Using R to analyse a simple survey

Nathaniel Phillips

24 March 2017

The survey

For these examples, we'll load the following data representing psychology study:

id	sex	age	condition	response	accuracy	time
1	m	25	A	34	31	173
2	m	23	B	32	10	197
3	f	26	C	85	32	463
4	m	27	A	46	54	291
5	m	21	B	37	36	139
6	m	27	C	65	16	272

Installing packages

To use functions or data from a package, like `yarrr` or `BayesFactor`, you first must install the package from the internet on your computer with `install.packages()`. Note, you must be connected to the internet to install the package, but you only need to do this once.

```
# Install some packages from the internet (you only need to install packages to your computeronce)

install.packages("yarrr")          # For R piratery
install.packages("BayesFactor")    # For Bayesian analyses
install.packages("papaja")         # For APA style results
install.packages("rprojroot")      # For working directory handling
install.packages("dplyr")          # For data wrangling
```

Loading packages

To use a package you've installed, you must always load it with `library()`

```
# Load packages for my analyses. You must always load all packages you plan to use!

library("yarrr")          # For R piratery
library("BayesFactor")    # For Bayesian analyses
library("papaja")         # For APA style results
library("rprojroot")      # For working directory handling
library("dplyr")          # For data wrangling
```

Loading data

If you have a .txt file that you want to read into R, use the `read.table()` function.

Argument	Description
`file`	The document's file path relative to the working directory unless specified otherwise. For example `file = "mydata.txt"` looks for the text file directly in the working directory, while `file = "data/mydata.txt"` will look for the file in an existing folder called `data` inside the working directory.If the file is outside of your working directory, you can also specify a full file path (`file = "/Users/CaptainJack/Desktop/OctoberStudy/mydata.txt"`)
`header`	A logical value indicating whether the data has a header row – that is, whether the first row of the data represents the column names.
`sep`	A string indicating how the columns are separated. For comma separated files, use `sep = ","`, for tab–delimited files, use `sep = "\t"`
`stringsAsFactors`	A logical value indicating whether or not to convert strings to factors. I **always** set this to FALSE because I *hate*, *hate*, *hate* how R uses factors

```
# Set working directory to directory containing an .Rproj file
setwd(rprojroot::is_rstudio_project$find_file())

# Example: Loading data from a text file called mydata.txt into a new object called x

study <- read.table(file = "data/study.txt", # The file is called mydata.txt and is in the working dire
                    sep = "\t",              # Columns in the data are separated by tabs
                    header = TRUE,           # There is a header row
                    stringsAsFactors = FALSE)   # Do not convert strings to Factors

# Note, if you do not have a copy of the study.txt data, you can also read it directly from the interne
study <- read.table(file = "https://raw.githubusercontent.com/ndphillips/PsyKo-2017/master/data/study.t
                    sep = "\t",              # Columns in the data are separated by tabs
                    header = TRUE,           # There is a header row
                    stringsAsFactors = FALSE)   # Do not convert strings to Factors
```

Exploring data

Function	Description
`head(study), tail(study)`	Print the first few rows (or last few rows).
`View(study)`	Open the entire object in a new window
`nrow(study), ncol(study), dim(study)`	Count the number of rows and columns
`rownames(), colnames(), names()`	Show the row (or column) names
`str(study), summary(study)`	Show the structure of the dataframe (ie., dimensions and classes) and summary statistics

```
head(study)    # Show me the first few rows of study
View(study)    # View the entire dataframe in a new window
str(study)     # Show me basic information about the dataframe
dim(study)     # How many rows and columsn are there?
```

Working with column names

To access columns in a dataframe by name, use the `$` operator. For example, `study$sex` will return the sex values in a dataframe called study

```
names(study)                      # Show me the names of the columns in study
study$age                         # Return the column age in the dataframe
study$age.months <- study$age * 12  # Add a new column age.months to the dataframe
names(study)[8] <- "months"     # Change the name of the 7th column to "months"
```

Descriptive statistics functions

Continuous, numeric data

```
sum(study$age)
min(study$response)
max(study$time)
mean(study$time)
median(study$response)
sd(study$time)
var(study$age)
summary(study$time)
```

Categorical data

```
unique(study$age)                       # What are all unique values of age?
table(study$condition, exclude = NULL)  # How many times did each condition value occur
```

Missing (NA) Values

If your data contains missing (NA) values, you may need to include additional arguments in your functions to prevent errors:

```
# include na.rm = TRUE to ignore missing values

mean(study$response, na.rm = TRUE)    # Show me the mean of response, but ignore missing (NA) values
max(study$time, na.rm = TRUE)         # Show me the median of time, but ignore NA values

table(study$sex, exclude = NULL)      # Show me a table of the values of sex, and include counts of NA v
```

Grouped aggregation

```
# Show me the mean age for each sex
aggregate(formula = age ~ sex,
          data = study,
          FUN = mean)


# Show me the median response for each condition, but only for males
```

```r
aggregate(formula = response ~ condition,
          data = study,
          subset = sex == "m",
          FUN = median)


# Show me the mean time for each condition and sex,
aggregate(formula = time ~ condition + sex,
          data = study,
          FUN = mean)
```

Plotting

Colors

```r
colors()                                    # See all of the named colors in R
yarrr::piratepal()                          # Show the yarrr color palettes
yarrr::piratepal("pony", plot.result = TRUE)  # Show the pony palette
```

Histograms

```r
# Histogram of responses
hist(study$response)

# Fancier histogram
hist(study$response,
     xlim = c(0, 100),                # x axis limits
     breaks = 20,                     # number of bins
     col = "dodgerblue",              # Color of bins
     border = "white",                # Bin border color
     main = "Response Distribution",
     xlab = "Response",
     yaxt = "n",                      # turn off y-axis
     ylab = "")
```

Scatterplots

```r
# Scatterplot of responses and time
plot(x = study$response,
     y = study$time)


# Fancier scatterplot of reponse and time
plot(x = study$response,
     y = study$time,
     xlab = "Response",
     ylab = "Time (milliseconds)",
     pch = 16,                        # Solid points
     col = gray(.5, .5),              # Transparent gray points
```

```
     xlim = c(0, 100))

grid()                                # Add gridlines

# Add a regression line
abline(lm(time ~ response,
          data = study),
       lty = 2)                      # Dashed line
```

Barplot

```
# Barplot of response by condition using papaja::apa_barplot
papaja::apa_barplot(data = study,
                    id = "id",
                    factors = c("condition"),
                    dv = "accuracy")

# Now including two IVs
papaja::apa_barplot(data = study,
                    id = "id",                          # Column indicating individual participants
                    factors = c("condition", "sex"),    # What are the IVs?
                    dv = "accuracy",                    # What is the DV
                    args_legend = list(x = "topleft"))  # Put legend on top left
```

pirateplots

```
# Pirateplot of response by condition
yarrr::pirateplot(response ~ condition,
                  data = study)

#  Theme 2
yarrr::pirateplot(response ~ condition,
                  data = study,
                  theme = 2,
                  back.col = "oldlace")

# Theme 3 and 2 IVs
yarrr::pirateplot(response ~ condition + sex,
                  data = study,
                  pal = "google",             # use the google color palette
                  theme = 3)
```

Hypothesis tests

t-tests

```
# One sample t-test comparing the mean response to 100
t.test(study$response,
       alternative = "two.sided",
```

```
       mu = 500)

# Two sample t-test comparing the mean response of males to females
t.test(response ~ sex,
       data = study,
       alternative = "two.sided")

# Two sample t-test comparing the mean time between condition A and B
t.test(time ~ condition,
       data = study,
       subset = condition %in% c("A", "B"),
       alternative = "two.sided")
```

Correlation test

```
# Correlation test between age and time
cor.test(formula = ~ age + time,
         data = study)

# Correlation test between response and time, but only for females
cor.test(formula = ~ response + time,
         data = study,
         subset = sex == "f")
```

ANOVA

```
# ANOVA comparing response by condition
response.aov <- aov(formula = response ~ condition,
                    data = study)

# Summary results
summary(response.aov)

# Post-hoc tests
TukeyHSD(response.aov)
```

Regression

```
# Regression analysis predicting response by age, sex and condition
response.lm <- lm(formula = response ~ age + sex + condition,
                  data = study)

# Summary results
summary(response.lm)
```

Bayesian Statistics

```
# Bayesian t-test comparing times of men and women
time.bf <- BayesFactor::ttestBF(formula = time ~ sex,
                                data = study)

time.bf       # Show results
plot(time.bf)

# Bayesian ANOVA comparing reponse by condition

study$condition.f <- factor(study$condition)  # Convert condition to a factor

response.bf <- BayesFactor::anovaBF(formula = response ~ condition.f,
                                    data = study)

response.bf       # Show results
plot(response.bf)

# Bayesian ANOVA comparing reponse by condition and age
study$age.f <- factor(study$age)           # Convert condition to a factor

response.B.bf <- BayesFactor::anovaBF(formula = response ~ condition.f + age.f,
                                      data = study)

response.B.bf       # Show results
plot(response.B.bf)
```

Simulations

What is the distribution of p-values given a specified mean and sd?

- If we take a sample size of size N from a normal distribution with mean `true.mean` and standard deviation `true.sd`, and conduct a one-sample t-test, what will the distribution of p-values be?

```
# Simulate 1000 p-values from a normal distribution with specified mean and standard deviation

# --------------------
# Determine simulation parameters
# --------------------

true.mean <- 0     # The true population mean
true.sd <- 1       # The true population standard dviation
N <- 100           # The number of samples

# --------------------
# Start pulation!
# --------------------

# Create a place-holder vector for the p-values
p <- rep(NA, 1000)

# Loop over 1000 simulations
```

```r
for(i in 1:1000) {

  # Generate random data
  data <- rnorm(n = N, mean = true.mean, sd = true.sd)

  # Calculate a one-sample t-test
  test <- t.test(data)

  # Get the p-value
  p.value <- test$p.value

  # Assign the p-value to the ith place in p
  p[i] <- p.value

}

# --------------------
# Show the results!
# --------------------

# What percent are less than or equal to .05?
mean(p < .05)

# Histogram of distribution of p-values
hist(p,
     xlim = c(0, 1),
     xlab = "p-value",
     main = paste("p(p-value < .05) =", round(mean(p < .050), 2)))

abline(v = .05, lty = 2)
```

Given that a p-value is 'significant' what is the probability that the null hypothesis is actually false?

- If a p-value is significant, what is the probability that the null hypothesis is actually false?

```r
# Simulate 100 p-values from a normal distribution with mean = 0

# --------------------
# Determine simulation parameters
# --------------------

n.H0 <- 500      # The number of cases where the null is TRUE
n.H1 <- 500      # The number of cases where the null is FALSE

N <- 100
mean.H0 <- 0    # The mean when the null is TRUE
mean.H1 <- .25  # The mean when the null is FALSE


# --------------------
# Start simulation!
# --------------------
```

```r
# Create a dataframe of parameters and a placeholder for the p-values
sim <- data.frame(mu = c(rep(mean.H0, n.H0), rep(mean.H1, n.H1)),
                  sd = rep(1, n.H0 + n.H1),
                  p = NA)

# Loop over 1000 simulations
for(i in 1:1000) {

  # Get the true mean and standard deviation for the ith simulation
  true.mu <- sim$mu[i]
  true.sd <- sim$sd[i]

  # Generate random data
  data <- rnorm(n = N, mean = true.mu, sd = true.sd)

  # Conduct a one-sample t-test
  test <- t.test(data)

  # Get the p-value
  p.value <- test$p.value

  # Assign the p-value to the ith place in the dataframe
  sim$p[i] <- p.value

}


# --------------------
# Show the results!
# --------------------

# What percent of H0
mean(sim$p[sim$mu == 0] < .05)
mean(sim$p[sim$mu != 0] < .05)

# Given a significant p-value, what is the probability that the null hypothesis is false?
with(subset(sim, p < .05), mean(mu != 0))

## Plotting

par(mar = c(5, 6, 4, 1) + .1)
plot(x = sim$p[sim$mu != 0],
     y = rep(1, n.H1) + rnorm(n.H1, mean = 0, sd = .1),
     ylim = c(.5, 2), col = "red", xlab = "p value", ylab = "", yaxt = "n", xlim = c(0, 1))

points(x = sim$p[sim$mu == 0],
       y = rep(1.5, n.H0) + rnorm(n.H0, mean =0, sd = .1), col = "blue")

mtext("Null = TRUE", side = 2, at = 1.5, las = 1, line = 1)
mtext("Null = FALSE", side = 2, at = 1, las = 1, line = 1)

abline(v = .05, lty = 2)
```

```
mtext(paste("p(Null is FALSE | p < .05) = ", round(with(subset(sim, p < .05), mean(mu != 0)), 2)), side
```