Economics 217 - Data Science and Time Series

- Topics covered in this lecture
  - Pattern matching
  - Anomaly Detection
  - Optimal clusters with the "Gap Statistic"
  - Textual Analysis
- General Question: How do we find similar and dissimilar time series?

Time Series and Clustering

- Our previous treatment of clustering tried to group geo-coordinates by their proximity
  - Each observation was paired with a latitude and longitude
- One can, of course, define observations by their time series
  - Financial data or macroeconomic data
  - Textual data (a ordering of alpha-numeric characters, or words)
  - Medical data (eg. EKG, etc..)
- Technique is similar - define a distance, and clustering method
- Distance
  - **Numeric:** Similar to before (Euclidean distance, other metric)
  - **Textual:** Measures of similarity (eg. intersection of words/characters)

Data: US Housing Prices by MSA

- Like your first exam, a housing price index by metropolitan statistical area
  - Housing prices normalized the 1990
  - Only complete series are included (no missing prices)
- Objectives
  - Find similar series
  - Detect "outliers" or anomalous series
  - Find optimal housing clusters

Data Processing

- Load Data

```
x<-read.csv("/Users/acspearot/Documents/HPI_AT_metro.csv",
header=TRUE,na="-")
```

- Restrict years to 1990 and onward

```
x<-subset(x,Year>=1990)
```

- Create a Year-Quarter dummy

```
x$YearQuarter<-paste(x$Year,x$Quarter,sep="-")
```

- Create a list of MSAs

```
MSAs<-sort(unique(as.character(x$MSA)))
```

- Create a matrix

```
y<-matrix(nrow=length(MSAs),ncol=nrow(subset(x,MSA==MSAs[1])))
for(i in 1:length(MSAs)){
   sx<-subset(x,MSA==MSAs[i])
   y[i,]<-sx$Index/sx$Index[grep("2000-1",sx$YearQuarter)]
}
```

- Keep observations with a complete time series since 1990

```
completeseries<-grep(0,rowSums(is.na(y)))
y2<-y[completeseries,]
MSAs<-MSAs[completeseries]
```

Use hclust to find housing clusters

- Use the hierarchical clustering procedure to find similiar housing time series
- Define distance matrix between series:

```
distmat<-dist(y2)
```

- Run the full hierarchical clustering procedure:

```
results.complete<-hclust(distmat,method="complete")
```

- "Cut the tree" at 10 clusters:

```
cluster.complete10 <- cutree(results.complete, 10)
```

- Create data frame to view the results:

```
res<-data.frame(MSAs,cluster.complete10)
```

- Do these clusters look sensible? Where are the metro areas you've heard of?
- Try with 50 clusters:

```
cluster.complete50 <- cutree(results.complete, 50)
res<-data.frame(MSAs,cluster.complete50)
```

Outlier Detection with DBSCAN

- DBSCAN groups observations into high density areas, with outliers considered noise. Density determined by:
  - *eps* is the radius to determine density.
  - *MinPoints* determines a minimum number of points within a radius
- Three types of points:
  - Core: The interior. A point is a core point if there are at least *MinPoints* within a distance of *eps*.
  - Border: A border point is not a core point, but falls within *eps* of a core point.
  - Noise: Neither a core point nor a border point.
- For our purposes, we wish to use these parameters to iteratively find the most dissimilar observations (or anomalous time series). To do this, we will:
  - Start with a very high value of *eps*, such that there is only 1 cluster.
  - Sequentially reduce eps until outliers are identified

Outlier Detection with DBSCAN (cont.)

- Load dbscan library

```
library(dbscan)
```

- Loop through different values of *eps* to find the first outlier,

```
for(i in seq(10,1,by=-0.1)){
   results.dbscan<-dbscan(distmat,eps=i)
   count<-sum(results.dbscan$cluster==0,na.rm=TRUE)
   if(count>0)break
}
```

- Collect the results

```
res$dbscan1<-results.dbscan$cluster
```

- Loop through different values of *eps* to find the 10 first outliers

```
for(i in seq(10,1,by=-0.1)){
   results.dbscan<-dbscan(distmat,eps=i)
   count<-sum(results.dbscan$cluster==0,na.rm=TRUE)
   if(count>10)break
}
res$dbscan2<-results.dbscan$cluster
```

Optimal Clustering - the "Gap Statistic"

- There are a number of methods to evaluate the optimal number of clusters
  - Compare clusters to existing classifications.
  - Use a modified cross-validation, other ad-hoc techniques
- One particular method, the "Gap Statistic", is more theoretically motivated
  - Tibshirani, Walther, and Hastie (2001), *Journal of the Royal Statistical Society*
- The Gap Statistic is based on the idea that a clustering algorithm will find clusters from random data
  - Random points can be arranged in clusters, by chance
  - Compare how a clustering procedure on real data compares to a clustering procedure on random data
  - Choose the smallest number of clusters such that the algorithm maximizes the "tightness" of clusters on the real data compared to fake data. (there is a specific metric for this)
- The Gap statistic can be used with any clustering technique that requires the choice of number of clusters to begin with.

Calculating the Gap Statistic

- Define:
  - x_{ij}, $i = 1, 2, .., n$, $j = 1, 2, ...p$, where i are observations and j are characteristics
  - $d_{ii'}$ is the distance between i and i'
  - C_r is defined as a set of observations in a cluster, r.
  - n_r is the number of observations in a cluster, r.
- The Gap statistic is defined as:

$$Gap_n(k) = \mathbf{E}_n^* (\log (W_k)) - \log (W_k)$$

  where
  - $\mathbf{E}_n^* (\log (W_k))$: the null within-cluster average distance with k clusters
  - $\log (W_k)$: the model within-cluster average distance with k clusters
- To calculate using the model:

$$W_k = \sum_{r=1}^{k} \frac{1}{2n_r} \sum_{i,i' \in C_R} d_{ii'}$$

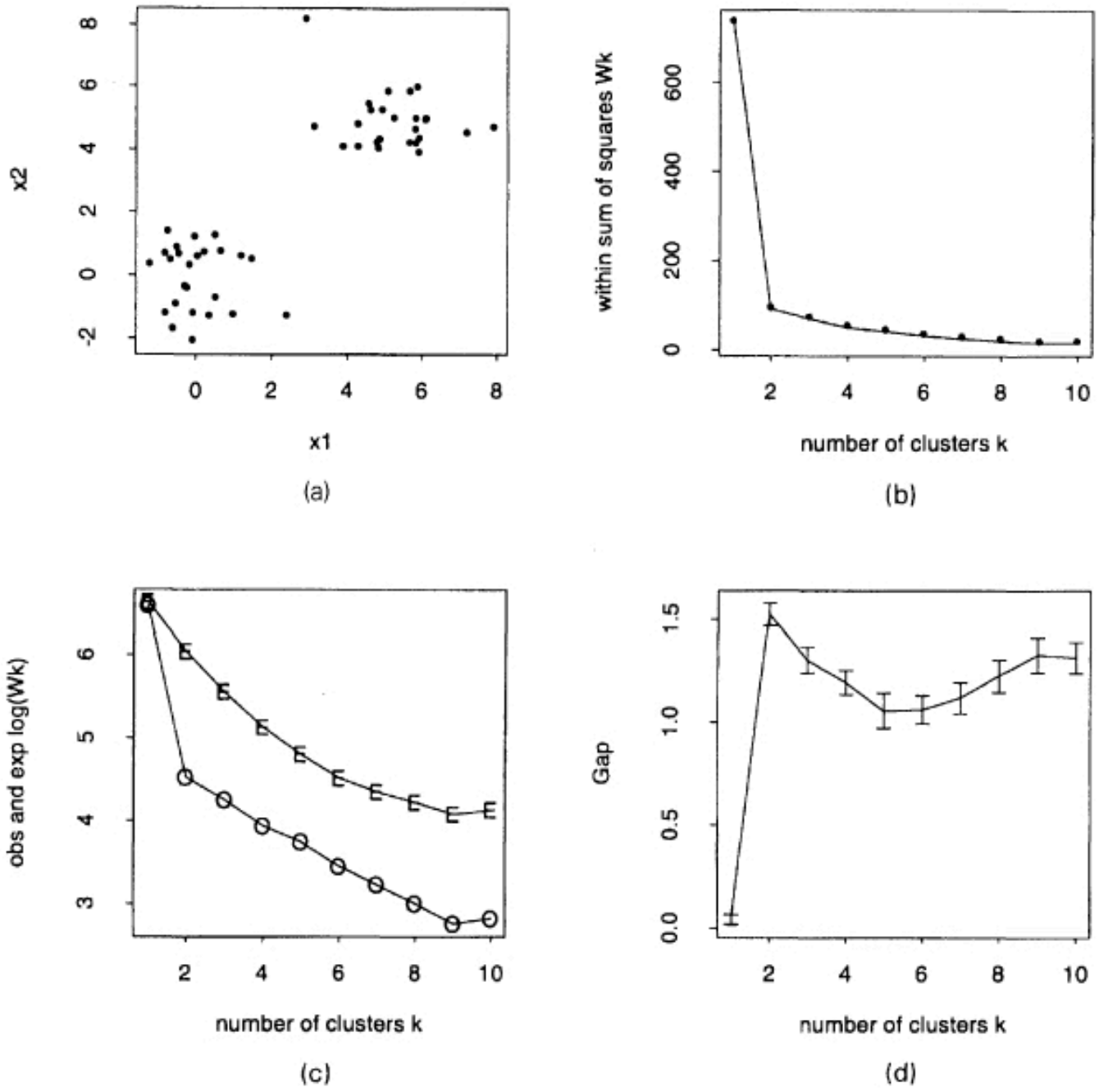

Fig. 1. Results for the two-cluster example: (a) data; (b) within sum of squares function W_k; (c) functions $\log(W_k)$ (O) and $\hat{E}_n^*\{\log(W_k)\}$ (E); (d) gap curve

The Idea Behind the Gap Statistic

- For the null, use a bootstrap with B replications.
  - Draw random data from a distribution similar to each j (eg. from the same range of data for each j)
- Then calculate the "null" log within-cluster average distance:

$$\mathbf{E}_n^*(\log(W_k)) = \frac{1}{B}\sum_{b=1}^{B}\log\left(\sum_{r=1}^{k}\frac{1}{2n_r^b}\sum_{i,i'\in C_R^b} d_{ii'}^b\right)$$

  where sd_k is the standard deviation of the null statistic at k.

- **Optimal Cluster Rule**: Use smallest k such that:

$$Gap(k) > Gap(k+1) - sd_{k+1}$$

- Intuition:
  - We want the tight clusters relative to the null: larger $Gap(k)$
  - If we are to accept more clusters, we need the new $Gap(k+1)$ to be sufficiently higher than $Gap(k)$
  - If there exist multiple k such that this condition is satisfied, choose parsimony (fewer clusters)

An Example with the Gap Statistic

- The library NbClust has techniques to test for cluster performance
- Load NbClust library

```
library(NbClust)
```

- Assign row names to the data matrix:

```
row.names(y2)<-MSAs
```

- Run a hierarchical clustering procedure (with the complete linkage), and determine optimal clustering via the gap statistic

```
GapComplete<-NbClust(y2,index="gap",method="complete")
```

- Report number of clusters

```
GapComplete$Best.nc
```

- Report clusters

```
as.matrix(GapComplete$Best.partition)
```

- Report smallest cluster

```
SmallCluster<-GapComplete$Best.partition==2
as.matrix(GapComplete$Best.partition[SmallCluster])
```

Textual Distance

- In the course of your data science career, you may be increasingly asked to engage on textual data to compare objects or extract information
- Edit-based distances
  - Distance is represented by the number of edits to turn one string into another
  - Eg. Insertions, substitutions, deletion, etc..
- Q-grams
  - Comparison of q-character sequences between strings
  - Eg. (q-gram), Jaccard, and cosine
- Hueristics - practical rules
- We'll given some examples for each using the "stringdist" library

```
install.packages("stringdist")
library(stringdist)
```

Edit Distances

- The basic types of edits:
  - Substitution: 'foo' → 'boo'
  - Deletion: 'foo' → 'oo'
  - Insertion: 'foo' → 'floo'
  - Transposition: 'foo' → 'ofo'
- **Generalized Levenshtein Distance:**
  - Weighted number of insertions, deletions, and substitutions to turn one string into another
- **Damerau-Levenshtein Distance:**
  - Allows for adjacent characters to be transposed
- Levenshtein distance comparing "alan" and "allen"

```
stringdist('alan', 'allen', method='lv')
```

Q-Grams

- Q-gram based distances deal with matching strings of q consecutive characters
- **Jaccard Distance** is a simple comparison of the intersection and union of all unique q-length sequences

$$Distance = 1 - \frac{|Q(s;q) \cap Q(t;q)|}{|Q(s;q) \cup Q(t;q)|}$$

  where $Q(s;q)$ is the set of q-length sequences in s
- **Q-gram distance** calculates the number of non-shared q-grams between the two strings
- In both Jaccard and Q-gram, you can use the functions "intersect" and "union" after vectorizing text strings, but there are packages that do this automatically
- Example with "alan" and "allen"

```
stringdist('alan', 'allen', method='jac',1)
stringdist('alan', 'allen', method='qgram',1)
stringdist('alan', 'allen', method='jac',2)
stringdist('alan', 'allen', method='qgram',2)
```

Q-Grams (cosine)

- **Cosine distance**: measures one minus the cosine of the angle between two vectors
- Translate text into vectors using "term frequency"
- Eg: "alan" vs. "allen", q=1
  - Vector space: (a, e, l, n)
  - $v('alan', q = 1) = (2, 0, 1, 1)$
  - $v('allen', q = 1) = (1, 1, 2, 1)$
- Eg, "alan" vs. "allen", q=2
  - Vector space: (al, la, an, ll, le, en)
  - $v('alan', q = 2) = (1, 1, 1, 0, 0, 0)$
  - $v('allen', q = 2) = (1, 0, 0, 1, 1, 1)$
- Formula:

$$CosDistance = 1 - \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

where A_i and B_i are vectors for two strings based on term frequency

Application: Textual Analysis of Fed Statements

- Textual Analysis is used in economics and business
  - Analyzing fed minutes/transcripts
  - Analyzing corporate earnings calls
  - An unsurprisingly, your posts on any social media
- On the website, I have compile 6 opening paragraphs of fed statements from 2013

```
x<-read.delim("/Users/acspearot/Documents/Classes/Econ 217/Fed Statement", sep="\t")
x$Date<-as.character(x$Date)
x$Text<-as.character(x$Text)
```

- As a basic command, you can search for phrases using "grep"

```
grep("economic activity has been expanding",x$Text[1])
grep("economic activity has been expanding",x$Text[2])
grep("economic activity has been expanding",x$Text[3])
grep("economic activity has been expanding",x$Text[4])
grep("economic activity has been expanding",x$Text[5])
grep("economic activity has been expanding",x$Text[6])
```

Application: Textual Analysis of Fed Statements

- There are countless ways of analyzing textual data for economic information, but let's just measure their distance
  - The Fed is careful with their language - analyzing deviations could indicate changes in economic conditions.
- Compare fed minutes with a 20 character cosine distance

```
stringdist(x$Text[1],x$Text[2], method='cosine',q=20)
stringdist(x$Text[1],x$Text[3], method='cosine',q=20)
stringdist(x$Text[1],x$Text[4], method='cosine',q=20)
stringdist(x$Text[1],x$Text[5], method='cosine',q=20)
stringdist(x$Text[1],x$Text[6], method='cosine',q=20)
```

- Obviously this is overly simplistic, but can imagine enriching this in a number of dimensions.