Data Visualization and Exploration with R

A practical guide to using R, RStudio, and Tidyverse for data visualization, exploration, and data science applications.

GeoSpatial Training Services

Eric Pimpler

Data Visualization and Exploration with R

A practical guide to using R, RStudio, and Tidyverse for data visualization, exploration, and data science applications.

Eric Pimpler

Introduction to R and RStudio

The R Project for Statistical Computing, or simply named R, is a free software environment for statistical computing and graphics. It is also a programming language that is widely used among statisticians and data miners for developing statistical software and data analysis. Over the last few years, they were joined by enterprises who discovered the potential of R, as well as technology vendors that offer R support or R-based products.

Although there are other programming languages for handling statistics, R has become the de facto language of statistical routines, offering a package repository with over 6400 problem-solving packages. It is also offers versatile and powerful plotting. It also has the advantage of treating tabular and multi-dimensional data as a labeled, indexed series of observations. This is a game changer over typical software which is just doing 2D layout, like Excel.

In this chapter we'll cover the following topics:

- Introduction to RStudio
- Creating variables and assigning data
- Using vectors and factors
- Using lists
- Using data classes
- Looping statements
- Decision support statements
- Using functions
- Introduction to tidyverse

Introduction to RStudio

There are a number of integrated development environments (IDE) that you can use to write R code including Visual Studio for R, Eclipse, R Console, and RStudio among others. You could also use a plain text editor as well. However, we're going to use RStudio for the exercises in this book. RStudio is a free, open source IDE for R. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management.

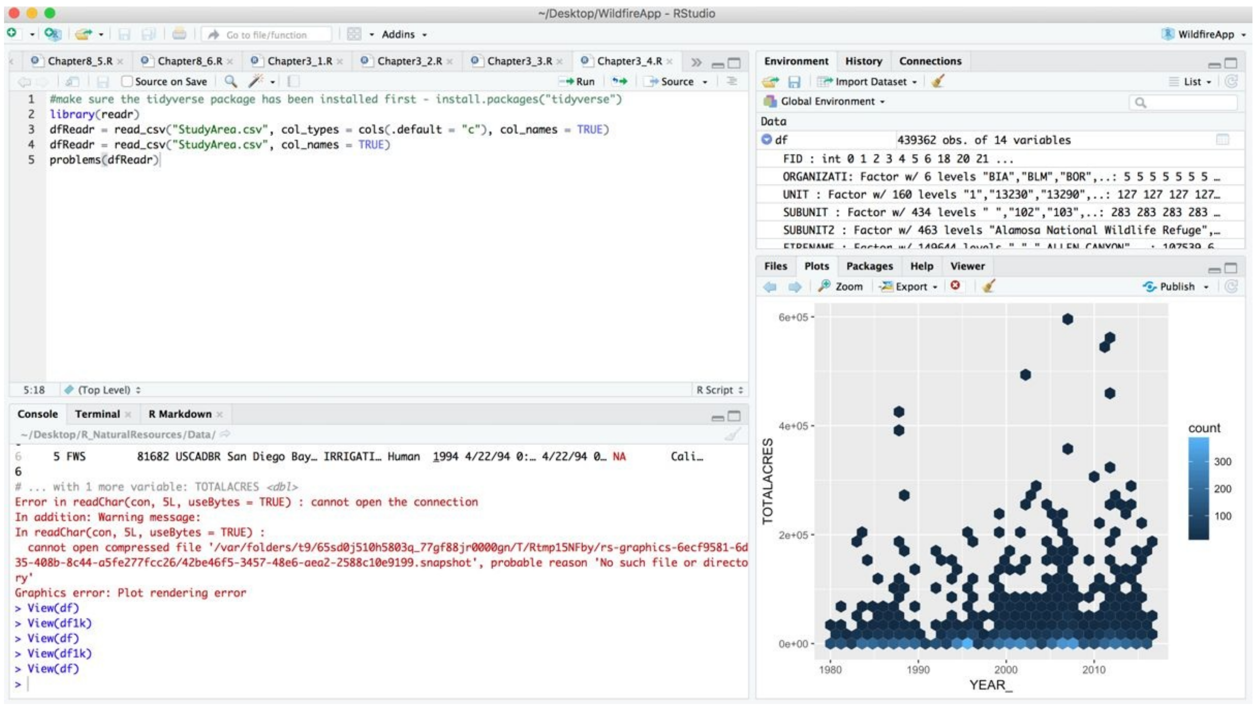
RStudio is available in open source and commercial editions and runs on the desktop (Windows, Mac, and Linux) or in a browser connected to RStudio Server or RStudio Server Pro (Debian/Ubuntu, RedHat/CentOS, and SUSE Linux).

Although there are many options for R development, we're going to use RStudio for the exercises in this book. You can get more information on RStudio at

https://www.rstudio.com/products/rstudio/

The RStudio Interface

The RStudio Interface, displayed in the screenshot below, looks quite complex initially, but when you break the interface down into sections it isn't so overwhelming. We'll cover much of the interface in the sections below. Keep in mind though that the interface is customizable so if you find the default interface isn't exactly what you like it can be changed. You'll learn how to customize the interface in a later section.



```
#make sure the tidyverse package has been installed first - install.packages("tidyverse")
library(readr)
dfReadr = read_csv("StudyArea.csv", col_types = cols(.default = "c"), col_names = TRUE)
dfReadr = read_csv("StudyArea.csv", col_names = TRUE)
problems(dfReadr)
```

```
df                439362 obs. of 14 variables
  FID : int 0 1 2 3 4 5 6 18 20 21 ...
  ORGANIZATI: Factor w/ 6 levels "BIA","BLM","BOR",..: 5 5 5 5 5 5 5 …
  UNIT : Factor w/ 160 levels "1","13230","13290",..: 127 127 127 127…
  SUBUNIT : Factor w/ 434 levels " ","102","103",..: 283 283 283 283 …
  SUBUNIT2 : Factor w/ 463 levels "Alamosa National Wildlife Refuge",…
```

```
6     5 FWS        81682 USCADBR San Diego Bay… IRRIGATI… Human  1994 4/22/94 0:… 4/22/94 0… NA        Cali…
6
# ... with 1 more variable: TOTALACRES <dbl>
Error in readChar(con, 5L, useBytes = TRUE) : cannot open the connection
In addition: Warning message:
In readChar(con, 5L, useBytes = TRUE) :
  cannot open compressed file '/var/folders/t9/65sd0j510h5803q_77gf88jr0000gn/T/Rtmp15NFby/rs-graphics-6ecf9581-6d
35-408b-8c44-a5fe277fcc26/42be46f5-3457-48e6-aea2-2588c10e9199.snapshot', probable reason 'No such file or directo
ry'
Graphics error: Plot rendering error
> View(df)
> View(df1k)
> View(df)
> View(df1k)
> View(df)
> 
```

To simplify the overview of RStudio we'll break the IDE into quadrants to make it easier to reference each component of the interface. The screenshot below illustrates each of the quadrants. We'll start with the panes in quadrant 1 and work through each of the quadrants.

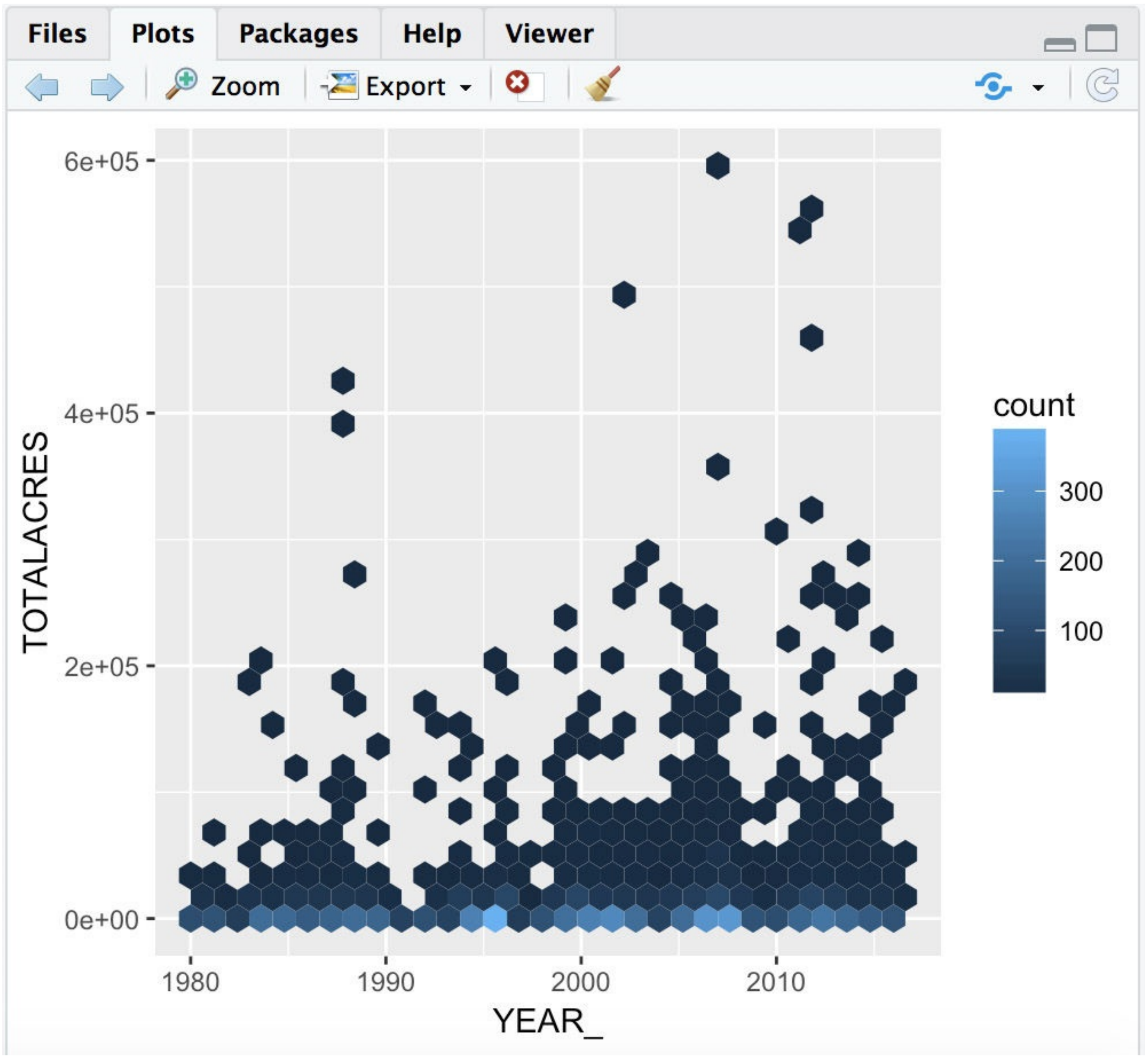
Files Pane – (Q1)

The **Files**pane functions like a file explorer similar to Windows Explorer on a Windows operating system or Finder on a Mac. This tab, displayed in the screenshot below, provides the following functionality:

1. Delete files and folders
2. Create new folders
3. Rename folders
4. Folder navigation
5. Copy or move files
6. Set working directory or go to working directory
7. View files
8. Import datasets

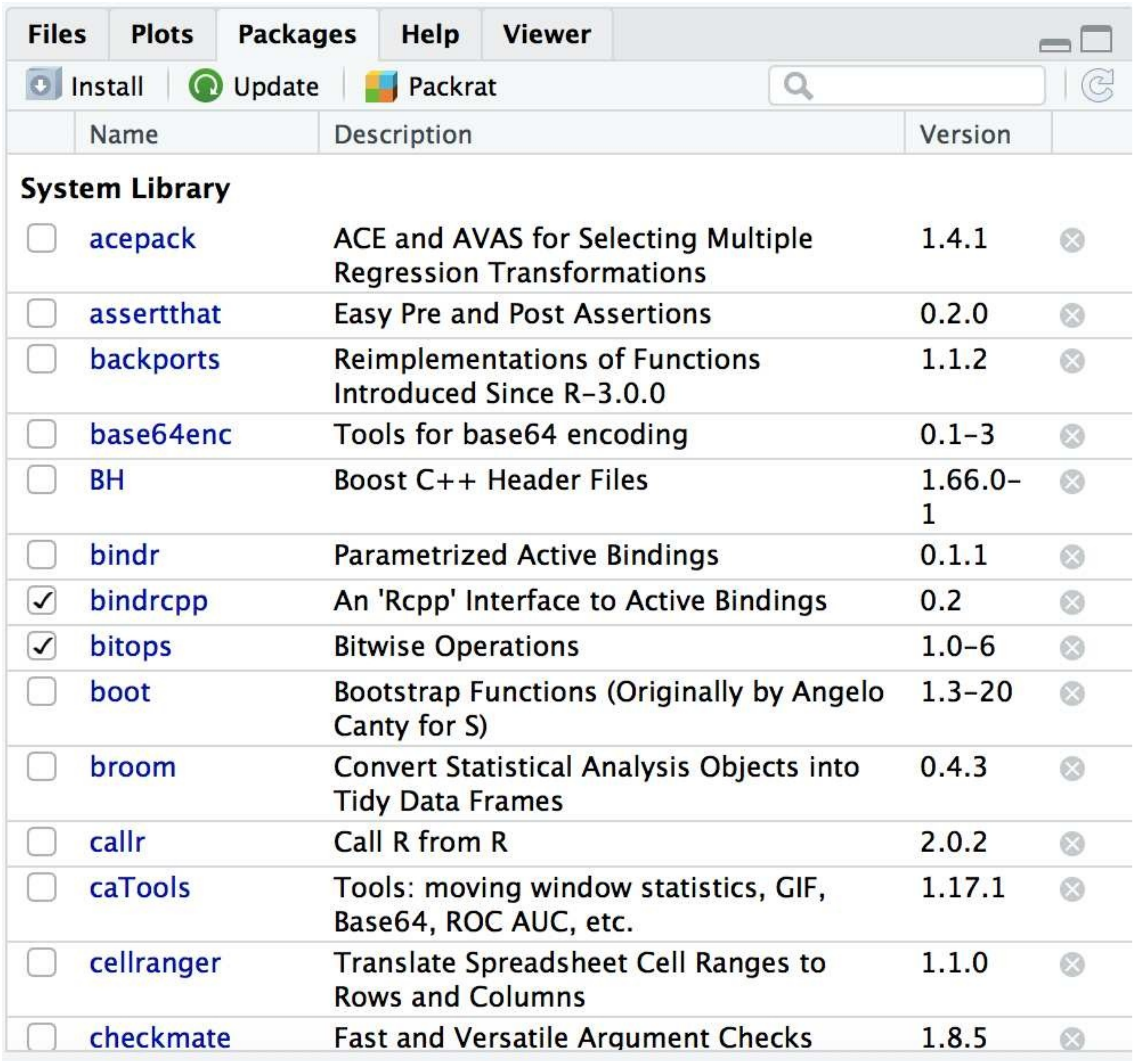
Plots Pane – (Q1)

The **Plots**pane, displayed in the screenshot below, is used to view output visualizations produced when typing code into the **Console** window or running a script. Plots can be created using a variety of different packages, but we'll primarily be using the ggplot2 package in this book. Once produced, you can zoom in, export as an image, or PDF, copy to the clipboard, and remove plots. You can also can navigate to previous and next plots.



Packages Pane – (Q1)

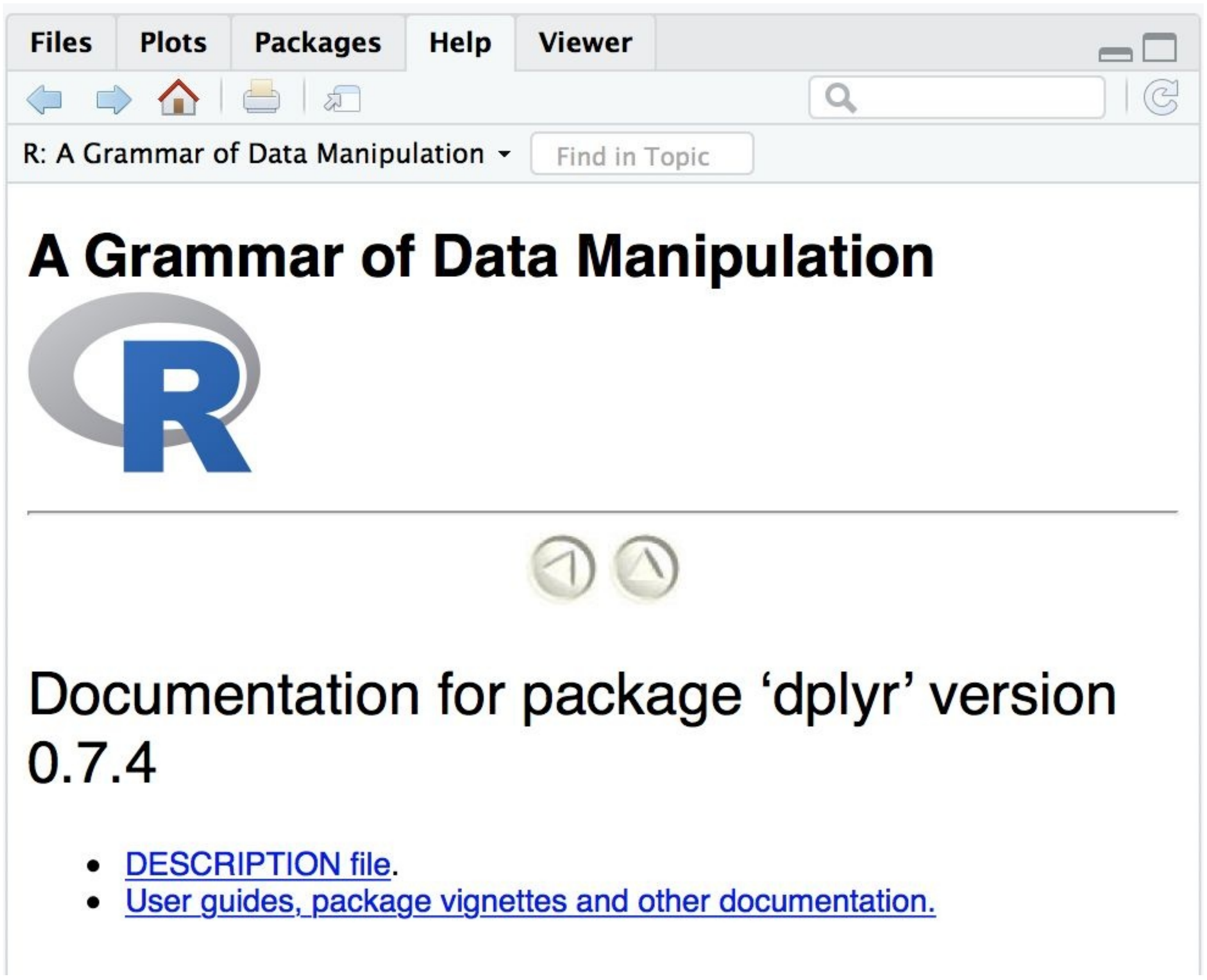
The **Packages** pane, shown in the screenshot below, displays all currently installed packages along with a brief description and version number for the package. Packages can also be removed using the x icon to the right of the version number for the package. Clicking on the package name will display the help file for the package in the **Help** tab. Clicking on the checkbox to the left of the package name loads the library so that it can be used when writing code in the **Console** window.



	Name	Description	Version	
System Library				
☐	acepack	ACE and AVAS for Selecting Multiple Regression Transformations	1.4.1	⊗
☐	assertthat	Easy Pre and Post Assertions	0.2.0	⊗
☐	backports	Reimplementations of Functions Introduced Since R-3.0.0	1.1.2	⊗
☐	base64enc	Tools for base64 encoding	0.1-3	⊗
☐	BH	Boost C++ Header Files	1.66.0-1	⊗
☐	bindr	Parametrized Active Bindings	0.1.1	⊗
✓	bindrcpp	An 'Rcpp' Interface to Active Bindings	0.2	⊗
✓	bitops	Bitwise Operations	1.0-6	⊗
☐	boot	Bootstrap Functions (Originally by Angelo Canty for S)	1.3-20	⊗
☐	broom	Convert Statistical Analysis Objects into Tidy Data Frames	0.4.3	⊗
☐	callr	Call R from R	2.0.2	⊗
☐	caTools	Tools: moving window statistics, GIF, Base64, ROC AUC, etc.	1.17.1	⊗
☐	cellranger	Translate Spreadsheet Cell Ranges to Rows and Columns	1.1.0	⊗
☐	checkmate	Fast and Versatile Argument Checks	1.8.5	⊗

Help Pane – (Q1)

The **Help** pane, shown in the screenshot below, displays linked help documentation for any packages that you have installed.



A Grammar of Data Manipulation

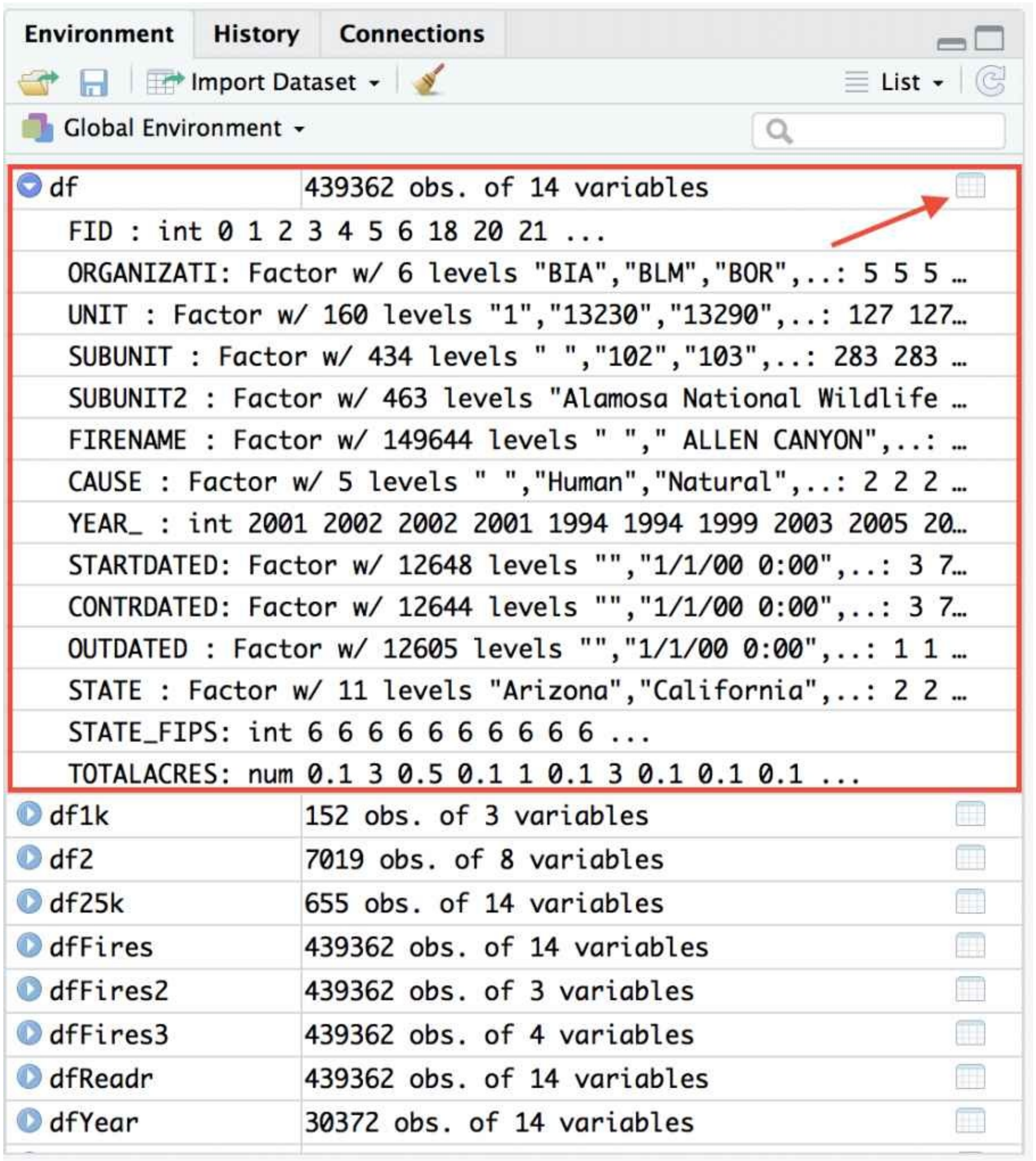
Documentation for package ‘dplyr’ version 0.7.4

- DESCRIPTION file.
- User guides, package vignettes and other documentation.

Viewer Pane – (Q1)
RStudio includes a **Viewer**pane that can be used to view local web content. For example, web graphics generated using packages like googleVis, htmlwidgets, and RCharts, or even a local web application created with Shiny. However, keep in mind that the Viewer pane can only be used for local web content in the form of static HTML pages written in the session’s temporary directory or a locally run web application. The Viewer pane can’t be used to view online content.

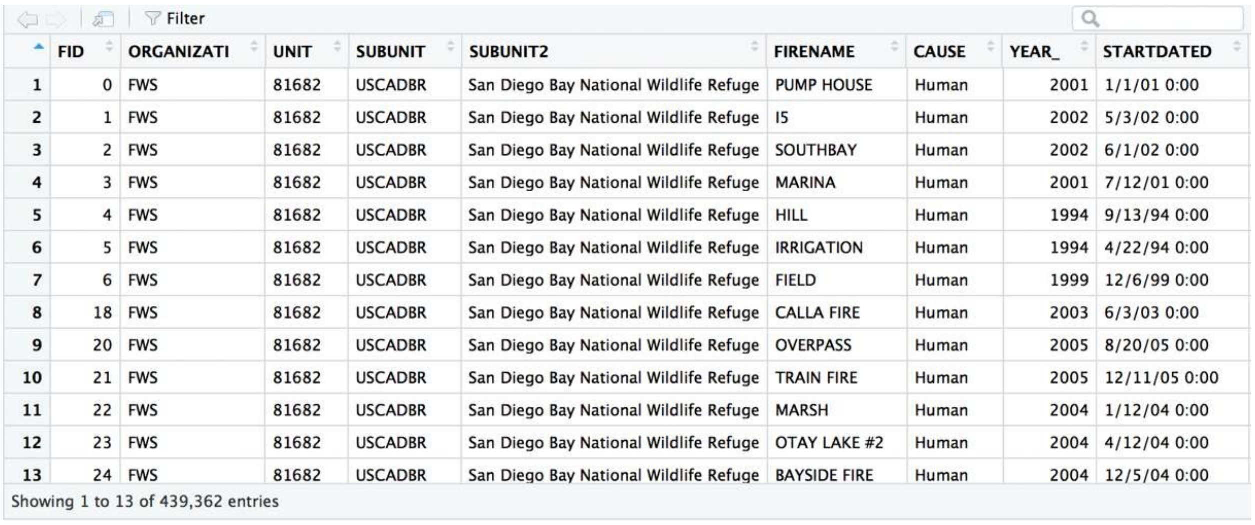
Environment Pane – (Q2)

The **Environment** pane contains a listing of variables that you have created for the current session. Each variable is listed in the tab and can be expanded to view the contents of the variable. You can see an example of this in the screenshot below by taking a look at the df variable. The rectangle surrounding the df variable displays the columns for the variable.



```
df          439362 obs. of 14 variables
  FID : int 0 1 2 3 4 5 6 18 20 21 ...
  ORGANIZATI: Factor w/ 6 levels "BIA","BLM","BOR",..: 5 5 5 …
  UNIT : Factor w/ 160 levels "1","13230","13290",..: 127 127…
  SUBUNIT : Factor w/ 434 levels " ","102","103",..: 283 283 …
  SUBUNIT2 : Factor w/ 463 levels "Alamosa National Wildlife …
  FIRENAME : Factor w/ 149644 levels " "," ALLEN CANYON",..: …
  CAUSE : Factor w/ 5 levels " ","Human","Natural",..: 2 2 2 …
  YEAR_ : int 2001 2002 2002 2001 1994 1994 1999 2003 2005 20…
  STARTDATED: Factor w/ 12648 levels "","1/1/00 0:00",..: 3 7…
  CONTRDATED: Factor w/ 12644 levels "","1/1/00 0:00",..: 3 7…
  OUTDATED : Factor w/ 12605 levels "","1/1/00 0:00",..: 1 1 …
  STATE : Factor w/ 11 levels "Arizona","California",..: 2 2 …
  STATE_FIPS: int 6 6 6 6 6 6 6 6 6 6 ...
  TOTALACRES: num 0.1 3 0.5 0.1 1 0.1 3 0.1 0.1 0.1 ...
df1k        152 obs. of 3 variables
df2         7019 obs. of 8 variables
df25k       655 obs. of 14 variables
dfFires     439362 obs. of 14 variables
dfFires2    439362 obs. of 3 variables
dfFires3    439362 obs. of 4 variables
dfReadr     439362 obs. of 14 variables
dfYear      30372 obs. of 14 variables
```

Clicking the table icon on the far-right side of the display (highlighted with the arrow in the screenshot above) will open the data in a tabular viewer as seen in the screenshot below.



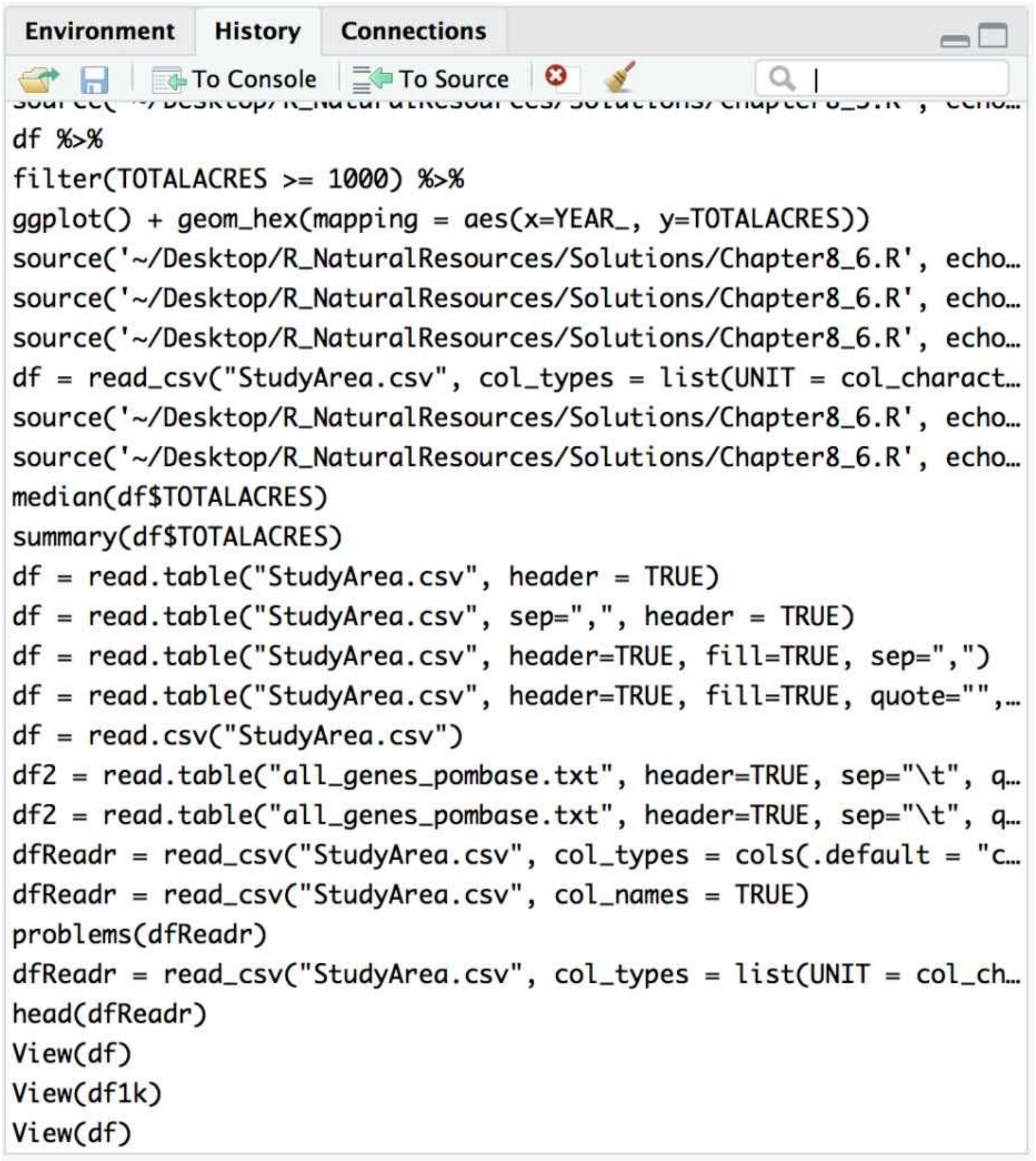
	FID	ORGANIZATI	UNIT	SUBUNIT	SUBUNIT2	FIRENAME	CAUSE	YEAR_	STARTDATED
1	0	FWS	81682	USCADBR	San Diego Bay National Wildlife Refuge	PUMP HOUSE	Human	2001	1/1/01 0:00
2	1	FWS	81682	USCADBR	San Diego Bay National Wildlife Refuge	I5	Human	2002	5/3/02 0:00
3	2	FWS	81682	USCADBR	San Diego Bay National Wildlife Refuge	SOUTHBAY	Human	2002	6/1/02 0:00
4	3	FWS	81682	USCADBR	San Diego Bay National Wildlife Refuge	MARINA	Human	2001	7/12/01 0:00
5	4	FWS	81682	USCADBR	San Diego Bay National Wildlife Refuge	HILL	Human	1994	9/13/94 0:00
6	5	FWS	81682	USCADBR	San Diego Bay National Wildlife Refuge	IRRIGATION	Human	1994	4/22/94 0:00
7	6	FWS	81682	USCADBR	San Diego Bay National Wildlife Refuge	FIELD	Human	1999	12/6/99 0:00
8	18	FWS	81682	USCADBR	San Diego Bay National Wildlife Refuge	CALLA FIRE	Human	2003	6/3/03 0:00
9	20	FWS	81682	USCADBR	San Diego Bay National Wildlife Refuge	OVERPASS	Human	2005	8/20/05 0:00
10	21	FWS	81682	USCADBR	San Diego Bay National Wildlife Refuge	TRAIN FIRE	Human	2005	12/11/05 0:00
11	22	FWS	81682	USCADBR	San Diego Bay National Wildlife Refuge	MARSH	Human	2004	1/12/04 0:00
12	23	FWS	81682	USCADBR	San Diego Bay National Wildlife Refuge	OTAY LAKE #2	Human	2004	4/12/04 0:00
13	24	FWS	81682	USCADBR	San Diego Bay National Wildlife Refuge	BAYSIDE FIRE	Human	2004	12/5/04 0:00

Showing 1 to 13 of 439,362 entries

Other functionality provided by the **Environment** pane includes opening or saving a workspace, importing dataset from text files, Excel spreadsheets, and various statistical package formats. You can also clear the current workspace.

History Pane – (Q2)

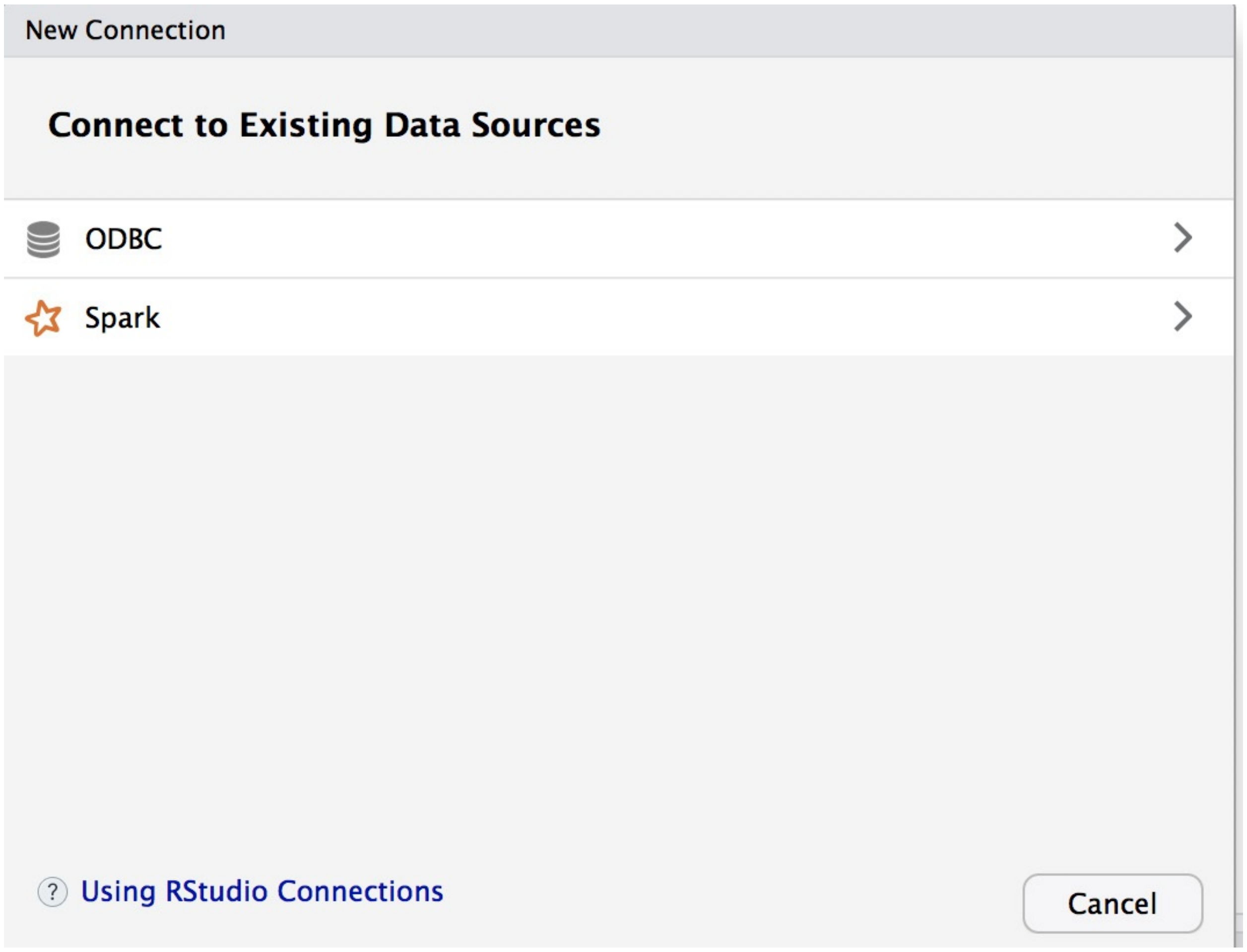
The **History** pane, shown in the screenshot below, displays a list of all commands that have been executed in the current session. This tab includes a number of useful functions including the ability to save these commands to a file or load historical commands from an existing file. You can also select specific commands from the History tab and send them directly to the console or an open script. You can also remove items from the **History** pane.



```
df %>%
filter(TOTALACRES >= 1000) %>%
ggplot() + geom_hex(mapping = aes(x=YEAR_, y=TOTALACRES))
source('~/Desktop/R_NaturalResources/Solutions/Chapter8_6.R', echo…
source('~/Desktop/R_NaturalResources/Solutions/Chapter8_6.R', echo…
source('~/Desktop/R_NaturalResources/Solutions/Chapter8_6.R', echo…
df = read_csv("StudyArea.csv", col_types = list(UNIT = col_charact…
source('~/Desktop/R_NaturalResources/Solutions/Chapter8_6.R', echo…
source('~/Desktop/R_NaturalResources/Solutions/Chapter8_6.R', echo…
median(df$TOTALACRES)
summary(df$TOTALACRES)
df = read.table("StudyArea.csv", header = TRUE)
df = read.table("StudyArea.csv", sep=",", header = TRUE)
df = read.table("StudyArea.csv", header=TRUE, fill=TRUE, sep=",")
df = read.table("StudyArea.csv", header=TRUE, fill=TRUE, quote="",…
df = read.csv("StudyArea.csv")
df2 = read.table("all_genes_pombase.txt", header=TRUE, sep="\t", q…
df2 = read.table("all_genes_pombase.txt", header=TRUE, sep="\t", q…
dfReadr = read_csv("StudyArea.csv", col_types = cols(.default = "c…
dfReadr = read_csv("StudyArea.csv", col_names = TRUE)
problems(dfReadr)
dfReadr = read_csv("StudyArea.csv", col_types = list(UNIT = col_ch…
head(dfReadr)
View(df)
View(df1k)
View(df)
```

Connections Pane – (Q2)

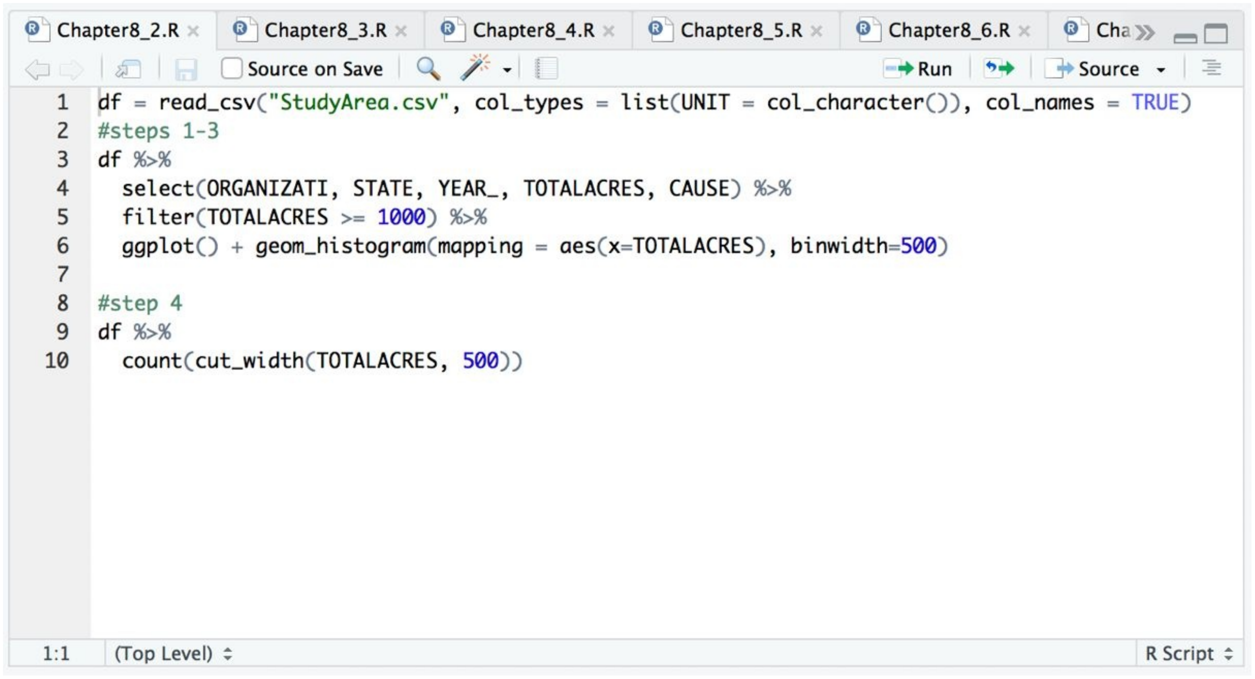
The **Connections**tab can be used to access existing or create new connections to ODBC and Spark data sources.



Source Pane – (Q3)

The **Source**pane in RStudio, seen in the screenshot below, is used to create scripts, and display datasets An R script is simply a text file containing a series of commands that are executed together. Commands can also be written line by line from the **Console** pane as well. When written from the **Console**pane, each line of code is executed when you click the Enter (Return) key. However, scripts are executed as a group.

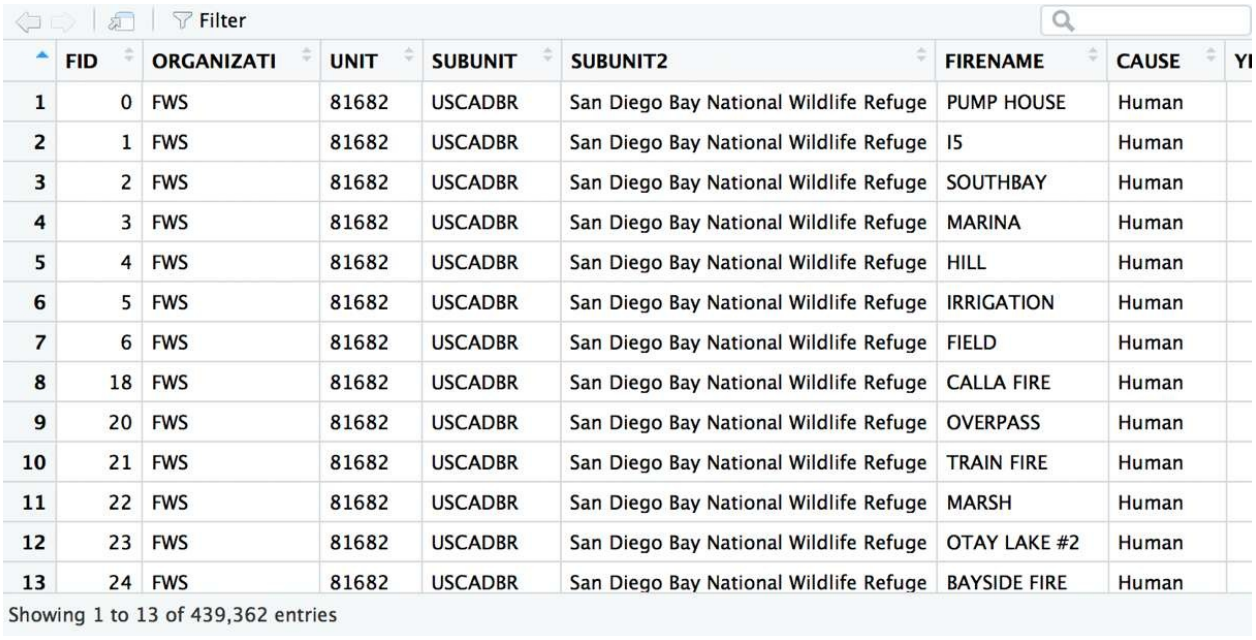
Multiple scripts can be open at the same time with each script occupying a separate tab as seen in the screenshot. RStudio provides the ability to execute the entire script, only the current line, or a highlighted group of lines. This gives you a lot of control over the execution the code in a script.



```
df = read_csv("StudyArea.csv", col_types = list(UNIT = col_character()), col_names = TRUE)
#steps 1-3
df %>%
  select(ORGANIZATI, STATE, YEAR_, TOTALACRES, CAUSE) %>%
  filter(TOTALACRES >= 1000) %>%
  ggplot() + geom_histogram(mapping = aes(x=TOTALACRES), binwidth=500)

#step 4
df %>%
  count(cut_width(TOTALACRES, 500))
```

The **Source** pane can also be used to display datasets. In the screenshot below, a data frame is displayed. Data frames can be displayed in this manner by calling the View(<data frame>) function.

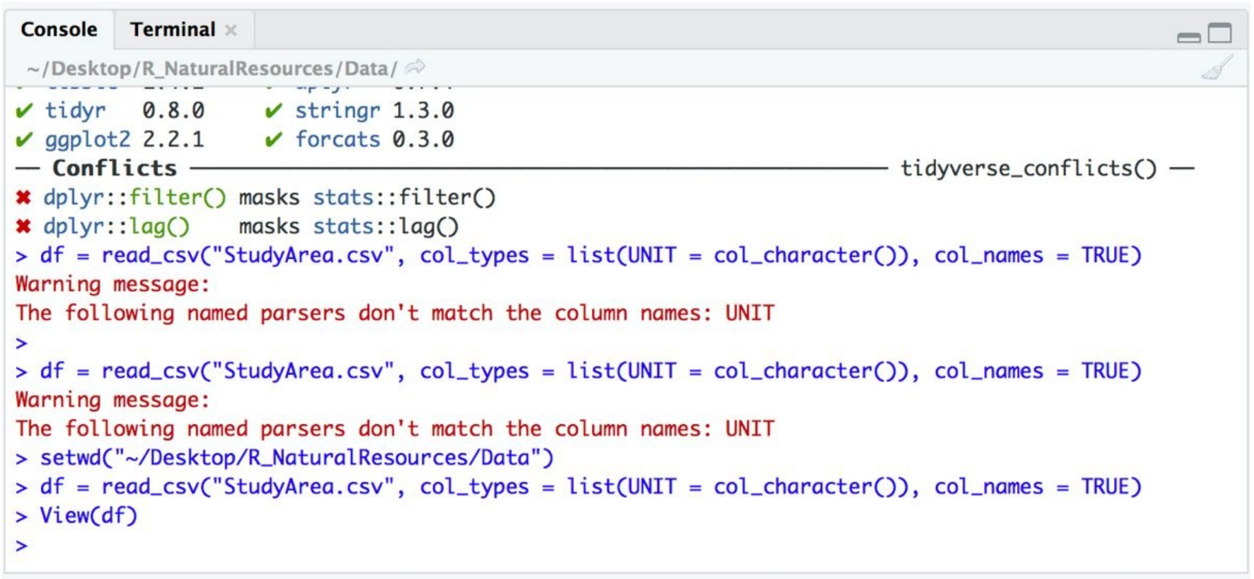


	FID	ORGANIZATI	UNIT	SUBUNIT	SUBUNIT2	FIRENAME	CAUSE	YI
1	0	FWS	81682	USCADBR	San Diego Bay National Wildlife Refuge	PUMP HOUSE	Human	
2	1	FWS	81682	USCADBR	San Diego Bay National Wildlife Refuge	I5	Human	
3	2	FWS	81682	USCADBR	San Diego Bay National Wildlife Refuge	SOUTHBAY	Human	
4	3	FWS	81682	USCADBR	San Diego Bay National Wildlife Refuge	MARINA	Human	
5	4	FWS	81682	USCADBR	San Diego Bay National Wildlife Refuge	HILL	Human	
6	5	FWS	81682	USCADBR	San Diego Bay National Wildlife Refuge	IRRIGATION	Human	
7	6	FWS	81682	USCADBR	San Diego Bay National Wildlife Refuge	FIELD	Human	
8	18	FWS	81682	USCADBR	San Diego Bay National Wildlife Refuge	CALLA FIRE	Human	
9	20	FWS	81682	USCADBR	San Diego Bay National Wildlife Refuge	OVERPASS	Human	
10	21	FWS	81682	USCADBR	San Diego Bay National Wildlife Refuge	TRAIN FIRE	Human	
11	22	FWS	81682	USCADBR	San Diego Bay National Wildlife Refuge	MARSH	Human	
12	23	FWS	81682	USCADBR	San Diego Bay National Wildlife Refuge	OTAY LAKE #2	Human	
13	24	FWS	81682	USCADBR	San Diego Bay National Wildlife Refuge	BAYSIDE FIRE	Human	

Showing 1 to 13 of 439,362 entries

Console Pane – (Q4)

The **Console**pane in RStudio is used to interactively write and run lines of code. Each time you enter a line of code and click **Enter**(Return) it will execute that line of code. Any warning or error messages will be displayed in the **Console** window as well as output from **print()** statements.



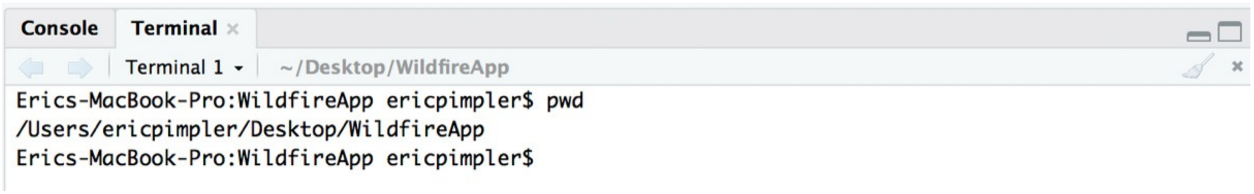
```
✔ tidyr   0.8.0     ✔ stringr 1.3.0
✔ ggplot2 2.2.1     ✔ forcats 0.3.0
— Conflicts ——————————————————————————————————— tidyverse_conflicts() —
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
> df = read_csv("StudyArea.csv", col_types = list(UNIT = col_character()), col_names = TRUE)
Warning message:
The following named parsers don't match the column names: UNIT
>
> df = read_csv("StudyArea.csv", col_types = list(UNIT = col_character()), col_names = TRUE)
Warning message:
The following named parsers don't match the column names: UNIT
> setwd("~/Desktop/R_NaturalResources/Data")
> df = read_csv("StudyArea.csv", col_types = list(UNIT = col_character()), col_names = TRUE)
> View(df)
>
```

Terminal Pane – (Q4)

The RStudio **Terminal**pane provides access to the system shell from within the RStudio IDE. It supports xterm emulation, enabling use of full-screen terminal applications (e.g. text editors, terminal multiplexers) as well as regular command-line operations with lineediting and shell history.

There are many potential uses of the shell including advanced source control operations, execution of long-running jobs, remote logins, and system administration of RStudio.

The **Terminal**pane is unlike most of the other features found in RStudio in that it’s capabilities are platform specific. In general, these differences can be categorized as either Windows capabilities or other (Mac, Linux, RStudio Server).

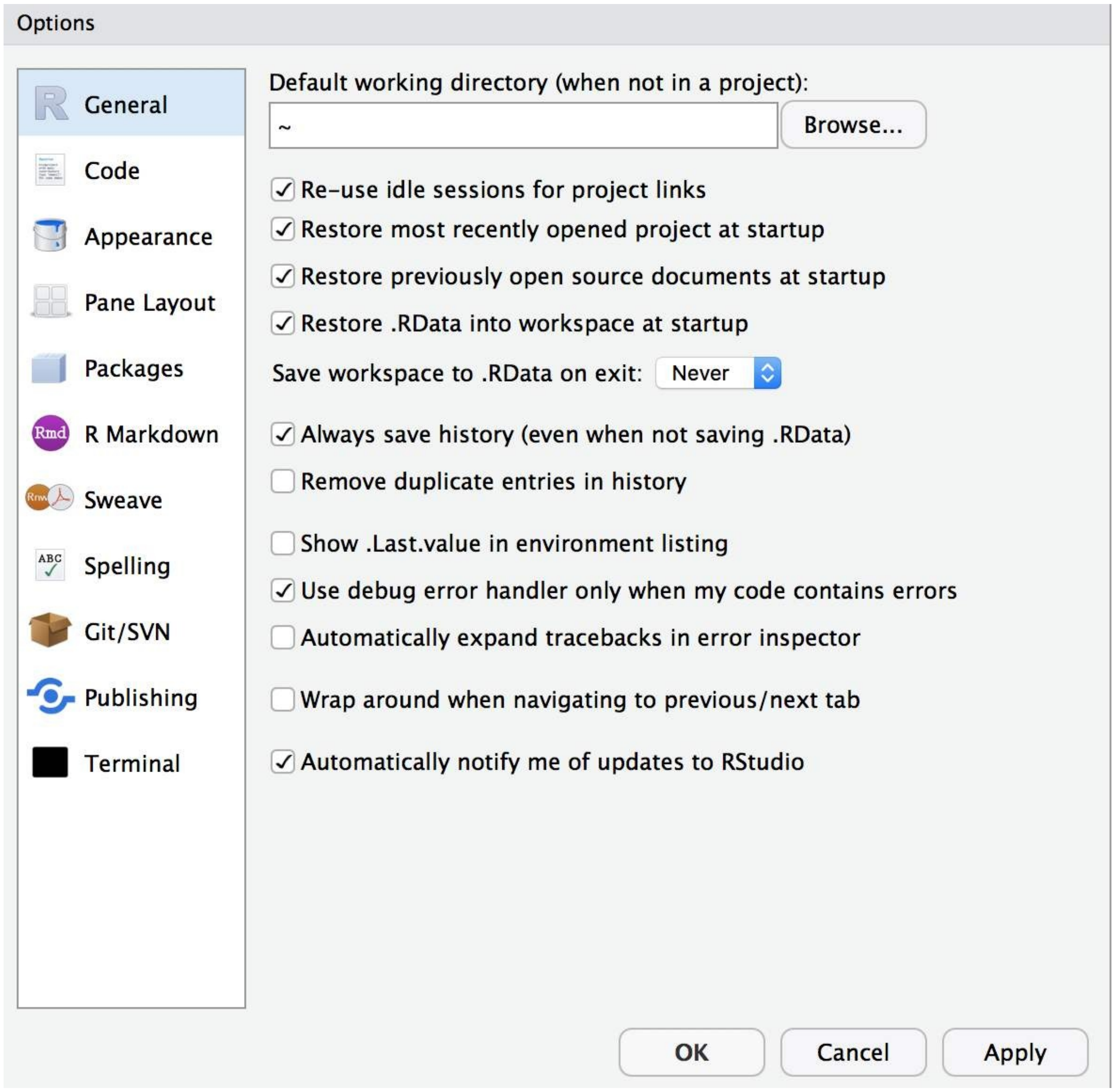


```
Erics-MacBook-Pro:WildfireApp ericpimpler$ pwd
/Users/ericpimpler/Desktop/WildfireApp
Erics-MacBook-Pro:WildfireApp ericpimpler$
```

Customizing the Interface

If you don’t like the default RStudio interface, you can customize the appearance. To do so, go to **Tool | Options** (**RStudio | Preferences**on a Mac).

The dialog seen in the screenshot below will be displayed.



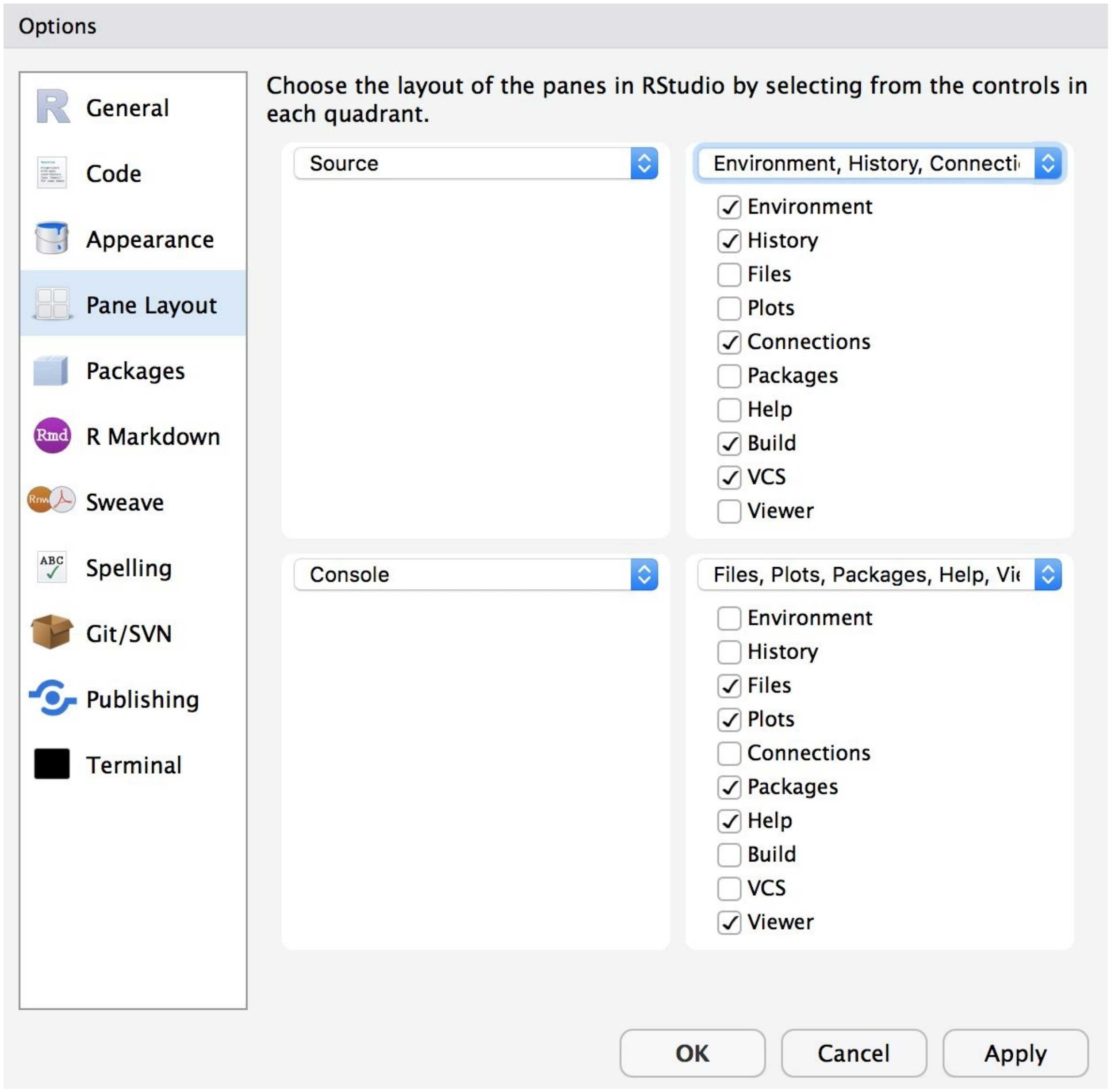
Default working directory (when not in a project): ~

- ✓ 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: Never

- ✓ 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
- ☐ Wrap around when navigating to previous/next tab
- ✓ Automatically notify me of updates to RStudio

The **Pane Layout** tab is used to change the locations of console, source editor, and tab panes, and set which tabs are included in each pane.


Options

General
Code
Appearance
Pane Layout
Packages
R Markdown
Sweave
Spelling
Git/SVN
Publishing
Terminal

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

Menu Options

There are also a multitude of options that can be accessed from the RStudio menu items as well. Covering these items in depth is beyond the scope of this book, but in general here are some of the more useful functions that can be accessed through the menus.

1. Create new files and projects
2. Import datasets
3. Hide, show, and zoom in and out of panes
4. Work with plots (save, zoom, clear)

5. Set the working directory
6. Save and load workspace
7. Start a new session
8. Debugging tools
9. Profiling tools
10. Install packages
11. Access help system

You’ll learn how to use various components of the RStudio interface as we move through the exercises in the book.

Installing RStudio

If you haven’t already done so, now is a good time to download and install RStudio. There are a number of versions of RStudio, including a free open source version which will be sufficient for this book. Versions are also available for various operating systems including Windows, Mac, and Linux.

1. Go to https://www.rstudio.com/products/rstudio/download/ find RStudio for Desktop, the Open Source License version, and follow in the instructions to download and install the software.
In the next section we’ll explore the basic programming constructs of the R language including the creation and assigning of data to variables, as well as the data types and objects that can be assigned to variables.

Installing the Exercise Data

This is intended as a hands-on exercise book and is designed to give you as much handson coding experience with R as possible. Many of the exercises in this book require that you load data from a file-based data source such as a CSV file. These files will need to be installed on your computer before continuing with the exercises in this chapter as well as the rest of the book. Please follow the instructions below to download and install the exercise data.

1. In a web browser go to
https://www.dropbox.com/s/5p7j7nl8hgijsnx/ IntroR.zip?dl=0.
2. This will download a file called IntroR.zip.

3. The exercise data can be unzipped to any location on your computer. After unzipping the IntroR.zip file you will have a folder structure that includes IntroR

as the top-most folder with sub-folders called Data and Solutions. The Data folder contains the data that will be used in the exercises in the book, while the Solutions folder contains solution files for the R script that you will write.

RStudio can be used on Windows, Mac, or Linux so rather than specifying a specific folder to place the data I will leave the installation location up to you. Just remember where you unzip the data because you'll need to reference the location when you set the working directory.

4. For reference purposes I have installed the data to the desktop of my Mac computer under IntroR\Data. You will see this location referenced at various locations throughout the book. However, keep in mind that you can install the data anywhere.

Exercise 1: Creating variables and assigning data

In the R programming language, like other languages, variables are given a name and assigned data. Each variable has a name that represents its area in memory. In R, variables are case sensitive so use care in naming your variable and referring to them later in your code.

There are two ways that variables can be assigned in R. In the first code example below, a variable named x is created. The use of a less than sign immediately followed by a dash then precedes the variable name. This is the operator used to assign data to a variable in R. On the right-hand side of this operator is the value being assign to the variable. In this case, the value 10 has been assigned to the variable x. To print the value of a variable in R you can simple type the variable name and then click the **Enter** key on your keyboard.

```
x <- 10
x
[1] 10
```

The other way of creating and assigning data to a variable is to use the equal sign. In the second code example we create a variable called y and assign the value 10 to the variable. This second method of creating and assigning data to a variable is probably more familiar to you if you've used other languages like Python or JavaScript.