IS5 in R: Displaying and Describing Data (Chapter 2)

Nicholas Horton (nhorton@amherst.edu)

2025-01-20

Table of contents

Introduction and background

This document is intended to help describe how to undertake analyses introduced as examples in the Fifth Edition of *Intro Stats* (2018) by De Veaux, Velleman, and Bock. This file as well as the associated Quarto reproducible analysis source file used to create it can be found at http://nhorton.people.amherst.edu/is5.

This work leverages initiatives undertaken by Project MOSAIC (http://www.mosaic-web.org), an NSF-funded effort to improve the teaching of statistics, calculus, science and computing in the undergraduate curriculum. In particular, we utilize the `mosaic` package, which was written to simplify the use of R for introductory statistics courses. A short summary of the R needed to teach introductory statistics can be found in the mosaic package vignettes (https://cran.r-project.org/web/packages/mosaic). A paper describing the mosaic approach was published in the *R Journal*: https://journal.r-project.org/archive/2017/RJ-2017-024.

We begin by loading packages that will be required for our analyses.

```
library(mosaic)
library(tidyverse)
```

Chapter 2: Displaying and Describing Data

Section 2.1: Summarizing and Displaying a Categorical Variable

```
options(digits = 3) # make decimal results easier to read
Titanic <- read_csv("http://nhorton.people.amherst.edu/is5/data/Titanic.csv")
```

By default, the `read_csv()` function (loaded with the `tidyverse` package) prints the variable names. These messages were suppressed using the `message: false` code chunk option to save space and improve readability.

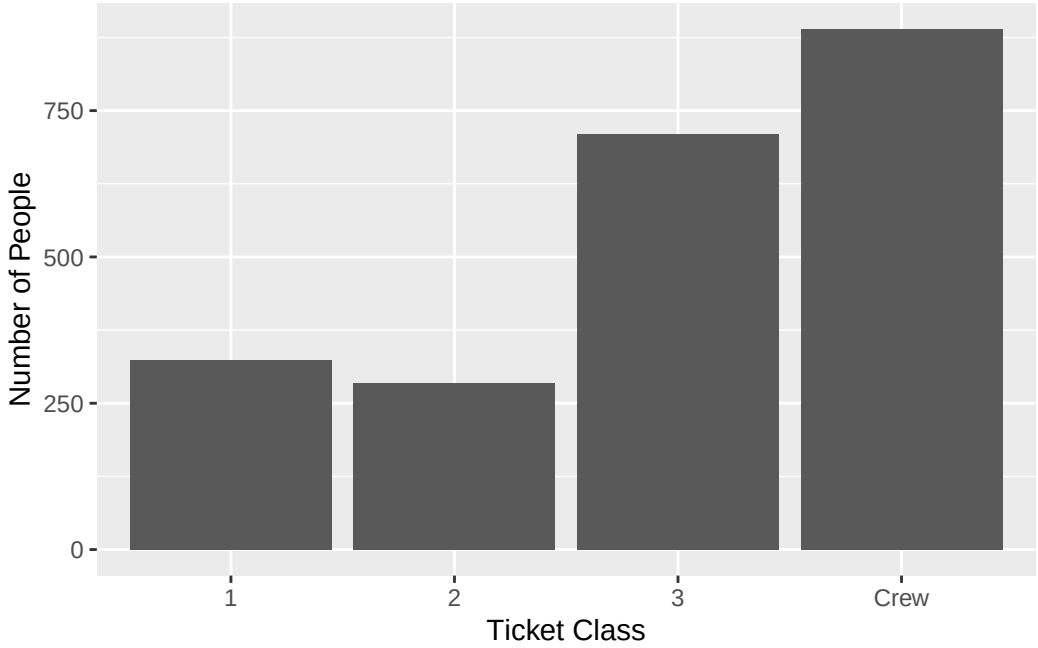
```
# Table 2.2, page 19
tally(~ Class, data = Titanic)
```

```
Class
   1    2    3 Crew
 324  285  710  889
```

```
# Table 2.3
tally(~ Class, format = "percent", data = Titanic)
```

```
Class
   1    2    3 Crew
14.7 12.9 32.2 40.3
```

```
# Figure 2.2, page 19
gf_bar(~ Class, data = Titanic) |>
  gf_labs(x = "Ticket Class", y = "Number of People")
```



`GOAL(~ X)` is the general form of the modeling language for one variable in the `mosaic` package. We use `gf_bar()` to make a bar graph using the `ggformula` system, which is automatically make accessible by loading the `mosaic` package.

See the Mosaic Minimal Guide for more details:

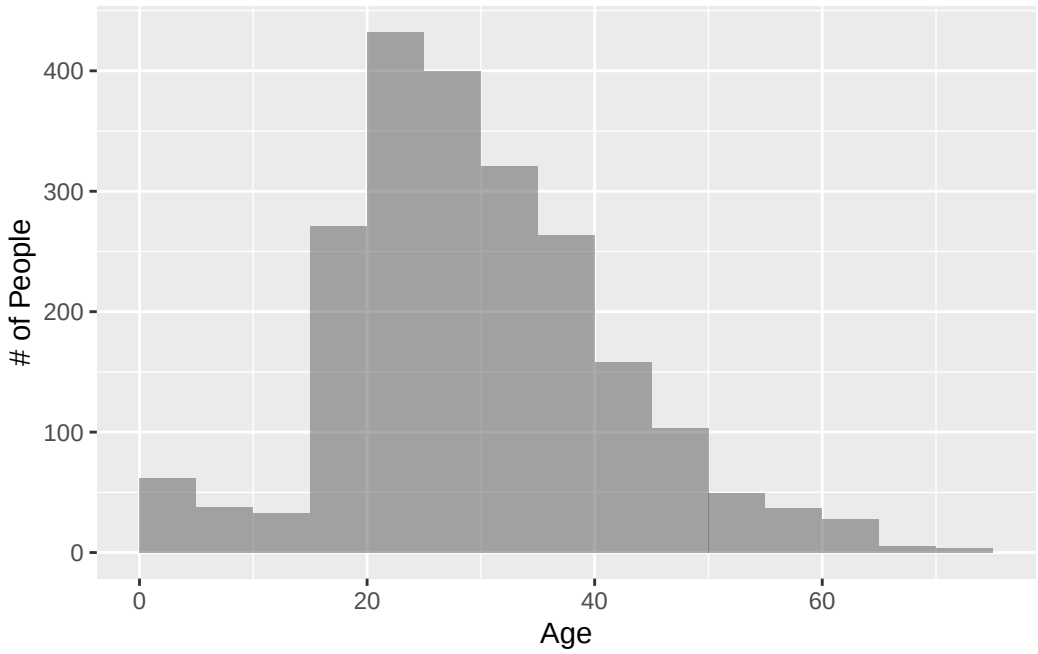
https://cran.r-project.org/web/packages/mosaic/vignettes/MinimalRgg.pdf

Section 2.2: Displaying a Quantitative Variable

Ages of Those Aboard the Titanic

```
# Figure 2.7, page 24
gf_histogram(~ Age, data = Titanic, binwidth = 5, ylab = "# of People", center = 5 / 2)
```

```
Warning: Removed 3 rows containing non-finite outside the scale range
(`stat_bin()`).
```

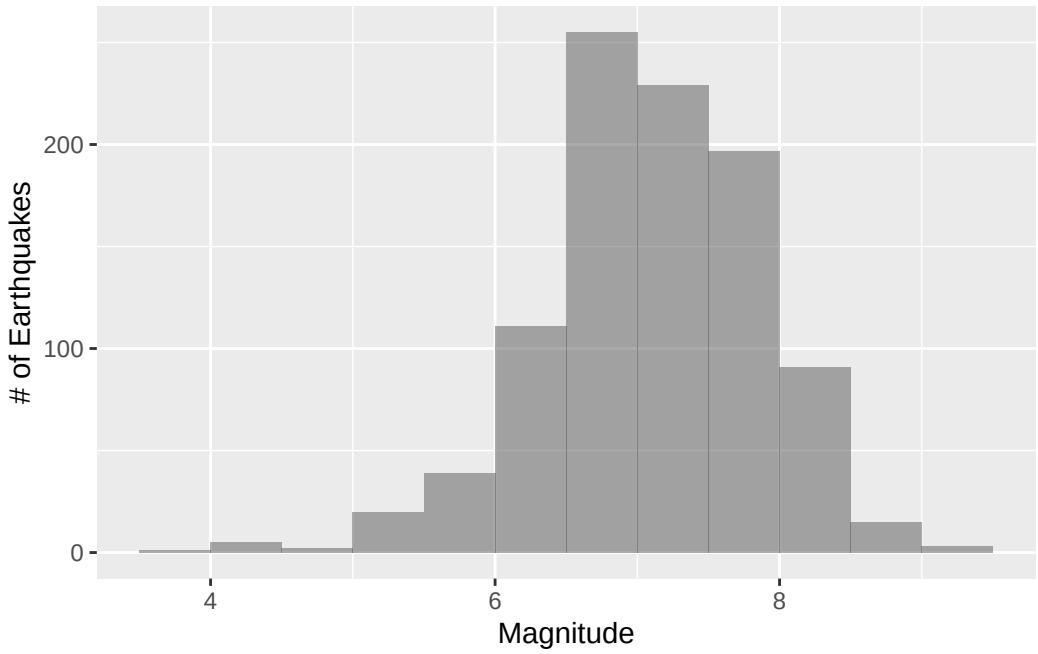


The function generates a warning because three of the ages are missing; this output can (and should!) be suppressed by adding `warning: false` as an option in this code chunk. Note: be sure to double check about a warning before suppressing it!

Earthquakes and Tsunamis

We begin by reading in the data.

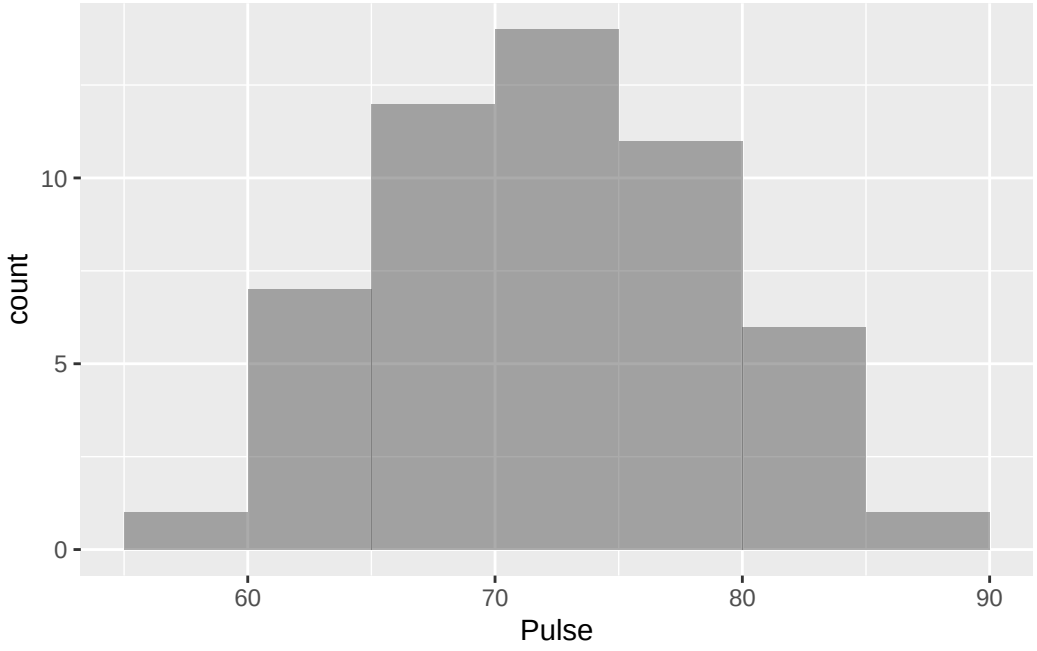
```
# Example 2.3, page 25
Earthquakes <- read_csv("http://nhorton.people.amherst.edu/is5/data/Tsunamis_2016.csv")
gf_histogram(~ Primary_Magnitude,
  data = Earthquakes, binwidth = 0.5,
  ylab = "# of Earthquakes", xlab = "Magnitude", center = 0.25
)
```



Stem-and-Leaf Displays

See page 26.

```
# Figure 2.8, page 26
Pulse_rates <- read_csv("http://nhorton.people.amherst.edu/is5/data/Pulse_rates.csv")
gf_histogram(~ Pulse, data = Pulse_rates, binwidth = 5, center = 5 / 2)
```




```
with(Pulse_rates, stem(Pulse)) # base R function that doesn't have a `data =` option.
```

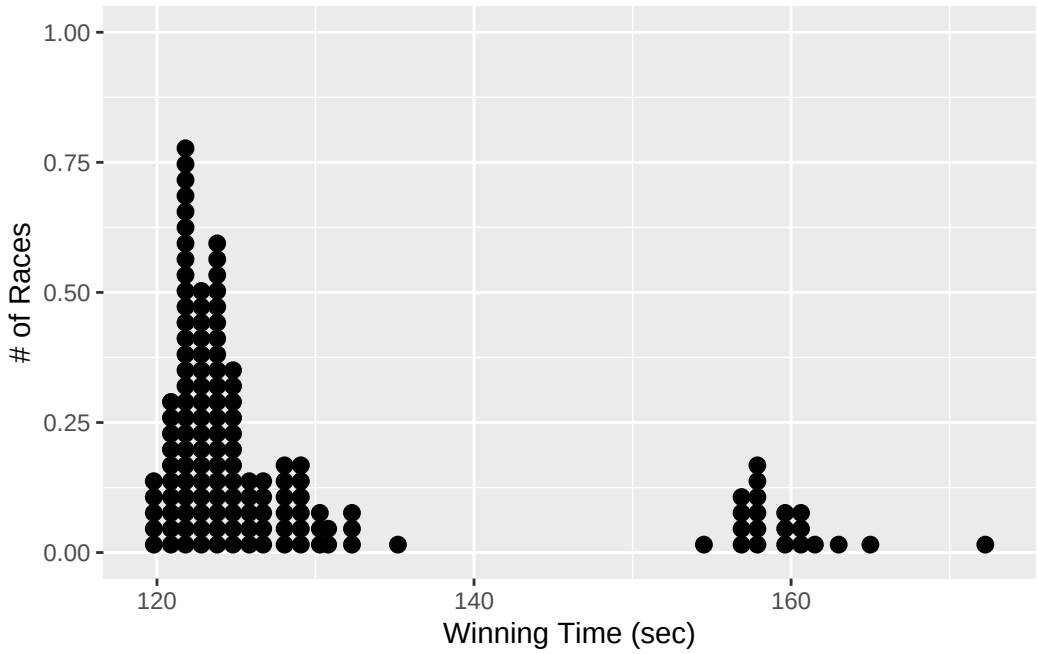
```
  The decimal point is 1 digit(s) to the right of the |

  5 | 7
  6 | 13444
  6 | 556668888899
  7 | 0012223333444
  7 | 5557777888889
  8 | 0112233
  8 | 6
```

Dotplot

See Figure 2.9, page 27

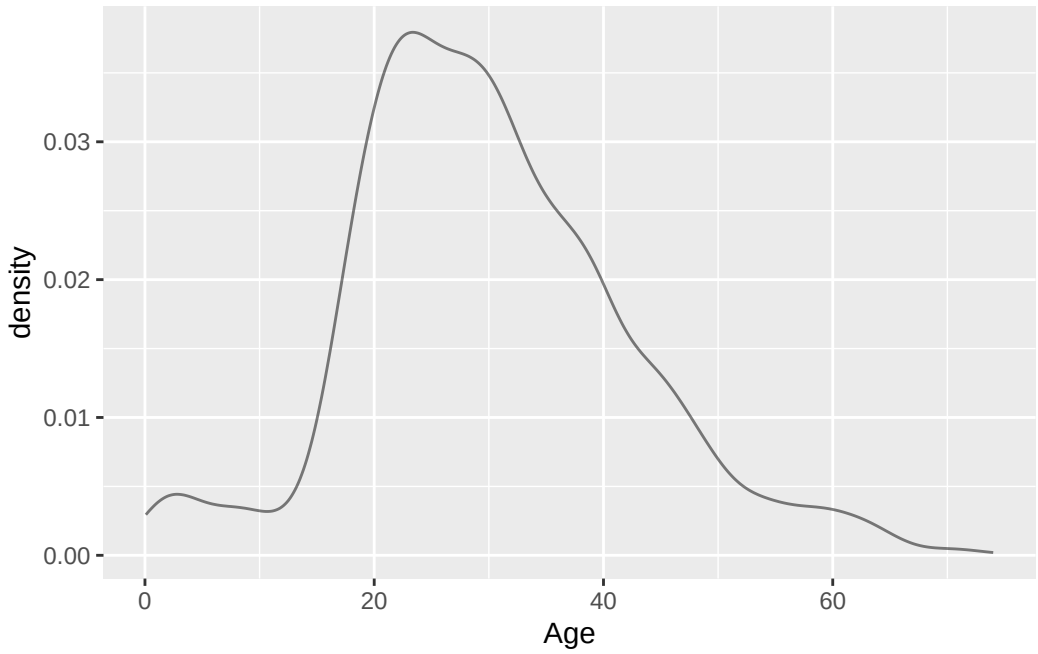
```
Derby <- readr::read_csv("http://nhorton.people.amherst.edu/is5/data/Kentucky_Derby_2016.csv
gf_dotplot(~Time_Sec, data = Derby, binwidth = 1) |>
  gf_labs(x = "Winning Time (sec)", y = "# of Races")
```



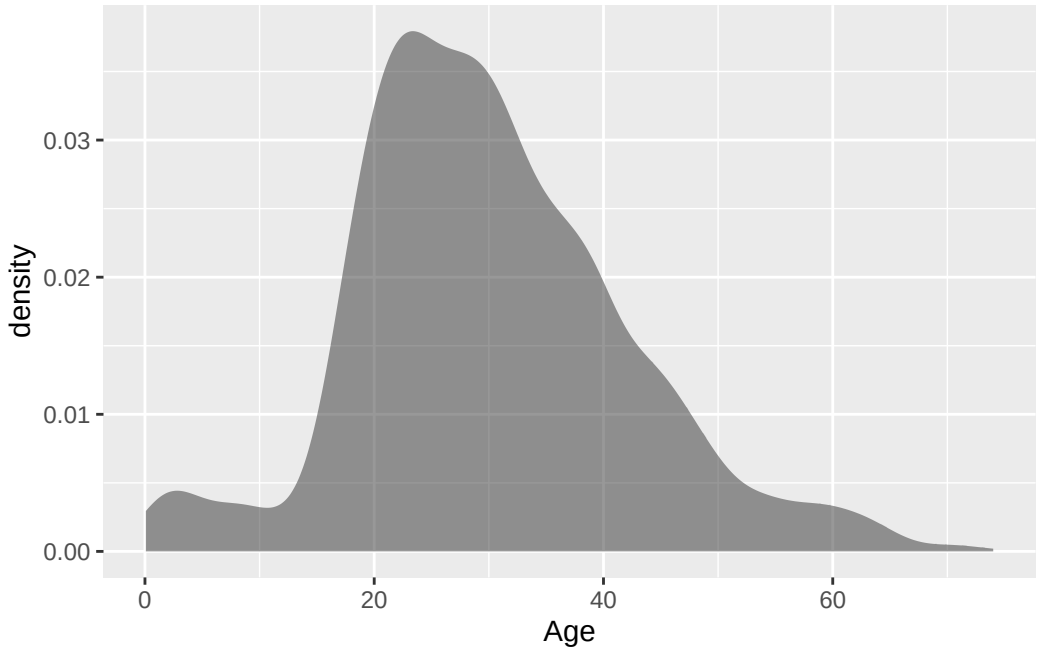
Density Plots

There are two forms of density plots: not-shaded, and shaded. The former will be useful when comparing multiple densities.

```
# Figure 2.10, page 27
gf_dens(~ Age, data = Titanic)
```




```
gf_density(~ Age, data = Titanic)
```



Section 2.3: Shape

See displays on pages 28-29.

Consumer Price Index

First we need to load the data.

```
CPI <- read_csv("http://nhorton.people.amherst.edu/is5/data/CPI_Worldwide.csv") |>
  janitor::clean_names()
```

The pipe operator (`|>`) takes the output of the first command and uses it as input for the next.

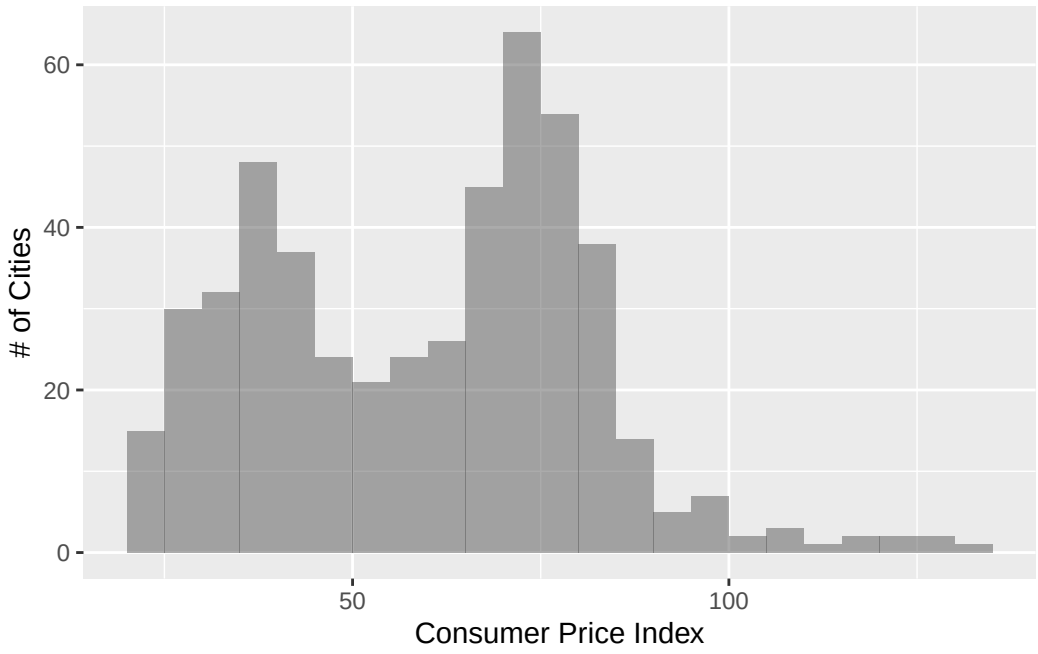
The variable names coming out of spreadsheet files can sometimes be quite wonky. The results from `read_csv()` are passed to the `clean_names()` function from the `janitor` package to make them more consistent.

```
names(CPI)
```

```
[1] "city"                          "consumer_price_index"
[3] "rent_index"                    "consumer_price_plus_rent_index"
[5] "groceries_index"               "restaurant_price_index"
[7] "local_purchasing_power_index"
```

You can use the `names()` function to check the reformatted names.
(We'll often use the `glimpse()` command to provide even more information about a dataset.)

```
# Example 2.5, page 30
gf_histogram(~consumer_price_index,
  data = CPI, ylab = "# of Cities",
  xlab = "Consumer Price Index", binwidth = 5, center = 5 / 2
)
```



Credit Card Expenditures

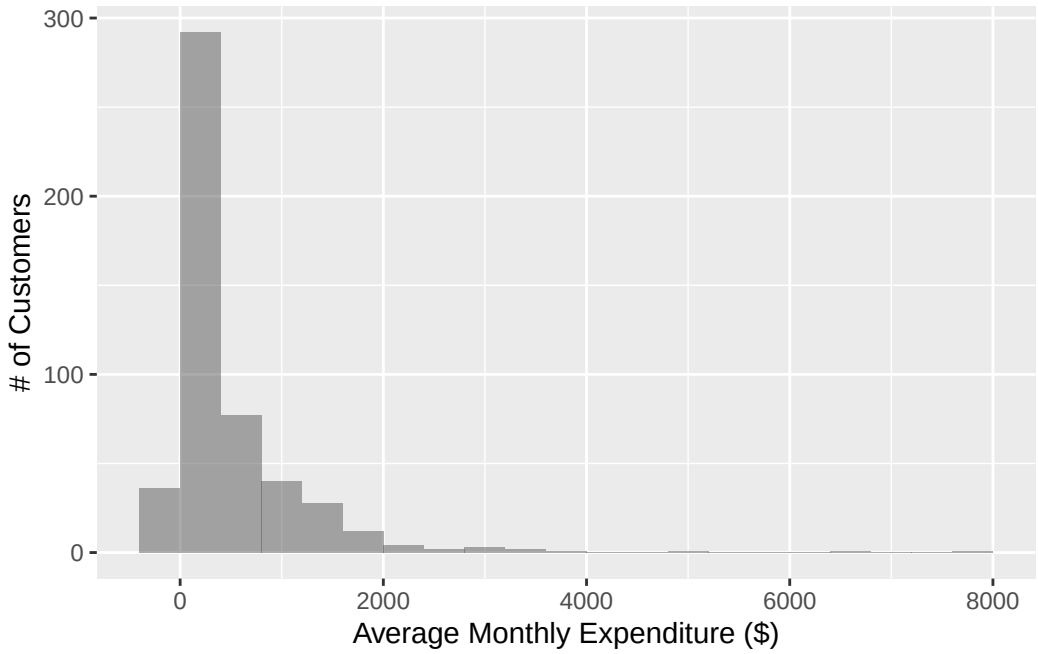
First we load the data.

```
#! message: false
CreditCardEx <- read_csv("http://nhorton.people.amherst.edu/is5/data/Credit_card_charges.csv
  janitor::clean_names()
```

```
Rows: 500 Columns: 1
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (1): Charges($)

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Figure 2.6, page 30
gf_histogram(~charges,
  data = CreditCardEx, ylab = "# of Customers",
  xlab = "Average Monthly Expenditure ($)", binwidth = 400, center = 200
)
```

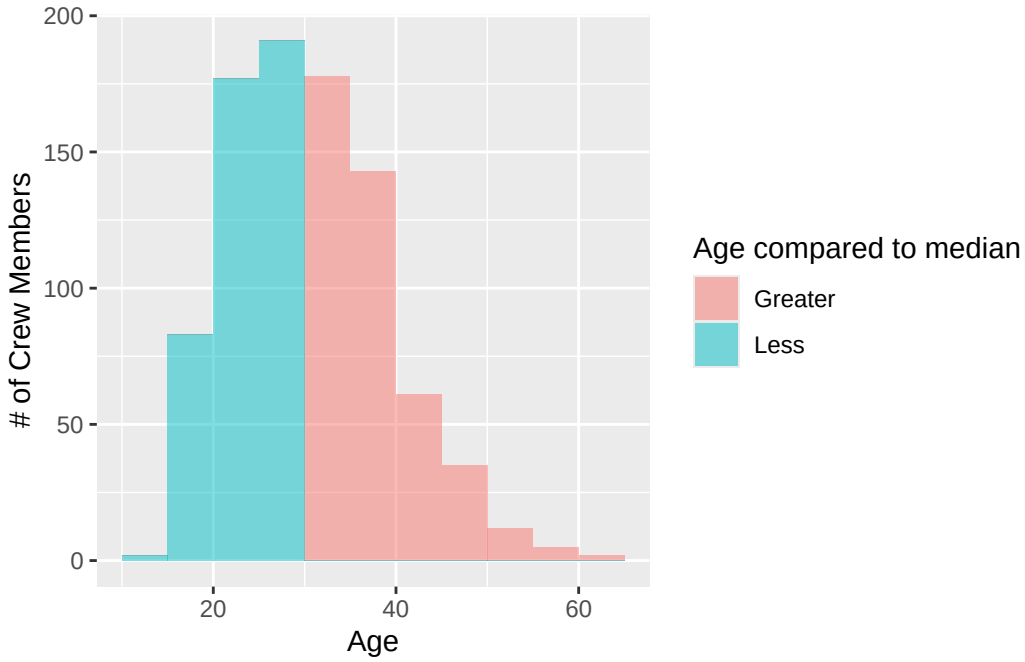


Section 2.4: Center

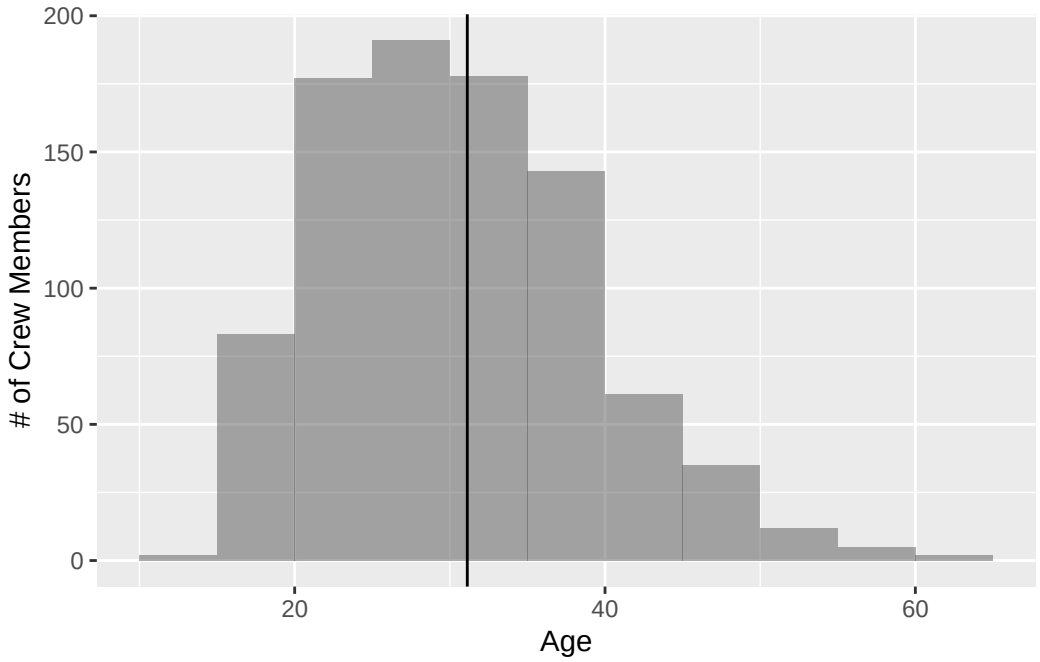
Finding Median and Mean

First we need to load the data.

```
TitanicCrew <- filter(Titanic, Class == "Crew")
# Figure 2.15, page 32
TitanicCrew |>
  mutate(color = ifelse(Age <= median(Age), "Less", "Greater")) |>
  gf_histogram(~ Age, fill = ~color, binwidth = 5, center = 5 / 2, ylab = "# of Crew Members"
  gf_labs(fill = "Age compared to median")
```




```
# Figure 2.16
gf_histogram(
  ~Age,
  data = TitanicCrew,
  ylab = "# of Crew Members",
  binwidth = 5,
  center = 5 / 2) |>
  gf_vline(xintercept = mean(~Age, data = TitanicCrew))
```




```
df_stats(~Age, data = TitanicCrew)
```

```
  response min Q1 median Q3 max mean   sd   n missing
1      Age  14 24     30 37  62 31.1 8.55 889       0
```

Another way to generate summary statistics is the `favstats()` command (we will stick to `df_stats()` because it is more flexible).

```
favstats(~Age, data = TitanicCrew)
```

```
 min Q1 median Q3 max mean   sd   n missing
  14 24     30 37  62 31.1 8.55 889       0
```

Section 2.5: Spread

The Range

```
range(~Age, data = TitanicCrew)
```

```
[1] 14 62
```

```
diff(range(~Age, data = TitanicCrew))
```

```
[1] 48
```

The `range()` function returns the maximum and minimum values, so we can use the `diff()` function to find the difference between the two values.

The Interquartile Range

```
df_stats(~ Age, data = TitanicCrew)
```

```
  response min Q1 median Q3 max mean   sd   n missing
1      Age  14 24     30 37  62 31.1 8.55 889       0
```

```
IQR(~ Age, data = TitanicCrew)
```

```
[1] 13
```

Using the `IQR()` function allows us to avoid having to manually find the IQR by subtracting Q1 from Q3 from the `df_stats()` output.

Standard Deviation

```
sd(~Age, data = TitanicCrew)
```

```
[1] 8.55
```

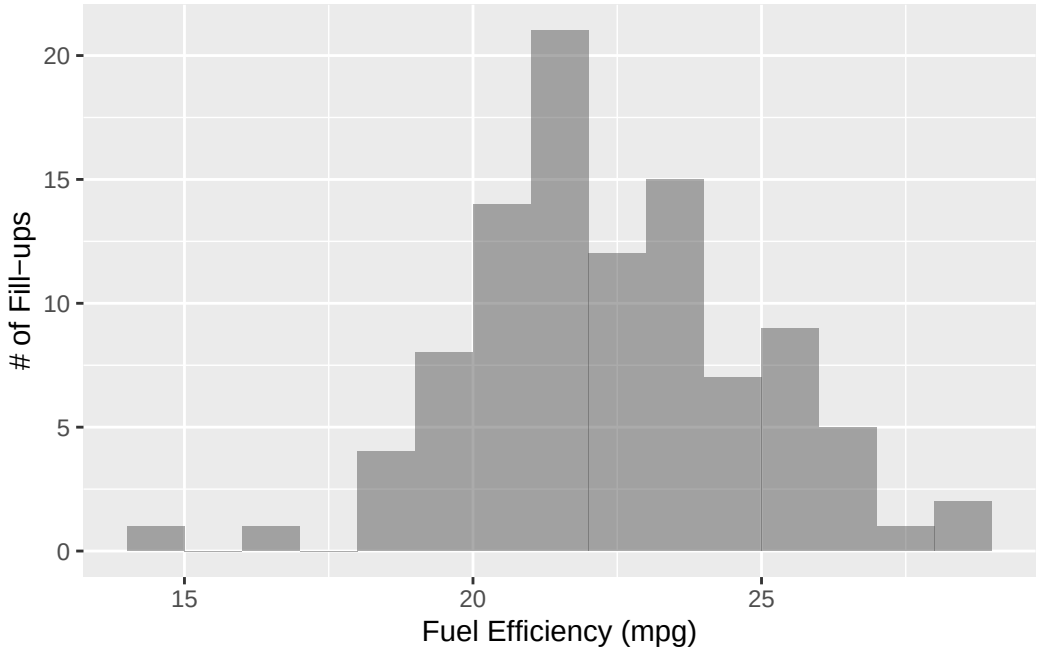
```
var(~Age, data = TitanicCrew)
```

```
[1] 73.1
```

Summarizing a Distribution

First we need to load the data.

```
Nissan <- read_csv("http://nhorton.people.amherst.edu/is5/data/Nissan.csv")
# Step-by-Step Example, page 39
gf_histogram(~mpg,
  data = Nissan, binwidth = 1, xlab = "Fuel Efficiency (mpg)",
  ylab = "# of Fill-ups", center = 5 / 2
)
```



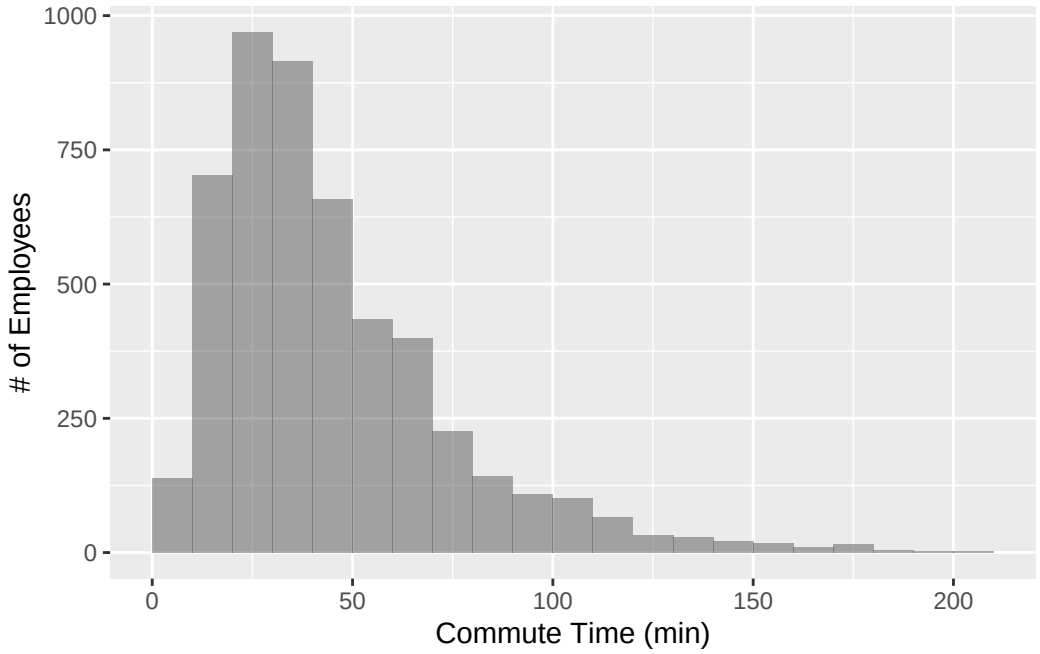

```
df_stats(~ mpg, data = Nissan)
```

```
  response  min   Q1 median Q3  max mean   sd   n missing
1      mpg 14.7 20.8   22.1 24 28.2 22.4 2.45 100       0
```

Random Matters

First we need to load the data.

```
Commute <- read_csv("http://nhorton.people.amherst.edu/is5/data/Population_Commute_Times.csv
  janitor::clean_names()
# Figure 2.19, page 40
gf_histogram(~ commute_time,
  data = Commute, binwidth = 10, xlab = "Commute Time (min)",
  ylab = "# of Employees", center = 5
)
```


1000
750
500
250
0
# of Employees
0
50
100
150
200
Commute Time (min)