Introduction to

Introduction to

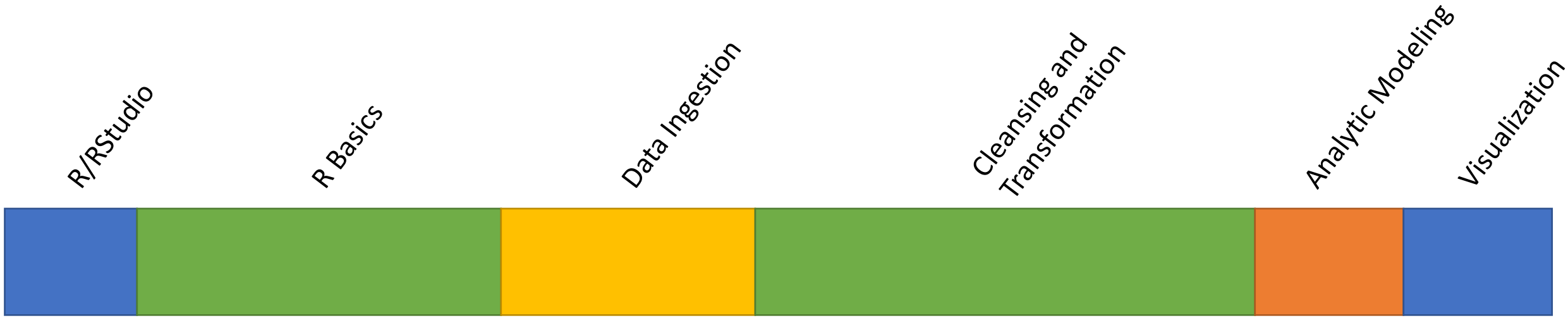
@joshuafennessy

linkedin.com/in/joshuafennessy

josh@joshuafennessy.com

http://www.blue-granite.com/blog

Our Day…


R/RStudio
R Basics
Data Ingestion
Cleansing and Transformation
Analytic Modeling
Visualization

You can expect to learn:

- The syntax of the R language and basic data types
- One way to ingest data from commonly used sources in business
- One way to prepare data for analytics
- Basic modeling techniques

Before we begin:

1. Make sure you have R installed

2. Make sure you have RStudio installed

3. Clone this GitHub repository: https://github.com/joshuafennessy/LearnRInADay

R and RStudio

R is....

- a **functional** programing language specifically geared towards analytics
- Open Source - covered under the GNU General Public License
- Free (as in beer)*
- Cross-platform
- Delivered by **The Comprehensive R Archive Network** (CRAN), who provide base R and additional mirrors for a large collection of add-on packages

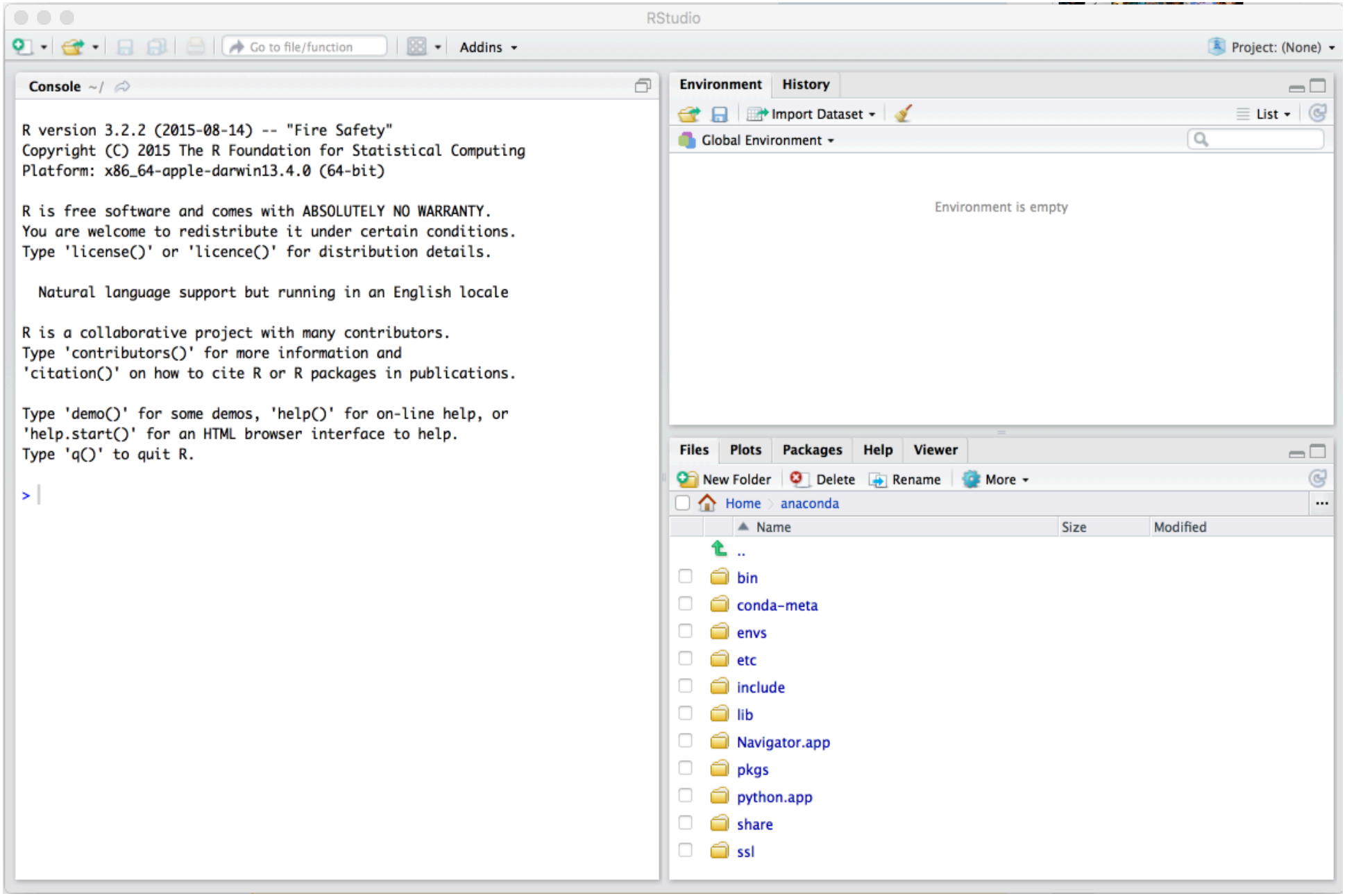
  http://www.cran.r-project.org

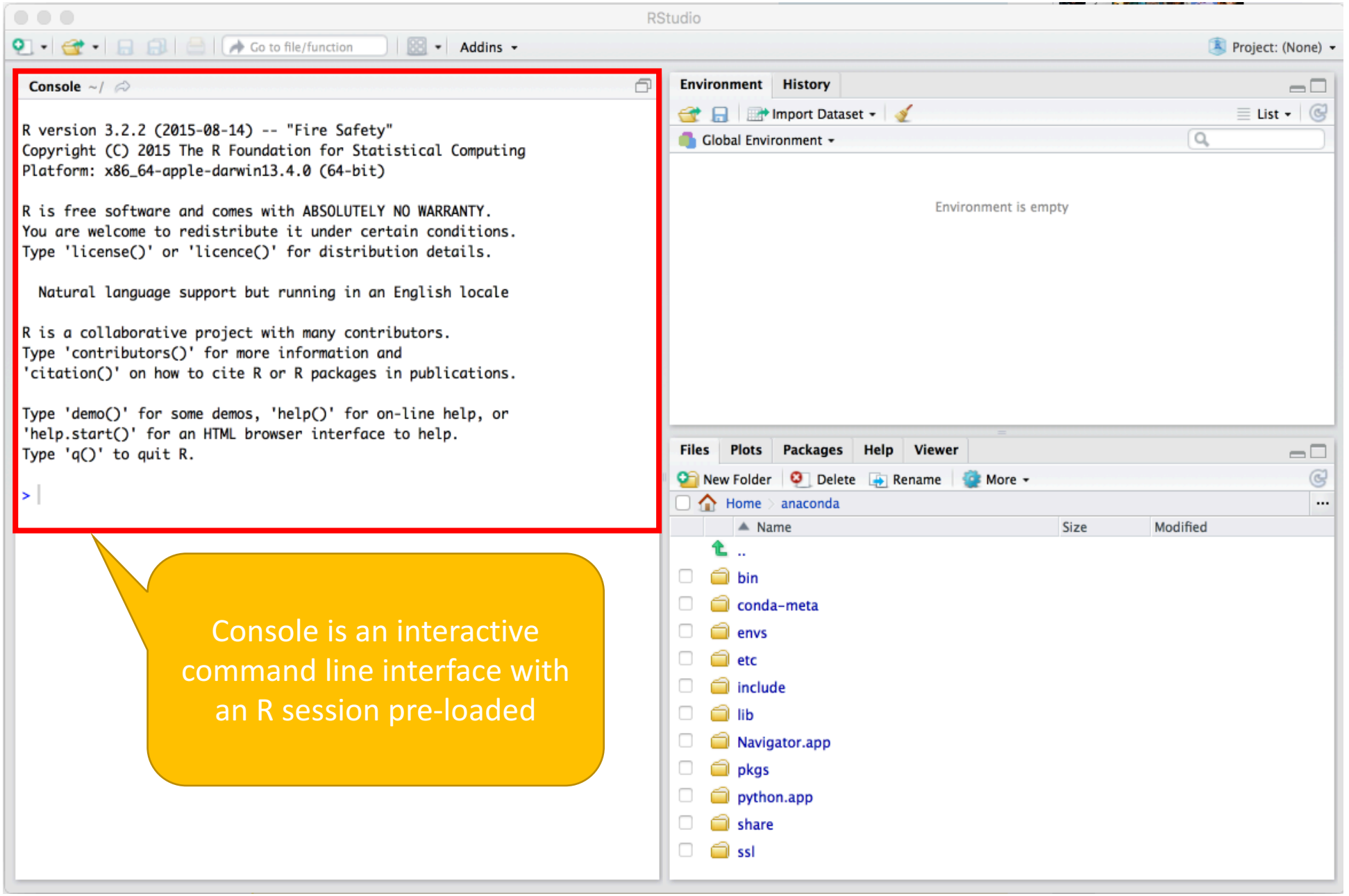
You might choose to get R from Microsoft…

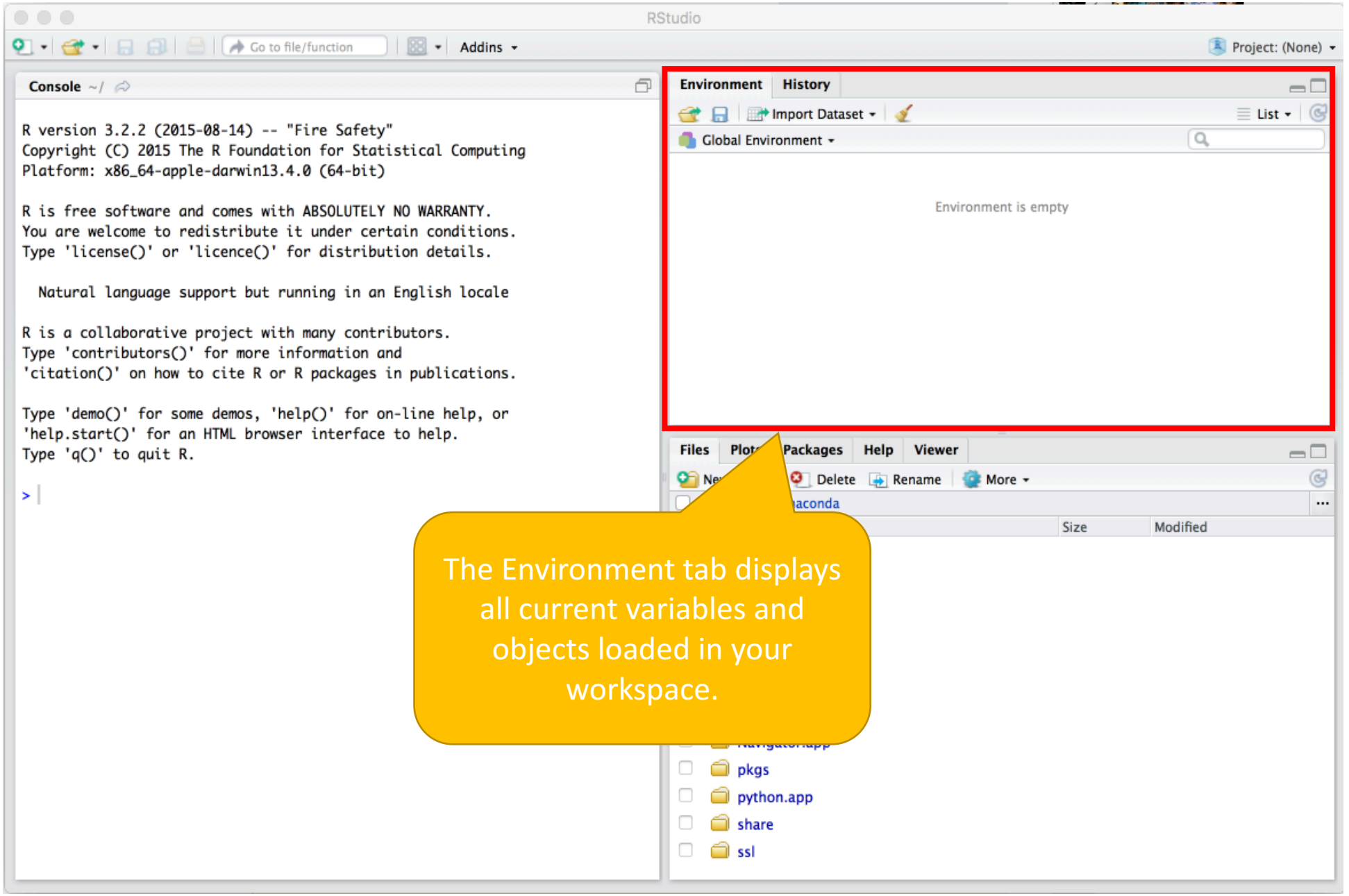
- Fully CRAN compatible
- Performance improvements through the use of multi-core processing
- Static CRAN repository based on the date downloaded
- Included checkpoint package to simplify code sharing and package versioning
- Additional bundled packages that provide additional features to R and aren’t included with base R.

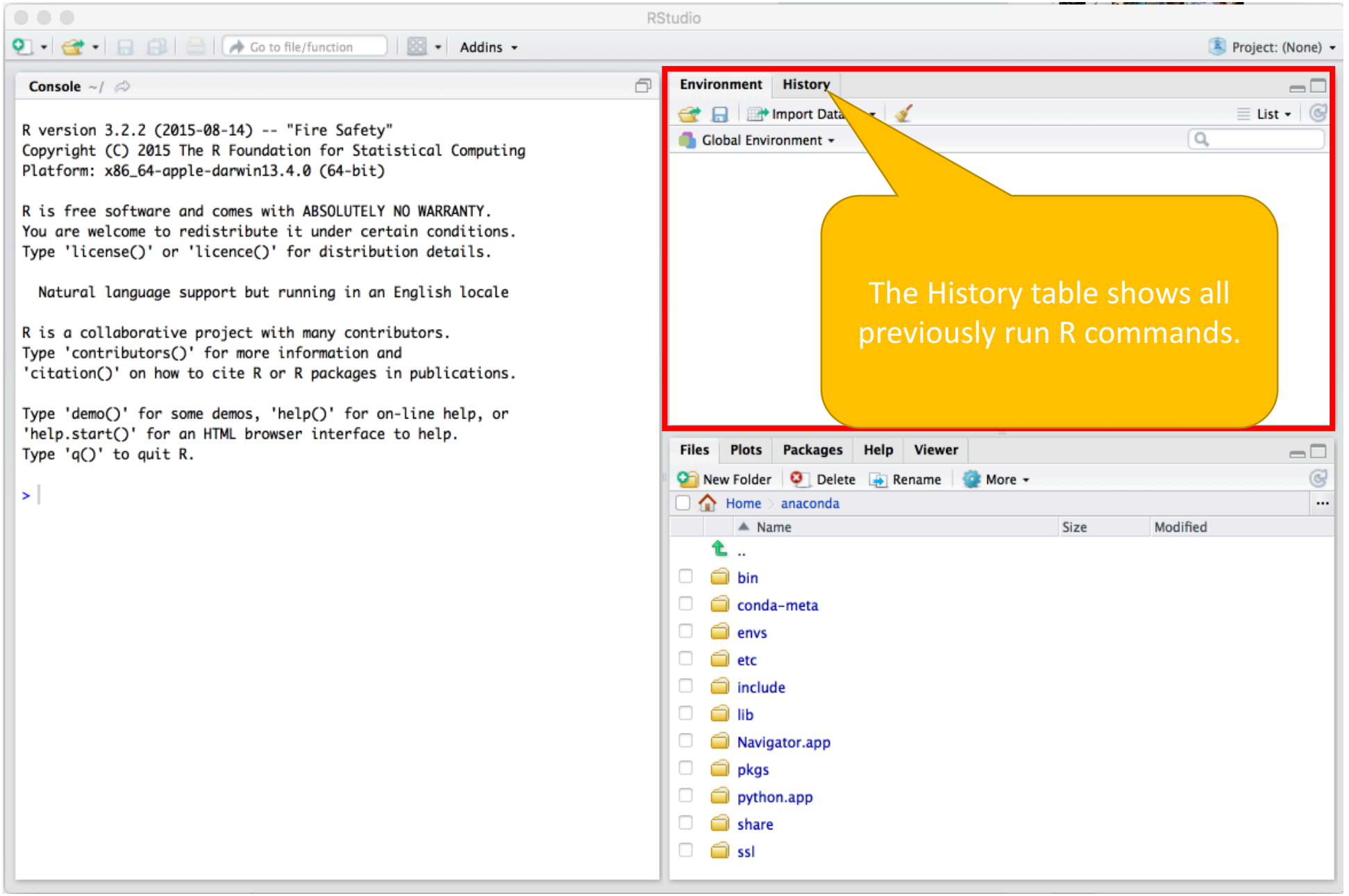
We typically develop R in…

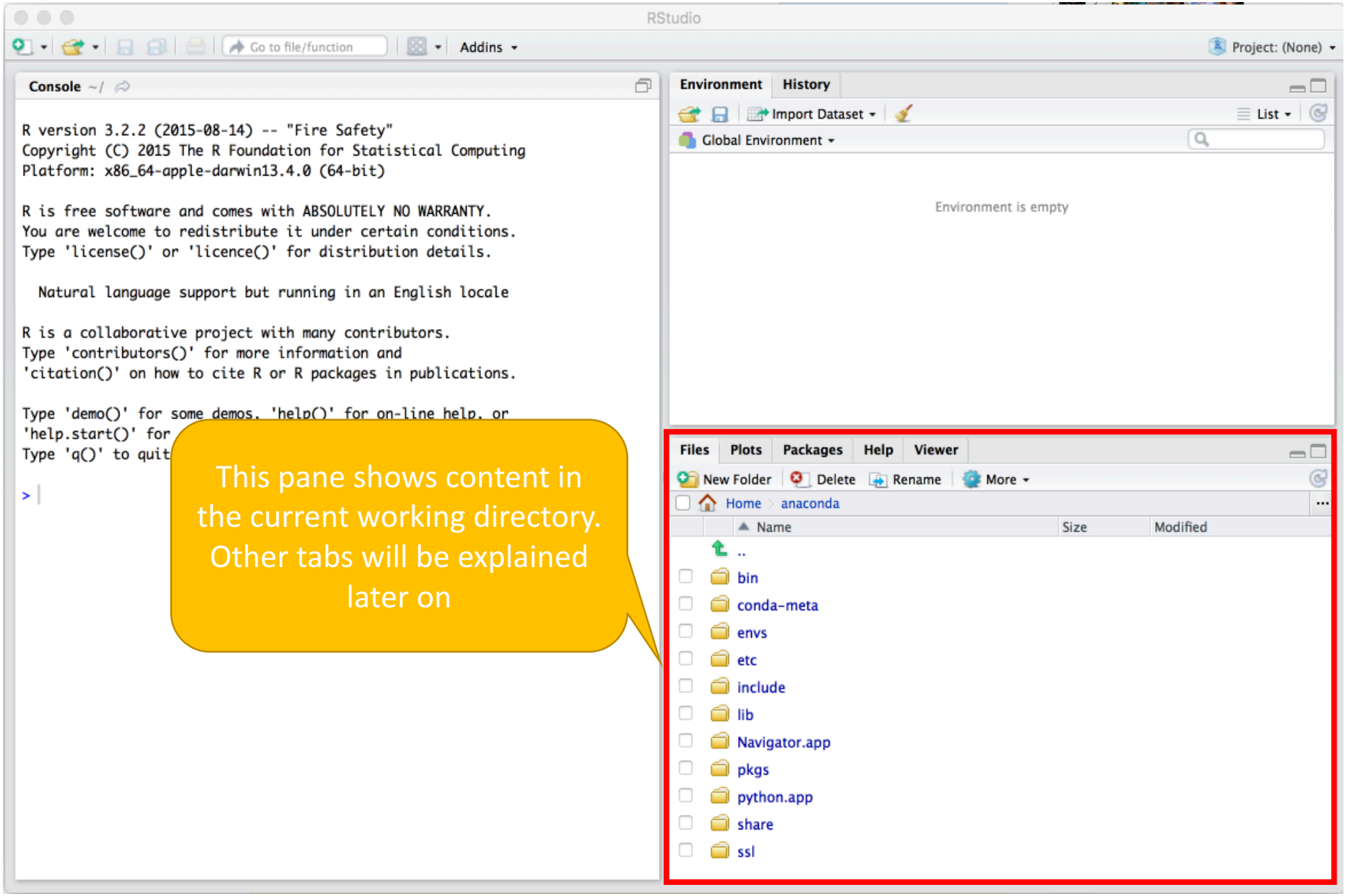
- Rstudio (http://www.rstudio.com)
- Installed as a client application (Windows/Mac/*nix) with Free (as in beer) and Enterprise versions available
- Also installed as a server, allowing you to access RStudio in a browser.

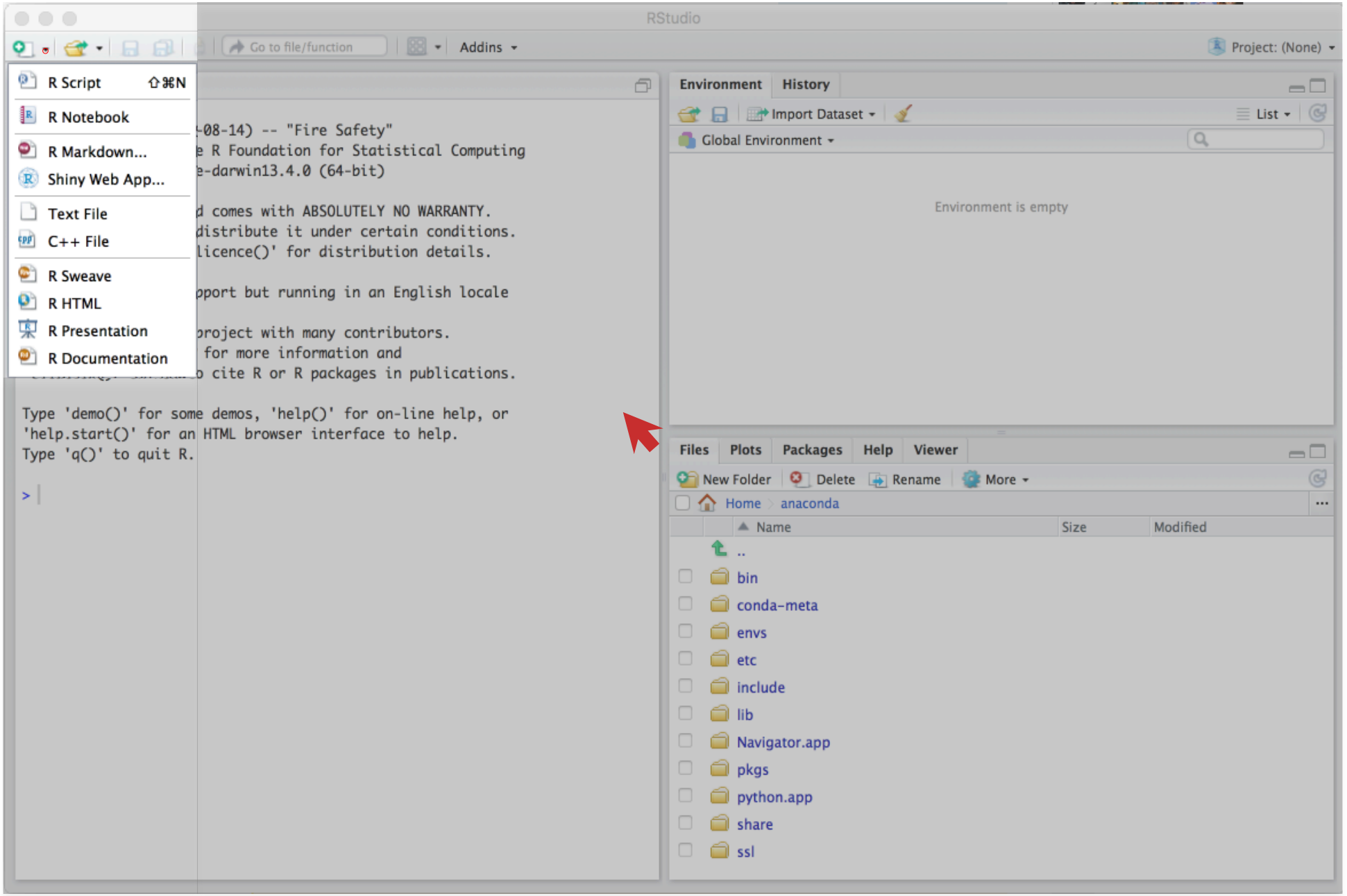

RStudio
Go to file/function
Addins
Project: (None)
Console ~/
R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>
Environment
History
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Home
anaconda
Name
Size
Modified
..
bin
conda-meta
envs
etc
include
lib
Navigator.app
pkgs
python.app
share
ssl

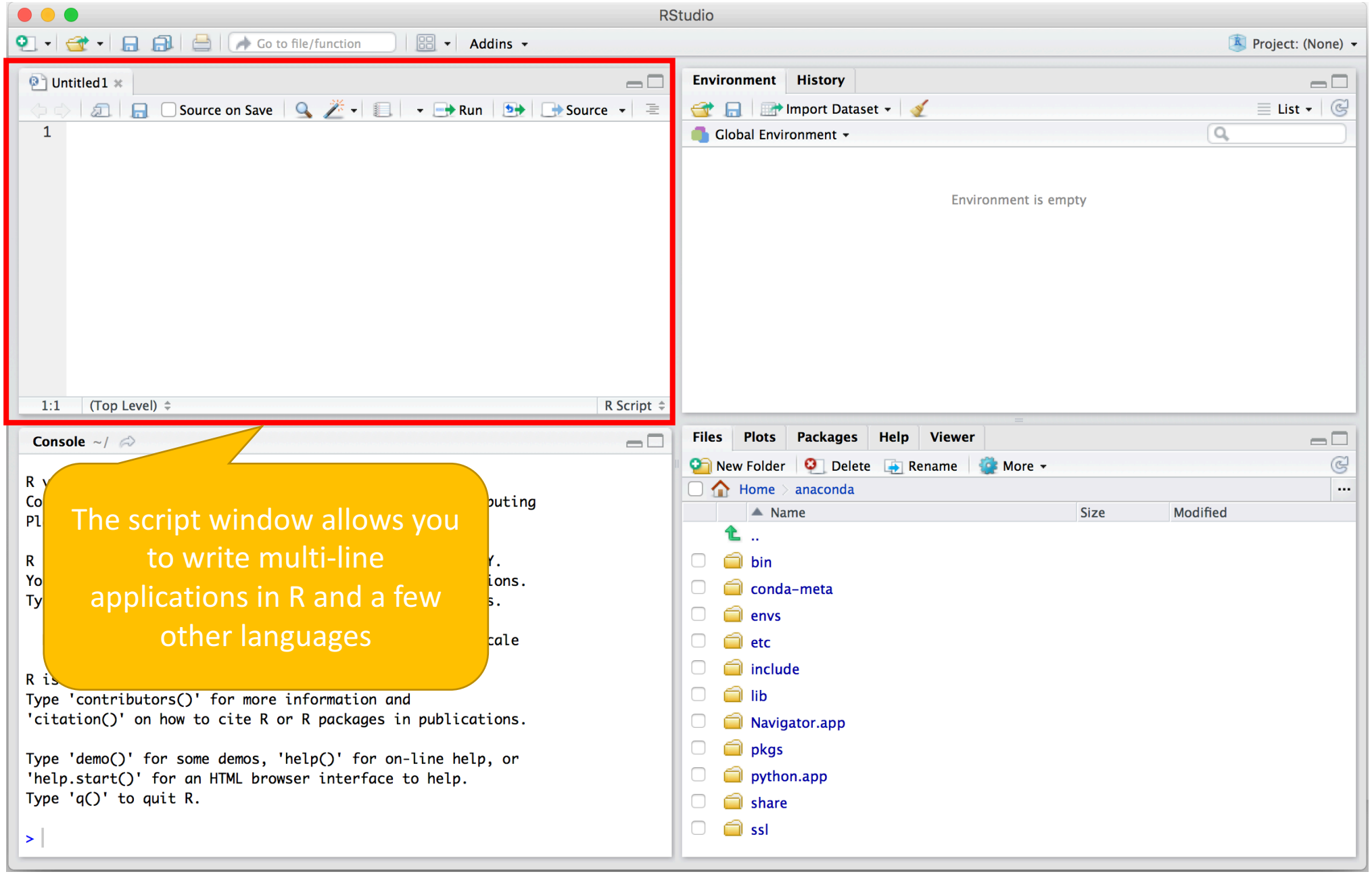

RStudio
Go to file/function
Addins
Project: (None)
Console ~/
R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>
Environment
History
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Home
anaconda
Name
Size
Modified
..
bin
conda-meta
envs
etc
include
lib
Navigator.app
pkgs
python.app
share
ssl
Console is an interactive command line interface with an R session pre-loaded

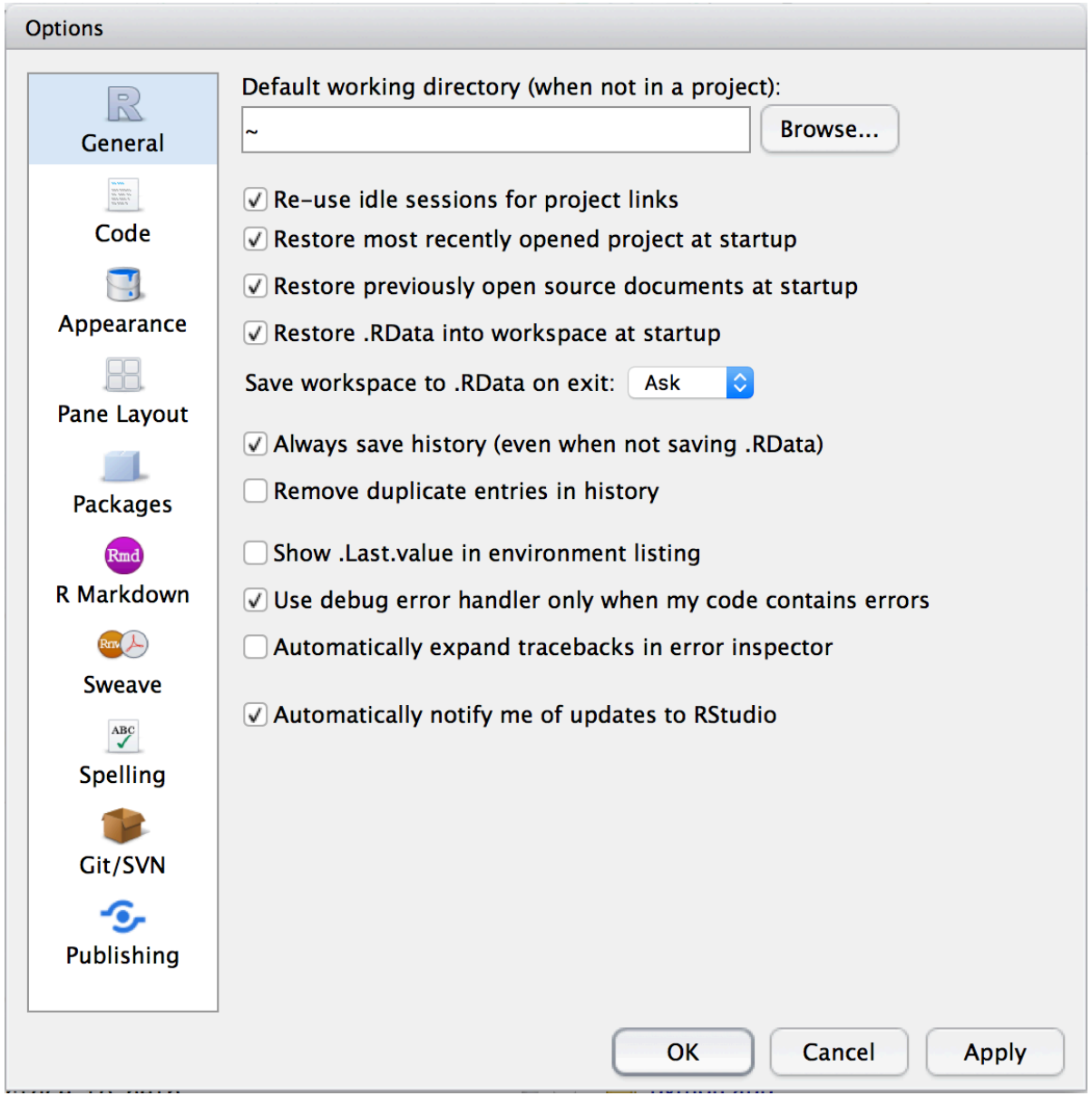

RStudio
Go to file/function
Addins
Project: (None)
Console ~/
R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>
Environment
History
Import Dataset
List
Global Environment
Environment is empty
Files
Packages
Help
Viewer
Delete
Rename
More
Size
Modified
pkgs
python.app
share
ssl
The Environment tab displays all current variables and objects loaded in your workspace.

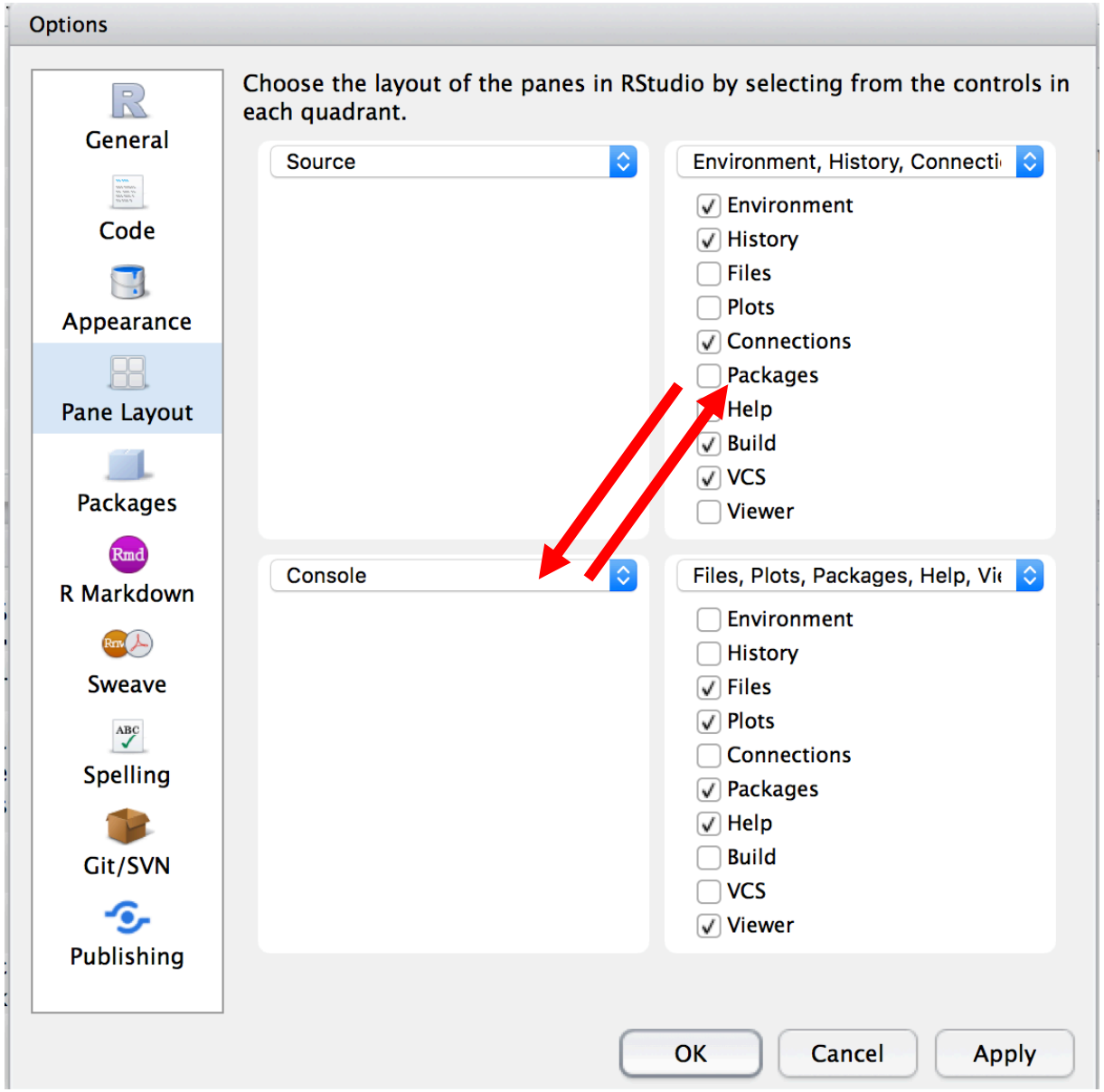

RStudio
Go to file/function
Addins
Project: (None)
Console ~/
R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>
Environment
History
Import Data
List
Global Environment
The History table shows all previously run R commands.
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Home
anaconda
Name
Size
Modified
..
bin
conda-meta
envs
etc
include
lib
Navigator.app
pkgs
python.app
share
ssl

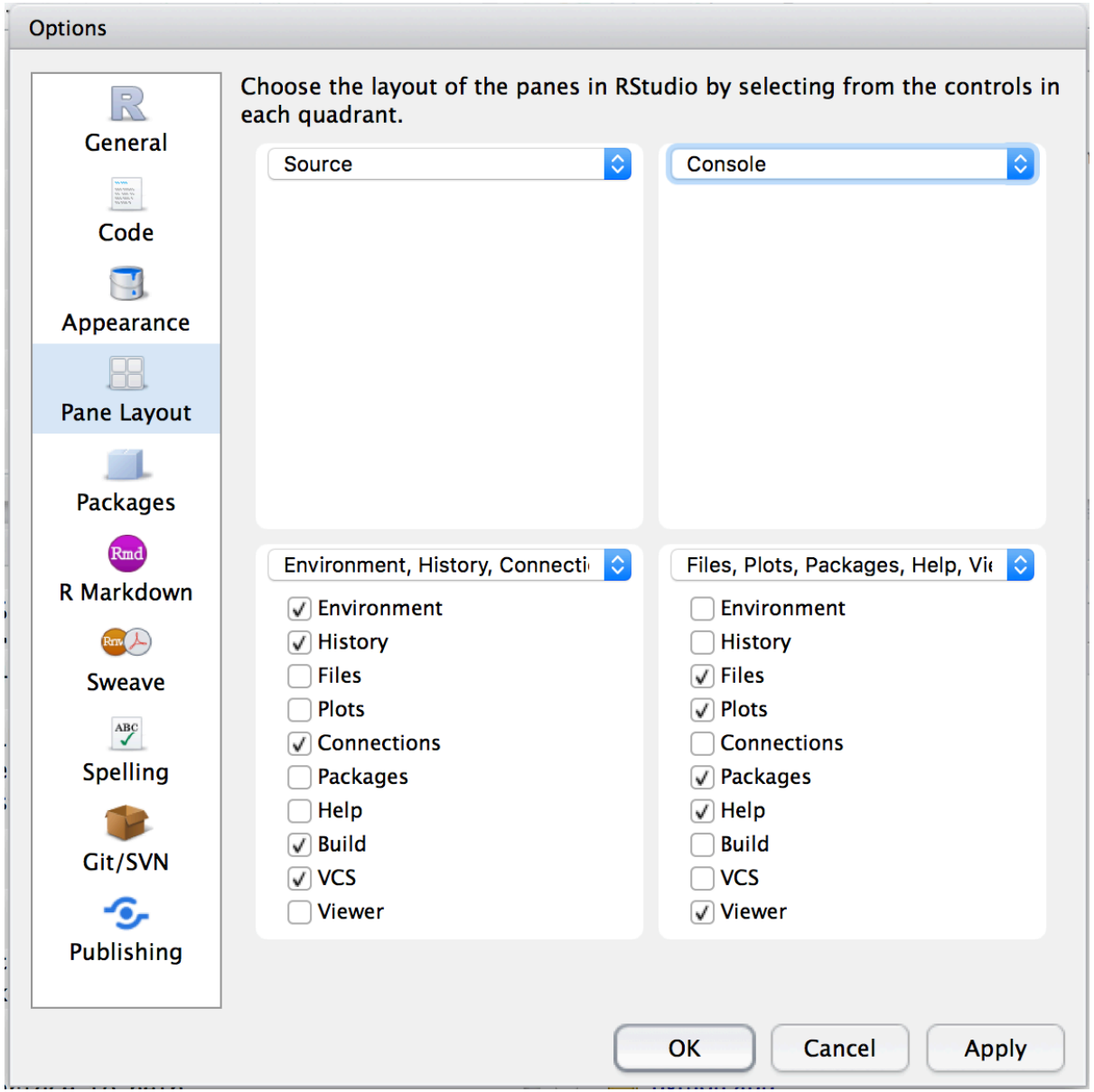

RStudio
Go to file/function
Addins
Project: (None)
Console ~/
R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for
Type 'q()' to quit
>
This pane shows content in the current working directory. Other tabs will be explained later on
Environment
History
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Home
anaconda
Name
Size
Modified
..
bin
conda-meta
envs
etc
include
lib
Navigator.app
pkgs
python.app
share
ssl

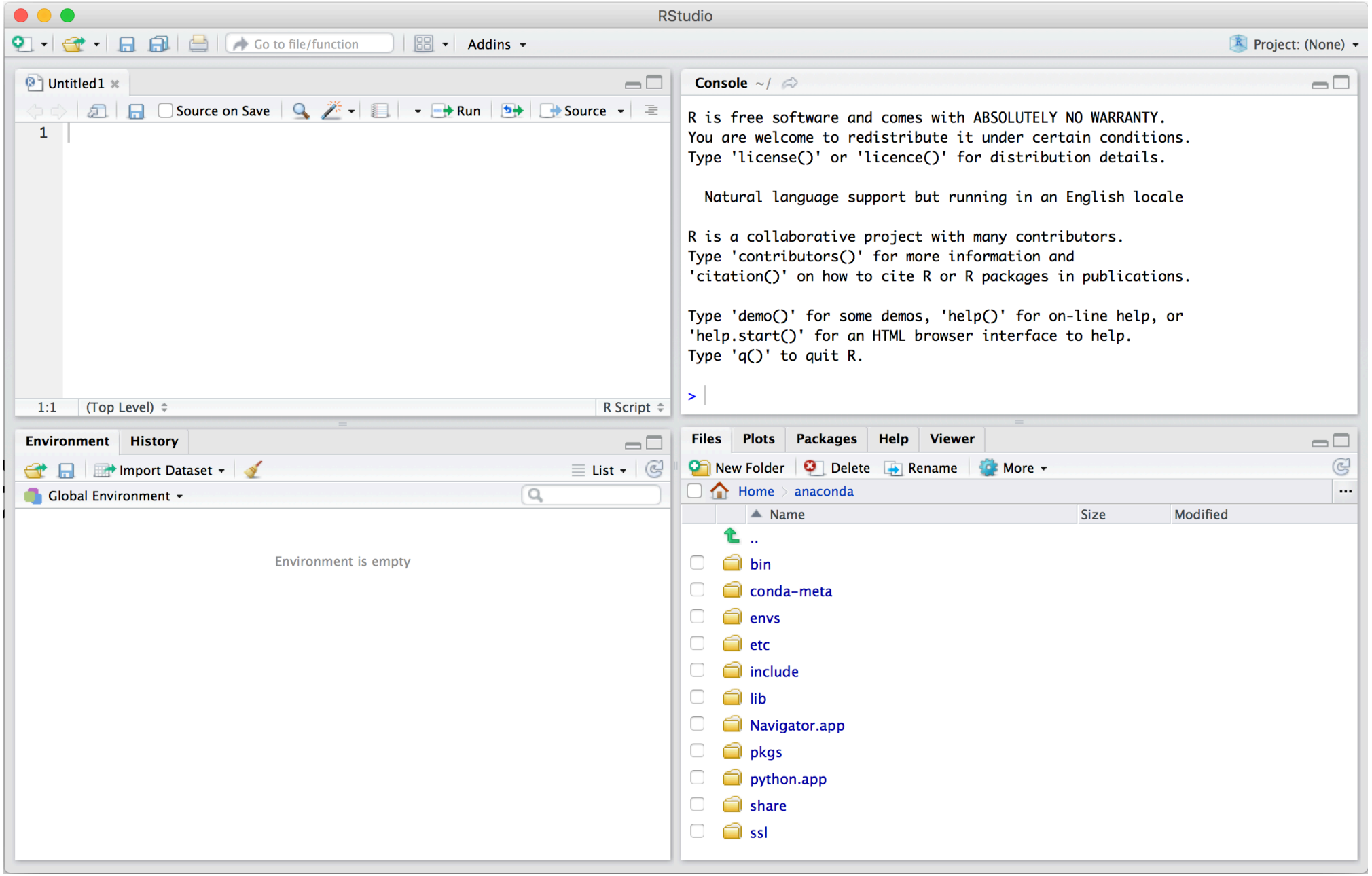

RStudio
Go to file/function
Addins
Project: (None)
R Script ⇧⌘N
R Notebook
R Markdown...
Shiny Web App...
Text File
C++ File
R Sweave
R HTML
R Presentation
R Documentation
-08-14) -- "Fire Safety"
e R Foundation for Statistical Computing
e-darwin13.4.0 (64-bit)
d comes with ABSOLUTELY NO WARRANTY.
distribute it under certain conditions.
licence()' for distribution details.
pport but running in an English locale
project with many contributors.
for more information and
o cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>
Environment
History
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Home
anaconda
Name
Size
Modified
..
bin
conda-meta
envs
etc
include
lib
Navigator.app
pkgs
python.app
share
ssl

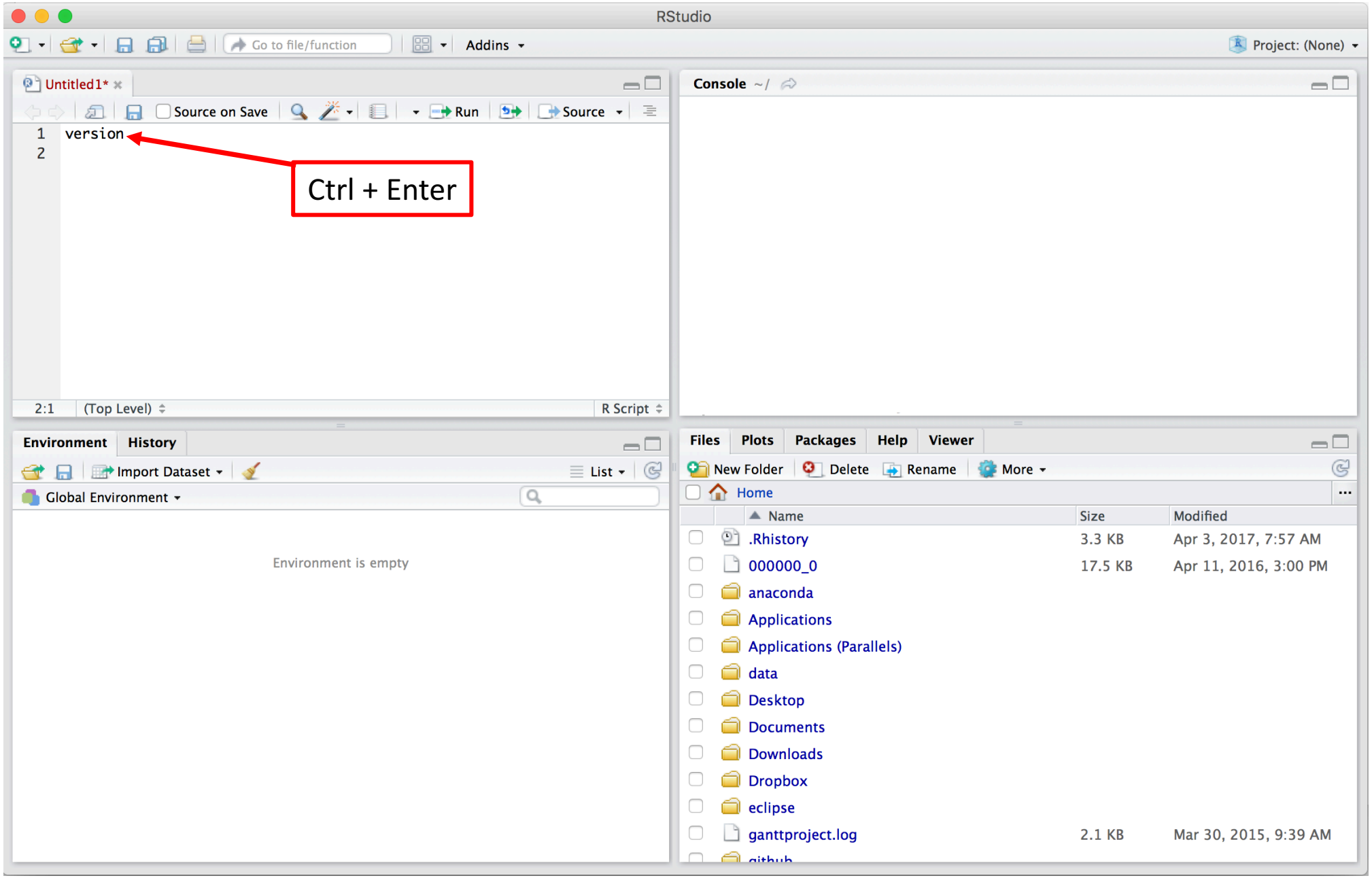

RStudio
Go to file/function
Addins
Project: (None)
Untitled1
Source on Save
Run
Source
1
1:1
(Top Level)
R Script
Environment
History
Import Dataset
List
Global Environment
Environment is empty
Console ~/
The script window allows you to write multi-line applications in R and a few other languages
Type '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.
>
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Home
anaconda
Name
Size
Modified
..
bin
conda-meta
envs
etc
include
lib
Navigator.app
pkgs
python.app
share
ssl

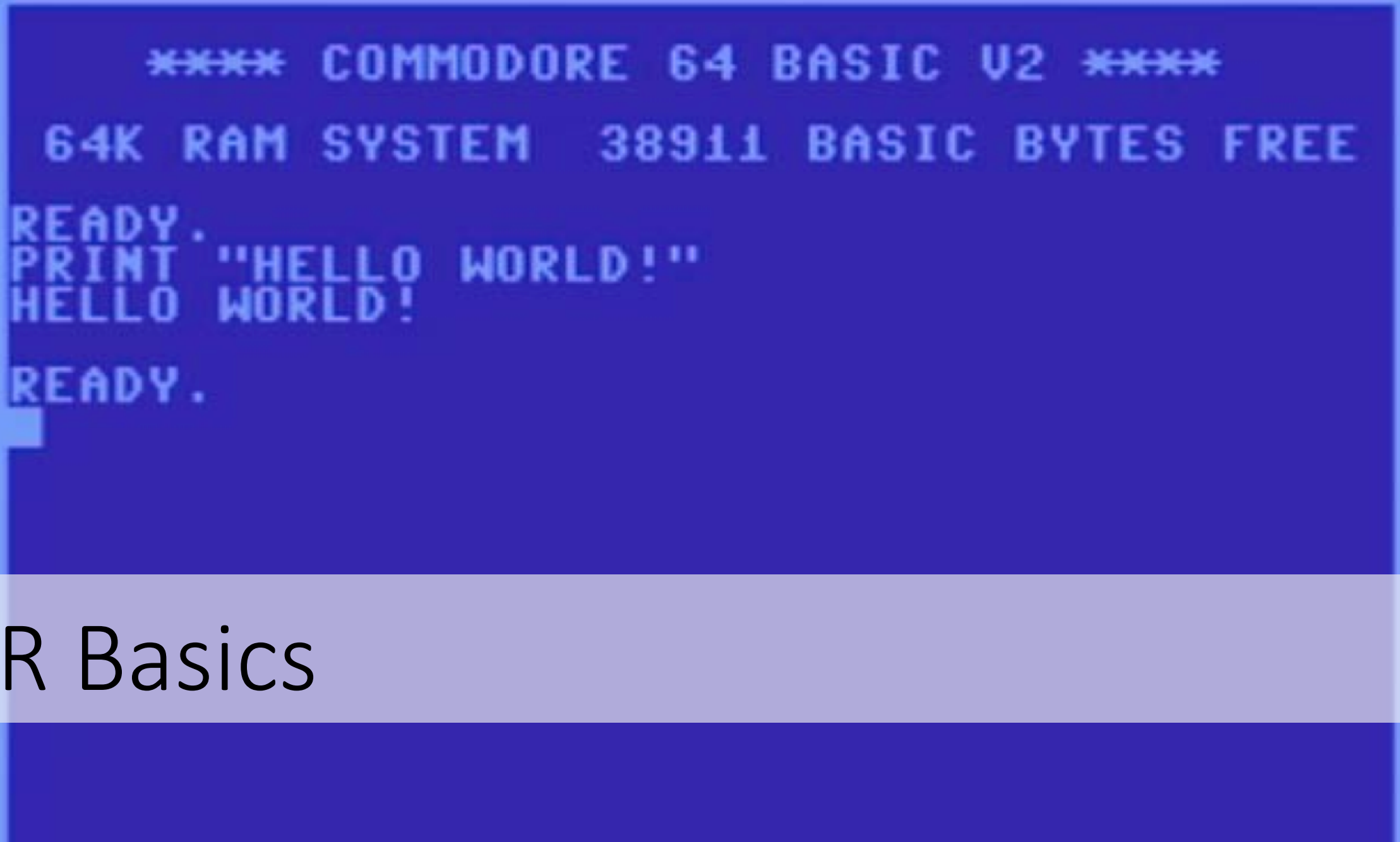

Options
General
Code
Appearance
Pane Layout
Packages
R Markdown
Sweave
Spelling
Git/SVN
Publishing
Default working directory (when not in a project):
~
Browse...
Re-use idle sessions for project links
Restore most recently opened project at startup
Restore previously open source documents at startup
Restore .RData into workspace at startup
Save workspace to .RData on exit:
Ask
Always save history (even when not saving .RData)
Remove duplicate entries in history
Show .Last.value in environment listing
Use debug error handler only when my code contains errors
Automatically expand tracebacks in error inspector
Automatically notify me of updates to RStudio
OK
Cancel
Apply


Options
General
Code
Appearance
Pane Layout
Packages
R Markdown
Sweave
Spelling
Git/SVN
Publishing
Choose the layout of the panes in RStudio by selecting from the controls in each quadrant.
Source
Environment, History, Connecti
Environment
History
Files
Plots
Connections
Packages
Help
Build
VCS
Viewer
Console
Files, Plots, Packages, Help, Vi
Environment
History
Files
Plots
Connections
Packages
Help
Build
VCS
Viewer
OK
Cancel
Apply


Options
General
Code
Appearance
Pane Layout
Packages
R Markdown
Sweave
Spelling
Git/SVN
Publishing
Choose the layout of the panes in RStudio by selecting from the controls in each quadrant.
Source
Console
Environment, History, Connecti
Environment
History
Files
Plots
Connections
Packages
Help
Build
VCS
Viewer
Files, Plots, Packages, Help, Vi
Environment
History
Files
Plots
Connections
Packages
Help
Build
VCS
Viewer
OK
Cancel
Apply


RStudio
Go to file/function
Addins
Project: (None)
Untitled1
Source on Save
Run
Source
1
1:1
(Top Level)
R Script
Console ~/
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>
Environment
History
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Home
anaconda
Name
Size
Modified
..
bin
conda-meta
envs
etc
include
lib
Navigator.app
pkgs
python.app
share
ssl


RStudio
Go to file/function
Addins
Project: (None)
Untitled1*
Source on Save
Run
Source
1 version
2
Ctrl + Enter
2:1
(Top Level)
R Script
Console ~/
Environment
History
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Home
Name
Size
Modified
.Rhistory 3.3 KB Apr 3, 2017, 7:57 AM
000000_0 17.5 KB Apr 11, 2016, 3:00 PM
anaconda
Applications
Applications (Parallels)
data
Desktop
Documents
Downloads
Dropbox
eclipse
ganttproject.log 2.1 KB Mar 30, 2015, 9:39 AM


**** COMMODORE 64 BASIC V2 ****
64K RAM SYSTEM 38911 BASIC BYTES FREE
READY.
PRINT "HELLO WORLD!"
HELLO WORLD!
READY.

R Basics

```
> 1 + 1
```

You can enter commands in the console

```
> 1 + 1
[1] 2
```

Enter commands in the console

Results will appear after the command

```
> 1 + 1
[1] 2
```

```
> print("Hello World")
```

Enter commands in the console

Results will appear after the command

Functions are called with parenthesis

```
> 1 + 1
[1] 2
```

```
> print("Hello World")
[1] "Hello World"
```

Enter commands in the console

Results will appear after the command

Functions are called with parenthesis

Functions accept zero to many parameters

```
> x <- 2
```

Variables don't require declaration. Assign with the <- operator.

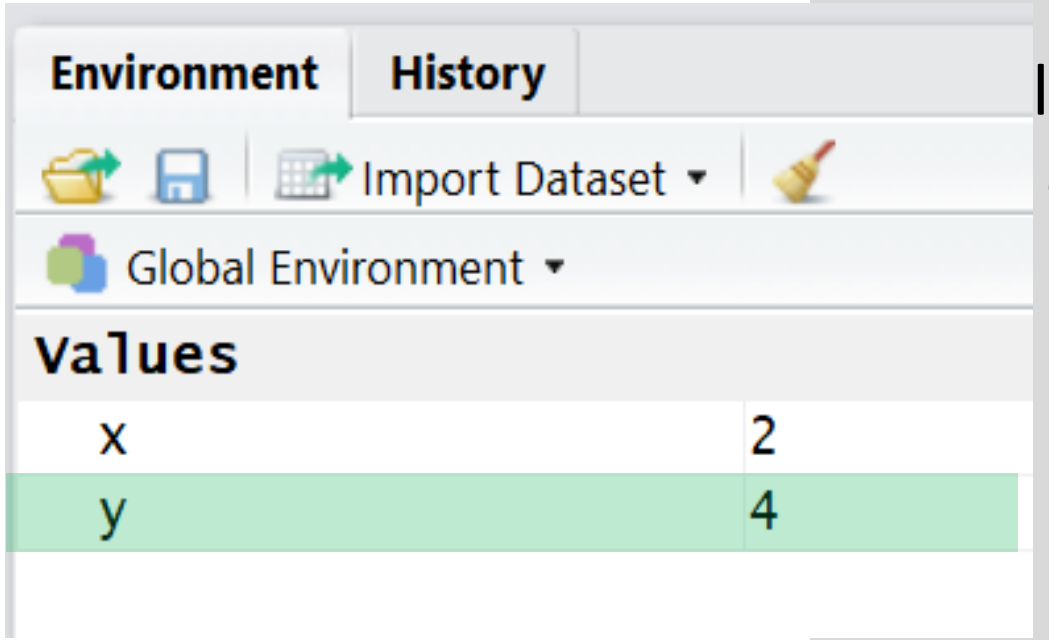
```
> x <- 2
> y <- x * x
```

Variables don't require declaration. Assign with the <- operator.

Variables can be used in future operations.

```
> x <- 2
> y <- x * x
```

Variables don't require declaration. Assign with the <- operator.



les can be used in operations.

```
> x <- 2
> y <- x * x

> (y <- x * x * x)
```

Variables don't require declaration. Assign with the <- operator.

Variables can be used in future operations.

Variables can be overwritten

```
> x <- 2
> y <- x * x
```

```
> (y <- x * x * x)
[1] 8
```

Variables don't require declaration. Assign with the <- operator.

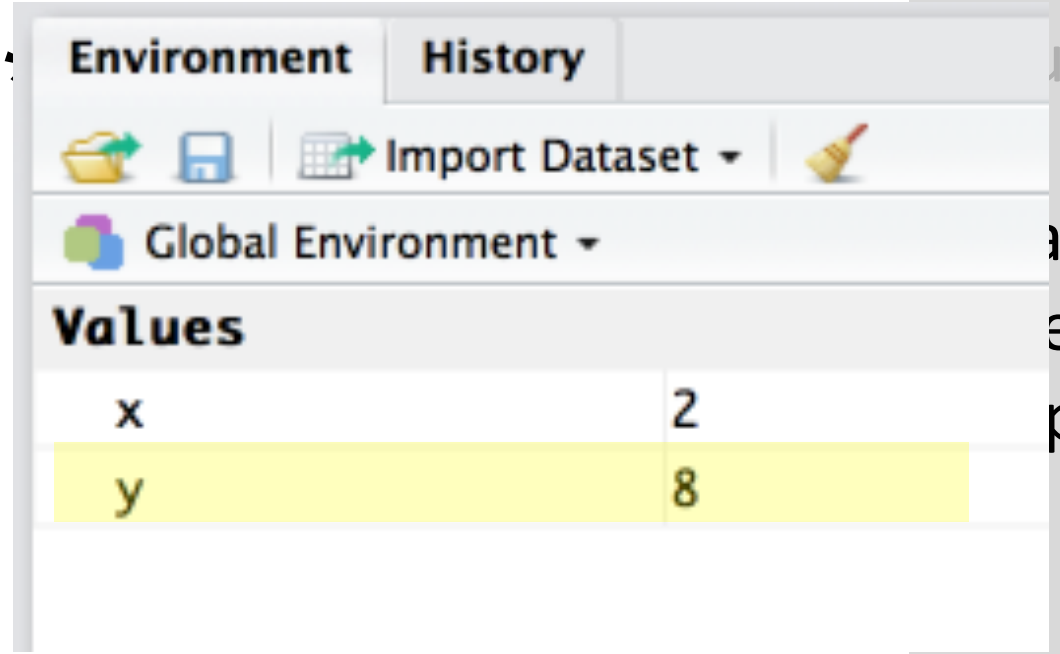
Variables can be used in future operations.

Variables can be overwritten

Wrapping commands in parenthesis forces console output

```
> x <- 2
> y <- x * x
```

```
> (y <- x
[1] 8
```



Variables don't require declaration. Assign with the <- operator.

Variables can be used in
ure operations.

apping commands in
enthesis forces console
put

```
> x <- 1
> typeof(x)
```

Data types in R are inferred by the data

```
> x <- 1
> typeof(x)
[1] "double"
```

Data types in R are inferred by the data

Numbers are inferred as doubles

```
> x <- 1
> typeof(x)
[1] "double"
```

Data types in R are inferred by the data

Numbers are inferred as doubles

```
> x <- 1
> typeof(integer(x))
```

You can manually coerce data types as needed

```
> x <- 1
> typeof(x)
[1] "double"
```

```
> x <- 1
> typeof(integer(x))
[1] "integer"
```

Data types in R are inferred by the data

Numbers are inferred as doubles

You can manually coerce data types as needed

Why do this? Integers require less memory than doubles. Or maybe you want to express a number as a character string.

```
> x <- 1
> class(x)
```

Everything in R is an object.

```
> x <- 1
> class(x)
[1] "numeric"
```

Everything in R is an object.

The class function shows what your object is inherited from.

```
> x <- 1
> class(x)
[1] "numeric"

> is.numeric(x)
```

Everything in R is stored as an object.

The class function shows what your object is inherited from.

You can also test for classes before performing an operation.

```
> x <- 1
> class(x)
[1] "numeric"

> is.numeric(x)
[1] TRUE
```

Everything in R is an object.

The class function shows what your object is inherited from.

You can also test for classes before performing an operation.

This is useful for certain types of arithmetic or models

```
> (myvec <- c(1,2,3))
```

The first kind of set is called a **vector**. It's like an array in other languages. It's a set of values, all of the same data type.

```
> (myvec <- c(1,2,3))
[1] 1 2 3
```

The first kind of set is called a **vector**. It's like an array in other languages. It's a set of values, all of the same data type.

Vectors are created with the c() function.

```
> (myvec <- c(1,2,3))
[1] 1 2 3
```

```
> str(myvec)
```

The first kind of set is called a vector. It's like an array in other languages. It's a set of values, all of the same data type.

Vectors are created with the c() function.

The str() function shows an object's structure

```
> (myvec <- c(1,2,3))
[1] 1 2 3
```

```
> str(myvec)
num [1:3] 1 2 3
```

The first kind of set is called a **vector**. It's like an array in other languages. It's a set of values, all of the same data type.

Vectors are created with the c() function.

The str() function shows an object's structure

myvec has one row and three columns; stored as numeric

```
> v2 <- c("a","b","c")
> str(v2)
```

Here is another vector, but this one has character values.

```
> v2 <- c("a","b","c")
> str(v2)
chr [1:3] "a" "b" "c"
```

Here is another vector, but this one has character values.

Like before, str() shows the structure of the vector. This time it's stored as characters.

```
> summary(myvec)
```

Summary() is another useful function that will display basic summary information about your object

```
> summary(myvec)
Min. 1st Qu. Median …
1.0   1.5     2.0 …
```

Summary() is another useful function that will display basic summary information about your object

Because **myvec** is numeric, summary stats are calculated

```
> summary(myvec)
Min. 1st Qu. Median …
1.0   1.5     2.0 …

> summary(v2)
```

Summary() is another useful function that will display basic summary information about your object

Because **myvec** is numeric, summary stats are calculated

But what about v2, our character vector?

```
> summary(myvec)
Min. 1st Qu. Median …
1.0   1.5     2.0 …
```

```
> summary(v2)
Length     Class      Mode
     3 character character
```

Summary() is another useful function that will display basic summary information about your object

Because **myvec** is numeric, summary stats are calculated

But what about v2, our character vector?

Summary shows a different set of information for a character vector

CHALLENGE

Creating a vector

Remember that a vector contains values of the same data type.

What will happen with this vector?

```
> quiz <- c(1,"2",3)
```

- R will raise an error
- Vector of numerics
- Vector of characters

```
> quiz <- c(1,"2",3)

> str(quiz)
```

Remember that a vector contains values that are all the same data type.

Reminder: use the str() function to examine the structure.

```
> quiz <- c(1,"2",3)

> str(quiz)
chr [1:3] "1" "2" "3"
```

Remember that a vector contains values that are all the same data type.

Reminder: use the str() function to examine the structure.

R will coerce all values in a vector to a common type, in this case, character

```
> lgvec <-
    sample(1:1000, 100)
```

Sometimes you'll need to select certain values from a vector. Consider this vector with 100 random values.

```
> lgvec <-
    sample(1:1000, 100)

> lgvec[75]
[1] 96
```

Sometimes you'll need to select certain values from a vector. Consider this vector with 100 random values.

Single values can be extracted using an index.

```
> lgvec <-
    sample(1:1000, 100)

> lgvec[75]
[1] 96

> lgvec[1:10]
[1] 642 756 263 463 …
```

Sometimes you'll need to select certain values from a vector. Consider this vector with 100 random values.

Single values can be extracted using an index.

You can also select a range of values. In this example I'm asking for the first 10 values from the vector.

```
> lgvec > 500
```

Another way to filter is by using a logical condition.

```
> lgvec > 500
[1] TRUE TRUE FALSE FA.
[11] TRUE FALSE FALSE .
```

Another way to filter is by using a logical condition.

This will return a vector containing the response to the condition for each value

```
> lgvec > 500
[1] TRUE TRUE FALSE FA.
[11] TRUE FALSE FALSE .
```

```
> gt500 <- lgvec > 500
> lgvec[gt500]
```

Another thing you can do with vector is to filter based on a condition.

This will return a vector containing the response to the condition for each value

So, if you save the result of the condition into an object, you can filter the original integer vector using the logical vector.

```
> lgvec > 500
[1] TRUE TRUE FALSE FA.
[11] TRUE FALSE FALSE .
```

```
> gt500 <- lgvec > 500
> lgvec[gt500]
[1] 642 756 579 536 790
[16] 525 765 598 577 …
```

Another thing you can do with vector is to filter based on a condition.

This will return a vector containing the response to the condition for each value

So, if you save the result of the condition into an object, you can filter the original integer vector using the logical vector.

CHALLENGE

Filtering Vectors

Using lgvec (our vector with 100 random integers) return only the even numbers; Or only the odd numbers, you decide.

```
-- HINT: R has a mod operator
> 9 %% 3
[1] 0
```

02:00

```
> (onlyEven
   <- lgvec %% 2 == 0)
```

Staring with lgvec, we first need to build a logical vector containing results of the conditional statement.

```
> (onlyEven
  <- lgvec %% 2 == 0)

> lgvec[onlyEven]
```

Staring with lgvec, we first need to build a logical vector containing results of the conditional statement

Now, we simply need to filter **lgvec** by the new logical vector, **onlyEven**

```
> (onlyEven
  <- lgvec %% 2 == 0)

> lgvec[onlyEven]

> lgvec[!onlyEven]
```

Staring with lgvec, we first need to build a logical vector containing results of the conditional statement

Now, we simply need to filter **lgvec** by the new logical vector, **onlyEven**

Did you want to return odd numbers? Easy enough with vector operations

```
> (l1 <-
     list(1, "a", 4))
```

A list is a set of **objects** with varying data types.

```
> (l1 <-
     list(1, "a", 4))
[[1]]
[1] 1

[[2]]
[1] "a"

[[3]]
[1] 4
```

A list is a set of objects with varying data types.

When you create this list, the output you'll see is the value of each list item.

```
> (l1 <-
      list(1, "a", 4))
[[1]]
[1] 1

[[2]]
[1] "a"

[[3]]
[1] 4
```

This is a **list slice** in position 1

This is the first value in slice 2

A list is a set of objects with varying data types.

When you create this list, the output you'll see is the value of each list item.

Since a list can contain different object types, it's organized differently than a vector

```
> l1[2]
```

You can select a list slice by using single square brackets.

```
> l1[2]
[[2]]
[1]"a"
```

You can select a list slice by using single square brackets

This will return the entire slice.

```
> l1[2]
[[2]]
[1]"a"
```

```
> l1[[2]]
[1] "a"
```

You can select a list slice by using single square brackets

This will return the entire list slice

To select only the value, use two square brackets

```
> typeof(l1[2])
```

Let's look at that another way.

```
> typeof(l1[2])
[1] "list"
```

Let's look at that another way

With single brackets, a list is returned

```
> typeof(l1[2])
[1] "list"
```

```
> typeof(l1[[2]])
[1] "character"
```

Let's look at that another way

With single brackets, a list is returned

But with double brackets a string type is returned – the type of the value in slice 2

```
> v1 <- c("a","b","c")
> v2 <- c(1,2,3)
> name <- "Josh"
```

Now for the big reveal…

```
> (l1 <-
       list(v1,v2,name)
[[1]]
[1] "a" "b" "c"
[[2]]
[1] 1 2 3
[[3]]
[1] "Josh"
```

Now for the big reveal…

Lists can contain vectors AND single values. You can mix many types of objects in a list.

CHALLENGE

List Selection

Given list **l1**, return the first item of the second slice. Then return the third item of the first slice.

```
-- Remember single brackets
-- vs double brackets
> l1[[1]] --returns the object
> l1[1] - returns a list
```

01:00

```
> vec2 <- l1[[2]]
```

First, remember that the second slice is a numeric vector. Use square brackets to isolate it

```
> vec2 <- l1[[2]]
> vec2[1]
[1] 1
```

First, remember that the second slice is a numeric vector. Use square brackets to isolate it

Then, select the first value from the vector object

```
> vec2 <- l1[[2]]
> vec2[1]
[1] 1
```

```
> l1[[1]][3]
[1] "c"
```

First, remember that the second slice is a numeric vector. Use square brackets to isolate it

Then, select the first value from the vector object

Because R is a functional language, you can also shortcut the reference

```
> names(l1)
NULL
```

In the real world, we don't often reference data by index. We name things.

```
> names(l1)
NULL
```

```
> names_l1 <-
      c("letters",
        "numbers",
        "me")
```

In the real world, we don't often reference data by index. We name things

To name slices in a list, start by creating a chr vector with the name of each item

```
> names(l1) <- names_l1
```

Now, pass the new vector of item names, **names_l1**, into the **names()** function

```
> names(l1) <- names_l1
> l1
$letters
[1] "a" "b" "c"

$numbers
[1] 1 2 3

$me
[1] "Josh"
```

Now, pass the new vector of item names, **names_l1**, into the **names()** function

Now when you return l1, each slice is named using the **$** notation

```
> l1$letters
[1] "a" "b" "c"
```

Now you can select an object by referencing the name using the "$" notation

```
> l1$letters
[1] "a" "b" "c"
```

```
>str(l1)
>summary(l1)
```

Now you can select an object by referencing the name using the "$" notation

Have you tried looking at the list structure or summary yet? Give it a shot.

Take a look at this database table. Does it remind you of anything?

	Stock Item Key	Stock Item	Color
1	0	Unknown	N/A
2	1	Void fill 400 L bag (White) 400L	N/A
3	2	Void fill 300 L bag (White) 300L	N/A
4	3	Void fill 200 L bag (White) 200L	N/A
5	4	Void fill 100 L bag (White) 100L	N/A
6	5	Air cushion machine (Blue)	N/A
7	6	Air cushion film 200mmx200mm 325m	N/A
8	7	Air cushion film 200mmx100mm 325m	N/A
9	8	Large replacement blades 18mm	N/A

	Stock Item Key	Stock Item	Color
1	0	Unknown	N/A
2	1	Void fill 400 L bag (White) 400L	N/A
3	2	Void fill 300 L bag (White) 300L	N/A
4	3	Void fill 200 L bag (White) 200L	N/A
5	4	Void fill 100 L bag (White) 100L	N/A
6	5	Air cushion machine (Blue)	N/A
7	6	Air cushion film 200mmx200mm 325m	N/A
8	7	Air cushion film 200mmx100mm 325m	N/A
9	8	Large replacement blades 18mm	N/A

Take a look at this database table. Does it remind you of anything?

The columns kind of look like a vector…

	Stock Item Key	Stock Item	Color
1	0	Unknown	N/A
2	1	Void fill 400 L bag (White) 400L	N/A
3	2	Void fill 300 L bag (White) 300L	N/A
4	3	Void fill 200 L bag (White) 200L	N/A
5	4	Void fill 100 L bag (White) 100L	N/A
6	5	Air cushion machine (Blue)	N/A
7	6	Air cushion film 200mmx200mm 325m	N/A
8	7	Air cushion film 200mmx100mm 325m	N/A
9	8	Large replacement blades 18mm	N/A

Take a look at this database table. Does it remind you of anything?

The columns kind of look like a vector…

A row kind of looks like a list…

	Stock Item Key	Stock Item	Color
1	0	Unknown	N/A
2	1	Void fill 400 L bag (White) 400L	N/A
3	2	Void fill 300 L bag (White) 300L	N/A
4	3	Void fill 200 L bag (White) 200L	N/A
5	4	Void fill 100 L bag (White) 100L	N/A
6	5	Air cushion machine (Blue)	N/A
7	6	Air cushion film 200mmx200mm 325m	N/A
8	7	Air cushion film 200mmx100mm 325m	N/A
9	8	Large replacement blades 18mm	N/A

[[1]] [[2]] [[3]]

Take a look at this database table. Does it remind you of anything?

The columns kind of look like a vector…

A row kind of looks like a list…

The collection of all rows looks like a list of vectors with named slices…

```
> (d1 <-
      data.frame(l1))
  letters numbers   me
1       a       1 Josh
2       b       2 Josh
3       c       3 Josh
```

In R, this is called a data.frame, and it's the most important set object you'll use at work.

```
> typeof(d1)
[1] "list"
```

Under the covers, a data.frame is stored as a list

```
> typeof(d1)
[1] "list"
```

```
> class(d1)
[1] "data.frame"
```

Under the covers, a **data.frame** is stored as a list

But it inherits from the **data.frame** class, so you'll get extra functionality on top of everything you know about working with a list

```
> typeof(d1)
[1] "list"

> class(d1)
[1] "data.frame"

> str(d1)
> summary(d1)
```

Under the covers, a **data.frame** is stored as a list

But it inherits from the **data.frame** class, so you'll get extra functionality on top of everything you know about working with a list

We'll spend the rest of the course working mostly with data.frames. For now, look at the str() and summary() to get acquainted.

```
> d1[3,]
```

You can select a single row of a dataframe

```
> d1[3,]
```

```
> d1[1:3,]
```

You can select a single row of a dataframe

Or, you can select a range of rows

```
> d1[3,]
```

```
> d1[1:3,]
```

```
> d1[1,1:2]
```

You can select a single row of a data.frame

Or, you can select a range of rows

Likewise, you can select columns along with (or without rows. Single columns or ranges are supported.

```
> d1[3,]
```

```
> d1[1:3,]
```

```
> d1[1,1:2]
```

```
> d1$field1[1,]
```

You can select a single row of a data.frame

Or, you can select a range of rows

Likewise, you can select columns along with (or without rows. Single columns or ranges are supported.

Just like a list, you can reference column names with the "$" notation

Congratulations!

You survived the introduction.

A winner is you!

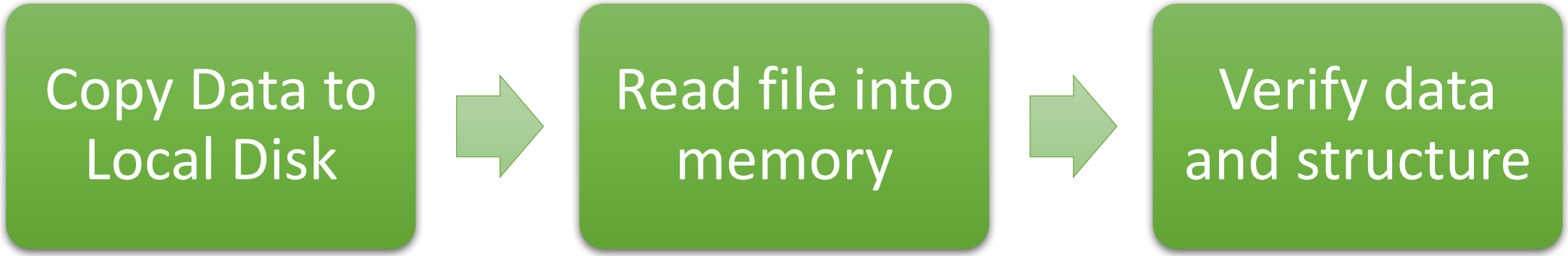
Acquiring Data with R

R can read data from many sources

In this workshop, we'll cover three major types you'll probabaly encounter at work

- CSV
- Excel
- Database

Loading data from files has three main steps



Breaking into secret NYC Taxi servers

Well, maybe we'll just download the CSV file of the open data they provide

https://s.thestreet.com/files/tsc/v2008/photos/contrib/uploads/nyctaxis_600x400.jpg

```
> getwd()
[1] "C:/Users/Josh/Documents"
```

R includes functions to manipulate directories and files on local disk

getwd() returns the current working directory

```
> getwd()
[1] "C:/Users/Josh/Documents"

> setwd(tempdir())
```

R includes functions to manipulate directories and files on local disk

getwd() returns the current working directory

setwd() allows you to change your working directory -- tempdir() creates a new temp directory

This line sets your current working directory to a new temp directory

```
> getwd()
[1] "C:/Users/Josh/Documents"

> setwd(tempdir())

> getwd()
[1] "C:/Users/AppData/Local/Temp…"
```

setwd() allows you to change your working directory -- tempdir() creates a new temp directory

This line sets your current working directory to a new temp directory

Your current working directory is now in your system's temp folder

```
-- check if a directory exists
> dir.exists()

-- create a new directory
> dir.create()

-- build a file path string
> file.path()
> file.path(getwd(),
                  "newfile.txt")
```

There are additional functions that help with manipulating directories and files on disk

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

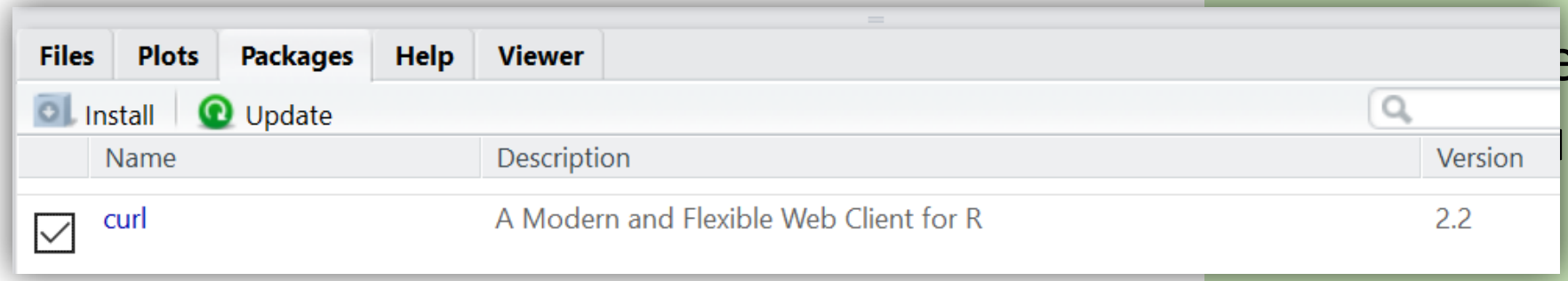
The curl package provides basic network interface functionality, like HTTP/FTP to allow you to easily download files from remote sources

R Packages are developed by the open source community.

Packages first must be installed.

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

```
> library("curl")
```



R Packages are developed by the open source community

Packages first must be installed

ed, load the
sing the library

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

> library("curl")

> curl_download(
                fileToDownload,
                destfile="file")
```

R Packages are developed by the open source community

Packages first must be installed

Once installed, load the package(s) using the library function

The curl package includes a curl_download() function to download files

Your turn - Download a zipped CSV

1. Create a new Rscript to be used for the workshop
2. Ensure the **curl** package is installed and loaded
3. Set your working directory to a new temp directory
4. Download the file using **`curl_download()`**
5. BONUS - Unzip the file with the **`unzip()`** function

https://bgmrstrainingfiles.blob.core.windows.net/lab-data/NYC_sample.zip

05:00

Solution - Download a CSV

```
1  install.packages("curl")
2  library("curl")
3
4  Setwd(tempdir())
5  fileurl <- "https://bgmrstrainingfiles.blob.core.windows.net/lab-data/NYC_sample.zip"
6
7  curl_download(fileurl, destfile="NYC_Sample.zip")
8
9
10
```

Explore a file: read the first 3 lines

```
print(readLines(file('NYC_Sample.csv'), n=3))
```

```
[1] "vendorid,tpep_pickup_datetime,tpep_dropoff_datetime,passenger_count,trip_distance,pickup_long
fwd_flag,dropoff_longitude,dropoff_latitude,payment_type,fare_amount,extra,mta_tax,improvement_sur
"
[2] "1,2016-01-29 09:29:04,2016-01-29 09:39:46,1,1.40,-73.994697570800781,40.725814819335938,1,N,-
,0,0.5,1,0,0.3,10.3"

[3] "1,2016-01-29 09:29:09,2016-01-29 09:37:48,1,2.60,-73.974021911621094,40.746917724609375,1,N,-
5,0,0.5,1,0,0.3,12.3"
```

R and CSV files

R prefers "clean" CSV files:

- Headers
- No-quoted strings
- Standard seperators
- Well formatted line endings

Fill a data frame

Base R

```
target_df <- read.csv('file.csv')
```

Dates read as strings

Slow performance

No feedback

Fill a data frame

A better option

```
install.packages('readr')
library(readr)

target_df <- read_csv(file='file.csv')
```

Your turn – Load the taxi data with ***readr***

1. Ensure the **readr** package is installed and loaded
2. Fill a *data.frame*, `nyc_taxi`, using `read_csv()`
3. BONUS - Answer the following questions about `nyc_taxi` using `str()` and `summary()`

`How many variables are in nyc_taxi?`

`What is the data type of the tpep_dropoff_datetime variable?`

`What was the largest tip_amount?`

Solution – Load the taxi data

```
1   install.packages("readr")
2   library("readr")
3
4   nyc_taxi <- read_csv("NYC_Sample.csv")
5
6   ##examine the data.frame structure and data
7   str(nyc_taxi)
8   summary(nyc_taxi)
9
10  ##show first few rows
11  head(nyc_taxi)
12
```

Identifying Holidays

Which happen to be stored in an Excel file

http://magneticmediatv.com/wp-content/uploads/2015/09/holiday-2.jpg

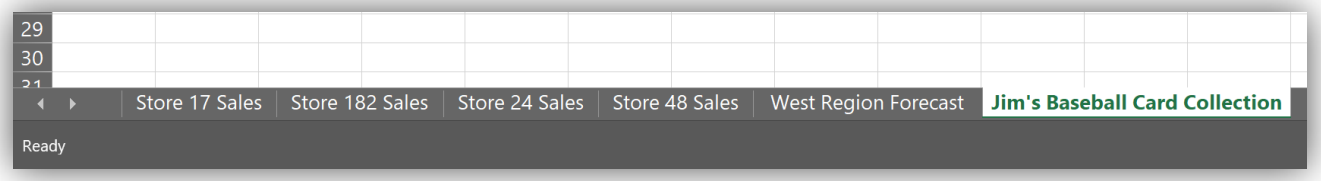
Consuming Excel files

Excel files can be a challenge:

Data in arbitrary locations

	A	B	C	D	E
1					
2		Kids			
3	Name	Age			
4	Sam	8			
5	James	6			
6	Tiffany	11		Parents	
7	Jack	13		Name	Age
8				Josh	37
9				Ned	26
10				Staci	25
11				Monica	31
12					

Multiple Sheets



Consuming Excel files

Readxl package

```
install.packages("readxl")
library("readxl")
```

Consuming Excel files

Readxl package

```
##read one sheet of a file
read_excel("file.xlsx", sheet = "Sheet 1")
read_excel("file.xlsx",
                sheet = "Sheet 1",
                range="C25:G75")
```

Consuming Excel files

Readxl package

```
##read multiple sheets
excel_sheets("file.xslx") #returns a list

lapply(excel_sheets("file.xslx"),
       read_excel,
       path = "file.xslx")
```

Your turn – Load the holidays with readXL

1. Ensure the **readXL** package is installed and loaded
2. Fill a *data.frame*, **holidays**, using **read_excel()**
3. BONUS - Answer the following questions about **holidays**

How many observations are in **holidays?**

Which **`holidayname`** is in the 13th row?

What **date** is exactly in the middle of the data.frame?

Solution – Load the taxi data

```
1   install.packages("readxl")
2   library("readxl")
3
4   holidays <- read_excel("holidays.xlsx")
5
6   ##examine the data.frame structure and data
7   str(holidays)
8   summary(holidays)
9
10  ##peek at the 13th row
11  holidays[13,]
12
```

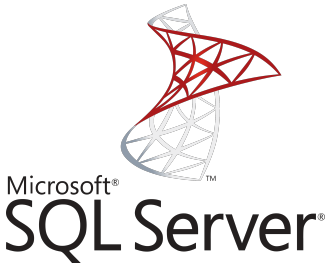
Don't check the weather

Actually, do check it, it might come in handy for our analysis

https://cdn20.patch.com/inline_images/22859565/1490711638.jpg

Connecting to a database

In order to build business ready R applications, we'll need to be able to connect to a database.



Connecting to a database

RODBC package

```
install.packages("RODBC")
library("RODBC")
```

NOTE ** You may have to install/update unixODBC on Mac or Linux systems

Connecting to a database

Step 1: Make a connection

Using a DSN (System or User)

```
dsn <- "MyDatabaseConnection"
openChannel <-
        odbcConnect(dsn=, uid=, pwd=)
```

Using an explicit connection string

```
conStr <- "driver=;server=;database;uid=;pwd="
openChannel <-
        odbcDriverConnect(connection=conStr)
```

Connecting to a database

Step 2: Check connection info

Use the channel object created in the connection step

```
odbcGetInfo(openChannel)
```

```
      DBMS_Name                                  DBMS_Ver
        "MySQL"                                  "5.6.26.0"
Driver_ODBC_Ver                          Data_Source_Name
        "03.80"                                  "03.80"
     Driver_Name                               Driver_Ver
"libmyodbc5w.so"                             "05.03.0008"
       ODBC_Ver                               Server_Name
  "03.52.0000" "jf.mysql.database.azure.com via TCP/IP"
```

Connecting to a database

Step 3: List tables found in the database

Use the channel object created in the connection step

```
sqlTables(openChannel)
```

```
  TABLE_CAT TABLE_SCHEM   TABLE_NAME TABLE_TYPE REMARKS
1   weather             dailyrecords      TABLE
```

Connecting to a database

Step 4: Examine table schema

Use the channel object created in the connection step

```
sqlColumns(channel = openChannel,
           sqtable = "dailyrecords",
           schema = "weather")
```

	TABLE_CAT	TABLE_SCHEM	TABLE_NAME	COLUMN_NAME	DATA_TYPE	TYPE_NAME	COLUMN_SIZE	BUFFER_LENGTH	DECIMAL_DIGITS
1		<NA>	dailyrecords	id	12	varchar	20	20	NA
2		<NA>	dailyrecords	yyyymmdd	4	integer	10	4	0
3		<NA>	dailyrecords	prcp	3	decimal	12	14	9
4		<NA>	dailyrecords	tmax	3	decimal	6	8	2
5		<NA>	dailyrecords	tmin	3	decimal	6	8	2

	NUM_PREC_RADIX	NULLABLE	REMARKS	COLUMN_DEF	SQL_DATA_TYPE	SQL_DATETIME_SUB	CHAR_OCTET_LENGTH	ORDINAL_POSITION	IS_NULLABLE
1	NA	1	<NA>	<NA>	12	NA	20	1	YES
2	10	1	<NA>	<NA>	4	NA	NA	2	YES
3	10	1	<NA>	<NA>	3	NA	NA	3	YES
4	10	1	<NA>	<NA>	3	NA	NA	4	YES
5	10	1	<NA>	<NA>	3	NA	NA	5	YES

Connecting to a database

Step 5: Read data

Use the channel object created in the connection step

```
sqlFetch(channel = openChannel,
             sqtable = "[tablename]",
             schema = "[schema]")
```

Connecting to a database

Additional Functionality

Use the channel object created in the connection step

```
sqlSave() #creates a new table (if not exists)
          #and appends rows

sqlUpdate() #updates existing rows in a table

sqlClear() #removes all rows (TRUNCATE)

sqlDrop() #drops a table
```

Your turn – Load the holidays data with RODBC

Option 1 – SQL Server
Server: jfdw.database.windows.net
Database: weather
Username: weatherReader
Password: Weather1

Option 2 – MySQL
Server: jf.mysql.database.azure.com
Schema: weather
Username: weatherReader@jf
Password: Weather1

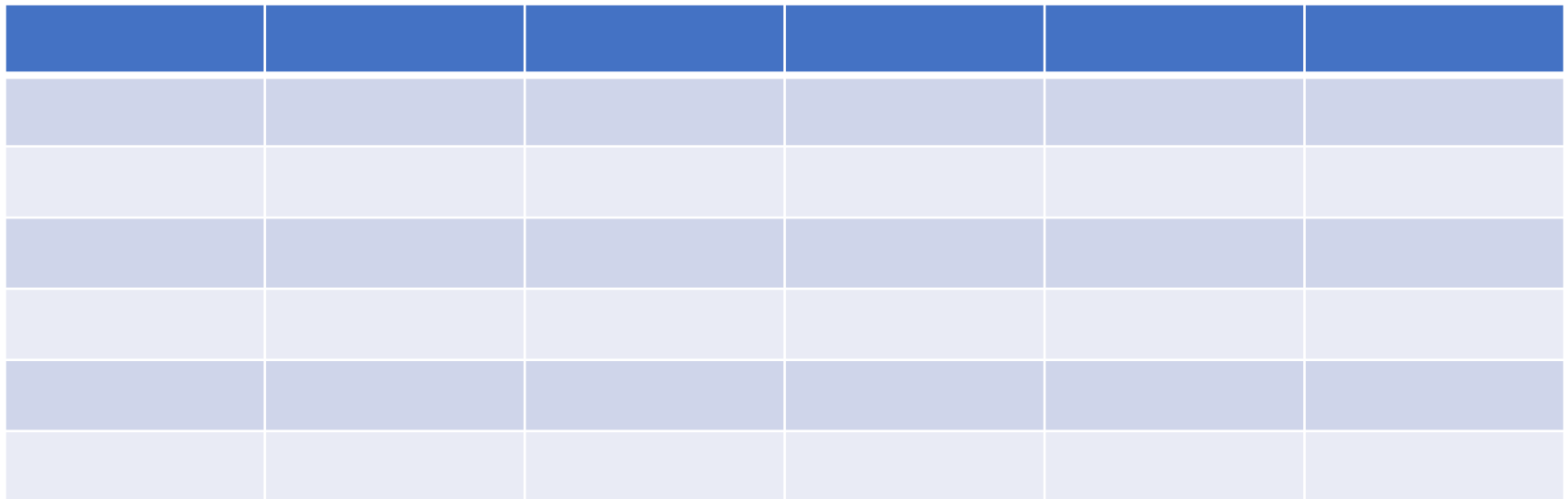
1. Ensure the **RODBC** package is installed and loaded
2. Fill a *data.frame*, **`weather`**,
3. BONUS Questions

`How many days had precipitation?`

`What was the average temp on July 4?`

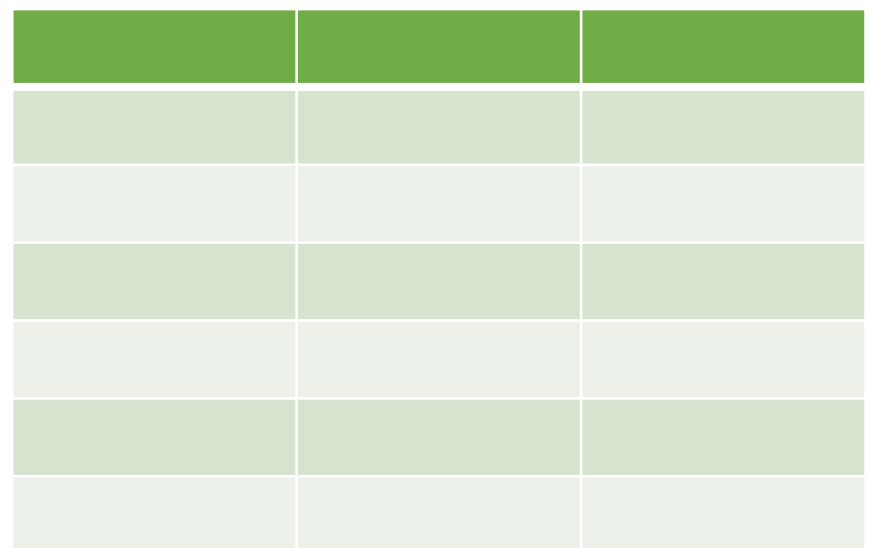
`What day had the lowest temp?`

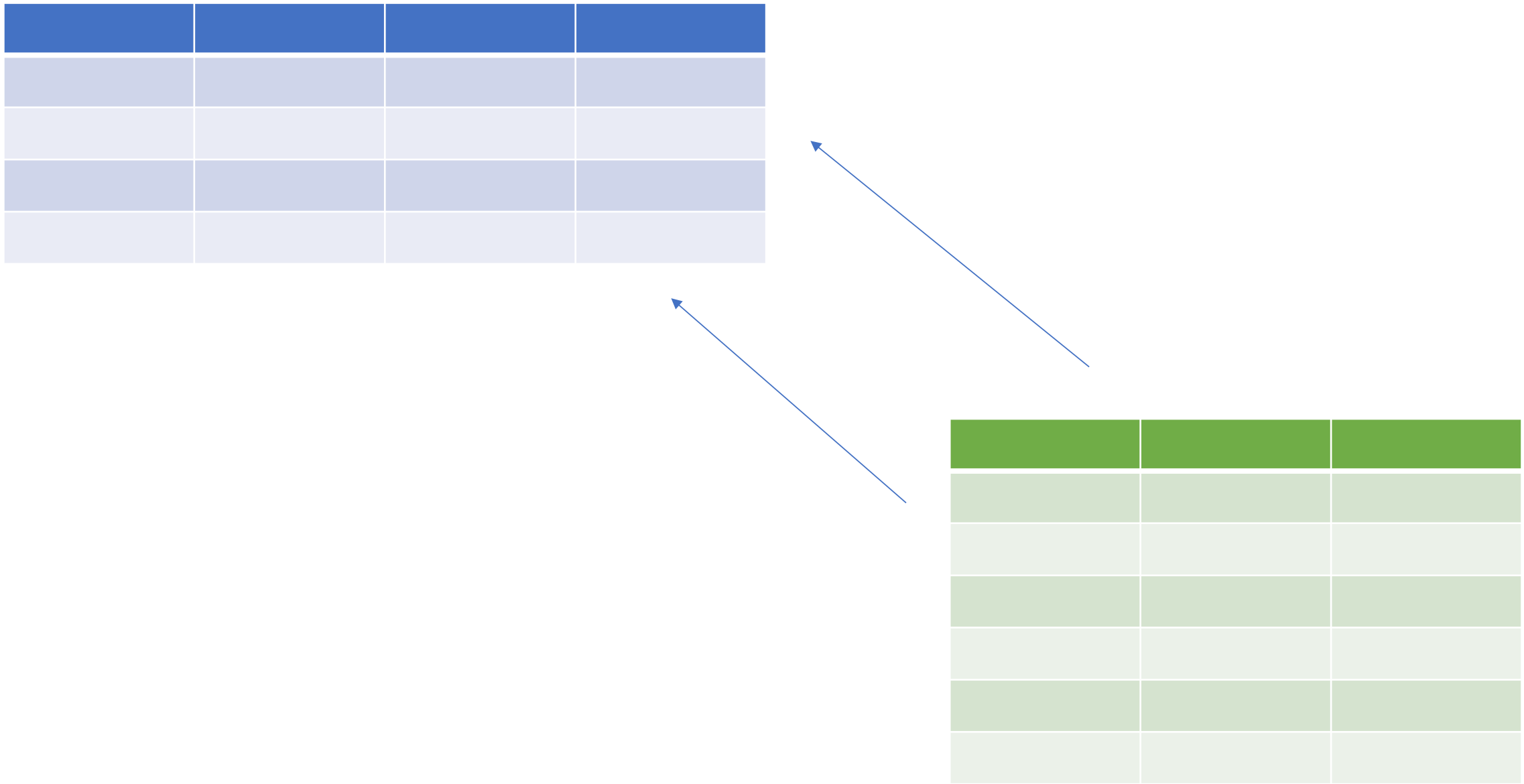
Data Cleansing and Transformation

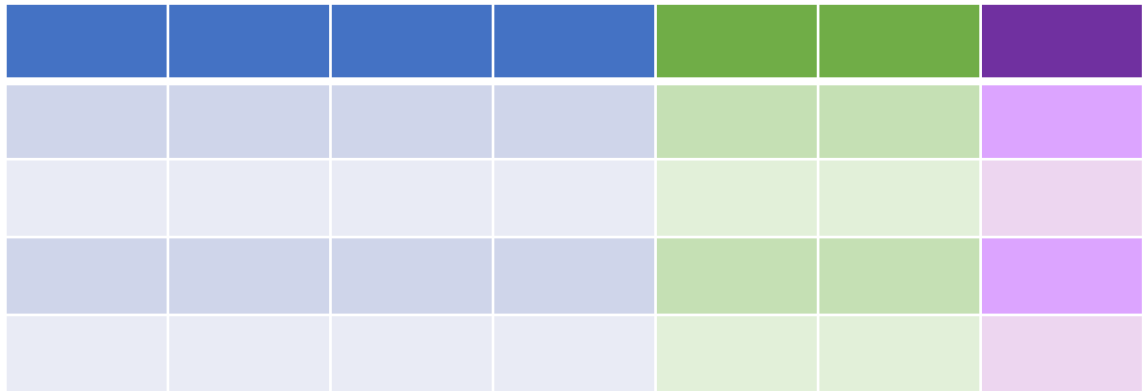


After data is loaded…

You'll probably have multiple `data.frames`







The multitool of R Development - dplyr


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

df <- filter(df, variable1 >= 100)
```

The multitool of R Development - dplyr


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

df <- filter(df, variable1 >= 100)
```

dplyr function

data.frame (or tibble)

Function arguments

Making hard tasks easier to perform…


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

df <- select(
          filter(df, variable1 >= 100),
          variable1:variable5)
```

… and read


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

df <- filter(df, variable1 >= 100) %>%
      select(variable1:variable5)
```

Common tasks

If you want to….	Use the *dplyr* function
Return a subset of rows	`filter()`
Return a subset of columns	`select()`
Reorder rows	`arrange()`
Rename an existing column	`rename()`
Create a new column	`mutate()`
Replace an existing column	`transmute()`
Merge columns of two tables	`left_join()`, `right_join()`, `inner_join()`, `full_join()`
Find all rows of one table matching rows in another (or vice versa)	`semi_join()`, `anti_join()`

dplyr::rename()

Rename one column

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

df <- rename(df, new_name = old_name)
```

dplyr::rename()

Rename more than one column

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

df <- rename(df,
             new_name = old_name,
             next_new = next_old)
```

Your turn – rename some columns

1. Here's a softball. Use `dplyr` to rename the following columns.
    - tpep_dropoff_datetime to dropoff_time
    - tpep_pickup_datetime to pickup_time

2:00

dplyr::select()

Select specific columns

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

df <- select(df, f1, f2, f3, f4, f5)
```

dplyr::select()

Select a range of columns

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

df <- select(df, f1:f5)
```

dplyr::select()

Select all columns except

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

df <- select(df, -c(f6, f7, f8))
```

dplyr::select()

Mix it up…

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

df <- select(df, f1:f6, -(f4))
```

dplyr::select()

Use a vector to select

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

colnames <- c("f1", "f2", "f3")
df <- select(df, .dots = colnames)
```

dplyr::select()


```
   vendorid     tpep_pickup_datetime          tpep_dropoff_datetime         passenger_count trip_distance
Min.   :1.000   Min.   :2016-01-01 00:00:03   Min.   :2016-01-01 00:00:11   Min.   :0.00    Min.   :     0.0
1st Qu.:1.000   1st Qu.:2016-02-17 20:40:37   1st Qu.:2016-02-17 20:54:03   1st Qu.:1.00    1st Qu.:     1.0
Median :2.000   Median :2016-04-01 17:24:37   Median :2016-04-01 17:41:41   Median :1.00    Median :     1.7
Mean   :1.531   Mean   :2016-04-01 12:58:44   Mean   :2016-04-01 13:14:36   Mean   :1.66    Mean   :     4.4
3rd Qu.:2.000   3rd Qu.:2016-05-15 08:52:33   3rd Qu.:2016-05-15 09:07:12   3rd Qu.:2.00    3rd Qu.:     3.2
Max.   :2.000   Max.   :2016-06-30 23:59:52   Max.   :2016-07-01 21:48:41   Max.   :9.00    Max.   :1587540.0
pickup_longitude  pickup_latitude    ratecodeid     store_and_fwd_flag dropoff_longitude dropoff_latitude  payment_type
Min.   :-121.93   Min.   : 0.00   Min.   : 1.000   Length:1388631     Min.   :-121.93   Min.   : 0.00   Min.   :1.000
1st Qu.: -73.99   1st Qu.:40.74   1st Qu.: 1.000   Class :character   1st Qu.: -73.99   1st Qu.:40.73   1st Qu.:1.000
Median : -73.98   Median :40.75   Median : 1.000   Mode  :character   Median : -73.98   Median :40.75   Median :1.000
Mean   : -72.93   Mean   :40.18   Mean   : 1.052                      Mean   : -73.00   Mean   :40.21   Mean   :1.343
3rd Qu.: -73.97   3rd Qu.:40.77   3rd Qu.: 1.000                      3rd Qu.: -73.96   3rd Qu.:40.77   3rd Qu.:2.000
Max.   :   0.00   Max.   :48.51   Max.   :99.000                      Max.   :   0.00   Max.   :44.62   Max.   :4.000
 fare_amount         extra              mta_tax         improvement_surcharge   tip_amount        tolls_amount
Min.   :-198.00   Min.   :-16.6100   Min.   :-1.0000   Min.   :-14.320       Min.   :-10.5000   Min.   :-0.3000
1st Qu.:   6.50   1st Qu.:  0.0000   1st Qu.: 0.5000   1st Qu.:  0.000       1st Qu.:  0.0000   1st Qu.: 0.3000
Median :   9.50   Median :  0.0000   Median : 0.5000   Median :  1.330       Median :  0.0000   Median : 0.3000
Mean   :  12.88   Mean   :  0.3323   Mean   : 0.4975   Mean   :  1.801       Mean   :  0.3195   Mean   : 0.2997
3rd Qu.:  14.50   3rd Qu.:  0.5000   3rd Qu.: 0.5000   3rd Qu.:  2.360       3rd Qu.:  0.0000   3rd Qu.: 0.3000
Max.   :3013.50   Max.   : 51.0000   Max.   :17.4100   Max.   :399.990       Max.   :905.6200   Max.   : 0.7600
 total_amount
Min.   :-208.80
1st Qu.:   8.75
Median :  11.80
Mean   :  16.13
3rd Qu.:  17.80
Max.   :3014.30
```

Your turn – clean up columns with select()

1. Ensure the `dplyr` packages in installed and loaded
2. Using `dplyr`, devise three (3) ways to remove the following columns from `nyc_taxi`
    - `Vendorid`
    - `Ratecodeid`
    - `Store_and_fwd_flag`
    - `Payment_type`

10:00

dplyr::filter()

Select rows based on a single condition

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

df <- filter(sales, total_price >= 100.00)
```

dplyr::filter()

Select rows based on multiple conditions

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

df <- filter(sales,
                  total_price >= 100.00,
                  store_name = "Grand Rapids")
```

dplyr::filter()

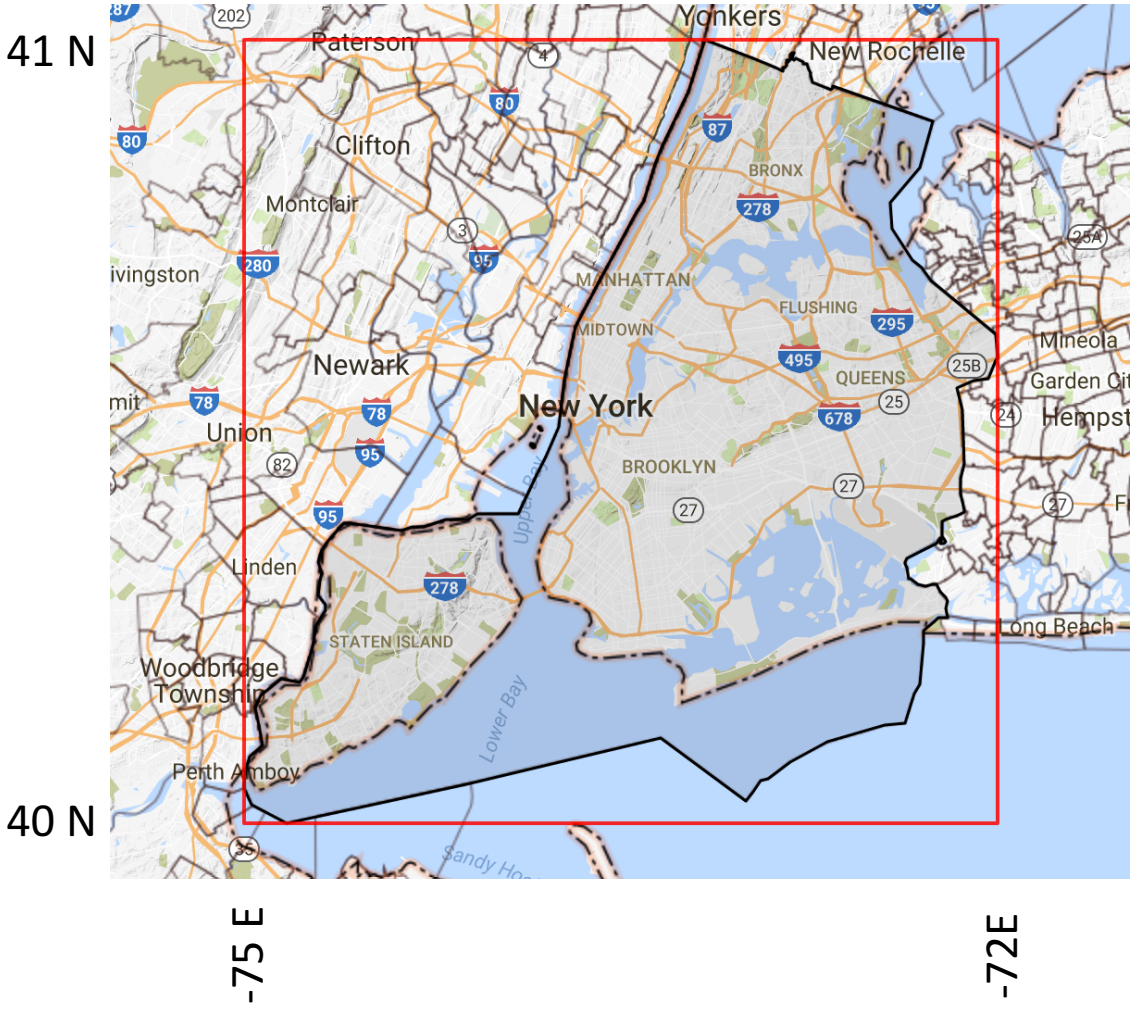
Select rows based on multiple conditions using std logic

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

df <- filter(sales,
                  total_price >= 100.00 ||
                  store_name = "Grand Rapids")
```

dplyr::filter()

Let's take a look at a lat/long values



dplyr::filter()

Let's take a look at a lat/long values

```
summary(nyc_taxi[ , grep('long|lat', names(nyc_taxi), value = TRUE)])
```

```
 pickup_longitude  pickup_latitude dropoff_longitude dropoff_latitude
 Min.   :-121.93   Min.   : 0.00   Min.   :-121.93   Min.   : 0.00
 1st Qu.: -73.99   1st Qu.:40.74   1st Qu.: -73.99   1st Qu.:40.73
 Median : -73.98   Median :40.75   Median : -73.98   Median :40.75
 Mean   : -72.93   Mean   :40.18   Mean   : -73.00   Mean   :40.21
 3rd Qu.: -73.97   3rd Qu.:40.77   3rd Qu.: -73.96   3rd Qu.:40.77
 Max.   :   0.00   Max.   :48.51   Max.   :   0.00   Max.   :44.62
```

Your turn – get rid of bad rows with filter()

1. Ensure the `dplyr` packages in installed and loaded
2. Using `dplyr`, filter the `nyc_taxi` data.frame to only include rows that fall within our geofence boundaries of (40N, -75E) and (41N, -72E)

   *`Remember that our data includes pickup and dropoff locations`

10:00

Your turn – uh-oh we've found more bad rows

1. Good job filtering out bad locations. But we've found some bad values too. `total_amount` should be a positive number

2. Modify your filter to ensure that only rows with positive `total_amount` are included

5:00

*dplyr::*_join()*

Joins are use to merge two datasets together and combine their columns. Join will match on common variable names

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

df1 <- left_join(df1, df2)
```

*dplyr::*_join()*

Control which variables are used for matching

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

df1 <- left_join(df1, df2, by = c("a", "b"))
```

*dplyr::*_join()*

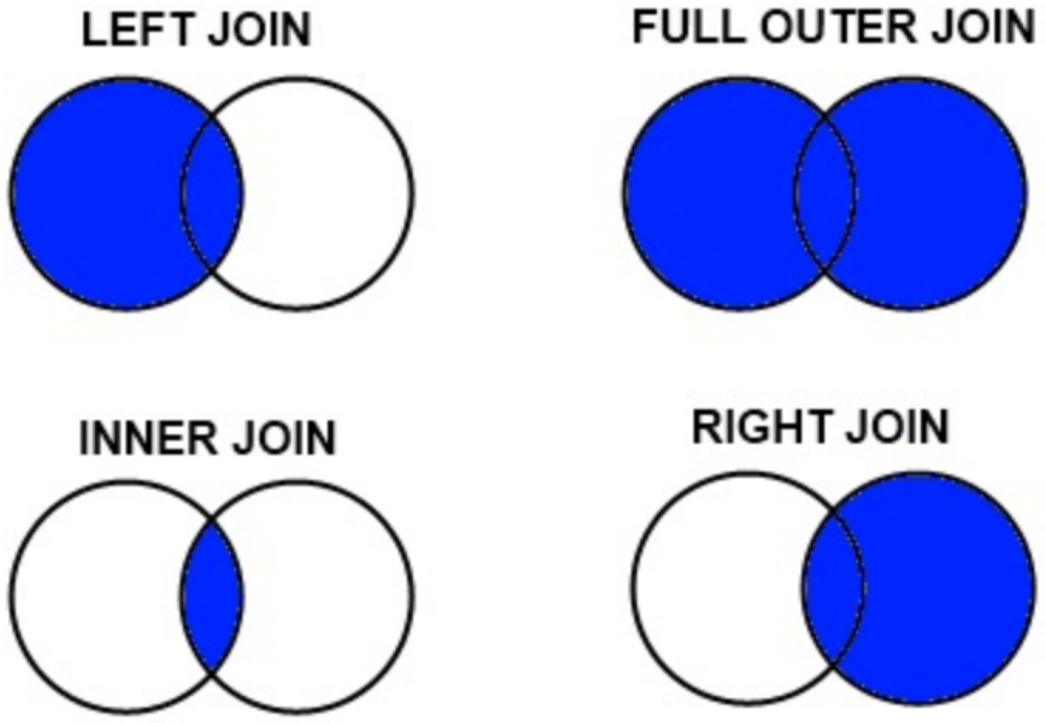
Specify a non-matching variable name

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

df1 <- left_join(df1, df2, by = c("a" = "b"))
```

*dplyr::*_join()*

dplyr supports the following join types



*dplyr::*_join()*

dplyr supports the following join types

`semi_join`: return all rows from tbl1 where matched in tbl2. Only return columns from tbl1

`anti_join`: return all rows from tbl1 where NOT matched in tbl2. Only return columns from tbl1

dplyr::mutate()

Mutate adds a new column to a data.frame using an expression

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

sales <- mutate(sales,
                20percdisc = sale_amt * .8)
```

dplyr::mutate()

You can also use logic to create a new value

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

sales <- mutate(sales,
                freq_shopper =
                if_else(numVists > 10, TRUE, FALSE)
              )
```

*dplyr::mutate() and dplyr::*_join()*

Often a new key needs to be created for a join

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

sales <- mutate(sales,
                datekey = year(saleDate))
            )
sales <- join (sales, storeHours, by(c("datekey" = "date"))
```

Your turn – join nyc_taxi to holidays

1. Use mutate to create a integer based date based on pickup_time (hint load the lubridate package first).

```
year() * 10000 + month() * 100 + day()
```

2. left_join() to the holidays data.frame using the new column matched with the date column of holidays

10:00

Your turn – join nyc_taxi to weather

1. Now join nyc_taxi to the weather table using the pickup_time column.

2. Do you need to mutate again, or can you do it by building on your previous work?

3. Check to see if there are any columns that you can get rid of.

10:00

Engineering features with dplyr

Two functions for engineering features

`mutate` when you want to add a new column

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

sales <- mutate(sales,
                datekey = year(saleDate),
                monthKey = datekey / 100)
            )
```

Engineering features with dplyr

Two functions for engineering features

`transmute` when you want to replace all columns

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

sales <- transmute(sales,
                   discPerc = salePrice / listPrice)
              )
```

Your turn – engineer some features

1. Use your dplyr skills to engineer the following new features.

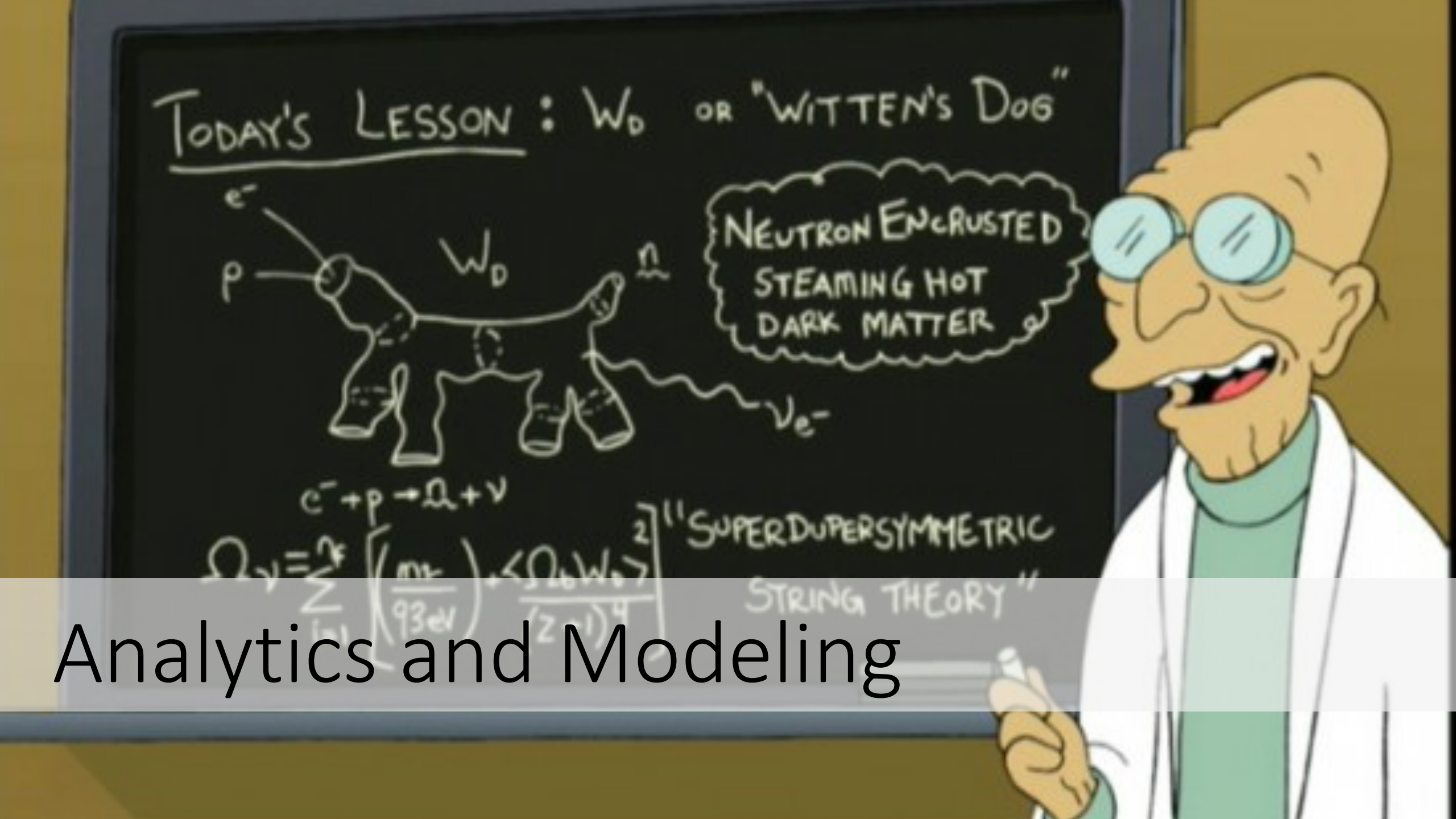
Feature Name	Rule
Pickup/dropoff hour	The hour of pickup and dropoff. Hint: use lubridate.
Pickup/dropoff day	The day of pickup and dropoff. Hint: use lubridate.
isWetDay	Precipitation was more then .05"
isHoliday	The day was a holiday
tripLength – use integer value	Trips less than .5 mile are short (1), up to 1.5 miles are medium (2), and over 1.5 miles are long (3)
custTipped	Indicates if the customer tipped or not
tipPercent	The tip_amount divided by the fare_amount
isBigTip	The tipPercent is larger than 20%

Final piece: adding geospatial features

Just follow along with this section and run the code

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

sales <- transmute(sales,
                   discPerc = salePrice / listPrice)
           )
```



Analytics and Modeling

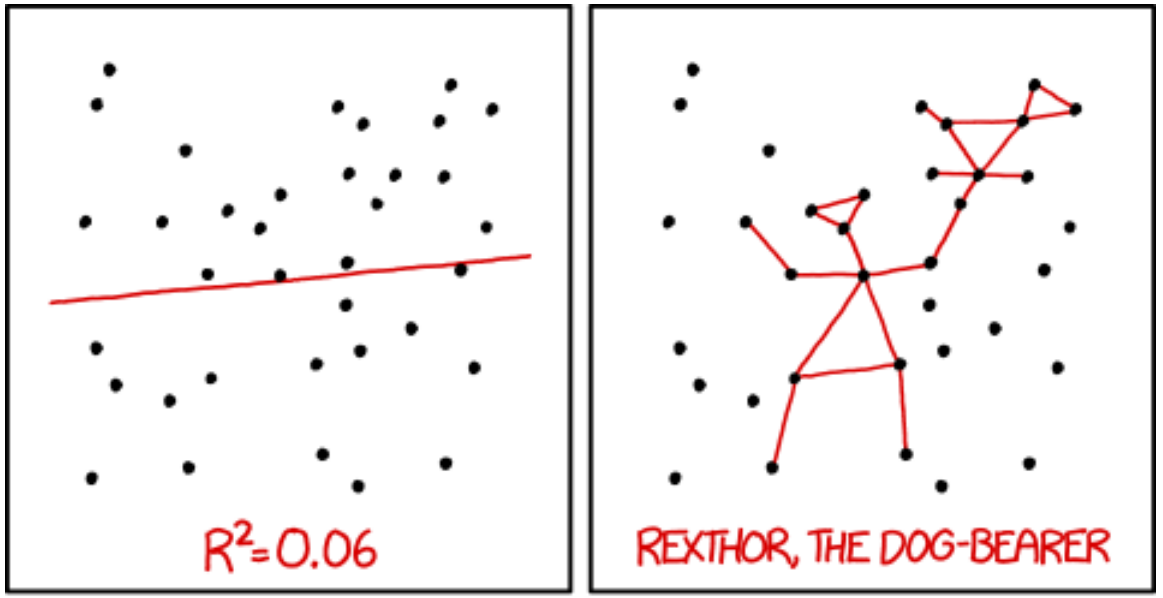
Analytic models

Analytic models are mathematical functions that are used to:

- Predict values
- Classify variables
- Determine probabilities

Prediction

Regression analysis is a common method of predicting an outcome

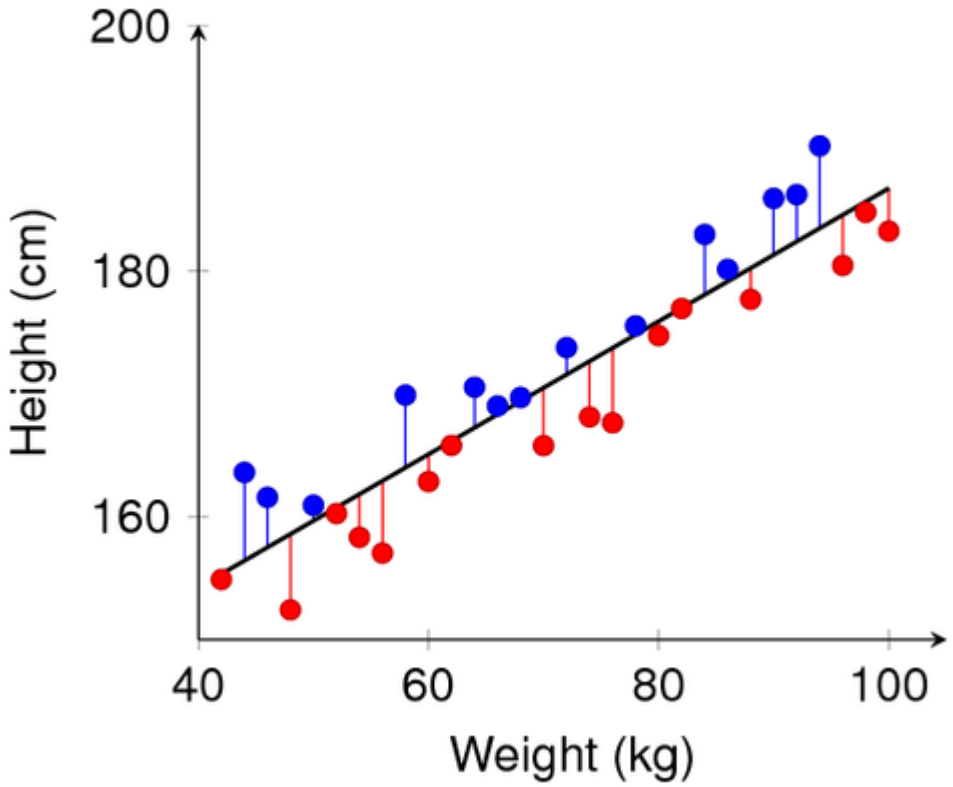


Linear regression is good for predicting a continuous variable. Like an amount, or amount of something consumed.

https://imgs.xkcd.com/comics/linear_regression.png

Prediction

Linear regression: slope, intercept, and residuals.

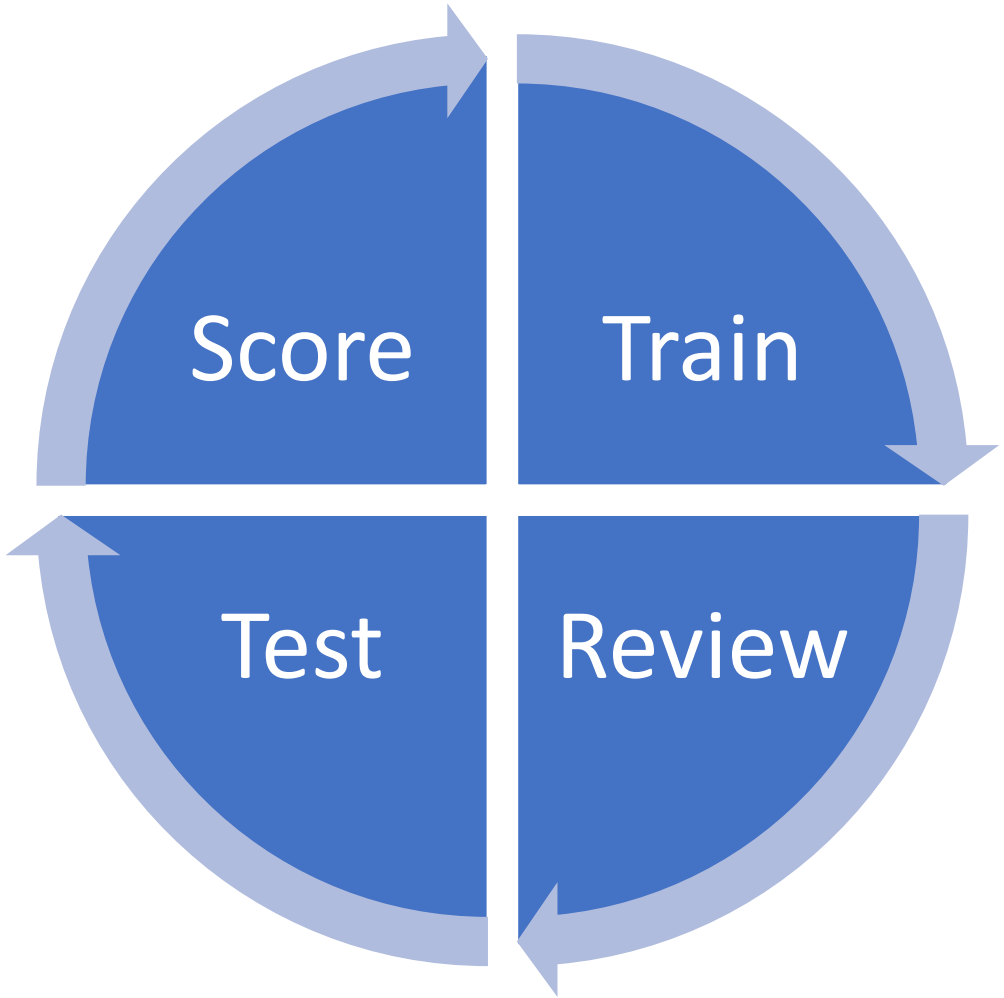


A line is fitted to the data points to attempt to correlate the dependent variable to the independent variable(s).

The difference between an observed value and the predicted value is called a residual.

Prediction

Supervised linear regression: The process.



1. ***Train*** the model using a sample of the dataset
2. ***Review*** the trained model's coefficients and values
3. ***Test*** the model against known observations
4. ***Score*** new data with the model

Linear Regression with R

Start by splitting `nyc_taxi` into training and testing datasets

`dplyr` makes this really easy with the `sample_frac()` function

```
train <- sample_frac(df, 0.8)

rowids <- as.numeric(rownames(train))
test <- df[-rowids,]
```

Linear Regression with R

The `lm()` function calculates a linear regression model

```
model <- lm(indep ~ dep1 + dep2 + dep3:dep4, data = df)
```

Linear Regression with R

Once the model is trained, you can review the diagnostics

```
print(model) #shows the model coefficients

summary(model) #returns summary statistics

str(model) #lists all info associated with the model
```

Linear Regression with R

Pay special attention to the model summary

```
Call:
lm(formula = mpg ~ disp + hp, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-4.7945 -2.3036 -0.8246  1.8582  6.9363

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 30.735904   1.331566  23.083  < 2e-16 ***
disp        -0.030346   0.007405  -4.098 0.000306 ***
hp          -0.024840   0.013385  -1.856 0.073679 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.127 on 29 degrees of freedom
Multiple R-squared:  0.7482,    Adjusted R-squared:  0.7309
F-statistic: 43.09 on 2 and 29 DF,  p-value: 2.062e-09
```

P-value < .05

Your turn – predict fare_amount

1. Write a linear regression function to predict `fare_amount`.

2. Use many of our independent variables in your formula. Remember that `dropoff_nhood` and `pickup_nhood` interact with each other. NOTE – using factors or character based variables will greatly increase the amount of time it takes to train your model.

3. Which variables are the appear to be significant? Retrain your model with only the significant variables.

15:00

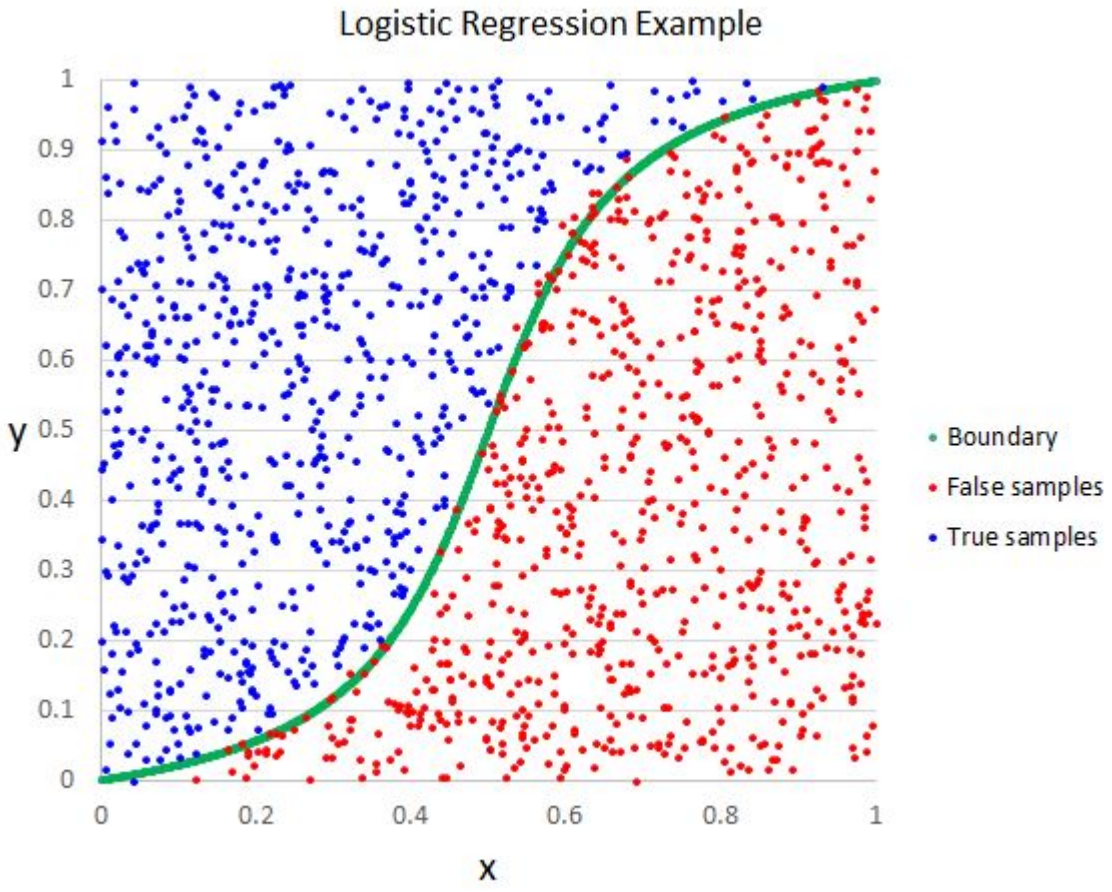
Linear Regression with R

After reviewing, it's a good idea to test your model

```
modelPred <- predict(model, test)
model <- cbind(test, modelPred)
```

Prediction

Logistic regression.



A curve is fit to the data to predict an outcome of a categorical variable.

Useful in determine if something happened or not.

Linear Regression with R

The `glm()` function calculates a linear regression model

```
model <- glm(indep ~ dep1 + dep2 + dep3:dep4, data = df)
summary(model)
```

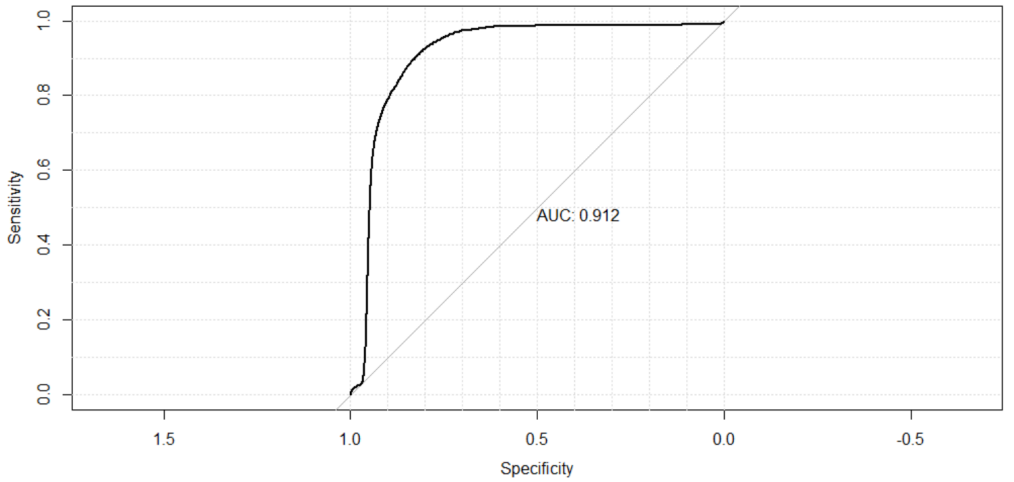
Linear Regression with R

In addition to p-values, generating a ROC curve is a good indicator of variable significance to a model

```
library(pROC)

roc(indep ~ dep,
    data = df,
    plot = TRUE)
```

Away from 45deg



Your turn – predict custTipped

1. Write a logistic regression function to predict if a customer will tip.

2. Use many of our independent variables in your formula. Remember that `dropoff_nhood` and `pickup_nhood` interact with each other. NOTE: using factor or character based variables will greatly increase the amount of time it takes to train your model.

3. Which variables are the appear to be significant? Retrain your model with only the significant variables.

4. Try to plot a ROC curve to see how significant your variables are.

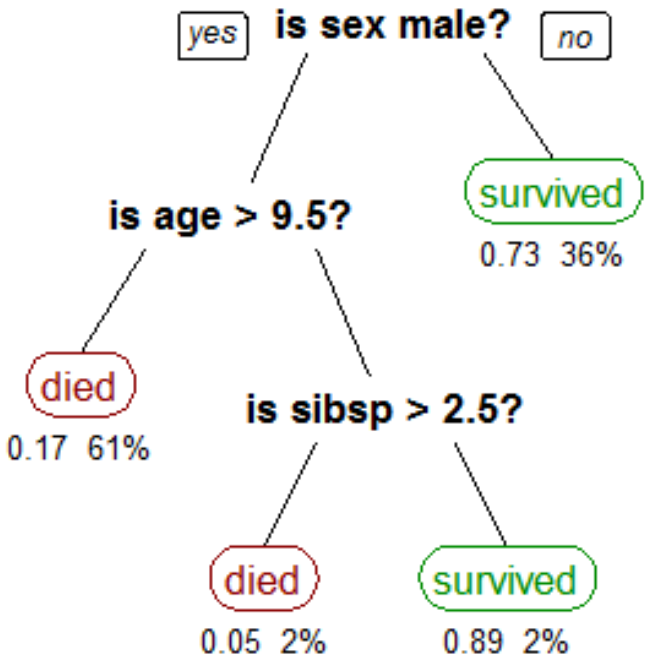
15:00

Prediction

Decision Tree

Decision Trees are a type of model that is use to show decision steps that are made to reach a binary outcome.

It displays the probabilities and weights of each decision point and is a great way to visualize a complex model.



Decision Trees with R

The rpart package is one way to create decision trees

```
install.packages("rpart")
library("rpart")
model <- rpart(indep ~ dep1 + dep2 + dep3:dep4,
               data = df)
summary(model)
```

Decision Trees with R

To plot an rpart tree, use the rpart.plot package

```
install.packages("rpart.plot")
library("rpart.plot")

rpart.plot(model)
```

Your turn – predict tip_amount

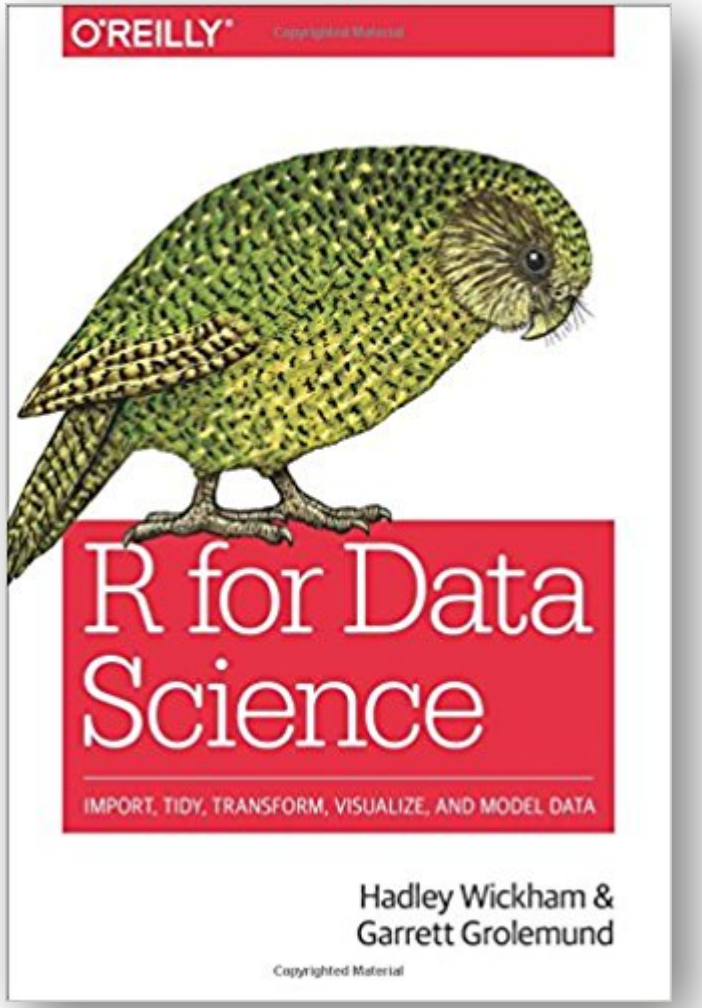
1. Build a decision tree model to predict how much a customer will tip.

2. Use many of our independent variables in your formula. Remember that `dropoff_nhood` and `pickup_nhood` interact with each other. Decisions trees work great with factor or character based variables. Often a better choice when you have many variables of these formats.

3. Which variables are the appear to be significant? Retrain your model with only the significant variables.

4. Plot your tree using `rpart.plot()`

15:00



R for Data Science

http://amzn.to/2sk8fNN

Introduction to R for Data Science

Learn the R statistical programming language, the lingua franca of data science in this hands-on course.

Microsoft

http://bit.ly/2sZd5g4

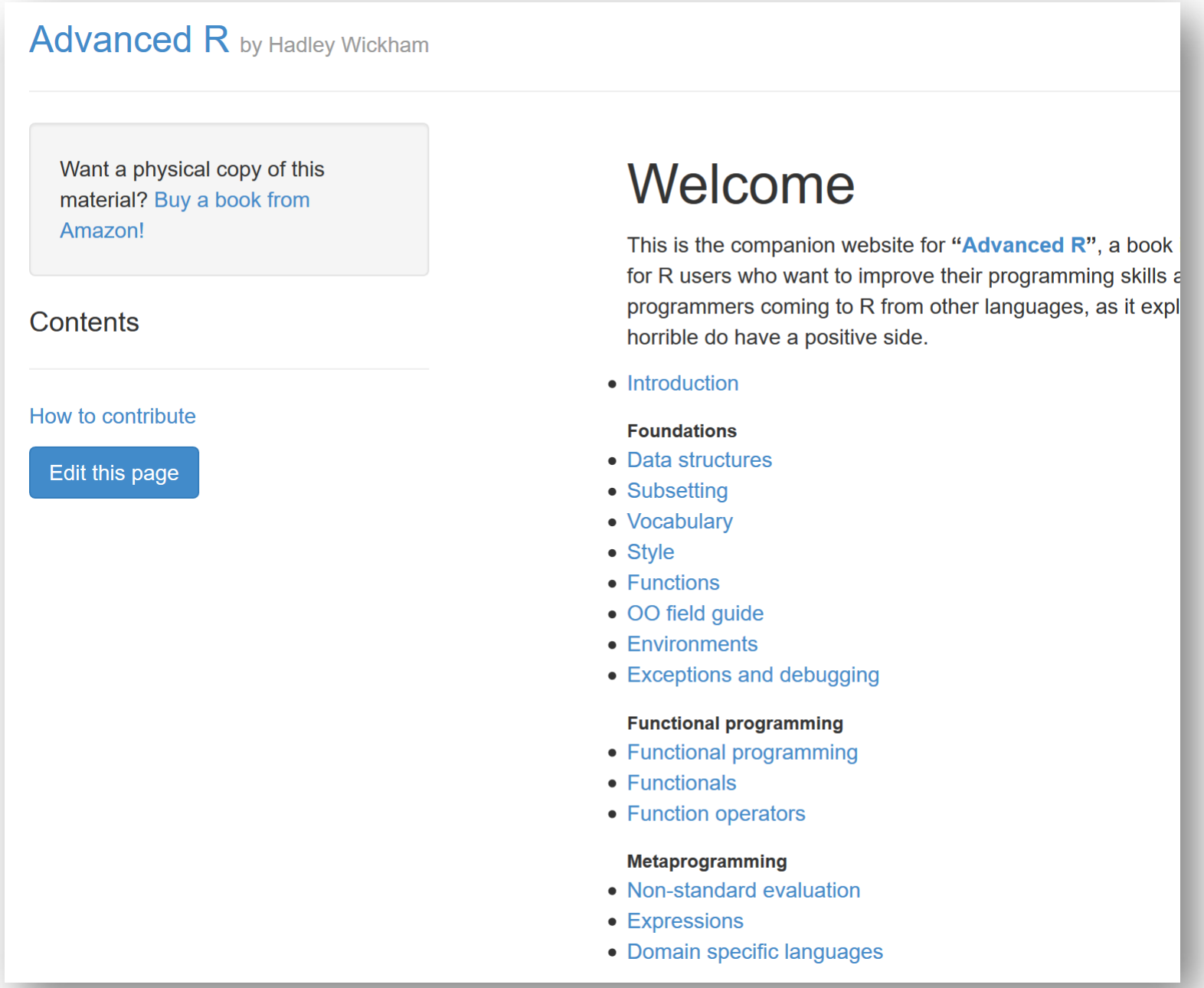
http://bit.ly/2sItbLD

Programming with R for Data Science

Learn the fundamentals of programming with R, from reading and writing data to customizing visualizations and performing predictive analysis.

Microsoft

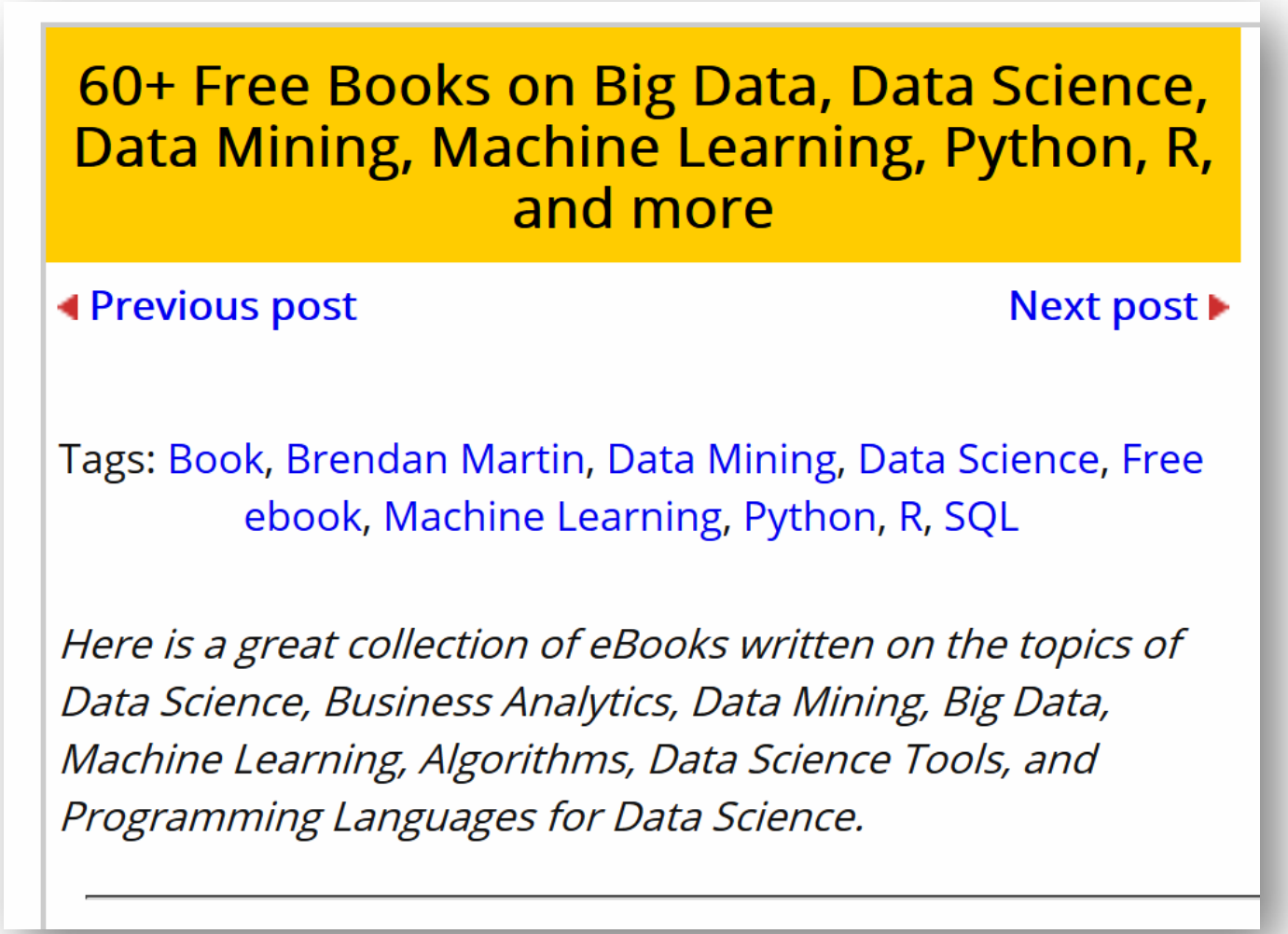
edX



Advanced R

By Hadley Wickham

http://adv-r.had.co.nz/



60+ Free Books on Big Data, Data Science, Data Mining, Machine Learning, Python, R, and more

◂ Previous post Next post ▸

Tags: Book, Brendan Martin, Data Mining, Data Science, Free ebook, Machine Learning, Python, R, SQL

Here is a great collection of eBooks written on the topics of Data Science, Business Analytics, Data Mining, Big Data, Machine Learning, Algorithms, Data Science Tools, and Programming Languages for Data Science.

Huge collection of free eBooks

http://bit.ly/2sYZYeD