Data Analysis in R

Orientation to R Studio

Michael E DeWitt Jr

2018-09-06 (updated: 2018-09-06)

Our mission

- Introduce R as a programming language and `<<data science>>` environment
- Introduce R Studio as an IDE
- Get you working on your use cases quickly

A little bit about R

- R started with S... 1976-05-05 at Bell Labs

A little bit about R

- R started with S... 1976-05-05 at Bell Labs

- Developed out of the statistics department

A little bit about R

- R started with S... 1976-05-05 at Bell Labs

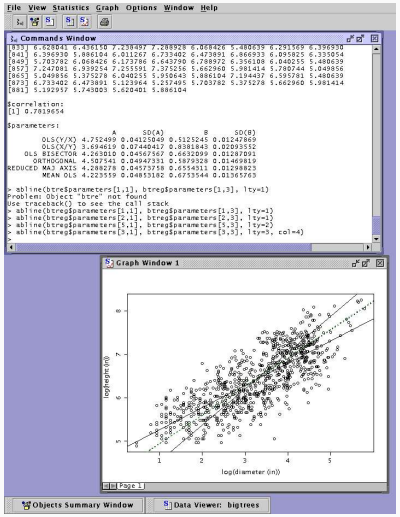
- Developed out of the statistics department
- Non-standard methods were needed

A little bit about R

- R started with S... 1976-05-05 at Bell Labs

- Developed out of the statistics department
- Non-standard methods were needed
- Solution: Develop programming language and environment designed explicitly for **interactive** data analysis using the best available computation methods and flexible enough to meet new use cases!

If you can do something well, then have



S was then sold to the open market as S

Later distributed as S-Plus

Bought by TIBCO

After S, R?

- Ross Ihaka and Robert Gentleman met at University of Auckland while Dr. Gentleman on sabbatical

After S, R?

- Ross Ihaka and Robert Gentleman met at University of Auckland while Dr. Gentleman on sabbatical

- Common Problem: `$A$`, `$-plus`, `$TATA`, `$P$$`

After S, R?

- Ross Ihaka and Robert Gentleman met at University of Auckland while Dr. Gentleman on sabbatical

- Common Problem: `$A$`, `$-plus`, `$TATA`, `$P$$`
- Solution: Develop an **open source** implementation of S!

After S, R?

- Ross Ihaka and Robert Gentleman met at University of Auckland while Dr. Gentleman on sabbatical

- Common Problem: `$A$`, `$-plus`, `$TATA`, `$P$$`
- Solution: Develop an **open source** implementation of S!
- Allow for new methods to be adopted and shared through **packages** *

[*] Packages are ways of collecting, documenting and sharing R (or other language) scripts that have been pre-written. They can be installed and then run as new functions extending base-R

CRAN Established

- Comprehensive R Archive Network Established cran

CRAN Established

- Comprehensive R Archive Network Established cran
- Hosting of the software itself and packages

CRAN Established

- Comprehensive R Archive Network Established cran
- Hosting of the software itself and packages
- R-Core established to review and approve packages for inclusion on CRAN

Why chose R?

- It's Free!!!

Why chose R?

- It's Free!!!
- It's flexible

Why chose R?

- It's Free!!!
- It's flexible
- Access to the newest methods

Why chose R?

- It's Free!!!
- It's flexible
- Access to the newest methods
- The community support is strong

Why chose R?

- It's Free!!!
- It's flexible
- Access to the newest methods
- The community support is strong
- It has been extended into webpage generation, documentation, dashboards, etc

Why not R?

Why not R?

- Syntax is *not* consistent between packages

```
# From randomForest
rf_1 <- randomForest(x, y, mtry = 12, ntree = 2000, importance = TRUE)

# From ranger
rf_2 <- ranger(
  y ~ .,
  data = dat,
  mtry = 12,
  num.trees = 2000,
  importance = 'impurity'
)
```

Why not R?

- Syntax is *not* consistent between packages

```
# From randomForest
rf_1 <- randomForest(x, y, mtry = 12, ntree = 2000, importance = TRUE)

# From ranger
rf_2 <- ranger(
  y ~ .,
  data = dat,
  mtry = 12,
  num.trees = 2000,
  importance = 'impurity'
)
```

- Packages have not necessarily been vetted and completely de-bugged

Why not R?

- Syntax is *not* consistent between packages

```
# From randomForest
rf_1 <- randomForest(x, y, mtry = 12, ntree = 2000, importance = TRUE)

# From ranger
rf_2 <- ranger(
  y ~ .,
  data = dat,
  mtry = 12,
  num.trees = 2000,
  importance = 'impurity'
)
```

- Packages have not necessarily been vetted and completely de-bugged
- R **can** be slower than some other languages

Why not R?

- Syntax is *not* consistent between packages

```
# From randomForest
rf_1 <- randomForest(x, y, mtry = 12, ntree = 2000, importance = TRUE)

# From ranger
rf_2 <- ranger(
  y ~ .,
  data = dat,
  mtry = 12,
  num.trees = 2000,
  importance = 'impurity'
)
```

- Packages have not necessarily been vetted and completely de-bugged
- R **can** be slower than some other languages
- Performs calculations *in-memory* (e.g. data < RAM)

R Studio

- R Studio, a for profit company, developed an IDE for R

R Studio

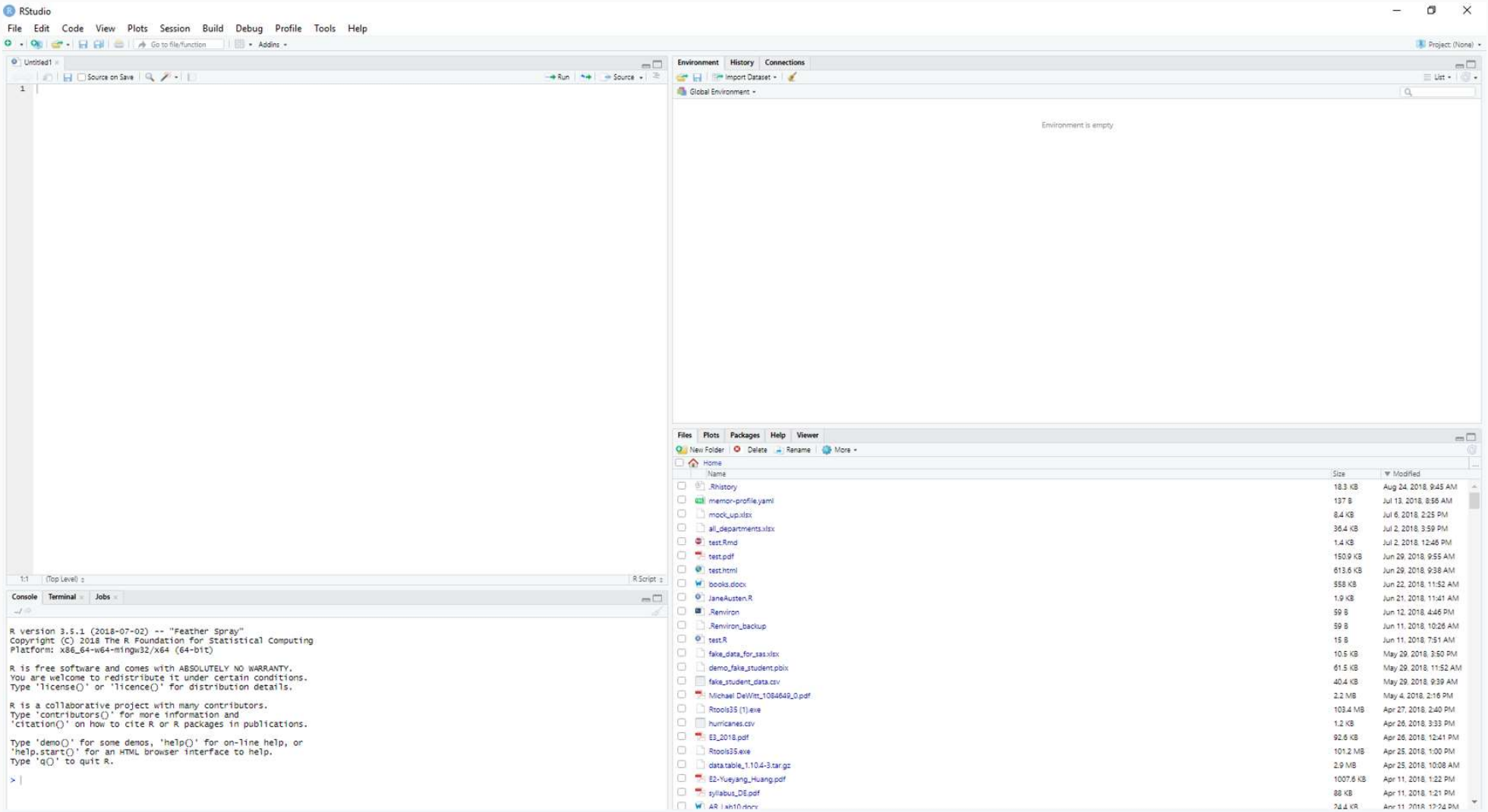
- R Studio, a for profit company, developed an IDE for R
- IDE is an Integrated Development Environment

R Studio

- R Studio, a for profit company, developed an IDE for R
- IDE is an Integrated Development Environment
- Syntax highlighting, auto-complete, visual exploration, integration with version control systems and more!

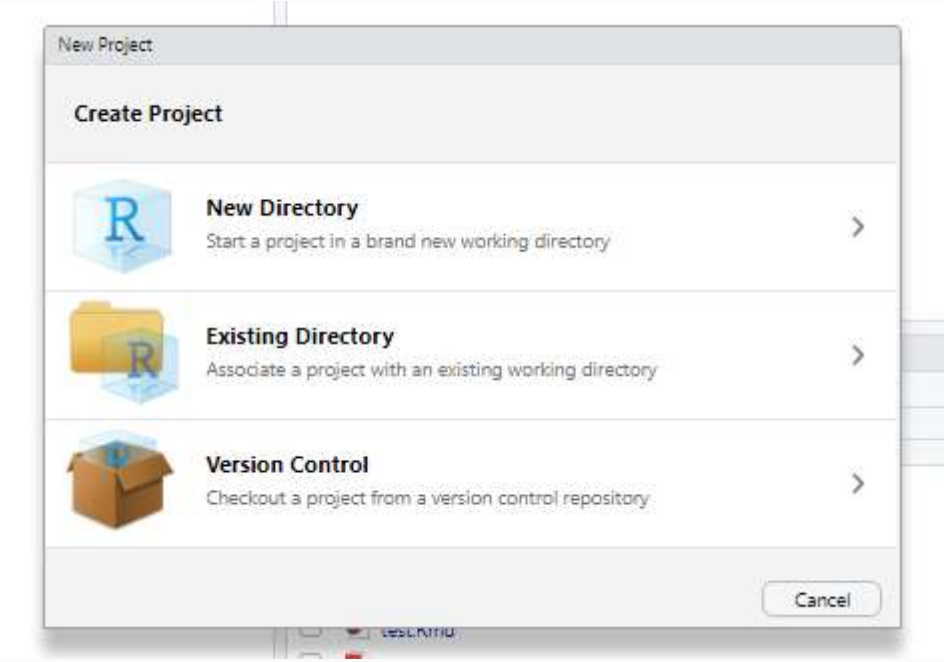
So Let's Explore it

Press CTRL + SHIFT + N or CMD + SHIFT + N to create a new script



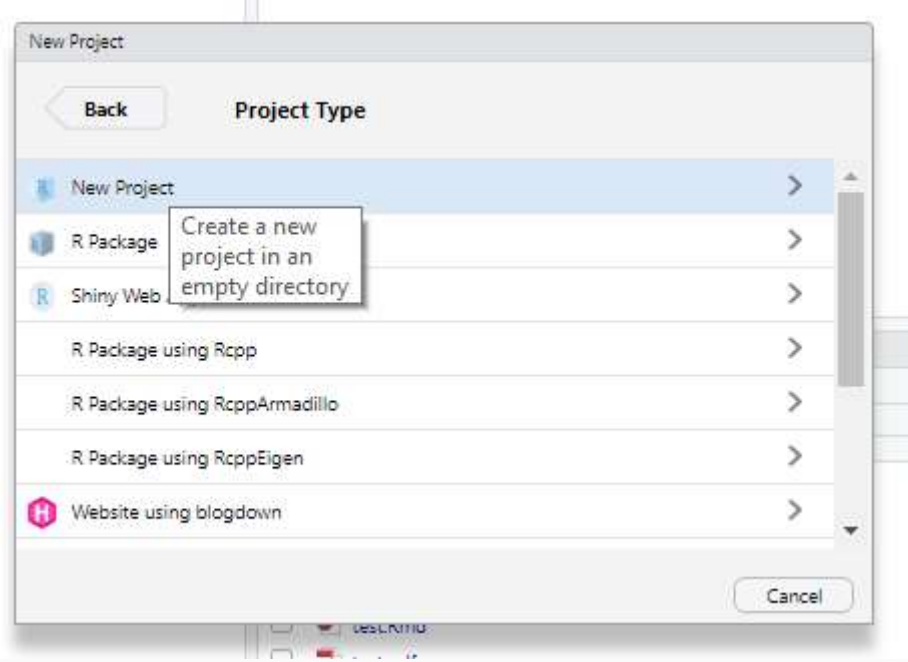
Start a new project

We want to start a new project



Put the project in a new directory (typically the default)

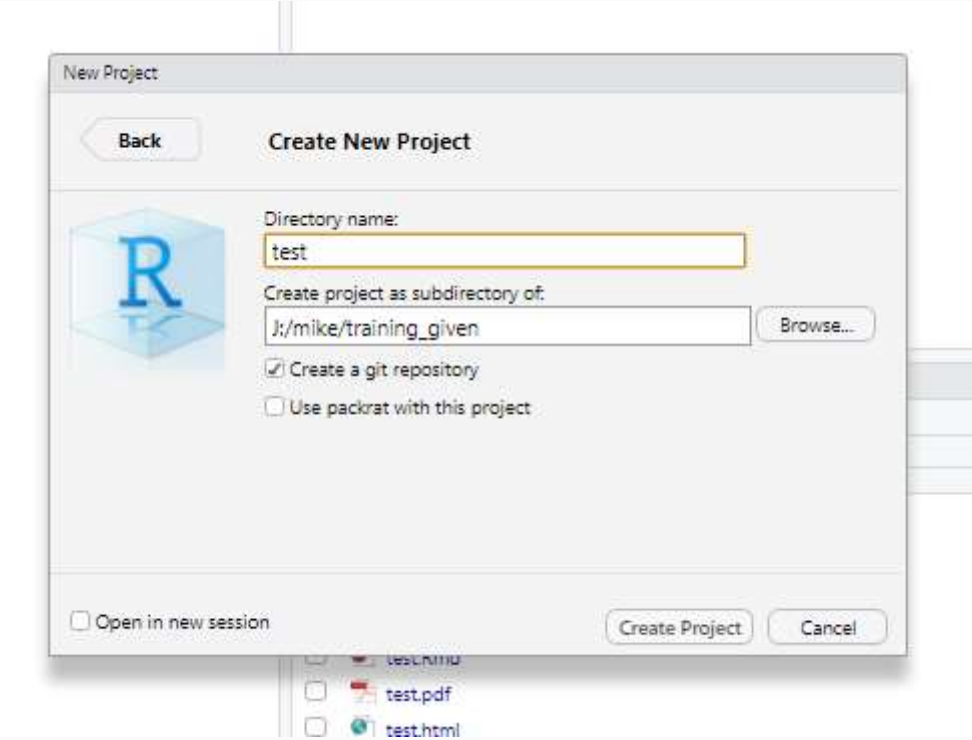
Create a new project



We would like a new project (other items are for more advanced topics)

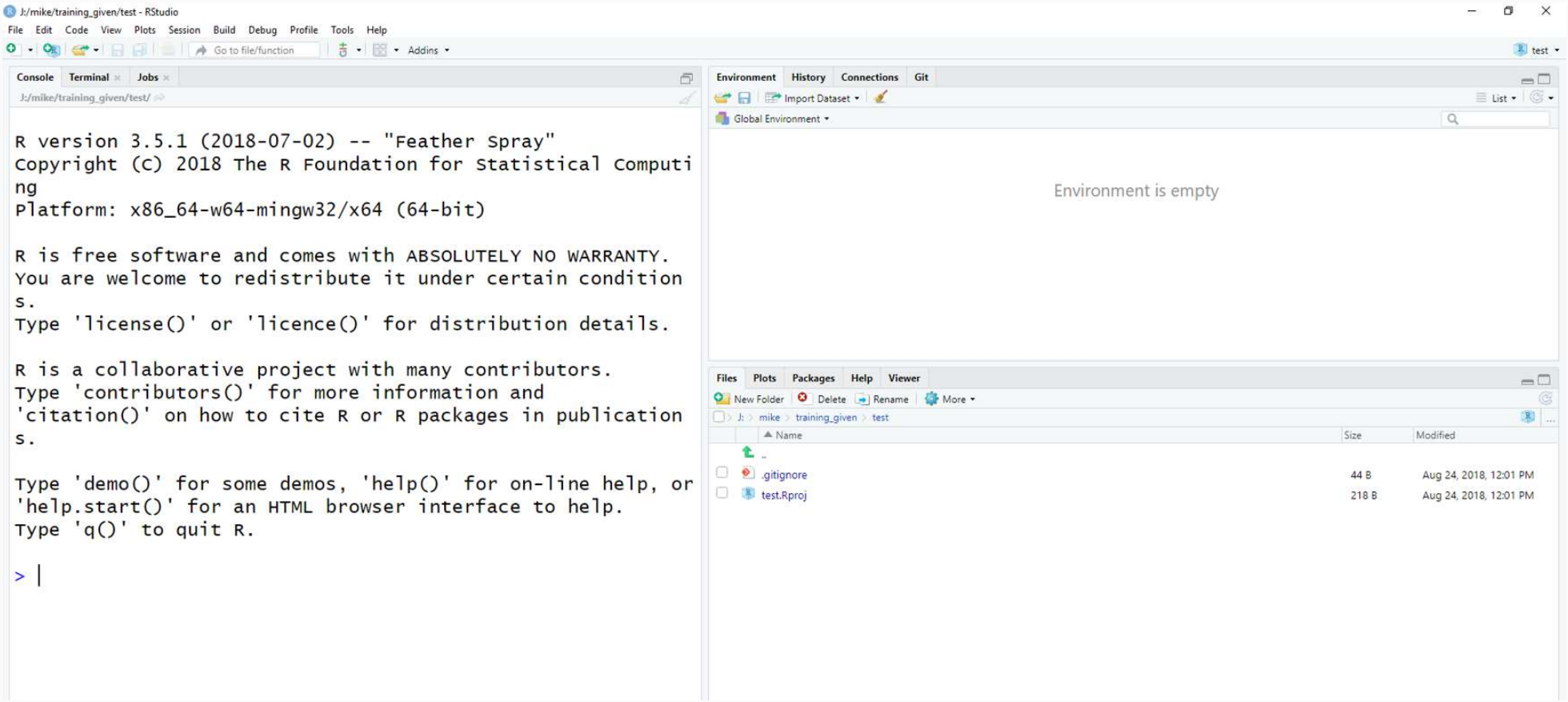
Name it

Name the new project and place it in whatever directory works for you



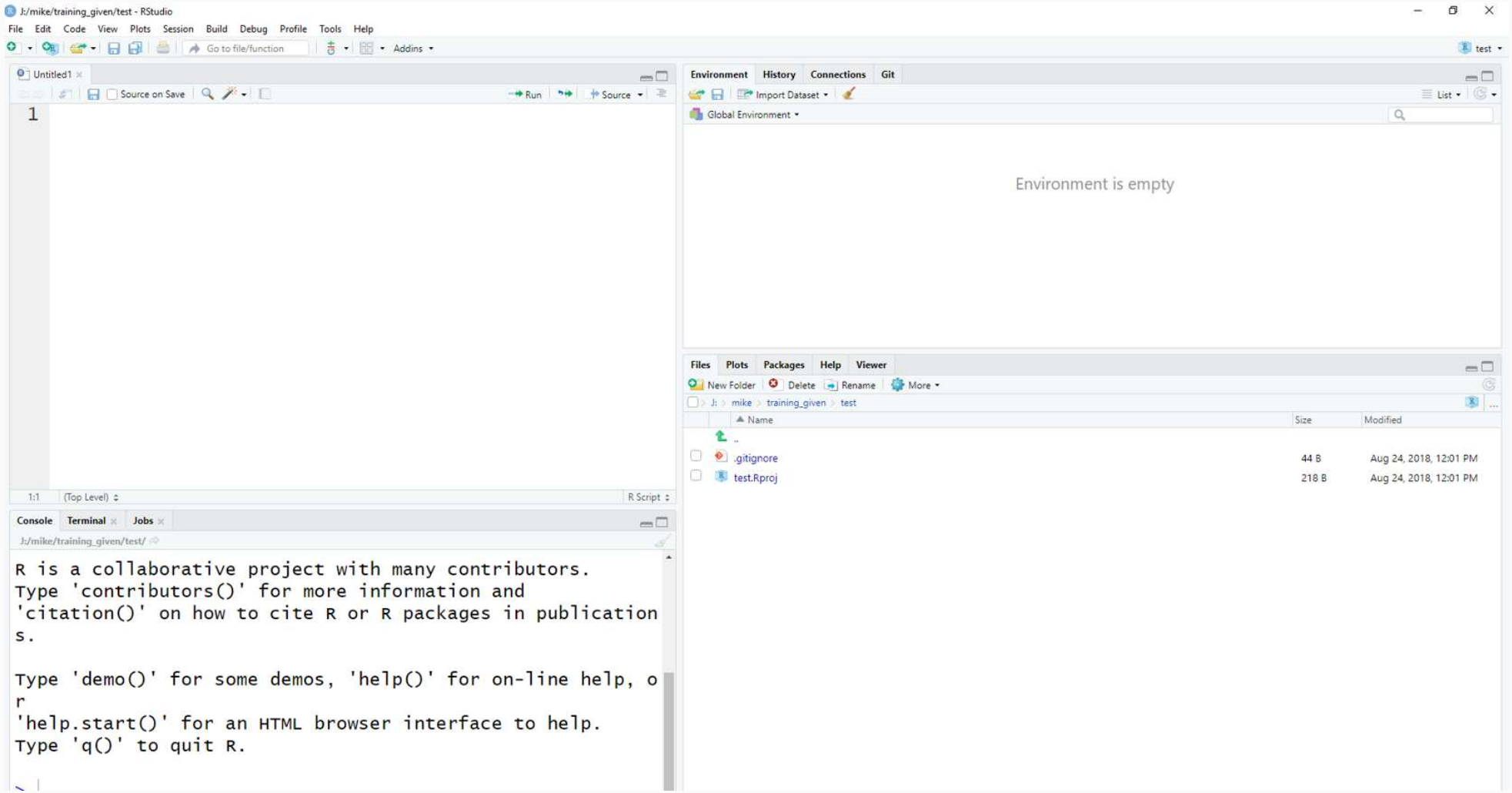
Ta Da!

This should be your starting point for each new project

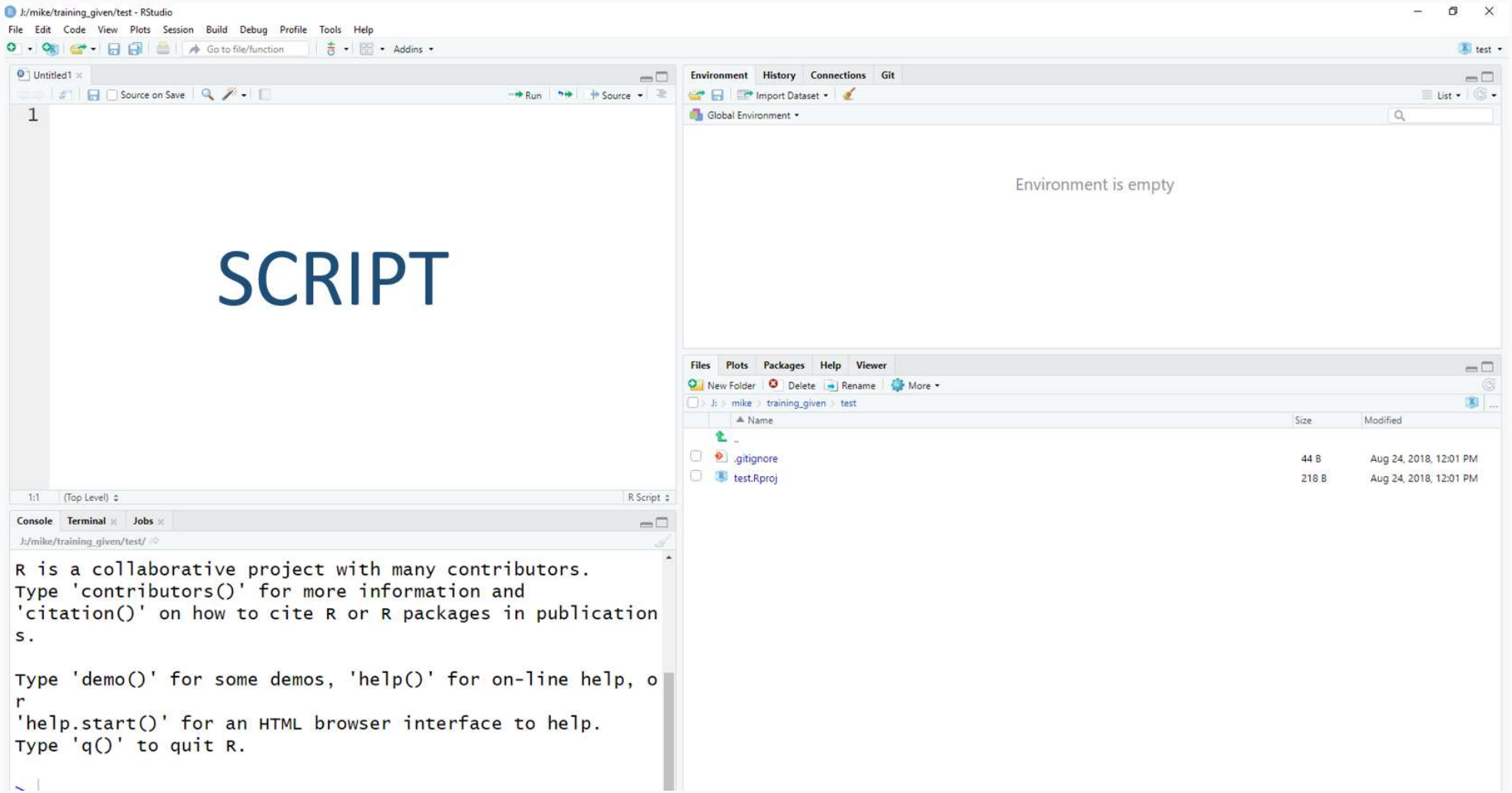


Now open a new scripting pane

`CRTL+N` or `CMD+N` to open a new script



Scripting Pane



Scripting Pane (Where the magic happens)

Generally you want to write all of your code here in either:

- Rmarkdown document (.Rmd)
- R script (.R)

This way you can save your code and submit it to R!

Scripting Pane (Where the magic happens)

Generally you want to write all of your code here in either:

- Rmarkdown document (.Rmd)
- R script (.R)

This way you can save your code and submit it to R!

(And save a copy to run again later)

Scripting Pane (Where the magic happens)

Generally you want to write all of your code here in either:

- Rmarkdown document (.Rmd)
- R script (.R)

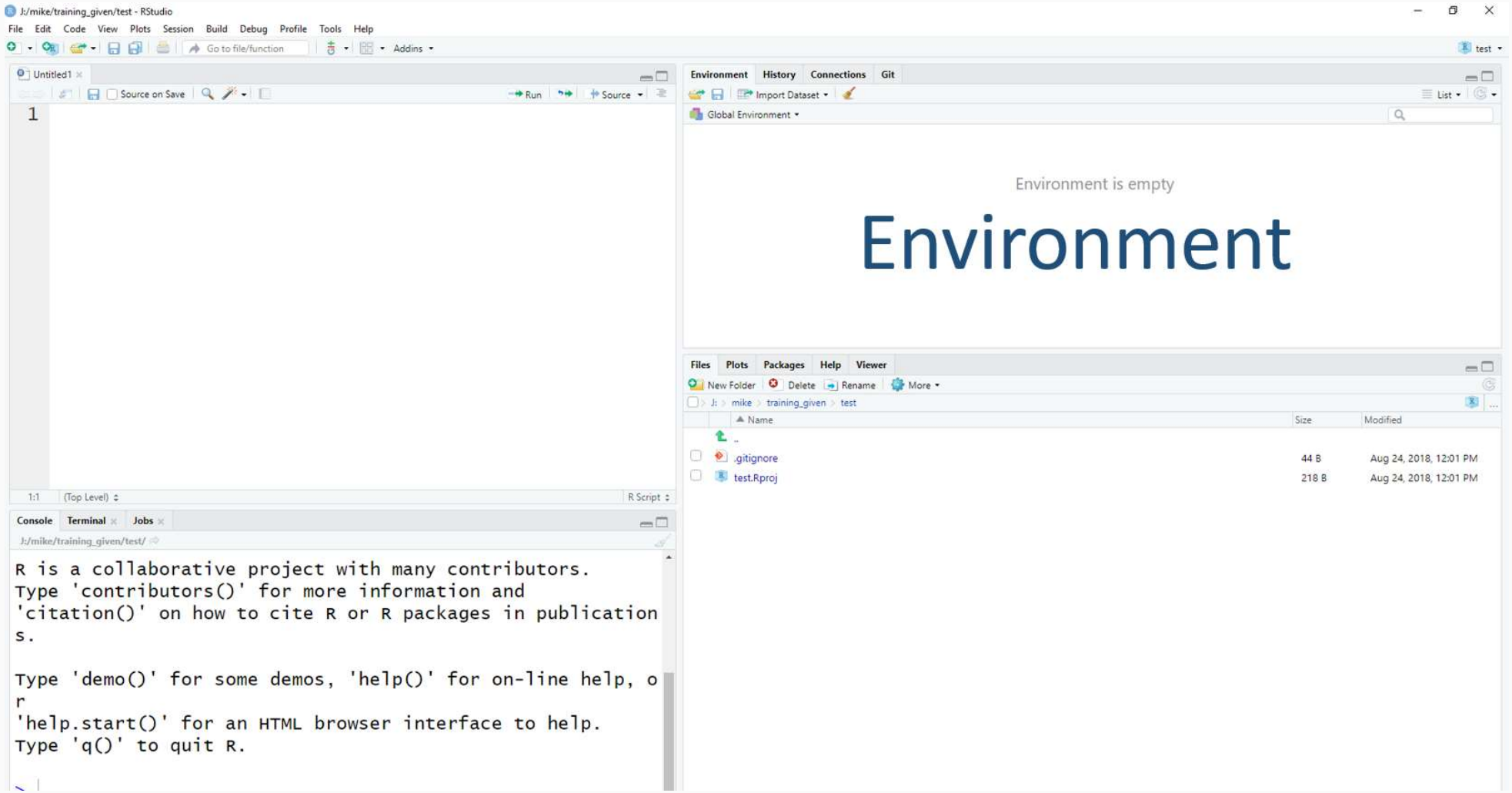
This way you can save your code and submit it to R!

(And save a copy to run again later)

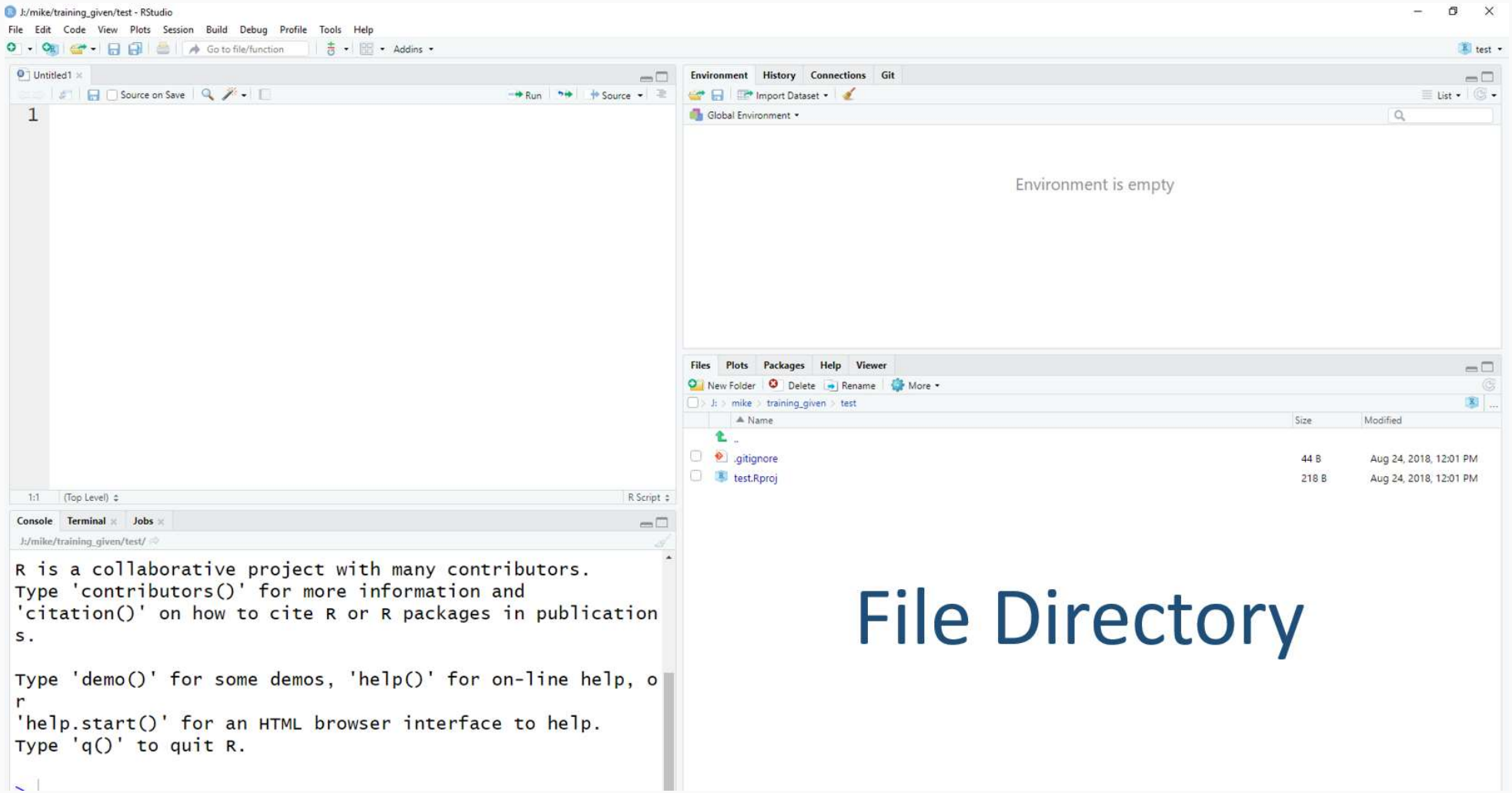
You can send the script to R using the `CRTL + ENTER` / `CMD + ENTER` Shortcut

Environment Pane

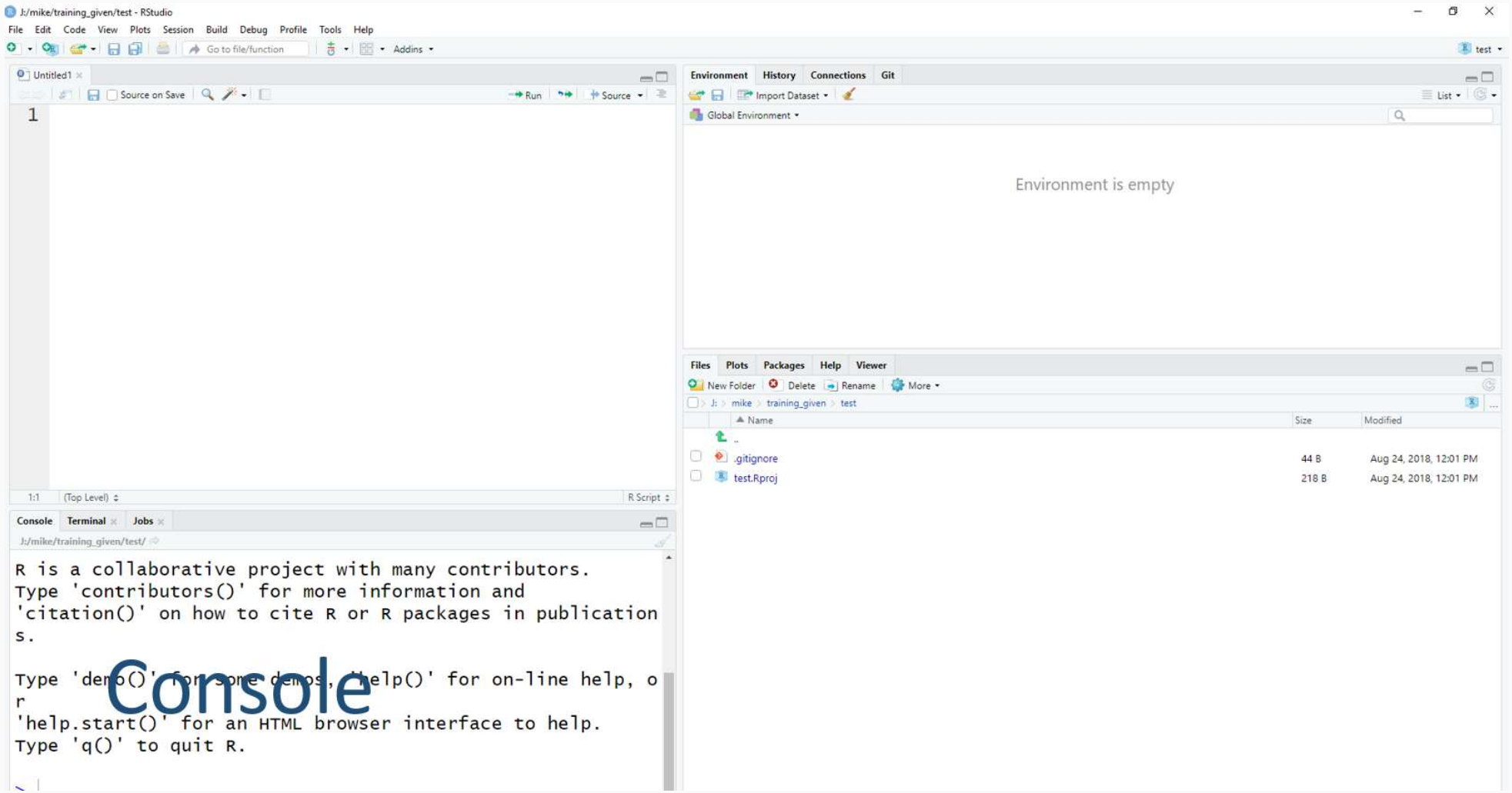
This is where you will find objects that are `in-memory` or available to call



File Directory Pane



Console Pane



Console

The console is the heart of R and the best calculator in the world

```
4+6
```

```
## [1] 10
```

If you type directly the output will be printed, but *not* saved

Console

The console is the heart of R and the best calculator in the world

```
4+6
```

```
## [1] 10
```

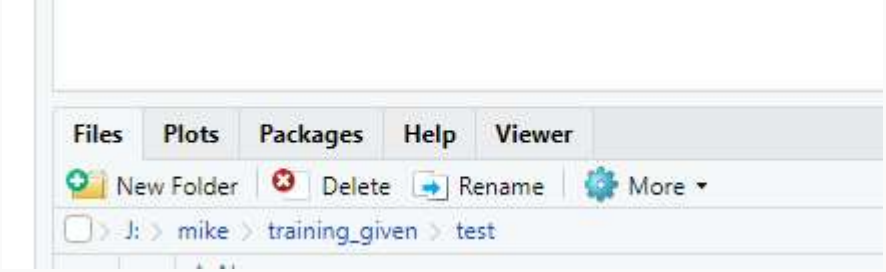
If you type directly the output will be printed, but *not* saved

To save you need to use the assignment operator `<-` to store the object in memory

```
my_output <- 4+6
my_output
```

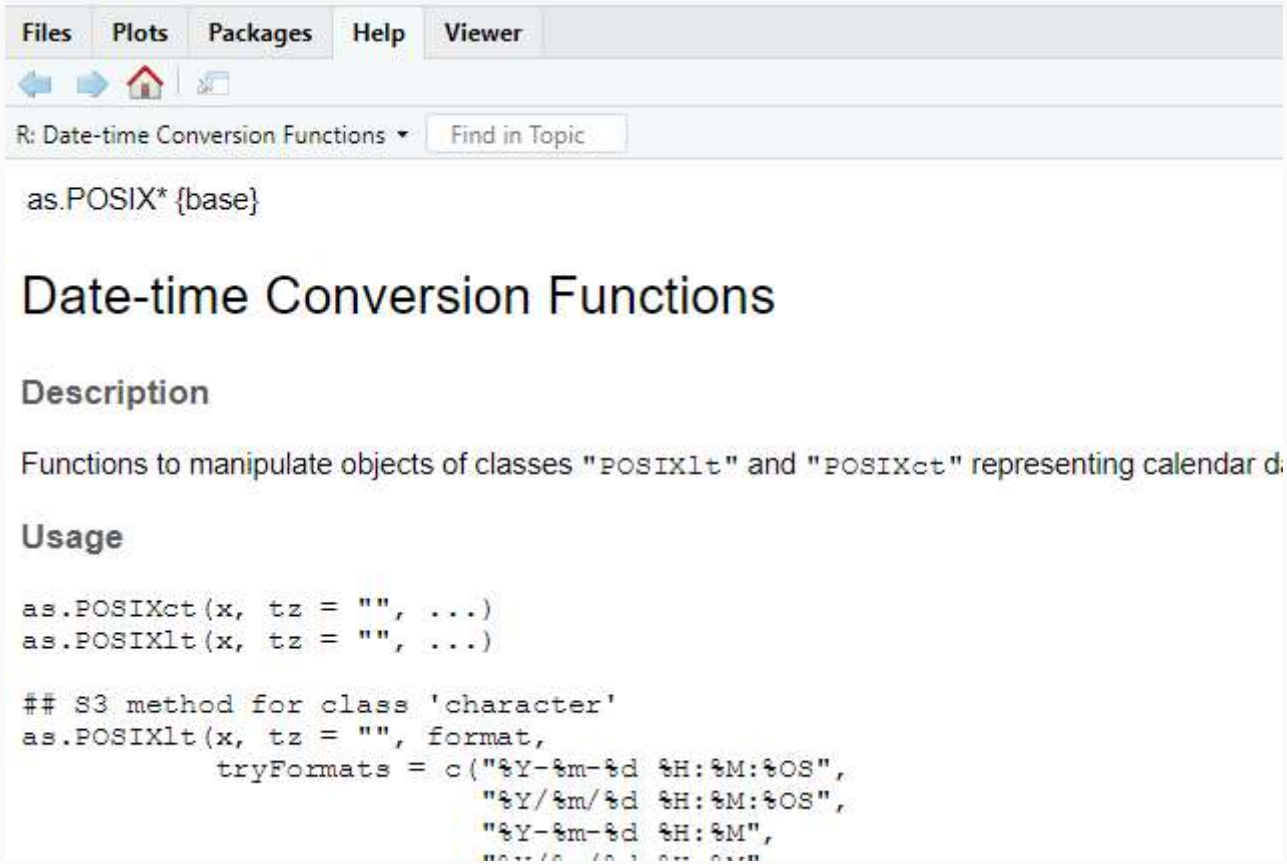
```
## [1] 10
```

Additional Tabs



Additional tabs contain other objects include files, plots, packages, and viewer

Help!



A handy feature is the searchable help pane where you can get information on functions

Additional Help

You can also use the console to help you *find help*

To call the help file for a particular function:

```
??lm
```

Additional Help

You can also use the console to help you *find help*

To call the help file for a particular function:

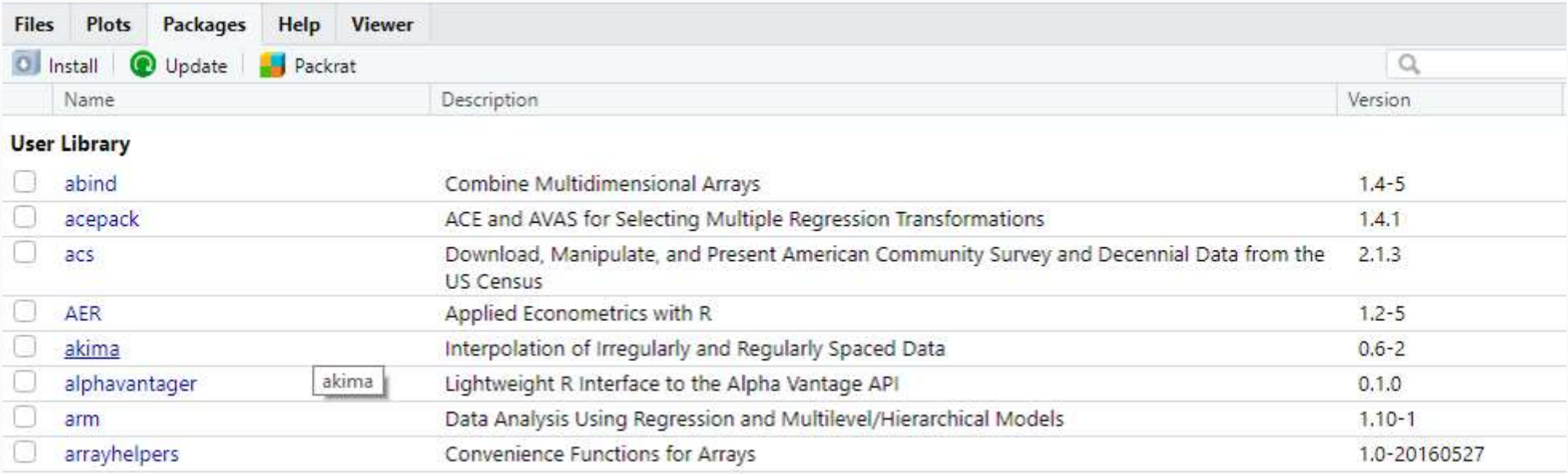
```
??lm
```

For a function that you can't quite remember...

```
apropos("glm")
```

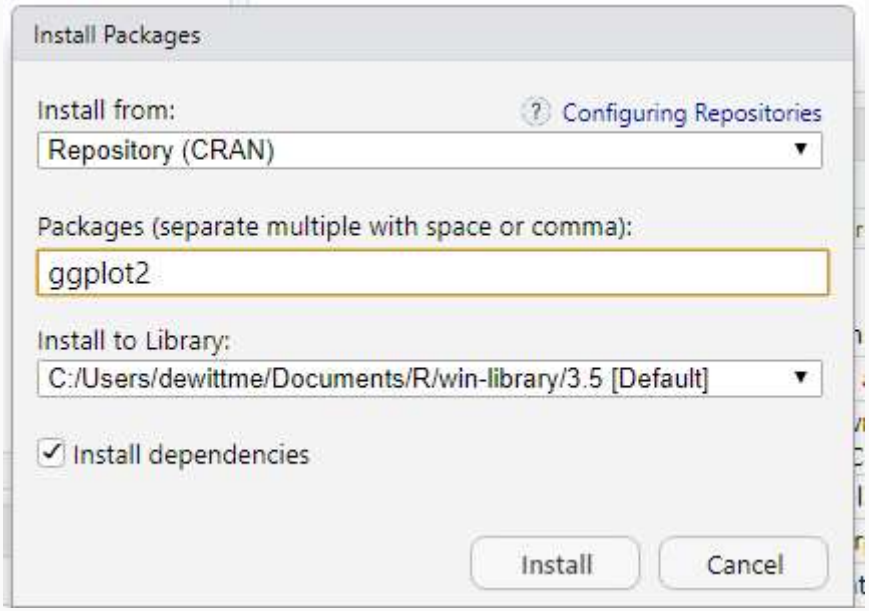
```
## [1] "glm"           "glm.control"   "glm.fit"       "predict.glm"
## [5] "residuals.glm" "summary.glm"
```

Package Viewer



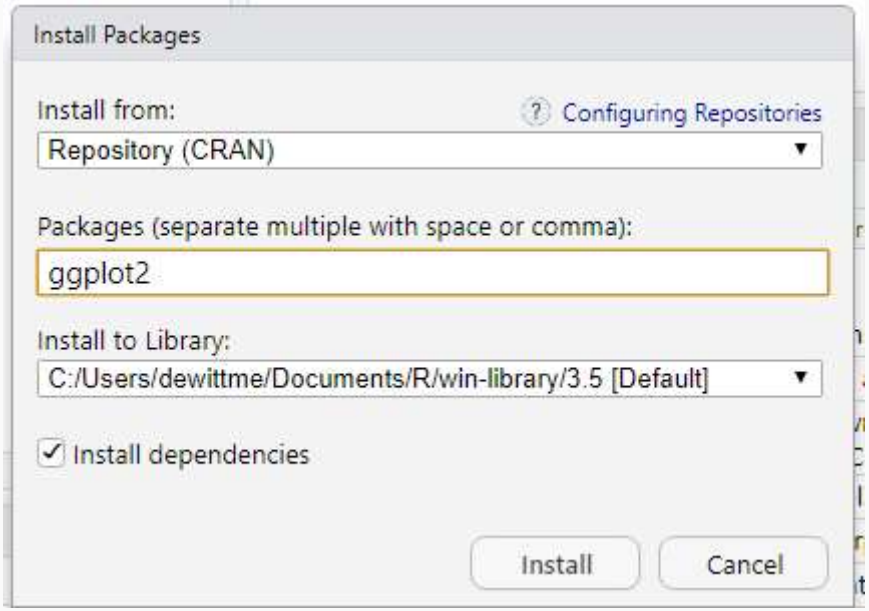
The package viewer allows you to see your install packages (with links to the function descriptions)

Package Install



You can install packages from `cran` using the package install feature

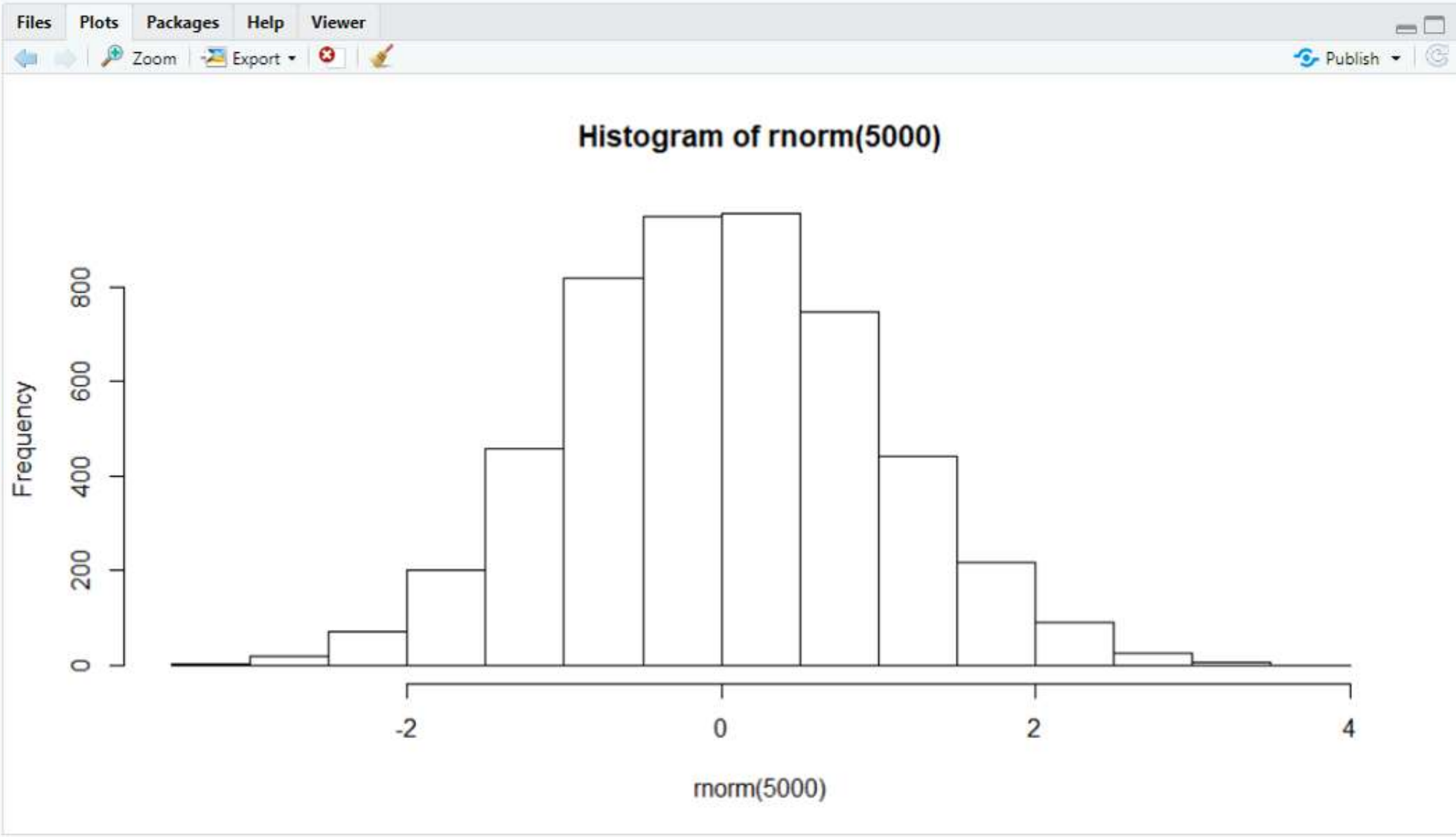
Package Install



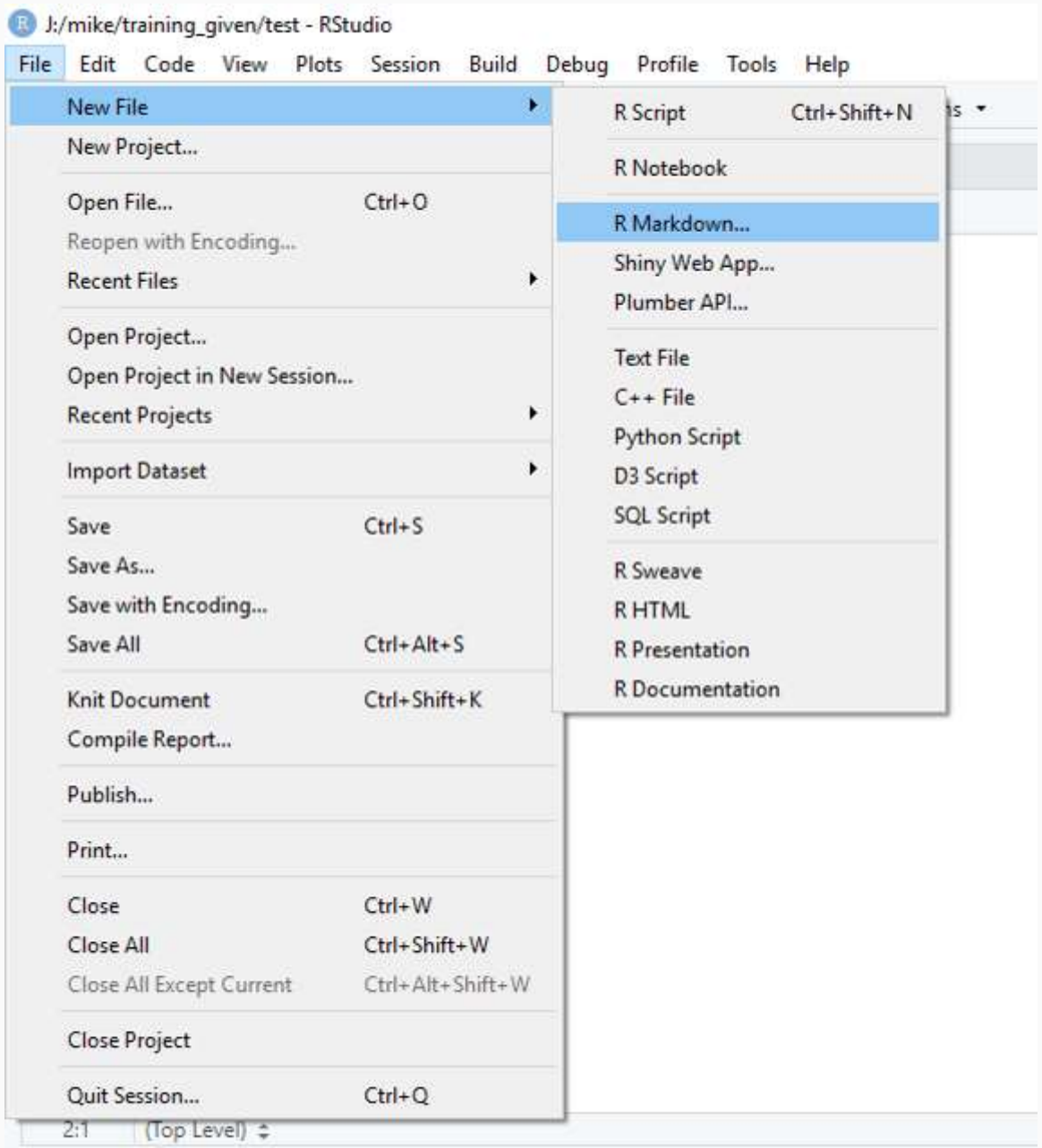
You can install packages from `cran` using the package install feature

Later we will install packages from other sources like github

Plots

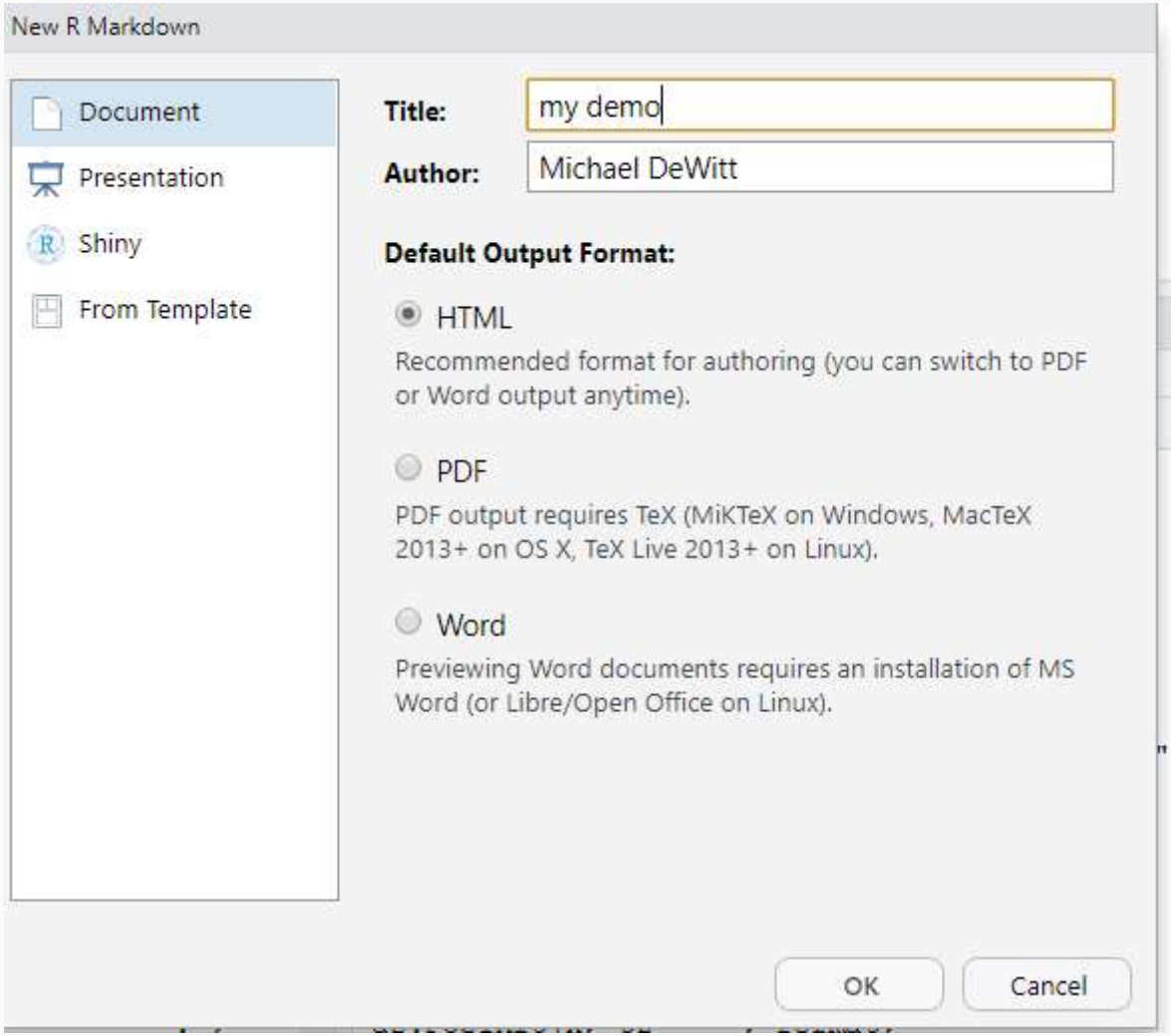

Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
Histogram of rnorm(5000)
Frequency
0
200
400
600
800
-2
0
2
4
rnorm(5000)

Creating a new file



New Rmarkdown

Markdown documents are a way to weave text and code into the same documents



Rmarkdown Example File

Untitled1* Untitled2

Knit Insert Run

```
---
title: "my demo"
author: "Michael DeWitt"
date: "8/24/2018"
output: html_document
---

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

R Markdown

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

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

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

Parts of an Rmarkdown Document - YAML

The YAML header block is separated by three _ symbols

```
---
title: "Untitled"
author: "Michael DeWitt"
date: "9/5/2018"
output: html_document
---
```

Parts of an Rmarkdown Document - YAML

The YAML header block is separated by three `_` symbols

```
---
title: "Untitled"
author: "Michael DeWitt"
date: "9/5/2018"
output: html_document
---
```

Specifies to `pandoc` how to convert the document and into what format to render the document

- html_output = html
- pdf_output = pdf
- word_output = Microsoft Word

Parts of an Rmarkdown Document - Code

Code chunks can be inserted with `CTRL + ALT + I` or `CMD + ALT + I`

```
fit <- lm(mpg ~ disp + wt, data = mtcars)
```

Parts of an Rmarkdown Document - Code

Code chunks can be inserted with `CTRL + ALT + I` or `CMD + ALT + I`

```
fit <- lm(mpg ~ disp + wt, data = mtcars)
```

Here you can type R code, and determine if the output will be printed in the document

Parts of an Rmarkdown Document - Code

Code chunks can be inserted with `CTRL + ALT + I` or `CMD + ALT + I`

```
fit <- lm(mpg ~ disp + wt, data = mtcars)
```

Here you can type R code, and determine if the output will be printed in the document

You can run each code chunk by pressing the green arrow in the chunk

Parts of an Rmarkdown Document - Code

Code chunks can be inserted with `CTRL + ALT + I` or `CMD + ALT + I`

```
fit <- lm(mpg ~ disp + wt, data = mtcars)
```

Here you can type R code, and determine if the output will be printed in the document

You can run each code chunk by pressing the green arrow in the chunk

Or running the code line by line

Parts of an Rmarkdown Document - Plain Text

The rest of the Rmarkdown is plain text

Parts of an Rmarkdown Document - Plain Text

The rest of the Rmarkdown is plain text

This is where you can write down ideas, your opinions, your findings

Parts of an Rmarkdown Document - Plain Text

The rest of the Rmarkdown is plain text

This is where you can write down ideas, your opinions, your findings

You can use this for both exploration (think lab notebook) and final documents

Parts of an Rmarkdown Document - Plain Text

The rest of the Rmarkdown is plain text

This is where you can write down ideas, your opinions, your findings

You can use this for both exploration (think lab notebook) and final documents

Rmarkdown also provides ways to add some additional formatting

Rmarkdown Markup

Formatting

`# # Header 1`

`## ## Header 2`

`*italics*` or `_italics_` for *italics*

`**bold**` or `__bold__` for **bold**

`-` or `*` for bullets

- Bullet 1
- Bullet 2
  - Bullet 2a

`1` for numbered lists

1 Item 1

2 Item 2

`$\int_{a}^bx^2dx$` for $\LaTeX$ e.g. $\int_a^b x^2 dx$

Knit it!

Rmarkdown to html!

Check out the gallery at https://rmarkdown.rstudio.com/gallery.html

Some final words about the course

Some final words about the course

The emphasis is on *application* and not on *theory*

Some final words about the course

The emphasis is on *application* and not on *theory*

I will teach using the `tidyverse` paradigm

Some final words about the course

The emphasis is on *application* and not on *theory*

I will teach using the `tidyverse` paradigm

We will learn data structures as we go along (not a programming class)

Some final words about the course

The emphasis is on *application* and not on *theory*

I will teach using the `tidyverse` paradigm

We will learn data structures as we go along (not a programming class)

My way isn't always the best way (always alternatives)