Data Motivation

Sunday, July 31, 2016 1:56 PM

- Finding files with data
- Extracting raw data to be used
- Data typically does not look neat and organized
    - Raw file with data that must be parsed and analyzed to find meaning
- Goal:
    - Raw data -> Processing script -> tidy data

Raw and Processed Data

Sunday, July 31, 2016 2:00 PM

- Raw data
  - Original source of data
  - Hard for data analyses
  - Data analysis includes pr

Raw:

- Orig. source
- Hard for analysis
- May only need to be processed once
- processing is a step

Processed:

- Ready for analysis
- Standards are available
- Steps should be recorded

*All technically raw

Processing Pipeline:

DNA sequencing:

DNA Chunks → colors attached

Based on levels of shown color, different sequences produced in text

take pictures and images of

Components of Tidy Data

Sunday, July 31, 2016 2:06 PM

Raw → Tidy: Components:

- Raw dataset
- Tidy dataset
- Code book - each var + values in tidy set
  - meta data
- Explicit recording of process of data

Raw data:

1. No software on data
2. No manipulations
3. No removed data
4. No summarizing data

Tidy Data:

1. Variable each column
2. Observations in rows
3. Table per "kind" of variable
4. Multiple tables - link via column
   - Tips:
     - Row at top w/ var names
     - Variable names human readable
     - One file per table

Codebook (Meta data)

1. info about vars (units!)
2. Summary choices made
3. Study design used

- Typically word / markdown

Instruction List:

- Ideally a script,
  - no parameters
  - input → raw
  - output → tidy

If not 1 script, separate into steps

Downloading Files

Sunday, July 31, 2016 2:26 PM

- Best to do downloading via program language (eg R), so that step can be done via instruction script
- Get/set working directory
- Check for + create necessary directories
- Getting from Internet: download.file():
    - Url
    - Destfile (destination files)
    - Method (sometimes necessary to specify for secure connections)
- Record when downloaded for tracking purposes

Reading Files (with R)

Sunday, July 31, 2016 2:33 PM

- Excel:
  - Download/grab file
  - Pretty straightforward from there… Google it
- XML:
  - Tags = general labels
    - Start/End tags: <section> </section>
    - Empty tags: <phrase />
  - Elements are examples of tags, specific tags
    - <Greeting>
    - <Swag>
  - Attributes are components of tags/labels
  - XML Package
  - RootNode <- xmlRoot(doc)
  - Names of root node:
    - xmlNames(RootNode)
    - Also can access like an array (rootNode[[1]])
    - xmlSApply()- apply across XML tree

Reading Databases

Sunday, July 31, 2016 3:56 PM

- MySQL
  - Open source database
  - Package: RMySQL
  - Method:
    - ucscDb <- dbConnect(MySQL(), user="",host="")
    - Result <- dbGetQuery(ucscDb, "show databases;");
    - Disconnect: dbDisconnect(ucscDb)
    - Can grab whole tables, or subsets via specific queries
    - Always close the connection when you don't need it anymore
- HDF5
  - Large datasets
  - Range of data types
  - Hierarchical data format
  - Read in groups, different indices of tables
- From the Web
  - Extract data from HTML code
  - Watch for breaching terms of service
  - Reading too much quickly can get IP blocked
  - Open connection
  - Can use XML library, htmlparser to recognize general HTML tags
  - GET(url)- Parse HTML, get info from the page
    - Websites with passwords
  - Handles: root urls that can be built upon
- APIs
  - Combine R with APIs for different apps to get data from these apps (eg Twitter)
  - After retrieving data paired to the software, process the data
  - API documentation will show what interactions with server are available
  - Use httr package
- Other sources
  - There's an R package for almost anything
    - Databases: RPostressSQL, RODBC, Rmongo
    - Images: jpeg, readbitmap, png
    - GIS data
    - Music data: tuneR, seewave

Subsetting Tricks

Sunday, July 31, 2016 11:24 PM

x[, 1] → 1st col

x[, "var1"] → name of 1st col

x[1:2, "var1"] → 1st row var1 col.

Logicals:

x[(x$var1 <= 3 & x$var3 > 11),]

x[(x$var1 <= 3 | x$var3 > 11),]

Missing values:

x[which(—),]

↳ returns corresponding indices

Sort command

sort(——)

Order

x[order(—),]

↳ orders rows in terms of that

... rows
in terms of that
increasing order

---

cbind() = Binding cols
rbind() = Binding rows

Summarizing Data

Sunday, July 31, 2016 11:32 PM

head = 6 top rows

tail = bottom 3

summary():

- Text/factor vars => give # occur
- Quantitative => give basic stat data (min, max, median, etc)

quantile():

- Different dists. + probs of quant data

table():

- auto gen table for specified set of data

Check for missing values

sum(is.na())

or

anyC)

all(condition) => does condition ever occur?

Specific Characteristics

% in % => ~~this value for another?~~

%in% => which values fall in this condition

as.data.frame( )

(ross Tabi

xtabs(Var ~ colums to grab from)

Creating New Variables

Sunday, July 31, 2016 11:41 PM

Why?

- Raw data doesn't have needed value
- Transforms necessary
- Easy add to data frames
- Common:
  - Missingness indicators
  - Cutting up quant.
  - Transforms

- Sequences create indexed datasets

Dplyr

Monday, August 1, 2016 1:41 AM

dplyr = package for data frames

dplyr assumptions:
- 1 obs per row
- column = var/measure/char.
- Primary imp. = R
- Other langs good for relational databases

- data frame
- methods of manipulation

→ dplyr → new data frame

Verbs:

- Select: Choose columns by name
  -
```
> names(chicago)
[1] "city"      "tmpd"      "dptp"      "date"      "pm25tmean2" "pm10tmean2" "o3tmean2"
[8] "no2tmean2"
> head(select(chicago, city:dptp))
  city tmpd   dptp
1 chic 31.5 31.500
2 chic 33.0 29.875
3 chic 33.0 27.375
4 chic 29.0 28.625
5 chic 32.0 28.875
6 chic 40.0 35.125
> head(select(chicago, -
```
- Filter: subset rows based on conditions
  -
```
> chic.f <- filter(chicago, pm25tmean2 > 30)
> head(chic.f, 10)
   city tmpd dptp       date pm25tmean2 pm10tmean2  o3tmean2 no2tmean2
1  chic   23 21.9 1998-01-17      38.10   32.46154  3.180556  25.30000
2  chic   28 25.8 1998-01-23      33.95   38.69231  1.750000  29.37630
3  chic   55 51.3 1998-04-30      39.40   34.00000 10.786232  25.31310
4  chic   59 53.7 1998-05-01      35.40   28.50000 14.295125  31.42905
5  chic   57 52.0 1998-05-02      33.30   35.00000 20.662879  26.79861
6  chic   57 56.0 1998-05-07      32.10   34.50000 24.270422  33.99167
7  chic   75 65.8 1998-05-15      56.50   91.00000 38.573007  29.03261
8  chic   61 59.0 1998-06-09      33.80   26.00000 17.890810  25.49668
9  chic   73 60.3 1998-07-13      30.30   64.50000 37.018865  37.93056
10 chic   78 67.1 1998-07-14      41.40   75.00000 40.080902  32.59054
> chic.f <- filter(chicago, pm25tmean2 > 30 & tmpd > 80)
> head(chic.f)
  city tmpd dptp       date pm25tmean2 pm10tmean2 o3tmean2 no2tmean2
1 chic   81 71.2 1998-08-23    39.6000       59.0 45.86364  14.32639
2 chic   81 70.4 1998-09-06    31.5000       50.5 50.66250  20.31250
3 chic   82 72.2 2001-07-20    32.3000       58.5 33.00380  33.67500
4 chic   84 72.9 2001-08-01    43.7000       81.5 45.17736  27.44239
5 chic   85 72.6 2001-08-08    38.8375       70.0 37.98047  27.62743
6 chic   84 72.6 2001-08-09    38.2000       66.0 36.73245  26.46742
```

- Arrange: reorder rows of data frame
- Rename: rename variable to something more intuitive
- Mutate: transform existing vars to make new ones
- Groupby: split into categorical data
- Summarize: self explanatory
- %>%: pipeline operator, feed into the right

Regular Expressions

Monday, August 1, 2016 4:26 AM

- Literals + metacharacters
- Literal = actual word
- ^(phrase) = phrase at beginning of line
- (phrase)$ = phrase at end of line
- [Character class]- set of chars acceptable in match (eg [Bb] takes B or b)
- Range of characters:
    - [a-z] = all lowercase
    - [a-zA-Z]
    - [0-9]
- ^ inside a char class indicates not what's in the char class
    - Eg [^?.]$ = not ? Or . At end
- . = any character
- | = or
- ? = after any expression, that it's optional
- \ = precede other metacharacters when used as liberals
- * = any number including none of item
- + = any number including 1
- {a, b} = indicates interval, i.e. The maximum and minimum occurrences
- \number = repetitions of found text expected