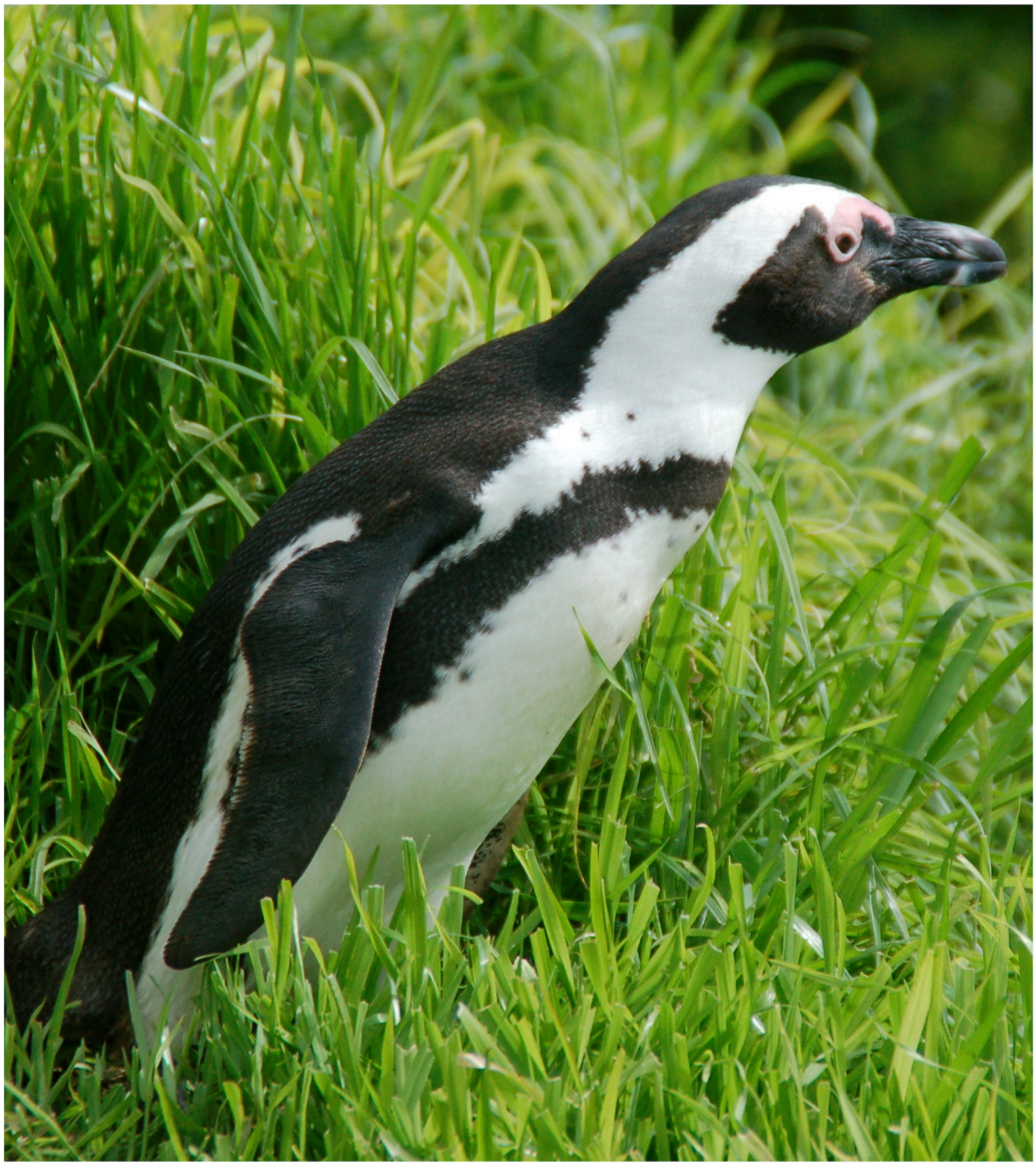
R Programming for Data Science



Roger D. Peng

R Programming for Data Science

Roger D. Peng

This book is for sale at http://leanpub.com/rprogramming

This version was published on 2015-07-20

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Also By Roger D. Peng

Exploratory Data Analysis with R

Contents

Preface

I started using R in 1998 when I was a college undergraduate working on my senior thesis. The version was 0.63. I was an applied mathematics major with a statistics concentration and I was working with Dr. Nicolas Hengartner on an analysis of word frequencies in classic texts (Shakespeare, Milton, etc.). The idea was to see if we could identify the authorship of each of the texts based on how frequently they used certain words. We downloaded the data from Project Gutenberg and used some basic linear discriminant analysis for the modeling. The work was eventually published[1] and was my first ever peer-reviewed publication. I guess you could argue it was my first real "data science" experience.

Back then, no one was using R. Most of my classes were taught with Minitab, SPSS, Stata, or Microsoft Excel. The cool people on the cutting edge of statistical methodology used S-PLUS. I was working on my thesis late one night and I had a problem. I didn't have a copy of any of those software packages because they were expensive and I was a student. I didn't feel like trekking over to the computer lab to use the software because it was late at night.

But I had the Internet! After a couple of Yahoo! searches I found a web page for something called R, which I figured was just a play on the name of the S-PLUS package. From what I could tell, R was a "clone" of S-PLUS that was free. I had already written some S-PLUS code for my thesis so I figured I would try to download R and see if I could just run the S-PLUS code.

It didn't work. At least not at first. It turns out that R is not exactly a clone of S-PLUS and quite a few modifications needed to be made before the code would run in R. In particular, R was missing a lot of statistical functionality that had existed in S-PLUS for a long time already. Luckily, R's programming language was pretty much there and I was able to more or less re-implement the features that were missing in R.

After college, I enrolled in a PhD program in statistics at the University of California, Los Angeles. At the time the department was brand new and they didn't have a lot of policies or rules (or classes, for that matter!). So you could kind of do what you wanted, which was good for some students and not so good for others. The Chair of the department, Jan de Leeuw, was a big fan of XLisp-Stat and so all of the department's classes were taught using XLisp-Stat. I diligently bought my copy of Luke Tierney's book[2] and learned to really love XLisp-Stat. It had a number of features that R didn't have at all, most notably dynamic graphics.

But ultimately, there were only so many parentheses that I could type, and still all of the research-level statistics was being done in S-PLUS. The department didn't really have a lot of copies of S-PLUS lying around so I turned back to R. When I looked around at my fellow students, I realized that I was basically the only one who had any experience using R. Since there was a budding interest in R

[1] http://amstat.tandfonline.com/doi/abs/10.1198/000313002100#.VQGiSELpagE

[2] http://www.amazon.com/LISP-STAT-Object-Oriented-Environment-Statistical-Probability/dp/0471509167/

around the department, I decided to start a "brown bag" series where every week for about an hour I would talk about something you could do in R (which wasn't much, really). People seemed to like it, if only because there wasn't really anyone to turn to if you wanted to learn about R.

By the time I left grad school in 2003, the department had essentially switched over from XLisp-Stat to R for all its work (although there were a few hold outs). Jan discusses the rationale for the transition in a paper[3] in the *Journal of Statistical Software*.

In the next step of my career, I went to the Department of Biostatistics[4] at the Johns Hopkins Bloomberg School of Public Health, where I have been for the past 12 years. When I got to Johns Hopkins people already seemed into R. Most people had abandoned S-PLUS a while ago and were committed to using R for their research. Of all the available statistical packages, R had the most powerful and expressive programming language, which was perfect for someone developing *new* statistical methods.

However, we didn't really have a class that taught students how to use R. This was a problem because most of our grad students were coming into the program having never heard of R. Most likely in their undergradute programs, they used some other software package. So along with Rafael Irizarry, Brian Caffo, Ingo Ruczinski, and Karl Broman, I started a new class to teach our graduate students R and a number of other skills they'd need in grad school.

The class was basically a weekly seminar where one of us talked about a computing topic of interest. I gave some of the R lectures in that class and when I asked people who had heard of R before, almost no one raised their hand. And no one had actually used it before. The main selling point at the time was "It's just like S-PLUS but it's free!" A lot of people had experience with SAS or Stata or SPSS. A number of people had used something like Java or C/C++ before and so I often used that a reference frame. No one had ever used a functional-style of programming language like Scheme or Lisp.

To this day, I still teach the class, known a Biostatistics 140.776 ("Statistical Computing"). However, the nature of the class has changed quite a bit over the past 10 years. The population of students (mostly first-year graduate students) has shifted to the point where many of them have been introduced to R as undergraduates. This trend mirrors the overall trend with statistics where we are seeing more and more students do undergraduate majors in statistics (as opposed to, say, mathematics). Eventually, by 2008–2009, when I'd asked how many people had heard of or used R before, everyone raised their hand. However, even at that late date, I still felt the need to convince people that R was a "real" language that could be used for real tasks.

R has grown a lot in recent years, and is being used in so many places now, that I think it's essentially impossible for a person to keep track of everything that is going on. That's fine, but it makes "introducing" people to R an interesting experience. Nowadays in class, students are often teaching me something new about R that I've never seen or heard of before (they are quite good at Googling around for themselves). I feel no need to "bring people over" to R. In fact it's quite the opposite–people might start asking questions if I *weren't* teaching R.

[3] http://www.jstatsoft.org/v13/i07

[4] http://www.biostat.jhsph.edu

This book comes from my experience teaching R in a variety of settings and through different stages of its (and my) development. Much of the material has been taken from by Statistical Computing class as well as the R Programming[5] class I teach through Coursera.

I'm looking forward to teaching R to people as long as people will let me, and I'm interested in seeing how the next generation of students will approach it (and how my approach to them will change). Overall, it's been just an amazing experience to see the widespread adoption of R over the past decade. I'm sure the next decade will be just as amazing.

[5] https://www.coursera.org/course/rprog

History and Overview of R

> There are only two kinds of languages: the ones people complain about and the ones nobody uses —*Bjarne Stroustrup*

Watch a video of this chapter[6]

What is R?

This is an easy question to answer. R is a dialect of S.

What is S?

S is a language that was developed by John Chambers and others at the old Bell Telephone Laboratories, originally part of AT&T Corp. S was initiated in 1976[7] as an internal statistical analysis environment—originally implemented as Fortran libraries. Early versions of the language did not even contain functions for statistical modeling.

In 1988 the system was rewritten in C and began to resemble the system that we have today (this was Version 3 of the language). The book *Statistical Models in S* by Chambers and Hastie (the white book) documents the statistical analysis functionality. Version 4 of the S language was released in 1998 and is the version we use today. The book *Programming with Data* by John Chambers (the green book) documents this version of the language.

Since the early 90's the life of the S language has gone down a rather winding path. In 1993 Bell Labs gave StatSci (later Insightful Corp.) an exclusive license to develop and sell the S language. In 2004 Insightful purchased the S language from Lucent for $2 million. In 2006, Alcatel purchased Lucent Technologies and is now called Alcatel-Lucent.

Insightful sold its implementation of the S language under the product name S-PLUS and built a number of fancy features (GUIs, mostly) on top of it—hence the "PLUS". In 2008 Insightful was acquired by TIBCO for $25 million. As of this writing TIBCO is the current owner of the S language and is its exclusive developer.

The fundamentals of the S language itself has not changed dramatically since the publication of the Green Book by John Chambers in 1998. In 1998, S won the Association for Computing Machinery's Software System Award, a highly prestigious award in the computer science field.

[6] https://youtu.be/STihTnVSZnI

[7] http://cm.bell-labs.com/stat/doc/94.11.ps

The S Philosophy

The general S philosophy is important to understand for users of S and R because it sets the stage for the design of the language itself, which many programming veterans find a bit odd and confusing. In particular, it's important to realize that the S language had its roots in data analysis, and did not come from a traditional programming language background. Its inventors were focused on figuring out how to make data analysis easier, first for themselves, and then eventually for others.

In Stages in the Evolution of S[8], John Chambers writes:

> "[W]e wanted users to be able to begin in an interactive environment, where they did not consciously think of themselves as programming. Then as their needs became clearer and their sophistication increased, they should be able to slide gradually into programming, when the language and system aspects would become more important."

The key part here was the transition from *user* to *developer*. They wanted to build a language that could easily service both "people". More technically, they needed to build language that would be suitable for interactive data analysis (more command-line based) as well as for writing longer programs (more traditional programming language-like).

Back to R

The R language came to use quite a bit after S had been developed. One key limitation of the S language was that it was only available in a commericial package, S-PLUS. In 1991, R was created by Ross Ihaka and Robert Gentleman in the Department of Statistics at the University of Auckland. In 1993 the first announcement of R was made to the public. Ross's and Robert's experience developing R is documented in a 1996 paper in the *Journal of Computational and Graphical Statistics*:

> Ross Ihaka and Robert Gentleman. R: A language for data analysis and graphics. Journal of Computational and Graphical Statistics, 5(3):299–314, 1996

In 1995, Martin Mächler made an important contribution by convincing Ross and Robert to use the GNU General Public License[9] to make R free software. This was critical because it allowed for the source code for the entire R system to be accessible to anyone who wanted to tinker with it (more on free software later).

In 1996, a public mailing list was created (the R-help and R-devel lists) and in 1997 the R Core Group was formed, containing some people associated with S and S-PLUS. Currently, the core group controls the source code for R and is solely able to check in changes to the main R source tree. Finally, in 2000 R version 1.0.0 was released to the public.

[8] http://www.stat.bell-labs.com/S/history.html
[9] http://www.gnu.org/licenses/gpl-2.0.html

Basic Features of R

In the early days, a key feature of R was that its syntax is very similar to S, making it easy for S-PLUS users to switch over. While the R's syntax is nearly identical to that of S's, R's semantics, while superficially similar to S, are quite different. In fact, R is technically much closer to the Scheme language than it is to the original S language when it comes to how R works under the hood.

Today R runs on almost any standard computing platform and operating system. Its open source nature means that anyone is free to adapt the software to whatever platform they choose. Indeed, R has been reported to be running on modern tablets, phones, PDAs, and game consoles.

One nice feature that R shares with many popular open source projects is frequent releases. These days there is a major annual release, typically in October, where major new features are incorporated and released to the public. Throughout the year, smaller-scale bugfix releases will be made as needed. The frequent releases and regular release cycle indicates active development of the software and ensures that bugs will be addressed in a timely manner. Of course, while the core developers control the primary source tree for R, many people around the world make contributions in the form of new feature, bug fixes, or both.

Another key advantage that R has over many other statistical packages (even today) is its sophisticated graphics capabilities. R's ability to create "publication quality" graphics has existed since the very beginning and has generally been better than competing packages. Today, with many more visualization packages available than before, that trend continues. R's base graphics system allows for very fine control over essentially every aspect of a plot or graph. Other newer graphics systems, like lattice and ggplot2 allow for complex and sophisticated visualizations of high-dimensional data.

R has maintained the original S philosophy, which is that it provides a language that is both useful for interactive work, but contains a powerful programming language for developing new tools. This allows the user, who takes existing tools and applies them to data, to slowly but surely become a developer who is creating new tools.

Finally, one of the joys of using R has nothing to do with the language itself, but rather with the active and vibrant user community. In many ways, a language is successful inasmuch as it creates a platform with which many people can create new things. R is that platform and thousands of people around the world have come together to make contributions to R, to develop packages, and help each other use R for all kinds of applications. The R-help and R-devel mailing lists have been highly active for over a decade now and there is considerable activity on web sites like Stack Overflow.

Free Software

A major advantage that R has over many other statistical packages and is that it's free in the sense of free software (it's also free in the sense of free beer). The copyright for the primary source code for R is held by the R Foundation[10] and is published under the GNU General Public License version

[10] http://www.r-project.org/foundation/

2.0[11].

According to the Free Software Foundation, with *free software*, you are granted the following four freedoms[12]

- The freedom to run the program, for any purpose (freedom 0).
- The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbor (freedom 2).
- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this.

You can visit the Free Software Foundation's web site[13] to learn a lot more about free software. The Free Software Foundation was founded by Richard Stallman in 1985 and Stallman's personal web site[14] is an interesting read if you happen to have some spare time.

Design of the R System

The primary R system is available from the Comprehensive R Archive Network[15], also known as CRAN. CRAN also hosts many add-on packages that can be used to extend the functionality of R.

The R system is divided into 2 conceptual parts:

1. The "base" R system that you download from CRAN: Linux[16] Windows[17] Mac[18] Source Code[19]
2. Everything else.

R functionality is divided into a number of *packages*.

- The "base" R system contains, among other things, the `base` package which is required to run R and contains the most fundamental functions.
- The other packages contained in the "base" system include `utils`, `stats`, `datasets`, `graphics`, `grDevices`, `grid`, `methods`, `tools`, `parallel`, `compiler`, `splines`, `tcltk`, `stats4`.

[11] http://www.gnu.org/licenses/gpl-2.0.html
[12] http://www.gnu.org/philosophy/free-sw.html
[13] http://www.fsf.org
[14] https://stallman.org
[15] http://cran.r-project.org
[16] http://cran.r-project.org/bin/linux/
[17] http://cran.r-project.org/bin/windows/
[18] http://cran.r-project.org/bin/macosx/
[19] http://cran.r-project.org/src/base/R-3/R-3.1.3.tar.gz

- There are also "Recommended" packages: `boot`, `class`, `cluster`, `codetools`, `foreign`, `KernSmooth`, `lattice`, `mgcv`, `nlme`, `rpart`, `survival`, `MASS`, `spatial`, `nnet`, `Matrix`.

When you download a fresh installation of R from CRAN, you get all of the above, which represents a substantial amount of functionality. However, there are many other packages available:

- There are over 4000 packages on CRAN that have been developed by users and programmers around the world.
- There are also many packages associated with the Bioconductor project[20].
- People often make packages available on their personal websites; there is no reliable way to keep track of how many packages are available in this fashion.
- There are a number of packages being developed on repositories like GitHub and BitBucket but there is no reliable listing of all these packages.

Limitations of R

No programming language or statistical analysis system is perfect. R certainly has a number of drawbacks. For starters, R is essentially based on almost 50 year old technology, going back to the original S system developed at Bell Labs. There was originally little built in support for dynamic or 3-D graphics (but things have improved greatly since the "old days").

Another commonly cited limitation of R is that objects must generally be stored in physical memory. This is in part due to the scoping rules of the language, but R generally is more of a memory hog than other statistical packages. However, there have been a number of advancements to deal with this, both in the R core and also in a number of packages developed by contributors. Also, computing power and capacity has continued to grow over time and amount of physical memory that can be installed on even a consumer-level laptop is substantial. While we will likely never have enough physical memory on a computer to handle the increasingly large datasets that are being generated, the situation has gotten quite a bit easier over time.

At a higher level one "limitation" of R is that its functionality is based on consumer demand and (voluntary) user contributions. If no one feels like implementing your favorite method, then it's *your* job to implement it (or you need to pay someone to do it). The capabilities of the R system generally reflect the interests of the R user community. As the community has ballooned in size over the past 10 years, the capabilities have similarly increased. When I first started using R, there was very little in the way of functionality for the physical sciences (physics, astronomy, etc.). However, now some of those communities have adopted R and we are seeing more code being written for those kinds of applications.

If you want to know my general views on the usefulness of R, you can see them here in the following exchange on the R-help mailing list with Douglas Bates and Brian Ripley in June 2004:

[20] http://bioconductor.org

> **Roger D. Peng**: I don't think anyone actually believes that R is designed to make *everyone* happy. For me, R does about 99% of the things I need to do, but sadly, when I need to order a pizza, I still have to pick up the telephone.
>
> **Douglas Bates**: There are several chains of pizzerias in the U.S. that provide for Internet-based ordering (e.g. www.papajohnsonline.com) so, with the Internet modules in R, it's only a matter of time before you will have a pizza-ordering function available.
>
> **Brian D. Ripley**: Indeed, the GraphApp toolkit (used for the RGui interface under R for Windows, but Guido forgot to include it) provides one (for use in Sydney, Australia, we presume as that is where the GraphApp author hails from). Alternatively, a Padovian has no need of ordering pizzas with both home and neighbourhood restaurants ….

At this point in time, I think it would be fairly straightforward to build a pizza ordering R package using something like the `RCurl` or `httr` packages. Any takers?

R Resources

Official Manuals

As far as getting started with R by reading stuff, there is of course this book. Also, available from CRAN[21] are

- An Introduction to R[22]
- R Data Import/Export[23]
- Writing R Extensions[24]: Discusses how to write and organize R packages
- R Installation and Administration[25]: This is mostly for building R from the source code)
- R Internals[26]: This manual describes the low level structure of R and is primarily for developers and R core members
- R Language Definition[27]: This documents the R language and, again, is primarily for developers

[21] http://cran.r-project.org
[22] http://cran.r-project.org/doc/manuals/r-release/R-intro.html
[23] http://cran.r-project.org/doc/manuals/r-release/R-data.html
[24] http://cran.r-project.org/doc/manuals/r-release/R-exts.html
[25] http://cran.r-project.org/doc/manuals/r-release/R-admin.html
[26] http://cran.r-project.org/doc/manuals/r-release/R-ints.html
[27] http://cran.r-project.org/doc/manuals/r-release/R-lang.html

Useful Standard Texts on S and R

- Chambers (2008). *Software for Data Analysis*, Springer
- Chambers (1998). *Programming with Data*, Springer: This book is *not* about R, but it describes the organization and philosophy of the current version of the S language, and is a useful reference.
- Venables & Ripley (2002). *Modern Applied Statistics with S*, Springer: This is a standard textbook in statistics and describes how to use many statistical methods in R. This book has an associated R package (the `MASS` package) that comes with every installation of R.
- Venables & Ripley (2000). *S Programming*, Springer: This book is a little old but is still relevant and accurate. Despite its title, this book is useful for R also.
- Murrell (2005). *R Graphics*, Chapman & Hall/CRC Press: Paul Murrell wrote and designed much of the graphics system in R and this book essentially documents the underlying details. This is not so much a "user-level" book as a developer-level book. But it is an important book for anyone interested in designing new types of graphics or visualizations.
- Wickham (2014). *Advanced R*, Chapman & Hall/CRC Press: This book by Hadley Wickham covers a number of areas including object-oriented programming, functional programming, profiling and other advanced topics.

Other Resources

- Major technical publishers like Springer, Chapman & Hall/CRC have entire series of books dedicated to using R in various applications. For example, Springer has a series of books called *Use R!*.
- A longer list of books can be found on the CRAN web site[28].

[28] http://www.r-project.org/doc/bib/R-books.html

Getting Started with R

Installation

The first thing you need to do to get started with R is to install it on your computer. R works on pretty much every platform available, including the widely available Windows, Mac OS X, and Linux systems. If you want to watch a step-by-step tutorial on how to install R for Mac or Windows, you can watch these videos:

- Installing R on Windows[29]
- Installing R on the Mac[30]

There is also an integrated development environment available for R that is built by RStudio. I really like this IDE—it has a nice editor with syntax highlighting, there is an R object viewer, and there are a number of other nice features that are integrated. You can see how to install RStudio here

- Installing RStudio[31]

The RStudio IDE is available from RStudio's web site[32].

Getting started with the R interface

After you install R you will need to launch it and start writing R code. Before we get to exactly how to write R code, it's useful to get a sense of how the system is organized. In these two videos I talk about where to write code and how set your working directory, which let's R know where to find all of your files.

- Writing code and setting your working directory on the Mac[33]
- Writing code and setting your working directory on Windows[34]

[29] http://youtu.be/Ohnk9hcxf9M
[30] https://youtu.be/uxuuWXU-7UQ
[31] https://youtu.be/bM7Sfz-LADM
[32] http://rstudio.com
[33] https://youtu.be/8xT3hmJQskU
[34] https://youtu.be/XBcvH1BpIBo

R Nuts and Bolts

Entering Input

At the R prompt we type expressions. The `<-` symbol is the assignment operator.

```
> x <- 1
> print(x)
[1] 1
> x
[1] 1
> msg <- "hello"
```

The grammar of the language determines whether an expression is complete or not.

```
x <-  ## Incomplete expression
```

The # character indicates a comment. Anything to the right of the # (including the # itself) is ignored. This is the only comment character in R. Unlike some other languages, R does not support multi-line comments or comment blocks.

Evaluation

When a complete expression is entered at the prompt, it is evaluated and the result of the evaluated expression is returned. The result may be *auto-printed*.

```
> x <- 5  ## nothing printed
> x       ## auto-printing occurs
[1] 5
> print(x)  ## explicit printing
[1] 5
```

The `[1]` shown in the output indicates that `x` is a vector and `5` is its first element.

Typically with interactive work, we do not explicitly print objects with the `print` function; it is much easier to just auto-print them by typing the name of the object and hitting return/enter. However, when writing scripts, functions, or longer programs, there is sometimes a need to explicitly print objects because auto-printing does not work in those settings.

When an R vector is printed you will notice that an index for the vector is printed in square brackets `[]` on the side. For example, see this integer sequence of length 20.

```
> x <- 10:30
> x
 [1] 10 11 12 13 14 15 16 17 18 19 20 21
[13] 22 23 24 25 26 27 28 29 30
```

The numbers in the square brackets are not part of the vector itself, they are merely part of the *printed output*.

With R, it's important that one understand that there is a difference between the actual R object and the manner in which that R object is printed to the console. Often, the printed output may have additional bells and whistles to make the output more friendly to the users. However, these bells and whistles are not inherently part of the object.

Note that the `:` operator is used to create integer sequences.

R Objects

R has five basic or "atomic" classes of objects:

- character
- numeric (real numbers)
- integer
- complex
- logical (True/False)

The most basic type of R object is a vector. Empty vectors can be created with the `vector()` function. There is really only one rule about vectors in R, which is that **A vector can only contain objects of the same class**.

But of course, like any good rule, there is an exception, which is a *list*, which we will get to a bit later. A list is represented as a vector but can contain objects of different classes. Indeed, that's usually why we use them.

There is also a class for "raw" objects, but they are not commonly used directly in data analysis and I won't cover them here.

Numbers

Numbers in R are generally treated as numeric objects (i.e. double precision real numbers). This means that even if you see a number like "1" or "2" in R, which you might think of as integers, they are likely represented behind the scenes as numeric objects (so something like "1.00" or "2.00"). This isn't important most of the time...except when it is.