Open Intro Fall 2020

Aleksandr Fisher

Getting the Basics

- Don't forget to check and set your working directory

```
source("http://www.openintro.org/stat/data/cdc.R")
```

Getting the Basics

- The Behavioral Risk Factor Surveillance System (BRFSS) is an annual telephone survey of 350,000 people in the United States. The BRFSS Web site contains a complete description of the survey, including the research questions that motivate the study and many interesting results derived from the data.

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------
```

```
## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  3.0.0     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------
## x dplyr::filter() masks stats::filter()
```

Explore the data

- You can see and inspect the data set comfortably in RStudio with the View() command, which invokes a spreadsheet-style data viewer on a matrix-like R object.

```
CDC
```

```
## # A tibble: 20,000 x 9
##    genhlth   exerany hlthplan smoke100 height weight wtd
##    <fct>       <dbl>    <dbl>    <dbl>  <dbl>  <int>
##  1 good            0        1        0     70    175
##  2 good            0        1        1     64    125
##  3 good            1        1        1     60    105
##  4 good            1        1        0     66    132
##  5 very good       0        1        0     61    150
##  6 very good       1        1        0     64    114
##  7 very good       1        1        0     71    194 4
```

Explore the data

- After loading the data and converting it into a tibble, one should inspect the data to get some understanding about the structure and content. Common funtions for these tasks are:
- `<name-of-data-tibble>`: Display the first 10 rows and all columns that fit on one screen. It also prints an abbreviated description of the column type.
- `head(<name-of-df>)`, `tail(<name-of-df>)`: Return the first or last part. Use these commands if it is not a tibble but a data frame
- `dim()`: Retrieve the dimension
- `names()`: Get the names
- `str()`: Display compactly the internal structure
- `glimpse()`: is the dplyr version of str() showing values of each

Install Packages

```
library(dplyr)
library(ggplot2)

source("http://www.openintro.org/stat/data/arbuthnot.R")
```

##Look as Data

The Arbuthnot data set refers to Dr. John Arbuthnot, an 18th century physician, writer, and mathematician. He was interested in the ratio of newborn boys to newborn girls, so he gathered the baptism records for children born in London for every year from 1629 to 1710. We can view the data by typing its name into the console.

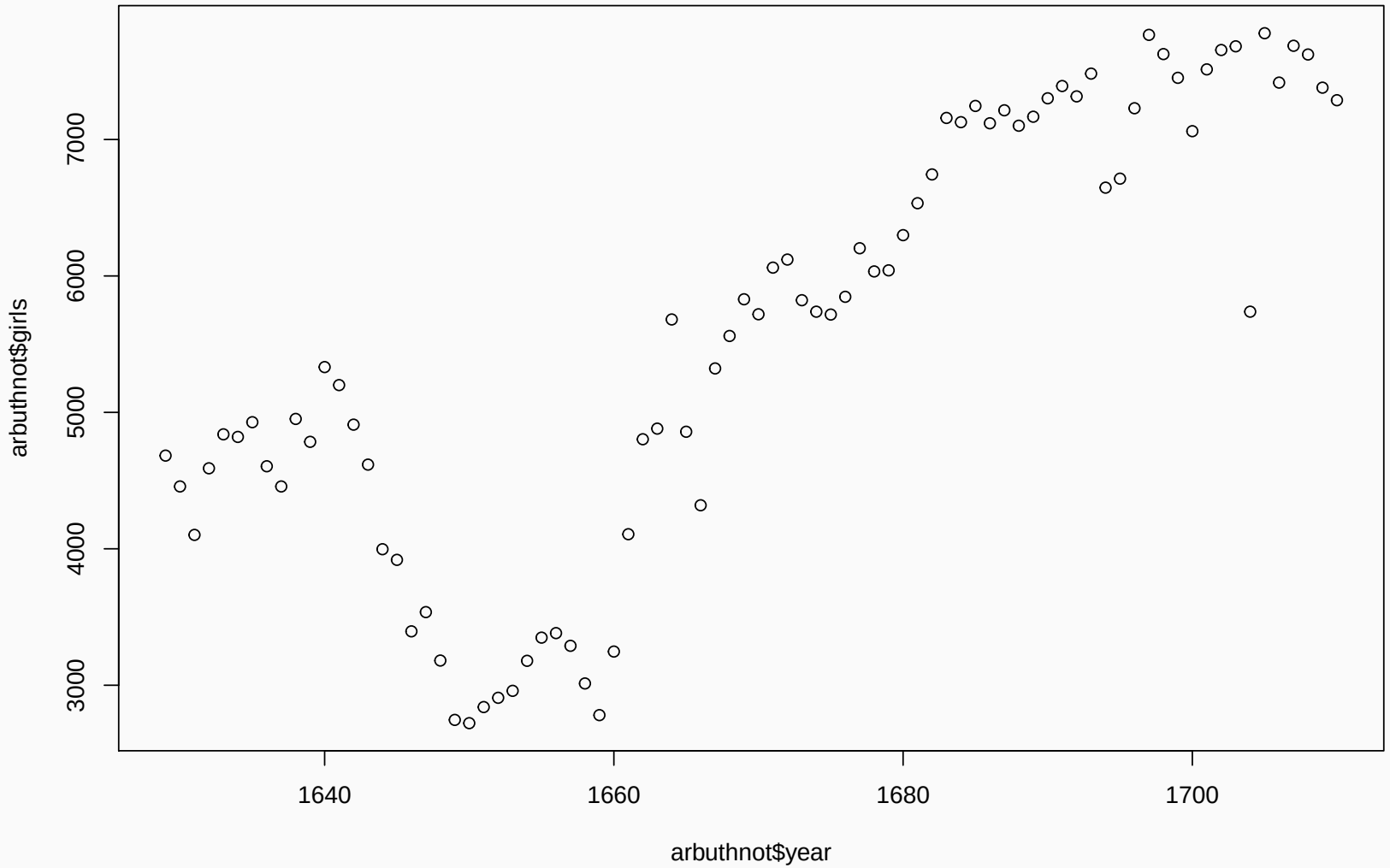
```
dim(arbuthnot)
```

```
## [1] 82  3
```

```
glimpse(arbuthnot)
```

Plot Data - Base R

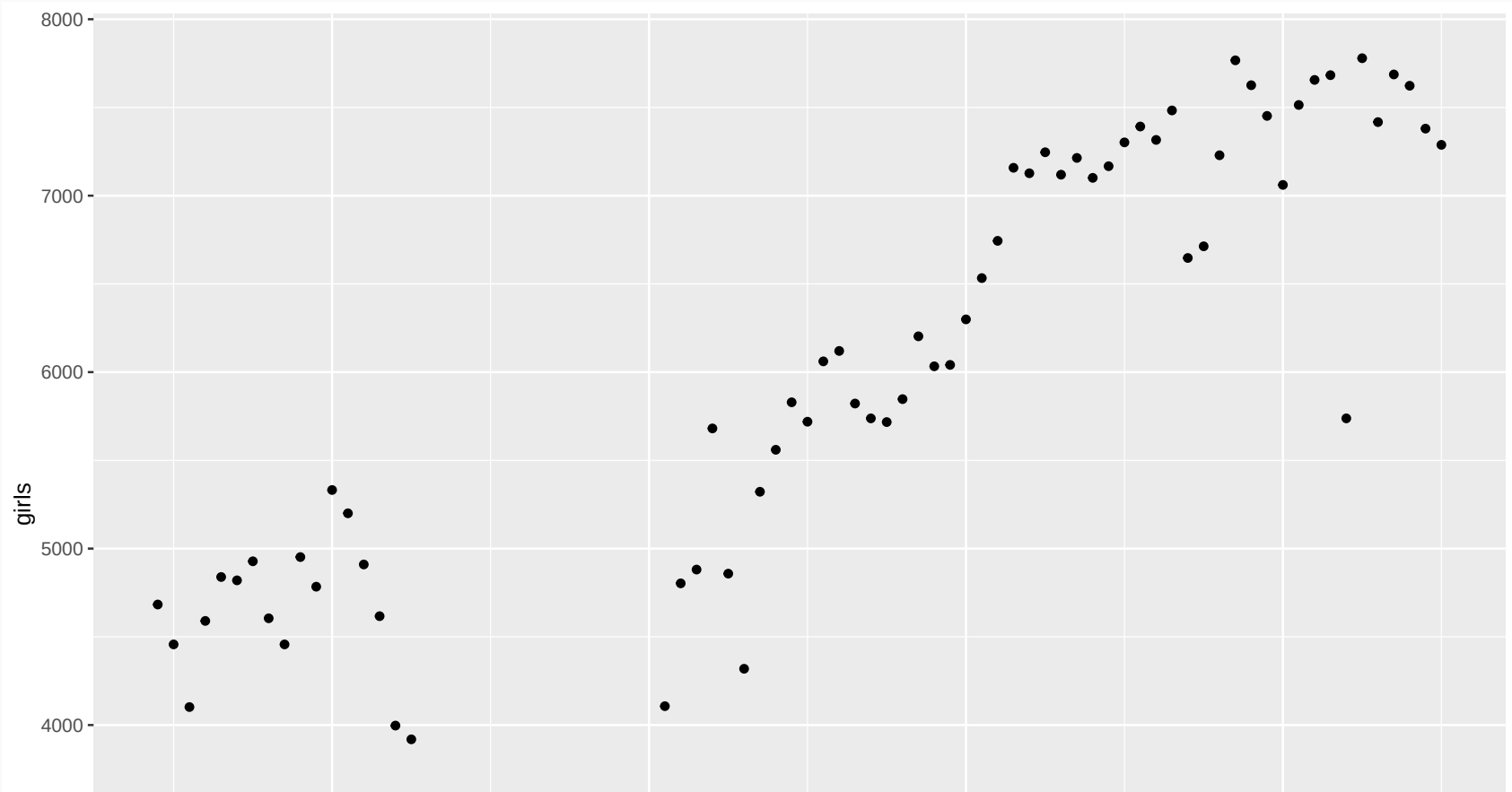
```
plot(x = arbuthnot$year, y = arbuthnot$girls)
```



Plot Data - qplot R

- R has some powerful functions for making graphics. We can create a simple plot of the number of girls baptized per year with qplot.

```
qplot(x = year, y = girls, data = arbuthnot)
```

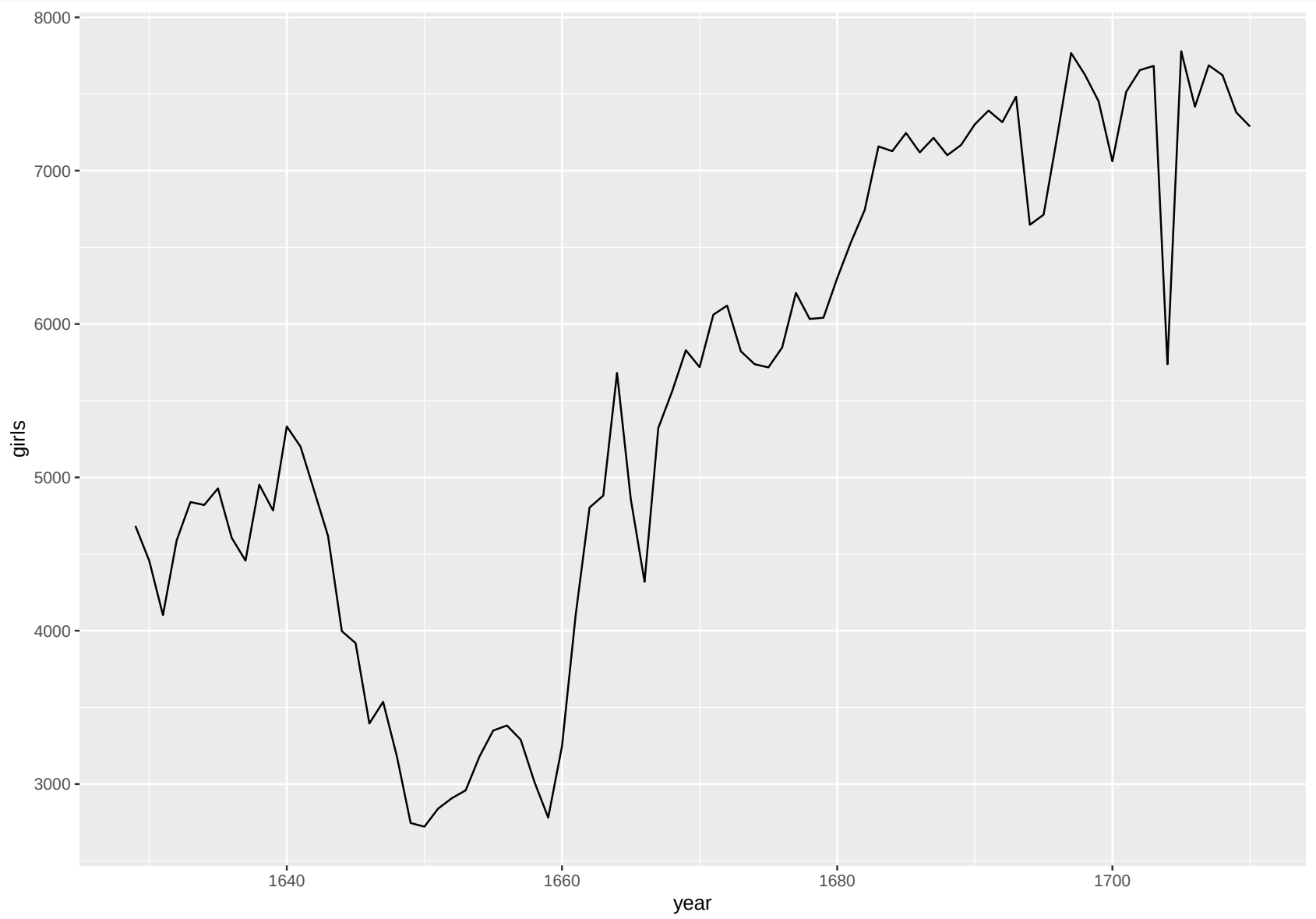


Plot Data - line graph

```
plot(x = arbuthnot$year, y = arbuthnot$girls, type = "l")
```

Plot Data - line graph

```
qplot(x = year, y = girls, data = arbuthnot, geom = "line")
```



Manipulating Data - Creating New Variables

- Now, suppose we want to plot the total number of baptisms. To compute this, we could use the fact that R is really just a big calculator. We can type in mathematical expressions like

```
5218 + 4683
```

```
## [1] 9901
```

```
arbuthnot$boys + arbuthnot$girls
```

```
##  [1]  9901  9315  8524  9584  9997  9855 10034  9522  91
## [13] 10670 10370  9410  8104  7966  7163  7332  6544  58
## [25]  6155  6620  7004  7050  6685  6170  5990  6971  88
## [37]  9972  8997 10938 11633 12335 11997 12510 12563 118
## [49] 12626 12601 12288 12847 13355 13653 14735 14702 147
## [61] 14771 15211 15054 14918 15159 13632 13976 14861 158
## [73] 15616 15687 15448 11851 16145 15369 16066 15862 152
```

Manipulating Data - Creating New Variables (mutate)

- We'll be using this new vector to generate some plots, so we'll want to save it as a permanent column in our data frame.

```
arbuthnot <- arbuthnot %>%
  mutate(total = boys + girls)
```

- The %>% operator is called the piping operator. It takes the output of the previous expression and pipes it into the first argument of the function in the following one. To continue our analogy with mathematical functions, `x %>% f(y)` is equivalent to `f(x, y)`.

Manipulating Data - Creating New Variables

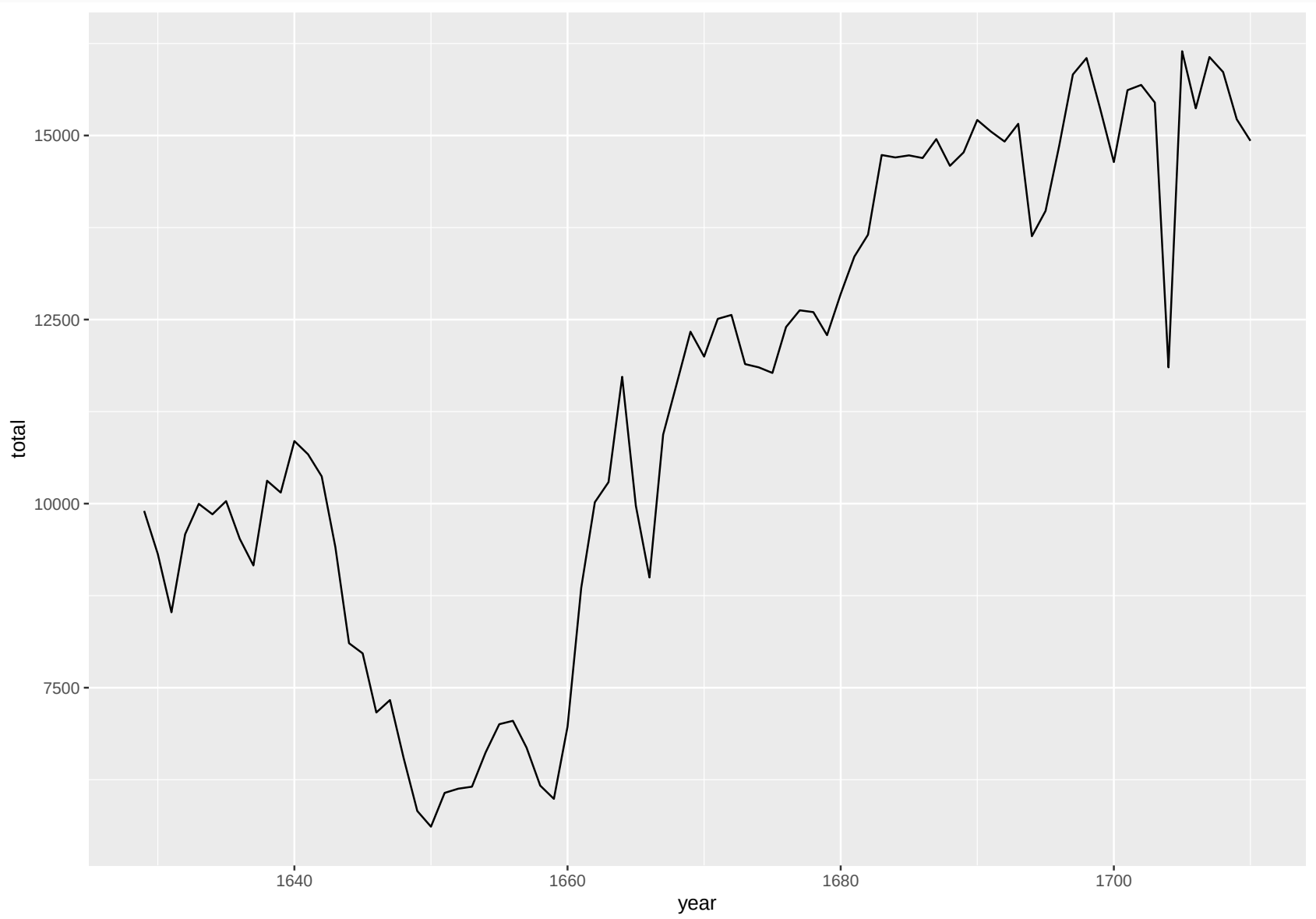
- **A note on piping:** Note that we can read these three lines of code as the following:
- "Take the arbuthnot dataset and pipe it into the mutate function. Mutate the arbuthnot data set by creating a new variable called total that is the sum of the variables called boys and girls.
- Then assign the resulting dataset to the object called arbuthnot, i.e. overwrite the old arbuthnot dataset with the new one containing the new variable."

Manipulating Data - Adding in plot

```
plot(arbuthnot$year, arbuthnot$boys + arbuthnot$girls, type
```

Manipulating Data - Adding in qplot

```
qplot(x = year, y = total, data = arbuthnot, geom = "line")
```



Manipulating Data - creating ratios

```
5218 / 4683
```

```
## [1] 1.114243
```

Manipulating Data - creating ratios

```
arbuthnot$boys / arbuthnot$girls
```

```
##  [1] 1.114243 1.089971 1.078011 1.088017 1.065923 1.0446
##  [9] 1.055194 1.082189 1.121656 1.034884 1.051923 1.1120
## [17] 1.032661 1.109867 1.073529 1.057215 1.121267 1.0617
## [25] 1.080095 1.082416 1.091371 1.084565 1.032533 1.0477
## [33] 1.156075 1.085988 1.108584 1.063369 1.052697 1.0831
## [41] 1.116143 1.097744 1.064016 1.052778 1.043112 1.0653
## [49] 1.035467 1.088679 1.034100 1.039530 1.044237 1.0244
## [57] 1.032846 1.064054 1.072498 1.054359 1.060974 1.0831
## [65] 1.025792 1.050850 1.081931 1.055748 1.037981 1.1049
## [73] 1.078254 1.048981 1.010673 1.065354 1.075460 1.0721
## [81] 1.062331 1.048299
```

```
arbuthnot <- arbuthnot %>%
  mutate(boy_to_girl_ratio = boys / girls)
```

Manipulating Data - ratios using mutate

```
5218 / (5218 + 4683)
```

```
## [1] 0.5270175
```

```
arbuthnot$boys / (arbuthnot$boys + arbuthnot$girls)
```

```
##  [1] 0.5270175 0.5215244 0.5187705 0.5210768 0.5159548 0
##  [8] 0.5163831 0.5134279 0.5197362 0.5286700 0.5085714 0
## [15] 0.5093518 0.5067868 0.5080341 0.5260366 0.5177305 0
## [22] 0.5149679 0.5322023 0.5254569 0.5192526 0.5197885 0
## [29] 0.5080030 0.5116694 0.5357262 0.5342132 0.5361942 0
## [36] 0.5153557 0.5128359 0.5199511 0.5134394 0.5220493 0
## [43] 0.5155076 0.5128552 0.5105507 0.5158214 0.5144798 0
## [50] 0.5212285 0.5083822 0.5096910 0.5108199 0.5060426 0
## [57] 0.5080788 0.5155165 0.5174905 0.5132301 0.5147925 0
## [64] 0.5095857 0.5063659 0.5123973 0.5196766 0.5135590 0
## [71] 0.5149385 0.5176583 0.5188268 0.5119526 0.5026541 0
```

Manipulating Data - True/False

```
arbuthnot$boys > arbuthnot$girls
```

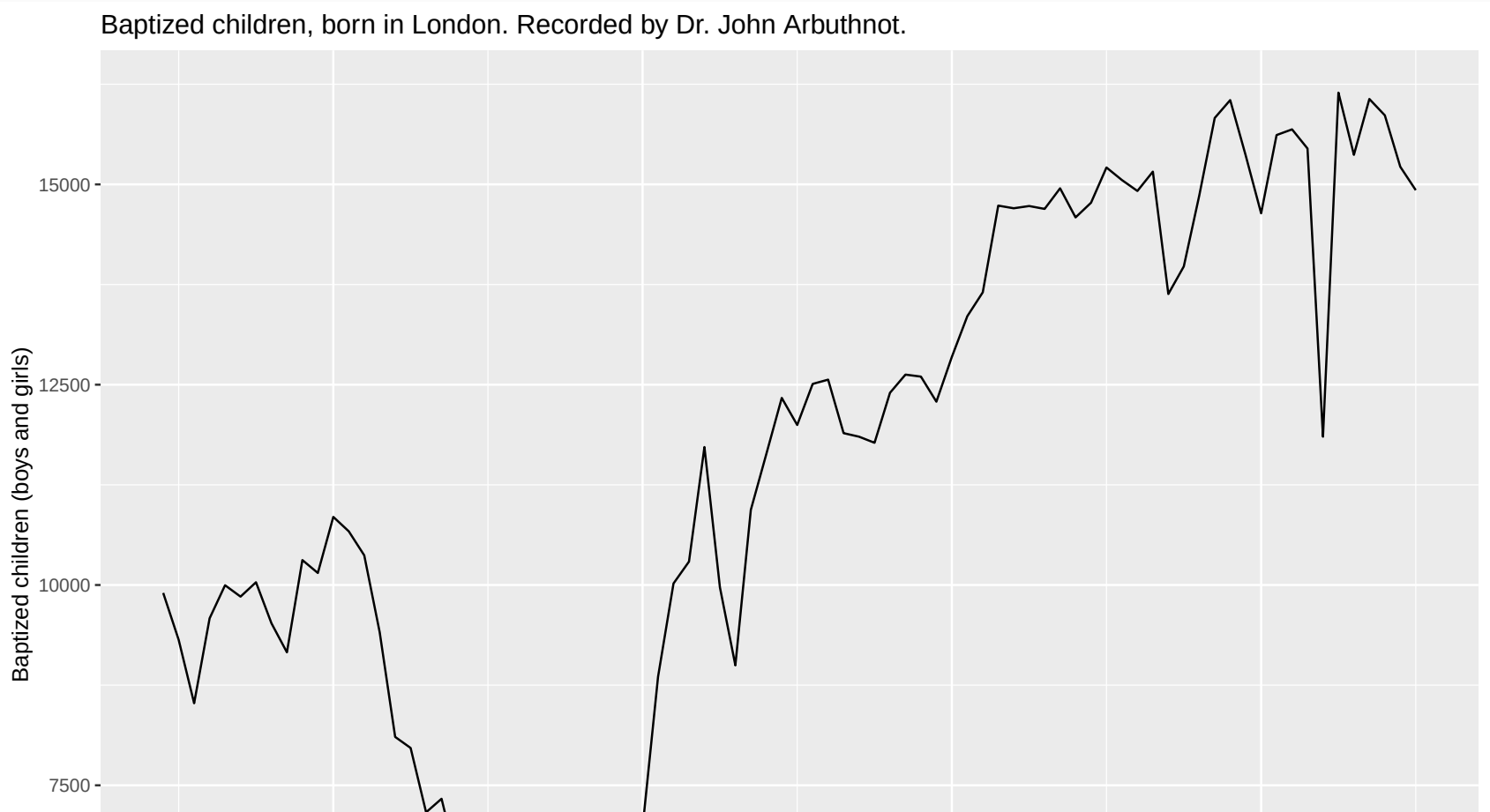
```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE T
## [16] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE T
## [31] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE T
## [46] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE T
## [61] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE T
## [76] TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

Manipulating Data

- Make a plot that displays the boy-to-girl ratio for every year in the data set. What do you see? Does Arbuthnot's observation about boys being born in greater proportion than girls hold up in the U.S.? Include the plot in your response.
- In what year did we see the most total number of births in the U.S.? You can refer to the help files or the R reference card http://cran.r-project.org/doc/contrib/Short-refcard.pdf to find helpful commands.

Some More Plots

```
arbuthnot %>% ggplot(aes(year, total)) +
        geom_line() +
        xlab("Year") + ylab("Baptized children (boys and g
        ggtitle("Baptized children, born in London. Recorde
```



Some More Plots

- Similarly to how we computed the total number of births, we can compute different kinds of ratios (boys to girls, boys to total, girls to total).

```
arbuthnot <- arbuthnot %>%
  mutate(boy_to_girl_ratio = boys / girls)
arbuthnot <- arbuthnot %>%
  mutate(boy_ratio = boys / total)
arbuthnot <- arbuthnot %>%
  mutate(girl_ratio = girls / total)
arbuthnot %>%
        ggplot(aes(year, boy_to_girl_ratio)) +
        geom_line() +
        xlab("Year") + ylab("Baptized ratio (boys to girls)
        ggtitle("Baptized children, born in London. Recorde
```

```
source("http://www.openintro.org/stat/data/cdc.R")

names(cdc)

## [1] "genhlth"  "exerany"  "hlthplan" "smoke100" "height"
## [8] "age"      "gender"
```

- First 6 observations

```
head(cdc)
```

```
##      genhlth exerany hlthplan smoke100 height weight wtde
## 1       good       0        1        0     70    175
## 2       good       0        1        1     64    125
## 3       good       1        1        1     60    105
## 4       good       1        1        0     66    132
## 5 very good       0        1        0     61    150
## 6 very good       1        1        0     64    114
```

- Last 6 observations

```
tail(cdc)
```

```
##           genhlth exerany hlthplan smoke100 height weight
## 19995        good        0        1        1     69    224
## 19996        good        1        1        0     66    215
## 19997   excellent        0        1        0     73    200
## 19998        poor        0        1        0     65    216
## 19999        good        1        1        0     67    165
## 20000        good        1        1        1     69    170
```

Introduction to Data - Summary

```
summary(cdc$weight)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    68.0   140.0   165.0   169.7   190.0   500.0
```

Introduction to Data - Mean, Variance, Median

```
mean(cdc$weight)
```

```
## [1] 169.683
```

```
var(cdc$weight)
```

```
## [1] 1606.484
```

```
median(cdc$weight)
```

```
## [1] 165
```

Introduction to Data -Tables and Plots

```
table(cdc$smoke100)
```

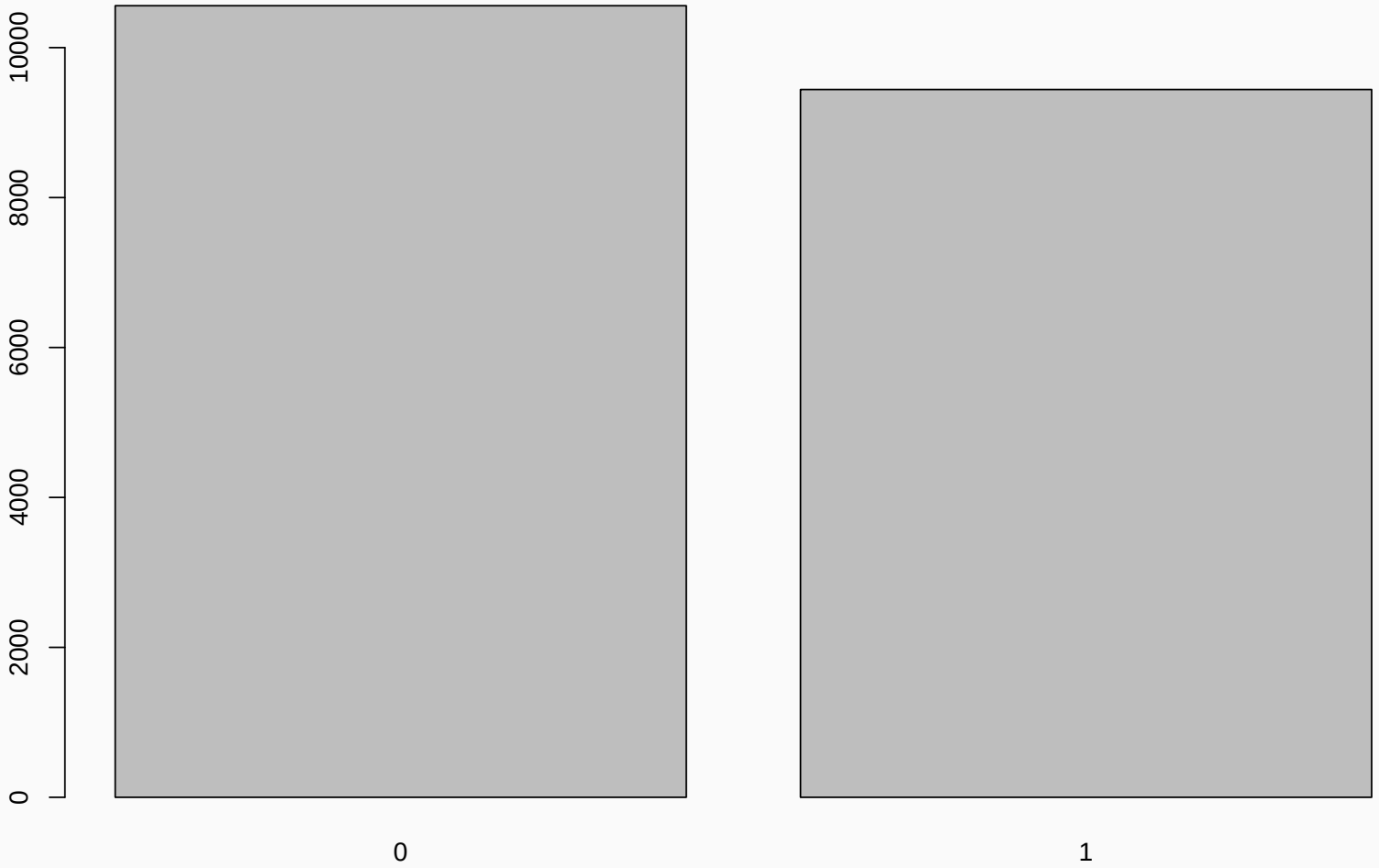
```
##
##     0     1
## 10559  9441
```

```
table(cdc$smoke100)/20000
```

```
##
##       0       1
## 0.52795 0.47205
```

Introduction to Data -Tables and Plots

```
barplot(table(cdc$smoke100))
```



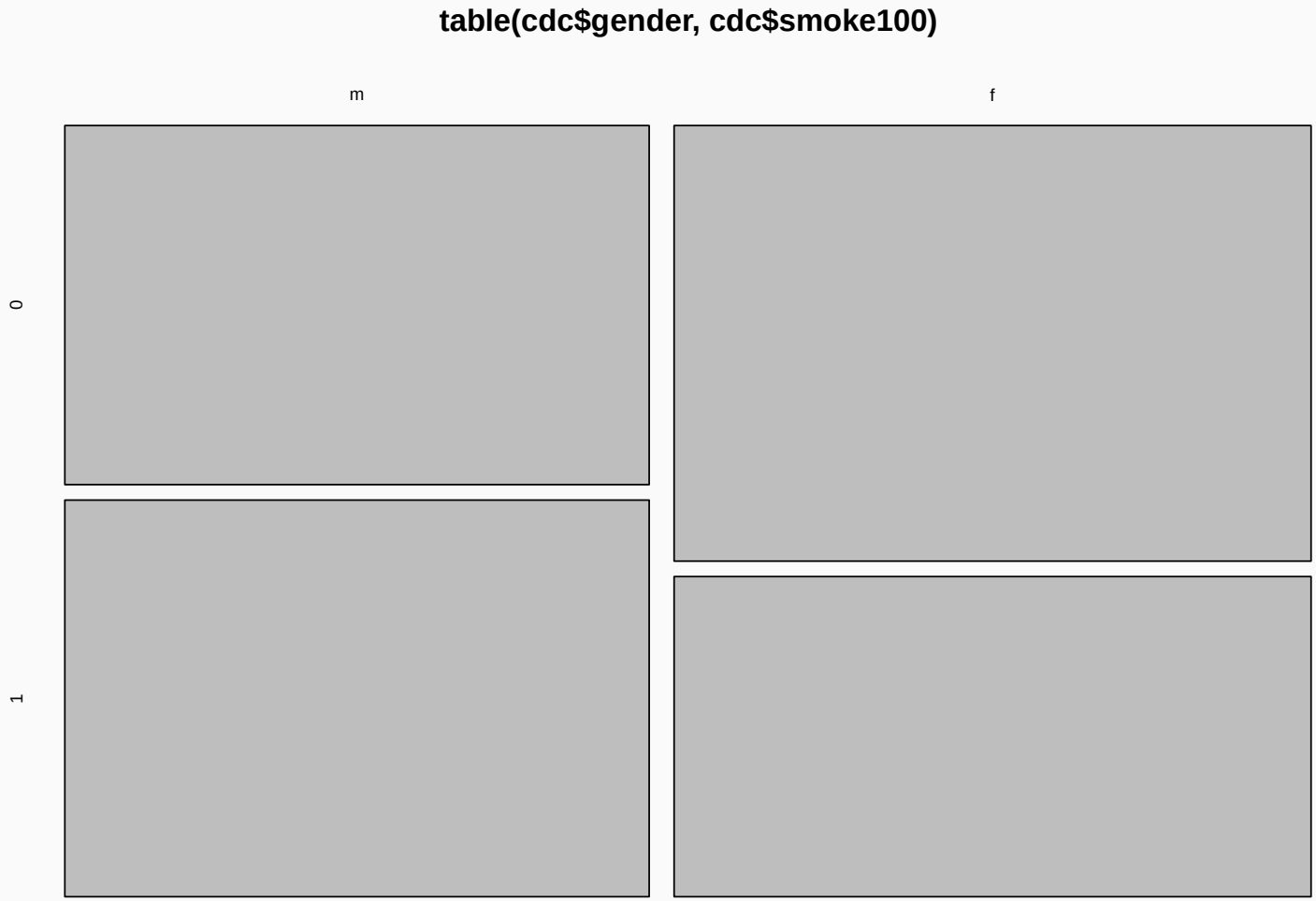
Introduction to Data - 2X2 Table

```
table(cdc$gender,cdc$smoke100)
```

```
##
##        0    1
##   m 4547 5022
##   f 6012 4419
```

Introduction to Data - Mosaic Plot

```
mosaicplot(table(cdc$gender,cdc$smoke100))
```



Introduction to Data - How R Stores Data

```
dim(cdc)
```

```
## [1] 20000     9
```

```
cdc[567,6]
```

```
## [1] 160
```

```
names(cdc)
```

```
## [1] "genhlth"  "exerany"  "hlthplan" "smoke100" "height"
## [8] "age"      "gender"
```

Introduction to Data - How R Stores Data

```
cdc[1:10,6]
```

```
##  [1] 175 125 105 132 150 114 194 170 150 180
```

```
1:10
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
cdc[1:4,]
```

```
##   genhlth exerany hlthplan smoke100 height weight wtdesi
## 1    good        0        1        0     70    175      1
## 2    good        0        1        1     64    125      1
## 3    good        1        1        1     60    105      1
## 4    good        1        1        0     66    132      1
```

```
head(cdc)
```

```
##      genhlth exerany hlthplan smoke100 height weight wtde
## 1       good        0        1        0     70    175
## 2       good        0        1        1     64    125
## 3       good        1        1        1     60    105
## 4       good        1        1        0     66    132
## 5 very good        0        1        0     61    150
## 6 very good        1        1        0     64    114
```

```
cdc[,6]
```

```
##     [1] 175 125 105 132 150 114 194 170 150 180 186 168
##    [19] 200 125 200 160 160 165 105 190 190 160 115 185
##    [37] 160 190 125 160 124 143 118 210 200 145 175 130
##    [55] 150 172 125 168 120 180 186 144 250 160 164 155
##    [73] 217 215 190 215 112 136 142 210 180 120 225 200
##    [91] 265 166 145 138 137 175 230 140 180 100 118 142
##   [109] 211 220 210 160 132 163 132 207 145 190 200 170
##   [127] 160 250 190 205 200 160 144 185 140 200 280 180
##   [145] 136 190 150 130 172 107 187 125 110 205 140 128
##   [163] 195 187 142 225 180 180 180 128 130 215 180 152
##   [181] 220  92 185 150 129 135 140 110 183 212 165 171
##   [199] 150 210 130 200 155 135 200 134 128 115 155 112
##   [217] 149 135 140 165 126 180 107 165 120 210 145 200
##   [235] 207 155 212 122 160 125 170 135 148 160 150 165
```

```
cdc$weight[567]
```

```
## [1] 160
```

```
cdc$weight[1:10]
```

```
##  [1] 175 125 105 132 150 114 194 170 150 180
```

Introduction to Data - Select Variable

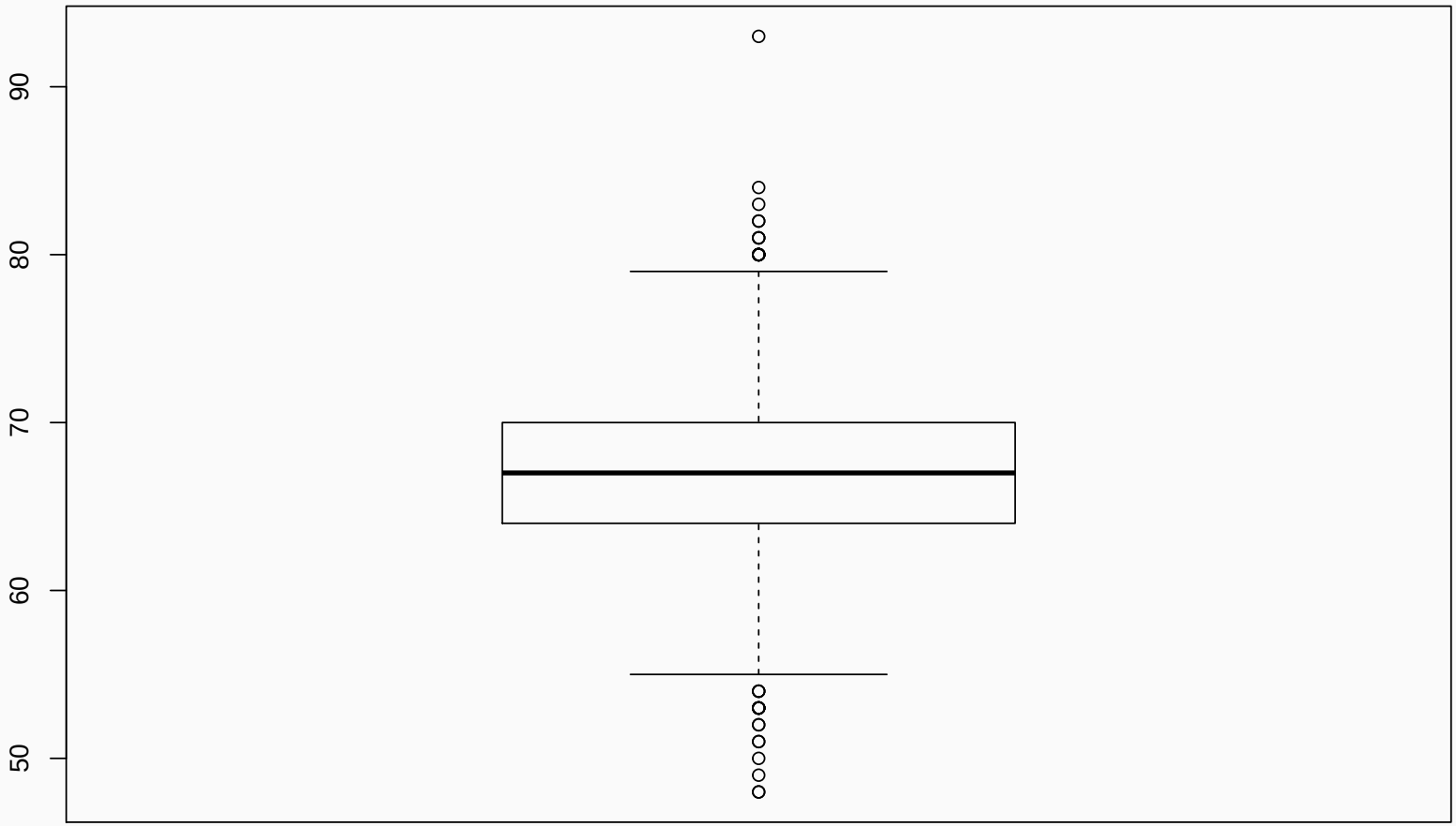
```
cdc$gender == "m"
```

##Introduction to Data - Select Variable

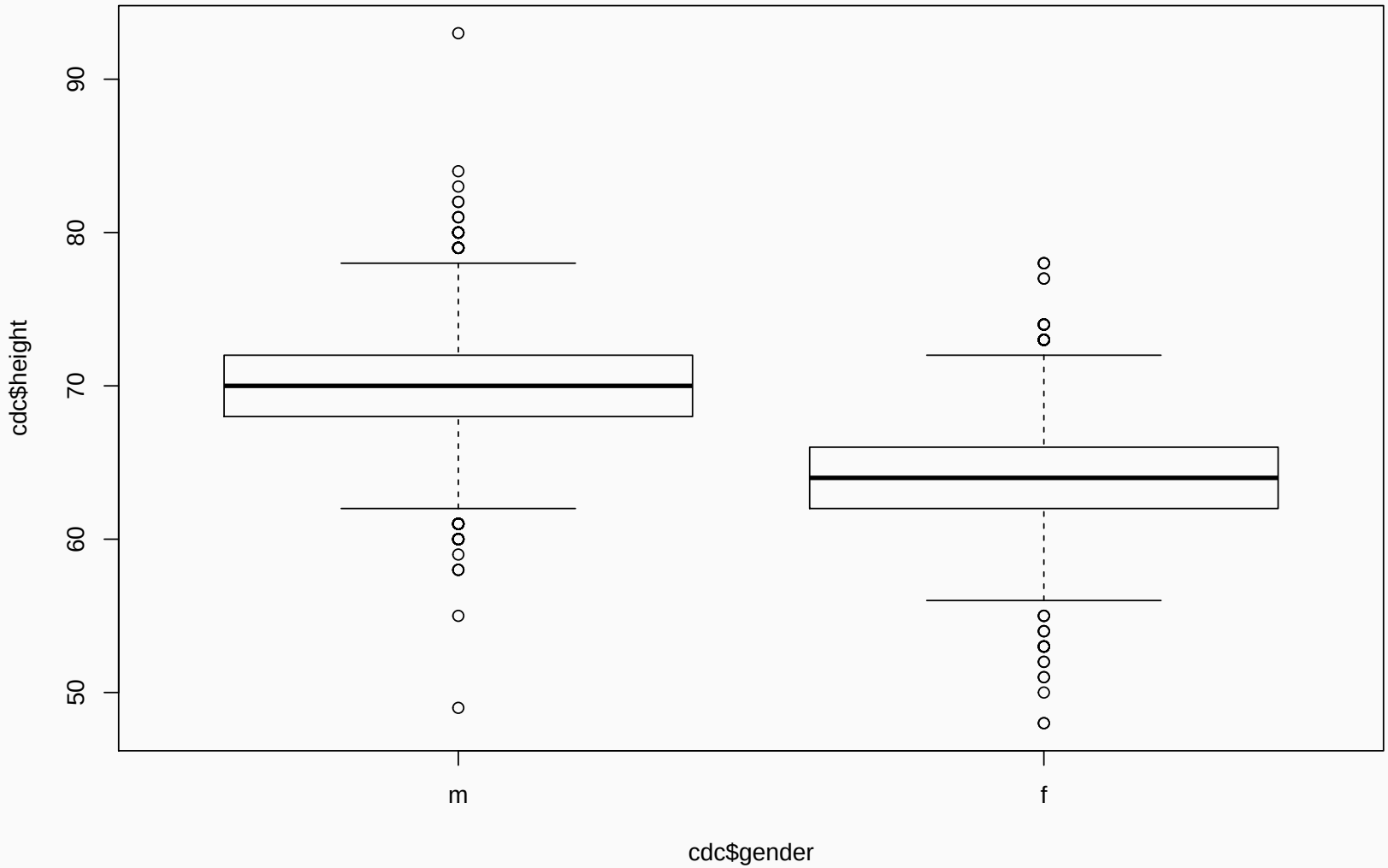
```
cdc$age > 30
```

```
##      [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##     [13] FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
##     [25] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##     [37] FALSE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
##     [49]  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE
##     [61]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##     [73] FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##     [85]  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
##     [97]  TRUE  TRUE  TRUE FALSE  TRUE FALSE FALSE  TRUE
##    [109]  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
```

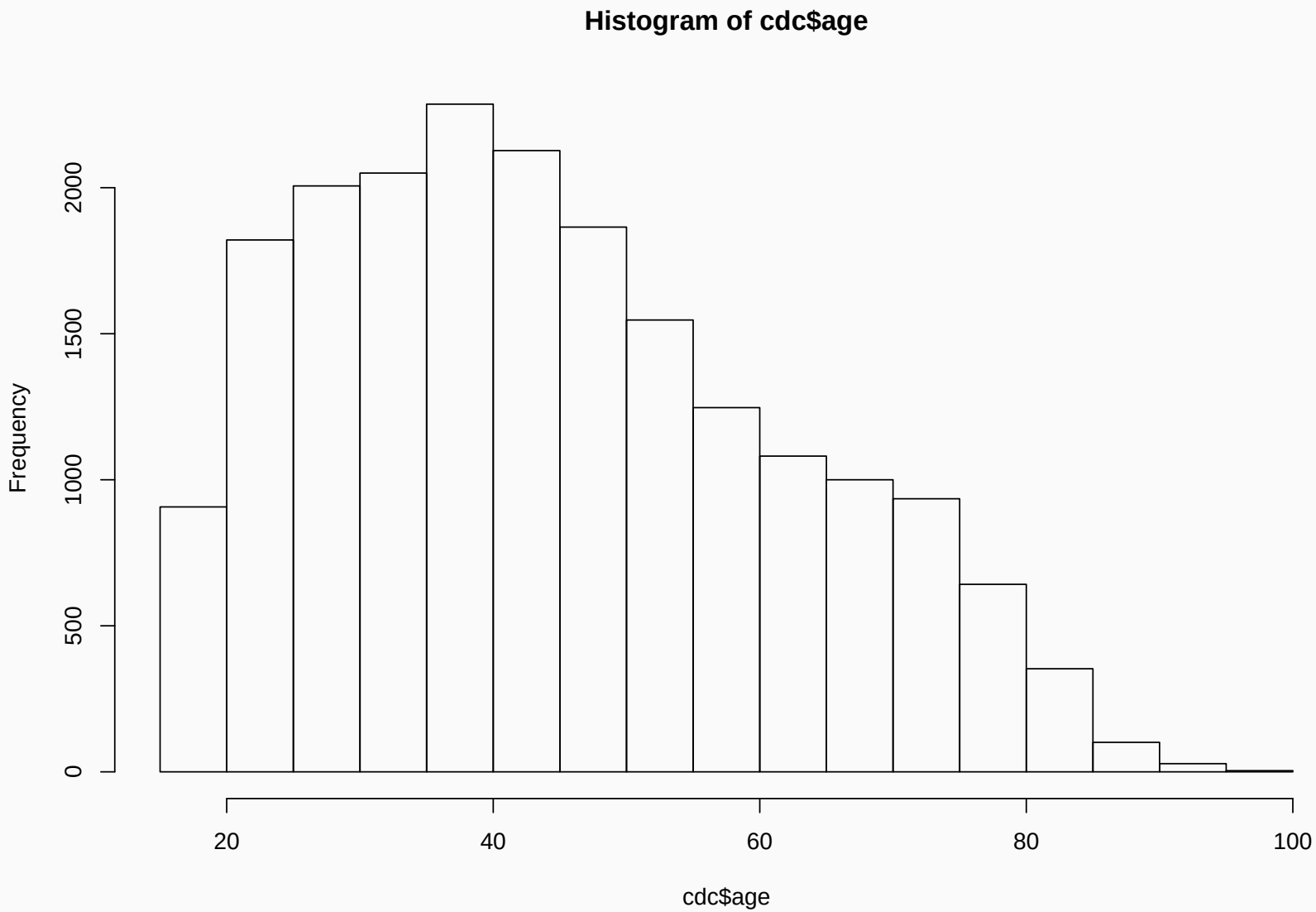
```
boxplot(cdc$height)
```



```
boxplot(cdc$height ~ cdc$gender)
```



```
hist(cdc$age)
```


Histogram of cdc$age
Frequency
0
500
1000
1500
2000
20
40
60
80
100
cdc$age

Introduction to Data - with dyplyr

- The Bureau of Transportation Statistics (BTS) is a statistical agency that is a part of the Research and Innovative Technology Administration (RITA). As its name implies, BTS collects and makes available transportation data, such as the flights data we will be working with in this lab.
- We begin by loading the nycflights data frame. Type the following in your console to load the data:

```
library(statsr)
```

```
## Loading required package: BayesFactor

## Loading required package: coda

## Loading required package: Matrix

##
```

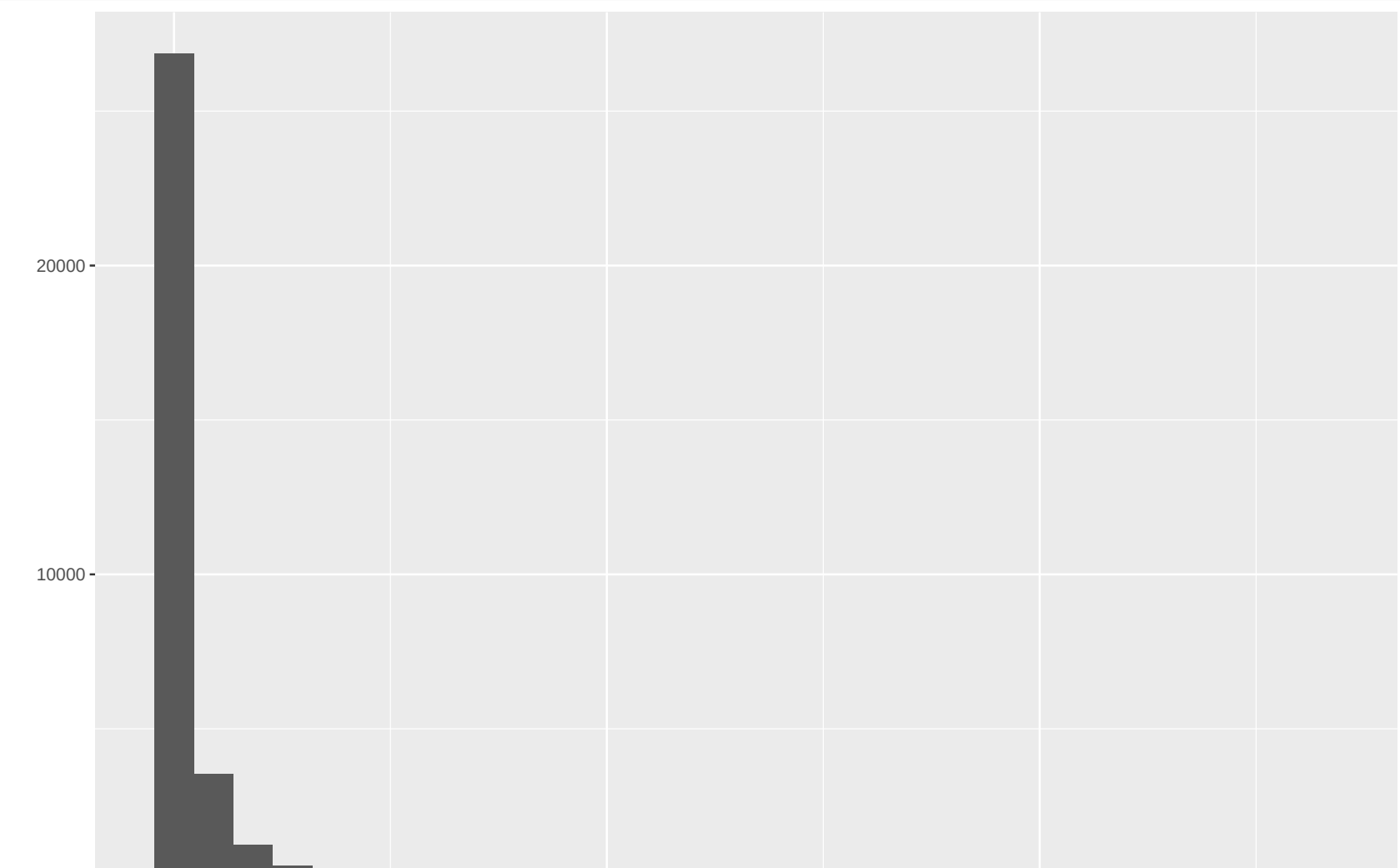
- A very useful function for taking a quick peek at your data frame and viewing its dimensions and data types is `str`, which stands for structure.

```
str(nycflights)
```

```
## tibble [32,735 x 16] (S3: tbl_df/data.frame)
##  $ year     : int [1:32735] 2013 2013 2013 2013 2013 201
##  $ month    : int [1:32735] 6 5 12 5 7 1 12 8 9 4 ...
##  $ day      : int [1:32735] 30 7 8 14 21 1 9 13 26 30 .
##  $ dep_time : int [1:32735] 940 1657 859 1841 1102 1817
##  $ dep_delay: num [1:32735] 15 -3 -1 -4 -3 -3 14 85 -10
##  $ arr_time : int [1:32735] 1216 2104 1238 2122 1230 200
##  $ arr_delay: num [1:32735] -4 10 11 -34 -8 3 22 71 -8 6
##  $ carrier  : chr [1:32735] "VX" "DL" "DL" "DL" ...
##  $ tailnum  : chr [1:32735] "N626VA" "N3760C" "N712TW" "
```

- Let's start by examing the distribution of departure delays of all flights with a histogram.

```
qplot(x = dep_delay, data = nycflights, geom = "histogram")

## `stat_bin()` using `bins = 30`. Pick better value with `
```



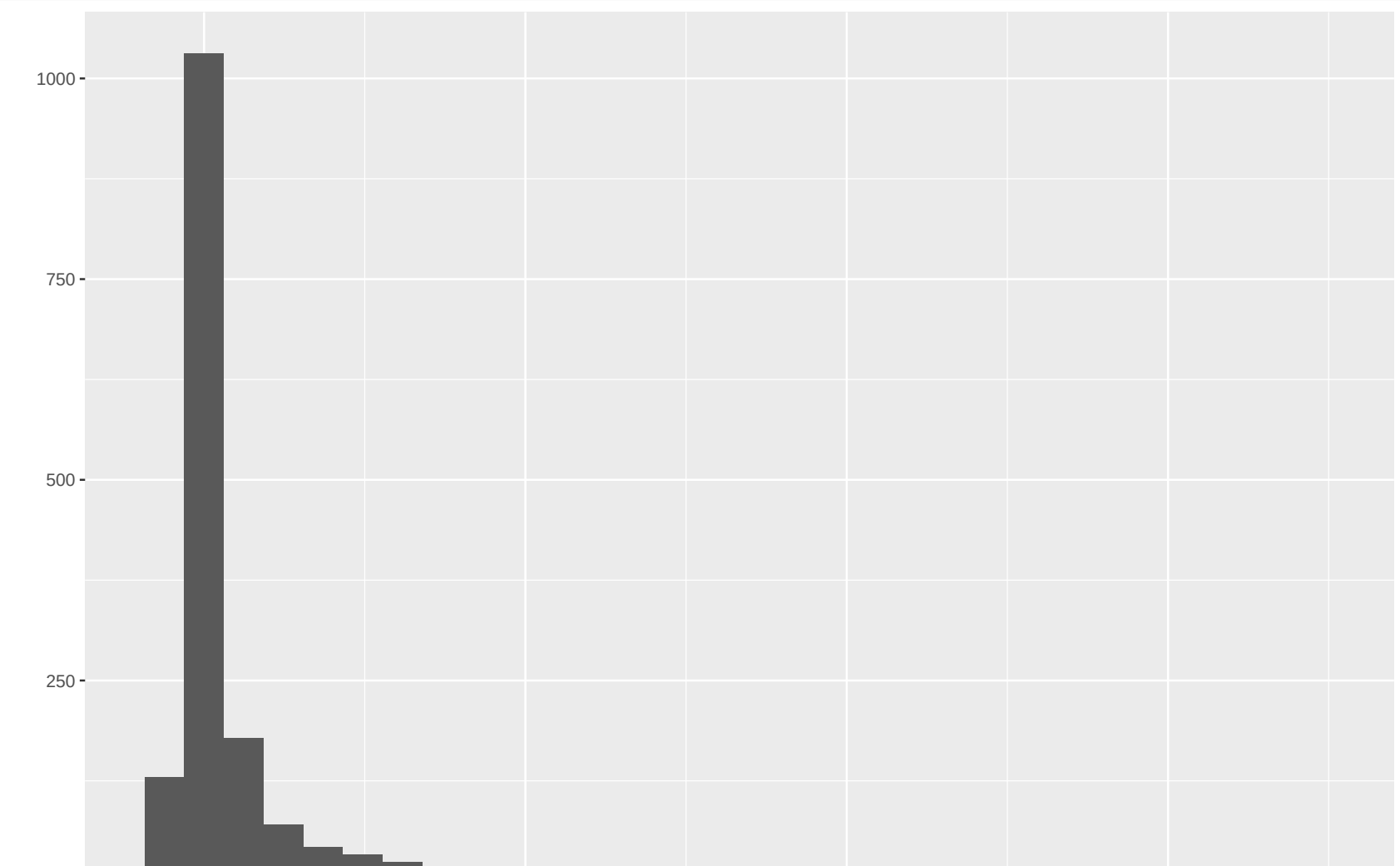
- If we want to focus only on departure delays of flights headed to Los Angeles, we need to first filter the data for flights with that destination (dest == "LAX") and then make a histogram of the departure delays of only those flights.

```
lax_flights <- nycflights %>%
  filter(dest == "LAX")
```

Introduction to Data - histogram

```
qplot(x = dep_delay, data = lax_flights, geom = "histogram"
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `
```



- Command 1: Take the `nycflights` data frame, filter for flights headed to LAX, and save the result as a new data frame called `lax_flights`.
  - == means "if it's equal to".
  - LAX is in quotation marks since it is a character string.
- Command 2: Basically the same qplot call from earlier for making a histogram, except that it uses the smaller data frame for flights headed to LAX instead of all flights.

```
lax_flights %>%
  summarise(mean_dd = mean(dep_delay), median_dd = median(d
```

```
## # A tibble: 1 x 3
##   mean_dd median_dd     n
##     <dbl>     <dbl> <int>
## 1    9.78        -1  1583
```

- We can also filter based on multiple criteria. Suppose we are interested in flights headed to San Francisco (SFO) in February:

```
sfo_feb_flights <- nycflights %>%
  filter(dest == "SFO", month == 2)
```

- Another useful technique is quickly calculating summary statistics for various groups in your data frame. For example, we can modify the above command using the group_by function to get the same summary stats for each origin airport:

```
sfo_feb_flights %>%
  group_by(origin) %>%
  summarise(median_dd = median(dep_delay), iqr_dd = IQR(dep
```

```
## # A tibble: 2 x 4
##   origin median_dd iqr_dd n_flights
##   <chr>       <dbl>  <dbl>     <int>
## 1 EWR           0.5   5.75         8
## 2 JFK          -2.5  15.2         60
```

- Which month would you expect to have the highest average delay departing from an NYC airport?
- Let's think about how we would answer this question:
- First, calculate monthly averages for departure delays. With the new language we are learning, we need to group_by months, then summarise mean departure delays.
- Then, we need to arrange these average delays in descending order

```
nycflights %>%
  group_by(month) %>%
  summarise(mean_dd = mean(dep_delay)) %>%
  arrange(desc(mean_dd))
```



```
## # A tibble: 12 x 2
```

Probability

```
library(dplyr)
library(ggplot2)
library(oilabs)
```

```
##
## Attaching package: 'oilabs'
```

```
## The following objects are masked from 'package:statsr':
##
##     calc_streak, inference, plot_ss, rep_sample_n
```

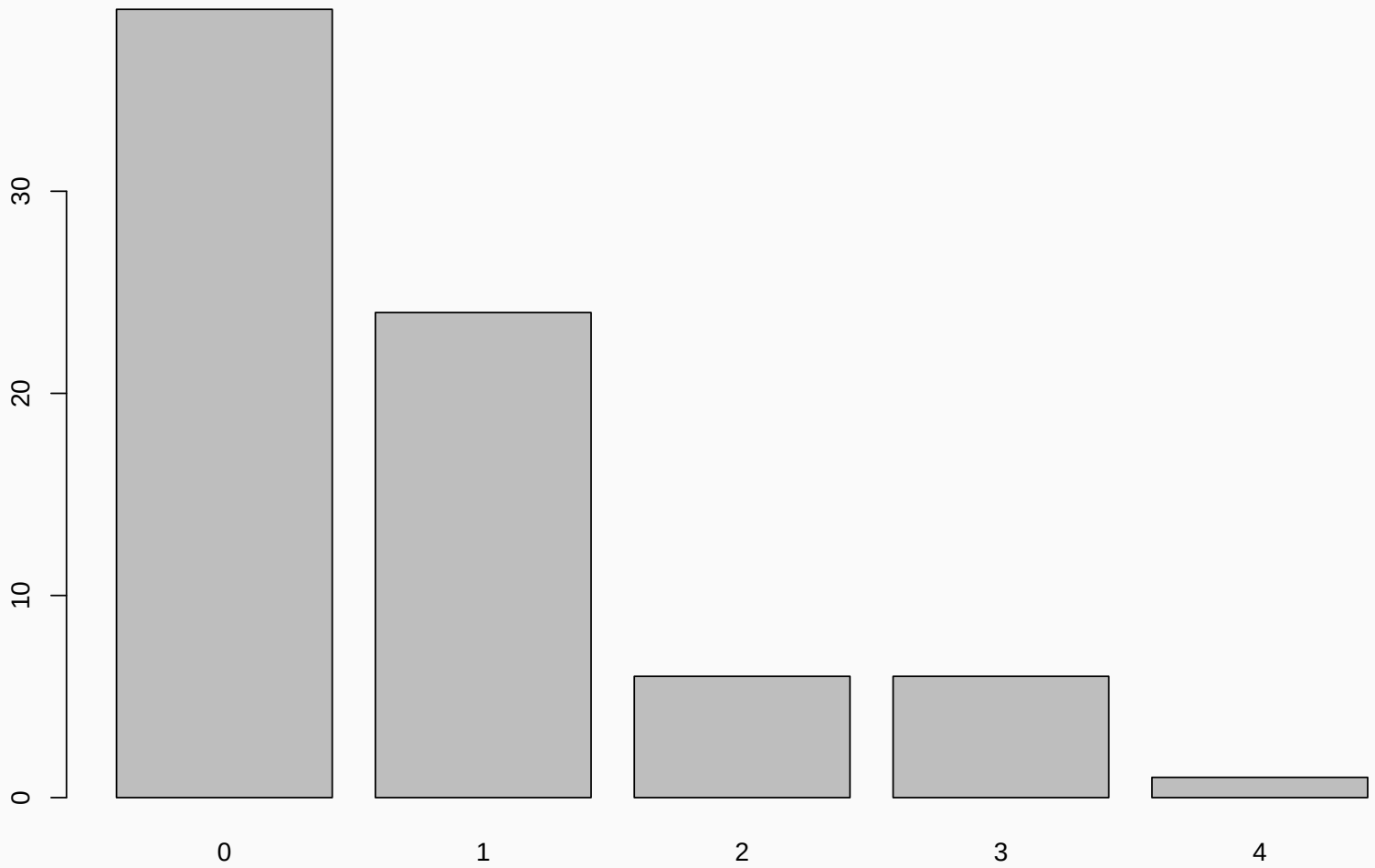
```
data(kobe_basket)
```

##Probability

- Our investigation will focus on the performance of one player: Kobe Bryant of the Los Angeles Lakers. His performance against the Orlando Magic in the 2009 NBA Finals earned him

Probability

```
barplot(table(kobe_streak))
```



Probability

- We've shown that Kobe had some long shooting streaks, but are they long enough to support the belief that he had a hot hand? What can we compare them to?

```
outcomes <- c("heads", "tails")
sample(outcomes, size = 1, replace = TRUE)
```

```
## [1] "heads"
```

```
sim_fair_coin <- sample(outcomes, size = 100, replace = TRU
table(sim_fair_coin)
```

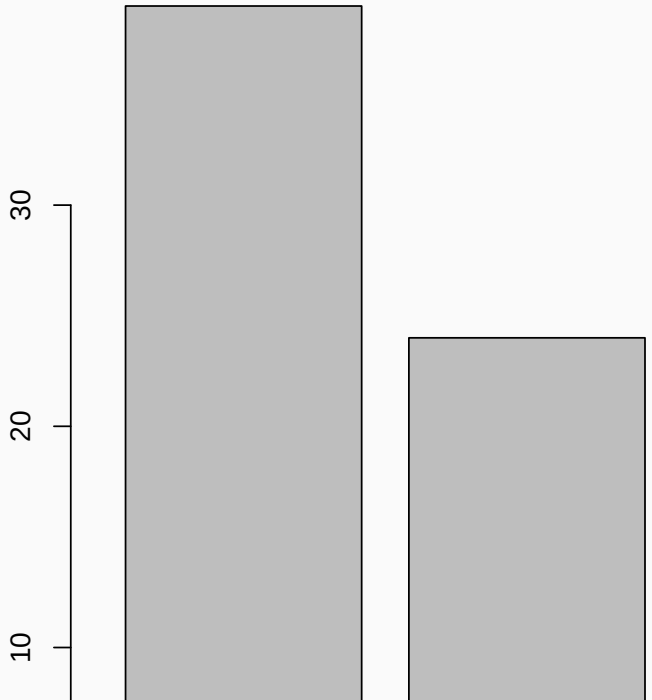
```
## sim_fair_coin
## heads tails
##    52    48
```

```
sim_unfair_coin <- sample(outcomes, size = 100, replace = T
table(sim_unfair_coin)
```

```
## sim_unfair_coin
## heads tails
##    14    86
```

Probability

```
outcomes <- c("H", "M")
sim_basket <- sample(outcomes, size = 133, replace = TRUE)

kobe_streak <-calc_streak(kobe_basket$shot)
barplot(table(kobe_streak))
```

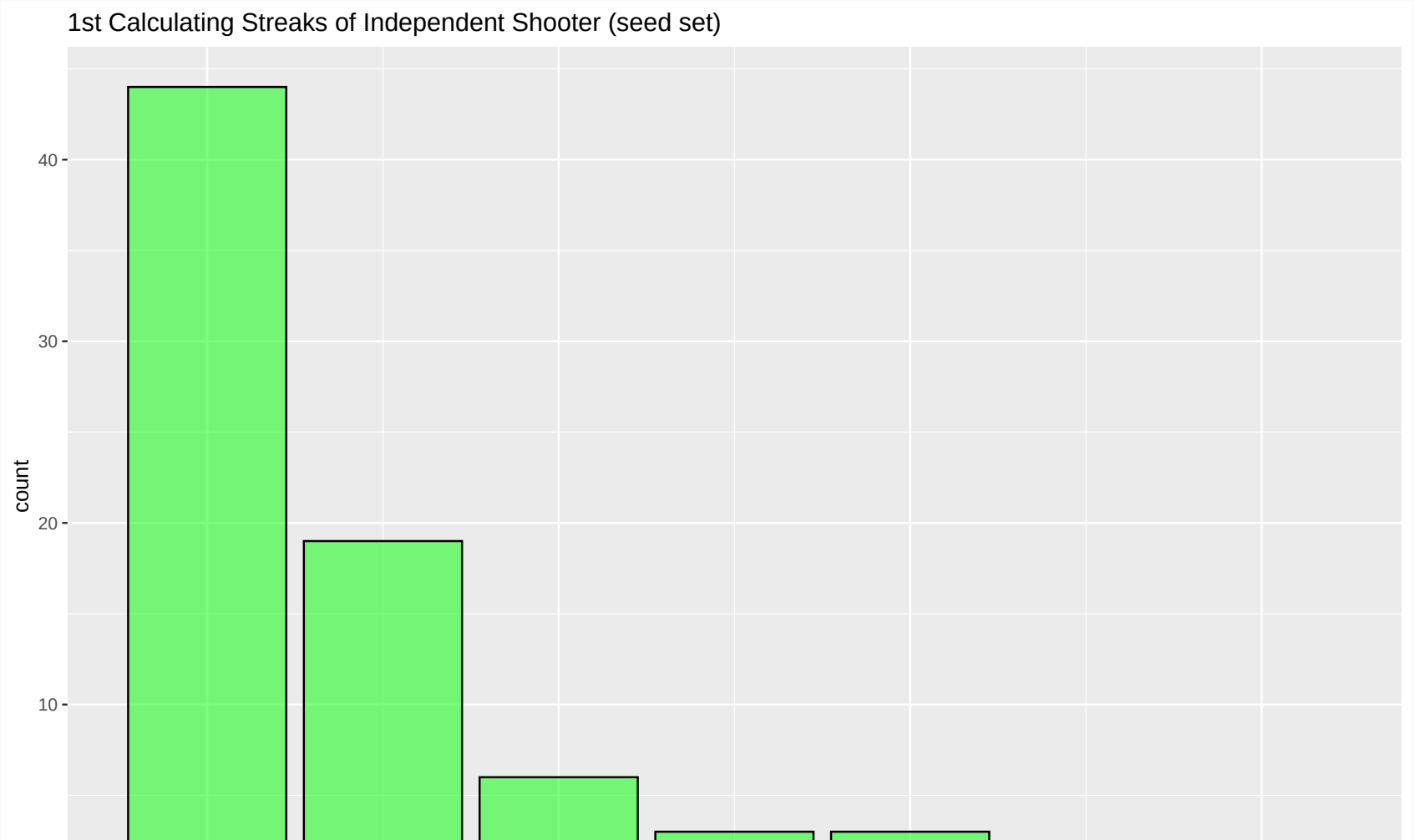


Probability

```
set.seed(22322)
shot_outcomes <- c("H", "M")
sim_basket <- sample(shot_outcomes, size = 133, replace = T
                     prob = c(0.45, 0.55))


sim_streak <- calc_streak(sim_basket)
```

Probability - ggplot

```
ggplot(sim_streak, aes(x = length)) +
                  ggtitle("1st Calculating Streaks of Indepen
                  geom_bar(alpha = 0.5, fill = "green", colou
```



The Normal Distribution

- We'll be working with measurements of body dimensions. This data set contains measurements from 247 men and 260 women, most of whom were considered healthy young adults. Let's take a quick peek at the first few rows of the data.

```
library(dplyr)
library(ggplot2)
library(oilabs)
```

The Normal Distribution

```
data(bdims)
head(bdims)
```

```
## # A tibble: 6 x 25
##   bia.di bii.di bit.di che.de che.di elb.di wri.di kne.d
##    <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl
## 1   42.9   26     31.5   17.7   28     13.1   10.4   18.
## 2   43.7   28.5   33.5   16.9   30.8   14     11.8   20.
## 3   40.1   28.2   33.3   20.9   31.7   13.9   10.9   19.
## 4   44.3   29.9   34     18.4   28.2   13.9   11.2   20.
## 5   42.5   29.9   34     21.5   29.4   15.2   11.6   20.
## 6   43.3   27     31.5   19.6   31.3   14     11.5   18.
## # ... with 14 more variables: wai.gi <dbl>, nav.gi <dbl>
## #   thi.gi <dbl>, bic.gi <dbl>, for.gi <dbl>, kne.gi <db
## #   ank.gi <dbl>, wri.gi <dbl>, age <int>, wgt <dbl>, hg
```

The Normal Distribution

- We'll be focusing on just three columns to get started: weight in kg (wgt), height in cm (hgt), and sex (m indicates male, f indicates female).
- Since males and females tend to have different body dimensions, it will be useful to create two additional data sets: one with only men and another with only women.

```
mdims <- bdims %>%
  filter(sex == "m")
fdims <- bdims %>%
  filter(sex == "f")
```

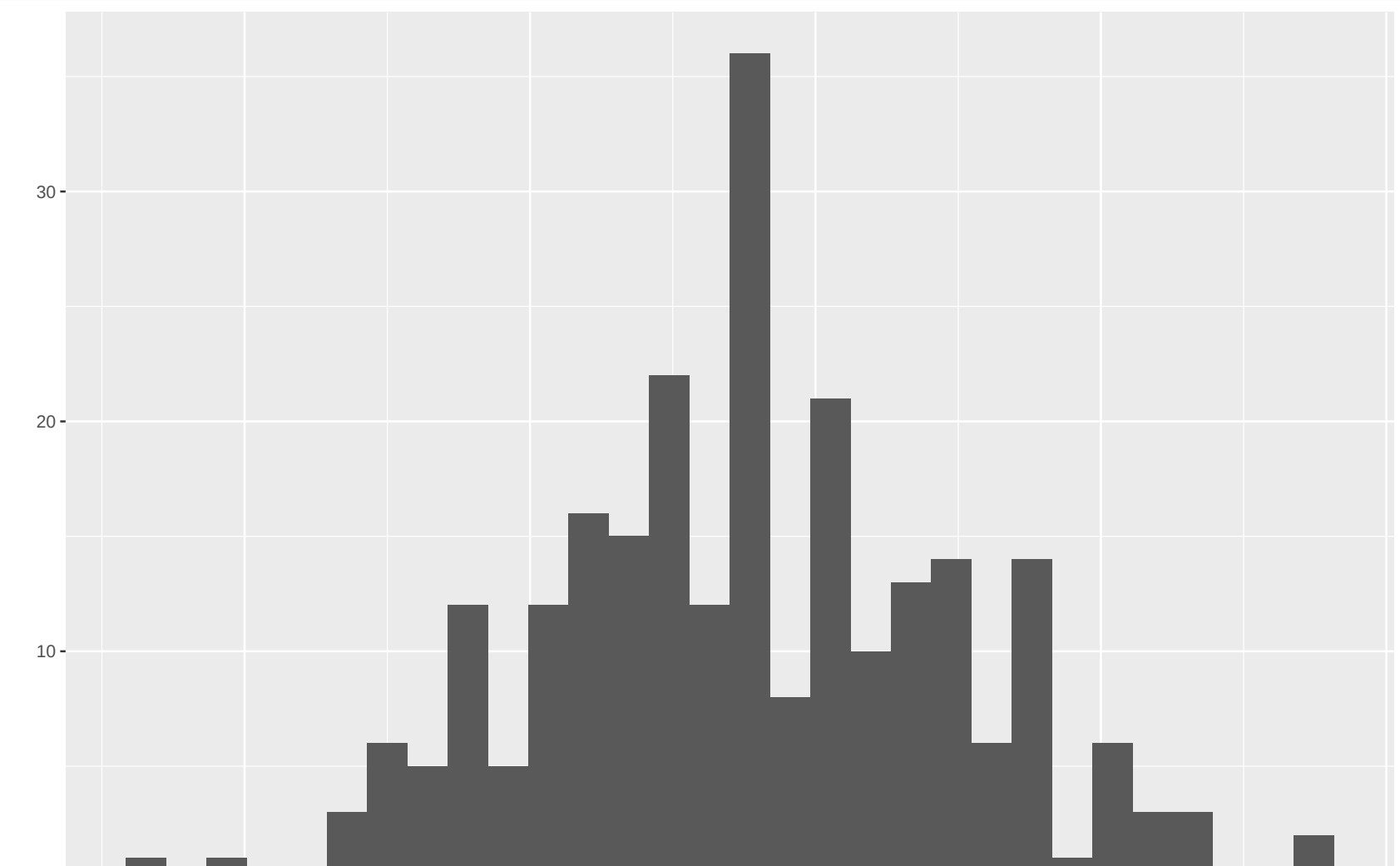
The Normal Distribution

- Make a plot (or plots) to visualize the distributions of men's and women's heights. How do their centers, shapes, and spreads compare?

The Normal Distribution - Histogram

```
qplot(data = mdims, x = hgt, geom = "histogram")
```

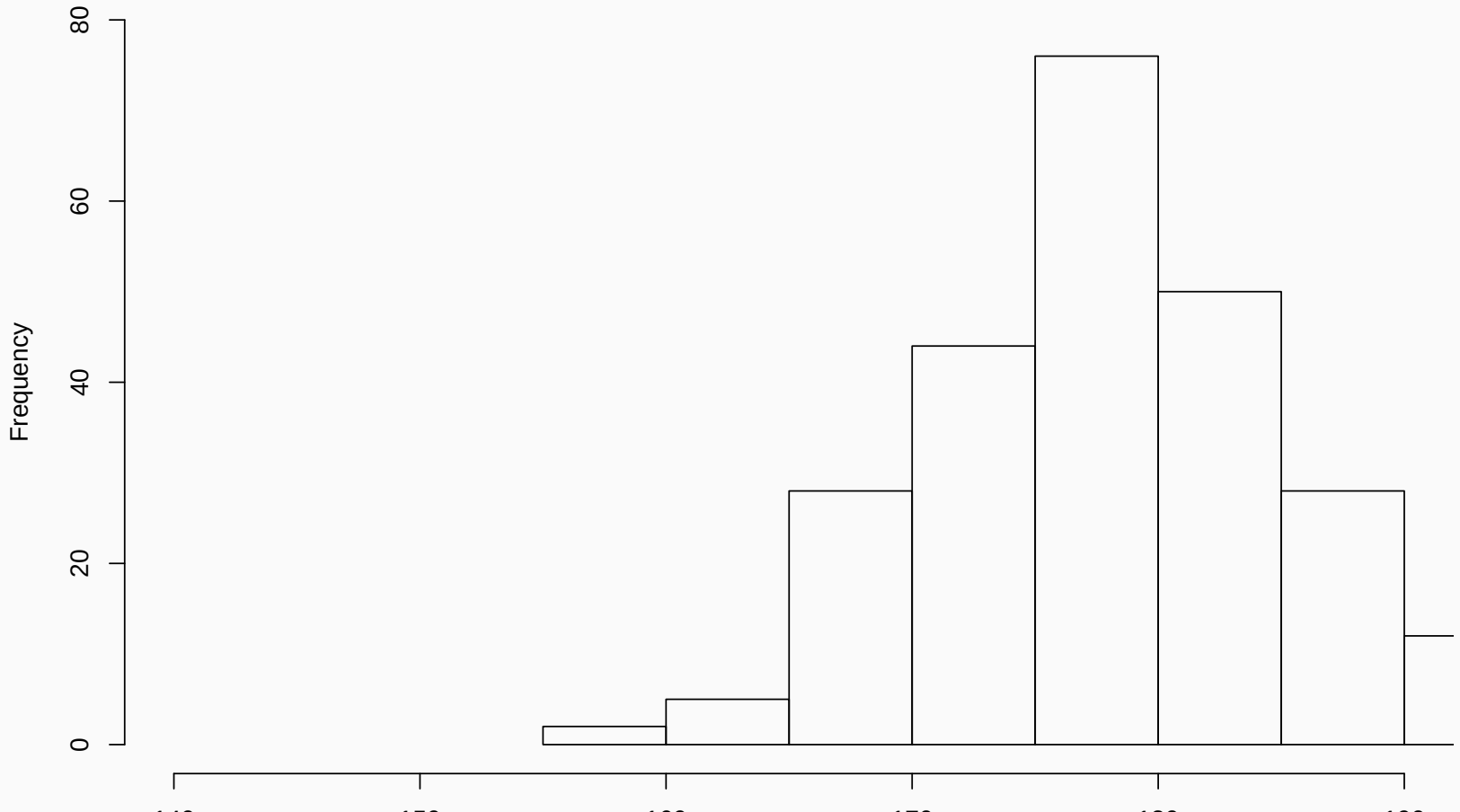
```
## `stat_bin()` using `bins = 30`. Pick better value with `
```



The Normal Distribution

- How does this compare to the regular histogram function

```
hist(mdims$hgt, xlab = "Male Height", main = "", xlim = c(1
```



The Normal Distribution

- Does it look bell-shaped or normal? It's tempting to say so when faced with a unimodal symmetric distribution.
- To see how accurate that description is, we can plot a normal distribution curve on top of a histogram to see how closely the data follow a normal distribution.
- This normal curve should have the same mean and standard deviation as the data.
- We'll be working with women's heights, so let's store them as a separate object and then calculate some statistics that will be referenced later.

The Normal Distribution

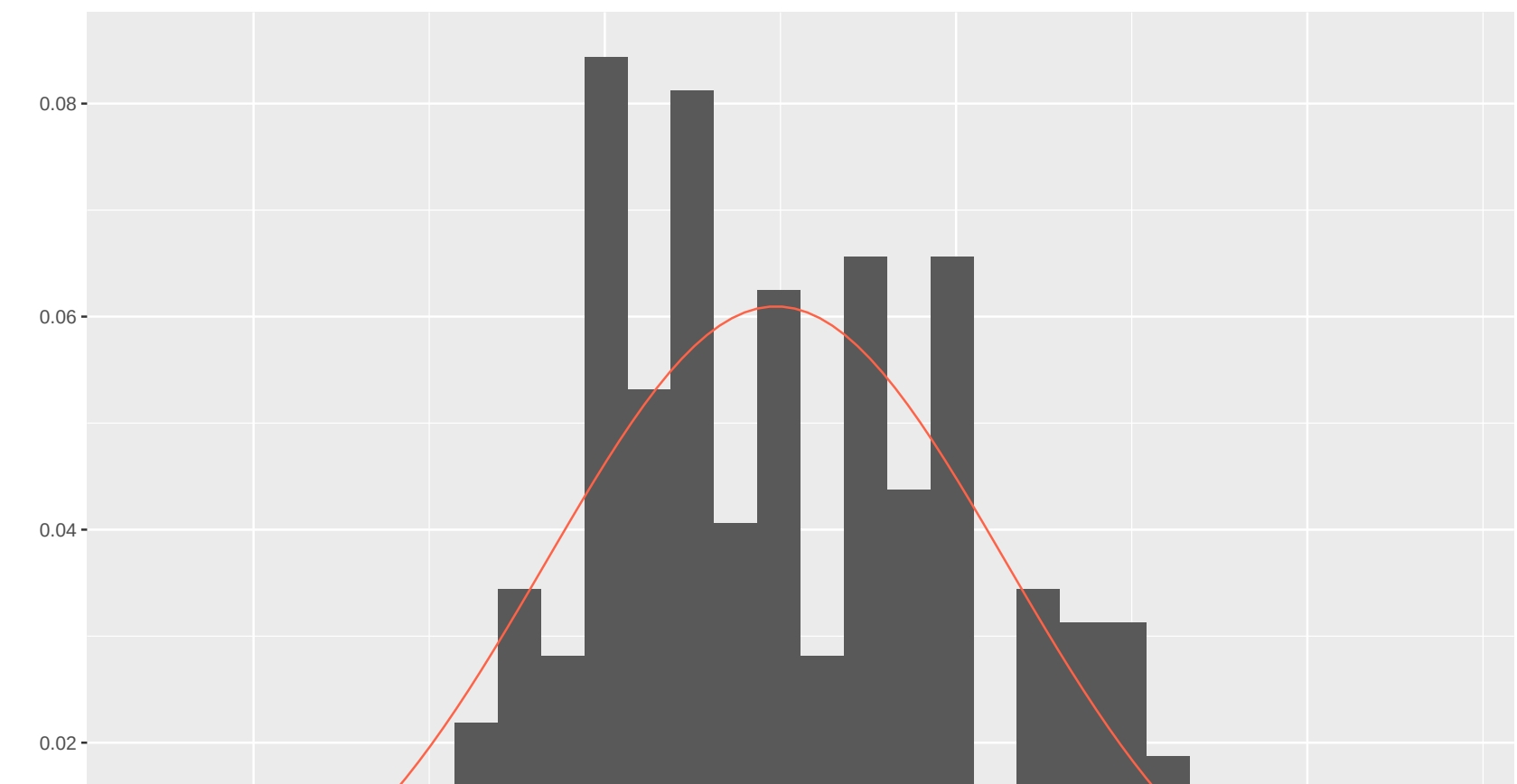
```
fhgtmean <- mean(fdims$hgt)
fhgtsd   <- sd(fdims$hgt)
```

- Next we make a density histogram to use as the backdrop and use the lines function to overlay a normal probability curve.
- The difference between a frequency histogram and a density histogram is that while in a frequency histogram the heights of the bars add up to the total number of observations, in a density histogram the areas of the bars add up to 1.
- The area of each bar can be calculated as simply the height times the width of the bar. Using a density histogram allows us to properly overlay a normal distribution curve over the histogram since the curve is a normal probability density function that also has area under the curve of 1.

The Normal Distribution

```
qplot(x = hgt, data = fdims, geom = "blank") +
  geom_histogram(aes(y = ..density..)) +
  stat_function(fun = dnorm, args = c(mean = fhgtmean, sd =
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `
```

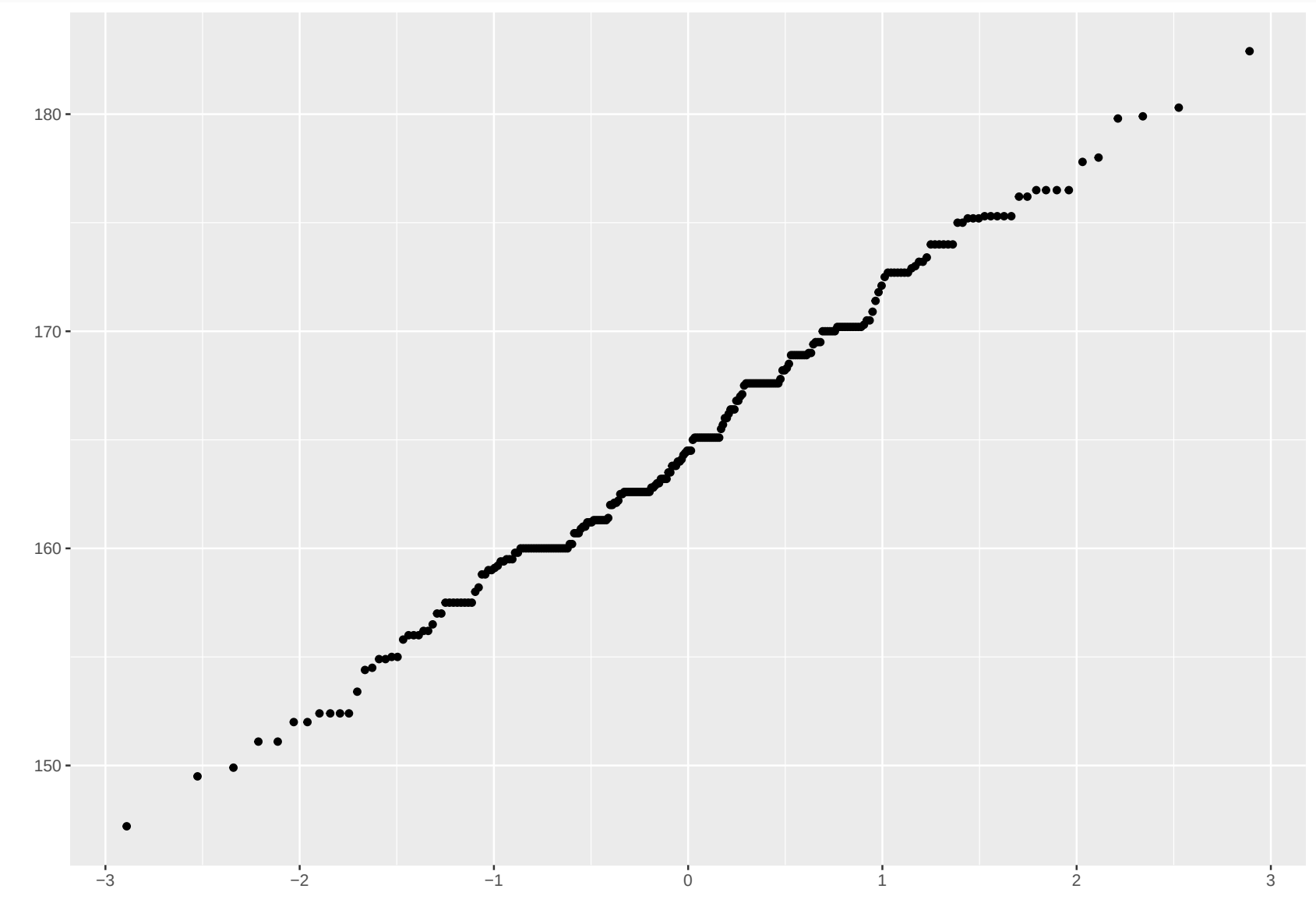


The Normal Distribution

- Eyeballing the shape of the histogram is one way to determine if the data appear to be nearly normally distributed, but it can be frustrating to decide just how close the histogram is to the curve.
- An alternative approach involves constructing a normal probability plot, also called a normal Q-Q plot for "quantile-quantile".

The Normal Distribution: Q-Q plot

```
qplot(sample = hgt, data = fdims, geom = "qq")
```



The Normal Distribution

- The x-axis values correspond to the quantiles of a theoretically normal curve with mean 0 and standard deviation 1 (i.e., the standard normal distribution).
- The y-axis values correspond to the quantiles of the original unstandardized sample data. However, even if we were to standardize the sample data values, the Q-Q plot would look identical.

The Normal Distribution

- A data set that is nearly normal will result in a probability plot where the points closely follow a diagonal line. Any deviations from normality leads to deviations of these points from that line.
- The plot for female heights shows points that tend to follow the line but with some errant points towards the tails. We're left with the same problem that we encountered with the histogram above: how close is close enough?

The Normal Distribution

- A useful way to address this question is to rephrase it as: what do probability plots look like for data that I know came from a normal distribution?
- We can answer this by simulating data from a normal distribution using rnorm.

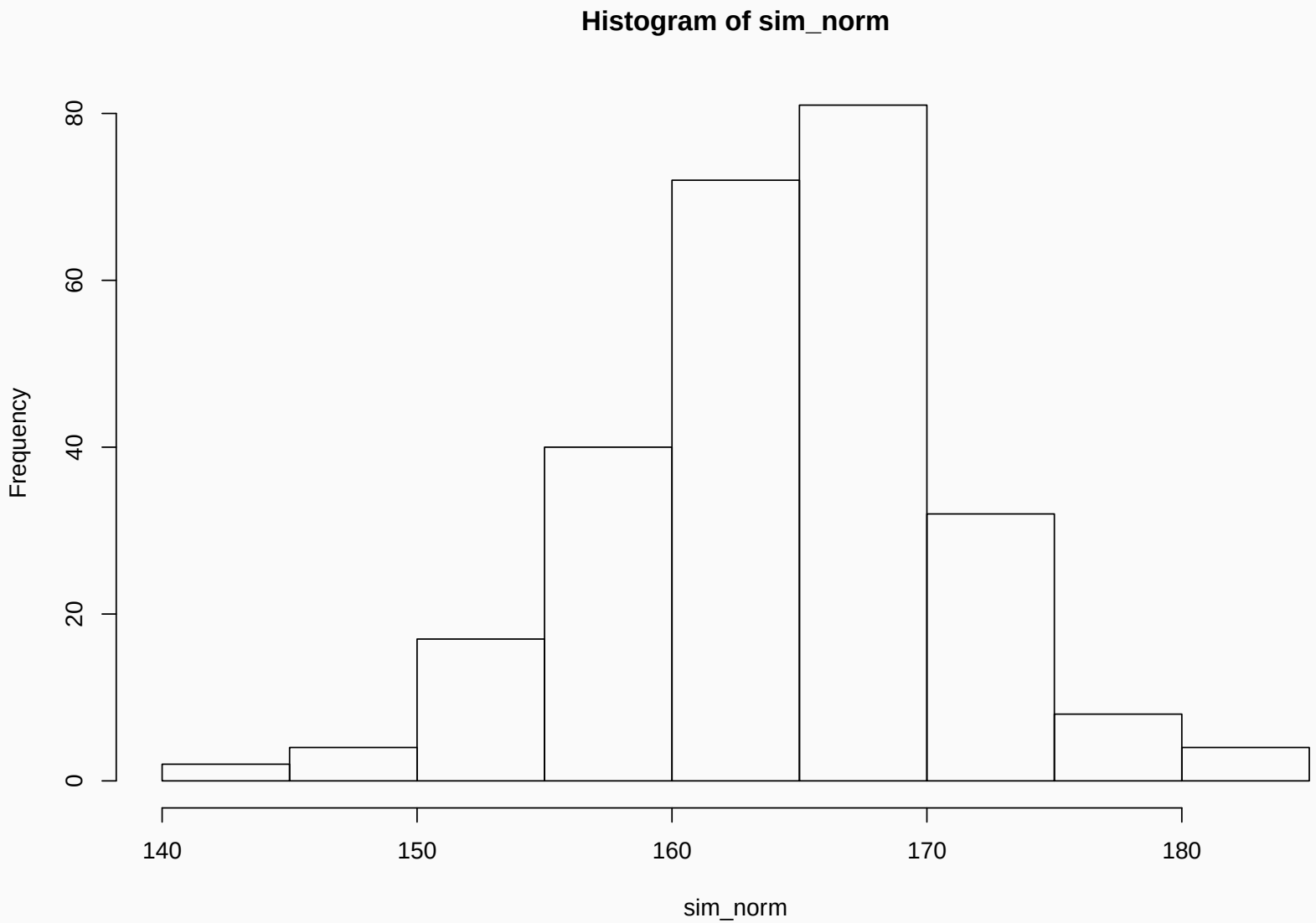
The Normal Distribution

```
sim_norm <- rnorm(n = nrow(fdims), mean = fhgtmean, sd = fh
```

- The first argument indicates how many numbers you'd like to generate, which we specify to be the same number of heights in the fdims data set using the nrow() function.
- The last two arguments determine the mean and standard deviation of the normal distribution from which the simulated sample will be generated.
- We can take a look at the shape of our simulated data set, sim_norm, as well as its normal probability plot.

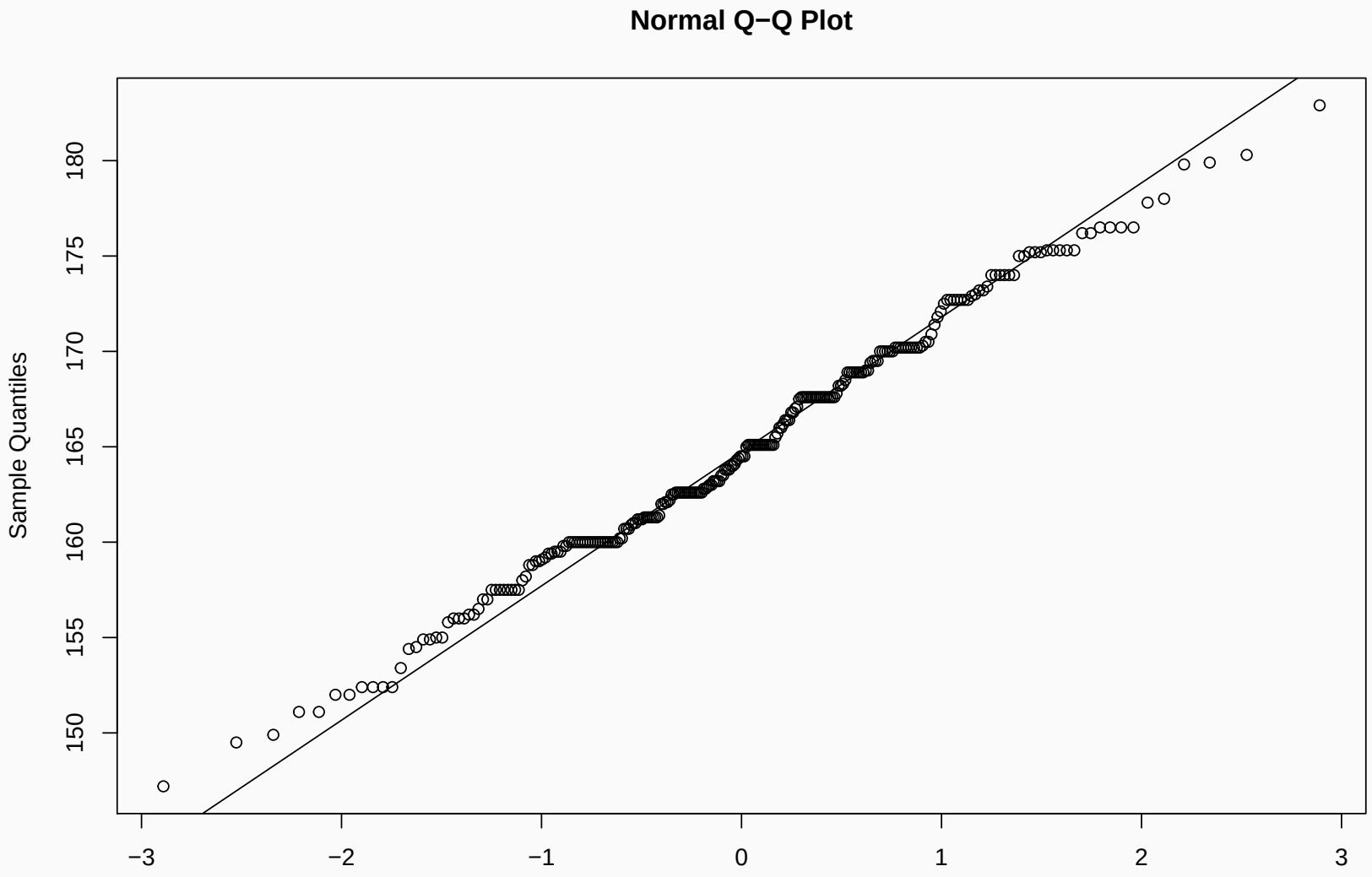
The Normal Distribution

```
hist(sim_norm)
```



The Normal Distribution

```
qqnorm(fdims$hgt)
qqline(fdims$hgt)
```

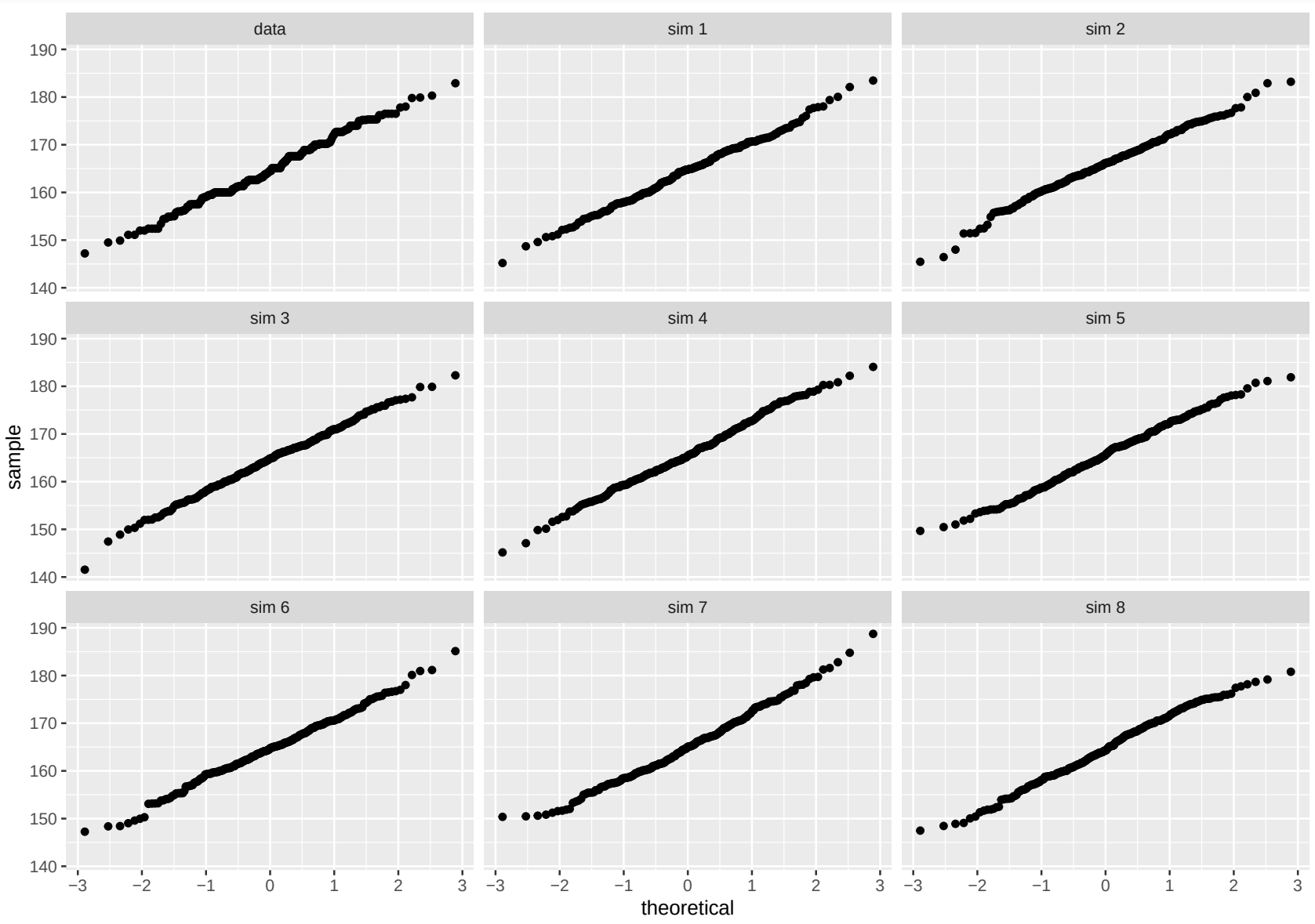


The Normal Distribution

- Even better than comparing the original plot to a single plot generated from a normal distribution is to compare it to many more plots using the following function. It shows the Q-Q plot corresponding to the original data in the top left corner, and the Q-Q plots of 8 different simulated normal data.

The Normal Distribution

```
qqnormsim(sample = hgt, data = fdims)
```



The Normal Probabilities

- Okay, so now you have a slew of tools to judge whether or not a variable is normally distributed. Why should we care?
- It turns out that statisticians know a lot about the normal distribution. Once we decide that a random variable is approximately normal, we can answer all sorts of questions about that variable related to probability.
- Take, for example, the question of, "What is the probability that a randomly chosen young adult female is taller than 6 feet (about 182 cm)?"

The Normal Probabilities

- If we assume that female heights are normally distributed (a very close approximation is also okay), we can find this probability by calculating a Z score and consulting a Z table (also called a normal probability table).
- In R, this is done in one step with the function pnorm().

```
1 - pnorm(q = 182, mean = fhgtmean, sd = fhgtsd)
```

```
## [1] 0.004434387
```

- Note that the function pnorm() gives the area under the normal curve below a given value, q, with a given mean and standard deviation.
- Since we're interested in the probability that someone is taller than 182 cm, we have to take one minus that probability.

The Normal Probabilities

- Assuming a normal distribution has allowed us to calculate a theoretical probability. If we want to calculate the probability empirically, we simply need to determine how many observations fall above 182 then divide this number by the total sample size.

The Normal Probabilities

```
fdims %>%
  filter(hgt > 182) %>%
  summarise(percent = n() / nrow(fdims))
```

```
## # A tibble: 1 x 1
##   percent
##     <dbl>
## 1 0.00385
```

```
sum(fdims$hgt > 182) / length(fdims$hgt)
```

```
## [1] 0.003846154
```

Sampling Distributions

- We consider real estate data from the city of Ames, Iowa. The details of every real estate transaction in Ames is recorded by the City Assessor's office.
- Our particular focus for this lab will be all residential home sales in Ames between 2006 and 2010. This collection represents our population of interest.
- We would like to learn about these home sales by taking smaller samples from the full population.

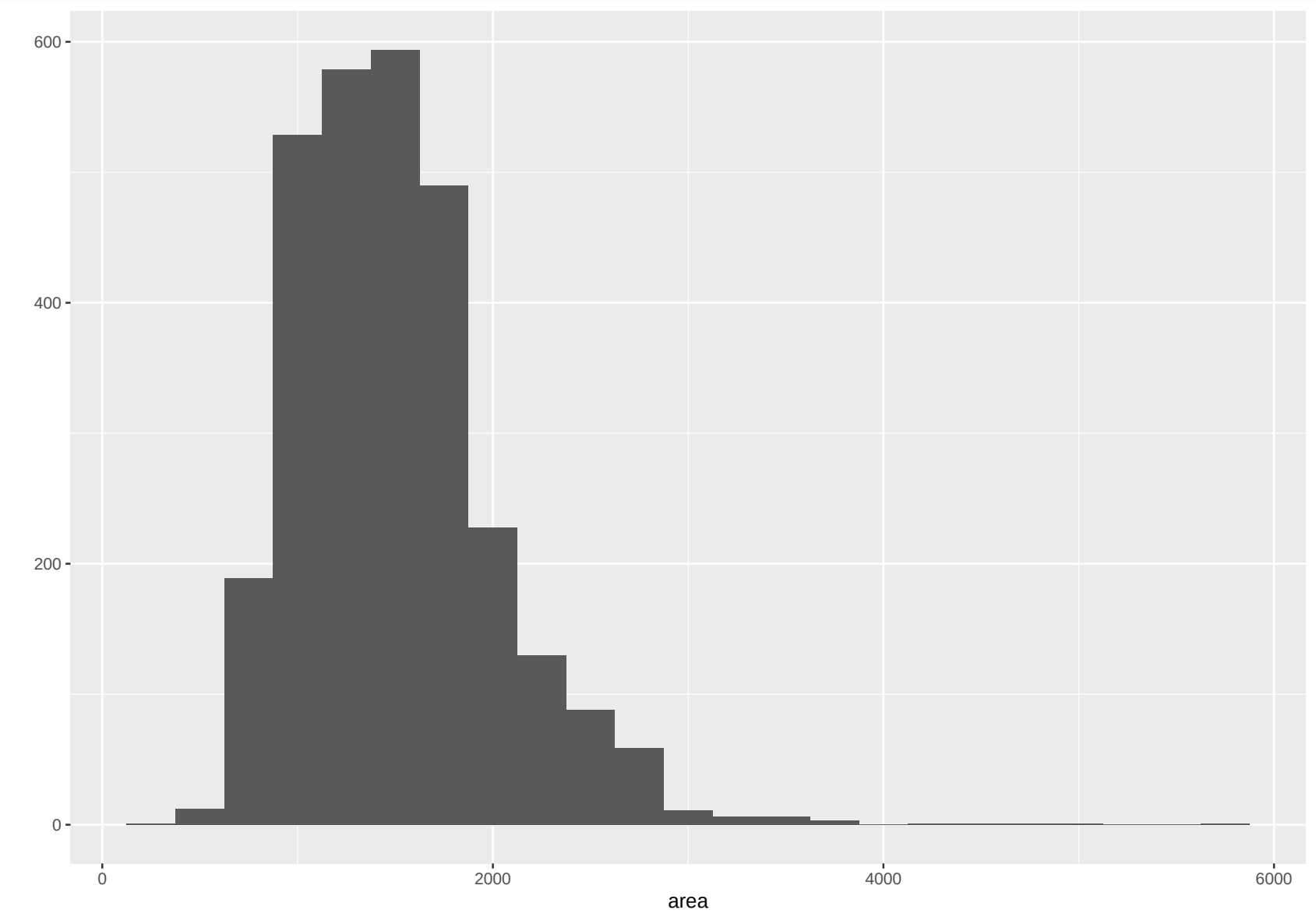
```
data(ames)
```

Sampling Distributions

- We see that there are quite a few variables in the data set, enough to do a very in-depth analysis. For this lab, we'll restrict our attention to just two of the variables: the above ground living area of the house in square feet (area) and the sale price (price).
- We can explore the distribution of areas of homes in the population of home sales visually and with summary statistics. Let's first create a visualization, a histogram:

Sampling Distributions

```
qplot(data = ames, x = area, binwidth = 250, geom = "histog
```



Sampling Distributions

- Let's also obtain some summary statistics. Note that we can do this using the summarise function.
- Some of the functions below should be self explanatory (like mean, median, sd, IQR, min, and max). A new function here is the quantile function which we can use to calculate values corresponding to specific percentile cutoffs in the distribution.
- Finding these values is useful for describing the distribution, as we can use them for descriptions like "the middle 50% of the homes have areas between such and such square feet".

Sampling Distributions

```
ames %>%
  summarise(mu = mean(area), pop_med = median(area),
            sigma = sd(area), pop_iqr = IQR(area),
            pop_min = min(area), pop_max = max(area),
            pop_q1 = quantile(area, 0.25),  # first quarti
            pop_q3 = quantile(area, 0.75))  # third quarti
```

```
## # A tibble: 1 x 8
##      mu pop_med sigma pop_iqr pop_min pop_max pop_q1 pop
##   <dbl>   <dbl> <dbl>   <dbl>   <int>   <int>  <dbl>  <d
## 1 1500.    1442  506.    617.     334    5642   1126  17
```

The unknown sampling distribution

- Here, we have access to the entire population, but this is rarely the case in real life.
- Gathering information on an entire population is often extremely costly or impossible. Because of this, we often take a sample of the population and use that to understand the properties of the population.
- If we were interested in estimating the mean living area in Ames based on a sample, we can use the sample_n command to survey the population.

```
samp1 <- ames %>%
  sample_n(50)
```

The unknown sampling distribution

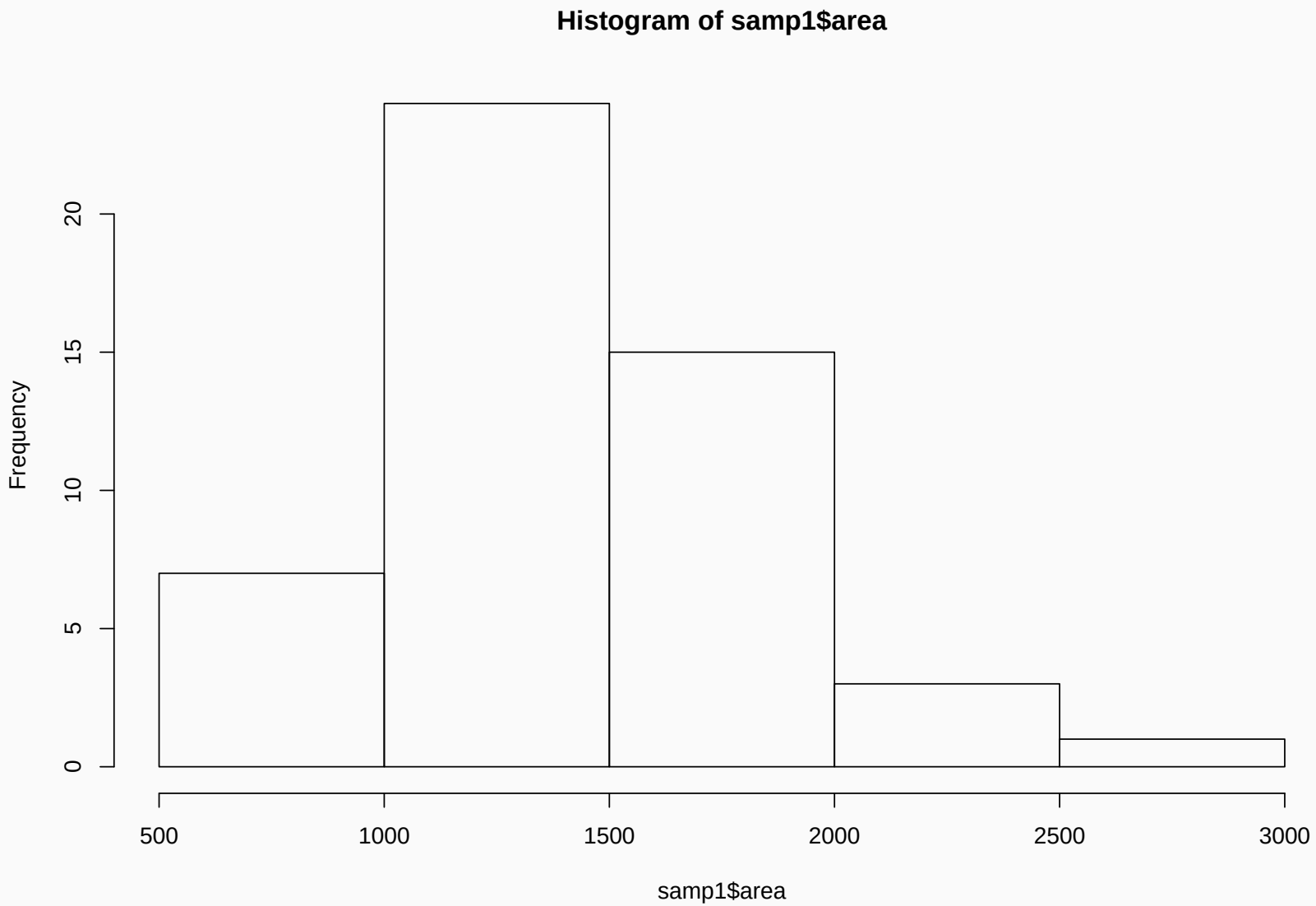
- If we're interested in estimating the average living area in homes in Ames using the sample, our best single guess is the sample mean.

```
samp1 %>%
  summarise(x_bar = mean(area))
```

```
## # A tibble: 1 x 1
##   x_bar
##   <dbl>
## 1 1420.
```

The unknown sampling distribution

```
hist(samp1$area)
```



The unknown sampling distribution

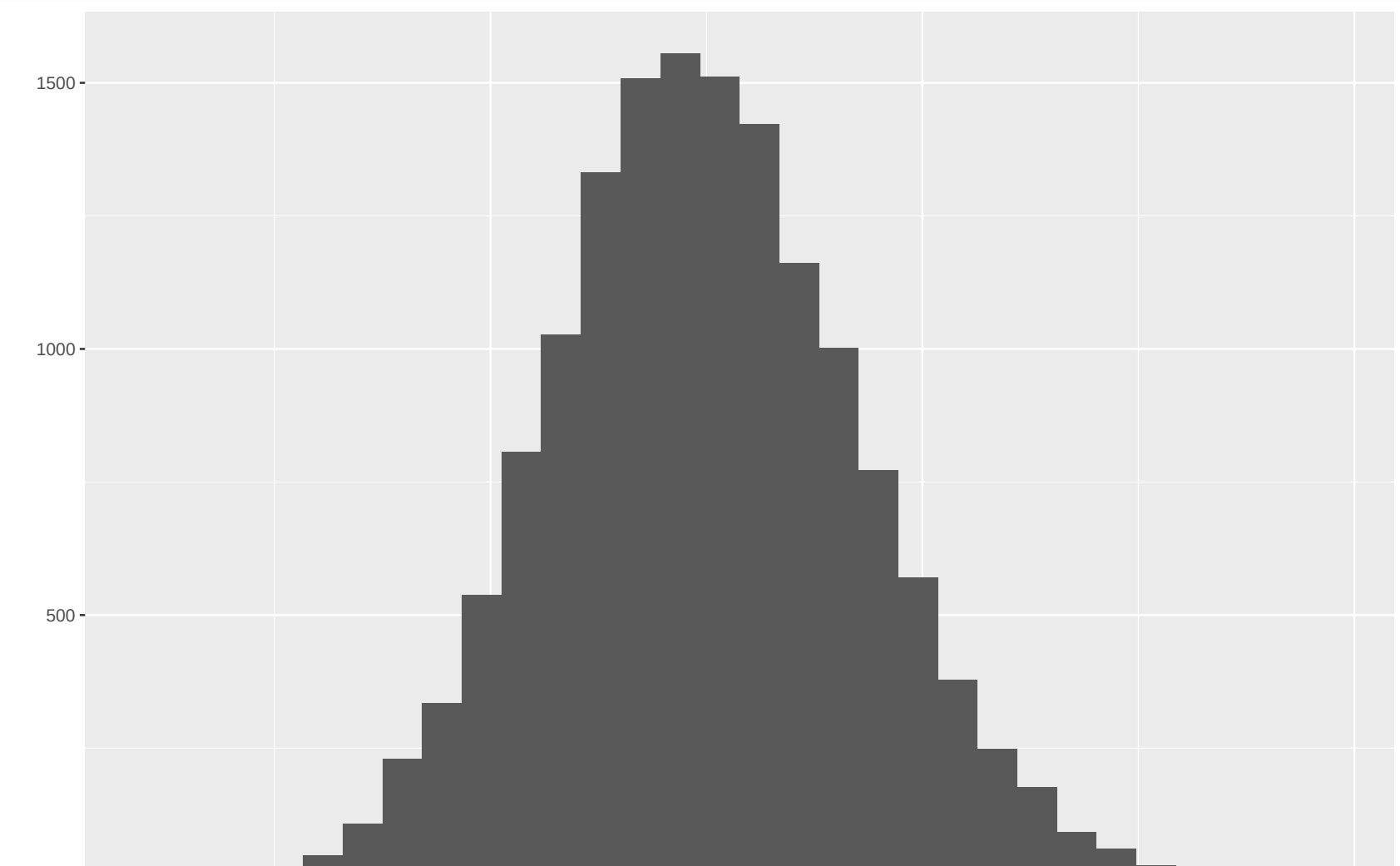
- Not surprisingly, every time we take another random sample, we get a different sample mean.
- It's useful to get a sense of just how much variability we should expect when estimating the population mean this way.
- The distribution of sample means, called the sampling distribution (of the mean), can help us understand this variability. In this lab, because we have access to the population, we can build up the sampling distribution for the sample mean by repeating the above steps many times.
- Here we will generate 15,000 samples and compute the sample mean of each. Note that we specify that replace = TRUE since sampling distributions are constructed by sampling with replacement.


```
sample_means50 <- ames %>%
```

The unknown sampling distribution

```
qplot(data = sample_means50, x = x_bar)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `
```



Interlude: Sampling distributions

- The idea behind the rep_sample_n function is repetition. Earlier we took a single sample of size n (50) from the population of all houses in Ames. With this new function we are able to repeat this sampling procedure rep times in order to build a distribution of a series of sample statistics, which is called the sampling distribution.
- Note that in practice one rarely gets to build true sampling distributions, because we rarely have access to data from the entire population.

Interlude: Sampling distributions

- Without the rep_sample_n function, this would be painful. We would have to manually run the following code 15,000 times

```
ames %>%
  sample_n(size = 50) %>%
  summarise(x_bar = mean(area))
```

```
## # A tibble: 1 x 1
##   x_bar
##   <dbl>
## 1 1316.
```

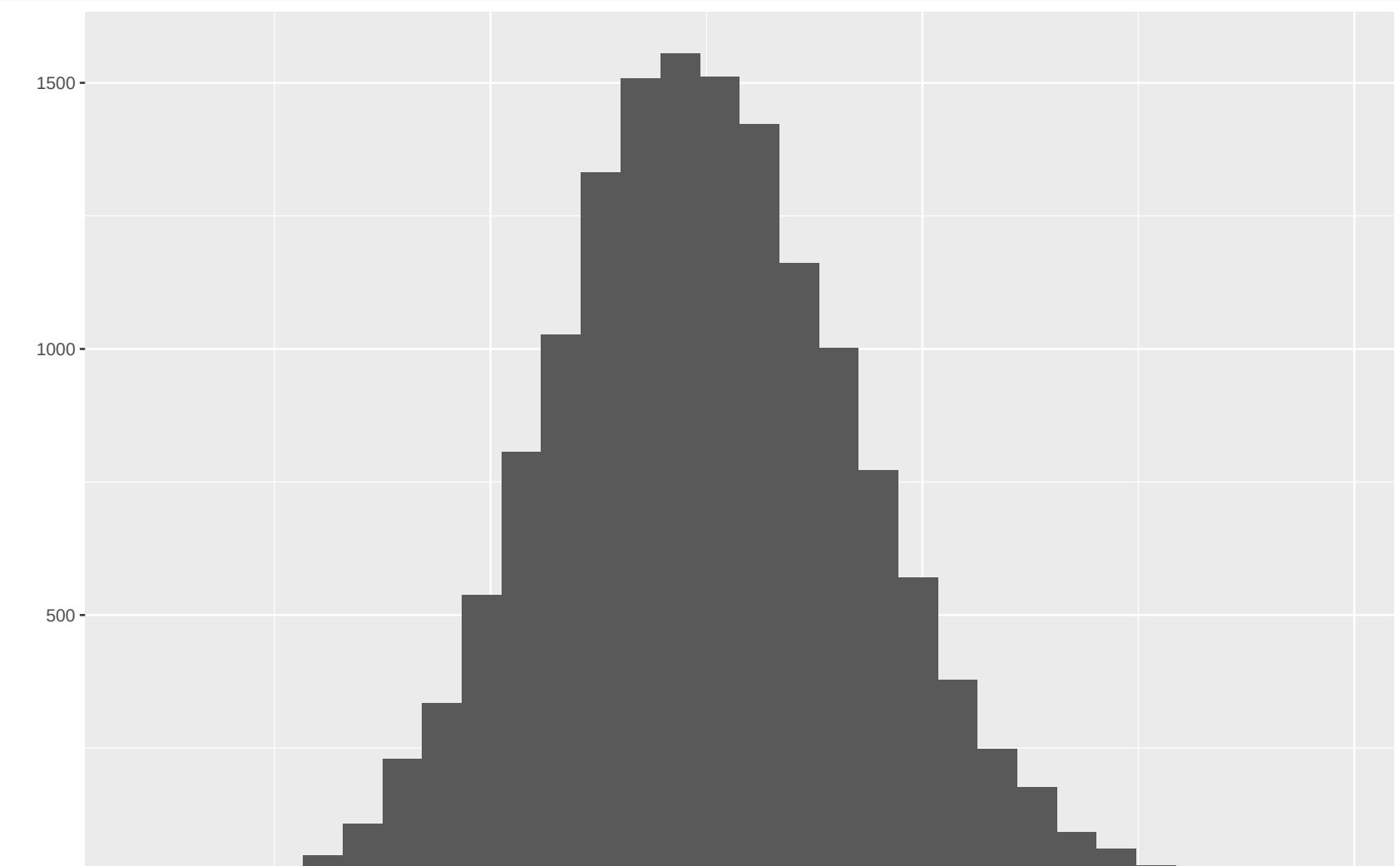
Sample size and the sampling distribution

- Mechanics aside, let's return to the reason we used the `rep_sample_n` function: to compute a sampling distribution, specifically, the sampling distribution of the mean home area for samples of 50 houses.

Sample size and the sampling distribution

```
qplot(data = sample_means50, x = x_bar, geom = "histogram")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `
```



- We consider real estate data from the city of Ames, Iowa. This is the same dataset used in the previous lab. The details of every real estate transaction in Ames is recorded by the City Assessor's office.
- Our particular focus for this lab will be all residential home sales in Ames between 2006 and 2010. This collection represents our population of interest. In this lab we would like to learn about these home sales by taking smaller samples from the full population. Let's load the data.

- We'll start with a simple random sample of size 60 from the population.

```
n <- 60
samp <- sample_n(ames, n)
```

Standard Deviations

- The **Standard Deviation** is a measure of how spread out numbers are.
- **68%** of values are within **1 standard deviation** of the mean
- **95%** of values are within **2 standard deviations** of the mean
- **99.7%** of values are within **3 standard deviations** of the mean

Standard Deviations

- The number of standard deviations from the mean is also called the "Standard Score", "sigma" or "z-score".
- Example:

$$\mu = 1.40$$

$$\sigma = 0.15$$

- How many standard deviations is 1.85m from the mean?

Standard Score

- To calculate the **standard score** ("z-score")
- First, subtract the mean from number
- Second, divide by the standard deviation
- Here is the formula for z-score that we have been using:

$$z = \frac{x - \mu}{\sigma}$$

Confidence intervals

- Based only on this single sample, the best estimate of the average living area of houses sold in Ames would be the sample mean, usually denoted as $\bar{x}$ (here we're calling it `x_bar`).
- That serves as a good point estimate but it would be useful to also communicate how uncertain we are of that estimate. This uncertainty can be quantified using a confidence interval.
- A confidence interval for a population mean is of the following form $[\bar{x} + z^\star \frac{s}{\sqrt{n}}]$

Confidence intervals

- Remember that confidence levels and percentiles are not equivalent. For example, a 95% confidence level refers to the middle 95% of the distribution, and the critical value associated with this area will correspond to the 97.5th percentile.

Confidence intervals

- We can find the critical value for a 95% confidence interal using

```
z_star_95 <- qnorm(0.975)
z_star_95
```

```
## [1] 1.959964
```

Confidence intervals

- Let's finally calculate the confidence interval:

```
samp %>%
  summarise(x_bar = mean(area),
            se = sd(area) / sqrt(n),
            me = z_star_95 * se,
            lower = x_bar - me,
            upper = x_bar + me)
```

```
## # A tibble: 1 x 5
##   x_bar    se    me lower upper
##   <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 1598.  75.4  148. 1450. 1746.
```

- To recap: even though we don't know what the full population looks like, we're 95% confident that the true average size of houses in Ames lies between the values lower and upper. There

95% Confidence Intervals

- A 95% confidence interval is a range of values that you can be 95% certain contains the true mean of the population. This is not the same as a range that contains 95% of the values. The graph below emphasizes this distinction.
- With large samples, you know that mean with much more precision than you do with a small sample, so the confidence interval is quite narrow when computed from a large sample.

95% Confidence Intervals

- It is correct to say that there is a 95% chance that the confidence interval you calculated contains the true population mean. It is not quite correct to say that there is a 95% chance that the population mean lies within the interval.
- What's the difference?

95% Confidence Intervals

- The population mean has one value. You don't know what it is (unless you are doing simulations) but it has one value. If you repeated the experiment, that value wouldn't change (and you still wouldn't know what it is). Therefore it isn't strictly correct to ask about the probability that the population mean lies within a certain range.
- In contrast, the confidence interval you compute depends on the data you happened to collect. If you repeated the experiment, your confidence interval would almost certainly be different. So it is OK to ask about the probability that the interval contains the population mean.

Confidence intervals

- Using R, we're going to collect many samples to learn more about how sample means and confidence intervals vary from one sample to another.
- Here is the rough outline:
- Obtain a random sample.
- Calculate the sample's mean and standard deviation, and use these to calculate and store the lower and upper bounds of the confidence intervals.

Confidence intervals

- Repeat these steps 50 times.

```
ci <- ames %>%
        rep_sample_n(size = n, reps = 50, replace = TRUE) %
        summarise(x_bar = mean(area),
                  se = sd(area) / sqrt(n),
                  me = z_star_95 * se,
                  lower = x_bar - me,
                  upper = x_bar + me)
```

Confidence intervals

- Let's view the first five intervals:

```
ci %>%
  slice(1:5)
```

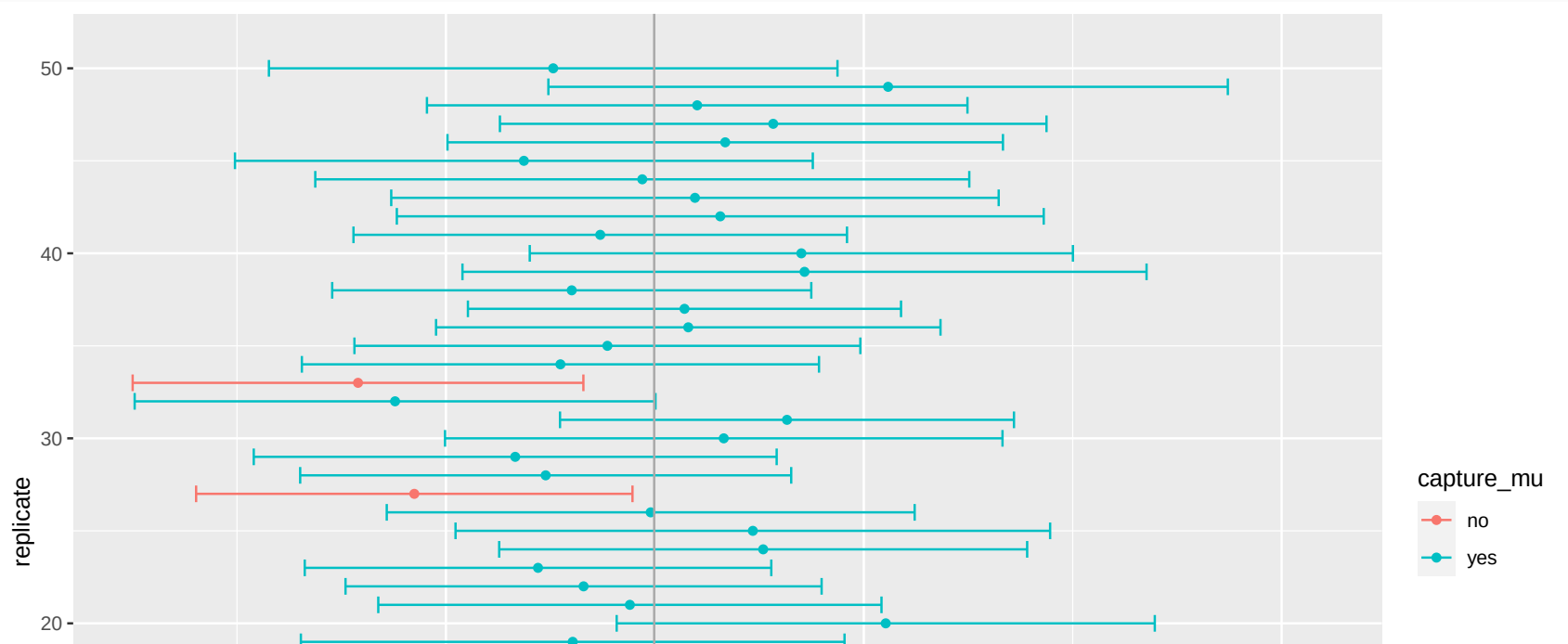
```
## # A tibble: 5 x 6
##   replicate x_bar    se    me lower upper
##       <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1         1 1561.  68.7  135. 1426. 1695.
## 2         2 1428.  61.6  121. 1307. 1549.
## 3         3 1605.  74.9  147. 1458. 1751.
## 4         4 1414.  59.9  117. 1296. 1531.
## 5         5 1534.  76.8  151. 1383. 1685.
```

Confidence Intervals - Plot

```
params <- ames %>%
  summarise(mu = mean(area))
```

Confidence intervals

```
ci <- ci %>%
  mutate(capture_mu = ifelse(lower < params$mu & upper > pa

qplot(data = ci, x = replicate, y = x_bar, color = capture_
  geom_errorbar(aes(ymin = lower, ymax = upper)) +
  geom_hline(data = params, aes(yintercept = mu), color = '
  coord_flip()
```



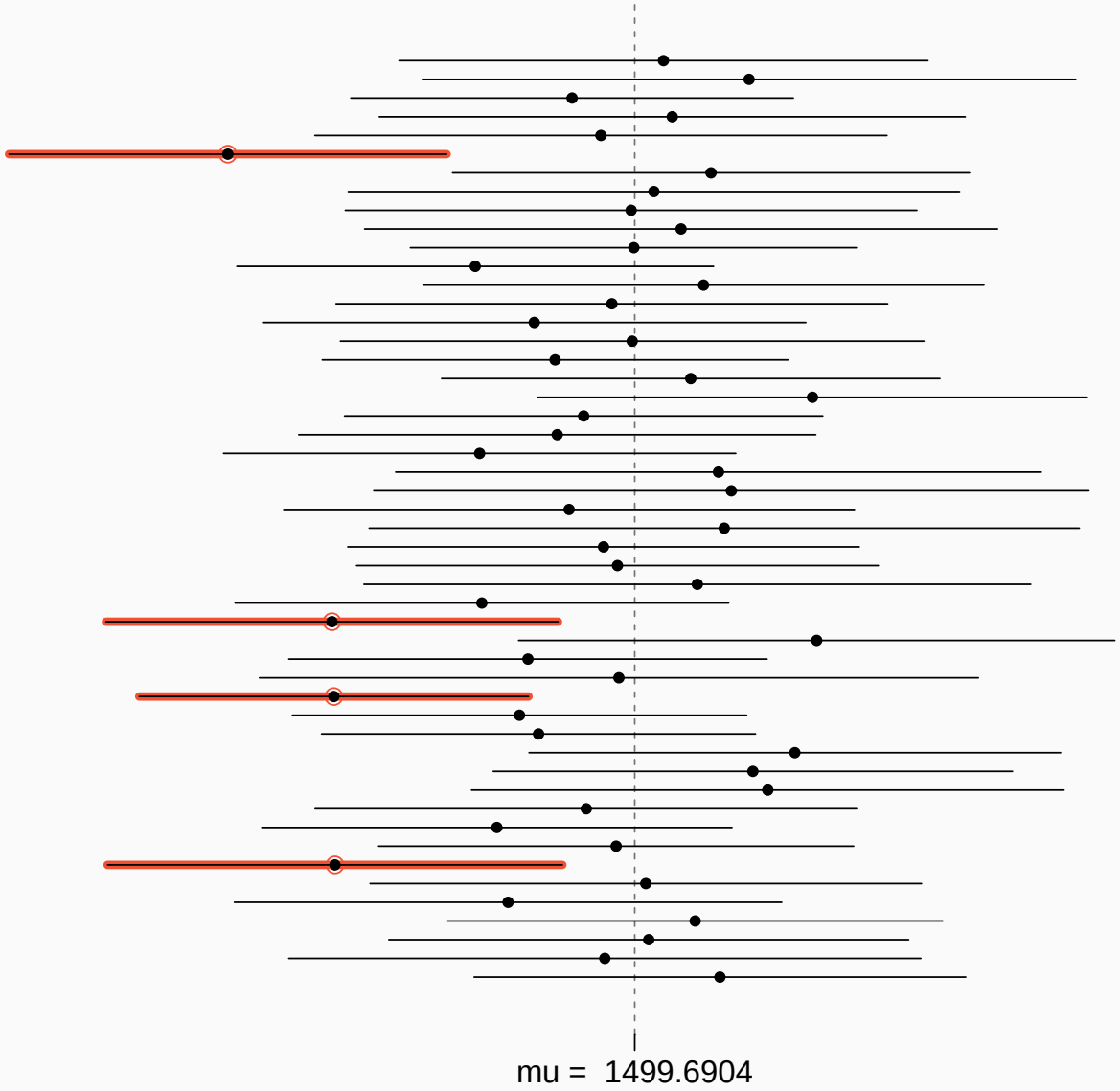
Confidence intervals - Plot

- We want a plot that indicates whether the interval does or does not capture the true population mean.

```
download.file("http://www.openintro.org/stat/data/ames.RDat
load("ames.RData")

population <- ames$Gr.Liv.Area
samp <- sample(population, 60)

samp_mean <- rep(NA, 50)
samp_sd <- rep(NA, 50)
n <- 60

for(i in 1:50){
  samp <- sample(population, n) # obtain a sample of size
  samp_mean[i] <- mean(samp)    # save sample mean in ith
```

Confidence Intervals - Plot

```
plot_ci(lower_vector, upper_vector, mean(population))
```



Inference for Numerical Data

- In 2004, the state of North Carolina released a large data set containing information on births recorded in this state. This data set is useful to researchers studying the relation between habits and practices of expectant mothers and the birth of their children. We will work with a random sample of observations from this data set.
- Load the nc data set into our workspace.

```
data(nc)
```

Explaratory Data Analysis

- What are the cases in this data set? How many cases are there in our sample?

```
glimpse(nc)
```

```
## Rows: 1,000
## Columns: 13
## $ fage          <int> NA, NA, 19, 21, NA, NA, 18, 17, N
## $ mage          <int> 13, 14, 15, 15, 15, 15, 15, 15, 1
## $ mature        <fct> younger mom, younger mom, younger
## $ weeks         <int> 39, 42, 37, 41, 39, 38, 37, 35, 3
## $ premie        <fct> full term, full term, full term,
## $ visits        <int> 10, 15, 11, 6, 9, 19, 12, 5, 9, 1
## $ marital       <fct> married, married, married, marrie
## $ gained        <int> 38, 20, 38, 34, 27, 22, 76, 15, N
## $ weight        <dbl> 7.63, 7.88, 6.63, 8.00, 6.38, 5.3
```

Explaratory Data Analysis

- Using visualization and summary statistics, describe the distribution of weight gained by mothers during pregnancy. The summary function can be useful.

```
summary(nc$gained)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.00   20.00   30.00   30.33   38.00   85.00      27
```

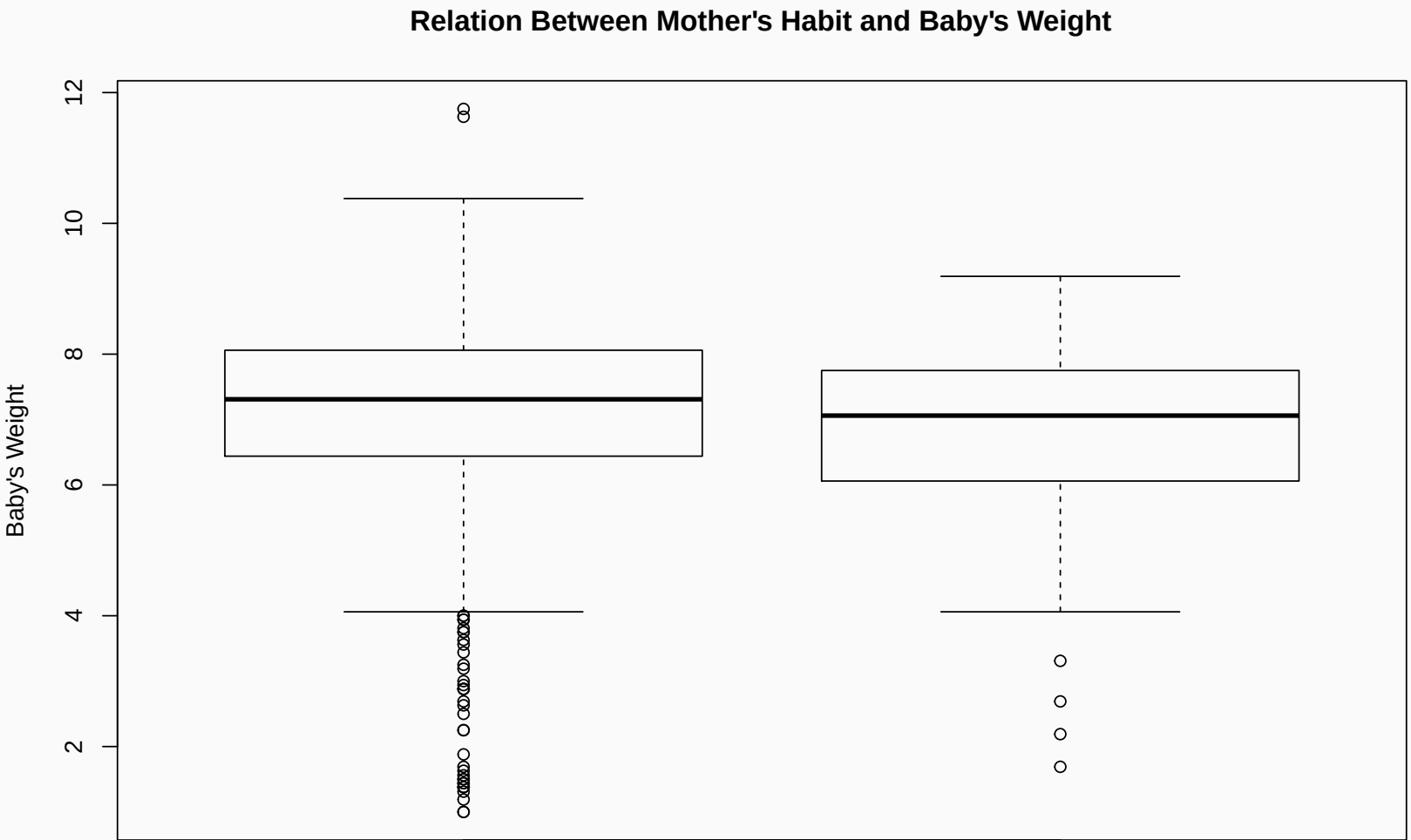
- We can also looking at the possible relationship between a mother's smoking habit and the weight of her baby

```
nc %>%
  group_by(habit) %>%
  summarise(mean_weight = mean(weight))
```

```
## Warning: Factor `habit` contains implicit NA, consider u
## `forcats::fct_explicit_na`
```

Explaratory Data Analysis - boxplot

```
# Basic box plot
boxplot(weight~habit,data=nc, main="Relation Between Mother
    ylab="Baby's Weight", xlab="Mother Smoker/Non-Smoker")
```

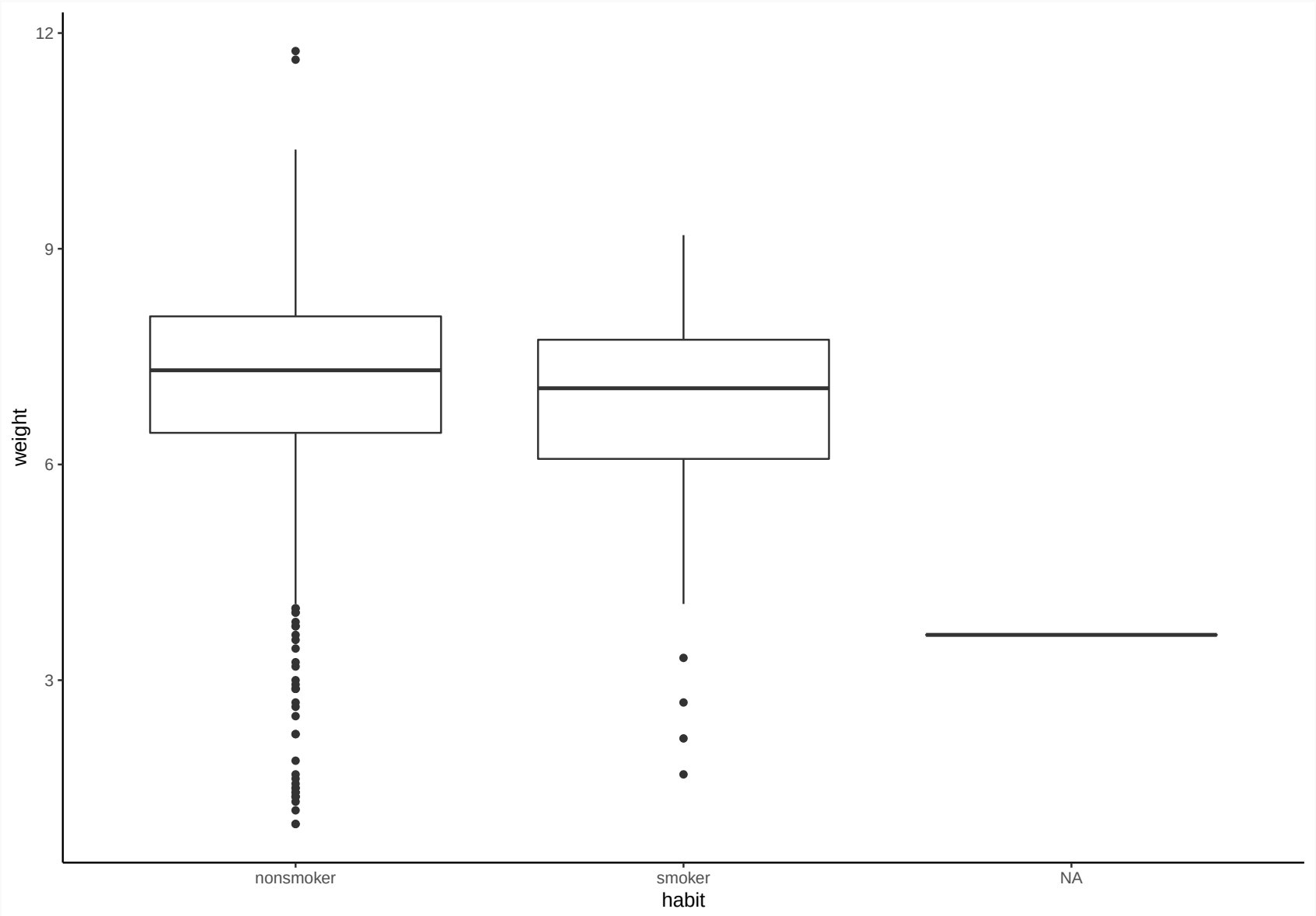


Explaratory Data Analysis - ggplot boxplot

```
library(ggplot2)
# gpplot box plot
p<-ggplot(nc, aes(x=habit, y=weight)) +
  geom_boxplot()
```

Explaratory Data Analysis - ggplot boxplot

```
p + scale_color_grey() + theme_classic()
```

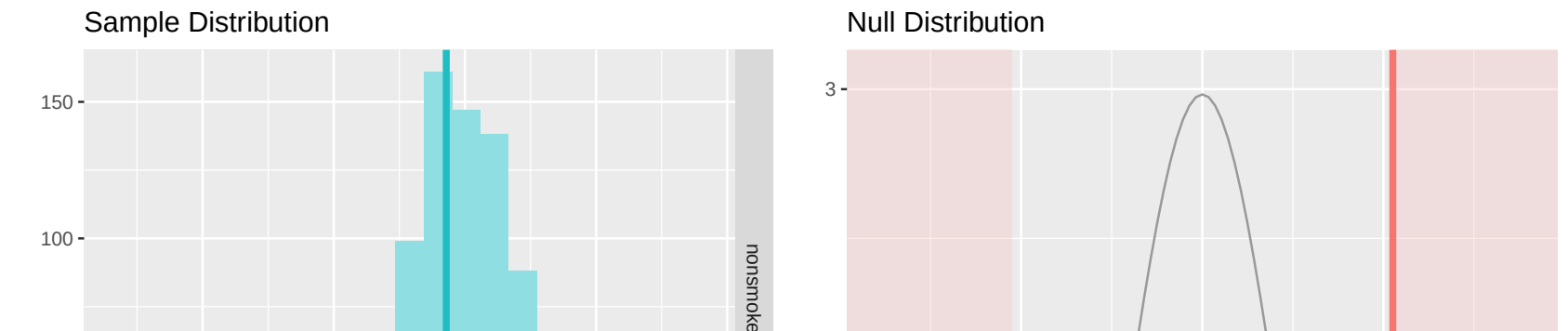


Explaratory Data Analysis - inference

- There is an observed difference, but is this difference statistically significant? In order to answer this question we will conduct a hypothesis test.
- Next, we introduce a new function, inference, that we will use for conducting hypothesis tests and constructing confidence intervals.

Explaratory Data Analysis - inference

```
inference(y = weight, x = habit, data = nc, statistic = "me
          alternative = "twosided", method = "theoretical")

## Response variable: numerical
## Explanatory variable: categorical (2 levels)
## n_nonsmoker = 873, y_bar_nonsmoker = 7.1443, s_nonsmoker
## n_smoker = 126, y_bar_smoker = 6.8287, s_smoker = 1.3862
## H0: mu_nonsmoker =  mu_smoker
## HA: mu_nonsmoker != mu_smoker
## t = 2.359, df = 125
## p_value = 0.0199
```



Explaratory Data Analysis - inference

- Let's pause for a moment to go through the arguments of this custom function.
    - The first argument is `y`, which is the response variable that we are interested in: `weight`.

Explaratory Data Analysis - inference

- The second argument is the explanatory variable, `x`, which is the variable that splits the data into two groups, smokers and non-smokers: `habit`.
- The third argument, `data`, is the data frame these variables are stored in.

Explaratory Data Analysis - inference

- Next is statistic, which is the sample statistic we're using, or similarly, the population parameter we're estimating.
- When performing a hypothesis test, we also need to supply the null value, which in this case is 0, since the null hypothesis sets the two population means equal to each other.
- The alternative hypothesis can be `"less"`, `"greater"`, or `"twosided"`. Lastly, the method of inference can be `"theoretical"` or `"simulation"` based.