Welcome to Introduction to R!

- Use the link to download the files we will be using today: https://github.com/ChristopherSwader/R_introduction
- Download the Day1 folder.

Introduction to R

R workshop:
Instructor: Christopher Swader, LU Sociology (Assoc. Prof) and LMU (Munich, Researcher). christopher.swader@soc.lu.se

Teaching Assistant: Maximilian Hornung (MS programme in Social Scientific Data Analysis, LU)

Slides adapted from and inspired by:
Irina Vartanova, Institute for Future Studies, Stockholm

The following sources were used for the slides under CC BY-NC 4.0:
Course content for Remaster the Tidyverse
RStudio Primers

Lesson 1 of 5
Getting Started

Intro to R workshop, LU Skills School

Instructor:

Christopher Swader
LU Sociology (Assoc. Prof) and LMU (Munich, Researcher)

Teaching Assistant:

Maximillian Hornung
(MS programme in Social Scientific Data Analysis, LU)

Today’s agenda

- Why R?
- Getting started with R
- Functions
- Objects
- Data Structures
- R packages/libraries

Introductions!

Why R?

New data (e.g. api, web scraping, survey experiments), New methods (machine learning)

Using Twitter as a data source: an overview of social media research tools (2019)

*Twitter and other social media platforms represent a large and largely untapped resource for social data and evidence. In this post, **Wasim Ahmed** updates his recurring series on the Impact Blog, to bring you the latest developments in digital methods and methodologies for researching Twitter and other social media platforms.*

This post builds upon the 2015, and 2017 editions of this post, captures key trends and events which are shaping social media research for social scientists and provides a collection of research methods and tools for the analysis of social media data.

Since the 2017 edition of this blog post, I have seen even more unique and interesting uses of social media data across a wide variety of research disciplines, such as sociology, computer science, media and communication, political science, and engineering to name only a few. Social media platforms generate a vast amount of data on a daily basis on a variety of topics and consequently represent a key source of information for anyone seeking to study 21st century society.

Twitter remains the most popular platform for academic research, as it still provides its data via a number of Application Programming Interfaces (API). In contrast, the aftermath of the Cambridge

The Multiple Burdens of Foreign-Named Men—Evidence from a Field Experiment on Gendered Ethnic Hiring Discrimination in Sweden

Moa Bursell

European Sociological Review, Volume 30, Issue 3, June 2014, Pages 399–409,
https://doi.org/10.1093/esr/jcu047
Published: 27 March 2014 **Article history** ▾

Cite Permissions Share ▾

Scholars have documented ethnic and gender discrimination across labour markets since the 1970s by using field experiments (correspondence tests) in which pairs of equally qualified applications are sent to employers with job openings. In these experiments, discrimination is measured by documenting group differences in callbacks. However, the gendered nature of ethnic discrimination has been neglected thus far in this literature. Drawing on the results of a correspondence test, this study presents evidence of extensive ethnic discrimination in the Swedish labour market against applicants with Arabic and North African names but no evidence of discrimination against women. However, the findings also reveal gendered ethnic employer preferences: employers in male-dominated occupations practice gender overcompensation favouring female-named applicants, whereas employers in female-dominated occupations practice both ethnic and gender overcompensation, favouring foreign-named men in particular.

Number of R packages available on CRAN by year
(Excluding archived packages)
Count
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
1998
2000
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022
Year
Source: CRAN Metadata from {pkgsearch}

The Comprehensive R Archive

Secure https://cran.r-project.org

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

Available CRAN Packages By Name

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A3	Accurate, Adaptable, and Accessible Error Metrics for Predictive Models
abbyyR	Access to Abbyy Optical Character Recognition (OCR) API
abc	Tools for Approximate Bayesian Computation (ABC)
ABCanalysis	Computed ABC Analysis
abc.data	Data Only: Tools for Approximate Bayesian Computation (ABC)
abcdeFBA	ABCDE_FBA: A-Biologist-Can-Do-Everything of Flux Balance Analysis with this package
ABCoptim	Implementation of Artificial Bee Colony (ABC) Optimization
ABCp2	Approximate Bayesian Computational Model for Estimating P2
ABC.RAP	Array Based CpG Region Analysis Pipeline
abcrf	Approximate Bayesian Computation via Random Forests
abctools	Tools for ABC Analyses
abd	The Analysis of Biological Data
abf2	Load Gap-Free Axon ABF2 Files
ABHgenotypeR	Easy Visualization of ABH Genotypes
abind	Combine Multidimensional Arrays
abjutils	Useful Tools for Jurimetrical Analysis Used by the Brazilian Jurimetrics Association
abn	Modelling Multivariate Data with Additive Bayesian Networks
abodOutlier	Angle-Based Outlier Detection

Reproducible workflow

RESEARCH ARTICLE

PSYCHOLOGY

Estimating the reproducibility of psychological science

Open Science Collaboration*†

Reproducibility is a defining feature of science, but the extent to which it characterizes current research is unknown. We conducted replications of 100 experimental and correlational studies published in three psychology journals using high-powered designs and original materials when available. Replication effects were half the magnitude of original effects, representing a substantial decline. Ninety-seven percent of original studies had statistically significant results. Thirty-six percent of replications had statistically significant results; 47% of original effect sizes were in the 95% confidence interval of the replication effect size; 39% of effects were subjectively rated to have replicated the original result; and if no bias in original results is assumed, combining original and replication results left 68% with statistically significant effects. Correlational tests suggest that replication success was better predicted by the strength of original evidence than by characteristics of the original and replication teams.

Reproducibility is a core principle of scientific progress (*1*–*6*). Scientific claims should not gain credence because of the status or authority of their originator but by the replicability of their supporting evidence. Scientists attempt to transparently describe the methodology and resulting evidence used to support their claims. Other scientists agree or disagree whether the evidence supports the claims, citing theoretical or methodological reasons or by collecting new evidence. Such debates are

results are false and therefore irreproducible (*9*). Some empirical evidence supports this analysis. In cell biology, two industrial laboratories reported success replicating the original results of landmark findings in only 11 and 25% of the attempted cases, respectively (*10*, *11*). These numbers are stunning but also difficult to interpret because no details are available about the studies, methodology, or results. With no transparency, the reasons for low reproducibility cannot be evaluated.

Other investigations point to practices and

facilitated each step of the process and maintained the protocol and project resources. Replication materials and data were required to be archived publicly in order to maximize transparency, accountability, and reproducibility of the project (https://osf.io/ezcuj).

In total, 100 replications were completed by 270 contributing authors. There were many different research designs and analysis strategies in the original research. Through consultation with original authors, obtaining original materials, and internal review, replications maintained high fidelity to the original designs. Analyses converted results to a common effect size metric [correlation coefficient (r)] with confidence intervals (CIs). The units of analysis for inferences about reproducibility were the original and replication study effect sizes. The resulting open data set provides an initial estimate of the reproducibility of psychology and correlational data to support development of hypotheses about the causes of reproducibility.

Sampling frame and study selection

We constructed a sampling frame and selection process to minimize selection biases and maximize generalizability of the accumulated evidence. Simultaneously, to maintain high quality, within this sampling frame we matched individual replication projects with teams that had relevant interests and expertise. We pursued a quasi-random sample by defining the sampling frame as 2008 articles of three important psychology journals: *Psychological Science* (*PSCI*), *Journal of Personality and Social Psychology* (*JPSP*), and *Journal of Experimental Psychology: Learning, Memory, and Cognition* (*JEP:*

And CRAN is not the only place you can find R packages and code:

Also check GITHUB

And for debugging help and code snippets, ***Stack Overflow***

Learning R is really about learning how to more efficiently google. You need to do so less often, but mainly you learn to phrase your questions better and better apply the results to your problem.

It is free!

It has great, active communities

useR!

RStudio

R Ladies

Posit Community

R-bloggers

https://www.reddit.com/r/RStudio/

You can work **faster** and also gain a more **profound** understanding of your data and methods.

Great graphs

Advanced models

Animations

https://github.com/thomasp85/gganimate

Interactive html applications ('shinyr')

https://irinavrt.shinyapps.io/moral_opinion_change/

Develop your own routines

(e.g. multiverse analysis, or webscraping)

, algorithms

(e.g. a new kind of statistical procedure or machine learning algorithm)

, visualizations

(e.g. proportional rectangular Venn for visualizing multi level charateristics)

https://github.com/ChristopherSwader

If statistics programs/languages were cars...

Learning a language

- This will be sometimes frustrating
- A lot to learn: structure, syntax, and vocabulary
- Need to look up functions, errors

Kick A$$ Threshold

Author Kathy Sierra

Getting Started

Setup before you begin

- It is assumed that your computer is **already installed** with R gui (version >= 4.4.1), Rstudio (v >= 2023.12.1), and Rtools (v >= 4.3.555). The software center at LU, or your IT support can help you with this. If you have full rights to your computer, you can do this yourself through online pages.

- Also install tidyverse *install.packages(”tidyverse”)*

A language

Writing software

A language

R: Engine

Writing software

RStudio: Dashboard

Note that you MUST have the base R language (and its attached GUI) also installed, but I strongly recommend using Rstudio as your way to work with the R language (it makes use of the base installation). **Updating R** means usually updating 'base R' and its attached GUI, any packages that you use, and sometimes R studio itself. Recommend: if it ain't broke, don't fix it. If it is broken suddenly without obvious changes, update everything.

Data analysis framework

Console Terminal Jobs

~/

```
R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin17.0 (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.

>
```

Environment History Connections Tutorial

Import Dataset List

Global Environment

Environment is empty

Files Plots Packages Help Viewer

New Folder Delete Rename More

Home > Dropbox > projects > R course > Introduction to R

Name	Size	Modified
..		
exercises		
slides		

Console: this is where you can both execute your code and see the results.

A language

Your turn

- Type *1:10* and press enter
- Type *10:1* and press enter
- Type *1:10* + *10:1* and press enter

All executed code is saved in *History*

Console | Terminal | Jobs

~/Dropbox/projects/R course/

```
R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin17.0 (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.

> 3+2
[1] 5
>
```

Environment | History | Connections | Tutorial

To Console | To Console | To Source

```
3+2
```

Files | Plots | Packages | Help | Viewer

New Folder | Delete | Rename | More

pbox > projects > R course > Introduction to R > Day1_First-Steps

Name	Size	Modified
..		
00-Introduction-Slides.pdf	7.5 MB	Oct 4, 2020, 1:1
01-First-Steps-Slides.pdf	7.2 MB	Oct 4, 2020, 1:1
01_First-Steps-Exercises.Rmd	4.7 KB	Sep 21, 2020, 4:

Navigate through your computer's files and folders.

Console | Terminal | Jobs

~/Dropbox/projects/R course/

```
R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin17.0 (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.

> 3+2
[1] 5
>
```

Environment | History | Connections | Tutorial

To Console | To Source

3+2

Files | Plots | Packages | Help | Viewer

New Folder | Delete | Rename | More

pbox > projects > R course > Introduction to R > Day1_First-Steps

Name	Size	Modified
..		
00-Introduction-Slides.pdf	7.5 MB	Oct 4, 2020, 1:1
01-First-Steps-Slides.pdf	7.2 MB	Oct 4, 2020, 1:1
01_First-Steps-Exercises.Rmd	4.7 KB	Sep 21, 2020, 4:

Browse your computer file system

Console Terminal Jobs

~/Dropbox/projects/R course/

```
R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin17.0 (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.

> 3+2
[1] 5
>
```

Environment History Connections Tutorial

To Console To Source

3+2

Files Plots Packages Help Viewer

New Folder Delete Rename More

pbox > projects > R course > Introduction to R > Day1_First-Steps

Name	Size	Modified
..		
00-Introduction-Slides.pdf	7.5 MB	Oct 4, 2020, 1:1
01-First-Steps-Slides.pdf	7.2 MB	Oct 4, 2020, 1:1
01_First-Steps-Exercises.Rmd	4.7 KB	Sep 21, 2020, 4:

Environment temporarily saves all of your relevant data and variables used during the session.

But to *permanently* save our routine and make it reproducible, we should rely on saving an R script. (A script is simply a file with code).

Your turn

- Make your own R script for today. I recommend a separate one for each day (to keep it navigable).(I recommend that you work on your own script rather than merely adapting and annotating mine. (you will remember better that way and will better keep track of your own work.)

- Keep it in a place that you will find it. You can supplement, compare it to mine, found on the Github page for this course.

LLM’s for coding?

- Yes, LLM’s like ChatGPT can code (at least the basics). But so can your colleague down the hallway.

- Because you came here to learn to develop your **own** skill set, I recommend you first code by yourself. If you get stuck, google the answer and get used to interpreting the results to your case.

- If you get really stuck, you can also ask AI, but in this case, give your code, your intention, and **ask what is wrong** rather than asking it to generate a full solution.

- My own experience of using AI for coding assistance showed it failing in some more complex tasks, creating annoying bugs in some intermediate tasks, doing fine in the basics, and giving some helpful starting points for some very niche programming languages.

Your Turn

- What does this command return?

```
getwd()
```

I recommend you run your commands now within your script for the day. You can run the current line by press CNTL + ENTER (PC) or COMMAND + RETURN (MAC)

Working directory

R associates itself with a folder (i.e. directory) on your computer.

- This folder is known as your "working directory"
- When you save files, R will save them here
- When you load files, R will look for them here

  … unless you specify a full path, it uses your working directory as the default.

R intro 2020 - RStudio

Go to file/function | Addins | R intro 2020

Console | Terminal | Jobs

~/Dropbox/projects/R course/R intro 2020/

```
R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin17.0 (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.

>
```

Environment | History | Connections | Tutorial

Import Dataset | List

Global Environment

Environment is empty

Files | Plots | Packages | Help | Viewer

New Folder | Delete | Rename | More

Home > Dropbox > projects > R course > R intro 2020

Name	Size	Modified
..		
cheatsheets		
data		
Day1_First-Steps		
Day2_Visualise		
Day3_Transform		
Day4_Model		
Day5_Communicate		
R intro 2020.Rproj	205 B	Oct 6, 2020, 8:2

You can navigate to today's folder and then set it as the working directory

Your Turn

- Now set your working directory either by using setwd() with the full path inside or by clicking on Files → More → set as working directory.

RStudio project

- Helps to keep all the files associated with a project together
- And separate analysis

RStudio File Edit Code View Plots Session Build Debug Profile Tools Window Help
New File
New Project...
Open File... ⌘O
Recent Files
Open Project...
Open Project in New Session...
Recent Projects
Import Dataset
Save ⌘S
Save As...
Save All ⌥⌘S
Print...
Close ⌘W
Close All ⇧⌘W
Close All Except Current ⌥⇧⌘W
Close Project
Quit Session... ⌘Q
RStudio
Addins
Console
~/
R version "Taking Off Again"
Copyright on for Statistical Computing
Platform: (64-bit)
R is free ABSOLUTELY NO WARRANTY.
You are w t under certain conditions.
Type 'lic for distribution details.
Natural nning in an English locale
R is a co 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.
>

New Project Wizard
Back
Create Project from Existing Directory
Project working directory:
~/Dropbox/projects/R course/Introduction to R
Browse...
Open in new session
Create Project
Cancel

Your Turn

- Make an RStudio project from the parent of the existing directory which is the folder you use for the workshop. (Your new project will take that name of that directory)

R intro 2020 - RStudio

Go to file/function | Addins | R intro 2020

Console | Terminal | Jobs

~/Dropbox/projects/R course/R intro 2020/

```
R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin17.0 (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.

>
```

Environment | History | Connections | Tutorial

Import Dataset | List

Global Environment

Environment is empty

Files | Plots | Packages | Help | Viewer

New Folder | Delete | Rename | More

Home > Dropbox > projects > R course > R intro 2020

Name	Size	Modified
..		
cheatsheets		
data		
Day1_First-Steps		
Day2_Visualise		
Day3_Transform		
Day4_Model		
Day5_Communicate		
R intro 2020.Rproj	205 B	Oct 6, 2020, 8:2

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Untitled1*
Source on Save
Run
Source
Here is my starting script
A series of numbers
10:1
6:1
(Top Level)
R Script
Console
Terminal
Background Jobs
R 4.4.1 · ~/
> 10:1
[1] 10 9 8 7 6 5 4 3 2 1
>
Environment
History
Connections
Tutorial
184 MiB
List
R
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Home
Name
Size
-workfolders.RData 524.6 KB
-workfolders.Rhistory 15 KB
.RData 40.2 MB
.Rhistory 13.5 KB
.Rprofile 0 B
Advanced Courses information_
agglomeration_tree.txt 831 B
Anpassade Office-mallar
Bilder - genväg.lnk 1006 B
chicago-author-date no url 21.6 KB
Regularly save your script to a place that you remember (e.g. a folder for this workshop.
Annotate it with # so that you remember what you were doing and exactly reproduce your steps.
Annotation is really important for you and for others to make your code transparent!!

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins | Project: (None)

Document outline

Untitled1*

Source on Save | Run | Source

```
# Here is my starting script

# A series of numbers####

10:1

##Descending####
```

A series of numbers
Descending

7:17 | Descending | R Script

You can create bookmarks in your script by adding four # signs after your annotations and any number before (to set the level of hierarchy)

Console | Terminal | Background Jobs

R 4.4.1 · ~/

```
> 10:1
 [1] 10  9  8  7  6  5  4  3  2  1
>
```

Environment | History | Connections | Tutorial

122 MiB | List

R | Global Environment

Environment is empty

Files | Plots | Packages | Help | Viewer | Prese

Home

Name	Size
-workfolders.RData	524.6 KB
-workfolders.Rhistory	15 KB
.RData	40.2 MB
.Rhistory	13.5 KB
.Rprofile	0 B
Advanced Courses information_	
agglomeration_tree.txt	831 B
Anpassade Office-mallar	
Bilder - genväg.lnk	1006 B
chicago-author-date no url	21.6 KB

Your Turn

- Practice making a couple of bookmarks in your R script to indicate what we are doing and make it easier to navigate.

- This is especially important if you are adapting the script that I have given you, so that you can easily find your own code (vs. Mine).

Basic R syntax

Function ”arguments”.

```
my_dist <- rnorm(n=100, mean=0, sd=1)
```

”rnorm” is the name of the function, indicated with parentheses

Your Turn

- Learn about the function rnorm() and its arguments by typing: *?rnorm*.
- Plot a histogram of my_dist using *hist().*
- Try a different *n* for rnorm() and see how the histogram changes. Try 10, and 1000 cases.
- Change the ’breaks’ argument in hist() and observe the results.

Basic R syntax

Function ”arguments”.

```
my_dist <- rnorm(n=100, mean=0, sd=1)
```

”rnorm” is the name of the function, indicated with parentheses

”x” argument: your data

```
hist(my_dist)
```

function

R code lines do not (usually) need to end with any special character. Just put new lines below the old. R executes them sequentially.
Upper/lower case matters!

There is a VAST array of R functions.

```
Sys.Date()
## [1] "2023-10-02"
sqrt(9)
## [1] 3
seq(1, 10)
## [1]  1  2  3  4  5  6  7  8  9 10
id <- paste0("Swader", paste0(sample(1:10, 5, replace = F,),
collapse = ""))
## [1] "Swader57219"
```

And you can combine and create (almost) any that you can imagine.

The existing ones are stored in ’packages’ or ’libraries’. You will often use a combination of these.

If you like, Use R ’cheatsheets’ for reference

- We will use a lot of code that you will forget already next week
- Try to understand how it works
- Remember where to look it up. Google!
- Through the rstudio ’help’ menu
- https://posit.co/resources/cheatsheets/

Data Wrangling with dplyr and tidyr

Cheat Sheet

RStudio

Syntax - Helpful conventions for wrangling

dplyr::**tbl_df(iris)**

Converts data to tbl class. tbl’s are easier to examine than data frames. R displays only the data that fits onscreen:

```
Source: local data frame [150 x 5]

   Sepal.Length Sepal.Width Petal.Length
1           5.1         3.5          1.4
2           4.9         3.0          1.4
3           4.7         3.2          1.3
4           4.6         3.1          1.5
5           5.0         3.6          1.4
..          ...         ...          ...
Variables not shown: Petal.Width (dbl),
  Species (fctr)
```

dplyr::**glimpse(iris)**

Information dense summary of tbl data.

utils::**View(iris)**

View data set in spreadsheet-like display (note capital V).

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa

dplyr::**%>%**

Passes object on left hand side as first argument (or . argument) of function on righthand side.

`x %>% f(y)` *is the same as* `f(x, y)`

`y %>% f(x, ., z)` *is the same as* `f(x, y, z )`

"Piping" with %>% makes code more readable, e.g.

```
iris %>%
  group_by(Species) %>%
  summarise(avg = mean(Sepal.Width)) %>%
  arrange(avg)
```

Tidy Data - A foundation for wrangling in R

In a tidy data set:

Each **variable** is saved in its own **column**

Each **observation** is saved in its own **row**

Tidy data complements R’s **vectorized operations**. R will automatically preserve observations as you manipulate variables. No other format works as intuitively with R.

M * A

Reshaping Data - Change the layout of a data set

tidyr::**gather(cases, "year", "n", 2:4)**

Gather columns into rows.

tidyr::**spread(pollution, size, amount)**

Spread rows into columns.

tidyr::**separate(storms, date, c("y", "m", "d"))**

Separate one column into several.

tidyr::**unite(data, col, ..., sep)**

Unite several columns into one.

dplyr::**data_frame(a = 1:3, b = 4:6)**

Combine vectors into data frame (optimized).

dplyr::**arrange(mtcars, mpg)**

Order rows by values of a column (low to high).

dplyr::**arrange(mtcars, desc(mpg))**

Order rows by values of a column (high to low).

dplyr::**rename(tb, y = year)**

Rename the columns of a data frame.

Subset Observations (Rows)

dplyr::**filter(iris, Sepal.Length > 7)**

Extract rows that meet logical criteria.

dplyr::**distinct(iris)**

Remove duplicate rows.

dplyr::**sample_frac(iris, 0.5, replace = TRUE)**

Randomly select fraction of rows.

dplyr::**sample_n(iris, 10, replace = TRUE)**

Randomly select n rows.

dplyr::**slice(iris, 10:15)**

Select rows by position.

dplyr::**top_n(storms, 2, date)**

Select and order top n entries (by group if grouped data).

Logic in R - ?Comparison, ?base::Logic

<	Less than	!=	Not equal to
>	Greater than	%in%	Group membership
==	Equal to	is.na	Is NA
<=	Less than or equal to	!is.na	Is not NA
>=	Greater than or equal to	&, |, !, xor, any, all	Boolean operators

Subset Variables (Columns)

dplyr::**select(iris, Sepal.Width, Petal.Length, Species)**

Select columns by name or helper function.

Helper functions for select - ?select

select(iris, contains(".t."))
Select columns whose name contains a character string.

select(iris, ends_with("Length"))
Select columns whose name ends with a character string.

select(iris, everything())
Select every column.

select(iris, matches(".t."))
Select columns whose name matches a regular expression.

select(iris, num_range("x", 1:5))
Select columns named x1, x2, x3, x4, x5.

select(iris, one_of(c("Species", "Genus")))
Select columns whose names are in a group of names.

select(iris, starts_with("Sepal"))
Select columns whose name starts with a character string.

select(iris, Sepal.Length:Petal.Width)
Select all columns between Sepal.Length and Petal.Width (inclusive).

select(iris, -Species)
Select all columns except Species.

RStudio® is a trademark of RStudio, Inc. • CC BY RStudio • info@rstudio.com • 844-448-1212 • rstudio.com

devtools::install_github("rstudio/EDAWR") for data sets

Learn more with browseVignettes(package = c("dplyr", "tidyr")) • dplyr 0.4.0• tidyr 0.2.0 • Updated: 1/15

Tidyverse

A popular family of packages

R code of high quality is....

- Easy to write
- Easy to read
- Easy to reproduce

Tidyverse and 'Base R'

- Ecosystem of packages
- The same code syntax
- Intuitive code, that follows the way (some of us) think

- Sometimes more efficient (and easier to remember)

And *many thousands of* others

Overall: Use what works for you! You will certainly mix packages. Remember 'Mad Max' style. :-)

Your turn

- Open RStudio.
- *library("wbstats") #this is how we load a library!!*
- *Error in library("wbstats") : there is no package called ‘wbstats’*
- Google ”how to install a package in R?” and install ”wbstats”

- *install.packages(”wbstats”)*

Installing a package is something you only need to do once, not in every session!

But LOADING a package with library() is something you do once you have already installed it and want to use it.

Your turn

- Type *“here is some text”* (with quotes) in the console, press enter

- Double click on the last line (at the bottom) in history to call it up it one more time. Press enter.

- Assign it to an object to save it for later use. *my_text <- ”some important text”*

- Create a string vector with three pieces of text in a series:

- *my_text <- c(my_text, ”Welcome”, ”User”)*

  A ’vector’ is a series of values of a similar type (string, integers, numeric).

Some shortcuts!

I use these key ones in **<u>every single</u>** R session:

- alt-dash: ← (creates assignment arrow!) or opt-dash on a MAC.
- cntl-ENT (cmd-ret): excutes current line of code or selection of code in your script
- up and down arrows: cycle through recently used code
- highlight code and shift-cntl-C: comments in or out code!
- Highlight code and quotes/parentheses/brackets: inserts quotes on both sides
- Autofill: start typing the name of any loaded R function or object and you should see an option to complete the typing
- Look them up! https://support.posit.co/hc/en-us/articles/200711853-Keyboard-Shortcuts-in-the-RStudio-IDE

Some basic elements in R

Objects

Functions and their arguments

Data structures:
- **vectors**
- **lists (data frames)**

Functions

Functions

```
Sys.Date()
sqrt(9)
seq(1, 10)
```

function name

comma

argument

Using the *argument names* when supplying the argument is optional. R expects them in the order of the function definition, helpfile. You can change that order by supplying the name. e.g. *seq(to=10,from= 1)* is equivalent to seq(1, 10)

Arguments

```
seq(from = 1, to = 10, by = 1)
```

Argument name

Argument value

The function help (e.g. *?seq()*) will tell you which data type, format is needed as the function values.

A very common error is that you supply the wrong kind of data.

Names and order

```
seq(1, 10, 1)
## [1]  1  2  3  4  5  6  7  8  9 10

seq(10, 1, 1)
## Error in seq.default(10, 1, 1) : wrong sign
in 'by' argument

seq(10, 1, by = -1)
## [1] 10  9  8  7  6  5  4  3  2  1
```

Names and order

```
seq(1, 10, 1)
## [1]  1  2  3  4  5  6  7  8  9 10

seq(10, 1, 1)
## Error in seq.default(10, 1, 1) : wrong sign
in 'by' argument

seq(10, 1, by = -1)
## [1] 10  9  8  7  6  5  4  3  2  1

seq(to = 10, from = 1, by = 1)
 [1]  1  2  3  4  5  6  7  8  9 10
```

Optional arguments

```
log(100, base = 10)
## [1] 2
```

In the help of a function, e.g. ?log() some arguments have a default specified by *argument_name* = *value*. This means they are optional.

Those with no default are required.

Optional arguments

```
log(100, base = 10)
## [1] 2

log(100)
## [1] 4.60517

log(base = 10)
## Error: argument "x" is missing, with no
default
```

Your turn

- sample() is a function takes a sample of the specified size from the elements of x using either with or without replacement.
- Which arguments of sample() are required?
- Use sample() to generate 10 random draws of a vector of 5 names.
- You can make a vector by using combine: ***c(1, 2, 3, 4, 5)***

```
these_names <- c("Anna", "Bob", "Cindy", "David", "Edith")
these_names
##   [1] "Anna"  "Bob"   "Cindy" "David" "Edith"


this_draw <- sample(these_names, 10, replace = T)
this_draw
##  [1] "Bob"   "Bob"   "Cindy" "Edith" "Edith" "Bob"   "Bob"   "Cindy"
##  [9] "Edith" "Edith"
```

9th item in this string vector

10th item in this string vector

One of the most crucial base R utilities: Square bracket indexing.

Type *this_draw[9]*

This has major powerful uses also within datasets for selecting, filtering, modifying, etc.

Objects

Objects

An object is like a generic box that stores an individual value, a vector of values, a list, a dataframe, or more complex.

```
numbers <- rnorm(100, mean = 100, sd = 50)
```

object name

Assignment
ALT-dash
OPT-dash

value

rnorm() is similar to sample() except that it creates a normal distribution.

Object names do not:

- start with a number
- include spaces or special characters
- Recommend no periods

I recommend you do **<u>not</u>** use an equal sign for assignment and use this arrow instead! *(because equal signs are used in functions, as logical tests, and the arrow makes it more obvious when you are assigning something to an object)*

Objects

```
value <- 10
```

assignment

Use Alt + `-` to type
(Option + `-` on Mac)

*Objects are central to R as they are like **boxes** (or variables) where you add some kind of information in order to use it later.*

Pass these objects to other functions

```
mean(numbers)
## [1] 10.03636

sqrt(value)
## [1]  3.162278
```

Your turn

- *data(iris)* [This loads the data! They are among lots of practice data included in every R session and often used for training]

- Draw a random sample of 50% of the rows (You can see the number of observations in the environment window)

- Then use those indices to create a new subsample dataset, using square bracket indexing in the format ***[rows, columns]***.

Solution

- *data(iris)*
- *this_sample ← sample(1:nrow(iris), round(.5*nrow(iris)), replace=F)*
- *sub_iris ← iris[this_sample,]* **Square bracket indexing applied to a data.frame in the format [rows, columns].**
- And a taster of the graphics we will do later!

library(ggplot2)

ggplot(sub_iris, aes(x = Sepal.Length, y = Petal.Length, color = Species)) +

geom_point()

*We'll do more **ggplot** later. It can produce amazing custom graphics. This example is quite basic.*

Functions are also objects

If you type its name without the parentheses, you will get the base code.

e.g. type *sample*

For this reason, you never want to name a new object the same as an existing object/function. If you do so, you will save over the old function or object. You could reload the old one by reloading the relevant library

Your turn

- Data sets are also stored as objects in R. R has many pre-loaded data sets. Type data() to see them.
- Examine the contents of iris to see a classic toy data set.
- How could you learn more about the iris object?

```
iris
head(iris)
View(iris)
?iris
```

*In the Environment pane, you can also click on the **blue arrow** to the left of data or click on the table to the right to display it with **View()***

Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fctr>
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa

Quiz

- What happens if you assigned data to a new object named copy, like this?

```
data <- rnorm(100, mean = 100, sd = 15)
copy <- data
```

- What happens with copy when you change the data object?

```
data <- seq(1, 10, by = 1)
data
copy
```

```
data
##  [1]  1  2  3  4  5  6  7  8  9 10
copy
##   [1] 126.18245  95.37482 108.88396 102.53111 129.28824 102.40587
##   [8]  98.79536 131.72468 100.78783  85.71048 124.63236  80.74413
```

Data structures

Data types

```
my_numeric<- c(1, -3, 7.18)
my_string <- c("apple", "banana")
my_integers <- c(1L, 2L, 3L)
my_logicals <- c(TRUE, FALSE, FALSE)
```

```
typeof(my_numeric)
typeof(my_string)
typeof(my_integers)
typeof(my_logicals)
```

```
## [1] "double"
## [1] "character"
## [1] "integer"
## [1] "logical"
```

str() is one of my most-used commands. It tells me all of the data types within an object, including the different types in different columns of a data set.

Values, Vectors

```
this_value <- 10
this_value
##   [1] 10
this_vector <- 1:10
this_vector
##  [1]  1  2  3  4  5  6  7  8  9 10
this_other_vector <- c("Marie","Pontus", "Georgina")
this_vector
##  [1] "Marie"    "Pontus"  "Georgina"
```

c() ”combine”

```
c(1, 2, 3) also a vector
c(a = 1, b = 2, c = 3)named
vector
```

Vectors can only hold ***one type*** *of data*

Your turn

- Create a vector called vec that contains the integers from one to ten.
- Assign ”hello” as the 11th item of that vector.
- What happens to the data type of the vector?

```
vec <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
vec <- 1:10
vec <- seq(1, 10, by = 1)

vec

##  [1]  1  2  3  4  5  6  7  8  9 10

vec[11] <- ”hello”

vec  <- append(vec, ”hello”)

vec  <- c(vec, ”hello”)

str(vec)

##  chr  [1:11] "1" "2" "3" "4" "5" "6" "7"
"8" "9" "10" "hello"
```

e.g. you can’t take the mean of a string vector...even if the strings are numbers! e.g. mean(vec[1:10])

Type coercion

Vectors have only type

```
typeof( c(3.14, TRUE, 6L, "hello") )

## [1] "character"
```

Extract elements of vectors

```
vec[3]
##  [1]  3
```

Consider learning **square brackets** really well. They are soooo useful, efficient.

Quiz

What will the following code return?

```
vec[c(1,2,11)]
```

Your turn

Extract the element named gamma from the vector below.

```
vec2 <- c(alpha = 1, beta = 2, gamma = 3)
```

```
vec2 <- c(alpha = 1, beta = 2, gamma = 3)
 vec2["gamma"] #or vec2[3]
## gamma
##     3
```

```
vec2 <- c(alpha = 1, beta = 2, gamma = 3)
vec2[c("alpha", "gamma")]
## alpha gamma
##     1     3
```

So in a labeled vector, you can also index (subset) using the label!

Quiz

Can you guess what will the code return?

```
c(1, 2, 3, 4, 5) + c(1, 2, 3, 4, 5)
```

And this?

```
1 + c(1, 2, 3, 4, 5)
```

Vectorised operations

- Repeat operation for every pair of elements

```
c(1, 2, 3, 4, 5) + c(1, 2, 3, 4, 5)
##  [1]  2  4  6  8  10
```

- Shorter vectors repeated as needed

```
1 + c(1, 2, 3, 4, 5)
##  [1]  2  3  4  5  6
```

length 1

Once you get used to thinking in this way, your code will get way faster, as you can often avoid loops.

Reminder

```
str(iris)
```

```
iris[1,]
iris[,1]
```

In a data.frame or other 2 dimensional data, you use square brackets to extract [rows, columns]. The number refers to the index number, starting with one.

Quiz

- Which of these are numbers within R? What are the rest?

1 "1" "one" one

Quiz

- Which of these are numbers?

1	"1" "one"	one
Number (integer)	words (characters/ strings)	object

A very common error in R is having the wrong type of data entered into a function. Use str() to verify your data types! e.g. you cannot add "1"+"1"

Quiz

Which of these will work? Suppose **one <- 1**.

log(1) log("1") log("one") log(one)

Quiz

Which of these will work? Suppose **one <- 1**.

log(1) log("1") log("one") log(one)

list()

```
a_list <- list(numbers = 1:5,
               strings = "hello",
               logicals = FALSE)
a_list
## $numbers
## [1] 1 2 3 4 5
## $strings
## [1] "hello"
## $logicals
## [1] FALSE
```

Lists, unlike vectors, can hold a mixture of data types!

$

Extract an element from the list.

```
a_list$numbers
## [1] 1 2 3 4 5
a_list[[2]]
## [1] 1 2 3 4 5
a_list[["numbers"]]
## [1] 1 2 3 4 5
```

The dollar sign is also used to extract an entire column from a dataframe, e.g. a variable!

To square bracket index a list's elements directly, ***use double brackets****. [single brackets returns the elements still AS A list, instead of e.g. a vector]*

RStudio autocompletion

```
> a_list
$numbers
[1] 1 2 3 4 5
```

```
a_list
alarm          {utils}
alias          {stats}
alist          {base}
all            {base}
all.equal      {base}
```

```
> a_l
```

```
> a_list
$numbers
[1] 1 2 3 4 5

$strings
[1] "hello"

$logical
[1] FALS
```

```
numbers
strings
logicals
```

```
> a_list$
```

data.frame()

```
iris
```

Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fctr>
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa

*There are many table formats in R. They include the **tibble**, the **data.frame**, the **data.table**, and the **matrix**, among others. They have subtle differences. To start* **I would recommend choosing data.frame or tibble and stick to it**, *until you run into a problem (a good strategy generally in R). e.g. data.table() is excellent at handling sorting quickly in very large data sets.*

Your turn

Assemble the vectors below into a data.frame() called df with the column names numbers, logicals, strings. Extract the ”strings” column of the data frame. (Hint, try help ’?’ on dataframe to get an example).

```
   nums <- c(1, 2, 3, 4)
logs <- c(TRUE, TRUE, FALSE, TRUE)
strs <- c("apple", "banana", "carrot", "duck")
```

```
df <- data.frame(numbers = nums,
                 logicals = logs,
                 strings = strs)
df
##   numbers logicals strings
## 1       1     TRUE   apple
## 2       2     TRUE  banana
## 3       3    FALSE  carrot
## 4       4     TRUE    duck

df$strings
## [1] apple  banana carrot duck
#or df[,"strings"]
```

Notice you can use vectors as variables to create a dataframe (given they have the same length). And when you extract a column, variable… you have a vector.

In other words, dataframes can be seen as being built from vectors

Factors

R's representation of categorical data.

Consists of:

1. A set of values
2. An (ordered) set of valid levels

```
eyes <- factor(x = c("blue", ”brown, "green",
"green"),levels = c("blue", "brown", "green"))

## [1] blue  brown green green
Levels: blue brown green
```

Your turn

Run the code below. Extract the factor column of the data frame. How is it different from the strings column?

```
df$fruit <- factor(df$strings,
                   levels = c("carrot", "apple", "banana"))
```

```
df$strings
## [1] apple  banana carrot duck


df$fruit
## [1] apple  banana carrot <NA>
## Levels: carrot apple banana
```

Ordered Factor (ordinal variables)

```
df$pro_choice <- factor(c("agree", "disagree",     "agree",
  "strongly disagree"), ordered=TRUE,
  levels = c("strongly disagree", "disagree",      "neutral",
  "agree", "strongly agree"))

df$pro_choice

 [1] agree              disagree           agree
strongly disagree

 Levels: strongly disagree < disagree < neutral < agree <
strongly agree
```

levels()

Use **levels()** to access a factor's levels

```
moral_issues$issue <- as.factor(moral_issues$issue)

levels(moral_issues$issue)


 [1] "abany"    "abdefect" "abhlth"   "abnomore" "abpoor"   "abrape"   "absingle" "advfront"
 [9] "affrmact" "busing"   "cappun"   "colath"   "colcom"   "colhomo"  "colmil"   "colmslm"
[17] "colrac"   "fefam"    "fehelp"   "fehire"   "fehome"   "fejobaff" "fework"   "grass"
[25] "gunlaw"   "helpblk"  "helppoor" "helpsick" "hitbeatr" "hitchild" "hitdrunk" "hitmarch"
[33] "hitok"    "hitrobbr" "homosex"  "hubbywk1" "letdie1"  "libath"   "libcom"   "libhomo"
[41] "libmil"   "librac"   "marasian" "marblk"   "marhisp"  "marhomo"  "marwht"   "obeylaw"
[49] "pillok"   "polabuse" "polattak" "polescap" "polhitok" "polmurdr" "pornlaw"  "premarsx"
[57] "racchng"  "racfew"   "rachaf"   "racmar"   "racmost"  "racopen"  "racpush"  "racseg"
[65] "sexeduc"  "spanking" "spkath"   "spkcom"   "spkhomo"  "spkmil"   "spkmslm"  "spkrac"
[73] "suicide1" "suicide2" "suicide3" "suicide4" "teensex"  "twoincs1" "wirtap"   "wrkwayup"
[81] "xmarsex"
```

Resources for learning the basics

RStudio Primers

https://rstudio.cloud/learn/primers