Introduction to RStudio

Yuwu Chen

HPC @ LSU

Outline

- RStudio basics
  - R in PC and HPC
  - What is RStudio
  - RStudio IDE features
  - User environment
- Advanced features
  - Use Version Control with RStudio
  - Install and load R packages for advanced users
  - RStudio coding tools
  - Interactive graphics with ggvis and/or Shiny
  - Report Generation with R Markdown

What is R

- R is an integrated suite of software facilities for
  - importing, storing, exporting and manipulating data;
  - scientific computation;
  - conducting statistical analyses;
  - displaying the results by tables, graphs, etc.
- Highly customizable via thousands of freely available packages.
- R is also a platform for the development and implementation of new algorithms.

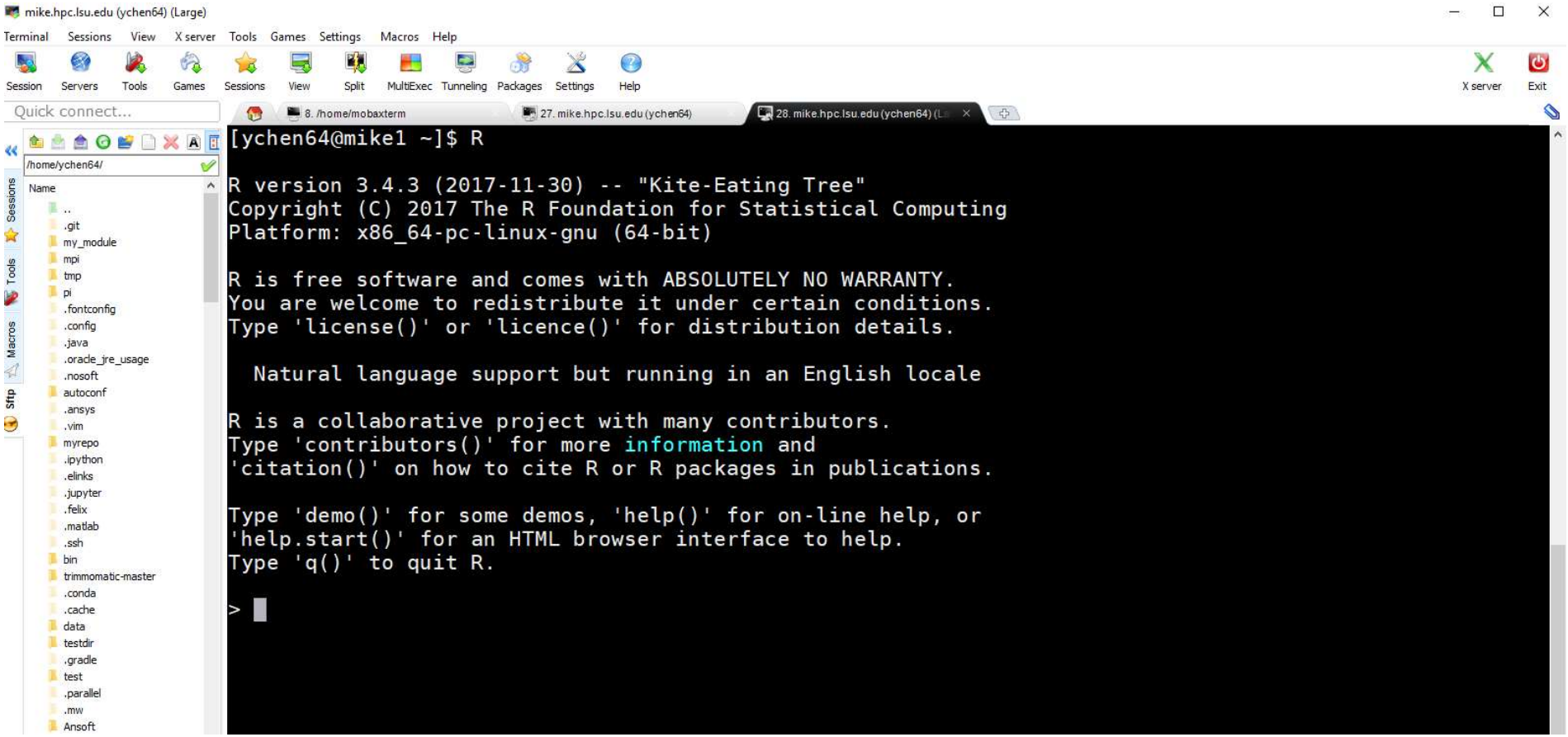
Installing and Loading R

- On your PC
  - RStudio is the de facto environment for R on a desktop system
  - R console from CRAN
  - Microsoft R Open
- On HPC cluster
  - R on all LONI and LSU HPC clusters
    - SuperMIC and QB2: `r/3.5.3/INTEL-18.0.1`
    - SuperMike2: `r/3.5.3/INTEL-18.0.0`
  - RStudio via Open OnDemand on SuperMike2

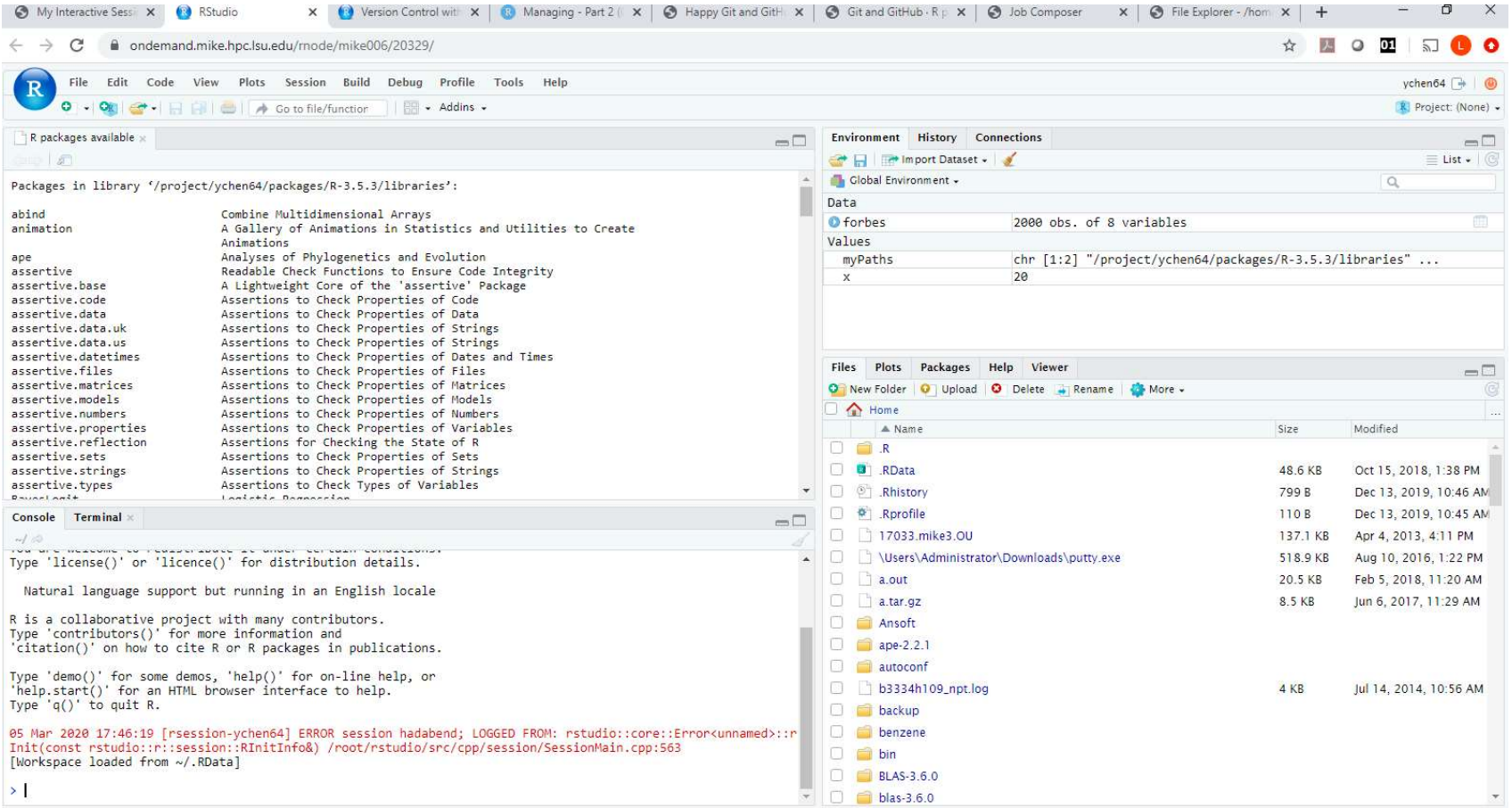
On LONI and LSU HPC Clusters

- Two modes to run R on clusters
  - Interactive mode
    - Type `R` command to launch the console, then run R commands in the console
    - RStudio Server at LSU HPC Open OnDemand (LSU HPC users only)
  - Batch mode
    - Write the R script first, then submit a batch job to run it (use the `Rscript` command)
    - This mode is better for production runs

On LONI and LSU HPC Clusters

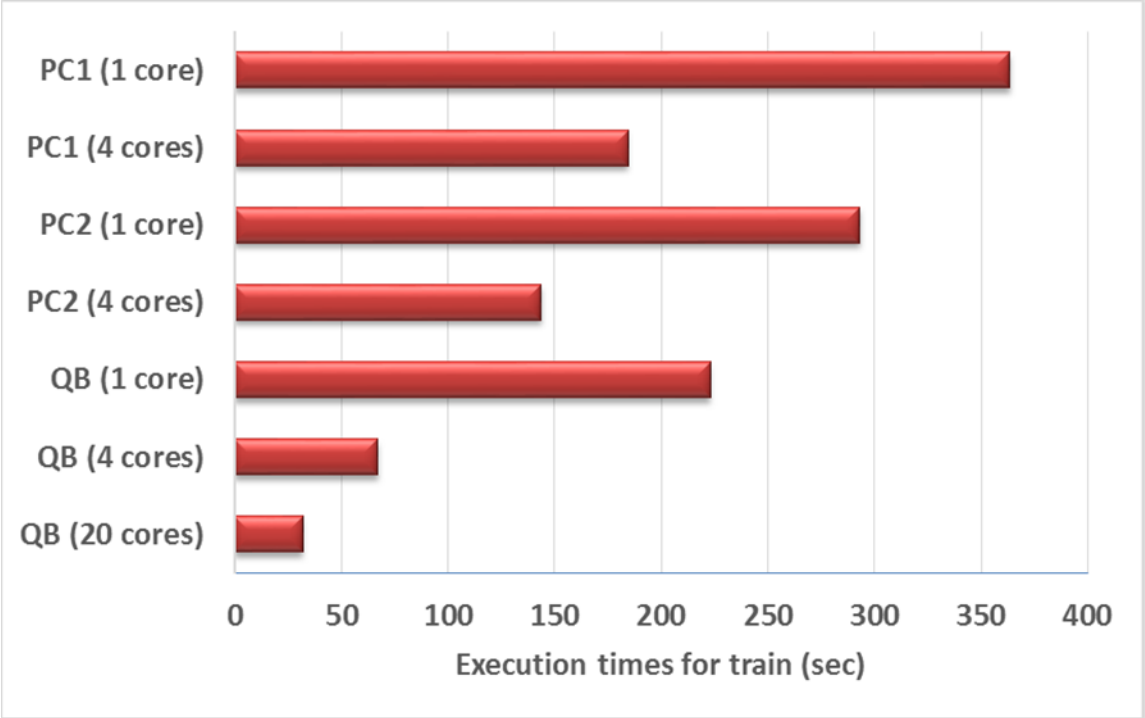


On LONI and LSU HPC Clusters



Clusters are Better for Resource-demanding Jobs

Training random forest model
Resampling method: 10-fold cross-validation



Outline

- RStudio basics
  - R in PC and HPC
  - What is RStudio
  - RStudio IDE features
  - User environment
- Advanced features
  - Use Version Control with RStudio
  - Install and load R packages for advanced users
  - RStudio coding tools
  - Interactive graphics with ggvis and/or Shiny
  - Report Generation with R Markdown

What is RStudio

- RStudio is an **integrated development environment** (IDE)for R
- RStudio is available in two formats:
  - RStudio Desktop
  - RStudio Server
- RStudio Desktop and RStudio Server are both available in free and fee-based (commercial) editions
- Initial release: 28 February 2011

Why RStudio


R STUDIO
...BUT WHY!


Why you guys
never teach RStudio??
imgflip.com


HPC User Services

Why RStudio

- RStudio integrates the tools you use with R into a single environment
- RStudio includes powerful coding tools
- RStudio enables rapid navigation to files and functions
- RStudio make it easy to start new or find existing projects
- RStudio has integrated support for Git and Subversion
- RStudio supports authoring HTML, PDF, Word Documents, and slide shows
- RStudio supports interactive graphics with Shiny and ggvis

Ref. https://rstudio.com/products/rstudio/features/

Why RStudio

https://i2.wp.com/res.cloudinary.com/syknapptic/image/upload/v1521320144/tidyverse_meme_oceake.png

Installing and Loading RStudio

- On your PC
  - RStudio Desktop

  https://rstudio.com/products/rstudio/download/
  - RStudio Server (available for some Linux platforms)
- On HPC cluster
  - RStudio Server via Open OnDemand on SuperMike2
  - Can be installed in your own directory (Not recommended)

Installing and Loading RStudio

- On your PC

RStudio Desktop 1.2.5033 - Release Notes

1. Install R. RStudio requires R 3.0.1+.
2. Download RStudio Desktop. Recommended for your system:

DOWNLOAD RSTUDIO FOR WINDOWS
1.2.5033 | 149.83MB

Requires Windows 10/8/7 (64-bit)

All Installers

Linux users may need to import RStudio's public code-signing key prior to installation, depending on the operating system's security policy.

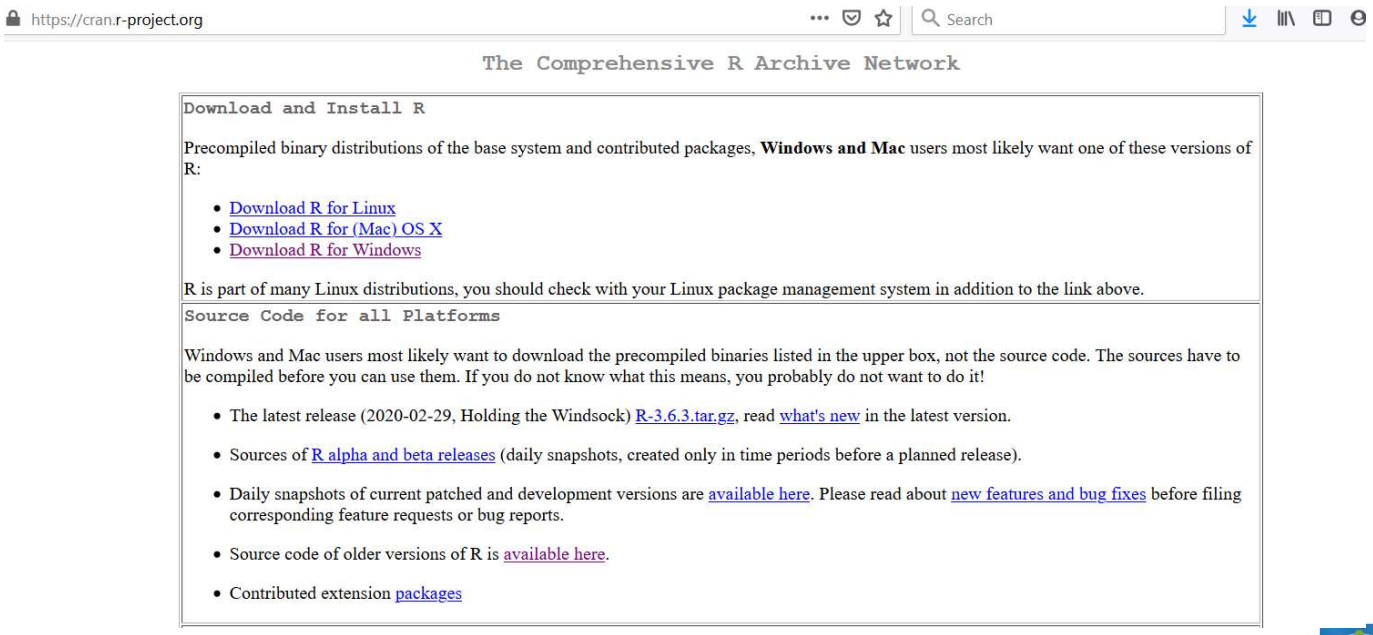
RStudio 1.2 requires a 64-bit operating system. If you are on a 32 bit system, you can use an older version of RStudio.

OS	Download	Size	SHA-256
Windows 10/8/7	RStudio-1.2.5033.exe	149.83 MB	7fd3bc1b
macOS 10.12+	RStudio-1.2.5033.dmg	126.89 MB	b67c9875

Installing and Loading RStudio

- On your PC
  1. Install R.

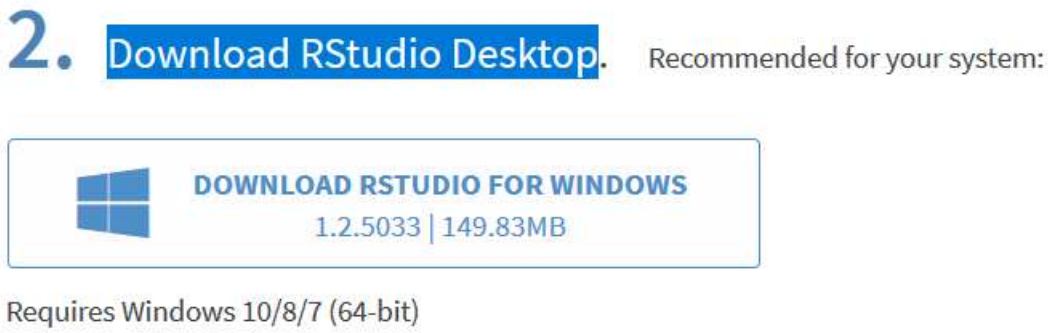
https://cran.r-project.org/



Note: HPC users better choose the same version as on the cluster.

Installing and Loading RStudio

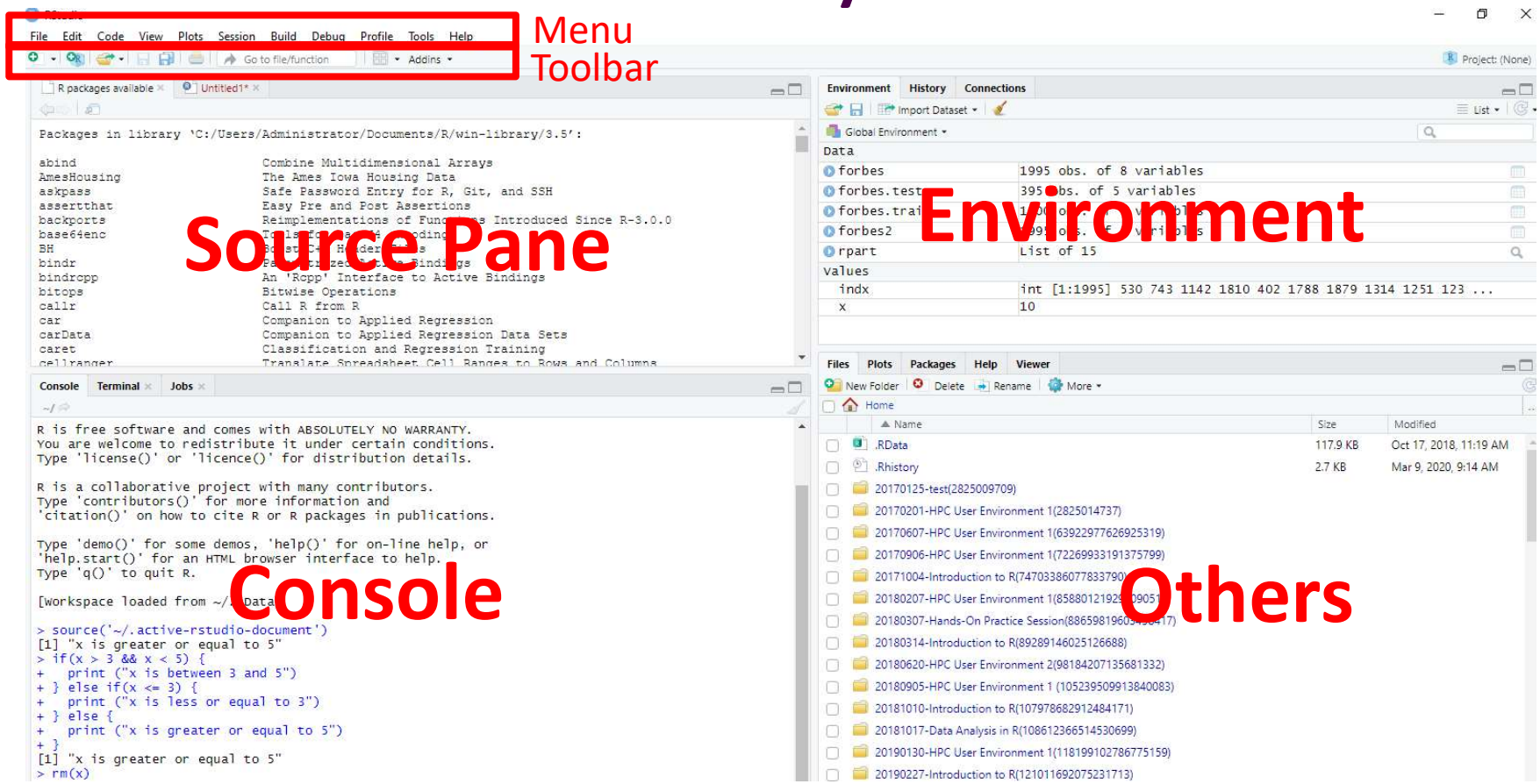
- On your PC
  2. Download RStudio Desktop



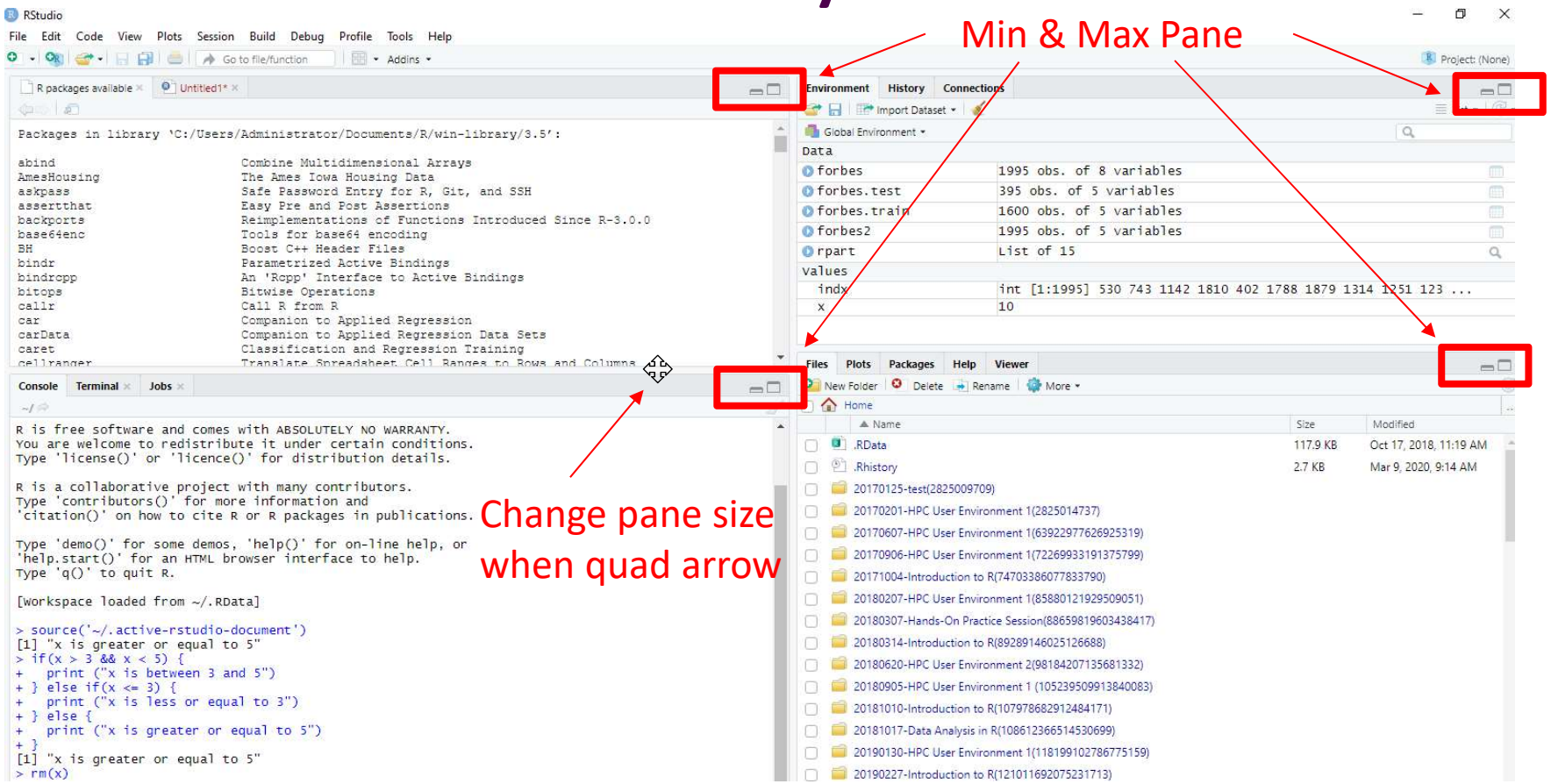
Outline

- RStudio basics
  - R in PC and HPC
  - What is RStudio
  - RStudio IDE features
  - User environment
- Advanced features
  - Use Version Control with RStudio
  - Install and load R packages for advanced users
  - RStudio coding tools
  - Interactive graphics with ggvis and/or Shiny
  - Report Generation with R Markdown

Pane Layout

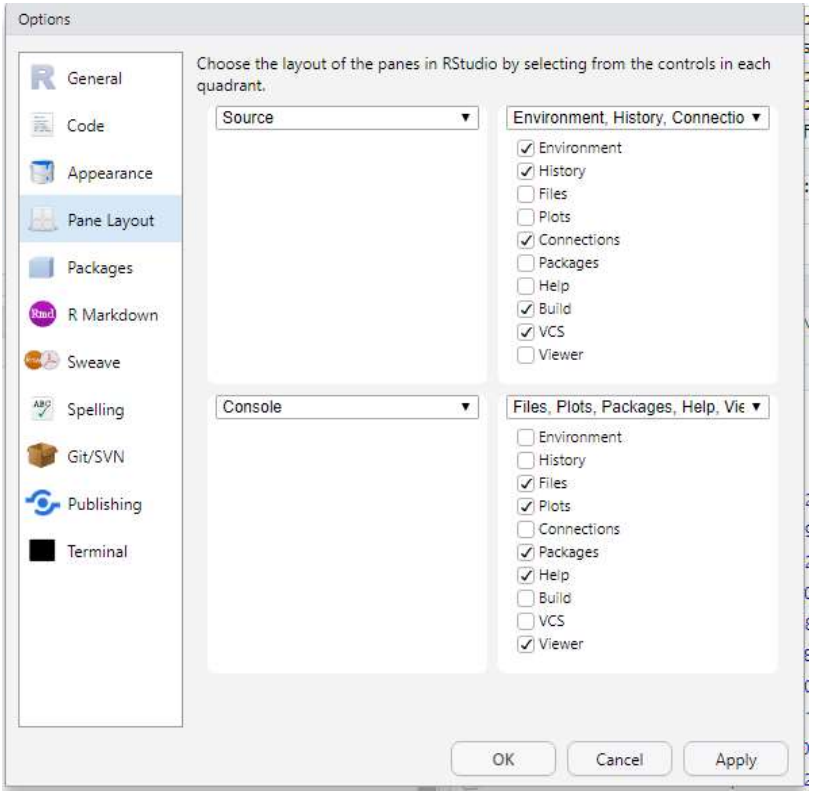


Pane Layout


Min & Max Pane
Change pane size when quad arrow
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
R packages available
Untitled1*
Environment History Connections
Import Dataset
Global Environment
Data
forbes 1995 obs. of 8 variables
forbes.test 395 obs. of 5 variables
forbes.train 1600 obs. of 5 variables
forbes2 1995 obs. of 5 variables
rpart List of 15
values
indx int [1:1995] 530 743 1142 1810 402 1788 1879 1314 1251 123 ...
x 10
Packages in library 'C:/Users/Administrator/Documents/R/win-library/3.5':
abind Combine Multidimensional Arrays
AmesHousing The Ames Iowa Housing Data
askpass Safe Password Entry for R, Git, and SSH
assertthat Easy Pre and Post Assertions
backports Reimplementations of Functions Introduced Since R-3.0.0
base64enc Tools for base64 encoding
BH Boost C++ Header Files
bindr Parametrized Active Bindings
bindrcpp An 'Rcpp' Interface to Active Bindings
bitops Bitwise Operations
callr Call R from R
car Companion to Applied Regression
carData Companion to Applied Regression Data Sets
caret Classification and Regression Training
Console Terminal Jobs
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
[Workspace loaded from ~/.RData]
> source('~/.active-rstudio-document')
[1] "x is greater or equal to 5"
> if(x > 3 && x < 5) {
+ print ("x is between 3 and 5")
+ } else if(x <= 3) {
+ print ("x is less or equal to 3")
+ } else {
+ print ("x is greater or equal to 5")
+ }
[1] "x is greater or equal to 5"
> rm(x)
Files Plots Packages Help Viewer
New Folder Delete Rename More
Home
Name Size Modified
.RData 117.9 KB Oct 17, 2018, 11:19 AM
.Rhistory 2.7 KB Mar 9, 2020, 9:14 AM
20170125-test(2825009709)
20170201-HPC User Environment 1(2825014737)
20170607-HPC User Environment 1(63922977626925319)
20170906-HPC User Environment 1(72269933191375799)
20171004-Introduction to R(74703386077833790)
20180207-HPC User Environment 1(85880121929509051)
20180307-Hands-On Practice Session(88659819603438417)
20180314-Introduction to R(89289146025126688)
20180620-HPC User Environment 2(98184207135681332)
20180905-HPC User Environment 1 (105239509913840083)
20181010-Introduction to R(107978682912484171)
20181017-Data Analysis in R(108612366514530699)
20190130-HPC User Environment 1(118199102786775159)
20190227-Introduction to R(121011692075231713)

Customizing Pane Layout

Menu “Tools” -> “Global Options”-> “Pane Layout”

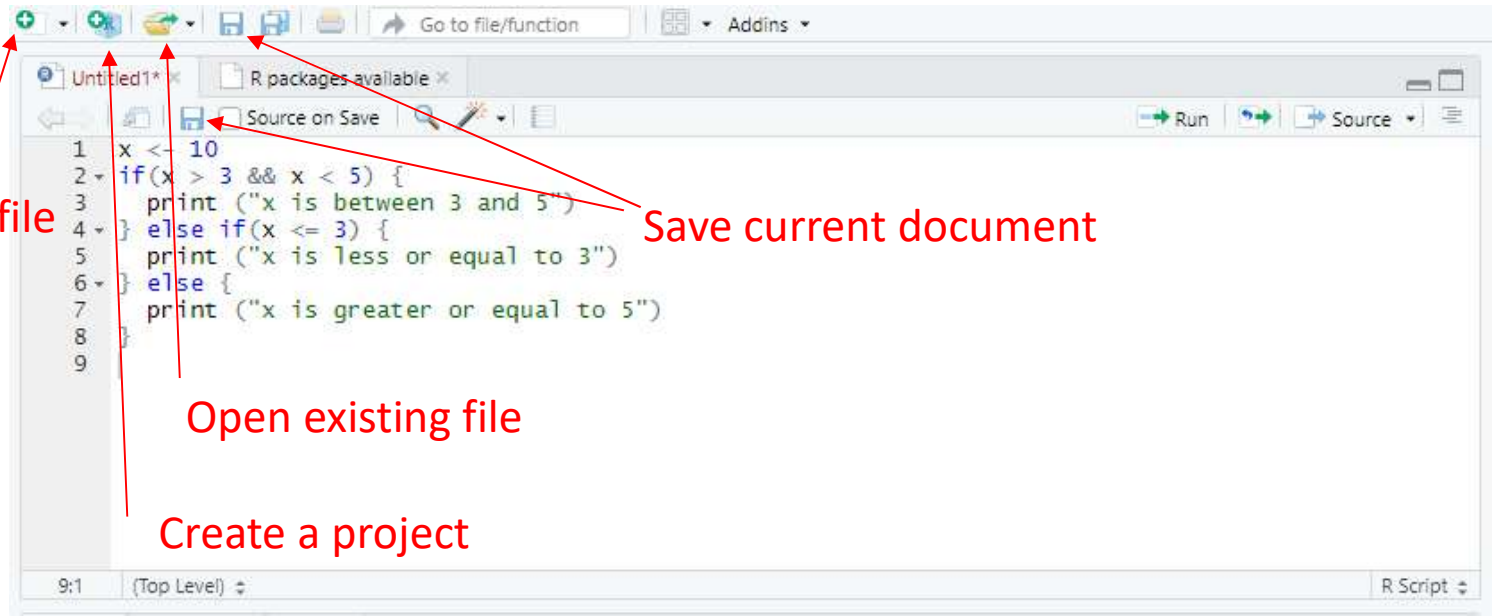


Toolbar and Source Pane

Creating or opening various files (e.g. R script)

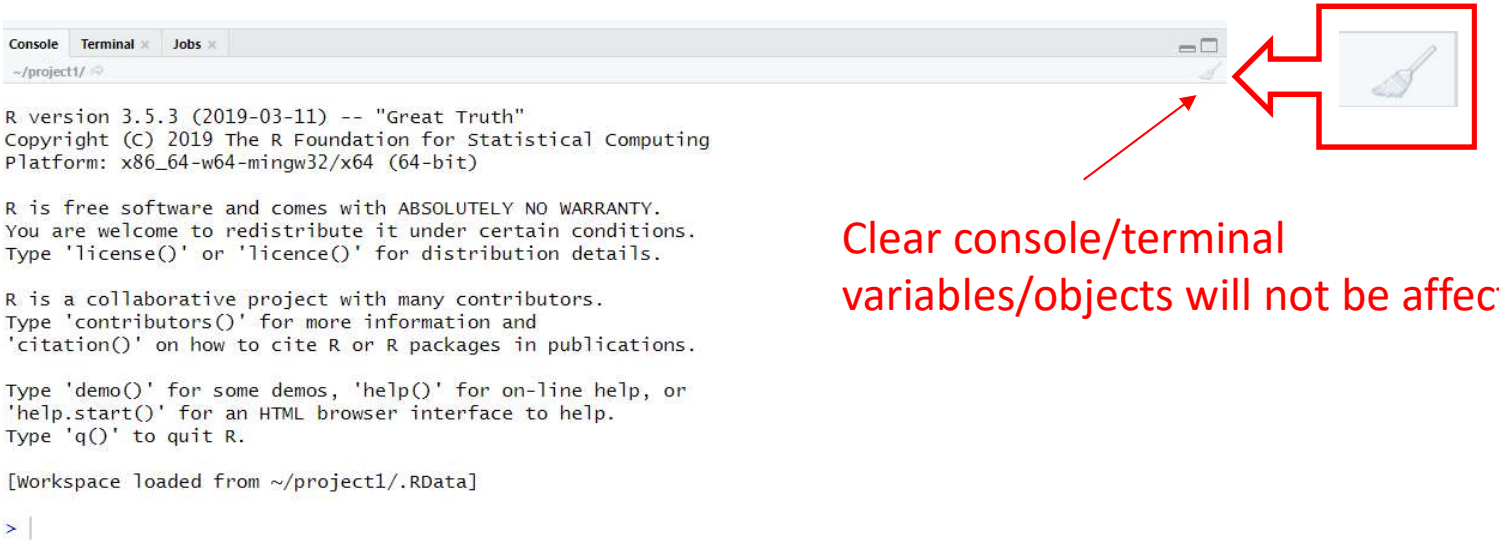
Visualizing data

Coding window



Note: raw data opened here CANNOT be accessed in R

Console & Terminal



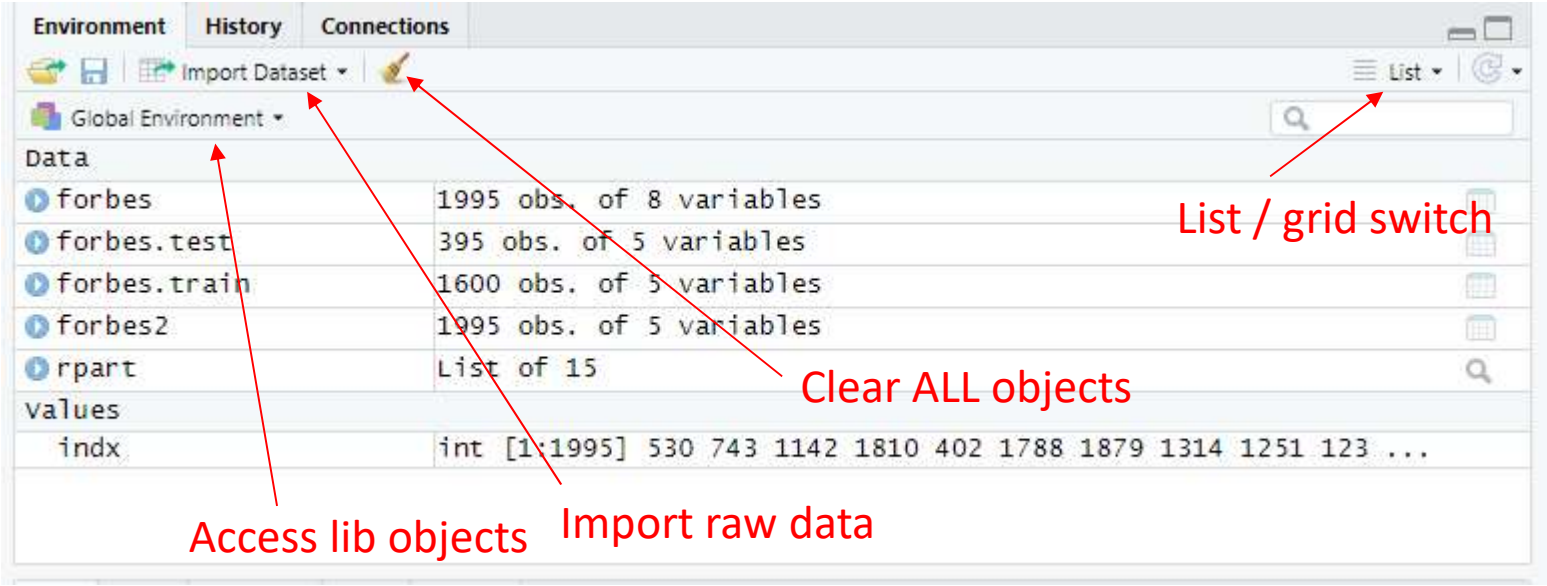
Clear console/terminal variables/objects will not be affected

- Tips: source, console and terminal all support:
  - Automatic completion of typing file, directory or command name via the `TAB` key
  - Recall previous commands using the up arrow (↑)

Keyboard Shortcuts

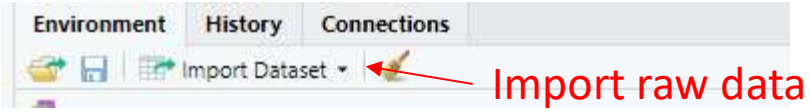
- Some of the more useful shortcuts
  - `Ctrl+1` — Move focus to the Source Editor
  - `Ctrl+2` — Move focus to the Console
  - `Ctrl+L` — Clear the Console
  - `Esc` — Interrupt R

Environment & History & Connections



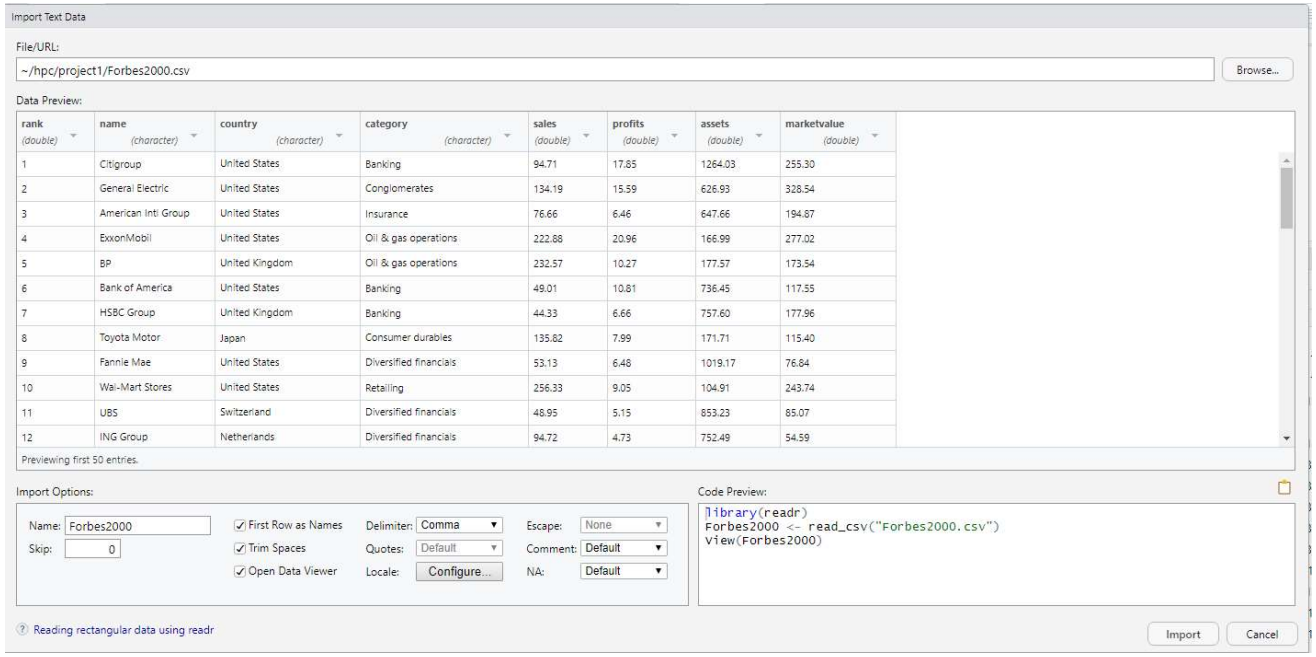
- Data to data, values to values
- Importing raw data (Next slide)

Environment & History & Connections



Import raw data

also in menu “File” -> “Import Dataset”



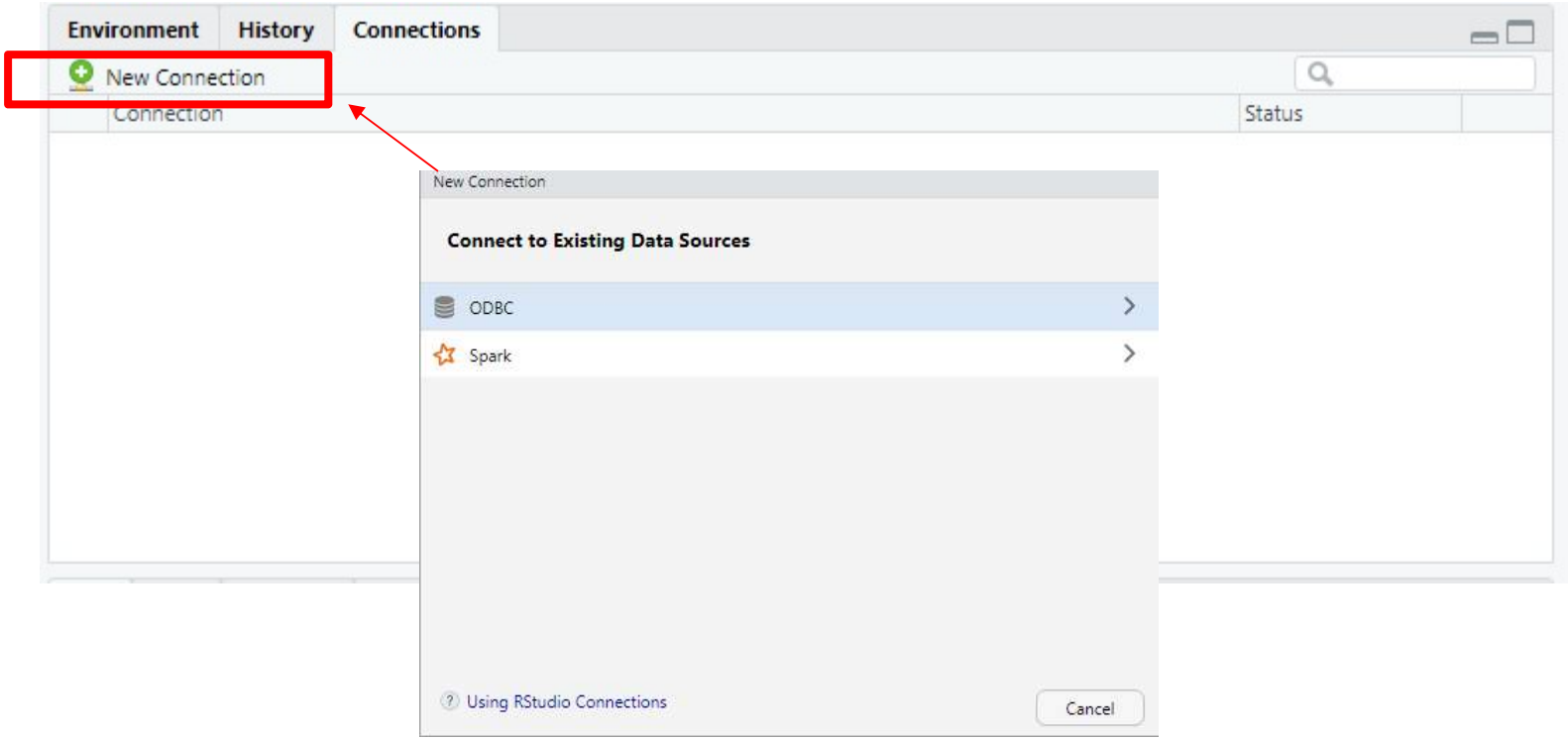
rank (double)	name (character)	country (character)	category (character)	sales (double)	profits (double)	assets (double)	marketvalue (double)
1	Citigroup	United States	Banking	94.71	17.85	1264.03	255.30
2	General Electric	United States	Conglomerates	134.19	15.59	626.93	328.54
3	American Intl Group	United States	Insurance	76.66	6.46	647.66	194.87
4	ExxonMobil	United States	Oil & gas operations	222.88	20.96	166.99	277.02
5	BP	United Kingdom	Oil & gas operations	232.57	10.27	177.57	173.54
6	Bank of America	United States	Banking	49.01	10.81	736.45	117.55
7	HSBC Group	United Kingdom	Banking	44.33	6.66	757.60	177.96
8	Toyota Motor	Japan	Consumer durables	135.82	7.99	171.71	115.40
9	Fannie Mae	United States	Diversified financials	53.13	6.48	1019.17	76.84
10	Wal-Mart Stores	United States	Retailing	256.33	9.05	104.91	243.74
11	UBS	Switzerland	Diversified financials	48.95	5.15	853.23	85.07
12	ING Group	Netherlands	Diversified financials	94.72	4.73	752.49	54.59

- Intuitive for Windows user
- Support raw file in plain text, Excel, SAS etc.
- Preview provided (use readr or later for better review)
- Code provided for later scripting
- Meet most of the requirements

Environment & History & Connections

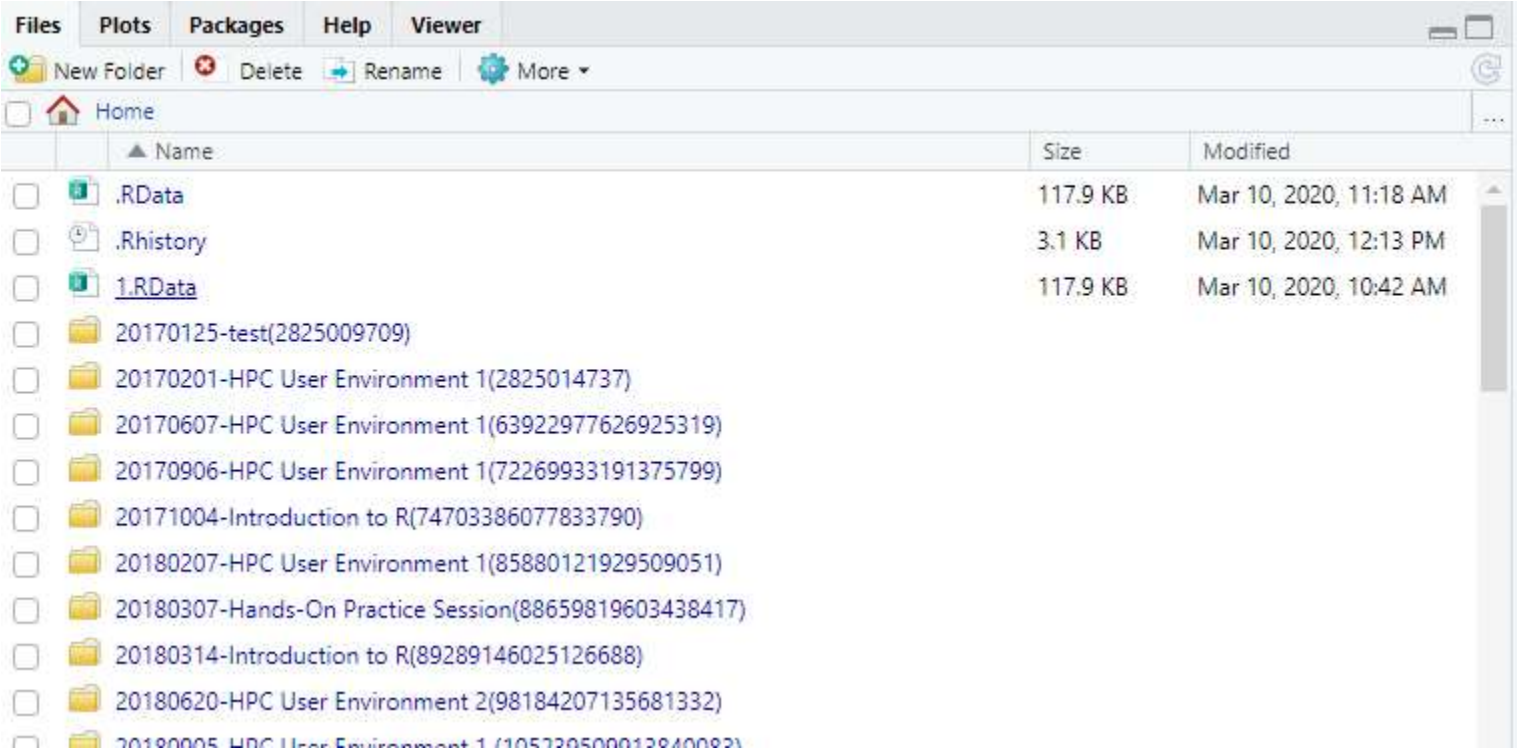
- Tips:
  - History can be saved to / loaded from a file
  - One or multiple selected commands can be sent to Console or Source
  - Remove selected or clear all of the entries

Environment & History & Connections



- The Connection Pane connects to a variety of data sources, and explore the objects and data inside the connection.

Files & Plots & Packages & Help & Viewer



- Tips: you CAN customize Pane Layout

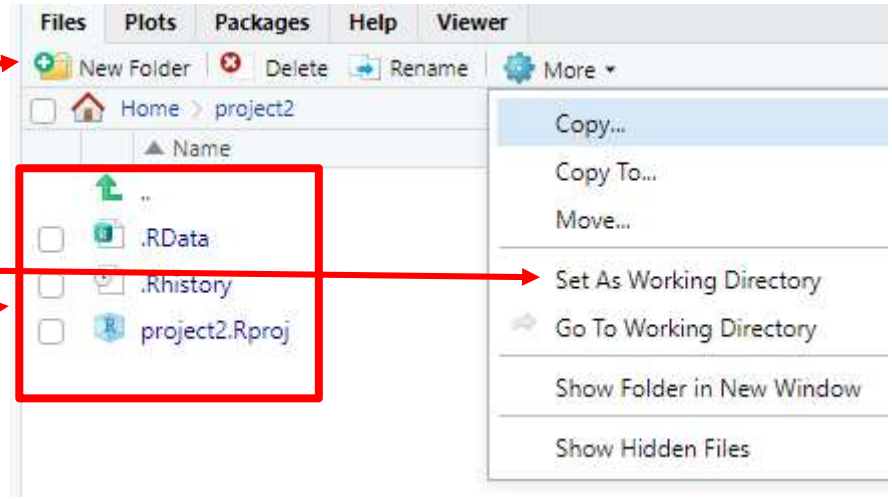
Outline

- RStudio basics
  - R in PC and HPC
  - What is RStudio
  - RStudio IDE features
  - User environment
- Advanced features
  - Use Version Control with RStudio
  - Install and load R packages for advanced users
  - RStudio coding tools
  - Interactive graphics with ggvis and/or Shiny
  - Report Generation with R Markdown

How does R user environment work

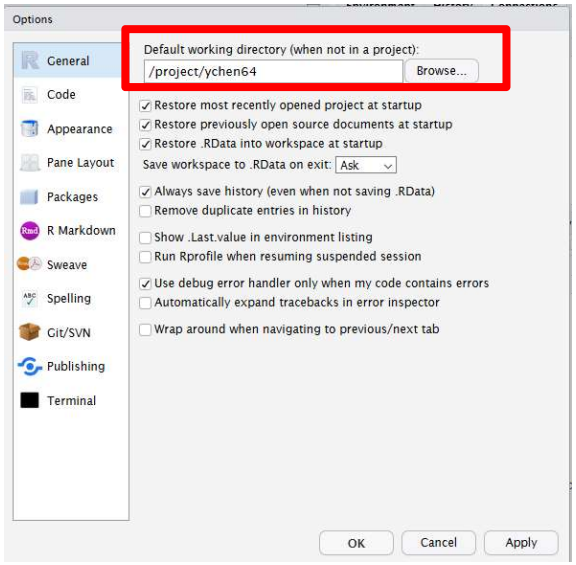
- R works best if you have a dedicated folder called “working directory”. Put all data files in the working directory (or in its subdirectories).

```
> getwd()  #Show current working directory
[1] "C:/Users/Administrator/Documents/project2"
> dir.create("data") #Create a new directory
> getwd()
[1] "/home/ychen64"
> setwd("data")
> getwd()
[1] "C:/Users/Administrator/Documents/project2/data"
> list.files()  # List files in current directory
```



How does R user environment work

- Default working directory (typically referenced using ~ in R)
  - Windows PC: C:\Users\Administrator(or your username)\Documents
  - Linux including HPC: the user home directory $HOME
- Change default working directory in menu “Tools” -> “Global Options” -> “General”



How does R user environment work

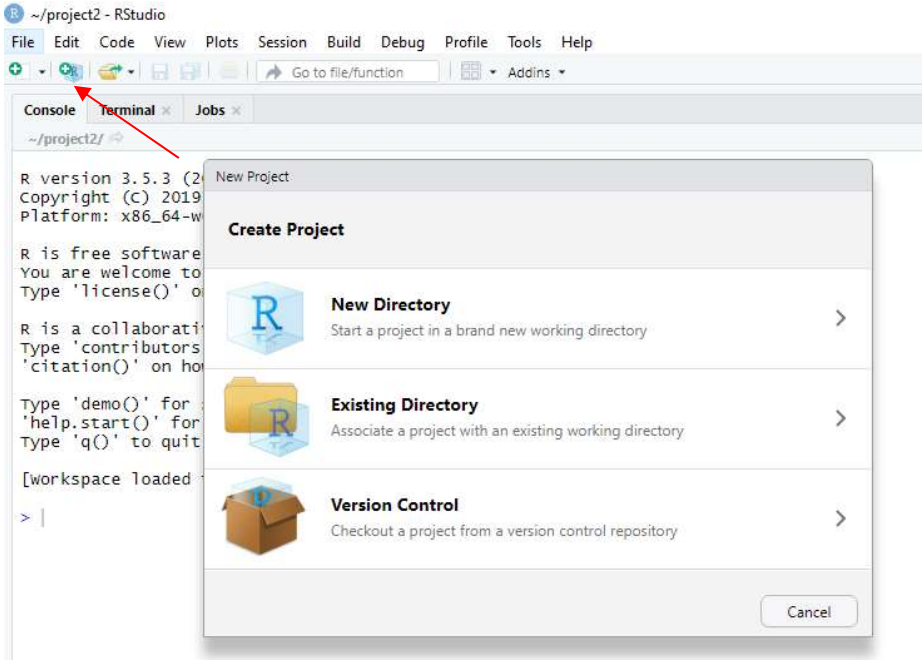
- Your objects will be automatically saved in the .RData file in the working directory.
- To quit use `q()` or `CTRL + D` or just kill the window. R will ask you "**Save workspace image to .RData**?". You can choose:
  - Save: leave R without saving your results in R;
  - Don't Save: save your results in .RData in your working directory;
  - Cancel: not quitting R.
- The commands you type in console or executed in Source Pane will be automatically saved in the .Rhistory file in the working directory.

Projects in RStudio

- RStudio Project is working directory “Pro”
  - includes the functionality of working directory
  - provides “Project Options” to set on a per-project basis to customize the behavior of RStudio

    Menu “Tools” -> “Project Options”
  - works with version control system

Creating Projects in RStudio



- When creating new project RStudio creates:
  - a project file (with an .Rproj extension) within the project directory
  - a hidden directory (named .Rproj.user) where project-specific temporary files are stored, which is also automatically added to .Rbuildignore, .gitignore, etc. if required.

Opening Projects in RStudio

- RStudio project can be opened
  - in menu “File” -> “Open Project…”
  - on the toolbar
  - by double-clicking the .Rproj file
- When a project is opened within RStudio the following actions are taken:
  - A new R session (process) is started
  - The .RData, .Rhistory and **.Rprofile** (if any) files in the project's main directory is sourced by R
  - The current working directory is set to the project directory

Closing Projects in RStudio

- RStudio project can be closed
  - in menu “File” -> “Close Project…” (w/o quitting RStudio)
  - in menu “File” -> “Open Project…”
  - when closing the RStudio (if project is closed in this way, it will be started automatically when opening RStudio next time)

Outline

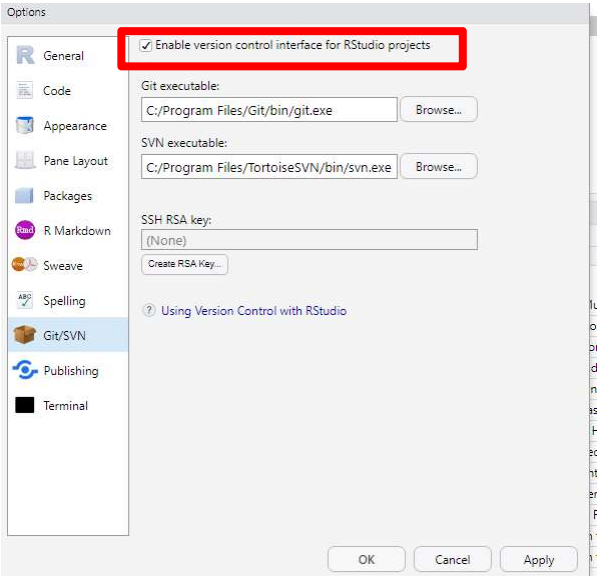
- RStudio basics
  - R in PC and HPC
  - What is RStudio
  - RStudio IDE features
  - User environment
- Advanced features
  - Use Version Control with RStudio
  - Install and load R packages for advanced users
  - RStudio coding tools
  - Interactive graphics with ggvis and/or Shiny
  - Report Generation with R Markdown

What is and Why Version Control

- Version Control is the management of changes to documents, computer programs, large web sites, and other collections of information.
- Version control is not only good for team collaboration but also benefits for individual work.
  - Why should I use version control?
  - R and version control for the solo data analyst
- RStudio IDE has integrated support for version control.:
  - Git
  - Subversion

Installing and Activating Git in RStudio

- Installation
  - PC: http://git-scm.com/downloads
  - HPC Clusters: load the Module key for Git
- Enable Git in menu “Tools” -> “Global Options” -> “Git/SVN”

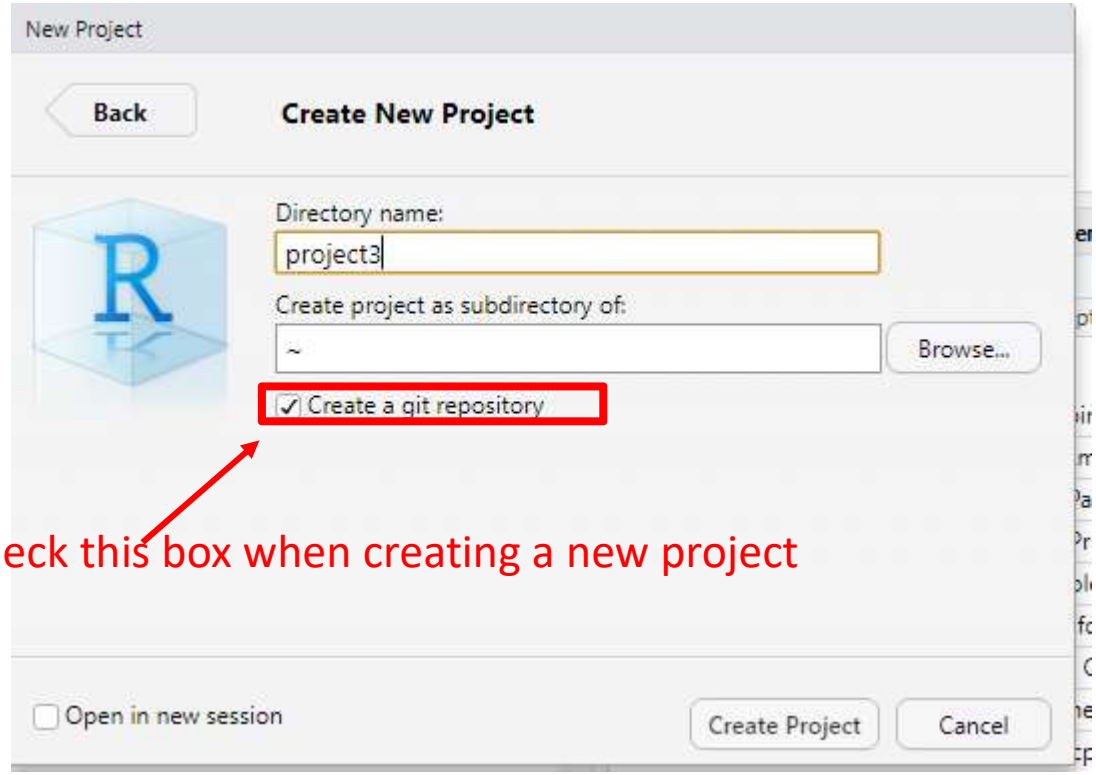


Various Scenarios of Initializing Git in RStudio

- Create a new project
  - with a totally new Git repository
  - based on an existing remote Git repository (Github)
  - using a directory already under version control
- Add version control to an existing project
  - using remote repositories
  - using local Git

Various Scenarios of Initializing Git in RStudio

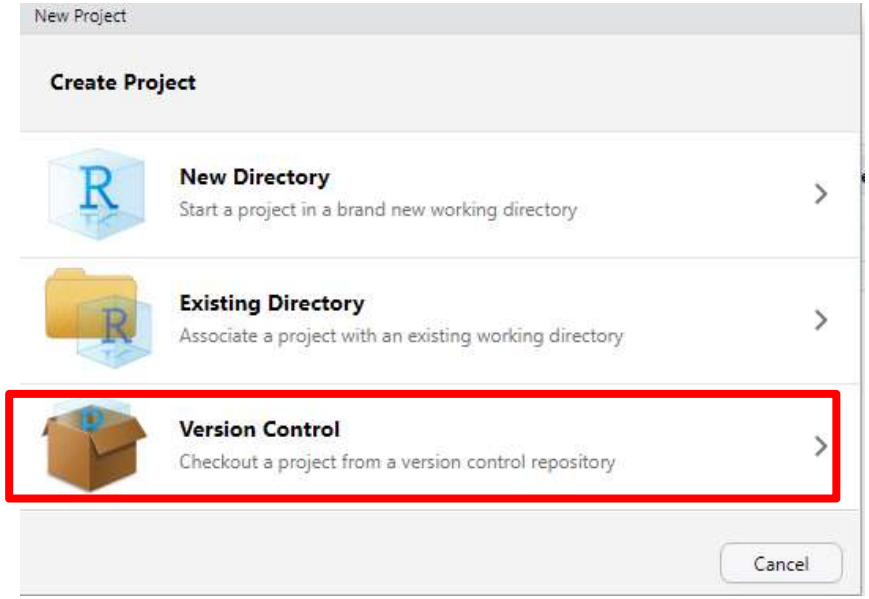
- Create a new project with a brand new Git repository

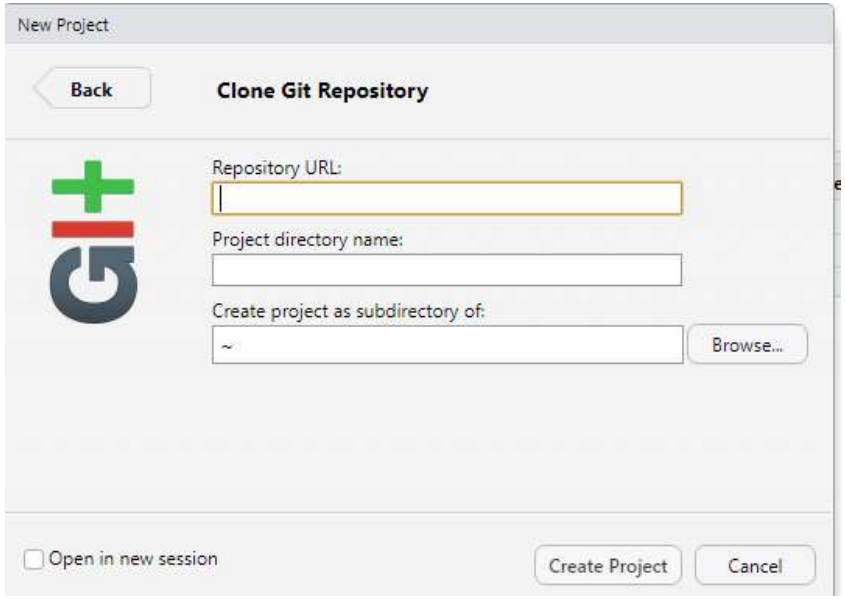


Check this box when creating a new project

Various Scenarios of Initializing Git in RStudio

- Create a new project from an existing remote Git repo (Github)

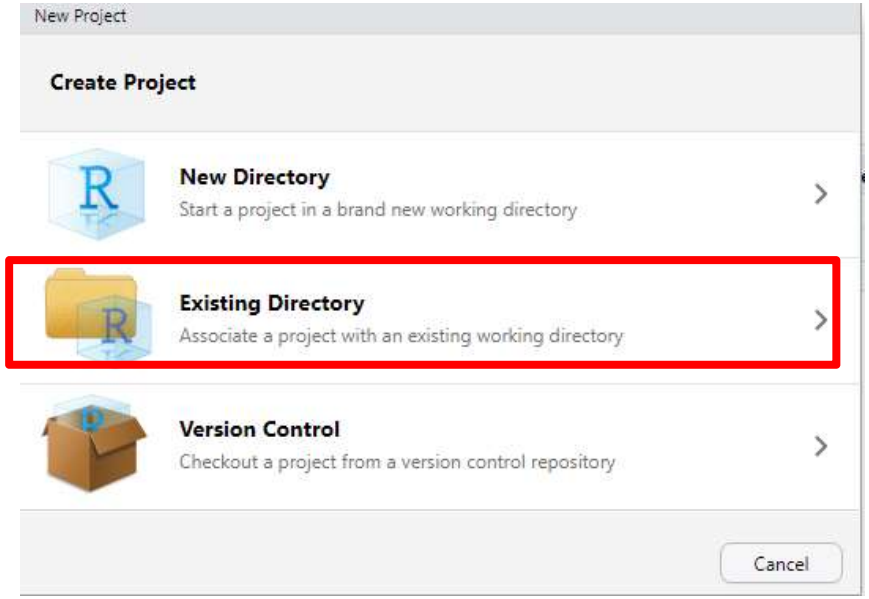


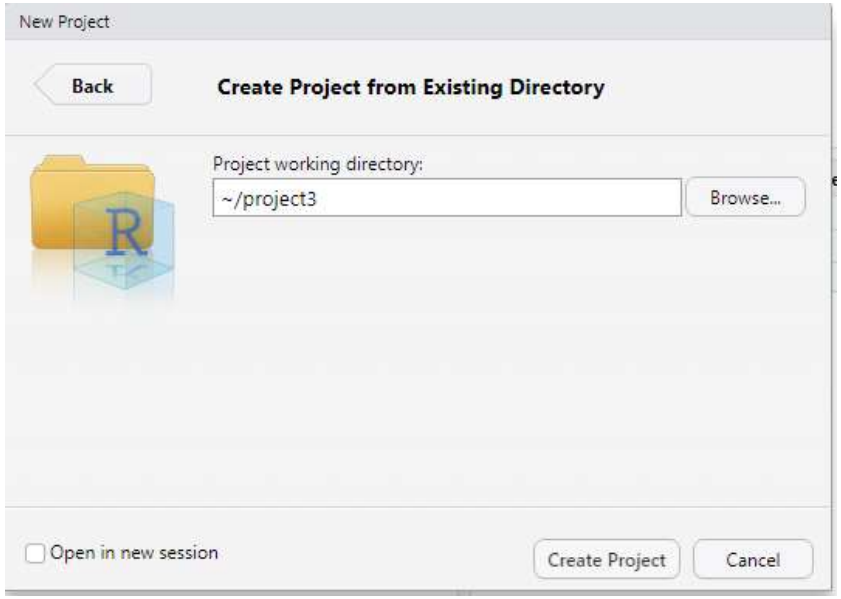


Equals to "git clone", then creating a new project based on the directory already under version control

Various Scenarios of Initializing Git in RStudio

- Create a new project using a directory already under version control

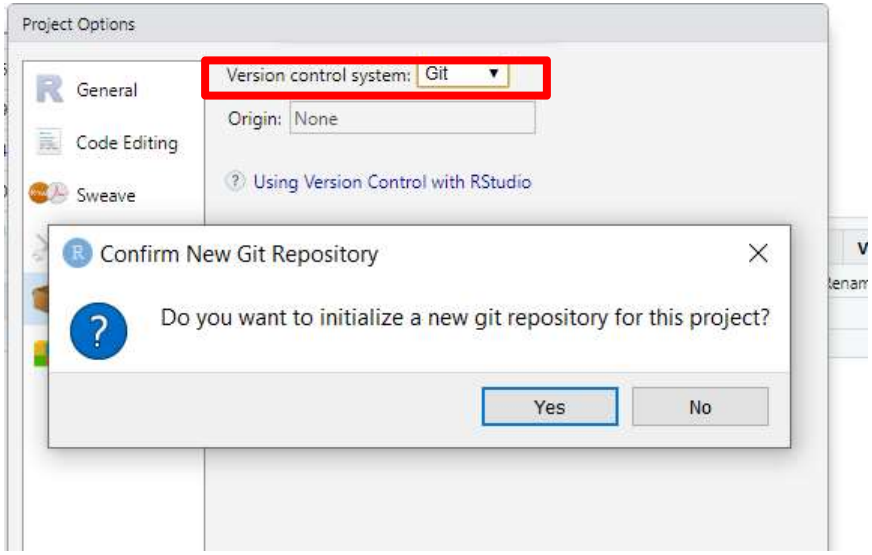




Various Scenarios of Initializing Git in RStudio

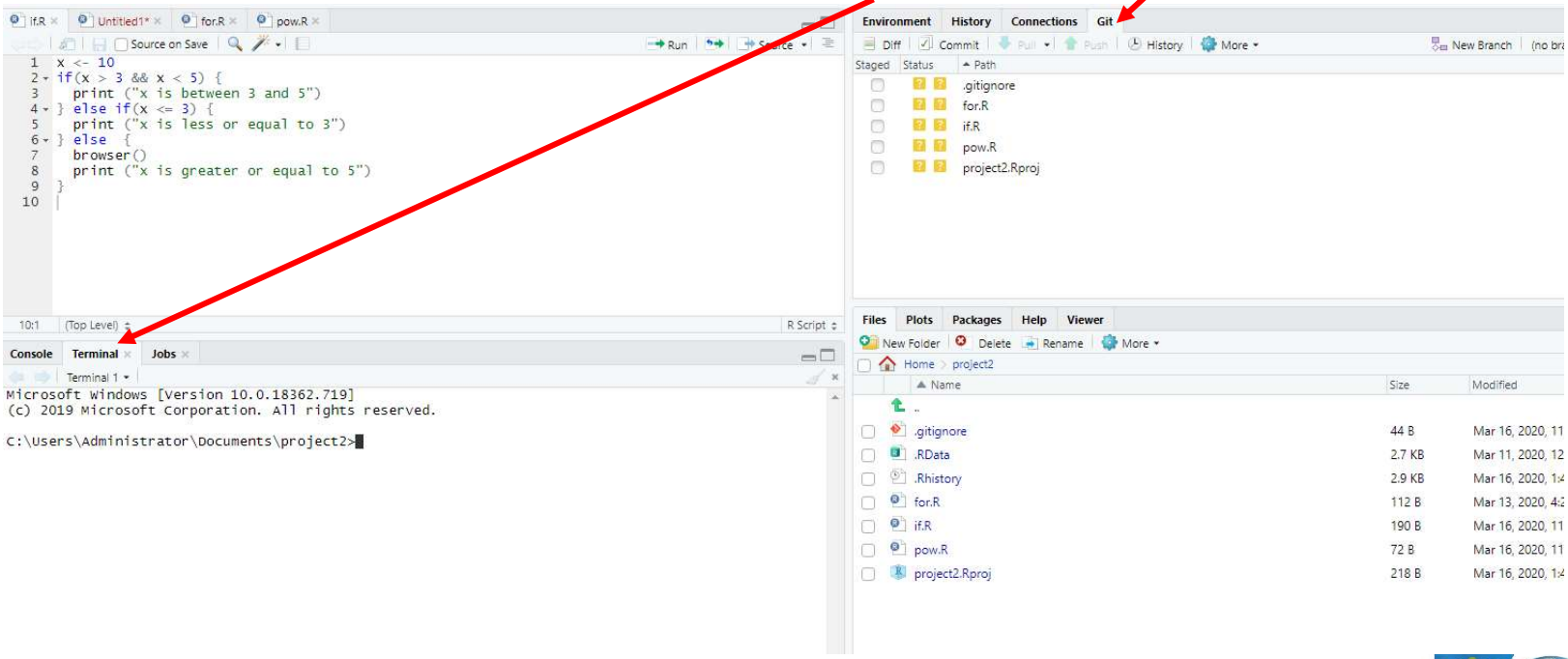
- Add version control to an existing project
  - using remote repositories (rare)
  - using local Git

  in menu “Tools” -> “Project Options” -> “Git/SVN”



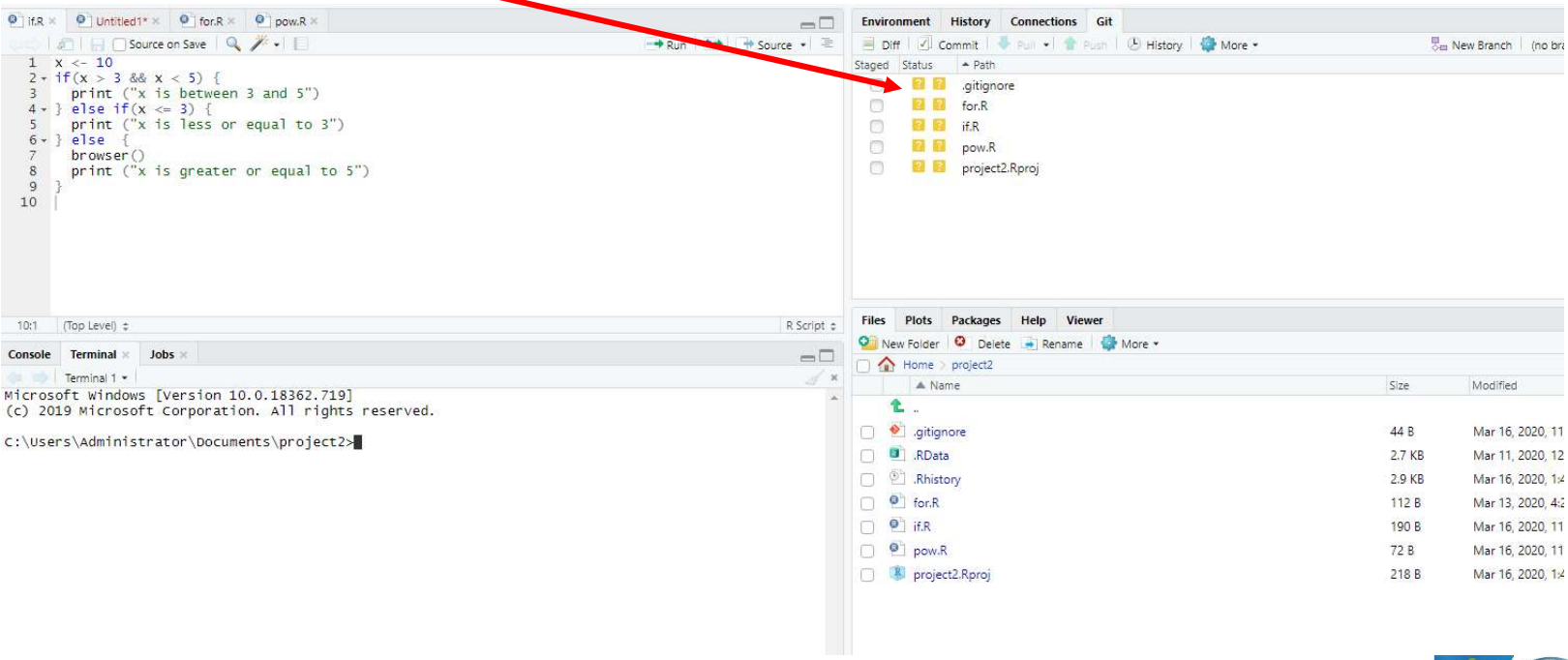
Git Panes

- The Git pane shows the file status (in terms of git)
- Git commands can be typed in Terminal



Working with Git

Untracked file, not in the Git yet (or no one cares about its change)



Working with Git

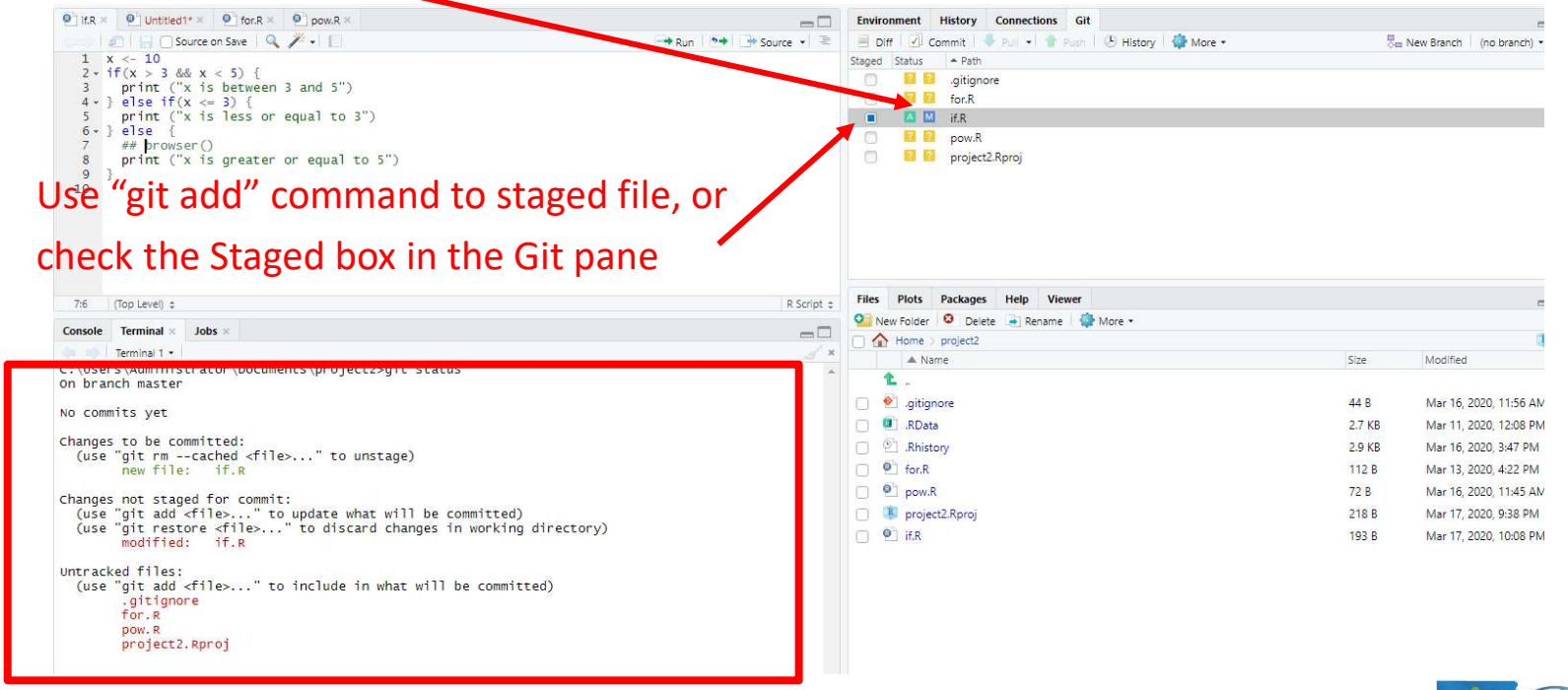
Added the untracked file

```
x <- 10
if(x > 3 && x < 5) {
  print ("x is between 3 and 5")
} else if(x <= 3) {
  print ("x is less or equal to 3")
} else  {
  browser()
  print ("x is greater or equal to 5")
}
```

```
C:\Users\Administrator\Documents\project2>git add if.R

C:\Users\Administrator\Documents\project2>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   if.R

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        for.R
        pow.R
        project2.Rproj
```

Working with Git

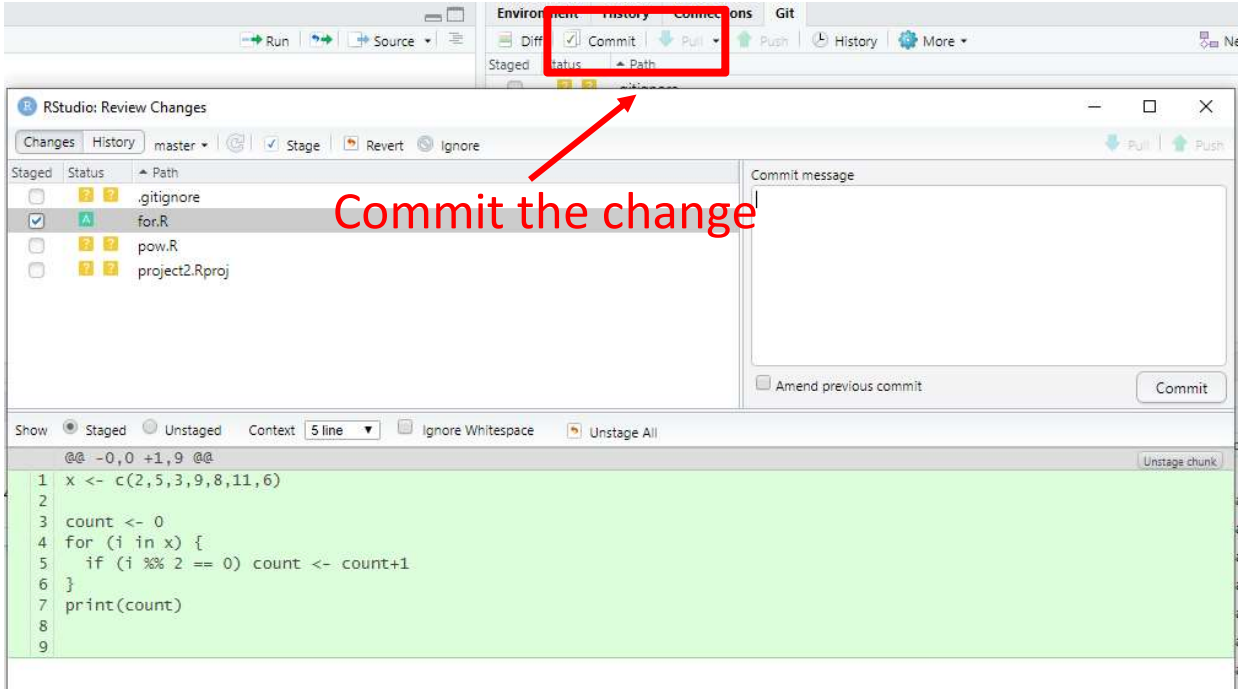
Modified file, may use "git add" to staged file, or the change can be reversed to let the status be unmodified



- Two icons for the same file ("dual" status in RStudio)
  - First icon is the RStudio status, second is Git status

Working with Git

Once a file is committed, its status is “unmodified” and will not shown in the Git pane (unless been changed/removed again)



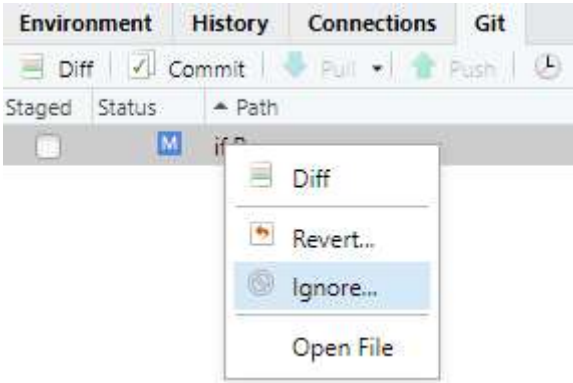
- git commit -m “commit message”

Ignoring Files

- Any Files such as temporary, very large or R project log files that you don't want Git to automatically add or even show you as being untracked can be
  - added to `.gitignore`

```
$ cat .gitignore
*.[oa]
# tells Git to ignore any files ending in ".o" or ".a" object
```

  - selected in the Git pane



Create & switch Git branch

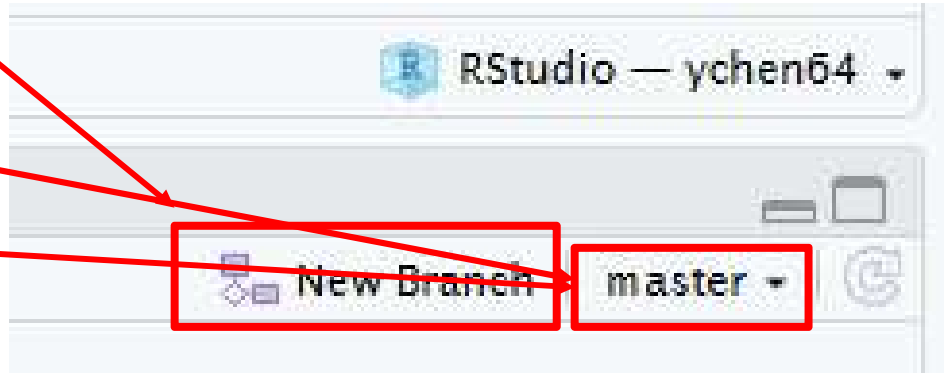
- Git branches are effectively a pointer to a snapshot of your changes such as adding a new feature or fixing a bug.
  - create new Git branch

    ```
    $ git checkout -b the_branch_name
    ```
  - check branch

    ```
    $ git branch
    ```
  - switch branch

    ```
    $ git checkout the_branch_name
    ```
  - delete branch

    ```
    $ git branch -d the_branch_name
    ```

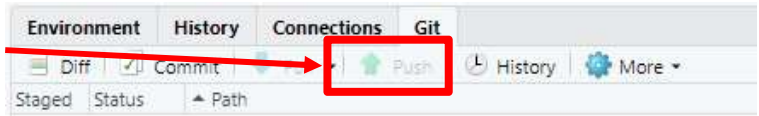


Push Current Repository to GitHub

- If the current repository was created from an existing repository on GitHub
  - `git remote` and `git push` commands:

```
$ git push -u origin master
```

  - or push button in the Git pane:

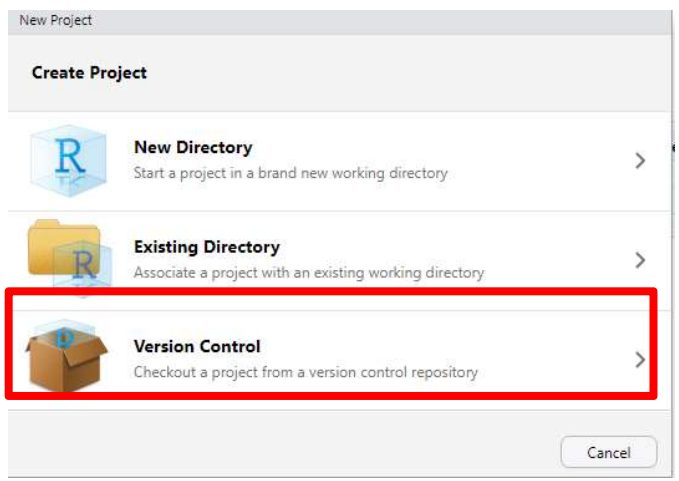


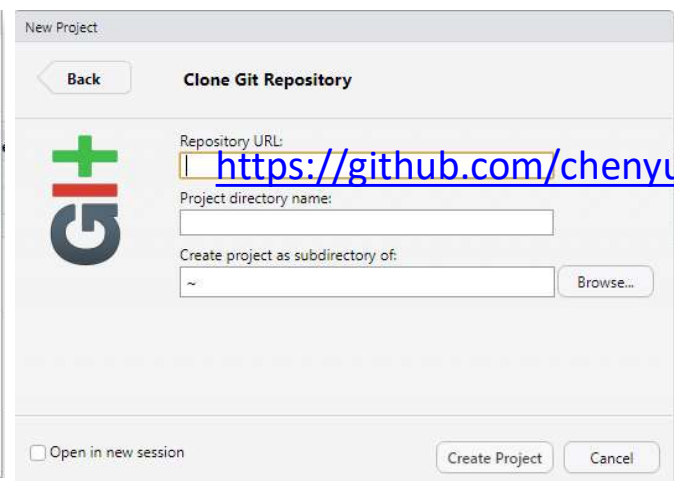
- If the current repository hasn't been connected to GitHub:
  - Create a new repo on GitHub: https://github.com/new. Give it the same name as your project.
  - `git remote` and then `git push` commands

```
$ git remote add origin https://github.com/chenyuetian/RStudio.git
$ git push -u origin master
```

Practice: Create a new project based on GitHub

- Installation
  - PC: http://git-scm.com/downloads
  - HPC Clusters: load the Module key for Git
- Enable Git in menu “Tools” -> “Global Options” -> “Git/SVN”
- Create a new project from an existing remote Git repo (Github)





Outline

- RStudio basics
  - R in PC and HPC
  - What is RStudio
  - RStudio IDE features
  - User environment
- Advanced features
  - Use Version Control with RStudio
  - Install and load R packages for advanced users
  - RStudio coding tools
  - Interactive graphics with ggvis and/or Shiny
  - Report Generation with R Markdown

Installing and Loading R Packages

- libraries that R currently searching can be shown with `.libPaths()` in the R console

```
> .libPaths ()
[1] "/home/ychen64/packages/R/libraries"  # user's own directory
[2] "/home/packages/r/3.4.3/INTEL-18.0.0/lib64/R/library"  #system path
```

- Installation:
  - Option 1: menu "Tools" -> "Install Packages"
  - Option 2: run `install.packages("<package name>")` function in the console
- On Windows the compiler collection needed for installing packages from source is called Rtools. Download it from http://cran.r-project.org/bin/windows/Rtools/
- Loading: the `library <package name>` function load previously installed packages

Installing and Loading R Packages - HPC Cluster

- You do NOT own a directory to install your packages by default, you need to specify it with one of these two options:
  - Option 1: Point the environment variable `R_LIBS_USER` to a desired location (**doesn't apply for OOD RStudio**)

```
[ychen64@mike002 ~]$ export R_LIBS_USER=/home/ychen64/packages/R/libraries
[ychen64@mike002 ~]$ echo $R_LIBS_USER
/home/ychen64/packages/R/libraries
```

  - Option 2: Save the path of packages to `.Rprofile` (particularly useful with RStudio project)

```
$ cat /home/ychen64/project1/.Rprofile
myPaths <- .libPaths() #system path
myPaths <- c("/home/ychen64/project1","/project/ychen64/packages/R-3.5.3/libraries",myPaths)
.libPaths(myPaths)
```

A new directory dependent to the project (CREATE IT FIRST!)

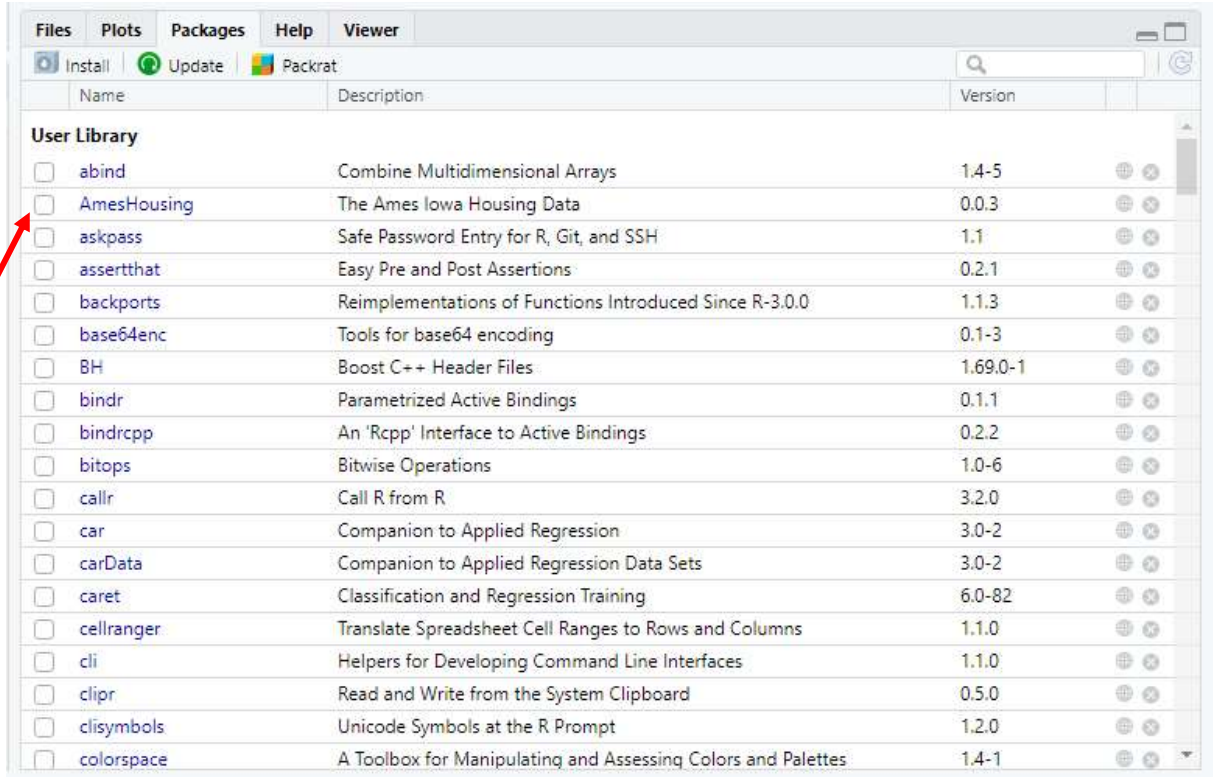
Your own directory by default

Listing and Unloading R Packages - Command Lines

- List all available packages `library()`
- List all packages in the default system library `library(lib = .Library)`
- Show currently loaded libraries: `search()` function or `sessionInfo()` function
- Check package version: `packageVersion ("<package name>")`
- Unload `detach(package:<package name>)`

```
[ychen64@mike002 ~]$ R

R version 3.3.3 (2017-03-06) -- "Another Canoe"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)
…

> library()
> library(lib = .Library)

> search()
 [1] ".GlobalEnv"        "package:swirl"     "package:stats"
 [4] "package:graphics"  "package:grDevices" "package:utils"
 [7] "package:datasets"  "package:methods"   "Autoloads"
[10] "package:base"
> packageVersion("swirl")
> detach(package:swirl)
```

Listing and Unloading R Packages - RStudio GUI



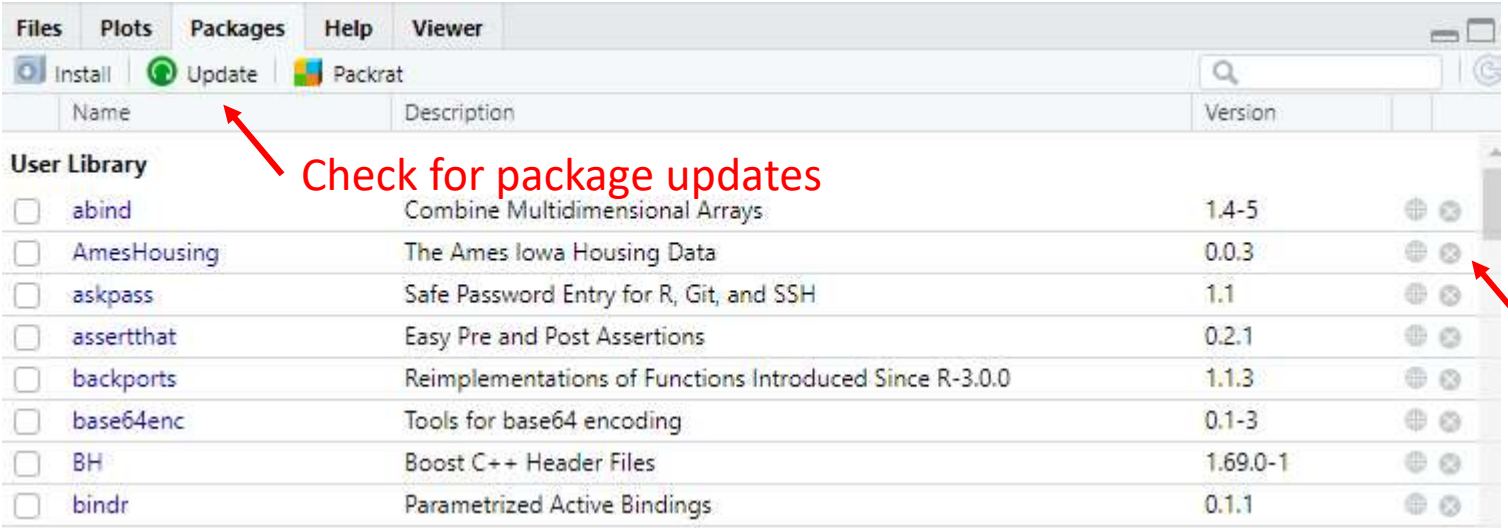
Load/unload package by selecting /deselecting

Scroll down to check system packages

Package version

Updating and Uninstall R Packages - PC and Cluster

- Update `update.packages("<package name>")`
- Uninstall `remove.packages("<package name>")`



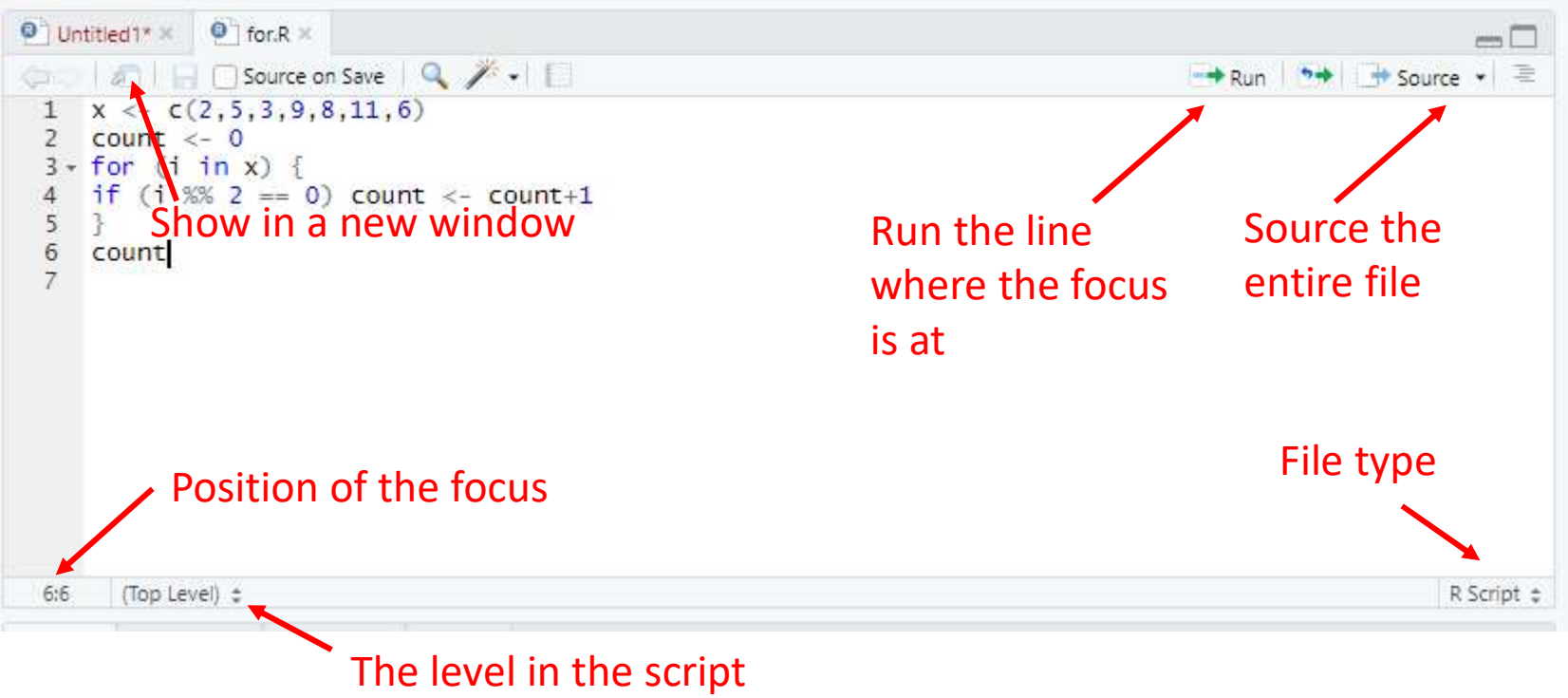
Do you need to update a package to the newest?

Please use good judgement

Outline

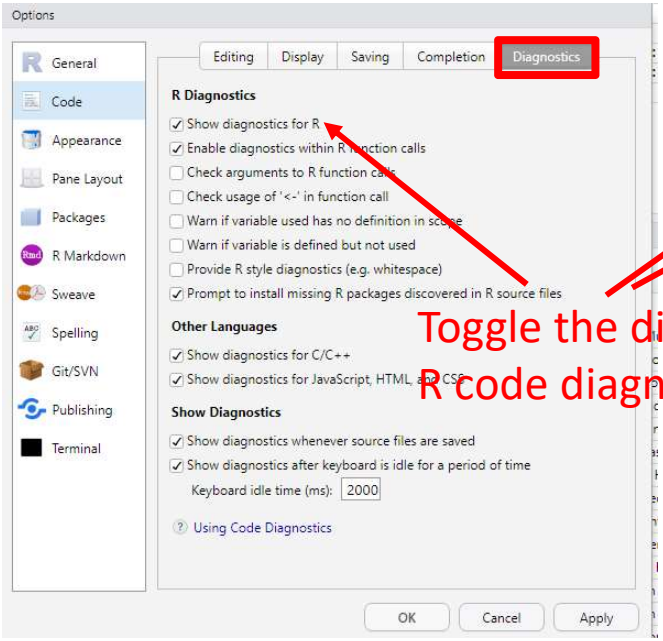
- RStudio basics
  - R in PC and HPC
  - What is RStudio
  - RStudio IDE features
  - User environment
- Advanced features
  - Use Version Control with RStudio
  - Install and load R packages for advanced users
  - RStudio coding tools
  - Interactive graphics with ggvis and/or Shiny
  - Report Generation with R Markdown

RStudio Coding Window Overview

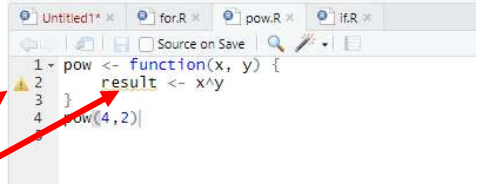


Code Diagnostics in RStudio

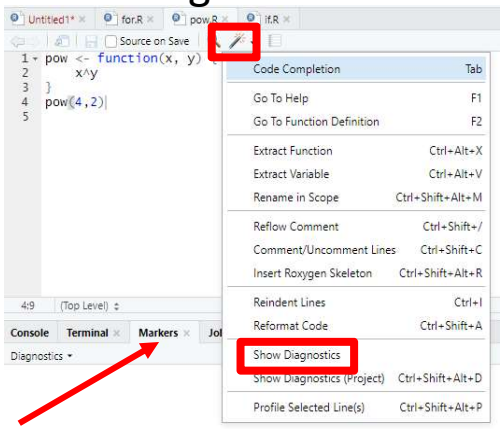
- Diagnostics can be enabled and options can be set in menu "Tools" -> "Global Options" -> "Code"



Toggle the display of R code diagnostics



Show diagnostics in new tab:

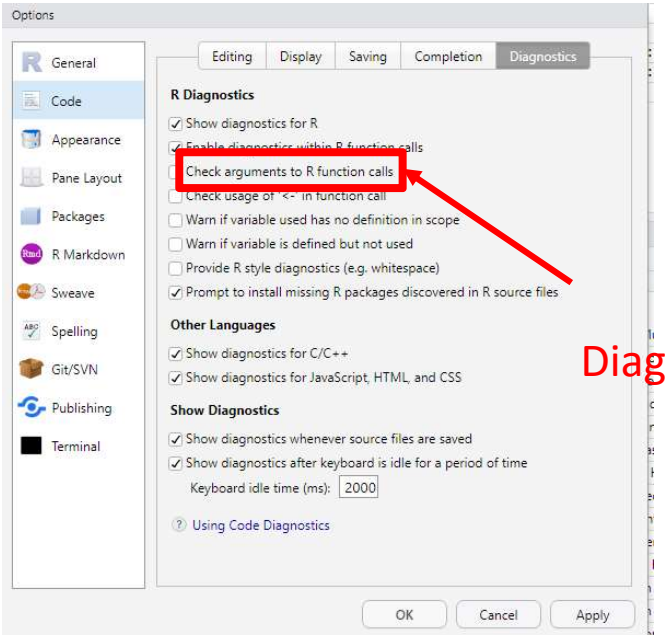


New tab

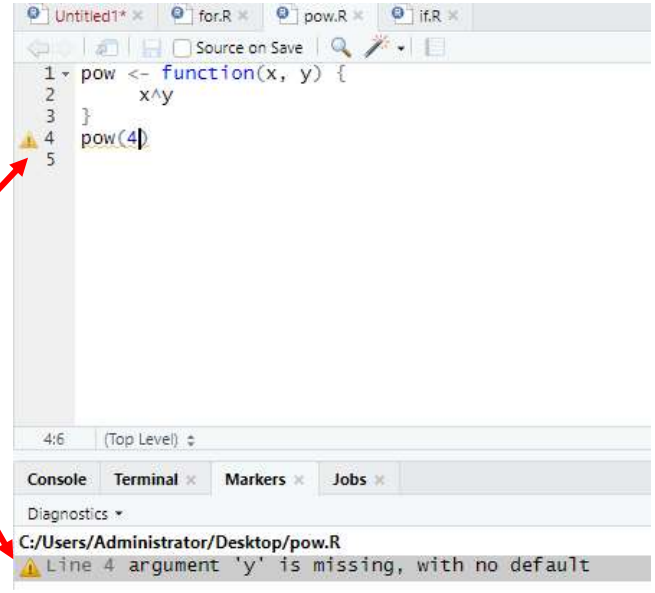
Ref. https://support.rstudio.com/hc/en-us/articles/205753617-Code-Diagnostics

Code Diagnostics in RStudio

- "Check arguments to R function calls" tries to detect whether a particular call to a function will succeed.



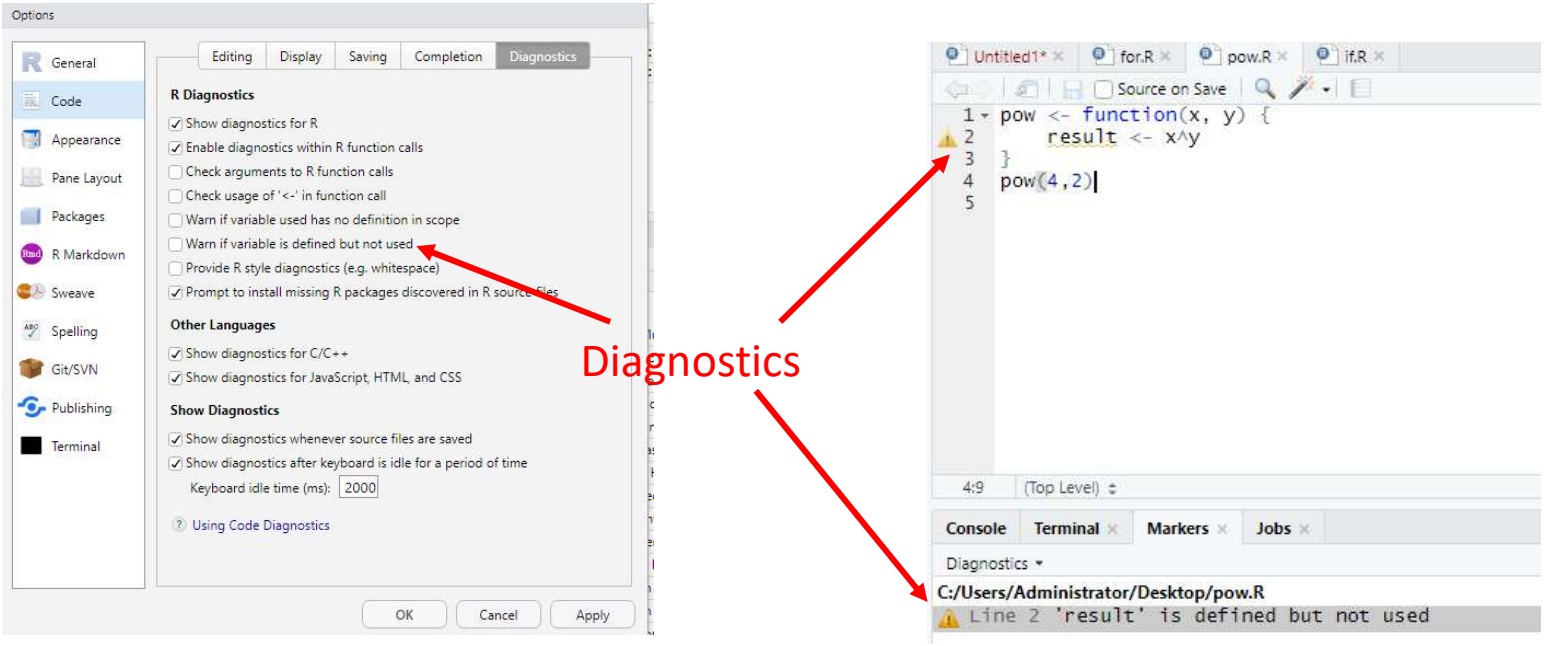
Diagnostics



Ref. https://support.rstudio.com/hc/en-us/articles/205753617-Code-Diagnostics

Code Diagnostics in RStudio

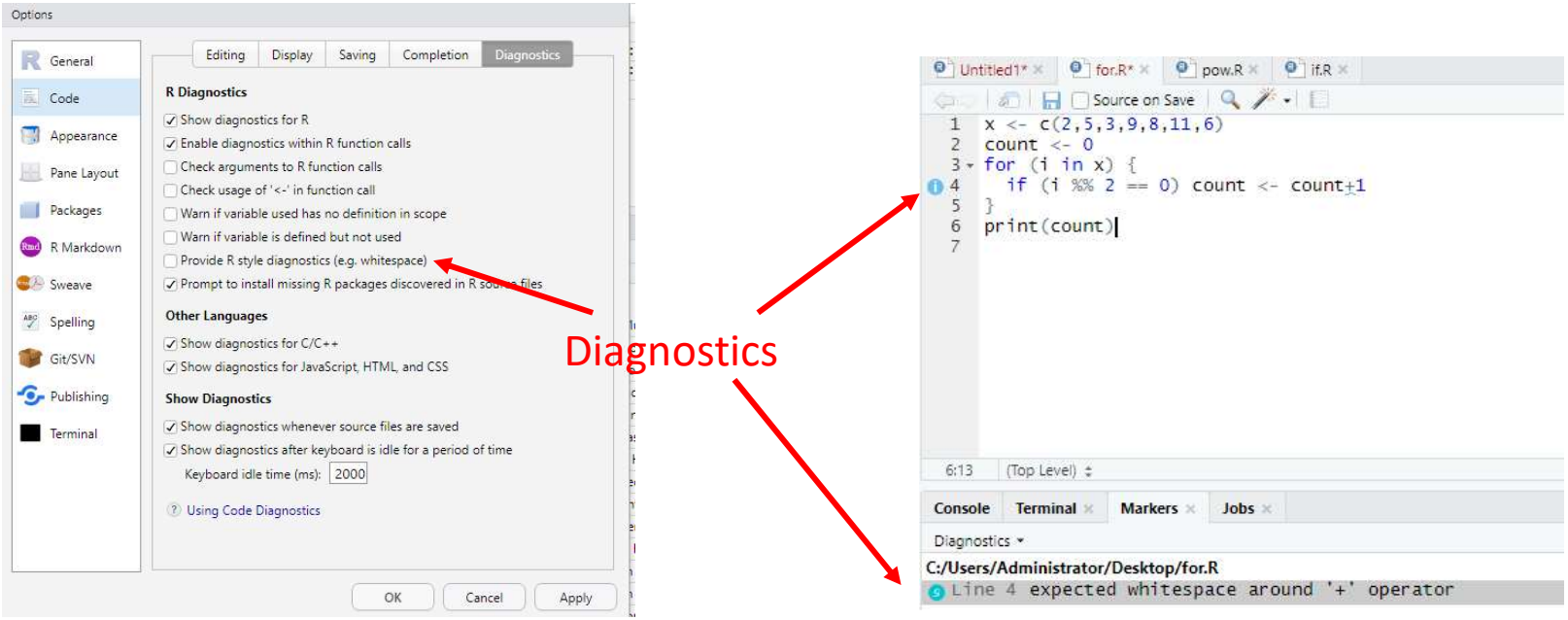
- "Warn if variable is defined but not used" helps to identify is a variable is created but never used.



Ref. https://support.rstudio.com/hc/en-us/articles/205753617-Code-Diagnostics

Code Diagnostics in RStudio

- "Provide R style diagnostics (e.g. whitespace)" checks to see if your code conforms to Hadley Wickham's style guide, and reports style warnings when encountered.



Ref. https://support.rstudio.com/hc/en-us/articles/205753617-Code-Diagnostics

Code Debugging in RStudio

- RStudio has integrated the R debugging tools.
- In order to enter debug mode, you'll need to tell R when you want to pause the computation.
  - R doesn't have a "pause now" feature (not useful as most R computations are too fast to stop in the middle)
  - Pick best way to pause calculation

Ref. https://support.rstudio.com/hc/en-us/articles/205612627-Debugging-with-RStudio#introduction

Entering the Debug Mode

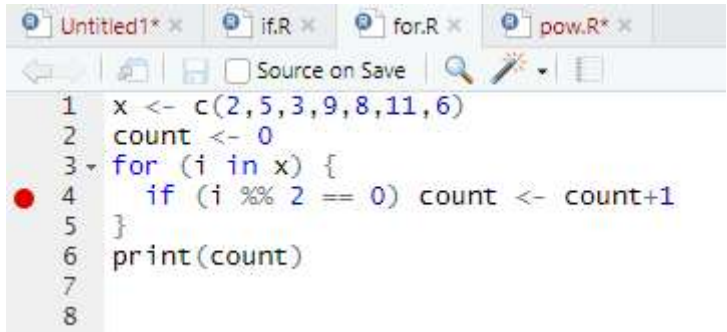
- Stopping on a line
  - Editor breakpoints
  - `browser()` breakpoints
- Stopping when a function executes
- Stopping when an error occurs

Ref. https://support.rstudio.com/hc/en-us/articles/205612627-Debugging-with-RStudio#introduction

Stopping on a Line

- Editor breakpoints (menu “Debug” -> “Toggle Breakpoint”)
  - is the most common (and easiest) way to stop on a line
  - takes effect immediately and don’t require you to change your code
  - works by injecting some tracing code into the R function object.

Red dot

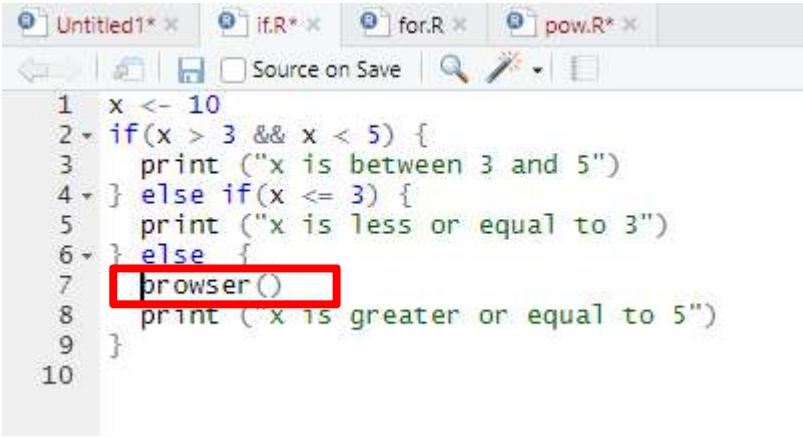


Editor breakpoints

[Ref. https://support.rstudio.com/hc/en-us/articles/205612627-Debugging-with-RStudio#introduction](https://support.rstudio.com/hc/en-us/articles/205612627-Debugging-with-RStudio#introduction)

Stopping on a Line

- `browser()` breakpoints
  - halts execution and invokes an environment browser when it is called
  - is actually part of the code, so it needs to be applied like any other code change in order to become active (e.g. source it)

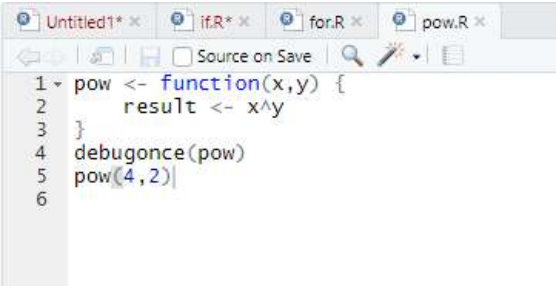


`browser()` breakpoints

Ref. https://support.rstudio.com/hc/en-us/articles/205612627-Debugging-with-RStudio#introduction

Stopping when a Function Executes

- Why need this type of stopping?
  - Sometimes you don't have the source file for the code you want to debug.
- The breakpoint causes the debugger to activate immediately when the function is run.
  - a one-shot breakpoint: `debugonce()`



  - debug a function every time it executes: `debug()` and `undebug()`

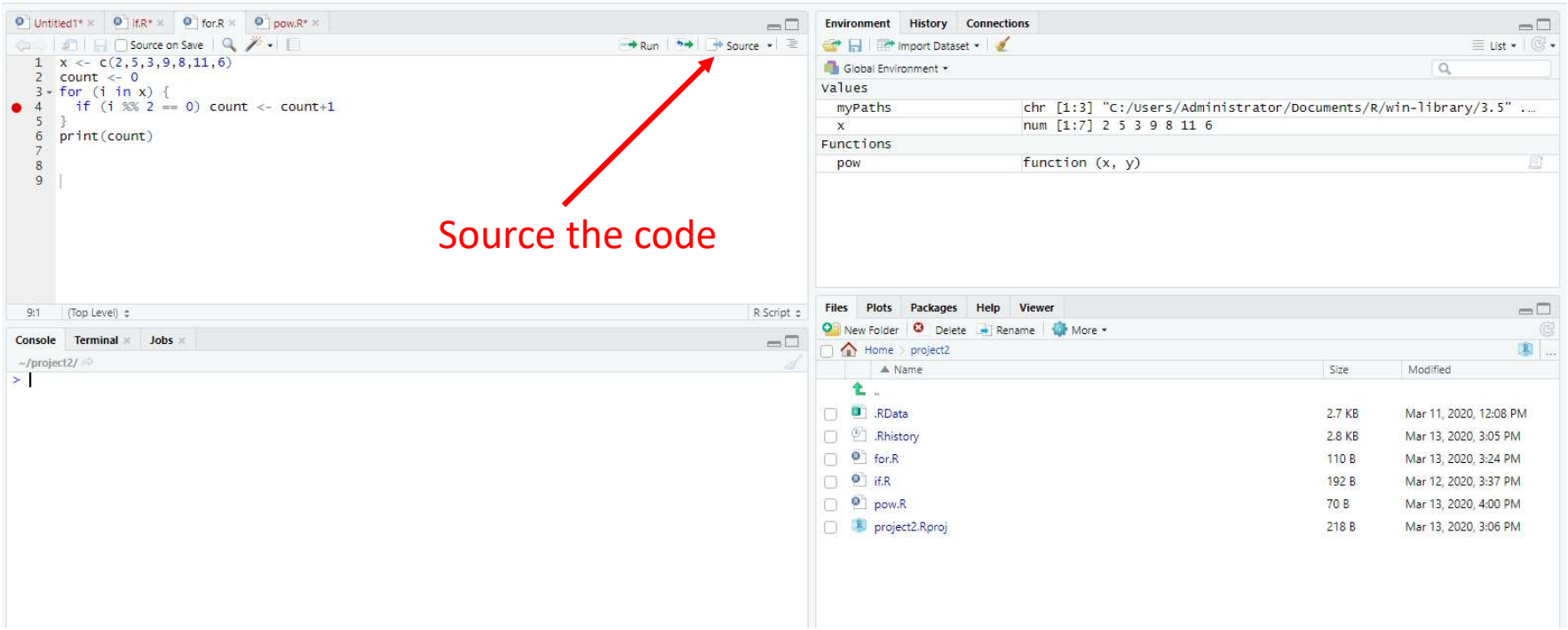
Ref. https://support.rstudio.com/hc/en-us/articles/205612627-Debugging-with-RStudio#introduction

Stopping when an Error Occurs

- RStudio halts execution at the point where the error is raised
- in menu “Debug” -> “On Error”, change the value from “Error Inspector” to “Break in Code”.

Ref. https://support.rstudio.com/hc/en-us/articles/205612627-Debugging-with-RStudio#introduction

Using the Debugger



Ref. https://support.rstudio.com/hc/en-us/articles/205612627-Debugging-with-RStudio#introduction

Using the Debugger

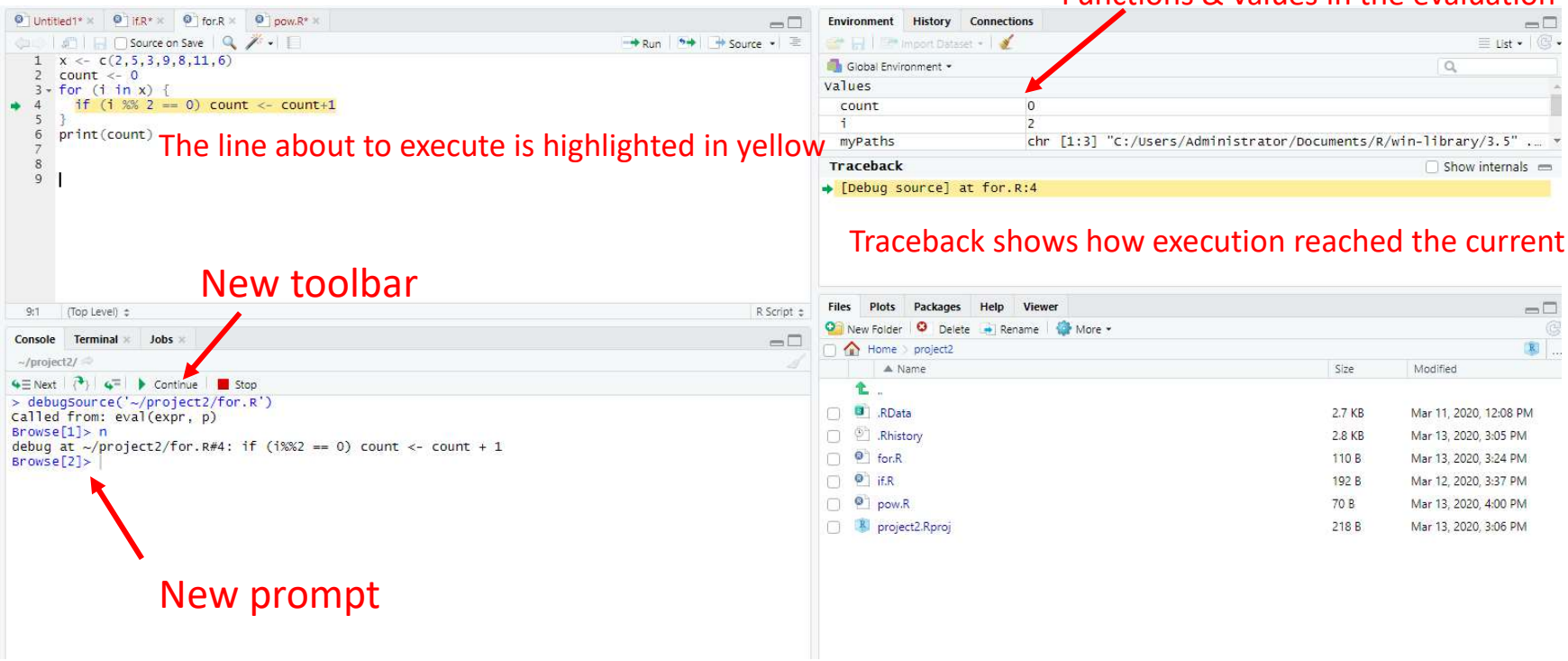
Functions & values in the evaluation

The line about to execute is highlighted in yellow

Traceback shows how execution reached the current point

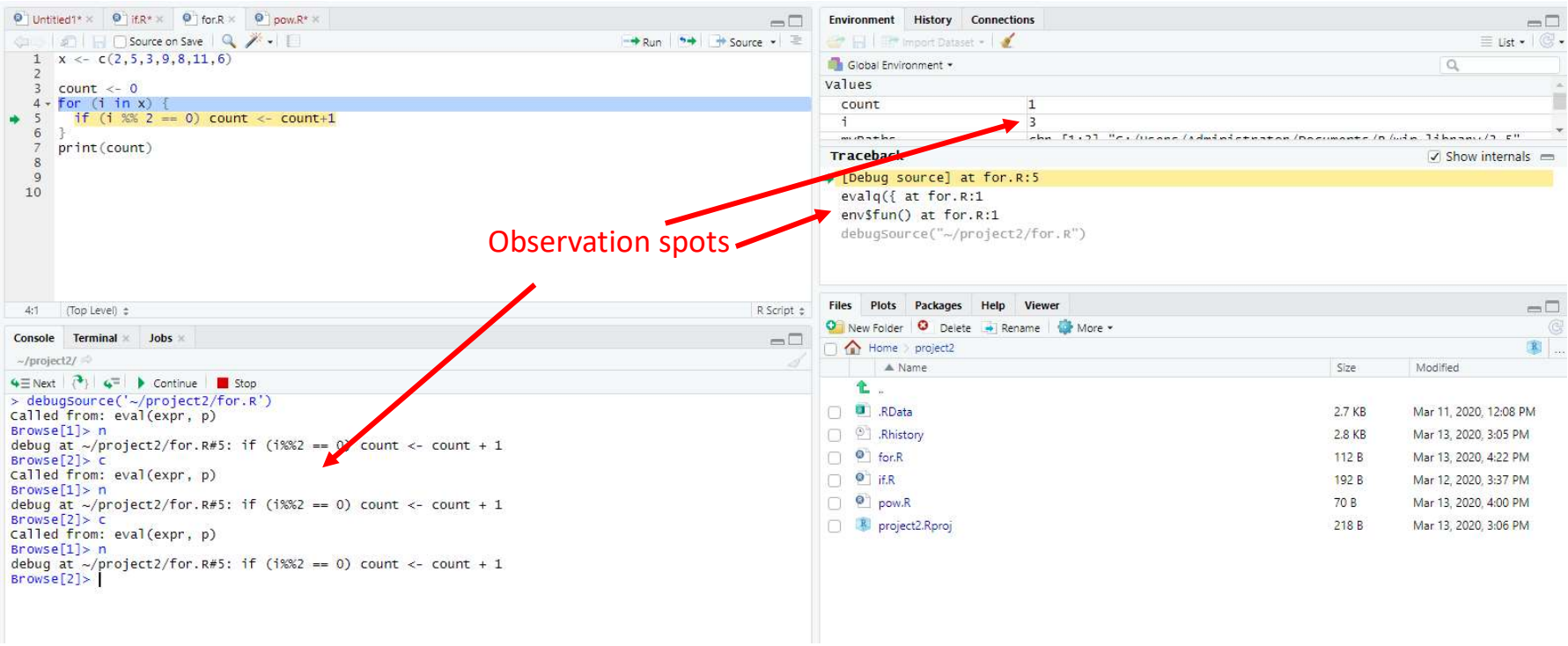
New toolbar

New prompt



Ref. https://support.rstudio.com/hc/en-us/articles/205612627-Debugging-with-RStudio#introduction

Using the Debugger



Ref. https://support.rstudio.com/hc/en-us/articles/205612627-Debugging-with-RStudio#introduction

Outline

- RStudio basics
  - R in PC and HPC
  - What is RStudio
  - RStudio IDE features
  - User environment
- Advanced features
  - Use Version Control with RStudio
  - Install and load R packages for advanced users
  - RStudio coding tools
  - Interactive graphics with ggvis and/or Shiny
  - Report Generation with R Markdown

Build interactive graphics for exploratory data analysis

- RStudio is easy to build interactive graphics for exploratory data analysis
  - RStudio includes a built-in browser so it can show the web graphics directly
- RStudio supports interactive graphics with Shiny and ggvis
  - ggvis

    https://ggvis.rstudio.com/ggvis-basics.html

    https://ggvis.rstudio.com/interactivity.html
  - Shiny (very detailed tutorials on RStudio's website)

    https://shiny.rstudio.com/

ggvis: interactive grammar of graphics

- ggvis has a similar underlying theory to ggplot2, but adds new features to make plots interactive
  - call to `ggvis()`

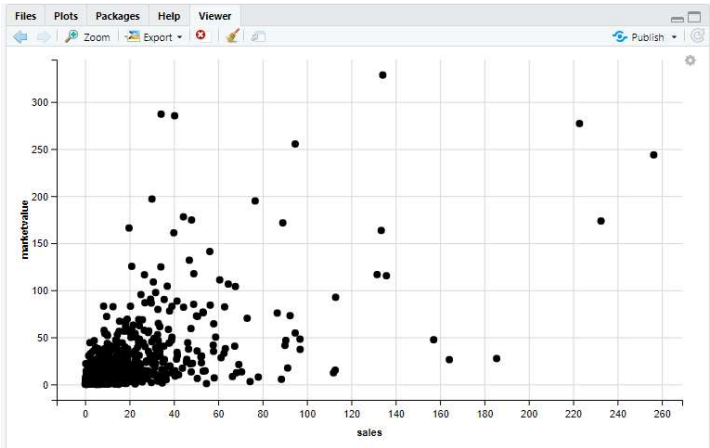
```
library(ggvis)
p <- ggvis(forbes, x = ~sales, y = ~marketvalue)
```

  - layer visual elements (by points)

```
layer_points(p)
```

  - rewrite with `%>%` (pronounced pipe) func:

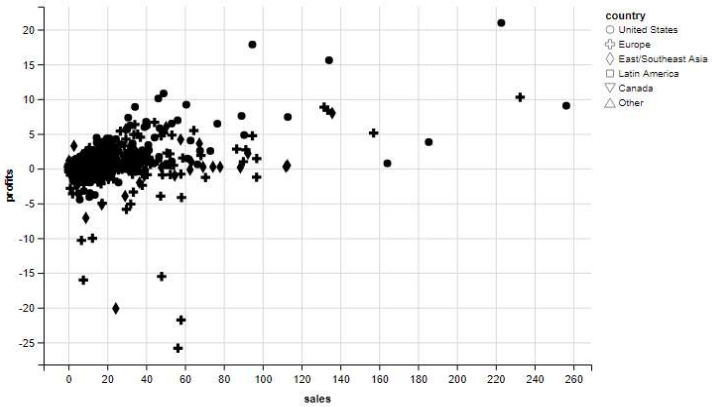
```
forbes %>%
  ggvis(~sales, ~marketvalue) %>%
  layer_points()
```



ggvis: interactive grammar of graphics

- Add more variables to the plot by mapping them to other visual properties
  - add `$country` to describe the shape (or fill, stroke, size)

```
forbes %>%
  ggvis(~sales, ~profits,shape=~country) %>%
  layer_points()
```



- Make the points a fixed property

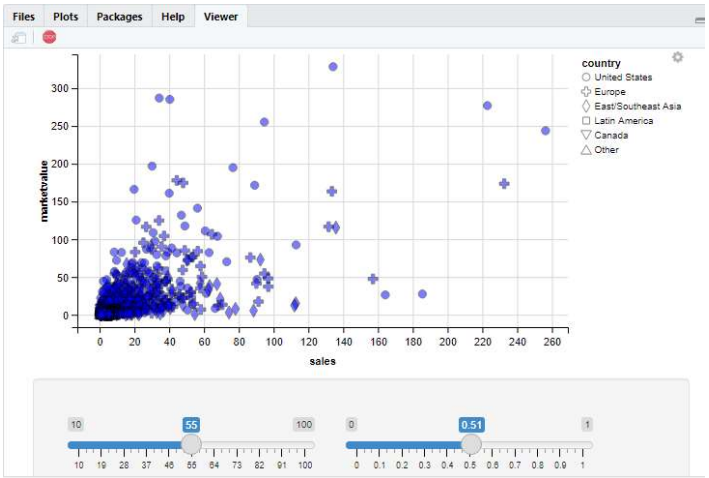
  (use `:=` instead of `=`)
  - red fill, black stroke and cross shape:

```
forbes %>%
  ggvis(~sales, ~marketvalue,fill:="red",stroke := "black",shape := "cross")
%>%
  layer_points()
```

ggvis: add interactivity

- Interactive plots are built with shiny
  - only have one running at a time in a given R session
- Add size and opacity sliders

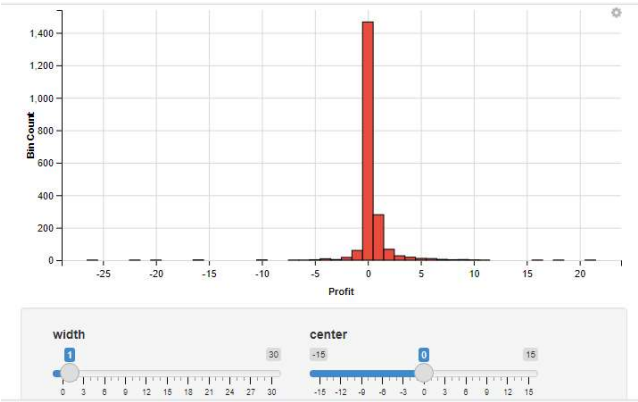
```
forbes %>%
  ggvis(x=~sales, y=~marketvalue,fill:="blue",stroke :=
"black",shape=~country,size := input_slider(10, 100),opacity :=
input_slider(0, 1)) %>%
  layer_points()
```



ggvis: add interactivity

- Layer visual elements by histograms with additional
  - fill up bins with red by using `fill :=`
  - label x and y axis

```
forbes %>%
  ggvis(~profits) %>%
  layer_histograms(width =  input_slider(0, 30, step = 1, label = "width"),
                   center = input_slider(-15, 15, step = 1, label =
"center"),fill:="#E74C3C") %>%
  add_axis("x", title = "Profit") %>%
  add_axis("y", title = "Bin Count")
```



ggvis: more interactive controls and layers

- Besides input_slider, ggvis provides more interactive controls:
  - input_checkbox(): a check-box
  - input_checkboxgroup(): a group of check boxes
  - input_numeric(): a spin box
  - input_radiobuttons(): pick one from a set options
  - input_select(): create a drop-down text box
  - input_text(): arbitrary text input
- Besides layer_points and layer_histograms, ggvis provides
  - Simple, which include primitives like points, lines and rectangles
  - Compound, which combine data transformations with one or more simple layers.

Outline

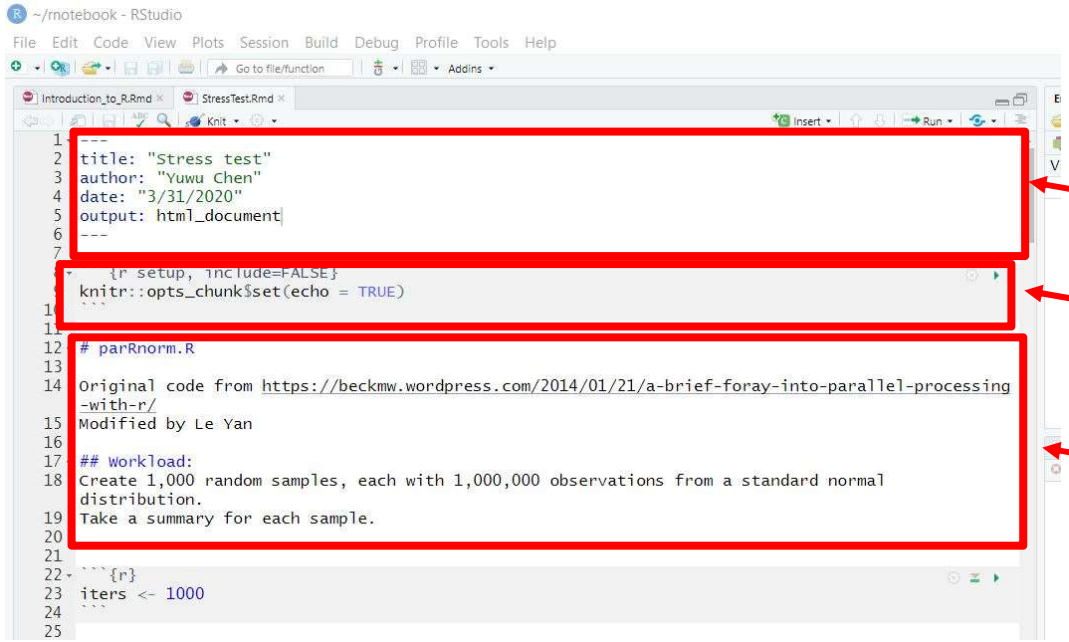
- RStudio basics
  - R in PC and HPC
  - What is RStudio
  - RStudio IDE features
  - User environment
- Advanced features
  - Use Version Control with RStudio
  - Install and load R packages for advanced users
  - RStudio coding tools
  - Interactive graphics with ggvis and/or Shiny
  - Report Generation with R Markdown

Generate report with R Markdown

- How R Markdown works
  - Weaves R code and human readable texts together into a plain text file that has the extension `.Rmd`
  - The rmarkdown package can convert `.Rmd` into documents of two types of output formats: documents, and presentations.
  - Also helps make your research reproducible

Generate report with R Markdown

- How `.Rmd` file looks like:



An (optional) YAML header surrounded by ---s

R code chunks surrounded by ```s

text mixed with simple text formatting

- The file above contains three types of content

Generate report with R Markdown

- R Markdown installation
  - R Markdown is free and open source

    ```
    $ install.packages("rmarkdown")
    ```

- Cheat sheet

  https://rstudio.com/wp-content/uploads/2016/03/rmarkdown-cheatsheet-2.0.pdf

- Reference guide

  https://rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf

Generate report with R Markdown

- Render `.Rmd` file

Take-home message

- What is R and RStudio
  - How to run R on PC and HPC clusters. When to consider to use HPC
- Why use RStudio
  - How to install RStudio
  - Basic IDE features, various panes
  - RStudio user environment: projects
- Advanced RStudio features
  - Version control structures
  - RStudio as a coding tool
  - Installing R packages with command lines into the desired locations
  - How to start interactive graphics with ggvis
  - Generating reports with R Markdown

Learning RStudio

- User documentation on RStudio
  - https://support.rstudio.com/hc/en-us
- Online tutorials (tons of them)
  - http://www.cyclismo.org/tutorial/R/
- Online courses (e.g. Coursera)
- Blogs
  - https://www.r-bloggers.com
- Educational R packages
  - Swirl: Learn R in R

More R Tutorials – Introduction to R

- R basics
  - What is R
  - How to run R codes
  - Basic syntax
  - Data classes and objects in R
- Flow control structures
- Statistical functions
- How to install and load R packages
- http://www.hpc.lsu.edu/training/archive/tutorials.php

More R Tutorials – Data Analysis in R

- Data analysis fundamentals with applications in R.
  - The data pre-processing
  - Basic statistical analysis methods such as linear regression, classification as well as re-sampling methods for the basic machine learning will be covered
- http://www.hpc.lsu.edu/training/archive/tutorials.php

More R Tutorials – Data Visualization in R

- This training provided an introduction to the R graphics in detail
- An overview on how to create and save graphs in R, then focus on the ggplot2 package.
- http://www.hpc.lsu.edu/training/archive/tutorials.php

More R Tutorials – Parallel Computing with R

- This training focused on how to use the "parallel" package in R and a few related packages to parallelize and enhance the performance of R programs
- http://www.hpc.lsu.edu/training/archive/tutorials.php

Next HPC Training

- Introduction to Python, March 17.
- Weekly trainings during regular semester
  - Wednesdays “9:00am-11:00am” session, Frey 307 CSC
- Programming/Parallel Programming workshops
  - Usually in summer

Getting Help

- User Guides
  - LSU HPC: http://www.hpc.lsu.edu/docs/guides.php#hpc
  - LONI:http://www.hpc.lsu.edu/docs/guides.php#loni
- Documentation: http://www.hpc.lsu.edu/docs
- Contact us
  - Email ticket system: sys-help@loni.org
  - Telephone Help Desk: 225-578-0900

Case Study: Forbes Fortune List

- The forbes dataset consists of 2000 rows (observations) describing companies' rank, name, country, category, sales, profits, assets and market value.

Getting Data

- Downloading files from internet
  - Manually download the file to the working directory
  - or with R function `download.file()`

```
> download.file("http://www.hpc.lsu.edu/training/weekly-materials/Downloads/Forbes2000.csv.zip", "Forbes2000.csv.zip")
> unzip("Forbes2000.csv.zip","Forbes2000.csv")
```

Reading and Writing Data

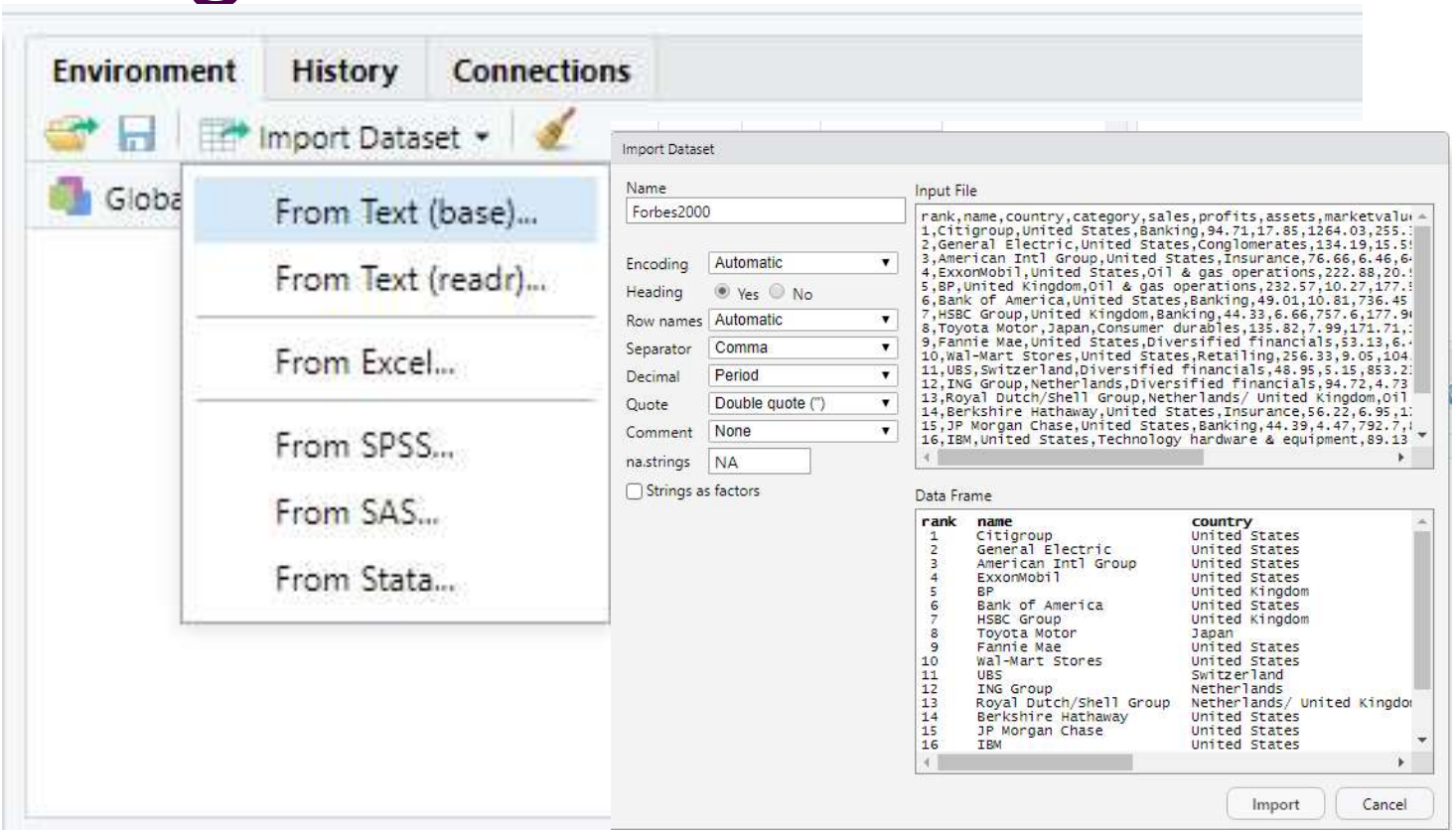
- R understands many different data formats and has lots of ways of reading/writing them (csv, xml, excel, sql, json etc.)

`read.table` `read.csv`	`write.table` `write.csv`	for reading/writing tabular data
`readLines`	`writeLines`	for reading/writing lines of a text file
`source`	`dump`	for reading/writing in R code files
`dget`	`dput`	for reading/writing in R code files
`load`	`save`	for reading in/saving workspaces

Reading Data with `read.csv`

```
# read.csv() is identical to read.table() except
that the default separator is a comma.
forbes <- read.csv("Forbes2000.csv",header=T,stringsAsFactors =
FALSE,na.strings ="NA",sep=",")
```

Reading Data in Environment Pane



Carefully choose the options of import

Steps for Data Analysis

- Get the data
- Read the data to R
- Inspect the data
- Preprocess the data (remove missing and dubious values, discard columns not needed etc.)
- Analyze the data
- Generate the report

More R Tutorials for Forbes Case Study

- Tutorials
  - Introduction to R
  - Data Analysis in R

http://www.hpc.lsu.edu/training/index.php