Linear Statistical Models

Donlapark Ponnoprat

2022-03-06

Table of contents

Preface

These are lecture notes that I wrote for the masters course Linear Statistical Models (208780) in Winter 2022.

After teaching statistics for a couple of years, I observed that many of our masters students lack the programming skills needed to apply what they learn in class to solve real-world problems. This motivated me to redesign the course to be more practice-oriented, featuring full hands-on code examples in R.

The first two sections of the notes closely follow the comprehensive textbook *Linear Regression and Other Stories* (Gelman, Hill, and Vehtari 2020). This excellent book covers all aspects of linear regression, including fitting, prediction, diagnostics, and practical issues that may arise. Moreover, the book features numerous coding examples throughout its content.

The third section covers causal inference, which mainly focuses on estimation of the effect of a treatment on an outcome. We shall see that, with careful experimental design and covariate "adjustment", causal questions can be answered using linear regression. The lecture notes for this section again follow the materials in Gelman, Hill, and Vehtari (2020). Additional topics such as tests for assumptions, panel data, and synthetic control follow the materials in Cunningham (2021), Huntington-Klein (2021) and Facure (2020).

The last section covers conformal prediction, which is a relatively new technique of constructing a prediction interval under minimal statistical assumptions. The materials in this section closely follow the lecture notes of Stats 300C taught by Emmanuel Candes at Stanford University (Candès 2022) and the references therein.

Any comments and suggestions are welcome (my homepage).

Contents

Part I

Linear regression

In this course, we will use linear regression as a building block to develop more complex tool study the relationship between variables. Since its first conception in Sir Francis Galton's work on heredity characteristics in 1886, linear regression had been extensively studied for its statistical properties and interpretation.

We will discuss these properties of linear regression in close detail, starting with model fitting and how to interpret the coefficients of the fitted model. Using the Bayesian framework, we will learn how to quantify uncertainties from the posterior distributions, and diagnose the model's fit via plotting and simulations. We will also cover several topics in model selection, which include variable and prior selection, illustrated with coding examples in R.

Chapter 1

Basic regression

We start with the basics of Bayesian regression on simulated data. We will go over the steps to in order to obtain the coefficient estimates, as well as their uncertainties. First, we load the `rstanarm` package, an R package for Bayesian regression modeling which we will use throughout the course.

```
library("rstanarm")
```

1.1 Simulation

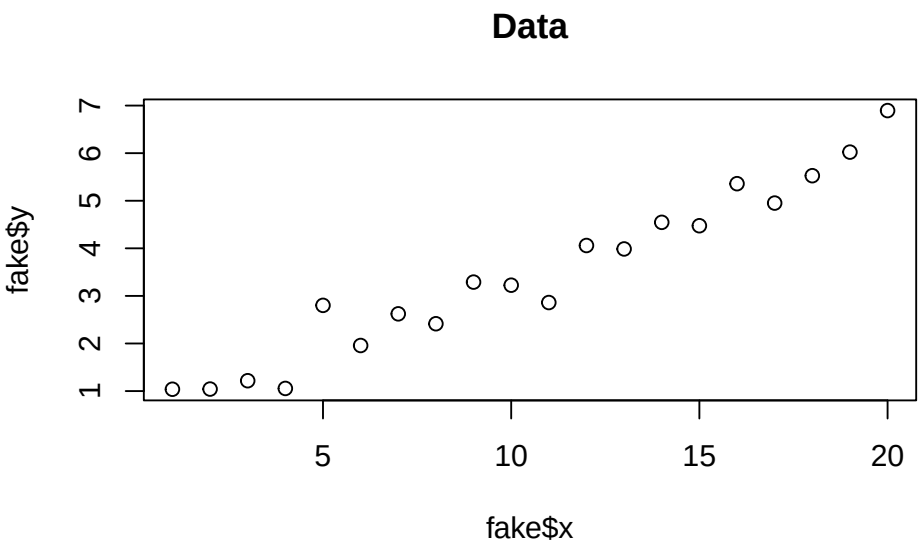
Simulate data as follows:

$$x = 1, 2, ..., 20 \tag{1.1}$$
$$y = 0.2 + 0.3x + \epsilon \tag{1.2}$$
$$\epsilon \sim N(0, 0.5) \tag{1.3}$$

```
x <- 1:20
n <- length(x)
a <- 0.2
b <- 0.3
sigma <- 0.5
y <- a + b*x + sigma*rnorm(n)

fake <- data.frame(x, y)
```

Plot the data

```
plot(fake$x, fake$y, main="Data")
```



Fit a regression model with `stan_glm`

```
fit_1 <- stan_glm(y ~ x, data=fake)
```

Here, the result shows the estimated coefficients with the uncertainties (the standard errors). It also estimates σ.

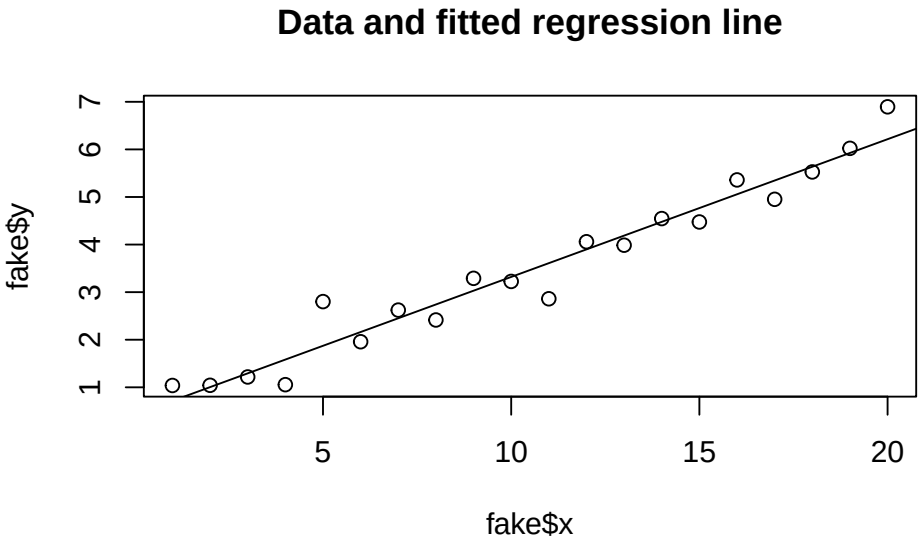
```
print(fit_1, digits=2)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      y ~ x
 observations: 20
 predictors:   2
------
            Median MAD_SD
(Intercept) 0.43   0.19
x           0.29   0.02

Auxiliary parameter(s):
      Median MAD_SD
sigma 0.42   0.07
```

```
------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

Plot the data with the fitted regression line

```
plot(fake$x, fake$y, main="Data and fitted regression line")
a_hat = coef(fit_1)[1]
b_hat = coef(fit_1)[2]
abline(a_hat, b_hat)
```



Here are the summary of the parameters.

Parameter	Assumed value	Estimate	Uncertainty
a	0.2	0.55	0.26
b	0.3	0.28	0.02
σ	0.5	0.54	0.09

From the properties of the standard normal distribution, for 68% of the time the true intercept a is between $0.55-0.26 = 0.29$ and $0.55+0.26 = 0.81$, and for 95% of the time it is between $0.55 - 2 \times 0.26 = 0.03$ and $0.55 + 2 \times 0.26 = 1.07$.

1.2 Earnings data

```
earnings <- read.csv("data/earnings.csv")
```

Fit a regression model

```
fit_2 <- stan_glm(earnk ~ height + male, data=earnings)
```

```
print(fit_2)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      earnk ~ height + male
 observations: 1816
 predictors:   3
------
            Median MAD_SD
(Intercept) -26.0   11.8
height        0.6    0.2
male         10.6    1.5

Auxiliary parameter(s):
      Median MAD_SD
sigma 21.4    0.3

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

The fitted model is:

$$\text{earnings} = -25.7 + 0.6 * \text{height} + 10.6 * \text{male} + \text{error}.$$

How should we interpret the coefficients here? Let's look at the following suggestions:

1. If we were to increase someone's height by one inch, his or her earning would increase by an expected amount of $600.
2. Comparing two people with the same sex but one inch different in height, the average difference in earnings is $600.

Between these two choices, the latter is more sensible. *Comparison* is a safer interpretation of the coefficient than the *effect*.

Similarly, it is more appropriate to say that, comparing two people with the same height but different sex, the man's earnings will be $10600 more than the woman's on average.

1.3 Historical origins of regression

The term *regression* comes from Francis Galton, a quantitative social scientist, who fit linear models to understand parent-child height relationship. He noticed that:

1. Children of tall parents tended to be taller than average but less tall than the parents.
2. Children of shorter parents tended to be shorter than the average but less short than the parents.

Thus, over time, people's heights have *regressed* to the mean, hence the term *regression*.

Let's look at the data of people's heights published by Karl Pearson and Alice Lee in 1903.

```
heights <- read.table("data/Heights.txt", header=TRUE)
print(heights[1:5,])
```

```
  daughter_height mother_height
1            52.5          59.5
2            52.5          59.5
3            53.5          59.5
4            53.5          59.5
5            55.5          59.5
```

Now we fit a regression model to the data.

```
fit_3 <- stan_glm(daughter_height ~ mother_height, data=heights)
```

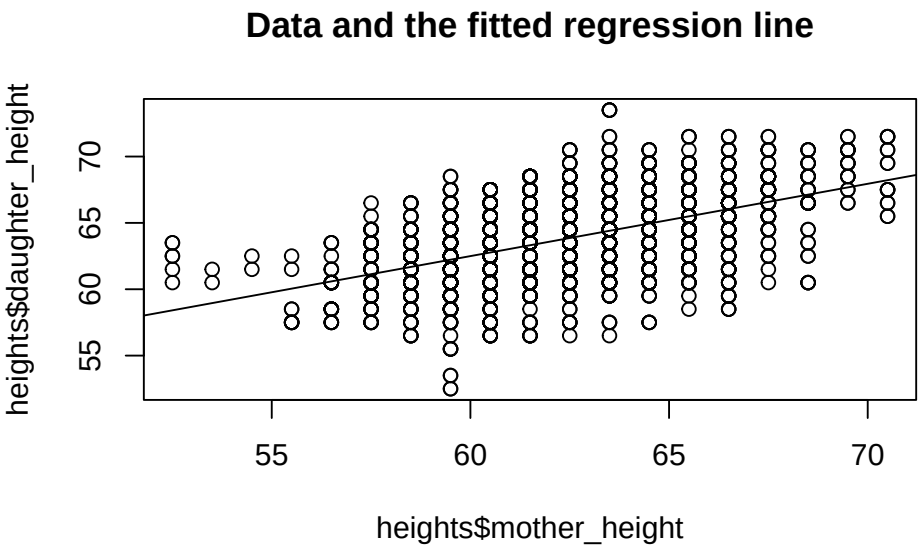
```
print(fit_3)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      daughter_height ~ mother_height
 observations: 5524
 predictors:   2
------
              Median MAD_SD
(Intercept)   29.8    0.8
```

```
mother_height  0.5    0.0

Auxiliary parameter(s):
      Median MAD_SD
sigma 2.3    0.0

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

Let's plot the data and the regression line.

```
plot(heights$mother_height, heights$daughter_height, main="Data and the fitted regression
a_hat = coef(fit_3)[1]
b_hat = coef(fit_3)[2]
abline(a_hat, b_hat)
```



Let's take a look at the average of daughters' and mothers' heights.

```
print(mean(heights$daughter_height))
```

```
[1] 63.85626
```

```
print(mean(heights$mother_height))
```

```
[1] 62.49873
```

The equation of the regression line is

$$y = 29.8 + 0.5x + \text{error}$$

The line's slope of 0.5 is easy to interpret—adding one inch to the mother's height corresponds to an increase in the daughter's height by 0.5 inches.

If the mother's height is 70 inches, then the daughter's height is 64.8 on average, so the daughter is less tall than the mother's but still taller than the average.

If the mother's height is 50 inches, then the daughter's height is 54.8 on average, so the daughter is less short than the mother's but still shorter than the average.

Looking back at the equation, we see < 1 coefficient which reduces variation of the daughters' heights, while the error term adds to the variation.

1.4 How regression to the mean can confuse people

Sometimes regression to the mean can lead people to mistakenly attribute it to causality. Here's a simulation of midterm and final scores of a group of students. Each score is composed of two components: the student's true ability and a random noise.

```
n <- 1000
true_ability <- rnorm(n, 50, 10)
noise_1 <- rnorm(n, 0, 10)
noise_2 <- rnorm(n, 0, 10)
midterm <- true_ability + noise_1
final <- true_ability + noise_2
exams <- data.frame(midterm, final)
```

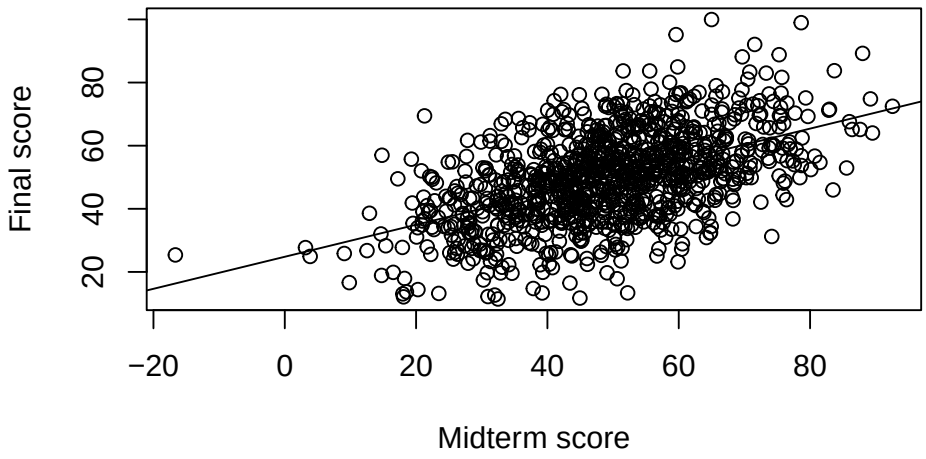
```
fit_4 <- stan_glm(final ~ midterm, data=exams)
```

```
print(fit_4)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      final ~ midterm
 observations: 1000
 predictors:   2
------
            Median MAD_SD
(Intercept) 24.8    1.4
```

```
midterm      0.5    0.0

Auxiliary parameter(s):
      Median MAD_SD
sigma 12.3    0.3

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

```r
plot(midterm, final, xlab="Midterm score", ylab="Final score")
abline(coef(fit_4))
```



One might infer from the coefficient of 0.5 that the students who did well on the midterm got overconfident and slacked off before the final, and the students who did poor on the midterm were motivated and tried extra hard for the final. But we know that this is not the case, as we generated this data ourselves! The real reason behind the regression to the mean is the variation between the midterms and the final scores: a student who scores very well on the first midterm (e.g. the two students with ~100 midterms score on the far right) are likely to have a high level of skill, and also was very lucky at the midterm (i.e. large positive noise) and so in the final exam, the student performs better than average but worse than on the midterm.

Chapter 2

Linear regression with a single predictor

As usual, we load the `rstanarm` package.

```
library("rstanarm")
```

2.1 Predicting presidential vote share from the economy

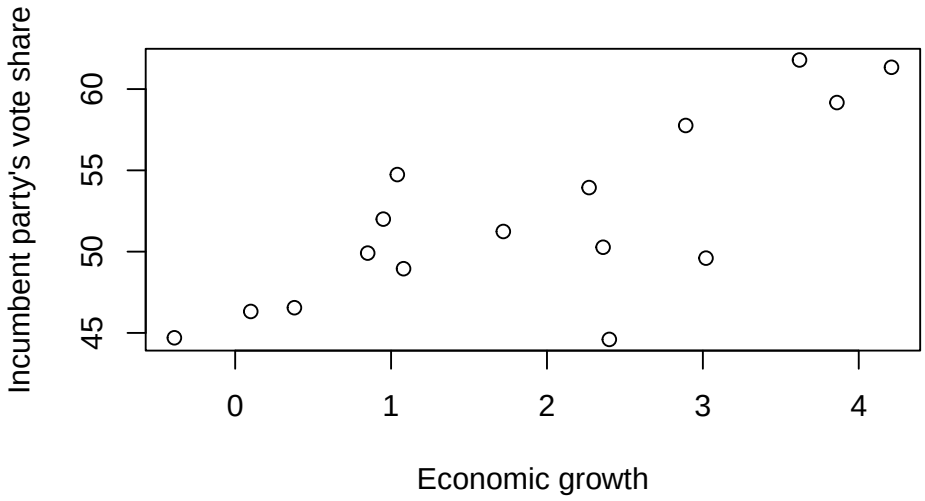
We will load the data from `hibbs.dat` which was created by Douglas Hibbs to forecast elections based on economic growth. Two important variables are: `growth`, the economic growth in the previous year and `vote`, the incumbent party's vote percentage.

```
hibbs <- read.table("data/hibbs.dat",
                    header=TRUE)
head(hibbs, 5)
```

```
  year growth  vote inc_party_candidate other_candidate
1 1952   2.40 44.60           Stevenson      Eisenhower
2 1956   2.89 57.76          Eisenhower       Stevenson
3 1960   0.85 49.91               Nixon         Kennedy
4 1964   4.21 61.34             Johnson       Goldwater
5 1968   3.02 49.60            Humphrey           Nixon
```

Plot the data

```
plot(hibbs$growth, hibbs$vote, xlab="Economic growth",
     ylab="Incumbent party's vote share")
```



Fit a regression model with `stan_glm`:

```
M1 <- stan_glm(vote ~ growth, data=hibbs,
               refresh=0)  # suppress output
```

Display the model:

```
print(M1)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      vote ~ growth
 observations: 16
 predictors:   2
------
            Median MAD_SD
(Intercept) 46.3    1.8
growth       3.0    0.7

Auxiliary parameter(s):
      Median MAD_SD
sigma 3.9    0.7

------
```

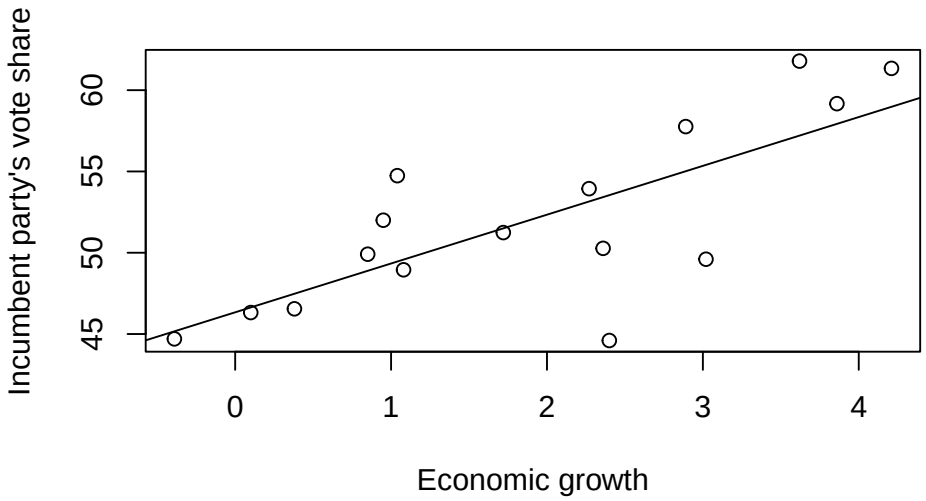
```
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

The fitted line is

$$y = 46.3 + 3.0x.$$

Plot the fitted regression line:

```
plot(hibbs$growth, hibbs$vote, xlab="Economic growth",
     ylab="Incumbent party's vote share")
abline(coef(M1))
```



- At $x = 0$ (zero economic growth), the incumbent party is predicted to receive 46.3% of the votes. This makes sense, as people are less likely to vote a party with poor performance.
- Every 1% of economic growth corresponds to an expected 3.0% increase in vote share for the incumbent party.
- The 68% confidence interval of the slope is $[3.0 \pm 0.7] = [2.3, 3.7]$ and the 95% confidence interval is $[3.0 \pm 2 \times 0.7] = [1.6, 4.4]$. It would be very unlikely that the data is generated from a model whose true slope is 0.
- The estimated *residual standard deviation* (the standard deviation of the error term) is 3.9%. This means that roughly 68% of the true vote percentages fall within ± 3.9 of the regression line.

2.1.1 Predicting the 2008 election

In the years leading up to the 2008 election, the economic growth was approximately 0.1% or $x = 0.1$. The linear model, $y = 46.3 + 3.0x$, predicted $y = 46.6$, or 46.6% of the vote going to the incumbent party, which was the Republicans at that time. It thus predicts 53.4% for Barack Obama, implying Democrats' victory in 2008.

2.1.2 Predicting the 2016 election

We now use the model to predict the 2016 presidential election of Democrat Hillary Clinton vs. Republican Donald Trump. At that time, the economic growth was approximately 2%. The linear model predicted

$$46.3 + 3.0 \times 2.0 = 52.3.$$

In other words, the model predicted that Clinton would have won the 2016 election, when in fact the winner was actually Trump.

Maybe we should have taken the uncertainty into account as well. We could ask ourselves: what is the chance that Clinton would win in that year? To answer this question, we recall that our model also has the error term:

$$y = 46.3 + 3.0x + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, 3.9).$$

Plugging in $x=2.0$ yields a random variable y:

$$y = 46.3 + 3.0 \times 2.0 + \varepsilon = 52.3 + \varepsilon \sim \mathcal{N}(52.3, 3.9).$$

The distribution of y is shown below. Clinton would have won if the vote share is greater than 50%. Thus the probability of Clinton winning the election is $\Pr[y > 50] = 0.72$.

The shaded area can be computed using the following code:

```
1-pnorm(50, 52.3, 3.9)
```

```
[1] 0.7223187
```

2.2 Checking the model's fit via simulation

We will demonstrate how to check the model's fit using the elections data above.

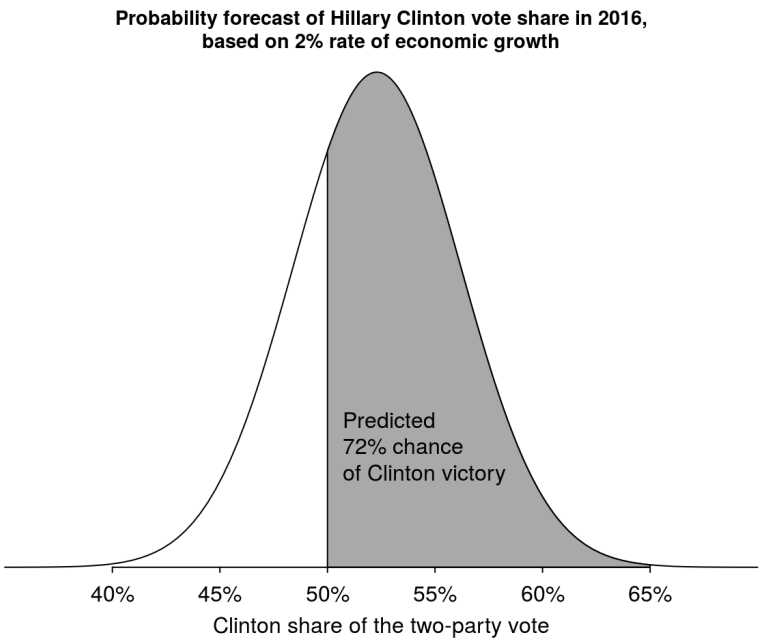


Figure 2.1: Forecast distribution

2.2.0.1 Step 1: Creating parameters from the fitted model

```
a <- 46.3
b <- 3.0
sigma <- 3.9
x <- hibbs$growth
n <- length(x)
```

2.2.0.2 Step 2: Simulating fake data

```
y <- a + b*x + rnorm(n, 0, sigma)
fake <- data.frame(x, y)
```

2.2.0.3 Step 3: Fitting the model and comparing fitted to assumed parameters

```
fit <- stan_glm(y ~ x, data=fake, refresh=0)
```

```
  print(fit)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      y ~ x
 observations: 16
 predictors:   2
------
            Median MAD_SD
(Intercept) 47.1    1.5
x            2.7    0.7

Auxiliary parameter(s):
      Median MAD_SD
sigma 3.5    0.7

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

The estimated coefficients (48.5 and 2.0) seem close enough to the assumed true values (46.3 and 3.0).

We can compare the coefficients formally by checking if the true values falls within 68% and 95% confidence intervals of the estimated coefficients. For simplicity, we will only do this for the slope b.

```
  b_hat <- coef(fit)["x"]
  b_se <- se(fit)["x"]
  print(b_hat)
```

```
       x
2.656575
```

```
  print(b_se)
```

```
        x
0.6575561
```

We then check whether the assumed true value of b falls within the 68% and 95% confidence intervals. However, since the original data is small, with a sample size of only 16, we need to use t-distribution instead of the normal distribution.

```r
t_68 <- qt(0.84, n-2)
t_95 <- qt(0.975, n-2)
cover_68 <- abs(b - b_hat) < t_68 * b_se
cover_95 <- abs(b - b_hat) < t_95 * b_se
paste("68% coverage: ", mean(cover_68))
```

```
[1] "68% coverage:  1"
```

```r
paste("95% coverage: ", mean(cover_95))
```

```
[1] "95% coverage:  1"
```

2.2.0.4 Step 4: Repeating the simulation in a loop

Now, we have to repeat the simulation several times and see if the *coverage probabilities*, that is, the probabilities that the confidence intervals contain the true coefficient, are close to 68% and 95%, respectively.

```r
n_fake <- 1000
cover_68 <- rep(NA, n_fake)  # c(NA, ... , NA)
cover_95 <- rep(NA, n_fake)  # c(NA, ... , NA)
t_68 <- qt(0.84, n-2)
t_95 <- qt(0.975, n-2)
pb <- txtProgressBar(min=0, max=n_fake, initial=0, style=3)
for (s in 1:n_fake){
  setTxtProgressBar(pb, s)
  y <- a + b*x + rnorm(n, 0, sigma)
  fake <- data.frame(x, y)
  fit <- stan_glm(y ~ x, data = fake, refresh = 0)

  b_hat <- coef(fit)["x"]
  b_se <- se(fit)["x"]

  cover_68[s] <- abs(b - b_hat) < t_68 * b_se
  cover_95[s] <- abs(b - b_hat) < t_95 * b_se
}
close(pb)
```

```r
paste("68% coverage: ", mean(cover_68))
```

```
[1] "68% coverage:  0.721"
```

```r
paste("95% coverage: ", mean(cover_95))
```

```
[1] "95% coverage:  0.957"
```

This simulation gives the desired result: close to 68% of 68% confidence intervals, and close to 95% of 95% confidence intervals, contain the true coefficients.

Chapter 3

Fitting linear regression

3.1 Least squares

The classic linear regression model is:

$$y_i = a + bx_i + \varepsilon.$$

Define the *residuals* as

$$r_i = y_i - (\hat{a} + \hat{b}x_i).$$

Note that this is different than the *errors* $\varepsilon_i = y_i - (a + bx_i)$, which cannot be obtained from the observed data.

In *least squares regression*, we estimate $(\hat{a}, \hat{b})$ that minimizes the *residual sum of squares*:

$$\text{RSS} = \sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n} (y_i - (\hat{a} + \hat{b}x_i))^2.$$

The $(\hat{a}, \hat{b})$ that minimizes RSS is called the *ordinary least squares* or *OLS estimate*, which is given by

$$\begin{aligned} \hat{b} &= \frac{\sum_{i=1}^{n}(x_i - \bar{x})y_i}{\sum_{i=1}^{n}(x_i - \bar{x})^2}, \\ \hat{a} &= \bar{y} - \hat{b}\bar{x}. \end{aligned}$$

Consequently, we can write the line equation as

$$y = \hat{a} + \hat{b}x = \bar{y} - \hat{b}\bar{x} + \hat{b}x = \bar{y} + \hat{b}(x - \bar{x}).$$

Thus, hte line goes through the mean of the data $(\bar{x}, \bar{y})$.

3.2 Estimation of residual standard deviation σ

Recall that we assume $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$. To find an unbiased estimator of σ^2, we use the following fact:

$$\frac{\text{RSS}}{\sigma^2} \sim \chi^2_{n-2}.$$

Combined with the fact that the expectation of a chi-square random variable equals its number of degrees of freedom, we have

$$\mathbb{E}\left[\frac{\text{RSS}}{\sigma^2}\right] = n - 2,$$

or equivalently,

$$\mathbb{E}\left[\frac{\text{RSS}}{n-2}\right] = \sigma^2.$$

Therefore, $\frac{\text{RSS}}{n-2}$ is an unbiased estimator of σ^2. Thus, we estimate the residual standard deviation σ using

$$\hat{\sigma} = \sqrt{\frac{\text{RSS}}{n-2}} = \sqrt{\frac{1}{n-2}\sum_{i=1}^{n}(y_i - (\hat{a} + \hat{b}x_i))^2}.$$

3.3 Maximum likelihood estimation

From the model $y_i = a + bx_i + \varepsilon_i$ where $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$, it follows that $y \sim N(a + bx_i, \sigma^2)$. The *likelihood function* is defined as the probability density function of the data, considered as a function of the parameters. Let $y = (y_1, \dots, y_n)$ and $X=(x_1,…,x_n). Then, the likelihood function in terms of $\hat{a}, \hat{b}$ and $\hat{\sigma}$ is

$$
\begin{aligned}
p(y \mid \hat{a}, \hat{b}, \hat{\sigma}, X) &= \prod_{i=1}^{n} p(y_i \mid \hat{a}, \hat{b}, \hat{\sigma}, X) \\
&= \frac{1}{(\sqrt{2\pi}\hat{\sigma})^n} \prod_{i=1}^{n} \exp\left(-\frac{1}{2} \left(\frac{y - (\hat{a} + \hat{b}x_i)}{\hat{\sigma}} \right)^2 \right) \\
&= \frac{1}{(\sqrt{2\pi}\hat{\sigma})^n} \exp\left(-\frac{1}{2\hat{\sigma}^2} \sum_{i=1}^{n} (y - (\hat{a} + \hat{b}x_i))^2 \right).
\end{aligned}
$$

Another way to estimate the parameters a, b and σ is to find $\hat{a}, \hat{b}$ and $\hat{\sigma}$ that maximizes the likelihood function. It is common to compute the log-likelihood function first.

$$
\log p(y \mid \hat{a}, \hat{b}, \hat{\sigma}, X) = -\frac{1}{2\hat{\sigma}^2} \sum_{i=1}^{n} (y - (\hat{a} + \hat{b}x_i))^2 - n \log \sqrt{2\pi}\hat{\sigma}.
$$

We can see the maximizing the log-likelihood function with respect to a and b. In other words, in linear regression, the *maximum likelihood estimates* of a and b are the same as the OLS estimates. To find the maximum likelihood estimate of σ, we apply the first derivative test of $\hat{\sigma}$.

$$
\frac{1}{\hat{\sigma}^3} \sum_{i=1}^{n} (y - (\hat{a} + \hat{b}x_i))^2 - \frac{n}{\hat{\sigma}} = 0.
$$

which gives

$$
\hat{\sigma}_{\text{mle}} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y - (\hat{a} + \hat{b}x_i))^2}.
$$

We observe that the maximum likelihood estimate is a biased estimate of σ.

Below is a plot of the likelihood function as a function of a and b, with σ fixed.

The second plot shows a level curve of the likelihood near the maximum likelihood estimate.

3.4 Bayesian linear regression

In Bayesian inference, we start by specifying a *prior distribution* on the parameters. In this case, the parameters are a, b and σ^2.

$$
p(a, b, \sigma^2).
$$

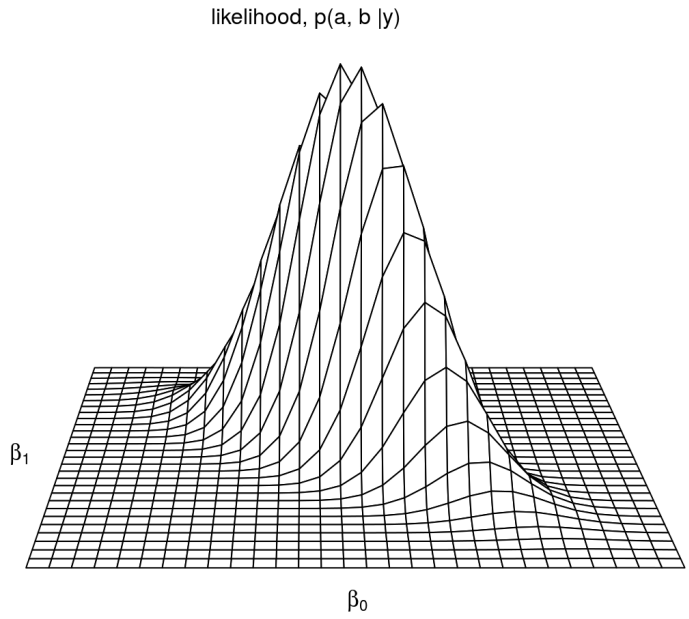


Figure 3.1: likelihood

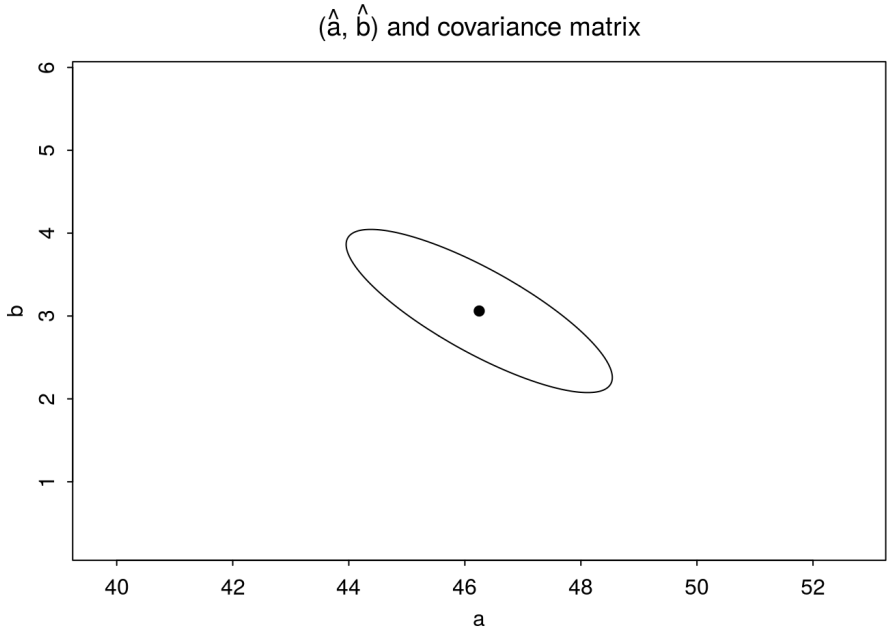


Figure 3.2: likelihood

Examples of prior distributions are: 1. Flat prior: $p(a, b, \sigma^2) = 1$ 2. $p(a, b, \sigma^2) = p(a, b \mid \sigma^2)p(\sigma^2)$ where $p(a, b \mid \sigma^2)$ is a normal distribution and $p(\sigma^2)$ is an inverse-gamma distribution.

Then, we specify the likelihood function. In linear regression, this is usually the normal likelihood.

$$p(y \mid a, b, \sigma^2, X) = \frac{1}{(\sqrt{2\pi}\sigma)^n} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y - (a + bx_i))^2\right).$$

We use the likelihood, which contains information about the data, to update our belief on the parameters via the Bayes' rule.

$$p(a, b, \sigma^2 \mid y, X) \propto p(y \mid a, b, \sigma^2, X)p(a, b, \sigma^2).$$

The left-hand side is called the *posterior distribution*. We then draw the parameters from the posterior distribution and use them to simulate posterior quantities, such as posterior mean of y or confidence intervals.

Here is a code example of Bayesian regression:

```
library(rstanarm)
```

```
x <- 1:10
y <- c(1, 1, 2, 3, 5, 8, 13, 21, 34, 55)
fake <- data.frame(x, y)
```

To fit the Bayesian regression with normal and inverse-gamma prior,

```
fit1 <- stan_glm(y ~ x, data=fake)
```

```
print(fit1)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      y ~ x
 observations: 10
 predictors:   2
------
            Median MAD_SD
(Intercept) -13.9    6.7
x             5.1    1.1
```

```
Auxiliary parameter(s):
      Median MAD_SD
sigma 9.9    2.5

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

To fit the Bayesian regression with flat prior (in this case, the posterior is the same as the likelihood),

```
fit2 <- stan_glm(y ~ x, data=fake, prior_intercept=NULL,
                 prior=NULL, prior_aux=NULL)
```

Here, `prior_intercept=NULL` sets a flat prior for the intercept, `prior=NULL` sets a flat prior for the other coefficients, and `prior_aux=NULL` sets a flat prior for σ^2.

```
print(fit2)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      y ~ x
 observations: 10
 predictors:   2
------
            Median MAD_SD
(Intercept) -14.1    7.0
x             5.1    1.1

Auxiliary parameter(s):
      Median MAD_SD
sigma 10.3    2.8

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

At default, `stan_glm` uses simulation to fit the model. To use optimization instead, set `algorithm="optimizing"`. The following code performs the maximum likelihood estimate.

```
fit3 <- stan_glm(y ~ x, data=fake, prior_intercept=NULL,
                 prior=NULL, prior_aux=NULL,
                 algorithm="optimizing")
```

```
  print(fit3)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      y ~ x
 observations: 10
 predictors:   2
------
            Median MAD_SD
(Intercept) -13.6    6.0
x             5.0    1.0

Auxiliary parameter(s):
      Median MAD_SD
sigma 9.8    2.3

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

3.5 Simulations from `stan_glm`

The fit from `stan_glm` yields a matrix of simulations. Here is an example of using these simulations to construct the 95% confidence interval of the coefficient of `x`.

```
  sims <- as.matrix(fit1)
  quantile(sims[, 2], c(0.025, 0.975))
```

```
    2.5%    97.5%
2.796289 7.533930
```

which is close to the approximation $[5.1 \pm 2 \times 1.1]$.

Chapter 4

Prediction and Bayesian inference

We go back to the Election vs Economy example.

```
library("rstanarm")
```

```
hibbs <- read.table("data/hibbs.dat",
                    header=TRUE)
M1 <- stan_glm(vote ~ growth, data=hibbs,
               refresh=0)  # suppress output
print(M1)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      vote ~ growth
 observations: 16
 predictors:   2
------
            Median MAD_SD
(Intercept) 46.3    1.7
growth       3.0    0.7

Auxiliary parameter(s):
      Median MAD_SD
sigma 3.9    0.8

------
* For help interpreting the printed output see ?print.stanreg
```

```
* For info on the priors used see ?prior_summary.stanreg
```

The `stan_glm` function, in addition to fitting the model, also performs 4000 simulations from the posterior distribution. The simulations can be obtained using the `as.matrix` function.

```
sims <- as.matrix(M1)
print(sims[1:5,])
```

```
          parameters
iterations (Intercept)   growth    sigma
      [1,]    45.82530 2.708003 4.354747
      [2,]    48.64418 2.151686 3.244081
      [3,]    47.25113 2.824543 3.689997
      [4,]    47.83557 2.971212 4.549891
      [5,]    46.71334 2.303495 4.524525
```

Then we can use these simulations to compute, for examples, posterior median and posterior median absolute deviation (MAD).

```
Median <- apply(sims, 2, median)
MAD_SD <- apply(sims, 2, mad)
print(cbind(Median, MAD_SD))
```

```
               Median    MAD_SD
(Intercept) 46.347337 1.7489130
growth       3.020284 0.7470974
sigma        3.937758 0.7548791
```

We can see that the numbers are similar to the results of the regression above.

4.1 Prediction and uncertainty: `predict`, `posterior_linpred`, and `posterior_predict`

These are functions to be called on the fitted regression (`M1`) with increasing levels of uncertainty.

1. `predict` returns the best point estimate for the average value of y given a new value of x.

$$\hat{y} = \hat{a} + \hat{b}x^{\text{new}}.$$

   First, we create a new dataframe with a single value $x^{\text{new}} = 2\%$.

```
new <- data.frame(growth=2.0)
print(new)
```

```
  growth
1      2
```

Then we use the `predict` function.

```
y_point_pred <- predict(M1, newdata=new)
print(y_point_pred)
```

```
       1
52.35994
```

This gives the same value as using the point estimates of the intercept and coefficient above: $46.3 + 3.0 \times 2 = 52.3$.

2. `posterior_linpred`

   This function returns a vector of posterior distributions of y:

   $$\hat{y} = a_i + b_i x^{\text{new}},$$

   over all simulations $a_1, \ldots, a_{4000}$ and $\$b_1, \ldots, b_{4000}$ from the posterior distributions of the intercept and slope, respectively. This is equivalent to:

```
sims <- as.matrix(M1)  # matrix of all 4000 simulations of a, b and sigma
a <- sims[,1]  # vector of 4000 simulations of intercept
b <- sims[,2]  # vector of 4000 simulations of slope
y_linpred <- a + b*as.numeric(new)
print(y_linpred[1:10])
```

```
 [1] 51.24131 52.94755 52.90022 53.77800 51.32033 51.29033 51.78258 52.81088
 [9] 50.47956 51.70579
```

Calling `posterior_linpred` gives us the same numbers:

```
y_linpred <- posterior_linpred(M1, newdata=new)
print(y_linpred[1:10])
```

```
 [1] 51.24131 52.94755 52.90022 53.77800 51.32033 51.29033 51.78258 52.81088
 [9] 50.47956 51.70579
```

3. `posterior_predict`

This function returns a vector of predictions, taking into account uncertainty of a, b and σ.

$$\hat{y} = a_i + b_i x^{\text{new}} + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma_i^2).$$

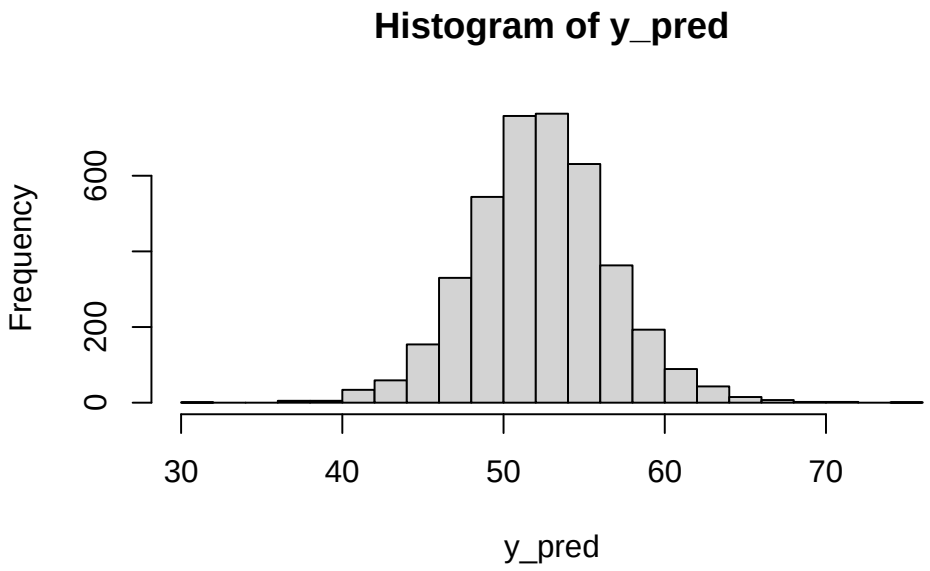
In addition to a_i's and b_i's as above, we also have $\sigma_1, \ldots, \sigma_{4000}$, the 4000 simulations from the posterior distribution. This is the same as the following code:

```
sims <- as.matrix(M1)  # matrix of all 4000 simulations of a, b and sigma
n_sims <- nrow(sims)  # number of rows in sims
a <- sims[,1]  # vector of 4000 simulations of intercept
b <- sims[,2]  # vector of 4000 simulations of slope
sigma <- sims[,3]  # vector of 4000 simulations of sigma
y_pred <- a + b*as.numeric(new) + rnorm(n_sims, 0, sigma)
print(y_pred[1:10])
```

```
[1] 53.25356 54.24320 49.29980 59.13072 45.90218 53.90072 46.59640 53.26520
[9] 49.48384 56.15140
```

Let us look at the histogram of the predictions.

```
hist(y_pred, breaks=20)
```



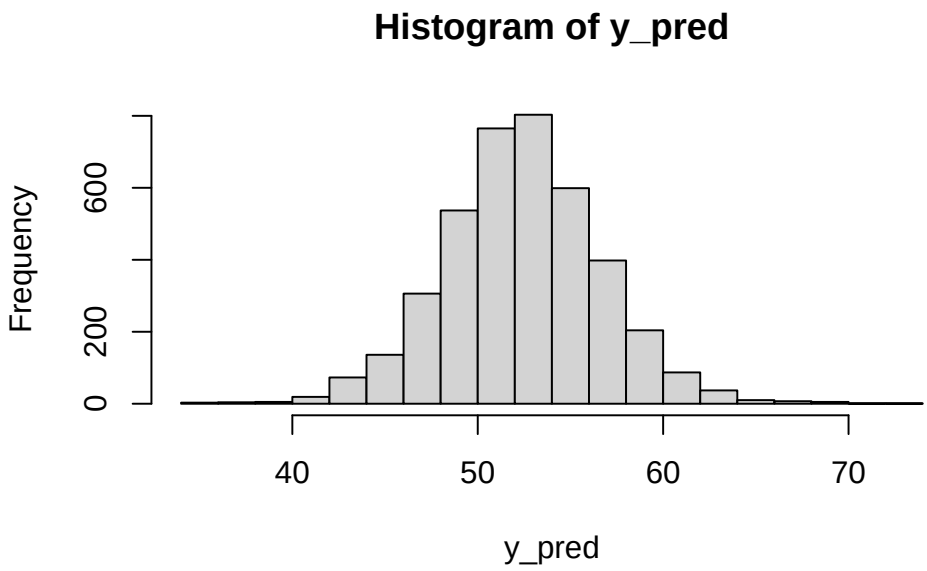
Now we try calling `posterior_predict` directly.

```r
y_pred <- posterior_predict(M1, newdata=new)
print(y_pred[1:10])
```

```
 [1] 55.78586 50.32929 54.14278 57.30579 58.52283 52.40129 48.17951 51.03949
 [9] 53.80145 54.73395
```

The predictions are not exactly the same as those from the direct simulation because of the noises. Nonetheless, we can compare the histograms between `posterior_predict` and direct simulation.

```r
hist(y_pred, breaks=20)
```



Now we can compute the point estimation of Clinton's voting share, the MAD and the winning probability.

```r
y_pred_mean <- median(y_pred)
y_pred_mad <- mad(y_pred)
win_indicator <- (y_pred > 50)
print(win_indicator[1:10])
```

```
 [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
```

```r
win_prob <- mean(win_indicator)
cat("Clinton's voting share: ", y_pred_mean, "\n")
```

```
Clinton's voting share:  52.36793
```

```
cat("MAD: ", y_pred_mad, "\n")
```

```
MAD:  3.973888
```

```
cat("Winning probability: ", win_prob, "\n")
```

```
Winning probability:  0.72925
```

4.1.1 Predictions on multiple inputs

We can also make predictions on a range of input values (in this case, outputs of `posterior_linpred` and `posterior_pred` are matrices of simulations of respective inputs.). Remember that we made a dataframe `new` consisting of a single value. In general, we can use a dataframe of a sequence of inputs. Here is an example where inputs are $-2.0\%, -1.5\%, \ldots 4.0\%$.

```
new_grid = data.frame(growth=seq(-2.0, 4.0, 0.5))
y_point_pred_grid = predict(M1, newdata=new_grid)
y_linpred_grid = posterior_linpred(M1, newdata=new_grid)
y_pred_grid = posterior_predict(M1, newdata=new_grid)

cat("Point estimations\n")
```

```
Point estimations
```

```
print(y_point_pred_grid)
```

```
       1        2        3        4        5        6        7        8
40.33822 41.84093 43.34365 44.84636 46.34908 47.85179 49.35451 50.85722
       9       10       11       12       13
52.35994 53.86265 55.36536 56.86808 58.37079
```

```
cat("\nLinear predictions with uncertainty\n")
```

```
Linear predictions with uncertainty
```

```
print(y_linpred_grid[1:5,])
```

```
iterations        1        2        3        4        5        6        7
      [1,] 40.40930 41.76330 43.11730 44.47130 45.82530 47.17930 48.53330
      [2,] 44.34081 45.41665 46.49250 47.56834 48.64418 49.72002 50.79587
      [3,] 41.60205 43.01432 44.42659 45.83886 47.25113 48.66341 50.07568
```

```
      [4,] 41.89315 43.37876 44.86436 46.34997 47.83557 49.32118 50.80679
      [5,] 42.10635 43.25810 44.40984 45.56159 46.71334 47.86509 49.01683

iterations        8        9       10       11       12       13
      [1,] 49.88731 51.24131 52.59531 53.94931 55.30331 56.65731
      [2,] 51.87171 52.94755 54.02340 55.09924 56.17508 57.25092
      [3,] 51.48795 52.90022 54.31249 55.72476 57.13704 58.54931
      [4,] 52.29239 53.77800 55.26360 56.74921 58.23481 59.72042
      [5,] 50.16858 51.32033 52.47208 53.62382 54.77557 55.92732
```

```
cat("\nPosterior predictions\n")
```

```
Posterior predictions
```

```
print(y_pred_grid[1:5,])
```

```
            1        2        3        4        5        6        7        8
[1,] 44.56453 39.33810 48.36911 36.91184 47.09549 45.61734 52.32406 43.72461
[2,] 43.47245 41.98679 46.74778 48.54136 44.28503 50.63784 49.28457 53.72223
[3,] 41.29061 44.92428 45.03022 44.89350 49.16970 53.57190 54.71974 53.06217
[4,] 44.32309 42.94489 50.09049 44.27604 48.09694 47.54070 46.53251 49.00634
[5,] 36.65576 37.83628 47.56969 51.50063 43.05292 45.83129 54.54305 43.97959
            9       10       11       12       13
[1,] 50.11013 46.43060 50.25066 52.48247 53.61879
[2,] 45.10324 53.13383 58.71023 54.39264 60.90816
[3,] 58.03417 56.40354 55.30158 52.04904 59.79294
[4,] 55.48687 57.27834 56.48904 51.28868 58.55687
[5,] 52.73422 54.00155 53.75586 55.41978 55.71907
```

The result of `predict` is a vector of length 13, `posterior_linpred` is a 4000×13 matrix, which contains 4000 predictions for each of the 13 values of growth, and `posterior_predict` is another 4000×13 matrix.

4.1.2 Predictions with input uncertainty

Previously, we have expressed uncertainty in the election outcome conditional on fixed values of economic growth. However, growth is usually estimated prior to the campaign, and updated by the government some time after. Hence, we have to take into account the uncertainty in growth when making predictions.

Let us assume that before the campaign, the economic growth was 2%, but after the campaign and just before the election, there would be a slight change in economic growth. We shall model the growth by $\mathcal{N}(0, 0.3^2)$. Let us simulate the growth from this distribution.

```
x_new <- rnorm(n_sims, 2.0, 0.3)  # create 4000 random numbers from N(0, 0.09)
```

We can then simulate the distribution of the prediction using the simulated a, b and σ.

```
sims <- as.matrix(M1)  # matrix of all 4000 simulations of a, b and sigma
n_sims <- nrow(sims)  # number of rows in sims
a <- sims[,1]  # vector of 4000 simulations of intercept
b <- sims[,2]

y_pred <- rnorm(n_sims, a + b*x_new, sigma)
```

Now we can compute the point estimation of Clinton's voting share, the MAD and the winning probability as before.

```
y_pred_mean <- median(y_pred)
y_pred_mad <- mad(y_pred)

win_indicator <- (y_pred > 50)
win_prob <- mean(win_indicator)

cat("Clinton's voting share: ", y_pred_mean, "\n")
```

```
Clinton's voting share:  52.44851
```

```
cat("MAD: ", y_pred_mad, "\n")
```

```
MAD:  4.206299
```

```
cat("Winning probability: ", win_prob, "\n")
```

```
Winning probability:  0.7195
```

Notice that the point prediction 52.3 is unchanged while the MAD has increased from 4.00 to 4.12 to reflect the extra uncertainty from the inputs.

4.2 Different types of priors in regression

Recall that in Bayesian inference, the likelihood is multiplied by a prior distribution to yield a posterior distribution.

In previous sections, we obtain the posterior distribution without being concerned with the prior distributions, as the data have strong linear relationship between the two variables. When the data are less informative, then we start to

think carefully about our choice of prior. We will consider three specific types of prior.

1. **Uniform prior distribution**

   This is sometimes called *non-informative* or *flat prior*. With a flat prior, the posterior is simply the product of the likelihood function and a constant. Thus the maximum likelihood estimate is the mode of the posterior distribution.

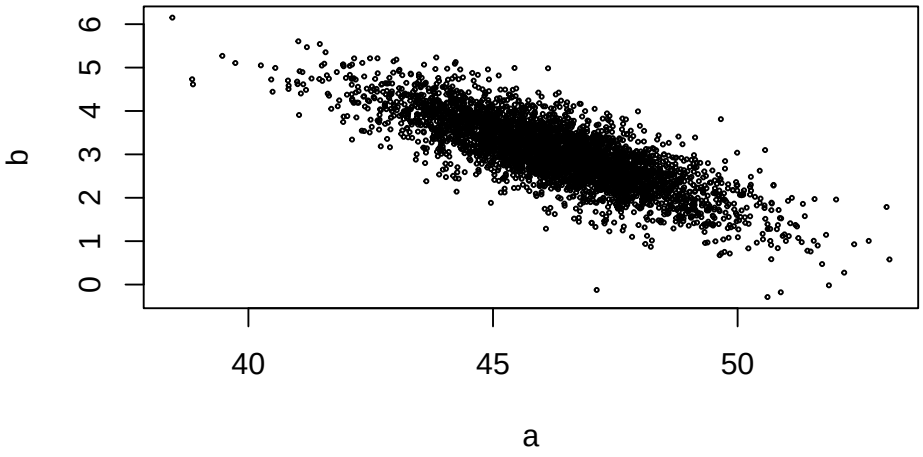
   As we mentioned in the previous chapter, to run with a flat prior, set the options of the coefficient and scale parameters to `NULL`.

```
M3 <- stan_glm(vote ~ growth, data=hibbs,
               prior_intercept=NULL, prior=NULL, prior_aux=NULL,
               refresh=0)
print(M3)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      vote ~ growth
 observations: 16
 predictors:   2
------
            Median MAD_SD
(Intercept) 46.2    1.7
growth       3.1    0.7

Auxiliary parameter(s):
      Median MAD_SD
sigma 4.0    0.8

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

   Then we plot the simulated values of a and b from the posterior distribution

```
sims <- as.matrix(M3)
a <- sims[,1]
b <- sims[,2]
plot(a, b, cex=0.3)
```



We can see that the (joint) posterior distribution of a and b is a normal distribution centered at the point estimates.

2. **Weakly informative prior**

   Weakly informative priors contain information about the scales of the parameters, where the scales are obtained from some appropriate analysis. They are not informative prior as they do not utilize prior knowledge about the parameters.

   At default, the `stan_glm` function uses a data-dependent weakly informative prior.

$$p(a, b, \sigma) = p(a|b)p(b)p(\sigma),$$

   where

   1. $p(b) = \mathcal{N}(0, 2.5 \text{ sd}(y)/\text{sd}(x))$
   2. $p(a|b) = p(a + b\bar{x}|b) = \mathcal{N}(\bar{y}, 2.5 \text{ sd}(y))$
   3. $p(\sigma) = \text{Exp}(1/\text{sd}(y))$.

   The scales of the intercept and slope are obtained via the following analysis:

   For a model of the form $y = a + bx + \text{error}$, the formulae of the OLS estimate are

$$\hat{b} = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$
$$\hat{a} = \bar{y} - \hat{b}\bar{x}.$$

Applying the Cauchy-Schwarz inequality: $\left|\sum_i a_i b_i\right| \le \sqrt{\sum_i a_i^2}\sqrt{\sum_i b_i^2}$, we obtain

$$\begin{aligned}
|\hat{b}| &= \frac{\left|\sum_{i=1}^n (y_i - \bar{y})(x_i - \bar{x})\right|}{\sum_{i=1}^n (x_i - \bar{x})^2} \\
&\le \frac{\sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2}}{\sum_{i=1}^n (x_i - \bar{x})^2} \\
&= \frac{\sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2}} \\
&= \frac{\mathrm{sd}(y)}{\mathrm{sd}(x)}.
\end{aligned}$$

We use this inequality for the estimate $|\hat{b}|$ to guide our belief about the value b. More precisely, the inequality suggests that the value of $|b|$ should not exceed sd(y)/sd(x). This motivates the weakly informative prior for b used in `stan_glm`, which is $\mathcal{N}(0, 2.5\ \mathrm{sd(y)}/\mathrm{sd}(x))$; such prior pulls the slope estimate towards the range from -2.5 sd(y)/sd(x)) to 2.5 sd(y)/sd(x)). Here, the 2.5 factor was chosen arbitrarily so that that prior does not have too much influence on the slope estimate when data are sufficiently informative.

For the intercept a, the choice of the distribution simply comes from the following computations with b fixed:

$$\begin{aligned}
\mathbb{E}[a + b\bar{x}] &= \mathbb{E}[\bar{y}] \\
\mathrm{sd}(a + b\bar{x}) &= \mathrm{sd}(a).
\end{aligned}$$

Lastly, Exp(1/sd(y)) is chosen as a prior for σ because it is a distribution over positive real numbers with mean equal sd(y).

To fit a linear regression with the weakly informative prior, simply run

```
M1 <- stan_glm(vote ~ growth, data=hibbs, refresh=0)
print(M1)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      vote ~ growth
 observations: 16
```

```
 predictors:   2
------
            Median MAD_SD
(Intercept) 46.3    1.7
growth       3.0    0.7

Auxiliary parameter(s):
      Median MAD_SD
sigma 3.9    0.7

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

3. **Informative prior**

   Informative priors are designed with prior knowledge about the variables. For the intercept, the `prior_intercept` argument in `stan_glm` is defined with our guess of $a+b\bar{x}$, which is the value of y when x is set to the average value. In the Election vs Economic growth example, this corresponds to Clinton's voting share when the growth is historically average, should be close to 50%, and it should not be less than 40% or more than 60%. Thus we set the prior for the intercept to be $\mathcal{N}(50, 10)$.

   For the slope, we should consider: how much is a swing in voting share if the economic growth were to increase by 1%. The swing is most likely not going to be more than 10%. With this information, we set the prior of the slope to be $\mathcal{N}(5, 5)$.

   Here is the code to fit a Bayesian regression model with these priors:

```r
M4 <- stan_glm(vote ~ growth, data=hibbs,
               prior=normal(5,5), prior_intercept=normal(50, 10),
               refresh=0)
print(M4)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      vote ~ growth
 observations: 16
 predictors:   2
------
            Median MAD_SD
(Intercept) 46.1    1.7
growth       3.1    0.7

Auxiliary parameter(s):
```

```
      Median MAD_SD
sigma 3.9    0.7

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

We see that the difference between this model and the previous models are not noticeable since the prior contains very little information compared to the data.

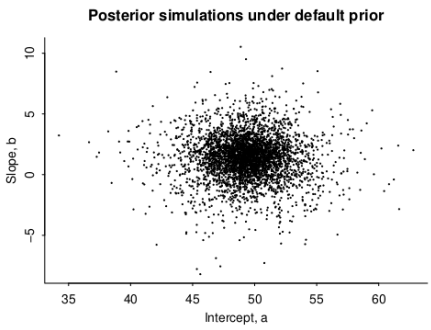
4.2.1 Example of regression with difference priors: Beauty and sex ratio

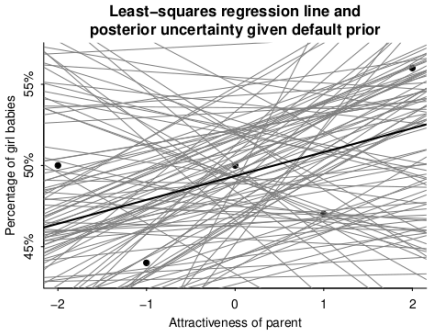
Here, we consider data in which the predictor is the parents' attractiveness of a five-point scale, and the target variable is the percentage of girls births among parents in each attractiveness category.

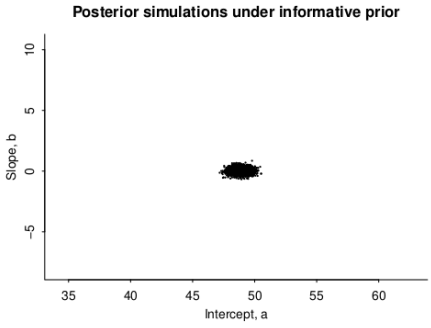
We fit two regression models, one with a weakly informative prior, and one with an informative prior. In informative prior, we set a prior for the following parameters:

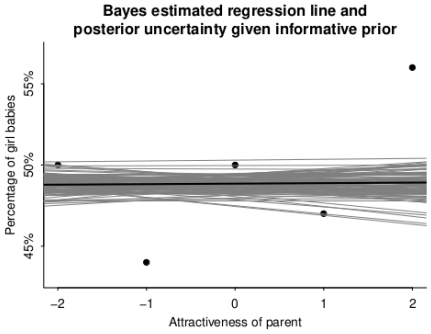
1. Intercept: to find a prior, we must guess the value of $a + b\bar{x}$, that is, the percentages of girls for parents of average beauty. We have prior knowledge of percentages of girl birth to be stable at roughly 48.5% to 49%. So we choose the prior to be $\mathcal{N}(48.8, 0.5^2)$.
2. Slope: We think that the parents' attractive should have barely any effect on the percentages of girl births, so we choose the prior to be $\mathcal{N}(0, 0.2^2)$.

Below are the plots of the regressions with the weakly informative prior and the informative prior, respectively. We can see that the data offers no information about a and b.


Posterior simulations under default prior
Slope, b
Intercept, a
10
5
0
−5
35
40
45
50
55
60


Least−squares regression line and
posterior uncertainty given default prior
Percentage of girl babies
Attractiveness of parent
55%
50%
45%
−2
−1
0
1
2


Posterior simulations under informative prior
Slope, b
Intercept, a
10
5
0
−5
35
40
45
50
55
60


Bayes estimated regression line and
posterior uncertainty given informative prior
Percentage of girl babies
Attractiveness of parent
55%
50%
45%
−2
−1
0
1
2

Chapter 5

Linear regression with multiple predictors

The linear regression with multiple predictors $x_1, \dots, x_p$ can be written in matrix-vector form (ignoring the error terms) as:

$$\begin{aligned}\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} &= \begin{pmatrix} \beta_0 + \beta_1 x_{11} + \dots + \beta_p x_{1p} + \varepsilon_1 \\ \beta_0 + \beta_1 x_{21} + \dots + \beta_p x_{2p} + \varepsilon_2 \\ \vdots \\ \beta_0 + \beta_1 x_{n1} + \dots + \beta_p x_{np} + \varepsilon_n \end{pmatrix} \\ &= \begin{pmatrix} 1 & x_{11} & \dots & x_{1p} \\ 1 & x_{21} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \dots & x_{np} \end{pmatrix} \cdot \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix},\end{aligned}$$

or, in short,

$$y = X\beta + \varepsilon.$$

Here, y is the vector of outputs, X is the *design matrix*, β is the vector of parameters and ε is the vector of the error terms. Assuming that ε^i are distributed as $\mathcal{N}(0,\sigma^2)$, another way of writing the model is

$$y \sim \mathcal{N}(X\beta, \sigma^2 I).$$

We attempt to find an estimator $\hat{\beta}$ of β by removing the error term to obtain an approximate equation:

$$
\begin{aligned}
y &\approx X\hat{\beta} \\
X^T y &\approx X^T X\hat{\beta} \\
\hat{\beta} &\approx (X^T X)^{-1} X^T y.
\end{aligned}
$$

Such $\hat{\beta}$ is called an *ordinary least squares* (OLS) estimate of β.

First, we import the `KidIQ` data, which contains data of children's and their mother's IQ.

```
kidiq <- read.csv("data/kidiq.csv")

head(kidiq)
```

```
  kid_score mom_hs    mom_iq mom_work mom_age
1        65      1 121.11753        4      27
2        98      1  89.36188        4      25
3        85      1 115.44316        4      27
4        83      1  99.44964        3      25
5       115      1  92.74571        4      27
6        98      0 107.90184        1      18
```

Here, `kid_score` is the child's IQ score, `mom_hs` is an indicator for whether the mother graduated from high school (0 or 1), and `mom_iq` is the mother's IQ score.

Now we fit a linear regression model of `kid_score` on two predictors: `mom_hs` and `mom_iq`.

```
library(rstanarm)
```

```
fit_1 <- stan_glm(kid_score ~ mom_hs + mom_iq, data=kidiq,
                  refresh=0)
print(fit_1)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      kid_score ~ mom_hs + mom_iq
 observations: 434
 predictors:   3
------
            Median MAD_SD
(Intercept) 25.6    5.8
```

```
mom_hs       5.9    2.2
mom_iq       0.6    0.1

Auxiliary parameter(s):
      Median MAD_SD
sigma 18.1    0.6

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

Thus the fitted line from the linear regression above is:

$$\text{kid\_score} = 26 + 6 * \text{mom\_hs} + 0.6 * \text{mom\_iq} + \text{error}.$$

Below are some suggestions for an interpretation of the `mom_iq`'s coefficient:

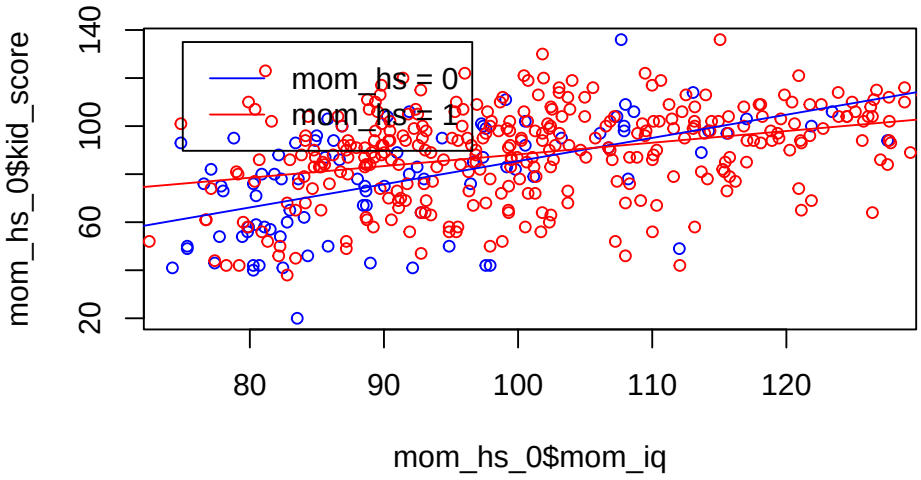
- *Predictive interpretation*: the difference between *two* children's IQ when their mothers' IQs differ by 1 and the other predictors (in this case, `mom_hs`) are identical is 0.6 on average.
- *Counterfactual interpretation*: changing the mother's IQ from 100 to 101, while leaving the other predictors unchanged, would lead to an expected increase of 0.6 in child's test score. This kind of reasoning arises in causal inference.

5.1 Interactions

The linear model with two predictors above imposes that the slope of `mom_iq` is the same for the subsets consisting of `mom_hs = 0` and `mom_hs = 1`. However, if we consider the linear regression on these two subsets:

```
mom_hs_0 = kidiq[kidiq$mom_hs == 0, ]
mom_hs_1 = kidiq[kidiq$mom_hs == 1, ]

fit_2 <- stan_glm(kid_score ~ mom_iq, data=mom_hs_0,
                  refresh=0)
fit_3 <- stan_glm(kid_score ~ mom_iq, data=mom_hs_1,
                  refresh=0)

plot(mom_hs_0$mom_iq, mom_hs_0$kid_score, col="blue", cex=0.8)
points(mom_hs_1$mom_iq, mom_hs_1$kid_score, col="red", cex=0.8)
abline(coef(fit_2), col="blue")
abline(coef(fit_3), col="red")
legend(75, 135, legend=c("mom_hs = 0", "mom_hs = 1"),
```

```
        col=c("blue", "red"), lty=1)
```



we can see that the slopes differ by a significant amount between the subsets. A remedy for this is to add an *interaction* term between `mom_hs` and `mom_iq`. This can be done by adding `mom_hs:mom_iq` to `stan_glm`.

```
fit_4 <- stan_glm(kid_score ~ mom_hs + mom_iq + mom_hs:mom_iq, data=kidiq,
                  refresh=0)
print(fit_4)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      kid_score ~ mom_hs + mom_iq + mom_hs:mom_iq
 observations: 434
 predictors:   4
------
             Median MAD_SD
(Intercept)  -10.0   13.6
mom_hs        49.4   15.1
mom_iq         1.0    0.1
mom_hs:mom_iq -0.5    0.2

Auxiliary parameter(s):
      Median MAD_SD
sigma 18.0    0.6

------
* For help interpreting the printed output see ?print.stanreg
```

```
* For info on the priors used see ?prior_summary.stanreg
```

The output tells us that the fitted model is:

$$\text{kid\_score} = -10.5 + 50 * \text{mom\_hs} + \text{mom\_iq} - 0.5 * \text{mom\_hs} * \text{mom\_iq} + \text{error}.$$

We now obtain equations of `kid_score` vs `mom_iq` for each `mom_hs` group by setting `mom_hs = 0` and `mom_hs = 1`. When `mom_hs = 0`, the equation becomes:

$$\text{kid\_score} = -10.5 + \text{mom\_iq} + \text{error}.$$

When `mom_hs = 1`, the equation becomes:

$$\text{kid\_score} = 39.5 + 0.5 * \text{mom\_iq} + \text{error}.$$

Comparing between three equations above, we can interpret the coefficients of the equation with the interaction term as follows:

- The intercept represents the predicted test scores for children whose mothers did not complete high school (`mom_hs = 0`) and had IQs of 0 (`mom_iq = 0`)—not a meaningful scenario.

  The intercept can be more interpretable if input variables are centered before including them as regression predictors.

- The coefficient of `mom_hs` is the difference between the predicted test scores for children whose mothers did not complete high school (`mom_hs = 0`) and children whose mothers did complete high school (`mom_hs = 1`), both with `mom_iq = 0`; this is inconceivable as no mothers have IQ of 0. To make the coefficient more interpretable, one might want to center the variable (i.e. subtract the observed values of `mom_hs` by their mean) first.

- The coefficient of `mom_iq` can be thought of as the comparison of mean test scores across children whose mothers did not complete high school (`mom_hs = 0`), but their mothers' IQs differ by 1 point.

- The coefficient on the interaction term is the difference between the slopes of the lines (regression on each `mom_hs` group) in the plot above.

When should we look for interactions? Interaction typically arises when the effects of a predictor are different across different groups. For example, when predicting the likelihood of cancer on smoking (0 or 1) and home radon exposure, the risk of cancer associated with the radon exposure is higher in the smoking group than non-smoking group.

5.2 Regression with multiple levels of a categorical predictor

We give an example of a regression with multiple inputs, some of which are categorical variables with multiple levels. Here, we will use the `Earnings` data.

```
earnings <- read.csv("data/earnings.csv")
```

```
summary(factor(earnings$ethnicity))
```

```
   Black Hispanic    Other    White
     180      104       38     1494
```

We fit a linear regression of `weight` on three predictors: `height`, `male` (0 or 1) and `ethnicity` (White, Black, Hispanic and Other).

```
fit_5 <- stan_glm(weight ~ height + male + ethnicity, data=earnings,
                  refresh=0)
print(fit_5)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      weight ~ height + male + ethnicity
 observations: 1789
 predictors:   6
------
                  Median MAD_SD
(Intercept)       -99.8   16.5
height              3.9    0.3
male               12.1    1.9
ethnicityHispanic  -6.2    3.7
ethnicityOther    -12.2    5.1
ethnicityWhite     -5.2    2.3

Auxiliary parameter(s):
      Median MAD_SD
sigma 28.7    0.5

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

We can see that `stan_glm` has automatically created three new *indicator variables* (also called *dummy variables*), namely `ethnicityHispanic`,

`ethnicityOther` and `ethnicityWhite` for each person. The value of each variable is 1 if the person belongs to the corresponding level, otherwise it is 0. For example, for a Hispanic person, `ethnicityHispanic = 1`, `ethnicityOther = 0` and `ethnicityWhite = 0`. But if we look closely at the `ethnicity` column in the dataset, we would notice that the group of Blacks is missing. This is because `stan_glm` took `Black` to be the *baseline* category against the other groups. Consequently, a black person is represented by `ethnicityHispanic = 0`, `ethnicityOther = 0` and `ethnicityWhite = 0`.

The model from the regression above is:

$$\begin{aligned}\text{weight} = &-100 + 3.9 * \text{height} + 12.1 * \text{male} \\ &- 6.1 * \text{Hispanic} - 12.3 * \text{Other} - 5.2 * \text{White} + \text{error}.\end{aligned}$$

The coefficient of `ethnicity`, can be interpreted as follows: between two persons with the same height and same gender,

- On average, a Hispanic person is 6.1 pounds lighter than a Black person, an Other person is 12.3 pounds lighter than a Black person, and a White person is 5.2 pounds lighter than a Black person.
- On average, a Hispanic person is $12.3 - 6.1 = 6.2$ pounds heavier than an Other person.
- On average, a Hispanic person is $6.1 - 5.2 = 0.9$ pounds lighter than a White person.
- On average, a White person is $12.3 - 5.2 = 7.1$ pounds heavier than an Other person.

Sometimes, we would like to change the baseline group; this can be done by specifying the order of the levels in the categorical variables. An example below shows how to set `White` as the baseline groups.

```
earnings$eth <- factor(earnings$ethnicity,
                       levels=c("White", "Black", "Hispanic", "Other"))
fit_5 <- stan_glm(weight ~ height + male + eth, data=earnings,
                  refresh=0)
print(fit_5)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      weight ~ height + male + eth
 observations: 1789
 predictors:   6
------
            Median MAD_SD
(Intercept) -105.1   17.1
```

```
height         3.9    0.3
male          12.1    2.0
ethBlack       5.2    2.4
ethHispanic   -0.9    3.0
ethOther      -7.0    4.9

Auxiliary parameter(s):
      Median MAD_SD
sigma 28.7    0.5

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

5.2.1 Simulation-based prediction

From the fitted model, we give an example of computing Pr(weight > 180) for a Hispanic male who is 70 inches tall. As in the previous chapter, we compute the probability using posterior simulation.

```
new = data.frame(height=70, male=1, eth="Hispanic")
y_pred <- posterior_predict(fit_5, newdata=new)
y_180_indicator <- (y_pred > 180)
y_180_prob <- mean(y_180_indicator)
cat("Probability of being heavier than 180 pounds: ", y_180_prob)
```

```
Probability of being heavier than 180 pounds:  0.4485
```

5.3 Paired and blocked designs as a regression problem

In the previous sections, we see that regression coefficients can be interpreted as comparisons. Conversely, comparison between two or more grounds can be treated a regressions.

1. **Completely randomized experiment**

   Suppose that in an experiment, people are randomly assigned into treatment and control groups. A standard estimate for the treatment effect is $\bar{y}_T - \bar{y}_C$. We can also obtain this estimate by assigning an indicator variable to each group: 0 for the control group and 1 for the treatment group. Then $\bar{y}_T - \bar{y}_C$ is precisely the coefficient of the following regression:

   ```
   fit <- stan_glm(y ~ treatment, data=experiment)
   ```

where `treatement` is 0 if the person is in the control group, and 1 if they are in the treatment group.

2. **Paired design**

   When the experiment consists of control-treatment pairs and the people in each pair might not be independent, we can assign a new indicator variable that indicates each pair e.g. the control and treatment in the first pair get `pair = 1`, the second pair gets `pair = 2`, so on and so forth.

   ```
   fit <- stan_glm(y ~ treatment + factor(pair), data=experiment)
   ```

3. **Block design**

   When considering the treatment effect over J groups of people, we can assign a new indicator variable `group` to indicate each group e.g. the first group gets `group = 1`.

   ```
   fit <- stan_glm(y ~ treatment + factor(group), data=experiment)
   ```

5.4 Weighted regression

The OLS estimate $\hat{\beta}$ minimizes the least squares objective:

$$\sum_i (y_i - X_i\hat{\beta})^2,$$

where $X_i = (1, x_{i1}, \dots, x_{ip})$. We can modify the objective with weights $w_1, \dots, w_n$ for each instance in the sum:

$$\sum_i w_i (y_i - X_i\hat{\beta})^2. \tag{1}$$

We typically require that $\sum_i w_i = 1$. Why do we want to add these weights to the objective? Sometimes, the proportions of different groups in the sample data do not match with those in the population, and we would like to make correction when fitting the model by adding the weights. For example, suppose that our data consists of 70 white persons and 30 black persons. To balance the model between these two groups, we can put weight 1/140 on all terms that correspond to white persons, and 1/60 to all terms that correspond to black persons.

Let $W = \text{Diag}(w_1, \dots, w_n)$. The estimate $\hat{\beta}_{\text{wls}}$ that minimizes the weighted objective (1) is

$$\hat{\beta}_{\text{wls}} = (X^T W^{-1} X)^{-1} X^T W^{-1} y.$$

To fit a weighted regression model using `stan_glm`, we just need to specify the `weights` parameter.

```
summary(factor(earnings$ethnicity))
```

```
   Black Hispanic    Other    White
     180      104       38     1494
```

```
N <- 180 + 104 + 38 + 1494
eth <- earnings$ethnicity
w <- (1/(4*180))*(eth == "Black") +
  (1/(4*104))*(eth == "Hispanic") +
  (1/(4*38))*(eth == "Other") +
  (1/(4*1494))*(eth == "White")

fit_6 <- stan_glm(weight ~ height + male + ethnicity, data=earnings,
                  weights=w, refresh=0)
print(fit_6)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      weight ~ height + male + ethnicity
 observations: 1789
 predictors:   6
------
                  Median MAD_SD
(Intercept)       -100.1   16.2
height               3.9    0.3
male                12.1    2.0
ethnicityHispanic   -6.2    3.5
ethnicityOther     -12.4    5.4
ethnicityWhite      -5.3    2.3

Auxiliary parameter(s):
      Median MAD_SD
sigma 28.6    0.5

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

The results of the weighted regression is not much different than those of the classical regression.

Chapter 6

Model diagnostics and evaluation

We will talk about several graphical and quantitative ways to check our model's fit to the data, and later we will talk about about cross validation—a technique for comparing between different models.

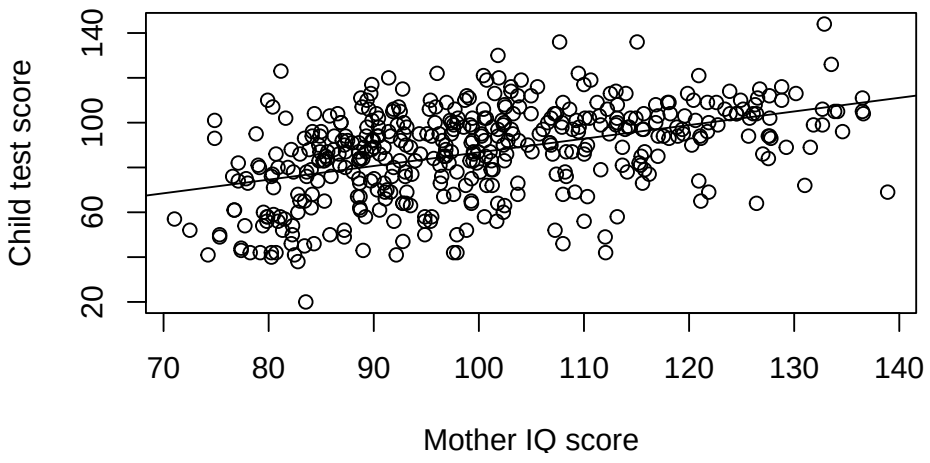
6.1 Plotting the data and the fitted model

6.1.1 Model with one predictor

In one-predictor cases, we can visualize the regression's fit by simply plotting the data and the regression line. Here, we plot the `KidIQ` data and the fitted lines.

```
library(rstanarm)
```

```
kidiq = read.csv("data/kidiq.csv")

fit_2 <- stan_glm(kid_score ~ mom_iq, data=kidiq, refresh=0)
plot(kidiq$mom_iq, kidiq$kid_score,
     xlab="Mother IQ score", ylab="Child test score")
abline(coef(fit_2)[1], coef(fit_2)[2])
```



6.1.2 Model with two predictors

6.1.2.1 Model with one categorical predictor and no interaction

We can plot each group and the fitted model on that group. Using the `KidIQ` data as an example, we recall that the non-interactive regression equation is

$$\text{kid\_score} = \hat{\beta}_1 + \hat{\beta}_2 * \text{mom\_hs} + \hat{\beta}_3 * \text{mom\_iq}.$$

Consequently, the equation of kid_score on mom_iq for the group mom_hs = 1 is

$$\text{kid\_score} = (\hat{\beta}_1 + \hat{\beta}_2) + \hat{\beta}_3 * \text{mom\_iq},$$

and the equation for the group mom_hs = 0 is

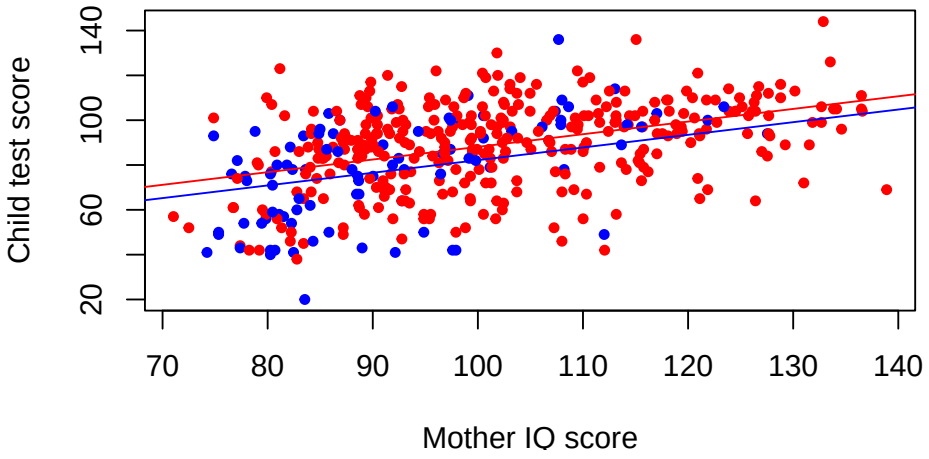
$$\text{kid\_score} = \hat{\beta}_1 + \hat{\beta}_3 * \text{mom\_iq}.$$

We thus obtain the intercept and the slope of each equation, which we use to plot the regression line of each mom_hs group below.

```
fit_3 <- stan_glm(kid_score ~ mom_hs + mom_iq, data=kidiq,
                  refresh=0)

# create a vector with value = "red" if mom_hs==1
# "blue" if mom_hs==0
colors <- ifelse(kidiq$mom_hs==1, "red", "blue")

print(colors[1:10])
```

```
 [1] "red"  "red"  "red"  "red"  "red"  "blue" "red"  "red"  "red"  "red"
```

```
plot(kidiq$mom_iq, kidiq$kid_score,
     xlab="Mother IQ score", ylab="Child test score",
     col=colors, pch=20)
b_hat <- coef(fit_3)
abline(b_hat[1] + b_hat[2], b_hat[3], col="red")
abline(b_hat[1], b_hat[3], col="blue")
```



6.1.2.2 Model with one categorical variable and an interaction

In this case, the equation is

$$\text{kid\_score} = \hat{\beta}_1 + \hat{\beta}_2 * \text{mom\_hs} + \hat{\beta}_3 * \text{mom\_iq} + \hat{\beta}_4 * \text{mom\_hs} * \text{mom\_iq}.$$

When $\text{mom\_hs} = 1$, the equation is

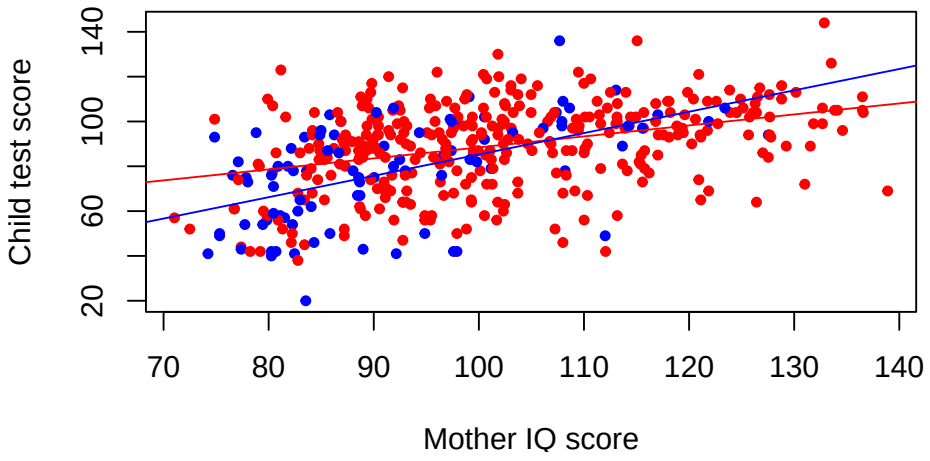
$$\text{kid\_score} = (\hat{\beta}_1 + \hat{\beta}_2) + (\hat{\beta}_3 + \hat{\beta}_4) * \text{mom\_iq},$$

and when $\text{mom\_hs} = 0$, we have the same equation as the no-interaction case:

$$\text{kid\_score} = \hat{\beta}_1 + \hat{\beta}_3 * \text{mom\_iq}.$$

Again, we can use the intercept and slope in each equation to plot the regression line on the corresponding level of mom_hs.

```
fit_4 <- stan_glm(kid_score ~ mom_hs + mom_iq + mom_hs:mom_iq, data=kidiq,
                  refresh=0)
colors <- ifelse(kidiq$mom_hs==1, "red", "blue")
plot(kidiq$mom_iq, kidiq$kid_score,
     xlab="Mother IQ score", ylab="Child test score",
     col=colors, pch=20)
b_hat <- coef(fit_4)
abline(b_hat[1] + b_hat[2], b_hat[3] + b_hat[4], col="red")
abline(b_hat[1], b_hat[3], col="blue")
```



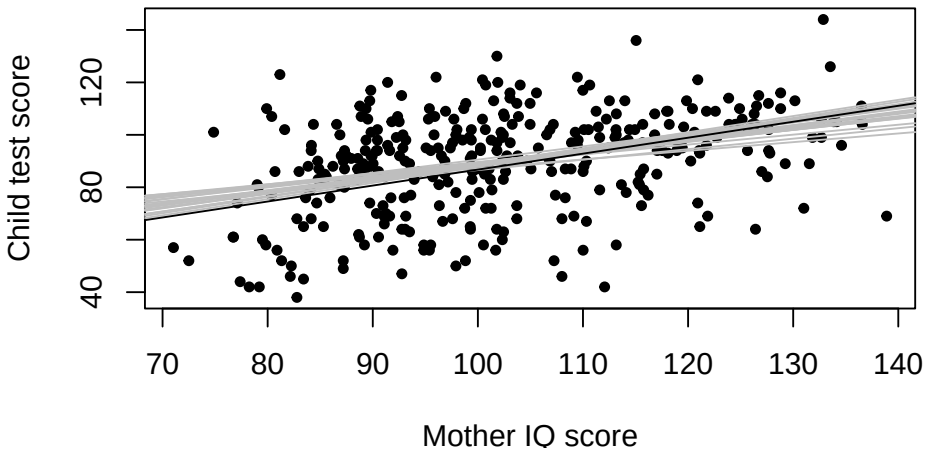
We can see that, unlike the no-interaction case, the lines are not parallel to each other.

6.1.2.3 Plotting the regression with uncertainty

We can use the posterior simulations to represent the uncertainty in the estimated regression coefficients. As an example, we plot the interactive model above on the subset of data with mom_hs = 1 along with 20 simulations drawn from the `stan_glm` fit.

```
sims <- as.matrix(fit_4)
n_sims <- nrow(sims)
mom_iq_1 <- kidiq$mom_iq[kidiq$mom_hs == 1]
kid_score_1 <- kidiq$kid_score[kidiq$mom_hs == 1]
plot(mom_iq_1, kid_score_1,
     xlab="Mother IQ score", ylab="Child test score",
     pch=20)
sims_display <- sample(n_sims, 20)  # sample 20 numbers between 1 and n_sims
```

```
for (i in sims_display){
  # plot with the coefficients from the i-th simulation
  abline(sims[i,1] + sims[i,2], sims[i,3] + sims[i,4], col="gray")
}
# plot with the estimated coefficients
abline(coef(fit_2)[1], coef(fit_2)[2], col="black")
```



6.1.3 Model with multiple predictors

Suppose that the regression equation is

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + ... + \hat{\beta}_p x_p.$$

For each predictor x_k, we can plot the line of y vs. x_k while holding the other predictors fixed at their averages. For example, the equation of y vs. x_1 is

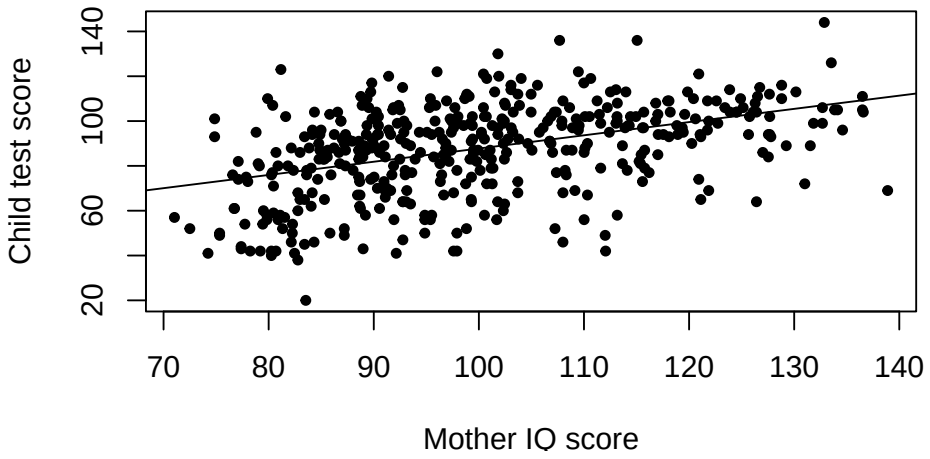
$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 \bar{x}_2 ... + \hat{\beta}_p \bar{x}_p.$$

In the example above, we can plot the line of `kid_score` vs `mom_iq` by fixing `mom_hs` at its average. The equation becomes:

$$\text{kid\_score} = (\hat{\beta}_1 + \hat{\beta}_2 * \overline{\text{mom\_hs}}) + (\hat{\beta}_3 + \hat{\beta}_4 * \overline{\text{mom\_hs}}) * \text{mom\_iq}.$$

```
mom_hs_mean = mean(kidiq$mom_hs)

plot(kidiq$mom_iq, kidiq$kid_score,
     xlab="Mother IQ score", ylab="Child test score",
```

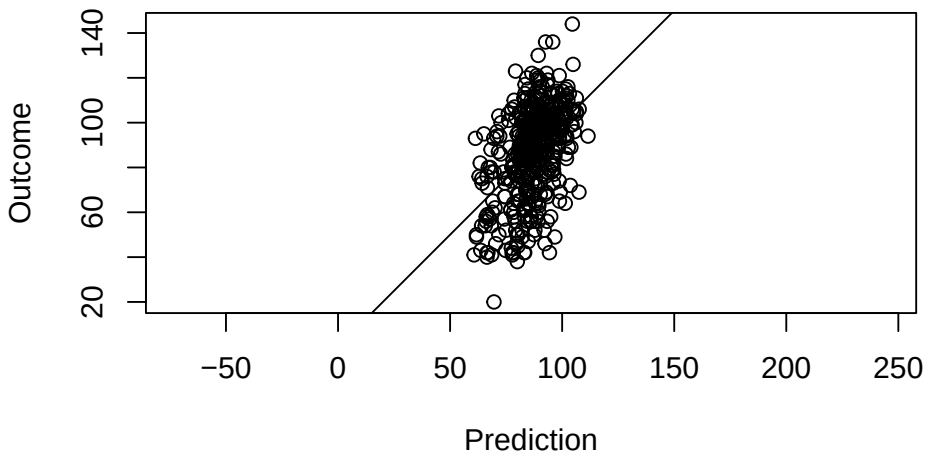
```
     pch=20)
b_hat <- coef(fit_4)
abline(b_hat[1] + b_hat[2] * mom_hs_mean,
       b_hat[3] + b_hat[4] * mom_hs_mean)
```



6.2 Plotting the outcome against the prediction

Another way to check the model's fit is to plot the outcome y against the linear prediction $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + ... + \hat{\beta}_p x_p$. Let's apply this technique to the `KidIQ` data above.

```
kid_score_pred_4 = predict(fit_4)

plot(kid_score_pred_4, kidiq$kid_score,
     xlab="Prediction", ylab="Outcome",
     asp=1)  # set the aspect ratio between x- and y-axis to 1
abline(0, 1)  # plot line y = x
```



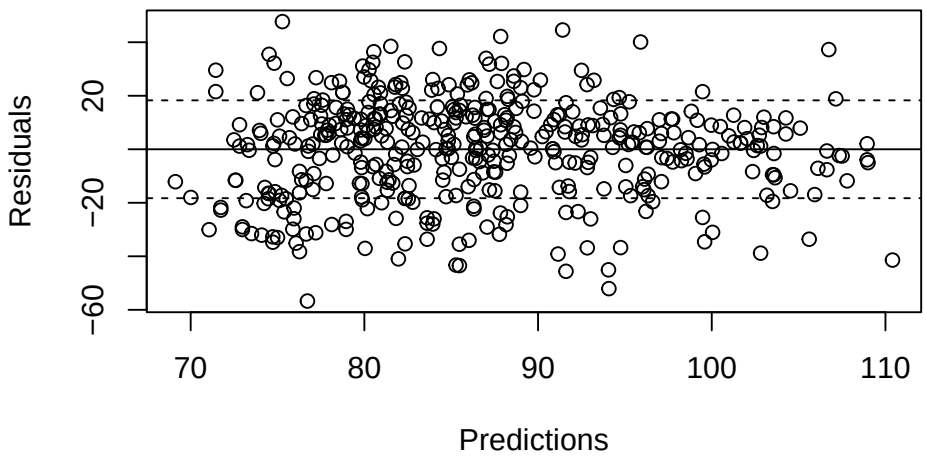
6.3 Residual plots

One way to evaluate the model's fit and independence of errors is to plot the residuals:

$$r_i = y_i - \hat{y}_i,$$

against the predictions $\hat{y}_i$. Let's see a residual plot for the simple regression of `kid_score` vs. `mom_iq`.

```
Predictions = predict(fit_2)

Residuals = kidiq$kid_score - Predictions

plot(Predictions, Residuals)
abline(0, 0)
abline(sigma(fit_2), 0, lty="dashed")  # +1 standard deviation
abline(-sigma(fit_2), 0, lty="dashed")  # -1 standard deviation
```



The residuals are relatively small compared to the outcomes, most of which are between 40-140, and they look sufficiently random; this suggests that the model's errors are independent with zero mean.

6.4 Comparing simulated data to real data

We can also simulate outcomes from the posterior predictive distribution and compare their distribution (that is, their histogram) to that of the original data. As an example, we take a look at `Newcomb` dataset from Stigler (1997), which contains data from an experiment to estimate the speed of light. Here, the outcome y represents the amount of time required for light to travel a distance of 7442 meters and are recorded as deviations from 24800 nanoseconds.

```
newcomb = read.csv("data/newcomb.txt")

head(newcomb)
```

```
   y
1  28
2  26
3  33
4  24
5  34
6 -44
```

We fit a simple normal distribution model on this data.

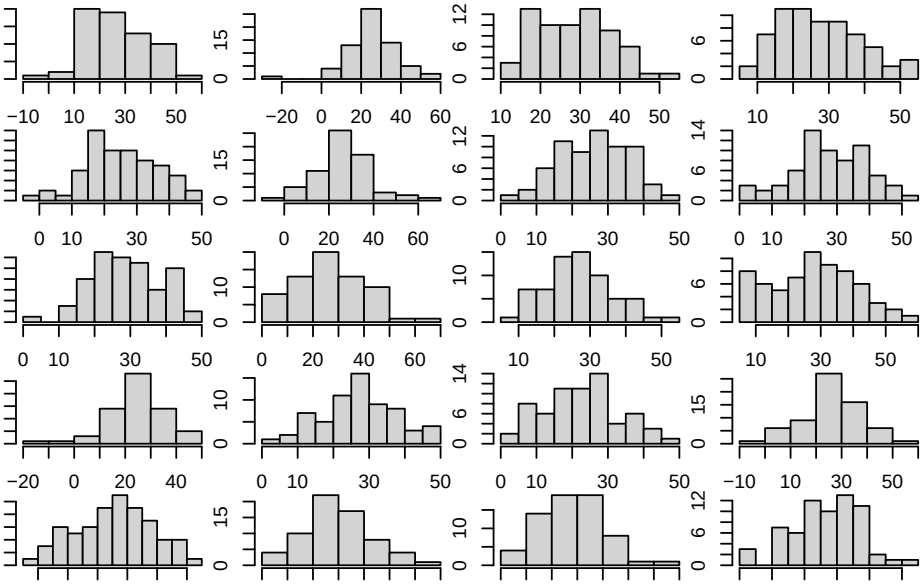
$$y = \beta_0 + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2),$$

which is equivalent to $y \sim \mathcal{N}(\beta_0, \sigma^2)$.

```
n_sample <- nrow(sims)

fit <- stan_glm(y ~ 1, data=newcomb,
                refresh=0)
y_sims <- posterior_predict(fit)
```

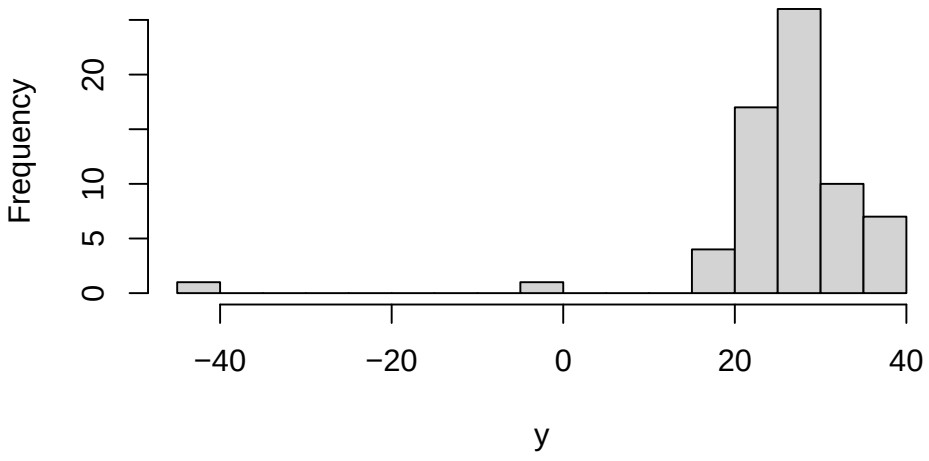
Here, `y_sims` contains 4000 simulations of y. To replicate the original dataset, which has 66 observations, we make 20 datasets, each of which consists of 66 numbers randomly sampled from `y_sims`.

```
par(mfrow=c(5, 4))
par(mar = c(1, 1, 1, 1))
for (s in sample(n_sample, 20)) {
  hist(y_sims[s,], main=NULL, ylab=NULL)
}
```



Now let us compare these histograms with the original data.

```
hist(newcomb$y, main=NULL, xlab="y", breaks=20)
```



The simulated histograms are noticably different than that of the original data, and we can see that the normal model is not suitable for the data. Alternatively, one might use an asymmetric contaminated normal distribution or a symmetric long-tailed distribution.

6.5 Explained variance R^2

The *coefficient of determination* (R^2) is calculated as follows:

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} = 1 - \frac{\text{RSS}}{\text{TSS}} = \frac{\text{TSS} - \text{RSS}}{\text{TSS}},$$

where RSS is the *residual sum of squares* and TSS is the *total sum of squares*.

- TSS measures the variance of the outcome y.
- RSS measures the amount of variance unexplained by the regression.
- Therefore, TSS − RSS measures the amount of variance *explained* by the regression.
- Consequently, R^2 measures the *proportion of variance in y that is explained by the regression.*

To understand R^2 further, we consider two special cases of the linear regression with one predictor: $\hat{y} = \hat{a} + \hat{b}x$.

- If the fitted line is the same as the horizontal line, that is, $\hat{y}_i = \bar{y}$ for all i, then RSS = TSS. So when the fitted model does not explain any of the variance in y, we have $R^2 = 0$.

- Suppose that the fitted line passes through all the points perfectly, so the residuals are all zeros. In this case, RSS $= 0$ and so $R^2 = 1$. Thus, $R^2 = 1$ indicates that the regression line explains all of the variance in y.

6.5.1 Bayesian R^2

There is one problem of using R^2 for Bayesian regression: R^2 is only guaranteed to be non-negative when $\hat{y}_i$ are predictions from the model with OLS coefficients. In general, however, we can have a regression model with a negative \$R^2\$. At an extreme, one can think of a linear model that is very far away from the data. Since the simulated coefficients from the Bayesian regression are not necessarily OLS, R^2 of the corresponding models might be negative. To this end, Gelman et al. (2019) proposed an alternative definition of R^2 for Bayesian regression:

$$R^2_{\text{Bayes}} = \frac{\frac{1}{n-1}\sum_i (\hat{y}_i - \bar{\hat{y}})^2}{\frac{1}{n-1}\sum_i (\hat{y}_i - \bar{\hat{y}})^2 + \sigma^2} = \frac{\text{Explained variance}}{\text{Explained variance} + \text{Residual variance}},$$

where $\bar{\hat{y}}$ is the mean of $\hat{y}_i$*'s. Thus,* R^2_{Bayes} is always between 0 and \$1\$. Again, $R^2_{\text{Bayes}} = 0$ if the fitted model is a horizontal line and R^2_{Bayes} is close to 1 when the explained variance dominates the residual variance.

In practice, we compute R^2_{Bayes} for each simulation of the coefficients and σ to obtain the posterior distribution of R^2_{Bayes}. This can be done in the `rstanarm` library by calling `bayes_R2` on a fitted model. To obtain a point estimate one can compute the median of the simulated values. For example, let us take the model of `KidIQ` with and without an interaction term from above.

```
R2_sims = bayes_R2(fit_3)
print(R2_sims[1:10])
```

```
 [1] 0.2322517 0.2087863 0.2294476 0.1647310 0.2443818 0.1713001 0.1666030
 [8] 0.2082350 0.2492992 0.1665608
```

```
cat("A point estimate of Bayesian R2 is: ", median(R2_sims))
```

```
A point estimate of Bayesian R2 is:  0.2142677
```

```
R2_sims = bayes_R2(fit_4)
print(R2_sims[1:10])
```

```
 [1] 0.1943837 0.2383272 0.2562779 0.2541789 0.2260046 0.2419281 0.2481928
 [8] 0.2294875 0.2716983 0.2399899
```

```
cat("A point estimate of Bayesian R2 is: ", median(R2_sims))
```

```
A point estimate of Bayesian R2 is:  0.2298335
```

The results show that R^2_{Bayes} of the model with an interaction term is larger than that without an interaction term.

6.6 Cross validation

A model should be evaluated on how well they make predictions on new data. However, sometimes we would like to evaluate and compare models without waiting for new data. One can instead hold out a subset of existing data, train the model on the remaining data, and then evaluate on the held-out data: this is called *cross validation*.

6.6.1 Leave-one-out cross validation

In leave-one-out cross validation (LOO), the model is fitted on all but a single data point, and then it is evaluated on that data point. We suggest two ways of evaluating the model:

6.6.1.1 1. Log score and deviance

Suppose that β_i and σ_i are the parameters fitted on the data consisting of all but the i-th data point. The likelihood of the i-th data point (X_i, y_i) is

$$p(y_i \mid \beta_i, \sigma_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{1}{2\sigma_i^2}(y_i - X_i\beta_i)^2\right).$$

A larger likelihood suggests a better fit of the model to this data point. The *log score* is the log of the likelihood without the constant term:

$$-\log\sigma_i - \frac{1}{2\sigma_i^2}(y_i - X_i\beta_i)^2.$$

We perform LOO for all data points. The model's performance is measured by taking the sum of the log scores. This is called the *expected log predictive density* (elpd).

$$\text{elpd} = \sum_{i=1}^{n} -\log\sigma_i - \frac{1}{2\sigma_i^2}(y_i - X_i\beta_i)^2.$$

To compare the performance between two models, we compare their elpd.

The exact computation of elpd is quite slow since it requires fitting the model n times. The `loo` function in R implement an approximation of elpd that is much faster to compute.

As an example, we compute the elpd of the following model:

$$\text{kid\_score} = \text{mom\_iq}.$$

```
fit_1 <- stan_glm(kid_score ~ mom_hs, data=kidiq,
                  refresh=0)

loo_1 <- loo(fit_1)
print(loo_1)
```

```
Computed from 4000 by 434 log-likelihood matrix

         Estimate   SE
elpd_loo  -1914.8 13.8
p_loo         3.1  0.3
looic      3829.6 27.6
------
Monte Carlo SE of elpd_loo is 0.0.

All Pareto k estimates are good (k < 0.5).
See help('pareto-k-diagnostic') for details.
```

The output can be interpreted as follows:

- `elpd_loo` is the estimated elpd along with a standard error representing uncertainty due to using only 434 data points.
- `p_loo` is the estimated "effective number of parameters" in the model, which is essentially the number of parameters that accounts for information in the prior and the data. The above model has 3 parameters, so it makes sense that `p_loo` is close to 3 here.
- `looic` is the LOO information criterion, which is $-2\times$`elpd_loo`.

Let us compare `fit_1` with `fit_3`: $\text{kid\_score} = \text{mom\_iq} + \text{mom\_hs}$.

```
loo_3 <- loo(fit_3)
print(loo_3)
```

```
Computed from 4000 by 434 log-likelihood matrix

         Estimate   SE
elpd_loo  -1876.0 14.2
```

```
p_loo         4.0  0.4
looic      3752.0 28.5
------
Monte Carlo SE of elpd_loo is 0.0.

All Pareto k estimates are good (k < 0.5).
See help('pareto-k-diagnostic') for details.
```

We can see that `fit_3`'s elpd of -1875.9 is higher than `fit_1`'s elpd of -1914.8. This suggests that `fit_3`'s posterior predictive distribution matches the data better than that of `fit_1`.

To also obtain the standard error of the difference in elpd between `fit_3` and `fit_1`, we can use the `loo_compare` function.

```
loo_compare(loo_3, loo_1)
```

```
      elpd_diff se_diff
fit_3   0.0       0.0
fit_1 -38.8       8.3
```

The output tells us that `fit_1`'s elpd is 39.0 smaller than that of `fit_3`, with the standard error of the difference equals 8.4. As a rule of thumb, if `eldp_diff` is greater than 4, the number of observations is greater than 100, and the model is not badly misspecified, then `se_diff` is a reliable measure of uncertainty in the difference between `elpd`'s.

Let us compare the `eldp` between `fit_3`, the model with two predictors and no interaction term, and `fit_4`, the model with two predictors and an interaction term.

```
loo_4 <- loo(fit_4)

loo_compare(loo_3, loo_4)
```

```
      elpd_diff se_diff
fit_4  0.0       0.0
fit_3 -3.4       2.7
```

The difference is less than 4, so there is no clear improvement by adding the interaction term.

Chapter 7

Logarithmic transformations

We might want to try a logarithmic transformation when

- Additivity and linearity are not reasonable assumptions.
- The outcomes are all positive.

In this case, we run the regression with $\log y_i$ as target values:

$$\log y_i = \beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip} + \varepsilon_i.$$

To obtain a model that predicts the outcome from the input, we exponentiate both sides of the equation.

$$\begin{aligned} y_i &= e^{\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip} + \varepsilon_i} \\ &= B_0 B_1^{X_{i1}} ... B_p^{X_{ip}} E_i, \end{aligned}$$

where $B_0 = e^{\beta_0}, B_1 = e^{\beta_1}, ...$

Consider the logarithmic regression on the `Earnings` data. This can be done in are by simply replacing `earn` in the formula by `log(earn)` (notice that we only regress on the subset of earn > 0.

```
library(rstanarm)
```

```
earnings = read.csv("data/earnings.csv")

head(earnings)
```

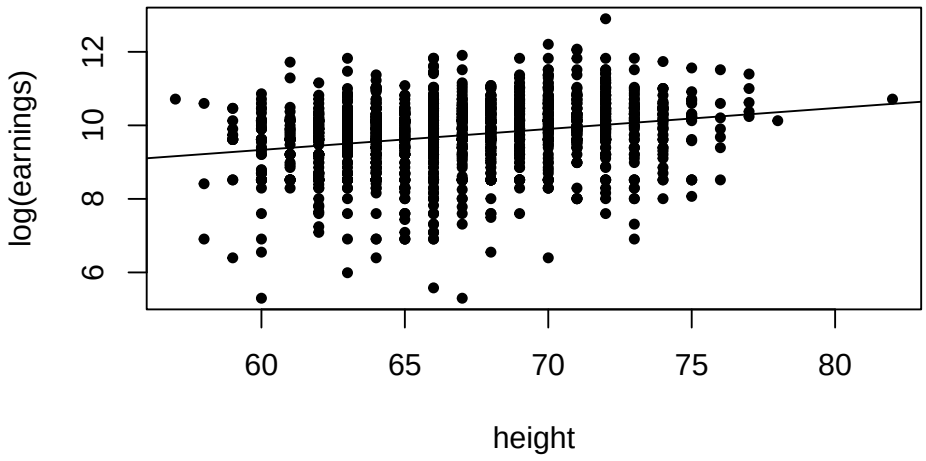
```
  height weight male  earn earnk ethnicity education mother_education
1     74    210    1 50000    50     White        16               16
2     66    125    0 60000    60     White        16               16
3     64    126    0 30000    30     White        16               16
4     65    200    0 25000    25     White        17               17
5     63    110    0 50000    50     Other        16               16
6     68    165    0 62000    62     Black        18               18
  father_education walk exercise smokenow tense angry age
1               16    3        3        2     0     0  45
2               16    6        5        1     0     0  58
3               16    8        1        2     1     1  29
4               NA    8        1        2     0     0  57
5               16    5        6        2     0     0  91
6               18    1        1        2     2     2  54
```

```
logmodel_1 <- stan_glm(log(earn) ~ height, data=earnings,
                       subset=earn>0, refresh=0)
print(logmodel_1)
```

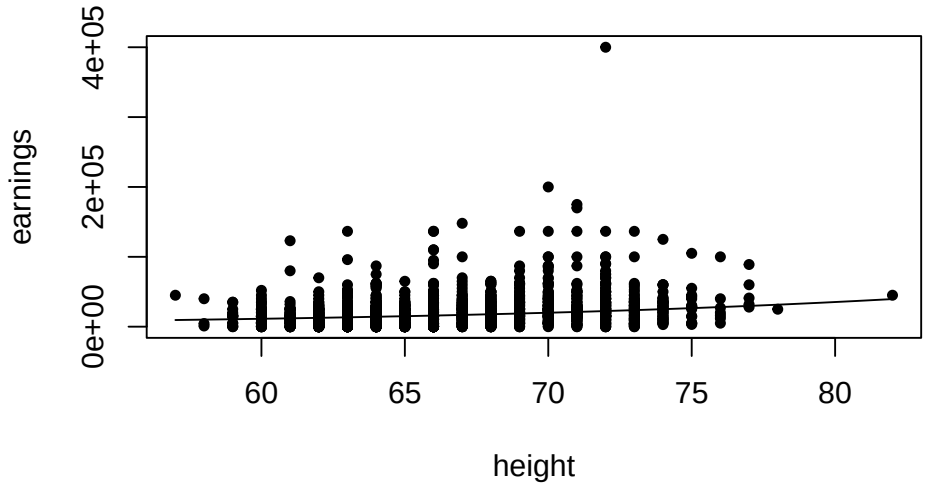
```
stan_glm
 family:       gaussian [identity]
 formula:      log(earn) ~ height
 observations: 1629
 predictors:   2
 subset:       earn > 0
------
            Median MAD_SD
(Intercept) 5.9    0.4
height      0.1    0.0

Auxiliary parameter(s):
      Median MAD_SD
sigma 0.9    0.0

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

Here are what the plot of this model on the log scale and the original scale.

```r
plot(earnings$height, log(earnings$earn), pch=20,
     xlab="height", ylab="log(earnings)")
abline(coef(logmodel_1))
```




```r
beta = coef(logmodel_1)

plot(earnings$height, earnings$earn, pch=20,
     xlab="height", ylab="earnings")
curve(exp(beta[1] + beta[2]*x), add=TRUE)
```



In the latter plot, the curve is moving upwards like an exponential function.

7.1 Interpreting the coefficients

The fitted model from the logarithmic regression above is

$$\log(\text{earnings}) = 5.9 + 0.1 * \text{height},$$

This model suggests that, for two people, say A and B, whose heights differ by 1, their log-earnings differ by 0.1 on average:

$$\begin{aligned}
\log(\text{earnings}_A) - \log(\text{earnings}_B) &= 0.1 \\
\log\left(\frac{\text{earnings}_A}{\text{earnings}_B}\right) &= 0.1 \\
\frac{\text{earnings}_A}{\text{earnings}_B} &= e^{0.1} \approx 1.1,
\end{aligned}$$

Here, we have used an approximation $e^x \approx 1 + x$, which is reasonably accurate for $x < 1$.

From this, we can interpret the slope of 0.1 as follows:

> 1 inch increase in height corresponds to an expected 10% increase in earnings.

Such possible interpretation is why we have (implicitly) used the logarithms base e instead of base \$10\$. If we were to use base 10, we would instead obtain $\frac{\text{earnings}_A}{\text{earnings}_B} = 10^{\hat{\beta}_1}$ where $\hat{\beta}_1$ is the slope from fitting the regression with logarithms base 10. In this case, we cannot estimate the percentage increase in earnings just by looking at the coefficient.

7.1.1 When there are zero-valued outcomes

Sometimes, we might face a situation where some of the outcomes are zeros, which cannot take the logarithm directly. One way to model such data is by running a *classification* model that can classify whether an instance has non-zero outcome (for example, a linear regression model, which will be introduced next chapter). We then use this model to tell us whether a new data point has non-zero outcome. If that is the case, then we use the fitted logarithmic model to predict the actual value of the outcome.

7.2 Model checking with simulations

We will compare the logarithmic regression with the linear regression by replicating a dataset from each model and compare it to the observed dataset.

We first fit the linear model.

```
fit_1 <- stan_glm(earn ~ height, data=earnings,
                  subset=earn>0, refresh=0)
```

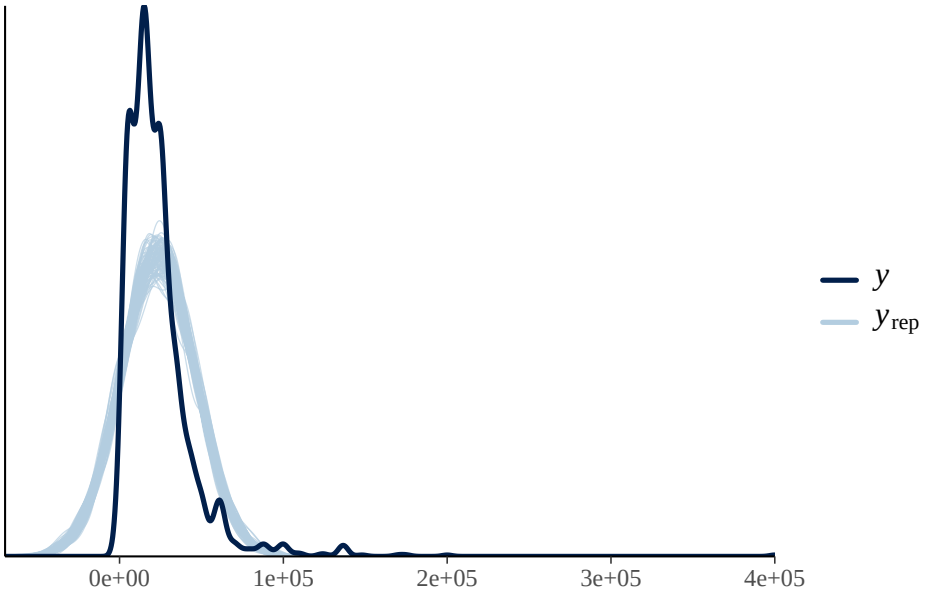
and then we simulate outcomes from the posterior predictive distribution.

```
yrep_1 <- posterior_predict(fit_1)
n_sims <- nrow(yrep_1)  # number of rows in the simulation
subset <- sample(n_sims, 100)  # randomly pick 100 rows
yrep_1_100 <- yrep_1[subset,]  # pick rows of yrep_1 by row indices in subset
```

There is a convenient library `bayesplot` that allows us to make density plots of the simulations and the data via `ppc_dens_overlay` function.

```
library(bayesplot)
```

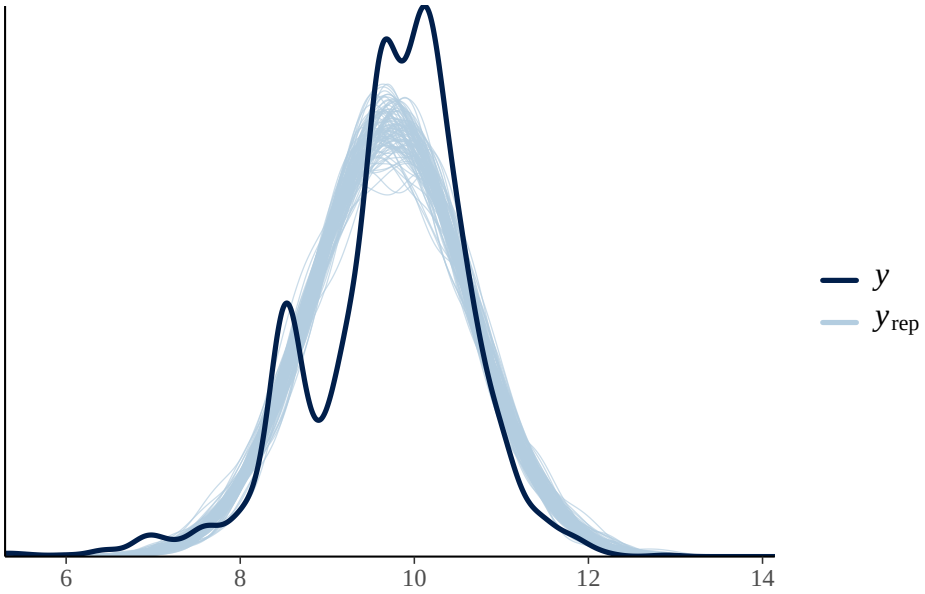
```
ppc_dens_overlay(earnings$earn[earnings$earn>0], yrep_1[subset,])
```



We can see that the distributions do not quite match as the observed data is more concentrated and is non-negative.

Now let us try the same for the logarithmic model.

```
yrep_log_1 <- posterior_predict(logmodel_1)
n_sims <- nrow(yrep_log_1)  # number of rows in the simulation
subset <- sample(n_sims, 100)  # randomly pick 100 rows
yrep_log_1_100 <- yrep_log_1[subset,]  # pick rows by list of indices in subset

ppc_dens_overlay(log(earnings$earn[earnings$earn>0]), yrep_log_1_100)
```



Visually, the logarithmic model has a better fit than the linear model.

7.3 elpd for the logarithmic regression

First, let us look at the elpd between the linear and logarithmic model.

```
loo_1 = loo(fit_1)
```

```
Warning: Found 1 observation(s) with a pareto_k > 0.7. We recommend calling 'loo' again with
```

```
print(loo_1)
```

```
Computed from 4000 by 1629 log-likelihood matrix

         Estimate    SE
elpd_loo -18608.4 165.4
```

```
p_loo        26.5  19.7
looic     37216.7 330.7
------
Monte Carlo SE of elpd_loo is NA.

Pareto k diagnostic values:
                         Count Pct.    Min. n_eff
(-Inf, 0.5]   (good)     1628  99.9%   876
 (0.5, 0.7]   (ok)          0   0.0%   <NA>
   (0.7, 1]   (bad)         0   0.0%   <NA>
   (1, Inf)   (very bad)    1   0.1%   7
See help('pareto-k-diagnostic') for details.
```

```
loo_log_1 = loo(logmodel_1)
print(loo_log_1)
```

```
Computed from 4000 by 1629 log-likelihood matrix

         Estimate   SE
elpd_loo  -2100.6 38.8
p_loo         3.9  0.4
looic      4201.1 77.6
------
Monte Carlo SE of elpd_loo is 0.0.

All Pareto k estimates are good (k < 0.5).
See help('pareto-k-diagnostic') for details.
```

Notice that `elpd_loo` between these two models are on different scales; this is because the likelihood of the linear model and the transformed model are totally different, and we have to make correction for this difference in the computation of LOO. Initially, the equation for a logarithmic model is:

$$\log y = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p + \varepsilon = X\beta + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \sigma^2).$$

Define a new random variable $z = \log y$. The equation above tells use that

$$z \sim \mathcal{N}(X\beta, \sigma^2).$$

To obtain the elpd of the logarithmic model, we need to compute the density $p(y \mid \beta, \sigma, X)$. This can be achieved by applying the change-of-variable formula between two variables y and z.

$$p(y \mid \beta, \sigma, X) = p(z \mid \beta, \sigma, X) \left| \frac{dz}{dy} \right| = \frac{1}{y} p(z \mid \beta, \sigma, X).$$

Here, $|y| = y$ since we assume that y is positive. Thus, for a held-out data point (X_i, y_i), with $z_i = \log y_i$, the log-likelihood of this point is

$$\log p(y_i \mid \beta, \sigma, X_i) = \underbrace{\log p(z_i \mid \beta, \sigma, X_i)}_{\text{elpd of } \log y_i} - \log y_i.$$

In other words, we can make a correction for the elpd of the logarithmic model by subtracting $\log y_i$ for each instance (X_i, y_i).

We can inspect the elpd of each instance by calling `pointwise` attribute of the `loo` result. The following code shows the first five rows of the elpd from the logarithmic regression:

```
print(loo_log_1$pointwise[1:5,])
```

```
       elpd_loo mcse_elpd_loo        p_loo    looic influence_pareto_k
[1,] -1.1011337  0.0006686739 0.0017736218 2.202267         0.01135110
[2,] -1.9367326  0.0007254570 0.0020959857 3.873465         0.06864000
[3,] -1.1537610  0.0004255231 0.0007318929 2.307522        -0.10228210
[4,] -0.9577587  0.0003146193 0.0003845926 1.915517        -0.15439004
[5,] -1.9174803  0.0009185109 0.0034125379 3.834961        -0.07356066
```

From this, we can subtract $\log y_i$ (`earn` in this case) from the `elpd_loo` (first column).

```
loo_log_1$pointwise[,1] <- loo_log_1$pointwise[,1] - log(earnings$earn[earnings$earn>0])
```

We can now sum `elpd_loo` over all instances.

```
elpd_with_correction <- sum(loo_log_1$pointwise[,1])
print(elpd_with_correction)
```

```
[1] -17932.98
```

The elpd is now in the same scale as the linear model. We can now compare the elpd between the linear and logarithmic model.

```
loo_compare(loo_log_1, loo_1)
```

```
Warning: Not all models have the same y variable. ('yhash' attributes do not
match)
```

```
           elpd_diff se_diff
logmodel_1    0.0       0.0
fit_1      -675.4     153.1
```

7.4 Log-log model

If the log transformation is applied to both an input variable and the outcome, the coefficient can be interpreted as the percentage difference in y per percentage difference in x.

Let us try this technique on the `Earnings` data.

```
logmodel_5 <- stan_glm(log(earn) ~ log(height) + male, data=earnings,
                       subset=earn>0, refresh=0)

print(logmodel_5)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      log(earn) ~ log(height) + male
 observations: 1629
 predictors:   3
 subset:       earn > 0
------
            Median MAD_SD
(Intercept) 2.8    2.2
log(height) 1.6    0.5
male        0.4    0.1

Auxiliary parameter(s):
      Median MAD_SD
sigma 0.9    0.0

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

- The coefficient 1.6 of log(height) implies that 1% increase in height corresponds to 1.6% increase in earnings.
- The coefficient 0.4 of male implies that, on average, a male earns 40% more than a female with the same height.

Chapter 8

Comparing regression models

General principles

1. Use prior knowledge to pick relevant input variables.
2. For inputs that have large effects, consider including their interactions.
3. If the coefficient of a predictor has a small standard error, then we might want to keep it in the model.
4. If the coefficient of a predictor has a large standard error, and there is no reason for the predictor in the model, then we might want to remove it.
5. If a coefficient contradicts the reality (for example, a negative coefficient for education in an income regression),
   - If the standard error is large, then the unusual estimate can be explained from its uncertainty.
   - If the standard error is small, try to understand how it could happen. In the income vs. education example, the negative coefficient might be because the data was collected from a subpopulation in which the more educated people are younger and hence tend to have lower average income.

8.1 Example: predicting the yields of mesquite bushes

We apply the model checking techniques to `Mesquite` data, which is used to estimate the weight (in grams) of yield harvested from a mesquite bush.

Figure 8.1: A mesquite tree

```
mesquite <- read.table("data/mesquite.dat",
                       header=TRUE)

head(mesquite)
```

```
  obs group diam1 diam2 total_height canopy_height density weight
1   1   MCD   1.8  1.15         1.30          1.00       1  401.3
2   2   MCD   1.7  1.35         1.35          1.33       1  513.7
3   3   MCD   2.8  2.55         2.16          0.60       1 1179.2
4   4   MCD   1.3  0.85         1.80          1.20       1  308.0
5   5   MCD   3.3  1.90         1.55          1.05       1  855.2
6   6   MCD   1.4  1.40         1.20          1.00       1  268.7
```

The input variables are:

Variable name	Description
`diam1`	diameter along the longer axis of the canopy (meters)
`diam2`	diameter along the shorter axis of the canopy (meters)
`canopy_height`	height of the canopy (meters)
`total_height`	total height of the bush
`density`	number of primary stems per plant unit
`group`	`MCD` or `ALS`, indicating two different times of measurement

To start off, we regress `weight` on all of the predictors.

```
library(rstanarm)
```

```
fit_1 <- stan_glm(weight ~ diam1 + diam2 + canopy_height +
                  total_height + density + group,
                  data=mesquite, refresh=0)

print(fit_1)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      weight ~ diam1 + diam2 + canopy_height + total_height + density +
       group
 observations: 46
 predictors:   7
------
             Median  MAD_SD
(Intercept)  -1091.3   183.8
diam1          190.5   111.8
diam2          370.8   128.4
canopy_height  355.2   209.1
total_height  -103.7   188.6
density        131.8    34.4
groupMCD       362.7    99.9

Auxiliary parameter(s):
      Median MAD_SD
sigma 272.7   31.7

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

We evaluate this model using elpd.

```
loo_1 <- loo(fit_1)
```

```
Warning: Found 2 observation(s) with a pareto_k > 0.7. We recommend calling 'loo' again with
```

```
print(loo_1)
```

```
Computed from 4000 by 46 log-likelihood matrix
```

```
         Estimate   SE
elpd_loo   -334.1 12.6
p_loo        16.0  8.5
looic       668.2 25.1
------
Monte Carlo SE of elpd_loo is NA.

Pareto k diagnostic values:
                         Count Pct.    Min. n_eff
(-Inf, 0.5]   (good)     43    93.5%   779
 (0.5, 0.7]   (ok)        1     2.2%   136
   (0.7, 1]   (bad)       1     2.2%   18
   (1, Inf)   (very bad)  1     2.2%   4
See help('pareto-k-diagnostic') for details.
```

Three instances with $\hat{k} > 0.7$ indicate that the approximate elpd computation might not be accurate. Aki Vehtari has provided a quick guideline on how to diagnose the model from `p_loo` and Pareto $\hat{k}$ values:

- If all Pareto k small, model is likely to be ok (although there can be better models)
- If high Pareto k values
  - If p_loo « the number of parameters p, then the model is likely to be misspecified. PPC is likely to detect the problem, too. Try using overdispersed model, or add more structural information (nonlinearity, mixture model, etc.).
  - If p_loo > the number of parameters p, then the model is likely to be badly misspecified. If the number of parameters p«n, then PPC is likely to detect the problem, too. Case example https://rawgit.com/avehtari/modelselection_tutorial/master/roaches.html 189
  - If p_loo > the number of parameters p, then the model is likely to be badly misspecified. If the number of parameters p is relatively large compared to the number of observations p>n/5 (more accurately we should count number of observations influencing each parameter as in hierarchical models some groups may have small n and some groups large n), it is possible that PPC doesn't detect the problem. Case example Recommendations for what to do when k exceeds 0.5 in the loo package? 299
  - If p_loo < the number of parameters p and the number of parameters p is relatively large compared to the number of observations p>n/5, it is likely that model is so flexible or population prior is so weak that it's

difficult to predict for left out observation even if the model is true one. Case example is the simulated 8 schools in https://arxiv.org/abs/1507.04544 90 and Gaussian processes and spatial models with short correlation lengths.

Let us compare try fitting a logarithmic model. Again, since `group` is categorical, we do not apply the logarithmic transformation to this predictor.

```
fit_2 <- stan_glm(log(weight) ~ log(diam1) + log(diam2) +
                  log(canopy_height) + log(total_height) +
                  log(density) + group,
                  data=mesquite, refresh=0)

print(fit_2)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      log(weight) ~ log(diam1) + log(diam2) + log(canopy_height) +
       log(total_height) + log(density) + group
 observations: 46
 predictors:   7
------
                  Median MAD_SD
(Intercept)       4.8    0.2
log(diam1)        0.4    0.3
log(diam2)        1.1    0.2
log(canopy_height) 0.4    0.3
log(total_height)  0.4    0.3
log(density)      0.1    0.1
groupMCD          0.6    0.1

Auxiliary parameter(s):
      Median MAD_SD
sigma 0.3    0.0

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

```
loo_2 <- loo(fit_2)
```

```
Warning: Found 1 observation(s) with a pareto_k > 0.7. We recommend calling 'loo' again with
```

```
  print(loo_2)
```

```
Computed from 4000 by 46 log-likelihood matrix

         Estimate   SE
elpd_loo    -19.4  5.3
p_loo         7.6  1.6
looic        38.9 10.7
------
Monte Carlo SE of elpd_loo is NA.

Pareto k diagnostic values:
                         Count Pct.    Min. n_eff
(-Inf, 0.5]   (good)     43    93.5%   1046
 (0.5, 0.7]   (ok)        2     4.3%   427
   (0.7, 1]   (bad)       1     2.2%   395
   (1, Inf)   (very bad)  0     0.0%   <NA>
See help('pareto-k-diagnostic') for details.
```

The approximate elpd of this model is more reliable than the previous one. Now we adjust `loo_2` for the logarithmic transformation by subtracting the log score of each instance by the logarithm of its `weight`.

```
  loo_2a = loo_2

  loo_2a$pointwise[,1] <- loo_2a$pointwise[,1] - log(mesquite$weight)
  sum(loo_2a$pointwise[,1])
```

```
[1] -291.7312
```

```
  loo_compare(loo_1, loo_2a)
```

```
Warning: Not all models have the same y variable. ('yhash' attributes do not
match)
```

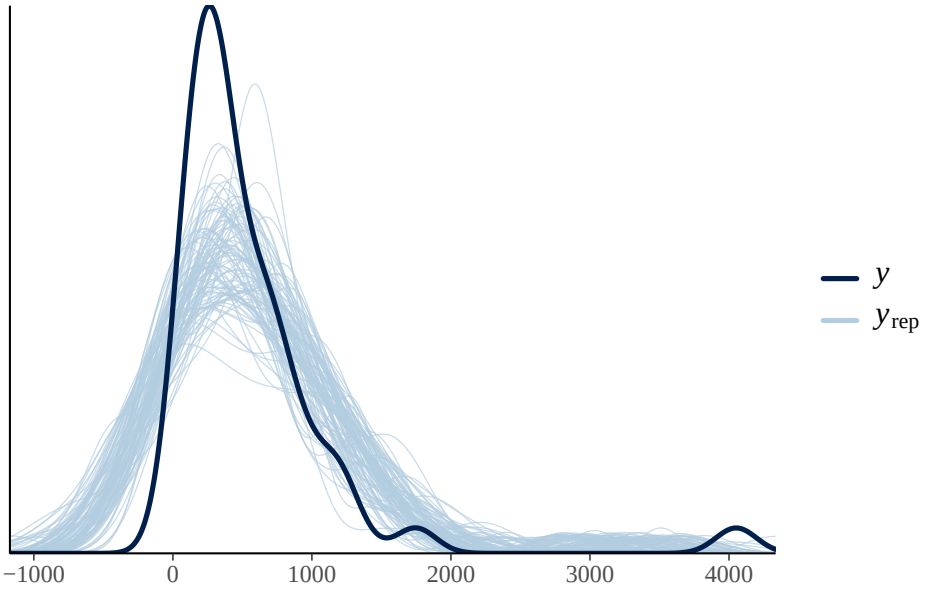
```
      elpd_diff se_diff
fit_2   0.0       0.0
fit_1 -42.4      10.9
```

The elpd of the logarithmic model is larger than the linear model, so we continue with the logarithmic model.

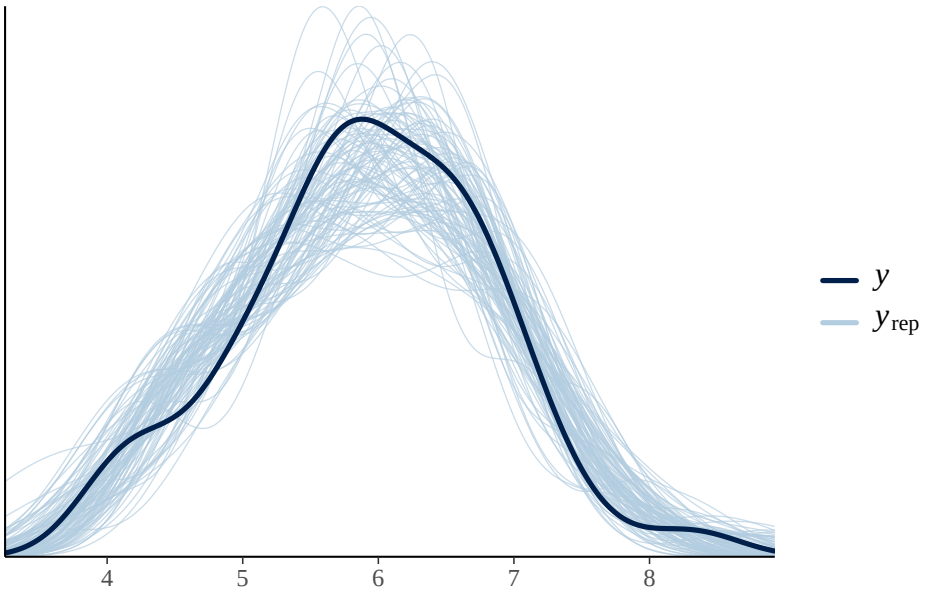
We can also simulated a dataset from each model and compare it with the original dataset.

```
library(bayesplot)
```

```
yrep_1 <- posterior_predict(fit_1)
n_sims <- nrow(yrep_1)
subset <- sample(n_sims, 100)
ppc_dens_overlay(mesquite$weight, yrep_1[subset,])
```




```
yrep_2 <- posterior_predict(fit_2)
ppc_dens_overlay(log(mesquite$weight), yrep_2[subset,])
```



The density plots show that the fit on the log scale is much better.

8.1.1 Constructing a simpler model

We have been throwing all predictors in our model. But sometimes we might want to look for a simpler model that is more interpretable. For example, we can create a new predictor that measures the volume of the canopy:

$$\texttt{canopy\_volume} = \texttt{diam1} * \texttt{diam2} * \texttt{canopy\_height}.$$

The technique of creating a new predictor from the old ones, in machine learning terms, is called *feature engineering*.

```
mesquite$canopy_volume <- mesquite$diam1 * mesquite$diam2 * mesquite$canopy_height

fit_3 <- stan_glm(log(weight) ~ log(canopy_volume), data=mesquite,
                  refresh=0)

print(fit_3)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      log(weight) ~ log(canopy_volume)
 observations: 46
 predictors:   2
```

```
------
                  Median MAD_SD
(Intercept)       5.2    0.1
log(canopy_volume) 0.7    0.1

Auxiliary parameter(s):
      Median MAD_SD
sigma 0.4    0.0

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

Let us compare this model with the previous logarithm model with all predictors.

```
loo_3 <- loo(fit_3)

loo_compare(loo_2, loo_3)
```

```
      elpd_diff se_diff
fit_2  0.0       0.0
fit_3 -7.2       5.0
```

There is only 7.4 difference in the estimated elpd but this new model is much easier to interpret.

Let us add more predictors: one is `canopy_shape`, which measures the ratio between the two diameters

$$\texttt{canopy\_shape} = \texttt{diam1}/\texttt{diam2},$$

and the others are `total_height`, `density` and `group`. We take an educated guess that having the ratio of diameters close to one is a sign of a healthy mesquite bush; thus if the predictors are not correlated, we expect the coefficient of `canopy_shape` to be negative.

```
mesquite$canopy_shape <- mesquite$diam1 / mesquite$diam2

fit_4 <- stan_glm(log(weight) ~ log(canopy_volume) + log(canopy_shape) +
                  log(total_height) + log(density) + group,
                  data=mesquite, refresh=0)

print(fit_4)
```

```
stan_glm
 family:       gaussian [identity]
```

```
 formula:      log(weight) ~ log(canopy_volume) + log(canopy_shape) + log(total_height) +
       log(density) + group
 observations: 46
 predictors:   6
------
                   Median MAD_SD
(Intercept)         4.9    0.1
log(canopy_volume)  0.7    0.1
log(canopy_shape)  -0.5    0.2
log(total_height)   0.2    0.3
log(density)        0.1    0.1
groupMCD            0.6    0.1

Auxiliary parameter(s):
      Median MAD_SD
sigma 0.3    0.0

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

As before, we will compare with model with the full logarithmic model.

```
loo_4 <- loo(fit_4)

loo_compare(loo_2, loo_4)
```

```
      elpd_diff se_diff
fit_4  0.0       0.0
fit_2 -0.1       1.2
```

The estimated difference of eldp is insignificant. However, the standard errors of `total_height` and `density` are large, so we might want to remove these predictors from the model.

Finally, we are left with a model with three predictors:

```
fit_5 <- stan_glm(log(weight) ~ log(canopy_volume) + log(canopy_shape) +
                  group, data=mesquite, refresh=0)

print(fit_5)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      log(weight) ~ log(canopy_volume) + log(canopy_shape) + group
 observations: 46
```

```
 predictors:   4
------
                  Median MAD_SD
(Intercept)        4.9    0.1
log(canopy_volume) 0.8    0.1
log(canopy_shape) -0.4    0.2
groupMCD           0.6    0.1

Auxiliary parameter(s):
      Median MAD_SD
sigma 0.3    0.0

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

```
loo_5 <- loo(fit_5)

loo_compare(loo_2, loo_5)
```

```
      elpd_diff se_diff
fit_5  0.0       0.0
fit_2 -1.4       1.5
```

In the end, we obtain a simple model with three predictors that performs as well as the full logarithmic model.

We can try adding more predictors or interaction terms but it would take a substantial amount of work to find a model that performs significantly better. Alternatively, one can try a different prior or a more complex model, such as a multilevel model.

8.2 Different priors for the coefficients

In the previous section, we have implicitly used the weakly informative prior for the coefficients.

We work on an example of predicting grades from a sample of high school students from Portugal.

```
data <- read.csv("data/student-merged.csv")

head(data)
```

```
  G1mat G2mat G3mat G1por G2por G3por school sex age address famsize Pstatus
1     7    10    10    13    13    13      0   0  15       0       0       1
```

```
2     8     6     5    13    11    11      0   0  15       0       0       1
3    14    13    13    14    13    12      0   0  15       0       0       1
4    10     9     8    10    11    10      0   0  15       0       0       1
5    10    10    10    13    13    13      0   0  15       0       0       1
6    12    12    11    11    12    12      0   0  15       0       0       1
  Medu Fedu traveltime studytime failures schoolsup famsup paid activities
1    1    1          2         4        1         1      1    1          1
2    1    1          1         2        2         1      1    0          0
3    2    2          1         1        0         1      1    1          1
4    2    4          1         3        0         1      1    1          1
5    3    3          2         3        2         0      1    1          1
6    3    4          1         3        0         1      1    1          1
  nursery higher internet romantic famrel freetime goout Dalc Walc health
1       1      1        1        0      3        1     2    1    1      1
2       0      1        1        1      3        3     4    2    4      5
3       1      1        0        0      4        3     1    1    1      2
4       1      1        1        0      4        3     2    1    1      5
5       1      1        1        1      4        2     1    2    3      3
6       1      1        1        0      4        3     2    1    1      5
  absences
1        2
2        2
3        8
4        2
5        8
6        2
```

We will predict the third period math grade given the other predictors.

```
predictors <- c("school","sex","age","address","famsize","Pstatus","Medu","Fedu",
"traveltime","studytime","failures","schoolsup","famsup","paid","activities",
"nursery", "higher", "internet", "romantic","famrel","freetime","goout","Dalc",
"Walc","health","absences")
data_G3mat <- subset(data, subset=G3mat>0, select=c("G3mat",predictors))
```

Let us try the standard regression model

```
fit1 <- stan_glm(G3mat ~ ., data=data_G3mat, refresh=0)
```

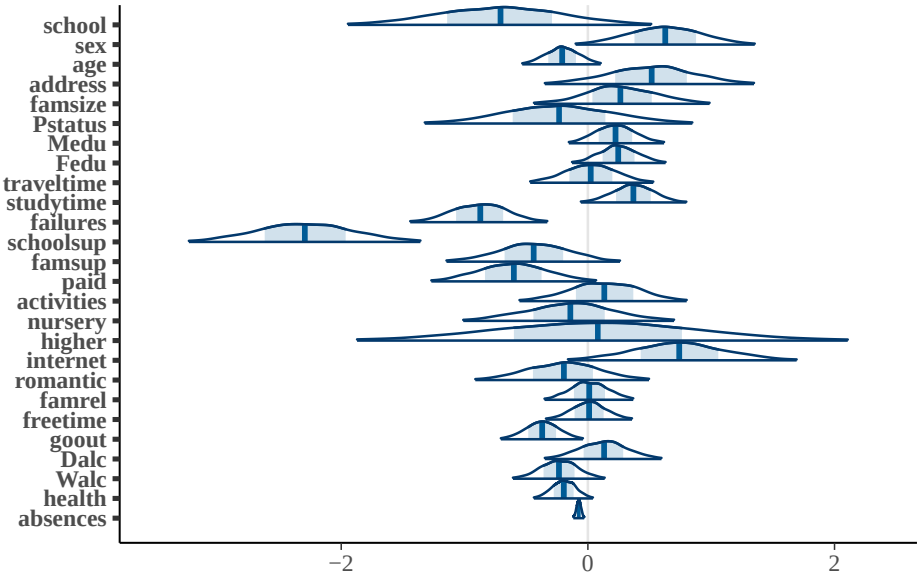
We plot the posterior distributions of the coefficients using the `mcmc_areas` function from `bayesplot` library.

p1 <- mcmc_areas(as.matrix(fit1), pars=vars(-'(Intercept)',-sigma),

prob_outer=0.95, area_method = "scaled height")

p1

```
p1 <- mcmc_areas(as.matrix(fit1), pars=vars(-'(Intercept)',-sigma),
                 prob_outer=0.95, area_method = "scaled height")
p1
```



The different amounts of uncertainty make it difficult to compare the coefficients. For example, it is really hard to see if `absences` is more relevant than `health`.

To compare between two predictors, we have to transform them so that they share the same scale. The common scaling technique is *standardization*, which consists of subtracting the observed values of each predictor by the mean, and then dividing by the sample standard deviation:

$$x \to \frac{x - \bar{x}}{\mathrm{sd}(x)}.$$

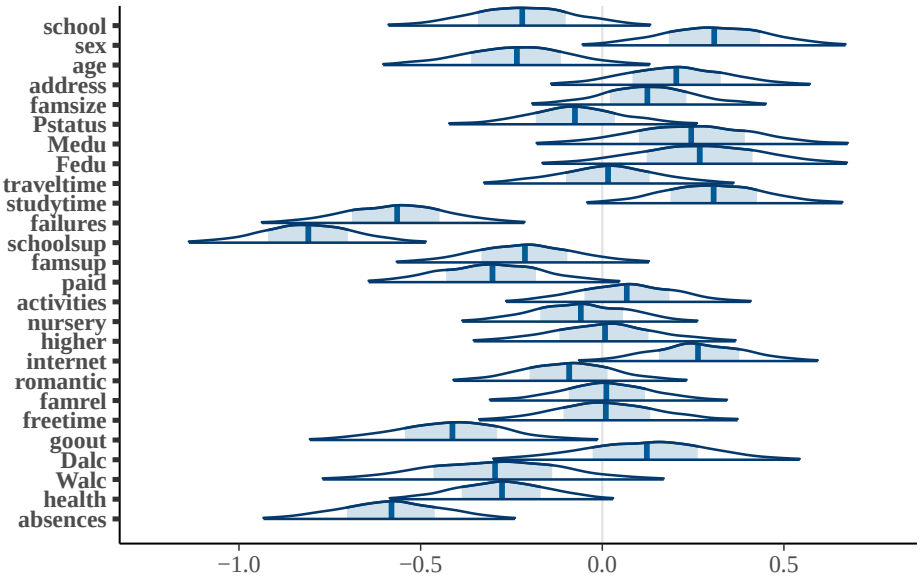
With this transformation, all predictors now have a mean of zero and a sample standard deviation of one.

In R, we can standardize all predictors by using the `scale` function.

```
datastd_G3mat <- data_G3mat
datastd_G3mat[,predictors] <-scale(data_G3mat[,predictors])
```

Now, let us fit the model on the standardized data and plot the coefficients again.

```
fit2 <- stan_glm(G3mat ~ ., data=datastd_G3mat, refresh=0)

p2 <- mcmc_areas(as.matrix(fit2), pars=vars(-'(Intercept)',-sigma),
                 prob_outer=0.95, area_method = "scaled height")
p2
```



The standard errors of the coefficients are now similar to each others. We can see that `absences` is more relevant than many of the predictors.

8.2.1 Priors for variable selection

To obtain a simpler model, we may assume that only some of the predictors are relevant, while the other predictors are negligible. One way to insert this assumption into the model is by using a prior whose distribution has a sharp peak around zero; examples of such priors are *regularized horseshoe prior* and *Laplace prior*.

8.2.1.1 Regularized horseshoe prior

The regularized horseshoe prior, as shown in the plot above, consists of prior $\mathcal{N}(0, \tau^2\lambda_j^2)$ for the j-th coefficient. Here, and λ_j is with the following priors:

1. τ is a *global scale* that shrinks all β_j towards zero. The prior for τ is

$$\text{Half-Cauchy}\,(0, \texttt{global\_scale})\,.$$

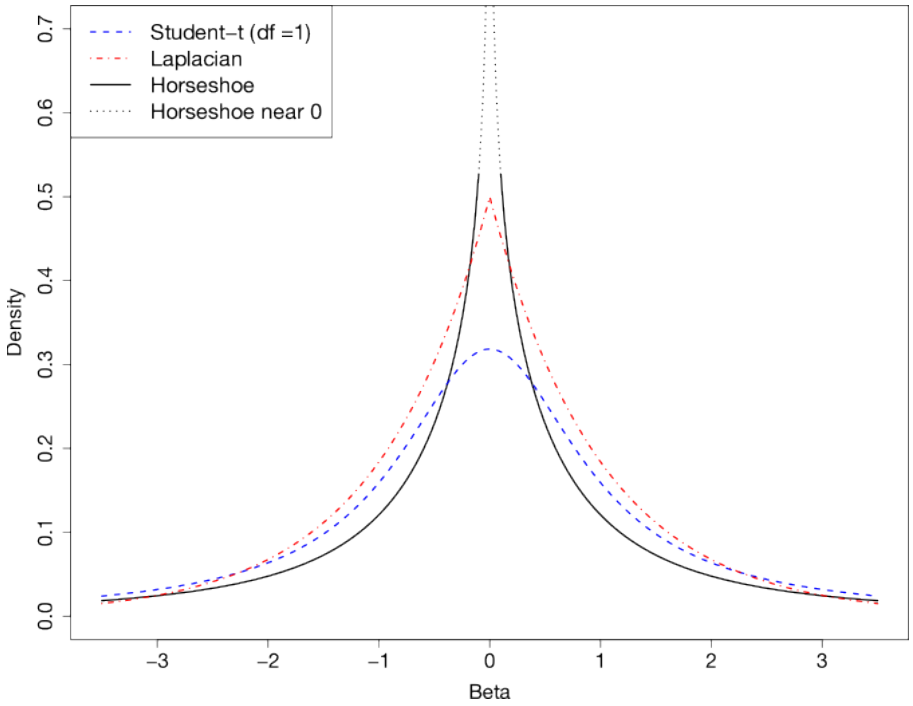


Figure 8.2: Priors for variable selection

A common choice for the global scale is

$$\texttt{global\_scale} = \frac{p_0}{p - p_0}\frac{\sigma}{\sqrt{n}},$$

where p is the number of predictors, p_0 is the expected number of relevant predictors, σ is the model's estimated standard deviation, and n is the number of observations.

2. λ_j is a *local scale* that allows some β_j to escape the shrinkage. The prior for λ_j is

$$\text{Half-Cauchy}\left(0, \texttt{slab\_scale}\right).$$

A common choice for the slab scale is

$$\texttt{slab\_scale} = \sqrt{\frac{\hat{R}^2}{p_0}}\text{sd}(y),$$

where $\hat{R}^2$ is an estimated proportion of variance explained by the model. With this choice of slab scale, the variance of the linear model, with the coefficients sampled from the priors and the predictors with zero mean and standard deviation 1, is precisely $\hat{R}^2\text{Var}(y)$:

$$\begin{aligned}\text{Var}\left(\beta_0 + \beta_1 x_1 + ... + \beta_{p_0} x_{p_0}\right) &= \text{Var}(\beta_1)\text{Var}(x_i) + ... + \text{Var}(\beta_p)\text{Var}(x_p) \\ &= \underbrace{\frac{\hat{R}^2}{p_0}\text{Var}(y) + ... + \frac{\hat{R}^2}{p_0}\text{Var}(y)}_{p_0 \text{ terms}} \\ &= \hat{R}^2\text{Var}(y).\end{aligned}$$

Below is an example of linear regression with the regularized horseshoe prior with $p_0 = 6$ and $\hat{R}^2 = 0.3$ (make sure that the data is standardized!):

```
p <- length(predictors)
n <- nrow(datastd_G3mat)
p0 <- 6
R2_hat <- 0.3
slab_scale <- sqrt(R2_hat/p0)*sd(datastd_G3mat$G3mat)
# global scale without sigma, as the scaling by sigma is done inside stan_glm
global_scale <- (p0/(p - p0))/sqrt(n)

fit3 <- stan_glm(G3mat ~ ., data=datastd_G3mat,
```

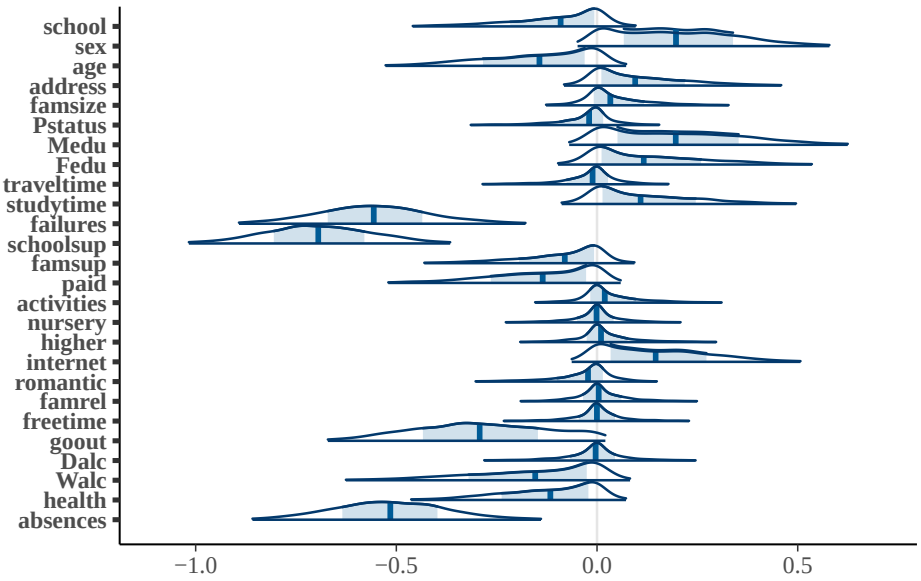
```
                prior=hs(global_scale=global_scale, slab_scale=slab_scale),
                refresh=0)

print(fit3)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      G3mat ~ .
 observations: 343
 predictors:   27
------
            Median MAD_SD
(Intercept) 11.6    0.2
school      -0.1    0.1
sex          0.2    0.2
age         -0.1    0.2
address      0.1    0.1
famsize      0.0    0.1
Pstatus      0.0    0.1
Medu         0.2    0.2
Fedu         0.1    0.2
traveltime   0.0    0.1
studytime    0.1    0.2
failures    -0.6    0.2
schoolsup   -0.7    0.2
famsup      -0.1    0.1
paid        -0.1    0.2
activities   0.0    0.1
nursery      0.0    0.1
higher       0.0    0.1
internet     0.1    0.2
romantic     0.0    0.1
famrel       0.0    0.1
freetime     0.0    0.1
goout       -0.3    0.2
Dalc         0.0    0.1
Walc        -0.2    0.2
health      -0.1    0.2
absences    -0.5    0.2

Auxiliary parameter(s):
      Median MAD_SD
sigma 2.9    0.1

------
```

```
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

We can see that most of the coefficient are zeros. The following distribution plots show that most of the coefficients are shrunk towards zero, making it easier to see relevant predictors.

```
p3 <- mcmc_areas(as.matrix(fit3), pars=vars(-'(Intercept)',-sigma),
                 prob_outer=0.95, area_method = "scaled height")
p3
```



From this plot, we see that the most relevant predictors are `failures`, `schoolsup`, `goout` and `absences`. Let us run the regression with only these four variables.

```
fit4 <- stan_glm(G3mat ~ failures + schoolsup +
                 goout + absences, data=datastd_G3mat,
                 refresh=0)

print(fit4)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      G3mat ~ failures + schoolsup + goout + absences
 observations: 343
 predictors:   5
```

```
------
            Median MAD_SD
(Intercept) 11.6    0.2
failures    -0.8    0.2
schoolsup   -0.7    0.2
goout       -0.5    0.2
absences    -0.6    0.2

Auxiliary parameter(s):
      Median MAD_SD
sigma 3.0    0.1

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

Not surprisingly, `failures`, `goout` and `absences` negatively affects the grades. The same goes for `schoolsup`: a student who requires extra educational support tends to perform worse than one who does not.

We compare its elpd to the model with all predictors.

```
loo2 <- loo(fit2)
loo4 <- loo(fit4)

loo_compare(loo2, loo4)
```

```
     elpd_diff se_diff
fit4  0.0       0.0
fit2 -1.2       6.3
```

The model with only four predictors performs as well as the model with all predictors.

8.2.1.2 LASSO regression

LASSO regression is the Bayesian regression with Laplace prior. To fit a LASSO model, simply run `stan_glm` with `prior=lasso(autoscale=TRUE)`.

```
fit5 <- stan_glm(G3mat ~ ., data=datastd_G3mat,
                 prior=lasso(autoscale=TRUE),
                 refresh=0)

print(fit5)
```

```
stan_glm
```

```
 family:       gaussian [identity]
 formula:      G3mat ~ .
 observations: 343
 predictors:   27
------
            Median MAD_SD
(Intercept) 11.6    0.2
school      -0.1    0.1
sex          0.2    0.2
age         -0.2    0.2
address      0.1    0.1
famsize      0.1    0.1
Pstatus      0.0    0.1
Medu         0.2    0.2
Fedu         0.2    0.2
traveltime   0.0    0.1
studytime    0.2    0.2
failures    -0.5    0.2
schoolsup   -0.7    0.2
famsup      -0.1    0.1
paid        -0.2    0.2
activities   0.0    0.1
nursery      0.0    0.1
higher       0.0    0.1
internet     0.2    0.2
romantic    -0.1    0.1
famrel       0.0    0.1
freetime     0.0    0.1
goout       -0.3    0.2
Dalc         0.0    0.1
Walc        -0.2    0.2
health      -0.2    0.2
absences    -0.5    0.2

Auxiliary parameter(s):
      Median MAD_SD
sigma 2.9    0.1

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

which gives us almost the same model as the one with the horseshoe prior. In general, the horseshoe prior is recommended over the LASSO.

Part II

Generalized linear models

In the previous part, we focused on linear models, which expect continuous outcomes. In many scenarios, however, we would like to model binary outcomes, such as having/not having lung cancer. In this part, we describe a model that can take such outcomes, namely *logistic regression*. After that, we introduct a large class of models called *generalized linear models*, which includes linear and logistic regression as special cases.

Chapter 9

Logistic regression

In this chapter we consider a prediction task with binary outcomes (0 or 1). Our running example is the poll data obtained before the presidential election in 1992. For each respondent i in the poll, we label $y_i = 1$ if he or she preferred George Bush or 0 if he or she preferred Bill Clinton. We predict the preferences from respondents' income levels, measured on a five-point scale.

```
nes <- read.table("data/nes.txt")
nes92 <- nes[nes$year == 1992 &
               !is.na(nes$rvote) &
               !is.na(nes$dvote) &
               (nes$rvote==1 | nes$dvote==1),]

head(nes92[, c("income", "rvote")])
```

```
      income rvote
32093      4     1
32094      2     1
32096      1     0
32097      2     1
32098      3     0
32099      4     0
```

We first attempt to predict the label from the linear function of the predictor(s).

$$X\beta = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p,$$

where $X = (1, x_1, ..., x_p)$ and $\beta = (\beta_0, ..., \beta_p)$. However, the range of the function is $(-\infty, \infty)$. To map this range to $(0, 1)$, we introduce the *logit* function:

$$\text{logit}(x) = \log\left(\frac{x}{1-x}\right),$$

which maps $(0, 1)$ to $(-\infty, \infty)$. What we need is the inverse of the logit function, which is commonly called the *logistic function*:

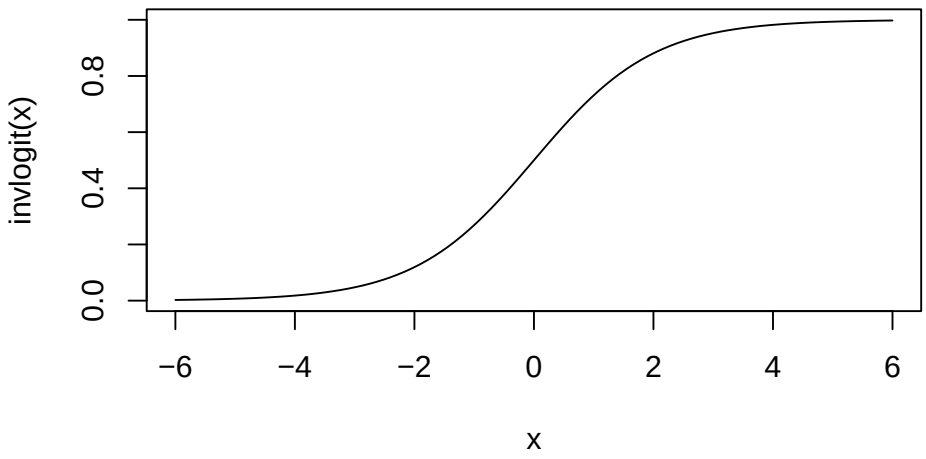
$$\text{logit}^{-1}(x) = \frac{e^x}{1+e^x}.$$

We can access the logit and logistic functions in R with `qlogis` and `plogis`, respectively.

```
logit <- qlogis
invlogit <- plogis
```

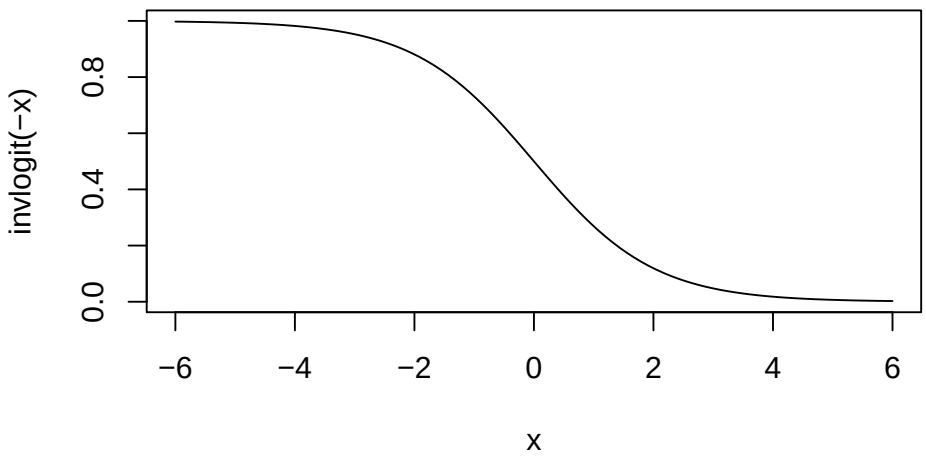
Below is the plot of the $\text{logit}^{-1}(x)$. We can see that the function is bounded above by 1, bounded below by 0, and is 0.5 at 0.

```
curve(invlogit(x), xlim=c(-6, 6))
```



Here is the plot of $\text{logit}^{-1}(-x)$:

```
curve(invlogit(-x), xlim=c(-6, 6))
```



We propose a model for the *probability* of preferring George Bush over Bill Clinton.

$$\Pr(y = 1|\beta, X) = \text{logit}^{-1}(X\beta) = \text{logit}^{-1}(\beta_0 + \beta_1 * \texttt{income}). \tag{9.1}$$

From the plots above, we see that:

- If β_1 is positive then the probability increases with the income.
- If β_1 is negative, then the probability decreases as income increases.

Under the logistic model (@eq-1), we can compute the conditional probability that $y = 0$.

$$\Pr(y = 0|\beta, X) = 1 - \text{logit}^{-1}(X\beta) = 1 - \text{logit}^{-1}(\beta_0 + \beta_1 * \texttt{income}). \tag{9.2}$$

Unlike the linear regression, there is no error term in this model.

9.1 Maximum likelihood for logistic regression

As in the linear regression, we fit the model by finding the parameters β that maximize the likelihood function. Given data $(X_1, y_1), \ldots, (X_n, y_n)$ and the probability model (Equation 19.1) and (Equation 9.2) above, the likelihood function of β is

$$\begin{aligned} p\left(y|\beta, X\right) &= p\left(y_1|\beta, X_1\right) p\left(y_2|\beta, X_2\right) \ldots p\left(y_n|\beta, X_n\right) \\ &= \Pr\left(y = y_1|\beta, X_1\right) \Pr\left(y = y_2|\beta, X_2\right) \ldots \Pr\left(y = y_n|\beta, X_n\right). \end{aligned}$$

Replacing each term in the product using (Equation 19.1) and (Equation 9.2), we can right the product in a compact form as

$$\begin{aligned} p\left(y|\beta,X\right) &= \prod_{i=1}^{n}\begin{cases}\text{logit}^{-1}(X_i\beta) & \text{if } y_i=1\\ 1-\text{logit}^{-1}(X_i\beta) & \text{if } y_i=0\end{cases}\\ &= \prod_{i=1}^{n}\left(\text{logit}^{-1}(X_i\beta)\right)^{y_i}\left(1-\text{logit}^{-1}(X_i\beta)\right)^{1-y_i}. \end{aligned}$$

We then maximize this expression over β. However, unlike the linear regression, using just standard calculus does not give us a closed-form solution. So we have to resort to some optimization algorithm that converges to a stationary point i.e. a point with zero partial derivatives. We will not discuss the algorithm here. Optimization theory tells us the maximization problem has a unique solution, unless there is colinearity or separation; we shall discuss these two conditions in a later chapter.

9.2 Bayesian inference for logistic regression

What we have just discussed in the previous section can be extended to Bayesian inference. As we have shown in Chapter 4, if the prior distributions of the parameters β are uniform, then the vector $\hat{\beta}$ that maximizes the posterior distribution is the same as the maximum likelihood estimate.

The default prior in `stan_glm` is again a weakly informative prior, which imposes the following prior distributions on the parameters:

- Each coefficient β_k for $k = 1,2,...,p$ is given a normal prior $\mathcal{N}\left(0,(2.5/\text{sd}(x_k))^2\right)$.
- The prediction at the mean $\beta_0+\beta_1\bar{x}_1+...+\beta_p\bar{x}_p$ is given a normal prior $\mathcal{N}(0,2.5^2)$.

But sometimes we have some prior information about the coefficients. For example, in a logistic regression with one predictor: $\Pr(y=1)=\text{logit}^{-1}(a+bx)$, sometimes we expect y to increase with x (a classic example is $x=$ smoking and $y=$ cancer). So we shall impose a "soft constraint" on b by giving it a normal prior $\mathcal{N}(0.5,0.5^2)$, which implies that b has a really high chance to be between 0 and 1.

9.3 Fitting a logistic regression model in R

To fit the model using `stan_glm`, we specify the parameter `family=binomial(link="logit")`.

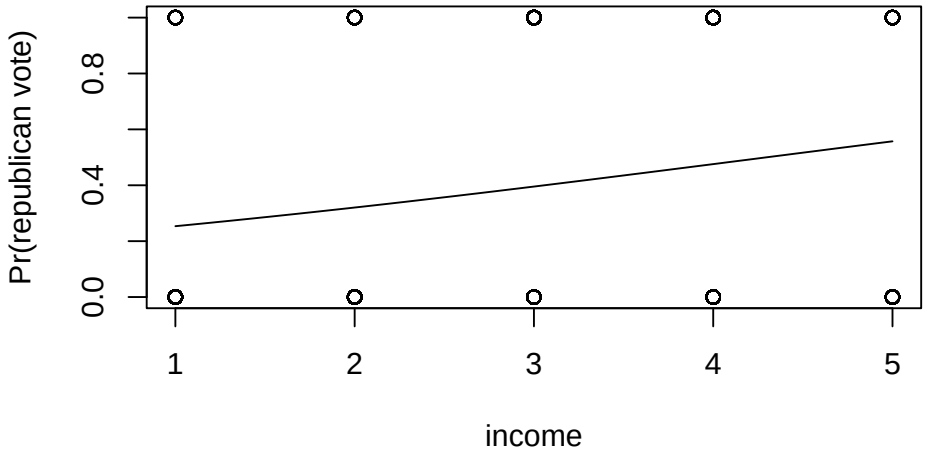
```
library(rstanarm)
```

```
fit_1 <- stan_glm(rvote ~ income, family=binomial(link="logit"),
                  data=nes92, refresh=0)

print(fit_1)
```

```
stan_glm
 family:       binomial [logit]
 formula:      rvote ~ income
 observations: 1179
 predictors:   2
------
            Median MAD_SD
(Intercept) -1.4    0.2
income       0.3    0.1

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

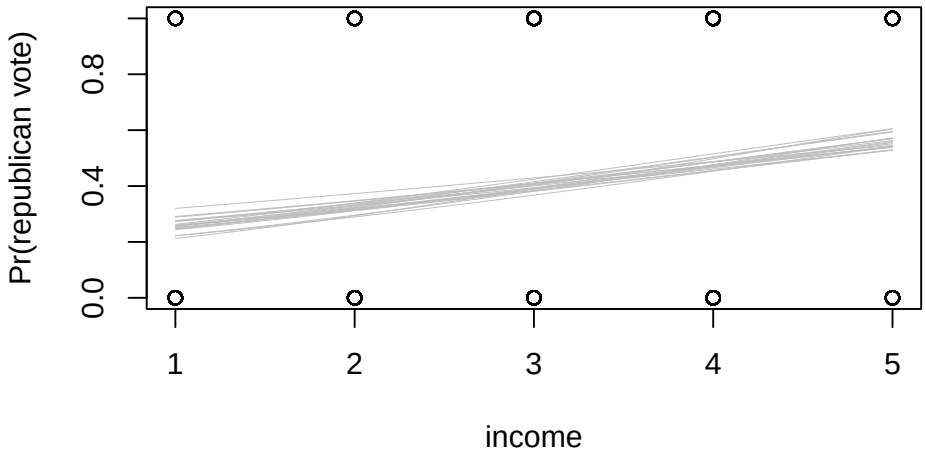
We can plot the actual y_i versus the predictions $\text{logit}^{-1}(-1.4 + 0.3x)$ using the `invlogit` function.

```
a <- coef(fit_1)[1]
b <- coef(fit_1)[2]

plot(nes92$income, nes92$rvote,
     xlab="income", ylab="Pr(republican vote)")
curve(invlogit(a + b * x), add=TRUE)
```



We can also plot with the parameter simulations to visualize the uncertainty in the coefficients. In the following code, we sample 20 draws from 4000 simulations.

```
sims_1 <- as.matrix(fit_1)  # a matrix of parameter simulations
n_sims <- nrow(sims_1)

plot(nes92$income, nes92$rvote,
     xlab="income", ylab="Pr(republican vote)")
for (j in sample(n_sims, 20)){
  a <- sims_1[j, 1]
  b <- sims_1[j, 2]
  curve(invlogit(a + b * x), col="gray",
        lwd=0.5, add=TRUE)
}
```



If, for some reason, we believe that people with higher income levels tend to prefer Bush over Clinton, then we might assume that the coefficient of `income` is somewhere between 0 to 1. Therefore, we use $\mathcal{N}(0.5, 0.5^2)$ as a prior of the coefficient.

```
fit_2 <- stan_glm(rvote ~ income, family=binomial(link="logit"),
                  data=nes92, prior=normal(0.5, 0.5),
                  refresh=0)

print(fit_2)
```

```
stan_glm
 family:       binomial [logit]
 formula:      rvote ~ income
 observations: 1179
 predictors:   2
------
            Median MAD_SD
(Intercept) -1.4    0.2
income       0.3    0.1

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

However, since the dataset is large (1179 instances), the prior has visibly no effect on the coefficient.

9.3.1 Interpreting the coefficients

From the fitted model $\Pr(y = 1) = \text{logit}^{-1}(-1.4 + 0.3x)$, we can interpret the intercept and the slope as follows:

- **Intercept**: As $y = 1$ corresponds to voting for Bush, the intercept -1.4 can be interpreted by assuming $x = 0$. However, assuming so is insensible as the income is on a 1-5 scale. Therefore, we have to indirectly interpret the intercept by evaluating the probability at some other value of x. For example, we can evaluate Pr(Bush support) at the average value of x.

  ```
  mean_income <- mean(nes92$income)
  cat("The mean income is", mean_income, "\n")
  ```

  ```
  The mean income is 3.075488
  ```

  ```
  invlogit(-1.4 + 0.3 * mean_income)
  ```

```
[1] 0.3828772
```

which tells us that at the average value of x, the probability of voting for Bush is 0.38.

- **Slope**: the positive slope implies that a respondent with a higher income level is more likely to vote for Bush. We measure the difference in $\Pr(y = 1)$ for two respondents whose incomes differ by 1. Keep in mind that, as logistic regression involves a nonlinear function, the difference does not stay the same across different values of x. As an example, let us measure the difference near the central value of x. Since $\bar{x} = 3.1$, we evaluate the difference in the probabilities between $x = 2$ and $x = 3$.

$$\text{logit}^{-1}(-1.4 + 0.3 * 3) - \text{logit}^{-1}(-1.4 + 0.3 * 2) = 0.068.$$

Thus, a respondent with income level 3 has 0.068 more probability of supporting Bush than one with income level 2.

9.3.1.1 Divide-by-4 rule for coefficient interpretation

If we look at the curve of the logistic function, we see that the slope of the curve is maximized at the center. Thus the function $\text{logit}^{-1}(a + bx)$ is maximized at $a + bx = 0$. To find out the slope at this point, we take the derivative of this function with respect to x.

$$\frac{d}{dx}\frac{e^{a+bx}}{1+e^{a+bx}} = \frac{d}{dx}\left(1 - \frac{1}{1+e^{a+bx}}\right) = \frac{be^{a+bx}}{(1+e^{a+bx})^2}.$$

Consequently, the slope at $a + bx = 0$ is $b/(1 + 1)^2 = b/4$. Therefore, we can interpret $\beta/4$ as the maximum difference in $\Pr(y = 1)$ corresponding to a 1 unit difference in x.

In our example, the slope is 0.3, which implies that a difference of 1 in income category corresponds no more than $0.3/4 = 0.075$ increase in probability of supporting Bush.

9.4 Different types of predictions

9.4.1 Point prediction

Suppose that we observe a new vector input X^{new}. Assuming the new data follow the logistic model: $\Pr(y^{\text{new}} = 1|\beta, X^{\text{new}}) = \text{logit}^{-1}(X^{\text{new}}\beta)$, a point prediction is the expected value of y^{new} with respect to the posterior distribution of β.

$$\begin{aligned}\mathbb{E}[y^{\text{new}} \mid X^{\text{new}}] &= \mathbb{E}_\beta\left[\mathbb{E}[y^{\text{new}} \mid \beta, X^{\text{new}}]\right] \\ &= \mathbb{E}_\beta\left[\Pr(y^{\text{new}} = 1|\beta, X^{\text{new}})\right] \\ &= \mathbb{E}_\beta\left[\text{logit}^{-1}(X^{\text{new}}\beta)\right].\end{aligned}$$

One might be inclined to estimate the expectation with $\text{logit}^{-1}(X^{\text{new}}\hat{\beta})$, where $\hat{\beta}$ is the vector of point estimates of the parameters. However, this leads to an incorrect estimation; as logit^{-1} is a nonlinear function, we have

$$\mathbb{E}_\beta\left[\text{logit}^{-1}(X^{\text{new}}\beta)\right] \neq \text{logit}^{-1}\left(\mathbb{E}_\beta\left[X^{\text{new}}\beta\right]\right).$$

Instead, we estimate the expectation by averaging $\text{logit}^{-1}(X^{\text{new}}\beta_1), \ldots, \text{logit}^{-1}(X^{\text{new}}\beta_S)$ over the posterior simulations $\beta_1, \ldots, \beta_S$ of β which can be obtained from the output of `stan_glm`. Below is an example of a point prediction for $x^{\text{new}} = 5$ (which corresponds to $X^{\text{new}} = (1, 5)$).

```
sims_1 <- as.matrix(fit_1)  # a matrix of parameter simulations
x_new <- 5
a <- sims_1[, 1]  # 4000 simulations of intercept
b <- sims_1[, 2]  # 4000 simulations of slope

epred <- invlogit(a + b * x_new)  # 4000 simulations of prediction

print(epred[1:5])
```

```
[1] 0.5501388 0.4903334 0.4732712 0.5600746 0.5665402
```

```
pred <- mean(epred)

print(pred)
```

```
[1] 0.5565013
```

All of this can be done in two lines by specifying `type="response"` in the `predict` function.

```
new <- data.frame(income=5)
pred <- predict(fit_1, type="response", newdata=new)

print(pred)
```

```
        1
0.5565013
```

9.4.2 Generating linear predictions

We can obtain simulations draws for the linear part of the model $X^{\text{new}}\beta_1, \ldots, X^{\text{new}}\beta_S$ using `posterior_linpred`.

```
linpred <- posterior_linpred(fit_1, newdata=new)

print(linpred[1:5])
```

```
[1]  0.20123166 -0.03867132 -0.10701705  0.24146501  0.26774879
```

9.4.3 Generating outcome probabilities

Another way of calculating $\Pr(y^{\text{new}} = 1|\beta, X^{\text{new}}) = \text{logit}^{-1}(X^{\text{new}}\beta)$ over the posterior simulations $\beta = \beta_1, \ldots, \beta_S$ is by calling the `posterior_epred` function.

```
epred <- posterior_epred(fit_1, newdata=new)

print(epred[1:5])
```

```
[1] 0.5501388 0.4903334 0.4732712 0.5600746 0.5665402
```

which gives the same outputs as the ones we used to calculate the point prediction above.

We can use these simulations to estimate the uncertainty in the predictions. For example, we can look at the mean and standard deviation of the probabilities that people with income level 5 would support Bush.

```
print(c(mean(epred), sd(epred)))
```

```
[1] 0.55650132 0.02976798
```

The mean of 0.56 and the standard deviation of 0.03 tell us that, among the people with income level 5, the percentage of Bush supporters is probably in the rage $56\% \pm 3\%$.

9.4.4 Generating binary outcomes

The outputs of `posterior_epred` are S different predicted probabilities $p_i = \Pr(y^{\text{new}} = 1|\beta_i, X^{\text{new}})$; $i = 1, \ldots, S$ of an individual voter with income X^{new}. In other words, the distribution of the binary outcome y^{new} for the i-th simulation is Bernoulli(p_i). Thus, to sample new binary outcomes $y_1^{\text{new}}, \ldots, y_S^{\text{new}}$ from the posterior predictive distribution, we can sample y_i^{new} from the Bernoulli(p_i) for $i = 1, \ldots, S$. We can simulate binary outcomes using `posterior_predict`.

```
postpred <- posterior_predict(fit_1, newdata=new)

print(postpred[1:10])
```

```
 [1] 1 0 1 1 1 1 0 1 1 0
```

From theory, the average of these simulations should be close to $\mathbb{E}[y^{\text{new}} \mid X^{\text{new}}]$. Let us check if this is the case.

```
print(mean(postpred))
```

```
[1] 0.55725
```

The average of 0.56 is close to the point prediction above.

9.4.5 Predictions with multiple inputs

We can also use these functions to make predictions for a vector or a matrix of observations. For example, the following code computes different types of predictions for five new people whose income levels take on values 1 through 5:

```
new <- data.frame(income=1:5)
pred <- predict(fit_1, type="response", newdata=new)
linpred <- posterior_linpred(fit_1, type="response", newdata=new)
epred <- posterior_epred(fit_1, type="response", newdata=new)
postpred <- posterior_predict(fit_1, type="response", newdata=new)
```

Here, `pred` is a vector of length 5, and `linpred`, `epred` and `postpred` are matrices of size `n_sims` $\times$ 5. Let us check out the first few rows of `postpred`, for example.

```
print(epred[1:5,])
```

```
iterations         1         2         3         4         5
      [1,] 0.2459632 0.3121931 0.3871012 0.4677595 0.5501388
      [2,] 0.2545576 0.3067185 0.3643436 0.4261403 0.4903334
      [3,] 0.2620032 0.3092899 0.3609370 0.4160139 0.4732712
      [4,] 0.2246515 0.2955238 0.3778583 0.4678963 0.5600746
      [5,] 0.2202625 0.2929362 0.3779651 0.4712261 0.5665402
```

We notice that people with higher income levels are more likely to vote for Bush than those with lower income levels.

We can use these simulations to approximate various posterior quantities. For example, we can create a new variable which indicates that, for each simulation,

Bush is more popular among the people with income level 5 than those with income level 4.

```
indicator <- epred[, 5] > epred[, 4]

print(indicator[1:10])
```

```
 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

which can be used to computed the posterior probability that Bush is more popular among people with income level 5.

```
mean(indicator)
```

```
[1] 1
```

We can also compute the 95% confidence interval for the percentage difference of people with income level 4 and people with income level 5 who support of Bush. To do this, we use the `quantile` function.

```
quantile(epred[, 5] - epred[, 4], c(0.025, 0.975))
```

```
     2.5%     97.5%
0.0534826 0.1072301
```

This tells us that the difference is around $5.3\% - 10.8\%$.

Chapter 10

Logistic regression with multiple predictors

Including more predictors to the logistic regression adds more complexity, not only to the model but also its interpretability. In this chapter, we introduce a new concept for interpreting a coefficient, namely the *average predictive difference*. We also discuss logistic models with interactions and how to interpret their coefficients.

10.1 Example: wells in Bangladesh

Many of the wells in Bangladesh and other South Asian countries are contaminated with arsenic, which is a cumulative poison that may leads to cancer and other diseases. A research team from the United States and Bangladesh has inspected all the wells in Araihazar, Bangladesh and marked them as "safe" if the amount of arsenic is 50 micrograms per liter, and "unsafe" otherwise. People with unsafe wells were encouraged to switch to nearby private or community wells that were safe.

A few years later, the researchers returned surveyed the households in this area. The outcome variable is

$$\texttt{switch} = \begin{cases} 1 & \text{if household } i \text{ switched to a new well} \\ 0 & \text{if household } i \text{ continued using its own well.} \end{cases}$$

We will fit a logistic regression of this binary variable on the following inputs:

Name	Description
dist	The distance to the closest known safe well (meters)
arsenic	The arsenic level of the household's well (ug/l)
assoc	whether any member of the household are active member in community organizations
educ	the education level of the head of household

First, we fit a regression of `switch` on two predictors: the distance and the arsenic level. The distanced are scaled down by a factor of 100 so that the coefficient does not become too small.

```
library(rstanarm)

invlogit <- plogis
```

```
wells <- read.csv("data/wells.csv")
wells$dist100 <- wells$dist/100

head(wells)
```

```
  switch arsenic   dist dist100 assoc educ educ4
1      1    2.36 16.826 0.16826     0    0  0.00
2      1    0.71 47.322 0.47322     0    0  0.00
3      0    2.07 20.967 0.20967     0   10  2.50
4      1    1.15 21.486 0.21486     0   12  3.00
5      1    1.10 40.874 0.40874     1   14  3.50
6      1    3.90 69.518 0.69518     1    9  2.25
```

```
fit_1 <- stan_glm(switch ~ dist100 + arsenic,
                  family = binomial(link="logit"),
                  data = wells, refresh=0)

print(fit_1)
```

```
stan_glm
 family:       binomial [logit]
 formula:      switch ~ dist100 + arsenic
 observations: 3020
 predictors:   3
------
            Median MAD_SD
(Intercept)  0.0    0.1
```

```
dist100     -0.9    0.1
arsenic      0.5    0.0

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

Looking at the coefficients and the standard errors, both predictors are relevant to the prediction.

10.2 Average predictive difference for coefficient interpretation

A natural way to interpret the coefficient of `arsenic` is to compute the predictive difference between two values of arsenic with other predictors held at a constant. For example, we might want to see how the predicted probability change from `arsenic` $= 0.5$ (the lowest unsafe level) to `arsenic` $= 1.0$. Here, we fix `dist100` $= 0.4$.

$$\text{logit}^{-1}(-0.9 * 0.4 + 0.5 * 1.0) - \text{logit}^{-1}(-0.9 * 0.4 + 0.5 * 0.5) = 0.062.$$

We have to be careful on the values of the held-constant predictors; if we choose values that are far away from the actual range of inputs, the corresponding predictive difference might not be achievable. Below are the logistic plots of two examples of data with two variables u and v, where the values of v are at the extremes. We can see that, if we fixed v at its mean value, the predictive difference would be very large compared to those at the actual values of v.

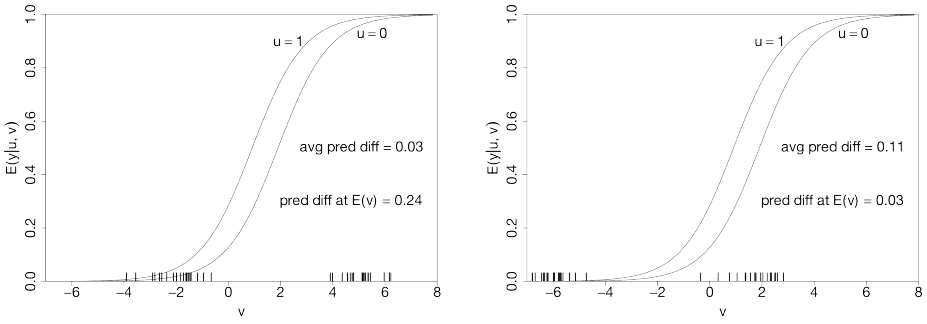


Figure 10.1: Plots of logistic regression between with two predictors: $u = 0$ and $u = 1$, and v is a continuous variable.

A better practice is to compute the predictive differences in one predictor, evaluated at the actual values of the other predictors in the data. Then, we take

the average of the differences. For example, let us denote the well-switching data by $\{(\texttt{switch}_1, \texttt{dist100}_1, \texttt{arsenic}_1), \ldots, (\texttt{switch}_n, \texttt{dist100}_n, \texttt{arsenic}_n)\}$. Then the *average predictive difference* (APD) between $\texttt{arsenic} = 0.5$ and $\texttt{arsenic} = 1.0$ can be calculated as follows:

$$\frac{1}{n}\sum_{i=1}^{n}\left[\text{logit}^{-1}(-0.9 * \texttt{dist100}_i + 0.5 * 1.0) - \text{logit}^{-1}(-0.9 * \texttt{dist100}_i + 0.5 * 0.5)\right].$$

Here is the computation in R:

```
arsenic_hi <- 1.0
arsenic_lo <- 0.5
b <- coef(fit_1)

# differences of logistics for all values of dist100 in the data
diffs <- invlogit(b[1] + b[2] * wells$dist100 + b[3] * arsenic_hi) -
         invlogit(b[1] + b[2] * wells$dist100 + b[3] * arsenic_lo)

mean(diffs)
```

```
[1] 0.05594791
```

In other words, on average, households with the arsenic level of 1.0 are 5.6% more likely to switch the well than those with the arsenic level of 0.5.

10.3 Logistic regression with interactions

Now suppose that we would like to add an interaction term between the distance and the arsenic level to the logistic regression. In R, this can be done by adding `dist100:arsenic` in the formula.

```
fit_2 <- stan_glm(switch ~ dist100 + arsenic + dist100:arsenic,
                  family = binomial(link="logit"), data = wells,
                  refresh=0)

print(fit_2)
```

```
stan_glm
 family:       binomial [logit]
 formula:      switch ~ dist100 + arsenic + dist100:arsenic
 observations: 3020
 predictors:   4
------
```

```
                Median MAD_SD
(Intercept)     -0.2    0.1
dist100         -0.6    0.2
arsenic          0.6    0.1
dist100:arsenic -0.2    0.1

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

According to this model, the probability of switching $\Pr(\texttt{switch} = 1)$ is:

$$\text{logit}^{-1}(-0.1 - 0.6 * \texttt{dist100} + 0.6 * \texttt{arsenic} - 0.2 * \texttt{dist100} * \texttt{arsenic}).$$

The coefficient of the interaction term seems to be relevant, so we keep it in the model for the time being. We now interpret each coefficient.

- **Intercept**: usually, we interpret the intercept by letting the other predictors be zero. However, it is impossible for the distance and the arsenic level to be zero; so we evaluate the prediction at the average values of $\texttt{dist100} = 0.48$ and $\texttt{arsenic} = 0.66$. The corresponding probability of switching is

  $$\text{logit}^{-1}(-0.1 - 0.6 * 0.48 + 0.6 * 1.66 - 0.2 * 0.48 * 1.66) = 0.61.$$

- **Effect of the distance**: we can rewrite the model as

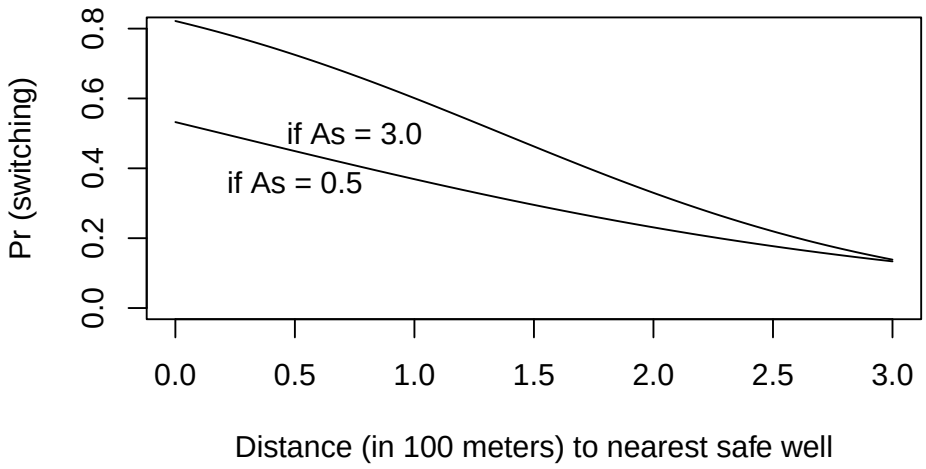
  $$\text{logit}^{-1}(-0.1 + 0.6 * \texttt{arsenic} - (0.6 + 0.2 * \texttt{arsenic}) * \texttt{dist100}).$$

  With `arsenic` fixed at a positive constant, then the coefficient of `dist100` is positive, which implies that households that are farther away from the nearest safe well are less likely to switch. Moreover, as the households' wells have higher arsenic levels, the *importance* of the distance to the probability of switching is increasing.

  We plot the logistic regression of probability of switching as a function of distance to the nearest safe well at two arsenic levels: $\texttt{arsenic} = 0.5$ and $\texttt{arsenic} = 3.0$.

```
b <- coef(fit_2)

curve(invlogit(b[1] + b[2]*x + b[3]*0.5 + b[4]*x*0.5),
      xlab="Distance (in 100 meters) to nearest safe well",
```

```
      ylab="Pr (switching)",
      xlim=c(0,3),
      ylim=c(0,0.8))
curve(invlogit(b[1] + b[2]*x + b[3]*3.0 + b[4]*x*3.0), add=TRUE)
text (0.50, 0.36, "if As = 0.5")
text (0.75, 0.50, "if As = 3.0")
```



For `arsenic` = 1.0, the probability of switching starts off higher and moves down faster; this is because the arsenic level became less relevant as the safe wells was too far away from the households. At 300 meters, the arsenic level barely affected the households' decisions at all.

We use the APD to measure the effect of the distance on the switching probability. Let us compute the APD of the distance from `dist100` = 0 to `dist100` = 1.

```
dist100_hi <- 1
dist100_lo <- 0
b <- coef(fit_2)

# differences of logistics for all values of dist100 in the data
diffs <- invlogit(b[1] + b[2] * dist100_hi + b[3] * wells$arsenic +
                  b[4] * dist100_hi * wells$arsenic) -
         invlogit(b[1] + b[2] * dist100_lo + b[3] * wells$arsenic +
                  b[4] * dist100_lo * wells$arsenic)

mean(diffs)
```

```
[1] -0.195182
```

which implies that, on average, households that are 100 meters from the nearest safe well are 20% less likely to switch compared to households that lived right next to a safe well.

- **Effect of the arsenic level**: we can rewrite the model as

$$\text{logit}^{-1}(-0.1 - 0.6 * \texttt{dist100} + (0.6 - 0.2 * \texttt{dist100}) * \texttt{arsenic}).$$

Notice that the coefficient of `arsenic` decreases as `dist100` increases. This means that, as the distance increases, the *importance* of arsenic to the probability of switching decreases.

Let us compute the APD between `arsenic` $= 0.5$ to `arsenic` $= 1.0$.

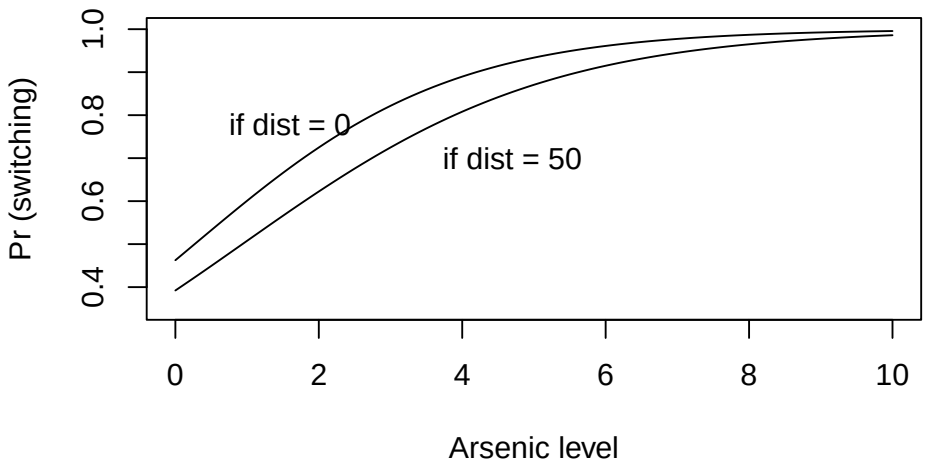
```
arsenic_hi <- 1.0
arsenic_lo <- 0.5
b <- coef(fit_2)

# differences of logistics for all values of dist100 in the data
diffs <- invlogit(b[1] + b[2] * wells$dist100 + b[3] * arsenic_hi +
                  b[4] * wells$dist100 * arsenic_hi) -
         invlogit(b[1] + b[2] * wells$dist100 + b[3] * arsenic_lo +
                  b[4] * wells$dist100 * arsenic_lo)

mean(diffs)
```

```
[1] 0.05814341
```

which implies that, on average, households with the arsenic level of 1.0 are 5.8% more likely to switch the well than those with the arsenic level of 0.5.

Here is the plot of the logistic regression lines of the probability of switching as a function of arsenic level, at distances of 0 meters and 50 meters.

```
b <- coef(fit_2)

curve(invlogit(b[1] + b[2]*0.0 + b[3]*x + b[4]*0.0*x),
      xlab="Arsenic level",
      ylab="Pr (switching)",
      xlim=c(0,10),
      ylim=c(0.35,1.0))
curve(invlogit(b[1] + b[2]*0.5 + b[3]*x + b[4]*0.5*x), add=TRUE)
text (1.6, 0.78, "if dist = 0")
text (4.7, 0.7, "if dist = 50")
```



Of course, households that lived next to a safe well were more likely two switch the well. The probabilities reach the same value as the arsenic level became dangerously high; this is because households almost always wanted to switch, regardless of the distance to the safe well.

Chapter 11

Diagnostics of logistic regression models

After fitting a logistic regression model, it is a good idea to inspect the model in more details. We discuss techniques of plotting the fitted model and checking the residuals. We also look into adding interaction terms to the model, as well as identifying problems that may arise when fitting the model.

11.1 Plotting logistic regression and binary data

11.1.1 Plotting binary data using binned averages

The usual scatterplot of data with binary outcomes might not be insightful as there is a lot of overlaps. A better way of plotting the data for is by binning a predictor, with other predictors held at constants, and plot the *binned averages*, the averages of the predictor and the outcome in each bin. We demonstrate this with the `wells` data from the previous chapter.

```
library(rstanarm)

invlogit <- plogis
```

```
wells <- read.csv("data/wells.csv")
wells$dist100 <- wells$dist/100
```

We fit a logistic regression model of well-switching on the distance.

```
fit_1 <- stan_glm(switch ~ dist100,
                  family = binomial(link="logit"),
                  data = wells, refresh=0)
```

Let us divide the distance into 8 bins.

```
K <- 8
# assign bin for each data point
bins <- as.numeric(cut(wells$dist100, K))

print(data.frame(wells$dist100, bins)[1:6,])
```

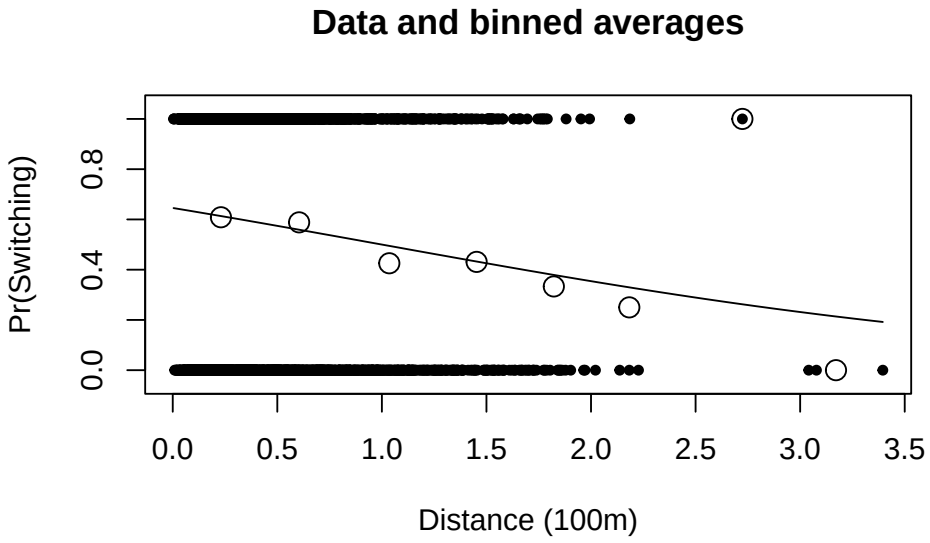
```
  wells.dist100 bins
1       0.16826    1
2       0.47322    2
3       0.20967    1
4       0.21486    1
5       0.40874    1
6       0.69518    2
```

Then we compute the average of the distance and the outcome for each bin.

```
x_bar <- rep(NA, K)  # initial vector of NAs
y_bar <- rep(NA, K)  # initial vector of NAs
for (k in 1:K){
  x_bar[k] <- mean(wells$dist100[bins==k])  # average of k-th bin
  y_bar[k] <- mean(wells$switch[bins==k])  # average of k-th bin
}
```

And now we plot the binned averages (the plot of white circles below). Since the averages of outcomes are no longer repeated values of 0 and 1, we can see some vertical variation in the plot. We also see that the binned averages are close to the fitted logistic regression line.

```
plot(wells$dist100, wells$switch, ylim=c(-0.05, 1.05),
     xlab="Distance (100m)", ylab="Pr(Switching)",
     pch=20, cex=1.0, main="Data and binned averages")
points(x_bar, y_bar, pch=21, cex=1.5)

# fitted logistic regression
curve(invlogit(0.6 - 0.6*x), add=TRUE)
```



11.1.2 Plotting decision boundaries when there are two predictors

Suppose that we have a logistic model with two predictors. Even though the model is nonlinear, our decision on the outcome of a new data point can be linear. A typical decision $\hat{y}$ of a new data point (x_1, x_2) is

$$\hat{y} = \begin{cases} 1 & \text{if } \Pr(y = 1|x_1, x_2) > 0.5 \\ 0 & \text{if } \Pr(y = 1|x_1, x_2) \leq 0.5. \end{cases}$$

Since $\Pr(y = 1|x_1, x_2) = \text{logit}^{-1}(\hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2)$ and $\text{logit}^{-1}(x) = 0.5$ when $x = 0$, so we can rewrite the decision as follows:

$$\hat{y} = \begin{cases} 1 & \text{if } \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 > 0 \\ 0 & \text{if } \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 \leq 0. \end{cases}$$

In this case, the linear function $\hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 = 0$ splits our decision, and so it is the decision boundary.

Let us fit a logistic regression of the well-switching on the distance and the arsenic level.

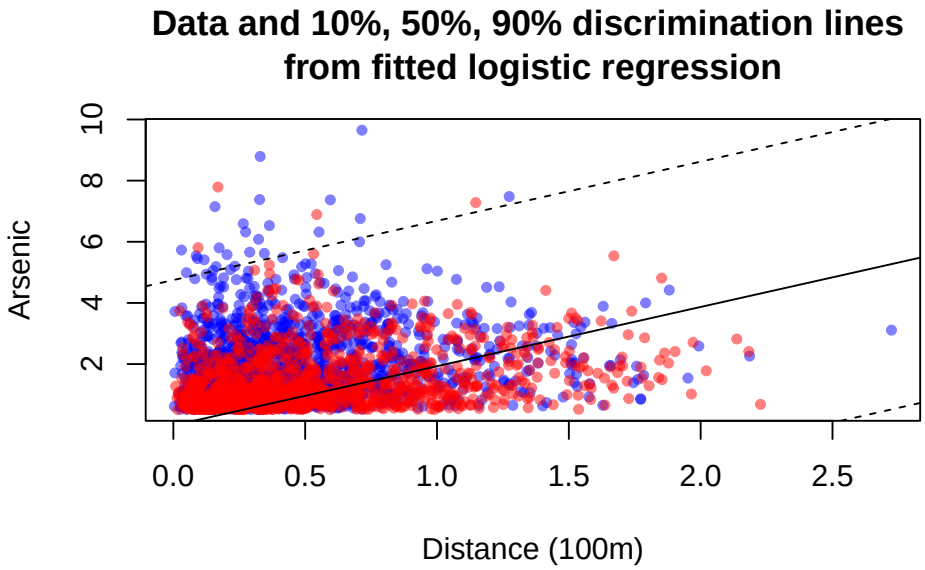
```
fit_2 <- stan_glm(switch ~ dist100 + arsenic,
                  family = binomial(link="logit"),
                  data = wells, refresh=0)
```

```
b = coef(fit_2)
```

Then, we plot the decision boundary of this model. In addition, we plot the lines of two extremes: $\Pr(y|x_1, x_2) = 0.1$ and $\Pr(y|x_1, x_2) = 0.9$, which is equivalent to $\hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 = \text{logit}(0.1)$ and $\hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 = \text{logit}(0.9)$, respectively. Note that to plot these equations with `abline`, we need to write x_2 as a function of x_1; the corresponding equations are:

$$
\begin{aligned}
x_2 &= -\frac{\hat{\beta}_0}{\hat{\beta}_2} - \frac{\hat{\beta}_1}{\hat{\beta}_2} x_1 \\
x_2 &= -\frac{\text{logit}(0.1) - \hat{\beta}_0}{\hat{\beta}_2} - \frac{\hat{\beta}_1}{\hat{\beta}_2} x_1 \\
x_2 &= -\frac{\text{logit}(0.9) - \hat{\beta}_0}{\hat{\beta}_2} - \frac{\hat{\beta}_1}{\hat{\beta}_2} x_1.
\end{aligned}
$$

```
plot(wells$dist100[wells$switch==1],
     wells$arsenic[wells$switch==1],
     main="Data and 10%, 50%, 90% discrimination lines
from fitted logistic regression",
     xlab="Distance (100m)",
     ylab="Arsenic",
     col = rgb(red=0, green=0, blue=1, alpha=0.5),
     pch=20)
points(wells$dist100[wells$switch==0],
       wells$arsenic[wells$switch==0],
       col = rgb(red=1, green=0, blue=0, alpha=0.5),
       pch=20)

abline(-b[1] / b[3],
       -b[2] / b[3])
abline((logit(0.9) - b[1]) / b[3],
       -b[2] / b[3], lty=2)
abline((logit(0.1) - b[1]) / b[3],
       -b[2] / b[3],  lty=2)
```

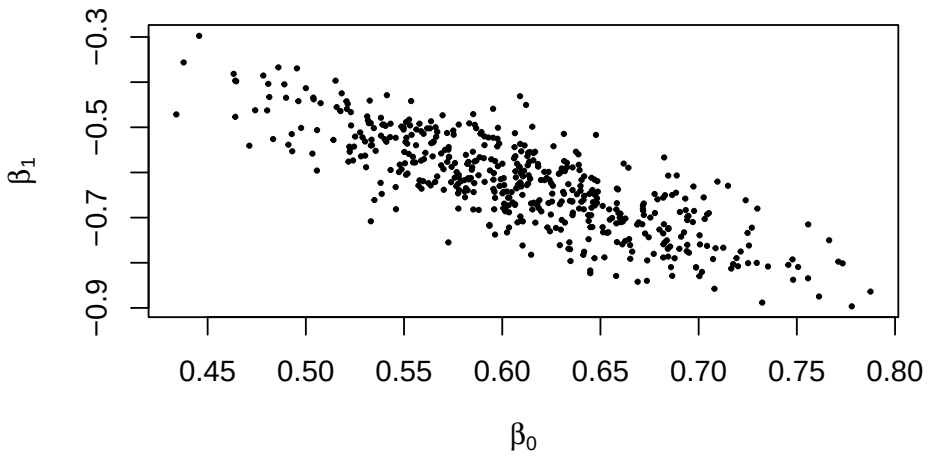


We notice from the plot that the model does not suit the data well, as many points in each class leak to the other side of the decision boundary.

11.2 Predictive simulation

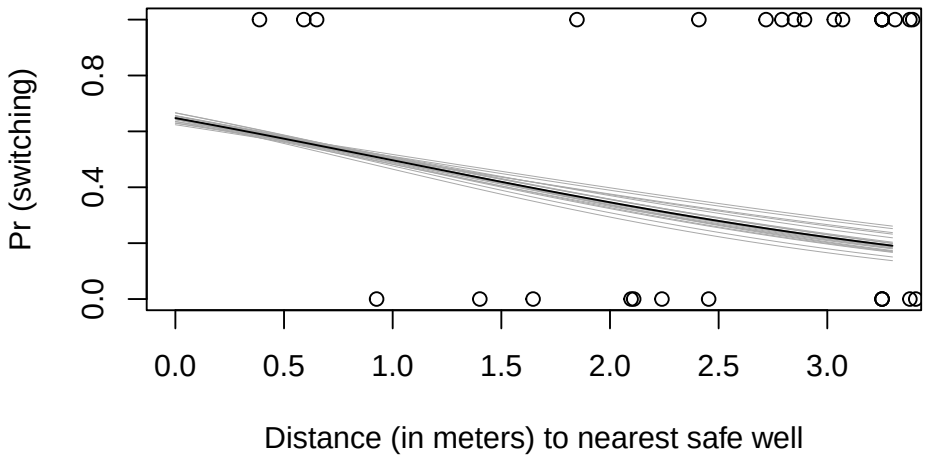
We can plot the uncertainty of the coefficients using `stan_glm`'s simulations. First, let us take the model with a single predictor `dist100` and plot the first 500 simulations of the intercept and the slope.

```
sims <- as.matrix(fit_1)
plot(sims[1:500,1], sims[1:500,2], xlab=expression(beta[0]),
     ylab=expression(beta[1]), pch=20, cex=.5)
```



We can also visualize the uncertainty in the predicted probabilities by plotting the logistic models with the simulated coefficients from above. The plot of the first 20 models below tells us that the probabilities of switching under shorter distances to the nearest safe well are more varied than those under longer distances.

```
plot(wells$dist, wells$switch,
     ylim=c(0, 1), xlim=c(0, 3.3),
     xlab="Distance (in meters) to nearest safe well",
     ylab="Pr (switching)")
for (j in 1:20) {
  a <- sims[j, 1]
  b <- sims[j, 2]
  curve (invlogit(a + b * x), lwd=.5,
         col="darkgray", add=TRUE)
}
a_hat <- coef(fit_1)[1]
b_hat <- coef(fit_1)[2]
curve(invlogit(a_hat + b_hat*x), lwd=1, add=TRUE)
```



11.3 Log score for logistic regression

In this section, we only consider the logistic regression model with the point estimate $\hat{\beta}$ (i.e. the numbers shown in the output of `stan_glm`). We recall the likelihood function of the logistic model.

$$p\left(y|\hat{\beta}, X\right) = \prod_{i=1}^{n} \begin{cases} \text{logit}^{-1}(X_i\hat{\beta}) & \text{if } y_i = 1 \\ 1 - \text{logit}^{-1}(X_i\hat{\beta}) & \text{if } y_i = 0. \end{cases}$$

Given m new labeled data points $(X_1^{\text{new}}, y_1) \dots, (X_m^{\text{new}}, y_m)$, we denote the probability prediction $p_i = \text{logit}^{-1}(X_i^{\text{new}}\hat{\beta})$ for $i = 1, \dots, m$. The log score is computed by taking the logarithm of the likelihood.

$$\text{out-of-sample log score} = \sum_{i=1}^{m} \begin{cases} \log p_i & \text{if } y_i = 1 \\ \log(1 - p_i) & \text{if } y_i = 0. \end{cases}$$

Let us see how the log score behaves in three cases.

1. Almost perfectly correct predictions; that is, $p_i \approx 1$ whenever $y_i = 1$ and $p_i \approx 0$ whenever $y_i = 0$. In the former case, we have $\log p_i \approx \log 1 = 0$, and in the latter, $\log(1 - p_i) \approx \log 1 = 0$ as well. Thus we expect the log score to be close to zero.
2. Completely wrong predictions; that is, $p_i \approx 0$ whenever $y_i = 1$ and $p_i \approx 1$ whenever $y_i = 0$. In the former case, we have $\log p_i \approx \log 0 = -\infty$ and in the latter, $\log(1 - p_i) \approx \log 0 = -\infty$. In other words, a large negative log score indicates that the model has bad predictive performance.
3. Random guessing; that is, $p_i = 0.5$ for all i, which implies $\log p_i = \log(1 - p_i) = \log 0.5$ for all i. The corresponding log score is $\sum_{i=1}^{m} \log 0.5 =$

$m \log 0.5$. Any model with meaningful predictions should do better than random guessing, so its log score should be more than $m \log 0.5$.

We measure the out-of-sample predictive performance with expected log predictive density (elpd) based on the above log score. We can compute the elpd of the model by simply calling the `loo` function.

11.3.1 Example of variable selection: well-switching example

```
loo_2 <- loo(fit_2)

print(loo_2)
```

```
Computed from 4000 by 3020 log-likelihood matrix

         Estimate   SE
elpd_loo  -1968.5 15.7
p_loo         3.3  0.1
looic      3937.1 31.4
------
Monte Carlo SE of elpd_loo is 0.0.

All Pareto k estimates are good (k < 0.5).
See help('pareto-k-diagnostic') for details.
```

The elpd of this model is -1968.4. In comparison, the elpd of random guessing is $3020 \log 0.5 = -2093$ (as their are 3020 data points in the dataset). So in terms of elpd, our model is better than random guessing.

Now, let us try adding an interaction term between the distance and the arsenic level.

```
fit_3 <- stan_glm(switch ~ dist100 + arsenic + dist100:arsenic,
                  family = binomial(link="logit"), data = wells,
                  refresh=0)

print(fit_3)
```

```
stan_glm
 family:       binomial [logit]
 formula:      switch ~ dist100 + arsenic + dist100:arsenic
 observations: 3020
 predictors:   4
------
                Median MAD_SD
```

```
(Intercept)     -0.1    0.1
dist100         -0.6    0.2
arsenic          0.6    0.1
dist100:arsenic -0.2    0.1

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

```
loo_3 <- loo(fit_3)

loo_compare(loo_2, loo_3)
```

```
      elpd_diff se_diff
fit_3  0.0       0.0
fit_2 -0.4       1.9
```

There is virtually no improvement over the previous model, and the standard error of the interaction term is not significantly smaller than the point estimate, so we decide to discard it from the model.

Now, let us add two more predictors that might be relevant: the years of education of the well user (`educ4`) and the status of association with any community organization (`assoc`).

```
fit_4 <- stan_glm(switch ~ dist100 + arsenic + educ4 + assoc,
                  family = binomial(link="logit"), data = wells,
                  refresh=0)

print(fit_4)
```

```
stan_glm
 family:       binomial [logit]
 formula:      switch ~ dist100 + arsenic + educ4 + assoc
 observations: 3020
 predictors:   5
------
            Median MAD_SD
(Intercept) -0.2    0.1
dist100     -0.9    0.1
arsenic      0.5    0.0
educ4        0.2    0.0
assoc       -0.1    0.1

------
* For help interpreting the printed output see ?print.stanreg
```

```
* For info on the priors used see ?prior_summary.stanreg
```

The coefficient of `assoc` is highly uncertain (high standard error compared to the point estimate) so we decide to remove the predictor.

```
  fit_5 <- stan_glm(switch ~ dist100 + arsenic + educ4,
                    family = binomial(link="logit"),
                    data = wells, refresh=0)

  print(fit_5)
```

```
stan_glm
 family:       binomial [logit]
 formula:      switch ~ dist100 + arsenic + educ4
 observations: 3020
 predictors:   4
------
            Median MAD_SD
(Intercept) -0.2    0.1
dist100     -0.9    0.1
arsenic      0.5    0.0
educ4        0.2    0.0

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

Every predictors seems to be significant, so now we add the interaction terms between the education and the other predictors.

```
  fit_6 <- stan_glm(switch ~ dist100 + arsenic + educ4 +
                    dist100:educ4 + arsenic:educ4,
                    family = binomial(link="logit"),
                    data = wells, refresh=0)

  print(fit_6)
```

```
stan_glm
 family:       binomial [logit]
 formula:      switch ~ dist100 + arsenic + educ4 + dist100:educ4 + arsenic:educ4
 observations: 3020
 predictors:   6
------
             Median MAD_SD
(Intercept)   0.1    0.1
dist100      -1.3    0.2
```

```
arsenic        0.4    0.1
educ4         -0.1    0.1
dist100:educ4  0.3    0.1
arsenic:educ4  0.1    0.0

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

Now let us compare the elpd of this model to the model with two predictors.

```
loo_6 <- loo(fit_6)

loo_compare(loo_2, loo_6)
```

```
      elpd_diff se_diff
fit_6   0.0       0.0
fit_2 -15.7       6.3
```

The elpd of the latest model is much higher, so we decide to keep this model for later sections.

11.4 Residuals for logistic regression

Let y be the actual outcome and $\Pr(y = 1)$ the predicted probability of a data point. The residual of the prediction is given by

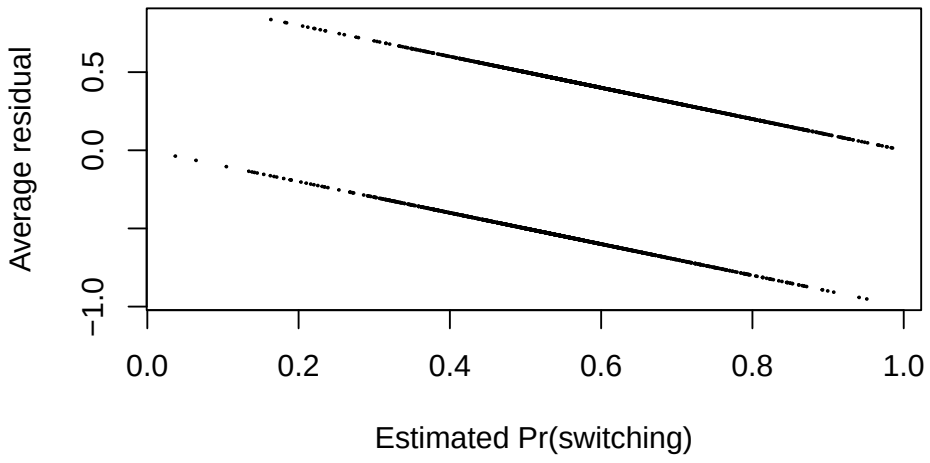
$$\text{residual} = y - \Pr(y = 1).$$

Unlike the linear regression, it does not make sense to plot the residual vs. the predicted probability as the points would belong to only one of the following functions of the predicted probability: $1 - \Pr(y = 1)$ and $-\Pr(y = 1)$. We illustrate this point by making a residual plot of the previous model of well-switching.

```
# We can use predict(fit_6, type="response", newdata=wells)
# but it is much slower than fitted.
pred6 <- fitted(fit_6)
pred6[1:5]
```

```
        1         2         3         4         5
0.6931220 0.4394061 0.7250128 0.6112158 0.6033322
```

```
residuals = wells$switch-pred6

plot(pred6, residuals, pch=20, cex=.2,
     xlab="Estimated Pr(switching)",
     ylab="Average residual")
```

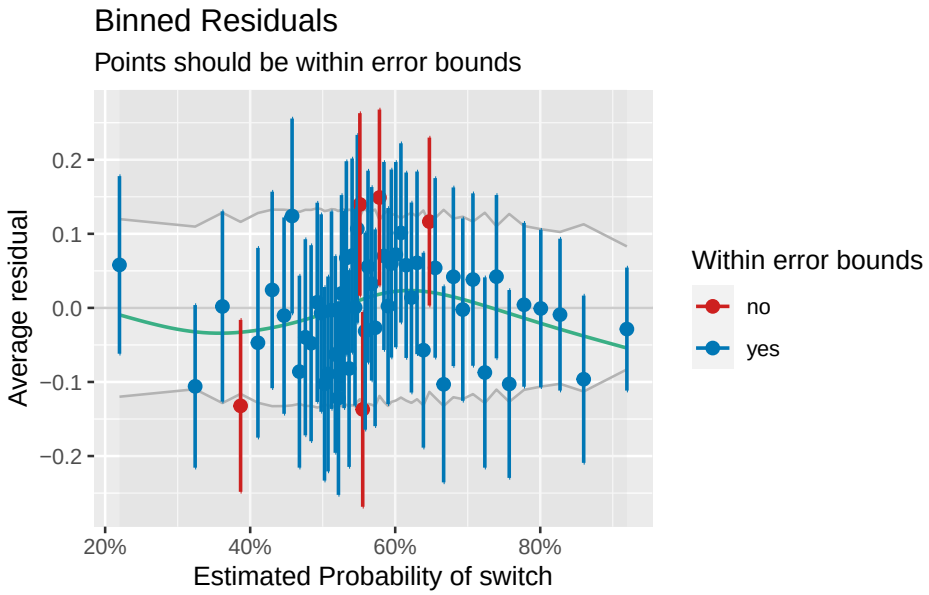


A recommended way to visualize the residuals is by plotting the *binned residuals*, which can be obtained by binning the data on the predicted probabilities, and then computing the averages of the probabilites and the residuals for each bin.

We compute and plot the binned residuals using the `binned_residuals` function from the `performance` library. Let p_j be the average predicted probability in bin j and n_j be the number of points in bin j. If the model were true, then, the j-th binned residual should fall inside the interval $[-2\sqrt{p_j(1-p_j)/n_j}, 2\sqrt{p_j(1-p_j)/n_j}]$ (the dotted lines) with probability 0.95.

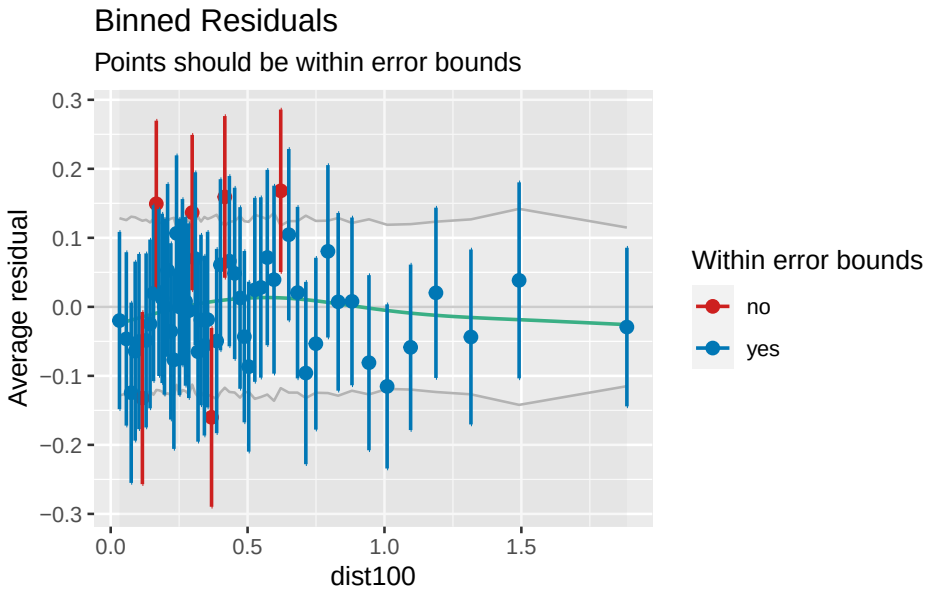
```
install.packages("performance")
install.packages("see")  # required to plot residuals

library(performance)
```

```
binnedres = binned_residuals(fit_6)

plot(binnedres)
```

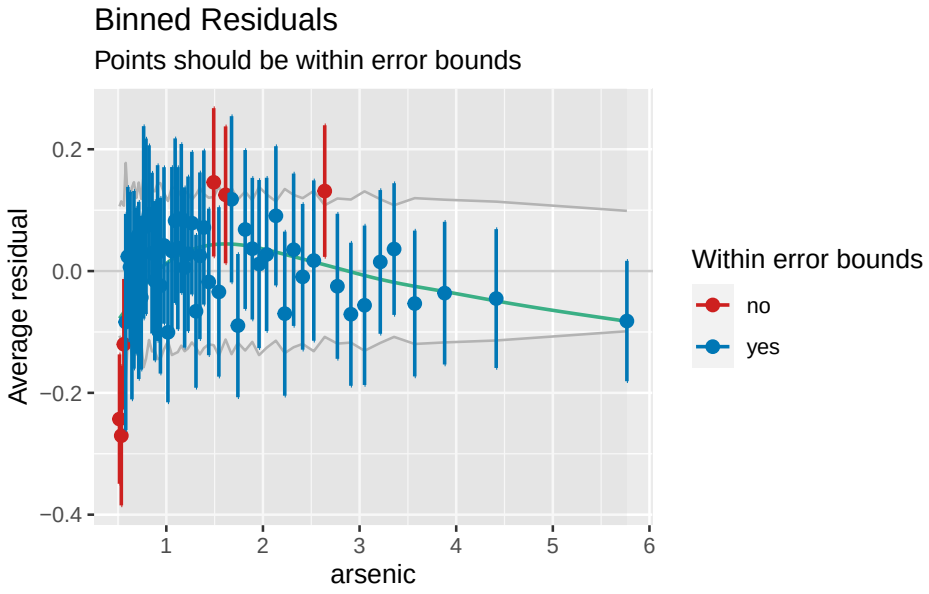


We can also plot binned residuals versus an input of interested by specifying the name of the input as an argument to the `binned_residuals` function. Here, we plot the binned residuals versus the distance and the arsenic level and see if there are any unusual patterns in the plots.

```
binnedres_dist = binned_residuals(fit_6, "dist100")
binnedres_arsenic = binned_residuals(fit_6, "arsenic")

plot(binnedres_dist)
```




```
plot(binnedres_arsenic)
```



The binned residual plot of the distance is mostly flat around zero, indicating that the model is a good fit. The plot of the arsenic level, however, has a rising and falling pattern, in which case we might want to apply the logarithmic

transformation.

11.5 Logarithmic transformation

As in the linear regression we can apply the logarithm on the variables to further improve the fit of the model.

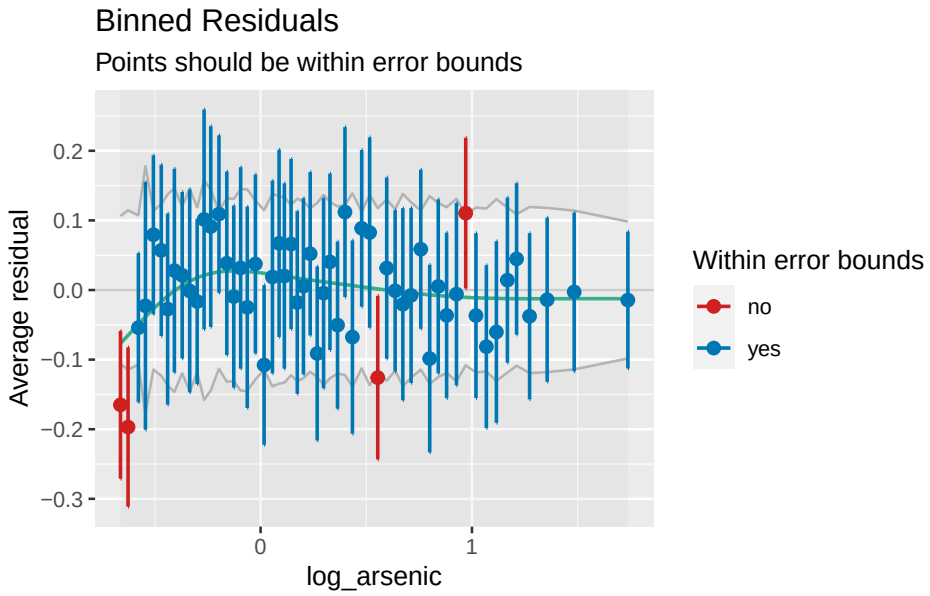
We continue the well-switching example. Detecting the unusual pattern in the binned residuals of the arsenic level, we decide to apply the logarithm on this predictor.

```
wells$log_arsenic <- log(wells$arsenic)

fit_7 <- stan_glm(switch ~ dist100 + log_arsenic + educ4 +
                  dist100:educ4 + log_arsenic:educ4,
                  family = binomial(link="logit"),
                  data = wells, refresh=0)

print(fit_7)
```

```
stan_glm
 family:       binomial [logit]
 formula:      switch ~ dist100 + log_arsenic + educ4 + dist100:educ4 + log_arsenic:educ4
 observations: 3020
 predictors:   6
------
                 Median MAD_SD
(Intercept)       0.5    0.1
dist100          -1.4    0.2
log_arsenic       0.8    0.1
educ4             0.0    0.1
dist100:educ4     0.3    0.1
log_arsenic:educ4 0.1    0.1

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

Then we plot the binned residuals of the arsenic level again.

```
binnedres_logarsenic = binned_residuals(fit_7, "log_arsenic")

plot(binnedres_logarsenic)
```



The residuals look better, though the problem remains at the low arsenic levels: the households with low arsenic levels are less likely to switch than predicted by the model.

Let us compare the elpd of the log model to the previous model.

```
loo_7 <- loo(fit_7)

loo_compare(loo_6, loo_7)
```

```
      elpd_diff se_diff
fit_7   0.0       0.0
fit_6 -14.8       4.3
```

The 14.8 increase in elpd suggests that the log model is a better fit than the previous model.

11.6 Error rate

Another way to measure the performance of a logistic regression model is by comparing the actual outcome y to the model's predicted outcome:

$$\hat{y} = \begin{cases} 1 & \text{if } \Pr(y = 1|x_1, x_2) > 0.5 \\ 0 & \text{if } \Pr(y = 1|x_1, x_2) \leq 0.5. \end{cases}$$

The *error rate* is the proportion of mismatches between y and $\hat{y}$ in the dataset. Thus lower error rate implies that the model's predictions are more accurate.

To compute the error rate in R, we first output the model's predicted probabilities.

```
pred8 <- fitted(fit_7)
```

Then we can compute the error rate as follows:

```
error_rate <- mean((pred8>0.5 & wells$switch==0) |
                   (pred8<=0.5 & wells$switch==1))

error_rate
```

```
[1] 0.363245
```

Let us compare this error rate with the one from a much simpler model, which predicts the majority outcome for all data points.

```
print(c("Proportion of 1's", mean(wells$switch)))
```

```
[1] "Proportion of 1's" "0.575165562913907"
```

```
print(c("Proportion of 0's", 1-mean(wells$switch)))
```

```
[1] "Proportion of 0's" "0.424834437086093"
```

The error rate of this simple model is 0.43. So in terms of error rates, our logistic model's predictions are more accurate than outputting the majority outcome.

11.7 Nonidentification

11.7.1 Collinearity

Collinearity happens when one of the predictors is a linear combination of the other predictors, which results in unstable fitting procedure. This results in coefficients having large standard errors. We can solve this issue by removing some of the predictors while keeping the elpd at the same level.

11.7.2 Separation

Separation happens when a combination of the predictors can be used to completely split the outcomes by their values. In the example below, we have data

of US election in 1964, where the `black` variable completely aligns the outcome (`rvote`), as there is no black respondent that votes for the republican.

```
nes <- read.table("data/nes.txt")
nes64 <- nes[nes$year == 1964 &
               !is.na(nes$rvote) &
               !is.na(nes$female) &
               !is.na(nes$black) &
               !is.na(nes$income),]

head(nes64[, c("year", "rvote", "black")])
```

```
     year rvote black
8467 1964     0     0
8468 1964     0     0
8470 1964     0     0
8471 1964     0     0
8473 1964     0     0
8474 1964     0     0
```

```
nes64[nes64$rvote==1 & nes64$black==1,]
```

```
 [1] year            resid           weight1         weight2
 [5] weight3         age             gender          race
 [9] educ1           urban           region          income
[13] occup1          union           religion        educ2
[17] educ3           martial_status  occup2          icpsr_cty
[21] fips_cty        partyid7        partyid3        partyid3_b
[25] str_partyid     father_party    mother_party    dlikes
[29] rlikes          dem_therm       rep_therm       regis
[33] vote            regisvote       presvote        presvote_2party
[37] presvote_intent ideo_feel       ideo7           ideo
[41] cd              state           inter_pre       inter_post
[45] black           female          age_sq          rep_presvote
[49] rep_pres_intent south           real_ideo       presapprov
[53] perfin1         perfin2         perfin          presadm
[57] age_10          age_sq_10       newfathe        newmoth
[61] parent_party    white           year_new        income_new
[65] age_new         vote.1          age_discrete    race_adj
[69] dvote           rvote
<0 rows> (or 0-length row.names)
```

In this case, the best maximum likelihood estimate of the coefficient of `black` is $-\infty$. In the summary of the regression on three predictors below, we notice that the standard error `black` is abnormally high.

```r
fit_8 <- glm(rvote ~ female + black + income,
             family=binomial(link="logit"),
             data=nes64)

summary(fit_8)$coefficients
```

```
              Estimate   Std. Error    z value     Pr(>|z|)
(Intercept)  -1.1509055   0.21594094 -5.3297235 9.836238e-08
female       -0.0873733   0.13623456 -0.6413446 5.212988e-01
black       -16.8337552 420.40038735 -0.0400422 9.680595e-01
income        0.1922987   0.05846259  3.2892611 1.004508e-03
```

We can handle this issue by adding some prior to the model. In fact, we can just use the default prior in `stan_glm` (the weakly informative prior) and the coefficient and the standard error becomes much smaller.

```r
fit_9 <- stan_glm(rvote ~ female + black + income,
                  family=binomial(link="logit"),
                  data=nes64, refresh=0)

print(fit_9)
```

```
stan_glm
 family:       binomial [logit]
 formula:      rvote ~ female + black + income
 observations: 1058
 predictors:   4
------
            Median MAD_SD
(Intercept) -1.1    0.2
female      -0.1    0.1
black       -8.7    4.1
income       0.2    0.1

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

Chapter 12

Generalized linear models

The linear regression and the logistic regression are examples of a more general class of models: the *generalized linear models* (GLM). As in the logistic regression, we can modify the nonlinear function and the model of the outcomes to handle various types of data, such as data with bounded outcomes, count data and data with multi-valued outcomes.

12.1 Definition of generalized linear models

Generalized linear models (GLMs) are a class of models that conform to the *transformed linear predictor* design. Specifically, a GLM consists of:

1. A vector of outcome data $y = (y_1, \dots, y_n)$.
2. A matrix of predictor $X = (\mathbf{1}_n, X_1, \dots, X_p)$ and a vector of coefficients $\beta = (\beta_0, \dots, \beta_p)^T$. A *linear predictor* is given by $X\beta$.
3. A *link function* g that transforms the linear predictor to the model's prediction through its inverse: $\hat{y} = g^{-1}(X\beta)$.
4. A distribution of the outcome, given the prediction: $p(y|\hat{y})$.
5. Other parameters such as variances, overdispersions and the outcome's upper and/or lower bounds.

Here are the link functions and the outcome distribution that we have used in the linear regression and logistic regression:

- In the linear regression, we used $g(x) = x$ (and so $g^{-1}(x) = x$) and $p(y|\hat{y}) = p(y|X\beta) = \mathcal{N}(y - X\beta, \sigma^2)$.
- In the logistic regression, we used the logit function $g(x) = \text{logit}(x)$ with the logistic inverse: $g^{-1}(x) = \text{logit}^{-1}(x) = e^x/(1+e^x)$. With the prediction $\hat{y} = \text{logit}^{-1}(X\beta)$, the outcome distribution is the Bernoulli distribution: $p(y|\hat{y}) = \hat{y}^y(1-\hat{y})^{1-y}$.

12.2 Poisson and negative binomial regression

12.2.1 Poisson regression

We start with the simplest regression model for count data.

$$y \sim \text{Poisson}(e^{X\beta}).$$

This is a GLM in which:

- The link function is $g(x) = \log x$.
- The prediction is $\hat{y} = g^{-1}(X\beta) = e^{X\beta}$.
- The outcome distribution is $p(y|\hat{y}) \sim \text{Poisson}(\hat{y})$.

From the properties of the Poisson distribution, we have

- $\mathbb{E}[y|X] = e^{X\beta}$.
- $\text{sd}(y|X) = \sqrt{\mathbb{E}[y]} = e^{X\beta/2}$.

Thus, in a Poisson model, the standard deviation of the outcome is already specified by the model. If the expected outcome is $\mathbb{E}[y] = 10$, then the prediction errors are mostly in the range of $\pm\sqrt{\mathbb{E}[y]} \approx \pm 3.33$. But for many datasets, the prediction errors might fall out of these ranges.

12.2.2 Overdispersion and underdispersion

Overdispersion and underdispersion refer to data that show more or less variation than the Poisson model. In other words, when fitting a Poisson model, the residuals of overdispersed data are often greater, while those of underdispersed data are mostly smaller than the square root of the predicted value. For such data, using Poisson models would be inappropriate. In the next section, we introduce another model that allows more prediction errors.

12.2.3 Negative binomial regression

We introduce another model for the count data.

$$y \sim \text{negativebinomial}(e^{X\beta}, \phi),$$

where $\text{negativebinomial}(p, \phi)$ models the number of failures in a sequence of iid $\text{Bernoulli}(p)$ trials before observing ϕ successes (but the range of ϕ can be extended to positive real numbers).

The predictive standard deviation is

$$\text{sd}(y|X) = \sqrt{\mathbb{E}[y|X] + \frac{1}{\phi}\mathbb{E}[y|X]^2}.$$

In the context of count data modeling, ϕ is the "reciprocal dispersion" parameter:

- Lower values of ϕ correspond to more overdispersion.
- Higher values of ϕ correspond to less overdispersion.
- The negative binomial distribution becomes the Poisson distribution in the limit $\phi \to \infty$ (that is, when there is no overdispersion).

12.2.4 Exposure and offset

In many cases, the outcomes depend on the amounts of *exposure* to the environment, and so two different outcomes may not be directly compared. For example, the number of daily deaths by country depends population size. The number of car accidents at an intersection depend on the number of cars running through that intersection.

If this is the case, we may instead let $e^{X\beta}$ be the expected *rate of outcomes* r, and the expected outcome is the product of exposure u and rate r.

$$\mathbb{E}[y] = ur = ue^{X\beta}, \tag{12.1}$$

which leads to a new model with the new predictor u:

$$y \sim \text{negativebinomial}(ue^{X\beta}, \phi).$$

We can separate the exposure and the exponential term by applying the logarithm on both sides of Equation 12.1.

$$\log \mathbb{E}[y] = \log u + X\beta.$$

With this, we call $\log u$ the *offset*.

12.2.5 Example: effect of pest management on reducing cockroach levels

We consider the `Roaches` dataset, which was used to study the effect of pest management on reducing cockroach levels in urban apartments. In the experiment, there were 158 apartments in the treatment group and 104 apartments in the control group. The data consists of the following variables:

Name	Description
y	post-treatment roach count
roach1	pre-treatment roach level
treatment	0 if control, 1 if treatment
senior	1 if the apartment is restricted to elderly
exposure2	number of days the traps had been laid

We create a new predictor named roach100, which isroach1 scaled down by a factor of 100.

```
roaches <- read.csv("data/roaches.csv")
roaches$roach100 <- roaches$roach1/100

head(roaches)
```

```
  X   y roach1 treatment senior exposure2 roach100
1 1 153 308.00         1      0  0.800000   3.0800
2 2 127 331.25         1      0  0.600000   3.3125
3 3   7   1.67         1      0  1.000000   0.0167
4 4   7   3.00         1      0  1.000000   0.0300
5 5   0   2.00         1      0  1.142857   0.0200
6 6   0   0.00         1      0  1.000000   0.0000
```

First, we fit Poisson regression by specifying family=poisson in stan_glm. The number of post-treatment roaches depend on the number of days the traps had been laid, so it makes sense to let exposure2 be the model's exposure.

```
library(rstanarm)
```

```
fit_1 <- stan_glm(y ~ roach100 + treatment + senior,
                  family=poisson,
                  offset=log(exposure2),
                  data=roaches, refresh=0)

print(fit_1)
```

```
stan_glm
 family:       poisson [log]
 formula:      y ~ roach100 + treatment + senior
 observations: 262
 predictors:   4
------
            Median MAD_SD
```

```
(Intercept)  3.1    0.0
roach100     0.7    0.0
treatment   -0.5    0.0
senior      -0.4    0.0

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

Interpreting the coefficients. The fitted Poisson model for the number of post-treatment roaches is

$$y \sim \text{Poisson}(e^{3.1+0.7*\texttt{roach100}-0.5*\texttt{treatment}-0.4*\texttt{senior}}).$$

We can interpret the coefficients of the regression as follows:

- The intercept is the prediction for `roach100` $= 0$, `treatment` $= 0$ and `senior` $= 0$. More precisely, for a non-senior apartment that was roach-free before and did not receive the pest management, the expected number of roaches is $e^3.1 \approx 22$. Note that sometimes some of the predictors cannot be zero, and the intercept cannot be interpreted in that case.
- The coefficient 0.7 of `roach100` indicates that, for each additional 100 roaches (while keeping `treatment` and `senior` at the same level), the expected number of post-treatment roaches increases by a factor of $e^{0.7} \approx 1 + 0.7 = 1.7$, or a 70% increase.
- The coefficient -0.5 of `treatment` indicates that the number of post-treatment roaches in an apartment with pest management is lower that that of an apartment without pest management (with the same level of `roach100` and `senior`) by a factor of $e^{-0.5} \approx 1 - 0.5 = 0.5$, or a 50% decrease.

- The coefficient -0.4 of `senior` indicates that the number of post-treatment roaches in a senior apartment is lower that that of a non-senior apartment (with the same level of `roach100` and `treatment`) by a factor of $e^{-0.4} \approx 1 - 0.4 = 0.6$, or a 40% decrease.

Checking the fit via simulation. Now we check the model's fit by looking at the posterior predictive distribution. As usual, we use the `posterior_predictive` function to generate 4000 numbers of post-treatment roaches, then we sample 400 of them.

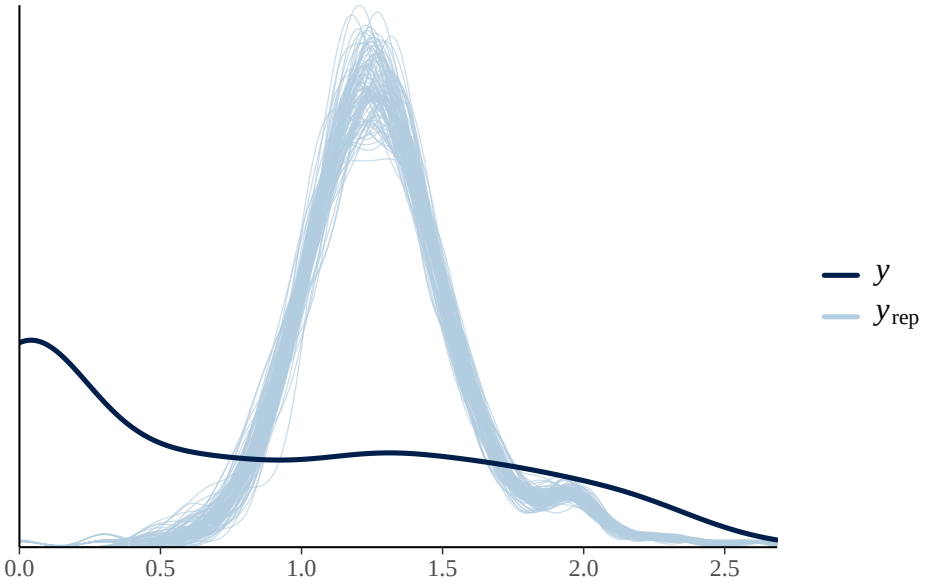
```
yrep_1 <- posterior_predict(fit_1)
n_sims <- nrow(yrep_1)
subset <- sample(n_sims, 100)
```

After that, we use `ppc_dens_overlay` to compare between the distributions of

original and simulated data. Here, we plot both data in the logarithmic scale to make the difference between the plots more pronounced.

```
library(bayesplot)
```

```
ppc_dens_overlay(log10(roaches$y+1), log10(yrep_1[subset,]+1))
```



The plots show that the original data is overdispersed and contains a lot of zeros; this indicates that the Poisson model might not be suitable as it only allows a relatively small number of zeros.

We thus turn to the negative binomial regression, which can be done by specifying `family=neg_binomial_2`. As before, we let `exposure2` be the exposure. There is no need to specify the reciprocal dispersion parameter ϕ—it can be estimated from the data.

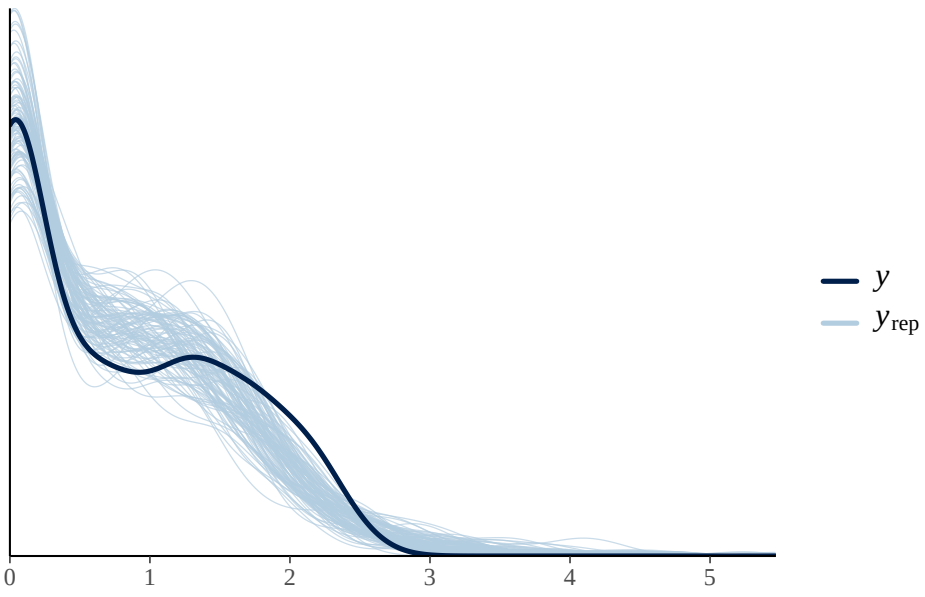
```
fit_2 <- stan_glm(y ~ roach100 + treatment + senior,
                  family=neg_binomial_2,
                  offset=log(exposure2),
                  data=roaches, refresh=0)

print(fit_2)
```

```
stan_glm
 family:       neg_binomial_2 [log]
```

```
 formula:      y ~ roach100 + treatment + senior
 observations: 262
 predictors:   4
------
            Median MAD_SD
(Intercept)  2.8    0.2
roach100     1.3    0.2
treatment   -0.8    0.3
senior      -0.3    0.3

Auxiliary parameter(s):
                     Median MAD_SD
reciprocal_dispersion 0.3    0.0

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

Let us see how the posterior predictive fits our data.

```
yrep_2 <- posterior_predict(fit_2)

n_sims <- nrow(yrep_2)
subset <- sample(n_sims, 100)
ppc_dens_overlay(log10(roaches$y+1), log10(yrep_2[subset,]+1))
```



The negative binomial looks like a better fit with a high probability of zero. However, the model allows the number of roaches to be as high as 10^5, which is unrealistic. We will see how we can make adjustment to this model later in the chapter.

12.3 Logistic-binomial and beta-binomial models

12.3.1 Logistic-binomial model

The logistic regression can be used to model the number of successes from n Bernoulli trials. In this setting, we can use the following GLM design:

- The link function is $g(x) = \text{logit}(x)$.
- The prediction is $\hat{p} = g^{-1}(X\beta) = \text{logit}^{-1}(X\beta)$.
- The outcome distribution is $p(y|\hat{p}) \sim \text{Binomial}(n, \hat{p})$, where n is the number of trials.

Let us try this model on the simulated data of basketball shooting. The following code produces $N = 100$ players, each shooting $n = 20$ shots. We also encode our assumption that the field goal percentage is inversely correlated to the weight.

```
N <- 100
weight <- rnorm(N, 216, 31)
p <- 0.6 - 0.1*(weight - 216)/31
n <- rep(20, N)
y <- rbinom(N, n, p)
data <- data.frame(n=n, y=y, weight=weight)

head(data)
```

```
  n  y   weight
1 20 11 229.9358
2 20  9 267.2749
3 20  9 255.7618
4 20 12 232.9686
5 20 15 197.2956
6 20 11 258.2237
```

To model the count data with the logistic regression, the outcome is a pair of number of successes and number of failures.

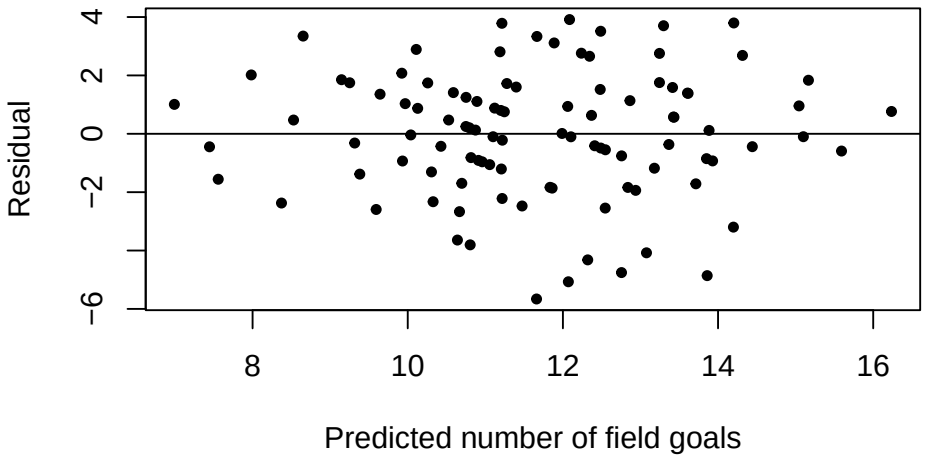
```
fit_1a <- stan_glm(cbind(y, 20-y) ~ weight,
                   family=binomial(link="logit"),
```

```
                    data=data, refresh=0)

  print(fit_1a, digits=3)
```

```
stan_glm
 family:       binomial [logit]
 formula:      cbind(y, 20 - y) ~ weight
 observations: 100
 predictors:   2
------
            Median MAD_SD
(Intercept)  3.407  0.375
weight      -0.014  0.002

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

Since the data is generated from the binomial model, so we expect the residuals to be very small. Let us see if that is the case.

```
  y <- fit_1a$y[,1]
  p_hat <- fitted(fit_1a)
  y_hat <- 20 * p_hat

  plot(y_hat, y - y_hat,
       xlab="Predicted number of field goals",
       ylab="Residual", pch=20)
  abline(0, 0)
```



12.3.2 Overdispersion

Logistic models usually have overdispersion problem, that is, the variation in the data is more than indicated by the model. To detect overdispersion, we first recall the standard deviation of $y \sim \text{Binomial}(n, \hat{p})$, which is $\sqrt{n\hat{p}(1-\hat{p})}$, where $\hat{p} = \text{logit}^{-1}(X\beta)$. Then, we consider the standardized residual:

$$z_i = \frac{y_i - \hat{y}_i}{\text{sd}(\hat{y}_i)} = \frac{y_i - n_i\hat{p}_i}{\sqrt{n_i\hat{p}_i(1-\hat{p}_i)}},$$

which has mean 0 and standard deviation 1. We then formally test for overdispersion by comparing $\sum_{i=1}^{N} z_i^2$ to the χ^2_{N-p} distribution, where p is the number of predictors.

12.3.3 Beta-binomial model

To handle the overdispersion, we modify the outcome distribution of the logistic-binomial model to obtain the *beta-binomial* model, which has the following GLM design:

- The link function is $g(x) = \text{logit}(x)$.
- The prediction is $\hat{p} = g^{-1}(X\beta) = \text{logit}^{-1}(X\beta)$.
- The outcome distribution is

  $$p(y|\hat{p}) \sim \text{Beta-Binomial}\left(n, \hat{p}\phi, (1-\hat{p})\,\phi\right),$$

  where n is the number of trials and $\phi \in (0, 1)$ is an overdispersion parameter.

The parameters of the beta-binomial distribution are chosen so that the mean of y is $n\hat{p}$ and the standard deviation of y is controlled by ϕ.

$$\begin{aligned}\mathbb{E}[y|\hat{p}] &= n\hat{p} \\ \text{sd}(y|\hat{p}) &= \sqrt{n\hat{p}(1-\hat{p})\left(\frac{n+\phi}{1+\phi}\right)}.\end{aligned}$$

Therefore, lowerr ϕ allows for more overdispersion, higher ϕ for less overdispersion. In the limit $\phi \to \infty$ (no overdispersion), we go back to the logistic-binomial model.

To fit the beta-binomial model, we use the `brms` library, which has the `brm` function that allows us to select `beta_binomial` family. We specify the number of trials via the `trials()` function. In this example, the number of trials (i.e. number of shots) for each basketball player is 20. The input of `trials` can

be a vector in the case that the number of trials varies by the data point. The dispersion parameter ϕ is estimated directly from the data, so there is no need to specify ϕ in `brm`.

```
# install.packages("brms")
library(brms)
```

```
fit_1b <- brm(y|trials(20) ~ weight, family=beta_binomial,
              data=data, seed=0, refresh=0)
```

```
Compiling Stan program...
```

```
Start sampling
```

```
print(fit_1b, digits=3)
```

```
 Family: beta_binomial
  Links: mu = logit; phi = identity
Formula: y | trials(20) ~ weight
   Data: data (Number of observations: 100)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI  Rhat Bulk_ESS Tail_ESS
Intercept    3.397     0.417    2.600    4.217 1.002     5104     3028
weight      -0.014     0.002   -0.017   -0.010 1.002     5344     2960

Family Specific Parameters:
    Estimate Est.Error l-95% CI u-95% CI  Rhat Bulk_ESS Tail_ESS
phi  138.750    84.195   43.240  357.844 1.000     2519     2327

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

The dispersion parameter $\phi = 127.2$ tells us that the predictive standard deviation of the beta-binomial model is $\sqrt{\frac{100+127.2}{1+127.2}} \approx 1.33$ times that of the logistic-binomial model.

12.4 Ordered and unordered categorical regression

Sometimes the outcomes can have more than two categories, which can be ordered or unordered.

Examples of ordered categories are:

- Easy, Medium, Hard
- Slow, Normal, Fast
- Urban, Sub-urban, Rural

Examples of unordered categories are:

- Football, Basketball, Baseball
- Car, Bus, Train
- A, B, O, AB

Here, we introduce two models: one to handle ordered categories, and another to handle unordered categories.

12.4.1 Ordered logistic regression

For the data with ordered categories $1, \dots, K$, we may use the following *logistic* model:

- The link function is $g(x) = \text{logit}(x)$, where g can be any strictly increasing function that maps $(0, 1)$ to $(-\infty, \infty)$.
  - If $g(x) = \text{logit}(x)$, then the resulting model is called the *ordered logit model* or *proportional odds model*.
  - If $g(x) = \Phi^{-1}(x)$, where Φ is the cdf of the standard normal distribution, then the resulting model is called the *ordered probit model*.
- The prediction for the k-th category is $\hat{p}_{\leq k} = g^{-1}(c_{k|k+1} + X\beta)$ for $1 \leq k \leq K-1$. Here, we have additional parameters $0 < c_{1|2} < c_{2|3} < \dots < c_{K-1|K}$, called the *cutpoints*.
- The outcome *cumulative* distribution is

  $$\Pr(y \leq k) = \hat{p}_{\leq k}, \qquad k = 1, \dots, K-1$$

  and $\Pr(y \leq K) = 1$. We can calculate the probabilities of individual categories as follows:

  $$\Pr(y = k) = \Pr(y \leq k) - \Pr(y \leq k-1), \qquad k = 1, \dots, K.$$

This formulation is well-defined, as $\{c_{k|k+1}\}_{k=1}^{K}$ is increasing and g (and hence g^{-1}) is increasing imply that $\hat{p}_{\leq k}$ is increasing as well. The two variations of

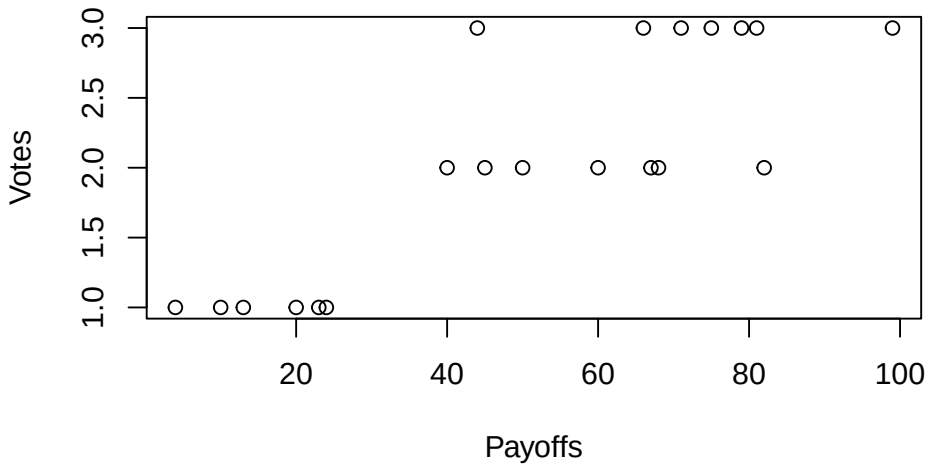
the model are used in different situations: the ordered logit model is used if the probabilities of y are expected to be evenly distributed across all categories, while the ordered probit model is used if the probabilities are expected to be concentrated at the middle values.

Let us try the ordered logit model on the `Storable` data. The data contains records of voting games played between 2-6 college students. A summary of the game is as follows:

- Each student was given a total of 4 votes to play in two rounds.
- The students had the choice to cast 1, 2 or 3 votes on the first round, and the remaining votes on the second round.
- Before casting the first votes, the students were told the payoffs for the winner, which were drawn from the uniform distribution on the interval $[1, 100]$.
- In addition, the students were told the distribution of the payoffs.

Here, we only take the results of the game played between two students. The `vote` column contains the number of the first votes and the `value` column contains the payoffs.

```
data_2player <- read.csv("data/2playergames.csv")
data_401 <- subset(data_2player, person == 401,
                   select = c("vote", "value"))
data_401$factor_vote <- factor(data_401$vote,
                               levels = c(1, 2, 3),
                               labels = c("1", "2", "3"),
                               ordered=TRUE)

plot(data_401$value, data_401$factor_vote,
     xlab="Payoffs", ylab="Votes")
```



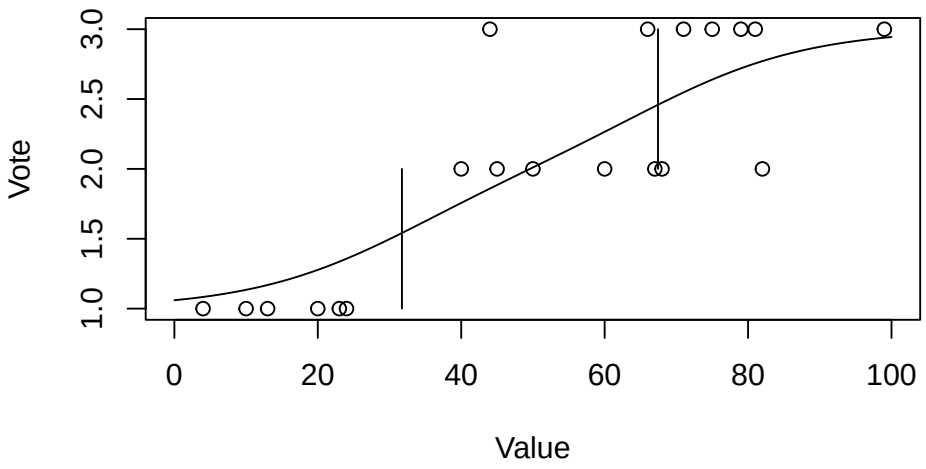
As expected, with more payoffs in the first round, the students were willing to cast more votes.

To fit the ordered logit model, we can use the `stan_polr` function (proportional odds logistic regression) provided in `rstanarm`. Here. we specify the prior mean of the R^2 of the prediction to be 0.3.

```
fit_1 <- stan_polr(factor(vote) ~ value, method="logistic",
                   prior=R2(0.3, "mean"), data=data_401,
                   refresh=0)

print(fit_1)
```

```
stan_polr
 family:       ordered [logistic]
 formula:      factor(vote) ~ value
 observations: 20
------
      Median MAD_SD
value 0.1    0.0

Cutpoints:
    Median MAD_SD
1|2 2.8    1.4
2|3 5.9    2.2

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

The results show two cutpoints: $c_{1|2} = 2.7$ and $c_{2|3} = 5.9$. Below is the plot of the expected vote given the payoffs from the model.



12.4.2 Unordered logistic regression

For the data with unordered categories $1, \dots, K$, we can use the *categorical logit* model, which has the following GLM specification:

- The link function is

$$g(x_1, \dots, x_K) = \left(\log \frac{x_1}{x_K}, \dots, \log \frac{x_{K-1}}{x_K} \right).$$

  The inverse of the link function is called the *multinomial logit*, also known as the *softmax* function.

$$g^{-1}(x_1, \dots, x_{K-1}) = \left(\frac{e^{x_1}}{1 + \sum_k e^{x_k}}, \dots, \frac{e^{x_{K-1}}}{1 + \sum_k e^{x_k}}, \frac{1}{1 + \sum_k e^{x_k}} \right).$$

- The prediction is $(\hat{p}_1, \dots, \hat{p}_K) = g^{-1}(X\beta_1, \dots, X\beta_{K-1})$ for $1 \le k \le K-1$. Notice that we now have $K-1$ sets of parameters: $\beta_1, \dots, \beta_{K-1}$.
- The outcome distribution is

$$\Pr(y = k) = \hat{p}_k, \qquad k = 1, \dots, K.$$

Let us fit this model on the `iris` dataset, which contains data of pedal height, pedal width, sepal height and sepal width of three different species of iris.

```
data(iris)

head(iris)
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

To fit the unordered logistic regression model of the species on the other predictors, we again use the `brm` function.

```
fit_2 <- brm(Species ~ ., family = categorical(link = "logit"),
             data=iris, prior=set_prior("normal (0, 8)"),
             refresh=0)

print(fit_2)
```

```
 Family: categorical
  Links: muversicolor = logit; muvirginica = logit
Formula: Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width
   Data: iris (Number of observations: 150)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Population-Level Effects:
                         Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
muversicolor_Intercept      -14.25     19.96   -55.14    22.06 1.00     2033
muvirginica_Intercept       -40.64     23.24   -89.32     1.26 1.00     2106
muversicolor_Sepal.Length     1.51      4.33    -6.81    10.23 1.01     1774
muversicolor_Sepal.Width     -4.21      4.04   -12.52     3.35 1.00     1522
muversicolor_Petal.Length     6.89      3.28     0.80    13.76 1.00     1481
muversicolor_Petal.Width      0.07      5.32   -10.38    10.37 1.00     2092
muvirginica_Sepal.Length     -0.78      4.38    -9.02     8.12 1.00     1778
muvirginica_Sepal.Width      -7.03      4.34   -15.76     1.04 1.00     1690
muvirginica_Petal.Length     13.49      3.99     6.20    21.85 1.00     1576
muvirginica_Petal.Width       9.97      5.67    -0.89    21.18 1.00     2111
                         Tail_ESS
muversicolor_Intercept       2213
muvirginica_Intercept        2219
muversicolor_Sepal.Length    1886
muversicolor_Sepal.Width     1494
muversicolor_Petal.Length    1740
muversicolor_Petal.Width     2353
muvirginica_Sepal.Length     2027
muvirginica_Sepal.Width      1774
muvirginica_Petal.Length     1938
```

```
muvirginica_Petal.Width       2530

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

Since the data has three categories, the fitted model has two sets of coefficients associated with two categories: versicolor and virginica.

12.5 Models with unequal error standard deviations

The usual linear regression model $y \sim \mathcal{N}(X\beta, \sigma^2)$ assumes that the error standard deviation σ is fixed. That said, we can allow the standard deviation to vary by the values of the predictors—such condition is called *heteroscedasticity*. For example, it is possible to fit the model $y \sim \mathcal{N}(X\beta_1, e^{X\beta_2})$, where β_1 and β_2 are the model's parameters. In the example of earnings data, we could fit the linear regression of log-earnings with the error standard deviation in the form of $e^{c+d*\texttt{male}}$, allowing different error standard deviations for women and men. This can be done using `brm`, with `bf` as a formula helper.

```
earnings <- read.csv("data/earnings.csv")
```

```
fit_1 <- brm(bf(log(earn)|subset(earn>0) ~ height + male,
                                    sigma ~ male),
             data=earnings, refresh=0)
```

```
Compiling Stan program...

Start sampling
```

```
print(fit_1)
```

```
 Family: gaussian
  Links: mu = identity; sigma = log
Formula: log(earn) | subset(earn > 0) ~ height + male
         sigma ~ male
   Data: earnings (Number of observations: 1816)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Population-Level Effects:
               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept          7.94      0.49     6.97     8.92 1.00     3518     3251
```

```
sigma_Intercept    -0.12      0.02    -0.16    -0.08 1.00     5246     2823
height              0.02      0.01     0.01     0.04 1.00     3497     3142
male                0.37      0.06     0.25     0.48 1.00     3578     2986
sigma_male         -0.06      0.04    -0.13     0.01 1.00     4798     2614

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

We can allow heteroskedasticity in other models as well. For example, the negative binomial model can be extended to $y \sim \text{negativebinomial}(e^{X\beta_1}, e^{X\beta_2})$, in which ϕ depends on the predictors. Let us try this on the roaches data, with ϕ (the shape parameter) depending on the treatment and the seniority.

```
fit_2 <- brm(bf(y ~ treatment + senior + offset(log(exposure2)),
                shape ~ treatment + senior),
             family=negbinomial,
             data=roaches, refresh=0)
```

```
Compiling Stan program...
```

```
Start sampling
```

```
print(fit_2)
```

```
 Family: negbinomial
  Links: mu = log; shape = log
Formula: y ~ treatment + senior + offset(log(exposure2))
         shape ~ treatment + senior
   Data: roaches (Number of observations: 262)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Population-Level Effects:
               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept          3.76      0.21     3.37     4.19 1.00     5559     2949
shape_Intercept   -1.27      0.14    -1.56    -0.98 1.00     6377     3419
treatment         -0.55      0.27    -1.09    -0.04 1.00     5399     2845
senior            -0.73      0.34    -1.36    -0.03 1.00     4933     2621
shape_treatment   -0.16      0.19    -0.53     0.21 1.00     5908     3136
shape_senior      -0.56      0.22    -1.01    -0.14 1.00     5716     2983

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

12.6 Mixture models for data with many zeros

12.6.1 Hurdle models

We have briefly mentioned back in Chapter 7.1.1 how to handle data with a lot of zero-valued outcomes, specifically for log-linear models. In summary, we can use the logistic regression to classify whether the outcome is zero, and then we use the linear regression to model non-zero outcomes. We demonstrate here how to do this on the earnings data with `stan_glm`. First, we fit a logistic regression model on whether the earning is zero.

```
  # (earn > 0) is an indicator of whether the earning is zero
  fit_1a <- stan_glm((earn == 0) ~ height + male,
                     family=binomial(link="logit"),
                     data=earnings, refresh=0)

  print(fit_1a)
```

```
stan_glm
 family:       binomial [logit]
 formula:      (earn == 0) ~ height + male
 observations: 1816
 predictors:   3
------
            Median MAD_SD
(Intercept)  3.0    1.9
height      -0.1    0.0
male        -1.7    0.3

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

Then, we fit a linear regression model on the data with non-zero earnings.

```
  fit_1b <- stan_glm(log(earn) ~ height + male,
                     data=earnings, subset=earn>0,
                     refresh=0)

  print(fit_1b)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      log(earn) ~ height + male
 observations: 1629
 predictors:   3
```

```
 subset:       earn > 0
------
            Median MAD_SD
(Intercept) 8.0    0.5
height      0.0    0.0
male        0.4    0.1

Auxiliary parameter(s):
      Median MAD_SD
sigma 0.9    0.0

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

The resulting model is a *mixture* of the linear and logistic models. Suppose that we want to simulate the earnings of a randomly chosen 68-inch tall woman from this model; this can be done in two steps: first, we use the logistic model to predict whether her earning is zero, and if it is not zero, we use the linear model to predict her earning. Both predictions can be obtained using `posterior_predict`.

```
new <- data.frame(height=68, male=0)
pred_1a <- posterior_predict(fit_1a, newdata=new)
pred_1b <- posterior_predict(fit_1b, newdata=new)
pred <- ifelse(pred_1a==1, 0, exp(pred_1b))

print(pred[1:10])
```

```
 [1] 14555.332 23870.347  9702.973  2905.343  5832.872 19438.731  3011.978
 [8]  5149.068 69719.996 18398.429
```

To fit the mixture model in a single step, we may use `brm` with the `hurdle_lognormal` family.

```
fit_2 <- brm(bf(earn ~ height + male, hu ~ height + male),
             family=hurdle_lognormal,
             data=earnings, refresh=0)
```

```
Compiling Stan program...
```

```
Start sampling
```

```
print(fit_2)
```

```
 Family: hurdle_lognormal
```

```
  Links: mu = identity; sigma = identity; hu = logit
Formula: earn ~ height + male
         hu ~ height + male
   Data: earnings (Number of observations: 1816)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Population-Level Effects:
             Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept        7.96      0.51     6.97     8.95 1.00     3423     3276
hu_Intercept     2.85      1.94    -0.94     6.68 1.00     3589     2963
height           0.02      0.01     0.01     0.04 1.00     3371     3171
male             0.37      0.06     0.25     0.50 1.00     3258     2985
hu_height       -0.07      0.03    -0.13    -0.01 1.00     3560     2968
hu_male         -1.66      0.32    -2.34    -1.08 1.00     3019     1972

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma     0.87      0.01     0.84     0.90 1.00     4635     2992

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

Notice that the coefficients of the linear regression (`Intercept`, `height` and `male`) and the logistic regression (`hu_Intercept`, `hu_height` and `hu_male`) are almost identical to the ones from the direct fits above. For count data with many zeros, one can use the `hurdle_poisson` or `hurdle_negbinomial` family in `brm`.

12.6.2 Zero-inflated models

As in hurdle models, a zero-inflated model also consists a logistic regression model that predicts whether the outcome is zero, followed by a regression model of our choice. The difference is that, in a hurdle model, outputs of the second model must be non-zero (e.g. log-linear models), while in a zero-inflated model, they can be zero (e.g. Poisson or negative binomial models).

In Section 12.2.5, we have fitted a negative binomial model to the roaches data. Let us try fitting a zero-inflated negative binomial model on this data using `brm`. Here, we use all predictors, including the exposure variable, to predict whether the number of post-treatment roaches is zero.

```
fit_3 <- brm(bf(y ~ roach100 + treatment + senior +
                    offset(log(exposure2)),
                zi ~ roach100 + treatment + senior +
```

```
                    offset(log(exposure2))),
                 family=zero_inflated_negbinomial(),
               data=roaches, refresh=0)
```

```
Compiling Stan program...
```

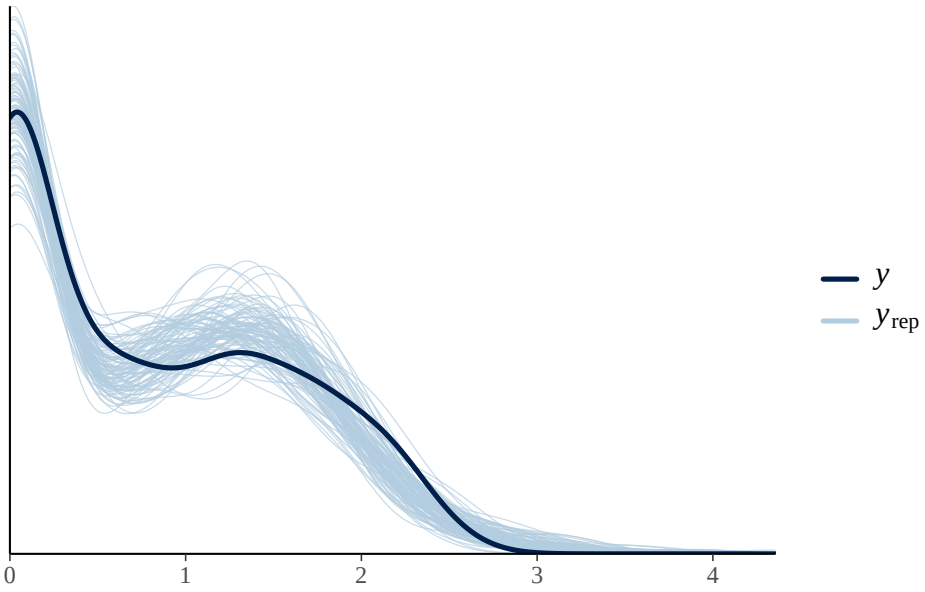
```
Start sampling
```

```
  print(fit_3)
```

```
 Family: zero_inflated_negbinomial
  Links: mu = log; shape = identity; zi = logit
Formula: y ~ roach100 + treatment + senior + offset(log(exposure2))
         zi ~ roach100 + treatment + senior + offset(log(exposure2))
   Data: roaches (Number of observations: 262)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Population-Level Effects:
             Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept        3.17      0.20     2.78     3.58 1.00     3549     2463
zi_Intercept    -1.03      0.52    -2.16    -0.13 1.00     2615     2772
roach100         0.87      0.17     0.55     1.22 1.00     3204     2663
treatment       -0.55      0.22    -0.99    -0.13 1.00     3567     2934
senior          -0.10      0.25    -0.59     0.43 1.00     3279     2833
zi_roach100    -12.72      4.27   -23.09    -6.34 1.00     1649     1242
zi_treatment     1.20      0.50     0.28     2.26 1.00     3213     2459
zi_senior        1.01      0.50     0.07     2.00 1.00     3143     2830

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
shape     0.49      0.06     0.38     0.61 1.00     2959     2929

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

We again check the model's fit with fake data simulation.

```
  yrep_3 <- posterior_predict(fit_3)

  n_sims <- nrow(yrep_3)
  subset <- sample(n_sims, 100)
  ppc_dens_overlay(log10(roaches$y+1), log10(yrep_3[subset,]+1))
```



The fit of the model is slightly better than that of the negative binomial model. Specifically, there are fewer apartments whose expected post-treatment numbers of roaches are larger than 10,000.

Chapter 13

Poststratification: regression with non-representative sample

Sometimes the sample that we use to fit the regression model does not represent the population well. For example, a randomized experiment can have equal numbers of males and females, but the same might not hold for the people in the city. In this case, the regression on the sample cannot be used to infer the population, but we can regress on each separate group and average the predictions according to the population; this technique is called *post-stratification*.

Suppose that we would like to predict y on a predictors x_1 with a model $\hat{y} = g^{-1}(\beta_0 + \beta_1 X_1)$ for some link function g. If the sample is not representative of the population, we cannot infer y on any level of X_1. But if we have access to the data of the population, such as the census, we can use it to recalibrate the prediction of y as follows: Suppose that there are records of additional variables $X_{-1} = (X_2, \dots, X_p)$ in both the sample and the population. We can instead fit a regression model of y on the other variables:

$$\hat{y} = g^{-1}(\hat{\beta}_0 + \hat{\beta}_1 X_1 + \dots + \hat{\beta}_p X_p).$$

Then, to infer y at level $X_1 = x_1$, we predict $\hat{y}$ at each *stratum*, that is, each observed values $x_{-1} = (x_2, \dots, x_p)$ of X_{-1}.

$$\hat{y}_{x_{-1}|x_1} = g^{-1}(\hat{\beta}_0 + \hat{\beta}_1 x_1 + \dots + \hat{\beta}_p x_p).$$

The final prediction $\hat{y}_{x_1}$ is then obtained by combining these per-stratum predictions, weighted by the stratum proportions $p_{x_{-1}|x_1}$ in the $X_1 = x_1$ subpopu-

lation.

$$\hat{y}_{x_1} = \sum_{x_{-1}} p_{x_{-1}|x_1} \hat{y}_{x_{-1}|x_1}.$$

We demonstrate using Monica Alexander's example of post-marriage name change (Alexander 2019). The data, collected by Philip Cohen, consists of responses from 5,000 people regarding their decisions for a name change after marriage. However, the respondents tend to have higher education levels and are younger than average. If we want to, for example, find out the proportion of women who kept their names after marriage, we could not infer it directly from the data.

Instead, we fit a logistic regression model on the responses, calculate the proportion, and recalibrate it using population's data, namely the U.S. census recorded at IPUMS USA.

Before anythin, we install `ipumsr` to read the census data and `haven` to read a `*.dta` file.

```
library(ipumsr)
library(haven)
library(rstanarm)
```

First, we import the survey data and only focus on ever-married women. We remove rows with missing data, divide ages into age groups, marriage years into decades, and education levels into pre-bachelor, bachelor and post-bachelor.

```
mncs <- read_dta("data/MNCS-PV2.dta")
mncs <- mncs[,c("yrmar",
                "agemar",
                "agemarc",
                "genmar",
                "spgenmar",
                "namechg",
                "ednow",
                "state")]
mncs <- mncs[!is.na(mncs$agemar) &
             !is.na(mncs$yrmar) &
             mncs$genmar == 2 &
             mncs$spgenmar == 1,]
mncs$kept_name <- as.numeric(mncs$namechg == 1)
mncs$state_name <- tolower(
    as.character(
        factor(
```

```
            mncs$state,
            levels = attributes(mncs$state)$labels,
            labels = names(attributes(mncs$state)$labels)
        )
    )
)
mncs$age <- mncs$agemar + (2019 - mncs$yrmar)
mncs$age_group <- (as.character(
         cut(mncs$age,
             breaks = c(seq(20, 80, by = 5), Inf),
             labels = seq(20, 80, by = 5),
             right = FALSE)
     )
)
mncs$decade_married <- (as.character(
         cut(mncs$yrmar,
             breaks=c(seq(1969, 2019, by = 10), Inf),
             labels=seq(1969, 2019, by = 10),
             right=FALSE
             )
     )
)
mncs$educ_group <- cut(mncs$ednow,
                       breaks = c(-1, 4.5, 5, 9),
                       labels = c("<BA", "BA", ">BA"))
mncs <- mncs[c("kept_name", "state_name", "age_group",
               "decade_married", "educ_group")]

head(mncs)
```

```
# A tibble: 6 x 5
  kept_name state_name     age_group decade_married educ_group
      <dbl> <chr>          <chr>     <chr>          <fct>
1         0 ohio           50        1979           >BA
2         0 virginia       35        1999           >BA
3         1 new york       35        2009           >BA
4         0 rhode island   55        1999           >BA
5         0 illinois       35        2009           >BA
6         0 north carolina 25        2009           >BA
```

We do not share the U.S. census data here, but you can sign up at the IPUMS US website and request for the data yourself. You may request for a single-year or 5-year data that includes the following variables: `AGE`, `PERWT`, `SEX`, `STATEFIP`, `MARST`, `YRMARR`, `EDUC`. In addition to the `*.dat` data, do not forget to download the accompanied `*.xml` file.

```
ddi <- read_ipums_ddi("data/usa_00001.xml")
ipums <- read_ipums_micro(ddi)
```

```
Use of data from IPUMS USA is subject to conditions including that users should
cite the data appropriately. Use command `ipums_conditions()` for more details.
```

```
ipums <- ipums[ipums$SEX == 2 &  # only woman
               ipums$AGE > 14 &
               ipums$MARST != 6 &  # exclude singles
               ipums$YRMARR > 1968,]
ipums$state_name <- tolower(
    as.character(
        factor(
            ipums$STATEFIP,
            levels = attributes(ipums$STATEFIP)$labels,
            labels = names(attributes(ipums$STATEFIP)$labels)
        )
    )
)
ipums$age_group <- as.character(
    cut(ipums$AGE,
        breaks = c(seq(20, 80, by = 5), Inf),
        labels = seq(20, 80, by = 5),
        right = FALSE
        )
)
ipums$decade_married <- as.character(
    cut(ipums$YRMARR,
        breaks = c(seq(1969, 2019, by = 10), Inf),
        labels = seq(1969, 2019, by = 10),
        right = FALSE
        )
)
ipums$educ_group <- cut(ipums$EDUC,
                        breaks = c(-1, 9.5, 10, 12),
                        labels = c("<BA", "BA", ">BA"))
ipums <- ipums[c("state_name", "PERWT", "age_group",
                 "decade_married", "educ_group")]

head(ipums)
```

```
# A tibble: 6 x 5
  state_name PERWT age_group decade_married educ_group
  <chr>      <dbl> <chr>     <chr>          <fct>
```

```
1 alabama       19 75        1989           <BA
2 alabama       30 50        1989           <BA
3 alabama       24 65        1969           <BA
4 alabama        3 60        1969           <BA
5 alabama        3 35        1999           BA
6 alabama       57 25        2009           <BA
```

Then, we aggregate the number of people (PERWT) by the other variables.

```
ipums <- aggregate(PERWT ~ age_group + state_name
                   + educ_group + decade_married,
                   ipums, sum)

head(ipums)
```

```
  age_group state_name educ_group decade_married PERWT
1        50    alabama        <BA           1969   160
2        55    alabama        <BA           1969 11971
3        60    alabama        <BA           1969 42487
4        65    alabama        <BA           1969 40086
5        70    alabama        <BA           1969 18581
6        75    alabama        <BA           1969  7568
```

Suppose that we would like to calculate the proportion of women who changed their names for each age groups. Then, in each age groups, we need to calibrate our model's predictions according to the proportion of the other variables.

```
ipums <- transform(ipums, prop = ave(PERWT,
                                     age_group,
                                     FUN = prop.table))
ipums$PERWT <- NULL

head(ipums)
```

```
  age_group state_name educ_group decade_married         prop
1        50    alabama        <BA           1969 1.714893e-05
2        55    alabama        <BA           1969 1.202384e-03
3        60    alabama        <BA           1969 4.400218e-03
4        65    alabama        <BA           1969 5.183921e-03
5        70    alabama        <BA           1969 4.467144e-03
6        75    alabama        <BA           1969 4.001971e-03
```

Now we train the model on the survey data. Here, we decide not to regress on the education group

```
fit <- stan_glm(kept_name ~ age_group
                + state_name
                + decade_married
                + educ_group,
                family = binomial(link = "logit"),
                data = mncs, refresh = 0)
```

We then make point predictions on the census data.

```
ipums$pred <- predict(fit, type = "response",
                      newdata = ipums)
print(ipums$pred[1:10])
```

```
 [1] 0.02413101 0.02965398 0.05119582 0.08420307 0.09098488 0.03837479
 [7] 0.03763646 0.08644370 0.10380387 0.16493608
```

Let us use these predictions to estimate the proportion of 25-to-30-year-old women who keep their names after marriage. The estimate is given by the sum of predictions for all strata in the 25-30 age group, weighted by the proportions of the corresponding stratum that we just calculated from the census.

```
age_group_25 <- (ipums$age_group == 25)
ipums25 <- ipums[age_group_25, ]

prop_age25 <- ipums25$prop
pred_age25 <- ipums25$pred
poststrat_est <- sum(prop_age25 * pred_age25)

cat("Predicted proportion for age group 25 =", poststrat_est, "\n")
```

```
Predicted proportion for age group 25 = 0.1870944
```

To predict the proportions at all age levels, we can use `aggregate` to sum the weighted predictions over the age groups. Here, we use `transform` to create a new column named `total`, which consists of the prediction-proportion products.

```
poststrat_pred <- aggregate(
    total ~ age_group,
    transform(ipums,
              total = prop * pred),
    sum)

print(poststrat_pred[1:10, ])
```

```
   age_group     total
1         20 0.2673061
2         25 0.1870944
3         30 0.2125474
4         35 0.2540660
5         40 0.2762589
6         45 0.3078861
7         50 0.3289182
8         55 0.3116307
9         60 0.3367970
10        65 0.3735635
```

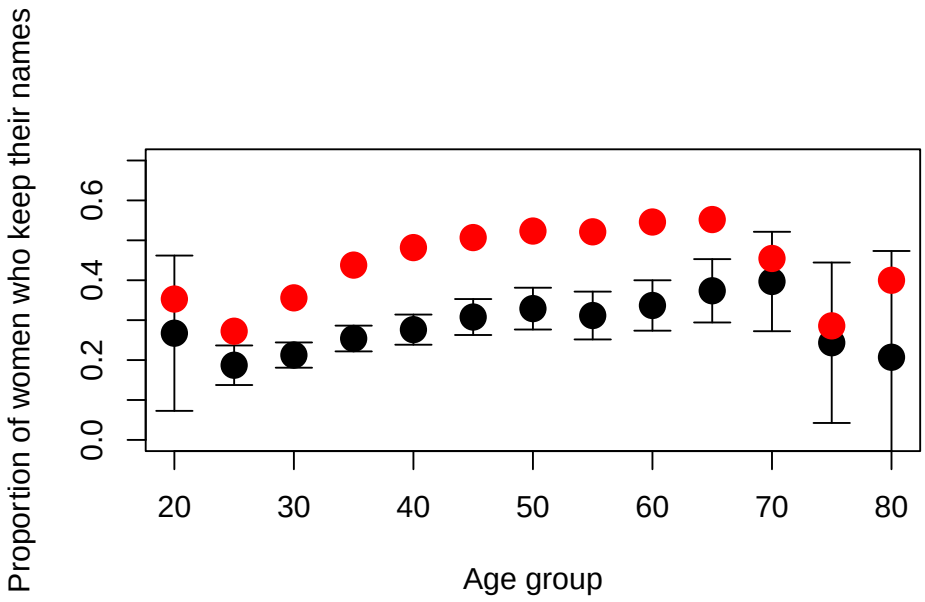
Instead of point predictions, we can post-stratify on the posterior predictive distributions at each age group. Here, each row of the dataframe `poststrat` consists of 4,000 posterior weighted predictions at each age group.

```
pred_sim <- t(posterior_epred(fit, newdata = ipums))
poststrat <- data.frame(
    age_group = strtoi(ipums$age_group),
    ipums$prop * pred_sim)
poststrat <- aggregate(. ~ age_group,
                       poststrat,
                       sum)

print(poststrat[, 1:6])
```

```
   age_group        X1        X2         X3        X4        X5
1         20 0.2047181 0.1992183 0.20903781 0.2086398 0.1852048
2         25 0.1757313 0.1777441 0.16962754 0.1555006 0.1953599
3         30 0.2062349 0.1976563 0.21615264 0.1979541 0.2152249
4         35 0.2151183 0.2420801 0.24137215 0.2223249 0.2508102
5         40 0.2495369 0.2679932 0.26539967 0.2521099 0.2672938
6         45 0.2991570 0.2816275 0.30214115 0.2560419 0.3045929
7         50 0.2841912 0.3113215 0.28739235 0.3083620 0.2990760
8         55 0.2913605 0.3210117 0.33632878 0.3055867 0.3186225
9         60 0.3122404 0.3507243 0.33530441 0.3475443 0.3378746
10        65 0.4055613 0.3253300 0.35137288 0.3250265 0.3639057
11        70 0.4154457 0.3584115 0.41041909 0.3292691 0.3303671
12        75 0.2459736 0.2422803 0.30387694 0.1464125 0.3225784
13        80 0.1458281 0.4780080 0.05933012 0.3092706 0.1037218
```

Now we can plot the prediction, with uncertainty, of the proportion of women in a particular age group who keep their names. The predictions across all age groups are compared with the simple predictions using the sample proportions.

```r
ncols <- ncol(poststrat)
means <- rowMeans(poststrat[, 2:ncols])
sds <- apply(poststrat[, 2:ncols], 1, sd)

simple_props <- aggregate(kept_name ~ age_group,
                          mncs,
                          mean)

# Plot stratified predictions
plot(poststrat$age_group,
     means,
     xlab = "Age group",
     ylab = "Proportion of women who keep their names",
     ylim = c(0, 0.7),
     pch = 16, cex = 2)

# Add error bars
arrows(x0 = poststrat$age_group,
       y0 = means - 2 * sds,
       x1 = poststrat$age_group,
       y1 = means + 2 * sds,
       code = 3,
       angle = 90,
       length = 0.1)

# Plot sample proportions
points(poststrat$age_group,
       simple_props$kept_name,
       col = "red", pch = 16,
       cex = 2)
```



We can see that the post-stratified predictions are lower than the sample proportions across all age groups. A possible explanation is that the respondents were mostly young and highly educated people, who are more likely to change their names after marriage.

Part III

Causal inference

We have been predicting the outcome using regression fitted on the data. In the following chapters, we are concerned with *causal* questions: what would happen to an outcome y if the unit is given a treatment, intervention, or exposure z? We will address challenges and discuss regression as a tool to answer such questions.

Chapter 14

Basics of causal inference

14.1 A running example

Consider the following hypothetical scenario: over the past few decades, omega-3 fatty acids has been promoted and advertised as an effective supplement for reducing blood pressure. Being skeptical about this clain, you decide to investigate. You found eight friends who agreed to join your experiment.

- Four of the friends were in the treatment group. They agreed to take the fish oil supplement every day for one year.
- The other four friends were in the control group. They agreed to simply maintaining their diets, free from fish oil supplement.

After one year, you measure systolic blood pressure of each of the eight participants. For simplicity, we consider systolic blood pressure of 160mmHg and higher as "high blood pressure".

14.1.1 Potential outcomes, counterfactuals, and causal effects

To formalize causal problems in this study, we assign several notations to each participant $i \in \{1, \dots, 8\}$

- Let z_i be the *treatment* variable of i.
  - $z_i = 0$ if i is in the control group (i.e. he/she did not take any oil supplement).
  - $z_i = 1$ if i is in the treatment group (i.e. he/she had been taking fish oil supplement).
- Let y_i^0 and y_i^1 be two *outcome* variables.
  - y_i^0 denotes the blood pressure of i if he/she did not take any supplement.

– y_i^1 denotes the blood pressure of i if he/she had been taking the supplement.

The outcome variables y_i^0 and y_i^1 are commonly referred to as **potential outcomes**. It is important to note that the potential outcomes are assigned to all participants, regardless of whether or not they had received the treatment (i.e. the supplement).

Thus, for everyone in the control group ($z_i = 0$), the value of y_i^0 is observed, while that of y_i^1 is unobserved. And for everyone in the treatment group ($z_i = 1$), the value of y_i^1 is observed while that of y_i^0 is unobserved. Sometimes, we might be interested in *what would happen* if a particular participant from the control group had recieved the treatment and vice versa. In this case, the outcomes of interest would be y_i^1 and y_i^0, respectively. Such outcomes are referred to as **counterfactual outcomes**. For each participant, it is impossible to directly measure his/her counterfactual outcome—this is commonly referred to as the **fundamental problem of causal inference**.

Let y_i be the observed outcome (not potential outcome) of person i. We can express it in terms of the potential outcomes:

$$y_i = y_i^0(1 - z_i) + y_i^1 z_i.$$

The **causal effect** of supplement versus non-supplement for person i is the difference between the two potential outcomes:

$$\tau_i = y_i^1 - y_i^0.$$

A hypothetical data of the experiment is shown in the table below. We see that at least one of the potential outcomes is always missing.

Unit i	Treatment z_i	Potential outcome #1 y_i^0	Potential outcome #2 y_i^1	Observed outcome y_i	Causal inference τ_i
Alex	0	140	?	140	?
Anna	0	140	?	140	?
Bill	0	150	?	150	?
Bob	0	150	?	150	?
Cindy	1	?	155	155	?
Carol	1	?	155	155	?
Dan	1	?	160	160	?
Dave	1	?	160	160	?

14.2 Average causal effects

We introduce several notions of causal effect.

First is the causal effect on an individual; this is sometimes called *individual treatment effect* (ITE).

$$\text{individual treatment effect: } \tau_i = y_i^1 - y_i^0.$$

The *sample average treatment effect* (SATE) is the average of ITE across all units in the sample.

$$\tau_{\text{SATE}} = \frac{1}{n}\sum_{i=1}^{n}(y_i^1 - y_i^0).$$

The *conditional average treatment effect* (CATE) is the average treatment effect of a subset $\mathcal{C}$ of the units, such "men" or "people who received the treatments".

$$\tau_{\text{CATE}} = \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}}(y_c^1 - y_c^0),$$

where $|\mathcal{C}|$ is the number of units in the subset $\mathcal{C}$.

The *population average treatment effect* (PATE) is the average treatment effect across the population.

$$\tau_{\text{PATE}} = \frac{1}{N}\sum_{i=1}^{N}(y_i^1 - y_i^0).$$

If the sample is a randon sample, then we can use SATE to estimate PATE. And any unbiased estimator of SATE is also an unbiased estimator of PATE.

Estimation of the average treatment effects is straightforward when the experiment is performed at completely random.

14.3 Randomized experiments

14.3.1 Completely randomized experiments

In a *completely randomized experiments*, everyone in the sample is equally likely to be assigned to the treatment group and the control group. With this, the averages of y_i^1's and y_i^0 are representative of those of the sample mean, and so we can estimate SATE with

$$\tau = \frac{1}{n/2} \sum_{i,z_i=0} y_i^0 - \frac{1}{n/2} \sum_{i,z_i=1} y_i^1,$$

which is the same as the coefficient τ of the regression $y_i = a + \tau z_i$. In other words, in a completely randomized experiment, we can estimate SATE using a regression of the outcome on the treatment assignment. If the sample is representative of the population, we can also use τ to estimate PATE.

To illustrate how the randomness affects the estimate of SATE, we compare between two sets of data, which also include the ages of the units. The first one is an ideal scenario for a randomized experiment. In each row, the bold potential outcome is the one actually seen, and the non-bold one is not observed.

Table 14.2: Data of the fish oil supplement experiment. Bold numbers are observed potential outcomes and non-bold number are unobserved.

Unit i	Age x_i	Treatment z_i	Potential outcome #1 y_i^0	Potential outcome #2 y_i^1	Observed outcome y_i
Alex	40	0	**140**	135	140
Anna	40	1	140	**135**	135
Bill	50	0	**150**	140	150
Bob	50	1	150	**140**	140
Cindy	60	0	**160**	155	160
Carol	60	1	160	**155**	155
Dan	70	0	**170**	160	170
Dave	70	1	170	**160**	160

In this case, the simple difference in means, $147.5 - 155.5$, is the same as SATE of -7.5.

Now let us compare this with a less ideal randomized scenario:

Table 14.3: Another data of the fish oil supplement experiment in which younger participants were more likely to receive the treatment.

Unit i	Age x_i	Treatment z_i	Potential outcome #1 y_i^0	Potential outcome #2 y_i^1	Observed outcome y_i
Alex	40	1	140	**135**	135
Anna	40	1	140	**135**	135
Bill	50	1	150	**140**	140
Bob	50	0	**150**	140	150
Cindy	60	0	**160**	155	160

Unit i	Age x_i	Treatment z_i	Potential outcome #1 y_i^0	Potential outcome #2 y_i^1	Observed outcome y_i
Carol	60	0	**160**	155	160
Dan	70	0	**170**	160	170
Dave	70	1	170	**160**	160

We can see that the the treatment is assigned to mostly younger participants, and the difference in the means, $142.5 - 160 = -17.5$, significantly underestimates the SATE of -7.5.

In many scenarios, the sample is not perfectly randomized, so we have to make some *adjustment* for the imbalance, a technique that we will introduce later.

14.3.2 Randomized blocks experiments

In some experiments, the participants can be divided by the observed values of a subset of variables into various *blocks*. If there are equal numbers of control and treated units within each block like in Table 14.2, we can simply estimate SATE using the difference of the means.

However, in some experiments, the ratios of control and treated units might be different across the blocks. In the fish oil supplement example, older people might be more in need of the supplement than the younger people, so the reseachers might simulate this pattern by assigning the treatment to more people in the older block than the younger block, as shown in the table below.

Unit i	Age x_i	Treatment z_i	Potential outcome #1 y_i^0	Potential outcome #2 y_i^1	Observed outcome y_i
Alex	40	0	**140**	135	140
Anne	40	0	**140**	135	140
Anna	40	1	140	**135**	135
Bill	50	0	**150**	140	150
Brad	50	0	**150**	140	150
Bob	50	1	150	**140**	140
Cindy	60	0	**160**	155	160
Carol	60	1	160	**155**	155
Chris	60	1	160	**155**	155
Dan	70	0	**170**	160	170
Dave	70	1	170	**160**	160
Drew	70	1	170	**160**	160

The difference between the means is -0.83 overestimates the SATE of -7.5.

A better estimate of SATE can be obtained by first computing the difference between the means in each block, and then taking a weighted average of the differences, with weights proportional to the number of units in each.

Another way to obtain the estimate is by fitting a linear regression on the treatment variable and indicators for the three of the four blocks:

$$y_i = a + \tau_{\text{RB}} z_i + \beta_1 b_{1i} + \beta_2 b_{2i} + \beta_3 b_{3i}.$$

Of course, this is an accurate estimator of SATE if there is only few variation of the outcomes within each block, or in other words, if the blocking variable is highly predictive of the outcome. Thus, in a randomized blocks experiment, we should select blocking variables that are predictive of the outcome, based on either theory or results from previous studies.

Randomized blocks experiments have one advantage over completely randomized experiments: their estimates of SATE (or PATE) have smaller standard deviations due to the homogeneity of the blocking variables.

14.3.3 Matched pairs experiments

A *matched pairs experiment* is a special case of a randomized block design with only two units in each block. For example, Table 14.2 shows data of a matched pairs experiment. In each block, we randomly select one unit (with 0.5 probability) to receive the treatment, and the other unit to receive the control.

This design is very effective when the members of each matched pair are similar to each other, because the difference of the observed outcomes in each pair is a good estimate for the treatment effect. Suppose that there are K pairs. Let y_j^T be the outcome of the treated unit y_j^C be the outcome of the controlled unit in pair j. Then, we can estimate SATE using the average of those K differences:

$$\bar{d} = \frac{1}{K} \sum_{k=1}^{K} (y_j^T - y_j^C).$$

$d_j = y_j^T - y_j^C$, , and Such pairs arise naturally in children of the same family, students in the same class or workers in the same department.

14.3.4 Group or cluster-randomized experiments

Sometimes, due to logistical or cost reasons, the treatment is assigned at the group level. For example, a schoolwide schedule reform requires assigning the new schedule to all students in a school; a new working hours policy requires changing the working hours to all employees in a company. A simple approach perform causal analysis at a group level is to treat each group as a single unit and use the aggregated value of the response variables as the outcome.

14.4 Assumptions of randomized experiments

In this section, we discuss several assumptions in the random designs for effective causal analysis.

14.4.1 Ignorability

The first assumption is *ignorability*, which differs by the random designs. We will state a version of this assumption for each design mentioned in the previous section.

Completely randomized design

that the distribution of each potential outcome is independent of the treatment assignment. This can be written formally as

$$z \perp y^0, y^1. \tag{14.1}$$

In our running example, this says that a participant with low blood pressure after one year is equally likely to be from control or treatment group.

The ignorability assumption implies that the difference in means is unbiased. To see this, we compute the expectations.

$$\begin{aligned}\mathbb{E}[y|z=1] &= \mathbb{E}[y^1|z=1] = \mathbb{E}[y^1] \\ \mathbb{E}[y|z=0] &= \mathbb{E}[y^0|z=0] = \mathbb{E}[y^0].\end{aligned}$$

In each line, the first equivalence follows from the fact that the potential outcome is observed for the corresponding treatment assignment, and the second equivalence follows from the ignorability. Consequently,

$$\mathbb{E}[y|z=1] - \mathbb{E}[y|z=0] = \mathbb{E}[y^1] - \mathbb{E}[y^0] = \tau_{\text{SATE}}.$$

If the data is also a random sample from the population, we can replace the right-hand side by τ_{PATE}. This agrees with the data in Table 14.2 and Table 14.3 that the difference in means is biased when the ignorability assumption is violated.

Randomized blocks experiments

Let b be the blocking variable. The ignorability assumption for the random block designs is:

$$z \perp y^0, y^1 \mid b.$$

In other words, within each block, all units have the same probability of being assigned in the treatment group. Note that if the probability of treatment is

the same across all blocks (that is, $z \perp b$), then Equation 14.1 is satisfied, and the difference in means in an unbiased estimator of SATE (or PATE).

Matched pairs experiments

This a special case of randomized block experiments with two units in each block. By the definition of a matched pairs experiment, every unit has the same probability (0.5) of being assigned the treatment; so Equation 14.1 is satisfied, and the difference in means in an unbiased estimator of SATE (or PATE).

14.4.2 Stable unit treatment value assumption (SUTVA)

The *stable unit treatment value assumption* (SUTVA) is simply

$$y_i^{z_i} = y_i^{z_i'} \quad \text{if } z_i = z_i'.$$

In other words, the potential outcome of unit i only depends only on the treatment, and nothing else. This assumption has several implications. First, it implies that the outcome of a unit does not depend on the other units' treatment assignments. Without this condition, causal estimation would quickly become intractable. In our running example, if a unit's outcome is also dependent on the other units' treatments, then there would be $2^8 = 256$ different combination of treatment assignments to 8 people. An we clearly do not have enough data to consider these 256 possibilities.

Here are some examples of SUTVA violations.

- In a study of effect of a new fertilizer, each of adjacent plots is randomly assigned to receive or not receive the fertilizer. However, the fertilizer from a treated plot might leak into a controlled plot, violating the SUTVA assumption.
- Vaccines of a contagious disease, randomly administered to people in a community could result in unvaccinated people having lower chance to contract the disease.
- A study that offered families from the same housing complex to move to a better neighborhood. However, a family accepting the offer and moving out might affect (positively or negatively) another family that did not receive the offer.

If we would like to perform an experiment in which SUTVA most likely does not hold due to unit "interference" as the examples show, one solution is to assign the treatment at a group level. For example, consider a study whose goal is to introduce a new technique to encourage physical activities among students. Suppose that the technique had been randomly assigned to a few students and turned out to be effective. This would improve physical activities of not only assigned students, which in turn improve those of non-assigned students as well. Thus it makes more sense to study the effect technique at the school level instead of individual level.

14.5 Some difficulties in causal inference

We address some concerns that are usualy present in causal studies.

- The ability to recover SATE (such as that of completely randomized experiments) is referred to as *internal validity*. And the extent to which the result of the study can be generalized to the population is referred to as *external validity*. Sometimes, it is difficult for an experiment to have external validity, so one has to adjust estimates of treatment effect to the population.
- The experiment can affect the behaviors of the participants. Participants in a study of effect of light on productivity are likely to be more productive during the experiment because they know they were being observed. Possible solutions include not revealing the goal of the experiment to the participants, and not telling them whether they are in the control or treatment group.
- Missing pre-treatment data is usually not fatal as they are independent of the treatment assignments. Missing outcome data, however, is very common the in control group since those in the control group are less likely to be emotionally engaged in the study. In this case, the ignorability assumption is destroyed since the missingness depends on the treatment assignments.
- Participants might not comply with the treatment assignment. In our running example, a participant who was assigned treatment might forget to take the supplement, or decide to stop taking it after a while. With such noncompliance, can make our estimate completely invalid.

Chapter 15

Causal inference with regression

Let us review all variables that arise in a randomized experiment.

- A *unit*, denoted by i, refers to an individual person or object in a random sample.
- Covariates x_i are pre-treatment measurements. These are not required for causal inference, but can be used to adjust for pre-treatment differences between the treatment and control groups.
- The treatment assignment z_i which is 1 for treated unit and 0 for controlled units.
- The potential outcomes:
  - y_i^1, the outcome if i was to receive the treatment,
  - y_i^0, the outcome if i was to receive the control.
- The observed outcomes y_i. So $y_i = y_i^1$ if $z_i = 1$ and $y_i = y_i^0$ if $z_i = 0$.

15.1 Regression for simple difference estimate

We start with causal estimation using a regression of the outcome on the treatment assignment variable.

$$y_i = a + bz_i + \varepsilon_i.$$

We use the Electric Company data, which is the data of a randomized experiment to study the effect of a new educational TV program, The Electric Company, on children's reading abilities. The experiment used the matched pairs design, where each pair consists of two classes in a grade from a school with the lowest reading scores. The students in these classes were assigned to take

a pre-test at the beginning of the school year and a post-test at the end. The data is contained in `electric.csv`.

```
library(bayesplot)
library(rstanarm)
```

Let us take a look at the data first.

```
electric <- read.csv("data/electric.csv")

head(electric)
```

```
  X post_test pre_test grade treatment supp pair_id
1 1      48.9     13.8     1         1    1       1
2 2      70.5     16.5     1         1    0       2
3 3      89.7     18.5     1         1    1       3
4 4      44.2      8.8     1         1    0       4
5 5      77.5     15.3     1         1    1       5
6 6      84.7     15.0     1         1    0       6
```

To see the effects of the TV program on the reading abilities, we plot the histograms of the post-test scores for both treatment and control groups. Here, the vertical lines are the averages.

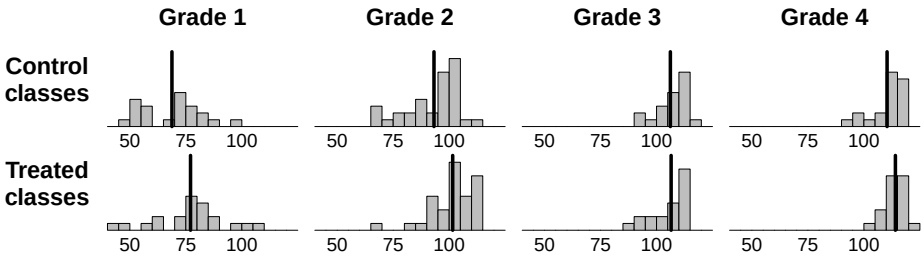


Figure 15.1: The histograms of the post-test scores

We observe that the effect of the program varies across the years. We fit the regression of the post-test on the treatment indicator.

```
fit_1 <- stan_glm(post_test ~ treatment, data=electric,
                  refresh=0)

print(fit_1)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      post_test ~ treatment
```

```
 observations: 192
 predictors:   2
------
            Median MAD_SD
(Intercept) 94.3    1.7
treatment    5.7    2.5

Auxiliary parameter(s):
      Median MAD_SD
sigma 17.6    0.9

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

The estimate is 5.7 with 2.5 standard deviation. Since Figure 15.1 indicates that the effect of the TV program varies across the grades, so we might consider fitting separate regression models by the grades. Foe each grade, we store the simulations of the difference estimate (the coefficient of `treatment`) in a matrix named `fit_2`.

```
fit_2 <- array(NA, c(4000, 4))
colnames(fit_2) <- c("Grade 1", "Grade 2",
                     "Grade 3", "Grade 4")
for (k in 1:4) {
  model <- stan_glm(post_test ~ treatment,
                    data=electric,
                    subset=(grade==k),
                    refresh=0)
  fit_2[, k] <- as.matrix(model)[, 'treatment']
}
```

From these simulations, we can plot the 50% and 95% uncertainty intervals of the difference estimate for each grade.

```
mcmc_intervals(fit_2)
```

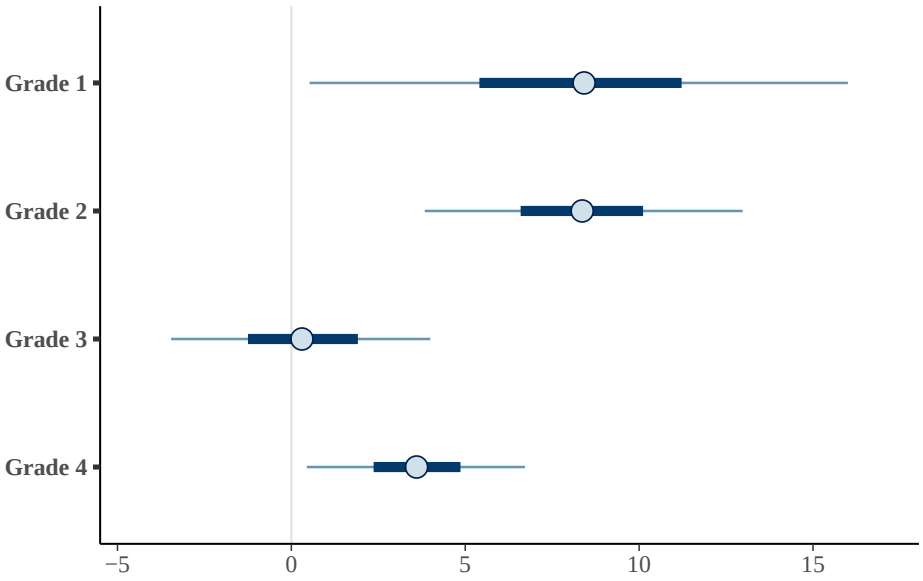


Figure 15.2: The point estimate of per-grade SATE, adjusted for the pre-test scores with uncertainties

The result agrees with Figure 15.1 that the program has a larger effect on lower grades. We also notice that the estimates for higher grades have lower standard errors.

15.2 Adding pre-treatment covariates to the model

We also have the pre-test scores as a covariate, which is highly correlated to the post-test scores. We need to adjust for this covariate, otherwise our estimate would be inaccurate due to the difference in the pre-test scores between the control group and the treatment group.

For each grade, we fit regression models with the pre-test scores x_i:

$$y = \beta_0 + \beta_1 z + \beta_2 x + \varepsilon.$$

Assume that the assignment is completely randomized, so it is independent of the pre-treatment covariate; This implies conditional ignorability: $z \perp y^0, y^1 | x$. Consequently,

$$\mathbb{E}[y|z=1,x] = \mathbb{E}[y^1|z=1,x] = \mathbb{E}[y^1|x]$$
$$\mathbb{E}[y|z=0,x] = \mathbb{E}[y^0|z=0,x] = \mathbb{E}[y^0|x].$$

This implies

$$\begin{aligned}
\tau_{\text{SATE}} &= \mathbb{E}[y^1 - y^0] \\
&= \mathbb{E}_x\mathbb{E}[y^1|x] - \mathbb{E}_x\mathbb{E}[y^1|x] \\
&= \mathbb{E}_x\mathbb{E}[y|z=1,x] - \mathbb{E}_x\mathbb{E}[y|z=0,x] \\
&= \mathbb{E}_x[(\beta_0 + \beta_1 + \beta_2 x) - (\beta_0 + \beta_2 x)] \\
&= \beta_1.
\end{aligned}$$

Since the coefficient obtained from fitting the model, say $\widehat{\beta}_1$, is an unbiased estimator of β_1, so it is an unbiased estimator of the average causal effect τ_{SATE} as well.

Let us estimate the average causal effect by fitting the regression model in R.

```
fit_3 <- array(NA, c(4000, 4))
colnames(fit_3) <- c("Grade 1", "Grade 2",
                     "Grade 3", "Grade 4")
for (k in 1:4) {
  model <- stan_glm(post_test ~ treatment + pre_test,
                    data=electric,
                    subset=(grade==k),
                    refresh=0)
  fit_3[, k] <- as.matrix(model)[, 'treatment']
}

mcmc_intervals(fit_3)
```

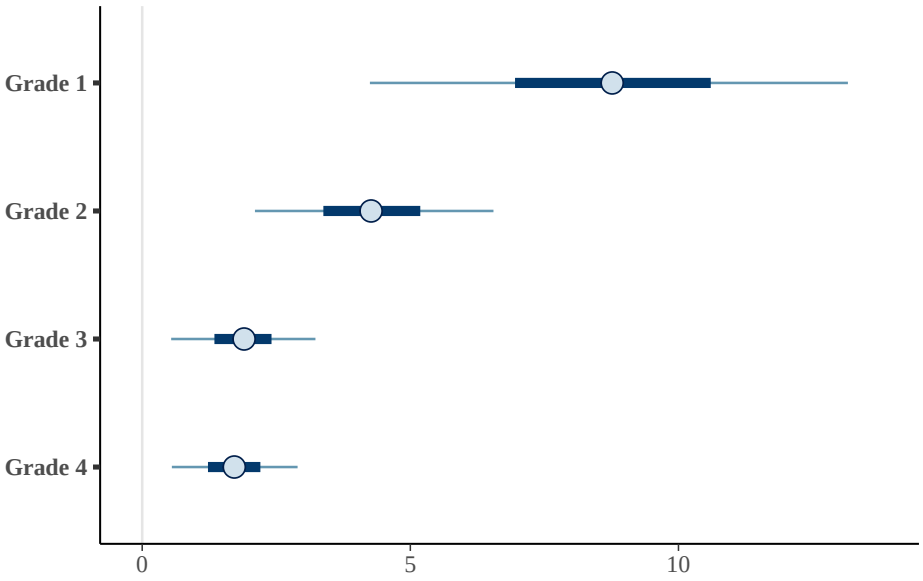


Figure 15.3: The point estimate of per-grade SATE, adjusted for the pre-test scores with uncertainties

With the pre-test scores, the standard errors of the estimated treatment effects become smaller.

15.2.1 Regression with interactions

We can add the interaction between the treatment and covariates to our model as well. To illustrate this, we take the data of grade 4 students in the Electric Company study, and fit a regression with an interaction between the treatment and the pre-test score.

```
fit_4 <- stan_glm(post_test ~ treatment + pre_test
                  + treatment:pre_test,
                  data=electric,
                  subset=(grade==4),
                  refresh=0)

print(fit_4)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      post_test ~ treatment + pre_test + treatment:pre_test
 observations: 42
 predictors:   4
```

```
 subset:       (grade == 4)
------
                   Median MAD_SD
(Intercept)        38.8    4.9
treatment          14.3    9.0
pre_test            0.7    0.0
treatment:pre_test -0.1    0.1

Auxiliary parameter(s):
      Median MAD_SD
sigma 2.2    0.3

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

The fitted model is the following:

$$\begin{aligned} y &= 38.6 + 14.6z + 0.7x - 0.1zx + \varepsilon \\ &= 38.6 + (14.6 - 0.1x)z + 0.7x + \varepsilon. \end{aligned}$$

Our estimate of the treatment effect is the coefficient of z, which is $14.6 - 0.1x$. This allows us to have a covariate-dependent causal estimate. For the grade 4 students, the minimum and maximum pre-test scores are 80 and 120, respectively. So the treatment effect varies from $14.6 - 0.1 * 120 = 2.6$ to $14.6 - 0.1 * 80 = 6.6$. This example illustrates that **including the interaction allows us to see how the treatment effect varies with the covariates**. We can also estimate SATE (and PATE) by computing the averate over the causal estimates of all units in the sample.

$$\hat{\tau}_{\text{SATE}} = \frac{1}{n}\sum_{i=1}^{n}(14.6 - 0.1x_i) = 14.6 - 0.1\bar{x},$$

which can be computed in R as follows:

```
sims <- as.matrix(fit_4)
n_sims <- nrow(sims)
is_grade_4 <- (electric$grade == 4)

pretest_4 <- electric$pre_test[is_grade_4]
mean_pretest_4 <- mean(pretest_4)

avg_effect <- sims[, 2] + sims[, 4]*mean_pretest_4
```

```
print(avg_effect[1:10])
```

```
 [1] 0.6597359 1.4787528 1.4710521 1.5079603 1.2501985 1.3890738 2.7665281
 [8] 0.6622269 2.8731758 1.1845089
```

With this, we can compute the point estimate and the uncertainty.

```
print(c(median(avg_effect), mad(avg_effect)))
```

```
[1] 1.7747740 0.6423454
```

We note that this is similar to the estimate of SATE shown in Figure 15.3. It also holds in general that the average of the causal estimates with an interaction is the same as the estimate of the average causal effect without the interaction.

This way of obtaining a causal estimate can also be extended to two interactions and above. For example, suppose we have two covariates x_1, x_2 and interactions zx_1 and zx_2. The regression model is

$$\begin{aligned} y &= \beta_0 + \beta_1 z + \beta_2 x_1 + \beta_3 x_2 + \beta_4 z x_1 + \beta_5 z x_2 + \varepsilon \\ &= \beta_0 + (\beta_1 + \beta_4 x_1 + \beta_5 x_2) z + \beta_2 x_1 + \beta_3 x_2 + \varepsilon. \end{aligned}$$

Thus, the estimated causal effect at covariate level x_1 and x_2 is $\beta_1 + \beta_4 x_1 + \beta_5 x_2$ and an estimate of SATE is $\beta_1 + \beta_4 \bar{x}_1 + \beta_5 \bar{x}_2$.

15.2.2 Do not add post-treatment covariates to the regression

In general, one should not adjust for post-treatment covariates (sometimes referred to as *mediator*). Here, *adjusting for* a variable means adding it as a regression input. To see why, we consider the following example of a study of the effect of child care services on the child's intelligence.

- y is the child's IQ score.
- z is the treatment assignment.
- x is a pre-treatment covariate that indicates whether both parents have a high school education.
- q is a post-treatment covariate that measures parenting quality.

Suppose that the relationship between these variables follow the linear model:

$$y = \beta_0 + \beta_1 z + \beta_2 x \beta_3 q + \varepsilon. \tag{15.1}$$

However, in most cases, the post-treatment covariate is not independent of the treatment assignment. For example, we could have the following relationship between the potential outcomes q and z.

$$q = 1 + 0.2z + \varepsilon'.$$

Assume further that the relationship between q and z follow SUTVA, which implies that ε' for $z = 0$ and $z = 1$. Let us compare two families with the same parenting quality, same high school education, but with different treatments:

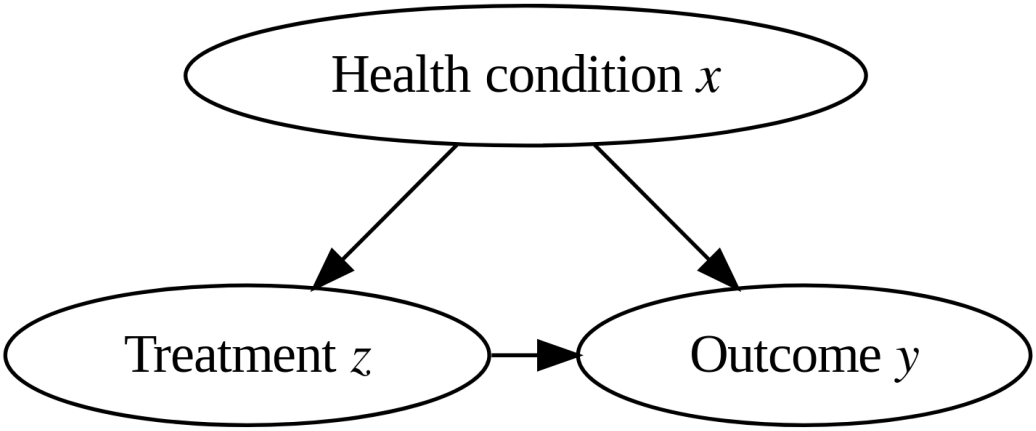
$$\begin{aligned}
&\text{Family 1: } z_1 = 0, q_1 = 1 \\
&\quad \Rightarrow \quad q_1^0 = 1, q_1^1 = 1.2, \varepsilon_1' = 0 \\
&\text{Family 2: } z_2 = 1, q_2 = 1 \\
&\quad \Rightarrow \quad q_2^0 = 0.8, q_2^1 = 2, \varepsilon' = -0.2.
\end{aligned}$$

Thus, we are actually comparing two families with different parenting skills: q_1^0 and q_2^0. This example illustrates that, among families with the same parenting quality (q) and high school education (x), there can be some variation in the parenting skills (q^0). However, to estimate the average causal effect, we would like to control q^0, the parenting skills, as it is more related to the child's IQ before the treatment. This suggests that the coefficient β_1 in Equation 15.1 would not be an appropriate estimate of the average causal effect.

Chapter 16

Causal inference with observational data

Observational data refers to data obtained from observing an event of interest; for example, data of outcomes of a costly treatment, which is only applied to patients with extremely poor health conditions. In this setting, we could have a covariate that affects both the treatment and the outcome; such covariate is usually called **confounding covariate** or **confounder**.



Here, "Health condition" is a confounder. Since the data does not come from a randomized experiment, the difference-of-means is unsuited for estimating the average causal effect; we can imagine that the average outcome among the treated patients (the patients with poor health conditions) must be lower than the average of the treated outcomes y^1, and that among the controlled patients

(the patients with poor health conditions) must be higher than the average of the controlled outcomes y^0.

16.1 Assumption in an observational study

Even when the data is not obtained from a randomized experiment, as long as the ignorability assumption is satisfied, we can turn the causal estimation problem into a linear regression problem. In an observational study, we must make sure that the variables satisfy the ignorability assumption:

$$y^0, y^1 \perp z|x,$$

which is similar to the assumption for a randomized block experiment. The difference is that, in an obversational study, the assumption is not implied by the design of an experiment, but by our prior knowledge of the relationship among the variables. If the ignorability assumption holds, the average causal effect can be estimated using the coefficient β_1 of the treatment assignment in the regression model:

$$y = \beta_0 + \beta_1 z + \beta_2 x + \varepsilon.$$

The proof of this can be found in Section 15.2.

So far, we have discussed causal estimation when there is only a single *observed* confounder. In general, the causal effect can be estimated if confounders are all observed. If not all confounders are observed, then we might risk introducing some bias in our estimate.

16.1.1 Omitted variable bias

We can quantify the bias from omitting the confounder x when the relationship between the variables can be described with a linear regression model:

$$y = \beta_0 + \beta_1 z + \beta_2 x + \varepsilon. \tag{16.1}$$

Suppose that we did not aware of a potential confounder x and fit a misspecified model:

$$y = \beta_0' + \beta_1' z + \varepsilon', \tag{16.2}$$

where β' and β' is another set of coefficients. To measure the biased introduce from using this model, we fit a regression of the confounder x on the treatment z.

$$x = \gamma_0 + \gamma_1 z + \varepsilon''. \tag{16.3}$$

Substituting Equation 16.3 back into Equation 16.1 yields

$$y = \beta_0 + \beta_2\gamma_0 + (\beta_1 + \beta_2\gamma_1)z + \varepsilon + \beta_2\varepsilon''. \tag{16.4}$$

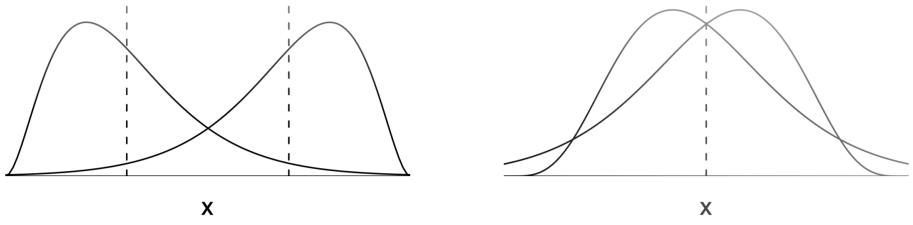
Equating the coefficient of z in Equation 16.2 and Equation 16.4 yields

$$\beta_1' = \beta_1 + \beta_2\gamma_1.$$

We can see that our causal estimate β_1' is biased. This also implies that, if a covariate x is not associated with the treatment ($\gamma_1 = 0$) or if the covariate is not associated with the outcome ($\beta_2 = 0$).

16.1.2 Imbalance of confounder distributions

An observed confounder is **imbalanced** when the distribution of the confounder for the treatment group differs from that of the control group. Examples of imbalanced confounder x are shown in the following plots:



In the left plot, the distributions of x for the control and treatment groups have different means; while in the right plot, assuming that the mean is non-zero, the distributions would have different second moments.

Causal estimation with imbalanced confounder distribution would force us to rely more on the correctness of our model. For example, suppose that the true model of the population is:

$$\begin{aligned} \text{Treatment: } & y = \beta_0 + \beta_1 x + \beta_2 x^2 + \theta + \varepsilon \\ \text{Control: } & y = \beta_0 + \beta_1 x + \beta_2 x^2 + \varepsilon. \end{aligned}$$

Consequently, the causal effect θ can be estimated by taking the averages of both equations, which yields

$$\theta = \bar{y}_1 - \bar{y}_0 - \beta_1(\bar{x}_1 - \bar{x}_0) - \beta_2(\bar{x_1^2} - \bar{x_0^2}),$$

where $\bar{y}_1, \bar{x}_1, \bar{x_1^2}$ are the averages of the treatment group and $\bar{y}_0, \bar{x}_0, \bar{x_0^2}$ are the averages of the control group.

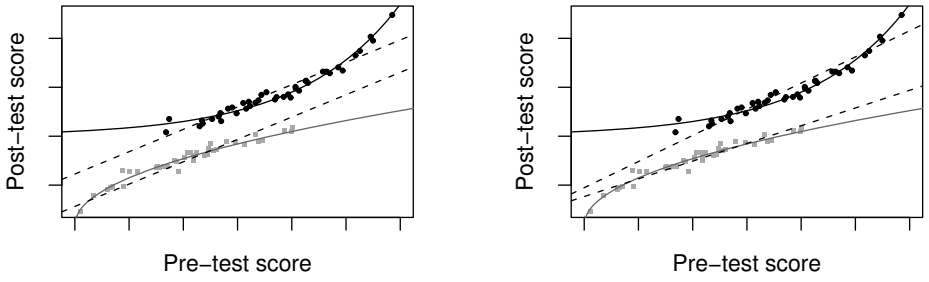
Suppose that we wanted to keep it simple and estimate the causal effect using the difference between the means:

$$\theta' = \bar{y}_1 - \bar{y}_0,$$

Then, our estimate would be off the true estimate by $\beta_1(\bar{x}_1 - \bar{x}_0) + \beta_2(\bar{x_1^2} - \bar{x_0^2})$. This bias would be small if the confounder distributions for the treatment and the control groups are almost identical, which implies $\bar{x}_1 \approx \bar{x}_0$ and $\bar{x_1^2} \approx \bar{x_0^2}$. On the other hand, if the distributions are vastly different, then hte bias would become large.

16.1.3 Lack of complete overlap

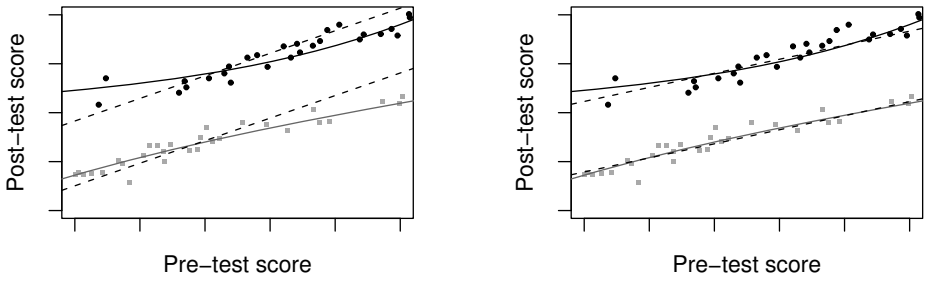
Overlap (or *common support*) is the intersection of the ranges of the confounder data for the treatment and control groups. We say that the distributions have *complete overlap* if their ranges coincide. Lack of complete overlap in the confounders leads to causal estimation problem, because for some observed values of the confounder, we have no information on the counterfactual outcomes. The plots below show examples the Electric Company data, which has the pre-test score as a confounder. Here, the solid curve are the true confounder distributions for the treatment group (black dots) and the control group (gray dots). The dashed lines in the left plot are regression lines of the post-test scores on the treatment and the pre-test score, while the dashed lines on the right also allow for an interaction between the two predictors. The causal effect at any level of pre-test score is simply the vertical distance between the two solid lines.



As the confounder distributions for the treatment and control groups do not completely overlap, our causal estimate (the vertical distance between the dashed lines) totally underestimates the true average treatment effect (the vertical distance between the solid lines).

Nonetheless, it is still possible to estimate the treatment effect in the region where the confounder is observed for both groups. As shown in the plots below, by restricting our analysis to this region and fitting a linear regression (without

or with an interaction) as before, we obtain an estimate of treatment effect that is very accurate in this region.



16.2 The Electric Company example

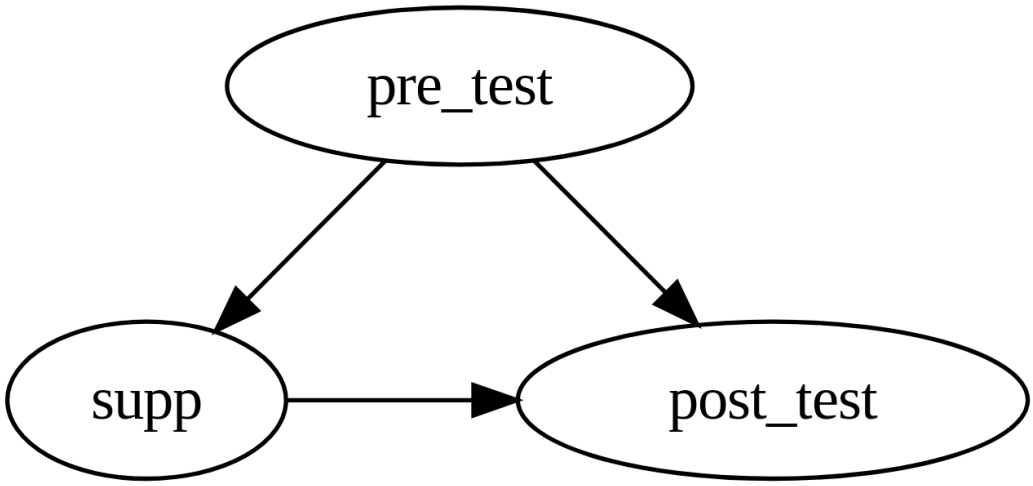
The Electric Company data that we used in the previous chapter, in fact, has an additional covariate: the teacher for each class in the treatment group had the choice of *replacing* or *supplementing* the current regular reading program by the TV program; the choice is indicated by the covariate `supp` (0, 1, or NA for every controlled class).

```
library(bayesplot)
library(rstanarm)
```

```
electric <- read.csv("data/electric.csv")

head(electric)
```

```
  X post_test pre_test grade treatment supp pair_id
1 1      48.9     13.8     1         1    1       1
2 2      70.5     16.5     1         1    0       2
3 3      89.7     18.5     1         1    1       3
4 4      44.2      8.8     1         1    0       4
5 5      77.5     15.3     1         1    1       5
6 6      84.7     15.0     1         1    0       6
```

Suppose that we would like to estimate the causal effect of the supplement versus the replacement among the classes that were assigned to watch the TV program. Assuming that the pre-test score also affects the choice of supplement (this is just for demonstration, as there can be many factors that affect the choice of supplement), the relationship between the variables is illustrated by the following graphical model:



As `post_test` is affected by `pre_test`, we must adjust for the covariate in our linear regression.

```
fit_supp <- array(NA, c(4000, 4))
colnames(fit_supp) <- c("Grade 1", "Grade 2",
                        "Grade 3", "Grade 4")
for (k in 1:4) {
  model <- stan_glm(post_test ~ supp + pre_test,
                    data=electric,
                    subset=(grade==k) & (!is.na(supp)),
                    refresh=0)
  fit_supp[, k] <- as.matrix(model)[, 'supp']
}

mcmc_intervals(fit_supp)
```

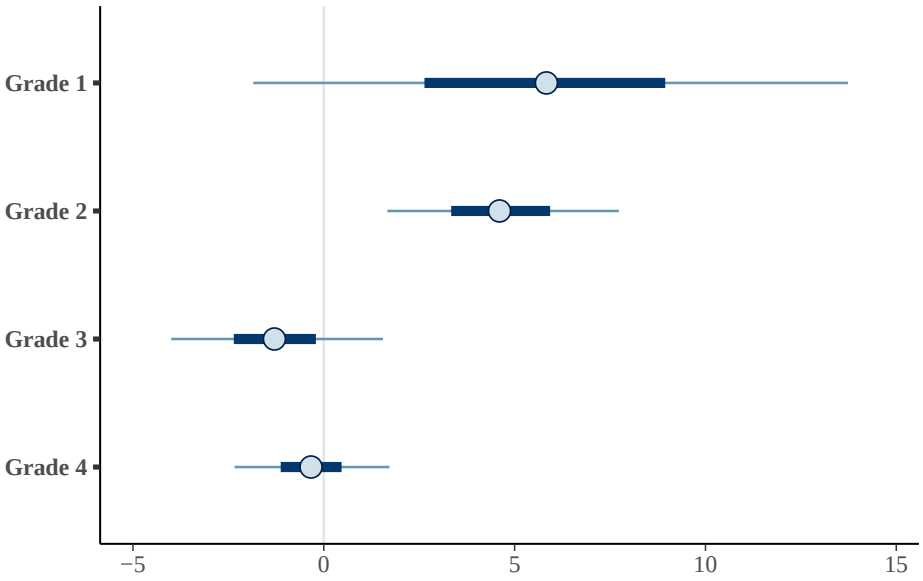


Figure 16.1: The point estimate of per-grade SATE with uncertainties, adjusted for the pre-test scores and the supplement indicators

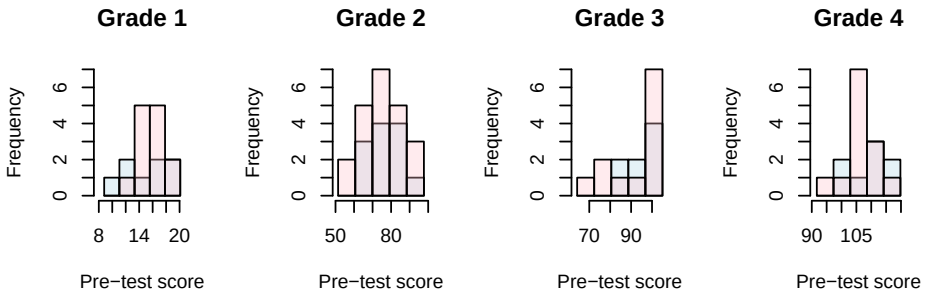
We conclude from the plot that supplementing is more effective than replacing the TV program in lower grades.

16.2.1 Examining overlap of the confounder distribution

We can plot histograms of the confounder (the pre-test score) for the treatment and control groups. In each plot, the pink histogram is that of the treatment group, and the blue histograms is that of the control group.

```
blue <- rgb(173,216,230, max=255,
          alpha=80, names="lt.blue")
pink <- rgb(255,192,203, max=255,
          alpha=80, names="lt.pink")

par(mfrow=c(1,4))
for (k in 1:4){
    grade_k_data <- electric$pre_test[electric$grade==k &
                                      !is.na(electric$supp)]
    min_score <- min(grade_k_data)
    max_score <- max(grade_k_data)
    hist(grade_k_data[electric$supp==0],
         breaks=seq(min_score, max_score, length.out=6),
         xlim=c(min_score-1, max_score+1),
```

```
          ylim=c(0, 7),
          main=paste("Grade", k), col=blue,
          xlab="Pre-test score",
          freq=TRUE)
    hist(grade_k_data[electric$supp==1],
          breaks=seq(min_score, max_score, length.out=6),
          col=pink, freq=TRUE, add=TRUE)
}
```



We clearly see the imbalance between the treatment and control groups in Grade 1 and Grade 4, and there is lack of complete overlap in Grade 3. In particular, there are some classes in Grade 3 that supplemented the TV program and their average pre-test scores are lower than those that replaced the regular reading program with the TV program. We should keep these observations in mind when assessing the accuracy of our causal estimates.

Chapter 17

Subclassification and propensity score matching

17.1 Subclassification

When the confounder is a discrete variable, *subclassification* is an easy way to estimate the average causal effect. We demonstrate this on an example of data more than 4000 children born in the 1980s. Some of the children were received high-quality child care from the Infant Health and Development Program (IHDP). We want to measure the effect of the child care on the cognitive abilities, evaluated with an IQ-like test at age 3. The data is contained in `cc2.csv`. Below, we show some of the attributes, namely age in months (`age`), body weight (`bw`), mother's education (`educ`), treatment (`treat`) and the IQ score at age 3 (`ppvtr.36`).

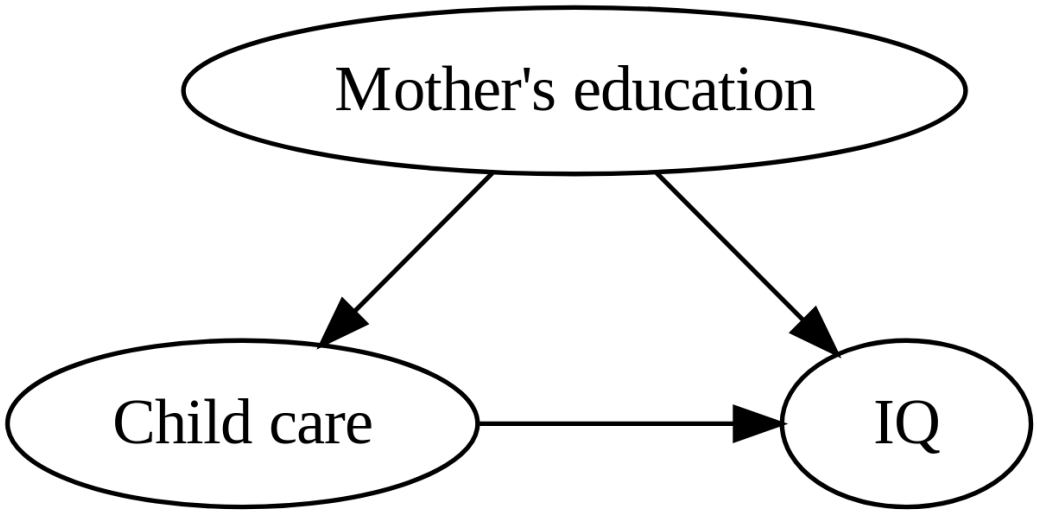
```
set.seed(0)
library(rstanarm)
library(survey)
```

```
cc2 <- read.csv("data/cc2.csv")

head(cc2[, c('age', 'bw', 'educ', 'treat', 'ppvtr.36')])
```

```
       age   bw educ treat ppvtr.36
1 60.79671 1559    4     1      111
2 59.77823 2240    1     1       81
3 59.51540 1900    1     1       92
4 59.18686 1550    4     1      103
```

```
5 58.79261 2270    1      1        81
6 58.49692 1550    2      1        94
```

To make it simple, we assume that the only confounder is the mother's education. This is the case when the program specifically targetted those families who received only basic or no education.



The `Childcare` data recorded four levels of mother's education: not a high school graduate (`lths`), a high school graduate (`hs`), at some college (`ltcoll`), and a college gradute (`college`)

```
head(cc2[, c("lths", "hs", "ltcoll", "college")])
```

```
  lths hs ltcoll college
1    0  0      0       1
2    1  0      0       0
3    1  0      0       0
4    0  0      0       1
5    1  0      0       0
6    0  1      0       0
```

In this case, the conditional treatment effect (CATE) for each level of mother's education is the difference of means within the level.

```
educ_list <- c("lths", "hs", "ltcoll", "college")

df <- data.frame(matrix(nrow = 4, ncol = 5))
```

```
colnames(df) <- c("Mother\'s education", "Treatment effect", "Standard error", "#treated"
df$"Mother\'s education" <- educ_list

for (k in 1:4){
    edu <- educ_list[k]

    iq_treat <- cc2$ppvtr.36[cc2$treat==1 & cc2[edu]==1]
    iq_contr <- cc2$ppvtr.36[cc2$treat==0 & cc2[edu]==1]

    n_treat <- length(iq_treat)
    n_contr <- length(iq_contr)

    tr_effect <- mean(iq_treat) - mean(iq_contr)
    tr_se <- sqrt(var(iq_treat)/n_treat
                  +var(iq_contr)/n_contr)

    df[k, "Treatment effect"] <- tr_effect
    df[k, "Standard error"] <- tr_se
    df[k, "#treated"] <- n_treat
    df[k, "#controls"] <- n_contr
}

df
```

```
  Mother's education Treatment effect Standard error #treated #controls
1               lths         9.298590       1.461121      126      1232
2                 hs         4.057315       1.873075       82      1738
3             ltcoll         7.871995       2.402038       48       789
4            college         4.622168       2.322062       34       332
```

Notice that most mothers in this dataset did not finish high school, which should be reflected when we combine the mother's education-specific estimates to obtain the average treatment effect. To this end, we calculate a weighted average of the estimates, with weights defined by the number of children in each group; such estimation method is referred to as *subclassification*.

$$\hat{\tau}_{\text{ATE}} = \frac{9.3 * 1358 + 4.1 * 1820 + 7.9 * 837 + 4.6 * 366}{1358 + 1820 + 837 + 366} = 6.5,$$

and the standard error is $\sqrt{\frac{1.5^2 * 1358^2 + 1.9^2 * 1820^2 + 2.4^2 * 837^2 + 2.3^2 * 366^2}{(1358+1820+837+366)^2}} = 1.04$.

17.1.1 Average effect of treatment on the treated

In observational data, sometimes we only care about the treatment effect on the treated group. For example, we would like to measure the treatment effect

on the children that were eligible for the child care program, which are those in the treatment group. In this case, we would like to measure the *average effect of treatment on the treated* (ATT). We can also use subclassification to estimate ATT, only counting the children in the treatment group.

$$\hat{\tau}_{\text{ATT}} = \frac{9.3 * 126 + 4.1 * 82 + 7.9 * 48 + 4.6 * 34}{126 + 82 + 48 + 34} = 7.0,$$

and the standard error is $\sqrt{\frac{1.5^2 * 126^2 + 1.9^2 * 82^2 + 2.4^2 * 48^2 + 2.3^2 * 34^2}{(126+82+48+34)^2}} = 0.9$.

So there is no visible difference between the estimates of the average treatment over all children (ATE) and the average treatment over the treated children (ATT) in this case.

17.2 Propensity score matching

Matching refers to any method of transforming the original data to make it look like a sample from a randomized experiment, from which we can estimate causal effects with little bias, even when our model is misspecified.

We will go over a specific type of matching, called *propensity score matching*. Essentially, we use a logistic regression to compute a "score" for each unit; these scores will be used to "match" between treated and untreated units.

We shall detail the propensity score matching as a five-step process below.

17.2.1 Step 1: Choose the confounders and estimand

Choosing confounders. Researchers often choose a list of confounders based on previous literature. In the child care example, there are not many covariates, so we decide to include them all.

Choosing estimands. We have talked about two estimands: the average treatment effect (ATE) and the average treatment effect on the treated (ATT). We will also occasionally mention the average treatment effect on the controlled (ATC). The choice of estimand depends on the purpose of the estimation. In the child care example, we want to evaluate how the child care program affects the children that are eligible for the program, so we choose to estimate ATT.

To estimate ATT, we will keep the treatment group the same, while transforming the control group to look like the treatment group. If we were to estimate the effect of the treatment on the control, we would transform the treatment group to match the control group instead.

17.2.2 Step 2: Estimate the propensity score

In this step, we fit a logistic model to estimate the probability of a child receiving the treatment, conditioning on the covariates.

```
ps_fit_1 <- stan_glm(treat ~ bw + bwg + hispanic + black
                     + b.marr + lths + hs + ltcoll
                     + work.dur + prenatal + sex + first
                     + preterm + momage + dayskidh
                     + income,
                     family=binomial(link='logit'),
                     data=cc2,
                     algorithm='optimizing', refresh=0)
```

We then use the fitted model to calculate the *propensity score* of each child, that is, the predicted linear function for each child in the treatment group (the reason we use the linear function instead of the predicted probability is because it is easier to compare the distribution before and after the matching; see Step 4 below). Each score plays a role of summarizing the covariates of each unit as a single number.

```
pscores <- predict(ps_fit_1)

print(pscores[1:10])
```

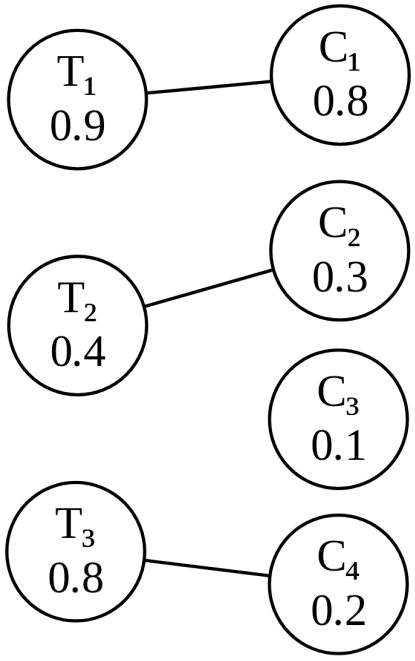
```
          1           2           3           4           5           6
4.262056670 -0.210344313  0.554438239  2.917770699  0.009094453  1.744454213
          7           8           9          10
1.576030355 -2.535481435 -0.702373486  0.532513542
```

17.2.3 Step 3: Match controlled units to the treated units

Our goal now is to transform the control group so that its distribution of the propensity scores matches that of the treatment group. The transformation that we use here is simply choosing, for each treated unit T_i , the controlled unit that has the closest propensity score to that of T_i. We can either match with or without replacement.

Matching without replacement

In matching without replacement for ATT estimation, we match each treated unit with the closest controlled unit that has not been matched yet. The diagram below shows a hypothetical example of matching without replacement; the numbers shown are the propensity scores from the logistic regression.



When matching for ATC estimation, we swtich the roles between the treated units and the controlled units.

We can match a controlled unit to each treated unit using the provided `matching` function.

```
source("library/matching.R")
```

```
matches <- matching(z=cc2$treat, score=pscores, replace=FALSE)

print(summary(matches))
```

```
          Length Class  Mode
match.ind 4381   -none- numeric
cnts      4381   -none- numeric
pairs     4381   -none- numeric
```

```
print(matches$match.ind[1:10])
```

```
 [1] 1150 2899  730 2913 1455 2433 4131 1406 2689 3910
```

Here, the number in the i-th row is the index of the unit that has been matched

with the i-th unit. Since we are matching without replacement, the number of matched controlled units has to be the same as the number of units in the treatment group, which is 290 in this example. Thus, the total number of matched units is $290 * 2 = 580$.

```
matched <- cc2[matches$match.ind,]

nrow(matched)
```

```
[1] 580
```

With this new dataset, we will have to check that the covariates between the treatment and control groups are balanced, and the propensity scores in both groups sufficiently overlap; this is detailed in Step 4. But before that let us have a glimpse of what is going to happen in the final step: we will run a linear regression on the matched dataset and use the coefficient of the treatment assignment to estimate ATT.

Code example for fitting a weighted regression model is shown in Listing 17.1 below.

Listing 17.1 Regression for matching without replacement

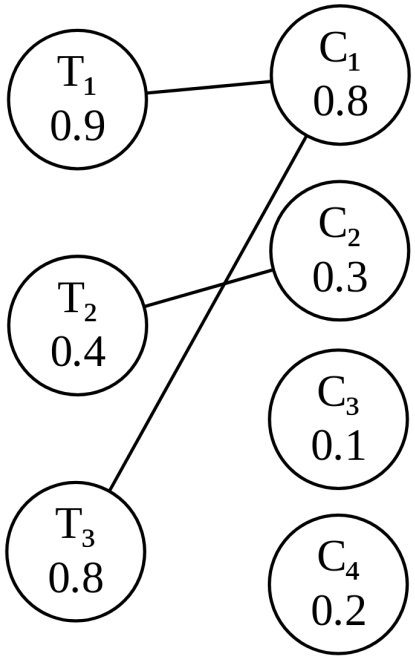
```
reg_ps <- stan_glm(ppvtr.36 ~ treat + bw + bwg + hispanic
                   + black + b.marr + lths + hs + ltcoll
                   + work.dur + prenatal + sex + first
                   + preterm + momage + dayskidh + income,
                   data=cc2[matches$match.ind,],
                   algorithm='optimizing')

summary(reg_ps)['treat', 1:2]
```

```
   Median    MAD_SD
10.426381  1.520227
```

Note that fitting a regression model on a matched data without replacement is the same as fitting on a weighted data, with weight equals 1 for each unit that has been matched, and weight equals 0 for each unit that has not been matched. This kind of "weighted data" interpretation will be relevant for the next type of matching.

Matching with replacement

In matching with replacement for ATT estimation, we match each treated unit with the closest controlled unit that might or might not have been matched. The diagram below shows a hypothetical example of matching with replacement; the numbers shown are the propensity scores from the logistic regression.



When matching for ATC estimation, we swtich the roles between the treated units and the controlled units.

For this type of matching, we can again use the `matching` function with `replace=TRUE`.

```
matches.wr <- matching(z=cc2$treat, score=pscores, replace=TRUE)
wts.wr <- matches.wr$cnts

print(wts.wr[780:800])
```

```
 [1] 1 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Here, the i-th number in `wts.wr` is the number of times that the i-unit has been matched. In this example, the 781st unit has been matched twice, which means that we should include two copies of this unit in the restructured control group. In other words, we can treat each numbers in `wts.wr` as the weight of the corresponding unit.

To fit a linear regression model with weighted data, we may use functions provided by the `survey` library. Specifically, we use `svydesign` to specify the weigths and `svyglm` to fit the model. Here, we use `ids=~1` for data with no clusters.

Code example for fitting a weighted regression model is shown in Listing 17.2 below.

Listing 17.2 Regression for matching with replacement

```
ps_fit_1_design <- svydesign(ids=~1,
                             weights=matches.wr$cnts,
                             data=cc2)
reg_ps.wr <- svyglm(ppvtr.36 ~ treat + bw + bwg + hispanic
                    + black + b.marr + lths + hs + ltcoll
                    + work.dur + prenatal + sex + first
                    + preterm + momage + dayskidh + income,
                    design=ps_fit_1_design,
                    data=cc2)

summary(reg_ps.wr)$coef['treat', 1:2]
```

```
Estimate Std. Error
9.590492   2.015102
```

17.2.4 Step 4: Inspect balance and overlap in propensity scores

As alluded in Step 3, we will check if (1) the covariates between the treatment and control groups are balanced, and (2) the propensity scores in both groups sufficiently overlap. Here, we will use `balance` function from the provided `balance.R` to compute the difference in covariate means between treatment and control groups.

```
# Uncomment to install required package
#install.packages("Hmisc")
source("library/balance.R")
```

```
covs <- c('bw', 'preterm', 'dayskidh', 'sex', 'first',
          'age', 'black', 'hispanic', 'white', 'b.marr',
          'lths', 'hs', 'ltcoll', 'college', 'work.dur',
          'prenatal', 'momage')
bal_nr <- balance(rawdata=cc2[,covs], treat=cc2$treat,
                  matched=matches$cnts, estimand='ATT')
bal_nr.wr <- balance(rawdata=cc2[,covs], treat=cc2$treat,
                     matched=matches.wr$cnts, estimand='ATT')
```

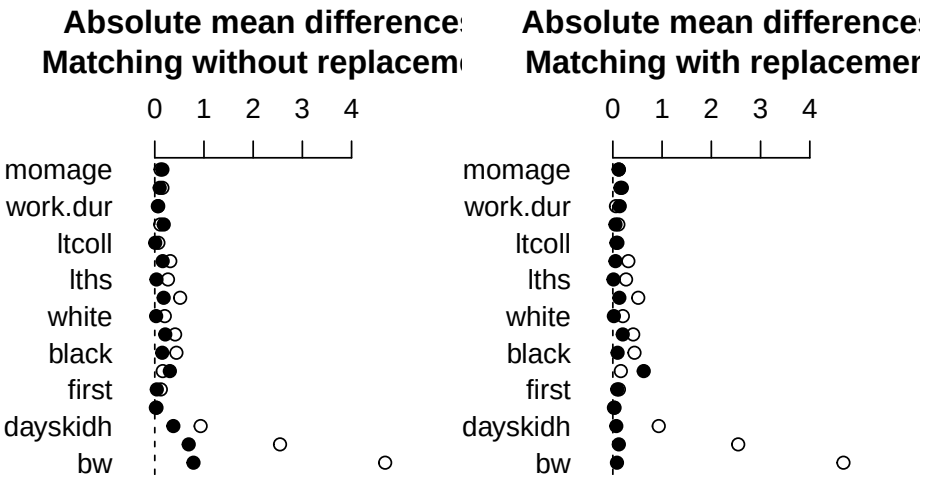
After calling `balance`, the mean differences over all covariates before the matching are stored in `output$diff.means.raw`, and those after the matching are

stored in `output$diff.means.matched`. Here, we define a function named `plot_mean_diffs` to plot these differences from a `balance`'s output.

```
plot_mean_diffs <- function(bal, title) {
    pts <- bal$diff.means.raw[,4]
    pts2 <- bal$diff.means.matched[,4]

    K <- length(pts)

    plot(c(pts,pts2), c(1:K, 1:K),
         bty='n', xlab='', ylab='',
         xaxt='n', yaxt='n', type='n',
         main=title)
    abline(v=0, lty=2)
    points(pts, 1:K, cex=1)
    points(pts2, 1:K, pch=19, cex=1)
    axis(3)
    axis(2, at=1:K, labels=covs,
         las=2, hadj=1, lty=0)
}
```

Now, let us compare the absolute mean differences obtained from matching with replacement and without replacement.

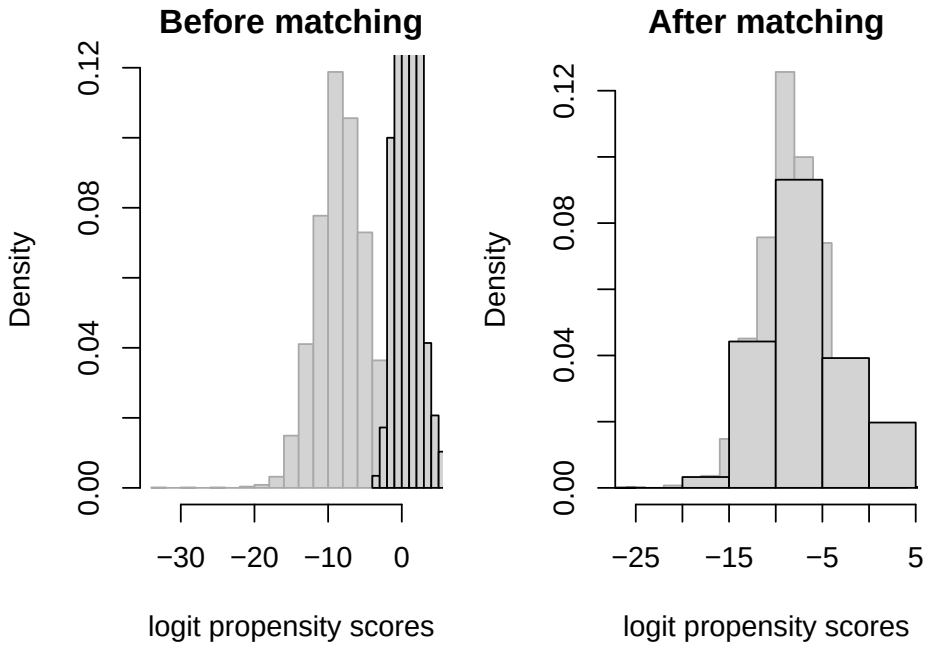
```
par(mfrow=c(1,2))

par(mar=c(1,4.3,7,2))
plot_mean_diffs(bal_nr, "Absolute mean differences\n Matching without replacement")

par(mar=c(1,4.3,7,2))
plot_mean_diffs(bal_nr.wr, "Absolute mean differences\n Matching with replacement")
```



We see that matching with replacement has smaller mean differences, which implies that it has a better balance.

To check the overlap, we inspect the histograms of the propensity scores in the treatment and control groups.

```
par(mfrow=c(1,2))
# Plot the histograms of the propensity scores before matching
par(mar=c(5,4,2,1))
hist(pscores[cc2$treat==0], main="Before matching",
     border="darkgrey", xlab="logit propensity scores",
     freq=FALSE)
hist(pscores[cc2$treat==1], freq=FALSE, add=TRUE)
# Plot the histograms of the propensity after matching
par(mar=c(5,4,2,1))
hist(pscores[cc2[matches.wr$match.ind, 'treat']==0],
     main="After matching", border="darkgrey",
     xlab="logit propensity scores", freq=FALSE)
hist(pscores[cc2[matches.wr$match.ind, 'treat']==1], freq=FALSE, add=TRUE)
```



We can see that the the propensity scores have a better overlap after matching.

17.2.5 Before step 5: Repeat steps 2-4 until a good balance is achieved

We do not want to proceed to Step 5 yet until we obtain a good balance in the covariates. There are generally two ways to achieve a better balance:

- Changing the model for the propensity score. This can be done in several ways:
  - Adding interactions between the covariates
  - Finding other potential confounders
  - Transforming existing continuous variables
- Changing the way the propensity scores are used to restructure the data, for example, to a different matching method.

17.2.6 Step 5: Fit the regression on the restructured data

As mentioned in Step 3, after we are satisfied with the balance, we can now fit the linear regression model on the restructured dataset. Example code for fitting a regression on matched data without replacement can be found in Listing 17.1, and code for matched data with replacemenet can be found in Listing 17.2. The estimate of average treatment effect is 10.4 ± 1.5 and 9.6 ± 2.0, respectively.

17.2.7 Other considerations

There are several considerations that one has to keep in mind when performing propensity score matching.

- It is not always a good idea to include all covariates in the list of potential confounders. Here is a basic guideline on which covariates to choose:
  - Do not include post-treatment covariates
  - Do not include covariates that are strongly related to the treatment but not strongly related to the outcome. An example of such covariate is instrumental variable with will be introduced in the next chapter.
  - Do include covariates that are strongly related to the outcome.
- Instead of using the propensity scores, we can simply compute the distance between two observations using a known distance functions; for example, the Euclidean distance: $d(X_i, X_j) = \sqrt{\sum_{k=1}^{K}(X_{ik} - X_{jk})^2}$ and the Mahalanobis distance: $d(X_i, X_j) = (X_i - X_j)^T \Sigma^{-1}(X_i - X_j)$, where Σ is the covariance matrix of the data.
- It is not neccessary to aim for an accurate model for the propensity score. In fact, an accurate model might cause more problems than it solves. For example, consider a logistic model that can perfectly predicts whether an unit received the treatment or control. In this case, the propensity scores are all 0 or 1, indicating that there is no overlap, and matching is no differen than randomly assigned a controlled unit to each treated unit.
- There are several modifications for the matching algorithm:
  - We can match each treated unit with $k > 1$ controlled unit closest to it; this is sometimes called k-to-1 matching.
  - We can specify a threshold $d > 0$ and match each treated unit with all controlled units that are less than d distance away.
  - In the two algorithms above, we can give more weights to closer matches and less weights to farther matches.

There is also an R library `MatchIt` which is dedicated to matching for causal estimations. Check out this detailed example to get started.

17.3 Inverse probability weighting

We can also use propensity scores as units' weights without any matching. The idea is to weight the sample so that it is representative of the group of interest.

To illustrate the idea of inverse probability weighting, we consider the following hypothetic scenario: Suppose for simplicity that there is only one confounder X which has only two possible values: x_1 and x_2. Assume further that the potential outcomes are:

$$y^1(x_1) = 3, \quad y^0(x_1) = 1$$
$$y^1(x_2) = 2, \quad y^0(x_2) = 1,$$

and the probability scores are:

$$p(x_1) = \Pr[Z = 1|X = x_1] = 0.7, \quad p(x_2) = \Pr[Z = 1|X = x_2] = 0.4. \quad (17.1)$$

Below is an example of count data generated from these conditional distributions.

	$X = x_1$	$X = x_2$
$Z = 1$	70	80
$Z = 0$	30	120
τ	2	1

The ATE is $\frac{2*70+2*30+1*80+1*120}{70+30+80+120} = 1.33$. Due to the imbalance in the numbers of observed y^1 and y^0 for each value of X, the difference-in-mean estimate of $\frac{3*70+2*80}{70+80} - \frac{1*30+1*120}{30+120} = 1.47$ does not exactly match the ATE.

The can improve our estimate by balancing the numbers of treated and controlled units. If we were to know the conditional distribution Equation 19.1, we could have scaled the number of treated units down by a factor of $\Pr[Z = 1|X = x_i]$ and the number of controlled units by a factor of $\Pr[Z = 0|X = x_i]$ to obtain the so-called *pseudo-population* as shown below:

	$X = x_1$	$X = x_2$
$Z = 1$	100	200
$Z = 0$	100	200
τ	2	1

Now the difference-in-mean estimate is accurate: $\frac{3*100+2*200}{100+200} - \frac{1*100+1*200}{100+200} = 1.33$, which exactly matches the ATE. Since we generally do not know the conditional distributions Equation 19.1, we estimate them using a logistic regression $\hat{z} = \hat{p}(x) = \text{logit}^{-1}(\beta_0 + \beta_1 x)$.

In general, the inverse probability weighting estimate (IPW) is:

$$\begin{aligned}\hat{\tau}^{\text{IPW}} &= \frac{1}{n}\sum_{i;z_i=1}\frac{y_i}{\hat{p}(x_i)} - \frac{1}{n}\sum_{i;z_i=0}\frac{y_i}{1-\hat{p}(x_i)} \\ &= \frac{1}{n}\sum_{i=1}^{n}\frac{y_i z_i}{\hat{p}(x_i)} - \frac{1}{n}\sum_{i=1}^{n}\frac{y_i(1-z_i)}{1-\hat{p}(x_i)}.\end{aligned}$$

Theoretically, if $\hat{p} = p$ and the ignorability assumption holds, then $\hat{\tau}^{\text{IPW}}$ is an unbiased estimator of τ_{ATE}. To show this, we first compute the expectation of the first term.

$$\begin{aligned}
\mathbb{E}\left[\frac{YZ}{p(X)}\right] &= \mathbb{E}\left[\mathbb{E}\left[\frac{YZ}{p(X)}\middle| X\right]\right] \\
&= \mathbb{E}\left[\mathbb{E}\left[\frac{Y^1\ T}{p(X)}\middle| X\right]\right] \\
&= \mathbb{E}\left[\frac{\mathbb{E}[Y^1|X]\ \mathbb{E}[T|X]}{p(X)}\right] \\
&= \mathbb{E}\left[\mathbb{E}[Y^1|X]\right] \\
&= \mathbb{E}[Y^1].
\end{aligned}$$

Similarly, we have $\mathbb{E}\left[\frac{Y(1-Z)}{1-\hat{p}(X)}\right] = \mathbb{E}[Y^0]$. It follows that $\mathbb{E}[\tau^{\text{IPW}}] = \mathbb{E}[Y^1] - \mathbb{E}[Y^0]$.

For each unit (x, z, y), we let $\hat{p}(X)$ be the propensity score obtained from the logistic regression. We will put a weight on each unit as follows:

- Estimating the average treatment effect (ATE)
  - For every treated unit (x, z, y), we put a weight of $\frac{1}{\hat{p}(x)}$.
  - For every controlled unit (x, z, y), we put a weight of $\frac{1}{1-\hat{p}(x)}$.
- Estimating the average effect of treatment on the treated (ATT)
  - For every treated unit (x, z, y), we put a weight of 1.
  - For every controlled unit (x, z, y), we put a weight of $\frac{\hat{p}(x)}{1-\hat{p}(x)}$.
- Estimating the average effect of treatment on the controlled (ATC)
  - For every treated unit (x, z, y), we put a weight of $\frac{1-\hat{p}(x)}{\hat{p}(x)}$.
  - For every controlled unit (x, z, y), we put a weight of 1.

Below is example code for estimating ATT of the child care program using the inverse probability weighting.

```
inv.logit <- plogis

wt.iptw <- inv.logit(pscores) / (1 - inv.logit(pscores))
wt.iptw[cc2$treat==0] <- wt.iptw[cc2$treat==0]
wt.iptw[cc2$treat==1] <- 1

ps_fit_iptw_design <- svydesign(ids=~1, weights=wt.iptw, data=cc2)
reg_ps.iptw <- svyglm(ppvtr.36 ~ treat + bw + bwg + hispanic
                      + black + b.marr + lths + hs + ltcoll
                      + work.dur + prenatal + sex + first
                      + preterm + momage + dayskidh + income,
```

```
                      design=ps_fit_iptw_design, data=cc2)

summary(reg_ps.iptw)$coef['treat', 1:2]
```

```
Estimate Std. Error
8.372319   2.332639
```

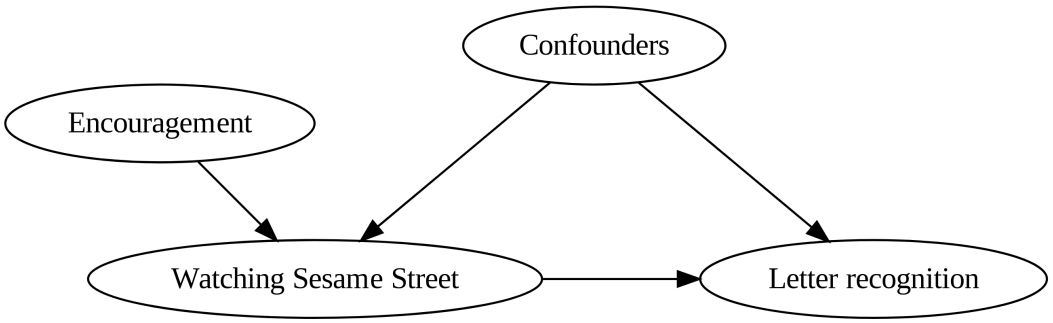
Chapter 18

Instrumental variables

18.1 Motivation

Suppose that we want to estimate the effect of watching a TV show Sesame Street on preschool children's recognition of English letters. We might consider an experiment where the treatment is watching Sesame Street. However, it would be difficult for us to force the children to watch the TV show for prevent them from watching it.

Instead, what we can encourage a random subgroup of the children to watch the show. Now, we have an additional variable, the *encouragement* that affects the treatment variable.



Let us take a look at the data, which is contained `sesame.csv`.

```
set.seed(0)
library(brms)
library(rstanarm)
```

```r
sesame <- read.csv("data/sesame.csv")

head(sesame[, c("encouraged", "watched", "setting", "site", "postlet")])
```

```
  encouraged watched setting site postlet
1          1       0       2    1      30
2          1       1       2    1      37
3          0       1       2    1      46
4          0       0       2    1      14
5          0       1       2    1      63
6          0       1       2    1      36
```

The outcome is `postlet`, which is the post-treatment measurement of the letter recognition task.

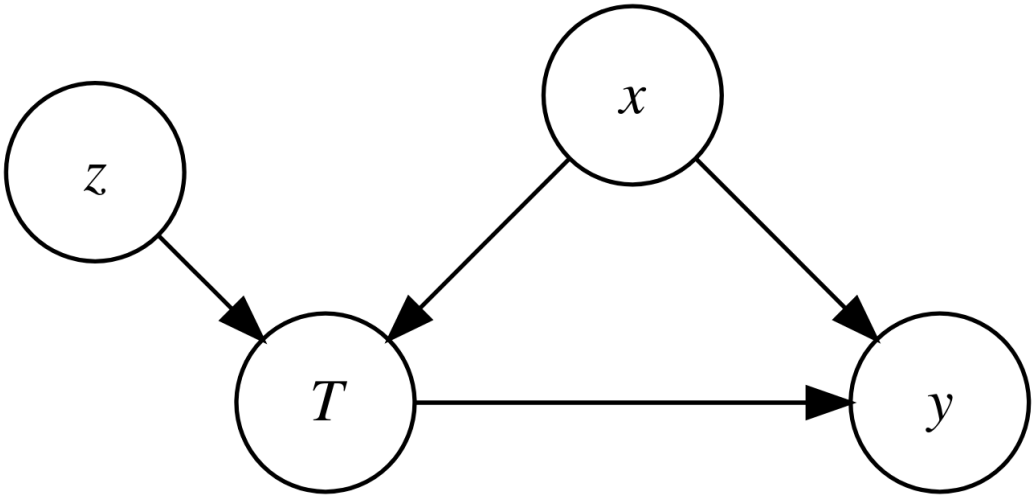
The encouragement is an example of an *instrumental variable* (IV), a variable that affects the treatment but does not affect the outcome. This is an example of *quasi-experiment*, that is, an experiment to study the causal effect without random treatment assigment.

18.2 Terminologies for instrumental varialbes

In addition to the usual causal notations of the treatment variable T, the outcome variable y, and the confounders x, we also have:

- the instrumental variable z affecting the treatment,
- the *potential* treatments T^0 and T^1 which is observed only if $z = 0$ and $z = 1$, respectively.
- the *potential* outcomes y^0 and y^1 which is observed only if $z = 0$ and $z = 1$, respectively.

We summarize the variables in the graphical model below.



We can estimate the effect of the instrument on the outcome—sometimes called *intent-to-treat* (ITT) effect—using the difference in the means of encouraged and not encouraged units effect. This is different than the treatment effect which is usually the quantity of interest. We will discuss the formulations and assumptions imposed by this model and ways to estimate the average treatment effect.

We now categorize the units into four groups by their treatment response to the instrument.

1. **Compliers** Instrument has a positive effect on their treatments, that is, $T_i^1 = 1$ and $T_i^0 = 0$.
2. **Defiers** Instrument has a negative effect on their treatments, that is, $T_i^1 = 0$ and $T_i^0 = 1$.
3. **Never-takers** Units who *never* take the treatment, that is, $T_i^1 = T_i^0 = 0$.
4. **Always-takers** Units who *always* take the treatment, that is, $T_i^1 = T_i^0 = 1$.

We are only interested in estimating the average treatment effect on the compliers—this is called the *complier average causal effect* (CACE).

A good instrument induces many compliers and no defiers; we will discuss more about desired properties of an instrument in the next section.

18.3 Assumptions for instrumental variables

In addition to SUTVA, the method of intrumental variables relies on several assumptions.

Ignorability of the instrument. We assume that the randomization in the instrumental variable is independent of the potential treatments and the outcomes:

$$y^1, y^0, T^1, T^0 \perp z.$$

Unlike the standard causal analysis, the ignorability of the treatment does not have to be satisfied.

Monotonicity. We assume that there is no defiers, that is, no unit who would have taken the treatment if they were not encouraged and would not take the treatment if they were encouraged.

Relevance. Of course, in an instrumental variable design, there would be no hope in estimating the treatment effect if the instrument is unrelated to the treatment. This assumption can be written as:

$$\Pr[T_i^1 = 1 | z_i = 1] > 0 \quad \text{and} \quad \Pr[T_i^1 = 0 | z_i = 0] > 0.$$

Exclusion restriction. We assume that there is no effect of the instrument on the outcomes of the never-takers (who would not have taken the treatment either way) and always-takers (who would have taken the treatment either way).

We can come up with a story that violates the exclusion restriction. In the Sesame Street example, we could have parents who prohibited their children from watching television (so the children were never-takers). But after being encouraged to have their children watch the Sesame Street, they decided to purchases a similar educational material for their children to read instead.

If there are some covariates x, we might instead assume the conditional ignorability:

$$y^1, y^0, T^1, T^0 \perp z | x,$$

We will see that these assumptions lead to unbiasedness of the difference-in-means estimation in the next section.

18.4 Intent-to-treat (ITT) effect and complier average causal effec (CACE)

The following table shows hypothetical data of the Sesame Street experiment. The bold numbers are observed values.

Unit i	Potential treatment T_i^0	Potential treatment T_i^1	Unit Type c_i	Encouragement z_i	Potential outcome y_i^0	Potential outcome y_i^1	Instrument effect $y_i^1 - y_i^0$
1	**0**	1	complier	0	**67**	76	9
2	**0**	1	complier	0	**72**	80	8
3	**0**	0	never-taker	0	**68**	68	0
4	**1**	1	always-taker	0	**76**	76	0
5	**1**	1	always-taker	0	**74**	74	0
6	0	**1**	complier	1	74	**81**	7
7	0	**1**	complier	1	68	**78**	10
8	0	**0**	never-taker	1	70	**70**	0
9	1	**1**	always-taker	1	80	**80**	0
10	1	**1**	always-taker	1	82	**82**	0

Assuming that the instrument assignments are random, this data satisfies all the four assumptions above. Specifically, there is no defiers (no $T_i^0 = 1$ and $T_i^1 = 0$) and no instrument effect on the never-takers and always-takers.

Now let us calculate the intent-to-treat (ITT) effect using the difference in the means of encouraged and not encouraged children, which in turn is the average of the instrument effect τ_i.

$$\text{ITT} = \frac{9+8+0+0+0+7+10+0+0+0}{10} = 3.4.$$

We can also calculate the *complier average causal effect* (CACE), using the difference in the means of the treated and controlled compliers.

$$\text{CACE} = \frac{9+8+7+10}{4} = 8.5.$$

In our hypothetical data, we can write CACE in terms of ITT as follows:

$$\text{CACE} = \frac{\text{ITT}}{4/10} = \frac{\text{ITT}}{\Pr[c = \text{complier}]} = \frac{\mathbb{E}[y|z=1] - \mathbb{E}[y|z=0]}{\Pr[c = \text{complier}]}.$$

18.4.1 Compute CACE using ITT

In a instrumental variable design, our quantity of interest is the average treatment effect over the compliers, that is, the CACE. However, we generally do

not observe which units are complier, so we cannot compute CACE directly from the definition. But following computations will show that, under the four assumptions above, we can compute CACE using ITT.

First, we have

$$\begin{aligned}
\text{ITT} &= \mathbb{E}[y|z=1] - \mathbb{E}[y|z=0] \\
&= \mathbb{E}[y^1 - y^0] \\
&= \mathbb{E}[y^1 - y^0|c = \text{complier}]\Pr[c = \text{complier}] \\
&\quad + \mathbb{E}[y^1 - y^0|c = \text{defier}]\Pr[c = \text{defier}] \\
&\quad + \mathbb{E}[y^1 - y^0|c = \text{never-taker}]\Pr[c = \text{never-taker}] \\
&\quad + \mathbb{E}[y^1 - y^0|c = \text{always-taker}]\Pr[c = \text{always-taker}] \\
&= \mathbb{E}[y^1 - y^0|c = \text{complier}]\Pr[c = \text{complier}] \\
&= \text{CACE} * \Pr[c = \text{complier}],
\end{aligned}$$

where the last equality follows from the monotonicity assumption ($\Pr[c = \text{defier}] = 0$) and the exclusion restriction assumption ($Y^1 - Y^0 = 0$ for the never-takers and always-takers). To remove $\Pr[c = \text{complier}]$, we consider $\mathbb{E}[T|z = 1] - \mathbb{E}[T|z = 0]$. Using the monotonicity assumption as before, we obtain

$$\begin{aligned}
\mathbb{E}[T|z=1] - \mathbb{E}[T|z=0] &= \mathbb{E}[T^1 - T^0] \\
&= \mathbb{E}[T^1 - T^0|c = \text{complier}]\Pr[c = \text{complier}] \\
&\quad + \mathbb{E}[T^1 - T^0|c = \text{defier}]\Pr[c = \text{defier}] \\
&= \mathbb{E}[T^1 - T^0|c = \text{complier}]\Pr[c = \text{complier}] \\
&= \Pr[c = \text{complier}],
\end{aligned}$$

which cannot be zero because of the assumption that the instrument must have an effect on the treatment. Dividing these two expressions yields

$$\text{CACE} = \frac{\mathbb{E}[y|z=1] - \mathbb{E}[y|z=0]}{\mathbb{E}[T|z=1] - \mathbb{E}[T|z=0]}.$$

Now, using the ignorability assumption, we can estimate the numerator using the difference in the means of the outcomes of the encouraged children ($z = 1$) and not encouraged children ($z = 0$), which can be computed with a linear regression.

```
itt_zy <- stan_glm(postlet ~ encouraged, data=sesame,
                   refresh=0)
```

```
coef(itt_zy)["encouraged"]
```

The coefficient of `encouraged` is our estimate of ITT. And we can estimate the denominator using the difference in the means of the treatment assignments of those two groups of children.

```
itt_zt <- stan_glm(watched ~ encouraged, data=sesame,
                   refresh=0)

coef(itt_zt)["encouraged"]
```

```
encouraged
 0.3626182
```

The coefficient of `encouraged` is the proportion of compliers in the data. Dividing the ITT estimate by the proportion, we obtain a CACE estimate, sometimes called a *Wald estimate*.

```
wald_est <- coef(itt_zy)["encouraged"] / coef(itt_zt)["encouraged"]

wald_est
```

```
encouraged
  7.901558
```

When there are covariates x, the CACE needs to be defined for each level of x, and all of our derivations above have to be conditioned on x. More precisely, with the conditional ignorability $y^1, y^0, T^1, T^0 \perp z|x$ and the corresponding assumptions, we have

$$\text{CACE}(x) = \frac{\mathbb{E}[y|z=1,x] - \mathbb{E}[y|z=0,x]}{\mathbb{E}[T|z=1,x] - \mathbb{E}[T|z=0,x]}.$$

18.5 Two-stage least squares

We discuss a general method of estimating CACE, called *two-stage least square* (2SLS).

In 2SLS, we pe form two linear regressions:

1. Regression of the treatment variable on the instrument:

$$\hat{T} = \alpha_0 + \alpha_1 z_1.$$

2. Regression of the outcome on the first model's *predicted* treatment:

$$y = \beta_0 + \beta_1 \hat{T} + \varepsilon.$$

Here is an example of 2SLS on the Sesame Street data.

```
fit_2a <- stan_glm(watched ~ encouraged, data=sesame,
                   refresh=0)
sesame$watched_hat <- fit_2a$fitted
fit_2b <- stan_glm(postlet ~ watched_hat, data=sesame,
                   refresh=0)

fit_2b
```

```
stan_glm
 family:       gaussian [identity]
 formula:      postlet ~ watched_hat
 observations: 240
 predictors:   2
------
            Median MAD_SD
(Intercept) 20.5    4.0
watched_hat  8.1    5.0

Auxiliary parameter(s):
      Median MAD_SD
sigma 13.3    0.6

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

Here, the estimated treatment effect is 8.1. However, the second regression does not give the correct standard error as it has to be computed from the observed treatment assignments, not the predicted one. Instead, we can fit 2SLS using the `brms` library and return the correct standard error.

```
f1 <- bf(watched ~ encour)
f2 <- bf(postlet ~ watched)
IV_brm_a <- brm(f1 + f2, data=sesame, refresh=0)
```

```
IV_brm_a
```

```
 Family: MV(gaussian, gaussian)
  Links: mu = identity; sigma = identity
         mu = identity; sigma = identity
```

```
Formula: watched ~ encour
         postlet ~ watched
   Data: sesame (Number of observations: 240)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Population-Level Effects:
                  Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
watched_Intercept     0.55      0.04     0.47     0.63 1.00     3977     3069
postlet_Intercept    20.44      3.71    13.37    28.21 1.00     2076     2310
watched_encour        0.36      0.05     0.26     0.46 1.00     4128     2913
postlet_watched       8.10      4.69    -1.61    17.05 1.00     1956     1862

Family Specific Parameters:
              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma_watched     0.38      0.02     0.35     0.42 1.00     4661     2943
sigma_postlet    12.64      0.69    11.47    14.19 1.00     2790     2448

Residual Correlations:
                        Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
rescor(watched,postlet)     0.16      0.15    -0.13     0.45 1.00     1839
                        Tail_ESS
rescor(watched,postlet)     2117

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

We can also incorporate the covariates by adding them to both regression models. For example, let us add two covariates: `site` and `setting` to our models.

```
f1 <- bf(watched ~ encour + prelet + setting + factor(site))
f2 <- bf(postlet ~ watched + prelet + setting + factor(site))
IV_brm_b <- brm(f1 + f2, data=sesame, refresh=0)
```

```
IV_brm_b
```

```
 Family: MV(gaussian, gaussian)
  Links: mu = identity; sigma = identity
         mu = identity; sigma = identity
Formula: watched ~ encour + prelet + setting + factor(site)
         postlet ~ watched + prelet + setting + factor(site)
   Data: sesame (Number of observations: 240)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000
```

```
Population-Level Effects:
                   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
watched_Intercept      0.66      0.11     0.45     0.87 1.00     3995     3119
postlet_Intercept      1.33      4.51    -7.62    10.27 1.00     1803     1713
watched_encour         0.34      0.05     0.24     0.44 1.00     5514     3055
watched_prelet         0.01      0.00    -0.00     0.01 1.00     6257     3176
watched_setting       -0.05      0.05    -0.16     0.05 1.00     5656     2961
watched_factorsite2    0.03      0.07    -0.10     0.17 1.00     3359     2622
watched_factorsite3   -0.11      0.07    -0.24     0.02 1.00     3467     2980
watched_factorsite4   -0.34      0.07    -0.49    -0.20 1.00     3891     3161
watched_factorsite5   -0.29      0.10    -0.49    -0.10 1.00     3778     3169
postlet_watched       13.90      3.86     6.40    21.33 1.00     1721     1917
postlet_prelet         0.70      0.08     0.56     0.85 1.00     5399     2797
postlet_setting        1.60      1.42    -1.20     4.39 1.00     3497     3150
postlet_factorsite2    8.39      1.87     4.64    12.05 1.00     4126     3028
postlet_factorsite3   -3.98      1.76    -7.40    -0.43 1.00     3237     2916
postlet_factorsite4    0.87      2.32    -3.67     5.58 1.00     2094     2162
postlet_factorsite5    2.69      2.81    -2.85     8.26 1.00     3138     2528

Family Specific Parameters:
             Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma_watched     0.36      0.02     0.33     0.39 1.00     5986     2795
sigma_postlet     9.44      0.54     8.51    10.63 1.00     3014     2271

Residual Correlations:
                        Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
rescor(watched,postlet)    -0.18      0.15    -0.46     0.13 1.00     1692
                        Tail_ESS
rescor(watched,postlet)     2050

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

The instrumental variables' treatment effect estimate is the coefficient of `postlet_watched`, which is 13.9 with standard error 3.9.

The 2SLS method can be easily extended to continuous instrument and treatment variables. But we must be careful in the case of binary instrument and continuous treatment as there is no universal way—as least without parametric assumptions—to quantify the instrument's effect on the treatment.

18.6 Multiple instruments, treatments and covariates

We can have multiple variables playing different roles in the instrumental variables model. In general, we have

- The outcome y
- Multiple treatments $T_1, \dots, T_k$
- Multiple instruments $z_1, \dots, z_m$; these variables only affect the outcome through the treatments.
- Multiple covariates $x_1, \dots, x_l$; these variables affect the outcomes directly.

For each treatment and each instrument, we categorized the units by the compliance as follows:

- **Compliers** Instrument has a positive (negative) effect on their treatments.
- **Never-takers** and **Always-taker** Instrument has no effect on their treatments.
- **Defiers** Instrument has a negative (positive) effect on their treatments.

In order to estimate the CACE of each treatment's the following assumptions must be satisfied:

- **Ignorability of the instrument** $y^1, y^0, T_i^1, T_i^0 \perp z_j | x_1, \dots, x_l$ for $i = 1, \dots, k$ and $j = 1 \dots, m$.
- **Monotonicity** There is no defier, that is, no unit who reacts in the opposite direction of what we expect.
- **Relevance** The instrument and the treatment must be related.
- **Exclusion restriction** There is no instrument effect on the outcomes of the never-takers and always-takers.

In addition, to identify all treatment effects, we have to ensure that any combination of the observed treatments can be attained by adjusting the instruments; this only happens when the number of instruments is equal or greater than the number of treatments.

- **Full rank model** $m = k$ (exactly identified) or $m > k$ (overidentified).

With these assumptions, we can estimate the treatment effect(s) using the following 2SLS.

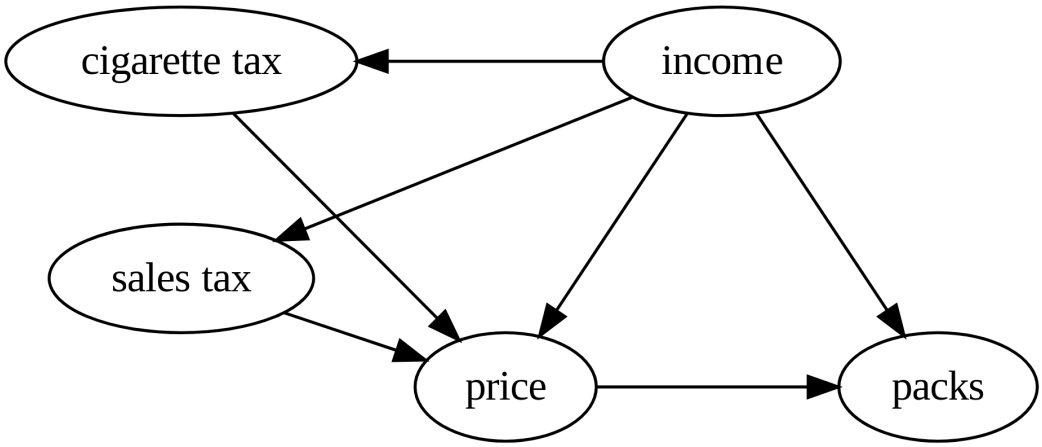
$$
\begin{aligned}
\hat{T}_1 &= \alpha_1 + \sum_{j=1}^{k} \alpha_{1j} z_j + \sum_{j=1}^{l} \delta_{1j} x_j \\
\hat{T}_2 &= \alpha_2 + \sum_{j=1}^{k} \alpha_{2j} z_j + \sum_{j=1}^{l} \delta_{2j} x_j \\
&\vdots \\
\hat{T}_m &= \alpha_m + \sum_{j=1}^{k} \alpha_{mj} z_j + \sum_{j=1}^{l} \delta_{mj} x_j \\
y &= \beta_0 + \sum_{i=1}^{m} \beta_i \hat{T}_i + \sum_{i=1}^{l} \gamma_i x_i + \varepsilon.
\end{aligned}
$$

As an example, we apply this method on cigarette sales data in the 48 continental US States in 1995 from Webel (2011).

```
cigarette <- read.csv("data/cigarette.csv")

head(cigarette[, c("packs", "rprice", "rincome", "salestax", "cigtax")])
```

```
      packs   rprice  rincome  salestax   cigtax
1 101.08543 103.9182 12.91535 0.9216975 26.57481
2 111.04297 115.1854 12.16907 5.4850193 36.41732
3  71.95417 130.3199 13.53964 6.2057067 42.86964
4  56.85931 138.1264 16.07359 9.0363074 40.02625
5  82.58292 109.8097 16.31556 0.0000000 28.87139
6  79.47219 143.2287 20.96236 8.1072834 48.55643
```

We would like to estimate the treatment effect of cigarette price (`rprice`) on the number of cigarette packs sold (`packs`). However, the price itself might be affected by the demand for cigarettes; so we instead add two instrumental variables that are rather affected by fiscal policy, namely sales tax (`salestax`) and cigarette tax (`cigtax`). We also has income (`rincome`) as a covariate. Our model is summarized in the following graph:



With this, we can now perform 2SLS using `brm`.

```
lm1 <- bf(log(rprice) ~ salestax + cigtax + log(rincome))
lm2 <- bf(log(packs) ~ log(rprice) + log(rincome))
IV_cig <- brm(lm1 + lm2, data=cigarette, refresh=0)
```

```
IV_cig
```

```
 Family: MV(gaussian, gaussian)
  Links: mu = identity; sigma = identity
         mu = identity; sigma = identity
Formula: log(rprice) ~ salestax + cigtax + log(rincome)
         log(packs) ~ log(rprice) + log(rincome)
   Data: cigarette (Number of observations: 48)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Population-Level Effects:
                    Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
logrprice_Intercept     4.10      0.10     3.89     4.30 1.00     3290
logpacks_Intercept      9.90      1.10     7.75    12.02 1.00     4381
logrprice_salestax      0.01      0.00     0.01     0.02 1.00     6481
logrprice_cigtax        0.01      0.00     0.01     0.01 1.00     4017
logrprice_logrincome    0.11      0.04     0.03     0.19 1.00     3254
logpacks_logrprice     -1.28      0.28    -1.83    -0.74 1.00     2792
logpacks_logrincome     0.29      0.25    -0.18     0.77 1.00     2662
                    Tail_ESS
logrprice_Intercept     2875
logpacks_Intercept      2757
```

```
logrprice_salestax       3128
logrprice_cigtax         3607
logrprice_logrincome     2812
logpacks_logrprice       2425
logpacks_logrincome      2699

Family Specific Parameters:
                Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma_logrprice     0.03      0.00     0.03     0.04 1.00     3110     2668
sigma_logpacks      0.20      0.02     0.16     0.25 1.00     2959     2399

Residual Correlations:
                           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
rescor(logrprice,logpacks)    -0.25      0.15    -0.53     0.05 1.00     2788
                           Tail_ESS
rescor(logrprice,logpacks)     2582

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

The point estimate of the coefficient is -1.28 with standard error 0.28, which implies that 1% increase in cigarette price would decrease cigarette consumption by $-1.00\% - 1.56\%$. Note that what we just estimated is the treatment effect on the complier states, that is, the states that would increase the cigarette price as a result of the tax raises.

18.7 Testing the assumptions

The method of instrument variables require many assumptions, some of which can be explained away with the prior knowledge about the variables. Even if this is not the case, we can still test some of the assumptions from the data.

18.7.1 Testing relevance

Not only the instrument must be related to the treatment, the relationship must be strong enough so that we can estimate the treatment effect reliably. A common problem is when we have a *weak instrument*, an instrument that does not have a strong relationship with the treatment, which can lead to biased estimates.

To test the relevance, we can use the *joint F-test* to compare the first-stage regression with and without the instruments; and we reject the null hypothesis if the model with the instruments has better predictive power. For more details on the join F-test, see e.g. James et al. (2021, chap. 3.2) and Hanck et al. (2019, chap. 7.3).

The usual statistical test with a specified type I error does not exclude weak instrument, so we have to come up with a new criteria on the F-statistic in order to decide if the instrument is strong enough. A rule of thumb says that an F-statistic greater than 10 is sufficient. For more reliable cutoffs and other testing methods for weak instruments, see Stock and Yogo (2002) for an extensive study.

18.7.2 Testing exclusion restriction

One way to test the exclusion restriction is by performing 2SLS only on the group of always-takers and never-takers and see if the effect of the instrument is negligible.

Another way is by fitting the second-stage regression with the *observed* treatment and the instrument:

$$y = \beta_0 + \sum_{i=1}^{k} \alpha_i z_i + \sum_{i=1}^{m} \beta_i T_i + \sum_{i=1}^{l} \gamma_i x_i + \varepsilon.$$

The exclusion restriction says that the instrument only affects the outcome through the treatment, so we can use the joint F-test to see if some of $\alpha_i \neq 0$, in which case we say that the assumption is violated.

When the 2SLS model is overidentified, that is, when there is more instruments than the treatments, there is also Sargan test, which look at the relationship between the residuals of the second-stage model and the instruments, and we reject the null if the relationship is significant.

Chapter 19

Regression discontinuity

Instead of a randomized experiment, we can design an experiment with no random element, and our variables still satisfy the ignorability assumption.

One possible way to do this is by *regression continuity*. The basic idea is to study the behavior of the outcome y over a range of a continuous variable X, often called the **forcing variable**. Assume that, from prior knowledge, we expect y to be a smooth function of X. If there is a sudden jump in y at a particular value $X = c$, it is possible that the sudden jump is caused by the effect of the treatment.

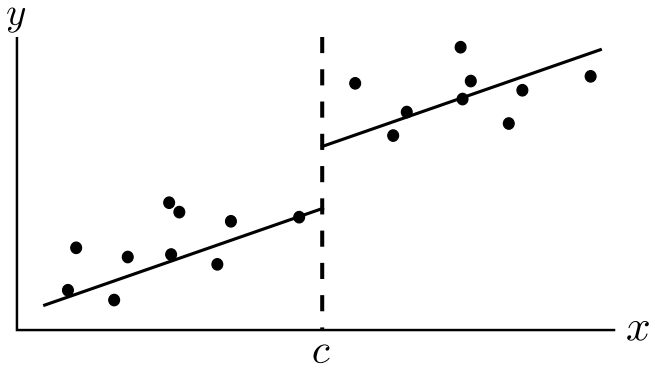


Figure 19.1: An example of regression discontinuity

In this case, we can define the treatment z with $z = 1$ if $z \geq c$ and $z = 0$ otherwise. This design has a severe overlap problem as the values of x between the treatment and control groups are disjoint. We can only estimate where the groups overlap, that is, at $X = c$. Thus we are interested in the *local* average treatment effect (LATE) at $X = c$, which can be computed as follows:

$$\tau_{\text{SRD}} = \mathbb{E}[y^1 - y^0 | X = c].$$

Right now, it is not possible to directly compute τ_{SRD} as it involves the counterfactual. So we have to make some assumptions first.

Assumptions

1. Conditional ignorability:

$$y^1, y^0 \perp z | X,$$

   which is automatically satisfied since z is deterministic.

2. Continuity of the expected potential outcomes with respect to the forcing variable:

$$\mathbb{E}[y^1|X] \text{ and } \mathbb{E}[y^0|X] \text{ are continuous in } X.$$

With these assumptions, the average treatment effect can be computed as follows:

$$\tau_{\text{SRD}} = \mathbb{E}[y^1 - y^0 | X = c]. \tag{19.1}$$

In other words, we can estimate SATE using the difference between the values of the two linear functions at $X = c$.

We now show that Equation 19.1 holds under the assumptions. By the continuity of the outcome, the ignorability, and SUTVA, we have

$$\mathbb{E}[y^1|X = c] = \lim_{x \uparrow c} \mathbb{E}[y^1|X = x] = \lim_{x \uparrow c} \mathbb{E}[y^1|z = 1, X = x] = \lim_{x \uparrow c} \mathbb{E}[y|X = x].$$

Similarly, we have $\mathbb{E}[y^0|X = c] = \lim_{x \downarrow c} \mathbb{E}[y|X = x]$. It follows that

$$\tau_{\text{SRD}} = \lim_{x \uparrow c} \mathbb{E}[y|X = x] - \lim_{x \downarrow c} \mathbb{E}[y|X = x].$$

Consequently, we can estimate LATE using the difference between the values of the linear functions at $X = c$. As there is no overlap at any other value of X, it is not possible to estimate LATE at these points since we cannot observe couterfactuals. Instead, we have to extrapolate the LATE at $X = c$ other values of X. The estimates of LATE can be seen as the vertical distance between two red lines in the plot below.

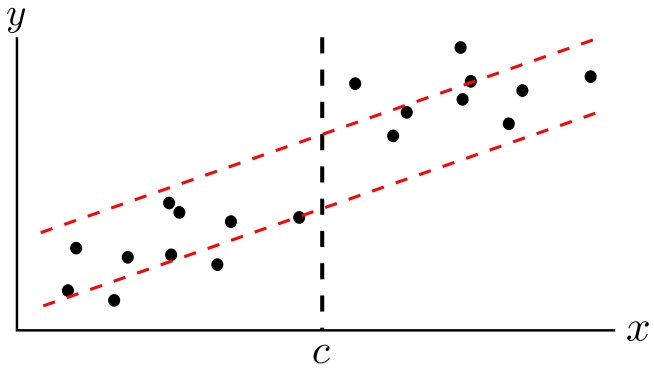


Figure 19.2: Extrapolation of LATE estimate at the cutoff

19.1 Deriving the linear regression

To find the linear regression that gives us an estimate of LATE, we first subtract the forcing variable by the cutoff.

$$x' = x - c.$$

Now we can write the equations for the two linear regression with intercepts α_0 and $\alpha_0 + \delta$ as follows:

$$\begin{aligned} \text{Control:} \quad & y^0 = \alpha_0 + \alpha_1 x' + \text{error} \\ \text{Treatment:} \quad & y^1 = \alpha_0 + \delta + \beta_1 x' + \text{error}. \end{aligned}$$

Taking the expectation on both equations, conditioning on $x = c$ (or $x' = 0$),

$$\mathbb{E}[y^1 | x = c] - \mathbb{E}[y^0 | x = c] = (\alpha_0 + \delta) - \alpha_0 = \delta.$$

The combination of two models above is equivalent to a single interactive model:

$$y = \alpha_0 + \delta z + \alpha_1 x' + (\beta_1 - \alpha_1) z x' + \text{error}.$$

Notice that δ is a coefficient of the treatment variable z. So we can fit a linear regression (possibly with an interaction between the treatment and forcing variable) and obtain the coefficient of the treatment variable as an estimate of LATE.

Only regress near the cutoff. It is a good practice to fit the regression only on a small interval around the cutoff—this is to prevent the points that are far away from the cutoff to have any effect on the local estimate, as the following figure illustrates.

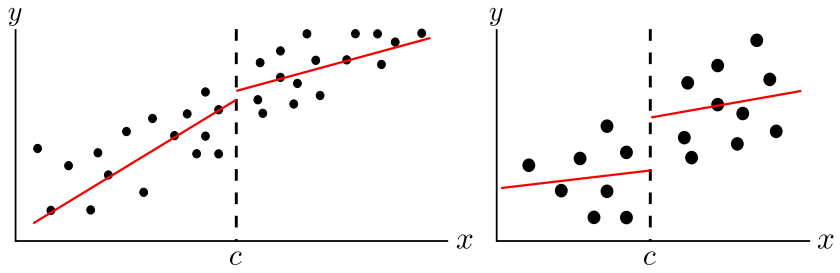


Figure 19.3: Left: Regressions on all points. Right: Regression on a small interval around the cutoff

19.2 Example: The effect of educational support on test scores in Chile

As an example, we consider the data from the Chilean government, who implemented "900 schools program" in an attempt to improve the performance of struggling public schools. The educational resources have been distributed to schools whose mean fourth-grade test scores are below a cutoff. Here, the forcing variable is the mean test score and the outcome is the follow-up reading test score in 1992.

Here is the list of variables that we are going to use:

Name	Description
`read92`	The score of reading test in 1992
`eligible`	1 if `rule2` is less than `cutoff`, 0 otherwise
`rule2`	the earlier test score minus `cutoff`
`cutoff`	The test score cutoff for educational support
`read88`	The score of reading test in 1988
`math88`	The score of math test in 1988

The data is contained in `chile.csv`.

```
set.seed(0)
library(brms)
library(rstanarm)
```

```
chile <- read.csv("data/chile.csv")

head(chile[, c("eligible", "rule2", "cutoff", "read88", "math88", "read92")])
```

```
  eligible       rule2 cutoff read88 math88 read92
```

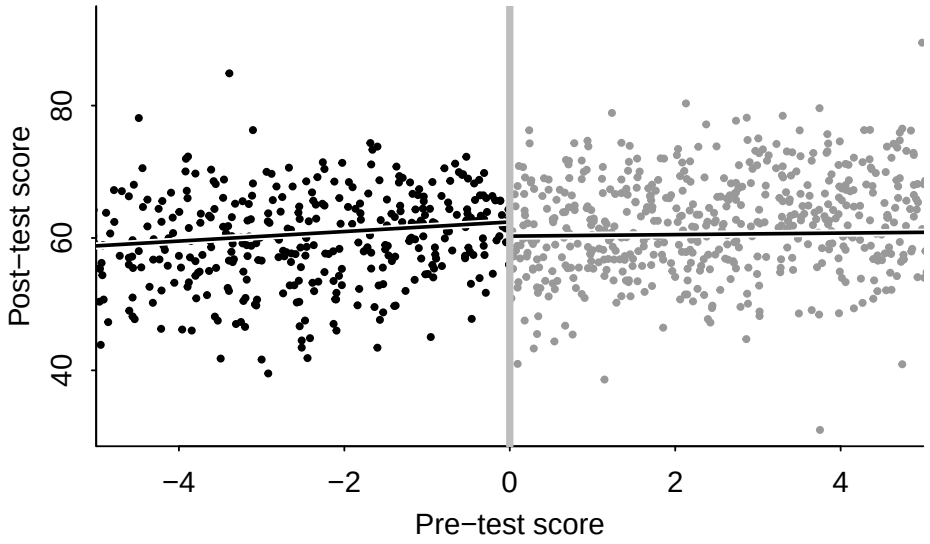
```
1        0  3.41499853   49.4  54.37  51.26 57.000
2        0 31.81499672   43.4  79.06  71.37 85.515
3        0  4.44500113   43.4  47.76  47.93 51.971
4        0 16.79999733   46.4  65.13  61.27 66.374
5        0  0.09499893   49.4  49.26  49.73 52.500
6        1 -2.03000116   46.4  50.51  38.23 55.333
```

Here, `eligible` is the treatment variable and `rule2` is the forcing variable. In each group, the expected reading score in 1992 should move continuously along the earlier test score, so expect the continuity assumption to be satisfied. The forcing variable `rule2` is already subtracted by the score cutoff; hence, we are ready to fit the model with an interaction. As mentioned before, we fit the regression only on a small interval around the cutoff. In this example, the cutoff for `rule2` is zero, so we shall fit on the subset of schools whose `rule2` is between -5 and 5.

```
fit_1 <- stan_glm(read92 ~ eligible + rule2 + eligible:rule2
                  + read88 + math88,
                  data=chile, subset = abs(rule2)<5,
                  refresh=0)

fit_1
```

```
stan_glm
 family:       gaussian [identity]
 formula:      read92 ~ eligible + rule2 + eligible:rule2 + read88 + math88
 observations: 883
 predictors:   6
 subset:       abs(rule2) < 5
------
               Median MAD_SD
(Intercept)    23.4    4.4
eligible        2.1    0.9
rule2           0.1    0.2
read88          0.6    0.1
math88          0.2    0.1
eligible:rule2  0.1    0.3

Auxiliary parameter(s):
      Median MAD_SD
sigma 6.9    0.2

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

The estimate of LATE is 2.1 with standard error 0.9. Let us take a look at the regression lines.



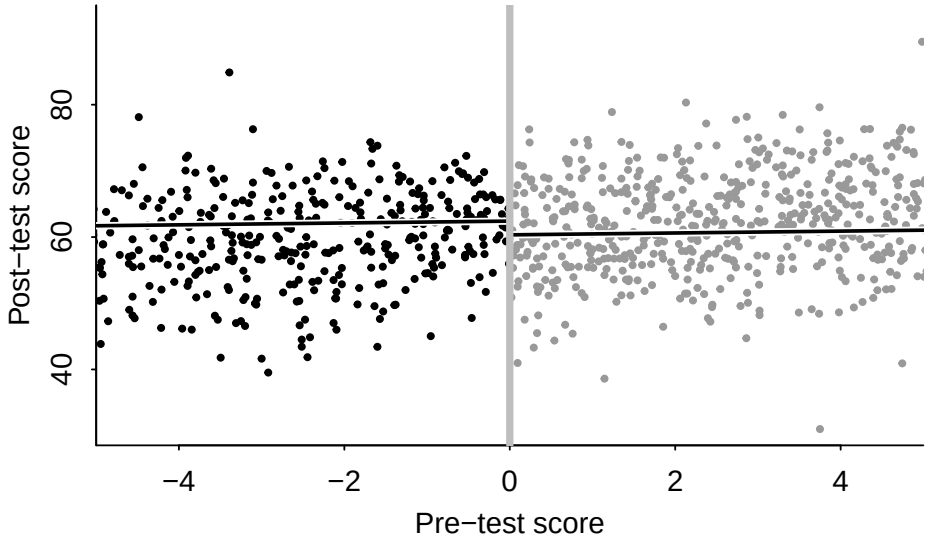
The data is too noisy to estimate the interaction, so we might instead fit the model without the interaction. In this case, the two lines are parallel to each other.

```
fit_2 <- stan_glm(read92 ~ eligible + rule2
                  + read88 +  math88,
                  data=chile, subset = abs(rule2)<5,
                  refresh=0)

fit_2
```

```
stan_glm
 family:       gaussian [identity]
 formula:      read92 ~ eligible + rule2 + read88 + math88
 observations: 883
 predictors:   5
 subset:       abs(rule2) < 5
------
            Median MAD_SD
(Intercept) 23.5    4.3
eligible     2.1    0.9
rule2        0.1    0.2
read88       0.6    0.1
math88       0.2    0.1
```

```
Auxiliary parameter(s):
      Median MAD_SD
sigma 6.9    0.2

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

We see that the estimate of LATE is the same as before. Let us take a look at the regression lines.



Chapter 20

Difference-in-differences

20.1 Example: effect of minimum wage on employment

Suppose that we would like to estimate the effect of raising the minimum wage on employment. With a lot of money and power, we could perform a randomized experiment by flipping a coin for each local market in countries. If it comes up head, we raise the minimum wage; if it comes up tail, we keep it the same.

Of course, this is just a thought experiment—the randomized experiment is not feasible. Nonetheless, it is possible to estimate the treatment effect when we have before-after data of a pair of units: both are controlled before, but only one of them is treated after. This is what Card and Krueger (1993) did after seeing that New Jersey's minimum wage was about to be raised from $4.25 to $5.05 in November 1992, while a neighboring Pennsylvania's minimum wage stayed the same at $4.25. They seized this opportunity and fielded two surveys to 400 fast food restaurants in both states: the first one in February 1992 and the second one in November 1992.

Let α and β be the deployment in New Jersey and Pennsylvania, respectively. Let δ be the effect of raising the minimum wage, and assume that any other factor had the same effect γ on both states (which might be possible since these two are adjacent). The data of employment obtained from the surveys would look like the following table:

	February 1992	November 1992	Difference
New Jersey	α	$\alpha + \gamma + \delta$	$\gamma + \delta$
Pennsylvania	β	$\beta + \gamma$	γ
Difference			δ

We see that the treatment effect δ is the difference between the state-wise before-and-after differences, or the *difference-in-differences*. Of course, this generally does not match the treatment effect due to noises, in which case we have the difference-in-differences (DID) estimate of the treatment effect.

The raw data can be downloaded from Card's personal website. Here, we will use the preprocessed data stored in `wage92.csv`.

```
set.seed(0)
library(rstanarm)
```

```
wage92 <- read.csv("data/wage92.csv")
wage92 <- na.omit(wage92)  # remove NA rows

head(wage92[, c("d_nj",
                "y_ft_employment_before",
                "y_ft_employment_after")])
```

```
   d_nj y_ft_employment_before y_ft_employment_after
4     0                  34.0                  20.0
5     0                  24.0                  35.5
7     0                  70.5                  29.0
8     0                  23.5                  36.5
9     0                  11.0                  11.0
10    0                   9.0                   8.5
```

Below are descriptions of the relevant variables:

Name	Description
`d_nj`	1 if New Jersey; 0 if Pennsylvania (**Treatment**)
`y_ft_employment_before`	Full time equivalent employment before treatment (**Outcome**)
`y_ft_employment_after`	Full time equivalent employment after treatment (**Outcome**)

Now we can compute the difference-in-differences estimate using the difference in the means of the employments.

```
wage_nj <- subset(wage92, d_nj == 1)
wage_pa <- subset(wage92, d_nj == 0)

before_nj <- mean(wage_nj$y_ft_employment_before)
after_nj <- mean(wage_nj$y_ft_employment_after)
```

```r
diff_nj <- after_nj - before_nj


before_pa <- mean(wage_pa$y_ft_employment_before)
after_pa <- mean(wage_pa$y_ft_employment_after)
diff_pa <- after_pa - before_pa

did <- diff_nj - diff_pa
```

Let us summarize this in a table as shown above.

```r
result <- data.frame(State = c("New Jersey", "Pennsylvania", "Difference"),
                     Before = c(before_nj, before_pa, NA),
                     After = c(after_nj, after_pa, NA),
                     Difference = c(diff_nj, diff_pa, did))

result
```

```
         State   Before    After Difference
1   New Jersey 20.65775 21.04842   0.390669
2 Pennsylvania 23.70455 21.82576  -1.878788
3   Difference       NA       NA   2.269457
```

The DID estimate tells us that raising the minimum wage from $4.25 to $5.05 would increase the employment by 2.27 on average.

20.2 Regression for the difference-in-differences estimate

We can also use a linear regression to obtain the DID estimate. Let y_{before} and y_{after} be the outcome before and after the time period, and z be the treatment assignment. We can regress the difference on the treatment variable:

$$y_{\text{after}} - y_{\text{before}} = \beta + \delta z + \varepsilon. \tag{20.1}$$

Then, the coefficient of the interaction term δ is the DID estimate. This is because

$$\mathbb{E}[y_{\text{after}} - y_{\text{before}}|z = 1] - \mathbb{E}[y_{\text{after}} - y_{\text{before}}|z = 0] = (\beta_0 + \delta) - \beta_0 = \delta.$$

Let us try this method on the employment data. First, we have to combine the employments before and after the wage raise into a single column, and add a time indicator.

```
fit_1 <- stan_glm((y_ft_employment_after - y_ft_employment_before) ~ d_nj,
                  data=wage92,
                  seed=0, refresh=0)

print(fit_1, digit=2)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      (y_ft_employment_after - y_ft_employment_before) ~ d_nj
 observations: 350
 predictors:   2
------
            Median MAD_SD
(Intercept) -1.88   1.09
d_nj         2.23   1.17

Auxiliary parameter(s):
      Median MAD_SD
sigma 8.74   0.33

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

The DID estimate is 2.23, with 1.17 standard error, which is close to the point estimate of 2.27 that we just computed directly from the differences between the means.

20.2.1 Different observations before and after the treatment time

Let P be a time indicator with $P = 0$ and $P = 1$ signifies the time before and after the treatment took effect, respectively. If the observations at $P = 0$ are different than those at $P = 1$, then we cannot compute $y_{\text{after}} - y_{\text{before}}$. Assuming that the observations in each of the treatment and control groups are independently from the same distribution, we can instead fit the following regression with an interaction term:

$$y = \beta_0 + \beta_1 z + \beta_2 P + \delta z P + \varepsilon.$$

The DID estimate is the coefficient δ of the interaction term, as it is the difference between the two coefficients of z from fitting $y = a + bz$ on the data with $P = 1$ and $P = 0$, respectively. More explicitly,

$$\begin{aligned}\mathbb{E}[y|z=1,P=1]-\mathbb{E}[y|z=0,P=1] &= (\beta_0+\beta_1+\beta_2+\delta)-(\beta_0+\beta_2+\delta)\\ &= \beta_1+\delta\\ \mathbb{E}[y|z=1,P=0]-\mathbb{E}[y|z=0,P=0] &= (\beta_0+\beta_1)-\beta_0\\ &= \beta_1.\end{aligned}$$

Subtracting these two equalities yields

$$\text{DID} = (\beta_1+\delta)-\beta_1 = \delta.$$

20.2.2 Difference-in-differences by matching

Alternatively, we can use propensity score matching to match each unit that was observed before the treatment time to a unit in the same group that was observed after. Then, we treat each pair as a single observation with the observed values of y_{before} and y_{after}. With these new observations, we can obtain the DID estimate by fitting the regression Equation 20.1.

In all cases, we have made a strong assumption that the changes in the outcomes without the treatment effect would be the same in both New Jersey and Pennsylvania. We will discuss more about the assumptions for the DID estimate in the next section.

20.3 Parallel trends assumption

From Equation 20.1, we define the *potential changes* as the difference between the potential outcome, with or without the treatment, and the outcome observed before applying the treatment.

$$d^1 = y^1 - y_{\text{before}}, \quad d^0 = y^0 - y_{\text{before}},$$

where y^1, y^0 are the potential outcomes. In view of Equation 20.1, in order for the coefficient δ to be a valid causal estimate, the dependent variable in the regression must be independent of the treatment assignment, which is guaranteed when

$$d^0 \perp z. \tag{20.2}$$

This is referred to as *parallel trends assumption*, as it implies that the change in a treated unit would be the same as that of a controlled unit had it not received the treatment. We can show that the DID estimate is an unbiased estimate of the **ATT** as follows:

$$
\begin{aligned}
&\mathbb{E}[y - y_{\text{before}}|z = 1] - \mathbb{E}[y - y_{\text{before}}|z = 0] \\
&= \mathbb{E}[y^1 - y_{\text{before}}|z = 1] - \mathbb{E}[y^0 - y_{\text{before}}|z = 0] \\
&= \mathbb{E}[d^1|z = 1] - \mathbb{E}[d^0|z = 0] \\
&= \mathbb{E}[d^1|z = 1] - \mathbb{E}[d^0|z = 1] \\
&= \mathbb{E}[y^1 - y^0|z = 1] \\
&= \mathbb{E}[y^1|z = 1] - \mathbb{E}[y^0|z = 1],
\end{aligned}
$$

where we used Equation 20.2 to show the third equality. Two comments are in order:

- If we instead have a stronger assumption: $d^1, d^0 \perp z$. Then **ATT** is the same as **ATE**, in which case we can estimate both with DID.
- If $y_{\text{before}} \perp z$ (which implies $\mathbb{E}[y_{\text{before}}|z = 1] = \mathbb{E}[y_{\text{before}}|z = 0]$), we can instead assume $y^0 \perp z$ and modify the proof to show that the DID estimate is an unbiased estimate of the ATT (if $y^1, y^0 \perp z$, then the ATT is the same as ATE).

With confounder covariates, however, this assumption might not be satisfied. For example, in the 1992 survey, almost half of the fast food restaurants were Burger King's, and around 80% of them were from New Jersey; so if Burger King was very responsive to the minimum wage raise compared to the other fast food restaurants, the potential employments in New Jersey would be lower than that in Pennsylvania.

Thus, we have to adjust for these confounder covariates, say x, in the ignorability assumption.

$$
d^1, d^0 \perp z|x.
$$

With this assumption, we obtain the DID estimate by regressing on the confounders as well. In the employment example, we can adjust for the five franchise indicators

```
fit_2 <- stan_glm((y_ft_employment_after - y_ft_employment_before) ~
                      d_nj + x_burgerking + x_kfc
                  + x_roys + x_wendys + x_co_owned,
                  data=wage92,
                  seed=0, refresh=0)

print(fit_2, digit=2)
```

```
stan_glm
 family:       gaussian [identity]
```

```
 formula:      (y_ft_employment_after - y_ft_employment_before) ~ d_nj + x_burgerking +
       x_kfc + x_roys + x_wendys + x_co_owned
 observations: 350
 predictors:   7
------
             Median MAD_SD
(Intercept)  -3.22  26.83
d_nj          2.35   1.16
x_burgerking  1.69  26.74
x_kfc         1.87  26.76
x_roys       -0.30  27.07
x_wendys      1.06  26.85
x_co_owned    0.36   1.10

Auxiliary parameter(s):
      Median MAD_SD
sigma 8.73   0.34

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

Nonetheless, in this example, the DID estimate of 2.35 with standard error 1.16 is not noticably different that the previous one.

Note. For the reasons explained in Section 15.2.2, do not adjust for post-treatment covariates.

20.3.1 Checking the parallel trends assumption

It is possible to check for the parallel trends assumption if the data was recorded at multiple time points before the treatment took effect. If this is the case, there are mainly three ways to check for the parallel trend assumptions.

1. *Check the plot over time.* We can compare the graphs of the average outcomes between the treatment and control group over a period of time leading up to when the treatment occurred. If the graphs are moving apart or approaching each other, the parallel trend assumption might not be satisfied.
2. *Statistical test.* To see whether the trends different between the treatment and control groups, we can fit the following regression with an interaction term on the data before the treatment occurred:

$$y = \beta_0 + \beta_1 * \text{Time} + \beta_2 * \text{Time} * z + \varepsilon,$$

and perform a statistical test to see if $\beta_2 = 0$, in which case it is unlikely that the trends are different. On the other hand, if we reject $\beta_2 = 0$, we still have

to look at the graphs and see if the difference in trends is visually small but the test was performed with a large sample size, or if the outcomes vastly differ only over a brief moment, outside of which the trends are very similar.

3. *The placebo test.* The idea is to treat some of untreated data as *fake* treated data and see if our DID estimate is significant, even though there should not be any effect. More precisely, we first remove the time period that the treatment took effect. Then, we choose an earlier time period, and let the outcomes over this period be the results of a *fake treatment*. If the DID estimate with this fake treatment is significant, then the parallel trends assumption might be violated.

Chapter 21

Panel data

Panel data, or longitudinal data, is data that contains multiple observations of each unit. As we can see from the method of difference-in-differences, under certain assumptions, the temporal nature of the data can be exploit to extract the treatment effect. Below are relevant variables in panel data analysis:

- $t = 1, 2, \ldots, T$ is the time.
- y_{it} is unit i's time-varying outcome.
- x_{it} is unit i's vector of time-varying predictors, including the treatment.
- u_i is unit i's vector of unobserved time-invariant effect.

Here, we impose that the unobserved individual effect is not changing over time. For example, u_i might be i's health condition before receiving scheduled treatments. The difference-in-differences setting is a special case of panel data with $T = 2$ and u_i is the same for all units in each of the treatment and control groups.

Throughout this chapter, we will use panel data of 545 observations of young men from 1980 to 1987 from Vella and Verbeek (1998). The outcome (`wage`) is the log of wage, and the predictors that we will use are: years of experience (`exper`), whether the wage is set by collective bargaining (`union`) and the marriage status (`married`). We will also use the unique identifier (`nr`) and the year identified (`year`).

```
set.seed(0)
library(rstanarm)
```

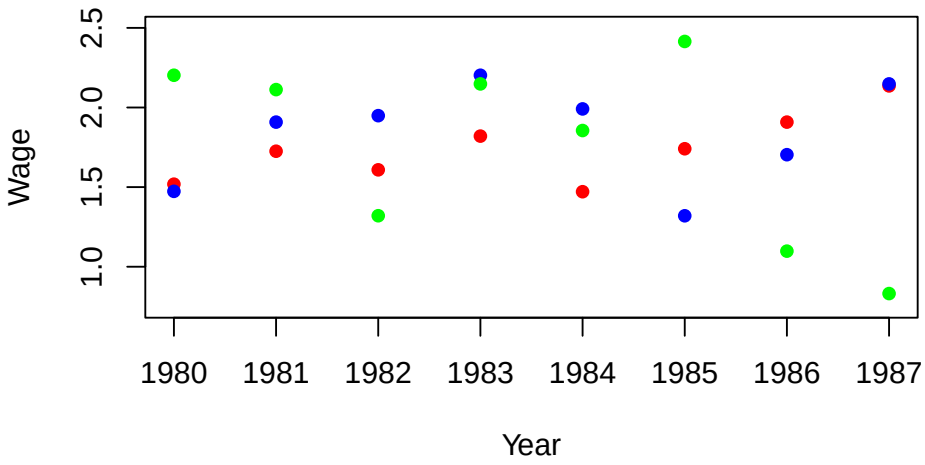
```
wage97 <- read.csv("data/wage97.csv")
wage97 <- wage97[, c("nr", "year", "exper",
                     "union", "married", "wage")]
```

```
wage97$nr <- factor(wage97$nr)
wage97$year <- factor(wage97$year)

lookup <- c("yes" = 1, "no" = 0)
wage97$union <- lookup[wage97$union]
wage97$married <- lookup[wage97$married]

head(wage97)
```

```
  nr year exper union married     wage
1 13 1980     1     0       0 1.197540
2 13 1981     2     1       0 1.853060
3 13 1982     3     0       0 1.344462
4 13 1983     4     0       0 1.433213
5 13 1984     5     0       0 1.568125
6 13 1985     6     0       0 1.699891
```

Notice that Unit #13 had been observed once a year since 1980. To see what kind of model we are looking for, we take a look at the wages of three people in the dataset.

```
colors <- c("red", "blue", "green")
plot(1, type = "n", xlab = "Year", ylab = "Wage",
     xlim = c(1980,1987), ylim = c(0.75, 2.5))

for (i in 1:3) {
    points(1980:1987, wage97$wage[8*i+1:8*(i+1)],
           pch = 16, col = colors[i])
}
```



This suggests that simply fitting a regression on this data would be a big mistake, since such model assumes the same level of wages for every young men. Instead, we have to come up with a model that allows for the difference in wages.

21.1 Fixed effects model

We assume that the outcome is composed of the individual time-invariant effect and time-specific treatment effect, resulting in the following model:

$$y_{it} = \beta x_{it} + u_i + \varepsilon_{it}, \quad \varepsilon_{it} \sim N(0, \sigma). \tag{21.1}$$

The vectors of coefficients β is the same for all units and times, but the individual effect u_i varies across the units.

The inclusion of the fixed effect term u_i means that we are controlling for anything that does not change over time; this is where we exploit the panel structure to control for unobserved confounders, given that they are constant over time. After fitting the model, the confounder effects will be stored in u_i.

One way to obtain an estimate of u_i is by adding an indicator variable that is unique for each unit, but this would be computationally inefficient if we observe a large number of units. Instead, we use a trick that will remove u_i from the equation. First, we compute

- $\bar{y}_i$ the mean of User i's time-varying outcomes,
- $\bar{x}_i$ the mean of User i's time-varying predictors.

Then, we subtract the variables in Equation 21.1 by their means. Since u_i is fixed over time, the mean of u_i is u_i itself.

$$\begin{aligned} y_{it} - \bar{y}_i &= \beta(x_{it} - \bar{x}_i) + (u_i - u_i) + (\varepsilon_{it} - \bar{\varepsilon}_i) \\ &= \beta(x_{it} - \bar{x}_i) + (\varepsilon_{it} - \bar{\varepsilon}_i). \end{aligned}$$

Denoting $\ddot{y}_{it} = y_{it} - \bar{y}_i$, $\ddot{x}_{it} = x_{it} - \bar{x}_i$ and $\ddot{\varepsilon}_{it} = \varepsilon_{it} - \bar{\varepsilon}_i$, we obtain a new regression equation:

$$\ddot{y}_{it} = \beta \ddot{x}_{it} + (\varepsilon_{it} - \bar{\varepsilon}_i).$$

Thus, we can instead fit on the *demeaned* data to obtain the estimate of the coefficients, and in particular, the average treatment effect.

Let us try this on the wage data. We can subtract the mean from each variable by using the `scale` function with the `scale` argument set to `FALSE`.

```
wage97$exper_c <- with(wage97, exper - ave(exper, nr))
wage97$union_c <- with(wage97, union - ave(union, nr))
wage97$married_c <- with(wage97, married - ave(married, nr))
wage97$wage_c <- with(wage97, wage - ave(wage, nr))

head(wage97[, c("nr", "year", "exper_c",
                "union_c", "married_c", "wage_c")])
```

```
  nr year exper_c union_c married_c      wage_c
1 13 1980    -3.5  -0.125         0 -0.05811187
2 13 1981    -2.5   0.875         0  0.59740792
3 13 1982    -1.5  -0.125         0  0.08880961
4 13 1983    -0.5  -0.125         0  0.17756126
5 13 1984     0.5  -0.125         0  0.31247301
6 13 1985     1.5  -0.125         0  0.44423887
```

Now we can fit the regression model with e.g. `stan_glm`. However, the differences in the number of observations, predictors' variances, etc. suggest that we should assume heterogeneous standard errors across the units. We can fit such model using `stan_glmer` from the `rstanarm` library. In the formula, we add `(1 | nr)`, which means that we are varying the intercept (`1`) by the unit.

```
fit_1 <- stan_glmer(wage ~ married + union + exper
                    + (1 | nr),
                    data=wage97, seed=0, refresh=0)

print(fit_1, digits=3)
```

```
stan_glmer
 family:       gaussian [identity]
 formula:      wage ~ married + union + exper + (1 | nr)
 observations: 3115
------
            Median MAD_SD
(Intercept) 1.241  0.027
married     0.080  0.020
union       0.102  0.022
exper       0.055  0.003

Auxiliary parameter(s):
      Median MAD_SD
sigma 0.348  0.005

Error terms:
 Groups   Name        Std.Dev.
```

```
 nr       (Intercept) 0.3852
 Residual             0.3485
Num. levels: nr 429

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

The model tells us that marriage increases the wage by 8% on average (since the outcome is the log of the wage).

The intercept of each individual's model is a sum of two components:

- The first component is the *fixed effects* which is the same for all units; this is the value of `intercept` shown above,
- The second component is the *random effects* which varies from unit to unit. To see the random effects of all units, we can use `ranef` on the fitted model.

```
ranef(fit_1)$nr[1:5, ]
```

```
[1] -0.22613167 -0.01290594  0.17485998  0.26456645 -0.10256147
```

The actual intercept u_i of each individual's model can be obtained using `coef` on the fitted model.

```
head(coef(fit_1)$nr)
```

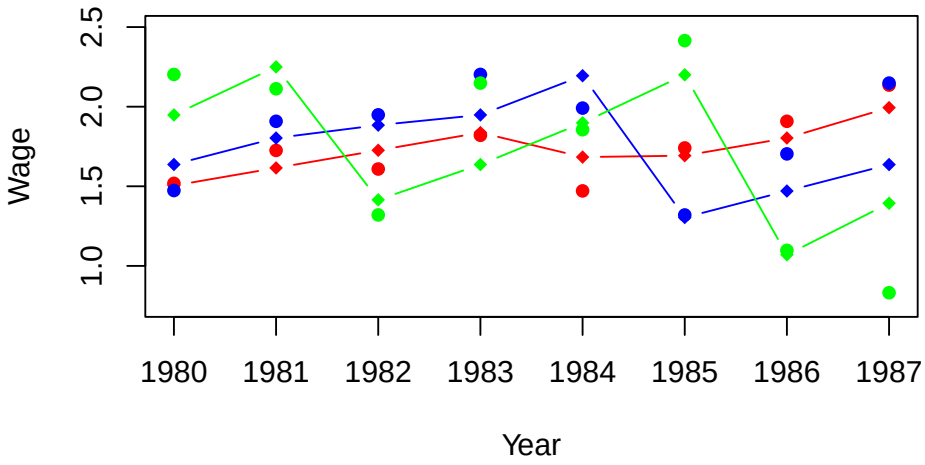
```
    (Intercept)    married     union      exper
13     1.014496 0.08003359 0.1019362 0.05538296
17     1.227722 0.08003359 0.1019362 0.05538296
45     1.415488 0.08003359 0.1019362 0.05538296
110    1.505195 0.08003359 0.1019362 0.05538296
120    1.138067 0.08003359 0.1019362 0.05538296
126    1.621628 0.08003359 0.1019362 0.05538296
```

Notice that each intercept is equal to the sum of the fixed effects and random effects above.

Let us take a look at the actual wage versus the predicted wage from the fixed effects model. Overall, the predictions (the line plots) are close to the actual wages.

```
colors <- c("red", "blue", "green")
plot(1, type = "n", xlab = "Year", ylab = "Wage",
     xlim = c(1980,1987), ylim = c(0.75, 2.5))
```

```r
for (i in 1:3) {
    points(1980:1987, wage97$wage[8*i+1:8*(i+1)],
           pch = 16, col = colors[i])
    lines(1980:1987, fit_1$fitted.values[8*i+1:8*(i+1)],
          pch = 18, col = colors[i], type = "b")
}
```

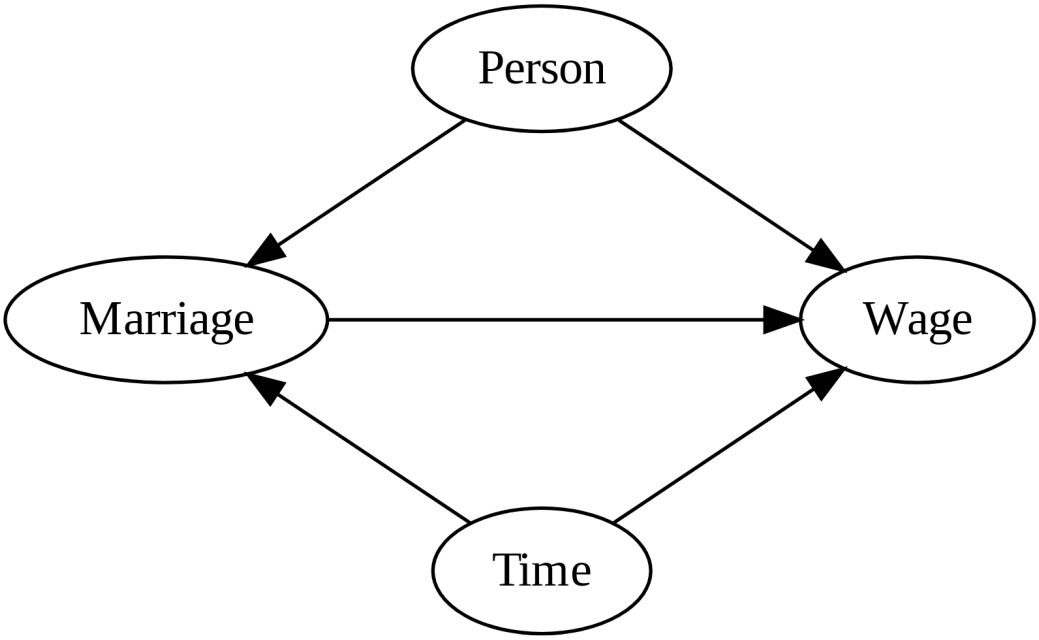


21.2 Time effects

In addition to the user's effect, we can have a fixed effect at each time as well.

$$y_{it} = \beta x_{it} + u_i + v_t + \varepsilon_{it}, \quad \varepsilon_{it} \sim N(0, \sigma). \tag{21.2}$$

This is applicable when the time is a counfounder that affects both the treatment and the outcome. For example, due to cultural shift, the number of marriages increases with time. And due to the inflation, the wage also increases with time. The graphical model of our example is shown below:



To fit the fixed effects model with `stan_glmer`, simply add `(1 | year)` to the formula.

```
fit_2 <- stan_glmer(wage ~ married + union + exper
                    + (1 | nr) + (1 | year),
                    data=wage97, seed=0, refresh=0)

print(fit_2, digits=3)
```

```
stan_glmer
 family:       gaussian [identity]
 formula:      wage ~ married + union + exper + (1 | nr) + (1 | year)
 observations: 3115
------
            Median MAD_SD
(Intercept)  1.673  0.124
married      0.074  0.020
union        0.108  0.022
exper       -0.013  0.015

Auxiliary parameter(s):
      Median MAD_SD
sigma 0.349  0.005
```

```
Error terms:
 Groups   Name        Std.Dev.
 nr       (Intercept) 0.3653
 year     (Intercept) 0.2202
 Residual             0.3486
Num. levels: nr 429, year 8

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

With this new model, the average effect of marriage on wage decreases from 8% to 7.4%, but it is still significant.

We can use `ranef` to inspect the random effects as before.

```
ranef(fit_2)$nr[1:5, ]
```

```
[1] -0.3381000  0.0525898  0.1198533  0.3938407 -0.1562601
```

We can look at the time's random effects as well.

```
ranef(fit_2)$year[1:5, ]
```

```
[1] -0.26494327 -0.15934185 -0.09733085 -0.03082152  0.03349460
```

And we can see that the effect increases by the years.

21.3 Assumptions and Cautions

Ignorability. For the coefficient of the treatment ot be a valid estimate of the treatment effect, we have to assume independent conditional to the grouping variable—in this case the unit i—and pre-treatment covariates x. That is,

$$y^0, y^1 \perp z | i, x.$$

If the model has time effects, we have to condition on the time variable as well.

Confounders must be constant over time. The inclusion of the time-invariant variable u_i imposes that confounding must remain constant at all time. Any unobserved confounders that are changing over time will not be detected by the fixed effects model.

No reverse causality. The problem of assuming a wrong causal direction usually comes up in panel data. For example, instead of marriage having an effect on the wage, it might be the other way around, as people with more

wages have more chance of getting married. As another example. when one considers between the police spending and crime rate, the causal effect can go both ways: higher crime rate can cause more police spending, or having more police spending actually reduces the crime rate. Review of previous research is a common way to find the right direction, as well as asking domain experts.

Chapter 22

Synthetic control

In difference-in-differences, we saw that it is possible to estimate the treatment effect if the treated and controlled units are similar before applying the treatment. But more often than not, we might not find a controlled unit that exactly matches the treated unit that, we have, especially when there are many confounders to adjust for.

The idea of *synthetic control* is to make a fake unit that resembles the pre-treatment treated unit by combining the existing controlled units.

22.1 Example: study of the effect of taxation on cigarette consumption

The method of synthetic control was demonstrated in a study of the effect of cigarette taxation on its consumption (Abadie, Diamond, and Hainmueller 2010). One could argue that imposing the tax would decrease cigarette consumption. On the other hand, since cigarettes are addictive, the consumption would not decrease by much.

In particular, Abadie, Diamond, and Hainmueller studied the effect of *Proposition 99* which imposes several restrictions on cigarette sales in California since 1988: a 25-cent tax per cigarette pack, a ban on cigarette vending machines in public areas accessible by juveniles, and a ban on the sale of single cigarettes.

To study the effect of Proposition 99 using synthetic control, the authors collected the data of cigarette consumption in California and other states, before and after imposing the statute. The data `prop99.csv` contains the information regarding cigarette sales and taxes across all states from 1970 to 2000, on which we shall perform a couple of preprocessing steps:

1. Add a column that represents states by numbers.

2. There were several states that imposed similar cigarette restrictions. These states do not represent those in the control group, so we have to remove them.
3. There are several information regarding Proposition 99 in the `SubMeasureDesc` column, but we will only use the cigarette consumption as the outcome.

```
library(Synth)
```

```
# Load and preprocess the cigarette consumption data
prop99 <- read.csv("data/prop99.csv")

# Add a column that represents states by numbers
prop99$state <- as.numeric(factor(prop99$LocationDesc))

# Remove states that imposed similar cigarette restrictions
bad_states <- c('Massachusetts', 'Arizona', 'Oregon',
                'Florida', 'Alaska', 'Hawaii', 'Maryland',
                'Michigan', 'New Jersey', 'New York',
                'Washington', 'District of Columbia')
prop99 <- prop99[
    prop99$SubMeasureDesc=="Cigarette Consumption (Pack Sales Per Capita)" &
    !(prop99$LocationDesc %in% bad_states),
    ]

prop99 <- prop99[, c("state", "LocationDesc", "Year", "Data_Value")]

head(prop99)
```

```
   state LocationDesc Year Data_Value
2      1      Alabama 2014       61.7
20     4     Arkansas 2014       54.4
26     5   California 2014       22.7
32     6     Colorado 2014       36.7
38     7  Connecticut 2014       30.1
44     8     Delaware 2014       71.2
```

We also need additional information about these states in order to "match" pre-intervention California with the other states. We will use the `smoking.rda` data which contains cigarette sales `cigsale` , average log of income `lnincome` , beer sales `beer` , proportion of 15-24 age group in the population `age15to24` , and the cigarette's retail price `retprice`.

```r
# Load the state data
load("data/smoking.rda")

head(smoking)
```

```
  state year cigsale lnincome beer age15to24 retprice
1     1 1970    89.8       NA   NA 0.1788618     39.6
2     1 1971    95.4       NA   NA 0.1799278     42.7
3     1 1972   101.1 9.498476   NA 0.1809939     42.3
4     1 1973   102.9 9.550107   NA 0.1820599     42.1
5     1 1974   108.2 9.537163   NA 0.1831260     43.1
6     1 1975   111.7 9.540031   NA 0.1841921     46.6
```

Now we join these two dataframes by the state numbers and the years. We also make a new variable `allstates` that stores the remaining 39 states.

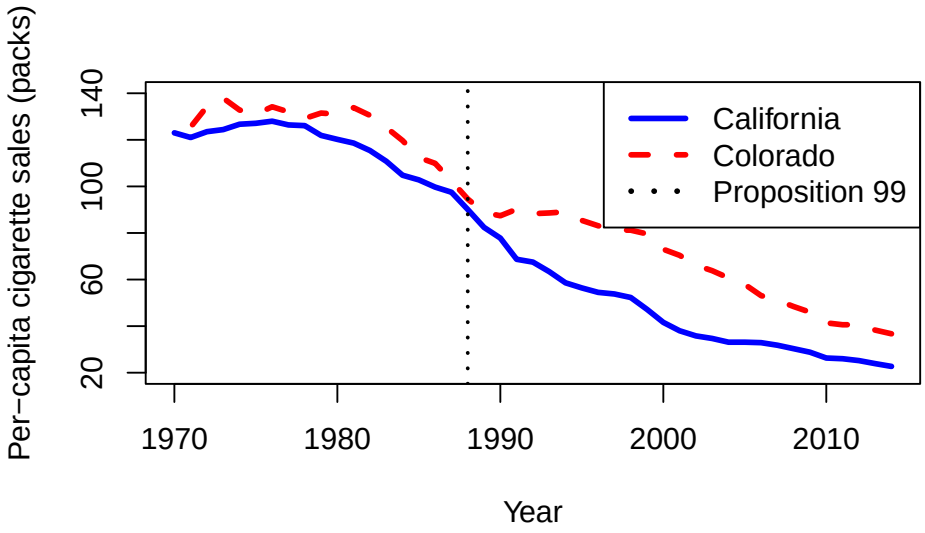
```r
# Join the two dataframes
colnames(smoking)[2] <- "Year"
prop99_full <- merge(prop99, smoking, by=c("state", "Year"))

# Obtain the list of states
allstates <- unique(prop99_full$LocationDesc)

head(prop99_full)
```

```
  state Year LocationDesc Data_Value cigsale lnincome beer age15to24 retprice
1     1 1970      Alabama       89.8    89.8       NA   NA 0.1788618     39.6
2     1 1971      Alabama       95.4    95.4       NA   NA 0.1799278     42.7
3     1 1972      Alabama      101.1   101.1 9.498476   NA 0.1809939     42.3
4     1 1973      Alabama      102.9   102.9 9.550107   NA 0.1820599     42.1
5     1 1974      Alabama      108.2   108.2 9.537163   NA 0.1831260     43.1
6     1 1975      Alabama      111.7   111.7 9.540031   NA 0.1841921     46.6
```

Let us visualize and compare the cigarette consumption in California and Colorado. We also plot a vertical line that splits between the pre-intervention period (1970-1987) and the post-intervention period (1988-2000).

```r
ca_data <- prop99[prop99$LocationDesc == "California", ]
co_data <- prop99[prop99$LocationDesc == "Colorado", ]

plot(ca_data$Year, ca_data$Data_Value, type = "l",
     ylab = "Per-capita cigarette sales (packs)",
     xlab = "Year", ylim = c(20, 140),
     col = "blue", lwd = 3)
```

```
lines(co_data$Year, co_data$Data_Value, lty = 2,
      col = "red", lwd = 3)
abline(v = 1988, lty = 3, lwd = 2)
legend("topright",
       legend = c("California",
                  "Colorado",
                  "Proposition 99"),
       col = c("blue", "red", "black"),
       lty = 1:3, lwd = 3)
```



The trends of the cigarette consumption between these two states are similar up until 1988. After that, there is a gap which might be a result of cigarette taxation in California.

22.2 The method of synthetic control

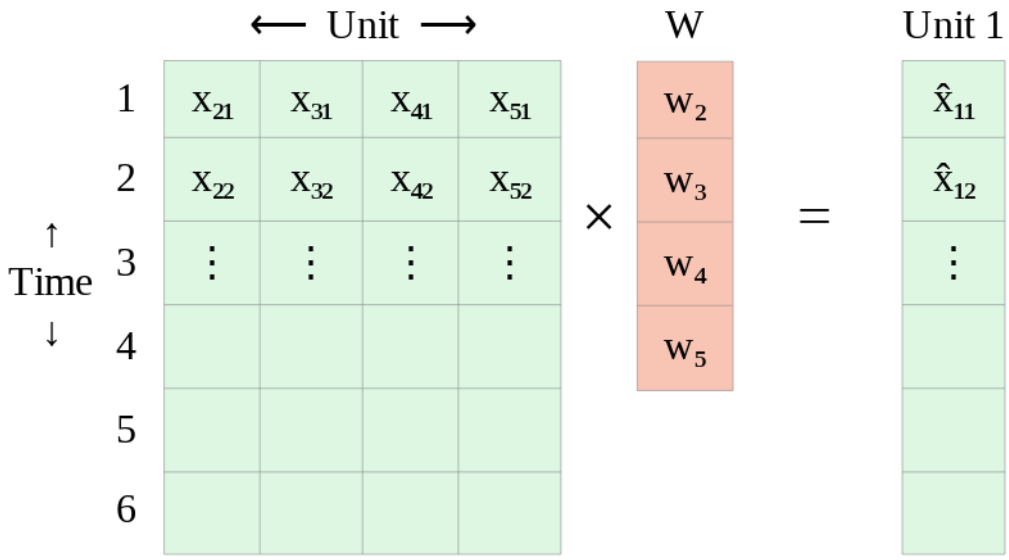
We now explain the method of constructing a new controlled unit that has similar features to those of the treated unit before the intervention. Let us start with a simple case of only one covariate x. Here, we introduce some notations:

- Suppose that there are J units: Unit 1 is the treated unit and unit $2, \dots, J$ are the controlled units.
- Suppose that the outcomes were observed at time $t = 1, \dots, T$, and the effect of the intervention occurred at time $T_0 < T$.
- y_{jt} is the outcome of j-th unit at time t.
- x_{jt} is the covariate of j-th unit at time t.

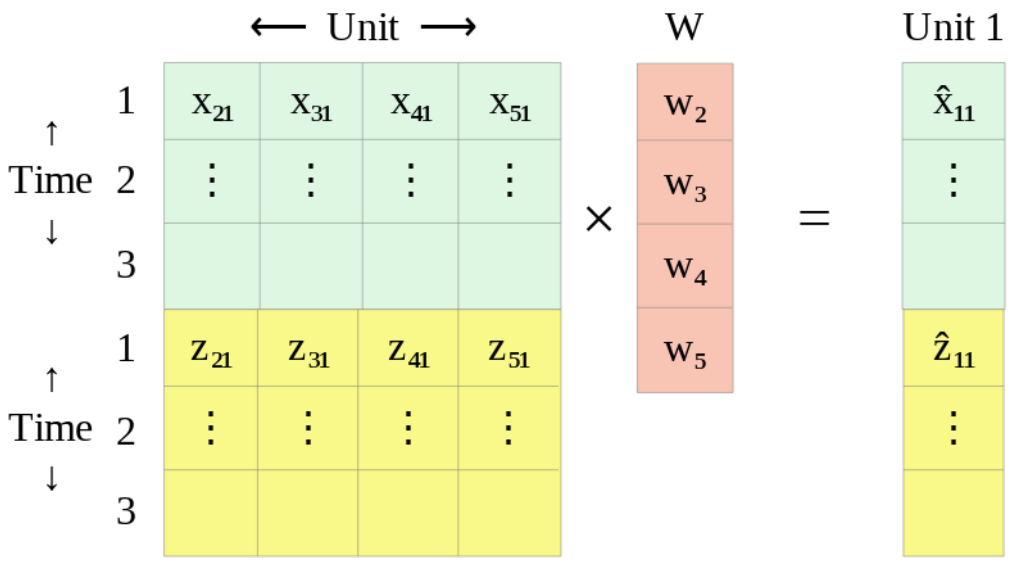
The idea is to construct Unit 1's *synthetic covariate* $\hat{x}_{1t}$ as a linear combination of the controlled units' covariates:

$$\hat{x}_{1t} = w_2 x_{2t} + ... + w_J x_{Jt}; \qquad t = 1, ..., T$$

where $w_2, ..., w_J$ are positive weights that sum to one. Our goal now is to find the combination of weights so that Unit 1's synthetic covariate $\hat{x}_{1t}$ is "close" to its actual covariate x_{1t}; this problem can be cast as a linear regression of x_{1t} on $x_{2t}, ..., x_{Jt}$, with $w_2, ..., w_J$ as the coefficients. The diagram below illustrates our regression problem:



Suppose that we have an additional covariate z, we can stack them next to x as follows:



Now suppose that we have m covariates: $x_{jt}^1, \ldots, x_{jt}^m$, which give rise to a synthetic covariate $\hat{x}_{1t}^k = \sum_{j=2}^J w_j x_{jt}^k$ for $k = 1, \ldots, m$. We can cast the linear regression as minimizing the least-squares objective with respect to $w_2, \ldots, w_J$.

$$\min_{w_2,\ldots,w_J} \sum_{k=1}^m \sum_{t=1}^T \left(x_{1t}^k - \sum_{j=2}^J w_j x_{jt}^k \right)^2$$
$$\text{subject to} \quad w_2 + \ldots + w_J = 1.$$

However, some covariates might be more important than the others in predicting the outcome; for example, one of our covariates is the cigarette sales, which should be more predictive of the cigarette consumption than the beer sales. We thus multiply the term associated with the k-th covariate by a positive weight v_k to reflect its relative importance:

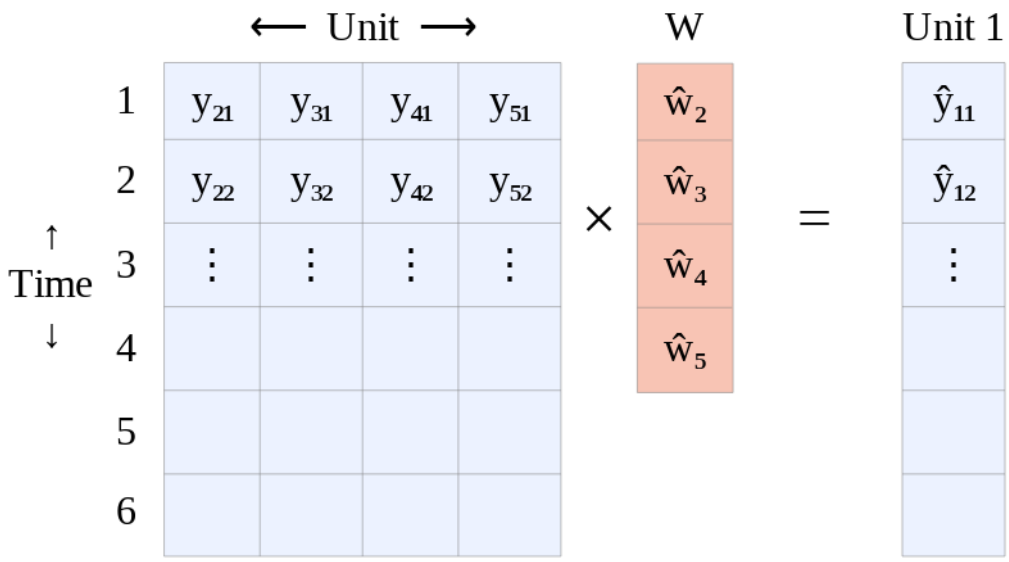
$$\min_{w_2,\ldots,w_J} \sum_{k=1}^m \sum_{t=1}^T v_k \left(x_{1t}^k - \sum_{j=2}^J w_j x_{jt}^k \right)^2 \tag{1}$$
$$\text{subject to} \quad w_2 + \ldots + w_J = 1. \tag{22.1}$$

We will discuss on how to choose $v_1, \ldots, v_m$ in a moment, but let us assume for now that these weights are known.

Solving (1) yields a solution $\hat{w}_2, \ldots, \hat{w}_J$. We then use this solution to obtain Unit 1's synthetic outcomes $\hat{y}_{1t}$ as follows:

$$\hat{y}_{1t} = \hat{w}_2 y_{2t} + \ldots + \hat{w}_J y_{Jt}, \qquad t = 1, \ldots, T.. \tag{2}$$

The diagram below illustrates the relationship between Unit 1's synthetic and actual outcomes.



We can now continue our discussion on the choice of the weights $v_1, \ldots, v_m$. Since the pre-intervention outcomes of the synthetic control should be similar to those of Unit 1, we typically choose $v_1, \ldots, v_m$ that minimize the corresponding least-squares objective:

$$\min_{v_1,\ldots,v_m} \sum_{t=1}^{T_0-1} \left(y_{1t} - \sum_{j=2}^{J} \hat{w}_j y_{jt} \right)^2 \tag{3}$$

$$\text{where} \quad \hat{w}_2, \ldots, \hat{w}_J \text{ solve (1).} \tag{22.2}$$

Notice that the sum only consists of the pre-intervention data. After solving (3) for $v_1, \ldots, v_m$, we obtain the associated $\hat{w}_2, \ldots, \hat{w}_J$—inserting these into (2) yields our estimate of Unit 1's post-intervention counterfactual outcome $\hat{y}_{1t}, t \geq T_0$. We can then estimate the causal effect using:

$$\hat{\tau}_{1t} = y_{1t} - \hat{y}_{1t}, \qquad t = T_0, \ldots, T. \tag{4}$$

22.3 Synthetic control in R using the `Synth` package

Now we continue our example of synthetic control in R. We already have the preprocessed dataframe `prop99_full` which contains both the covariates and the outcomes over 39 states. We can then use the `dataprep` function provided in the `Synth` package to split the data into dataframes of the covariates and the outcomes for synthetic control. For simple usage, we define a wrapper function named `prepare_data` that only asks for the name of the treated unit. Full descriptions of the parameters of `dataprep` can be found here.

```
# Prepare data for synthetic control of a specified state
prepare_data <- function(state) {
    return(
        dataprep(
            foo = prop99_full,
            predictors = c("cigsale", "lnincome", "beer", "age15to24"),
            predictors.op = "mean",
            time.predictors.prior = 1970:1987,
            dependent = "Data_Value",
            unit.variable = "state",
            unit.names.variable = "LocationDesc",
            time.variable = "Year",
            treatment.identifier = state,
            controls.identifier = allstates[!allstates == state],
            time.optimize.ssr = 1970:1987,
            time.plot = 1970:2000
        )
    )
}

prop99_prep <- prepare_data("California")
```

To perform the method of synthetic control, we simply call the `synth` function on the prepared data.

```
# Perform synthetic control
prop99_synth <- synth(data.prep.obj = prop99_prep)
```

After that, we call `synth.tab` on the output and the actual data to create tables summarizing the optimal weights $\hat{v}_1, \dots, \hat{v}_m$ and $\hat{w}_2, \dots, \hat{w}_J$ and the corresponding minimum values of the lease-squares (1) and (3).

```r
# Table summarizing the result
prop99_tab <- synth.tab(prop99_synth, prop99_prep)

prop99_tab$tab.w
```

```
   w.weights     unit.names unit.numbers
1      0.000        Alabama            1
4      0.000        Georgia           11
6      0.000          Idaho           13
7      0.000       Illinois           14
8      0.000        Indiana           15
11     0.000           Iowa           16
13     0.000         Kansas           17
14     0.000       Kentucky           18
15     0.000      Louisiana           19
16     0.002          Maine           20
17     0.000      Minnesota           24
18     0.000    Mississippi           25
19     0.000       Missouri           26
20     0.000        Montana           27
24     0.000       Nebraska           28
25     0.000         Nevada           29
26     0.000  New Hampshire           30
27     0.429     New Mexico           32
28     0.508 North Carolina           34
29     0.000   North Dakota           35
30     0.000           Ohio           36
32     0.000       Oklahoma           37
34     0.061   Pennsylvania           39
35     0.000       Arkansas            4
36     0.000       Colorado            6
37     0.000    Connecticut            7
39     0.000       Delaware            8
```

We can see that North Carolina is the most representative state in the synthetic control, while New Mexico is the second most.

Unfortunately, `synth` only outputs the weights and the least-squares minimum. To compute the synthetic outcome (2), we have to multiply the controlled's outcome matrix, which is stored in the prepared data's `Y0plot` column, and the vector of optimal weights, stored in the result's `solution.w` column.

```r
print(prop99_prep$Y0plot[1:5, 1:8])
```

```
        1    11    13    14    15    16    17    18
```

```
1970  89.8 100.3 124.8 120.0 155.0 109.9 102.4 124.8
1971  95.4 104.1 125.5 117.6 161.1 115.7 108.5 125.6
1972 101.1 103.9 134.3 110.8 156.3 117.0 126.1 126.6
1973 102.9 108.0 137.9 109.3 154.7 119.8 121.8 124.4
1974 108.2 109.7 132.8 112.4 151.3 123.7 125.6 131.9
```

```
print(prop99_synth$solution.w[1:8])
```

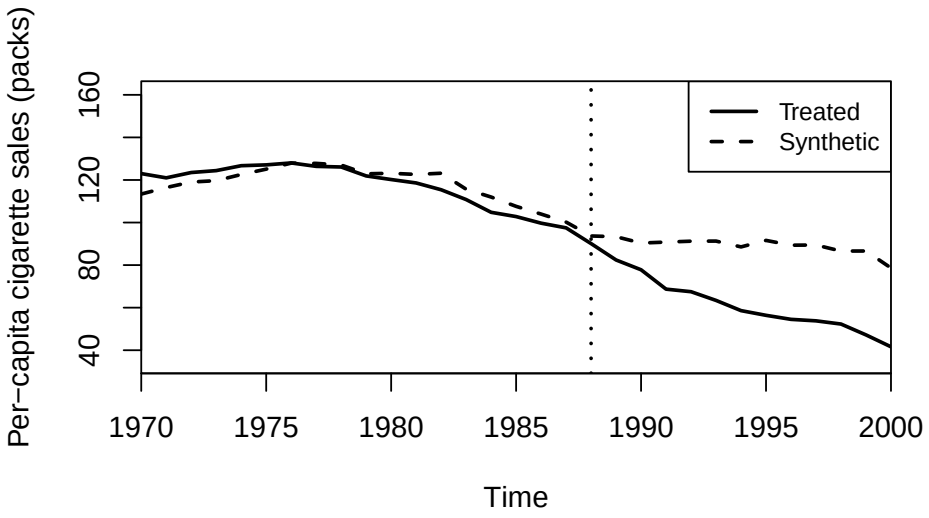
```
[1] 2.518675e-06 8.593394e-06 4.654868e-06 3.732791e-06 5.340892e-08
[6] 1.424996e-05 1.815872e-06 1.289167e-06
```

```
# Calculate the outcomes of the synthetic control
synth_control <- prop99_prep$Y0plot%*%prop99_synth$solution.w

tail(synth_control)
```

```
     w.weight
1995 91.60042
1996 89.35529
1997 89.43415
1998 86.59770
1999 86.62141
2000 78.64532
```

To plot both the actual outcome and the synthetic outcome, we can use the `path.plot` function. Here, we also plot a vertical line that indicates the intervention (the enactment of Proposition 99) in 1988.

```
# Plot the outcomes of the treated unit and the synthetic control
path.plot(prop99_synth, prop99_prep,
          tr.intake = 1988,
          Ylab = "Per-capita cigarette sales (packs)")
```



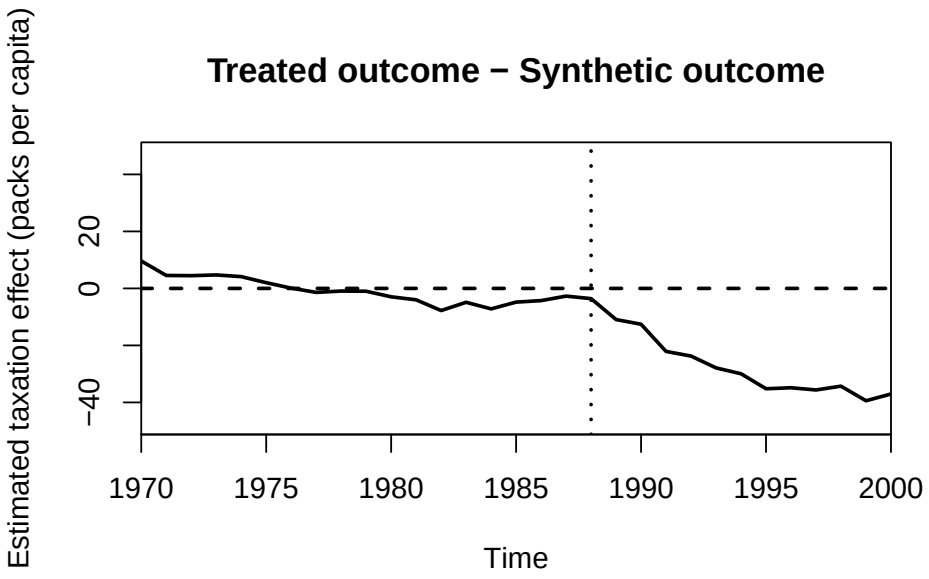
To compute an estimate of the treatment effect using (4), we subtract the treated's outcome, stored in the prepared data's `Y1plot` column, by the synthetic control's outcome.

```
# Calculate an estimate of the treatment effect
tax_effects <- prop99_prep$Y1plot - synth_control

tail(tax_effects)
```

```
            5
1995 -35.20042
1996 -34.85529
1997 -35.63415
1998 -34.29770
1999 -39.42141
2000 -37.04532
```

We can also plot our estimate of the treatment effect over time using `gaps.plot` function.

```
gaps.plot(prop99_synth, prop99_prep,
          tr.intake = 1988,
          Main = "Treated outcome - Synthetic outcome",
          Ylab = "Estimated taxation effect (packs per capita)")
```



22.4 Permutation test

Without a frame of reference, it is not clear if our estimate displayed above is statistically significant. To show this, we can use a permutation test, whose hypotheses are:

H_0 : The effects of treatment on the treated and the controlled are the same.
H_1 : The effect of treatment on the treated is less than that on the controlled.

To perform the test, we follow the same procedure as above for all other units to obtain estimates of the *placebo effects* and see if the estimate of the treatment effect is sufficiently smaller than that of the placebo effects. In addition, we only consider units whose pre-intervention synthetic outcomes are close to the actual outcomes in terms of the *mean-squared error* (MSE):

$$\text{MSE}_j = \frac{1}{T_0}\sum_{t=1}^{T_0}(y_{jt} - \hat{y}_{jt})^2.$$

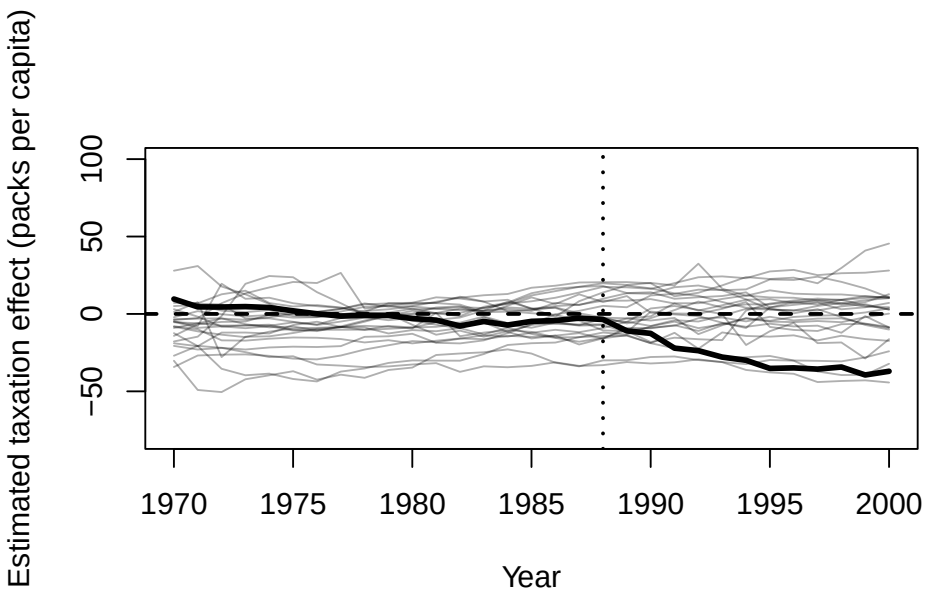
In particular, we discard any state whose pre-intervention error is greater than 1600.

```
prop99_placebos <- data.frame(row.names=1970:2000)
```

```
for (state in allstates) {
    # Estimate the treatment effect for the state
    prop99_prep <- prepare_data(state)
    prop99_synth <- synth(data.prep.obj = prop99_prep)
    synth_control <- prop99_prep$Y0plot%*%prop99_synth$solution.w
    tax_effects <- prop99_prep$Y1plot - synth_control

    # Only consider state with small pre-intervention error
    if(mean(tax_effects[1:18]^2) < 1600) {
        prop99_placebos[, state] <- tax_effects[, 1]
    }
}
```

To visualize the permutation test, we plot the estimate of treatment effect and placebo effects.

```
plot(1970:2000, prop99_placebos$California, xlab = "Year",
     ylab = "Estimated taxation effect (packs per capita)",
     ylim = c(-80, 100), type = "l", lwd = 3)

for (state in colnames(prop99_placebos)) {
    lines(1970:2000, prop99_placebos[, state], col='#00000050')
}
abline(a = 0, b = 0, lty = 2, lwd = 2)
abline(v = 1988, lty = 3, lwd = 2)
```



The good thing about this test is that the p-value—the proportion of the controlled units whose estimates of the placebo effects are smaller than the treated unit's synthetic control estimate—can be easily computed. For example, let us compute the p-value of the estimate in year 2000.

```
effect2000 <- prop99_placebos["2000", ]
p_value <- mean(effect2000 < effect2000$California)

print(p_value)
```

```
[1] 0.04166667
```

Chapter 23

Use cases of causal inference in industry

This section is for keeping track of blog articles and papers related to use cases of causal inference in industry. Many of these methods use meta-learners (Künzel et al. 2019) or doubly-robust estimators (Funk et al. 2011) which are not covered in this course.

23.1 Matching

- At Uber, researchers studied the effect of using Uber Eats, in addition to Uber Rides, on the amount spent on Uber Rides. They used propensity score matching with 100+ covariates to match each user who used both Uber Eats and Uber Rides to a user who only used Uber Rides. Their results suggest that using UberEats drives up spending on Uber Rides (Python tutorial).

23.2 Instrumental variables

- Researchers at Twitch studied the effect of number of friends on the user's likelihood to return to the site. Their instrumental variable is the random assignment of receiving a prompt to add more friends (Forter 2017).
- Researchers at Roblox studied the impact of the Avatar Shop on the community engagement. Their instrumental variable is the random assignment of getting a recommendation for items in the Avatar Shop (Kharel 2021).
- Farre-Mensa, Hegde, and Ljungqvist (2017) studied the impact of having patents on a startup's subsequent growth. The judges are the examiners of

the patents, with different levels of leniency, and the instrumental variable is the random assignment of judges to the patent applications.
- Researchers at TripAdvisor studied the effect of being a membership on user engagement. Since it is not possible to force users to comply and become members, they instead used a instrumental variable design in which a randomized group of users were provided with a single-click sign-up button, which was much easier than the previous sign-up process. In this study, the instrumental variable was whether the user was offered the easier sign-up option (Python tutorial).

23.3 Difference-in-differences

- At Spotify, researchers studied the treatment effects of adopting a streaming service on the total music consumption. The study was performed using difference-in-differences on a unique panel data that contains individual-level music consumption. Since the sampled consumers may not be representative of the larger population of potential adopters, they instead studied local average treatment effects (LATE) among those consumer segments who adopt streaming (Datta, Knox, and Bronnenberg 2018).

23.4 Panel data

- Researchers at Uber studied the impact of increased pricing during high demand on the supply of Uber rides. They fitted a fixed effect model on panel data of drivers' trips, using the hourly fare multiplier (depending of the supply and demand) as the treatment variable, an indicator of whether the trip was the driver's last one of the session as the outcome variable, and individual driver, day of week, hour of day, and region of city as fixed effects. The results of the model suggest that a surge in hourly fares significantly increases the supply of rides on the Uber platform (Chen 2016).

23.5 Synthetic control

- Researchers at Spotify proposed Bayesian structural time-series model (Brodersen et al. 2015) to constructs a counterfactual artist popularity outcome using a set of synthetic controls. Their findings suggest that releasing a new track has a positive impact on the popularity of other tracks by the same artist, and other related and competing artists also benefit from a new track release (Mehrotra, Bhattacharya, and Lalmas 2020).

Part IV

Conformal prediction

In the remaining part of the course, we shall focus on models' predictions with uncertainties. More precisely, given a predictive model and a new data point, our goal is to construct an interval that has a high probability of containing the outcome associated with the data point. To do this, we will use *conformal prediction*, a frequentist method of constructing a prediction interval that relies on minimal assumption on the data distribution. In a sense, conformal prediction is an exact opposite of predictive Bayesian inference, which heavily relies on distributional assumptions of the model's parameters (through the prior) and that of the data (through the likelihood). Of course, if the data distribution exactly matched our assumptions, the posterior predictive distribution would give us an accurate prediction interval. On the other hand, if our Bayesian model was misspecified, then the conformal prediction would give us a better prediction interval.

In the following chapters, we will cover fundamental ideas of conformal prediction in the context of regression and classification. We will discuss its computational issue and, introduce a couple of methods that are designed to solve this issue.

Chapter 24

Full & split conformal prediction

There are many situations where a wrong prediction can lead to costly consequences. For example, a wrong prediction on a patient's health condition after receiving the treatment could lead to a fatal outcome. Thus it is important to understand the reliability and uncertainty of our predictions.

In the first few chapters, we have talked about Bayesian regression where we used the posterior predictive distribution to measure the uncertainty. One downside of such method is the need to specify the distribution that generates the data in the form of the likelihood. In this chapter, we introduce a method with minimal distributional assumptions to estimate prediction intervals that exploit the exchangeability of the data points.

24.1 Review: quantile

Let us first review the concept of quantile. Given a random variable X with continuous density, the quantile function $\mathrm{Q}(\beta; X)$ is the smallest value x of X such that $\Pr[X \leq x] = \beta$. But if X is discrete or its density is not continuous, such x might not exist, so we have to relax the definition a bit. More precisely, the quantile function is:

$$\text{Quantile}(\beta, X) = \inf\{x : \Pr[x \leq X] \geq \beta\}.$$

Suppose that we observe data $x = (x_1, \ldots, x_n)$, we can define the *empirical quantile* function as follows:

$$\text{Quantile}(\beta, x) = \min\left\{x_i : \frac{\#\{j : x_j \leq x_i\}}{n} \geq \beta\right\}. \tag{24.1}$$

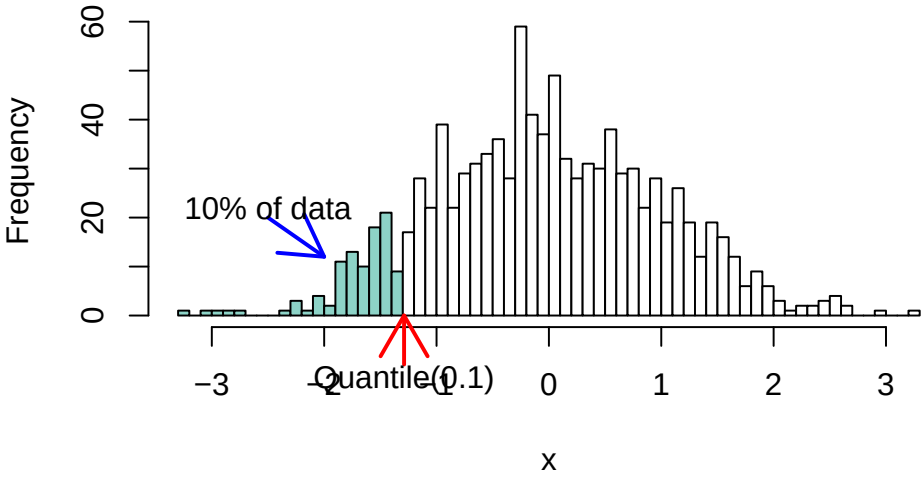
In other words, one can compute the quantile by sorting the observed values in ascending order and finding x_i such that the proportion of values less than or equal to x_i is *just* above β.

To compute quantiles in R, we can use the `quantile` function. Below is an example of calculating $\text{Quantile}(0.1, x)$ where x is a sample from the standard normal distribution.

```
set.seed(0)

x <- rnorm(1000)
q <- quantile(x, 0.1)

print(q)
```

```
     10%
-1.28896
```

When plotting a histogram of this data, the area associated with the values smaller than the quantile should take up approximately 10% of the total area.



24.2 Quantile Lemma

Exchangeability. Our method for constructing prediction intervals relies on the main assumption that the random variables of interest are exchangeable, that is, the joint distribution of the random variables $R_1, \ldots, R_n$ does not change upon any permutation. Thus one must be careful and check whether the data

at hand satisfies this assumption. For example, data points that were generated from a distribution that shifts over time are not exchangeable.

Suppose that $R_1, \ldots, R_n$, together with a new variable R_{new} are exchangeable, then the following *quantile lemma* tells us that it is possible to know something about R_{new} in relation to $R_1, \ldots, R_n$ without knowing its underlying probability distribution. The lower bound, which is a standard result in conformal prediction, is due to Vovk, Gammerman, and Shafer (2005) and the upper bound is due to Lei et al. (2018).

Lemma 24.1 (Quantile Lemma)**.** *Denote* $R_{1:n} = \{R_1, \ldots, R_n\}$. *If* $R_1, \ldots, R_n, R_{new}$ *are exchangeable, then for any* $\beta \in (0, 1)$*,*

$$\Pr\left[R_{new} \leq \text{Quantile}(\beta, R_{1:n} \cup \{\infty\})\right] \geq \beta. \tag{24.2}$$

If in addition the probabilities of ties are zero i.e. $\Pr[R_i = R_j] = 0$ *for all* $i \neq j$*, then we have an upper bound:*

$$\Pr\left[R_{new} \leq \text{Quantile}(\beta, R_{1:n} \cup \{\infty\})\right] \leq \beta + \frac{1}{n+1}. \tag{24.3}$$

Proof. We go over the proof in five steps.

1. Let $q = \text{Quantile}(\beta, R_{1:n} \cup \{\infty\})$. If we modify the data $R_{1:n} \cup \{\infty\}$ by moving ∞ to any other value larger than q, the β-quantile is still unchanged. Specifically, we move ∞ to R_{new} so that

   $$R_{\mathsf{new}} > \text{Quantile}(\beta, R_{1:n} \cup \{\infty\}) \iff R_{\mathsf{new}} > \text{Quantile}(\beta, R_{1:n} \cup \{R_{\mathsf{new}}\}),$$

   which is equivalent to

   $$R_{\mathsf{new}} \leq \text{Quantile}(\beta, R_{1:n} \cup \{\infty\}) \iff R_{\mathsf{new}} \leq \text{Quantile}(\beta, R_{1:n} \cup \{R_{\mathsf{new}}\}).$$

2. From the definition of empirical quantile (Equation 24.1),

   $$\begin{aligned} R_{\mathsf{new}} \leq \text{Quantile}(\beta, R_{1:n} \cup \{R_{\mathsf{new}}\}) &\iff \frac{\#\{j : R_j \leq R_{\mathsf{new}}\}}{n+1} \leq \beta \\ &\iff \text{rank}(R_{\mathsf{new}}) \leq \lceil \beta(n+1) \rceil. \end{aligned}$$

3. Let us detour a bit and consider $\Pr[\text{rank}(R_{\mathsf{new}}) = r]$ for any $r \in \{1, \ldots, n+1\}$. With $I = \{1, \ldots, n, \mathsf{new}\}$, we have

$$\sum_{i \in I} \Pr[\text{rank}(R_i) = r] \geq \Pr[\text{rank}(R_i) = r \text{ for some } i \in I] = 1. \tag{24.4}$$

By exchangeability, $\Pr[\text{rank}(R_i) = r]$ is the same for all $i \in I$, so Equation 24.4 implies $\Pr[\text{rank}(R_{\text{new}}) = r] \geq \frac{1}{n+1}$ for all r.

4. Continuing from Step 2., we obtain the lower bound in Equation 24.2:

$$\begin{aligned}
\Pr\left[\text{rank}(R_{\text{new}}) \leq \lceil \beta(n+1) \rceil\right] &= \sum_{r=1}^{\lceil \beta(n+1) \rceil} \Pr[\text{rank}(R_{\text{new}}) = r] \\
&\geq \frac{\lceil \beta(n+1) \rceil}{n+1} \\
&\geq \beta.
\end{aligned}$$

5. If the probabilities of ties are zero, then the distribution of the rank of R_{new} is exactly Uniform$\{1, \ldots, n+1\}$, from which we can compute the exact probability and obtain the upper bound in Equation 24.3:

$$\Pr\left[\text{rank}(R_{\text{new}}) \leq \lceil \beta(n+1) \rceil\right] = \frac{\lceil \beta(n+1) \rceil}{n+1} \leq \beta + \frac{1}{n+1}.$$

□

24.3 Full conformal prediction

The main idea of constructing the prediction intervals is to let $R_1, \ldots, R_n, R_{\text{new}}$ be a "non-conformity score" of our data points $(x_1, y_1), \ldots, (x_n, y_n), (x_{\text{new}}, ?)$, from which we use the quantile lemma to find the prediction intervals of the outcome associated with x_{new}, say y_{new}.

Let us consider the problem of predicting a child's IQ score from the mother's education, IQ score, years of employment and age. We load the `KidIQ` dataset and make a new hypothetic data point where the child IQ has not been yet observed.

```
kidiq <- read.csv("data/kidiq.csv")

n <- nrow(kidiq)
new_data <- data.frame(kid_score = NA,
                       mom_hs = 0,
                       mom_iq = 90,
                       mom_work = 1,
```

```
                        mom_age = 20)

head(kidiq)
```

```
  kid_score mom_hs    mom_iq mom_work mom_age
1        65      1 121.11753        4      27
2        98      1  89.36188        4      25
3        85      1 115.44316        4      27
4        83      1  99.44964        3      25
5       115      1  92.74571        4      27
6        98      0 107.90184        1      18
```

Our goal is to construct a 95% prediction interval of the child's IQ. Here, we introduce our first method, namely the *full conformal prediction* (Vovk, Gammerman, and Shafer 2005) which utilizes the quantile lemma to obtain a prediction inverval with any model of choice (for example, a linear regression model). There are mainly two variants of full conformal predictions.

24.3.1 Deleted full conformal prediction

Let $\mathcal{D}_{1:n} = \{(x_1, y_1), \dots, (x_n, y_n)\}$ be the observed data and $(x_{\mathsf{new}}, ?)$ be a new data point. We can compute the prediction intervals for the outcome y_{new} associated with x_{new} as follows:

For each possible value of y

1. Add a new point (x_{new}, y) to the dataset.
2. For each $i \in \{1, \dots, n, \mathsf{new}\}$,

   2.1. Fit our predictive model on all but the i-th data point: $\mathcal{D}_{1:n} \cup \{(x_{\mathsf{new}}, y)\} - \{(x_i, y_i)\}$. Let $\hat{\mu}^y_{-i}$ be the fitted model.

   2.2. Compute the non-conformity score $R^y_i = |y_i - \hat{\mu}^y_{-i}(x_i)|$.
3. Sort $R^y_1, \dots, R^y_n$ in increasing order: $R^y_{(1)}, \dots, R^y_{(n)}$.
4. Let $k = \lceil \beta(n+1) \rceil$. We keep y if $R^y_{\mathsf{new}} \leq R^y_{(k)}$ and discard it otherwise.

After we are done with all y, our prediction interval $C_\beta(x_{\mathsf{new}})$ consists of the values of y that we keep. In other words, $C_\beta(x_{\mathsf{new}}) = \{y : R^y_{\mathsf{new}} \leq R^y_{(k)}\}$.

$C_\beta(x_{\mathsf{new}})$ being a β-prediction interval is a simple consequence of the quantile lemma: let y_{new} be the actual outcome associated with x_{new}. Assuming that $(x_1, y_1), \dots, (x_n, y_n), (x_{\mathsf{new}}, y_{\mathsf{new}})$ are exchangeable, it directly follows from the quantile lemma that $C_\beta(x_{\mathsf{new}})$ is a β-prediction interval: since $R^{y_{\mathsf{new}}}_{(k)}$ is the β-quantile of $R^{y_{\mathsf{new}}}_{(1)}, \dots, R^{y_{\mathsf{new}}}_{(n)}$ (which are also exchangeable by their symmetric contruction), it follows from Equation 24.2 that

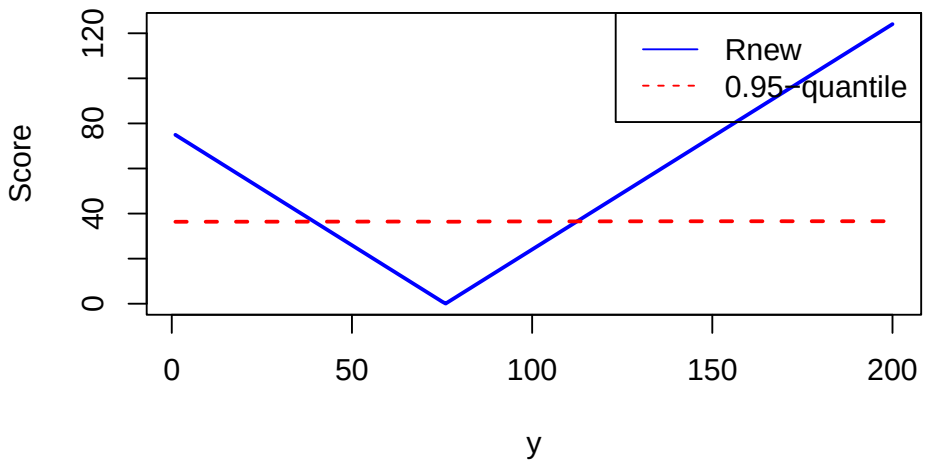
$$\Pr[y_{\text{new}} \in C_\beta(x_{\text{new}})] = \Pr\left[R_{\text{new}}^{y_{\text{new}}} \leq R_{(k)}^{y_{\text{new}}}\right] \geq \beta.$$

Below is an implementation of the procedure on the `KidIQ` data. Here, we assume that the possible values of IQ range from 1 to 200. To speed up the process, we convert the dataframe into a matrix and fit linear regression using a bare bone `.lm.fit` instead.

```
kidiq_mat <- as.matrix(kidiq)
kidiq_mat <- rbind(kidiq_mat, as.matrix(new_data))
X <- kidiq_mat[, -1]
X <- cbind(rep(1, nrow(X)), X)
Y <- kidiq_mat[, 1]

beta <- 0.95
k <- ceiling(beta * (n+1))

Rnew <- rep(NA, 200)
Rk <- rep(NA, 200)

for (y in 1:200) {
    Y[n+1] <- y
    R <- rep(NA, n+1)

    for (i in 1:(n+1)) {
        model <- .lm.fit(X[-i, ], Y[-i])
        yhat <- X[i, ] %*% coef(model)
        R[i] <- abs(Y[i] - yhat)
    }

    Rnew[y] <- R[n+1]
    Rk[y] <- sort(R[1:n])[k]
}
```

Let us plot R_{new}^y and R_k^y in order to find a prediction interval of y.

```
plot(1:200, Rnew, type = "l", lwd = 2, col = "blue",
     xlab = "y", ylab = "Score")
lines(1:200, Rk, lwd = 2, lty = 2, col = "red")
legend("topright", legend=c("Rnew", "0.95-quantile"),
       col=c("blue", "red"), lty=1:2)
```



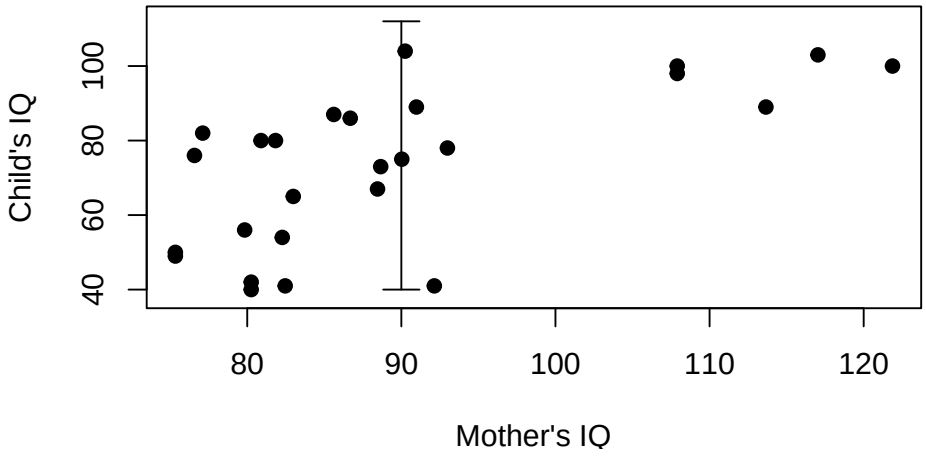
The prediction interval consists of all y's whose scores are below the β-quantile. We can also compute the upper and lower bound of the interval directly.

```
which_y_conform <- which(Rnew < Rk)
last <- length(which_y_conform)

lower <- which_y_conform[1]
upper <- which_y_conform[last]
cat("Prediction interval: [", lower, ",", upper, "]")
```

```
Prediction interval: [ 40 , 112 ]
```

To see how well the prediction interval covers the data points, we plot the data points that have similar characteristics as those of the new data point. Specifically, we take the data points consisting of children whose mothers were in the 16-24 age group, never graduated from high school (`mom_hs = 0`) and had been working for one year (`mom_work = 1`).

```
kidiq_subset <- kidiq[kidiq$mom_hs == 0 &
                      kidiq$mom_work == 1 &
                      abs(kidiq$mom_age - 20) < 5,]

plot(kid_score ~ mom_iq, data = kidiq_subset,
     ylab = "Child's IQ", xlab = "Mother's IQ",
     ylim = c(38, 113), pch = 19)
arrows(x0 = new_data$mom_iq, y0 = lower,
       x1 = new_data$mom_iq, y1 = upper,
       code = 3, angle = 90, length = 0.1)
```



The plot indicates that the interval has sufficient coverage over the data points.

24.3.2 Ordinary full conformal prediction

This variant of full conformal prediction fits on *all* data only once for each value of y. Thus it is a lot faster to run compared to the deleted variant.

Let $\mathcal{D}_{1:n} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be the observed data and $(x_{\text{new}}, ?)$ be a new data point.

For each possible value of y

1. Add a new point (x_{new}, y) to the dataset.
2. Fit our predictive model on $\mathcal{D}_{1:n} \cup \{(x_{\text{new}}, y)\}$. Let $\hat{\mu}^y$ be the fitted model.
3. For each $i \in \{1, \ldots, n, \text{new}\}$, compute the non-conformity score $R_i^y = |y_i - \hat{\mu}^y(x_i)|$.
4. Sort $R_1^y, \ldots, R_n^y$ in increasing order: $R_{(1)}^y, \ldots, R_{(n)}^y$.
5. Let $k = \lceil \beta(n+1) \rceil$. We keep y if $R_{\text{new}}^y \leq R_{(k)}^y$ and discard it otherwise.

After we are done with all y, our prediction interval $C_\beta(x_{\text{new}})$ consists of the values of y that we keep. In other words, $C_\beta(x_{\text{new}}) = \{y : R_{\text{new}}^y \leq R_{(k)}^y\}$.

```
beta <- 0.95
k <- as.integer(beta * (n+1))

Rnew <- rep(NA, 200)
Rk <- rep(NA, 200)

for (y in 1:200) {
```

```
    new_data[, "kid_score"] <- y

    model <- lm(kid_score ~ ., data = rbind(kidiq, new_data))
    R <- abs(residuals(model))

    Rnew[y] <- R[n+1]
    Rk[y] <- sort(R[1:n])[k]
}
```

From this, let us compute the prediction interval.

```
which_y_conform <- which(Rnew < Rk)
last <- length(which_y_conform)

lower <- which_y_conform[1]
upper <- which_y_conform[last]
cat("Prediction interval: [", lower, ",", upper, "]")
```

```
Prediction interval: [ 40 , 112 ]
```

There is little to no difference between the two predictions intervals. See Abad et al. (2022) and Vovk et al. (2019) for more discussions and experiments on these two variants.

24.4 Split conformal prediction

One downside of the full conformal prediction is that it requires fitting the predictive model as many times as the number of possible values of y. Alternatively, we can split the data into two sets: a *training set* to fit the model, and *a calibration set* to calculate non-conformity scores. The β-quantile of the scores is then used to obtain a prediction inverval as before. The fact that the training set is not involved in the scoring process has two implications:

- We only need to assume that the data points in the calibration set and the new data point are exchangeable,
- The model is fitted only once on the training set. In particular, we no longer have to fit the model iteratively over all possible values of y,

which lead to a faster computation at a cost of statistical efficiency since the model is only fitted on a part of the dataset. This method, referred to as *split conformal prediction*, can be performed as follows:

Let $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ be the observed data and $(x_{\mathsf{new}}, ?)$ be a new data point.

1. Split $\{1, \ldots, n\}$ into a training set Tr and a calibration set Cal.

2. Fit our model on $\{(x_i, y_i) : i \in \mathsf{Tr}\}$. Denote the fitted model by $\hat{\mu}$.
3. For each $j \in \mathsf{Cal}$, compute the (non-conformity) score $R_j = |y_j - \hat{\mu}(x_j)|$.
4. Sort $R_1, \ldots, R_n$ in increasing order: $R_{(1)}, \ldots, R_{(n)}$.
5. Let $k = \lceil \beta(n+1) \rceil$. The prediction interval is

$$C_\beta(x_{\mathsf{new}}) = \left[\hat{y}_{\mathsf{new}} - R_{(k)}, \hat{y}_{\mathsf{new}} + R_{(k)}\right].$$

Again, $C_\beta(x_{\mathsf{new}})$ being a β-prediction interval is a simple consequence of the quantile lemma: let y_{new} be the actual outcome associated with x_{new} and $R_{\mathsf{new}} = |y_{\mathsf{new}} - \hat{y}_{\mathsf{new}}|$. As before, there is greater or equal to β probability that $R_{\mathsf{new}} \leq R_{(k)}$, which is equivalent to $y_{\mathsf{new}} \in [\hat{y}_{\mathsf{new}} - R_{(k)}, \hat{y}_{\mathsf{new}} + R_{(k)}]$.

Here is an example of split conformal prediction on the `KidIq` dataset:

```
beta <- 0.95
m <- floor(n/2)
k <- ceiling(beta * (m+1))

calib_id <- sample(seq_len(n), size = m)
kidiq_train <- kidiq[-calib_id, ]
kidiq_calib <- kidiq[calib_id, ]

model <- lm(kid_score ~ ., data = kidiq_train)

yhat <- predict(model, newdata = kidiq_calib)
y <- kidiq_calib$kid_score
R <- abs(y - yhat)

R <- sort(R)[k]

ynew_hat <- predict(model, newdata = new_data)
lower <- ynew_hat - R
upper <- ynew_hat + R
cat("Prediction interval: [", lower, ",", upper, "]")
```

```
Prediction interval: [ 36.34356 , 111.7025 ]
```

The prediction interval is a bit wider than those of the full conformal predictions, which agree with our comment regarding the statistical efficiency of split conformal prediction at the beginning of the section.

Chapter 25

Jackknife+, CV+ and Quantile regression

A downside of the full conformal is the need to refit the predictive model every time a new data point is introduced, making it computational expensive. On the other hand, the split conformal is fast but less efficient in the amount of data used to fit to the model. Between these two extremes, we would like to find another method that can utilize each data point for both fitting and scoring, while being reasonably fast to compute; two of such methods are Jackknife+ and CV+ (Barber et al. 2021).

25.1 Jackknife+

A simple way to fix the data efficiency problem is by fitting the model and calibrating on the whole dataset. However, this method is likely to overfit, as the non-conformity scores on the training data are naturally smaller than those on the unseen data, resulting in a prediction interval that does not cover enough data points.

Jackknife is introduced to address this issue using leave-one-out fitting and calibration: let $\mathcal{D}_{1:n} = \{(x_1, y_1), \dots, (x_n, y_n)\}$ be the observed data. For each $i \in \{1, \dots, n\}$, we fit the model on all but one data point $\mathcal{D}_{1:n} - \{(x_i, y_i)\}$ to obtain a fitted model $\hat{\mu}_{-i}$. The non-conformity score is then

$$R_i = |y_i - \hat{\mu}_{-i}(x_i)|.$$

Let $\hat{\mu}$ be the model fitted on all observed data $\mathcal{D}_{1:n}$. Let x_{new} be a new data point. The 90% Jackknife prediction interval is similar to the one in the split conformal:

$$[\text{5th-perc. of } \{\hat{\mu}(x_{\text{new}}) - R_i\}, \text{95th-perc. of } \{\hat{\mu}(x_{\text{new}}) + R_i\}].$$

However, we are looking at percentiles of distances R_i from a fixed point $\hat{\mu}$, which might be too restrictive and result in not enough coverage.

Jackknife+. To solve Jackknife's issue, we simply use $\hat{\mu}_{-i}$ to predict the outcome of the new data point instead of $\hat{\mu}$. Here are the steps to perform Jackknife+ in full details:

Let $\mathcal{D}_{1:n} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be the observed data and $(x_{\text{new}}, ?)$ be a new data point.

For each $i \in \{1, \ldots, n\}$,

1. Fit the model on all but one data point $\mathcal{D}_{1:n} - \{(x_i, y_i)\}$. Let $\hat{\mu}_{-i}$ be the fitted model.
2. Compute the non-conformity score: $R_i = |y_i - \hat{\mu}_{-i}(x_i)|$.
3. Compute a lower and upper bound of $\hat{\mu}_{-i}$'s prediction interval for the new data point:

$$L_i = \hat{\mu}_{-i}(x_{\text{new}}) - R_i, \quad U_i = \hat{\mu}_{-i}(x_{\text{new}}) + R_i.$$

The Jackknife+ prediction interval is

$$C^{\text{JK+}}_{1-2\alpha}(x_{\text{new}}) = [\alpha\text{-quantile of } \{L_1, \ldots, L_n\}, (1-\alpha)\text{-quantile of } \{U_1, \ldots, U_n\}].$$

The following theorem from Barber et al. (2021) shows that this prediction interval has $1 - 2\alpha$ probability coverage. So, for example, to obtain a 90% prediction interval we would set $\alpha = 0.05$.

Theorem 25.1. *If $(x_1, y_1), \ldots, (x_n, y_n), (x_{new}, y_{new})$ are exchangeable, then*

$$\Pr[y_{new} \in C^{JK+}_{1-2\alpha}(x_{new})] \geq 1 - 2\alpha.$$

The following diagram visualizes the difference between Jackknife and Jackknife+.

Let us try the Jackknife+ method on the `KidIQ` dataset.

```
kidiq <- read.csv("data/kidiq.csv")

n <- nrow(kidiq)
new_data <- data.frame(kid_score = NA,
                       mom_hs = 0,
```

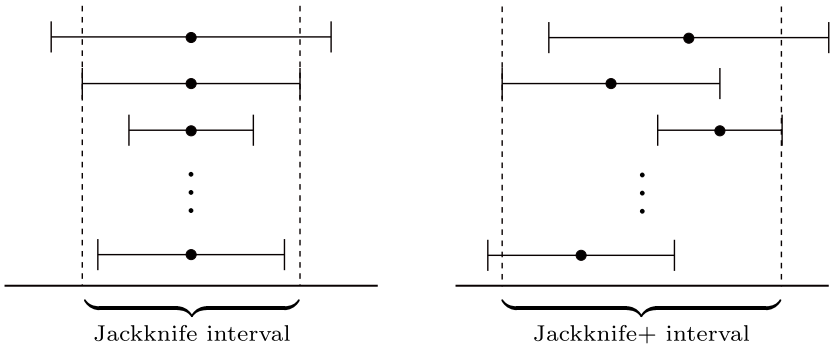


Figure 25.1: Comparison between Jackknife and Jackknife+ intervals.

```
                    mom_iq = 90,
                    mom_work = 1,
                    mom_age = 20)

head(kidiq)
```

```
  kid_score mom_hs    mom_iq mom_work mom_age
1        65      1 121.11753        4      27
2        98      1  89.36188        4      25
3        85      1 115.44316        4      27
4        83      1  99.44964        3      25
5       115      1  92.74571        4      27
6        98      0 107.90184        1      18
```

Suppose that we want to find a 90% prediction interval of the new data point; then we have to set $\alpha = 0.05$.

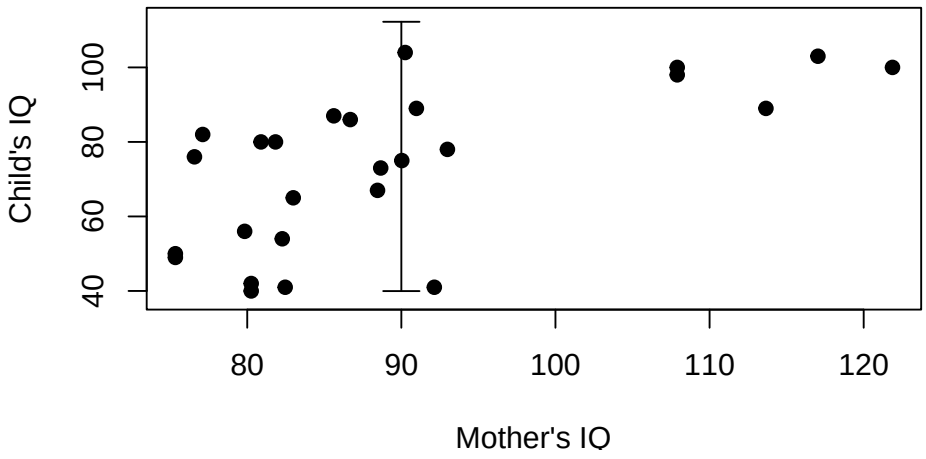
```
alpha <- 0.05

lowers <- rep(NA, n)
uppers <- rep(NA, n)

for (i in 1:n) {
    model <- lm(kid_score ~ ., data = kidiq[-i, ])
    ynew_hat <- predict(model, new_data)
    yi_hat <- predict(model, kidiq[i, ])
    yi <- kidiq[i, "kid_score"]
    Ri <- abs(yi - yi_hat)

    lowers[i] <- ynew_hat - Ri
```

```
    uppers[i] <- ynew_hat + Ri
}

lower <- quantile(lowers, alpha)
upper <- quantile(uppers, 1 - alpha)
cat("Prediction interval: [", lower, ",", upper, "]")
```

```
Prediction interval: [ 39.96481 , 112.2441 ]
```

The prediction interval is similar to those obtained in the previous chapters, so it should have sufficient coverage over the data points. To see this, we plot the interval on the data that have similar features as those of the new data point.

```
kidiq_subset <- kidiq[kidiq$mom_hs == 0 &
                      kidiq$mom_work == 1 &
                      abs(kidiq$mom_age - 20) < 5,]

plot(kid_score ~ mom_iq, data = kidiq_subset,
     ylab = "Child's IQ", xlab = "Mother's IQ",
     ylim = c(38, 113), pch = 19)
arrows(x0 = new_data$mom_iq, y0 = lower,
       x1 = new_data$mom_iq, y1 = upper,
       code = 3, angle = 90, length = 0.1)
```



25.2 CV+

CV+ is a generalization of Jackknife+ in which we split the data into several folds instead of leaving one point out for calibration. Here are the steps to perform CV+ in full details:

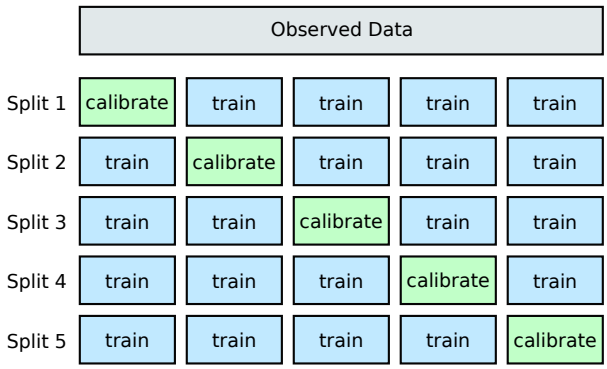


Figure 25.2: A 5-fold split of the observed data.

Let $\mathcal{D}_{1:n} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be the observed data, $(x_{\text{new}}, ?)$ be a new data point, and K be a pre-specified number of folds.

1. Split $\{1, \ldots, n\}$ into K folds $I_1, \ldots, I_K$. Define a function $f : \{1, \ldots, n\} \to \{1, \ldots, K\}$ such that $f(i) = k$ if $i \in I_k$.

2. For each $k \in \{1, \ldots, K\}$,

   2.1. Fit the model on $\{(x_i, y_i) : i \in \{1, \ldots, n\} - I_k\}$. Let $\hat{\mu}_k$ be the fitted model.

   2.2. For each $i \in I_k$,

   - Compute the non-conformity score $R_i = |y_i - \hat{\mu}_k(x_i)|$.
   - Compute a lower and upper bound of $\hat{\mu}_k$'s prediction interval for the new data point:

$$L_i = \hat{\mu}_k(x_{\text{new}}) - R_i, \quad U_i = \hat{\mu}_k(x_{\text{new}}) + R_i.$$

The CV+ prediction interval is

$$C^{\text{CV+}}_{1-2\alpha}(x_{\text{new}}) = [\alpha\text{-quantile of } \{L_1, \ldots, L_n\}, (1-\alpha)\text{-quantile of } \{U_1, \ldots, U_n\}] .$$

The following theorem from Barber et al. (2021) shows that this prediction interval has $1 - 2\alpha$ probability coverage. So, for example, to obtain a 90% prediction interval we would set $\alpha = 0.05$.

Theorem 25.2. ***Theorem 2.*** *If $(x_1, y_1), \ldots, (x_n, y_n), (x_{new}, y_{new})$ are exchangeable, then*

$$\Pr[y_{new} \in C^{CV+}_{1-2\alpha}(x_{new})] \geq 1 - 2\alpha.$$

Let us use CV+ to contruct a 90% prediction interval on `KidIQ`'s new data point. Here, we use `createFolds` from `caret` package to split the data into 10 folds.

```
library(caret)
```

```
alpha <- 0.05

lowers <- rep(NA, n)
uppers <- rep(NA, n)

folds <- createFolds(kidiq$kid_score, k = 10)
for (fold in folds) {
    model <- lm(kid_score ~ ., data = kidiq[-fold, ])
    ynew_hat <- predict(model, new_data)
    yi_hat <- predict(model, kidiq[fold, ])
    yi <- kidiq[fold, "kid_score"]
    Ri <- abs(yi - yi_hat)

    lowers[fold] <- ynew_hat - Ri
    uppers[fold] <- ynew_hat + Ri
}

lower <- quantile(lowers, alpha)
upper <- quantile(uppers, 1 - alpha)
cat("Prediction interval: [", lower, ",", upper, "]")
```

```
Prediction interval: [ 39.80893 , 111.6746 ]
```

25.3 Quantile regression

In the split conformal, the prediction interval is $[\widehat{y}_{\mathsf{new}} - R_{(k)}, \widehat{y}_{\mathsf{new}} + R_{(k)}]$. Notice that the interval has constant width independent of the new data point. So the split conformal might not be appropriate for heteroskedastic data.

Alternatively, we could instead estimate a lower and upper quantiles of the prediction interval. To estimate a specific quantile of the outcome given a set of predictors, we can use the *quantile regression*. Below is an example of using the quantile regression to estimate the 0.1-quantile and 0.9-quantiles on simulated data in R.

```
library(quantreg)
```

```r
n_points <- 500
slope <- 2

x <- runif(n_points, 0, 2)
u <- x * rnorm(n_points) # heteroskedastic errors
y <- x * slope + u

simdata <- data.frame(x = x, y = y)

plot(x, y, pch = 20, ylim = c(-0.5, 7))
abline(rq(y ~ x, tau = 0.9), col = "blue", lwd = 2)
abline(rq(y ~ x, tau = 0.1), col = "red", lwd = 2)
legend("topleft", legend = c("0.9-quantile", "0.1-quantile"),
       col = c("blue", "red"), lty = 1, lwd = 2)
```

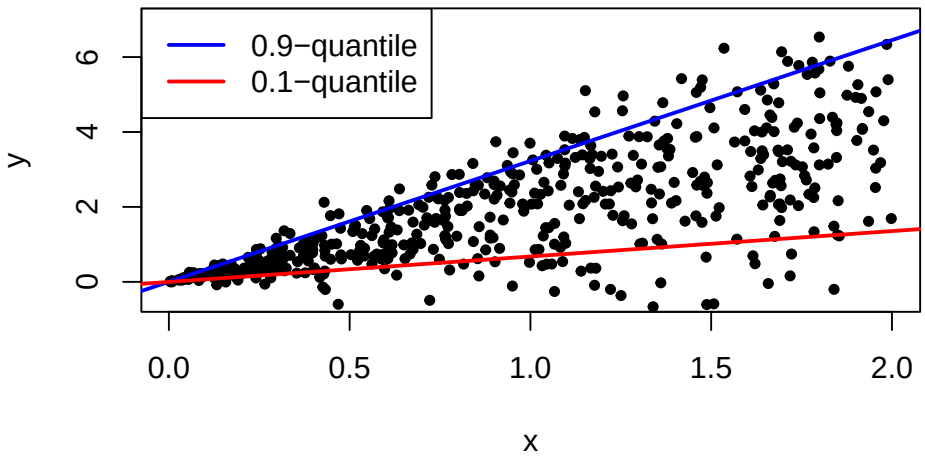


Figure 25.3: 0.1-quantile regression and 0.9-quantile regression.

This plot suggests that the quantile estimates can be used to construct a prediction interval.

Motivated by this observation, Romano, Patterson, and Candes (2019) introduced *conformal quantile regression* (CQR), which uses the quantile estimates to construct a prediction interval. The steps to perform CQR are as follows:

Let $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ be the observed data, $(x_{\mathsf{new}}, ?)$ be a new data point and $\gamma \in (0, 0.5)$ is a pre-specified quantile of the prediction.

1. Split $\{1, \ldots, n\}$ into a training set Tr and a calibration set Cal.
2. Fit γ-quantile and $(1-\gamma)$-quantile regressions on $\{(x_i, y_i) : i \in \mathsf{Tr}\}$. Denote the fitted models by $\widehat{\mu}^-$ and $\widehat{\mu}^+$, respectively.

3. For each $j \in \mathsf{Cal}$, compute the non-conformity score:

$$R_j = \max\left\{\widehat{q}^{-}(x_j) - y_j, y_j - \widehat{q}^{+}(x_j)\right\}.$$

In other words, R_j is a signed distance form y_i to one of the regression lines, where R_j is negative if $\widehat{q}^{-}(x_i) < y_i < \widehat{q}^{+}(x_i)$ and positive otherwise.

4. Sort $R_1, \ldots, R_n$ in increasing order: $R_{(1)}, \ldots, R_{(n)}$.
5. Let $k = \lceil \beta(n+1) \rceil$. The prediction interval is

$$C_\beta^{\mathsf{CQR}}(x_{\mathsf{new}}) = [\widehat{q}^{-}(x_{\mathsf{new}}) - R_{(k)}, \widehat{q}^{+}(x_{\mathsf{new}}) + R_{(k)}].$$

The following theorem from Romano, Patterson, and Candes (2019) shows that this prediction interval has β probability coverage.

If $(x_1, y_1), \ldots, (x_n, y_n), (x_{\mathsf{new}}, y_{\mathsf{new}})$ are exchangeable, then

$$\Pr[y_{\mathsf{new}} \in C_\beta^{\mathsf{CQR}}(x_{\mathsf{new}})] \geq \beta.$$

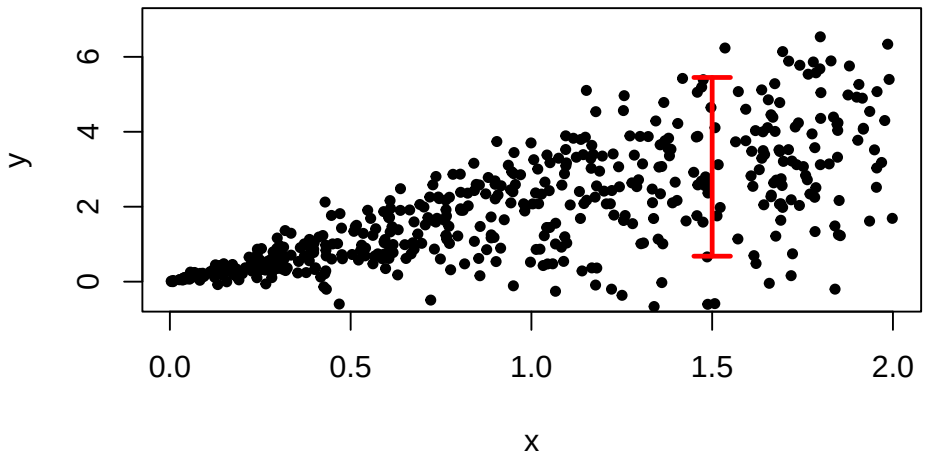
```r
beta <- 0.95
q <- 0.2
m <- floor(n_points/2)

new_xy <- data.frame(x = 1.5, y = NA)

calib_id <- sample(seq_len(n_points), size = m)
simdata_train <- simdata[-calib_id, ]
simdata_calib <- simdata[calib_id, ]

# Training
model_lo <- rq(y ~ x, tau = q, data = simdata_train)
model_hi <- rq(y ~ x, tau = 1 - q, data = simdata_train)

# Calibration
yhat_lo <- predict(model_lo, newdata = simdata_calib)
yhat_hi <- predict(model_hi, newdata = simdata_calib)
y <- simdata_calib$y
R <- pmax(yhat_lo - y, y - yhat_hi)

Rk <- quantile(R, beta)

# Estimate the quantiles of the new data point
ynew_hat_lo <- predict(model_lo, newdata = new_xy)
ynew_hat_hi <- predict(model_hi, newdata = new_xy)
```

```r
# Construct the prediction interval
lower <- ynew_hat_lo - Rk
upper <- ynew_hat_hi + Rk
cat("Prediction interval: [", lower, ",", upper, "]")
```

```
Prediction interval: [ 0.6791725 , 5.449896 ]
```

Let us plot the prediction interval to see how it fares on the simulated data,

```r
plot(y ~ x, data = simdata, pch = 20, ylim = c(-0.5, 7))
arrows(x0 = new_xy$x, y0 = lower,
       x1 = new_xy$x, y1 = upper,
       code = 3, angle = 90, length = 0.1,
       col = "red", lwd = 2.5)
```



which shows that the interval has sufficient coverage at $x = 1.5$.

Chapter 26

Conformal prediction for classification

Consider the following classification task: let $\mathcal{D}_{1:n} = \{(x_1, y_1), \dots, (x_n, y_n)\}$ be the observed data with $y_i \in \{1, \dots, K\}$, and $(x_{\text{new}}, ?)$ be a new data point. The goal is to find a class $y_{\text{new}} \in \{1, \dots, K\}$ that is best associated with x_{new}.

As in regression, conformal prediction allows us to obtain a *prediction set* which contains multiple classes. Let $\hat{p}(y|x)$ be a model that estimates the probability that an example with predictor x is in class y; for example, $\hat{p}$ could be a logistic regression model. We compute the model's likelihood on (x_i, y_i):

$$P_i = \hat{p}_{-i}(y_i|x_i).$$

One candidate for the non-conformity score is the negative likelihood:

$$R_i = -P_i.$$

We then sort $R_1, \dots, R_n$ in increasing order: $R_{(1)}, \dots, R_{(n)}$. Let $\alpha \in (0.5, 1)$ and $\beta = 1 - \alpha$. The quantile lemma in Section 24.2 tells us that the prediction set:

$$\{y \in \{1, \dots, K\} : -\hat{p}(y|x_{\text{new}}) \leq \lceil \beta(n+1) \rceil\text{-quantile of } \{R_1, \dots, R_n\}\},$$

has $1 - \alpha$ coverage probability. We can also write this set in terms of likelihood:

$$\{y \in \{1, \dots, K\} : \hat{p}(y|x_{\text{new}}) \geq \lfloor \alpha(n+1) \rfloor\text{-quantile of } \{P_1, \dots, P_n\}\},$$

which is easier to interpret. From this observation, we present here two approaches to obtain a prediction set.

26.1 Full conformal approach

Let $\mathcal{D}_{1:n} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be the observed data with $y_i \in \{1, \ldots, K\}$, and $(x_{\mathsf{new}}, ?)$ be a new data point.

For each $y_{\mathsf{new}} \in \{1, \ldots, K\}$,

1. Add the new point $\{(x_{\mathsf{new}}, y_{\mathsf{new}})\}$ to the dataset.
2. Fit the model on $\mathcal{D}_{1:n} \cup \{(x_{\mathsf{new}}, y_{\mathsf{new}})\}$. Let $\widehat{p}$ be the fitted model.
3. For each $i \in \{1, \ldots, n, \mathsf{new}\}$, compute the probability estimate $P_i = \widehat{p}(y_i|x_i)$.
4. Sort the probabilities $P_1, \ldots, P_n$ in increasing order: $P_{(1)}, \ldots, P_{(n)}$.
5. Let $k = \lfloor \alpha(n+1) \rfloor$. Include y_{new} in the prediction set if $\widehat{p}(y_{\mathsf{new}}|x_{\mathsf{new}}) \geq P_{(k)}$, otherwise we discard it.

In summary, the prediction set is

$$\mathcal{S}_{1-\alpha}(x_{\mathsf{new}}) = \{y_{\mathsf{new}} \in \{1, \ldots, K\} : \widehat{p}(y_{\mathsf{new}}|x_{\mathsf{new}}) \geq P_{(k)}\}.$$

It follows from Lemma 24.1 that this prediction set has $1-\alpha$ probability coverage: let (x_{new}, y) be a new data point, then

$$\Pr[y \in \mathcal{S}_{1-\alpha}(x_{\mathsf{new}})] \geq 1 - \alpha.$$

26.2 Jackknife+ approach

The problem with the full conformal approach is the need to refit the model on every new data point. To avoid this problem, we can take the Jackknife+ approach and fit the model on the training set and make a prediction on the new data point. Here are the steps to obtain a prediction set using the Jackknife+ approach in full details:

Let $\mathcal{D}_{1:n} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be the observed data with $y_i \in \{1, \ldots, K\}$, and $(x_{\mathsf{new}}, ?)$ be a new data point.

1. For each $i \in \{1, \ldots, n, \mathsf{new}\}$,

   1.1. Fit the model on $\mathcal{D}_{1:n}$. Let $\widehat{p}_{-i}$ be the fitted model.

   1.2. Compute the probability estimate $P_i = \widehat{p}_{-i}(y_i|x_i)$.

2. Sort the probabilities $P_1, \ldots, P_n$ in increasing order: $P_{(1)}, \ldots, P_{(n)}$.
3. For each $y_{\mathsf{new}} \in \{1, \ldots, K\}$,

   Include y_{new} in the prediction set if $\widehat{p}_{-i}(y_{\mathsf{new}}|x_{\mathsf{new}}) \geq P_{(k)}$ where $k = \lfloor \alpha(n+1) \rfloor$.

In summary, the prediction set is

$$\mathcal{S}^{\mathsf{JK+}}_{1-2\alpha}(x_{\mathsf{new}}) = \{y_{\mathsf{new}} \in \{1, \ldots, K\} : \widehat{p}_{-i}(y_{\mathsf{new}}|x_{\mathsf{new}}) \geq P_{(k)}\}.$$

This set has $1-2\alpha$ probability coverage (Romano, Patterson, and Candes 2019). More precisely, let (x_{new}, y) be a new data point, then

$$\Pr[y \in \mathcal{S}^{\mathsf{JK+}}_{1-\alpha}(x_{\mathsf{new}})] \geq 1 - 2\alpha.$$

References

Abad, Javier, Umang Bhatt, Adrian Weller, and Giovanni Cherubin. 2022. "Approximating Full Conformal Prediction at Scale via Influence Functions." In.

Abadie, Alberto, Alexis Diamond, and Jens Hainmueller. 2010. "Synthetic Control Methods for Comparative Case Studies: Estimating the Effect of California's Tobacco Control Program." *Journal of the American Statistical Association* 105 (490): 493–505. https://doi.org/10.1198/jasa.2009.ap08746.

Alexander, Monica. 2019. "Analyzing Name Changes After Marriage Using a Non-Representative Survey." *Personal Blog*. https://www.monicaalexander.com/posts/2019-08-07-mrp/.

Barber, Rina Foygel, Emmanuel J. Candès, Aaditya Ramdas, and Ryan J. Tibshirani. 2021. "Predictive Inference with the Jackknife+." *The Annals of Statistics* 49 (1): 486–507. https://doi.org/10.1214/20-AOS1965.

Brodersen, Kay H., Fabian Gallusser, Jim Koehler, Nicolas Remy, and Steven L. Scott. 2015. "INFERRING CAUSAL IMPACT USING BAYESIAN STRUCTURAL TIME-SERIES MODELS." *The Annals of Applied Statistics* 9 (1): 247–74. http://www.jstor.org/stable/24522418.

Candès, Emmanuel. 2022. "Lecture Notes of Stats 300C: Theory of Statistics." 2022. https://candes.su.domains/teaching/stats300c/lectures.html.

Card, David, and Alan B. Krueger. 1993. "Minimum Wages and Employment: A Case Study of the Fast Food Industry in New Jersey and Pennsylvania." Working {Paper}. Working Paper Series. National Bureau of Economic Research. https://doi.org/10.3386/w4509.

Chen, M. Keith. 2016. "Dynamic Pricing in a Labor Market." In *Proceedings of the 2016 ACM Conference on Economics and Computation*. ACM. https://doi.org/10.1145/2940716.2940798.

Cunningham, Scott. 2021. *Causal Inference: The Mixtape*. Yale university press.

Datta, Hannes, George Knox, and Bart J. Bronnenberg. 2018. "Changing Their Tune: How Consumers' Adoption of Online Streaming Affects Music Consumption and Discovery." *Marketing Science* 37 (1): 5–21. https://doi.org/10.1287/mksc.2017.1051.

Facure, Matheus. 2020. "Python Causality Handbook." 2020. https://matheusfacure.github.io/python-causality-handbook/landing-page.html.

Farre-Mensa, Joan, Deepak Hegde, and Alexander Ljungqvist. 2017. "What Is a Patent Worth? Evidence from the U.S. Patent "Lottery"." Working {Paper}. Working Paper Series. National Bureau of Economic Research. https://doi.org/10.3386/w23268.

Forter, Carson. 2017. "Two-Stage Least Squares For A/B Tests." *Twitch Blog*. https://blog.twitch.tv/en/2017/06/30/two-stage-least-squares-for-a-b-tests-669d07f904f7/.

Funk, Michele Jonsson, Daniel Westreich, Chris Wiesen, Til Stürmer, M. Alan Brookhart, and Marie Davidian. 2011. "Doubly Robust Estimation of Causal Effects." *American Journal of Epidemiology* 173 (7): 761–67. https://doi.org/10.1093/aje/kwq439.

Gelman, Andrew, Jennifer Hill, and Aki Vehtari. 2020. *Regression and Other Stories*. Analytical Methods for Social Research. Cambridge University Press. https://doi.org/10.1017/9781139161879.

Hanck, Christoph, Martin Arnold, Alexander Gerber, and Martin Schmelzer. 2019. *Introduction to Econometrics with r. University of Duisburg-Essen*.

Huntington-Klein, Nick. 2021. *The Effect: An Introduction to Research Design and Causality*. CRC Press.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2021. *An Introduction to Statistical Learning*. Springer US. https://doi.org/10.1007/978-1-0716-1418-1.

Kharel, Ujwal. 2021. "Causal Inference Using Instrumental Variables." *Roblox Blog*. https://blog.roblox.com/2021/09/causal-inference-using-instrumental-variables.

Künzel, Sören R., Jasjeet S. Sekhon, Peter J. Bickel, and Bin Yu. 2019. "Metalearners for Estimating Heterogeneous Treatment Effects Using Machine Learning." *Proceedings of the National Academy of Sciences* 116 (10): 4156–65. https://doi.org/10.1073/pnas.1804597116.

Lei, Jing, Max G'Sell, Alessandro Rinaldo, Ryan J. Tibshirani, and Larry Wasserman. 2018. "Distribution-Free Predictive Inference for Regression." *Journal of the American Statistical Association* 113 (523): 1094–1111. https://doi.org/10.1080/01621459.2017.1307116.

Mehrotra, Rishabh, Prasanta Bhattacharya, and Mounia Lalmas. 2020. "Inferring the Causal Impact of New Track Releases on Music Recommendation Platforms Through Counterfactual Predictions." In *Proceedings of the 14th ACM Conference on Recommender Systems*, 687–91. RecSys '20. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/3383313.3418491.

Romano, Yaniv, Evan Patterson, and Emmanuel Candes. 2019. "Conformalized Quantile Regression." In *Advances in Neural Information Processing Systems*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/5103c3584b063c431bd1268e9b5e76fb-Paper.pdf.

Stock, James, and Motohiro Yogo. 2002. "Testing for Weak Instruments in Linear IV Regression." National Bureau of Economic Research. https://doi.org/10.3386/t0284.

Vella, Francis, and Marno Verbeek. 1998. "Whose Wages Do Unions Raise? A Dynamic Model of Unionism and Wage Rate Determination for Young Men." *Journal of Applied Econometrics* 13 (2): 163–83. http://www.jstor.org/stable/223257.

Vovk, Vladimir, Alexander Gammerman, and Glenn Shafer. 2005. *Algorithmic Learning in a Random World*. Springer-Verlag. https://doi.org/10.1007/b106715.

Vovk, Vladimir, Jieli Shen, Valery Manokhin, and Min-ge Xie. 2019. "Nonparametric Predictive Distributions Based on Conformal Prediction." *Machine Learning* 108 (3): 445–74. https://doi.org/10.1007/s10994-018-5755-8.

Webel, Karsten. 2011. *JH Stock, MW Watson: Introduction to Econometrics*. Springer Nature BV.