

Programming Fundamentals

Algorithms; Variable types; Syntax

Dr Josh Hodge

jhodge@ic.ac.uk

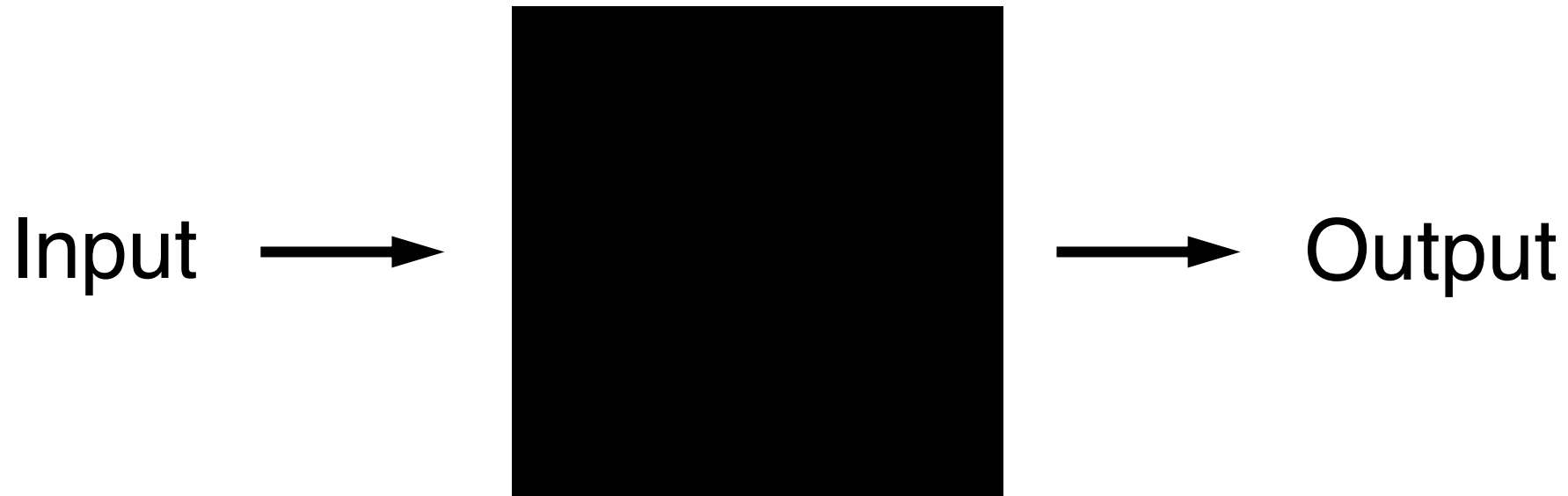
Intended Learning Outcomes

By end the day you will be able to:

- Define what an algorithm is
- Use pseudocode to help simplify programming
- Describe different data structures
- Identify different variable types
- Feel more comfortable with R and RStudio

Algorithms

“A process or set of rules to be followed to solve a problem or task.”



Examples of Algorithms



Directions

✓ Step 1

Preheat oven to 350 degrees F (175 degrees C). Grease and flour an 8-inch square pan.

✓ Step 2

In a large saucepan, melt 1/2 cup butter. Remove from heat, and stir in sugar, eggs, and 1 teaspoon vanilla. Beat in 1/3 cup cocoa, 1/2 cup flour, salt, and baking powder. Spread batter into prepared pan.

✓ Step 3

Bake in preheated oven for 25 to 30 minutes. Do not overcook.

✓ Step 4

To Make Frosting: Combine 3 tablespoons softened butter, 3 tablespoons cocoa, honey, 1 teaspoon vanilla extract, and 1 cup confectioners' sugar. Stir until smooth. Frost brownies while they are still warm.

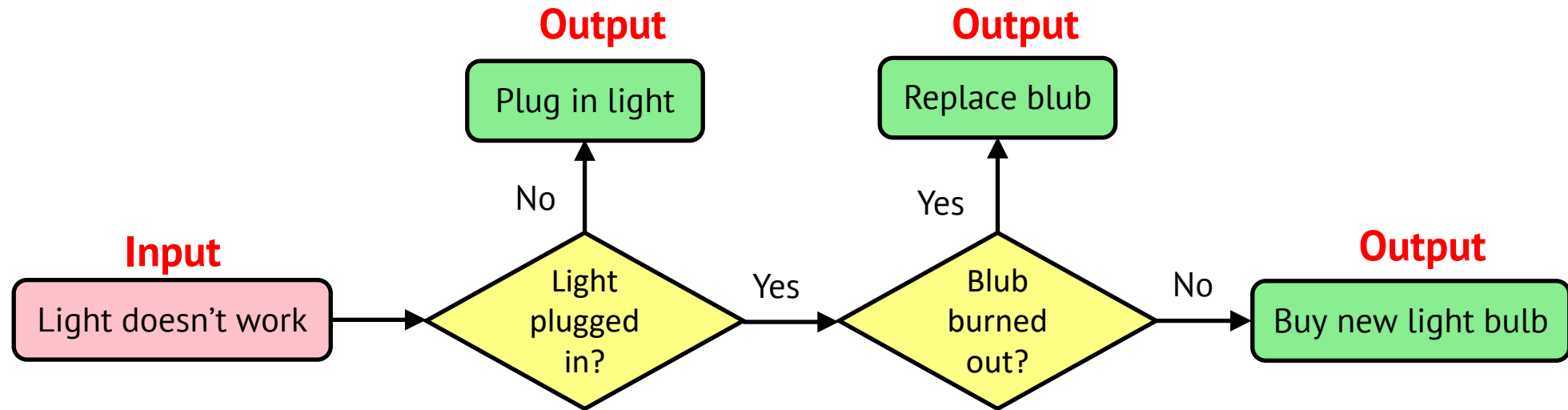


Chocolate Brownie



Pseudocode

- Plain English syntax of code
- Can be used to represent or write an algorithm
- Uses verbs, variables, conditional statements and loops to perform task (covered in more detail tomorrow)



Variables and Types

- Information is stored as variables

MyForename <- “Josh” **Character**

MyAge <- 31 **Numeric (integer or float)**

Iam31 <- TRUE **Logical**

Variables and Types



Variables with more information

```
MyFullname <- c("Josh", "Alexander", "Hodge-Grace")
```

Character Vector

```
MyDoB <- c(12, 11, 1989)
```

Numeric Vector

```
Iam31 <- c(TRUE, FALSE)
```

Logical Vector

Naming Variables



- R is case sensitive
- Keep variable names simple
- Keep column names simple

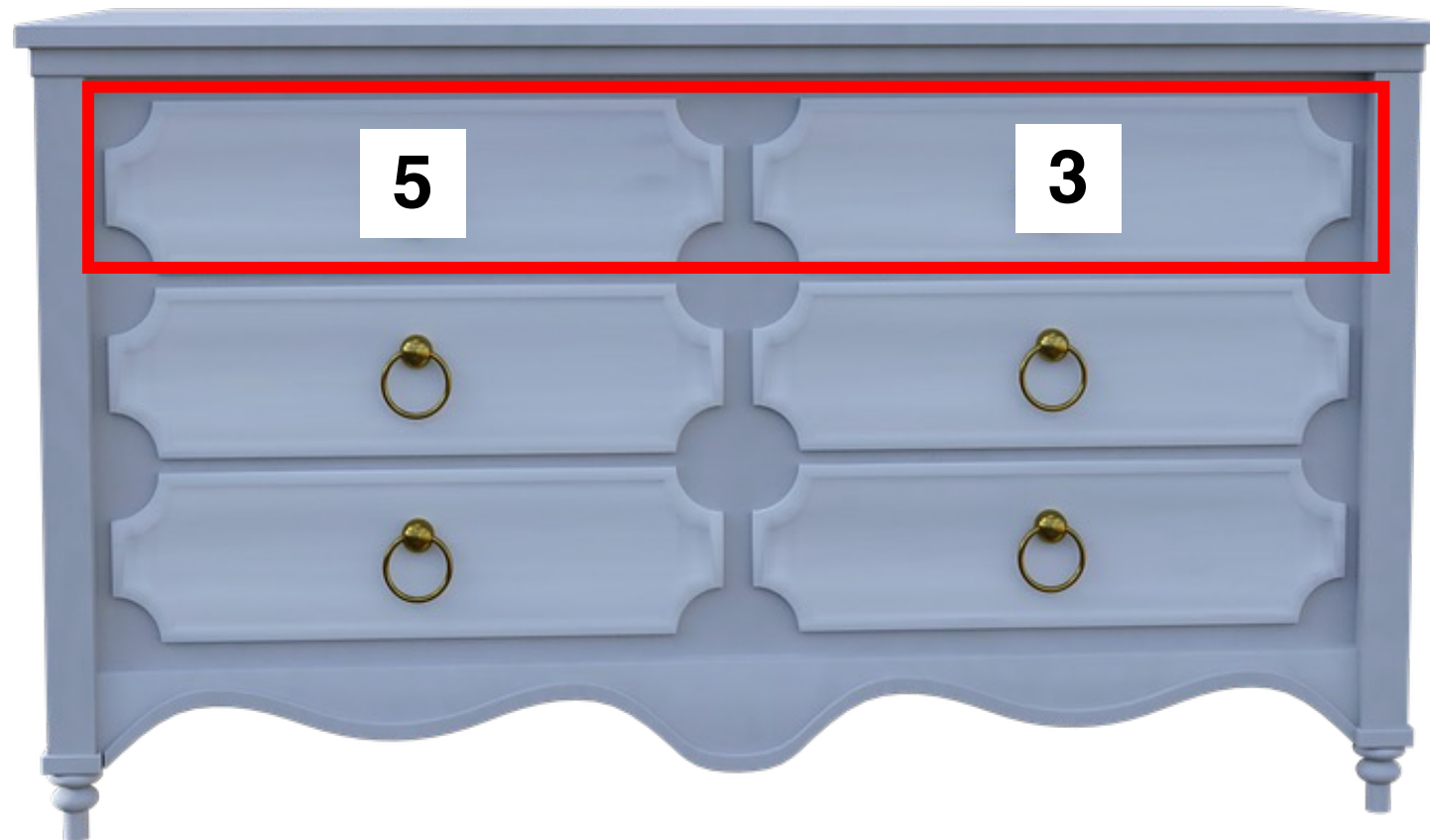
Data Structures

1. Vectors: one dimension of same data type
2. Matrices: two dimensions of the same data type
3. Arrays: n-dimensions of the same data type
4. Dataframes: two dimensions of mixed data types
5. Lists: n-dimensions of mixed data types

The Chest of Drawers of Vectors

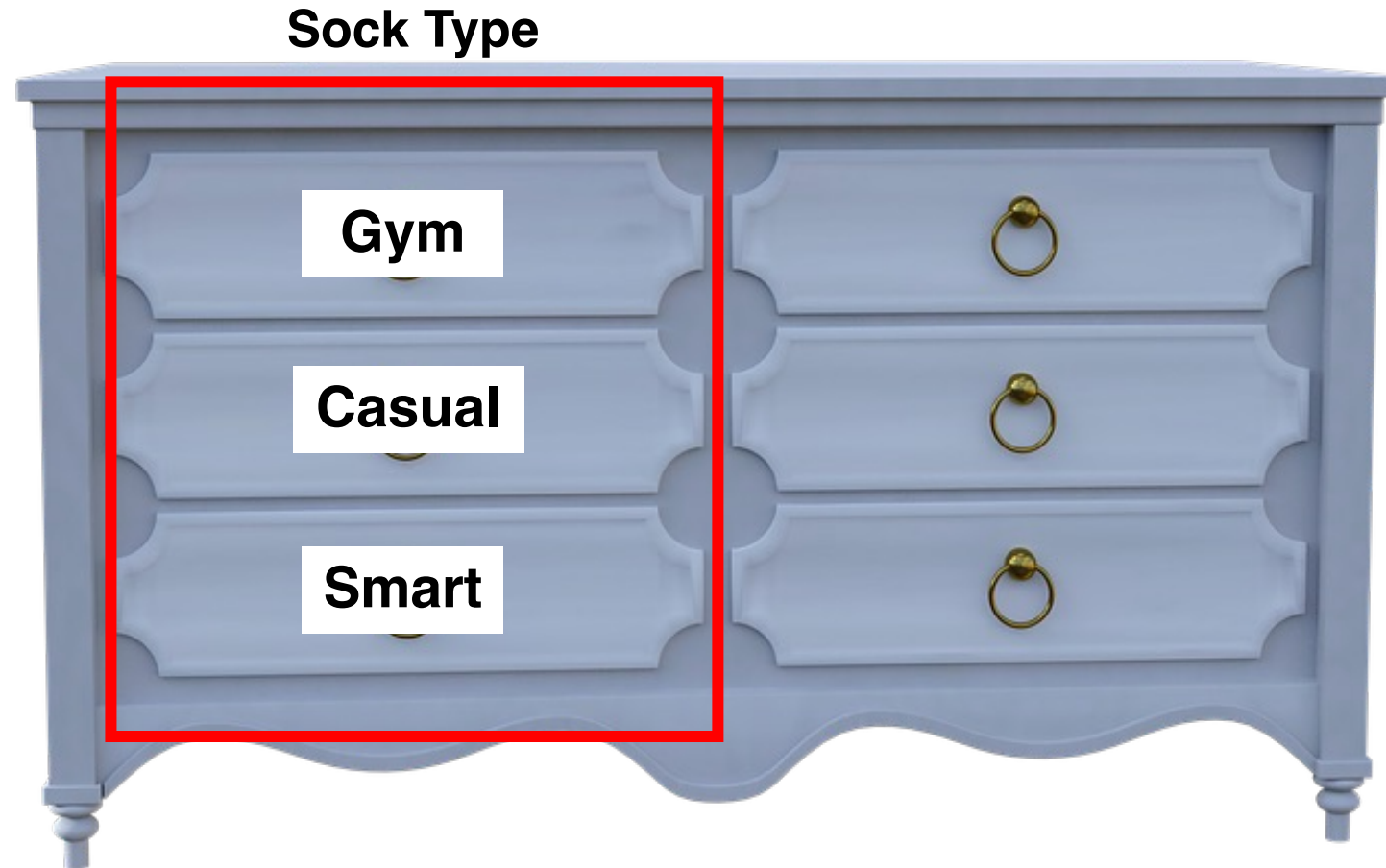
- Vectors: one dimension of same data type

Number of Socks



The Chest of Drawers of Vectors

- Vectors: one dimension of same data type



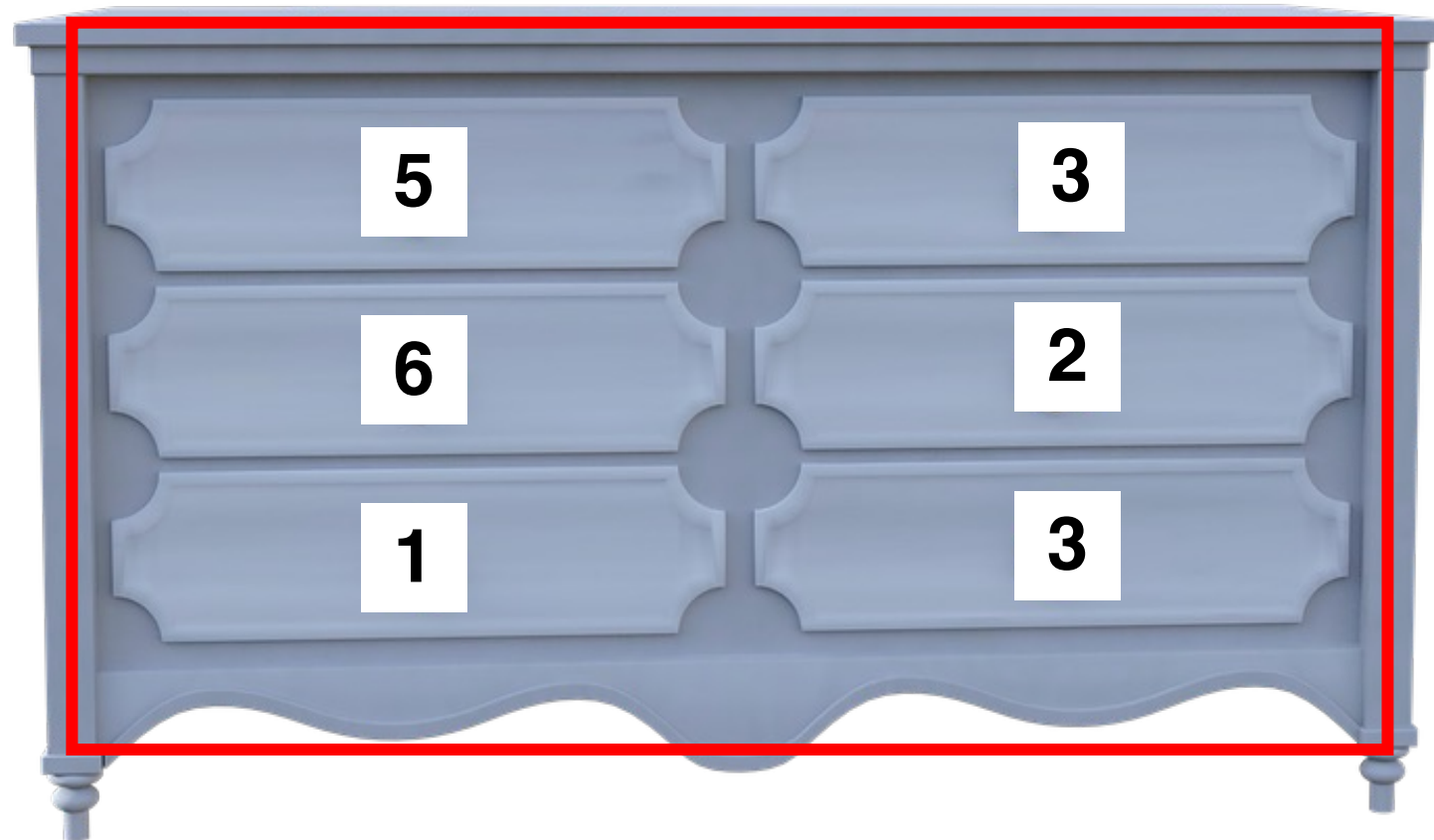
Data Structures

1. Vectors: one dimension of same data type
2. Matrices: two dimensions of the same data type
3. Arrays: n-dimensions of the same data type
4. Dataframes: two dimensions of mixed data types
5. Lists: n-dimensions of mixed data types

The Matrix of Drawers

- Matrix: two dimensions of the same data type

Number of Socks



5	3
6	2
1	3

Data Structures

1. Vectors: one dimension of same data type
2. Matrices: two dimensions of the same data type
3. Arrays: n-dimensions of the same data type
4. Dataframes: two dimensions of mixed data types
5. Lists: n-dimensions of mixed data types

The Array of Drawers

- Arrays: n-dimensions of the same data type

Number of Items

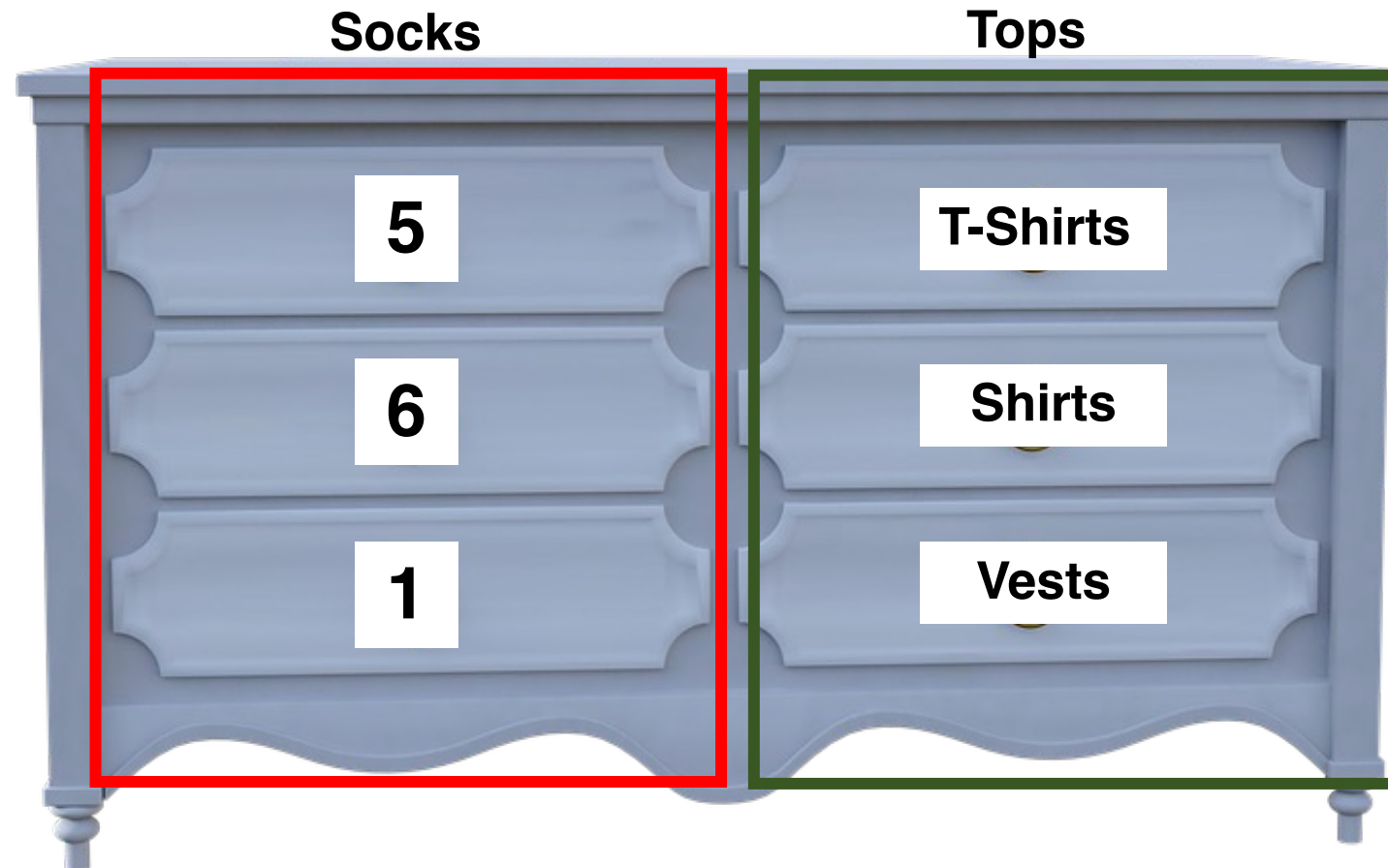


Data Structures

1. Vectors: one dimension of same data type
2. Matrices: two dimensions of the same data type
3. Arrays: n-dimensions of the same data type
4. Data frames: two dimensions of mixed data types
5. Lists: n-dimensions of mixed data types

Chest of Drawers as Dataframes




- Data Frames: two dimensions of mixed data types



Data Structures

1. Vectors: one dimension of same data type
2. Matrices: two dimensions of the same data type
3. Arrays: n-dimensions of the same data type
4. Dataframes: two dimensions of mixed data types
5. Lists: n-dimensions of mixed data types

Repositories, Packages and Functions

- A repository is a place where packages are located so you can install them.
 - The Comprehensive R Archive Network (CRAN) 
 - Bioconductor  **Bioconductor**
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS
 - Github 
- Packages are a collection of functions in a well-defined format
- Function is a piece of code that executes an action over an input

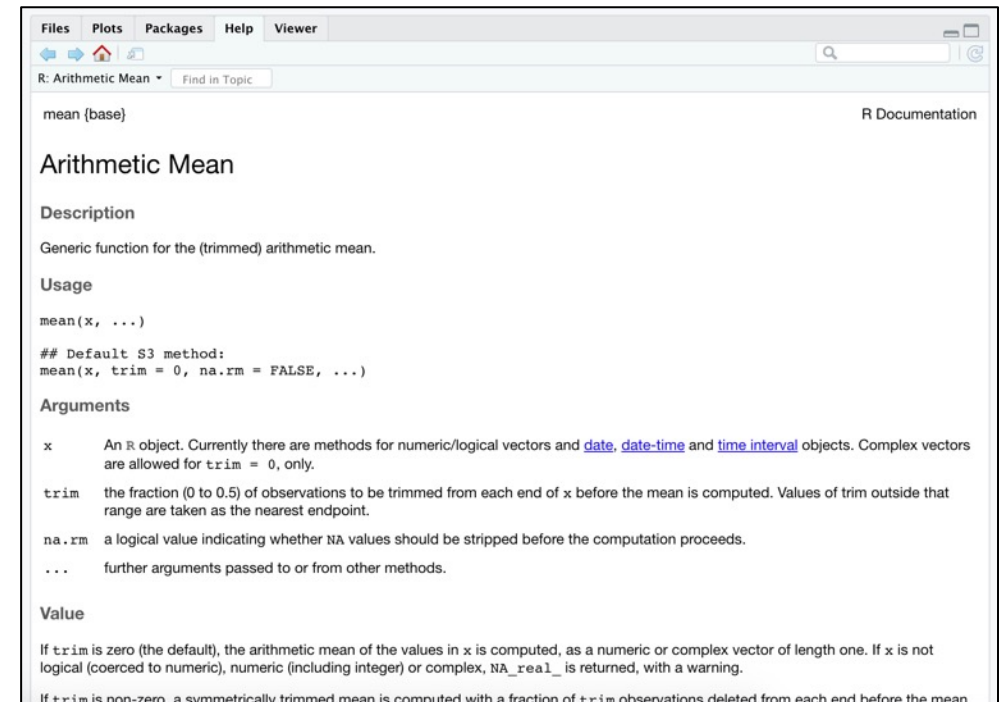
Building Blocks of a Function

function(arg1, arg2,...etc)

- Arguments are variables required for the function to be executed

```
mean(x)
x= numerical vector

vector <- c(1,2,3)
mean(vector)
2
```



R Syntax

- Syntax is the structure of statements in a computer language

R Syntax Comparison :: CHEAT SHEET

Dollar sign syntax

```
goal(data$x, data$y)
```

SUMMARY STATISTICS:

one continuous variable:

```
mean(mtcars$mpg)
```

one categorical variable:

```
table(mtcars$cyl)
```

two categorical variables:

```
table(mtcars$cyl, mtcars$am)
```

one continuous, one categorical:

```
mean(mtcars$mpg[mtcars$cyl==4])  
mean(mtcars$mpg[mtcars$cyl==6])  
mean(mtcars$mpg[mtcars$cyl==8])
```

PLOTTING:

one continuous variable:

```
hist(mtcars$disp)
```

```
boxplot(mtcars$disp)
```

one categorical variable:

```
barplot(table(mtcars$cyl))
```

two continuous variables:

```
plot(mtcars$disp, mtcars$mpg)
```

two categorical variables:

```
mosaicplot(table(mtcars$am, mtcars$cyl))
```

one continuous, one categorical:

```
histogram(mtcars$disp[mtcars$cyl==4])  
histogram(mtcars$disp[mtcars$cyl==6])  
histogram(mtcars$disp[mtcars$cyl==8])
```

```
boxplot(mtcars$disp[mtcars$cyl==4])  
boxplot(mtcars$disp[mtcars$cyl==6])  
boxplot(mtcars$disp[mtcars$cyl==8])
```

WRANGLING:

subsetting:

```
mtcars[mtcars$mpg>30, ]
```

making a new variable:

```
mtcars$efficient[mtcars$mpg>30] <- TRUE  
mtcars$efficient[mtcars$mpg<30] <- FALSE
```

Formula syntax

```
goal(y~x|z, data=data, group=w)
```

SUMMARY STATISTICS:

one continuous variable:

```
mosaic::mean(~mpg, data=mtcars)
```

one categorical variable:

```
mosaic::tally(~cyl, data=mtcars)
```

two categorical variables:

```
mosaic::tally(cyl~am, data=mtcars)
```

one continuous, one categorical:

```
mosaic::mean(mpg~cyl, data=mtcars)
```

tilde

PLOTTING:

one continuous variable:

```
lattice::histogram(~disp, data=mtcars)
```

```
lattice::bwplot(~disp, data=mtcars)
```

one categorical variable:

```
mosaic::bargraph(~cyl, data=mtcars)
```

two continuous variables:

```
lattice::xyplot(mpg~disp, data=mtcars)
```

two categorical variables:

```
mosaic::bargraph(~am, data=mtcars, group=cyl)
```

one continuous, one categorical:

```
lattice::histogram(~disp|cyl, data=mtcars)
```

```
lattice::bwplot(cyl~disp, data=mtcars)
```

The variety of R syntaxes give you many ways to “say” the same thing

read across the cheatsheet to see how different syntaxes approach the same problem

Tidyverse syntax

```
data %>% goal(x)
```

SUMMARY STATISTICS:

one continuous variable:

```
mtcars %>% dplyr::summarize(mean(mpg))
```

one categorical variable:

```
mtcars %>% dplyr::group_by(cyl) %>%  
dplyr::summarize(n())
```

two categorical variables:

```
mtcars %>% dplyr::group_by(cyl, am) %>%  
dplyr::summarize(n())
```

one continuous, one categorical:

```
mtcars %>% dplyr::group_by(cyl) %>%  
dplyr::summarize(mean(mpg))
```

PLOTTING:

one continuous variable:

```
ggplot2::qplot(x=mpg, data=mtcars, geom = "histogram")
```

```
ggplot2::qplot(y=disp, x=1, data=mtcars, geom="boxplot")
```

one categorical variable:

```
ggplot2::qplot(x=cyl, data=mtcars, geom="bar")
```

two continuous variables:

```
ggplot2::qplot(x=disp, y=mpg, data=mtcars, geom="point")
```

two categorical variables:

```
ggplot2::qplot(x=factor(cyl), data=mtcars, geom="bar") +  
facet_grid(~am)
```

one continuous, one categorical:

```
ggplot2::qplot(x=disp, data=mtcars, geom = "histogram") +  
facet_grid(~cyl)
```

```
ggplot2::qplot(y=disp, x=factor(cyl), data=mtcars,  
geom="boxplot")
```

WRANGLING:

subsetting:

```
mtcars %>% dplyr::filter(mpg>30)
```

making a new variable:

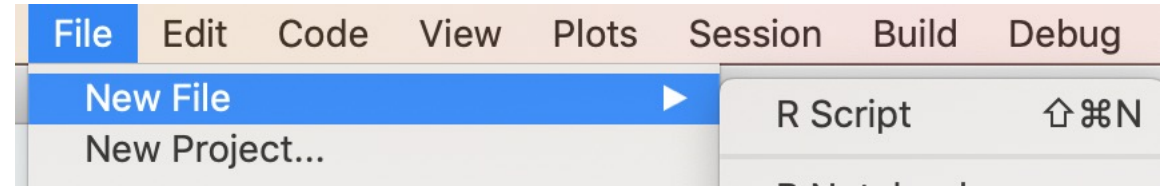
```
mtcars <- mtcars %>%  
dplyr::mutate(efficient = if_else(mpg>30, TRUE, FALSE))
```

the pipe

Rules of Scripting

1. Readability
2. Architecture First
3. Keep it Simple
4. Comments
5. Modularise and Develop Iteratively
6. Strive for Automation

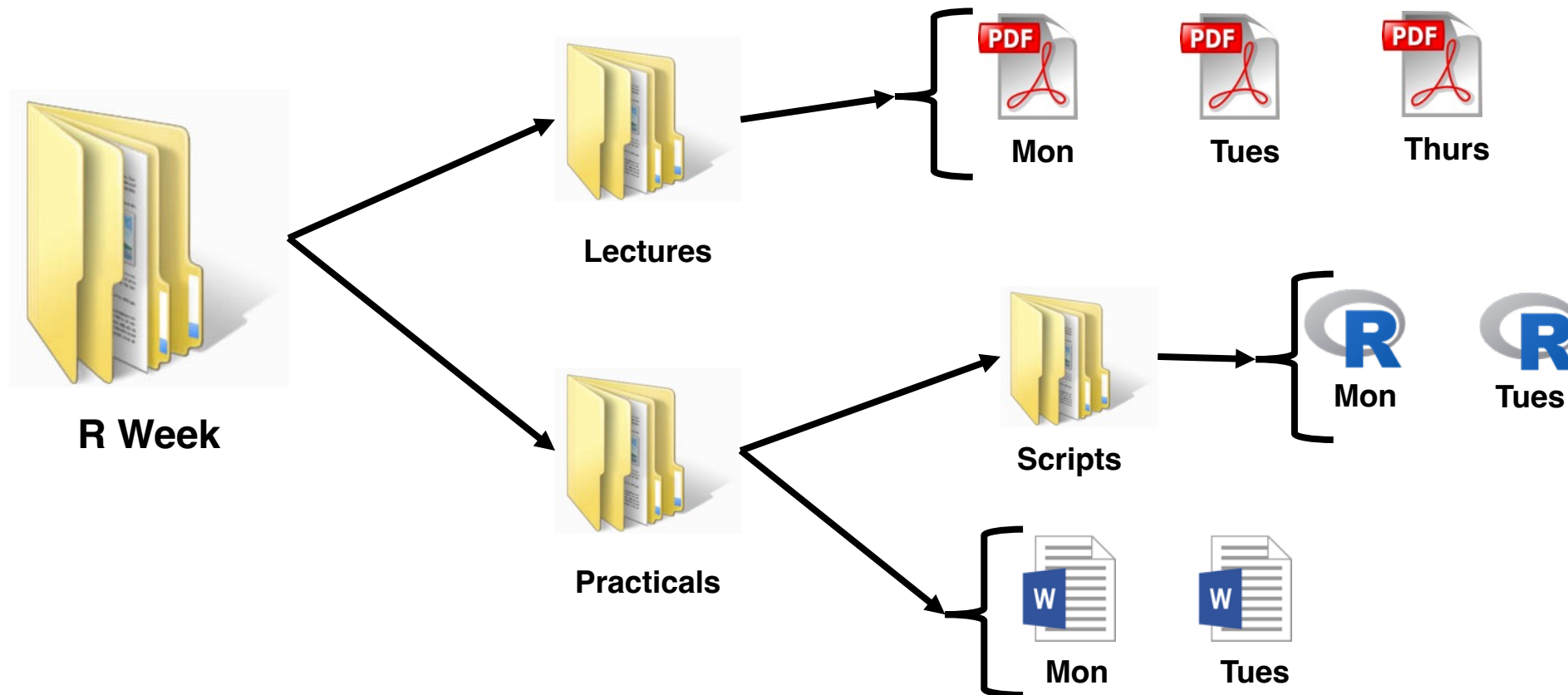
Rules of Scripting



1. Readability
2. Architecture First → **PLAN**
3. Keep it Simple
4. Comments `##Import Data##`
5. Modularise and Develop Iteratively
6. Strive for Automation

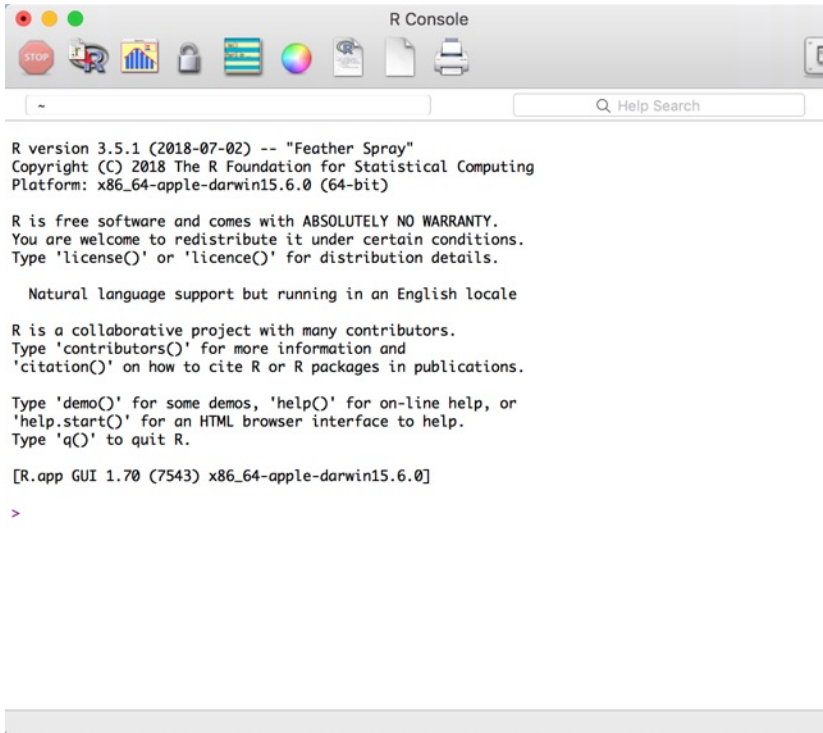
Directories

“File system cataloging computer files and other directories”



The R Environment

Basic



```
R Console

R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (64-bit)

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

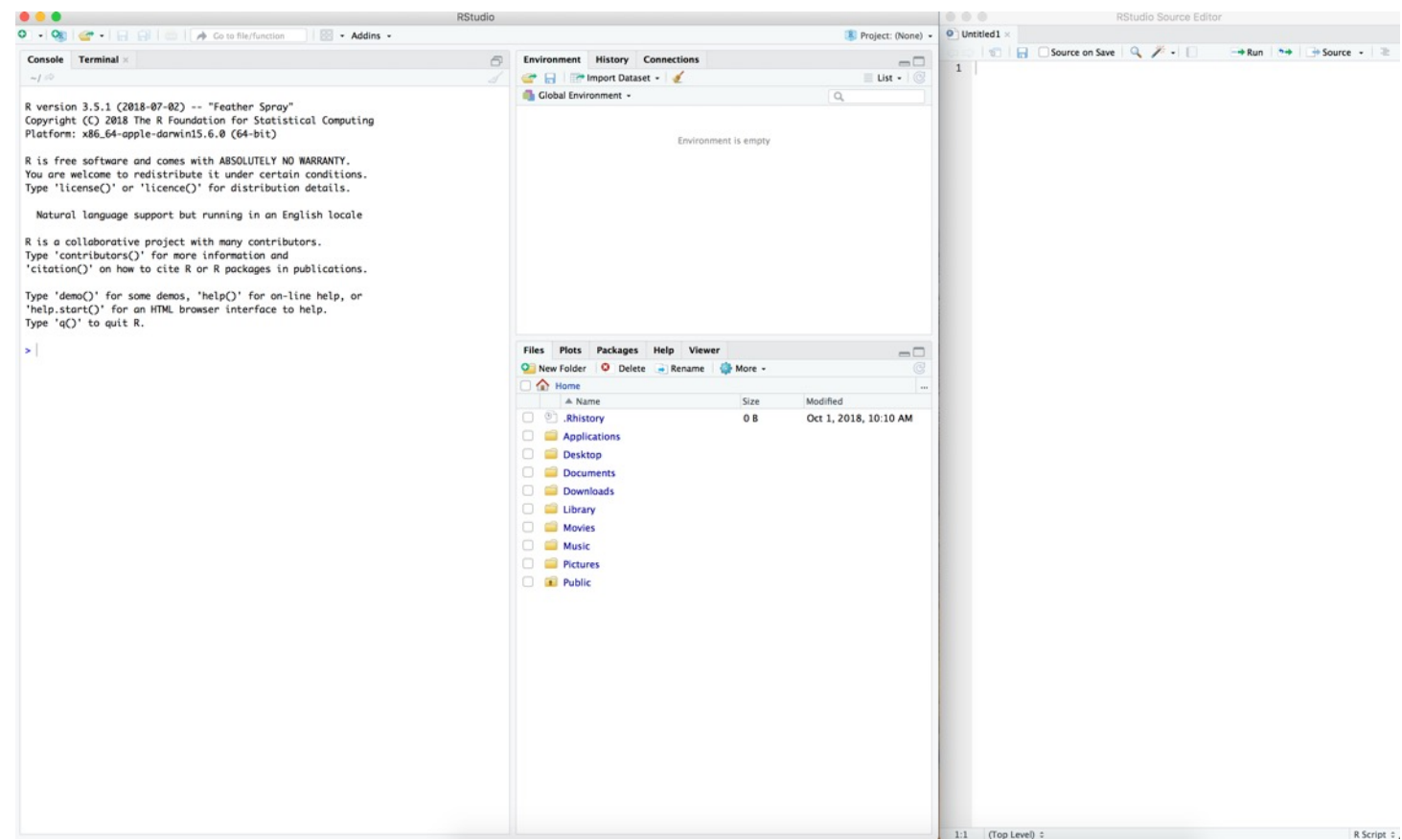
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

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.

[R.app GUI 1.70 (7543) x86_64-apple-darwin15.6.0]

>
```

IDE – RStudio





Accessing RStudio

Windows



You can access the Software Hub from College computers
and on your personal computer, for more information go to:
imperial.ac.uk/ict/software-hub

Mac



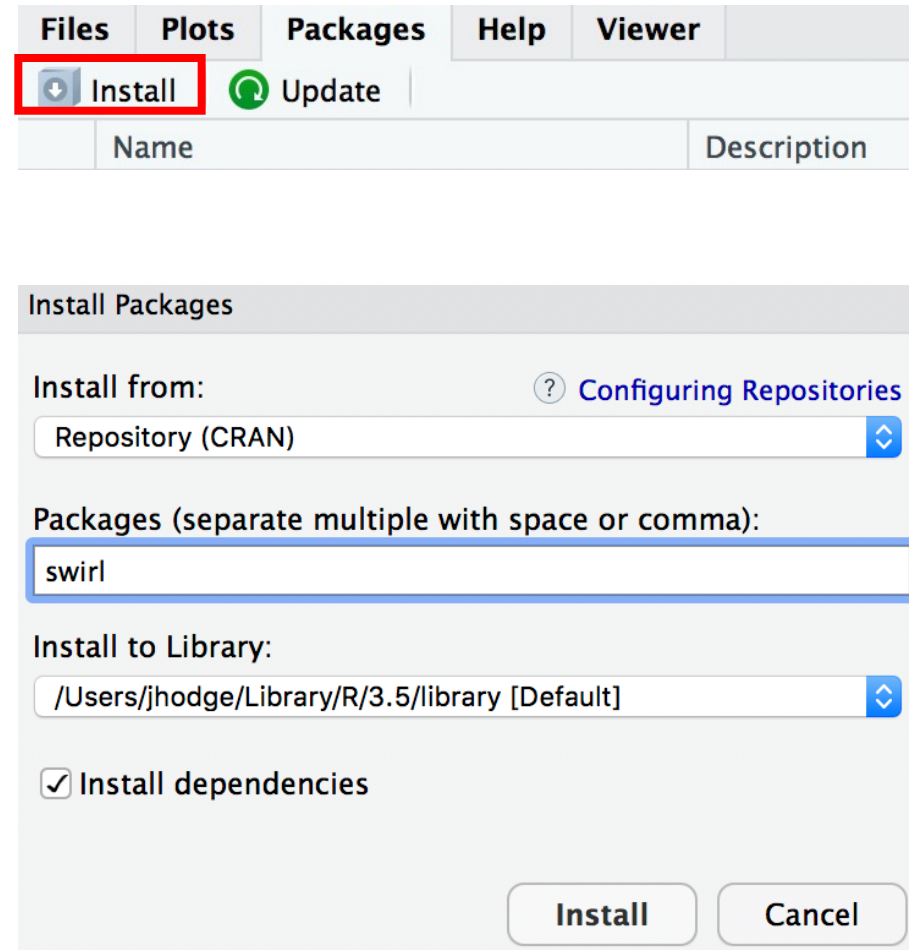
<https://cran.ma.imperial.ac.uk/>



<https://rstudio.com/products/rstudio/download/#download>

Installing and Loading Packages

```
##Install swirl##  
> install.packages("swirl")  
  
##Load swirl##  
> library("swirl")  
! Hi! Type swirl() when you are  
ready to begin.  
  
##Help for swirl function##  
> ?swirl
```



The screenshot shows the RStudio interface. The 'Packages' tab is selected in the top menu bar. Below it, the 'Install' button is highlighted with a red rectangle. The 'Update' button is also visible. Below the buttons, there is a table with columns 'Name' and 'Description'. Below the table, the 'Install Packages' dialog box is open. It has a title bar 'Install Packages'. Inside, there is a section 'Install from:' with a dropdown menu set to 'Repository (CRAN)' and a link '? Configuring Repositories'. Below that, there is a section 'Packages (separate multiple with space or comma):' with a text input field containing 'swirl'. Below that, there is a section 'Install to Library:' with a dropdown menu set to '/Users/jhodge/Library/R/3.5/library [Default]'. At the bottom, there is a checkbox 'Install dependencies' which is checked. At the bottom right, there are two buttons: 'Install' and 'Cancel'.

What's Next?



- Introduces R in a fun and easy way to learning R programming basics.
- Visit <https://swirlstats.com/>
- Complete by next session:
 - *R Programming*

1: Basic Building Blocks

4: Vectors

7: Matrices and Data Frames

5: Missing Values

8: Logic

3: Sequences of Numbers

6: Subsetting Vectors