Statistics with R

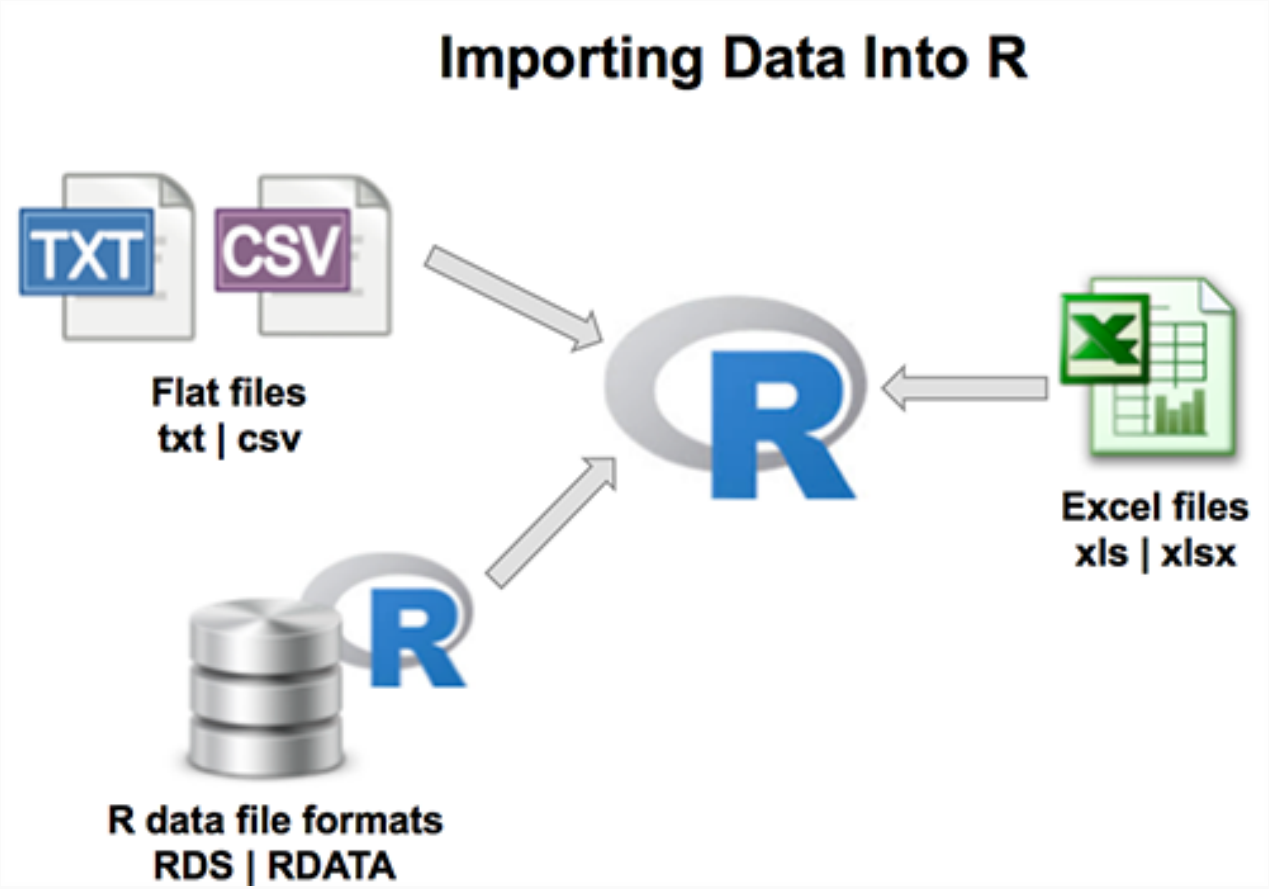
Data Visualization

Zhuanghua Shi (Strongway)

14 May, 2019

Last week: Import/Export Data

R can import and export many types of data. Most often format used are text, csv and excel files.



Import txt/csv files

- Read tabular data into R

```
read.table(file, header = FALSE, sep = "", dec = ".")
```

- Read "comma separated value" files (".csv")

```
read.csv(file, header = TRUE, sep = ",", dec = ".",
...)
```

- Or use read.csv2: variant used in countries that use a comma as decimal point and a semicolon as field separator.

```
read.csv2(file, header = TRUE, sep = ";", dec = ",",
...)
```

- Read TAB delimited files

```
read.delim(file, header = TRUE, sep = "\t", dec =
".", ...) read.delim2(file, header = TRUE, sep =
"\t", dec = ".", ...)
```

Import txt/csv files using readr

- `tidyverse` includes the package `readr`, a faster and friendly way to read table-like files.
  - `read_csv()`: comma separated (CSV) files
  - `read_tsv()`: tab separated files
  - `read_delim()`: general delimited files
  - `read_fwf()`: fixed width files
  - `read_table()`: tabular files where colums are separated by white-space.
  - `read_log()`: web log files
- `readr` provides consistence column specification (the most significant feature differs from the classical functions)

Import example

- read the data `exp1.csv` and show its head

```
data = read_csv('data/exp1.csv')
head(data, n = 3)
```

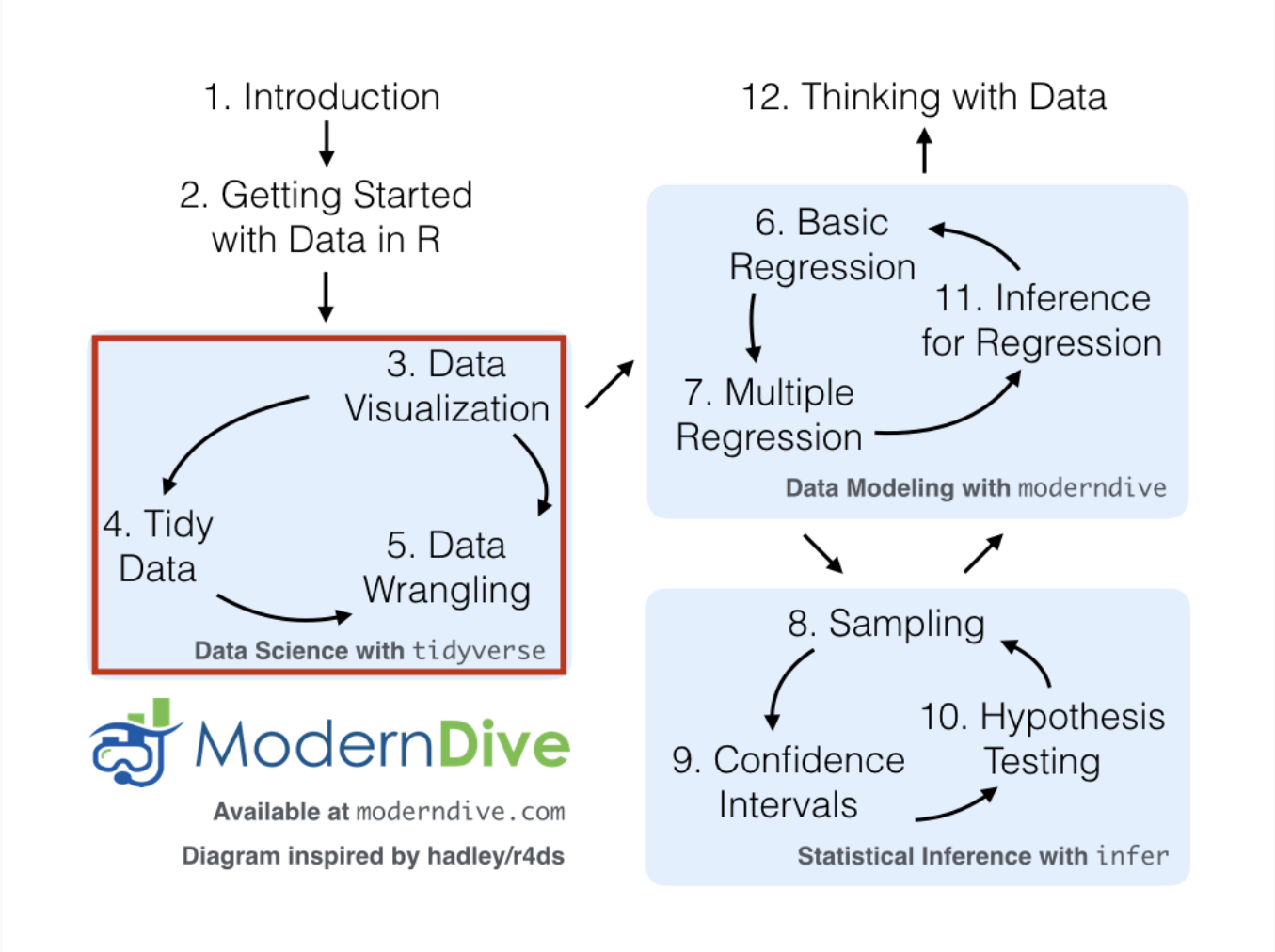
```
## # A tibble: 3 x 8
##   target  setsize dyn     resp    rt   sub   exp correct
##   <chr>     <int> <chr>  <int> <dbl> <int> <int>   <int>
## 1 Absent       12 Static     2 2.58      1     1       1
## 2 Absent        8 Static     2 0.936     1     1       1
## 3 Present      16 Static     1 0.795     1     1       1
```

Export/save Data

- Exporting data is similar to importing data. You can simply change the above mentioned functions from `read*` to `write*`.
  `write.csv()`, *`write.csv2()`*, *`write_csv()`*
- Import and export excel files requires additional package `readxl`.
- Save data for R Data Format: RDS
  Save an object to a file
  saveRDS(object, file = "my_data.rds")
  Restore the object
  readRDS(file = "my_data.rds")

Visualization with `ggplot2`

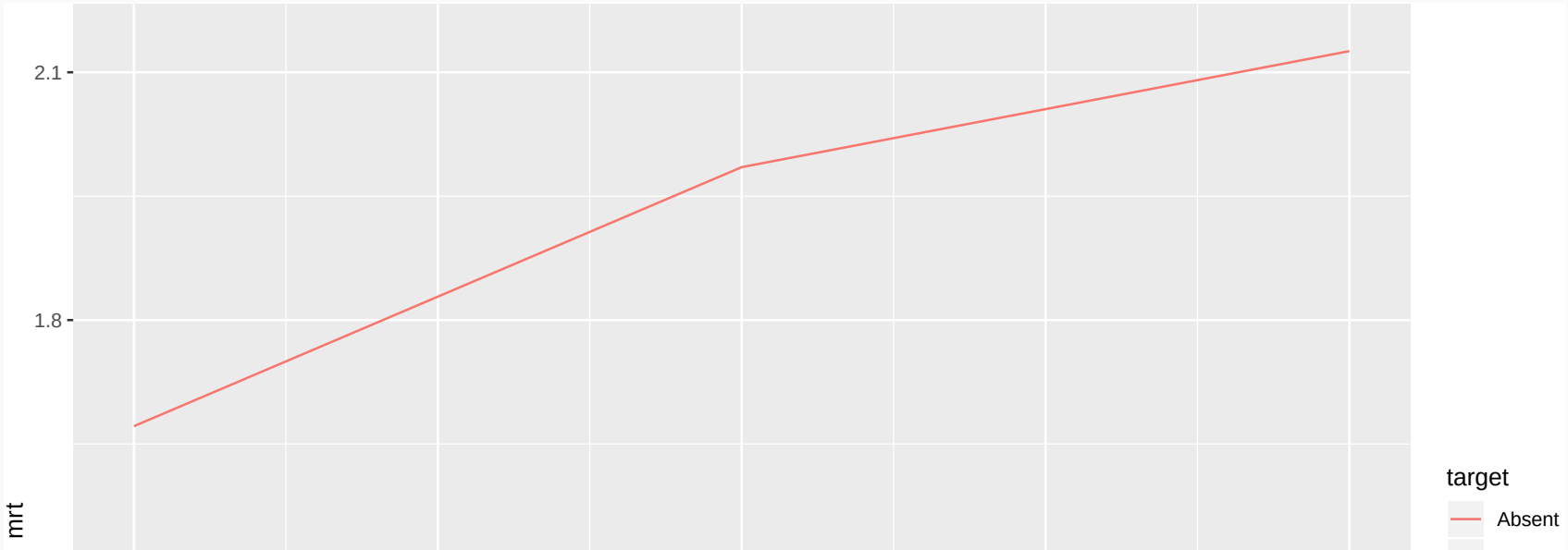
- Visualization is a critical step for explorative data analysis (EDA)



ggplot grammar

- Please refer to the cheatsheet.
  'ggplot(data =) + (mapping = aes(<MAPPINGs))
- Example

```
data %>% group_by(target, setsize) %>%
  summarise(mrt = mean(rt)) %>%
  ggplot(data = .) +
    geom_line(mapping = aes(x = setsize, y = mrt, color = t
```



common problem in plotting

- ggplot using + for layering.
  If you miss the last part, R doesn't think you've typed a complete expression and it's waiting for you to finishe it. ***ESCAPE to abort****.*
- + in a wrong place.
  It has to come at the end of the line, ***not*** *the start.*
  If mappings are the same, you can move it to `ggplot()`*.*

Five Named Graphs - The 5NG

- Scatterplots `geom_point()`
- linegraphs `geom_line()`
- histogram `geom_histogram()`
- Boxplot `geom_boxplot()`
- Barplots `geom_bar()`

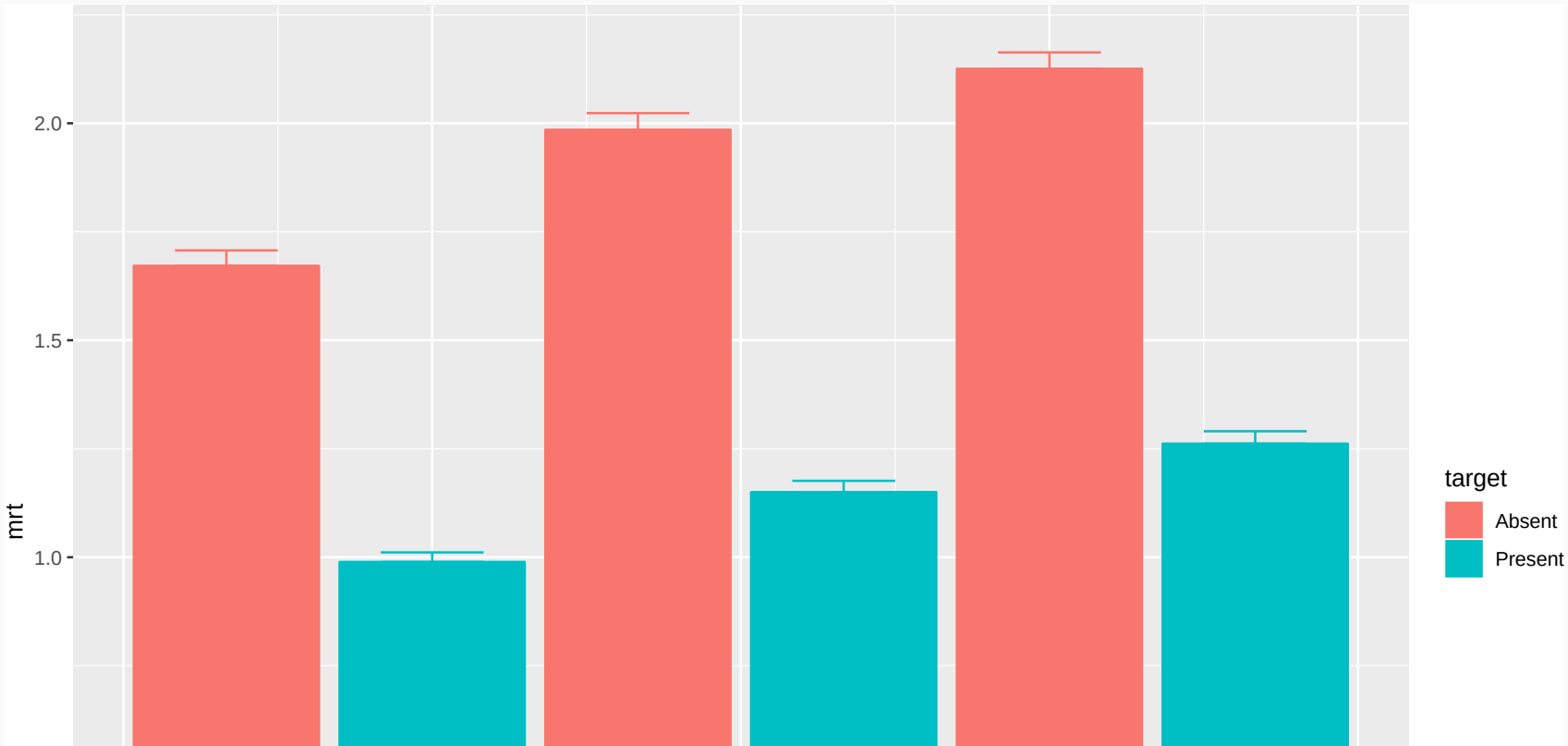
Some tipps

- Barplot by default only plots the counts. If you want to plot mean etc, you need to specify `stat = 'identity'`
- Multiple conditions in Barplot `position = 'dodge'`.
- Be aware of your type of data (category vs. continuous)
  - the data format will affect your graph. Using `factor()` or `as.numeric()` to convert your data type.
- `facet_*()` can be very helpful to examin individual participants

Error bars - an example

- Error bars are common in APA figures.

```
pos <- position_dodge(width = 4)
data %>% group_by( setsize, target) %>%
  summarise(mrt = mean(rt), n = n(), se = sd(rt)/sqrt(n-1))
  ggplot(., aes(x = setsize, y = mrt, color = target, fill
```



Practice

- Let's practise together