R WORKSHOP

RESEARCH METHODS

Rémi de Fleurian - r.defleurian@qmul.ac.uk
George Fazekas - g.fazekas@qmul.ac.uk
From materials by Marcus Pearce, Sarah Sauvé, and Daniel Müllensiefen

Queen Mary University of London

INTRODUCTION

WHAT IS R?

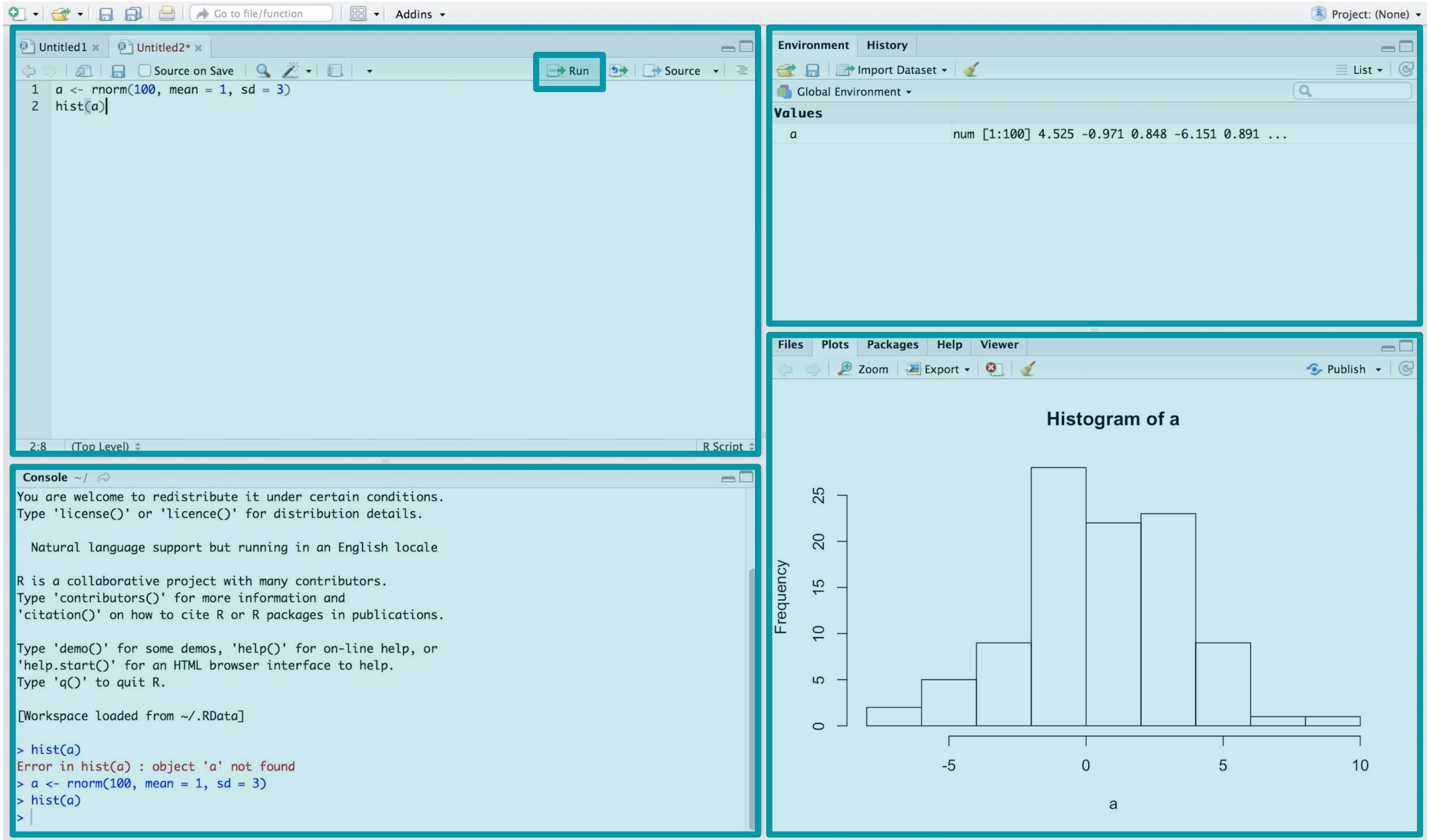
- Programming language for statistical computing and graphics
- Great alternative to SPSS, MATLAB, and many others
- Commonly used within the RStudio IDE
- Scripts vastly improve research workflow
- Extensive documentation (Wikibooks, and much more)
- 12,000+ packages (statistics, graphics, machine learning, etc.)

RSTUDIO

USER INTERFACE

Scripts and run code

Data and objects



Console and output

Plots, packages, and help

R SYNTAX

DATA

- R objects

```
x <- 2
```

- Create data

```
d <- c(8, 3, 5, 2, 3)
```

- Generate data

```
x.random <- runif(30, min = 0, max = 10)
x.normal <- rnorm(30, mean = 10, sd = 3)
```

- Import data
    - use “Import Dataset” in RStudio, or:

```
data <- read.csv("~/Desktop/R data.csv")
```

INDEXING

- Create vector

```
v <- c(9, 3, 5, 6, 2)
```

- Third element

```
v[3]
```

- Create matrix

```
m <- matrix(c(1:30), nrow = 6, ncol = 5, byrow = FALSE)
```

- Second column

```
m[,2]
```

- Fourth line

```
m[4,]
```

- Rows 2-4 of columns 1-3

```
m[2:4,1:3]
```

- Second column of data frame

```
data$Test_2
```

BASIC MATHS

- `1 + 2 * 3`
- `2 * 5^2 - 10 * 5`
- `4 * sin(pi / 2)`

^ or **	power
*, /	multiplication, division
+, -	adding, subtracting
%/%	division without remainder
%%	modulo division
max(), min()	extreme values
abs()	absolute value
sqrt()	square root
round(), floor(), ceiling()	rounding functions
sum(), prod()	sum, product
log(), log10(), log2()	logarithms
exp()	exponential function
sin(), cos(), tan(), asin(), acos(), atan()	trigonometric functions
pi	the number π
Inf, -Inf	infinity
NaN	not a number
NA	(not available)
NULL	empty set

FOR LOOPS, IF...ELSE STATEMENTS

- for loop

```
for (year in c(2010:2018)){
   print(paste("The year is ", year, ".", sep = ""))
}
```

- if...else statement

```
x <- -5
if(x >= 0){
   print("Positive number")
} else {
   print("Negative number")
}
```

HELP

- If you know the function

```
?mean
help(mean)
```

- If you don’t know the function

```
help.search('t-test')
```

- R documentation
- Wikibooks
- Stack Overflow

BASIC STATISTICS

DATA SET

- Populate a data frame

```
data <- data.frame(
    gender = rbinom(150, 1, 0.5),
    age = runif(150, min = 21, max = 55),
    fav.colour = rep(c("blue", "red", "green"), 50),
    test.before = rnorm(150, mean = 46, sd = 15),
    test.during = rnorm(150, mean = 62, sd = 13),
    test.after = rnorm(150, mean = 60, sd = 16)
)
```

DATA EXPLORATION

- Explore head and tail

```
head(data)
tail(data)
```

- Basic descriptive statistics

```
summary(data)
sd(data$test.before)
```

- More advanced descriptive statistics, with `moments` package

```
skewness(data$test.before)
kurtosis(data$test.before)
```

- Histogram

```
hist(data$test.before)
```

COMPARING DISTRIBUTIONS

- Useful (but more advanced) bit of code, using `ggplot2` package

```
d <- data.frame(score = c(data$test.before,
                          data$test.during,
                          data$test.after),
                treatment=rep(c("before", "during", "after"),
                              c(length(data$test.before),
                                length(data$test.during),
                                length(data$test.after))))
ggplot(d) +
  geom_density(aes(x=score, colour=treatment, fill=treatment),
               alpha=0.2)
```

- Data visualisation will be discussed in more details later

PARAMETRIC ASSUMPTIONS

- Normality (Shapiro-Wilk)

```
shapiro.test(data$age)
shapiro.test(data$test.before)
```

Careful! The null hypothesis is that the data is normally distributed. In other words, a statistically significant result means that the data is <u>not</u> normally distributed, and therefore that non-parametric tests should be used.

- Homogeneity of variance (for two-sample t-tests and ANOVAs)

```
var.test(data$age, data$test.before)
var.test(data$test.before, data$test.after)
```

Careful! Similarly, the null hypothesis is that the variances are homogenous.

INDEPENDENCE

- To measure association between **categorical independent variables** and **categorical dependant variables**

- Crosstabs

```
tab <- xtabs(~gender + fav.colour, data = data)
ftable(tab)
```

- Parametric test (Chi-squared)

```
chisq.test(tab)
```

- Non-parametric test (Kruskal-Wallis)

```
kruskal.test(gender ~ fav.colour, data = data)
```

CORRELATION

- To measure association between **a continuous independent variable** and **a continuous dependant variable**

- Add a new variable to the dataframe

```
new.test <- (data$test.after +
            runif(length(data$test.after), min = -10, max = 10))
data <- cbind(data, new.test)
```

- Scatterplots are great to visualise correlation

```
plot(data$test.before, data$test.after)
```

CORRELATION

- Titles and axis labels can easily be added

```
plot(data$test.after, data$new.test,
    main = "Plot for scores on test after treatment and new test",
    xlab = "Scores after treatment",
    ylab = "Scores on new test")
```

- As well as a regression line

```
abline(lm(data$test.after ~ data$new.test))
```

CORRELATION

- Parametric correlation test (Pearson)

```
cor.test(data$test.before, data$test.after, method="pearson")
cor.test(data$test.after, data$new.test, method="pearson")
```

- Non-parametric correlation test (Spearman)

```
cor.test(data$test.before, data$test.after, method="spearman")
cor.test(data$test.after, data$new.test, method="spearman")
```

- Interpreting *r*, the correlation coefficient
  - A positive value for *r* means a positive correlation, and vice versa
  - An absolute value close to 0 for *r* means a low degree of correlation
  - An absolute value close to 1 for *r* means a high degree of correlation

T-TEST

- To measure association between **a categorical independent variable with two levels** and **a continuous dependant variable**

- Parametric tests
  - One-sample t-test

```
t.test(data$test.before, mu = 46)
t.test(data$test.before, mu = 50)
```

  - Two-sample t-test

```
t.test(data$test.before, data$test.during)
t.test(data$test.during, data$test.after)
```

  - Paired-sample t-test (more powerful)

```
t.test(data$test.during, data$test.after, paired = TRUE)
```

- Non-parametric tests (Mann-Whitney U or Wilcoxon)
  - Simply replace `t.test` by `wilcox.test`

ANOVA

ANOVA

- To measure association between **categorical independent variables with more than two levels** and **a continuous dependant variable**

- General format

```
aov(DV ~ IVs, data = your.data)
```

- Independent variables
  - One-way ANOVA - for one independent variable

    ```
    test.score ~ treatment
    ```

  - Two-way ANOVA - for several IVs, including interaction effects

    ```
    test.score ~ treatment * gender
    ```

- Choice of non-parametric alternative to ANOVA depends on specific experimental design (Kruskal-Wallis, Friedman, etc.)

BETWEEN-SUBJECTS ONE-WAY ANOVA

- Run the ANOVA

```
aov1 <- aov(test.before ~ fav.colour, data = data)
summary(aov1)
```

- Show the means

```
model.tables(aov1, "means")
```

- Plot the means (`plot` can also be used instead of `plotmeans`)

```
library(gplots)
plotmeans(data$test.before ~ data$fav.colour,
      xlab="Favourite colour",
      ylab="Test scores before treatment",
      main="Mean plot with 95% CI")
```

- For plots that look better, use the `ggplot2` package! Instructions here.

BETWEEN-SUBJECTS TWO-WAY ANOVA

- Ensure gender is a factor, so it's not treated as a continuous variable

```
data$gender <- factor(data$gender)
```

- Run the ANOVA

```
aov2 <- aov(new.test ~ gender * fav.colour, data=data)
summary(aov2)
```

- Show the means

```
model.tables(aov2, "means")
```

- Post-hoc test for multiple comparisons of means (Tukey's HSD)

```
TukeyHSD(aov2)
```

- Plot the means (the DV is the last argument in `interaction.plot`)

```
interaction.plot(data$gender, data$fav.colour, data$test.after,
                 trace.label="Gender",
                 xlab="Favourite colour",
                 ylab="Test scores after treatment",
                 main="Interaction plot")
```

WITHIN-SUBJECTS ANOVAS

- The data needs some reformatting for within-subject ANOVAS
  - Add column for participant number in the data frame

    ```
    data <- cbind(participant = c(1:150), data)
    ```

  - Convert the data to the long format, using the `tidyr` package

    ```
    library(tidyr)
    data.long <- gather(data, treatment, test.score,
                        test.before:new.test)
    ```

- `gather` function in `tidyr` library

`data.long`	name of new data frame
`data`	name of old data frame
`treatment`	chosen name for categorical variable
`test.score`	chosen name for continuous variable
`test.before:new.test`	first and last columns to gather

WITHIN-SUBJECTS ONE-WAY ANOVA

- Ensure new categorical variables are not treated as continuous variables

```
data.long$participant <- factor(data.long$participant)
data.long$gender <- factor(data.long$gender)
```

- Run the ANOVA

```
aov3 <- aov(test.score ~ treatment + Error(participant/treatment),
            data = data.long)
summary(aov3)
```

- Show the means

```
model.tables(aov3, "means")
```

- Plot the means

```
library(gplots)
plotmeans(data$test.before ~ data$fav.colour,
      xlab="Favourite colour",
      ylab="Test scores before treatment",
      main="Mean plot with 95% CI")
```

MIXED-DESIGN ANOVA

- Run the ANOVA

```
aov4 <- aov(test.score ~ gender * treatment + Error(participant/treatment),
          data = data.long)
```

- Show the means

```
model.tables(aov4, "means")
```

- Plot the means

```
interaction.plot(data.long$treatment, data.long$gender, data.long$test.score,
               trace.label="Gender",
               xlab="Treatment",
               ylab="Test scores",
               main="Interaction plot")
```

- For complex designs, the `ezANOVA` function offers a more legible syntax

```
library(ez)
ezANOVA(data = data.long,
     dv = .(test.score),
     wid = .(participant),
     within = .(treatment),
     between = .(gender, fav.colour))
```

MODELLING

MODELLING

- To predict the value of a **dependent variable** based on the values of one or more **independent variables**

- A statistical model is a mathematical equation that is used to approximate the behaviour of the studied data, and to make predictions from this approximation

- The **parameters** of the model correspond to the coefficients of the equation, and the **residuals** (or errors) are the distances between the data points and the model

- Some simple, widely used forms of modelling are **linear regression**, **multiple regression**, and **logistic regression**, but many other approaches exist in the field of **machine learning**

MODELLING

- Currently, the two most popular languages for modelling are **Python** and **R**, due to the active development of relevant packages, and to the size of their respective machine learning communities

- Statistical modelling is a large and very active field of study, and can't be covered in a single workshop (or even in a single module)

- Some QMUL modules provide a good introduction to the topic:
  - CPD course on Statistics and R (6 half-days, starting May 2018)
  - ECS764P - Applied Statistics
  - ECS759P - Artificial Intelligence
  - ECS784P - Data Analytics
  - ECS708P - Machine Learning
  - MTH786P - Machine Learning with Python
  - ECS792P - Music and Speech Modelling

MODELLING

- There are also excellent online resources:
    - Andrew Ng's Machine Learning course on Coursera, for an excellent series of videos to introduce the reasoning and maths behind machine learning, though the practical aspects of the course are in MATLAB rather than Python or R
    - Kaggle, a machine learning competition platform, with good notebook-based introductions to Python, R, visualisation, machine learning, and deep learning
    - Google's crash course, for another solid introduction to machine learning and deep learning (though there are some prerequisites)
    - fast.ai, for a practice-based approach to becoming more familiar with deep learning
    - Andrew Ng's Deep Learning courses on Coursera, for a more theoretical look at neural networks and deep learning

MODELLING

- What follows is a very basic implementation of linear regression in R (for more details about this specific approach, see here)

- First, populate a data frame with data about alligator weight and length

```
alligator = data.frame(
  length = c(3.87, 3.61, 4.33, 3.43, 3.81, 3.83, 3.46, 3.76,
             3.50, 3.58, 4.19, 3.78, 3.71, 3.73, 3.78),
  weight = c(4.87, 3.93, 6.46, 3.33, 4.38, 4.70, 3.50, 4.50,
             3.58, 3.64, 5.90, 4.43, 4.38, 4.42, 4.25))
```

- Then, plot the relationship between the two variables

```
plot(alligator$length, alligator$weight,
     xlab = "Length",
     ylab = "Weight",
     main = "Alligators")
```

MODELLING

- The plot suggests a linear relationship, so a linear model can be fitted

```
model = lm(weight ~ length, data = alligator)
summary(model)
```

- The `Estimate` column lists the coefficients. In this case, a slope of 3.4311, and an intercept of -8.4761, which translates into the following model:

```
weight = 3.4311*length - 8.4761
```

- The `Pr(>|t|)` column lists the p-value, and show that in this case, the coefficients are significantly different to 0

MODELLING

- The residuals can be explored in a few ways to check whether the model captures most of the information present in the data, which is assumed to be the case when the residuals are normally distributed

- To do so, the residuals can be plotted in comparison to a reference line...

```
plot(resid(model) ~ fitted(model),
    xlab = "Fitted Values",
    ylab = "Residuals",
    main = "Residual Diagnostic Plot")
abline(h = 0)
```

- ... visualised with a histogram or a QQ plot, or simply checked with a test

```
hist(resid(model))
qqnorm(resid(model))
shapiro.test(resid(model))
```

MODELLING

- Finally, a model can be used to predict values for the dependent variable based on the values of the independent variables

- To do so, generate some new data

```
new.data <- data.frame(length = c(3.27, 3.81, 4.32))
```

- And use the `predict` command to make predictions on the new data, using the previously fitted model

```
predict(model, new.data)
```