R Basics

I HAVE NO IDEA WHAT I'M DOING
R Basics

NEW ZEALAND, TAKE YOUR MUM

NEW ZEALAND, LIKE LORD OF THE RINGS

NEW ZEALAND, WE SHOULD COME

NEW ZEALAND, TAKE YOUR MUM
Week 2.1: Descriptive Statistics in R

- R commands & functions
  - Tidyverse & the Pipe Operator
  - Multiple Functions
- Reading in data
- Saving R scripts
- Descriptive statistics
  - 1 variable
  - 2 variable
  - Grouping
- Plotting
  - Scatterplot
  - Bar graph
R Commands

- Simplest way to interact with R is by typing in commands at the > prompt:

R STUDIO

R
Typing in a simple calculation shows us the result:
- 608 + 28

What’s 11527 minus 283?

Some more examples:
- 400 / 65 (division)
- 2 * 4 (multiplication)
- 5 ^ 2 (exponentiation)
Functions

- More complex calculations can be done with functions:
  - $\sqrt{64}$
  - $\text{abs}(-7)$

  What the function is (square root)
  In parenthesis: What we want to perform the function on

- Can often read these left to right (“square root of 64”)
- What do you think this means?
Arguments

Some functions have **settings** ("arguments") that we can adjust:

- `round(3.14)`
  - Rounds off to the nearest integer (zero decimal places)

- `round(3.14, digits=1)`
  - One decimal place
Week 2.1: Descriptive Statistics in R

- R commands & functions
- Tidyverse & the Pipe Operator
  - Multiple Functions
- Reading in data
- Saving R scripts
- Descriptive statistics
  - 1 variable
  - 2 variable
  - Grouping
- Plotting
  - Scatterplot
  - Bar graph
Tidyverse & the Pipe Operator

- **tidyverse** is a very popular add-on package for many basic data-processing tasks in R

- 2 steps to using:
  - Install Tidyverse—only needs to be done once per computer
Installing Tidyverse: RStudio

- **Tools** menu -> **Install Packages**...
- Type in **tidyverse**
- Leave **Install Dependencies** checked
  - Grabs the other packages that **tidyverse** uses
  - Only need to do this once per computer!
Installing Tidyverse: R

- Packages & Data menu -> Package Installer -> Get List
- Find tidyverse
- Make sure to check Install Dependencies
  - Grabs the other packages that tidyverse uses
  - Only need to do this once per computer!
Tidyverse & the Pipe Operator

- **tidyverse** is a very popular add-on package for many basic data-processing tasks in R

- 2 steps to using:
  - Install Tidyverse—only needs to be done once per computer
  - Load Tidyverse—once per R session
    - `library(tidyverse)`
Tidyverse & the Pipe Operator

- **tidyverse** provides another interface to functions—the **pipe operator**
  
  - `a %>% b()`
    - Start with `a` and apply function `b()` to it
    - `3.14 %>% round()`
  
  - Helpful when we have multiple functions (as we’ll see in a moment)
Week 2.1: Descriptive Statistics in R

- R commands & functions
- Tidyverse & the Pipe Operator
- Multiple Functions
- Reading in data
- Saving R scripts
- Descriptive statistics
  - 1 variable
  - 2 variable
  - Grouping
- Plotting
  - Scatterplot
  - Bar graph
Multiple Functions

Self-Watering Palm Tree

Professor Butt's brain takes a nosedive and out comes his self-watering palm tree. String (A) works jumping jack (B), frightening cat (C) which raises back, and lifts trough (D), causing ball (E) to fall into teacup (F). Spring (G) makes ball rebound into cup (H). Pulling on string (J) which releases stick (I), causing shelf (K) to collapse. Milk can (L) drops on ladle (M) and tension on string (N) tilts shoe (O) against jigger on seltzer bottle (P). Squirting seltzer on ash-can spaniel who hasn't had a bath in four years surprise causes him to turn three somersaults over apparatus (R) and water splashes naturally into bowls, running through spray (S) watering palm (T), and saving yourself a trip to Havana for tropical atmosphere.
Multiple Functions

- The pipe operator makes it easy to do multiple functions in a row

- `-16 %>% sqrt() %>% abs()`
  - Start with -16
  - Then take the square root
  - Then take the absolute value

- Don't get scared when you see multiple pipes!
  - Just read left to right
Using Multiple Numbers at Once

- When we want to use multiple numbers, we **concatenate** them
  - \( c(2, 6, 16) \)  
    - A list of the numbers 2, 6, and 16

- Sometimes a computation requires multiple numbers
  - \( c(2, 6, 16) \) \( \%\%\% \) mean()

- Also a quick way to do the same thing to multiple different numbers:
  - \( c(16, 100, 144) \) \( \%\%\% \) sqrt()
Week 2.1: Descriptive Statistics in R

- R commands & functions
- Tidyverse & the Pipe Operator
- Multiple Functions

Reading in data
- Saving R scripts
- Descriptive statistics
  - 1 variable
  - 2 variable
  - Grouping
- Plotting
  - Scatterplot
  - Bar graph
Modules: Week 2.1: experiment.csv

- Reading **plausible** versus **implausible** sentences
- “Scott chopped the carrots with a **knife**.”

“Scott chopped the carrots with a **spoon**.”

Note: Simulated data; not a real experiment.
Modules: Week 2.1: experiment.csv

- Reading *plausible* versus *implausible* sentences
- Reading time on critical word

- **36 subjects**
  - Each subject sees 30 *items* (sentences): half plausible, half implausible
  - Interested in changes over time, so we’ll track *number of trials remaining* (29 vs 28 vs 27 vs 26…)
Reading in Data

- Make sure you have the dataset at this point if you want to follow along:

Canvas ➔ Modules ➔ Week 2.1 ➔ experiment.csv
Reading in Data – RStudio

- Navigate to the folder in lower-right
- More -> Set as Working Directory

- Open a “comma-separated value” file:
  - experiment <- read.csv('experiment.csv')

Name of the “dataframe” we’re creating (whatever we want to call this dataset)

read.csv is the function name

File name
Reading in Data – RStudio

- Navigate to the folder in lower-right
- More -> Set as Working Directory

- Open a “comma-separated value” file:
  - experiment <- read.csv('experiment.csv')

- General form of this:
  dataframe.name <- read.csv('filename')
Reading in Data – Regular R

- Read in a “comma-separated value” file:
  - `experiment <- read.csv('./Users/scottfraundorf/Desktop/experiment.csv')`

  - **Name of the “dataframe”** we’re creating (whatever we want to call this dataset)

  - **Folder & file name**

- Drag & drop the file into R to get the full folder & filename
Looking at the Data: Summary

- A “big picture” of the dataset:
  - `experiment %>% summary()`

```r
<table>
<thead>
<tr>
<th>Subject</th>
<th>Age</th>
<th>ItemName</th>
<th>Condition</th>
<th>TestingRoom</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length:1080</td>
<td>Length:1080</td>
<td>Length:1080</td>
<td>Length:1080</td>
<td>Min. :1.000</td>
</tr>
<tr>
<td>Class :character</td>
<td>Class :character</td>
<td>Class :character</td>
<td>Class :character</td>
<td>1st Qu.:2.000</td>
</tr>
<tr>
<td>Mode :character</td>
<td>Mode :character</td>
<td>Mode :character</td>
<td>Mode :character</td>
<td>Median :3.000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Mean :3.222</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3rd Qu.:4.250</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Max. :6.000</td>
</tr>
</tbody>
</table>
```

- `summary()` is a very important function!
  - Basic info & descriptive statistics
  - Check to make sure the data are correct
Looking at the Data: Summary

- A “big picture” of the dataset:
  - `experiment %>% summary()`

- We can use `$` to refer to a specific column/variable in our dataset:
  - `experiment$RT %>% summary()`
Looking at the Data: Raw Data

- Let’s look at the data!
  - experiment

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>S05</td>
<td>18</td>
<td>Bear</td>
<td>Plausible</td>
<td>6</td>
<td>22 526.567</td>
</tr>
<tr>
<td>129</td>
<td>S05</td>
<td>18</td>
<td>Jewels</td>
<td>Plausible</td>
<td>6</td>
<td>21 535.227</td>
</tr>
<tr>
<td>130</td>
<td>S05</td>
<td>18</td>
<td>Train</td>
<td>Plausible</td>
<td>6</td>
<td>20 622.015</td>
</tr>
<tr>
<td>131</td>
<td>S05</td>
<td>18</td>
<td>Traffic jam</td>
<td>Plausible</td>
<td>6</td>
<td>19 670.815</td>
</tr>
<tr>
<td>132</td>
<td>S05</td>
<td>18</td>
<td>Knights</td>
<td>Plausible</td>
<td>6</td>
<td>18 572.785</td>
</tr>
<tr>
<td>133</td>
<td>S05</td>
<td>18</td>
<td>Panther</td>
<td>Implausible</td>
<td>6</td>
<td>17 558.831</td>
</tr>
<tr>
<td>134</td>
<td>S05</td>
<td>18</td>
<td>Burglar</td>
<td>Implausible</td>
<td>6</td>
<td>16 766.705</td>
</tr>
<tr>
<td>135</td>
<td>S05</td>
<td>18</td>
<td>Astronaut</td>
<td>Plausible</td>
<td>6</td>
<td>15 510.578</td>
</tr>
<tr>
<td>136</td>
<td>S05</td>
<td>18</td>
<td>Horse</td>
<td>Implausible</td>
<td>6</td>
<td>14 593.603</td>
</tr>
<tr>
<td>137</td>
<td>S05</td>
<td>18</td>
<td>Firefighter</td>
<td>Plausible</td>
<td>6</td>
<td>13 786.298</td>
</tr>
<tr>
<td>138</td>
<td>S05</td>
<td>18</td>
<td>Cheese</td>
<td>Implausible</td>
<td>6</td>
<td>12 510.862</td>
</tr>
<tr>
<td>139</td>
<td>S05</td>
<td>18</td>
<td>Orange</td>
<td>Implausible</td>
<td>6</td>
<td>11 601.291</td>
</tr>
<tr>
<td>140</td>
<td>S05</td>
<td>18</td>
<td>Computer</td>
<td>Plausible</td>
<td>6</td>
<td>10 464.144</td>
</tr>
<tr>
<td>141</td>
<td>S05</td>
<td>18</td>
<td>Lawnmower</td>
<td>Plausible</td>
<td>6</td>
<td>9 734.754</td>
</tr>
<tr>
<td>142</td>
<td>S05</td>
<td>18</td>
<td>Fishing</td>
<td>Implausible</td>
<td>6</td>
<td>8 625.639</td>
</tr>
</tbody>
</table>

[ reached 'max' / getOption("max.print") -- omitted 938 rows ]
Looking at the Data: Raw Data

- Ack! That’s too much!
  How about just a few rows?
  - `experiment %>% head()`
  - `experiment %>% head(n=10)`
Reading in Data: Other Formats

- Excel:
  - Install the `readxl` package (only needs to be done once)
    - `install.packages('readxl')`
  - Then, to read in Excel data:
    - `library(readxl)`
    - `experiment <- read_excel('/Users/scottfraundorf/Desktop/experiment.xlsx', sheet=2)`

Excel files can have multiple sheets/tabs. In this case, we are saying to use sheet 2.
**Reading in Data: Other Formats**

- **SPSS:**
  - Uses the `haven` package—already installed as part of `tidyverse`
  - Then, to read in SPSS data:
    - `library(haven)`
    - `experiment <- read_spss('/Users/scottfraundorf/Desktop/experiment.spss')`
    - This package also includes `read_sas` and `read_stata`
Week 2.1: Descriptive Statistics in R

- R commands & functions
- Tidyverse & the Pipe Operator
- Multiple Functions
- Reading in data
- Saving R scripts
  - Descriptive statistics
    - 1 variable
    - 2 variable
    - Grouping
  - Plotting
    - Scatterplot
    - Bar graph
R Scripts

- Save & reuse commands with a script

R STUDIO

R

File -> New Document
**R Scripts**

- Run commands without typing them all again
- **R Studio:**
  - Code -> Run Region -> Run All: Run *entire* script
  - Code -> Run Line(s): Run just what you’ve highlighted/selected
- **R:**
  - Highlight the section of script you want to run
  - Edit -> Execute
- Keyboard shortcut for this:
  - Ctrl+Enter (PC), ⌘+Enter (Mac)
R Scripts

- Saves times when re-running analyses
- Other advantages?
- Some:
  - Documentation for yourself
  - Documentation for others
  - Reuse with new analyses/experiments
  - Quicker to run—can automatically perform one analysis after another
**R Scripts—Comments**

- Add `#` before a line to make it a comment
  - Not commands to R, just notes to self (or other readers)

```r
# This command gives us a summary of the
# "RT" variable using tidyverse
experiment$RT %>% summary()
```

- Can also add a `#` to make the rest of a line a comment
  - `experiment$RT %>% summary() #awesome`
Week 2.1: Descriptive Statistics in R

- R commands & functions
  - Tidyverse & the Pipe Operator
  - Multiple Functions
- Reading in data
- Saving R scripts
  - Descriptive statistics
    - 1 variable
      - 2 variable
      - Grouping
    - Plotting
      - Scatterplot
      - Bar graph
Descriptive Statistics

- So far, we’ve used `summary()` to get a high-level overview of our data

- Now, let’s use tidyverse to start testing specific descriptive statistics
Descriptive Statistics

- Let’s try getting the mean of the RT column
- We start with our `experiment` dataframe...

```r
experiment %>%
```
Descriptive Statistics

- Let’s try getting the mean of the RT column
- We start with our `experiment` dataframe...
  ...then, we start using `summarize()` to build a table of descriptive statistics...

```r
experiment %>% summarize()
```

Dataframe name
Descriptive Statistics

- Let’s try getting the mean of the RT column
- We start with our `experiment` dataframe...
- ...then, we start using `summarize()` to build a table of descriptive statistics...
- ..and, in particular, let’s get the `mean()` of the RT column

```
experiment %>% summarize(MyMean=mean(RT))
```

**Dataframe name**

**Name of the column in the resulting table (can be whatever you want)**

**Descriptive function**

**Variable of interest**
Week 2.1: Descriptive Statistics in R

- R commands & functions
- Tidyverse & the Pipe Operator
- Multiple Functions
- Reading in data
- Saving R scripts
  - Descriptive statistics
    - 1 variable
    - 2 variable
      - Grouping
      - Plotting
        - Scatterplot
        - Bar graph
Wow! That was complicated!
But, once we have learned this general format, we can easily make more complex tables...

```r
experiment %>% summarize(MyMean=mean(RT), MySD=sd(RT))
```

- Adds the SD (standard deviation) as a second column
  - Other relevant functions: `median()`, `min()`, `max()`
Descriptive Statistics: 2 Variables

```
experiment %>% summarize(MyMean=mean(RT),
                         MySD=sd(RT))
```

- Generic form:

```
dataframe.name %>%
  summarize(TableHeader1=function(VariableName),
            TableHeader2=function(VariableName))
```
Week 2.1: Descriptive Statistics in R

- R commands & functions
  - Tidyverse & the Pipe Operator
  - Multiple Functions
- Reading in data
- Saving R scripts
  - Descriptive statistics
    - 1 variable
    - 2 variable
- Grouping
- Plotting
  - Scatterplot
  - Bar graph
Descriptive Statistics: Grouping

- We often want to look at a dependent variable as a function of some independent variable(s)
  - e.g., RTs for Plausible vs. Implausible sentences

- We add an intermediate step – the `group_by()` function
  - `experiment %>% group_by(Condition) %>% summarize(M=mean(RT))`
  - “Group the data by Condition, then get the mean RT”
**Descriptive Statistics: Grouping**

- We often want to look at a dependent variable as a function of some independent variable(s).
- We add an intermediate step – the `group_by()` function.
- Can even group by 2 or more variables:
  - `experiment %>% group_by(Subject, Condition) %>% summarize(M=mean(RT))`
  - Each subject’s mean RT in each condition
Descriptive Statistics: Grouping

- We often want to look at a dependent variable as a function of some independent variable(s)
- We add an intermediate step – the `group_by()` function
- Can even group by 2 or more variables:
  - `experiment %>% group_by(Subject, Condition) %>% summarize(M=mean(RT))`
- Generic version of this:
  - `dataframe.name %>% group_by(IndependentVar1, IndependentVar2) %>% summarize(TableHeader=function(DependentVar))`
Descriptive Statistics: Grouping

- With `group_by()` and the `n()` function, we can create contingency tables for categorical variables:
  ```r
  experiment %>% group_by(Subject, Condition) %>% summarize(Observations=n())
  ```

Now, we are not getting the mean of any particular dependent variable.

We just want a frequency count of the number of observations for each subject in each condition.
Week 2.1: Descriptive Statistics in R

- R commands & functions
- Tidyverse & the Pipe Operator
- Multiple Functions
- Reading in data
- Saving R scripts
  - Descriptive statistics
    - 1 variable
    - 2 variable
    - Grouping
  - Plotting
    - Scatterplot
    - Bar graph
**Plotting**

- Tidyverse also includes a function for creating plots: `ggplot()`

- Each ggplot consists of two main elements:
  - *Mapping* the variables onto one or more aesthetic elements in your plot (e.g., X and Y axes, color, line type)
Plotting

- Tidyverse also includes a function for creating plots: `ggplot()`

- Each ggplot consists of two main elements:
  - Mapping the variables onto one or more aesthetic elements in your plot (e.g., X and Y axes, color, line type)
Plotting

- Tidyverse also includes a function for creating plots: `ggplot()`

- Each ggplot consists of two main elements:
  - *Mapping* the variables onto one or more aesthetic elements in your plot (e.g., X and Y axes, color, line type)
Plotting

- Tidyverse also includes a function for creating plots: `ggplot()`

- Each ggplot consists of two main elements:
  - **Mapping** the variables onto one or more aesthetic elements in your plot (e.g., X and Y axes, color, line type)
**Plotting**

- Tidyverse also includes a function for creating plots: `ggplot()`

- Each ggplot consists of two main elements:
  - **Mapping** the variables onto one or more aesthetic elements in your plot (e.g., X and Y axes, color, line type)
  - **Adding** one or more visual elements (**geoms**) to depict each observation (e.g., points, bars, lines)
Plotting: Scatterplot

- Does RT change over the course of the experiment?

- Basic scatterplot:
  - experiment %>%
    ggplot(aes(x=TrialsRemaining, y=RT)) +
    geom_point()

Here, we are saying how we want to translate the variables into visual form: the X axis will represent TrialsRemaining, and the Y axis will represent the RT variable.

Then, we want to represent each observation with a point.
Plotting: Scatterplot
Plotting: Scatterplot

- Our scatterplot:
  - `experiment %>%
    ggplot(aes(x=TrialsRemaining, y=RT)) +
    geom_point()`

- More generically:
  - `experiment %>%
    ggplot(aes(x=XAxisVariableName, y=YAxisVariableName)) +
    geom_point()`
**Plotting: Scatterplot**

- We can add additional variables into the plot by specifying what **aesthetic** element they should be mapped to:
  - `experiment %>% ggplot(aes(x=TrialsRemaining, y=RT, color=Condition)) + geom_point()`

- Now, we represent the **Condition variable** with the **color** of each point
Plotting: Scatterplot
Week 2.1: Descriptive Statistics in R

- R commands & functions
- Tidyverse & the Pipe Operator
- Multiple Functions
- Reading in data
- Saving R scripts
- Descriptive statistics
  - 1 variable
  - 2 variable
  - Grouping
- Plotting
- Scatterplot
- Bar graph
Plotting: Bar Graph

- Now let’s make a bar graph to compare conditions
- **KEY POINT:** A bar graph displays **means**—that is, **summarized data**
- Thus, we first need to compute those means
Plotting: Bar Graph

- Now let’s make a bar graph to compare conditions
- **KEY POINT:** A bar graph displays means—that is, *summarized data*
- Thus, we first need to compute those means

<table>
<thead>
<tr>
<th>Subject</th>
<th>Age</th>
<th>ItemName</th>
<th>Condition</th>
<th>TestingRoom</th>
<th>TrialsRemaining</th>
<th>RT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>35</td>
<td>Dolphin</td>
<td>Implausible</td>
<td>2</td>
<td>29710229</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>35</td>
<td>Traffic jam</td>
<td>Plausible</td>
<td>2</td>
<td>28522872</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>35</td>
<td>Jewels</td>
<td>Plausible</td>
<td>2</td>
<td>27708618</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>35</td>
<td>Senator</td>
<td>Plausible</td>
<td>2</td>
<td>26541017</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>35</td>
<td>Panther</td>
<td>Implausible</td>
<td>2</td>
<td>25718076</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>35</td>
<td>Train</td>
<td>Plausible</td>
<td>2</td>
<td>24630957</td>
<td></td>
</tr>
</tbody>
</table>
Plotting: Bar Graph

- Now let’s make a bar graph to compare conditions
- `experiment %>%
  group_by(Condition) %>%
  summarize(MeanRT = mean(RT)) %>%
  ggplot(aes(x = Condition, fill = Condition,
            y = MeanRT)) +
  geom_col()

Here, we are grouping by Condition and getting the mean RT in each condition.

The x-axis and bar color will represent Condition.

The y-axis (bar height) will represent MeanRT.

`geom_col()` for bar graphs.
Plotting: Bar Graph
Plotting: Bar Graph

- Generic form:

  ```r
dataframe.name %>%
  group_by(IndependentVariable) %>%
  summarize(M=mean(DependentVariable)) %>%

ggplot(aes(x=IndependentVariable, fill=IndependentVariable, y=M)) +

geom_col()
```