# rworkshop

*Helio, Giedre and Amy*

*27/11/2019*

## Data manipulation, visualisation and some tests

In this session you will learn to simulate, manipulate, analyse and visualise data.

We will be working with a pupilometry experiment on pupil adaptation. In this experiment people were shown a bright or a dark screen and the changes in the size of their pupil were measured using an eyetracker.

The basic idea is pupildiam = brightness (bright vs dark)

In the second experiment, they were shown words with bright or dark meaning and the words were presented with either bright or dark colors pupildiam = wordtype * brightness

### Required software

First we need to install all the packages that we need. These packages will have a lot of useful premade functionalities.

to install a package you write on your script, notebook or command line something like:

```
install.packages("nameOfThepackage") # you can easily find the name of the package
#on goolgle by searching a known function
```

After you install the package, you then need to load it. This will make sure the package is active to be used. In other words, you can have a package installed but if it is not loaded, you cannot use it.

After you install a package, it gets saved in our library of package, so to load a package you simply type

```
library("nameOfThePackage") # make use of the auto completion
```

## tip to run a line inside a script or inside a chunk on a notebook/markdown, simply place

the cursor on the line you want to run and use CTRL+ENTER (Windows) or CMD + ENTER (MAC)

Okay, so let's install and load the packages that we will use

```
# install packages
install.packages("lme4")                  # paclage for running multilevl mixed models,
#more about tat later
install.packages('afex')                  # useful for anovas and other methods
install.packages("broom.mixed")           # useful for maing tidy outputs
install.packages("devtools") #
devtools::install_github("debruine/faux") # this is how you install packages that are
#not yet publiched in CRAN but that are
#under development
install.packages("tidyverse")             # data wrangling and visualisation

# load required packages
```

```r
library("lme4")         # model specification / estimation
library("afex")         # anova and deriving p-values from lmer
library("broom.mixed") # extracting data from model fits
library("faux")         # data simulation
#NOTE: to install the 'faux' package, use:

library("tidyverse")    # data wrangling and visualisation
# ensure this script returns the same results on each run
set.seed(8675309)
```

we will start by simulating some data, the simulation part won't be discussed here, but those of you who are interested in data simulation can go over this at home. - For instance try to add a third condition in which the brightness level is a mean of both bright and dark and then run everything we do here; - try to reduce the number of subjects to 10? what happens with your analysis? - What happens if you have 100 subjects, instead ?

For now, simply run the script below to create a function that will simulate our data and then generate the data

```r
### Data simulation function
set.seed(42) # this makes sure your simulations give you the same result all of the time

my_sim_data <- function(nsubj  = 20, # number of subjects
                        nitem  = c(dark = 20, bright = 20),  # number of items
                        b0     = 3, # grand mean
                        b1     =  +.8, # effect of category
                        I0i_sd =  .4, # by-item random intercept sd
                        S0s_sd = .4, # by-subject random intercept sd
                        # S1s_sd =  40, # by-subject random slope sd
                        scor   = 0.3, # correlation between intercept and slope
                        err_sd = .6  # residual (standard deviation)
                        ) {
  # simulate items
    items <- faux::sim_design(
    between = list(brighteness = c("dark", "bright")),
    n = nitem,
    sd = I0i_sd,
    dv = "I0i",
    id = "item_id",
    plot = FALSE
  )

  # effect code category
  items$brighteness <- recode(items$brighteness, "dark" = +.5, "bright" = -.5)

  # simulate a sample of subjects
  subjects <- faux::sim_design(
    within = list(effect = c(S0s = "By-subject random intercepts")),
                              # S1s = "By-subject random slopes")),
    n = nsubj,
    sd = c(S0s_sd), #, S1s_sd)
    r = scor,
    id = "subj_id",
    plot = FALSE
  )
```

```
  # simulate trials
  dat_sim <- crossing(subj_id = subjects$subj_id,
                      item_id = items$item_id) %>%
    inner_join(subjects, "subj_id") %>%
    inner_join(items, "item_id") %>%
    mutate(err = rnorm(nrow(.), mean = 0, sd = err_sd)) %>%
    mutate(Pupdiam = b0 + I0i + S0s + (b1) * brighteness + err) %>%
    select(subj_id, item_id, brighteness, Pupdiam)

  dat_sim
}

# run the code below to simulate the data
dat_sim<- my_sim_data()
# dat_sim2 <- my_sim_data(nsubj = 1000)
```

It is a good idea to have a look at the data and get a sense of what it is there. In programs like excel you can simply look at the spreadsheet.

```
# View(dat_sim)

# now run the lines below and try to explain what it is telling you, you can also use
#?str on the command to get a hint
str(dat_sim)
head(dat_sim)
```

## Let's do a quick visualisation

and let's re-code the brightness categories. In pseudo code the line below reads: create a column called brightness cat in the dat_sim dataset, and to this column assign bright if the column brightness has a value of -.5, otherwise assign the word 'dark'

```
dat_sim$brighteness_cat<- if_else(dat_sim$brighteness == -.5, 'bright', 'dark')

ggplot(dat_sim, aes(x = brighteness_cat, y = Pupdiam))+
  geom_jitter(alpha = .3)+
  stat_summary(geom = 'pointrange', fun.data = 'mean_se')
  # geom_boxplot()
```

okay, we've now collected the data, what do we do with it?

A couple of things: 1 - aggregate it and summarise (if you want to run analysis like anova, t tests) 2 - make sure variable types are correctly specified 3 - run the analysis

Let's start with summarising, dplyr is a good way to do this in a 'tidy' manner we have already loaded tidyverse, so there is no need to load dplyr separately, as dplyr is part of the tidyverve universe

```
# okay, first in the experiment we asigned -.5 and +.5 for bright and dark condions,
#can we cerate a new column for the categories
dat_sim_mutated<- dat_sim%>% #%>% the pipe allows to access an object
  group_by(subj_id, brighteness_cat) %>%
  mutate(condition = if_else(brighteness == .5, 'Dark', 'Bright')) #mutate allows as
#to 'mjtate' the data frame, for instance
#if we want to create a new column
```

```
# usually we have multiple trials per participant and category, but we usually need
#to have a mean per condition and participant,
#so let's do that
dat_sim_aggreated<- dat_sim_mutated%>%
  group_by(subj_id, condition)%>%
  summarise_at(c('Pupdiam'), mean, na.rm = TRUE)
  # summarise_at(c('Pupdiam'), sum, na.rm = TRUE) if we needed a sum instead

# let's visualise it
ggplot(dat_sim_aggreated, aes(x = condition, y = Pupdiam, colour = condition))+
  stat_summary(geom = 'pointrange', colour = 'gray30')+
  geom_jitter(width = .1, aplha = .3) #it's a good idea to show the variability in the data


# did you know that the proportion of scientistis who a colour blind are is highter
#than the population? so let's make our
#colour scheme more friendly
a <- ggplot(dat_sim_aggreated, aes(x = condition, y = Pupdiam, colour = condition,))+
  stat_summary(geom = 'pointrange', colour = 'gray30')+
  geom_jitter(width = .1, aplha = .3)

a<- a+ theme_light() + scale_color_brewer(palette = 'Dark2')
a

# did all the particioants show eatcly the same level of change? we can link observations by particiant
b <- ggplot(dat_sim_aggreated, aes(x = condition, y = Pupdiam, colour = condition, group = 'subj_id'))+
  stat_summary(geom = 'pointrange', colour = 'gray30')+
  geom_jitter(width = .1, aplha = .3)+
  geom_line(aes(group = subj_id), colour = "grey70")

b<- b+ theme_light() + scale_color_brewer(palette = 'Dark2')
b
```

## let's run some tests

starting with an anova on this dataset (technically we can do a t-test but in this case it the same)

Remember what are the assumptions of the anova (or all linear models)? - normality - heteroscedacity - what else?

we can test and plot for assumptions

```
library(afex # load the afex package if we haven't already done that
# we can check if our dv is normally distributted

ggplot(dat_sim_aggreated, aes(x = Pupdiam))+
  geom_density()

# is it normal? roughly? why that shape?

ggplot(dat_sim_aggreated, aes(x = Pupdiam, colour = condition))+
  geom_density()
```

```r
# normality of residuals
normcheck<- lm(Pupdiam ~ condition, data = dat_sim_aggreated)

plot(normcheck)
# what do this plots tell us?

qqnorm(dat_sim_aggreated$Pupdiam)
```

### okay, now let's run the test

While we are at it, what are your hypothesis, and what do you expect the test to tell us?

```r
tests<- list() # it is good practice to store our stuff neatly, a list helps do that,
#as it an store multiple stuff, ideally
#pt related stuff in the same list. So we create and empty list where we will store our tests

# try to fill in the missing parts (solutions are provided all the eay to the buttom of this markdown)
# tests$a <- afex::aov_ez(id = '', dv = '',
#                         within = '',
#                         between = ,
#                         data = )

tests$a
summary(tests$a)

# emmeans::emmeans(tests$a, ~ condition)

# afex::test_levene(tests$a) we don't need this for within designs tho
```

## a few questions for you

okay answer, what is the difference between an anova and a linear regression? - try to run a linear regression for the same variables we used in the anova above?? the function for linear regression is 'lm' (try running

```r
?lm
```

in the command to get an idea of how to use it (solution provided in the end) - Comment on the output, hint, we have already ran the linear regression in this session

### can you run a correlation between condition contrast and pupil diameter

sample of code below:

```r
#cor.test(dat_sim_aggreated$condition_con, dat_sim_aggreated$Pupdiam, method=c("pearson", "kendall", "s
#decide which method to use and justify why
# cor.test(dat_sim_aggreated$condition_con, dat_sim_aggreated$Pupdiam, method=c("spearman"))
```

Where have you seen this (or something close that this) value before?

## let's work with Experiment 2 now

We will first start by simulating an interaction, again, if you are curious you can try to read an play with this simulation function on your own later, for now, just run the script below

```
### Data simulation function

my_sim_data <- function(nsubj  = 20, # number of subjects
                        nitem  = 100,  # number of items ; c(dark = 20, bright = 20)
                        b0     = 3, # grand mean
                        b1     =  +.6, # effect of brighteness
                        b2     = +.6, # effect of word content
                        b3   =.6,
                        I0i_sd =  .3, # by-item random intercept sd
                        S0s_sd = .2, # by-subject random intercept sd
                        # S1s_sd =  40, # by-subject random slope sd
                        scor   = 0.3, # correlation between intercept and slope
                        err_sd = .4  # residual (standard deviation)
                        ) {
  # simulate items brightness_word = c('dark_darkword', 'bright_darkword', 'dark_brightword', 'bright_b
    items <- faux::sim_design(
    between = list(word = c("darkword", "brightword"),
                   brightness_word = c('dark_darkword', 'bright_darkword', 'dark_brightword', 'bright_
    within = list(brighteness = c("dark", "bright")),
    n = nitem,
    sd = I0i_sd,
    dv = list(pupil = c("dark", 'bright')), #"I0i"
    id = "item_id",
    plot = TRUE
  )
   items2 <-gather(items, brightenesscond, I0i, 4:5, factor_key=TRUE)
   # items2 <- gather(items, value = "I0i", bright, dark)
       # Sepal.Length, Sepal.Width, Petal.Length, Petal.Width)

    # items2<- gather(items, key = 'item_id', )
  # effect code category
  # items$brighteness_cont <- recode(items$brighteness, "dark" = +.6, "bright" = -.6)
  items2$word_cont <- recode(items2$word, "darkword" = +.4, "brightword" = -.4)
  items2$bright_con <- if_else(items2$brightenesscond == "dark",  +.1, -.1)
  items2$bright_word_cont <- recode(items2$brightness_word, "dark.darkword" = +.6, "bright.darkword" = +
                                    "dark.brightword" = -.7, "bright.brightword" = -.4)

  # simulate a sample of subjects
  subjects <- faux::sim_design(
    within = list(effect = c(S0s = "By-subject random intercepts")),
                             # S1s = "By-subject random slopes")),
    n = nsubj,
    sd = c(S0s_sd), #, S1s_sd)
    r = scor,
    id = "subj_id",
    plot = FALSE
  )

  # simulate trials
```

```
  dat_sim <- crossing(subj_id = subjects$subj_id,
                      item_id = items$item_id) %>%
    inner_join(subjects, "subj_id") %>%
    inner_join(items2, "item_id") %>%
    mutate(err = rnorm(nrow(.), mean = 0, sd = err_sd)) %>%
    # mutate(Pupdiam = b0 + I0i + S0s + (b1) * brighteness_cont +  (b2)*word_cont + (b3)* brightness_wo
    mutate(Pupdiam =  b0 + I0i + S0s + (b1) * (bright_con) + (b2)*word_cont + (b3*b1*b2) *(bright_word_
    select(subj_id, item_id, word, word_cont, bright_con, Pupdiam, brightenesscond, brightness_word)

  dat_sim
}

# my_sim_data()
dat_sim_int<- my_sim_data()
# dat_sim2 <- my_sim_data(nsubj = 1000)
```

## let's name our variables properly

```
dat_sim_int$wordtype<- if_else(dat_sim_int$brighteness == "darkword", "darkword",
                               if_else(dat_sim_int$brighteness == "brightword", "brightword", NULL))
dat_sim_int$brighteness<- if_else(dat_sim_int$brighteness == "dark", "dark",
                               if_else(dat_sim_int$brighteness == "bright", "bright", NULL))

ggplot(dat_sim_int, aes(word, Pupdiam, colour = brightenesscond))+
  # geom_smooth(method = 'lm')
  stat_summary(aes(geom = 'pointrange', fun.data = 'mean'))

# okay, first in the experiment we asigned -.5 and +.5 for bright and dark conditions, can we cerate a
dat_sim_int_mutated<- dat_sim_int%>% #%>% the pipe allows to access an object
  group_by(subj_id, brighteness, word, brighteness_cont, word_cont) %>%
  mutate(color_type = if_else(brighteness == .5, 'Dark', 'Bright'))%>% #mutate allows as to 'mjtate' th
  #if we want to create a new column
  mutate(word_content = if_else(word == 0.2, 'Darkness', 'Brightness'))

# usually we have multiple trials per participant and category, but we usually need to have a mean per
#so let's do that
dat_sim_int_aggreated<- dat_sim_int_mutated%>%
  group_by(subj_id, word, brighteness)%>%
  summarise_at(c('Pupdiam'), mean, na.rm = TRUE)
  # summarise_at(c('Pupdiam'), sum, na.rm = TRUE) if we needed a sum instead

# let's have a peak
 ggplot(dat_sim_int_aggreated, aes(x = brighteness, y = Pupdiam, color = word))+ #group = 'subj_id'
  stat_summary(geom = 'pointrange')
  # geom_jitter(width = .1, aplha = .3)+
  # geom_line(aes(group = subj_id, colour =word))
```

**Now, using https://ggplot2.tidyverse.org and Google search, explore how you can make plots using ggplot2 that can be included in practical reports and publications.**

## additional exercises

1. • run an t-test on dat_sim with - that contains only one factor
2. • run an anova for the dat_sim_int that contains an interaction
3. • report those results in APA format

```
# 1. - run an t-test on dat_sim with that as only one factor
?t.test #use this to get help on how to run


# 2. 2. - run an anova for the dat_sim_int that contains and interaction, a slightly differnt way to ru

c<- afex::aov_ez(id = 'subj_id', data = dat_sim_int_aggreated,
             dv = 'Pupdiam',
             within = c('word', 'brightness'))

emmeans::emmeans(c, ~ word* brighteness)
```

#SOLUTIONS

## Anova

## solution

afex::aov_ez(id = 'subj_id', dv = 'Pupdiam', within = 'condition', between = NULL, data = dat_sim_aggreated)

## Linear regression

dat_sim_aggreated$condition_con < -if_else(dat_sim_aggreated$condition == 'Bright', -.5, +.5) tests$b < -lm(Pupdiam condition_con, data = dat_sim_aggreated)summary(tests$b)

For the very interested, a more rigorous way to conduct our analysis is by not aggregating trials and conditions, and instead fitting a linear mixed model predicting every single data point collected as a response variable.

We will walk you through an example below

run a liner mixed model on the dat_sim_int, example provided

```
# first let's load the package that allow us to use mixed models

require(lmerTest)
```

The linear mixed model is composed of two main parts, Fixed effects and random effects? Does anyone remember what those are from your stats class? any ideas?

in simple terms fixed effects are usually the conditions we are interested in in our experiments, the ones that we usually manipulate our are interested in comparing. What is this for our pupilometry study?

But now try to think, what other effects might determine how the pupil change in the experiment, regardless of what we are manipulating?

Those are usually random effects. In Experimental Psychology and neuroscience, those tend to be effects of participant? why? some participants will have bigger pupils than others or simply have different physiological reactivity to light, so some will be really responsive whereas others wont, this might also have to do with how attend each participant is during the experiment. When we average our data in anovas, we assume that all participants have the same degree of change? but is that a good assumption?

Another random effects that we experimenters want to control for are usually item or trial effects. Some items may generate higher responses than others, ideally we want this to vary randomly, but if that variability is great, and unaccounted for, it might be hard for out experiment conditions to reveal any effects that might be there (how is this situation called in your stats textbooks? hint, it is an error)

```
colnames(dat_sim_int)
tests$c<- lmer(Pupdiam ~ brightenesscond + word + (1 | item_id) + (1| subj_id),
               data = dat_sim_int, REML = FALSE) #always remeber
#to put REML to FALSE

# let's see the results
summary(tests$c)

# Now run
tests$d<- lmer(Pupdiam ~ brightenesscond * word + (1 | item_id) + (1| subj_id),
               data = dat_sim_int, REML = FALSE) #always remeber to
#put REML to FALSE

summary(tests$d)

# What is the difference betwen this and the previous model?

# now run this
anova(tests$c, tests$d)

# Take a shot at interpreting what all of this mean
```