### 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))**