Week 2

Structure of Data Analysis

Steps

- define the question (scientific or business question you'd like to be able to answer)
- ideal dataset (one you would collect given no shortage of time & money)
- determine what data you can access
- obtain data
- clean it up (so you cal analyse it)
- exploratory data analysis
- statistical modelling / prediction (to answer the question you're interested in)
- interpret results (what does it mean, it a plain language)
- challenge the result (what are the potential failings?)
- synthesize / write up all the results (using the data to answer the question - story)

- create reproduceable code (so you can share your analysis with other people)

① Defining a question

Start with a general question.

↓ Can I automatically detect messages that are SPAM?

Make it concrete

Can I use quantitative characteristics of emails to classify them SPAM?

② Ideal Dataset

(depends on your goal)

- descriptive - whole population
- exploratory - a random sample with many varables
- inferential - the right population, randomly sampled
- predictive - a training and test data sets from the same population
- casual - data from a randomized study
- mechanistic - data from all ~~co~~ components of the system

③ What data you can access?

- free data on the web
- you may buy
- or generate

⟹ UCI Machine Learning Repository

④ Obtain the [raw] data

reference the source
you need to record url and
time accessed

⑤ Clean the data

is it good enough?
not → change the data

understand the source (census, sample, etc)

may need reformatting, etc
(record these steps!)

if preprocessed - make sure you understand
how

⑥ Exploratory analysis

(playing in R, etc)

kernlab - data with mails from CRAN

install.packages("kernlab")

library(kernlab)

data(spam)

dim(spam) => 4601 58

We~~e~~ need to sample a test set
and a training set
↑
for prediction

either 1 or 0

set.seed(3435)
trainIndicator = rbinom(4601, size = 1,
prob = 0.5)
↑
fair coin

table(trainIndicator)
⇓ (summary)
2314 × 0, 2287 × 1

trainSet = spam[trainIndicator == 1,]
testSet = spam[trainIndicator == 0,]

dim(trainSet)

names(trainSet)
⇓
names of colums

head(trainSet)
⇓
a first few

table(trainSpam$type)

⇓

nonspam spam
1381 906

plot(trainSpam$capitalAve ~ trainSpam$type)

plot(log10(trainSpam[, 1:4]+1))

↓

first 4 colums vs each other
(16 plots)

clustering

hCluster = hclust(dist(t(~~transform~~
trainSpam[, 1:57])))

plot(hCluster) ⇓ (cluster dendagram)

puts variables that
have similar patterns
close together

We may
take log10 + 1
to see it better

⑦ Statistical / Prediction / modelling

- Should be informed by the results of your explanatory analysis

- methods depend on questions

- pay attention to what you've done with your data on pre-processing steps (log; etc)

- report all measures of uncertainty (number of mistake you did on the test set)

⑧ Interpret results

Use the appropriate language

describes
correlates with / associated with
leads to / causes
predicts

Give an explanation => Goal: it should be understood by non-tech audience

Interpret coefficients

Example:

The fraction of characters that are dollar signs can be used to predict if an email is spam

Anything with more than 6.6% \$s is classified as spam

More \$s always means more spam under our prediction

Our test set error ~~was~~ rate was 22.4%

(9) Challenge the result.

Challenge all steps:

- Question (was it right? could you make it more specific/general?)
- Data Source (was it right data? did you get right samples? right population?)
- Processing (correctly identified variables?)
- Analysis (do we pick the right predictors?)
- Conclusions (are you interpreting too much? are you trying to say smth you cannot?)

also challenge points of uncertainty

challenge choices of terms included in the models

think of potential alternative analyses

(10) Synthesize / write-up results

State the question you raised

Summarize the analyse into a story

Include an analysis if

- it's needed for the story
- it's needed to address the challenge

Order analyses according to the story rather than chronologically

Include figures

(11) Create reproducible code

(R code)

Example

- lead with the question

  Can I use quantitative characteristics of the emails to classify them as SPAM/HAM?

- Describe the approach

  Collected data from UCI →
  created training/test sets

  Explored relationships

  Choose logistic model on training set by cross validation

  Applied to the test, 78% accuracy

- Interpret results

  Number of dollar signs seems reasonable, e.g. „Make money with Viagra $$$$!“

- Challenge the results

  78% isn't that great

  I could use more variables

  Why logistic regression?

Organizing a data analysis

Files (directory structure)

```
/data
   /raw
   /processed          (tidy!)

/figures
   /explanatory
   / final

/R                     (code)
   /raw scripts
   / final
   / R markdown        (code + writing).

/text          either
   /readme
   /text of analysis
```

R markdown - for reproducible reports

R + text are integrated

text:

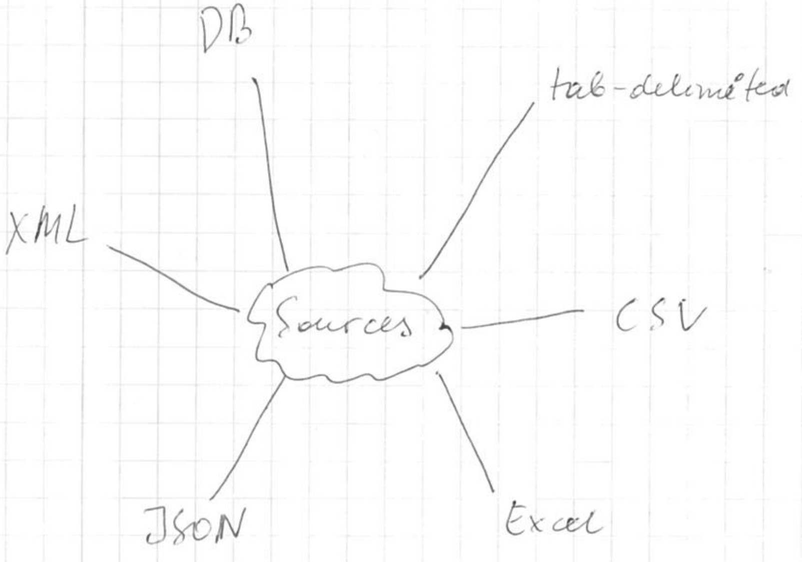
- title
- introduction (motivation)
- methods
- results
- conclusions

- references

project template! (from the cource)

Getting Data

getwd()   setwd()



from a ~~collegue~~ colleague
web
application programming int.
scrapping a web page

- download.file()   url
destfile
method

- download.file(fileUrl,
  destfile = "..",
  method = "curl") (for https)

  list.files("./data")

  dateDownload ← date()
  (for keeping the date when the file was downloaded)

- reading:

  read.table() read.csv() read.csv2()

  read.xlsx() read.xlsx2 {xlsx package)
  (maybe slower) (written in Java)

  file.choose() ← opens a file open dialog

- Connections

  file, url, gzfile, bzfile
  remember to close connection

  json {RJSONIO}
  fromJSON(con)
  └ file

- writing

  write.table
  save ← saves R object

  save.image - saves everything in your working directory
  (rda files)

  Load - opposite of save

paste and paste0

for pasting ~~string~~ character strings together

paste0 is the same as paste, but sep=""

- scrappy

  Lib(XML)
  ~~con = url("http...")~~

  ~~html = readlines~~

  html3 ← htmlTreeParse("http", ...)
  xpathSApply(html3, "//title", xmlValue)

RMySQL - mysql connection
R bigmemory - for big datasets

Data Resources

data.gov   open government
data.gouv.fr
data.gov.uk

gapminder.org   public health

etc   asdfree.com   survey data

kaggle.com - contests

+ lots of resources (see the 6th slide)

Summarizing data

Why?

- to big to look at
- find problems

missing values
outside of ranges
wrong units
mislabeled
wrong class

- dim() dimentions

x•yz (how many rows, cols)

names(X) columns' names

nrow ncol

· quantile (x) - like persentile

0% 25% 50% 75% 100%

range of variables

· summary (x)

summarizes quantitive quacitive variables.

· classes!

sapply (eData [1,], class)

⇓

class of every element of 1st row
is it loaded properly ?

· unique (x) length (x)

table ← unique + count of each time it appears

table (x) ← one var. (1 dim)

table (x, y) ← 2-dimentional table

· any() all()

any (e [1:10] > 10) all (e [..] > 10)

is there any TRUEs? are all TRUEs?

edata[edata$L > 0 & edat$n > 0,
c("L",7,"n"))

can be "|" (or)

sum(is.na(a$l))
how many NAs?

or

table(is.na(a$L))

⇓

FALSE TRUE
10 20

table by default doesn't show NAs,
table(x, useNA="ifany")

- summarizing cols/rows

rowSums	colSums
rowMeans	colMeans

colMeans(reviews, na.rm=TRUE)

Data Munging basics

(munge - вносить изменения в файл, обычно необратимые;

Data should be tidy!

+ names are easy to use and informative

obvious mistakes are removed

variables are internally consistent

appropriately translated

List of Munging operations (partial)

fix var names
create ~~vars~~ new vars
merge
reshape
deal with missing data
transform
remove inconsistent

} Should be Recorded!

- Character vectors!

tolower, toupper

stringsplit (remove dots, $, etc)

(g) sub ("_", "", names(x))

- cut() for quantitative vars in ranges

cut(x, seq(0,3600, by=600))

⇓ cuts into ranges (in what range a value is) => factor

- adding extra variables

f$new.var = data.vector

- merge() for merging data

x

merge(r, s, by.x="sol.id", by.y="id",
~~all~~=TRUE)

all

y

- sort(), order()

- reshaping

melt(X, id.vars="people", → what ise id
variable.name "treatment" → observation
value.name="value")

	treatA	treatB	people
1	NA	5	John
2	1	4	Jane
3	2	3	Mary

=>

	people	treatment	value
1	John	trA	NA
2	Jane	trA	1
3	Mary	trA	2
4	John	trB	5
5	Jane	trB	4
6	Mary	trB	3

↑ tidy!