1. Inroducing data analysis in R

Frank Edwards

9/4/2019

Introduction to Statistics

Overview of the course

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 (W 1:30-3:00, SCJ computer lab) provides us a chance to work through a problem set together
- Office hours (FE: Monday 10:00-2:00, CS: TBA) give you a chance to seek additional help with homework problems

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.

2. Wickham and Grolemund, *R for Data Science*

This book (available free at r4ds.had.co.nz/) provides a comprehensive overview of programming in R using the tidyverse packages

Supplemental materials

Arnold, *Quantitative Social Science: The R Tidyverse code*

jrnold.github.io/qss-tidy/

This website provides a complete translation of the base R code in QSS into tidyverse syntax

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 12PM the Tuesday before each lecture
- Complete homeworks using RMarkdown
- Provide adequate writing to explain and contextualize your responses / findings
- Email me (`frank.edwards@rutgers.edu`) the compiled RMarkdown output file and source code (.html and .rmd files) by the due date
- Everyone gets two free 3-day extensions throughout the semester, just let me know if you need to use it

Course communication

- Course website:

  `https://f-edwards.github.io/intro_stats/`
- Course Slack: `https://scj-introstats.slack.com`

Questions about the course?

Software installation

- R: `https://cran.r-project.org/`
- RStudio: `https://www.rstudio.com/`
- Git (optional): `https://git-scm.com/`
- GitHub course repo:

  `https://github.com/f-edwards/intro_stats`

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

```
5+3
```

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

```
5*3
```

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

```
5/3
```

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

```
5^3
```

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

R can make comparisons

```
5>3
```

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

```
5<=3
```

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

```
5==3
```

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

```
"a"=="a"
```

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

R works with objects

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

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

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

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

```
a<-a+1
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"
```

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
```

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

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

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

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

Vector 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
```

Vector indexing

```
vector2
```

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

```
vector2[1]
```

```
## [1] "a"
```

```
vector2[2]
```

```
## [1] "fancy"
```

```
vector2[3]
```

```
## [1] "vector"
```

Operations and vector indexing

```
vector1
```

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

```
vector1[2]
```

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

```
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)*

Functions!

```
vector1
```

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

```
min(vector1)
```

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

```
max(vector1)
```

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

```
mean(vector1)
```

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

```
sum(vector1)
```

Functions can work together

```r
sum(vector1)
```

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

```r
length(vector1)
```

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

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

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

```r
mean(vector1)
```

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

We can even write our own!

```
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 packaged 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-matrixes (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
```

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

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

Other means of indexing

```
UNpop[,1]
```

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

```
UNpop[1,]
```

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

```
UNpop$year
```

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

```
UNpop[,"world.pop"]
```

```
## [1] 2525779 3026003 3691173 4449049 5320817 6127700 6916
```

Some conveient 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
```

Other useful functions: *nrow()*, *head()*, *tail()*, *str()*, *ncol()*, *View()*

How to do operations over a data.frame column (the typical approach)

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

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

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

```
## [1] 1.000000 1.198047 1.461400 1.761456 2.106604 2.42606
```

We can also do this using tidyverse functions

```
library(tidyverse)
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
## Registered S3 method overwritten by 'rvest':
##   method            from
##   read_xml.response xml2
```

```
## -- Attaching packages ---------------------------------
```

```
## v ggplot2 3.1.1       v purrr   0.3.2
## v tibble  2.1.1       v dplyr   0.8.0.1
```

Common tidyverse functions

- select() selects columns by name
- filter() filters rows by condition
- mutate() creates new columns based on arguments
- rename() renames columns
- arrange() reorders rows based on value

We’ll experiment with these in lab