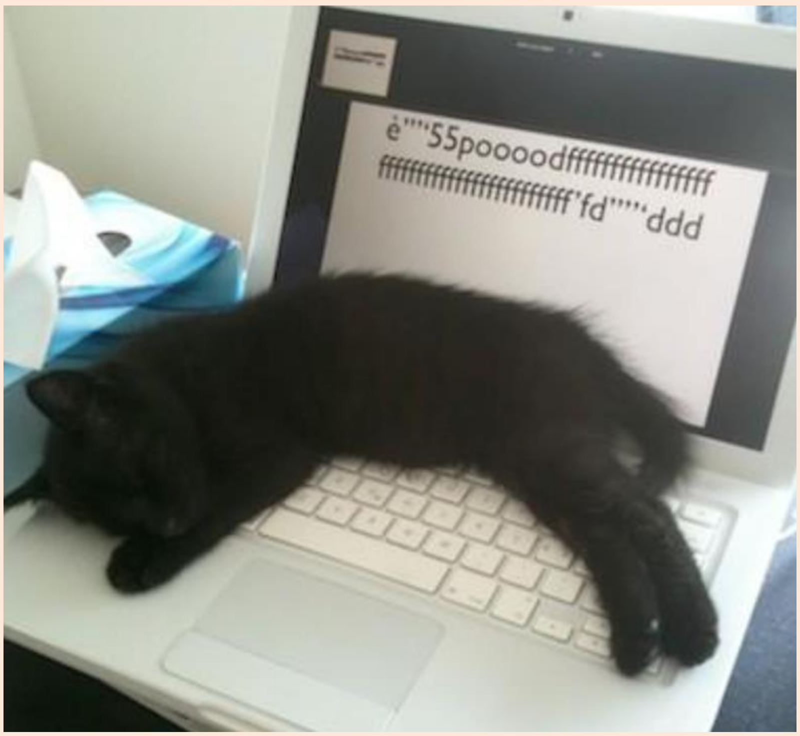
BIOSTATS 640 – Introduction to R
Fall 2024
https://people.umass.edu/biep640w/webpages/demonstrations.html



Source: https://www.engadget.com/2013-04-06-caturday-ruby-the-programmer-cat.html

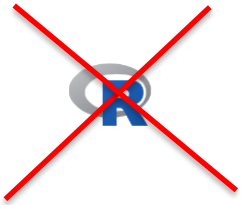
01
Up and Running with R Studio
September 4, 2024

Before you Begin -
This R lesson assumes that you have already downloaded and installed R and RStudio or, preferably for BIOSTATS 640, created an account in Posit in the Cloud.

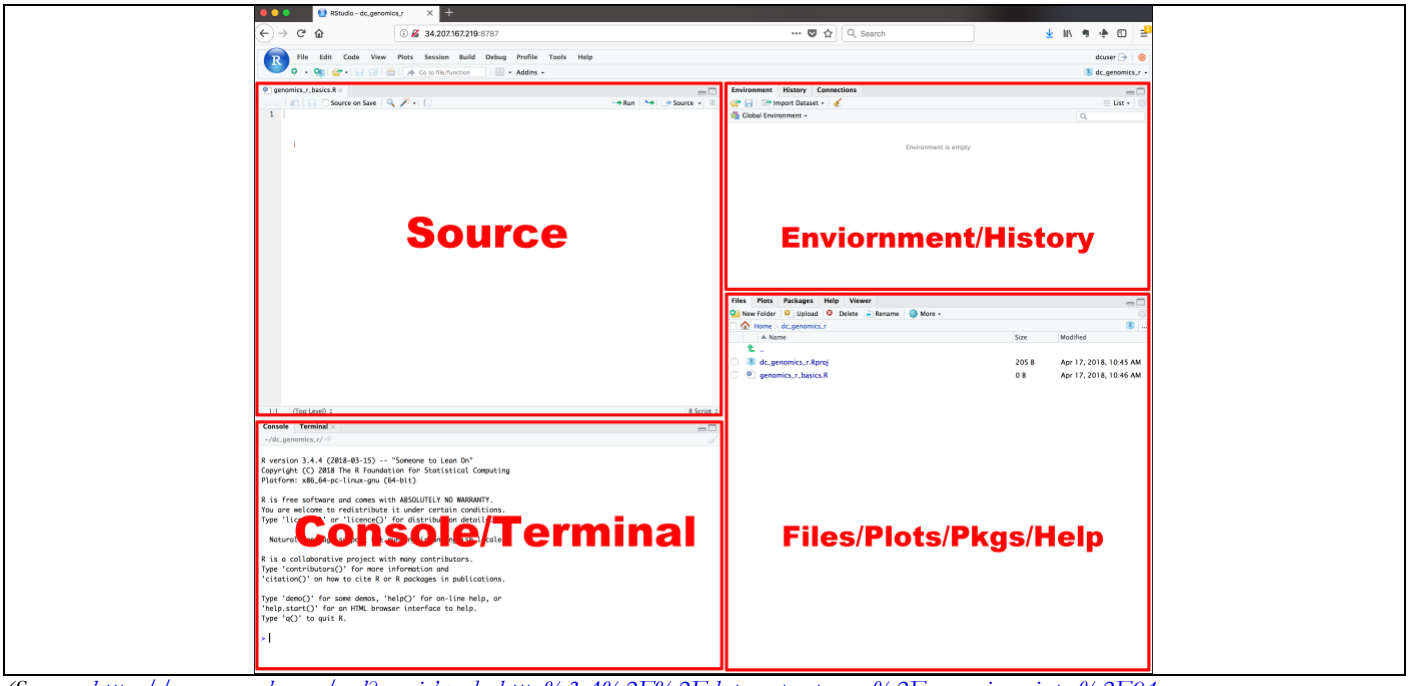
Welcome!
In this introduction, you will learn how to navigate among the panes in R Studio, issue some commands, and fix your mistakes!

1. Launch R Studio and Acquaint Yourself with its Interface

We will be doing all our work through the interface called R Studio; R is "under the hood".
We will be doing all our work in R Studio, ***NOT R***

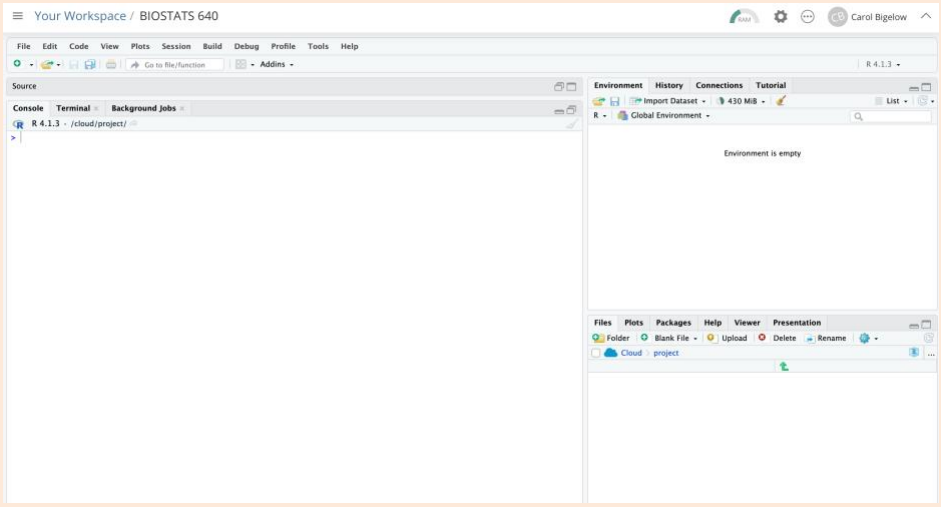
Launch R Studio	NOT R
	

Acquaint Yourself with the R Studio Interface.
Consider visiting this introduction, here:
https://ismayc.github.io/rbasics-book/3-rstudiobasics.html



(Source: https://www.google.com/url?sa=i&url=https%3A%2F%2Fdatacarpentry.org%2Fgenomics-r-intro%2F01-introduction%2Findex.html&psig=AOvVaw1hPui5N1bGL4CtSNUvA5Qi&ust=1600449355622000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCPiHvdLY8OsCFQAAAAAdAAAAABAs)

Example – My R Studio Interface in Posit/Cloud



Quick Overview of the Panes in R Studio

Pane	Description
Console/Terminal	• Default location: **lower left** • Code is executed from here • The prompt is a **">"** • IMPORTANT: Code typed into console is **NOT SAVED** • HACK: To retrieve previous command: **UP-arrow** • HACK: To clear window: **<control-l>** *this is the letter "el"* Good to know: No worries, your history is not lost
Source	• Default location: **upper left** • Here is where you will do your **R Script** and **R Markdown** work
Environment/History	• Default location: **upper right** • There are multiple tabs • **Environment tab**: Here you see all your stuff, called "objects" – datasets, variables, etc • **History tab:** Here you will see all your previous commands • HACK: To view your data: CLICK on its name
Files/Plots/Packages/Help	• Default location: **lower right** • There are multiple tabs.

	• Here you will find: plots, help w packages, importing from your computer, help

How to Move Between Panes (Note – To be honest, I hardly ever make use of these hacks. But you might!)

Shortcut	Moves you to:
< control > 1	Source/Editor (your script file)
< control > 2	Console
< control > 3	Help
< control > 4	History
< control > 5	Files
< control > 6	Plots
< control > 7	Packages
< control > 8	Environment
< control > 9	Viewer
< control > SHIFT 0	Returns you to original 4 panel display

Preliminary: > versus <- versus #

> **Prompt**. This is the prompt. R is waiting for you to issue a command

<- **Assignment operator**. This is the assignment operator (so is the equal sign)

**Beginning of a comment.** R ignores the rest of the line (good for documenting your code!)

More on the assignment operator, <-

Note: You could also use the equal sign, =, but this is not recommended

< - Translation: "Assign from the right to the left"

a < - 4 **Example: -** Assign the number "4" to an object that I choose to name "a"

More on the hashtag/pound symbol #

Comments in R begin with a #. R will ignore the rest of the line and continue its work at the start of the next line.

Tip – Make it a habit to insert comments in your work. A lot! A year from now, when you look at your old work, you'll be so glad.

2. Use R (console) as a Giant Calculator

```
#  HOW TO create a vector object that is NOT SAVED
#  Use c() to create a variable (R calls this a vector object) - unsaved
c(1,2, 4, 8, 12, 13, 15)
## [1]  1  2  4  8 12 13 15

# HOW TO create a vector object that you DO SAVE and name as v1.
v1 <- c(1,2, 4, 8, 12, 13, 15)
# Bummer.  R Studio doesn't give you the result.
# You have to tell R Studio to show you something.

# HOW TO tell R to show you something
# To view the contents of an object, simply type the name of the object
v1
## [1]  1  2  4  8 12 13 15

# HOW TO obtain the data type (character, numeric, etc)
# Use class() to show the data type
class(v1)
## [1] "numeric"

# addition – Show the result but do not save it
4+6
## [1] 10

# Subtraction – Show the result but do not save it
4-6
## [1] -2

# Basic math in two steps:  (1) create the object y that is the solution (2) display the object y
y <- 4+6
y
## [1] 10

# Basic math in 2 steps connected by a semi-colon:  (1) create ;  (2) display
x<-5+8; x
## [1] 13

# Basic math in one step but now using parentheses to force R Studio to display
(x<-5+8)
## [1] 13
```

Mathematical Functions in R (partial listing)

Function	Definition	Example
+	Addition	> 2+2 [1] 4
-	Subtraction	> 5-3 [1] 2
*****	Multiplication	> 5*4 [1] 20
/	Division	> 20/4 [1] 5
^	Exponentiation (raising to a power)	> 6^2 [1] 36
%/%	Integer part of division or quotient	> 48 %/% 5 What is whole number of 48/5? [1] 9
%%	Remainder part of division or quotient	> 48 %% 5 What is the remainder of 48/5? [1] 3
log()	logarithm to base e ("natural log") You may know this as ln( )	> log(34) [1] 3.526361 $e^{3.526361} = 34$
log10()	Logarithm to base 10	> log10(100) [1] 2 $10^2 = 100$
exp()	Exponentiation of the constant e Recall: e = 2.718 … (approx.)	> exp(4) [1] 54.59815 $e^4 = 54.59815$
sqrt()	Square root of	> sqrt(100) [1] 10 $\sqrt{100} = 10$
round(x,n)	Round x to the nth digit	

3. What Could Go Wrong

Unlike me who always forgets, don't you forget the 20 minute rule!!!!

__1. **Error: My assignment operator did not work**
Solution (for this example): The assignment operator is <- with no space between the < and the –

```
> v1 < - 4+6
Error: object 'v1' not found
> v1 <- 4+6
> v1
[1] 10
>
```

_2. **Error: I am not getting any result**
Solution (for this example): Creating something is just that. Only. You have to tell R to then show it.

```
>
> v2 <- 5^2
>
>
> v2
[1] 25
>
```

_3. Error: I made a mistake several commands back; how do I fix that?
Solution (for this example): In the console window, do UP-ARROW repeatedly to access and then edit your command.

_4. Error: I am getting a + and no result

Solution (for this example): The + means that you have submitted a command that is incomplete. R is waiting for youto finish it. You have two options for a solution: (1) finish the command; or (2) abandon the command

SOLUTION #1 – Finish the command.

At the + simply finish the command and enter

```
> (4+6)*(5
+
```

```
> (4+6)*(5
+ *7)
[1] 350
```

SOLUTION #2: Abandon the command.

Simply **Click <escape>** to return to the prompt

```
> v2 <- (4+6) * (5*
+
```

```
> v2 <- (4+6) * (5*
+
>
```

#4. Create Your First R Data

Let's begin by creating a very simple example.
Create an R dataset by combining columns (vectors)
Functions Used: c( ) and data.frame( )

In this example, we create a column of data for one numeric variable called **weight**. In R this is a vector of data type = numeric. We then create a 2nd column of data for one character variable called **town**. In R, this is a vector of data type = character. Finally, we combine these two variables (vectors in R) into a R dataset

```
> # create numeric vector called weight using c()
> weight <- c(161.3, 120.1, 223.2, 124.0, 88.2, 136.7, 140.0, 151.6)

> # create a character vector called town using c() with entries in quotes
> town <- c("amherst","amherst","hadley","amherst", "amherst","hadley","amherst", "amherst")

> # create an R dataset using data.frame(vector1, vector2) to combine vectors
> mydata <- data.frame(town,weight)

> # Show mydata by simply typing its name
> mydata
```

```
  town weight
1 amherst  161.3
2 amherst  120.1
3  hadley  223.2
4 amherst  124.0
5 amherst   88.2
6  hadley  136.7
7 amherst  140.0
8 amherst  151.6
```

#5. A First Look at Your Data

In this first "go" at R, we consider numerical summaries only. In a future introduction (after learning about the package ggplot2), we will learn ways to produce data visualizations. So, stay tuned! Pretty graphs are on their way.

In future introductions to R, we will be using packages to do all kinds of looks at data.
But here, we learn some simple ways to look at data using the functions that are included in your installation.
There's lots you can do, actually!

Welcome to the structure dataframe$variable
Yes, it seems a bit clunky but there you have it. In R, the convention for denoting a single variable in a dataframe is the two part notation: `dataframe$variable.`

dataframe$variable

dataframe

- **"dataframe"** is analogous to "dataset" in SAS or Stata or Minitab, etc
- **"dataframe"** is analogous to "sheet" in an Excel file that has data in a sheet

variable

- **"variable"** is just what you think it is. It is analogous to "variable" in SAS or Stata or Minitab, etc
- **"variable"** is analogous to "column" in an Excel file that has data in a sheet

Example

mydata$town

dataframe = **mydata** variable = **town**

Good to know:

R needs you to specify both dataframe and variable because :

You can have more than one dataframe (think dataset) active at a time.

Missing Values in R

NA

Note: There are other ways to denote missing values in R; we will get to these in a future introduction.

Preliminary: Use the function str( ) to learn about the structure of your data and to determine the data type for each variable. Why is this useful? Among other reasons, R will do some descriptives for some datatypes but not others. For example, R will not produce descriptives for a variable that is of data type = character.

```
> str(mydata)                                    # examine structure of dataframe/dataset
```

```
'data.frame':     8 obs. of  2 variables:
 $ town  : chr  "amherst" "amherst" "hadley" "amherst" ...
 $ weight: num  161.3 120.1 223.2 124 88.2 ...
```

Key:
- This is a dataframe
- There are 8 observations (sample size n=8)
- There are 2 variables: town and weight
- town is of data type = character
- weight is of data type = numeric

As needed:
Use factor( ) to create a categorical variable from your character variable.

Important:
R doesn't use the word "categorical" to refer to what we think of as "categorical". R calls these factors
We will learn a lot more about factors in future R introductions!

```
> mydata$townf <- factor(mydata$town)            # create a new variable called townf
> str(mydata)                                    # check that the new variable is there and correct
```

```
'data.frame':     8 obs. of  3 variables:
 $ town  : chr  "amherst" "amherst" "hadley" "amherst" ...
 $ weight: num  161.3 120.1 223.2 124 88.2 ...
 $ townf : Factor w/ 2 levels "amherst","hadley": 1 1 2 1 1 2 1 1
```

Key:
- This R object is a dataframe
- There are 8 observations (sample size n=8) of 3 variables.
- town is character, weight is numeric, and townf is factor type

Summary statistics for a continuous variable using the function summary( )

```
> summary(mydata$weight)
```

```
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   88.2   123.0   138.3   143.1   154.0   223.2
```

Good to know.

R does *not* produce descriptive statistics for a character variable

Why: A character variable has responses that are simply strings; it's not possible to produce descriptive statistics of strings!

```
> summary(mydata$town)
```

```
   Length     Class      Mode
        8 character character
```

Key:

- Because the variable town is character type, NO descriptive statistics are possible

Frequencies for a categorical variable (remember: R calls this a factor) using summary( )

```
> summary(mydata$townf)
```

```
  amherst  hadley
        6       2
```

Key:

- In contrast, because townf is factor type, R gives us the frequency of each outcome

Numerical Summary for EVERY variable in your dataframe/dataset using summary( )

```
> summary(mydata)
```

```
   town               weight          townf
Length:8           Min.   : 88.2   amherst:6
Class :character   1st Qu.:123.0   hadley :2
Mode  :character   Median :138.3
                   Mean   :143.1
                   3rd Qu.:154.0
                   Max.   :223.2
```

Key – Putting it all togther:
- NO descriptive statistics are produced for the character variable town
- As a general rule, do NOT overwrite an existing variable. Always preserve your original data. Here, I created townf as a new variable, leaving the source variable town UNchanged

Illustration: How to Calculate a Statistical Summary using functions such as mean( ) , var( ), etc

```
> mean(mydata$weight)
```

```
[1] 143.1375
```

Key:
- The mean of weight is 143.1375
- R provides a result ONLY if there are NO missing values

What Could Go Wrong: How to Calculate a Statistical Summary (e.g., mean, variance, etc)
Error: My calculation of a statistic produced NA
Solution (for this example): You need to tell R to exclude missing values NA in the calculation

```
> age <- c(33,12, NA, 67, 82, 91)
> mean(age)                              # calculate mean
[1] NA
```

```
> mean(age,na.rm=TRUE)                   # calculate mean with option na.rm=TRUE to remove missing valu
es
[1] 57
```

Key:
- Options are provided and are separated by a comma
- The option na.rm=TRUE stands for "remove NA" which tells R to remove missing values.

Some Statistical Functions in R

Function	Definition	Example

length(x)	Number of values in vector x	`x <- c(3,1,6,0,6)`
		`> length(x)`
		`[1] 5`
		
		`… alternatively, you could do ..`
		`> length(c(3,1,6,0,6))`
		`[1] 5`

max(x)	Maximum of values in vector x	`> x <- c(3,1,6,0,6)`
		`> max(x)`
		`[1] 6`

min(x)	Minimum of values in vector x	`> x <- c(3,1,6,0,6)`
		`> min(x)`
		`[1] 0`

mean(x)	Mean of values in vector x	`> x <- c(3,1,6,0,6)`
		`> mean(x)`
		`[1] 3.2`
		`> x <- c(3,1,NA,0,6)  Oops a missing!`
		`> mean(x,na.rm=TRUE)`
		`[1] 2.5`

median(x)	Median of values in vector x	`> x <- c(3,1,6,0,6)`
		`> median(x)`
		`[1] 3`

quantile(x,c(.25,.75))	Obtain 25th and 75th quantile values in vector x	`> x <- c(3,1,6,0,6)`
		`> quantile(x,c(0.25,0.75))`
		`25% 75%`
		`  1   6`

range(x)	Display minimum and maximum values in vector x	`> x <- c(3,1,6,0,6)`
		`> range(x)`
		`[1] 0 6`

sd(x)	Standard deviation of values in vector x	`> x <- c(3,1,6,0,6)`
		`> sd(x)`
		`[1] 2.774887`

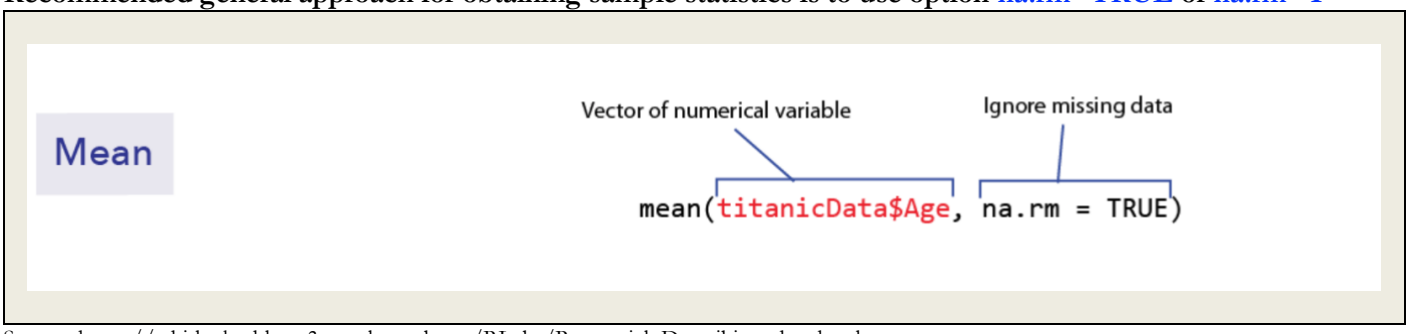
sum(x)	Total of values in vector x	`> x <- c(3,1,6,0,6)`
		`> sum(x)`
		`[1] 16`

var(x)	Variance of values in vector x	`> x <- c(3,1,6,0,6)`
		`> var(x)`
		`[1] 7.7`

abs(x)	Absolute values of values in vector x	`> abs(2-10)` `[1] 8`
factorial(x)	Calculate x! = x(x-1)(x-2)…(2)(1)	`> factorial(4)` `[1] 24  4! = 4*3*2*1 = 24`

rank(x)	Ranks of values in vector x	`> x <- c(3,1,6,0,6)`
		`> rank(x)`
		`[1] 3.0 2.0 4.5 1.0 4.5`

Recommended general approach for obtaining sample statistics is to use option na.rm=TRUE or na.rm=T



Source: https://whitlockschluter3e.zoology.ubc.ca/RLabs/R_tutorial_Describing_data.html

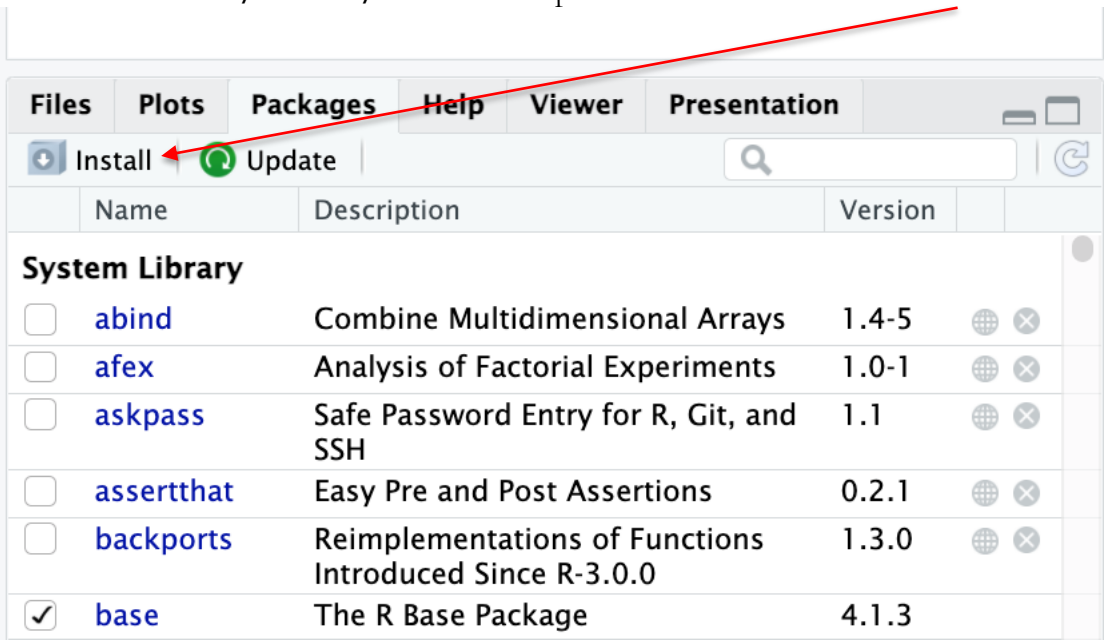
#6. *Your Turn*: Learn R Interactively Using the Package `{swirl}`

6.1 Install package (one time) `{swirl}`

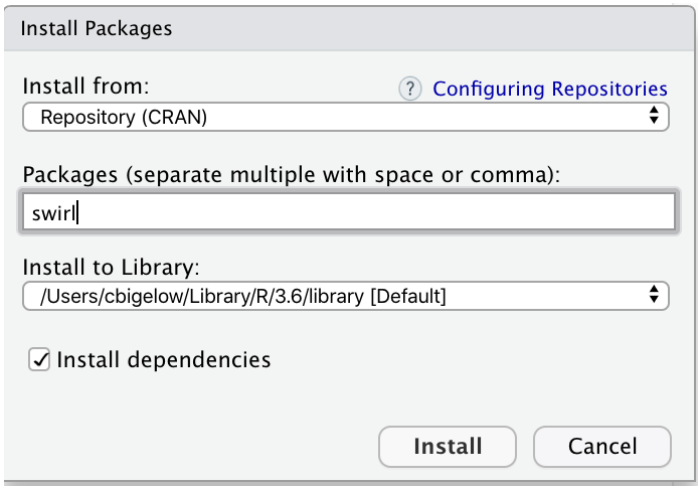
Method 1 – Using menus
Method 2 – Using command `install.packages("packagename")` in console pane

Method 1. Using Menus

From the **FILES/PLOTS/PACKAGES** pane choose **PACKAGES** and click **Install**



Key:



In Install from:
default (Repository CRAN) is fine

In Packages (separate multiple with space or comma:) **swirl**

In Install to Library: **leave as is**

Check box for "Install dependencies":
check

At bottom, click **Install**

Method 2 – Using command `install.packages("packagename")` in console pane

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

How to Load/Attach a Package to Your R Session
`library(packagename)`.
Required. In loading/attaching, the name of the package must NOT be enclosed in quotes

6.2 Load/attach using `library(swirl)`

```
library(swirl)
```

Good to Know!

- Installation is done ONE time
- Loading/attaching is done EVERY session

6.3 Start swirl using `swirl()`

```
swirl()
```

```
>
> library(swirl)
> swirl()

| Welcome to swirl! Please sign in. If you've been here before, use the same name as you did then. If
| you are new, call yourself something unique.

What shall I call you?
```

```
What shall I call you? Carol

| Would you like to continue with one of these lessons?

1: R Programming Basic Building Blocks
2: No. Let me start something new.

Selection:
```

Good to Know. Swirl Commands

`main()`	Return to main menu
`bye()`	Exit swirl. Your progress will be saved
`info()`	Display options again
`skip()`	Skip the current question
`play()` followed by `nxt()`	Experiment with R on your own until you type `nxt()` to return to swirl

#7. Some Good Videos to Get You Started

Pour yourself a tea or a coffee and watch them All!!!!

__1. *(Source: R Programming 101)*
R Programming for ABSOLUTE Beginners (video, 14:12)

__2. *(Source: MarinStats Lectures – R Tutorials 1.1*
What is R Studio and Why You Should Use it (video, 5:20)

__3. *(Source: MarinStats Lectures – R Tutorials 1.1*
Customizing R Studio (video, 3:53)

__4. *(Source: MarinStats Lectures – R Tutorials 1.1*
Setting Up Your Working Directory (video, 8:25)

__5. *(Source: MarinStats Lectures – R Tutorials 1.1*
Getting Started in R: Basic Mathematics and Coding in R (video, 7:47)