Research and Working with Data in R

Mark M. Fredrickson (mfredric@umich.edu)

Computational Methods in Statistics and Data Science (Stats 406)

Working with real data

Data Formats

Generally, we will find two types of data file formats in the wild:

- Proprietary: .xls or .xlsx (Excel), .dta (Stata), .sqlite (SQLite)
- Delimited: tab, comma, other

Proprietary

Support for these varies. For example, support for .xlsx files has been hit or miss in the past.

Best option: load in other software, convert to a delimited format.

The `foreign` library can read some formats (SPSS, Stata, SAS).

Delimited

A **delimited** data file is a **text file** where the columns are separated by special characters:

```
1, "some text", 8, 2.30
2, "other text" ,, 9.88
3, "text, with comma", 0, -5.23
```

Common delimiters include commas and tabs:

```
read.csv
read.table
```

Also the more generic `read.delim`.

Loading Data in R

Data will often come in compressed formats. For example, .zip or .gz. Your first step is to decompress it.

Then load it into a variable using a `read` function.

You may need to convert some columns to different data types (using `as.SOMETHING`)

Saving Data in R

R has its own proprietary format, .rda.

You can save not only single tables, but collections of information, functions, variables.

```
> k <- 100000
> myrands <- rnorm(k)
> save(file = "example.rda", k, myrands)
```

Loading saved files

```
> rm(k) ; rm(myrands)
> load("example.rda")
> print(k)

[1] 1e+05
```

Basic Data Cleaning

Real data often has missing values, bad values.

It is useful to investigate to see if there missing values or things that can't happen.

```
> summary(somedata)
```

```
       x1                x2
 Min.   :-3.533   Length:100
 1st Qu.:-0.682   Class :character
 Median :-0.031   Mode  :character
 Mean   :-0.014
 3rd Qu.: 0.656
 Max.   : 2.002
 NA's   :5
```

We could delete things with missing, but it might be better to **impute** a value

```
> somedata$x1[is.na(somedata$x1)] <- mean(somedata$x1, na.rm = TRUE)
> summary(somedata)
```

```
       x1                x2
 Min.   :-3.533   Length:100
 1st Qu.:-0.670   Class :character
 Median :-0.014   Mode  :character
 Mean   :-0.014
 3rd Qu.: 0.635
 Max.   : 2.002
```

Example: Loading data from the National Health And Nutrition Examination Survey

The **National Health And Nutrition Examination Survey (NHANES)** provides survey data on the dietary and health habits of people in the United States.

For many years, **low dose aspirin** was thought to be beneficial for those at risk of **heart disease**.

I downloaded data on **aspirin use** and **blood pressure exams** for survey participants.

Opening data

There are two data files, both in XPT format (a SAS format). rseek.org suggested the haven package:

```
> library(haven)
> aspirin <- read_xpt("./RXQASA_H.XPT")
> dim(aspirin)
```

```
[1] 3815    8
```

```
> bp <- read_xpt("./BPX_H.XPT")
> dim(bp)
```

```
[1] 9813   23
```

Coding “taking aspirin” variable

```
> ### Questions:
> # RXQ515 - Followed (doctor's) advice, took low-dose aspirin?
> # RXQ520 - Taking low-dose aspirin on your own?
>
> eq1 <- function(x) {
+     tmp <- x == 1
+     tmp[is.na(tmp)] <- FALSE
+     return(tmp)
+ }
> aspirin$taking_aspirin <- eq1(aspirin$RXQ515) | eq1(aspirin$RXQ520)
```

Aggregating multiple blood pressure readings

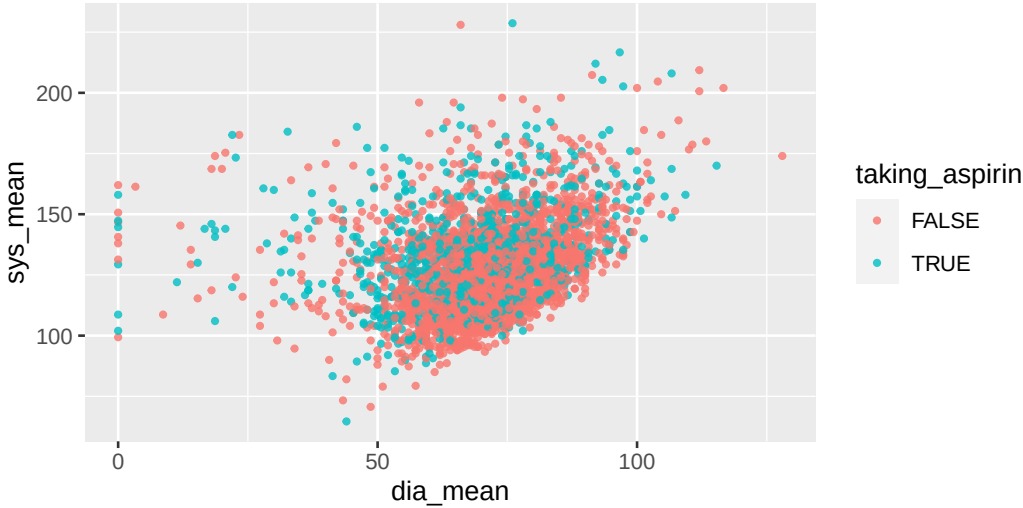
```
> bp$sys_mean <- rowMeans(bp[, c("BPXSY1", "BPXSY2", "BPXSY3", "BPXSY4")],
> bp$dia_mean <- rowMeans(bp[, c("BPXDI1", "BPXDI2", "BPXDI3", "BPXDI4")],
```

Combining

The `dplyr` (part of `tidyverse`) has several methods for **joining tables**:

- `inner_join` matches the tables on a **common key**, discard any entries that do not have a match (multiple matches possible)
- `left_join` keeps everything in the first table, even if no match (`NA` values for unmatched)
- `full_join` keeps everything in both tables, matching were it can (again, `NA` for unmatched)

```
> combined <- inner_join(aspirin, bp, "SEQN") # common "sequence number" I
```


sys_mean
200
150
100
0
50
100
dia_mean
taking_aspirin
FALSE
TRUE

Final Thoughts

- Document, document, document: use scripts/RMarkdown documents to record your changes to data
- Investigate outliers, missing values, strange patterns. Does -9 make sense for count data?
- Use `rseek.org` with your file extension to find packages.
- Open in other software and convert to CSV files.