UNIVERSITY OF CAMBRIDGE
Bioinformatics Training

Introduction to
for Biologists

Alison Horst
Artwork!

Gita Yadav

Introductions

Tutors

Participants Intro Document

Links- Course HandBook

Login to the OTE?

This course

- Aimed for **beginners** who never used R before.
  - Basics of R: Not Everything!
  - Tricks
  - How to get started
  - How you can learn more and improve yourself
  - Where to Find Help
- After this course you should be able to
  - Manipulate data and visualise it
  - Further your knowledge of R independently.
- As always! You need to practice frequently to get better in R

Time Table

Day 1	Day 2
Getting started	Data manipulation and visualisation with tidyverse
Introduction to R	
Starting with data	

RECOMMENDATION:

Use TWO screens to participate in this course!

One to Watch us

One for DIY!

SETUP INSTRUCTIONS

All the software required to run this course is installed on your OTE, so you do need to do anything.

In case you are not using the OTE, follow the instructions in COURSE HANDBOOK!

- **R** and **RStudio** are separate downloads and installations.
- R is the underlying statistical computing environment, but using R alone is no fun.
- RStudio is a graphical integrated development environment (IDE) that makes using R much easier and more interactive.
- You need to install R before you install RStudio.
- After installing both programs, you will need to install the **tidyverse** package from within RStudio.

SETUP INSTRUCTIONS

Dear Participant,

To take part in the exercise sessions during this R course you'll need to log into our training environment using the remote desktop interface provided. Click on the links below and enter the credentials given here to start. We'll go into further details during the course.

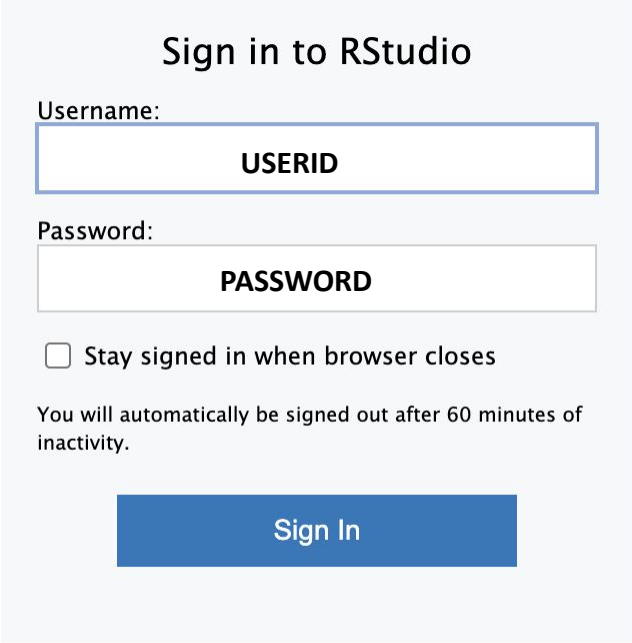
Please remember that we advise using **google chrome** and in particular avoiding the Safari browser for these exercises.

username='USERID" password= PASSWORD

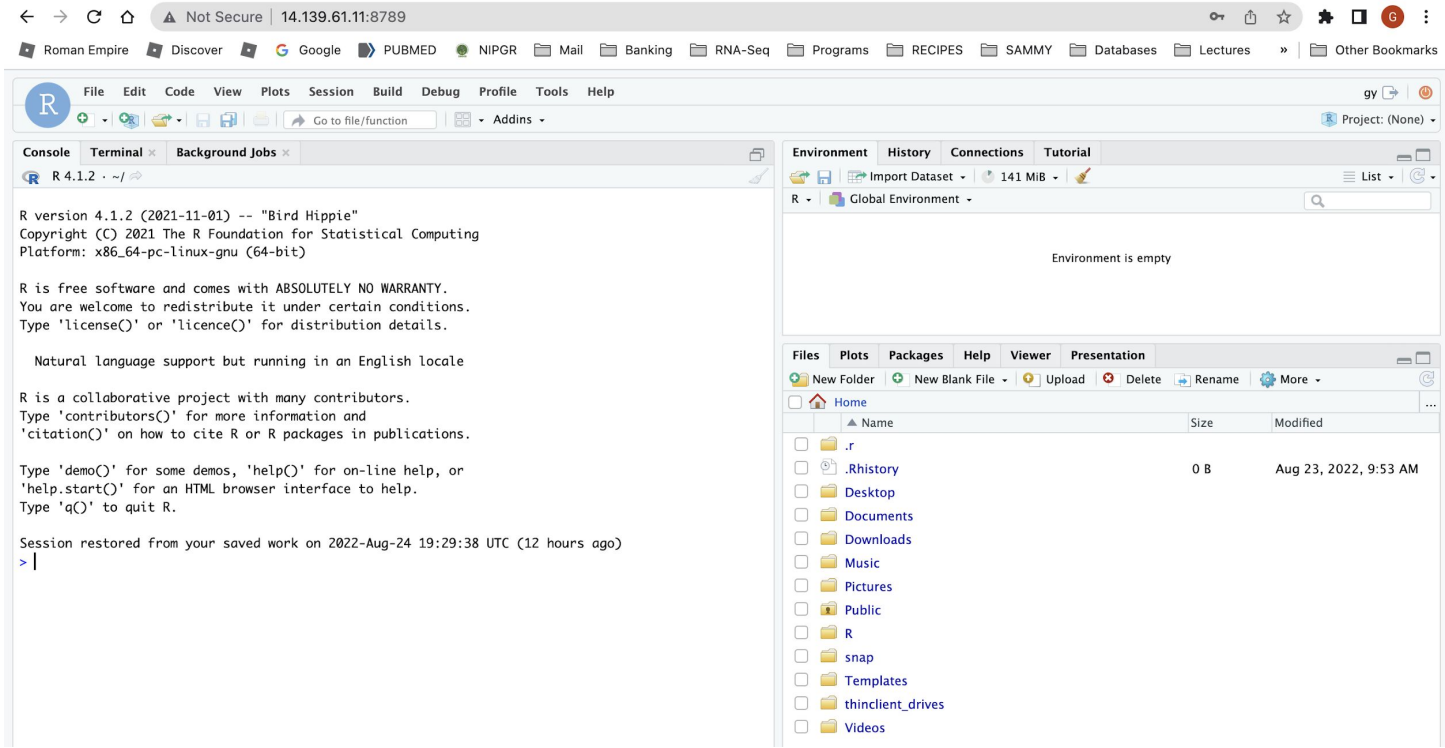
Remote Desktop Link

CLICK THIS LINK!

SETUP INSTRUCTIONS



SETUP INSTRUCTIONS



R is a statistical programming language.
It is very popular in the Data Science field, including Bioinformatics.
The term “R” is used to refer to both the programming language and the software that interprets the scripts written using

Course Materials and Links

COURSE DATA

> Ecology. 2016 Apr;97(4):1082. doi: 10.1890/15-2115.1.

Long-term monitoring and experimental manipulation of a Chihuahuan desert ecosystem near Portal, Arizona (1977-2013)

S K Morgan Ernest [1 2], Glenda M Yenni [1 2], Ginger Allington [3], Erica M Christensen [1 2], Keith Geluso [4], Jacob R Goheen [5], Michelle R Schutzenhofer [6], Sarah R Supp [7], Katherine M Thibault [8], James H Brown [9], Thomas J Valone [10]

Affiliations + expand

PMID: 28792597 DOI: 10.1890/15-2115.1

Free article

- Data is from the paper

S. K. Morgan Ernest, Thomas J. Valone, and James H. Brown. 2009. Long-term monitoring and experimental manipulation of a Chihuahuan Desert ecosystem near Portal, Arizona, USA. Ecology 90:1708.

- A simplified version of this data will be used in this course:

https://doi.org/10.6084/m9.figshare.1314459

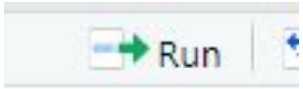
We will download this directly from R during the lessons when we need them.

This course has been created by adapting and adding new content to the Data Carpentry lesson in Ecology: 10.5281/zenodo.3264888

Why learn R?

- Run an R script with **one click/command.**

vs



Rscript <name of file>

Why learn R?

- R promotes **Reproducibility**

Reproducibility is when someone else (including your future self) can obtain the same results from the same dataset when using the same analysis.

- Automating your analysis.
- Generate reports.

Why learn R?

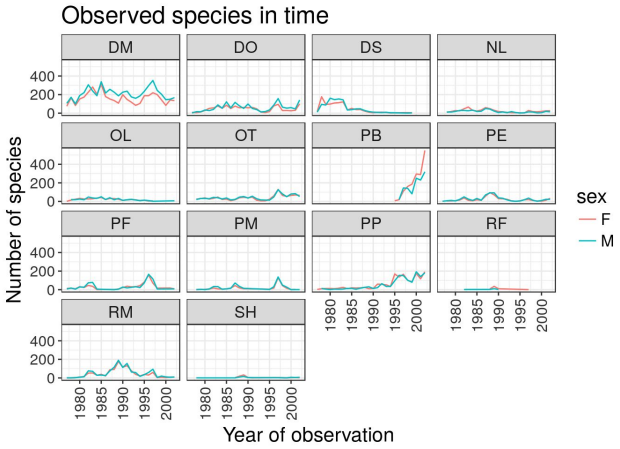
- R is **interdisciplinary & Extensible!**
  - Over 10,000 packages that cover different fields
    - Bioconductor repository for bioinformatics packages (GENOMICS)
    - Image Analysis
    - GIS
    - Time Series
    - Population genetics
    - Machine Learning
      - and a lot more!

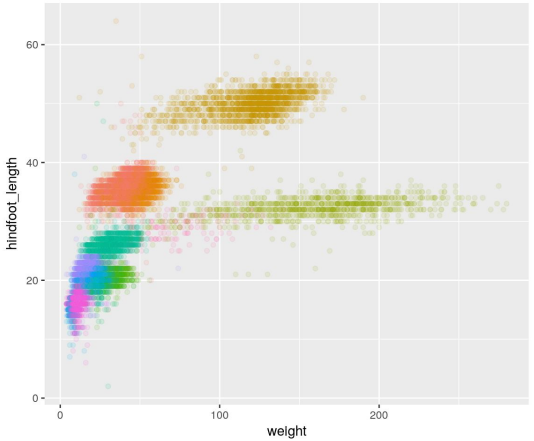
Why learn R?

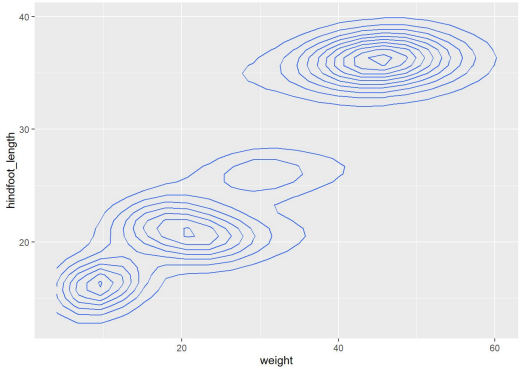
- R works on data of different sizes (100s to Millions of Lines!)
- The skills you learn with R scale easily with the size of your dataset.
- Designed for data analysis: Special data structures and data types that make handling of missing data and statistical factors convenient.
- Seamless connection to spreadsheets, databases, and many other data formats, on your **computer** or on the **web**.

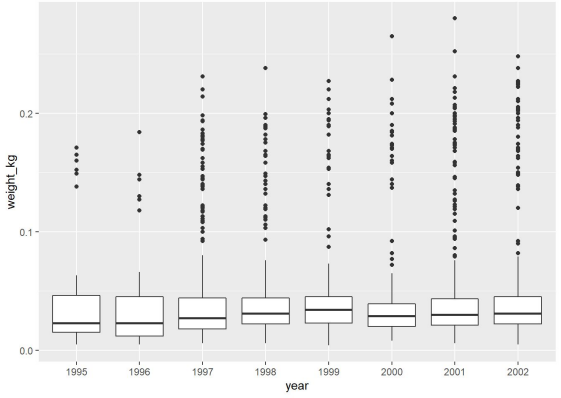
Why learn R?

- Best Feature: High-quality graphics!





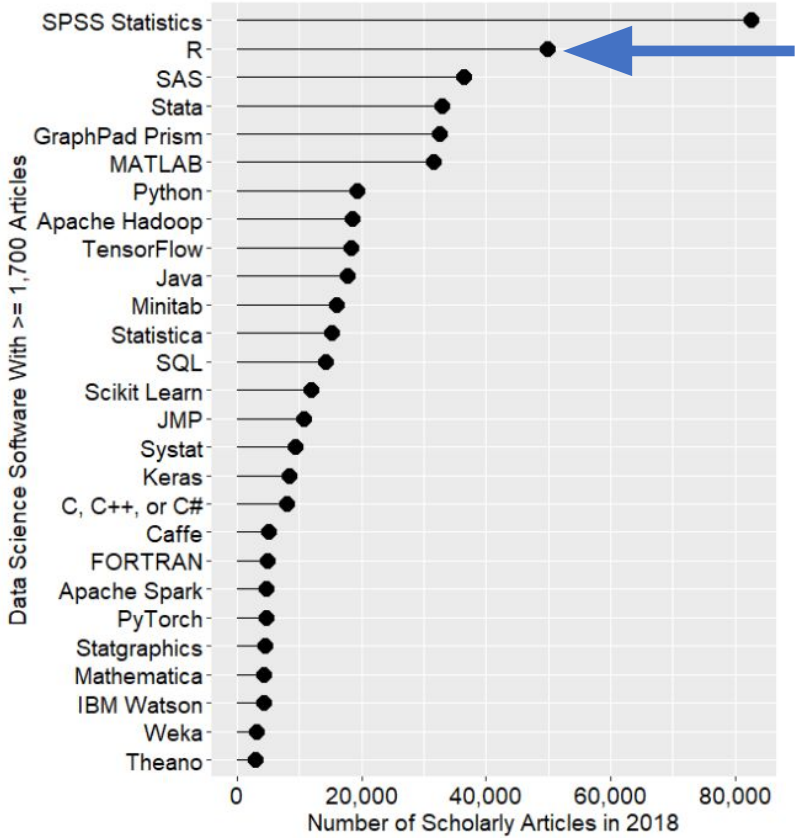




Why learn R?

- R has a large and welcoming community
  - RStudio community
  - Local R User Groups (meetup)
  - R-Ladies
  - R-bloggers
  - The Carpentries
  - Stack Overflow
- Popular in Data Science

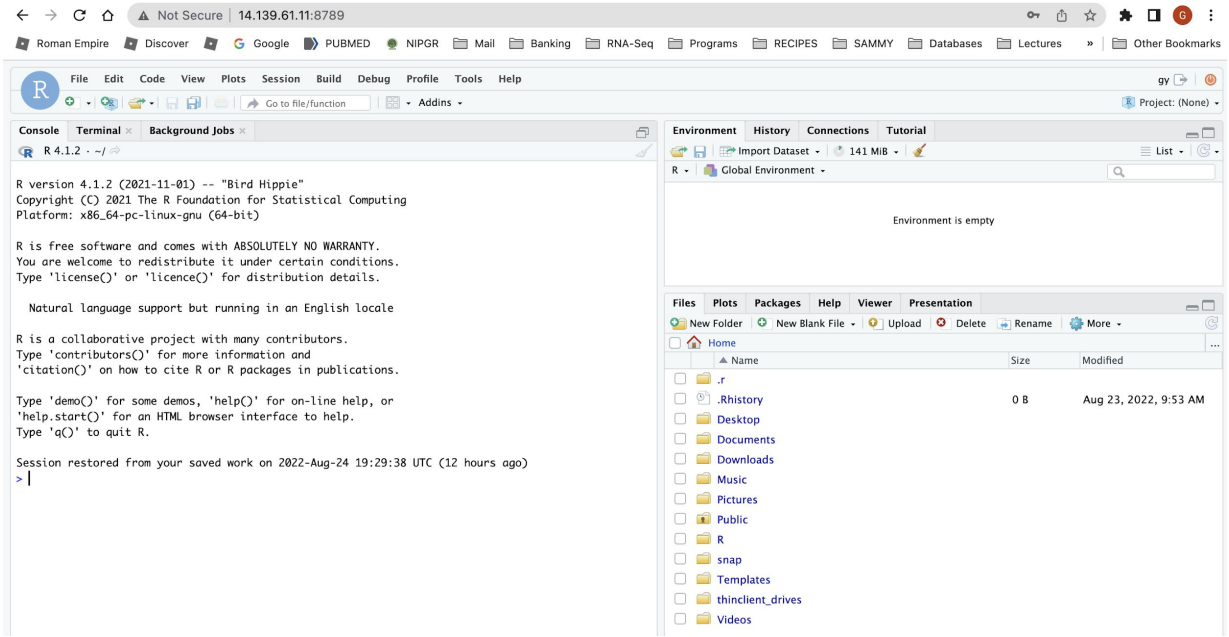
R tops other programming languages in academia



Why learn R?

- Free
- Open-source
- Cross-platform

RStudio


Not Secure | 14.139.61.11:8789
Roman Empire
Discover
Google
PUBMED
NIPGR
Mail
Banking
RNA-Seq
Programs
RECIPES
SAMMY
Databases
Lectures
Other Bookmarks
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Console
Terminal
Background Jobs
R 4.1.2 · ~/
R version 4.1.2 (2021-11-01) -- "Bird Hippie"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (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.
Session restored from your saved work on 2022-Aug-24 19:29:38 UTC (12 hours ago)
>
Environment
History
Connections
Tutorial
Import Dataset
141 MiB
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Presentation
New Folder
New Blank File
Upload
Delete
Rename
More
Home
Name
Size
Modified
.r
.Rhistory
0 B
Aug 23, 2022, 9:53 AM
Desktop
Documents
Downloads
Music
Pictures
Public
R
snap
Templates
thinclient_drives
Videos

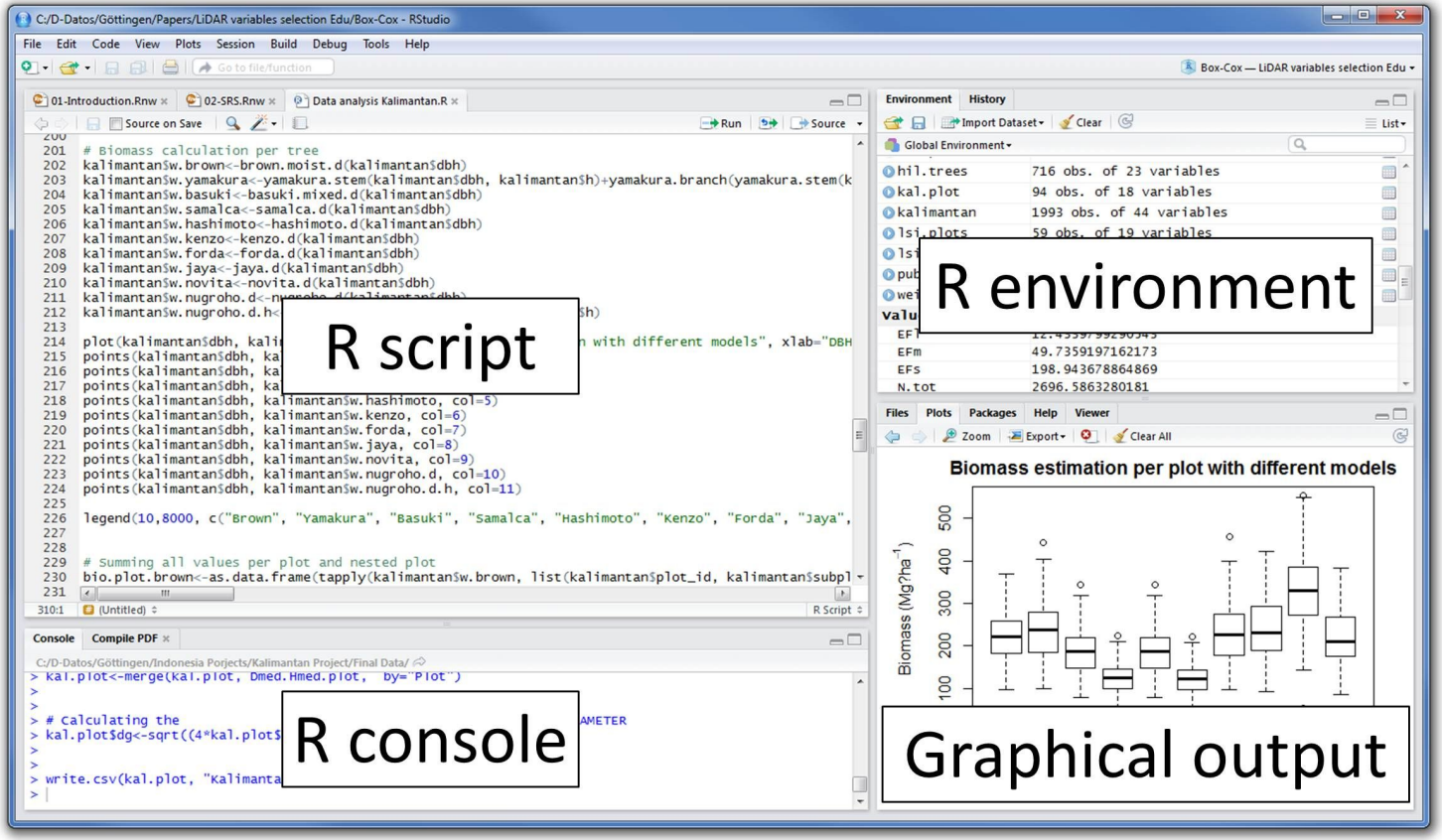
Rstudio IDE : Integrated Development Environment

- An IDE is an application used by software developers that faciliates programming by offering source code editing, building and debugging tools all integarated into one application.
- To function correctly, RStudio needs R and therefore both need to be installed on your computer.
- The RStudio Desktop open-source product is free under the Affero General Public License (AGPL) v3. Other versions of RStudio are also available.
- We will use RStudio IDE to:
  - write code,
  - navigate files on our computer,
  - inspect the variables we are going to create,
  - visualize the plots we will generate.
- RStudio can also be used for other things (*e.g.,* version control, developing packages, writing Shiny apps) that we will not cover during the workshop.

RStudio

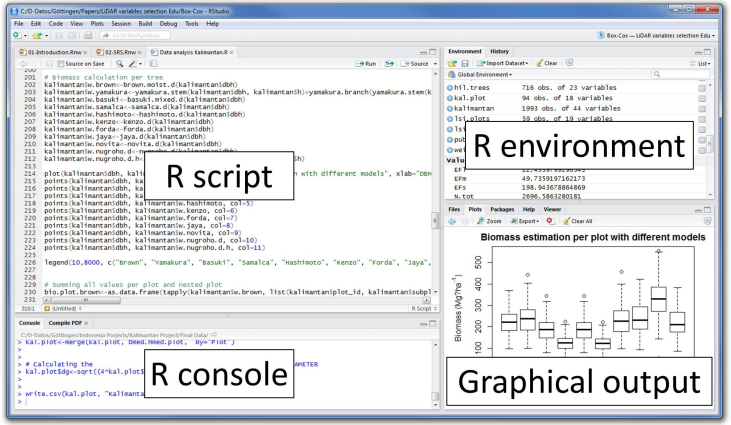
Let us start the action!

Rstudio IDE


C:/D-Datos/Göttingen/Papers/LiDAR variables selection Edu/Box-Cox - RStudio
File Edit Code View Plots Session Build Debug Tools Help
01-Introduction.Rnw
02-SRS.Rnw
Data analysis Kalimantan.R
Source on Save
Run
Source
Biomass calculation per tree
kalimantan$w.brown<-brown.moist.d(kalimantan$dbh)
kalimantan$w.yamakura<-yamakura.stem(kalimantan$dbh, kalimantan$h)+yamakura.branch(yamakura.stem(k
kalimantan$w.basuki<-basuki.mixed.d(kalimantan$dbh)
kalimantan$w.samalca<-samalca.d(kalimantan$dbh)
kalimantan$w.hashimoto<-hashimoto.d(kalimantan$dbh)
kalimantan$w.kenzo<-kenzo.d(kalimantan$dbh)
kalimantan$w.forda<-forda.d(kalimantan$dbh)
kalimantan$w.jaya<-jaya.d(kalimantan$dbh)
kalimantan$w.novita<-novita.d(kalimantan$dbh)
kalimantan$w.nugroho.d<-nugroho.d(kalimantan$dbh)
kalimantan$w.nugroho.d.h<
plot(kalimantan$dbh, kali
points(kalimantan$dbh, ka
points(kalimantan$dbh, ka
points(kalimantan$dbh, ka
points(kalimantan$dbh, kalimantan$w.hashimoto, col=5)
points(kalimantan$dbh, kalimantan$w.kenzo, col=6)
points(kalimantan$dbh, kalimantan$w.forda, col=7)
points(kalimantan$dbh, kalimantan$w.jaya, col=8)
points(kalimantan$dbh, kalimantan$w.novita, col=9)
points(kalimantan$dbh, kalimantan$w.nugroho.d, col=10)
points(kalimantan$dbh, kalimantan$w.nugroho.d.h, col=11)
legend(10,8000, c("Brown", "Yamakura", "Basuki", "Samalca", "Hashimoto", "Kenzo", "Forda", "Jaya",
# Summing all values per plot and nested plot
bio.plot.brown<-as.data.frame(tapply(kalimantan$w.brown, list(kalimantan$plot_id, kalimantan$subpl
(Untitled)
R Script
R script
Console
Compile PDF
C:/D-Datos/Göttingen/Indonesia Porjects/Kalimantan Project/Final Data/
> kal.plot<-merge(kal.plot, Dmed.Hmed.plot, by="Plot")
> # Calculating the
> kal.plot$dg<-sqrt((4*kal.plot$
> write.csv(kal.plot, "Kalimanta
R console
Environment
History
Import Dataset
Clear
List
Global Environment
hil.trees 716 obs. of 23 variables
kal.plot 94 obs. of 18 variables
kalimantan 1993 obs. of 44 variables
lsi.plots 59 obs. of 19 variables
EFm 49.7359197162173
EFs 198.943678864869
N.tot 2696.5863280181
R environment
Files
Plots
Packages
Help
Viewer
Zoom
Export
Clear All
Biomass estimation per plot with different models
Biomass (Mg?ha-1)
100
200
300
400
500
Graphical output

Rstudio IDE

- RStudio interface screenshot.
- Clockwise from top left: Source, Environment/History, Files/Plots/Packages/Help/Viewer, Console.
- RStudio is divided into 4 “Panes”
- The **Source/Script** for your scripts and documents (top-left, in the default layout),
- Your **Environment/History** (top-right),
- Your **Files/Plots/Packages/Help/Viewer** (bottom-right)
- R **Console** (bottom-left). The placement of these panes and their content can be customized (see menu, Tools -> Global Options -> Pane Layout).
- **Advantages** :
  - All information you need to write code is available in a single window.
  - Many shortcuts, autocompletion, and highlighting
  - RStudio makes typing easier and less error-prone



INTERACTING WITH R!

1. **Write down instructions for the computer to follow (CODE!)**

2. **Tell the computer to follow those instructions. (EXECUTE/RUN THE CODE!)**

We write, or code, instructions in R because it is a common language that both the computer and we can understand!

INTERACTING WITH R!

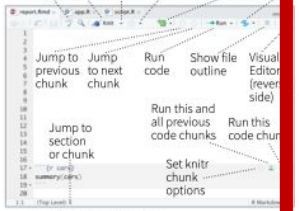
- **There are two main ways of interacting with R:**
  - **By using the Console**
  - **By using Script files (plain text files that contain your code).**

ALL R STUDIO CHEAT SHEETS ARE HERE :
https://www.rstudio.com/resources/cheatsheets/

RStudio IDE : : CHEAT SHEET

Documents and Apps

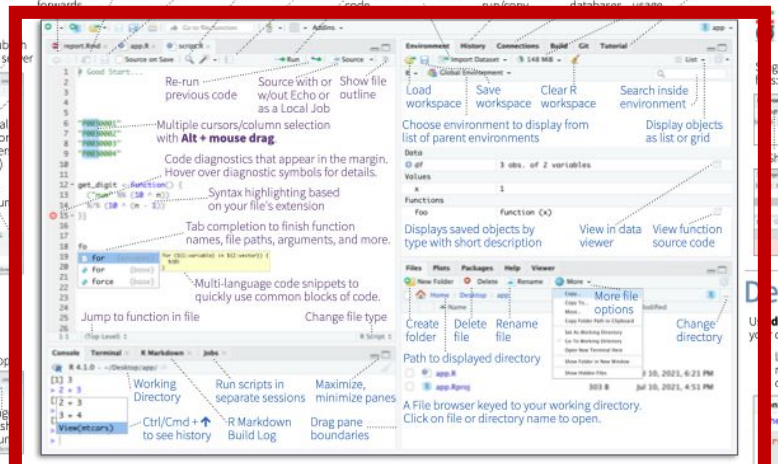
Open Shiny, R Markdown, knitr, Sweave, LaTeX, .Rd files and more in Source Pane



Access markdown guide at **Help > Markdown Quick Reference**
See reverse side for more on **Visual Editor**

RStudio recognizes that files named **app.R**, **server.R**, **ui.R**, and **global.R** belong to a shiny app

Source Editor



Tab Panes

Import data with wizard
History of past commands to run/copy
Manage external databases
View memory usage
R tutorials

Version Control

Turn on at **Tools > Project Options > Git/SVN**

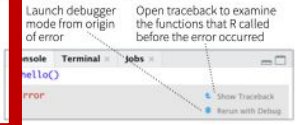
A Added
M Modified
? Untracked
D Deleted
R Renamed

Stage files:
Commit staged files
Push/Pull to remote
View History
Current branch

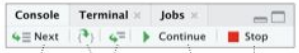
Show file diff to view file differences

Debug Mode

Use **debug()**, **browser()**, or a breakpoint and execute your code to open the debugger mode.



Click next to line number to add/remove a breakpoint.
Highlighted line shows where execution has paused



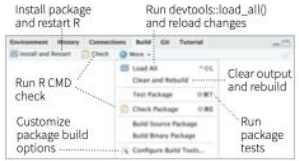
Package Development

Create a new package with **File > New Project > New Directory > R Package**

Enable roxygen documentation with **Tools > Project Options > Build Tools**

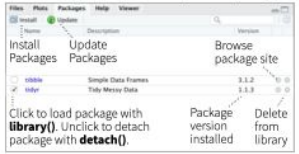
Roxygen guide at **Help > Roxygen Quick Reference**

See package information in the **Build Tab**

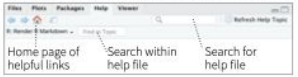


RStudio opens plots in a dedicated **Plots** pane

GUI **Package** manager lists every installed package



RStudio opens documentation in a dedicated **Help** pane



Viewer pane displays HTML content, such as Shiny apps, RMarkdown reports, and interactive visualizations

View(<data>) opens spreadsheet like view of data set

RStudio

Source Editor

Navigate backwards/ forwards

Open in new window

Save

Find and replace

Compile as notebook

Run selected code

Re-run previous code

Source with or w/out Echo or as a Local Job

Show file outline

Multiple cursors/column selection with **Alt + mouse drag**.

Code diagnostics that appear in the margin. Hover over diagnostic symbols for details.

Syntax highlighting based on your file's extension

Tab completion to finish function names, file paths, arguments, and more.

Multi-language code snippets to quickly use common blocks of code.

Jump to function in file

Change file type

Working Directory

Run scripts in separate sessions

Maximize, minimize panes

Ctrl/Cmd + ↑ to see history

R Markdown Build Log

Drag pane boundaries

Tab Panes

Import data with wizard

History of past commands to run/copy

Manage external databases

View memory usage

R tutorials

Load workspace

Save workspace

Clear R workspace

Search inside environment

Choose environment to display from list of parent environments

Display objects as list or grid

Displays saved objects by type with short description

View in data viewer

View function source code

Create folder

Delete file

Rename file

More file options

Change directory

Path to displayed directory

A File browser keyed to your working directory. Click on file or directory name to open.

Creating an R Project

- Before starting to write code in RStudio, we need to create an R Project.
- The idea behind an R project is to have a workspace where you can keep all the files and settings associated with the project together.

To create an "R Project":

1. Start RStudio.
2. Under the `File` menu, click on `New Project`. Choose `New Directory`, then `New Project`.
3. Enter a name for this new folder (or "directory"), and choose a convenient location for it. This will be your **working directory** for the rest of the day (*e.g.,* `~/my-first-project`).
4. Click on `Create Project`.
5. (Optional) Set Preferences to 'Never' save workspace in RStudio.

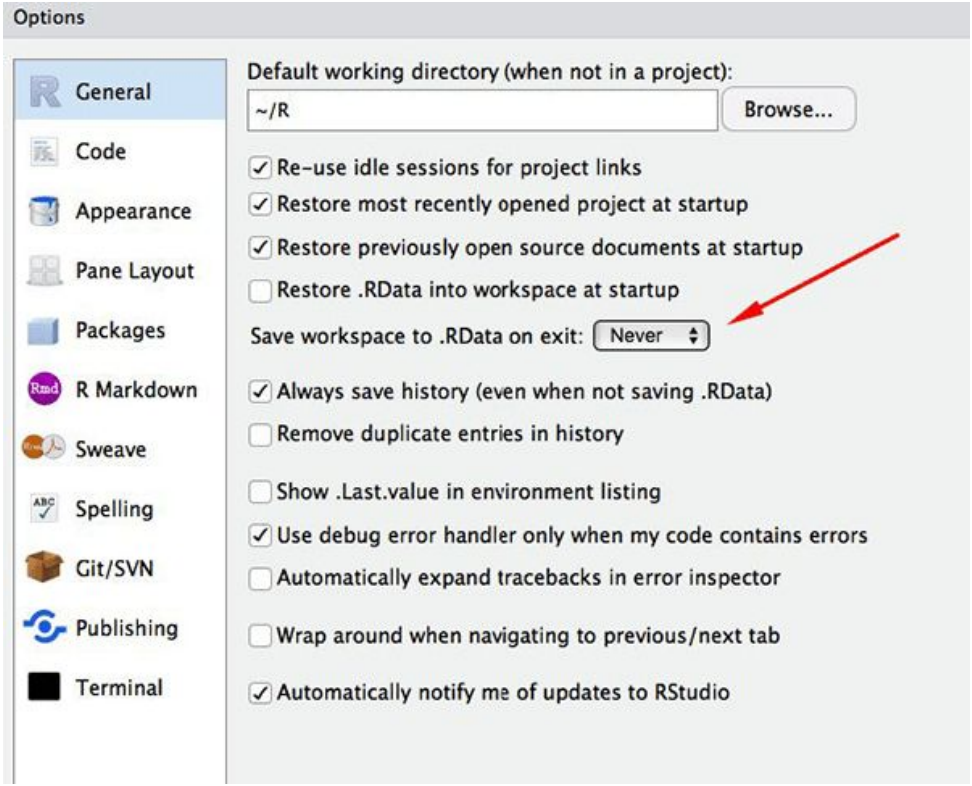
- Next time you open the R Project it would be easier to resume work.

Some Global Options!

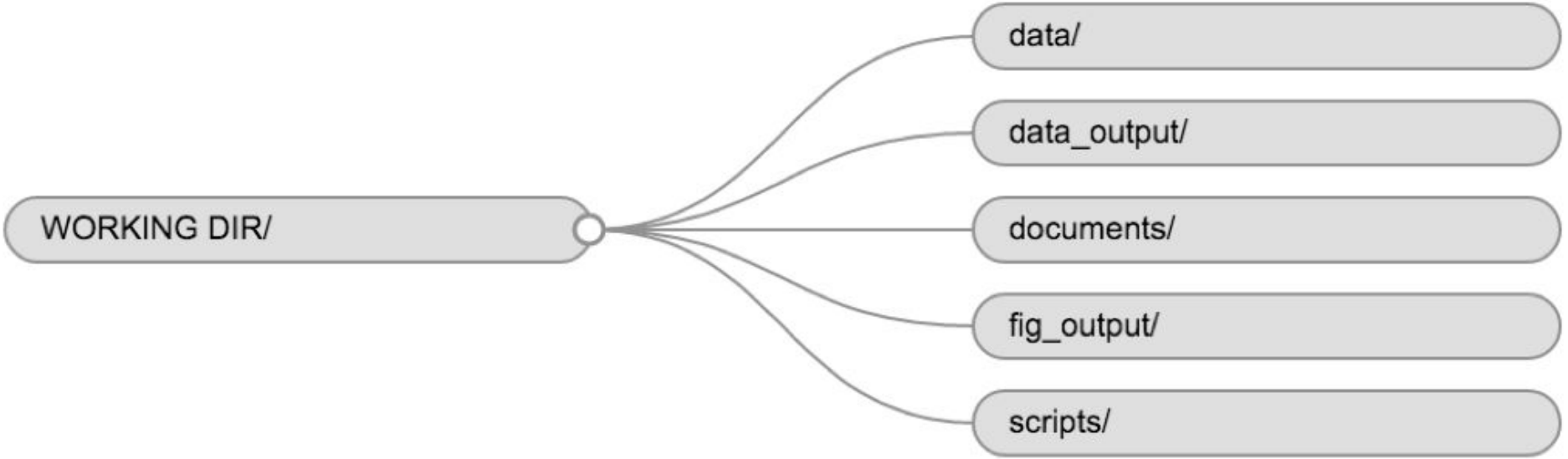
- RStudio's default preferences generally work well, but saving a workspace to .RData can be cumbersome, especially if you are working with larger datasets as this would save all the data that is loaded into R into the .RData file.

- To turn that off,
  - go to **Tools** –> Global Options
  - Select the 'Never' option for

  Save workspace to .RData' on exit.



Working Directory



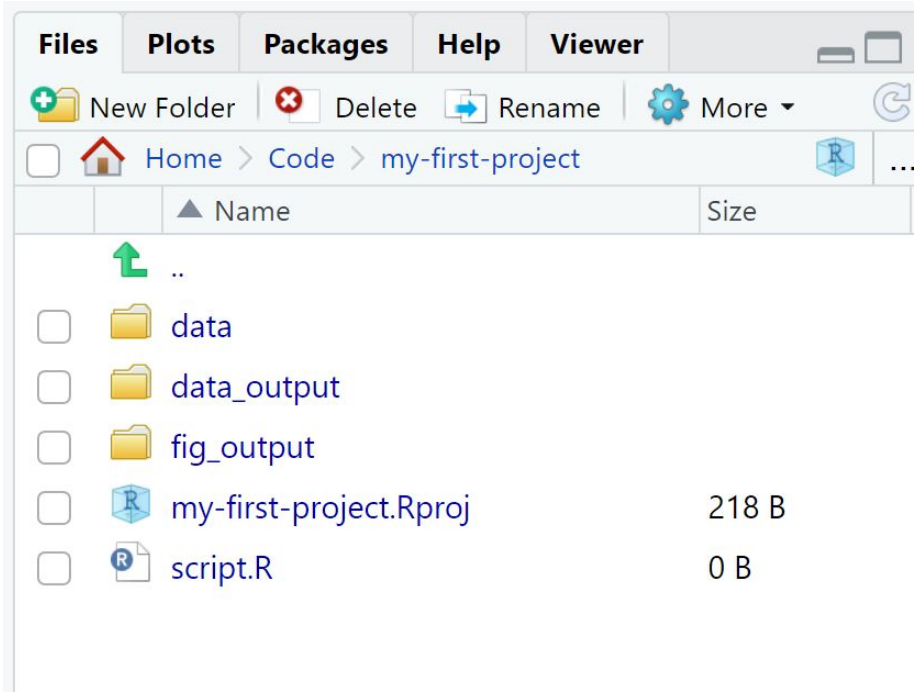
Your Working Directory

- **For this workshop, we will need a data/ folder to store our raw data,**
- **We will use data_output/ for when we learn how to export data as CSV files**
- **We will also create a fig_output/ folder for the figures that we will save**

- Under the `Files` tab on the right of the screen, click on `New Folder` and create a folder named `data` within your newly created working directory (e.g., `~/my-first-project/data`). Repeat these operations to create a `data_output/` and a `fig_output` folders.

- **We are going to keep the script in the root of our working directory because we are only going to use one file and it will make things easier.**

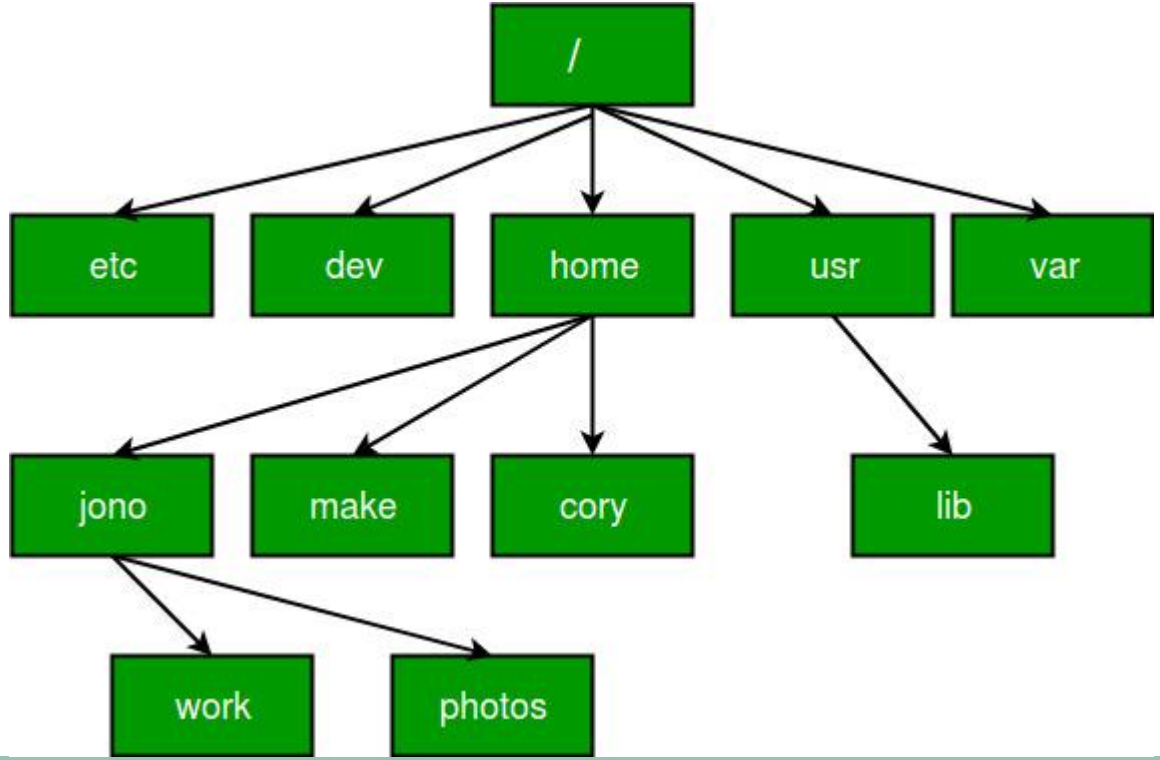
Your Working Directory



Computing Environments

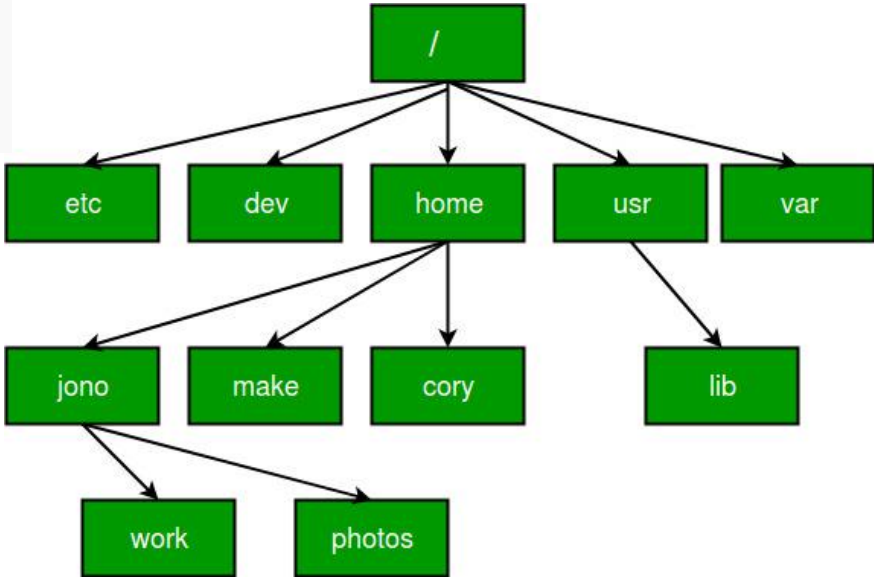
Concept of Working Directory

- **The working directory is an important concept to understand.**
- **It is the place from where R will be looking for and saving the files.**
- **When you write code for your project, it should refer to files in relation to the root of your working directory and only need files within this structure.**



WORKING DIRECTORY

*An **absolute path** is defined as specifying the location of a file or directory from the root directory (/). In other words, we can say that an absolute path is a complete path from start of actual file system from / directory.*



➢ **Working this way makes it a lot easier to move your project around on your computer and share it with others without worrying about whether or not the underlying scripts will still work!**

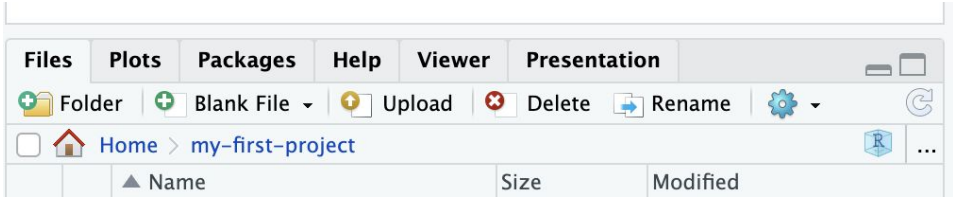
WHAT IS MY WORKING DIRECTORY?!

- **Using RStudio projects makes it easy to organise your files in the project and ensures that your working directory is set properly.**

- **RStudio shows your current working directory at the top of your window:**

WHAT IS MY WORKING DIRECTORY?!

- Another way to check your WD is
  - by typing getwd() in the console pane.
  - TRY IT NOW!



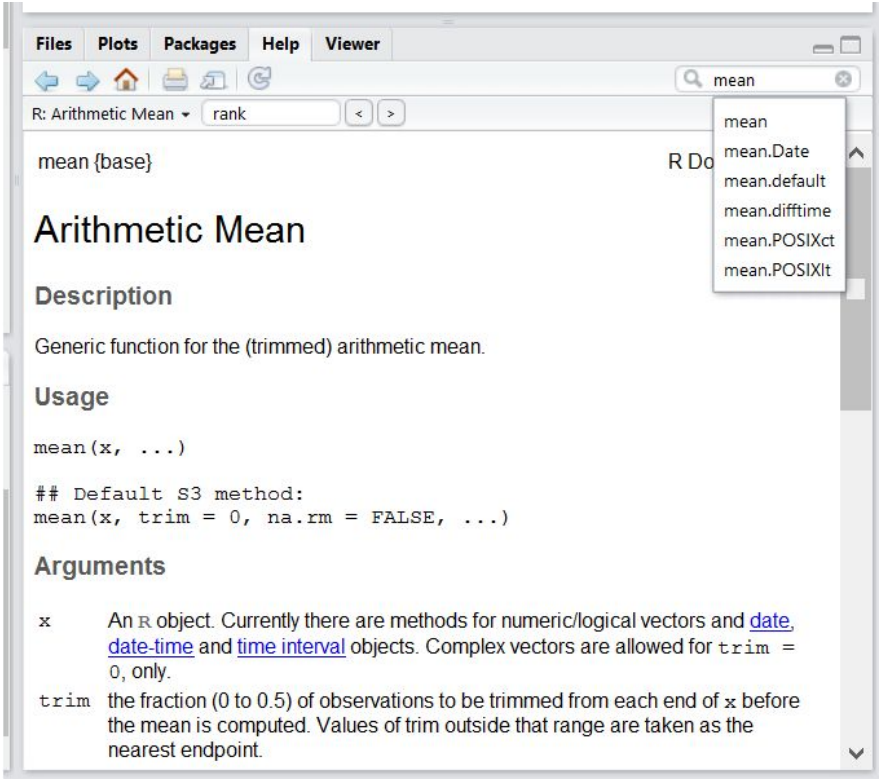
- If for some reason your working directory is not what it should be, you can change it in the RStudio interface by navigating in the file browser where your working directory should be, and clicking on the blue gear icon “More”, and select “Set As Working Directory”.
- Alternatively you can use setwd("/path/to/working/directory") to reset your working directory (not recommended). However, your scripts should not include this line because it will fail on someone else’s computer.

SEEKING HELP

RSTUDIO HELP INTERFACE

Use this to search for more information on R functions

TRY IT NOW!



SEEKING HELP

I know the name of the function I want to use, but I'm not sure how to use it

If you need help with a specific function, let's say `barplot()`, you can type:

```
?barplot
```

TRY IT NOW!

SEEKING HELP

I know the name of the function but I don't know how many / What inputs it requires to run?

If you just need to remind yourself of the names of the arguments, you can use:

```
args(lm)
```

TRY IT NOW!

SEEKING HELP

I want to use a function that does X, there must be a function for it but I don't know which one…

If you are looking for a function to do a particular task, you can use the `help.search()` function, which is called by the double question mark `??`. However, this only looks through the installed packages for help pages with a match to your search request

```
??kruskal
```

SEEKING HELP

I am stuck… I get an error message that I don't understand

Start by googling the error message. However, this doesn't always work very well because often, package developers rely on the error catching provided by R. You end up with general error messages that might not be very helpful to diagnose a problem (e.g. "subscript out of bounds"). If the message is very generic, you might also include the name of the function or package you're using in your query.

Asking for help

The key to receiving help from someone is for them to rapidly grasp your problem. You should make it as easy as possible to pinpoint where the issue might be.

PROGRAMMING IN R!

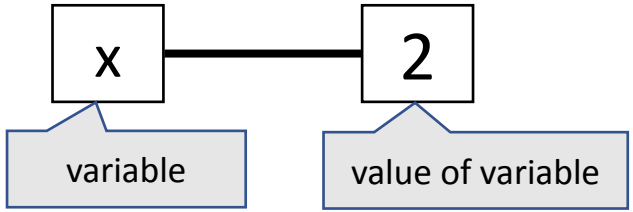
CREATING OBJECTS IN R!

CREATING OBJECTS IN R!

- **Type Math in the Console?**
  - **Fun- but not useful!**
- **Useful Stuff gets done when you can**
  - **Assign Values to Objects!**

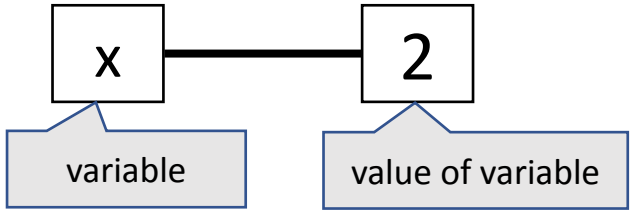
Variables

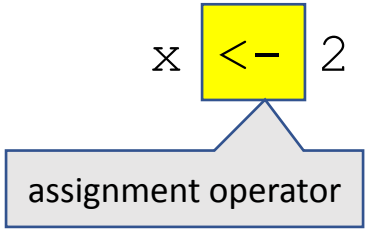
A variable is a letter or word that stores a value in it.



Variables

A variable is a letter or word that stores a value in it.





Variable names

- Case-sensitive
- Do not start with numbers
- Not too long
- Some names cannot be used as they are reserved

  e.g. if <- 66 ✗
- Do not use function names
- Use nouns if using a word to represent a variable

  e.g. gene_id <- 12345 ✓
- Be consistent - style guide

A SMALL EXERCISE!

Challenge

What are the values after each statement in the following?

```
mass <- 47.5            # mass?
age  <- 122             # age?
mass <- mass * 2.0      # mass?
age  <- age - 20        # age?
mass_index <- mass/age  # mass_index?
```

A SMALL EXERCISE!

▾ Answer

```
mass <- 47.5            # mass is 47.5
age  <- 122             # age is 122
mass <- mass * 2.0      # mass is 95
age  <- age - 20        # age is 102
mass_index <- mass/age  # mass_index is 0.931
```

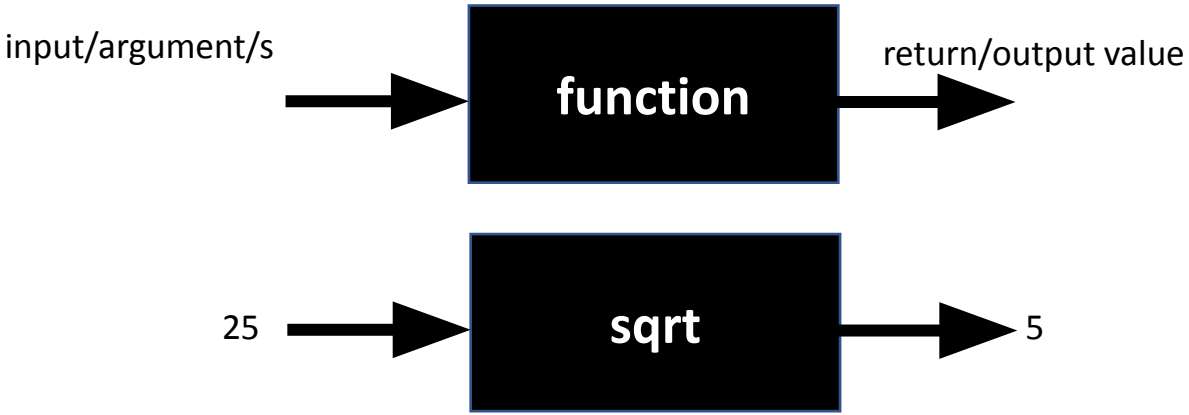
FUNCTIONS & THEIR ARGUMENTS

Calling functions

Functions execute a defined set of commands – automate a process

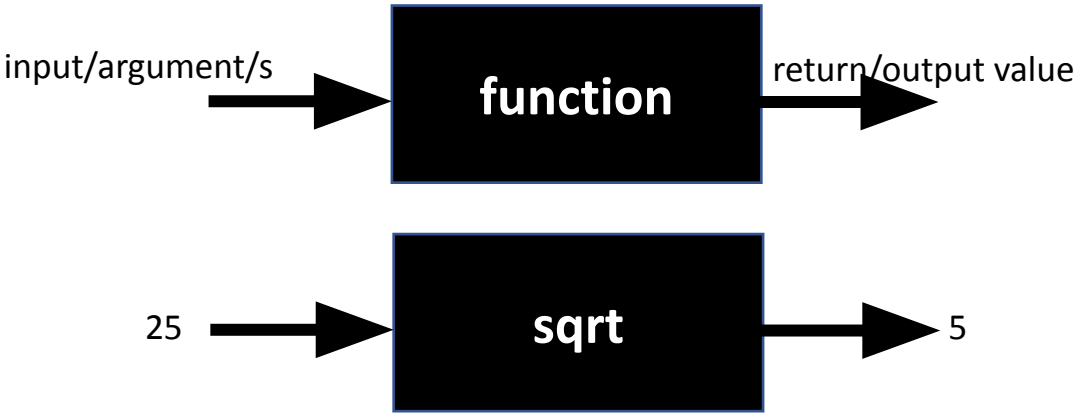
Calling functions

Functions execute a defined set of commands – automate a process



Calling functions

Functions execute a defined set of commands – automate a process




```
sqrt(25)
```

VECTORS & DATA TYPES

Data types

- **logical**: `TRUE FALSE`
- **integer**: whole numbers. *e.g.,* `40`
- **double**: numbers with decimal points. *e.g.,* `2.666`
- **character**: words or **strings**. *e.g.,* `"Hello"`

Data structures

- **vector**: list of items of the same data type. *e.g.*, 4, 6, 9, 12
- **factor**: categorical data (has to be a character vector). *e.g.*, Male, Female
- **data.frame**: contains tabular data – normally data is loaded into data.frame when reading in a file

Vector

- List of data types (must be same type)
- One-dimensional


```
c(4,12,7,9)
```

SUBSETTING VECTORS

Vector indices

1	2	3	4
4	12	7	9

CHALLENEGE

Challenge

1. Create a vector `height_mm` that contains the following heights of animals 100,150,99,87 in mm respectively.

▸ Answer

2. What is the total height of these animals in mm? (*Hint*: use function `sum`)

▸ Answer

3. Another animal was measured and its height is 220mm. Add this to the beginning of the vector `height_mm` and save the total height of the new set of animals in the `total_height` variable. What is the new total_height?

▸ Answer

FUN QUIZ!

- What is temperature?

```
temperature <- 26.789
round (x = temperature, digits = 1)
```

A. Variable

B. Function

C. Place holder

(Poll Question – Choose the right option!)

FUN QUIZ !

- What is the value of temperature after executing both lines of code?

```
temperature <- 26.789
round (x = temperature, digits = 1)
```

A. 27
B. 26.7
C. 26.789
D. 26.8

(Poll Question – Choose the right option!)

CONDITIONAL SUBSETTING

Relational Operators

Used to compute a condition/comparisons:

==	equal
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
!=	not equal to

Logical operators

& and

| or

! not

AND &

- TRUE & TRUE results in TRUE
- TRUE & FALSE results in FALSE
- FALSE & TRUE results in FALSE
- FALSE & FALSE results in FALSE

OR |

- TRUE | TRUE results in TRUE
- TRUE | FALSE results in TRUE
- FALSE | TRUE results in TRUE
- FALSE | FALSE results in FALSE

NOT !

- !TRUE results in FALSE
- !FALSE results in TRUE

HANDLING MISSING DATA

AN EXERCISE

Challenge

1. Using this vector of heights in inches, create a new vector, `heights_no_na`, with the NAs removed.

```
heights <- c(63, 69, 60, 65, NA, 68, 61, 70, 61, 59, 64, 69, 63, 63, NA,
```

2. Use the function `median()` to calculate the median of the `heights` vector.
3. How many people in the set are taller than 67 inches.

▸ Answer

COPY FROM CHAT BOX!: heights <- c(63, 69, 60, 65, NA, 68, 61, 70, 61, 59, 64, 69, 63, 63, NA, 72, 65, 64, 70, 63, 65)

AN EXERCISE

Answer

```
heights <- c(63, 69, 60, 65, NA, 68, 61, 70, 61, 59, 64, 69, 63, 63,

# 1.
heights_no_na <- heights[!is.na(heights)]
# or
heights_no_na <- na.omit(heights)
# or
heights_no_na <- heights[complete.cases(heights)]

# 2.
median(heights, na.rm = TRUE)

# 3.
heights_above_67 <- heights_no_na[heights_no_na > 67]
length(heights_above_67)
```

What we have learned so far

- How to create a Project in RStudio
- How to code and execute R code in RStudio
- Create variables *e.g.,* `weight <- 24`
- Data types: integer (`11`), double (`11.01`), character (`"Hello"`), logical (`TRUE/FALSE`)
- Calling functions *e.g.,* `sqrt(25)`
- Create vectors *e.g.,* `weight_mm <- c(22, 24, 10, 34)`
- Index vectors *e.g.,* `weight_mm[3]`
- Relational and Logical operators `(& | ! == < > <= >= !=)`
- Missing data `NA`

UP NEXT

STARTING WITH REAL DATA!