

Welcome to Workshop: Introduction to R, Rstudio, and Data

- I. Please sign in on the sign in sheet (this is so I can follow up to get feedback).
- II. If you haven't already, download R and Rstudio, install to your laptop.
- III. Download materials you'll need from my website (<http://faculty.washington.edu/jhrl/Teaching.html>) or google Janneke HilleRisLambers at University of Washington – go to Teaching tab, scroll down (zip file). Or ask me for a USB stick.

Introduction to R, Rstudio, and coding

I. What / Why R?

II. Rstudio & R

- A. The Source, Console, Help and Environment panes
- B. Functions and Data Objects

III. Getting started

- A. Data & Project Management
- B. Good Coding practice

IV. Data wrangling

- A. ChickenScript.R; A demo of how to read in and examine data, merge and subset, define variables.
- B. Nutnet data: explore in pairs

V. Further topics & Resources

But first, brief introductions...



Walker Endowed Professor of Natural History
University of Washington, Seattle (USA)

My Goals:

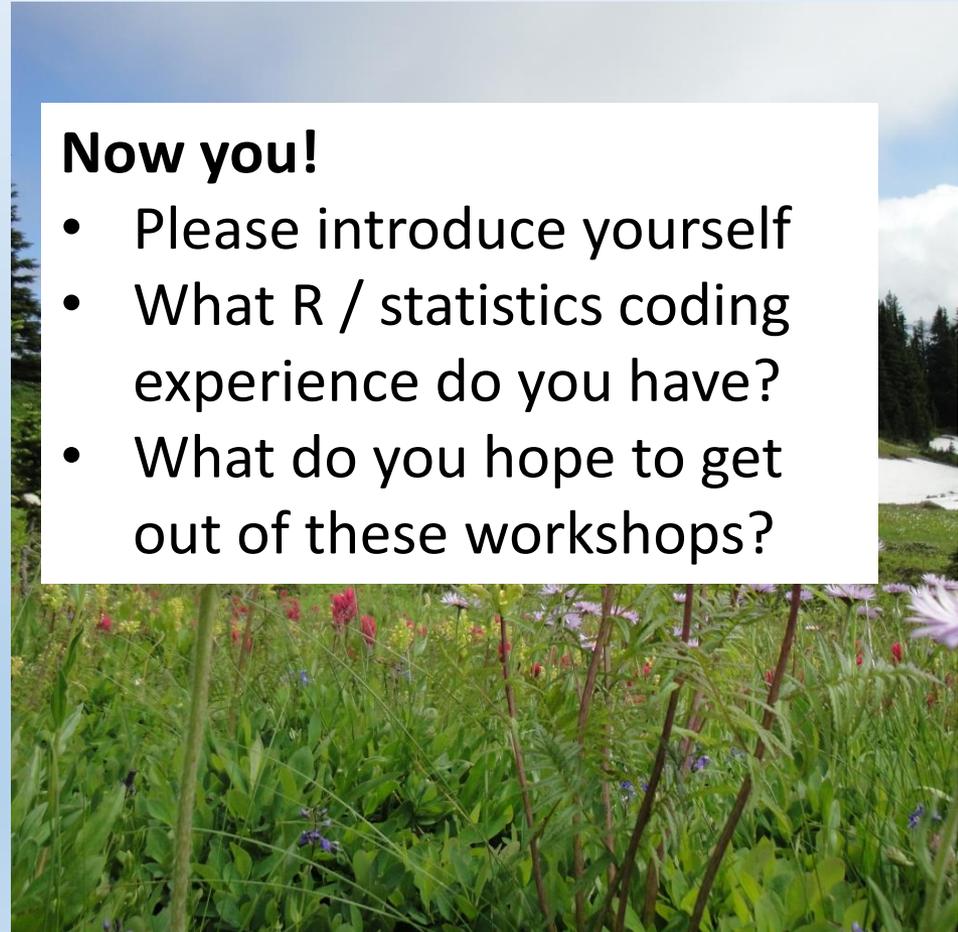
Introduction (no background required)
Not just coding / statistics (e.g. project
management, experimental design)
Collaborative: help each other
Feedback (what worked, what didn't)

Research interests:

Plant Community Ecology, Global Change
Statistics / Coding: since graduate school

Now you!

- Please introduce yourself
- What R / statistics coding experience do you have?
- What do you hope to get out of these workshops?

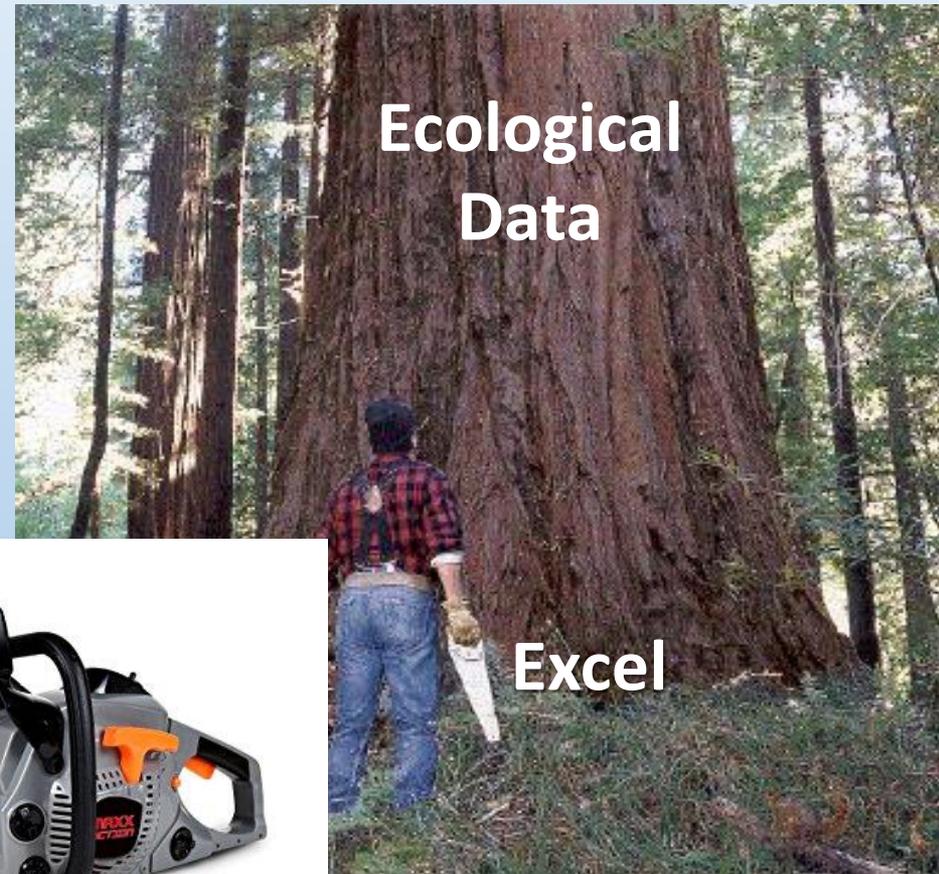


I. What is R?

- Computer language & environment for statistical computing & graphics. Script based (text computer code), not GUI based (menu / point & click).
- Tools for **Data Handling** and manipulation
- Large collection of statistical tools (packages) for **Data Analysis**; contributed by many experts
- Graphical interface for **Visualizing Data** & results from statistical analyses
- Relatively simple and effective, widely used, free, open source...

I. Why R?

- The right tool for (many of our) jobs

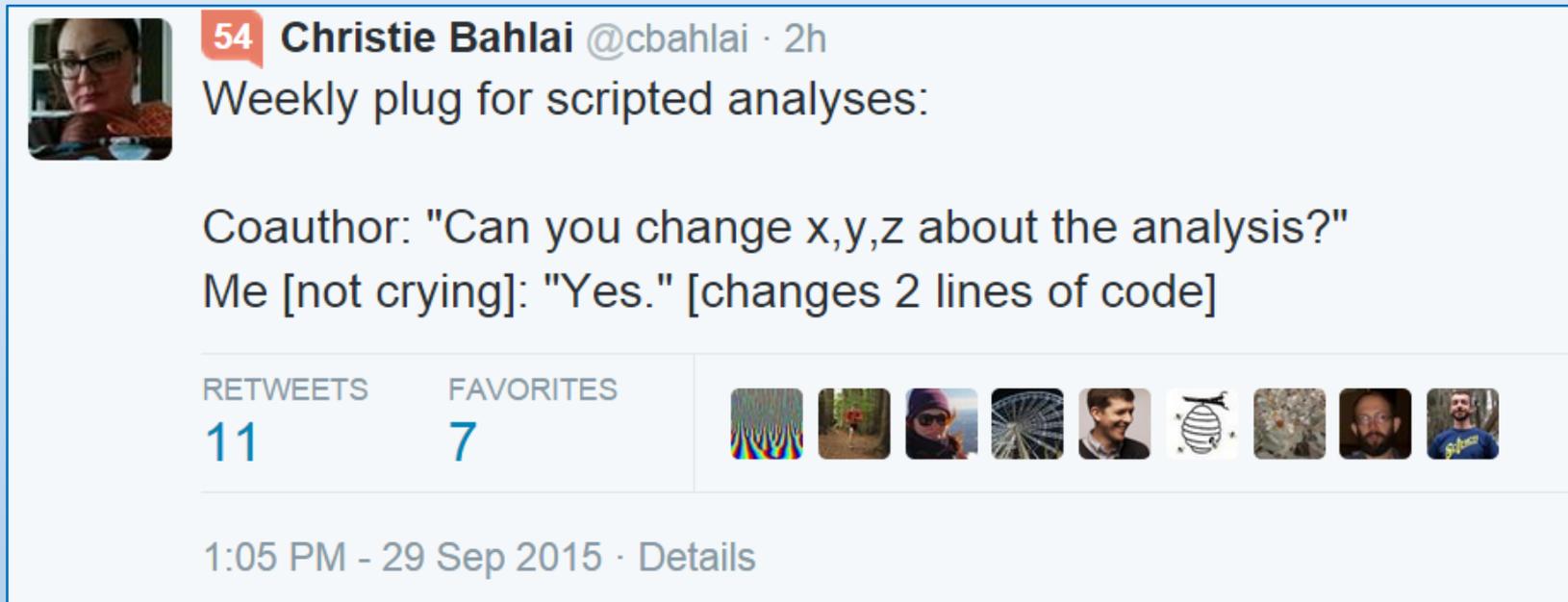


R statistics?



I. Why R?

- The right tool for (many of our) jobs
- Reproducible, shareable code & tools for collaboration



 **54** Christie Bahlai @cbahlai · 2h

Weekly plug for scripted analyses:

Coauthor: "Can you change x,y,z about the analysis?"
Me [not crying]: "Yes." [changes 2 lines of code]

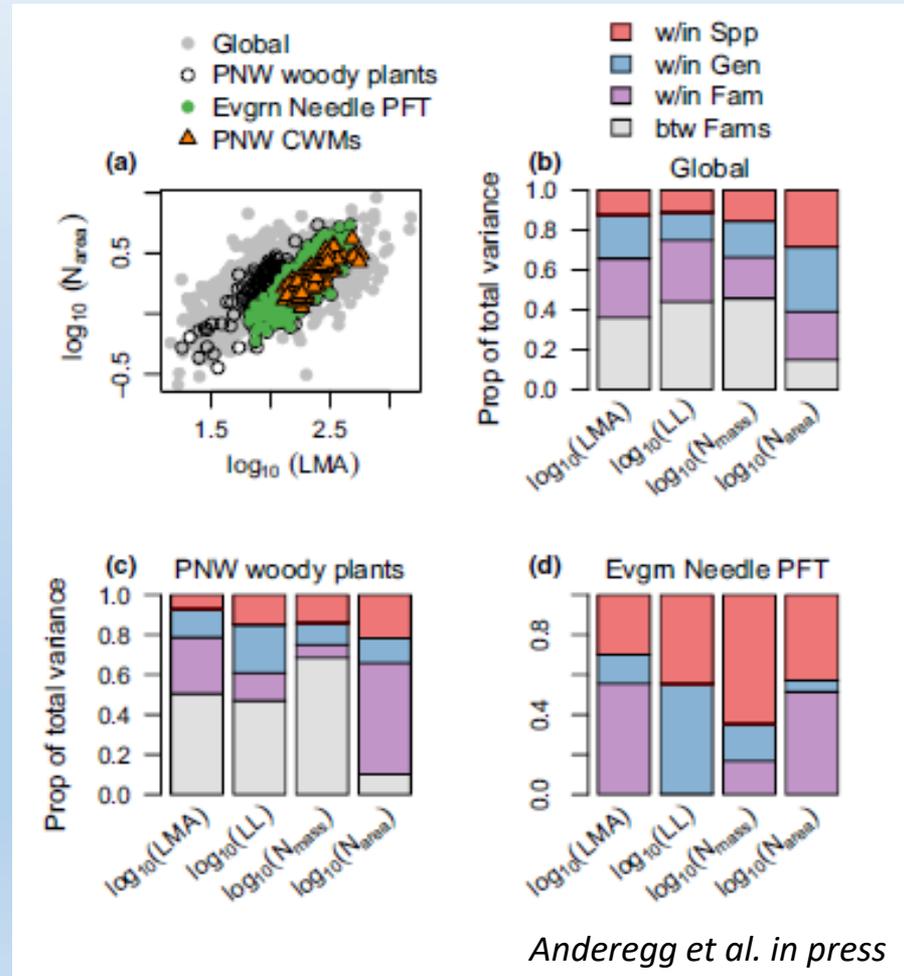
RETWEETS 11 FAVORITES 7

1:05 PM - 29 Sep 2015 · Details

Rule of thumb: every analysis you do on a dataset will have to be redone 10–15 times before publication. Plan accordingly.

I. Why R?

- The right tool for (many of our) jobs
- Reproducible, shareable code & tools for collaboration
- Publication quality plots (also easily reproducible)



I. Common uses of R

1. Explore data via summaries, plots or classical statistical analyses: ANOVA, LM, GLM, ...
2. Advanced analyses: Bayesian inference, random forests, spatial, mixed effects, ... (via packages)
3. Publication quality figures
4. Larger projects (e.g. publication): functions, scripts, documentation, reproducibility
5. Build your own R package and share via github.com

I. Learning R...

- R is a programming language, the learning curve can be steep so be patient
- Like human languages, what you get out is what you put in
- Increased productivity when fluent
- Many sources of help: online, books, labmates, etc.

NOTE:

- You learn to program by making mistakes
- Expect to make errors, learn from them, avoid them, but don't let errors frustrate you

II. Rstudio: what is it?

- Portal through which to use R (IDE).
- Simultaneously write code (in a script), execute code line by line, manage data, get help, view plots

LET'S GET STARTED!

These are instructions

Do / look / find this

> Type this (but not the >)

This is something useful / important

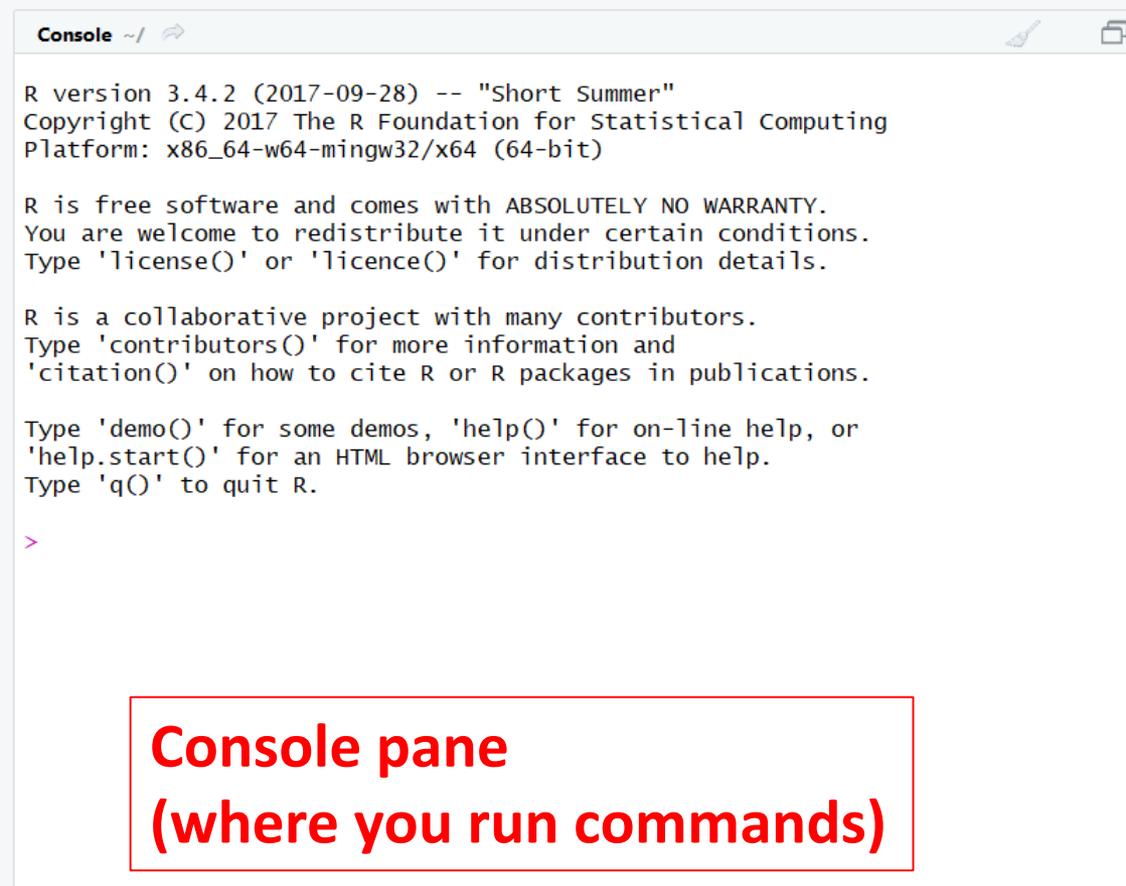
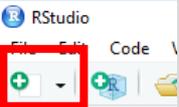
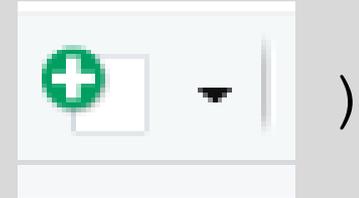
Instruction:

Open Rstudio

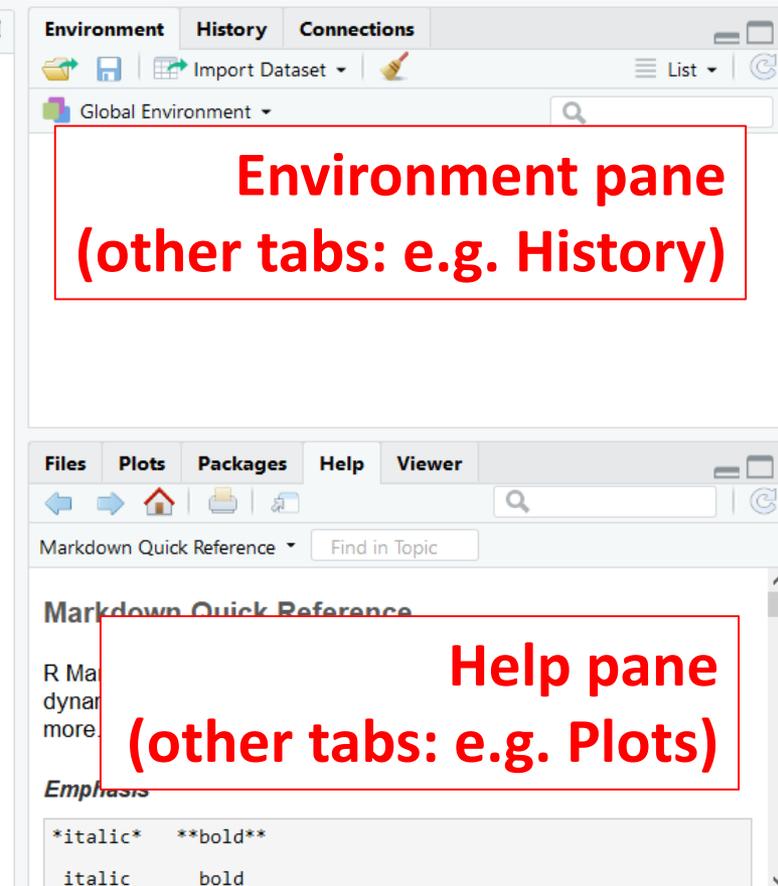
II. Rstudio

Instruction:

- Click on the Green Plus (i.e
- Then, click on R script

A screenshot of the RStudio Console pane. The text displayed is the R startup message, including the version (3.4.2), copyright (2017), and license information. The console prompt is shown as '>'. A red rectangular box is overlaid on the bottom part of the console, containing the text 'Console pane (where you run commands)'.

```
R version 3.4.2 (2017-09-28) -- "Short Summer"  
Copyright (C) 2017 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.  
  
>
```

A screenshot of the RStudio Environment and Help panes. The Environment pane is active, showing the 'Global Environment' tab. A red rectangular box is overlaid on the Environment pane, containing the text 'Environment pane (other tabs: e.g. History)'. Below it, the Help pane is visible, showing a 'Markdown Quick Reference' section. A red rectangular box is overlaid on the Help pane, containing the text 'Help pane (other tabs: e.g. Plots)'.

Environment History Connections

Global Environment

Environment pane
(other tabs: e.g. History)

Files Plots Packages Help Viewer

Markdown Quick Reference Find in Topic

Markdown Quick Reference

R Ma
dynam
more

Help pane
(other tabs: e.g. Plots)

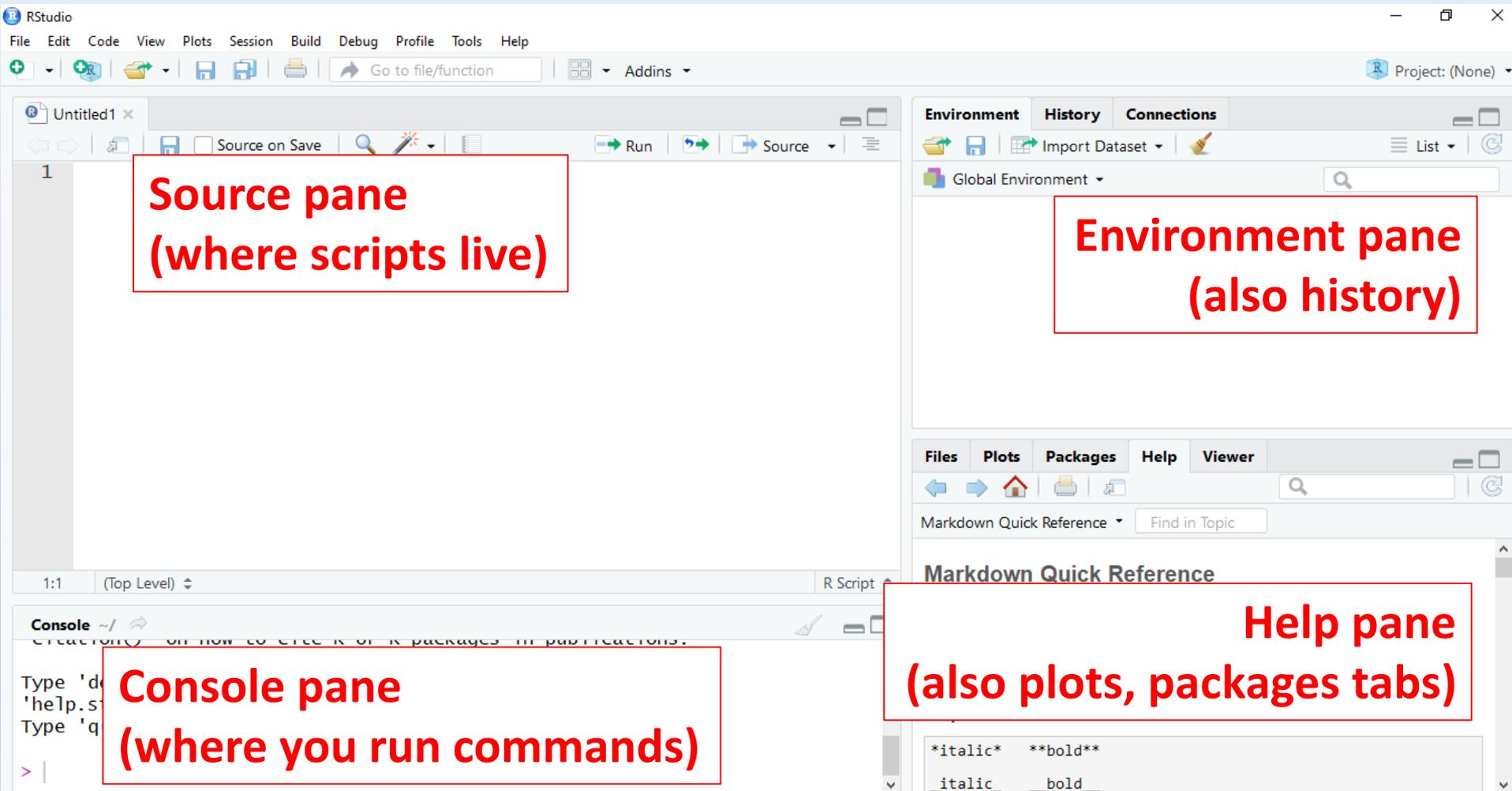
Emphasis

italic ****bold****

italic **__bold__**

II. Rstudio: the four panes

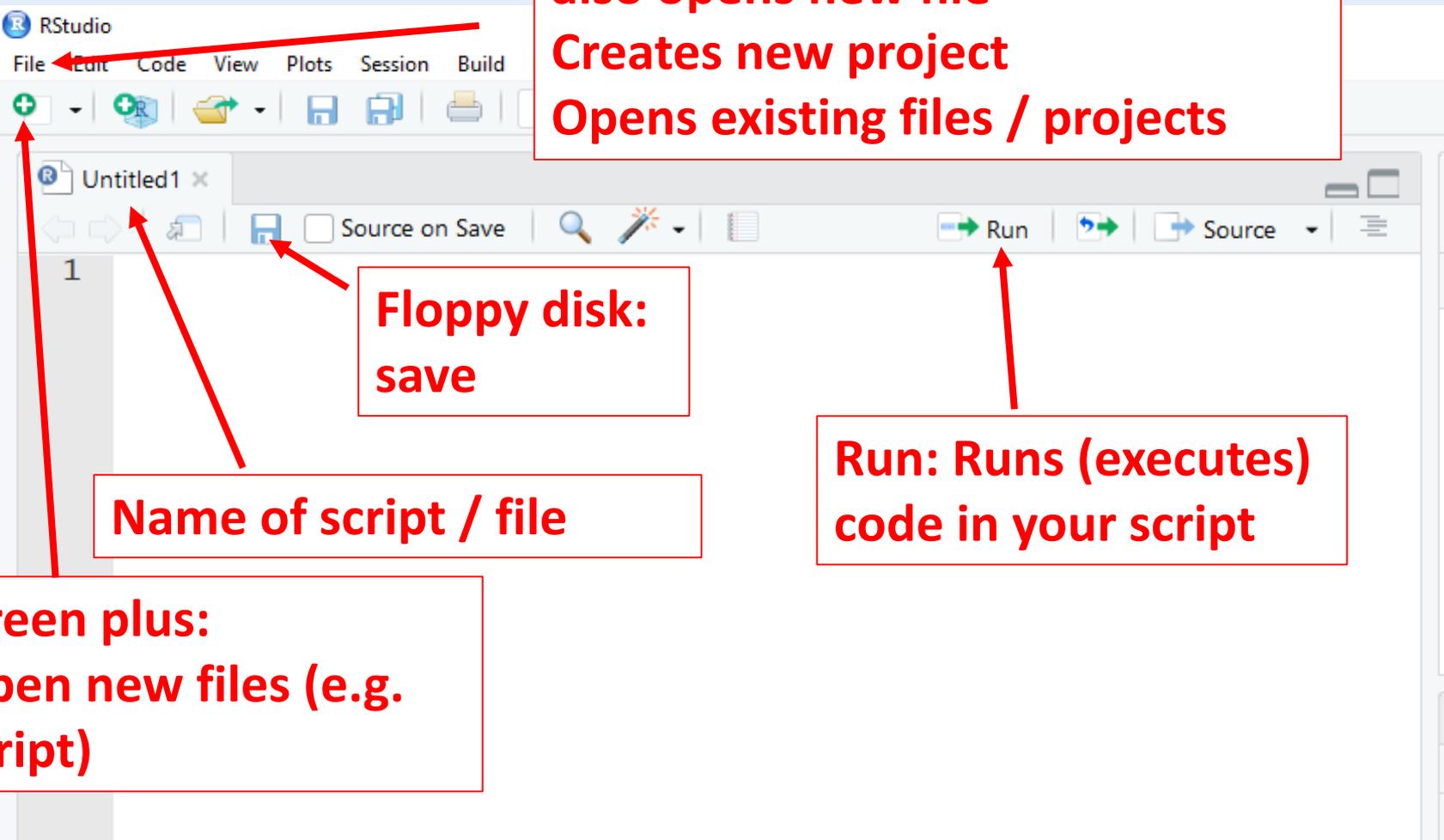
- This is what you'll usually see



II. Rstudio: the **source** pane

- Scripts (your code)

File:
also opens new file
Creates new project
Opens existing files / projects



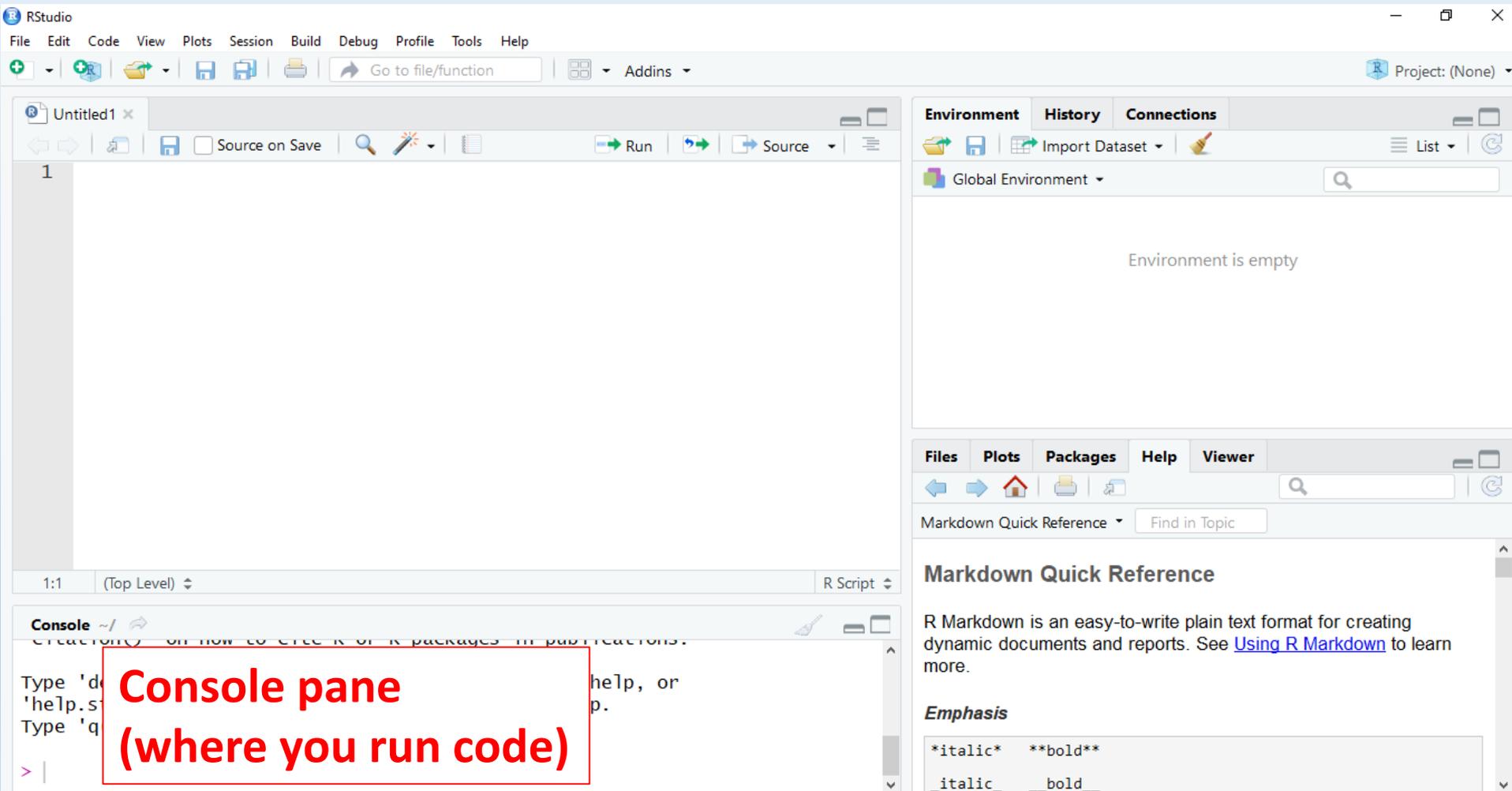
Floppy disk:
save

Name of script / file

**Run: Runs (executes)
code in your script**

Green plus:
Open new files (e.g.
script)

II. Rstudio: the console pane



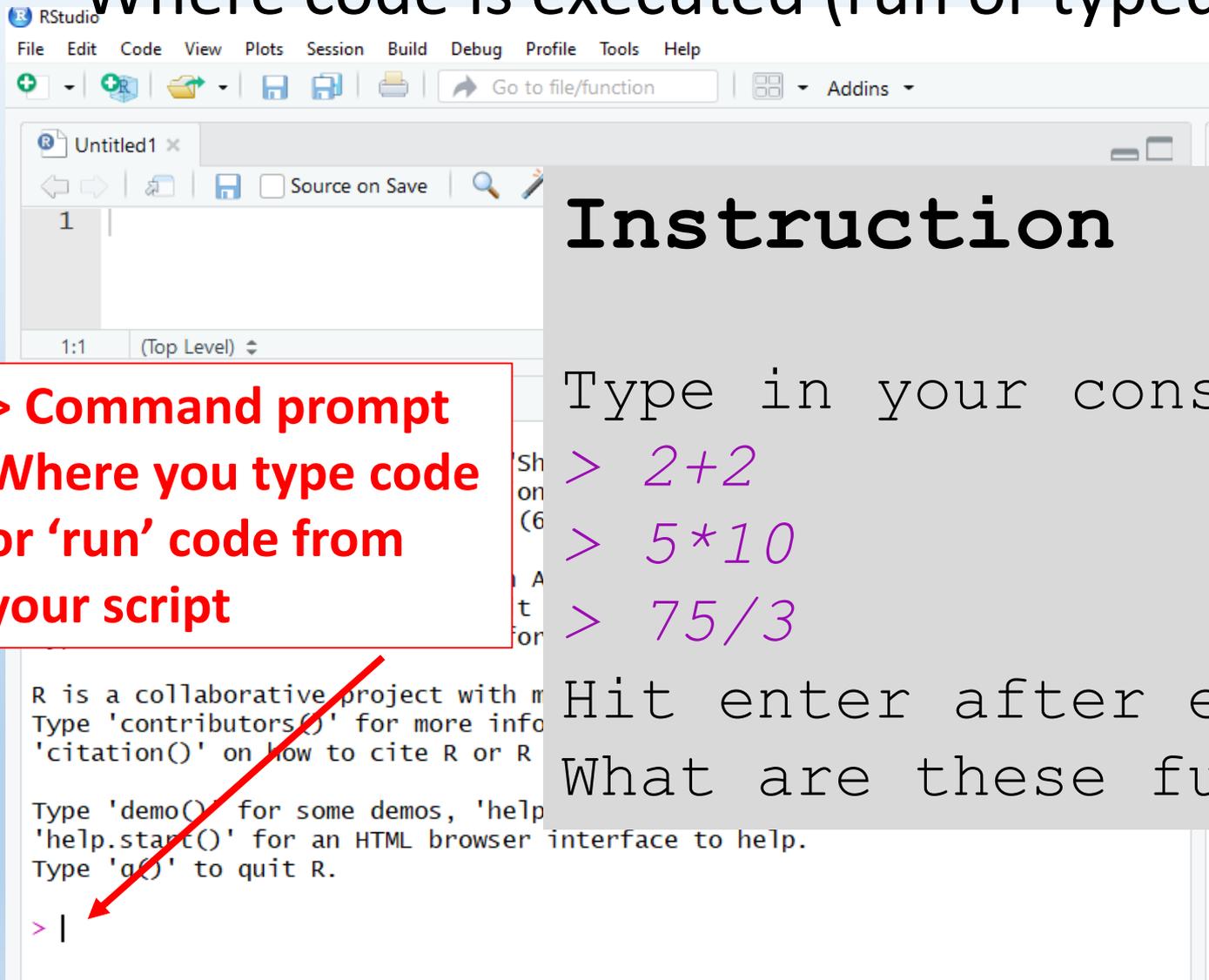
The image shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The toolbar contains icons for file operations and a search bar. The main editor pane shows a file named 'Untitled1' with a single line of code '1'. The right-hand side has three panes: Environment (showing 'Global Environment' and 'Environment is empty'), History, and Connections. Below these are panes for Files, Plots, Packages, Help, and Viewer. The Help pane is displaying a 'Markdown Quick Reference' with text about R Markdown and examples of emphasis formatting. The bottom pane is the Console, which is highlighted with a red box and contains the text 'Console ~/ >' and a red box with the text 'Console pane (where you run code)'.

Console ~/ >

**Console pane
(where you run code)**

II. Rstudio: the **console** pane

- Where code is executed (run or typed)



The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu bar is a toolbar with icons for file operations and a search bar labeled 'Go to file/function'. The main editor area shows a file named 'Untitled1' with a cursor on line 1. The console pane at the bottom displays the R prompt '> |' and some introductory text about R. A red box highlights the console pane with the text '> Command prompt Where you type code or 'run' code from your script'. A red arrow points from this box to the console prompt in the screenshot.

Instruction

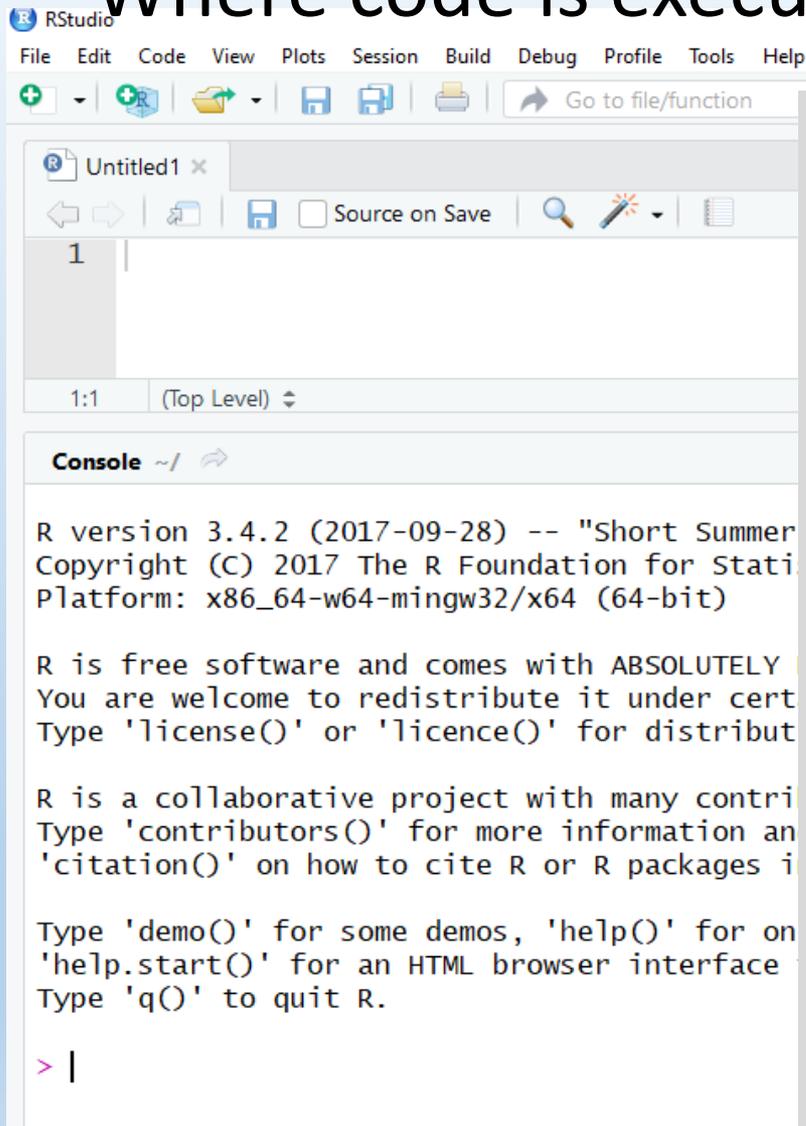
Type in your console:

```
> 2+2  
> 5*10  
> 75/3
```

Hit enter after each line.
What are these functions?

II. Rstudio: the **console** pane

- Where code is executed (run or typed)



The screenshot shows the RStudio interface. At the top is a menu bar with options: File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help. Below the menu bar is a toolbar with icons for file operations and a search bar labeled 'Go to file/function'. The main editor area shows a file named 'Untitled1' with a single line of code at line 1. Below the editor is the console pane, which displays the R version information and a prompt for user input.

```
R version 3.4.2 (2017-09-28) -- "Short Summer  
Copyright (C) 2017 The R Foundation for Stati  
Platform: x86_64-w64-mingw32/x64 (64-bit)  
  
R is free software and comes with ABSOLUTELY  
You are welcome to redistribute it under cert  
Type 'license()' or 'licence()' for distribut  
  
R is a collaborative project with many contri  
Type 'contributors()' for more information an  
'citation()' on how to cite R or R packages i  
  
Type 'demo()' for some demos, 'help()' for on  
'help.start()' for an HTML browser interface  
Type 'q()' to quit R.  
  
> |
```

Instructions

Type:

> 25/a

> log(-1)

*> (3*25)/*

What is going on
at each line?

II. Rstudio: the console pane

```
Console ~/   
>  
> 25/a  
Error: object 'a' not found  
>  
>  
> log(-1)  
[1] NaN  
Warning message:  
In log(-1) : NaNs produced  
>  
>  
> (3*25)/  
+  
+  
+ |
```

Error message: check and fix

Warning message: check and understand, perhaps fix

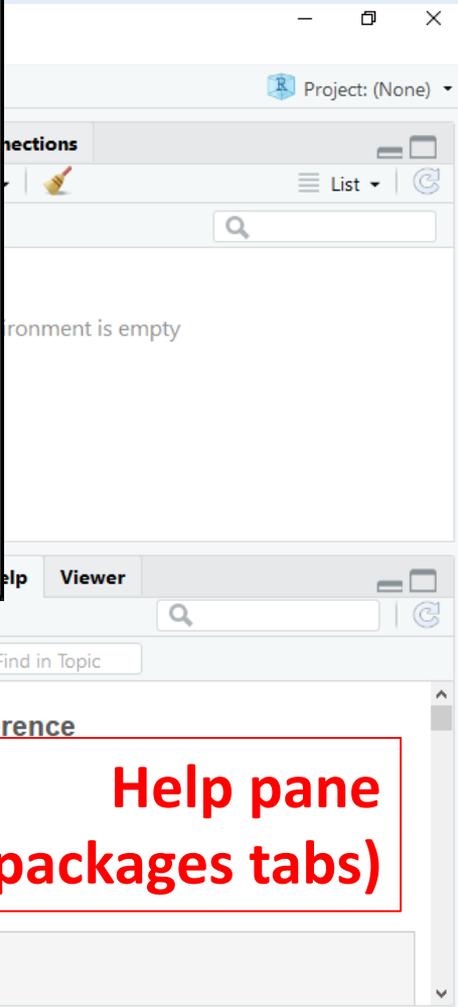
+ sign: R is expecting more from you

**Hit 'escape' to escape!
Also if R is generally bogged down...**



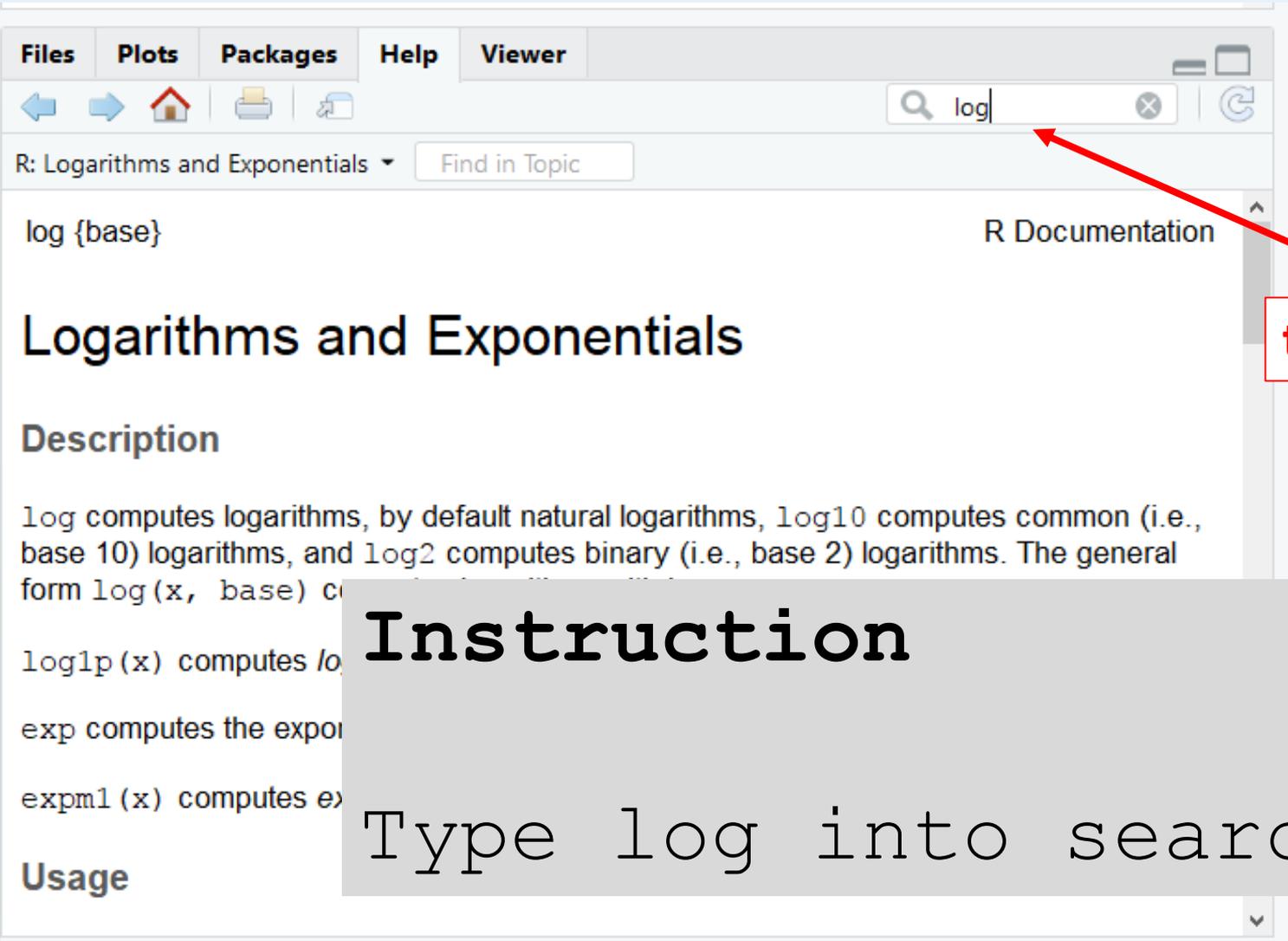
II. Rstudio: ...Plots/Packages/Help... pane

NEVER HAVE I FELT SO CLOSE TO ANOTHER SOUL AND YET SO HELPLESSLY ALONE AS WHEN I GOOGLE AN ERROR AND THERE'S ONE RESULT A THREAD BY SOMEONE WITH THE SAME PROBLEM AND NO ANSWER LAST POSTED TO IN 2003



II. Rstudio: ...Plots/Packages/Help... pane

- Use to get Help. Option 1: type in **Help** pane search bar



type here

Instruction
Type log into search bar

II. Rstudio: ...Plots/Packages/Help... pane

R: Logarithms and Exponentials Find in Topic

log {base} R Documentation

Logarithms and Exponentials

Description What the function does in general terms

log computes logarithms, by default natural logarithms, log10 computes common (i.e., base 10) logarithms, and log2 computes binary (i.e., base 2) logarithms. The general form log(x, base) computes logarithms with base base.

log1p(x) computes log(1+x) accurately also for |x| << 1 (and less accurately when x is approximately -1).

exp computes the exponential function.

expm1(x) computes exp(x) - 1 accurately also for |x| << 1.

Usage How to use the function

```
log(x, base = exp(1))
logb(x, base = exp(1))
log10(x)
log2(x)

log1p(x)

exp(x)
expm1(x)
```

Arguments What does the function need

x a numeric or complex vector.

base a positive or complex number: the base with respect to which logarithms are computed. Defaults to e=exp(1).

Details

All except logb are generic functions: methods can be defined for them individually or via the Math group generic.

log10 and log2 are only convenience wrappers, but logs to bases 10 and 2 (whether computed via log or the wrappers) will be computed more efficiently and accurately where supported by the OS. Methods can be set for them individually (and otherwise methods for log will be used).

logb is a wrapper for log for compatibility with S. If (S3 or S4) methods are set for log they will be dispatched. Do not set S4 methods on logb itself.

All except log are primitive functions.

R: Logarithms and Exponentials Find in Topic

Value What does the function return

A vector of the same length as x containing the transformed values. log(0) gives -Inf, and log(x) for negative values of x is NaN. exp(-Inf) is 0.

For complex inputs to the log functions, the value is a complex number with imaginary part in the range [-pi, pi]: which end of the range is used might be platform-specific.

S4 methods

exp, expm1, log, log10, log2 and log1p are S4 generic and are members of the Math group generic.

Note that this means that the S4 generic for log has a signature with only one argument, x, but that base can be passed to methods (but will not be used for method selection). On the other hand, if you only set a method for the Math group generic then base argument of log will be ignored for your class.

Source

log1p and expm1 may be taken from the operating system, but if not available there are based on the Fortran subroutine dlnrel by W. Fullerton of Los Alamos Scientific Laboratory (see <http://www.netlib.org/slatec/flnlib/dlnrel.f> and (for small x) a single Newton step for the solution of log1p(y) = x respectively.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole. (for log, log10 and exp.)

Chambers, J. M. (1998) *Programming with Data. A Guide to the S Language*. Springer. (for logb.)

See Also Discover other related functions

[Trig](#), [sqrt](#), [Arithmetic](#).

Examples Sample code showing how it works

```
log(exp(3))
log10(1e7) # = 7

x <- 10^-(1+2*1:9)
cbind(x, log(1+x), log1p(x), exp(x)-1, expm1(x))
```

[Package base version 3.0.1 [Index](#)]

20

II. Rstudio: ...Plots/Packages/Help... pane

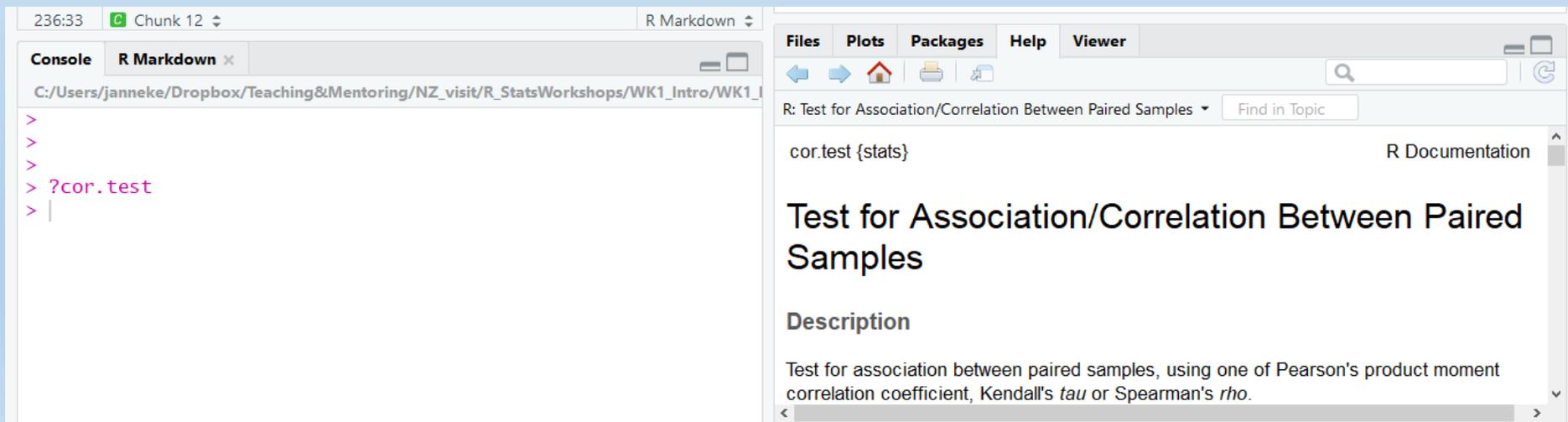
Option 2: type in **Console**, examine in **Help** pane

Instruction

Type in console

```
> ?cor.test
```

Examine output in Help pane



The screenshot displays the RStudio interface. On the left, the **Console** pane shows the command `?cor.test` entered. On the right, the **Help** pane displays the documentation for `cor.test`, titled "Test for Association/Correlation Between Paired Samples". The description states: "Test for association between paired samples, using one of Pearson's product moment correlation coefficient, Kendall's *tau* or Spearman's *rho*."

II. Rstudio: ...Plots/Packages/Help... pane

Option 3: type in **Console**, examine in **Help** pane

Instruction

Type in console

```
> ??"analysis of variance"
```

Examine in Help pane

Quotes are critical here

The screenshot shows the RStudio interface. On the left, the Console pane displays the command `> ??"analysis of variance"` with a red arrow pointing from the text 'Quotes are critical here' to the double quotes. On the right, the Help pane shows search results for the command, including the R logo and a list of help pages: [stats::aov](#) (Fit an Analysis of Variance Model), [stats::eff.aovlist](#) (Compute Efficiencies of Multistratum Analysis of Variance), and [stats::manova](#) (Multivariate Analysis of Variance).

II. Rstudio: The Environment pane

The image shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The toolbar contains icons for file operations and a search bar. The main editor window shows a script with the number '1' on the first line. The Environment pane on the right is highlighted with a red box and contains the text "Environment pane (also history)". Below the Environment pane is the Files pane, which shows a "Markdown Quick Reference" section. The console at the bottom displays the R prompt and some help text.

Environment pane (also history)

Markdown Quick Reference

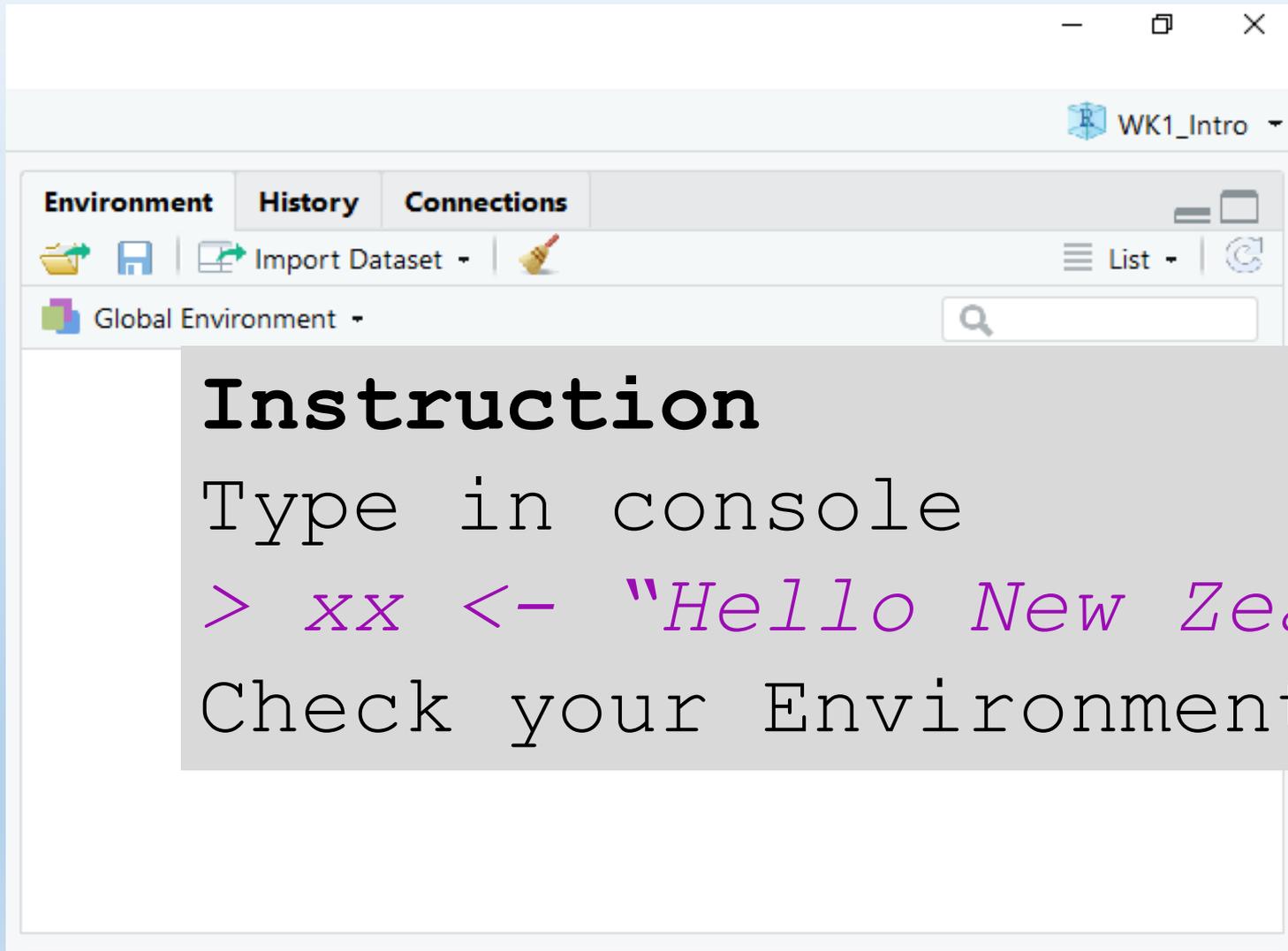
R Markdown is an easy-to-write plain text format for creating dynamic documents and reports. See [Using R Markdown](#) to learn more.

Emphasis

```
*italic*  **bold**  
_italic_  __bold__
```

II. Rstudio: The **Environment** pane

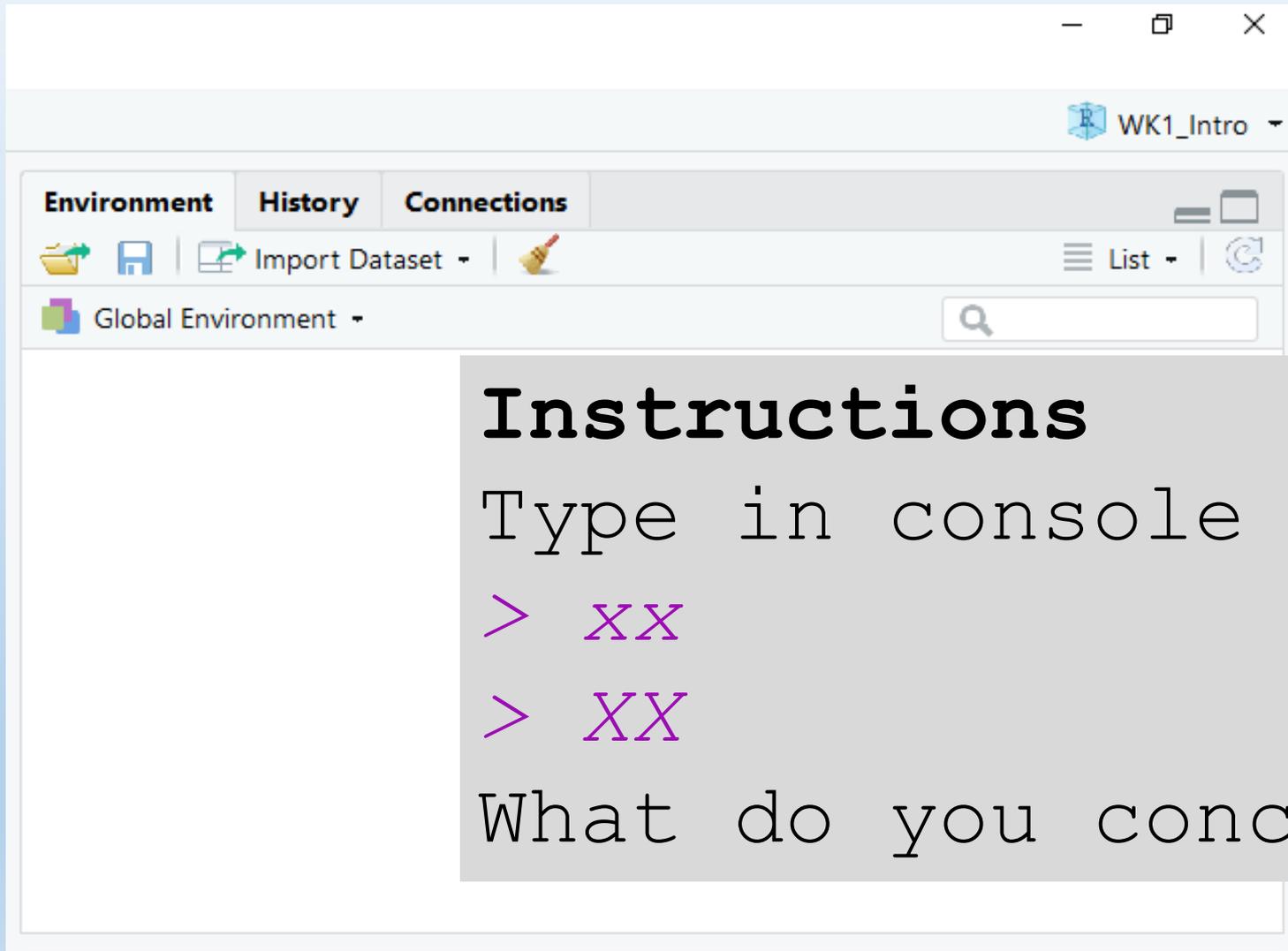
- All objects you've identified, created show up here

A screenshot of the RStudio interface showing the Environment pane. The pane is titled 'Global Environment' and contains a search bar. The Environment pane is currently empty. The console pane below it contains the following text:

```
Instruction
Type in console
> xx <- "Hello New Zealand"
Check your Environment pane
```

II. Rstudio: The **Environment** pane

- All objects you've identified, created show up here



II. R & Rstudio: functions and data objects

- A **function** is a chunk of code that conducts a specific task. R comes with many built in functions (e.g. cor.test, log, print, etc).

Instructions

Use help to look up these functions:

```
<-
```

```
read.csv
```

```
write.csv
```

```
plot
```

```
print
```

- Eventually, you may want to write your own **functions** (for tasks you repeat often)

II. R & Rstudio: functions and data objects

- Data comes in many types (i.e. **modes**) recognized by R and Rstudio. You'll use the first three most.

Data Type	Also know as:	Example
Numeric	float	42, 3.14, -19.2
Character	string or text	"a", "block1", "red", "Pinus palustris"
Logical	boolean	TRUE, FALSE
Integer		
Complex		
Raw		

II. R & Rstudio: functions and data objects

- A **vector** is a collection of items, that are all of the same type (the latter is important). We can create vectors and assign them to names using `c()`:

Instructions

Write in your console:

```
> example_vector <- c("beer", "cheese")  
> mode(example_vector)  
> example_vector
```

- Notice `example_vector` has now been added to your Environment.
- What does the `c` in `c()` refer to?

II. R & Rstudio: functions and data objects

- You can access the different elements within **vectors** with square brackets and numbers.

Instructions

Write in your console:

```
> example_vector[1]  
> example_vector[2]
```

- You can add to and replace elements in **vectors** using the <- assignment operator:

Write in your console:

```
> example_vector[1] <- "wine"  
> example_vector[3] <- "crackers"  
> example_vector
```

II. R & Rstudio: functions and data objects

- You can also create **vectors** with functions called `rep` and `seq` (stands for repeat and sequence).

Instructions

Write in your console, and examine:

```
> rvect <- rep(1, times=10)
> svect1 <- seq(1,10)
> svect2 <- seq(1,20,by=2)
```

Instructions

Create the following using `seq` and `rep` (and `c()` as little as possible):

- Odd integers between 1 and 99
- The numbers 1,1,1,2,2,2,3,3,3

II. R & Rstudio: functions and data objects

- **Matrices** are 2-dimensional collections of data. Like vectors, they only have 1 data type in them.

Instructions 1

Write in your console:

```
> example_matrix <- matrix(svect1,nrow=5,ncol=2)
> example_matrix
```

- Extract rows and columns (vectors), and cells from **matrices** using square brackets [row,col]:

Instructions

Extract from `example_matrix` (in your console) the 2nd column, the 5th row, and cell that is in the 3rd row and 1st column in 3 commands

II. R & Rstudio: functions and data objects

- **Data frames** are 2-dimensional, like matrices, but can hold several types of data. Try the following:

Instructions

Write in your console:

```
> spp <- c("Anemone", "Avalanche Lily",  
"Subarctic Lupine", "Mountain Daisy")  
> flowercol <- c("white", "white", "purple", "pink")  
> visits <- c(3, 5, 25, 7)  
> PlantDat <- data.frame(spp, flowercol, visits)
```

- As with matrices, rows / cols can be extracted with brackets. Columns can also be extracted by name:

Write in your console:

```
> PlantDat$visits
```

II. R & Rstudio: functions and data objects

- When you read a .txt or .csv file into R, it will be read in as a **data frame**, from which you can extract columns of data (vectors) as response and explanatory variables. These can also be transformed (e.g. if visits were monitored over 15 minutes and we want a per hour estimate):

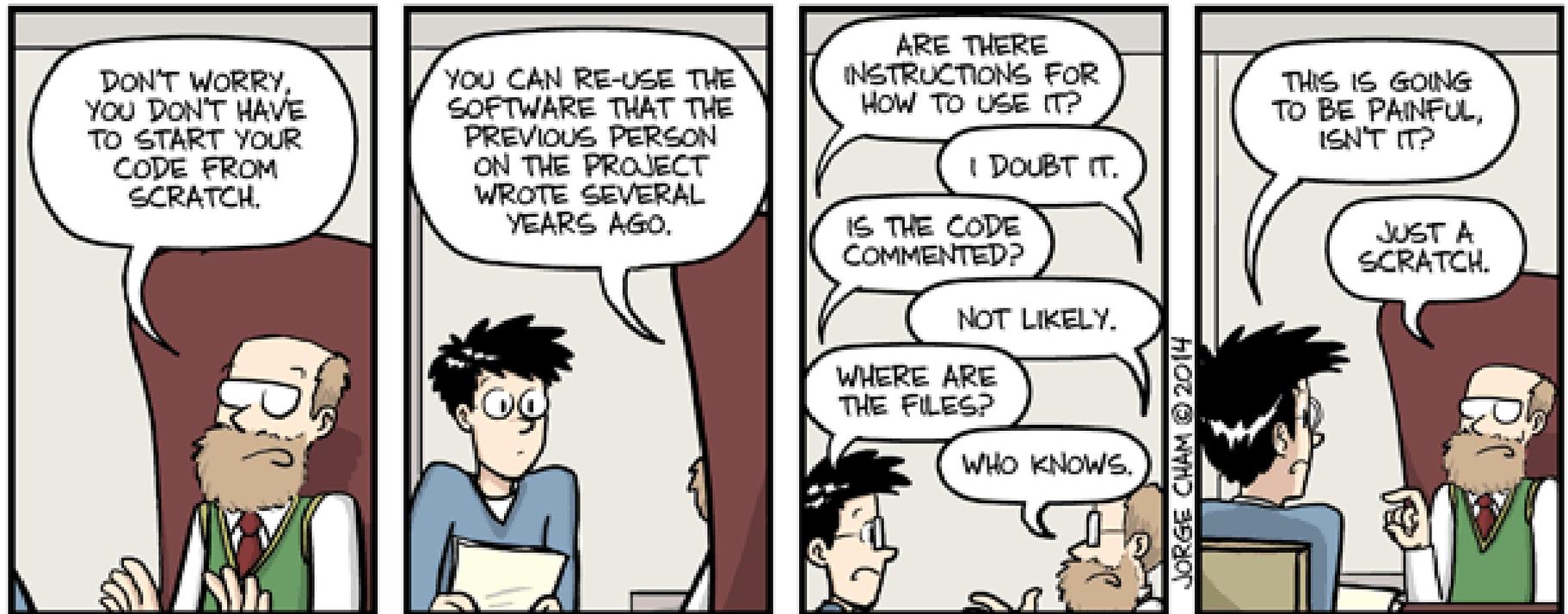
Instructions

Create a vector called `visitsperhour` from `PlantDat`.

```
> visitsperhour <- PlantDat$visits * 4
```

III. Organization = reproducibility (& clarity)

- Organize your files (R Projects)
- Organize your code (comments, structure)
- Presumption: you've organized your data (we'll talk about that if we have time)...



WWW.PHDCOMICS.COM

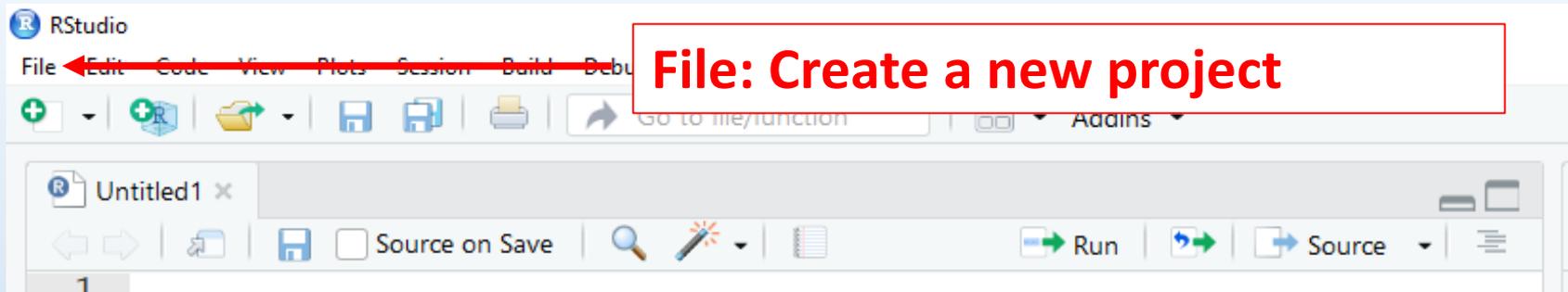
III. Getting Started: Project Management

SUGGESTIONS

- Create a directory for each project, naming it something informative (to you).
- Keep your data files, R script(s) and output files (e.g. figures, simulation results, summary tables) together in that directory (potentially w/ subfolders called data, results, etc).
- Treat data as read only, output as disposable (i.e. R generates everything from your script and data files)

**Potential Tool to do this:
create an R project**

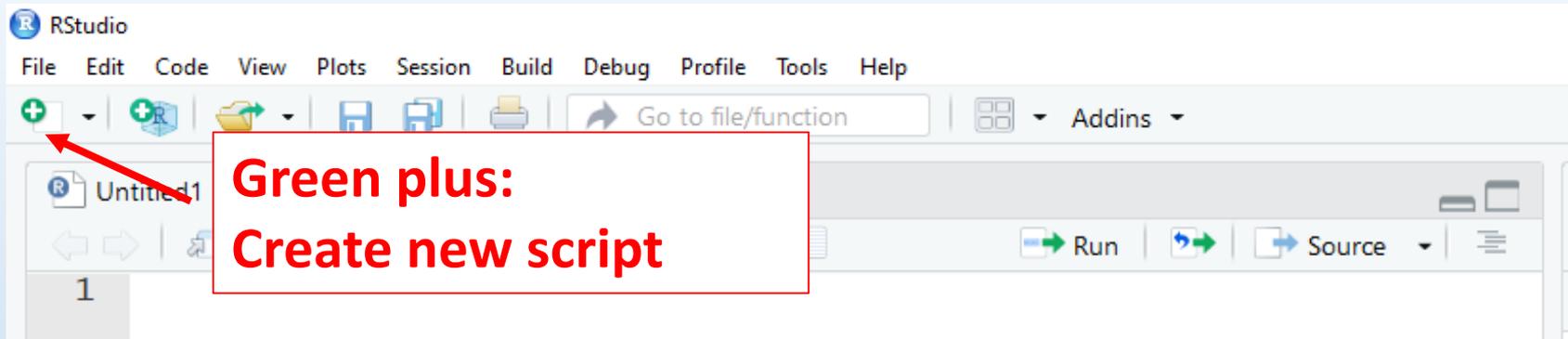
III. Getting Started: Creating an R Project



Instructions

- Go to File / New Project
- Choose New Directory / New Project
- Choose a directory / folder name (e.g. Workshop1) to write in top box
- Choose a location for this directory
- Copy all files for this workshop there

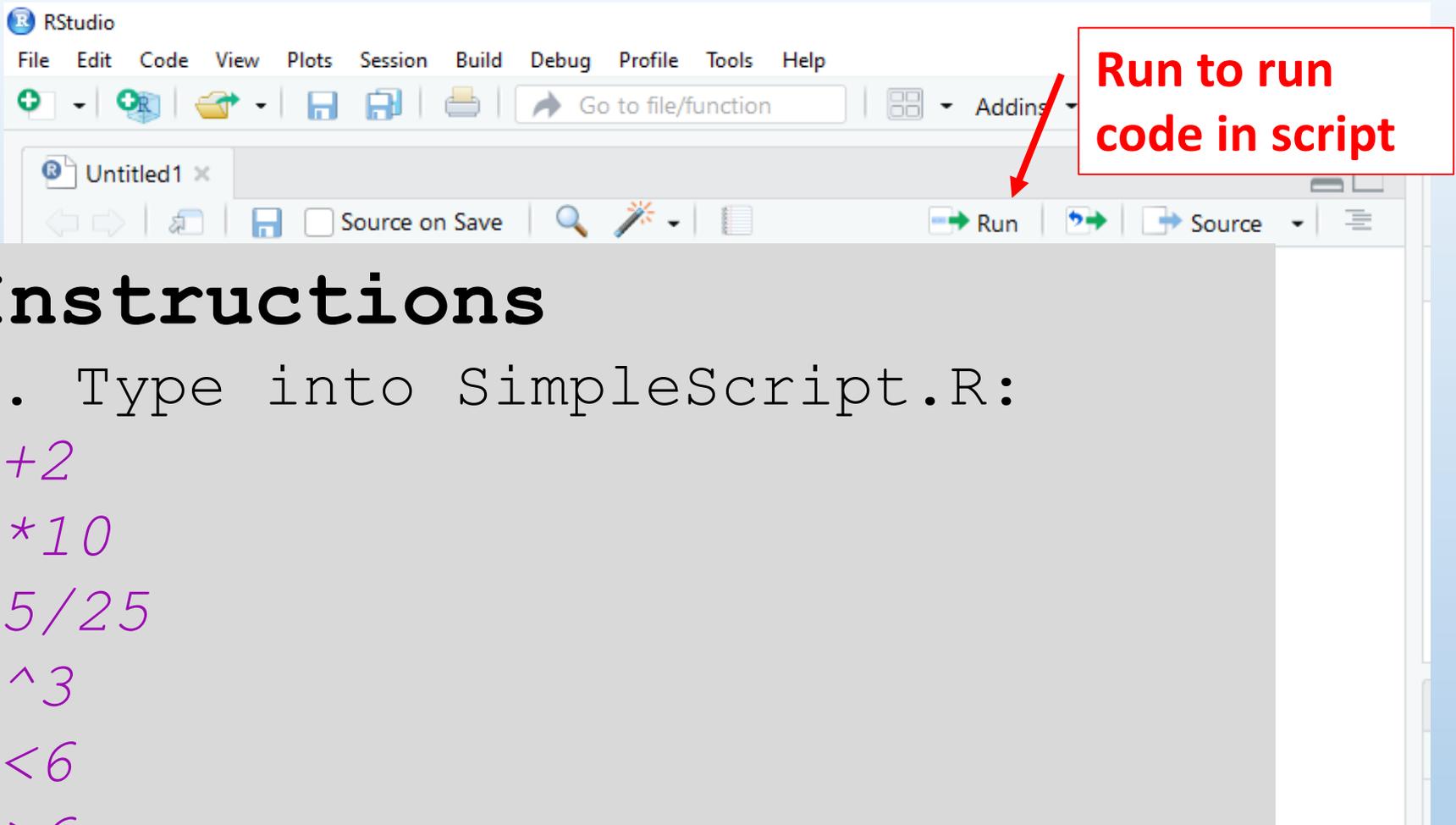
III. Getting Started: Starting a new script



Instructions

- Create a new script
- Save it as SimpleScript.R
- Notice that SimpleScript.R has shown up in the R project directory you created..
- NOTE - You can close your R project (under File / Close Project) when done, and reopen by clicking on the R project file (ends in .Rproj)

III. Getting Started: writing/running a script



Instructions

1. Type into SimpleScript.R:

$2+2$

$5*10$

$75/25$

4^3

$5<6$

$5>6$

2. Save

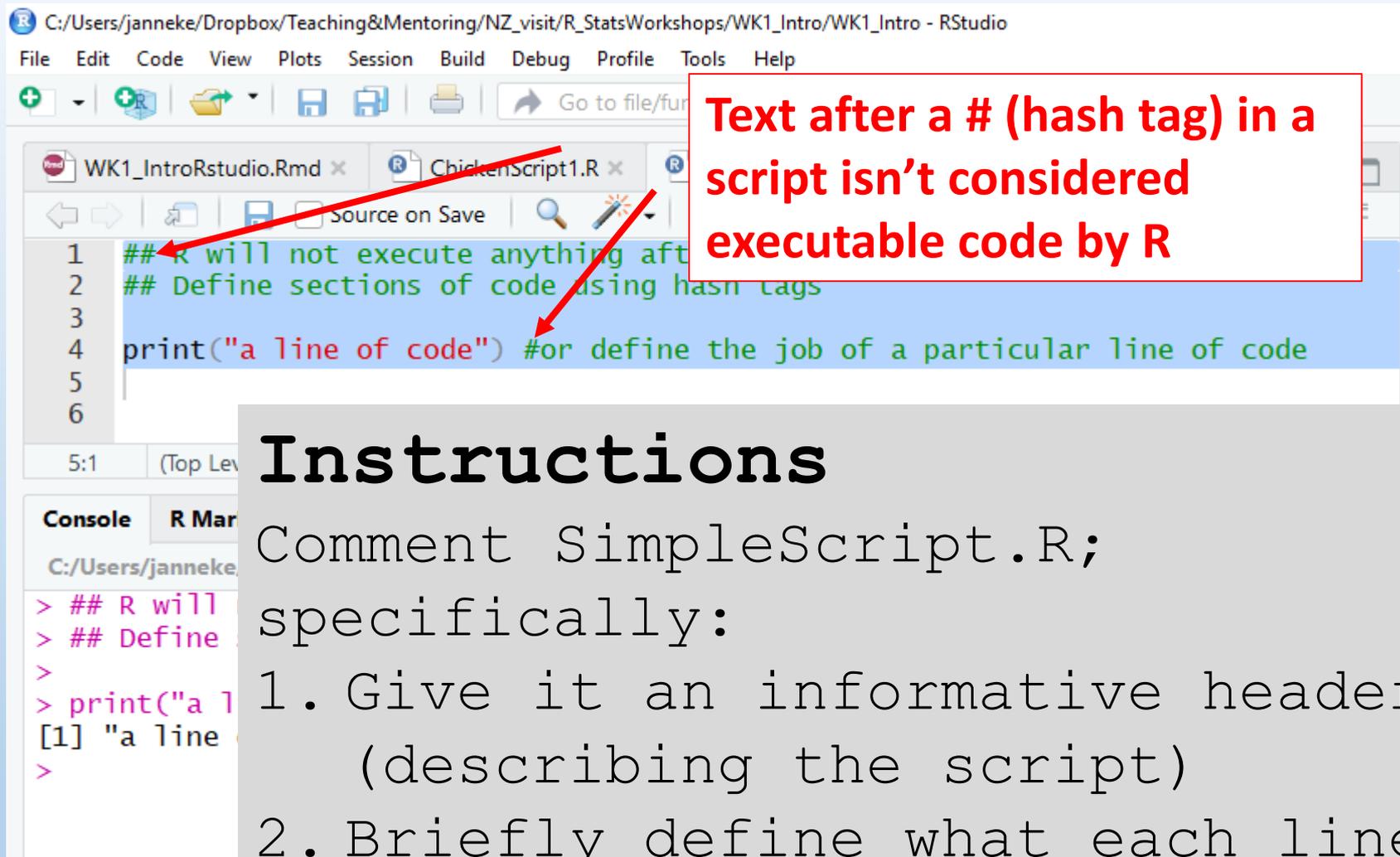
3. Highlight the text and hit run

III. Getting Started: what is 'clean' coding?

Instructions

1. Open `ChickenScript1.R`. It contains 57 lines of code.
2. Does it look easy to understand?
3. Imagine that you created this code, and reopened it after a field season (3 months later). Do you think you would understand it right away? How could you have written it to make it easier to use? **Discuss with your neighbor.**

III. Getting Started: comments



The screenshot shows the RStudio interface. The main editor window displays the following R code:

```
1 ## R will not execute anything after this line
2 ## Define sections of code using hash tags
3
4 print("a line of code") #or define the job of a particular line of code
5
6
```

A red box highlights the text: "Text after a # (hash tag) in a script isn't considered executable code by R". Red arrows point from this box to the hash tags in the code above.

The console window shows the output of the code:

```
> ## R will
> ## Define
>
> print("a line of code")
[1] "a line of code"
>
```

Instructions

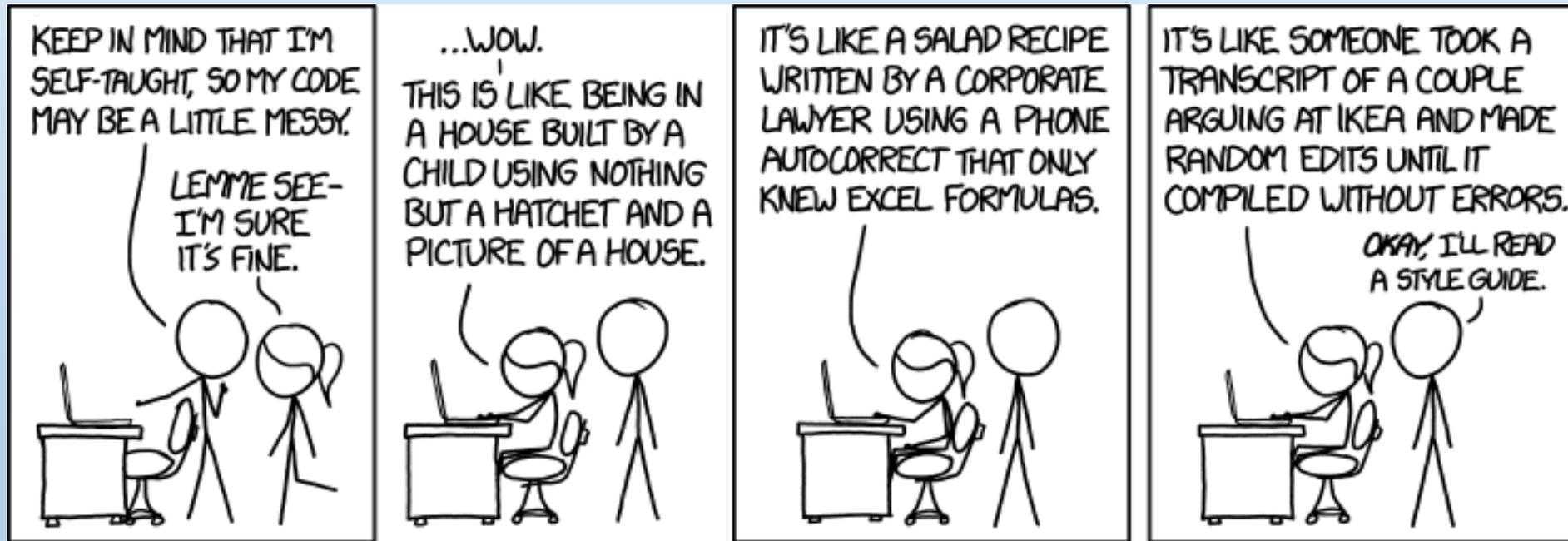
Comment SimpleScript.R;
specifically:

1. Give it an informative header (describing the script)
2. Briefly define what each line of code is doing.
3. Run your code.

III. Getting Started: Code formatting

SUGGESTIONS: Develop your own style guide; i.e.

- Create a script template, and use it for each new script (e.g. see `HRL_ScriptTemplate.R` for an example).



III. Getting Started: Code formatting

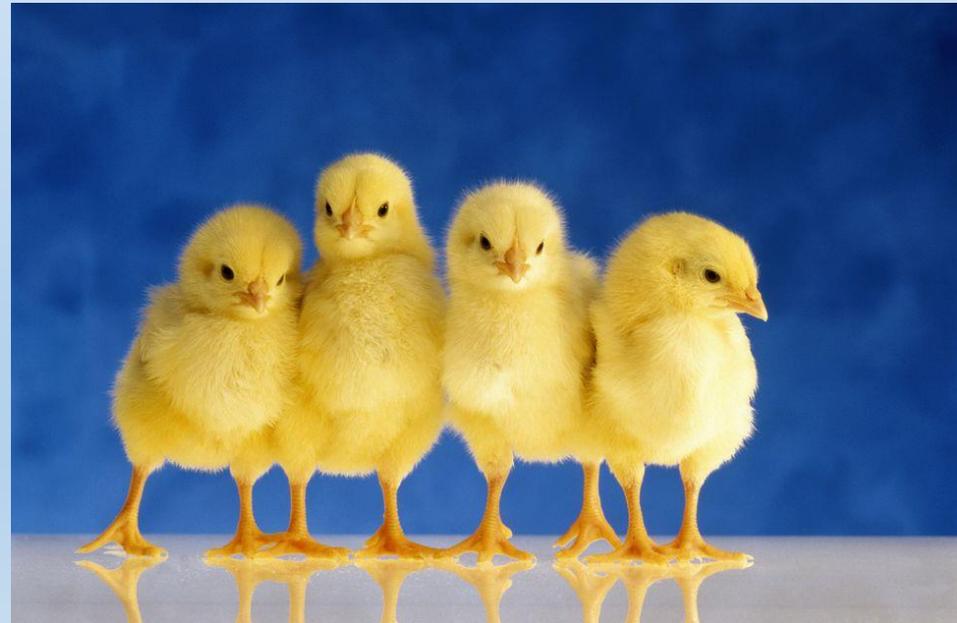
SUGGESTIONS: Develop your own style guide; i.e.

- Create a script template, and use it for each new script (e.g. see `HRL_ScriptTemplate.R` for an example).
- Comment liberally and informatively (this is useful for sharing, especially with a *future* you...). Aim to describe what and why, more than how
- Modularize your code (i.e. create chunks); use indents and paragraph breaks to keep your code visually appealing
- Use consistent and informative names (for data, functions, results, etc).

III. Getting Started: Data Wrangling I

What is data wrangling?

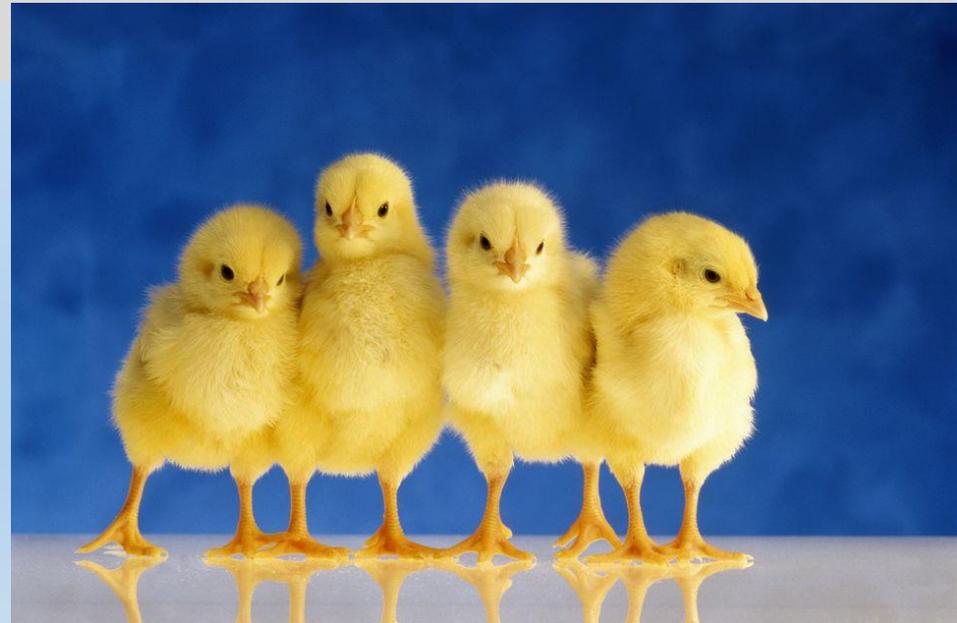
- Reading in data, examining the data file (ensure no errors), manipulating data to create summaries, different explanatory variables, exploratory plots.
- You will learn this by examining existing code, and trying to recreate similar code for another dataset...
- Chicks; 50 chicks weighed daily for 21 days
- Fed: Soybean, Sunflower, Linseed and Meatmeal (ugh)



III. Getting Started: Data Wrangling I

Instructions

1. Open `ChickenScript2.R`, and run the code line by line. Try to understand what the code is doing at each step.
2. Work with a partner
3. Raise your hand if you run into any problems.



III. Getting Started: Data Wrangling II

Nutrient Network Experiment: global, distributed, expt

- Nutrient addition of N, P, K (all combos; i.e. control, N, P, K, NP, NK, KP, NPK)
- Exclusion of large herbivores (only control, NPK)
- ≥ 3 reps per treatment / site (30 plots)
- Biomass, by functional group, collected yearly
- You'll be working with data from Smith Prairie, site in Washington State (2 nutnet files).



III. Getting Started: Data Wrangling II

Instructions

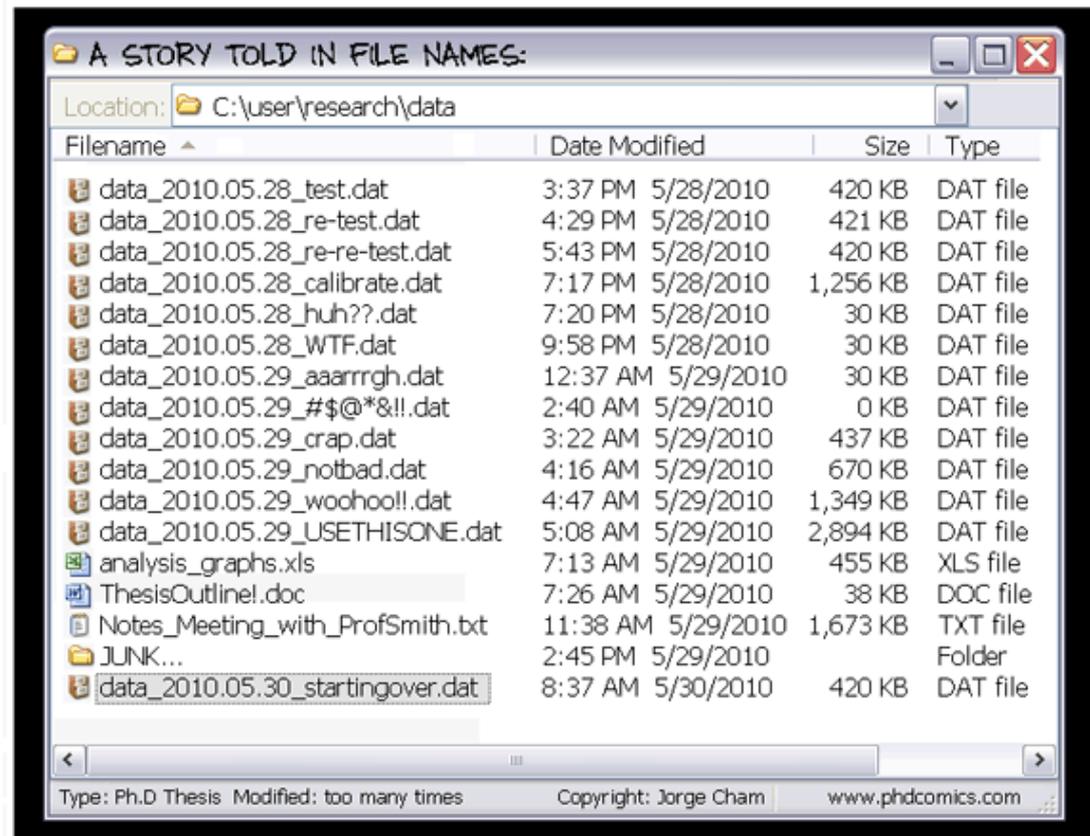
Write a script that does the following:

1. Reads in both data files & merges
2. Explore properties of the dataset
 - A. # years, functional groups sampled.
 - B. Exploratory plots: biomass vs. time; biomass vs. functional group; biomass vs. treatments
3. Subset data (e.g. final year, legumes only), and re-explore (as in #2)
4. A challenge: calculate the proportion of biomass that was legumes, explore by treatment (use `tapply`, `merge`, `subset`, create a new data frame)

IV. Further Topics

Data management

- Have a plan for data collection -> data entry -> error checking -> 'clean' data



IV. Further Topics

Data management

- Have a plan for data collection -> data entry -> error checking -> 'clean' data
- Include 'metadata'; in same file (e.g. worksheet in excel). Should include description of columns.
- Data structure: each row should be its' own observation, each column its' own variable.
- Anticipate the future (e.g. include year, 1st year)
- Order columns (explanatory, response), sort data
- No-no's: mixing data types (numbers + letters; NA's an exception); gaps / spaces, long names with spaces for column headers.
- Special consideration: consider NA's and true zeros

IV. Further Topics

Code sharing / reproducibility: Git and Github

- Git is a free online program that provides version control. Github is the webhosting version of Git.
- Keeps track of all versions of code, allows you to associate comments with changes, go back in time (to a previous version), and (if coding collaboratively) view changes by coder.
- More and more people are sharing code (in publications) by posting a link to a git repository.

	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

IV. Further Topics

Resources

- R and R studio online learning (hit home button in help)
- In-depth tutorial (web browser): Pirate-themed [TryR](#) via Code School
- Rstudio one page [cheatsheets](#)
- [Software Carpentry](#) has great workshops (free or virtually free), also online tutorials
- Books: I like Mick Crawley's [The R Book](#). Native NZ son Hadley Wickham's book [R for Data Science](#) is also meant to be good – he also has a set of packages (check the [Tidyverse](#)) that are excellent for munging, merging, data.

Acknowledgments



Clay Wright

UW Biology, R course (SAFS 552, 553) & Other



Trevor Branch

UW SAFS – R course (SAFS 552, 553)

Workshop 1 (15/03/2018)