1. Introducing data analysis in R

Frank Edwards

9/2/2019

Introduction to Statistics

Why learn statistics?

1. Quantitative methods allow us to discover or infer patterns when we have large amounts of data
2. Statistics provide methods for testing for differences between groups of data
3. Always remember two things: 1) all models are wrong, but some are useful; 2) social data come from people, and are always imperfect

Course Goals

1. Introduce students to statistical computing through the R programming language
2. Develop data manipulation, exploration, and visualization skills
3. Introduce core concepts in probability and statistics

Lecture / lab / office hours

- Lecture introduces new materials and concepts
- Lab provides interactive time to work on progamming
- Office hours (FE: Friday 9:30-11, RL: Monday 11 - 1)
- Raven's office hours provide a great opportunity to get help with the homework

Weekly routine for the course

1. Do the assigned readings
2. Attend lecture
3. Practice skills in lab
4. Do homework
5. Take a break
6. return to 1

Books and readings

1. Imai, *Quantitative Social Science*

This book is the foundation of the course. It introduces core social science methods, programming and statistics.

Homework

- I will assign a problem set at the end of each lecture
- Problem sets will be posted at github.com/f-edwards/intro_stats/
- Data is posted either on GitHub or available through the QSS package
- Problem sets are due at 10AM the day of the following lecture
- Explain your answersin writing. Don't just submit code.
- Homework is hard! Give it plenty of time.
- Email me (`frank.edwards@rutgers.edu`) the compiled RMarkdown output file and source code (.html and .rmd files) by the due date

Course communication

- Course website: `https://f-edwards.github.io/intro_stats/`
- Course Slack: `https://introstatsfall2020.slack.com`

Questions about the course?

Software installation

- R: `https://cran.r-project.org/`
- RStudio: `https://www.rstudio.com/`

Packages for the course

```
#install.packages(c(”devtools”, ”tidyverse”))
library(devtools)
### For book data
install_github(”kosukeimai/qss-package”,
               build_vignettes = TRUE)
### For problem set data
install_github(”conjugateprior/qss.student”)
```

Problems?

Introducing: R!

R is a calculator

```r
5+3
```

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

R is a calculator

```r
5*3
```

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

R is a calculator

```r
5/3
```

```
## [1] 1.666667
```

R is a calculator

```r
5^3
```

```
## [1] 125
```

R can make comparisons

```r
5>3
```

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

R can make comparisons

```r
5<=3
```

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

R can make comparisons

```r
5==3
```

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

R can make comparisons

```
”a”==”a”
```

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

R works with objects

```r
a<-2
a+1
```

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

R works with objects

```r
a<-2
a+1
```

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

R works with objects

```r
a<-2
a+1
```

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

```r
b<-a+2
```

R works with objects

```
a<-2
a<-a+1
```

R works with objects

```r
a
```

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

Objects can take many types

```
a<-2
class(a)
```

```
## [1] ”numeric”
```

```
b<-”howdy”
class(b)
```

```
## [1] ”character”
```

```
c<-TRUE
class(c)
```

```
## [1] ”logical”
```

Objects can take many types

```r
b<-”howdy”
class(b)
```

```
## [1] ”character”
```

Objects can take many types

```r
c<-TRUE
class(c)
```

```
## [1] ”logical”
```

Vectors

Vectors are one-dimensional arrays of values of any class

```
vector1<-c(2,3,4,5,6)
vector1
```

```
## [1] 2 3 4 5 6
```

Vectors

Vectors are one-dimensional arrays of values of any class

```
vector2<-c(”a”, ”fancy”, ”vector”)
vector2
```

```
## [1] ”a”      ”fancy”  ”vector”
```

Vectors

```
vector3<-c(TRUE, FALSE, TRUE, FALSE, FALSE)
vector3
```

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

Vectorized operations

```
vector1
```

```
## [1] 2 3 4 5 6
```

```
2 * vector1
```

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

```
vector3
```

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

```
vector3==FALSE
```

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

Vectorized operations

```
vector3
```

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

```
vector3==FALSE
```

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

Vector indexing

```
vector2
```

```
## [1] ”a”      ”fancy”  ”vector”
```

Vector indexing

```
vector2
```

```
## [1] ”a”      ”fancy”  ”vector”
```

```
vector2[1]
```

```
## [1] ”a”
```

Vector indexing

```
vector2
```

```
## [1] ”a”      ”fancy”  ”vector”
```

```
vector2[2]
```

```
## [1] ”fancy”
```

Vector indexing

```
vector2
```

```
## [1] ”a”      ”fancy”  ”vector”
```

```
vector2[3]
```

```
## [1] ”vector”
```

Operations and vector indexing

```
vector1
```

```
## [1] 2 3 4 5 6
```

```
vector1[2] + 3
```

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

Functions!

R has loads and loads of functions.

- Functions run a fixed set of operations on some argument(s)
- Functions return a value that can be assigned to an object
- Functions take the general form *function(arguments)*

Some common function in R for vectors

```r
vector1
```

```
## [1] 2 3 4 5 6
```

```r
min(vector1)
```

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

```r
max(vector1)
```

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

```r
mean(vector1)
```

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

```r
sum(vector1)
```

Some common function in R for vectors

```
vector1
```

```
## [1] 2 3 4 5 6
```

```
max(vector1)
```

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

Some common function in R for vectors

```r
vector1
```

```
## [1] 2 3 4 5 6
```

```r
mean(vector1)
```

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

Some common function in R for vectors

```r
vector1
```

```
## [1] 2 3 4 5 6
```

```r
sum(vector1)
```

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

Functions can work together

```
sum(vector1)
```

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

```
length(vector1)
```

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

```
sum(vector1)/length(vector1)
```

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

We can even write our own!

```r
redundantMean<-function(x){
  n<-length(x)
  sum_x<-sum(x)
  xbar<-sum_x/n
  return(xbar)
}
redundantMean(vector1)
```

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

Questions?

Data frames

```
### load the book package
library(qss)
### attach the UNpop data
data(UNpop)
head(UNpop)
```

```
##   year world.pop
## 1 1950   2525779
## 2 1960   3026003
## 3 1970   3691173
## 4 1980   4449049
## 5 1990   5320817
## 6 2000   6127700
```

As super-matrices?

Recall that we can obtain any element x_{ij} from a matrix X with row index i and column index j

```
UNpop
```

```
##   year world.pop
## 1 1950   2525779
## 2 1960   3026003
## 3 1970   3691173
## 4 1980   4449049
## 5 1990   5320817
## 6 2000   6127700
## 7 2010   6916183
```

As super-matrices?

Recall that we can obtain any element x_{ij} from a matrix X with row index i and column index j

```
UNpop[1,1]
```

```
## [1] 1950
```

As super-matrices?

Recall that we can obtain any element x_{ij} from a matrix X with row index i and column index j

```
UNpop[2,2]
```

```
## [1] 3026003
```

Other means of indexing

```
UNpop[,1]
```

```
## [1] 1950 1960 1970 1980 1990 2000 2010
```

Other means of indexing

```
UNpop[1,]
```

```
##   year world.pop
## 1 1950   2525779
```

Other means of indexing

```
UNpop$year
```

```
## [1] 1950 1960 1970 1980 1990 2000 2010
```

Other means of indexing

```r
UNpop$year[3]
```

```
## [1] 1970
```

Other means of indexing

```
UNpop[1, ”world.pop”]
```

```
## [1] 2525779
```

Some convenient data.frame functions

```
summary(UNpop)
```

```
##       year        world.pop
##  Min.   :1950   Min.   :2525779
##  1st Qu.:1965   1st Qu.:3358588
##  Median :1980   Median :4449049
##  Mean   :1980   Mean   :4579529
##  3rd Qu.:1995   3rd Qu.:5724258
##  Max.   :2010   Max.   :6916183
```

Some convenient data.frame functions

```r
nrow(UNpop)
```

```
## [1] 7
```

```r
ncol(UNpop)
```

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

```r
dim(UNpop)
```

```
## [1] 7 2
```

Some convenient data.frame functions

```r
head(UNpop)
```

```
##   year world.pop
## 1 1950   2525779
## 2 1960   3026003
## 3 1970   3691173
## 4 1980   4449049
## 5 1990   5320817
## 6 2000   6127700
```

Some convenient data.frame functions

```r
tail(UNpop)
```

```
##   year world.pop
## 2 1960   3026003
## 3 1970   3691173
## 4 1980   4449049
## 5 1990   5320817
## 6 2000   6127700
## 7 2010   6916183
```

Some convenient data.frame functions

```r
names(UNpop)
```

```
## [1] ”year”      ”world.pop”
```

How to do operations over a data.frame column

```r
mean(UNpop$world.pop)
```

```
## [1] 4579529
```

```r
sum(UNpop$world.pop)/nrow(UNpop)
```

```
## [1] 4579529
```

How to do operations over a data.frame column

```
UNpop$world.pop/UNpop$world.pop[1]
```

```
## [1] 1.000000 1.198047 1.461400 1.761456 2.106604 2.426063
```

Homework 1

- Complete HW1, posted at `https://github.com/f-edwards/intro_stats/tree/master/hw`
- Due Wednesday, September 9 at 10AM
- Submit homework as .Rmd and .html to `frank.edwards@rutgers.edu`
- I encourage you to work in groups, but make sure you submit your own code and write-up
- #HW1 is open on Slack for Q&A. Use it!
- You can access the data through the QSS package (see how I loaded the UNpop data above)

Lab

Introducing RMarkdown

- RMarkdown allows us to combine code and text in one document
- With the proper workflow, you can do all your academic writing in one place!
- Let's work through a demo!

Installing swirl

```
install.packages(”swirl”)
```

Install the QSS lessons

```
library(swirl) # load the swirl package
install_course_github(”kosukeimai”, ”qss-swirl”)
```

Start swirl

```r
swirl()
```

Work through Swirl Intro 1

```r
library(swirl)
swirl()
```