```
### Chapter 3 Example code to focus on:

library(mdsr)
library(tidyverse)

# Simple scatterplot using geom_point (Fig. 3.1):
g <- ggplot(data = CIACountries, aes(y = gdp, x = educ))
g + geom_point(size = 3)

# Addition of information on a third variable (categorical) using the Color aesthetic
(Fig. 3.2):
g + geom_point(aes(color = net_users), size = 3)

# Replacing the plotting characters with country names (Fig. 3.3):
g + geom_text(aes(label = country, color = net_users), size = 3)

# Adding information on a fourth variable using the size of the bubbles (Fig. 3.4):
g + geom_point(aes(color = net_users, size = roadways))

# Using facets rather than colors to separate the different levels of net_users (Fig.
3.7):
g +
  geom_point(alpha = 0.9, aes(size = roadways)) +
  coord_trans(y = "log10") +
  facet_wrap(~net_users, nrow = 1) +
  theme(legend.position = "top")


## Creates basic ggplot object that we can then build on:
g <- ggplot(data = SAT_2010, aes(x = math))


## Histogram of math SAT scores by state (Fig. 3.10)
g + geom_histogram(binwidth = 10) + labs(x = "Average Math SAT score")

# Could also specify the number of bins directly:
g + geom_histogram(bins = 8) + labs(x = "Average Math SAT score")


# Change the "adjust" argument to get a smoother or more wiggly density estimate:
# lower bandwidth = more wiggly estimate
g + geom_density(adjust = 0.1)
# higher bandwidth = smoother estimate
g + geom_density(adjust = 0.9)


## bar plot of average math SAT scores for a selection of states (Fig. 3.12)
ggplot(
  data = head(SAT_2010, 10),
  aes(x = reorder(state, math), y = math)
) +
  geom_col() +
  labs(x = "State", y = "Average Math SAT score")

library(mosaicData)
```

```
## Stacked VERTICAL Bar Plot with Color (like Fig. 3.13 but without flipping the
coordinates)
ggplot(data = mosaicData::HELPrct, aes(x = homeless)) +
  geom_bar(aes(fill = substance), position = "fill") +
  scale_fill_brewer(palette = "Spectral")

## Note the difference between position="stack" (shows counts) and position="fill"
(shows proportions):
ggplot(data = mosaicData::HELPrct, aes(x = homeless)) +
  geom_bar(aes(fill = substance), position = "stack") +
  scale_fill_brewer(palette = "Spectral")


## Creating a basic ggplot object
g <- ggplot(
  data = SAT_2010,
  aes(x = expenditure, y = math)
) +
  geom_point()
#plotting it:
g

## Plotting it as a scatterplot with a trend line:
g <- g +
  geom_smooth(method = "lm", se = FALSE) +     # Note "lm" will show a LINEAR trend
  xlab("Average expenditure per student ($1000)") +
  ylab("Average score on math SAT")
g

## Plotting the symbolic scatterplot with SAT categories separated by color (Fig.
3.14):
g + aes(color = SAT_rate)

## Plotting 3 separate scatterplots for the SAT categories (Fig. 3.15):
g + facet_wrap(~ SAT_rate)

## time series plot showing the change in temperature at the MacLeish field station in
2015 (Fig. 3.17)
# install.packages("macleish")
library(macleish)
ggplot(data = whately_2015, aes(x = when, y = temperature)) +
  geom_line(color = "darkgray") +
  geom_smooth() +
  xlab(NULL) +
  ylab("Temperature (degrees Celsius)")
```

```
######### Chapter 4 code to focus on:

## Another way to do the same thing, but with the pipe operation:
presidential %>%
  filter(lubridate::year(start) > 1973 & party == "Democratic") %>%
  select(name)

## The 'interval' function can calculate the duration of time between two date values:
library(lubridate)
my_presidents <- presidential %>%
  mutate(term.length = interval(start, end) / dyears(1))
my_presidents


## The 'year' function will pick out the year of a date value:
my_presidents <- my_presidents %>%
  mutate(elected = year(start) - 1)
my_presidents



## Putting missing values for presidents who were not actually elected
## the syntax for 'ifelse' is:  ifelse(test_condition, result_if_TRUE,
result_if_FALSE)
my_presidents <- my_presidents %>%
  mutate(elected = ifelse(elected %in% c(1962, 1973), NA, elected))
my_presidents


## Stylistic choice to prefer underscores to periods in variable names, function
names, etc.
my_presidents <- my_presidents %>%
  rename(term_length = term.length)
my_presidents


## ordering the rows based on one column's values
my_presidents %>%
  arrange(desc(term_length))

## A nested sorting.  Can you explain the sorted result?
my_presidents %>%
  arrange(desc(term_length), party, elected)


## Summary statistics for the whole data set
my_presidents %>%
  summarize(
    N = n(),
    first_year = min(year(start)),
    last_year = max(year(end)),
    num_dems = sum(party == "Democratic"),
    years = sum(term_length),
    avg_term_length = mean(term_length)
  )
```

```
## Summary statistics, separate by party:

my_presidents %>%
  group_by(party) %>%
  summarize(
    N = n(),
    first_year = min(year(start)),
    last_year = max(year(end)),
    num_dems = sum(party == "Democratic"),
    years = sum(term_length),
    avg_term_length = mean(term_length)
  )


## install and load package:
# install.packages("Lahman")
library(Lahman)

## Similar, but selecting more columns for this data frame 'mets_ben'
mets_ben <- Teams %>%
  select(yearID, teamID, W, L, R, RA) %>%
  filter(teamID == "NYN" & yearID %in% 2004:2012)
mets_ben


## Changing R ("runs") column name to RS ("runs scored")
mets_ben <- mets_ben %>%
  rename(RS = R)    # new name = old name
mets_ben


## Creating a new "winning percentage" column:
mets_ben <- mets_ben %>%
  mutate(WPct = W / (W + L))
mets_ben


## Picking which years the Mets won fewer games than "expected"
filter(mets_ben, W < W_hat)

## Sorting from "luckiest" years to "unluckiest" years:
mets_ben %>%
  mutate(Diff = W - W_hat) %>%
  arrange(desc(Diff))


## Summary statistics for a single variable:
mets_ben %>%
  skim(W)

## Summary statistics for all variables:
mets_ben %>%
  skim()
```

```
## Summary statistics for several variables:
mets_ben %>%
  summarize(
    num_years = n(),
    total_W = sum(W),
    total_L = sum(L),
    total_WPct = sum(W) / sum(W + L),
    sum_resid = sum(W - W_hat)
  )
# If an error, change 'summarize' to 'summarise'


## Summary statistics, separated by value of general manager:
mets_ben %>%
  group_by(gm) %>%
  summarize(
    num_years = n(),
    total_W = sum(W),
    total_L = sum(L),
    total_WPct = sum(W) / sum(W + L),
    sum_resid = sum(W - W_hat)
  ) %>%
  arrange(desc(sum_resid))
```

```
### Chapter 5 code to focus on (mainly focusing on the simpler examples here):

## loading packages
library(tidyverse)
library(mdsr)

# Creating a simple data frame with students in a Math class:
name <- c("Jenny", "James", "Ming", "Alisha", "Tara", "Niels")
test <- c(78, 81, 74, 82, 83, 91)
quiz1 <- c(9,10,8,9.5,8.5,8)
math <- data.frame(name,test,quiz1)
print(math)

# Creating a simple data frame with students in a Reading class:
student <- c("Kyle", "Jenny", "Alisha", "Bob", "Laura")
exercise <- c(3,4.5,5,4,5)
test <- c(72, 91, 90, 84, 88)
reading <- data.frame(student, exercise, test)
print(reading)

# A basic inner join, and seeing the result:
both_inner <- math %>%
 inner_join(reading, by = c("name" = "student"))
head(both_inner,10)
nrow(both_inner)

# A basic left join, and seeing the result:
both_left <- math %>%
 left_join(reading, by = c("name" = "student"))
head(both_left,10)
nrow(both_left)

# A basic right join, and seeing the result:
both_right <- math %>%
 right_join(reading, by = c("name" = "student"))
head(both_right,10)
nrow(both_right)

# A basic full join, and seeing the result:
both_full <- math %>%
 full_join(reading, by = c("name" = "student"))
head(both_full,10)
nrow(both_full)

# It can be convenient to rename some of the columns in the joined data set
# (with the pipe operation, this really could be done while doing the join):
both_inner <- both_inner %>%
 rename(math_test = test.x, reading_test=test.y)
head(both_inner,10)


## Inner join of the 'flights' data frame with the 'airlines' data frame.
## note the key (ID) column is called "carrier" in both tables.
flights_joined <- flights %>%
  inner_join(airlines, by = c("carrier" = "carrier"))
```

```
# Because of the identical name of the key column, could simply use:

flights_joined <- flights %>%
  inner_join(airlines, by = join_by(carrier))
```