Introduction to R for Epidemiologists

Jenna Krall, PhD

Thursday, January 15, 2015

Introduction to R for Epidemiologists

Course details

- Thursdays 4:00 PM-5:50 PM
- Room: GCR 119
- Course website: `www.jennakrall.com/IntrotoRepi`

Instructor and TA details

- Instructor: Dr. Jenna Krall, PhD
- E-mail: jenna.krall@emory.edu
- Office: GCR 369
- Office hours: 10 AM-11 AM Wednesdays and by appointment
- Lead TA: Brooke Alhanti
- TA: Anran Liu

Introduction to R for Epidemiologists

Office hours with Anran

- Monday 12-1, 1-2, 2-3, 3-4, 4-5
- Tuesday 10-11, 11-12, 12-1, 3-4, 4-5
- Thursday 1-2, 2-3, 3-4

Introduction to R for Epidemiologists

Prerequisites

- Introductory-level statistics course covering regression
- No programming experience required

What you need

- R and RStudio installed
- No required textbook

Introduction to R for Epidemiologists

Class format

- 60 minute lecture
- 50 minute lab

These two portions of the class provide two different ways of learning R

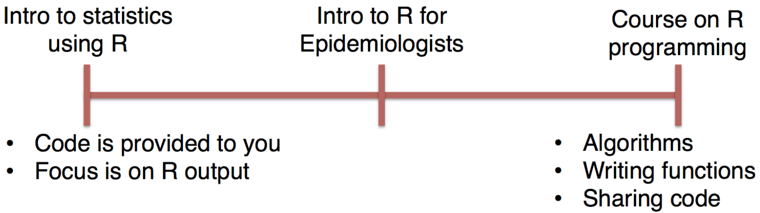
1. I tell you how to do things
2. Self-guided learning (with help from us)

Introduction to R for Epidemiologists

Grading

- 20% lab assignments
  - Must work on during class to receive credit
- 45% homework assignments (15% each)
  - May work with others
  - Must submit your own code
- 35% final project
  - Epidemiological analysis of real data
  - You may consult me or the TAs
  - You may not consult other students
- No late assignments will be accepted

Introduction to R for Epidemiologists



Intro to statistics using R

- Code is provided to you
- Focus is on R output

Intro to R for Epidemiologists

Course on R programming

- Algorithms
- Writing functions
- Sharing code

Introduction to R for Epidemiologists

Course objectives

- Read in and clean data
- Compute summary statistics
- Apply and interpret statistical methods
- Create well-formatted output tables
- Create publication-quality figures
- Create reproducible reports

If there is time

- R packages
- Shiny applications

Outline

1. History of R
2. R vs. other statistical programming languages
3. Orientation to R
4. Some basics
5. Getting data into R
6. Rules about programming
7. Resources

History of R

- Programming language for statistical computing
- Derived from S language (John Chambers, Bell labs)
- GNU (Gnu's not UNIX) project: free and open source
- Developed by
  - Ross Ihaka (University of Auckland)
  - Robert Gentleman (Genentech)
- R can do almost anything
  - Users can create their own R packages
- R project: `http://www.r-project.org/`
- Comprehensive R Archive Network (CRAN): `http://cran.r-project.org/`

R vs. other statistical programming languages

- The R language
  - We can create objects (e.g. a vector of heights)
  - Objects can interact with other objects (e.g. comparing height and age)
  - Programs consist of applying methods to objects (e.g. what is the mean height?)
- R is interactive
- R and RStudio
- R is easily customizable
  - Customize figures
  - Make nicely formatted tables
  - Write functions

R vs. other statistical programming languages

SAS

- No interactivity
- Easy to look at your data
- Difficult to create customized output
- Can be more stable for complex models
- Need a lot of code to perform simple tasks (e.g. to find a mean)

In SAS,

```
proc means data = data1;
var variable1;
run;
```

In R,

```
mean(variable1)
```

R vs. other statistical programming languages

- STATA
  - One dataset at a time
  - Intuitive and easy data cleaning
- MATLAB
  - Matrix computation
  - Powerful
- SPSS
  - Intuitive
  - No programming required

Orientation to R

The console

- Type code interactively
- Do not plan to save this code

Editor

- Save code for later
- Can run in console from editor

Orientation to R

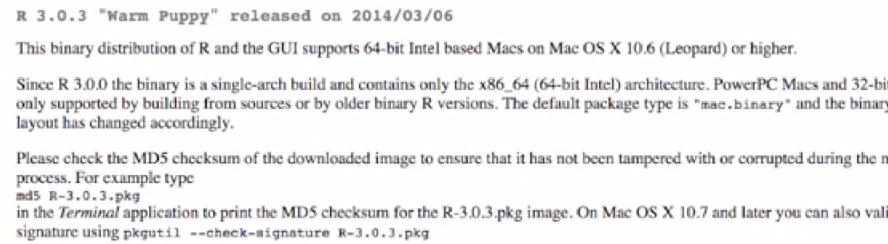
R vs. RStudio

- R opens the console
- RStudio
  - Runs usual R
  - Nice layout with different panels for editor, console, etc.
  - Additional features
  - Can customize the arrangement of the windows

Orientation to R

R version 3.1.2: codename “Pumpkin Helmet”

R version 3.0.2:



R 3.0.3 "Warm Puppy" released on 2014/03/06

This binary distribution of R and the GUI supports 64-bit Intel based Macs on Mac OS X 10.6 (Leopard) or higher.

Since R 3.0.0 the binary is a single-arch build and contains only the x86_64 (64-bit Intel) architecture. PowerPC Macs and 32-bi
only supported by building from sources or by older binary R versions. The default package type is `"mac.binary"` and the binar
layout has changed accordingly.

Please check the MD5 checksum of the downloaded image to ensure that it has not been tampered with or corrupted during the n
process. For example type
`md5 R-3.0.3.pkg`
in the *Terminal* application to print the MD5 checksum for the R-3.0.3.pkg image. On Mac OS X 10.7 and later you can also vali
signature using `pkgutil --check-signature R-3.0.3.pkg`

Orientation to R

The basics

```
5 + 3
```

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

```
123/2 + (2 * 17.2)
```

```
## [1] 95.9
```

The basics

Can store results using assignment operators

As an example:

```
nameofobject <- 5
print(nameofobject)
```

```
## [1] 5
```

```
nameofobject2 = 5
print(nameofobject2)
```

```
## [1] 5
```

Generally <- is preferred

The basics

Can calculate result as object:

```r
nameofsum <- 5 + 3
print(nameofsum)
```

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

```r
nameofsum
```

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

Objects can be functions of other objects:

```r
nameofnew <- nameofobject + nameofsum
nameofnew
```

```
## [1] 13
```

The basics

Objects can be reassigned

- Recall that we created an R object `nameofobject`

```
nameofobject
```

```
## [1] 5
```

- Suppose we reassign the name

```
nameofobject <- 1500
nameofobject
```

```
## [1] 1500
```

Classes

- So far, all the R objects we have dealt with have been numeric
- There are different data classes that R objects can belong to
- The ‘class‘ function in R can tell us the class of an R object

```
class(nameofobject)
```

```
## [1] "numeric"
```

- String or character objects that are non-numeric

```
nameofstring <- "IntrotoR"
nameofstring
```

```
## [1] "IntrotoR"
```

```
class(nameofstring)
```

```
## [1] "character"
```

Classes

class:small-code We can also create vectors by combining numbers or text. The c function in R combines objects together.

```
vector1 <- c(6, 2, 3, 4)
vector1
```

```
## [1] 6 2 3 4
```

```
class(vector1)
```

```
## [1] "numeric"
```

```
length(vector1)
```

```
## [1] 4
```

Classes

Vectors can also consist of strings

```
vectorstring <- c("R", "is", "awesome", "for", "Epidemiologists")
vectorstring
```

```
## [1] "R"               "is"              "awesome"         "for"
## [5] "Epidemiologists"
```

```
class(vectorstring)
```

```
## [1] "character"
```

```
length(vectorstring)
```

```
## [1] 5
```

Classes

Mixing data classes

```r
vectormix <- c("Intro", "to", "R", "version", 1)
vectormix
```

```
## [1] "Intro"   "to"      "R"       "version" "1"
```

```r
class(vectormix)
```

```
## [1] "character"
```

```r
length(vectormix)
```

```
## [1] 5
```

Classes

Factors in R

```
grades <- c("A", "B", "A", "A", "C", "F", "D", "B")
grades
```

```
## [1] "A" "B" "A" "A" "C" "F" "D" "B"
```

```
class(grades)
```

```
## [1] "character"
```

```
factorgrades <- factor(grades)
factorgrades
```

```
## [1] A B A A C F D B
## Levels: A B C D F
```

```
class(factorgrades)
```

```
## [1] "factor"
```

Classes

Why does class matter?

- Functions will perform differently for different classes
  - Print function for factor shows levels
  - Mean of a character vector is meaningless
- There are many other classes
  - Next class: matrices, data frames, lists, dates
  - Classes for output (e.g. linear model class lm)
  - In R, you can create your own classes of R objects

Selecting elements

Use brackets to select elements of a vector

```
vectorstring
```

```
## [1] "R"               "is"              "awesome"         "for"
## [5] "Epidemiologists"
```

```
vectorstring[1]
```

```
## [1] "R"
```

```
vectorstring[5]
```

```
## [1] "Epidemiologists"
```

R functions

- Used as nameoffunction(Robject)

```
vector1
```

```
## [1] 6 2 3 4
```

```
mean(vector1)
```

```
## [1] 3.75
```

- Functions can take several arguments separated by commas

```
mean(vector1, trim = 0.25)
```

```
## [1] 3.5
```

```
log(1)
```

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

R functions

```
length(vector1)
```

```
## [1] 4
```

```
median(vector1)
```

```
## [1] 3.5
```

```
summary(vector1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.00    2.75    3.50    3.75    4.50    6.00
```

```
ls()
```

```
##  [1] "factorgrades"  "grades"        "nameofnew"     "nameofobject"
##  [5] "nameofobject2" "nameofstring"  "nameofsum"     "vector1"
##  [9] "vectormix"     "vectorstring"
```

Packages

Packages include tools to perform many different analyses

- Some packages are preloaded
  - base (‘mean‘, ‘print‘, ‘length‘), stats (‘median‘, ‘lm‘)
  - datasets
  - graphics
- Can install and load additional packages from CRAN
  - ggplot2, RColorBrewer (advanced graphics)
  - dplyr, reshape2 (data cleaning/manipulation tools)

Installing a package (may need to specify a CRAN mirror)

```r
install.packages("ggplot2")
```

Loading a package

```r
library(ggplot2)
```

Working directory

Where R

- looks for data
- saves data, figures, etc.

```
getwd()
```

```
## [1] "/Users/jennakrall/Dropbox/IntrotoREpi/data"
```

Change your working directory using `setwd`

```
setwd("/Users/jennakrall/Dropbox/")
```

Working directory

- For Windows users, setwd("C:/Users/yourusername/etc.")
- For Mac users, setwd("/Users/yourusername/etc.")
- Can use home directory

```
setwd("~/")
getwd()
```

```
## [1] "/Users/jennakrall"
```

Getting data into R

R can read in data from many different sources

Types of data files

- R data files: (extension .RData or .rda)
- CSV file: comma separated values (extension .csv).
- Excel file: Microsoft Office (extension .xls or .xlsx). Can use excel to create a .csv file or read into R directly.
- SAS: datafiles (extension .sas7bdat) or Xport file (extension .xpt)

You must either specify the path to the data or set your working directory to the where the data are located

Getting data into R

Google flu trends

- Estimates flu activity using google searches
- This dataset includes estimated flu activity (by day) for the whole United States, Georgia, Atlanta, and Health and Human Services Region 4 (which includes Alabama, Florida, Georgia, Kentucky, Mississippi, North Carolina, South Carolina, and Tennessee).
- Google's Nature paper: `http://www.nature.com/nature/journal/v457/n7232/full/nature07634.html`
- Science paper: `http://www.sciencemag.org/content/343/6176/1203`
- Data Source: Google Flu Trends (`http://www.google.org/flutrends`), accessed 12/3/2014

Getting data into R

Example using .RData or .rda files

```
load("googleflu.RData")
ls()
```

```
## [1] "flu"
```

```
head(flu)
```

```
##         Date United.States Georgia Atlanta HHSRegion4
## 1 2003-09-28           902     514     519        631
## 2 2003-10-05           952     532     484        652
## 3 2003-10-12          1092     557     497        735
## 4 2003-10-19          1209     608     563        822
## 5 2003-10-26          1249     745     845        797
## 6 2003-11-02          1374     767     771        850
```

```
class(flu)
```

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

Getting data into R

Example using .csv file

```
flu <- read.csv("googleflu.csv", stringsAsFactors = FALSE)
head(flu)
```

```
##         Date United.States Georgia Atlanta HHSRegion4
## 1 2003-09-28           902     514     519        631
## 2 2003-10-05           952     532     484        652
## 3 2003-10-12          1092     557     497        735
## 4 2003-10-19          1209     608     563        822
## 5 2003-10-26          1249     745     845        797
## 6 2003-11-02          1374     767     771        850
```

```
class(flu)
```

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

Getting data into R

Example using .xls or .xlsx file

```r
library(XLConnect)
wkbook_flu <- loadWorkbook("googleflu.xlsx")
class(wkbook_flu)
```

```
## [1] "workbook"
## attr(,"package")
## [1] "XLConnect"
```

```r
flu <- readWorksheet(wkbook_flu, 1)
```

Getting data into R

Example using .xls or .xlsx file

```
head(flu)
```

```
##         Date United.States Georgia Atlanta HHSRegion4
## 1 2003-09-28           902     514     519        631
## 2 2003-10-05           952     532     484        652
## 3 2003-10-12          1092     557     497        735
## 4 2003-10-19          1209     608     563        822
## 5 2003-10-26          1249     745     845        797
## 6 2003-11-02          1374     767     771        850
```

```
class(flu)
```

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

Getting data into R

Example using SAS

1. Save SAS data as .csv and read into R
2. Use SAS Xport Transport file
3. Read SAS data directly
    - Only if SAS is installed
    - Do not recommend

Getting data into R

1. Save SAS data as .csv and read into R

- Save data (.sas7bdat) in SAS as .csv file

```
proc export data = googleflu
    outfile = "googleflu.csv"
    dbms = csv replace;
putnames = yes;
run;
```

- Then use `read.csv` function in R

```
flu <- read.csv("googleflu.csv", stringsAsFactors = FALSE)
```

Getting data into R

2. Use SAS Xport Transport file

SAS xport files

- Save data in SAS as SAS xport file

```
* Set xport filepath ;
libname lib1 xport "H:\googleflu.xpt";

* Set the xport file ;
data lib1.flu;
* This is your original data;
set flu;
run;
```

Getting data into R

2. Use SAS Xport Transport file

- Read xport file into R

```
library(Hmisc)
flu <- sasxport.get("googleflu.xpt")
```

```
## Processing SAS dataset FLU    ..
```

```
head(flu)
```

```
##         date   us georgia atlanta hhs
## 1 2003-09-28  902     514     519 631
## 2 2003-10-05  952     532     484 652
## 3 2003-10-12 1092     557     497 735
## 4 2003-10-19 1209     608     563 822
## 5 2003-10-26 1249     745     845 797
## 6 2003-11-02 1374     767     771 850
```

Getting data into R

3. Read SAS data directly (only if SAS is installed)

- This command sometimes yields ambiguous errors from SAS
- Need Windows/SAS
- Use `read.ssd` function to read `.sas7bdat` file directly into R
    - ‘read.ssd‘ function in R creates xport file (using SAS)
    - reads the resulting dataset into R using ‘read.xport‘.

```r
library(foreign)
lib1 <- "c:/"
flu <- read.ssd(lib1, "flu", sascmd = "filepath/to/where/SAS/is")
```

May need to tell R where to find SAS using sascmd argument.

Getting data into R

Other functions to read in data include

- read.table
- read.delim
- readLines
- scan
- many others

Rules about programming

- Always comment your code– your grade will depend on it (and so will your future self)

```r
# This is the mean of our numeric vector
mean(vector1)
```

```
## [1] 3.75
```

> *Your closest collaborator is you six months ago but you don't reply to email. – Erin Jonaitis (via Andrew Gelman)*

- Use appropriate spacing

```r
newvector <- c(5, 3, 4, 4)
```

Rules about programming

- Lines of code should be truncated before 80 characters

```
newvector <- c(28, 90, 10, 57, 66, 93, 29, 95, 19, 14, 96, 78, 61, 51, 1, 87,
    60, 46, 43, 35, 17, 64, 2, 55, 54, 25, 92, 32, 42, 94, 97, 86, 77, 6, 13,
    23, 20, 67, 30, 68, 12, 5, 24, 59, 33, 75, 26, 65, 88, 31, 47, 38, 53, 70,
    27, 98, 16, 8, 37, 15, 11, 40, 85, 83, 76, 91, 81, 48, 80, 7, 36, 22, 89,
    39, 4, 63, 21, 79, 99, 45, 56, 100, 44, 18, 3, 58, 73, 52, 62, 72, 69, 71,
    74, 84, 82, 49, 34, 50, 9, 41)
```

- Indent continued lines with two spaces

Rules about programming

Naming conventions

- The more descriptive, the better
- Names are case sensitive
- Capitalization and camelCase can make objects hard to type and remember
- Can use underscores to separate words in names

```
new_vector <- c(1, 2, 3, 4)
new_vector
```

```
## [1] 1 2 3 4
```

- Don't overwrite existing R functions or objects (e.g. c or T)

Style guides:

- Hadley Wickham: http://r-pkgs.had.co.nz/style.html
- Google's: https://google-styleguide.googlecode.com/svn/trunk/Rguide.xml

Resources

R help files (use `?function`)

```
?mean
help.search("mean")
```

Search engines

- Google: Append CRAN onto your google search for R.
- R seek: `http://www.rseek.org/`

Other resources `www.jennakrall.com/IntrotoRepi/resources.html`

- Stack Overflow: `http://stackoverflow.com/questions/tagged/r`
- UCLA R stats: `http://www.ats.ucla.edu/stat/r/`