F28ED: Stats Lab 2 | Introduction to R Studio for Data Analysis

Dr Peter McKenna

16 November 2020

Where to start this week

- In the last session we made it to plotting the continuous vector `conv_len`.
- The instructions for plotting the Likert data from the `like` vector are available in the Lab 1 instructions.
- Lab 1 materials compliment this session, so please finish them off when you get the chance.

Lab 2: Tips to getting started

- Ensure you open the `.Rproj` file as it will load the data and your scripts from last session.
- Check the data is in your directory (as a .csv), or the Global environment (as a tibble).
- If you do not have the data follow the `Reading in the data` instructions from lab 1.
- Create a new script for todays session; e.g., "lab-2.R".

```
# For ease, I'll load in the data again so you can get going quickly
# You only need to follow this step if the object `data` is not already loaded

# create tibble from data

data <-
  read_csv("f28ed_data.csv")
```

Lab 2 outline

- In this session we are going to be using the same data from Lab 1.
- I am going to demonstrate some data wrangling techniques offered by the Tidyverse, including:
  - extracting rows and columns
  - appending and adding columns
- To finish I will show you how to:
  - summarise the data
  - create barplots with error bars
  - analyse parametric & non-parametric paired samples data

Load packages

- load the necessary packages to get started.
- Do this by opening and running your `housekeeping.R` script.
- Note: you **do not** need to install these again, just to load them.

Fun with dplyr & tidyr

- Let's have a look at some of tidyverse functionality.
- *dplyr* and *tidyr* are useful for all sorts of simple operations.
- We'll try the functions `filter`, `mutate` and `select`:
  - `filter` : extract rows
  - `select` : extract columns
  - `mutate` : compute and append new/existing columns

Live Coding Demo 1: Fun with dplyr & tidyr

```
# before we start let's arrange the data by id
# this will show more clearly that there are 2 observations per participant

data <-       # to apply changes to your tibbles assign the changes to the original tibble name
  data %>%
  arrange(id)

# you can check the amendments quickly with `head`

head(data)

#### Filter

# let's have a look at the male voice speaker data only by filtering by row

data %>%
  filter(voice_type != "female") # "!=" denotes exclude

# what about from the first 15 participants?

data %>%
  filter(id %in% 1:15) # note the "%in%" operator denotes within

# you can also combine these arguments if you like

data %>%
  filter(voice_type != "female",
               id %in% 1:15)

#### Mutate

# say that the first 6 participants were female
```

```r
# we can use mutate to create a column with this info

data %>%
  filter(id %in% 1:6) %>%
  mutate(gender = "female")

# note that because we are not using the assign function these operations do not affect the `data` tibb

# if we knew that 1-6 were female and the rest were male we could use the logical argument `ifelse` to

data %>%
  mutate(gender = ifelse(id %in% 1:6, "female", "male")) # i.e., if the logical argument is satisfied t

#### Select

# let's select vectors of interest with `select`

data %>%
  select(id, like) # you can use a colon to select multiple vectors

# what about if you want to study the `conv_len` data

data %>%
  select(id, voice_type, conv_len)

# you could assign this to an object for later use

conv_data <-
  data %>%
  select(id, voice_type, conv_len)
```

Take home message

- These operations will be useful for you to tidy and manipulate your data.
- If they do not solve the query you have I recommend consulting:
  - R for Data Science chapters 9-12
  - Data Wrangling and Tidying cheatsheet
- The cheatsheet is a very useful tool for quickly navigating to the correct solution - I use it all the time.
- OK, let's move onto generating descriptive statistics from the data.

Generating descriptive statistics

- To generate a summary the data we use the `summarise` function.
- `summarise` allow you to specify the elements of the data you wish to generate a summary for.
- Remember, our ordinal data is non-parametric and our raio data is parametric.
- So, we are interested in the median and mode of `like`, and the mean and standard deviation of `conv_len`.
- R does not have a built in function for the mode so we have to make one first.

```r
# Create the function `getmode` for calculating the mode

getmode  <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}

# you'll see the function generated in the global environment, below the tibbles
```

Overall data summary

- Let's think back to the data and hypothesis before generating a summary.
- We are interested in whether there are measurable differences in paticipant's Likert ratings (i.e., our `like` vector) and conversation length (i.e., our `conv_len` vector) between the female or male voice smart speakers.
- We expect people to rate the female voice higer and spend more time talking to that speaker compared to the male voice speaker.
- So, we want to a generate summary of the outcome variables `like` and `conv_len` by the condition `speaker_type`.

Important word on summarising different data types

- Ordinal data that is non-parametric should be summarised in terms of the median and mode *NOT* the mean and standard deviation.
- Interval or Ratio data that is parametric should be summarised by the mean and standard deviation.

Live Coding Demo 2: Generating descriptive statistics

```r
# summarise can do all of the above in one go, but let's build it up

data %>%
  summarise(med_like = median(like)) # here we generate the median of `like` and call the summary "med_

# this works because R has some built in knowledge of mathematical operations (e.g., "mean", "sd" ...et

# we can also use the `getmode` function we created to generate both the median and mode of the Likert

data %>%
  summarise(med_rating = median(like),    # new vector `med_rating` is the median of like
            mode_rating = getmode(like))  # new vector `mode_rating` is the mode of like

# Ok, we've delt with the ordinal data, but what about the ratio data

data %>%
  summarise(med_rating = median(like),
            mode_rating = getmode(like),
            avg_conv = mean(conv_len),       # new vector `avg_conv` is the mean of conv_len
            sd_conv = sd(conv_len))          # new vector `sd_conv` is S.D. of conv_len

# Nice, so now we have a summary of all participants peformance
# The next step is to group the data by our condition of interest (voice_type) to generate by-condition
```

```r
# This summary should give a rough indication to whether our hypothesis has been supported or not

data %>%
  group_by(voice_type)  %>%                 # "group_by" allows us to group the data prior to summary ge
  summarise(med_rating = median(like),
            mode_rating = getmode(like),
            avg_conv = mean(conv_len),
            sd_conv = sd(conv_len))

# have a look at the output and see if it supports your hypothesis
```

Exercise 1

1. Report the descriptive statistics for both conditions. Here's a guide.
2. What trends in the data do you notice from the by-condition summary?
3. Would you expect there to be a difference between the two conditions?

Conversation Length barplot with error bars

- When we generate barplots we also want to include the standard error.
- This barplot compares the summary statistics for conversation length between female and male voice smart speakers.

Live Coding Demo 3: Conversation Length barplot with error bars

```r
# Some of this will be familiar from lab 1
# As before we begin by making a summary object (here I've called it 'bp') that contains the means and
# The summary needs to include the mean and S.D to create the plot

bp <-
  data %>%                             # create new object `bp` from data
  group_by(voice_type) %>%             # group by `voice_type`
  summarise(avg_conv = mean(conv_len),   # new vector avg_conv is the mean of conv_len
            sd_conv = sd(conv_len))      # new vector sd_conv is the S.D. of conv_len

# then pass to ggplot2
# However, on this occasion we are generating a bar plot with error bars, so the arguments are slightly

p <-                                            # we are going to call the plot "p"
  bp %>%
  ggplot(., mapping = aes(x = voice_type,
                          y = avg_conv,
                          fill = voice_type)) +  # notice mapping comes first here
  geom_bar(stat = "identity",                    # plot from objects values as they are
           colour = "black") +                   # apply black outline to bars
  geom_errorbar(aes(ymin = avg_conv - sd_conv/sqrt(30),
                    ymax = avg_conv + sd_conv/sqrt(30)), width = .2) +    # add standard error bars / s
  labs(x = "\nSmart Speaker Voice Type",
```

```
          y = "Conversation Length (seconds)\n") +
  theme_classic(base_size = 20) +                # use ggplot's calssic theme size 20 text
  theme(legend.position = "none")                # remove the legend - it's not useful here

# save the plot into the fig_output folder
# you can rename the plot so it has a meaningful label

ggsave("fig_output/conv_len_bar.png", p, width = 15, height = 10)
```

Testing the hypotheses | pt 1 likert data

- Have a look at the Likert plot your generated in lab 1; it should be in your `fig_out` folder.
- You can see that the female speaker was evaluated more favourably.
- We can test whether this difference is statistically significant using a non-parametric test.
- A Wilcoxon Test is appropirate for assessing differences between paired samples of the Likert data.
- Here's a useful guide on non-parametric tests of group differences in R Studio

Live Coding Demo 4: Testing the hypotheses

```
wilcox.test(like ~ voice_type,  # assess the likert data by voice_type
            data = data,        # use the tibble `data`
            paired = TRUE)      # data is paired samples (i.e., within subjects)
```

- **Note**: if your design is between-subjects you can add `paired=FALSE`.
- See the RStudio HELP section for further information.

Interpreting and reporting the Wilcoxon output

- You are interested in the 3rd line of output (the one beginning with V).
- $V = 318.5$, *p-value* = 2.045e-05 (or $p = 2.045$ * 10^-05).
- We are interested in the value of p, **NOT** V.
- Convention holds that this is statistically significant (begins at $p < 0.05$).

Reporting the result

- As $p < 0.001$ you can conclude that…
- Participant's Likert scale responses showed that they preferred the female voice speaker ($Mdn = 4$) to the male voice speaker ($Mdn = 2$), $p < 0.001$.

Testing the hypothesis | pt 2 conversation length data

- Ok, so let's move onto the conversation length data.
- This data is normally distributed (have a look again at the histogram plot) so we can used a paired samples t-test to peform the analysis.
- Here's a useful guide for completing a t-test in R Studio.

```r
# the syntax is similar to wilcox.test on this time you want to model the conv_len data

t.test(conv_len ~ voice_type,
       data = data,
       paired = TRUE)
```

Exercise 2

1. Report the findings of the t-test using APA conventions. Bear in mind the sample is paired, not independent, as participants took part in both conditions.
2. What conclusions can you draw from both statistical tests?
3. Were the hypothesis supported?