133 - Blend of 3 topics

- statistical, computational, technology (data)

Review will separate these 3 topics. Begin w/ Statistical

Statistical analysis is impacted by data provenance
and computational considerations

- population or sample
- representative sample
- observational or experimental

provenance - data source, how data collected
impacts what we can generalize from our findings
and approach we take in analysis
see, e.g. comparison of student evaluations of Gilberts shift

Exploratory Data Analysis (see §2.7)
Useful in cleaning data to ensure properly read & cleaned
Useful in first examination of data -

- keep an open mind; look for surprises
- look at the distribution of values for a variable:
  modes, tails, symmetry, gaps & outliers
- transformations: log, squareroot (often for counts)
  to symmetrize distribution & bank to 45° relationship & make variability more homogeneous

Method of Comparison

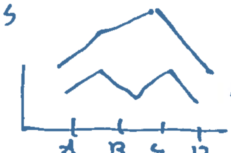
- Examine subgroups
- Compare to benchmarks
- Trends and patterns in relationship between variabl

§ 3.2 Graphics an important part of EDA & more formal analysis

Match plot type to data type

<u>univariate</u>
- continuous - density curve & histogram
- discrete - bar plot — few obs - strip plot
- categorical - bar plot — few obs - dot chart

<u>bivariate</u>
- 2 continuous - scatter plot
  - line plot (if one is time)
  - smooth curves (large # obs)
- 2 categorical - side-by-side bars
  - mosaic plot
  - line plot -



lines for 2nd var

categories of 1 var

- 1 cont + 1 categorical -
  - superposed density curves ; juxtaposed histograms
  - side-by-side box plots, violin plots

More than 2 variables
- Use color, plotting symbols, facets

maps if have lat + lon variables

§ 3.3 Guidelines

(A) Make the Data Stand out

- Fill the data region - choose limits to encompass all data (w/ possible exceptions (or outliers))
- use prominent plotting symbols
- avoid chart junk
- eliminate superfluous lines, colors, etc
- choose transformation to fill data region

(B) Facilitate Comparison

Use color to bring in additional information
Include reference markers
Color palette choice   sequential, diverging, qualitative
Scale - bank to 45°   — better able to discern relationship
symmetrize distribution
Super-pose & Juxtapose (depending on plot-type & variable type)
Jitter points to see density, and transparence
Determine the important comparison - and have that
dictate the arrangement of bars, lines, etc.
Length is easier to compare than angle & area
so avoid pie charts & scaled blocks (unless necessary)
Color that can be examined for long periods, and that
have similar luminance

(C) Information rich
Legend, Axis label, Title, Caption
Symbol size, shape, color have meaning
Reference lines & markers

Utilize randomness to understand what to expect and typical deviations

Monte Carlo Method - simulate an independent process
$x_1, x_2, \ldots, x_n$ from some distribution

Law Large Numbers: $\bar{x}$ converges to expected value
Prop of $x_i \leq c$ → chance a randomly generated value $\leq c$

Central Limit Theorem:
$\bar{x}$ is a random quantity and has its own distribution
as sampling grows $\bar{x}$'s distribution looks roughly normal and is centered at center of distribution generating the data with a spread shrinking like $1/\sqrt{n}$

Example Roulette $x_i = 1$ if #17 appears on i^{th} spin
0 otherwise

$\frac{\text{\# times 17 appears}}{\text{\# spins}}$ looks roughly normal for large # spins

```
urn = c(1, rep(0,37))
sum(sample(urn, n, replace=TRUE))/n
sampProps = replicate(4000, ↓ )
hist(sampProps)
```
looks normal for n large ~ 1000

Resampling - Use design of experiment or random what-if scenario to compare study outcome to chance distribution

see Gilbert example (and Final exam practice question)
see Student Eval example

Cross-Validation is another example - we review this notion after discussing modeling

Modeling - we divide this into 2 types

unsupervised - this is more exploratory
there is no prediction

supervised - there is a predictor variable
(we have almost exclusively examined classification type predictions)

Methods depend on a loss function (supervised) or a metric (unsupervised)

The choice depends on the type of data

L_2 - sum of squared differences or errors

L_1 - sum of absolute differences or errors

Jaccard index - counts of categories scaled

Likelihood Ratio - ratio of chances (see Naive Bayes)

Jensen-Shannon - similar to LR for term frequency

Gini Index - purity measure (see recursive partitioning)

Which Formulas to know?

L_1, L_2, Jaccard Index, Likelihood Ratio

For Jensen-Shannon, just know term freq & doc freq

Not Gini Index

Unsupervised Methods   Concepts & Algorithm

Hierarchical Clustering

Concept: Look for clusters of observations, i.e. pts close together

Agglomerative Algorithm

Begin with all points in separate clusters of size 1

Find distance between all pairs of clusters

Join the 2 closest clusters

Repeat until all clusters joined

Metric for distance between 2 clusters – complete linkage

largest distance between any 2 points where the points are in different clusters

Read and draw binary tree (2 clusters joined at each step)

AKA Dendrogram

slice into clusters for a particular value of distance

Multidimensional Scaling

Concept: Find a projection of observations into 2 dimensions that tries to preserve all pairwise distances

Look for clusters of observations

Algorithm: NA

Supervised Learning – Concepts & Algorithms

K-NN –

Concept: For any observation find the k closest observations and use the value of their outcome variable to predict the observation's outcome.

Here the outcome variable may be a classification (e.g. Android vs iPhone) or a continuous variable.

Algorithm – Find all n pairwise distances between the point that you are trying to predict the outcome for and the n points in your training set; order them; find the k-smallest; take the k-values of the outcome to estimate the outcome for the test point.

Combine via equal wts or weight inversely proportional to distance.

Recursive partitioning

Concept: Begin with all observations in one group/node. Split the group into 2 according to Yes/No response about the value of one of the variables, e.g.

$X \leq c$; $Y == \text{"m"}$
continuous categorical

Choose the question/split that makes the resulting nodes the most pure.

Continue splitting nodes in this way.
For the final tree – use these binary decisions to make predictions. For a test observation, answer

use the composition of the leaf node to predict the outcome variable.

<u>Algorithm</u>: NA There are several techniques to avoid overfitting, such as a lower bound on the number of obs in a node to consider splitting & a lower bound on the increase in purity to continue splitting,

<u>Naive Bayes</u>

concept: Approximate the chance a new observation belongs (outcome) to a particular category given its value for the other variables, e.g.

Chance (spam | word vector)

The approximation uses Bayes rule and a Naive assumption of independence of variables

Algorithm: for word vectors & binary classification

log-likelihood ratio:
$$\frac{\text{Prob}(\text{spam} \mid \text{wordvector})}{\text{Prob}(\text{ham} \mid \text{word vector})} \approx \prod_{i=1}^{\text{Bag of Words}} \frac{P(\text{word}_i \mid \text{spam})}{P(\text{word}_i \mid \text{ham})} \times \frac{P(\text{spam})}{P(\text{ham})}$$

Each term is either prop of spam with word or 1 − prop of spam w/ word, dependi on whether that word is in email or not

Take the log of this ratio

Model Assessment

Before we build our predictor, divide our data into 2 parts : test set and training set.

Put the test set aside. Build model with the training set

Make predictions for the test set

Check to see how accurate our predictions are :

Error - Average Loss $\frac{1}{\#\text{test}} \sum_{i=1}^{\#\text{test}} \ell(\text{outcome}_i, \text{Pred}_i)$

based on trained model

ℓ is the loss function,

Examples (similar to distances)

ℓ_2: $(\text{outcome}_i - \text{pred}_i)^2$ ℓ_1: $|\text{outcome}_i - \text{pred}_i|$

When outcome is binary - ℓ_1 & ℓ_2 both reduce to # of incorrect predictions. We often want to distinguish between the 2 types of errors (pred 1 & out 0 vs pred 0 & out 1)

We use the confusion matrix

		Prediction 0	Prediction 1	
Truth	0	n_0	n_0	1
	1	n_1	n_1	1

Cross-Validation - Model Building often involves nuisance (tuning aka) parameters, e.g. k in k-NN, complexity parameter in many.

We choose the tuning parameter with cross-validation where we imitate the model assessment scenario within the training set.

V-fold cross-validation There are many types of CV. With v-fold, we randomly partition the training set into v equi-sized sets P_j. Then for each P_j, we build a model based on the remaining observations and assess the model with P_j

$\frac{n}{v} = m$

$$\frac{1}{v} \sum_{j=1}^{v} \frac{1}{m} \sum_{obs_i \in P_j} l(\text{outcome}_i, \text{pred}_i, \lambda)$$

nuisance parameter

built using $\bigcup_{k \neq j} P_k$

Choose λ^* that minimizes the loss

After that, build the model with all the data in the training set and λ^*.

Finally assess model as on previous page

Computational Topics

We will not review the material prior to the midterm. Please review these topics on your own.

Here we focus on computational concepts related to simulation and model building. The topics are somewhat piece-meal.

Representation of information in the computer

Information is stored as sequences of bits (0s & 1s) organized into bytes (8 bits)

Characters today are Unicode w/ 8, 16 or 32 bits called UTF-8, UTF-16, and UTF-32

Unicode mappings are compatible with ASCII, which uses 7 bits to map to 128 characters. These include upper and lower case letters, digits, and a few symbols like # and $

Numbers are represented using binary system. For example

01101001

$$0\times 2^7 + 1\times 2^6 + 1\times 2^5 + 0\times 2^4 + 1\times 2^3 + 0\times 2^2 + 0\times 2^1 + 1\times 2^0$$

$$= 64+32+8+1 = 105$$

Numeric values are represented with double-precision floating point

8 bytes or 64 bits divided as

sign : 1 bit

exponent : 11 bits

mantissa : 53 bits (stored as 52 - with convention that first digit is a 1)

scientific notation $(-1)^{sign}\left(1+\sum_{i=1}^{52} b_{52-i}\, 2^{-i}\right)\times 2^{e-1023}$

Implications –

$x ==$ value not necessarily a good idea
use all.equal instead. Can set the tolerance in all.equal.

Order of operations can matter, e.g.
add a lot of small values then one large
may be different than start with a large value
and add small values to it
e.g. 1st eigenvalue in adhoc should be 1

Colors – not covered

Random Number Generation

Pseudo – not truly random – uses an algorithm and is deterministic

0) Begin with a seed x_0
1) From the seed (input) generate a number $f(x_0) \to x_1$
2) Use the previous number (input) generate the next number
3) Repeat step 2 until have n numbers $f(x_1) \to x_2$

Congruential generator $(b * x_i) \bmod m \to x_{i+1}$

Advantage: Can replicate simulation studies

Environment, Scope, Lazy Evaluation

Global environment contains objects defined and sourced into our workspace

When we call a function, a new workspace (aka frame) is created and it contains the inputs to the function call. This frame has a parent environment, which is the environment in which the function was defined.

When we call a function and pass in objects as inputs, copies of them are made and placed in the function's environment

As code is evaluated within the function, new variables are created in the function's frame. If a variable is referred to that doesn't exist in the frame, then R searches for it in the frame's parent environment (and if not found, look in the parent environ of that environ.)

Lazy Evaluation: The inputs for a function call are not evaluated until they are needed. R sets up a call frame for the function with the input arguments as variables. However these are only associated with an expression, and the expression is not evaluated until the variable is referenced in the code.

We can assign a value to a variable in the parent environment with <<-, e.g. $x <<- 2$

Specify inputs in a function call: Order named arguments, Then unnamed arguments assigned left to right to remaining unassigned parameters

Practice good programming style