assignment 2

10878862

10/12/2021

Hi, Dr. Andrew, this is my work of **assignment 2**, hope you like it~

In this assignment, I am going to working on these 3 dataset, conducting including wrangling data, visualising the data, constructing different ANOVA model, and comparing the relationship between ANOVA and linear model.

Load all Packages

First, I'm going to load all the packages that should be used in the three questions, having already used the `install.packages` function directly in the console, e.g., `install.packages("tidyverse")` .

Now we need to add these packages into our environment with function `library()` (e.g., `library(tidyverse)`). The main packages include:

- `tidyverse` : a collection of R packages using widely in data science. In the dataset, we specifically use ggplot2, dplyr and purrr.
  - `ggplot` : used to visualised the data
  - `dplyr` : includes `select` , `mutate` , `filter` , `summarise` and `arrange` .
  - `purrr` : used to exclude extreme value (by assign function for each rows).
- `afex` : provides a set of functions that make calculating ANOVAs easy.
- `emmeans` : is enormously useful for folks wanting to do post hoc comparisons among groups after fitting a model.

```
library(tidyverse)
library(afex)
library(emmeans)
library(knitr)
library(ggpubr)
library(ggridges)
library(car)
library(ggstatsplot)
library(rstatix)
library(broom)
```

Qestion 1

Summary of the experiment

The dataset that we are going to process are the results from an experiment of between-subjects design with ninety-six participants randomly assigned into two groups, which one group is presented to normally words and the other is exposed to degraded words, comparing the the relationship between visual quality and response time of words. There are 3 steps to cope with this dataset: 1. Data Wrangling; 2. Data summary; 3. Data Visualization; 4. Statistical Analysis; 5. Results output

Data Wrangling

read in data

Now, we will read in the experimental data that we are going to analyses as a variable in our environment.

The data comprises 96 rows with each subjects' ID number, condition of words assigned and response time of subjects.

```
Q1 <- read_csv("assignment_2_dataset_1.csv")
```

Understanding our data

As the tibble is one of the features of the tidyverse, we use the `as_tibble` default to change the type of the data to make sure our dataset is in tibble format.

```
as_tibble(Q1)
```

```
## # A tibble: 96 x 3
##    participant condition   response_time
##          <dbl> <chr>               <dbl>
##  1           1 condition_a          984.
##  2           2 condition_a         1005.
##  3           3 condition_a          979.
##  4           4 condition_a         1040.
##  5           5 condition_a         1008.
##  6           6 condition_a          979.
##  7           7 condition_a         1012.
##  8           8 condition_a         1018.
##  9           9 condition_a         1014.
## 10          10 condition_a          992.
## # ... with 86 more rows
```

Then, Using the function `head()` and `str()` , we get a basic understanding of our data's frame.

- With `head()` , we could see the name, type and first 6 rows of the data, variable "condition" belongs to "chr", which means character variable. And the other two is "dbl", which means "double".
- with `str()` , we could see more details, including the number of each variables, and the structure of this dataset.

```
head(Q1)
```

```
## # A tibble: 6 x 3
##   participant condition   response_time
##         <dbl> <chr>               <dbl>
## 1           1 condition_a          984.
## 2           2 condition_a         1005.
## 3           3 condition_a          979.
## 4           4 condition_a         1040.
## 5           5 condition_a         1008.
## 6           6 condition_a          979.
```

```
str(Q1)
```

```
## spec_tbl_df [96 x 3] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ participant  : num [1:96] 1 2 3 4 5 6 7 8 9 10 ...
##  $ condition    : chr [1:96] "condition_a" "condition_a" "condition_a" "condition_a" ...
##  $ response_time: num [1:96] 984 1005 979 1040 1008 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   participant = col_double(),
##   ..   condition = col_character(),
##   ..   response_time = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Change the Data into Suitable Format

As we have an understanding about our data, now we can start to manipulate it. The first step is placing it into a appropriate format that could be recognized by different functions we might want to use later with `tidyverse` packages.

We have known our data is in a long format (which is correct, because we could see the one observation (each participant in its condition) is in just one column). However, as I described above, the type of condition is *chr*, but we want a factor here corresponding to our experimental design.

So, first, we use `mutate()` to change the class of the **condition** from *character* (chr) to *factor* (fac).

Then, we change the Name of the Factors to present our experiment more clearly.

With **condition_a** and **condition_b**, we have no idea what are them except there are 2 conditions. So, we want to mutate the names of the **condition** to **Normal** (normally displayed) and **Degrade** (degraded displayed).

Here, we use : - `<-` (a function to assign a value to a data) to cover the original **Q1** with class-changed condition. - the mark `%>%` , called `pipe` . The `pipe` is used to ensure information is passed with code through different lines. - `mutate()` could replace variables and preserves existing ones. - `recode()` could switch the values based their name or position.

```
Q1_tided <- Q1 %>%
  mutate(condition = factor(condition)) %>%
  rename(RT = response_time) %>%
  mutate(condition = dplyr::recode(condition,
                            "condition_a" = "Normal",
                            "condition_b" = "Degrade"))
```

Filter the data

To exclude extreme value in the data (extreme value might affect our next statistical analysing, and might not satisfy the requirements of statistics). It might because the distraction of the participants.

We use map function in `purrr` packages (part of `tidyverse`). - `purrr` : - `map` : allow you to replace many for loops with code that is both more succinct and easier to read. - `append()` : allow you to add elements to a vector (combine two to one). - `filter()` : part of `dplyr` , retaining all rows that satisfy your requirements. Here, we set the threshold that the response time should be smaller than mean ± 3 Standard deviation (of specific condition). - `select()` : keeps only the variables you choose. - `unlist()` : to change a *list* to a vector, with `use.names = False` to drop the name information - `is.na()` : returns logical value showing if your value is *NA* or not, together with filter could drop the case with *NA*.

```
Normal_fil <- Q1_tided %>%
  filter(condition == "Normal") %>%
  select(RT) %>%
  map(function(x) ifelse(((x-mean(x))/sd(x)) <= 3, x, NA))

Degrade_fil <- Q1_tided %>%
  filter(condition == "Degrade") %>%
  select(RT) %>%
  map(function(x) ifelse(((x-mean(x))/sd(x)) <= 3, x, NA))

Q1_fil <- Q1_tided
Q1_fil$RT <- append(unlist(Normal_fil, use.names = FALSE), unlist(Degrade_fil, use.names = FALSE))
Q1_fil%>% filter(!is.na(RT))
```

```
## # A tibble: 96 x 3
##    participant condition    RT
##          <dbl> <fct>     <dbl>
##  1           1 Normal     984.
##  2           2 Normal    1005.
##  3           3 Normal     979.
##  4           4 Normal    1040.
##  5           5 Normal    1008.
##  6           6 Normal     979.
##  7           7 Normal    1012.
##  8           8 Normal    1018.
##  9           9 Normal    1014.
## 10          10 Normal     992.
## # ... with 86 more rows
```

Finally, let's view our data again to see what happened to our original data and is everything correct.

```
Q1_fil
```

```
## # A tibble: 96 x 3
##    participant condition    RT
##          <dbl> <fct>     <dbl>
##  1           1 Normal     984.
##  2           2 Normal    1005.
##  3           3 Normal     979.
##  4           4 Normal    1040.
##  5           5 Normal    1008.
##  6           6 Normal     979.
##  7           7 Normal    1012.
##  8           8 Normal    1018.
##  9           9 Normal    1014.
## 10          10 Normal     992.
## # ... with 86 more rows
```

Data Profile

To show the distribution, the representative value and other basic information of the dataset, we summary the data. For example, presenting the mean reaction, the standard deviance and the sample size of Reaction Time in two conditions.

Here, we use `summarise()` in `dplyr` packages to calculate these statistical variables.

To do this, first we use `group_by` function to divide our data into two conditions (groups), so then the `summarise` could return to the summaries in two different conditions.

To get a better understanding with our data, we calculate: - mean: the average of the values - sd: standard deviance - se: standard error - median: the middle number in a sorted, ascending or descending - max/min: the max and minimum value in a serious of data - quantile(25% and 75%): three/one quarters of the way up this rank order - variance: is the expectation of the squared deviation of a random variable from its population mean or sample mean - sample size: the number of observations

```
Q1_summarise <- Q1_fil %>%
  group_by(condition) %>%
  summarise(mean_RT = mean(RT),
            sd_RT = sd(RT),
            se = sd(RT) / sqrt(length(n)),
            median_RT = median(RT),
            max_RT = max(RT),
            min_RT = min(RT),
            quantile_RT25 = quantile(RT, 0.25),
            quantile_RT75 = quantile(RT, 0.75),
            var_RT = var(RT),
            number = n())
```

Make Table

Then, we use kable() function in knitr packages to draw a more formal table that could both show this information to our audience and also put in the future potential paper.

- Considering the unit of **response time** is *ms*, we drop all the decimals with `digits = 0` .
- `caption` to give a title to this table
- `col.names` to rename the name of columns in the table.
- `format.args` to cancel the scientific notation, and add , to represent *thousands*; and align left the characters in the table with `align = "lllllll"`

```
knitr::kable(
  Q1_summarise,
  caption = "Profile of reaction time (RT) in two conditions", col.names = c('condition', 'Mean(ms)', 'SD (ms)', 'SE (ms)',
'median (ms)', 'max (ms)', 'min (ms)', '25% quantile (ms)', '75% quantile (ms)', 'variance (ms)', 'N'), digits = 0, format.a
rgs = list(big.mark = ",",
  scientific = FALSE), align = "lllllllllll")
```

Profile of reaction time (RT) in two conditions

condition	Mean(ms)	SD (ms)	SE (ms)	median (ms)	max (ms)	min (ms)	25% quantile (ms)	75% quantile (ms)	variance (ms)	N
Normal	1,002	21	21	1,003	1,040	945	990	1,018	442	48
Degrade	1,020	24	24	1,021	1,075	970	1,003	1,032	554	48

Check Asumptions

Outliers

Outliers can be easily identified using box plot methods, implemented in the R function `identify_outliers()` from `rstatix` package.

We could see there were no extreme outliers.

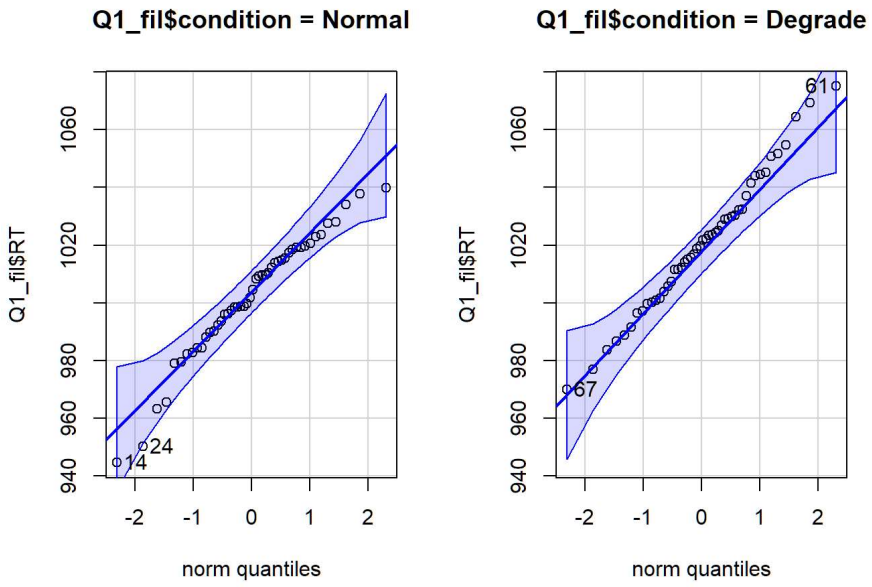
```
Q1_fil %>%
  group_by(condition) %>%
  identify_outliers(RT)
```

```
## # A tibble: 1 x 5
##   condition participant    RT is.outlier is.extreme
##   <fct>           <dbl> <dbl> <lgl>      <lgl>
## 1 Normal             14  945. TRUE       FALSE
```

Normality Assumption

qq plot is a way fitted to measure if the probabilities follow a uniform distribution. We load the `qqPlot` function in the `Car` package.

```
qqPlot(Q1_fil$RT, distribution="norm", Q1_fil$condition)
```



Compute Shapiro-Wilk test of normality

The score were normally distributed (p > 0.05) for each group, as assessed by Shapiro-Wilk's test of normality.

```
Q1_fil %>%
  group_by(condition) %>%
  shapiro_test(RT)
```

```
## # A tibble: 2 x 4
##   condition variable statistic     p
##   <fct>     <chr>        <dbl> <dbl>
## 1 Normal    RT           0.966 0.169
## 2 Degrade   RT           0.991 0.973
```

Homogeneity Test

ANOVA assumes that the variance of the residuals is equal for all groups.

Also, we test for homogeneity of variance using *levenTest*, *Fligner-Killeen* test or *Barlett's test* (These three are alternative with each other. Considering the level of robust, Barlett's > leven > FK test.

For all the tests, the p-value is not significant (p > 0.05), so there was homogeneity of variances for two conditions).

```
bartlett.test(RT~condition, data = Q1_fil)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  RT by condition
## Bartlett's K-squared = 0.5924, df = 1, p-value = 0.4415
```

```
leveneTest(RT~condition, data = Q1_fil)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value Pr(>F)
## group  1  0.4219 0.5176
##       94
```

```
fligner.test(RT~condition, data = Q1_fil)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  RT by condition
## Fligner-Killeen:med chi-squared = 0.29669, df = 1, p-value = 0.586
```

data visualization

Before formal statistical analysis, we also need make it visible to help us have a better intuitive understanding about our data. It could inspire us to determine which kind of statistic should be used.

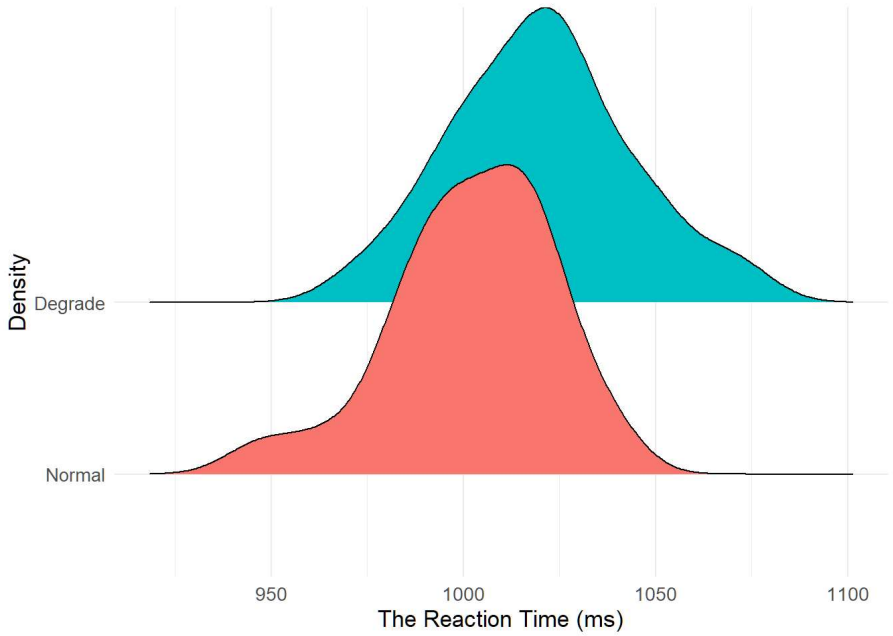
Here, we use density plot and scatter plot.

plot 1 - density plot

It is necessary to see the distribution of the data and the density curve is the intuitive and convenient way for us to know it.

- `geom_density_ridges` : used to generate the density(distribution) of the data.
- `guides` : used to set the properties of legends.
- `them` : combined with `axis.text.x` to set the properties of text in x axis.

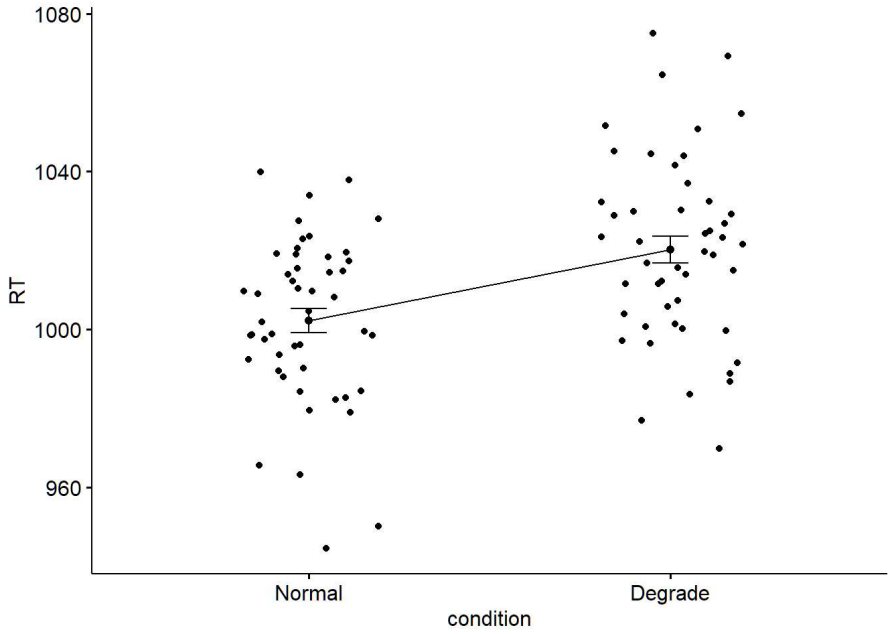
```
plot1_1 <- Q1_fil %>%
  ggplot(aes(x = RT, y = fct_reorder(condition, .fun = mean, RT))) +
  geom_density_ridges(height = .5, aes(fill = condition)) +
  theme_minimal() +
  theme(text = element_text(size = 13)) +
  guides(fill = 'none') +
  labs(x = "The Reaction Time (ms)",
       y = "Density")
plot1_1
```



plot 2 - scatter plot

It is useful to see the general “position” of the whole data. The simplest way is with scatter plot. Besides `ggplot() + geom_jitter()` . We could use `ggline` in **ggpubr** to draw a scatter plot easily.

```
Plot1_2 <- ggline(Q1_fil, x = "condition", y = "RT",
       add = c("mean_se", "jitter"),
       order = c("Normal",  "Degrade"),
       ylab = "RT", xlab = "condition")

Plot1_2
```



Statistical Analysis

Build Analysis of variance model

Let's first build an ANOVA model, and use summary() to get the effect and emmeans() to conduct the post-hoc analysis.

- `aov_4` : can be used for most situations of comparisons.
- `emmeans` : is used to do post hoc comparisons among groups after fitting a model.

```
model1 <- aov_4(RT ~ condition + (1 | participant), data = Q1_fil)

summary(model1)
```

```
## Anova Table (Type 3 tests)
##
## Response: RT
##           num Df den Df    MSE      F     ges    Pr(>F)
## condition      1     94 497.58 15.781 0.14375 0.0001393 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
emm1 <- emmeans(model1, pairwise ~ condition)

emm1
```

```
## $emmeans
##  condition emmean   SE df lower.CL upper.CL
##  Normal      1002 3.22 94      996     1009
##  Degrade     1020 3.22 94     1014     1027
##
## Confidence level used: 0.95
##
## $contrasts
##  contrast         estimate   SE df t.ratio p.value
##  Normal - Degrade    -18.1 4.55 94  -3.973  0.0001
```

Alternatively, when comparing the difference of two groups, the aov function is also suitable.

```
one.way <- aov(RT ~ condition, data = Q1_fil)

summary(one.way)
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## condition    1   7852    7852   15.78 0.000139 ***
## Residuals   94  46772     498
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Thus, we could know there is significant difference of response time between normally-displayed and degraded-displayed condition. Which we could conclude that participants spend more time to respond a degraded-displayed words than noramlly-displayed words.

Add P-values and Significance Levels to ggplots

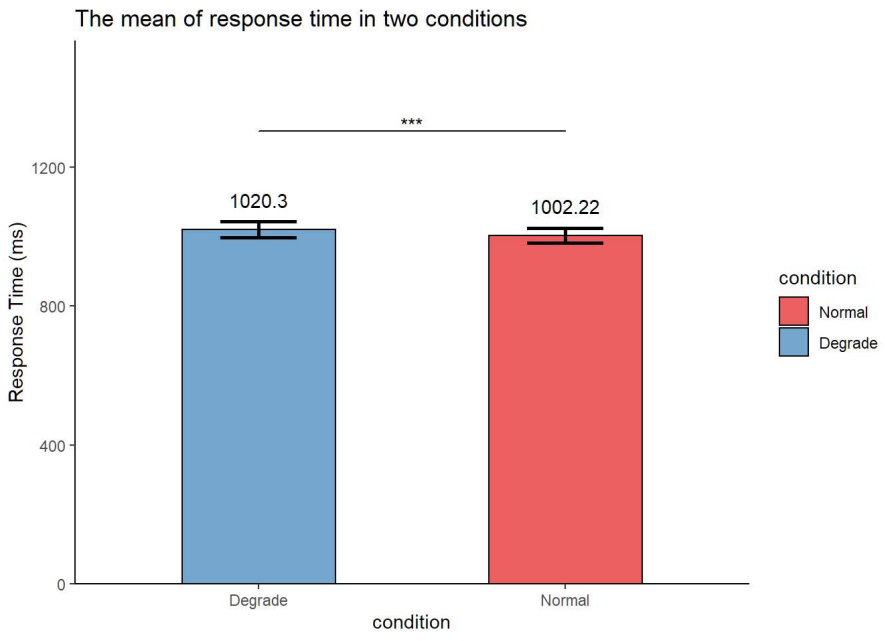
After get the p-value of the ANOVA, now we could draw new pictures with significance for make our results of analysis more clearly and preparing for our potential publication.

We could add the significance manually with `geom_signif()` based on previous results. Especially for barplot, we cannot automatically and directly compare the difference between groups.

plot3 - histogram plot with p value

- `summarise` : used to calculate mean and sd, prepared for following barplot.
- `geom_errorbar` : used to add errorbar to each bar.
- `geom_text` : used to add the value in the plot.
- `geom_signif` : used to add significant value and line in the plot
- `scale_y_continuous` : make the bar started in the zero and zoom the whole plot.
- `scale_x_discrete` : used to set the values for discrete x and y scale aesthetics.
- `labs` : change the characters of the lab.
- `theme` : using classic type of theme.
- `scale_fill_brewer` : change the color of the plot, here we use “Set1” in the whole processing.

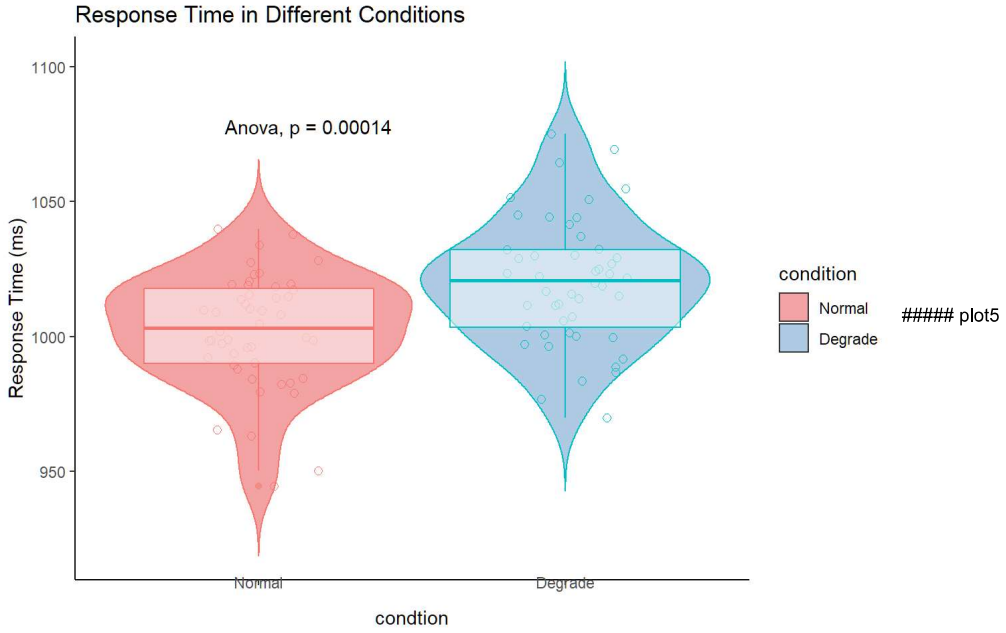
```
plot1_3 <- Q1_fil %>%
  group_by(condition) %>%
  summarise(mean_RT = mean(RT), sd_RT = sd(RT)) %>%
  ggplot(aes(x = condition, y = mean_RT)) +
  geom_bar(aes(fill = condition),
           stat="identity",
           alpha = 0.7,
           color ="black",
           width = 0.5) +
  geom_text(aes(label = round(mean_RT, 2)), vjust = -1.5) +
  geom_errorbar(aes(ymin = mean_RT - sd_RT,
                    ymax = mean_RT + sd_RT),
                width = 0.25,
                size = 1) +
  geom_signif(comparisons = list(c("Normal", "Degrade")),
              annotations="***",
              y_position = 1300, tip_length = 0, vjust=0.4)+
  scale_y_continuous(expand = expansion(mult = c(0, 0.2))) +
  scale_x_discrete(limits = rev) +
  labs(x = "condition",
       y = "Response Time (ms)",
       title = "The mean of response time in two conditions") +
  theme_classic() +
  scale_fill_brewer(palette = "Set1")

plot1_3
```



plot 4 - scatter plot with volin and boxplot

Another way to add the p-value of the comparisons is add `stat_compare_means` directly (a function from `ggpubr`), which is a easier way than above. - `geom_violin` : draw a violin-shaping plot showing a compact display of a continuous distribution - `geom_jitter` : adding a small amount of random variation to the location of each point - `geom_boxplot` : the boxplot compactly displays the distribution of a continuous variable - `set.seed(123)` : to make sure the plot we draw every time is the same (especially for the `gemo_jitter`)

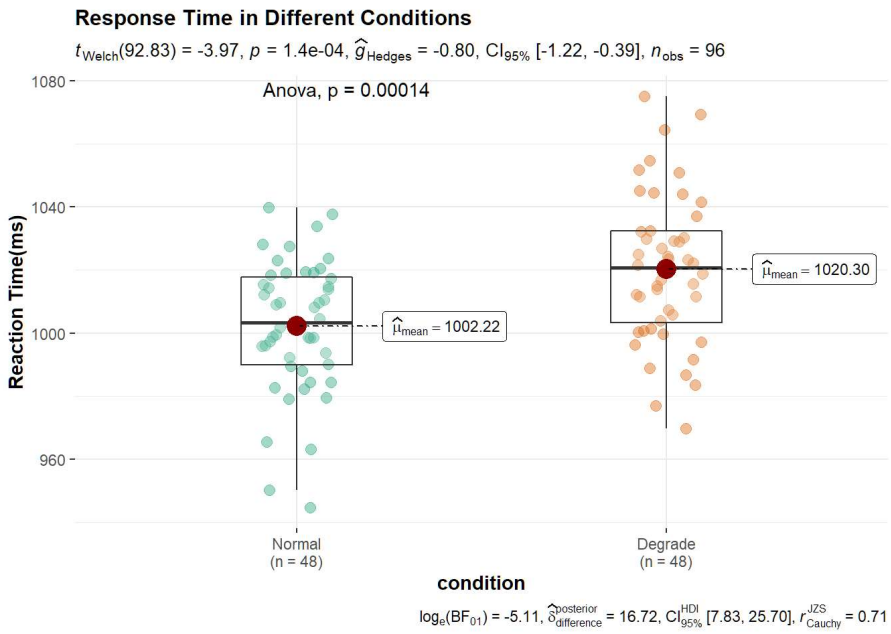
```
set.seed(123)

plot1_4 <- Q1_fil %>%
  ggplot(aes(x = condition, y = RT, colour = condition)) +
  geom_violin(aes(fill = condition),
              trim = FALSE,
              width = 1,
              alpha = 0.4) +
  geom_jitter(shape = 21,
              width = .2,
              alpha = 2,
              size = 2) +
  geom_boxplot(alpha = .5) +
  scale_fill_brewer(palette = "Set1") +
  guides(colour = 'none') +
  theme_classic() +
  theme(axis.text.x = element_text(angle = 0, vjust = 5, hjust = .5)) +
  labs(title = "Response Time in Different Conditions",
       x = "condtion",
       y = "Response Time (ms)") +
  stat_compare_means(method = "anova")

plot1_4
```



plot5

Another way to show plenty of paramater in a plot is using `ggbetweenstats` in `ggstatslpot`, `ggstatsplot` is an extension of `ggplot2` package. It creates graphics with details from statistical tests included in the plots themselves. It provides an easier API to generate information-rich plots for statistical analysis of continuous data.

```
plot1_5 <- ggbetweenstats(Q1_fil, x = condition, y = RT,
               plot.type="box",
               ylab = "Reaction Time(ms)",
               title = "Response Time in Different Conditions"
               ) +
               stat_compare_means(method = "anova")
plot1_5
```



Qestion 2

Question 2 is similar to question 1, except we take the caffeine consumption into our consideration. Because we knew there might be some relationship between response time and caffeine consumption.

Also, in this part, we want to compare the relationship between anova and linear model.

Because most part of script is the same as the question 1 above, so we omit some of the narrative description in the data wrangling and visualizing part, but emphasize the difference.

Data Wangling

reading in data

Similarly, first, we load the data and look through the format and type of the data.

```
Q2 <- read_csv("assignment_2_dataset_2.csv")
as_tibble(Q2)
```

```
## # A tibble: 96 x 4
##    participant condition   response_time caffeine
##          <dbl> <chr>               <dbl>    <dbl>
##  1           1 condition_a          984.        1
##  2           2 condition_a         1005.        1
##  3           3 condition_a          979.        3
##  4           4 condition_a         1040.        2
##  5           5 condition_a         1008.        2
##  6           6 condition_a          979.        3
##  7           7 condition_a         1012.        2
##  8           8 condition_a         1018.        0
##  9           9 condition_a         1014.        1
## 10          10 condition_a          992.        1
## # ... with 86 more rows
```

```
head(Q2)
```

```
## # A tibble: 6 x 4
##   participant condition   response_time caffeine
##         <dbl> <chr>               <dbl>    <dbl>
## 1           1 condition_a          984.        1
## 2           2 condition_a         1005.        1
## 3           3 condition_a          979.        3
## 4           4 condition_a         1040.        2
## 5           5 condition_a         1008.        2
## 6           6 condition_a          979.        3
```

```
str(Q2)
```

```
## spec_tbl_df [96 x 4] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ participant  : num [1:96] 1 2 3 4 5 6 7 8 9 10 ...
##  $ condition    : chr [1:96] "condition_a" "condition_a" "condition_a" "condition_a" ...
##  $ response_time: num [1:96] 984 1005 979 1040 1008 ...
##  $ caffeine     : num [1:96] 1 1 3 2 2 3 2 0 1 1 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   participant = col_double(),
##   ..   condition = col_character(),
##   ..   response_time = col_double(),
##   ..   caffeine = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Then, we revise the name of the data to make it more readable.

```
Q2_tided <- Q2 %>%
  mutate(condition = factor(condition)) %>%
  rename(RT = response_time) %>%
  mutate(condition = dplyr::recode(condition,
                            "condition_a" = "Normal",
                            "condition_b" = "Degrade"))
```

Now, I am going to exclude the extreme data.

```
Normal_fil2 <- Q2_tided %>%
  filter(condition == "Normal") %>%
  select(RT) %>%
  map(function(x) ifelse(((x-mean(x))/sd(x)) <= 3, x, NA))

Degrade_fil2 <- Q2_tided %>%
  filter(condition == "Degrade") %>%
  select(RT) %>%
  map(function(x) ifelse(((x-mean(x))/sd(x)) <= 3, x, NA))

Q2_fil <- Q2_tided
Q2_fil$RT <- append(as.numeric(unlist(Normal_fil2)), as.numeric(unlist(Degrade_fil2)))
Q2_fil%>% filter(!is.na(RT))
```

```
## # A tibble: 96 x 4
##    participant condition    RT caffeine
##          <dbl> <fct>     <dbl>    <dbl>
##  1           1 Normal     984.        1
##  2           2 Normal    1005.        1
##  3           3 Normal     979.        3
##  4           4 Normal    1040.        2
##  5           5 Normal    1008.        2
##  6           6 Normal     979.        3
##  7           7 Normal    1012.        2
##  8           8 Normal    1018.        0
##  9           9 Normal    1014.        1
## 10          10 Normal     992.        1
## # ... with 86 more rows
```

Summarising our Data

It's totally the same.

```
Q2_summarise <- Q2_fil %>%
  group_by(condition) %>%
  summarise(mean_RT = mean(RT),
            sd_RT = sd(RT),
            se = sd(RT) / sqrt(length(n)),
            median_RT = median(RT),
            max_RT = max(RT),
            min_RT = min(RT),
            quantile_RT25 = quantile(RT, 0.25),
            quantile_RT75 = quantile(RT, 0.75),
            var_RT = var(RT),
            number = n())

knitr::kable(
  Q2_summarise,
  caption = "Profile of reaction time (RT) in two conditions", col.names = c('condition', 'Mean(ms)', 'SD (ms)', 'SE (ms)',
'median (ms)', 'max (ms)', 'min (ms)', '25% quantile (ms)', '75% quantile (ms)', 'variance (ms)', 'N'), digits = 0, format.a
rgs = list(big.mark = ",",
  scientific = FALSE), align = "lllllllllll")
```

Profile of reaction time (RT) in two conditions

condition	Mean(ms)	SD (ms)	SE (ms)	median (ms)	max (ms)	min (ms)	25% quantile (ms)	75% quantile (ms)	variance (ms)	N
Normal	1,002	21	21	1,003	1,040	945	990	1,018	442	48
Degrade	1,020	24	24	1,021	1,075	970	1,003	1,032	554	48

Now, we started to build our anova model

builing our ANOVA model

First, we think this is an ANCOVA model, we need compare the participants' response time in two conditions controlling for the caffeine.

Check Asumptions

Normality test

First, we conduct linear model and augment data to add fitted values and residuals by using the `augment` function in `broom` packages.

```
# Fit the model, the covariate goes first
model2_nmt <- lm(RT ~ caffeine + condition, data = Q2_fil)
# Inspect the model diagnostic metrics
model2_nmt.metrics <- augment(model2_nmt) %>%
  select(-.hat, -.sigma, -.fitted) # Remove details
head(model2_nmt.metrics, 3)
```

```
## # A tibble: 3 x 6
##      RT caffeine condition .resid  .cooksd .std.resid
##   <dbl>    <dbl> <fct>      <dbl>    <dbl>      <dbl>
## 1  984.        1 Normal    -16.7  0.00458      -0.758
## 2 1005.        1 Normal      3.56 0.000208      0.162
## 3  979.        3 Normal    -26.9  0.0246       -1.23
```

```
shapiro_test(model2_nmt.metrics$.resid)
```

```
## # A tibble: 1 x 3
##   variable                  statistic p.value
##   <chr>                         <dbl>   <dbl>
## 1 model2_nmt.metrics$.resid     0.992   0.815
```

Homogeneity tests

This can be checked using the Levene's test:

```
model2_nmt.metrics %>% levene_test(.resid ~ condition)
```

```
## # A tibble: 1 x 4
##     df1   df2 statistic     p
##   <int> <int>     <dbl> <dbl>
## 1     1    94     0.122 0.727
```

Outlinears

There were no outliers in the data, as assessed by no cases with standardized residuals greater than 3 in absolute value.

```
model2_nmt.metrics %>%
  filter(abs(.std.resid) > 3) %>%
  as.data.frame()
```

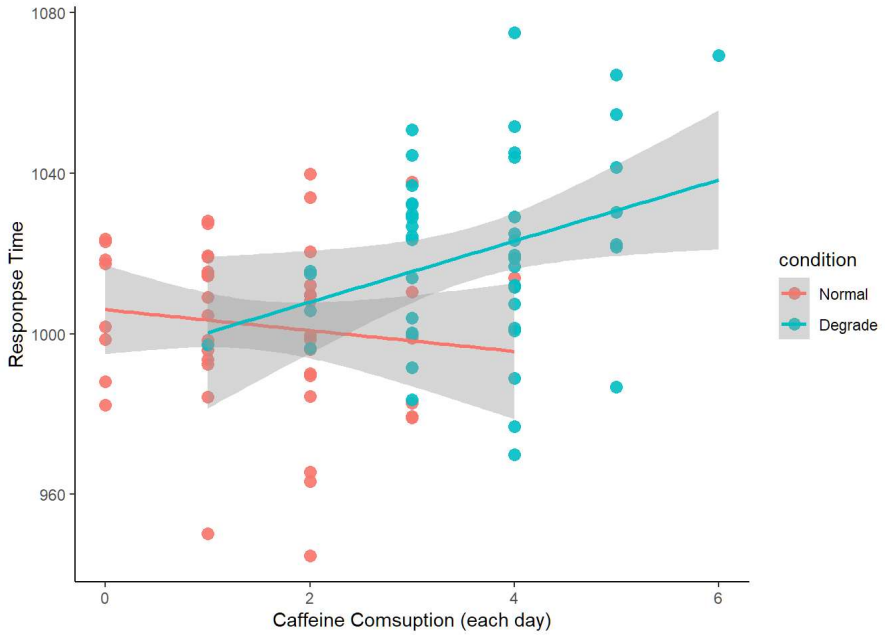
```
## [1] RT         caffeine   condition  .resid     .cooksd    .std.resid
## <0 rows> (or 0-length row.names)
```

Linearity assumption

Specifically, for ANCOVA, we create a scatter plot between the covariate (i.e., caffeine) and the dependent variable (i.e., response time) to test the linearity assumption.

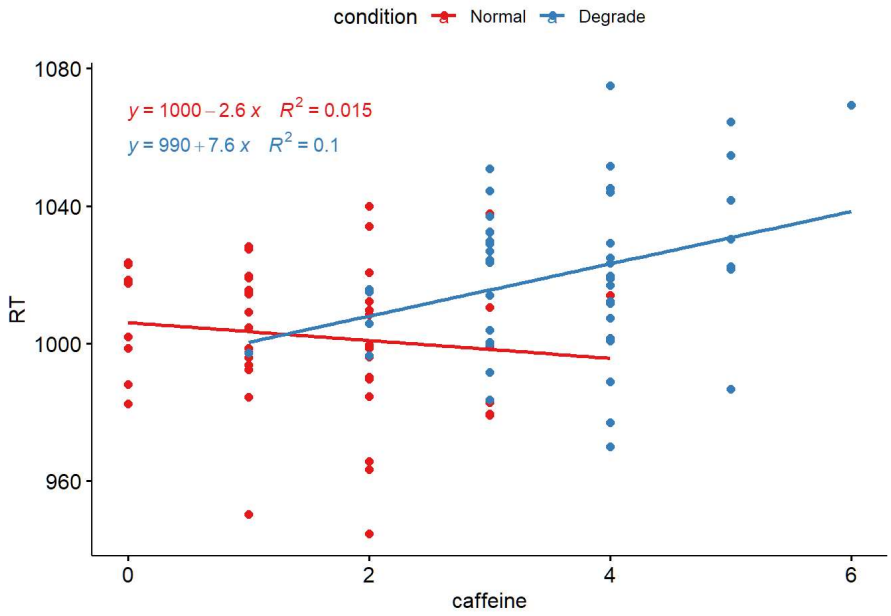
We could use `ggplot2` .

```
set.seed(123)
plot2_1 <- ggplot(Q2_fil, aes(x = caffeine, y = RT,  colour = condition)) +
  geom_point(size = 3, alpha = .9) +
  geom_smooth(method = lm)+
  labs(x = "Caffeine Comsuption (each day)",
       y = "Responpse Time") +
  theme_classic() +
  theme(text = element_text(size = 11))

plot2_1
```



Alternatively, we could also use `ggscatter` .

```
plot2_2 <- ggscatter(Q2_fil,
          x = "caffeine",
          y = "RT",
          color = "condition",
          add = "reg.line",
          palette = "Set1")+
  stat_regline_equation(
    aes(label =  paste(..eq.label.., ..rr.label.., sep = "~~~~"),
    color = condition))

plot2_2
```



According to the plot, we could see there is interaction between covariate and out come variable. So, it violates the linearity assumption. Technically, we could not conduct ANCOVA model or we at least could not “call” it “ANCOVA model” (although we could still construct this kind of model). But it remains controversial.

So, here, we conduct both ANCOVA and then factorial ANOVA.

Model construction

ANCOVA model

```
model2_ancova <- aov_4(RT ~ caffeine + condition + (1 | participant), data = Q2_fil, factorize = FALSE)

anova(model2_ancova)
```

```
## Anova Table (Type 3 tests)
##
## Response: RT
##           num Df den Df    MSE      F      ges  Pr(>F)
## caffeine       1     93 496.95 1.1188 0.011888 0.29291
## condition      1     93 496.95 3.5730 0.036998 0.06184 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
emmeans(model2_ancova, pairwise ~ condition)
```

```
## $emmeans
##  condition emmean   SE df lower.CL upper.CL
##  Normal      1005 4.08 93      997     1013
##  Degrade     1018 4.08 93     1010     1026
##
## Confidence level used: 0.95
##
## $contrasts
##  contrast         estimate   SE df t.ratio p.value
##  Normal - Degrade    -12.8 6.77 93  -1.890  0.0618
```

Linear Model

First, we need to set the contrasts. If we want to change the level of reference, we could use `fct_relevel`. But here, we don't need to.

```
contrasts(Q2_fil$condition)
```

```
##         Degrade
## Normal        0
## Degrade       1
```

According to this, we get the formula: RT = 998.564 + 2.469 * caffeine + 12.791 * condition (Degrade)

Then, we could calculate the adjust mean for each group.

For Normal group:

RT = 998.564 + 2.469 * 2.552083 + 12.791 * (0) = 1005

For Degrade group: RT = 998.564 + 2.469 * 2.552083 + 12.791 * (1) = 1018

We could see they're totally the same as the estimate results by `emmean()` function following the ANCOVA. Here, we build a relationship between ANCOVA and linear model.

```
model2_lm <- lm(RT ~ caffeine + condition, data = Q2_fil)
model2_lm
```

```
##
## Call:
## lm(formula = RT ~ caffeine + condition, data = Q2_fil)
##
## Coefficients:
##      (Intercept)          caffeine  conditionDegrade
##          998.564             2.469            12.791
```

```
mean(Q2_fil$caffeine)
```

```
## [1] 2.552083
```

Factorial ANOVA model

According to this, we could see a significant interaction between caffeine and condition.

```
model2_fact <- aov_4(RT ~ condition * caffeine + (1 | participant), data = Q2_fil, na.rm = TRUE, factorize = FALSE)

summary(model2_fact)
```

```
## Anova Table (Type 3 tests)
##
## Response: RT
##                   num Df den Df    MSE      F      ges  Pr(>F)
## condition              1     92 476.54 0.9835 0.010578 0.32393
## caffeine               1     92 476.54 1.2056 0.012934 0.27508
## condition:caffeine     1     92 476.54 4.9834 0.051384 0.02802 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
emmeans(model2_fact, pairwise ~ condition*caffeine)
```

```
## $emmeans
##  condition caffeine emmean   SE df lower.CL upper.CL
##  Normal        2.55    999 4.68 92      990     1009
##  Degrade       2.55   1012 4.70 92     1003     1021
##
## Confidence level used: 0.95
##
## $contrasts
##  contrast                                          estimate   SE df t.ratio
##  Normal 2.55208333333333 - Degrade 2.55208333333333    -12.7 6.63 92  -1.917
##  p.value
##   0.0583
```

linear model with interaction

We add the interaction (condition * caffeine) into the linear model and get the formula:

RT = 1006.051 + (-2.593) * caffeine + (-13.339) * condition (Degrade) + 10.204 * caffeine * condition(Degraded)

For Normal group:

RT = 1006.051 + (-2.593) * 2.552083 + (-13.339) * (0) + 10.204 * (2.552083) * (0) = 999

For Degrade group: RT = 1006.051 + (-2.593) * 2.552083 + (-13.339) * (1) + 10.204 * (2.552083) * (1) = 1012

Also, we could see when we reconstruct our ANOVA and corresponding linear model, the results of linear model could always calculate the adjusted value of the ANOVA model.

```
model2_lm2 <- lm(RT ~ caffeine + condition + condition * caffeine, data = Q2_fil)
model2_lm2
```

```
##
## Call:
## lm(formula = RT ~ caffeine + condition + condition * caffeine,
##     data = Q2_fil)
##
## Coefficients:
##               (Intercept)                  caffeine
##                  1006.051                    -2.593
##          conditionDegrade  caffeine:conditionDegrade
##                   -13.339                    10.204
```

```
mean(Q2_fil$caffeine)
```

```
## [1] 2.552083
```

In summary, with the covariate introduced, it is hard for us to test hypothesis correctly and simply. If we regard it as a simple ANCOVA, we could conclude that there is no group difference of response time no matter what level of visual quality is when controlling for the caffeine consumption. While, things are not that easy because it violate the linearity assumption, we could not treat it as traditional ANCOVA. Then, we conducted factorial model, it suggests that there is interaction between condition and caffeine consumption. Participants in normal group consume less caffeine than Degraded group, the interaction might be a reason why there is no group effect beween two conditions.

If in a real experiment, I may improve this study to control the consumption of caffeine of participants. For example, we could use mix design, setting caffeine consuming level (high/medium/low dose of caffeine consuming) as between subject factors, to explore the interaction between condition and caffeine consumption. I hypothesis: the effect of visual quality have different kind of influence on participant's pronounce time to a word when they consume different dose of caffeine. When they consume high dose of cafe, they would respond quicker when they see a normal word compared to the degraded word; while, in low-level caffeine consumption condition, there may be no difference between 2 types of words.

Qestion 3

Because in the beginning part of this script (including functions and logic) is the same as the question 1 or 2 above, so we omit some of the narrative description in the data wrangling and visualizing part, but emphasize the difference.

Data Wangling

```
Q3 <- read_csv("assignment_2_dataset_3.csv")
```

```
## Rows: 148 Columns: 5
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (5): participant, positiveprime_positivetarget, positiveprime_negativeta...
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
as_tibble(Q3)
```

```
## # A tibble: 148 x 5
##    participant positiveprime_positivetarget positiveprime_neg~ negativeprime_po~
##          <dbl>                         <dbl>             <dbl>             <dbl>
##  1           1                         1502.             1657.             1535.
##  2           2                         1535.             1581.             1623.
##  3           3                         1563.             1495.             1545.
##  4           4                         1492.             1692.             1601.
##  5           5                         1560.             1584.             1497.
##  6           6                         1552.             1603.             1598.
##  7           7                         1554.             1614.             1520.
##  8           8                         1606.             1571.             1541.
##  9           9                         1489.             1563.             1492.
## 10          10                         1613.             1511.             1736.
## # ... with 138 more rows, and 1 more variable:
## #   negativeprime_negativetarget <dbl>
```

```
head(Q3)
```

```
## # A tibble: 6 x 5
##   participant positiveprime_positivetarget positiveprime_neg~ negativeprime_pos~
##         <dbl>                         <dbl>             <dbl>             <dbl>
## 1           1                         1502.             1657.             1535.
## 2           2                         1535.             1581.             1623.
## 3           3                         1563.             1495.             1545.
## 4           4                         1492.             1692.             1601.
## 5           5                         1560.             1584.             1497.
## 6           6                         1552.             1603.             1598.
## # ... with 1 more variable: negativeprime_negativetarget <dbl>
```

```
str(Q3)
```

```
## spec_tbl_df [148 x 5] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ participant                : num [1:148] 1 2 3 4 5 6 7 8 9 10 ...
##  $ positiveprime_positivetarget: num [1:148] 1502 1535 1563 1492 1560 ...
##  $ positiveprime_negativetarget: num [1:148] 1657 1581 1495 1692 1584 ...
##  $ negativeprime_positivetarget: num [1:148] 1535 1623 1545 1601 1497 ...
##  $ negativeprime_negativetarget: num [1:148] 1596 1508 1479 1417 1592 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   participant = col_double(),
##   ..   positiveprime_positivetarget = col_double(),
##   ..   positiveprime_negativetarget = col_double(),
##   ..   negativeprime_positivetarget = col_double(),
##   ..   negativeprime_negativetarget = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

We are going to wrangle and revise our data to make it more clear and to a more manipulated format.

In this experiment, we use a long format to make it clearer for the audience to understand the meaning of the data. Also, some software requires us to input long format data. In this part, we use two functions:

- `pivot_longer` : to change it from wide format to long format.

Notice: we could also use function `gather` which might be used equivalently although might be more complete

- `mutate` : mutate could change the variable name of the data, cause they're too long and might influence our reading and drawing pictures later. So, we change it to a more remembered name, corresponding to our condition - priming valence (**positive/negative**) and following displaying type of valence (**positive/negative**). Also, to make it clear, we use the abbreviation (**pos/neg**). We could use function `rename(dataname, pos_pos = positiveprime_positivetarget, pos_neg = positiveprime_negativetarget, neg_pos = negativeprime_positivetarget,` to change the variable name.

```
Q3_longer <- Q3 %>%
  pivot_longer(cols = c(positiveprime_positivetarget,
                        positiveprime_negativetarget,
                        negativeprime_positivetarget,
                        negativeprime_negativetarget),
               names_to = "valence",
               values_to = "RT") %>%
    mutate(valence = dplyr::recode(valence,
                           "positiveprime_positivetarget" = "pos_pos",
                           "positiveprime_negativetarget" = "pos_neg",
                           "negativeprime_positivetarget" = "neg_pos",
                           "negativeprime_negativetarget" = "neg_neg"))
```

Now, we have made our data more pretty, but it yet not represent our design of study (which is 2x2 within-subjects design). Here, we use `separate` to separate the **valence** into 2 factors with 2 levels each. In `separate` function:

- `col` tells which variable we want to separate.
- `into` allows it to generate two new variables.

- `sep = "_"` tells the R how (here, we used underscore "_") we want to separate this data.
- `factor` used to change the class of variable.

```
Q3_tidied <- Q3_longer %>%
  separate(col = "valence", into = c("Prime", "Target"),
           sep = "_") %>%
      mutate(Prime = factor(Prime), Target = factor(Target))
```

We can now view what happend to our data through data wrangling.

```
Q3_tidied
```

```
## # A tibble: 592 x 4
##    participant Prime Target    RT
##          <dbl> <fct> <fct>  <dbl>
##  1           1 pos   pos    1502.
##  2           1 pos   neg    1657.
##  3           1 neg   pos    1535.
##  4           1 neg   neg    1596.
##  5           2 pos   pos    1535.
##  6           2 pos   neg    1581.
##  7           2 neg   pos    1623.
##  8           2 neg   neg    1508.
##  9           3 pos   pos    1563.
## 10           3 pos   neg    1495.
## # ... with 582 more rows
```

Data summarisation

Let's generate the Mean, Standard Deviation, and sample size for each of our four conditions.

```
Q3_summarised <- Q3_tidied %>%
  group_by(Prime, Target) %>%
  summarise(mean = mean(RT), sd = sd (RT), median_RT = median(RT), n())
```

Then we use summarised data to generate table.

```
knitr::kable(
  Q3_summarised,
  caption = "The profile of response time in four conditions", col.names = c('Prime', 'Target', 'mean (ms)', 'SD (ms)', 'med
ian (ms)', 'N'), digits = 0, format.args = list(big.mark = ",",
  scientific = FALSE), align = "llllll")
```

The profile of response time in four conditions

Prime	Target	mean (ms)	SD (ms)	median (ms)	N
neg	neg	1,547	52	1,550	148
neg	pos	1,563	50	1,562	148
pos	neg	1,567	54	1,570	148
pos	pos	1,547	45	1,550	148

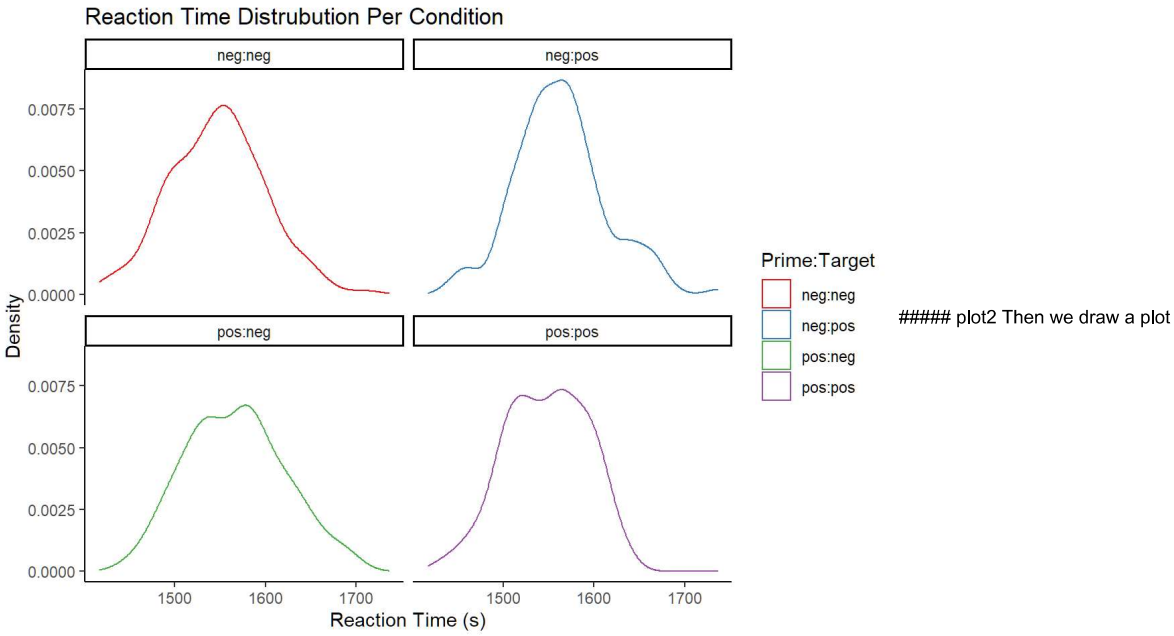
Data Visualisation

plot1 density plot

Here, we use geom_density to draw the density plot to see the distribution of the data and use:

- `facet_wrap` : to separate the plot
- `coord_flip` : to change the direction of the plot

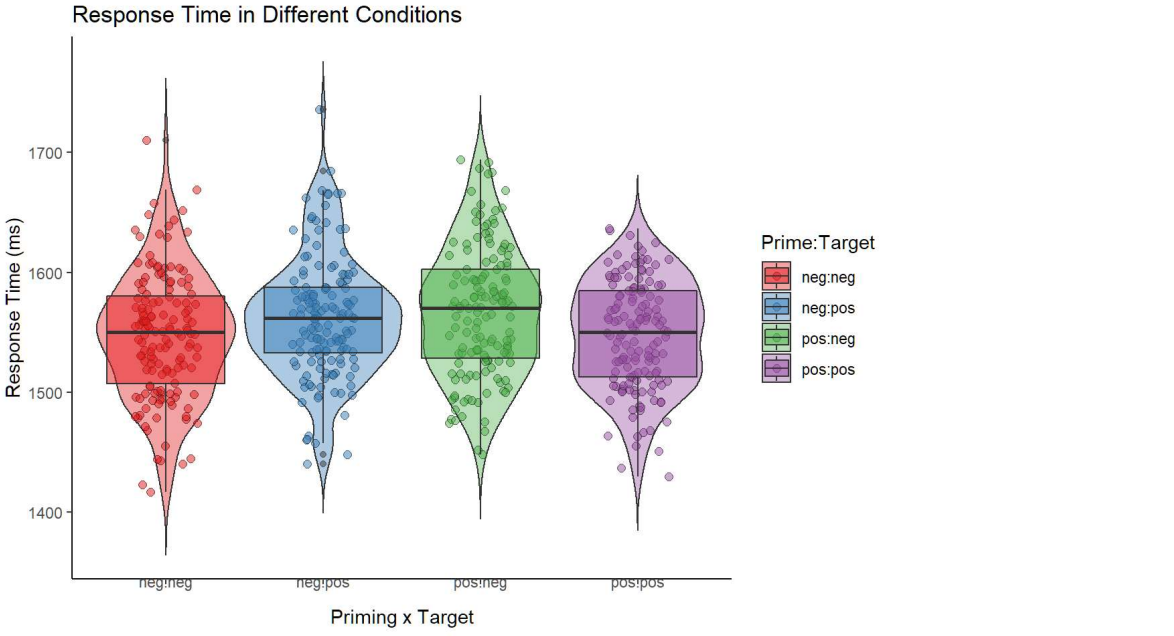
```
plot3_1 <- Q3_tidied %>%
  ggplot(aes(y = RT, colour = Prime:Target)) +
  geom_density(alpha = 0.4, linetype = 1) +
  coord_flip() +
  scale_colour_brewer(palette = "Set1") +
  facet_wrap(~Prime:Target)+
  theme_classic() +
  labs(title = "Reaction Time Distrubution Per Condition",
       y = "Reaction Time (s)",
       x = "Density")

plot3_1
```



plot2 Then we draw a plot

including violin, boxplot and scatter plot.

```
set.seed(123)

plot3_2 <- Q3_tidied %>%
  ggplot(aes(x = Prime:Target, y = RT)) +
  geom_violin(aes(fill = Prime:Target),
              trim = FALSE,
              width = 1,
              alpha = 0.4) +
  geom_jitter(aes(fill = Prime:Target),
              shape = 21,
              width = .2,
              alpha = 0.5,
              size = 2) +
  geom_boxplot(aes(fill = Prime:Target), alpha = .5) +
  scale_fill_brewer(palette = "Set1") +
  guides(colour = 'none') +
  theme_classic() +
  theme(axis.text.x = element_text(angle = 0, vjust = 5, hjust = .5)) +
  labs(title = "Response Time in Different Conditions",
       x = "Priming x Target",
       y = "Response Time (ms)")

plot3_2
```



Build Factorial ANOVA Model

We construct Factorial Model with aov_4 and generate the output using the anova() function.

```
model3 <- aov_4(RT ~ Prime * Target + (1 + Prime * Target | participant), data = Q3_tidied, na.rm = TRUE)

anova(model3)
```

```
## Anova Table (Type 3 tests)
##
## Response: RT
##              num Df den Df    MSE       F       ges    Pr(>F)
## Prime             1    147 2337.5  0.3157 0.0004908    0.5751
## Target            1    147 2730.5  0.2375 0.0004315    0.6267
## Prime:Target      1    147 2626.3 17.2541 0.0292770 5.523e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(model3)
```

```
##
## Univariate Type III Repeated-Measures ANOVA Assuming Sphericity
##
##                  Sum Sq num Df Error SS den Df    F value    Pr(>F)
## (Intercept)  1433445600      1   371410    147 5.6734e+05 < 2.2e-16 ***
## Prime              738      1   343608    147 3.1570e-01    0.5751
## Target             649      1   401386    147 2.3750e-01    0.6267
## Prime:Target     45315      1   386068    147 1.7254e+01 5.523e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We could know that there is interaction effect between to conditions. To further test our question, we use `emmeans()` to get the post-hoc comparisons.

```
emmeans(model3, pairwise ~ Prime * Target, adjust = "none")
```

```
## $emmeans
##  Prime Target emmean    SE  df lower.CL upper.CL
##  neg   neg      1547 4.304 147     1539     1556
##  pos   neg      1567 4.439 147     1558     1576
##  neg   pos      1563 4.151 147     1554     1571
##  pos   pos      1547 3.689 147     1540     1555
##
## Confidence level used: 0.95
##
## $contrasts
##  contrast          estimate   SE  df t.ratio p.value
##  neg neg - pos neg  -19.731 6.29 147  -3.138  0.0021
##  neg neg - neg pos  -15.405 5.98 147  -2.577  0.0110
##  neg neg - pos pos   -0.139 5.62 147  -0.025  0.9802
##  pos neg - neg pos    4.326 6.08 147   0.712  0.4776
##  pos neg - pos pos   19.591 6.05 147   3.236  0.0015
##  neg pos - pos pos   15.265 5.25 147   2.908  0.0042
```

The main contrasts we focus are the *neg neg - pos neg* and the *pos pos - neg pos*. Cause we compare for two times, so we need to use adjust p-value, which is 2 * original p-value (based on the Bonferroni correction). As a result, the adjust p-value are: 0.004 and 0.008 separately.

In summary, we could conclude that: We conducted a 2 (priming valence: Positive vs. Negative) x 2 (Target valence: Positive vs. Negative) repeated measures ANOVA to investigate the influence of priming valence on reaction times to following target of Positive or Negative valence. The ANOVA revealed no effect of priming valence (F < 1), no effect of target valence (F < 1), but an interaction between the valence of priming and target (F(1, 147) = 17.254, p = < 0.001, ηG2 = .029).

Bonferroni corrected post-hoc comparisons revealed that people responded faster to positive images following a positive prime (relative to following a negative prime) (1,547 ms. vs. 1,563 ms., t(147) = -2.908, *P* = 0.008), and faster to negative images following a negative prime (relative to following a positive prime) (1,547 ms. vs. 1567 ms., t(147) = -3.138, *P* = 0.004).

In general, we could also give audience a link to our some materials, including lab website, other supplementary materials, video description or any related information.

Here, I recommend you to use Youtube (http://youtube.com), you could find anything you like. (just for fun, haha)

Finally, I would like to finish with a meme to myself.

(an image caption Source: network)

Thanks for your patience and effort!