R Bootcamp

Ashley I Naimi, PhD
Associate Professor
Emory University

✉ ashley.naimi@emory.edu
@ashley_naimi
ainaimi

EMORY | ROLLINS SCHOOL OF PUBLIC HEALTH

Acknowledgements

Some of this material was taken from:

- Chris Paciorek's bootcamp
- Garret Grolemund's tidyverse, visualization, and manipulation basics
- RStudio Cheat Sheets

The NHEFS Data were taken from:

- Miguel Hernán and Jamie Robins' Causal Inference Book

A comprehensive list of additional resources:

- R Studio Training
- R Studio Webinars
- See also last slide of this presentation

What is R?

Environment for statistical computing and graphics

Relatively simple **programming language**

C is a compiled language

- Requires a complete program to run
- Fast

R is an interpreted language

- Commands run interactively
- Flexible but slow

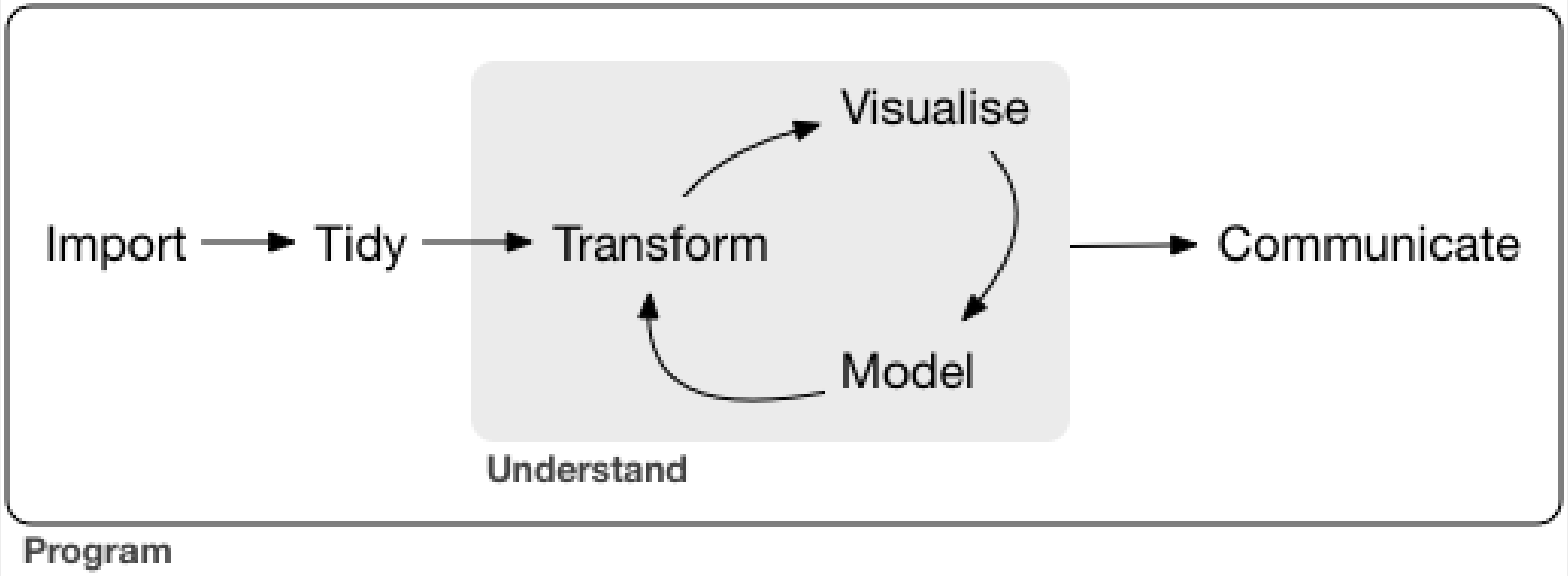
If C is faster, why use R?

- Over 10,000 Function Libraries
- Implements many common statistical procedures
- Exceptional graphics functions

How does R compare to SAS/Stata?

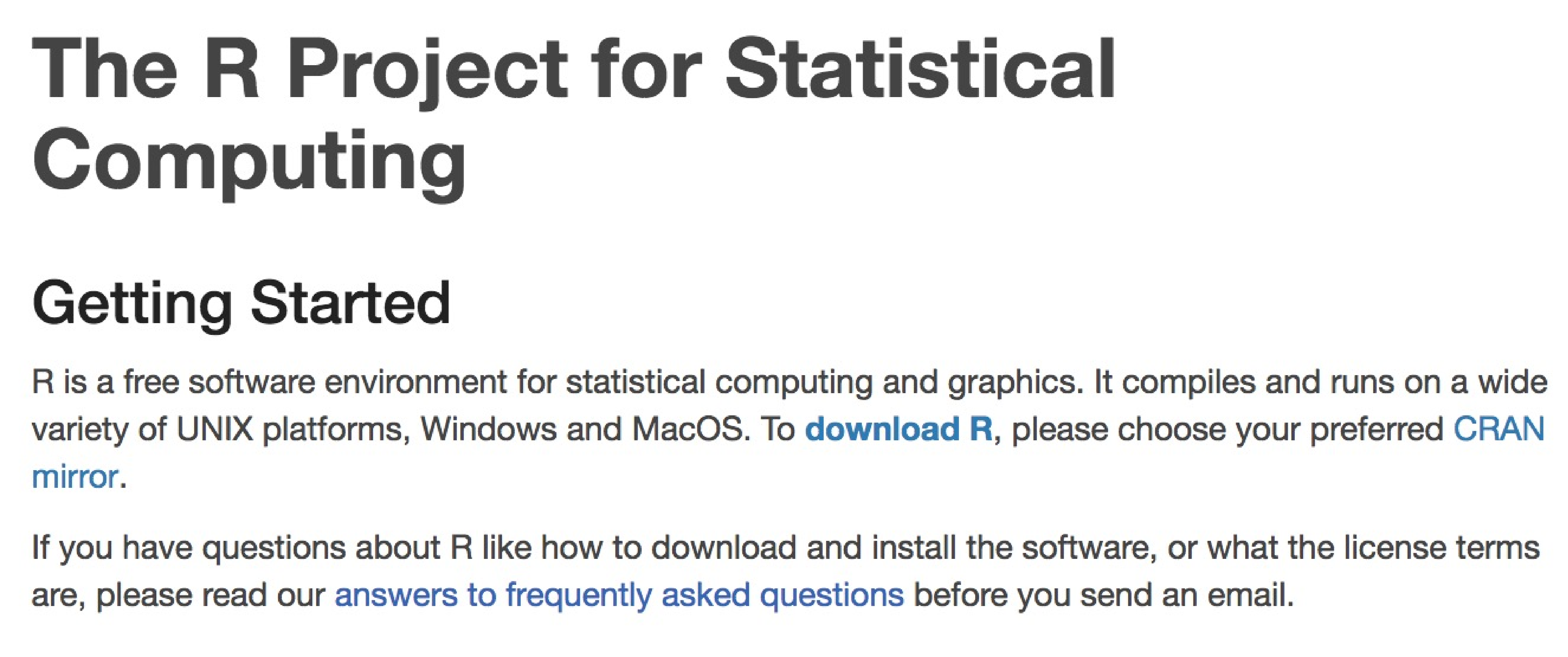
- Fairly similar committment for simple tasks (e.g., regression, EDA).
- Much better at plotting and visualizing data
- Much easier to use for complex tasks (e.g., advanced estimation, complex data)
- Overall, (much) greater ROI (IMO)
- For certain tasks (e.g., machine learning), the repertoire of tools available to R users is **unparalleled**.

R for Data Science



Installing R

www.r-project.org



The R Project for Statistical Computing

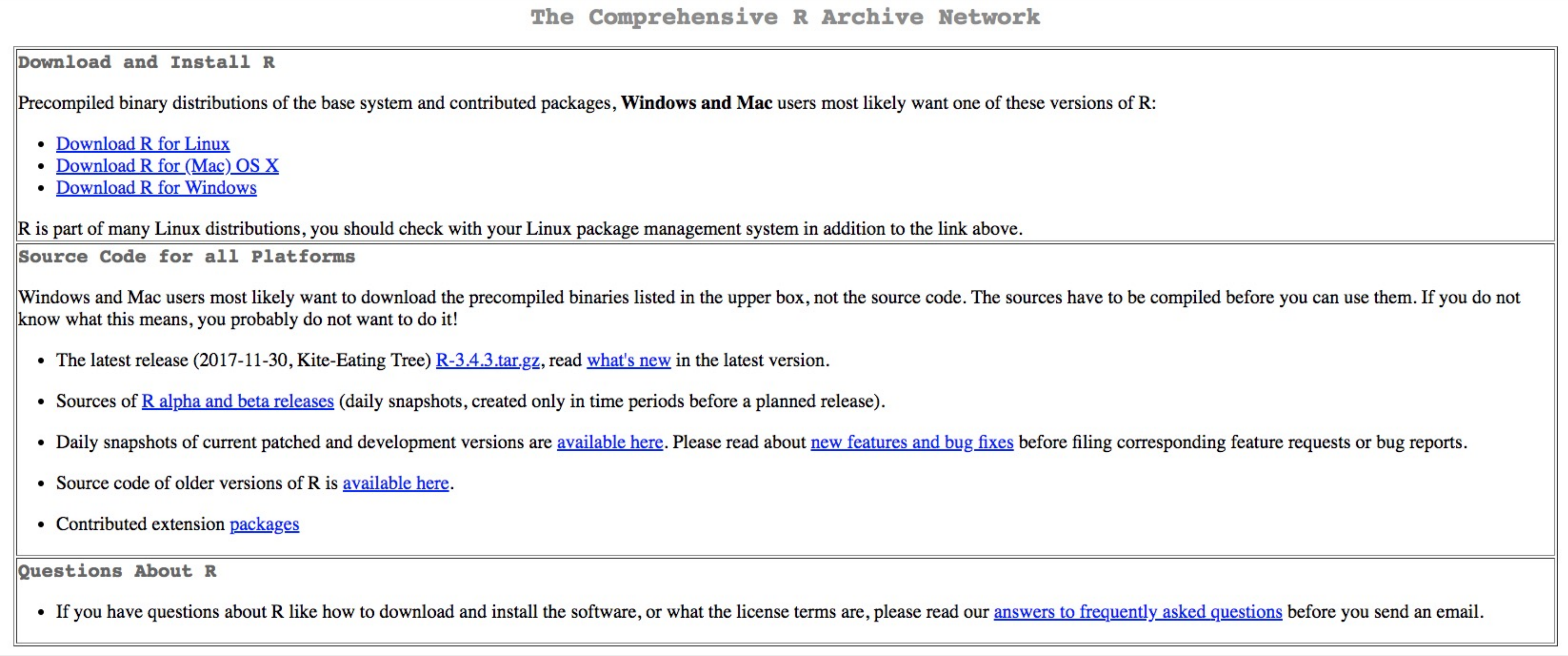
Getting Started

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To **download R**, please choose your preferred CRAN mirror.

If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

Installing R

http://lib.stat.cmu.edu/R/CRAN/



The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2017-11-30, Kite-Eating Tree) R-3.4.3.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

What is RStudio?

What is RStudio?

An integrated development environment (IDE) for R.

Highlights syntax/code

Code completion / indentation / navigation

Package development / debugging

Project management / organization

Version control (git)

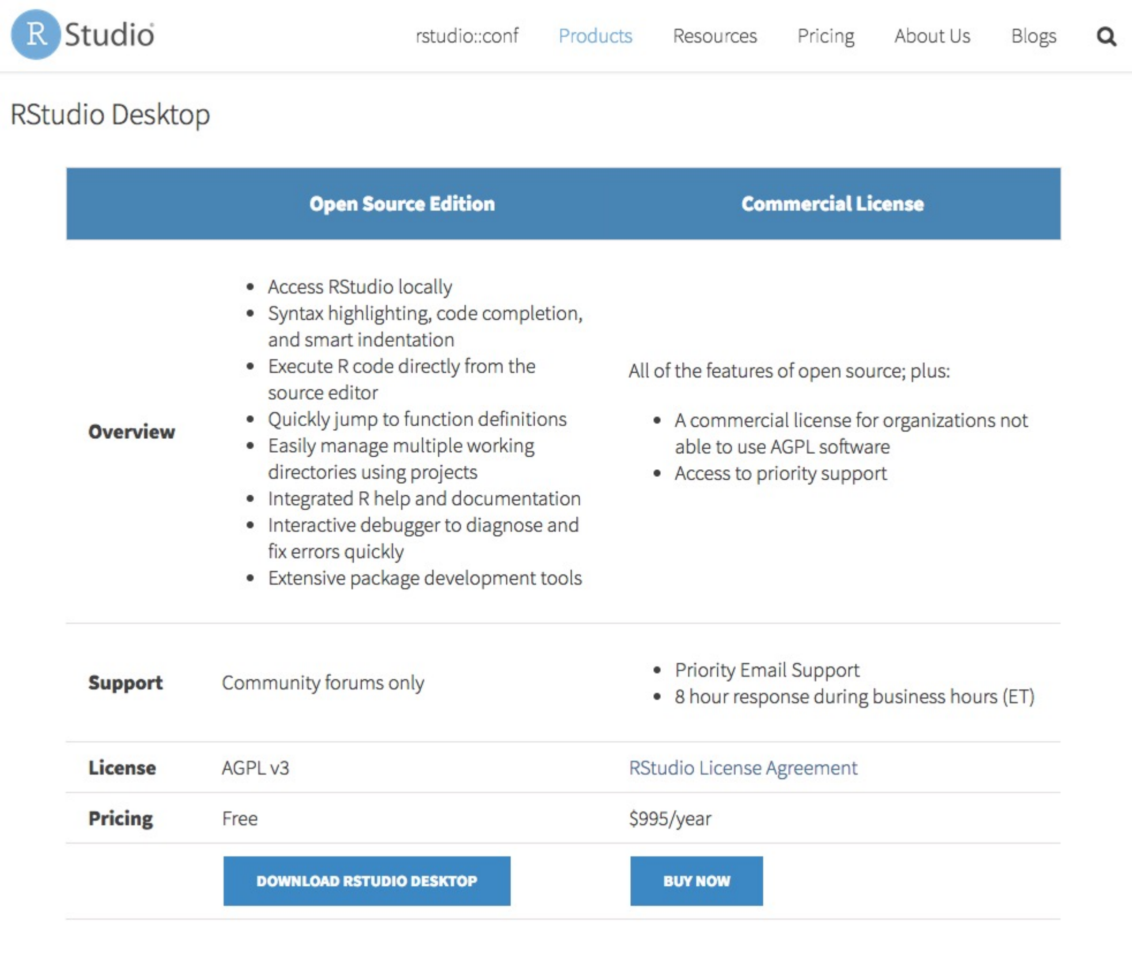
Creates HTML, Word, and pdf documents

A good option for reproducible research

Some really great insights on data science tools: INFO550

Installing RStudio

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

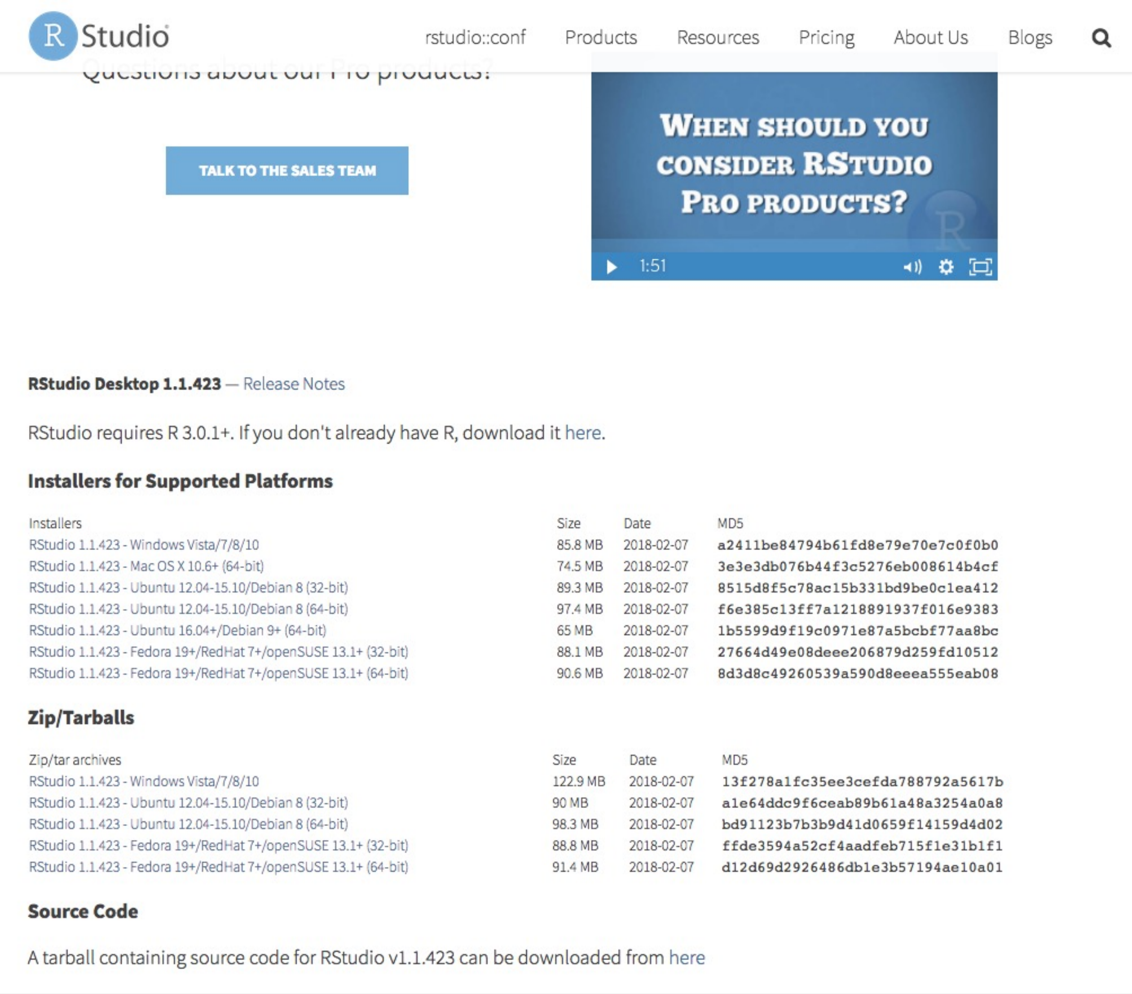


RStudio Desktop

	Open Source Edition	Commercial License
Overview	• Access RStudio locally • Syntax highlighting, code completion, and smart indentation • Execute R code directly from the source editor • Quickly jump to function definitions • Easily manage multiple working directories using projects • Integrated R help and documentation • Interactive debugger to diagnose and fix errors quickly • Extensive package development tools	All of the features of open source; plus: • A commercial license for organizations not able to use AGPL software • Access to priority support
Support	Community forums only	• Priority Email Support • 8 hour response during business hours (ET)
License	AGPL v3	RStudio License Agreement
Pricing	Free	$995/year
	DOWNLOAD RSTUDIO DESKTOP	BUY NOW

Installing RStudio

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



Questions about our Pro products?

RStudio Desktop 1.1.423 — Release Notes

RStudio requires R 3.0.1+. If you don't already have R, download it here.

Installers for Supported Platforms

Installers	Size	Date	MD5
RStudio 1.1.423 - Windows Vista/7/8/10	85.8 MB	2018-02-07	a2411be84794b61fd8e79e70e7c0f0b0
RStudio 1.1.423 - Mac OS X 10.6+ (64-bit)	74.5 MB	2018-02-07	3e3e3db076b44f3c5276eb008614b4cf
RStudio 1.1.423 - Ubuntu 12.04-15.10/Debian 8 (32-bit)	89.3 MB	2018-02-07	8515d8f5c78ac15b331bd9be0c1ea412
RStudio 1.1.423 - Ubuntu 12.04-15.10/Debian 8 (64-bit)	97.4 MB	2018-02-07	f6e385c13ff7a1218891937f016e9383
RStudio 1.1.423 - Ubuntu 16.04+/Debian 9+ (64-bit)	65 MB	2018-02-07	1b5599d9f19c0971e87a5bcbf77aa8bc
RStudio 1.1.423 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (32-bit)	88.1 MB	2018-02-07	27664d49e08deee206879d259fd10512
RStudio 1.1.423 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (64-bit)	90.6 MB	2018-02-07	8d3d8c49260539a590d8eeea555eab08

Zip/Tarballs

Zip/tar archives	Size	Date	MD5
RStudio 1.1.423 - Windows Vista/7/8/10	122.9 MB	2018-02-07	13f278a1fc35ee3cefda788792a5617b
RStudio 1.1.423 - Ubuntu 12.04-15.10/Debian 8 (32-bit)	90 MB	2018-02-07	a1e64ddc9f6ceab89b61a48a3254a0a8
RStudio 1.1.423 - Ubuntu 12.04-15.10/Debian 8 (64-bit)	98.3 MB	2018-02-07	bd91123b7b3b9d41d0659f14159d4d02
RStudio 1.1.423 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (32-bit)	88.8 MB	2018-02-07	ffde3594a52cf4aadfeb715f1e31b1f1
RStudio 1.1.423 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (64-bit)	91.4 MB	2018-02-07	d12d69d2926486db1e3b57194ae10a01

Source Code

A tarball containing source code for RStudio v1.1.423 can be downloaded from here

Running Code in R

To run a line of code in the R programming language, place your cursor at the end of a line, and press:

- COMMAND + RETURN (Mac)
- CTRL + ENTER (Windows)

```r
2*2*2
```

```
## [1] 8
```

Alternatively, highlight a single or multiple lines with your cursor, and press the same keys

R as a calculator

Most basically, R is a very advanced calculator:

```r
2 + 2 # add numbers
2 * pi # multiply by a constant
3^4 # powers
runif(5) # random number generation
sqrt(4^2) # functions
log(10) # natural log (i.e., base e)
log(100, base = 10) # log base 10
23 %/% 2 # integer division
23 %% 2 # modulus operator

# scientific notation
5000000000 * 1000
5e9 * 1e3
```

More operators: Quick-R

Assigning values to R objects

R is "object oriented". A basic task in R is to assign values to objects and perform functions on them:

```
a <- 10
a
```

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

```
a/100
```

```
## [1] 0.1
```

```
a+10
```

```
## [1] 20
```

```
# R is case sensitive!!!
A <- 15
print(c(a,A))
```

```
## [1] 10 15
```

Vectors

```
## Basic functional unit in R is a vector:
# numeric vector
nums <- c(1.1, 3, -5.7)
nums
```

```
## [1]  1.1  3.0 -5.7
```

```
nums <- rep(nums,2)
nums
```

```
## [1]  1.1  3.0 -5.7  1.1  3.0 -5.7
```

```
# integer vector
ints <- c(1L, 5L, -3L) # force storage as integer not decimal number
# 'L' is for 'long integer' (historical)

# sample nums with replacement
new_nums <- sample(nums,8,replace = TRUE)
new_nums
```

```
## [1] -5.7  1.1  3.0  3.0 -5.7 -5.7 -5.7 -5.7
```

Vectors

```r
# logical (i.e., Boolean) vector
bools <- c(TRUE, FALSE, TRUE, FALSE, T, T, F, F)
bools
```

```
## [1]  TRUE FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE
```

```r
# character vector
chars <- c("epidemiology is", "the study",
           "of the", "distribution",
           "and determinants", "of disease",
           "in", "a population")
chars
```

```
## [1] "epidemiology is"  "the study"        "of the"           "distribution"     "and determinants"
## [6] "of disease"       "in"               "a population"
```

Data Frames

Vectors can be combined into data frames (the basic data unit in R):

```
A <- data.frame(new_nums,bools,chars)
A
```

```
##   new_nums bools            chars
## 1     -5.7  TRUE  epidemiology is
## 2      1.1 FALSE        the study
## 3      3.0  TRUE           of the
## 4      3.0 FALSE     distribution
## 5     -5.7  TRUE and determinants
## 6     -5.7  TRUE       of disease
## 7     -5.7 FALSE               in
## 8     -5.7 FALSE     a population
```

Lists

And pretty much anything (vectors, data frames) can be combined into lists:

```
basic_list <- list(rep(1:3,5),
                   "what do you think of R so far?",
                   A)
basic_list[[1]]
```

```
##  [1] 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3
```

```
basic_list[[2]]
```

```
## [1] "what do you think of R so far?"
```

```
head(basic_list[[3]])
```

```
##   new_nums bools            chars
## 1     -5.7  TRUE  epidemiology is
## 2      1.1 FALSE        the study
## 3      3.0  TRUE           of the
## 4      3.0 FALSE     distribution
## 5     -5.7  TRUE and determinants
## 6     -5.7  TRUE       of disease
```

Subsetting

```
vals <- seq(2, 12, by = 2)
vals
```

```
## [1]  2  4  6  8 10 12
```

```
vals[3]
```

```
## [1] 6
```

```
vals[3:5]
```

```
## [1]  6  8 10
```

```
vals[c(1, 3, 6)];vals[-c(1, 3, 6)]
```

```
## [1]  2  6 12
```

```
## [1]  4  8 10
```

```
vals[c(rep(TRUE, 3), rep(FALSE, 2), TRUE)]
```

```
## [1]  2  4  6 12
```

Subsetting Data Frames

```
A[3,];A[,3]
```

```
##   new_nums bools  chars
## 3        3  TRUE of the
```

```
## [1] "epidemiology is"  "the study"        "of the"           "distribution"     "and determinants"
## [6] "of disease"       "in"               "a population"
```

```
A[2:3,];A[,2:3]
```

```
##   new_nums bools     chars
## 2      1.1 FALSE the study
## 3      3.0  TRUE    of the
```

```
##   bools            chars
## 1  TRUE  epidemiology is
## 2 FALSE        the study
## 3  TRUE           of the
## 4 FALSE     distribution
## 5  TRUE and determinants
## 6  TRUE       of disease
## 7 FALSE               in
## 8 FALSE     a population
```

```
subset(A,bools==F,select = -bools)
```

(Base) R Functions: Getting Help

```
# HELP!
?median

help.search('linear regression')

help(package='ggplot2')
```

(Base) R Functions: Object Structure

iris is a flower dataset included with R. The str() command gives the structure of the iris dataset:

```
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

The class() command tells us what kind of object this is:

```
class(iris)
```

```
## [1] "data.frame"
```

Using R & RStudio

R remains cutting edge through a network of users/maintainers who contribute **packages**. Packages are functions that are not part of base R. Without these packages, R would be much less useful.

For example:

- `VIM` is a package for the VIsualisation of Missing data
- `boot` is a package to get bootstrap CIs and standard errors
- `splines` is a package for including flexible regression splines in linear models
- `data.table` is a package for fast manipulation of data frames
- The `tidyverse` is a collection of packages that facilitate the practice of "tidy" data science.

Installing and loading packages

Let's install the tidyverse, and some other packages that are important for basic data visualization.

If this is your first time installing packages in R, you'll have to choose a CRAN mirror. This is done with the "repos = " (repository) argument (but can be done other ways too).

```
install.packages("tidyverse",repos='http://lib.stat.cmu.edu/R/CRAN')
```

```
##
## The downloaded binary packages are in
##     /var/folders/z_/cty0tpg97wz_x1d1zgdhwllr0000gs/T//Rtmp4xYbBq/downloaded_packages
```

```
library(tidyverse)
```

You should get a warning and other messages that I excluded here.

Installing and loading packages

Let's also install and load a package for the VIsualisation of Missing data:

```
install.packages("VIM",repos='http://lib.stat.cmu.edu/R/CRAN')
```

```
##
## The downloaded binary packages are in
##      /var/folders/z_/cty0tpg97wz_x1d1zgdhwllr0000gs/T//Rtmp4xYbBq/downloaded_packages
```

```
library(VIM)
```

Importing data into R

We can now use functions from the `tidyverse` to load our NHEFS data:

```
nhefs <- read_csv("./data/nhefs.csv")
```

```
## Rows: 1746 Columns: 61
```

```
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl (61): seqn, qsmk, death, yrdth, sbp, dbp, sex, age, race, income, marital, school, ht, wt71, wt82, wt8...
```

```
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Importing data into R

Using the `tidyverse` package to import data (as opposed to base R options) creates a tibble, which is an augmented data frame.

```
class(nhefs)
```

```
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

More options for importing data: R Studio Data Import Cheat Sheet

Exploring Data

Let's examine the structure of our NEHFS data:

```
dim(nhefs)
```

```
## [1] 1746   61
```

There are 1746 observations, and 61 columns in the nhefs tibble.

Let's select only specific columns from this tibble. We can do this using functions in the `dplyr` package, which is part of the `tidyvverse`:

```
nhefs <- nhefs %>% select(seqn,qsmk,sbp,dbp,sex,age,race,income,marital,school)
```

We'll learn more about the `%>%` (pipe) operator later. We've just re-written the nhefs object to include only the 10 variables in the `select()` function.

Exploring Data

This is what the selected columns look like:

```
head(nhefs)
```

```
## # A tibble: 6 × 10
##    seqn  qsmk   sbp   dbp   sex   age  race income marital school
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>   <dbl>  <dbl>
## 1   233     0   175    96     0    42     1     19       2      7
## 2   235     0   123    80     0    36     0     18       2      9
## 3   244     0   115    75     1    56     1     15       3     11
## 4   245     0   148    78     0    68     1     15       3      5
## 5   252     0   118    77     0    40     0     18       2     11
## 6   257     0   141    83     1    43     1     11       4      9
```

```
# can also use "tail" to see the end of the file
# tail(nhefs)
```

Functions and for loops

Functions are pieces of code written to accomplish specific tasks. Suppose we wanted to evaluate the proportion of missing data in each column in `nhefs`. We could do this by writing a function:

```r
propMissing <- function(x){
  mean(is.na(x))
}
propMissing(nhefs[,1])
```

```
## [1] 0
```

```r
propMissing(nhefs[,2])
```

```
## [1] 0
```

In the above code, `mean()` takes the sample average. In R, missing values are coded as `NA`, and `is.na()` is a base R function that returns a Boolean (true/false) value for each element in `x` that is missing. Thus, `mean(is.na(x))` returns the proportion of `x` that is missing.

Functions and for loops

Instead of copying and pasting the function over and over, we can put it in a `for` loop:

```
for (i in 1:10){
  output <- propMissing(nhefs[,i])
  print(output)
}
```

```
## [1] 0
## [1] 0
## [1] 0.0452
## [1] 0.0481
## [1] 0
## [1] 0
## [1] 0
## [1] 0.0378
## [1] 0
## [1] 0.067
```

Functions and for loops

Instead of a `for` loop, we can use the `apply` family of functions, which presents things in a way that is more informative. For example:

```
apply(nhefs,2,propMissing)
```

```
##    seqn    qsmk     sbp     dbp     sex     age    race  income marital  school
##  0.0000  0.0000  0.0452  0.0481  0.0000  0.0000  0.0000  0.0378  0.0000  0.0670
```

More information on the apply family: Apply tutorial

We can also make the above much more presentable and easier to read:

```
round(apply(nhefs,2,propMissing),3)*100
```

```
##    seqn    qsmk     sbp     dbp     sex     age    race  income marital  school
##     0.0     0.0     4.5     4.8     0.0     0.0     0.0     3.8     0.0     6.7
```

R & RStudio: Diving Deeper

Resources for further learning in R / Rstudio are endless:

- Chris Paciorek (UC Berkeley Bootcamp on youtube)
- R for Data Science (e-book)
- swirl
- Udacity Data Analysis with R
- Roger Peng's Coursera (advanced)
- r-bloggers