UPPSALA
UNIVERSITET

Uppsala Child & Baby Lab

Variables, operators, and data structures

R for Psychology Research

Marcus Lindskog, Docent
2019-09-02

Overview

1. Course info.
2. What is R?
3. Becoming familiar with RStudio.
4. Variables, operators, and data structures.

Course info

Lectures and Seminars

1. 10 weeks at 50% (7.5 hp)
2. 9 interactive lectures (no lecture w. 44)
3. 9 computer exercises (no session w.44)

Learning goals

> After completing this course, the student is expected to have gained sufficient knowledge of the R language to be able to use it to solve statistical problems relevant to psychology research.

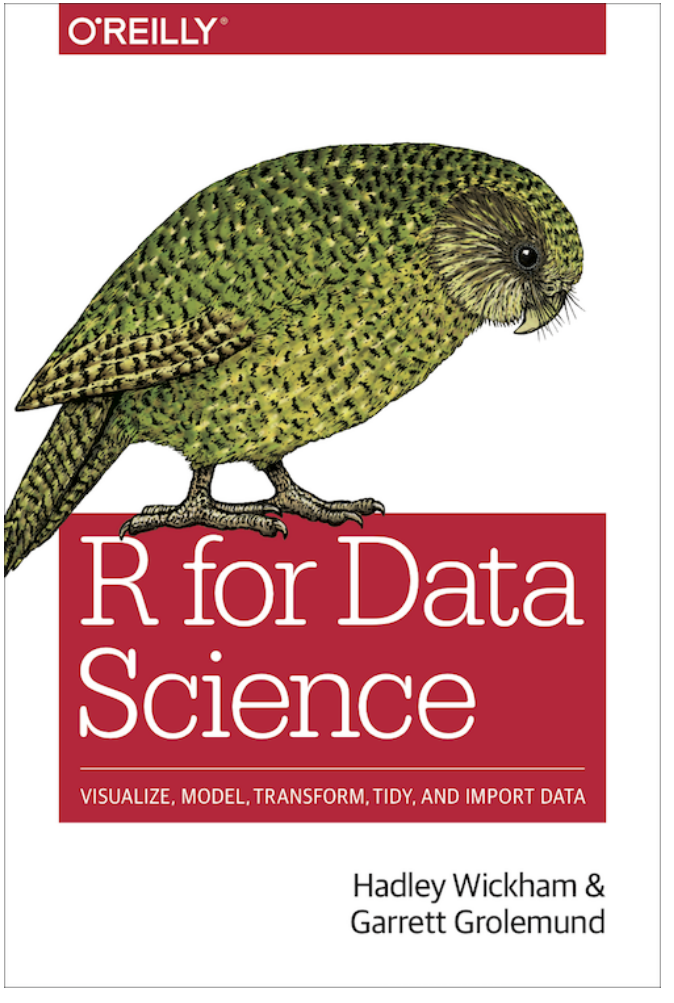
Examination

1. Tasks from each computer exercise that you will need to submit to me in the form of R-code.

2. A final examination in the form of a reproducible report, including statistical tests and visualizations of data, for a realistic data set, possibly from your own research.

Course materials and Examination

- Wickham, H. & Grolemund, G. (2017). *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data.* Sebastopol: O'Reilly. (https://r4ds.had.co.nz)
- All course material from me is available by email.
- Submit your examinations by e-mail to: marcus.lindskog@psyk.uu.se



What you can expect to pick up

- The basics of R.
- *Data wrangling* and *Graphing* in R.
- Often, for psychology researchers, used statistical tests.
- How to write and manage code to do the above.

What you **can not** expect to pick up

- When and why you should run a specific analysis.
- A quick and dirty introduction for your specific analyses.
- Knowledge of every possible package on CRAN.

What is R?

What is R?

- Statistical programming language.
- An implementation of the S programming language.
- An interpreted language.
- Developed by *R Development Core Team*

What is R?

- R is free.
- A major version comes out once a year.
- 2-3 minor releases each year.
- Anyone can contribute to R by writing a *package*.
- There are currently more than 15 000 packages available.

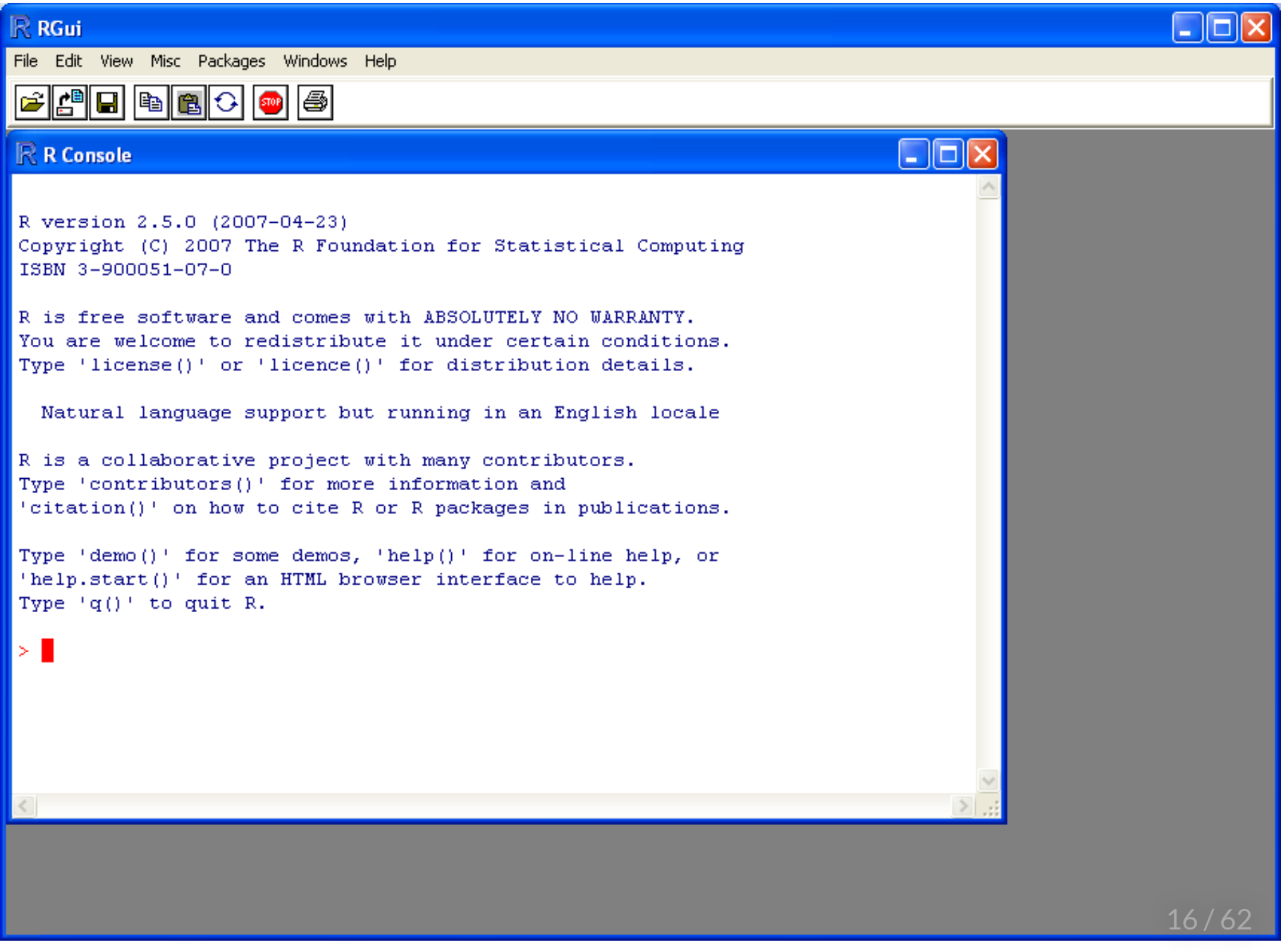
aRe we set up?

- R can be found at the *comprehensive R archive network* https://cran.r-project.org
- Rstudio can be found at https://www.rstudio.com/
- Make sure you keep R and Rstudio updated with the latest version.

Becoming familiar with RStudio

Say hello to Rstudio

- You can use R directly from its console.
- But, Rstudio makes your life much easier.
- RStudio is an integrated development environment (IDE).


RGui
File Edit View Misc Packages Windows Help
R Console
R version 2.5.0 (2007-04-23)
Copyright (C) 2007 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>

File Edit Code View Project Workspace Plots Tools Help

Go to file/function

Project: (None)

diamondPricing.R* | formatPlot.R | diamonds

Source on Save | Source

```
1  library(ggplot2)
2  source("plots/formatPlot.R")
3
4  View(diamonds)
5  summary(diamonds)
6
7  summary(diamonds$price)
8  aveSize <- round(mean(diamonds$carat), 4)
9  clarity <- levels(diamonds$clarity)
10
11 p <- qplot(carat, price,
12            data=diamonds, color=clarity,
13            xlab="Carat", ylab="Price",
14            main="Diamond Pricing")
15
```

15:1 (Top Level) R Script

Console ~/

```
       x                y                z
 Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
 1st Qu.: 4.710   1st Qu.: 4.720   1st Qu.: 2.910
 Median : 5.700   Median : 5.710   Median : 3.530
 Mean   : 5.731   Mean   : 5.735   Mean   : 3.539
 3rd Qu.: 6.540   3rd Qu.: 6.540   3rd Qu.: 4.040
 Max.   :10.740   Max.   :58.900   Max.   :31.800
> summary(diamonds$price)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    326     950    2401    3933    5324   18820
> aveSize <- round(mean(diamonds$carat), 4)
> clarity <- levels(diamonds$clarity)
> p <- qplot(carat, price,
+            data=diamonds, color=clarity,
+            xlab="Carat", ylab="Price",
+            main="Diamond Pricing")
>
> format.plot(p, size=24)
```

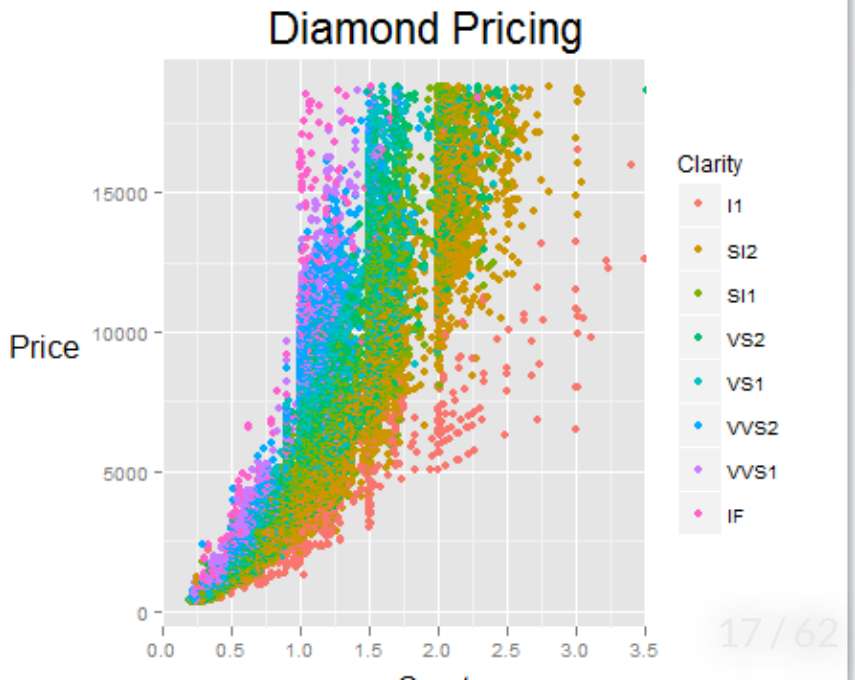
Workspace | History

Load | Save | Import Dataset | Clear All

Data	
diamonds	53940 obs. of 10 variables
Values	
aveSize	0.7979
clarity	character[8]
p	ggplot[8]
Functions	
format.plot(plot, size)	

Files | Plots | Packages | Help

Zoom | Export | Clear All



The console

- The console in R works just like a calculator.
- Anything you type after > will be evaluated.
- R uses the # sign for comments. Anything you add after # will not run as R code.
- Commenting your code using # is **very** important!

Variables, operators, and data structures

Arithmetic operators in R

- Addition: +
- Subtraction: -
- Multiplication: *
- Division: /
- Exponentiation: `^` or `**`
- Modulo: %%

Let's try

```
# Addition
6 + 8

# Subtraction
9 - 15

# Multiplication
10 * 4
```

Let's try

```
# Division
10 / 4

# Exponentiation
3^2

3**2

#Modulo
5 %% 2
```

Variable assignment

- R let's you store values (e.g. 4) and objects (e.g. a function) in variables.
- What you have store can be accessed using the variables name.
- A value (or object) is assigned to a variable using <-

```
# Assign 42 to the_answer
the_answer <- 42

# Access the value assigned to the_answer
the_answer
```

Variable operations

- Arithmetic operations works on variables.
- An answer can be assigned to new variables.

```
# Assign 42 to the_answer
the_answer <- 42

# Assign 15 to extra_everything
extra_everything <- 15

# Calculate sum of the_answer and extra_everything
the_answer + extra_everything

# Assign sum to new variable
new_truth <- the_answer + extra_everything
```

Variable assignment

- Variable assignment is not limited to integers.
- R allows you to assign various *data types* to variables.

```
# Assign 42 to the_answer
the_answer <- 42L

# Assign 42.76 to exact_answer
exact_answer <- 42.76

# Assign "question" to answer_to_what
answer_to_what <- "question"

# Assign FALSE to the_truth
the_truth <- FALSE
```

Data types in R

- character: `"a", "swc"`
- numeric: `2, 15.5`
- integer: `2L`
- logical: `TRUE, FALSE`
- (complex: `1+4i`)

What data type?

```
# Assign 42 to the_answer
class(the_answer)

# Assign 42.76 to exact_answer
class(exact_answer)

# Assign "question" to answer_to_what
class(answer_to_what)

# Assign FALSE to the_truth
class(the_truth)
```

Do we need to care?

```
# Difference between the_answer and exact_answer
exact_answer-the_answer

# Difference between the_answer and the_truth
the_truth-the_answer

# Difference between answer_to_what and exact_answer
answer_to_what - the_answer
```

Data structures

More than one data point?

- R let's you store collections of data points in various data structures.
- We will look at:
    - Vectors
    - Matrices
    - Data frames
    - Lists

Vectors

Vectors

- One dimensional collection of data.
- Can only contain data of the same type.
- Vectors are created with the `combine` fuction `c()`

```
numeric_vector <- c(8, 7, 6, 5, 4)

charcter_vector <- c("x", "y", "z", "t")

boolean_vector <- c(TRUE, FALSE, FALSE, TRUE)
```

Naming a vector

```r
# Your daily coffee consumption

coffee_cups <- c(6, 3, 4, 1, 10, 2, 4)

names(coffee_cups) <- c("Monday", "Tuesday", "Wednesday",
                        "Thursday", "Friday", "Saturday",
                        "Sunday")

coffee_cups
```

Getting lazy

```r
# Your daily coffee consumption

coffee_cups <- c(6, 3, 4, 1, 10, 2, 4)

week_days <- c("Monday", "Tuesday", "Wednesday",
               "Thursday", "Friday", "Saturday",
               "Sunday")

names(coffee_cups) <- week_days

coffee_cups
```

Arithmetic on vectors

```
# Your daily coffee consumption
coffee_cups_w34 <- c(6, 3, 4, 1, 10, 2, 4)
coffee_cups_w35 <- c(4, 7, 6, 9, 0, 8, 6)

names(coffee_cups_w34) <- week_days
names(coffee_cups_w35) <- week_days

two_week_cups <- coffee_cups_w34 + coffee_cups_w35

names(two_week_cups) <- week_days

two_week_cups
```

More arithmetics

```
#Total cups
sum(two_week_cups)

#Mean cups
mean(two_week_cups)

#Average daily cups
two_week_cups / 2
```

Logical operators

- Less than: <
- Less than or equal: <=
- Greater than: >
- Greater than or equal: >=
- Exactly qual to: ==
- Not equal to: !=
- Not y: !y
- x OR y: x|y
- x AND y: x&y

Logical operators

```
2 < 4
```

```
## [1] TRUE
```

```
3 > 5
```

```
## [1] FALSE
```

```
6 == 8
```

```
## [1] FALSE
```

```
"dog" == "dog"
```

```
## [1] TRUE
```

Logic on vectors

```
coffee_cups_w34 <- c(6, 3, 4, 1, 10, 2, 4)
coffee_cups_w35 <- c(4, 7, 6, 9, 0, 8, 6)

# Your daily coffee consumption
coffee_cups_w34 > coffee_cups_w35
```

```
## [1]  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE
```

```
coffee_cups_w34 == coffee_cups_w35
```

```
## [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
coffee_cups_w34 > 6
```

```
## [1] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
```

```
coffee_cups_w35 != 6
```

```
## [1]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE
```

Vector selection

```
# Your daily coffee consumption
monday_coffee <- coffee_cups_w34[1]

monday_coffee

weekend_coffee <- coffee_cups_w35[c(6,7)]
weekend_coffee
```

Vector selection using logic

```
# Your daily coffee consumption
high_volume <- coffee_cups_w34[coffee_cups_w34 >= 6]

high_volume

low_volume <- coffee_cups_w35[coffee_cups_w35 < 6]

low_volume
```

Matricies

Matricies

- Two dimensional (row, column) data structure.
- Can only contain one data type

Matrix assignment

```r
# Generate 4 x 5 numeric matrix
y<-matrix(1:20, nrow=4,ncol=5)

y

cell_values <- c(1,26,24,68, 32, 99)
row_names <- c("Row 1", "Row 2", "Row 3")
column_names <- c("Col 1", "Col 2")
my_matrix <- matrix(cell_values, nrow=3, ncol=2, byrow=TRUE,
  dimnames=list(row_names, column_names))

my_matrix
```

Matrix selection

```
# Generate 4 x 5 numeric matrix
y

# 4th column of matrix
y[,4]

# 3rd row of matrix
y[3,]

# rows 2,3,4 of columns 1,2,3
y[2:4,1:3]
```

Matrix selection

```
# Generate 4 x 5 numeric matrix
y

# What does this do?
y[y[,4] > 14,]
```

Data frames

Data frames

- Two dimensional data structure.
- Allows columns to be of different data types.
- Columns must be of same length.
- Similar to data sheet in standard statistical software.

Data frame assignment

```
var_1 <- c(49, 32, 24, 23, 43)
var_2 <- c("m", "f", "f", "m", "m")
var_3 <- c(102, 110, 123, 98, 112)
var_4 <- c(TRUE, TRUE, FALSE, FALSE, FALSE)

the_data <- data.frame(var_1, var_2, var_3, var_4)
```

Variable names

```
#Option 1
the_data <- data.frame(Age = var_1, Sex = var_2,
                       IQ = var_3, Bool = var_4)

#Option 2
names(the_data) <- c("Age", "Sex", "IQ", "Bool")
```

Access data

- There are three ways to access the data in a data frame.

1. Subscript: `the_data[3:4]`
2. Names: `the_data[c("Age","IQ")]`
3. dollar-notation: `the_data$Age`

```
#Option 1
the_data[,3:4]

#Option 2
the_data[c("Age","IQ")]

#Option 3
the_data$Age
```

Overview of the data

```
head(the_data, 2)

tail(the_data, 2)

str(the_data)

summary(the_data)
```

Subset data frame

- Using `subset()` you can select a portion of your data frame.
- `subset(my_df, subset = some_condition)`

```
subset(the_data, Age > 25)
```

Adding a variable

```
new_variable <- c(78, 98, 56, 22, 12)
the_data$New <- new_variable
str(the_data)
```

Removing a variable

```
#Option 1
the_data[,-c(2:3)]

# Option 2
subset(the_data, select = -c(Sex, IQ))

# Option 3
the_data[,!names(the_data) %in% c("Sex", "IQ")]
```

Lists

Lists

- Lists is the most flexible data structure in R
- It can basically contain anything.
- It helps us collect things in a structured way.

```
our_list <- list(matrix(3:14, nrow = 4), y, coffee_cups_w34)
our_list
```

Naming a list

```
names(our_list) <- c("M1", "M2", "Coffee")
our_list
```

Accessing list items 1

```
#Option 1
our_list[[1]]

our_list[[1]][2]

# Option 2
our_list$Coffee

our_list$Coffee[1]
```

Scripts

Scripts

- During this lecture we've written all our code in the console.
- This is not very practical in the long run.
- A better way is to collect all relevant code in a script.
- Let's have a look at how you do that in Rstudio.

That's all folks!