The Basics of R

Day 1 – Basic Functions

AFS Alaska Workshop

An Introduction to the R Programming Language for Fishery Biologists

February 2022

Instructor: Justin Priest

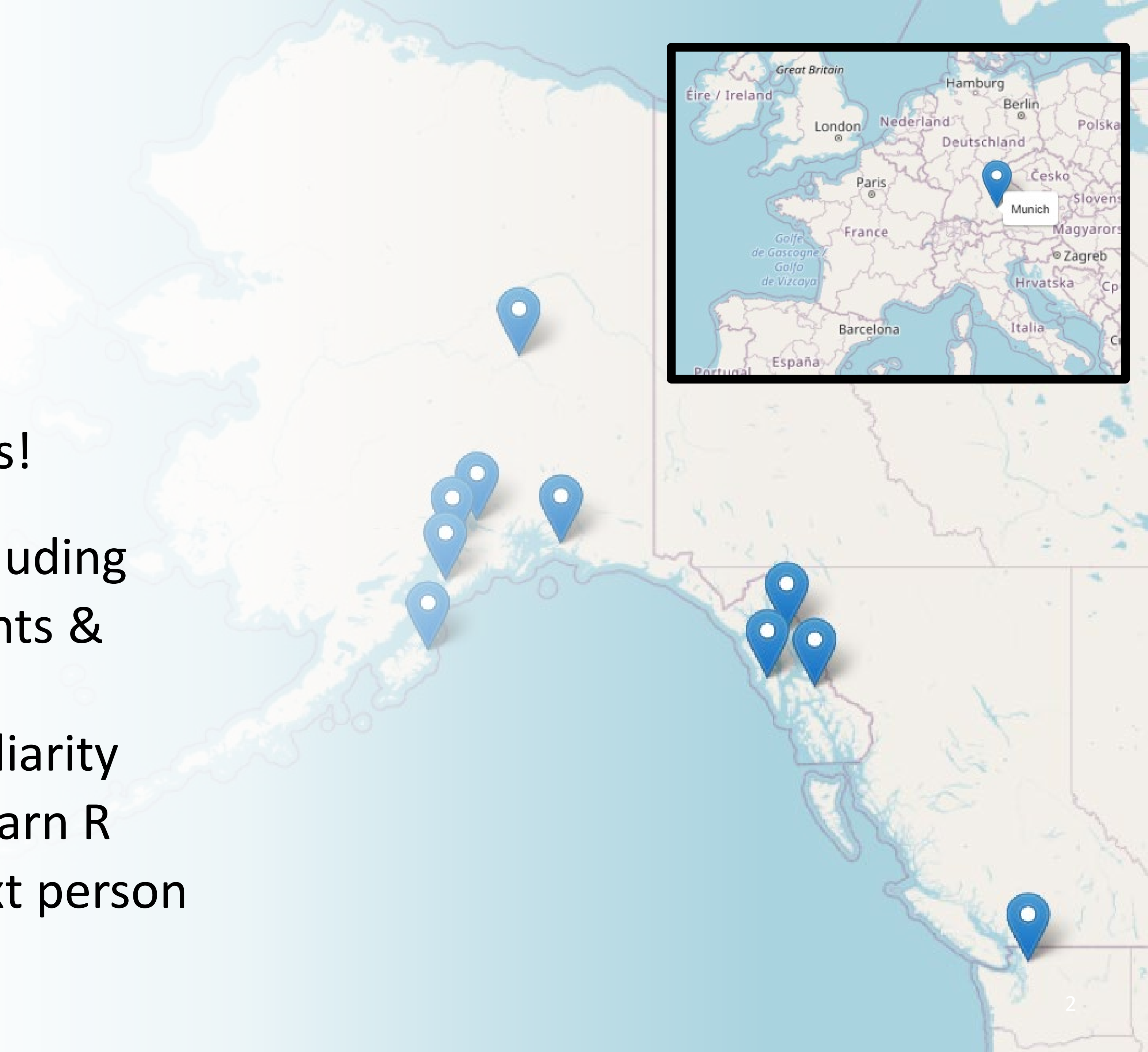
https://github.com/justinpriest/R_Intro_AFS

Introductions

Today we have 15 participants!

Briefly introduce yourself, including

- What types of assessments & figures you do for work
- Your R experience / familiarity
- Why you would like to learn R
- When done, pick the next person



Welcome

What you'll need today:

- Computer with R & RStudio installed
- Files that Justin emailed over

If there is something that is wrong (e.g., broken link) or could be explained better, please let me know and/or write it down.

After the course, there will be a survey to give me feedback on improvements

About Your Instructor

Justin Priest

2020 UAF CFOS Graduate

Learned R tools at CFOS (easy way) & taught myself (hard but fun way)

ADF&G Coho Salmon Researcher in Sitka & Juneau

Today's Agenda

09:30–10:45	1 – About R
10:45–12:00	2 – Basics of Programming
12:00–13:30	Lunch Break
13:30–15:00	3 – Working with your data
15:00–16:30	4 – Basic Data Manipulation
16:30–17:00	Review of Material

Each section will be organized by:

~20–30 min PowerPoint

~20–30 min RStudio demo

~10–20 min "learnr" code practice

Most learnr tutorials will not be able to be completed on time; that's ok! Finish later

Monday

09:00–09:30	Welcome and Review
09:30–11:30	5 – Charts
11:30–12:00	6 – Basics Analysis
12:00–13:30	Lunch
13:30–14:00	7 – Tidyverse
14:00–16:30	8 – Project
16:30–17:00	Review of Projects

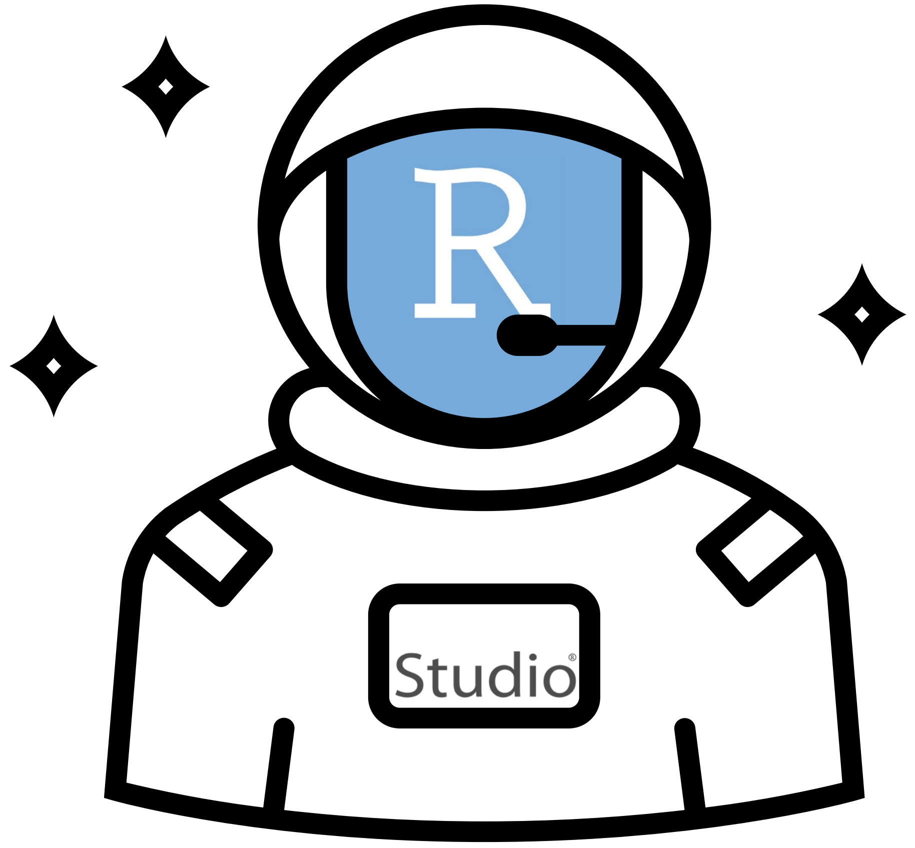
Project!

Throughout today, think of a project that you'll work on Monday, especially whether any issues we'll discuss pertain to your dataset.

Today will unfortunately be a little dry but tomorrow will be very visual!

Expectations

- This will be an intensive but rewarding course
- As such, it requires your **complete** attention
- Close out of your email. Please be fully present



Course Outcomes

- By the end of this class, you will be familiar with the basics of R
  - Hopefully, you'll be familiar enough to read it, and use these slides for future reference
- Don't get bogged down in the details or try to memorize everything, this is a language after all.
- We will go through code together, then you can run through it on your own later
  - Ask questions in the chat or raise hand
  - Answer each others' questions or "like" questions that are tricky

Learning Philosophy

- This is a "failure positive" zone!
  - You can't break anything. Be creative, think like a kid, see what happens when you type things.
- Ask questions! No dumb questions when you're learning a language
- Every mistake is a lesson: Ask me or yourself *why* something didn't work
- Communicate when I'm going too fast

One remote teaching request: if possible, keep video on so I can see your faces

Learning Philosophy

- This is a "failure positive" zone!
  - You can't break anything. Be creative, think like a ki type things.
- Ask questions! No dumb questions when you're
- Every mistake is a lesson: Ask me or yourself w
- Communicate when I'm going too fast

It's OK to fail!

Ask me questions!

Ask *why* it didn't work!

Tell if I'm going fast!

One remote teaching request: if possible, keep video on so I

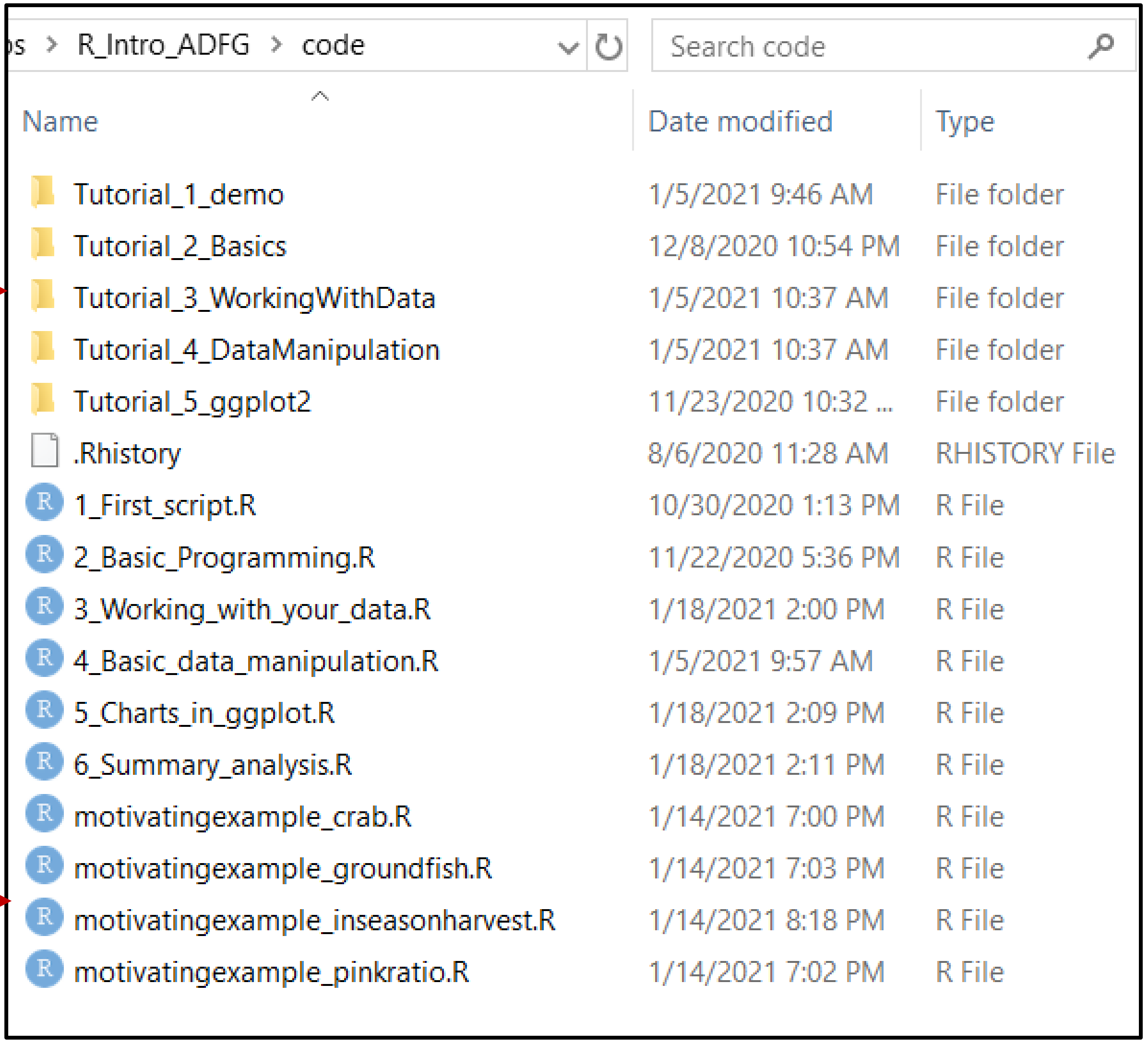
Guide

In the “code” folder, you’ll find:

Folders/files that start with “Tutorial” which review topics just learned with small & quick coding exercises

Files that start with numbers (e.g., 1_First_script.R) will be walked through together.

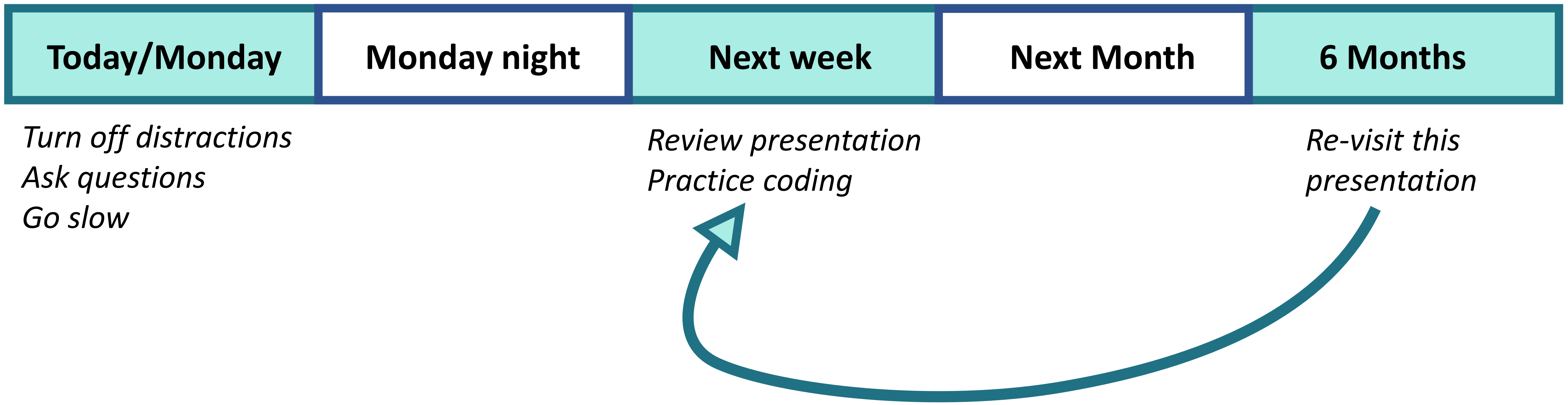
Files that start with “motivatingexample” are advanced scripts intended to show the usefulness of R. Don’t get bogged down in the details!



Name	Date modified	Type
Tutorial_1_demo	1/5/2021 9:46 AM	File folder
Tutorial_2_Basics	12/8/2020 10:54 PM	File folder
Tutorial_3_WorkingWithData	1/5/2021 10:37 AM	File folder
Tutorial_4_DataManipulation	1/5/2021 10:37 AM	File folder
Tutorial_5_ggplot2	11/23/2020 10:32 ...	File folder
.Rhistory	8/6/2020 11:28 AM	RHISTORY File
1_First_script.R	10/30/2020 1:13 PM	R File
2_Basic_Programming.R	11/22/2020 5:36 PM	R File
3_Working_with_your_data.R	1/18/2021 2:00 PM	R File
4_Basic_data_manipulation.R	1/5/2021 9:57 AM	R File
5_Charts_in_ggplot.R	1/18/2021 2:09 PM	R File
6_Summary_analysis.R	1/18/2021 2:11 PM	R File
motivatingexample_crab.R	1/14/2021 7:00 PM	R File
motivatingexample_groundfish.R	1/14/2021 7:03 PM	R File
motivatingexample_inseasonharvest.R	1/14/2021 8:18 PM	R File
motivatingexample_pinkratio.R	1/14/2021 7:02 PM	R File

How to Be Successful

This course is tough but if you do the following, you WILL learn R



Today/Monday
Turn off distractions
Ask questions
Go slow

Monday night
Review / complete assignments

Next week
Review presentation
Practice coding

Next Month
Work on your OWN R project!
Email Justin

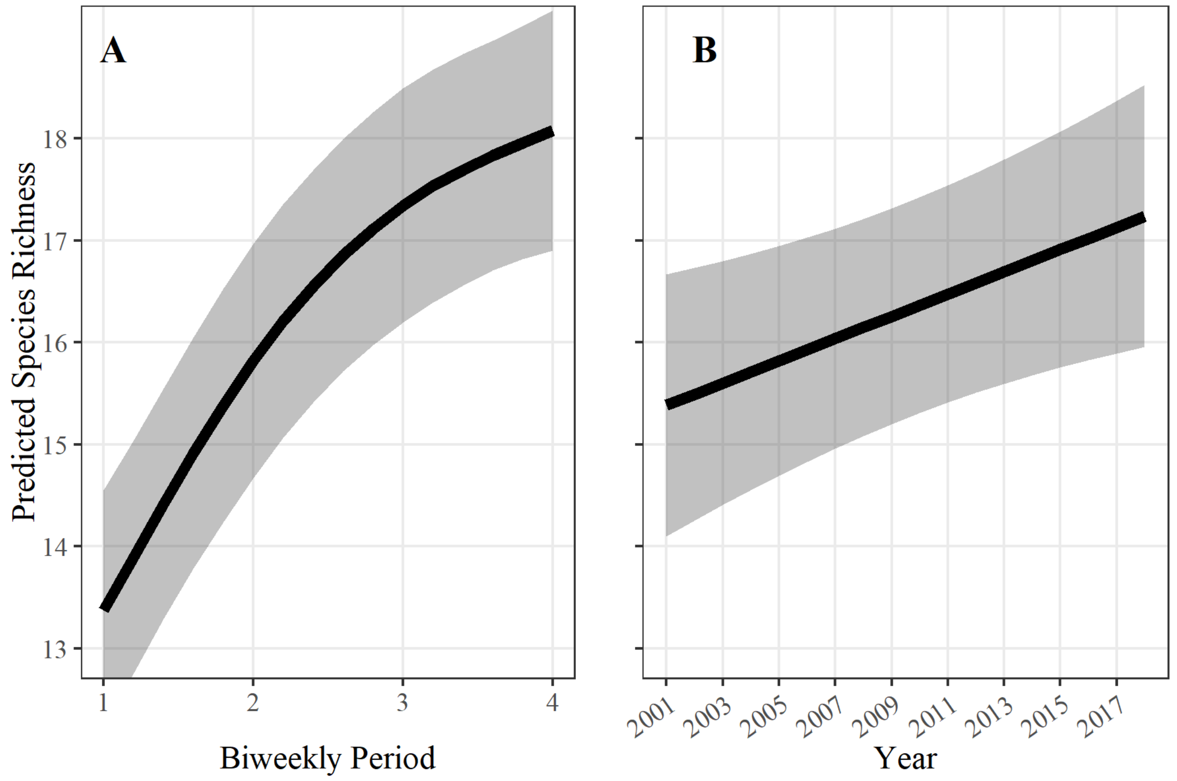
6 Months
Re-visit this presentation

1 – About R

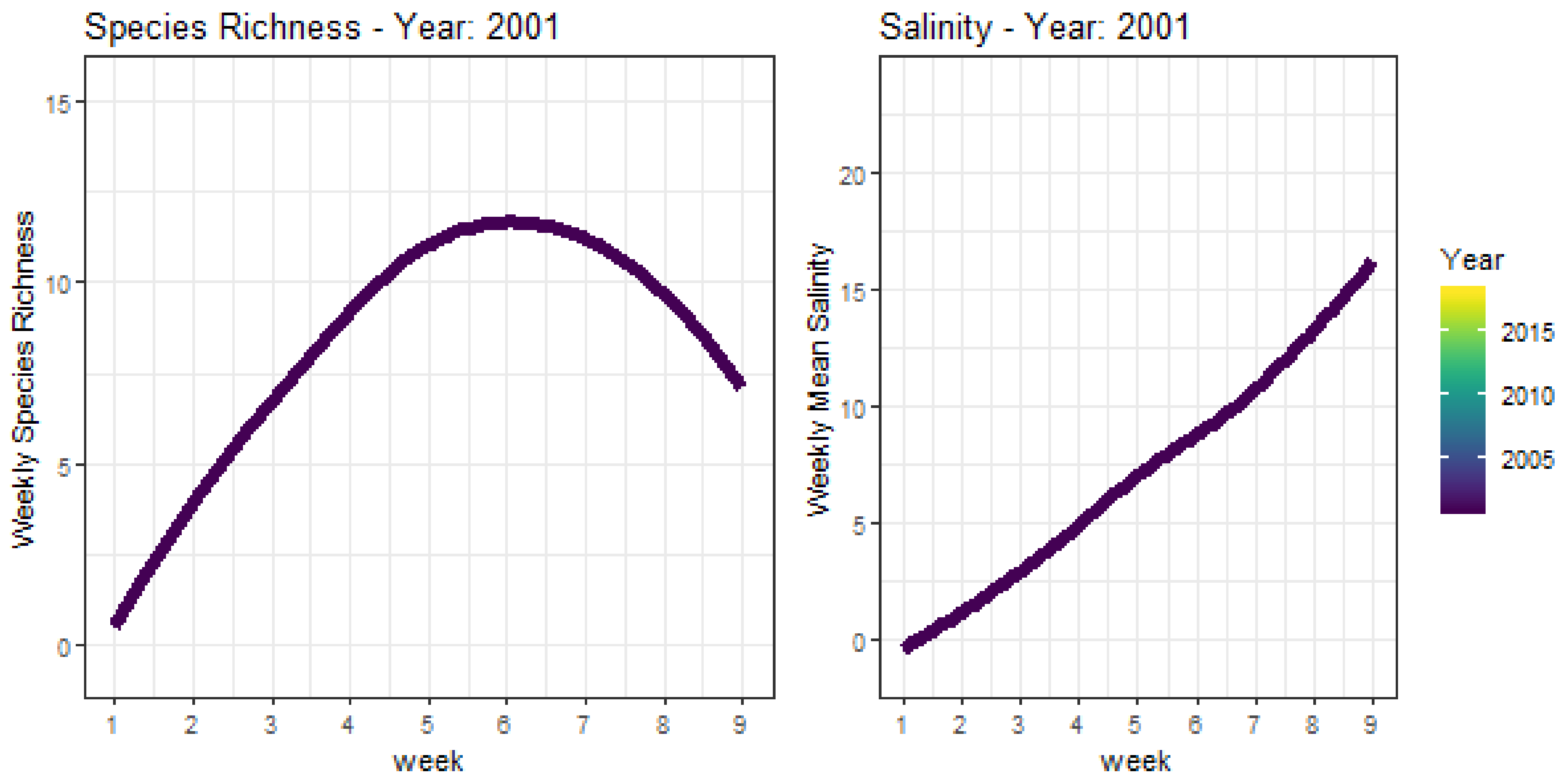
Hello, my name is _R_

Why Use R?

- Create publication quality figures!
- Allows for reproducible research
  - “Saves” all your steps in a file
- Easily automate steps you often do
  - E.g., data import from database
  - QA/QC’ing of data
- High-powered statistical modeling is quite straightforward

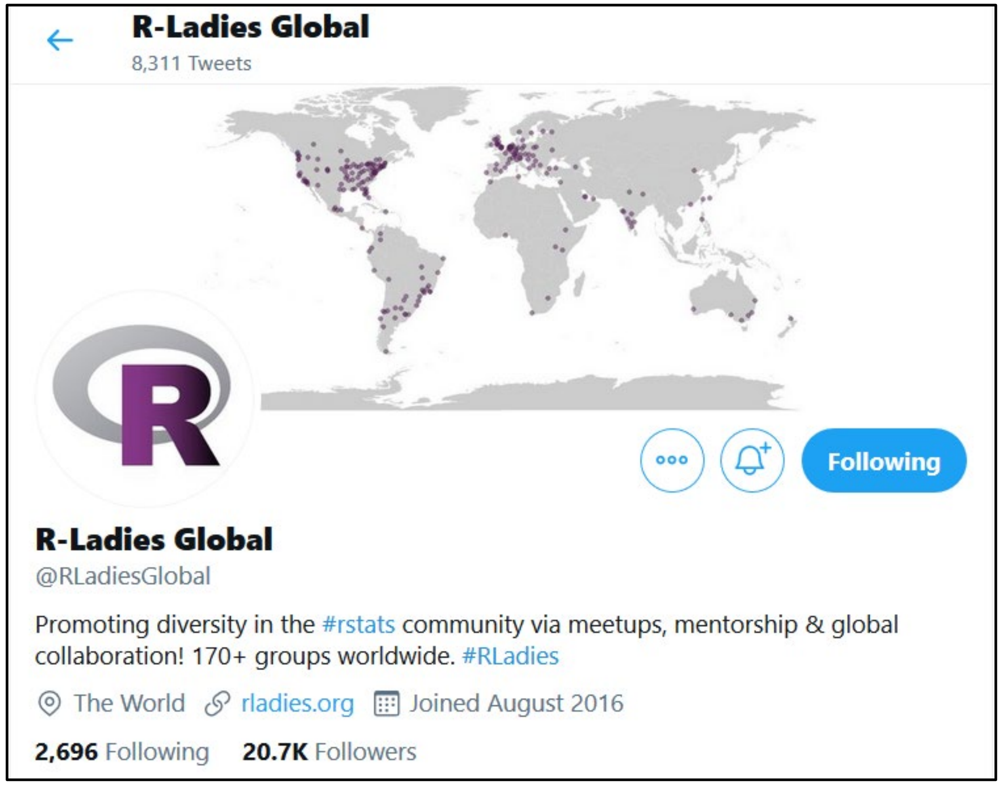


Why Use R?


Species Richness - Year: 2001
Weekly Species Richness
0
5
10
15
week
1
2
3
4
5
6
7
8
9
Salinity - Year: 2001
Weekly Mean Salinity
0
5
10
15
20
week
1
2
3
4
5
6
7
8
9
Year
2015
2010
2005

Why Use R?

- Relatively simple to learn
- Constant advances & evolving features due to:
  - Open-source
  - Modular design
  - Active community
- Online community is very friendly, helpful, and diverse!


R-Ladies Global
8,311 Tweets

Following

R-Ladies Global
@RLadiesGlobal

Promoting diversity in the #rstats community via meetups, mentorship & global collaboration! 170+ groups worldwide. #RLadies

The World rladies.org Joined August 2016

2,696 Following 20.7K Followers

When to Use R

USE R ☺

- You need modeling or any other statistics done.
- You will make a complex figure not possible in Excel.
- You will be updating the figure many times in the future.
- You will be working with biometricians.

DON'T USE R ☹

- You are making a figure that will never be updated.
- You are doing extremely quick analysis.
- Time is of the essence.

R vs RStudio

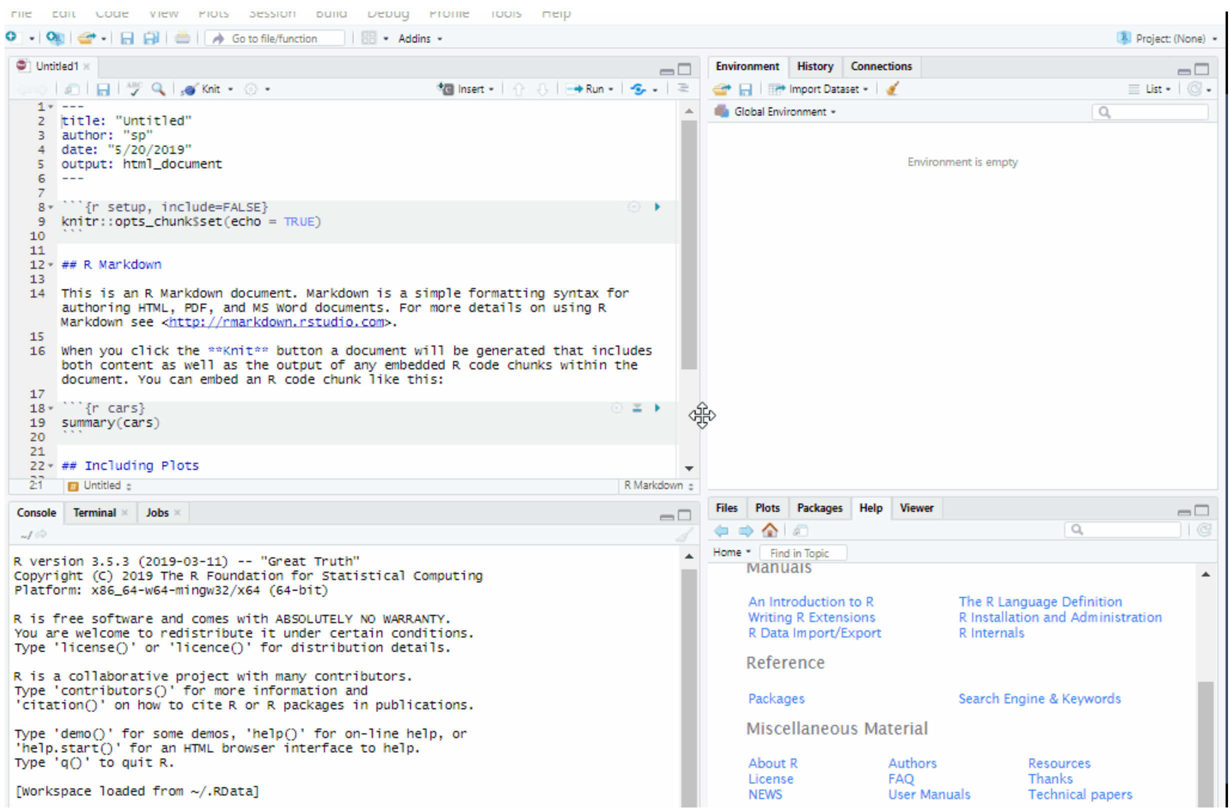
What is the difference between R (AKA "R Project") and RStudio?

- RStudio requires R to run in the background
  - R is the engine, RStudio is the rest of the car
- Think of R as notepad and RStudio as MS Word.
  - More powerful features and is much more user friendly
- From now on, pretty much only open RStudio
  - Easiest to double click the .RProj file

RStudio

You'll see 4 panes. By default, they are:

- ***Script editor*** in top left where you can write code before evaluating it,
- ***Console*** in bottom left where code is evaluated and output is seen,
- ***Environment*** in top right shows variables/dataframes (click on them to open and view!),
- ***Plot/Help*** area in bottom right.



~~~
---
title: "Untitled"
author: "sp"
date: "5/20/2019"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
~~~

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for
authoring HTML, PDF, and MS Word documents. For more details on using R
Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes
both content as well as the output of any embedded R code chunks within the
document. You can embed an R code chunk like this:

```{r cars}
summary(cars)
```

Including Plots
```

```
R version 3.5.3 (2019-03-11) -- "Great Truth"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (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.

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.

[Workspace loaded from ~/.RData]
```

Manuals

An Introduction to R
Writing R Extensions
R Data Import/Export
The R Language Definition
R Installation and Administration
R Internals

Reference

Packages
Search Engine & Keywords

Miscellaneous Material

About R
License
NEWS
Authors
FAQ
User Manuals
Resources
Thanks
Technical papers

gif: https://www.pipinghotdata.com/posts/2020-09-07-introducing-the-rstudio-ide-and-r-markdown/
```

# RStudio Demo

- Let's demo this in RStudio!
- We'll go over:
  - How to start / open RStudio
  - What each "pane" is
  - Where to type your code

```
myvariable
# Note that this printed to your console AND
# Look at the pane in the upper right and you

# There are more complicated objects. Let's ma
# Everything that we put inside the brackets w
# c is the concatenate function in R and squis
myvector = c(1, 3, 5, 7, 9)
myvector
c(2, 4, 6)     # Note that this prints to the co
mynewvector = c("Adult", "Juvenile", "Adult",
mynewvector
```

```
Values
  myvect…   num [1:5] 1 3 5 7 9
  x         num [1:100] 19.933…
  y         int [1:100] 1 2 3 …
```

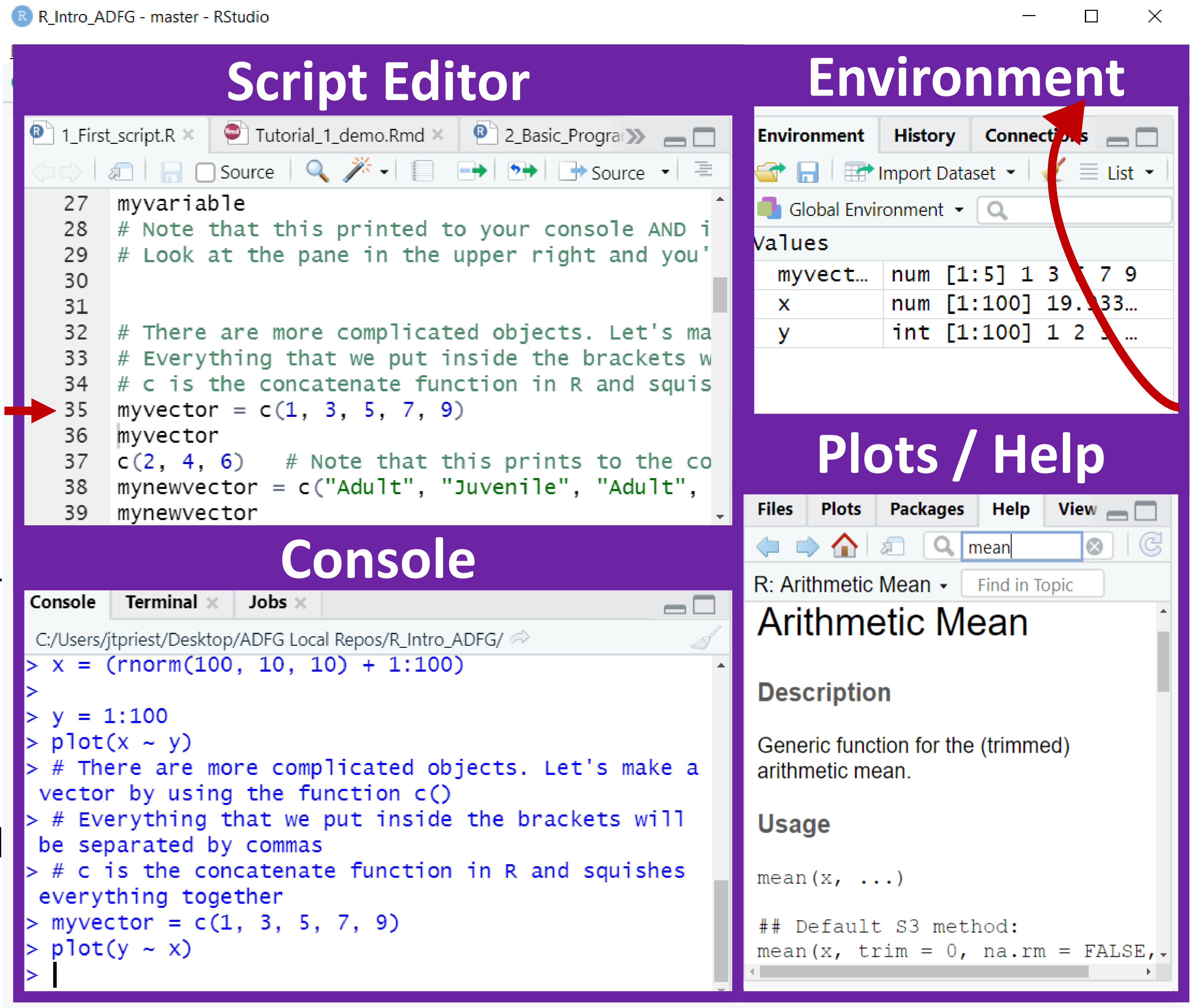
```
> x = (rnorm(100, 10, 10) + 1:100)
>
> y = 1:100
> plot(x ~ y)
> # There are more complicated objects. Let's make a
vector by using the function c()
> # Everything that we put inside the brackets will
be separated by commas
> # c is the concatenate function in R and squishes
everything together
> myvector = c(1, 3, 5, 7, 9)
> plot(y ~ x)
>
```

Script – A collection of lines of code saved to a file that can be run line by line, or all at once, usually the file extension is .R

- If you want to save the code to use it again, you must save it in a script file

Line – In script section each command has its own line. Lines must be complete bits of code otherwise they won't evaluate (to wrap an incomplete chunk of code to the next line, it must end in a comma or something similar)

Console – The bottom part of the program where you can enter code to be run and where executed code is shown



```
myvariable
# Note that this printed to your console AND i
# Look at the pane in the upper right and you'


# There are more complicated objects. Let's ma
# Everything that we put inside the brackets w
# c is the concatenate function in R and squis
myvector = c(1, 3, 5, 7, 9)
myvector
c(2, 4, 6)    # Note that this prints to the co
mynewvector = c("Adult", "Juvenile", "Adult",
mynewvector
```

```
Values
  myvect…   num [1:5] 1 3 5 7 9
  x         num [1:100] 19.333…
  y         int [1:100] 1 2 3 …
```

```
C:/Users/jtpriest/Desktop/ADFG Local Repos/R_Intro_ADFG/
> x = (rnorm(100, 10, 10) + 1:100)
>
> y = 1:100
> plot(x ~ y)
> # There are more complicated objects. Let's make a
 vector by using the function c()
> # Everything that we put inside the brackets will
 be separated by commas
> # c is the concatenate function in R and squishes
 everything together
> myvector = c(1, 3, 5, 7, 9)
> plot(y ~ x)
> |
```

mean

R: Arithmetic Mean

Find in Topic

Arithmetic Mean

Description

Generic function for the (trimmed) arithmetic mean.

Usage

```
mean(x, ...)

## Default S3 method:
mean(x, trim = 0, na.rm = FALSE,
```

Environment – Shows all your variables. If it's a dataframe, it's clickable

Project – Shows which project you're in. More on this later

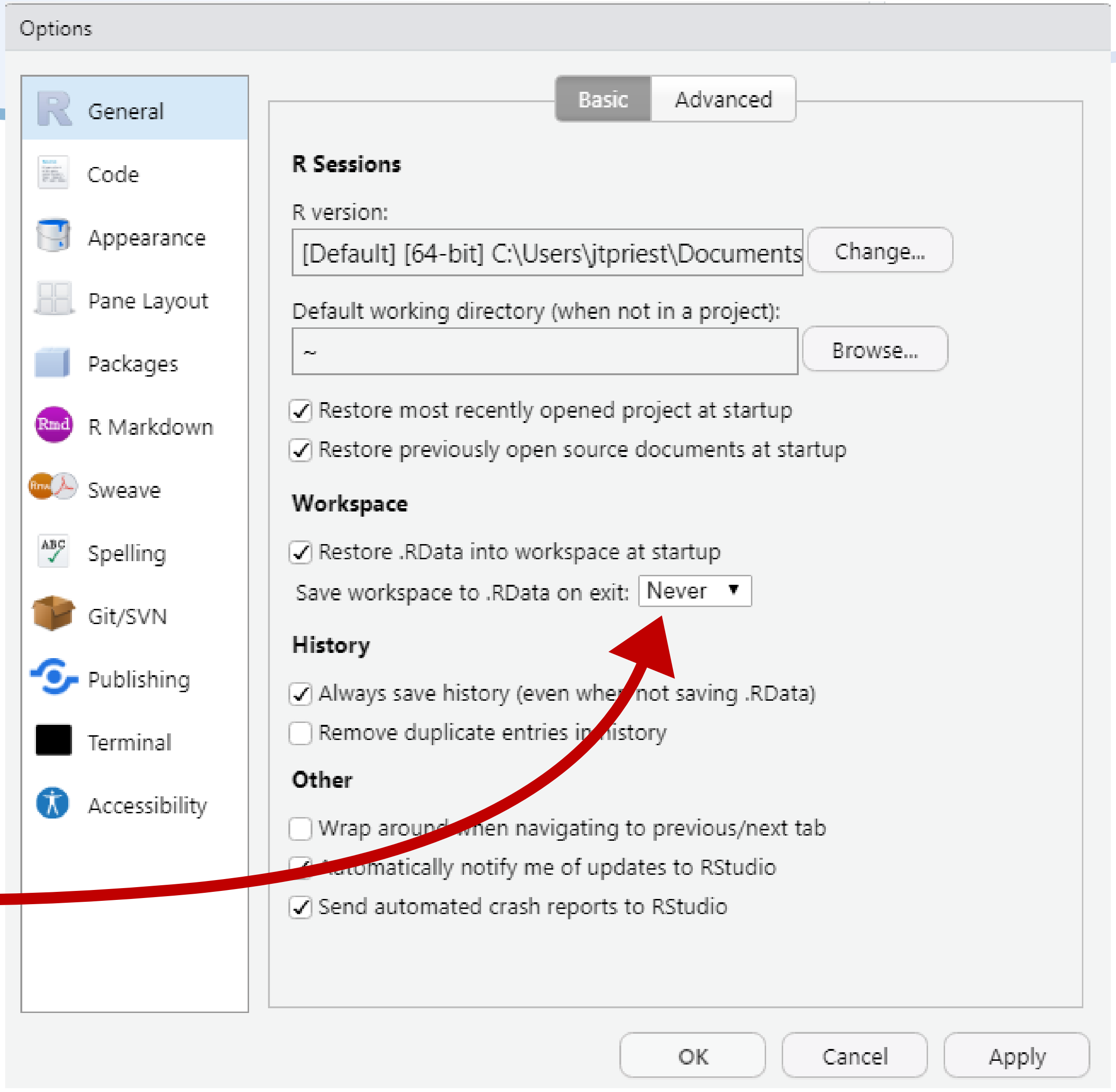
Plots – Shows your recent plots.

Help – Access the help file for every function

RStudio cont.

Highly recommended to change one default setting. Go to:

- Settings
- Click Tools -> General Options
- This will bring up these options
- Change “Save workspace to .RData on exit” to Never



R Sessions

R version:
[Default] [64-bit] C:\Users\jtpriest\Documents

Default working directory (when not in a project):
~

- [✓] Restore most recently opened project at startup
- [✓] Restore previously open source documents at startup

Workspace

- [✓] Restore .RData into workspace at startup

Save workspace to .RData on exit: Never

History

- [✓] Always save history (even when not saving .RData)
- [ ] Remove duplicate entries in history

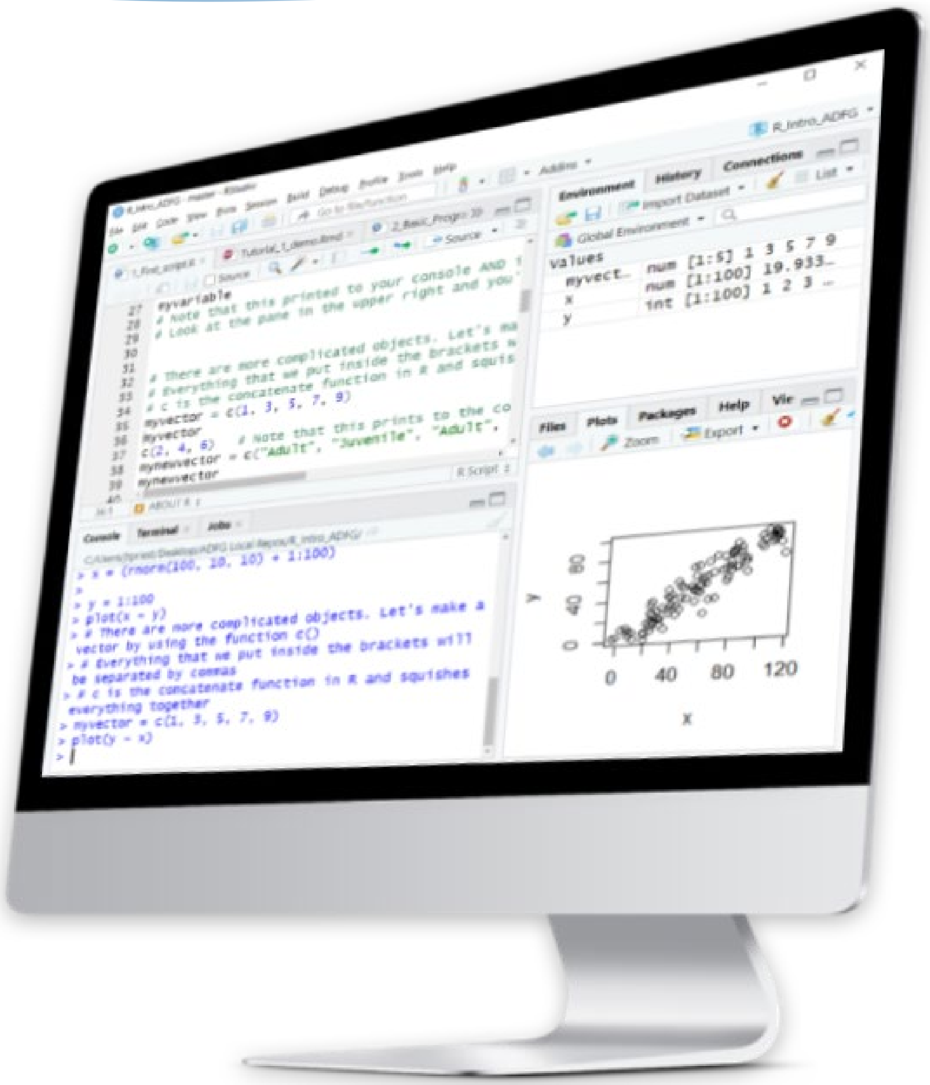
Other

- [ ] Wrap around when navigating to previous/next tab
- [✓] Automatically notify me of updates to RStudio
- [✓] Send automated crash reports to RStudio

RStudio cont.

- Back to RStudio!
- Let’s make those changes

- We’ll also go over:
  - What is an RProject?
  - How to open a script
  - Highlighting a word
  - Changing the theme
  - Using the help tab



```
myvariable
# Note that this printed to your console AND t
# Look at the pane in the upper right and you

# There are more complicated objects. Let's ma
# Everything that we put inside the brackets w
# c is the concatenate function in R and squis
myvector = c(1, 3, 5, 7, 9)
myvector
c(2, 4, 6)    # Note that this prints to the co
mynewvector = c("Adult", "Juvenile", "Adult",
mynewvector
```

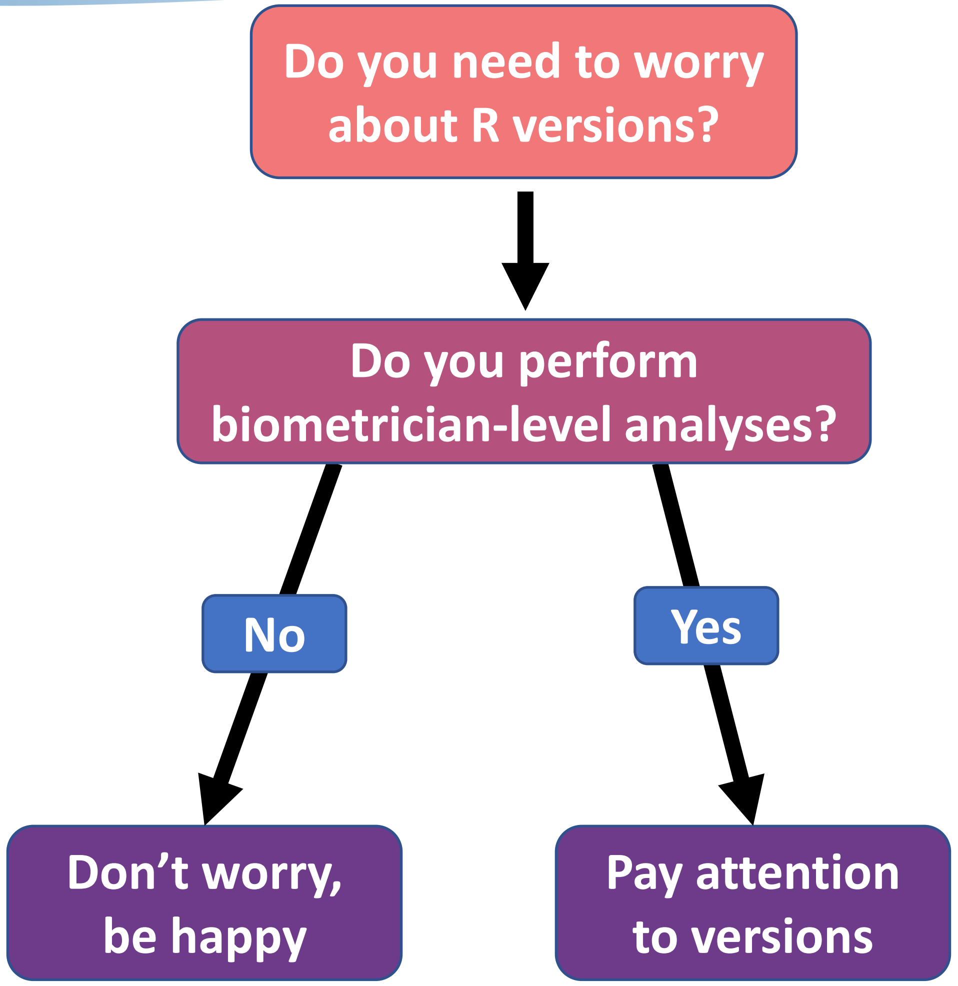
```
> x = (rnorm(100, 10, 10) + 1:100)
>
> y = 1:100
> plot(x ~ y)
> # There are more complicated objects. Let's make a
vector by using the function c()
> # Everything that we put inside the brackets will
be separated by commas
> # c is the concatenate function in R and squishes
everything together
> myvector = c(1, 3, 5, 7, 9)
> plot(y ~ x)
>
```

```
Values
  myvect…   num [1:5] 1 3 5 7 9
  x         num [1:100] 19.933…
  y         int [1:100] 1 2 3 …
```

Versions

- There are different “versions” of R, released to update software
- Occasionally (~once a year) you may need to update R
- Do this by downloading from CRAN and/or running:

  -
    ```
    install.packages("installr")
    library(installr)
    updateR()
    ```


Do you need to worry about R versions?
Do you perform biometrician-level analyses?
No
Yes
Don’t worry, be happy
Pay attention to versions

Quiz 1-1

When should you use "R Project" instead of RStudio?

a) Never. We always use RStudio.

b) Always. Kick it old school R

c) Always. I am a crusty old biometrician.

What are some advantages of using R projects?

a) Future advantages Justin hasn't covered yet.

b) Opens most recent scripts

c) Sets folder directory automatically

QUIZ!

Quiz 1-1 ANSWERS

When should you use "R Project" instead of RStudio?

[✓ a) Never. We always use RStudio.]

b) Always. Kick it old school R

c) Always. I am a crusty old biometrician.

What are some advantages of using R projects?

[✓ a) Future advantages Justin hasn't covered yet.]

[✓ b) Opens most recent scripts]

[✓ c) Sets folder directory automatically]

QUIZ!

Jargon

Function	A set of commands that evaluates your data in a specific way. You can write your own function, or they can be provided via other packages
Argument	In a function, this tells the function how to proceed For example: `mean(data, na.rm = TRUE)` uses an argument to remove NAs
Package	A group of new commands / functions to extend the usability of your analysis.
Variable	Something saved in R's memory for use later.

Jargon cont.

Dataframe	A group of rows and columns (like a spreadsheet). Each column has a specific type (TRUE/FALSE, integer, number, character, factor, etc.).
Factor	A type of variable that is a categorical grouping (“red” vs “blue”; “Treatment1” vs “Treatment2” vs “Treatment3”).
=	The equals sign sets something to a variable (by convention, this is a static number, character, etc., not a dataframe)
<-	The “leftward arrow” sets the items to the right to be equal to the variable on the left. For example: `x1 <- NewVariable`
c()	Lowercase c means concatenate which groups a bunch of things together. For example: `x1 <- c(“New Variable1”, “New Variable2”)`

Even more Jargon

[]	Square brackets mean selecting a subset
==	To check if things are equal, use two equals signs together
!=	To check if things are NOT equal, use this *(There are others that are used less commonly: & means AND, | means OR, ! means NOT, %in% means in a list)*
~	In base R the tilde usually means “regressed upon”

The last of the Jargon

rm()	Removes a variable from your workspace
str()	Inspects the structure of a variable
$	Selects a particular column of a dataframe For example: `dataframename$col1`
#	Makes a “comment” (not run code), telling R to not run anything on this line to the right

```
> print(“Hello!”) #This line only evaluates the Hello! part ignoring this
```

Also: Lines must end with a comma, bracket, or similar. If you want your code to continue to the next line, you will need a comma at the end of a line

Let’s head to RStudio

- Let’s familiarize ourselves with RStudio
  - You are learning two things at once: the new-to-you program RStudio and the R language
- We’ll put what we just learned to use in RStudio and review the basics of the program. You can watch my screen and/or run the lines yourselves. We’ll assess fisheries data that you use
- Later, revisit the script on your own. Consider modifying lines
  - Read the comments then run the line (“Run” or Ctrl+Enter/Cmd+Return)
  - Go slow, it isn’t a race
  - Ask yourself *why* something works or fails

One last RStudio tip

- If you'd like, you can change the appearance of RStudio (color & theme)
- Go to Tools -> Global Options -> Appearance
- Choose an "Editor Theme" that you like

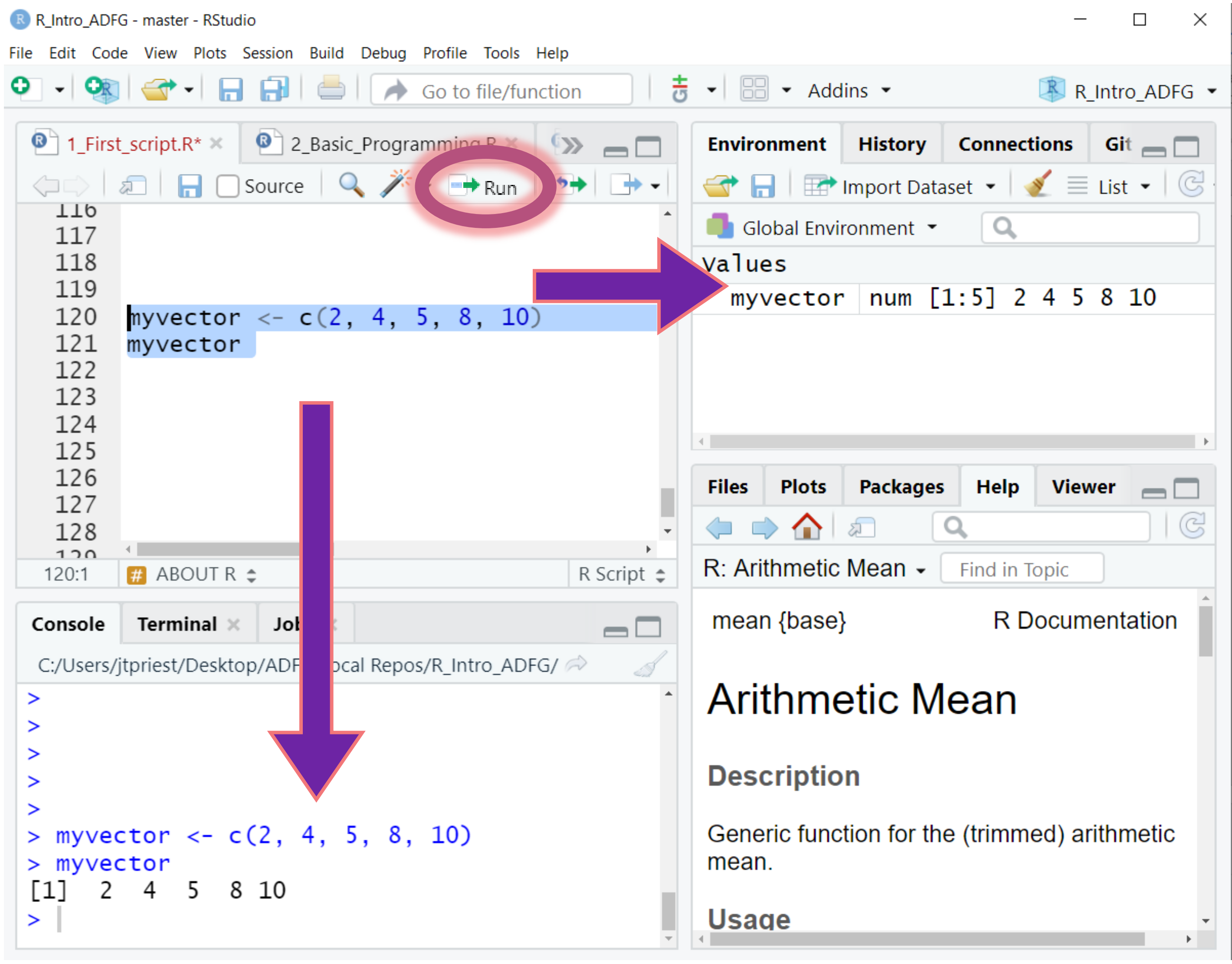
RStudio Summary

What we just learned

- How to open a R project
- 4 panes in RStudio
- How to run a line
- Difference between script & console
- Autocomplete for functions
- Highlighting a word
- Help
- Viewing a dataframe

RStudio Review

- Use "<-" to save a dataframe
    - Otherwise code doesn't save for access later (which is often fine!)
- Ctrl + Enter (or Run Button) evaluates the line of code (sending it to console)
    - Put cursor somewhere in that line, or highlight the chunk you want
- Objects in the environment disappear once we close R. We'll need to re-run whole script to get them back. They don't "live" anywhere permanently.



```
myvector <- c(2, 4, 5, 8, 10)
myvector
```

```
Values
 myvector    num [1:5] 2 4 5 8 10
```

```
> myvector <- c(2, 4, 5, 8, 10)
> myvector
[1]  2  4  5  8 10
>
```

Quiz 1-2

QUIZ!

Which symbols would we use to "set something equal to" and "make a comment / don't run line"?

a) `==` and `comment()`
b) `<-` and `#`
c) `equal` and `c()`
d) `$$` and RUN

How can we tell R to run (evaluate) a line of code?

a) Ask nicely
b) Ctrl + Enter (Cmd + Return)
c) Swear
d) "Run" button

Quiz 1-2 ANSWERS

Which symbols would we use to "set something equal to" and "make a comment / don't run line"?

a) `==` and `comment()`

[✓ b) `<-` and `#`]

c) `equal` and `c()`

d) `$$` and RUN

Why?

Leftward arrow sets something while the hashtag means don't run this line

How can we run (evaluate) a line of code?

a) Ask nicely

[✓ b) Ctrl + Enter (CMD + Return)]

c) Swear

[✓ d) "Run" button]

Why?
There are two ways to run a line;
I usually use the keyboard shortcut

QUIZ!

Quiz 1-3

How do we “save” a variable to access it later?

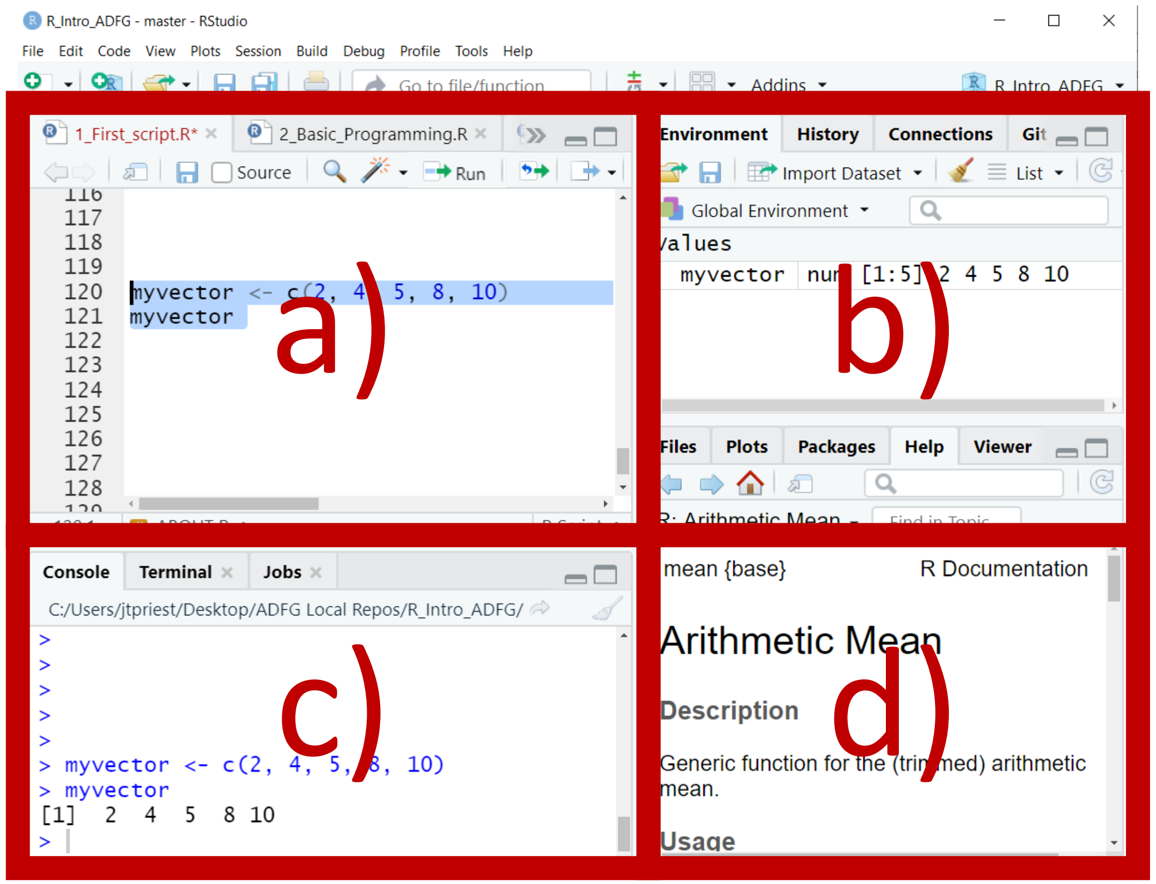
a) =

b) <-

c) variahble returno

d) save()

Which pane is the editor (1), and which is the console (2)?



```
myvector <- c(2, 4, 5, 8, 10)
myvector
```

```
Values
  myvector   num [1:5] 2 4 5 8 10
```

```
C:/Users/jtpriest/Desktop/ADFG Local Repos/R_Intro_ADFG/
>
>
>
>
>
> myvector <- c(2, 4, 5, 8, 10)
> myvector
[1]  2  4  5  8 10
>
```

mean {base}   R Documentation

Arithmetic Mean

Description

Generic function for the (trimmed) arithmetic mean.

Usage

Quiz 1-3 ANSWERS

How do we "save" a variable to access it later?

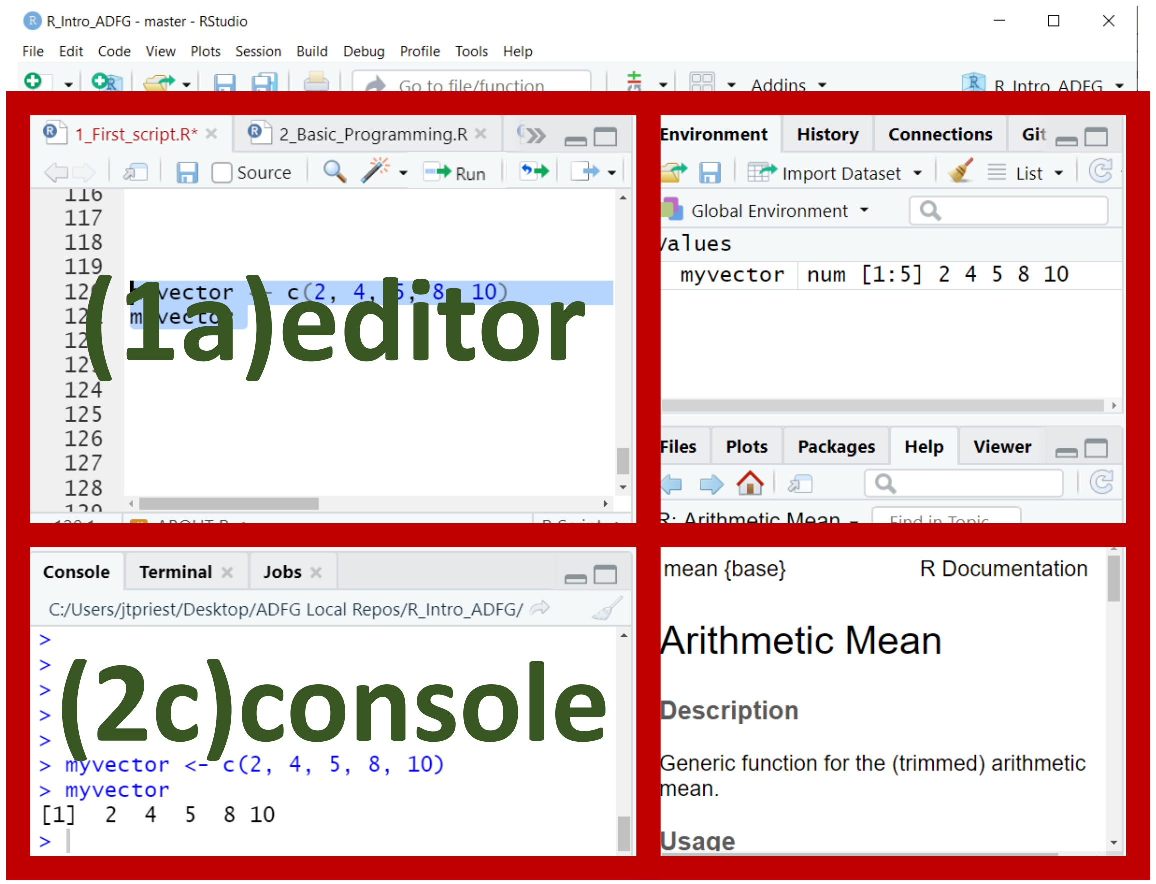
a) = ✓

b) <- ✓

c) variahble returno

d) save()

Which pane is the editor (1), and which is the console (2)?



```
myvector <- c(2, 4, 5, 8, 10)
myvector
```

```
myvector  num [1:5] 2 4 5 8 10
```

```
C:/Users/jtpriest/Desktop/ADFG Local Repos/R_Intro_ADFG/
> myvector <- c(2, 4, 5, 8, 10)
> myvector
[1]  2  4  5  8 10
>
```

Arithmetic Mean

Description

Generic function for the (trimmed) arithmetic mean.

Usage

2 – The Basics of Programming

Did you know today was the day you became a programmer?

Errors

If you use R, you WILL get errors

You'll get errors every other line (I do!) and that's OK!

Don't stress about an error, just find what the issue is

Errors don't go away, you just get faster at solving them

☺

Basic Programming

- To set objects we use “=“ or “<-”.
  - *(By convention, use equals for setting variables that don’t change (a=5) and the arrow for everything more complex)*

- You re-write over previous values every time you evaluate

- R is case sensitive

Use ctrl+Enter (CMD+Return) or run button to evaluate

```
> a = 5
> a
[1] 5
```

The [1] in the console output means it’s the 1st element

```
> a = 7
> a
[1] 7

> A
Error: object 'A' not
found
```

Basic Programming cont.

```
> x1 <- c(10, 8, 10, 12)
> mean(x1)
```

c() is short for concatenate, or combining things together

```
> x2 <- c(1, 2
```

Lines can't continue if left unfinished

```
log(), sqrt(), sum()
```

Explore what happens when you take log of something

```
seq(from=0, to=100, by=2)
```

Create a sequence

Basic Programming cont.

- Enter the following code:

```
x3 <- c(2,4,6,8,10)
x3[5]
```

- This returns the 5th element (10)

```
myfirstdf <- data.frame(sex = c("Male", "Male", "Female"),
   length = c(110, 112, 90),
   weight = c(3, 3.4, 2.4),
   age = c(2, 2, 1))
myfirstdf
```

- Look at this dataframe! You've got some data!

Quiz 2-1

What would be the result after running these 3 lines:

```
myvalue = 5
myvalue = 7
myvalue
```

5 OR 7

BUG HUNTER: Which lines of code contain no problems?

```
mean(c(1, 2, 3, 3, 4)
Seq(from = 1, to = 5)
myvalue == 5
c(1, 1, 3, )
sum(1, 2, 3)
```

QUIZ!

Quiz 2-1 ANSWERS

What would be the result after running these 3 lines:

```
myvalue = 5
myvalue = 7
myvalue
```

- 5 OR 7 ✓

BUG HUNTER: Which line(s) of code contain no problems?

```
mean(c(1, 2, 3, 3, 4)))
Seq(from = 1, to = 5)
myvalue == 5
c(1, 1, 3, )
```

sum(1, 2, 3) ✓

QUIZ!

Quotations

- Knowing what is and isn’t put in quotes is tricky when starting out
- Single quotes ‘x’ are usually interchangeable with double quotes “x”
- Often you differentiate between a character and a variable by putting characters in quotes
  - Conversely, sometimes you quote variables inside a function

NAs

- If you import your own data, you'll inevitably run into NA issues.
- An NA is just a known "blank"
- R will often throw errors for NAs
  - It just doesn't want to assume it knows best what you want
- You can usually remove NAs in a function (usually good idea) or filter them out ahead of time

```
> mean(c(3,4,5,6,NA))
[1] NA

> mean(c(3,4,5,6,NA), na.rm = TRUE)
[1] 4.5
```

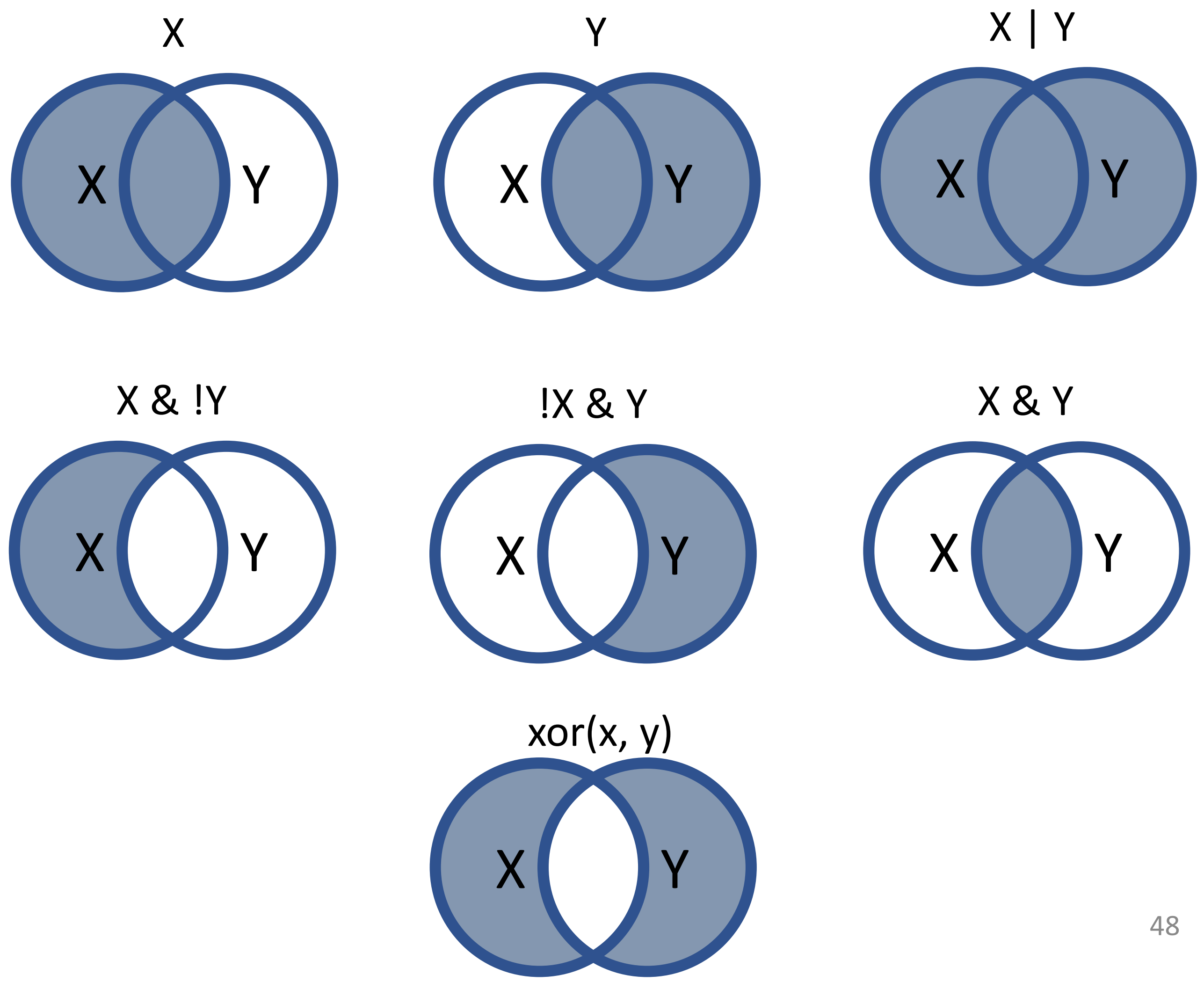
Remember:

"A blank is not a zero!"

Operators

- & means "and"
- | means "or"
- ! means "not"

- %in% checks if things are in a list
    - Helpful if you're filtering to a subset of species!



A Word About Types

All variables (objects) have attributes and these attributes MUST be compatible within themselves

For example, if you have this list

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

These are all numeric!

But if you have this list

```
> c(1,2,3,4, "five")
```

Then these will ALL be characters

Use `str()`, `typeof()` to check the structure and type of objects

```
> str(c(1,2,3,4, "five"))
 chr [1:5] "1" "2" "3" "4"
"five"
```

Object Types

- Vector
- Matrix
- Factor
- List
- Dataframe (made up of previous types!)
- *and others!*

Dataframes are what you'll use 90% of the time

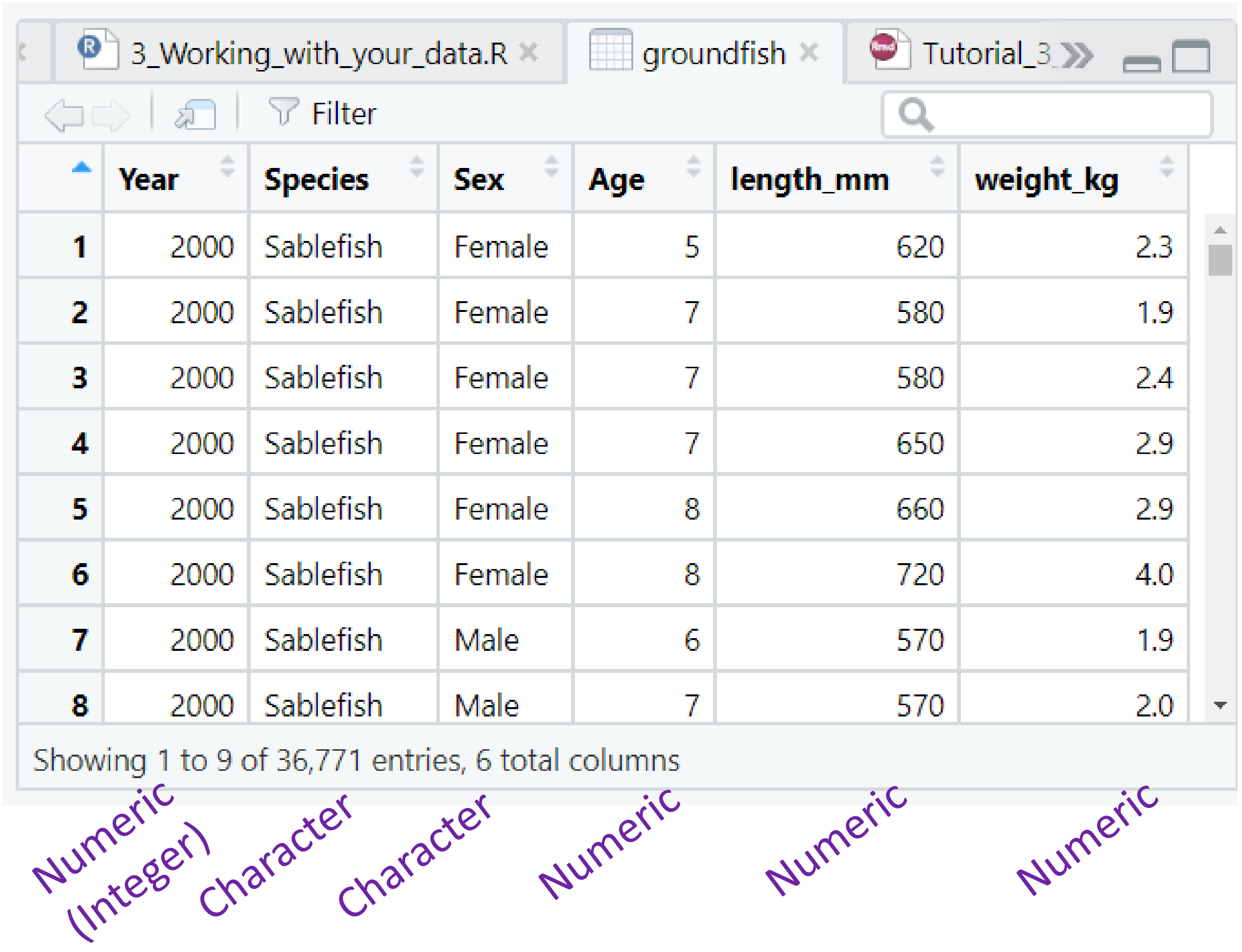
```
> as.vector(c(3,4,5,6))
[1] 3 4 5 6

> as.factor(c("area1", "area2", "area3"))
[1] area1 area2 area3
Levels: area1 area2 area3
```

Dataframes

What is a dataframe?

- Think of a dataframe as analogous to a spreadsheet. There are rows and columns

- Each column has a specific "type", and this type doesn't change between rows
  - Can't contain both characters and numbers
  - We can change it later if needed



	Year	Species	Sex	Age	length_mm	weight_kg
1	2000	Sablefish	Female	5	620	2.3
2	2000	Sablefish	Female	7	580	1.9
3	2000	Sablefish	Female	7	580	2.4
4	2000	Sablefish	Female	7	650	2.9
5	2000	Sablefish	Female	8	660	2.9
6	2000	Sablefish	Female	8	720	4.0
7	2000	Sablefish	Male	6	570	1.9
8	2000	Sablefish	Male	7	570	2.0

Showing 1 to 9 of 36,771 entries, 6 total columns

Numeric vs Factor Variables

It is important to understand the difference between a variable that is a factor (categorical) and a number.

Sometimes these distinctions are tricky (e.g., District Number is technically a category)

Year can be a funny variable. Could be a factor (compare ANOVA between years) or numerical (time trend)

Factor	**Numeric**
Species	Measurements
River	Counts
Station	Duration
Year	Year

Why does this matter?

Variable type controls plot type (e.g., scatterplot is numerical vs numerical; boxplot is factor vs numerical) and statistical models

Variables

- What is a variable?
  - Saves something by name to look at later

Variable Naming:

- Never name something the same as a function (e.g., "mean")
- Can't start with a number
- Can't contain a space, comma, etc.
- Make it memorable & recognizable to others

Good

juneaurain_2005-2019
blackcod_surveycount
tanner_chela

Bad

x
1 dataframe
dataframe1
mean
weight
cs125_10_g

Errors Revisited

Some common errors:

- Misspelling variable/function
- Not ending a line in parenthesis, comma, etc.
- Unpaired parentheses/quotes

RStudio will help minimize these!

- Capitalizing vs no caps
- Not loading package before calling function
- Wrong data type is used

Experience, going slow & thorough minimizes these!

Errors Revisted cont.

More common → **Less common**

Error Message	Possible Problem
unexpected symbol in ____	You made a syntax (coding grammar) error (extra comma, parenthesis, etc.)
could not find function	You forgot to load a package or misspelled the function name.
cannot open file .* : No such file or directory	You tried a path that doesn't exist.
there is no package called ___	You forgot to install a package or misspelled the package name
object .* not found	The object you are looking for might not exist.
non-numeric argument to ___	You tried to use a character vector where a numeric is needed.
attempt to apply non-function	You tried to use an object which is not a function as a function.
object of type 'closure' is not subsettable	You called \`$\` on a function.

Based on https://github.com/colinfay/argh

Stop! And Restarting

- Practice closing out of RStudio and then reopening. You can click save workspace or not. See what happens when you do both!

- Practice closing out of a long loop (stop button) and open-ended code (esc key in console)

Quiz 2-2

Which one of the following will make a numeric sequence?

a) `c(1,2,3,4, “five”)`
b) `c(1, 2, 3, 4, 5)`
c) `c(1, 2, 3, five)`
d) `C(1,2,3,4,5)`

- Now test your answers in R

Which of the following variables could be a factor?

a) `Fish sex`
b) `District`
c) `Gear`
d) `Year`
e) `Comment`

QUIZ!

Quiz 2-2 ANSWERS

Which one of the following will make a numeric sequence?

a) `c(1,2,3,4, “five”)`
b) `c(1, 2, 3, 4, 5)` ✓
c) `c(1, 2, 3, five)`
d) `C(1,2,3,4,5)`

Which of the following variables could be a factor?

a) Fish sex ✓
b) District ✓
c) Gear ✓
d) Year ✓
e) Comment

Why didn’t these work?

Case sensitivity, combining types, and calling an object that doesn’t exist

Why?

All can be categorical variables except for comments

QUIZ!

LUNCH!

Return at 13:30.
If you have questions or feedback from this morning, email me!

3 – Working With *Your* Data

It’s cooler than the underside of your pillow

But First! Directories

- Before getting your data in, you should know where R thinks you are

- By using modern tools in R (esp RProjects) we avoid many of the headaches with directories; regardless, it's important to be aware of these issues.

- Type `getwd()` to see what your working directory is.

What happens if you write:

```
> read.csv("input.csv")
```

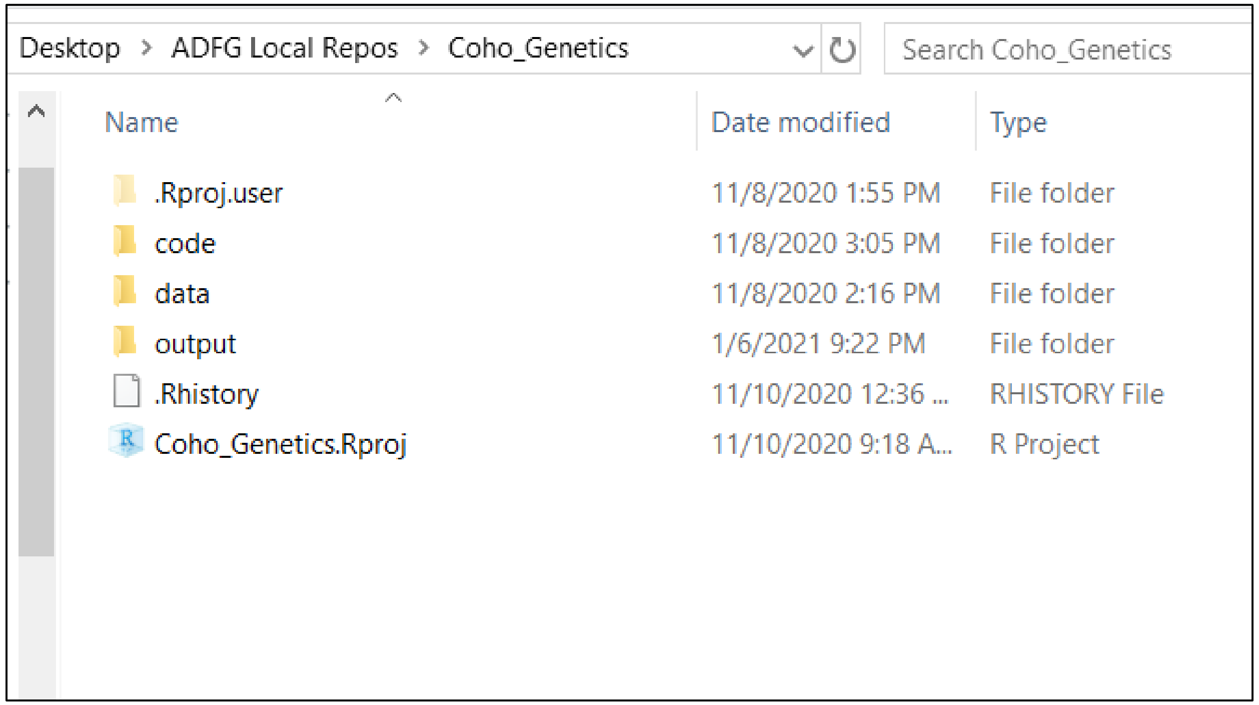
It can't find the file. Change this to

```
> read.csv("data/input.csv")
```

Directory Structure

- It is *very* helpful to standardize your structure

- Under your main project folder, use one .Rproj for that group of analyses. Use folders named:
  - data
  - code (or “analysis”)
  - output
  - etc.



Desktop › ADFG Local Repos › Coho_Genetics

Search Coho_Genetics

Name	Date modified	Type
.Rproj.user	11/8/2020 1:55 PM	File folder
code	11/8/2020 3:05 PM	File folder
data	11/8/2020 2:16 PM	File folder
output	1/6/2021 9:22 PM	File folder
.Rhistory	11/10/2020 12:36 ...	RHISTORY File
Coho_Genetics.Rproj	11/10/2020 9:18 A...	R Project

Packages

- A package is just a collection of new functions (commands). If you've never used the package before, run: `install.packages("packagename")`

- After this, you load it into R: `library(packagename)`

- You'll need to load your libraries every time you restart R, so add the libraries you'll need for that specific script at the top.

Run once!

```
> install.packages("dplyr")

> library(dplyr)
```

Run every time you open R

Packages cont.

- This is an important distinction, so I'll repeat it:
- Install a package once
  install.packages()
- Load a library every time R restarts
  library()

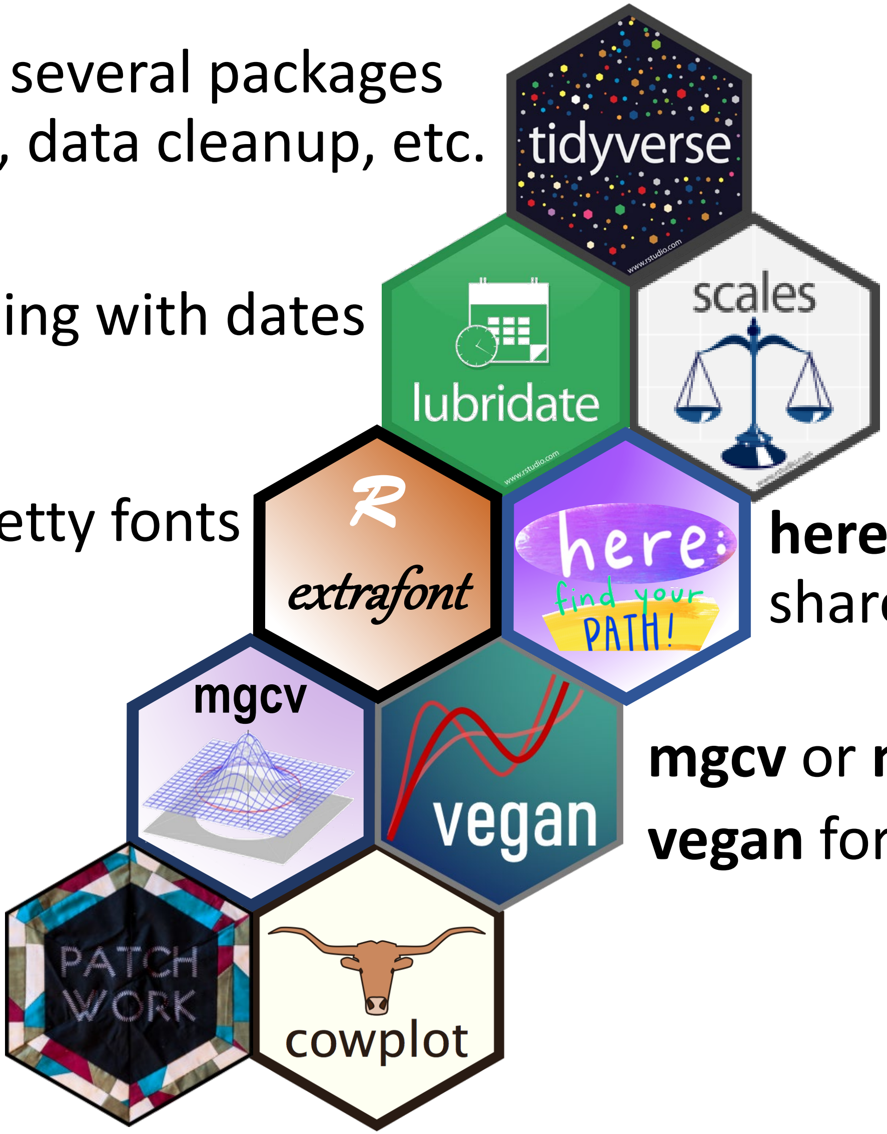
Packages cont.

Tidyverse contains several packages together for plotting, data cleanup, etc.

lubridate for dealing with dates

extrafont for pretty fonts

cowplot or **patchwork** put multiple plots together



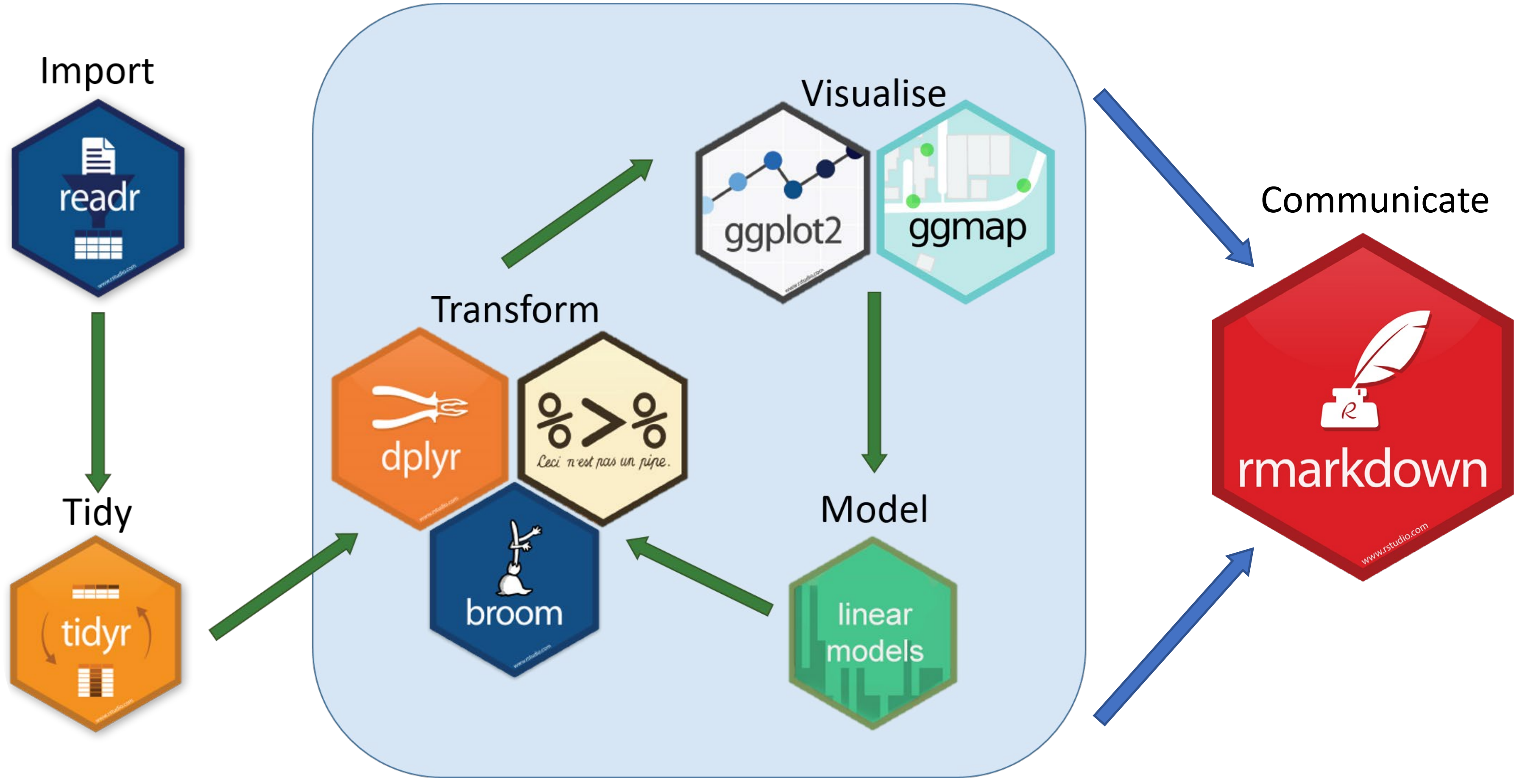
scales for dealing with ease of plotting axis scales

here for relative filenames (if you share / rename file directories),

mgcv or **nlme** for advanced modeling

vegan for multivariate statistics

Common Analysis Workflow



But Wait! There’s More!

Quiz 3-1

After running `library(mgcv)` you see "Error in library(mgcv) : there is no package called 'mgcv' ". What should you do?

a) Install package via `install.packages("mgcv")`

b) Double-check spelling of package

c) Call for help

d) Curl into a ball and cry

After running `read.csv("mydata.csv")` you see "Error in file(file, "rt") : cannot open the connection

In addition: Warning message:

In file(file, "rt") : cannot open file 'mydata.csv': No such file or directory". What should you do?

a) Check working directory (`getwd()`)

b) Double-check spelling of file

c) Check if file is in a subfolder

d) Curl into a ball and cry

QUIZ!

Quiz 3-1 Answers

After running `library(mgcv)` you see "Error in library(mgcv) : there is no package called 'mgcv' ". What should you do?

[✓ a) Install package via `install.packages("mgcv")`]

[✓ b) Double-check spelling of package]

c) Call for help

d) Curl into a ball and cry

After running `read.csv("mydata.csv")` you see "Error in file(file, "rt") : cannot open the connection

In addition: Warning message:

In file(file, "rt") : cannot open file 'mydata.csv': No such file or directory". What should you do?

[✓ a) Check working directory (`getwd()`)]

[✓ b) Double-check spelling of file]

[✓ c) Check if file is in a subfolder]

d) Curl into a ball and cry

QUIZ!

Data Do's & Data Don'ts

- Use unsummarized data (if possible)

- To the extent possible, do cleanup in R so that if you re-download from your database (e.g., OceanAK) you don't have to spend time changing things
  - Starting out, you can "cheat" and clean up in Excel first
  - It is *highly* recommended to bite the bullet and do all of this in R!

- Don't use "tabled" data, e.g., Year by Stream
  - Use "long" format with many rows where each row is a year and a stream

- Use "tidy" data where:

  - Every column is variable. - Every row is an observation. - Every cell is a single value.
  - More info here

Tidy vs non-tidy data

Which of these datasets are tidy data?

Year	Auke Creek	Berners River	Hugh Smith Lake
2018	146	3550	619
2019	345	9405	1235
2020	173	3296	634

Year	River	Count
2018	Auke Creek	619
2019	Auke Creek	1235
2020	Auke Creek	634
2018	Berners River	3550
2019	Berners River	9405
2020	Berners River	3296
2018	Hugh Smith	619
2019	Hugh Smith	1235
2020	Hugh Smith	634

This one!
Any guesses why?

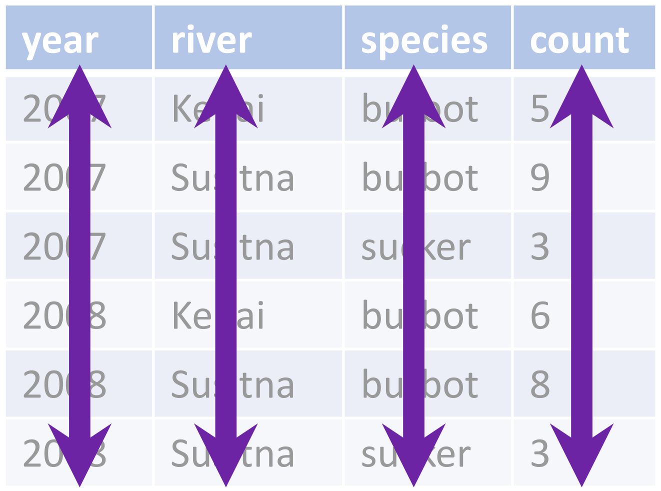
Tidy vs non-tidy data

To see why: if there were a comment, which does it refer to?

Year	Auke Creek	Berners River	Hugh Smith Lake	Comment
2018	146	3550	619	No fish 7/24-8/25
2019	345	9405	1235	Andy quit counting
2020	173	3296	634	Prob 100 fish low

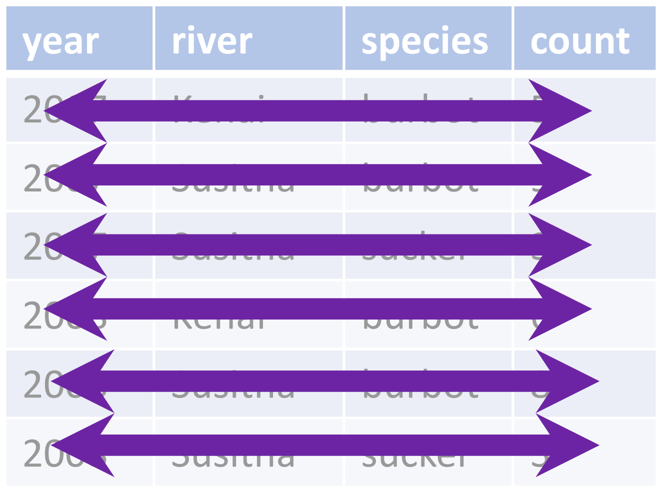
Year	River	Count	Comment
2018	Auke Creek	619	No fish 7/24-8/25
2019	Auke Creek	1235	Andy quit counting
2020	Auke Creek	634	High water 8/30
2018	Berners River	3550	
2019	Berners River	9405	
2020	Berners River	3296	Not a peak count
2018	Hugh Smith	619	
2019	Hugh Smith	1235	Prob 100 fish low
2020	Hugh Smith	634	Andy didn't count jacks

Tidy Data



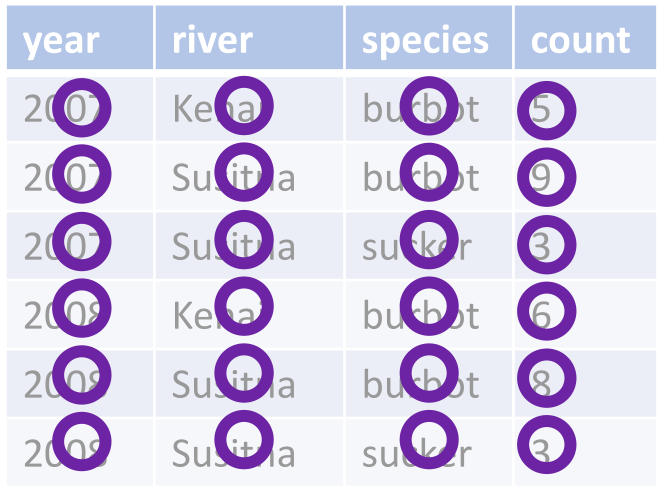
year	river	species	count
2007	Kenai	burbot	5
2007	Susitna	burbot	9
2007	Susitna	sucker	3
2008	Kenai	burbot	6
2008	Susitna	burbot	8
2008	Susitna	sucker	3

Each column is a variable



year	river	species	count
2007	Kenai	burbot	5
2007	Susitna	burbot	9
2007	Susitna	sucker	3
2008	Kenai	burbot	6
2008	Susitna	burbot	8
2008	Susitna	sucker	3

Each row is an observation



year	river	species	count
2007	Kenai	burbot	5
2007	Susitna	burbot	9
2007	Susitna	sucker	3
2008	Kenai	burbot	6
2008	Susitna	burbot	8
2008	Susitna	sucker	3

Each cell is a value

Why be Tidy?

- Why does it matter to be tidy?
- Though Excel worked quickly, it gave us all bad habits

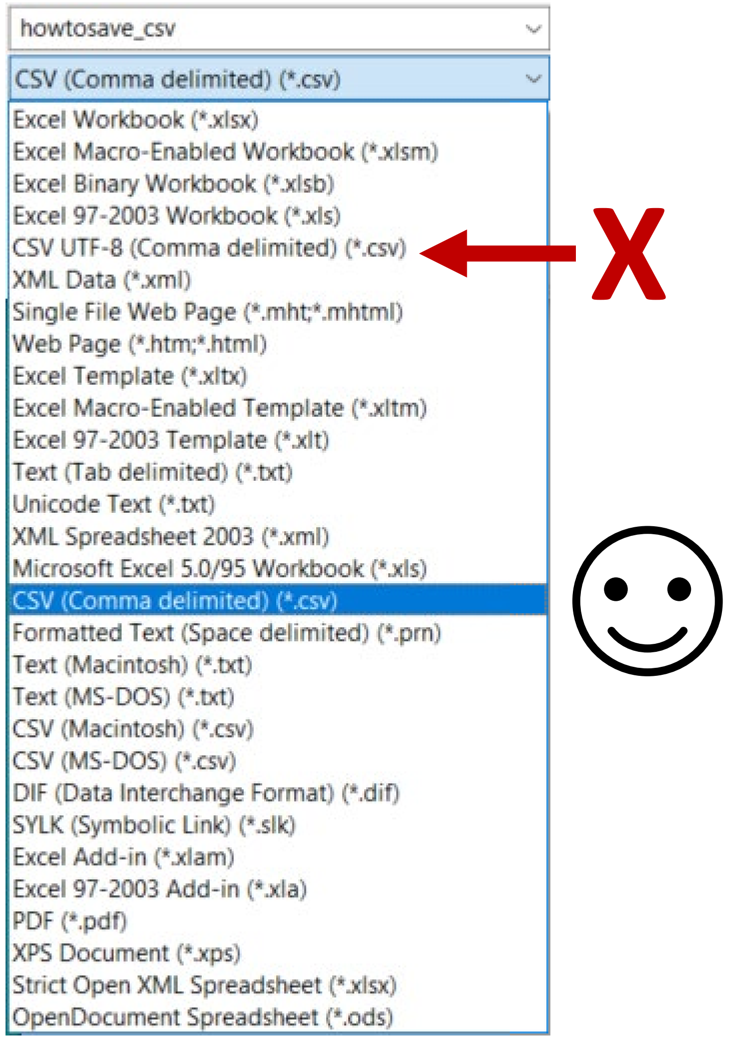
- By using tidy data, we can:
    - Add comments to each observation
    - Easily pair a variable to new data
    - Ensure all observations are of same type & structure

FINALLY! Getting data into R!

- Import things as a .CSV if you can
  - This prevents issues by using a static file (i.e., no formulas)
  - (All databases allow export as .CSV)

- You can still import files in .XLS or .XLSX but it's easier to use "flat" files

- Use "`read.csv()`" or "`read_csv()`"
  - We haven't yet used the "tidyverse", so we'll only use read.csv(). In future, read_csv() is better


howtosave_csv
CSV (Comma delimited) (*.csv)
Excel Workbook (*.xlsx)
Excel Macro-Enabled Workbook (*.xlsm)
Excel Binary Workbook (*.xlsb)
Excel 97-2003 Workbook (*.xls)
CSV UTF-8 (Comma delimited) (*.csv) X
XML Data (*.xml)
Single File Web Page (*.mht;*.mhtml)
Web Page (*.htm;*.html)
Excel Template (*.xltx)
Excel Macro-Enabled Template (*.xltm)
Excel 97-2003 Template (*.xlt)
Text (Tab delimited) (*.txt)
Unicode Text (*.txt)
XML Spreadsheet 2003 (*.xml)
Microsoft Excel 5.0/95 Workbook (*.xls)
CSV (Comma delimited) (*.csv) ☺
Formatted Text (Space delimited) (*.prn)
Text (Macintosh) (*.txt)
Text (MS-DOS) (*.txt)
CSV (Macintosh) (*.csv)
CSV (MS-DOS) (*.csv)
DIF (Data Interchange Format) (*.dif)
SYLK (Symbolic Link) (*.slk)
Excel Add-in (*.xlam)
Excel 97-2003 Add-in (*.xla)
PDF (*.pdf)
XPS Document (*.xps)
Strict Open XML Spreadsheet (*.xlsx)
OpenDocument Spreadsheet (*.ods)

Reading in a file

- Getting data into R is relatively simple! Use:

```
read.csv("folderlocation/filename.csv")
read.csv("folderlocation\\filename.csv")
```

NOT `read.csv("folderlocation\filename.csv")`

- You must use a forward slash "/" or 2 back slashes "\\" to designate a folder or directory
  - This is ***different*** than the way Windows shows

- Again, a better function will be read_csv()

SHOW ME THE DATA

OK so you have your data in. How can we look at it?

- Type the name in console (shows first few rows/columns)

- Type `View(dataframe_name)`, or click on the dataframe name in the top right pane (same as View).
  - Once opened, you can now filter, sort, scroll, etc.

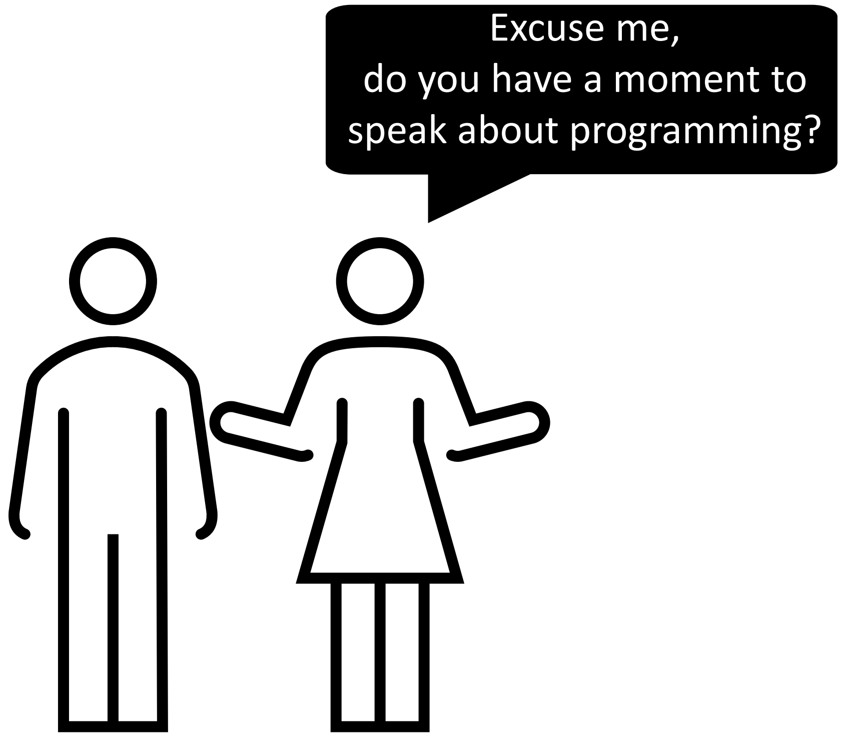
- Check structure: `str(dataframe_name)`

- `summary(dataframe_name)` to see info about dataframe

Data Import

- Simply *importing* the data is the easy part

- What takes time is being thorough, so your data is:
    - Clean, tidy data;
    - Clearly named
    - Correct column “type”



Data Import

Analysis Checklist

- Data is:
  - QA/QC'd
  - Flat file (.csv, etc.)
  - Not summarized
  - One row/observation
- Make a new directory (folder) & RProject

Quiz 3-2

Which of these correctly reads a file “mydata.csv” from folder “data” into R?

```
a) readcsv(“data/mydata.csv”)
b) read.csv(“data/mydata.csv”)
c) read(“data/mydata.csv”)
d) read.csv(“data\mydata.csv”)
```

What is tidy data?

a) All columns are variables
b) One row, one observation
c) Tabled data

QUIZ!

Quiz 3-2 ANSWERS

Which of these correctly reads a file "mydata.csv" from folder "data" into R?

a) `readcsv("data/mydata.csv")`
[✓ b) `read.csv("data/mydata.csv")`]
c) `read("data/mydata.csv")`
d) `read.csv("data\mydata.csv")`

What is tidy data?

[✓ a) All columns are variables]
[✓ b) One row, one observation]
c) "Tabled" data

QUIZ!

Let’s play with data

- Motivating example 2

4 – Basic Data Manipulation

Mmm, the Good Stuff

What is data manipulation

Now that you've got your data into R, you'll usually have to modify it in some way. For example:

- Excluding certain years or species
- Adding a new calculated column (CPUE)
- Pooling groups together

Why do this in R?

- You'll always have a record of what you did to your data
- Your script can repeat process (e.g., if you re-download data out of your database and need to rename data, add a calculated column, etc.)

The best package for this is "dplyr" (& other tidyverse packages)

Filtering: `filter()`

- You'll often need to exclude or subset a group of data
- Use `filter()` to keep certain rows
- You might want to exclude NAs
  - This is a little trickier because something can't be equal to an unknown. Use `is.na()`

```
> filter(dataframename, Year ==
2018, Species != “Pink salmon”)
```

```
> filter(dataframename,
!is.na(Length))
```

```
> filter(dataframename, Year == 2018
| Year == 2019)
```

```
> filter(dataframename, Year == 2018
& Year == 2019)
```

This would return no rows. Why?

Year can't be 2018 AND 2019

Rename Columns: **`rename()`**

- If columns come with spaces in them, R imports them to have backticks, e.g., `Column 1 Name`. Rename these to remove spaces

- *Depending on how you import, you'll want to pay attention to the names*

```
> rename(dataframename,
“newcolumnname1” = “oldcolumnname1”,
“newcolumnname2” = “oldcolumnname2”)
```

```
> rename(dataframename,
“newcolumnname1” = `old column`)
```

Add New Column: **`mutate()`**

- You can add, divide, multiply columns by each other

- Use an “if else” statement to make a conditional column

```
> mutate(dataframename, newcolumn =
column1 + column2)


> mutate(dataframename, newcolumn =
column1 / column2)


> mutate(dataframename, newcolumn =
“Good data”)


> mutate(dataframename, newcolumn =
ifelse(column1 > 1, “what to write
if true", “otherwise write this"))
```

Only Keep Certain Columns: `select()`

- List all columns to keep or add a negative to drop specific columns

```
> dplyr::select(dataframename,
columnname1, columnname2,
columnname3)
```

- Note that you can use `select()` to specify column order

```
> dplyr::select(dataframename,
columnname1, columnname2,
columnname3)
```

(Using the double colon "::" specifies which package the function should come from)

```
> dplyr::select(dataframename,
-columnname1)
> dplyr::select(dataframename,
columnname1:columnname4)
```

Note: There can occasionally be a conflict between dplyr's select and another package. To ensure that you are using the right select, use dplyr::select(). This is a *very* rare issue however.

The “pipe”

- dplyr also brings use the functionality of a “pipe”, written as %>%
- The pipe operator allows us to string together several functions and pass each of them to the next one

```
> dataframename <- filter(dataframename, Year == 2018)
> dataframename <- mutate(dataframename, surveymonth = month(surveydate))
> dataframename <- select(dataframename, Year, surveymonth, totalcount)

> dataframename <- dataframename %>% filter(Year == 2018) %>%
                         mutate(surveymonth = month(surveydate)) %>%
                         select(Year, surveymonth, totalcount)
```

%>% advantages

The pipe passes the object from the left to the right. Most often this passes a dataframe as an argument from the left to the right.

Stringing together many lines of code at once is:

- more readable,
- doesn’t repeat the same variables over and over,
- helps prevent “out of order” code problems,
- reduces excessive nesting of functions within functions

Replace a value: **`recode()`**

- You may often want to replace a value in the dataframe.
  - E.g., In column “Species” you want to change “coho” to “Coho Salmon”

```
dataframename %>%
   mutate(Species =
   recode(Species, “coho” = “Coho
   Salmon”))
```

- recode() is for replacing a value

```
dataframename %>%
   mutate(columnname =
   recode(columnname, “oldvalue” =
   “new and improved value”))
```

- replace_na() replaces all NAs in a column. (From package “tidyr”)
  - Be aware of the potential issue with replacing all NAs!

```
dataframename %>% mutate(columnname
   = replace_na(columnname, “what
   you want it replaced with”))
```

Quiz 4

Filter a dataframe named “df” to remove NAs from column “leng”

a) `filter(df, leng, na.rm=TRUE)`

b) `filter(df, leng, !NA)`

c) `filter(df, !is.na(leng))`

d) `filter(df$leng, is.na)`

e) `df %>% filter(!is.na(leng))`

Can test your answers in R using other existing dataframes

From dataframe “df”, calculate a new column of CPUE from columns “catch” and “effort”

a) `mutate(df, CPUE = catch / effort)`

b) `newcolumn(df %>% CPUE = catch / effort)`

c) `df$CPUE <- mutate(df$catch / effort)`

d) `df$CPUE <- Mutate(df, CPUE = catch / effort)`

QUIZ!

Quiz 4 ANSWERS

Filter a dataframe named “df” to remove NAs from column “leng”

a) `filter(df, leng, na.rm=TRUE)`

b) `filter(df, leng, !NA)`

[✓ c) `filter(df, !is.na(leng))`]

d) `filter(df$leng, is.na)`

[✓ e) `df %>% filter(!is.na(leng))`]

Why didn’t these run?

Only “is.na()” can find NAs

From dataframe “df”, calculate a new column of CPUE from columns “catch” and “effort”

[✓ a) `mutate(df, CPUE = catch / effort)`]

b) `newcolumn(df %>% CPUE = catch / effort)`

c) `df$CPUE <- mutate(df$catch / effort)`

d) `df$CPUE <- Mutate(df, CPUE = catch / effort)`

Why didn’t these run?

Incorrect use of $, wrong mix/match of pipe, capitalization

QUIZ!

Review & Day 1 Break

OH THANK GOODNESS

Review – WELL DONE

- First, you need to congratulate yourself! Today was TOUGH
- We covered a LOT of material. Great job covering 1–2 months of material in just a day
- Monday will be when everything all comes together

Review – About R

- RStudio 4 panes: Script editor, console, environment, plot/help
- Use <- or = to set (“save”) a variable
- Code must end on comma or similar
- A “function” is (usually) made up of argument(s)
- Variables (dataframe) are set to access later
- str(), $, [], c()

Review – Basics

- Errors
- NAs
- mean(), sum(), log()
- Make sure to use quotes when you describe a character
- Use descriptive variable names
- & means “AND”, | means “OR”
- A dataframe is group of columns and rows

Review – Working with your data

- Directories – Make sure that you are in the correct directory
  - An RProject makes this very simple
- Packages – Add more functions for us to use
  - Load them for use by using library()
- Tidy data – One row per observation, not summarized
- Check that structure of data matches what it should be
  - Use str(), especially checking for character vs factor or numerical vs character

Review – Data manipulation

- The basic commands to modify your data are:

select()	*Keep or remove columns*
filter()	*Keep rows that meet your conditions*
rename()	*Change column names*
mutate()	*Make new columns*
recode()	*Replace values*

- The pipe, %>%, makes your code cleaner and shorter

Review – Monday

- If today was difficult, that's expected! Pat yourself on the back!
- Remember our goals. By the end of Monday you will be able to:
  - Import your OWN data into R
  - Perform basic analyses
  - Make publication worthy figures
- If you didn't finish a learnr section, revisit it over the weekend
- Look at the dataset to bring to Monday's class. Is it tidy? Is it tabled?
- Questions or confused about anything? PLEASE email me