Hello,

The following work is one chapter out of an upcoming Handbook designed to give journalists the tools and information to be able to engage in contemporary, sophisticated, statistical analysis. The main editors of the Handbook are Teresa Bouza, and Leonid Pekelis. As this is only a (less than final) draft of a single chapter, we ask you not to distribute it too widely.

To this end, this work is licensed under a Creative Commons Attribution-NonCommercial 2.5 License, <http://creativecommons.org/licenses/by-nc/2.5/>. This means you are free to copy and reuse any information in this document (non-commercially), as long as you tell people where it's from.

We hope you enjoy the information inside, and find it useful.

Sincerely,

Teresa Bouza
Leonid Pekelis

Chapter 2: Classification & Clustering

author: Leonid Pekelis

The dataset:

The first dataset we'll look at is from data.gov - a collection of almost 6000 datasets published by the US government. The specific dataset is the US Overseas Loans and Grants (Greenbook) data. Here's a descriptive quote taken directly from the website.

> These data are U.S economic and military assistance by country from 1946 to 2010. This is the authoritative data set of U.S. foreign assistance. The data set is used to report U.S foreign assistance to Congress as required by the Foreign Assistance Act, Section 634.

What's more interesting for us is a description of the data itself: around 300 countries that the US has given aid to over the past 60 years. Each country is further subdivided into different program types such as Development Assistance, Child Survival and Health, HIV/AIDS directed assistance, etc. Any country that received over $500,000 in aid since 1945 has it's own entry, while countries receiving less than this are aggregated into general categories such as Eurasia (Other). So what we have is a dollar amount for each (country,program,year) triple.

The methods / Terms:

The analysis we'll use on this data set has been coined unsupervised learning. The best way to describe unsupervised learning is to define **supervised learning**. In supervised learning one has a few examples of what the right answer is. Going back to the intro, this might be the outcomes of coin flips for different flip strength. Or it might be a number of tweets that you painstakingly read and classified on a scale of political conservatism. The point is with supervised learning you are trying to infer a relationship between your data and the answer. This is usually with the intention of making future predictions.

With **unsupervised learning** you don't have any examples of "the answer." In fact, even the question is usually not well defined. The idea then, is to make some motivated inferences on patterns or associations in the data. The specific methods we'll use are clustering methods - k-means and hierarchical clustering - and association rules. An excellent reference is Chapter 14 of The Elements of Statistical Learning by Trevor Hastie, Rob Tibshirani, and Jerome Friedman. A free pdf of the book is available at the author's website at: http://www-stat.stanford.edu/~tibs/ElemStatLearn/ . It is especially useful for those seeking a more technical treatment of the materials presented here.

K-means is an algorithm used to partition a data set into K groups, where K is a number that the user (you!) picks beforehand. The output from K-means is a label for each data point indicating membership in groups 1,...,K, as we would expect. But in addition, we get a representative of each group - the average of all points belonging to the group. For example, the dashed red line in Figure x shows donations to the average country in group 1, and similarly orange for group 2. But how do we know the grouping is meaningful? The objective of K-means is to assign data points to groups in such a way that within any group, the members of that group are all close to the group's average. All the countries in group 1 have donation patterns that resemble the dashed red line fairly well. They're generally different from countries in group 2 in the same way that the dashed red line differs from dashed orange.

A **heatmap** is a plot of data that can be represented as a matrix (for example, counties by years with donation amount in the cells). Each cell is given a rectangle, and the color of the rectangle corresponds to the magnitude of the cell amount. The heatmaps in this section use red for low magnitude, moving through orange and yellow and finally to white for highest magnitude cells.

Bi-clustering is a term used to define simultaneous clustering of rows and columns of data that can represented as a matrix. Considering a heatmap of the data, single clustering of the rows or columns is like picking out either horizontal or vertical strips. Bi-clustering instead finds blocks, or the intersection between a horizontal and vertical strip. Generally, a good block will show similar coloring throughout. And a bi-clustered matrix will look like a plaid shirt.

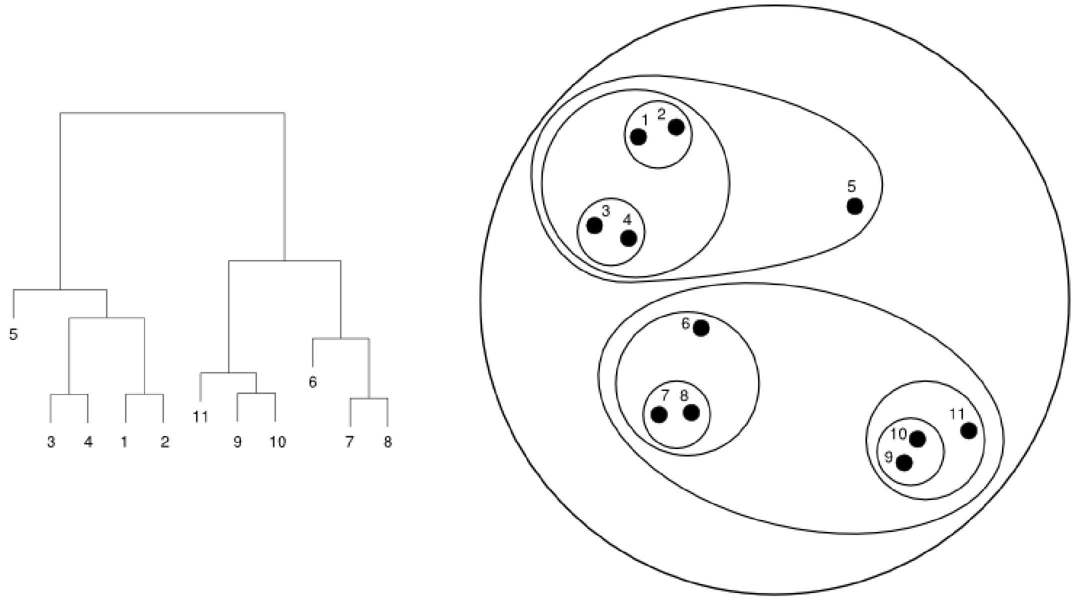
Scaling is a term that means we subtract the average from each observation, and then divide that difference by the standard deviation. For figure x, we scaled by rows, or countries, getting the following equation

$$\$_{(Country\,i,\,Year\,j)} \rightarrow \frac{\$_{(Co\,i,\,Yr\,j)} - Avg_{(Co\,i)}}{SD(Co\,i)} = \#\,SDs\ \$_{(Co\,i,\,Yr\,j)}\ is\ away\ from\ Avg_{(Co\,i)}.$$

Expressing data as the number of standard deviations from the average standardizes the data to the same units - a row of all $5m entries is the same as a row of all $5 entries, all zeros. This can be good or bad depending on the application, but for clustering years, it makes sense to put all the countries on the same footing, so one country isn't dominating how we choose clusters.

One advantage of **heirarchical clustering** over K-means is that you don't have to choose the number of groups K. In fact, that's the whole point. One view of heirarchical clustering (there are others) is you start with as many groups as observations. Then you start merging. First the closest observations to each other combine to form pairs. Then the next closest, and maybe a close third point is absorbed … all the way until only a single hive group remains. As usual, pictures help with the explanation, see figure below.[1]

[1] This image was borrowed from http://cs.jhu.edu/~razvanm/fs-expedition/tux3.html, a *very* extensive



The output from heriarchical clustering is almost always a *dendrogram* - a fancy word for a tree. Textbooks like to describe trees in terms of roots, parents, daughters and terminals, which is at least 3 too many metaphors for me. The lowest points of the tree are leaves, which correspond to individual observations. Lines are branches. They form nodes where they meet, corresponding to a new group. The top meeting point is the root node. The height of each node represents how big of a circle we had to draw in order to capture all the observations in that group (see figure above).

A useful property of these trees is we can chop the tree at any vertical level we want. This gives as many groups as branches the horizontal cut intersects, with group membership all the leaves connecting to the branch. If we cut the tree in figure x near the top, we would have 2 groups, points 1-5 in one group, and 6-11 in the other. Yet another useful property is large distances between successive meeting of branches indicate natural groups. For instance, in figure x, it takes a while for those last two groups to merge, indicating there might be a natural separation between those two, which we can confirm by looking at the plot of circles.

Here's a pretty big **caution** to interpreting a tree as a summary of the data. First, different methods for building trees, or small changes in the data can lead to very different trees. Second, heirarchical clustering methods give a tree structure to data regardless of whether a tree structure actually exists or not. I could give heriarchical clustering a bunch of random points and it would find some tree to represent those points. Section 14.3.12 of ESL gives a method for testing whether your data looks like a tree, called the *cophenetic correlation coefficient*. But it's

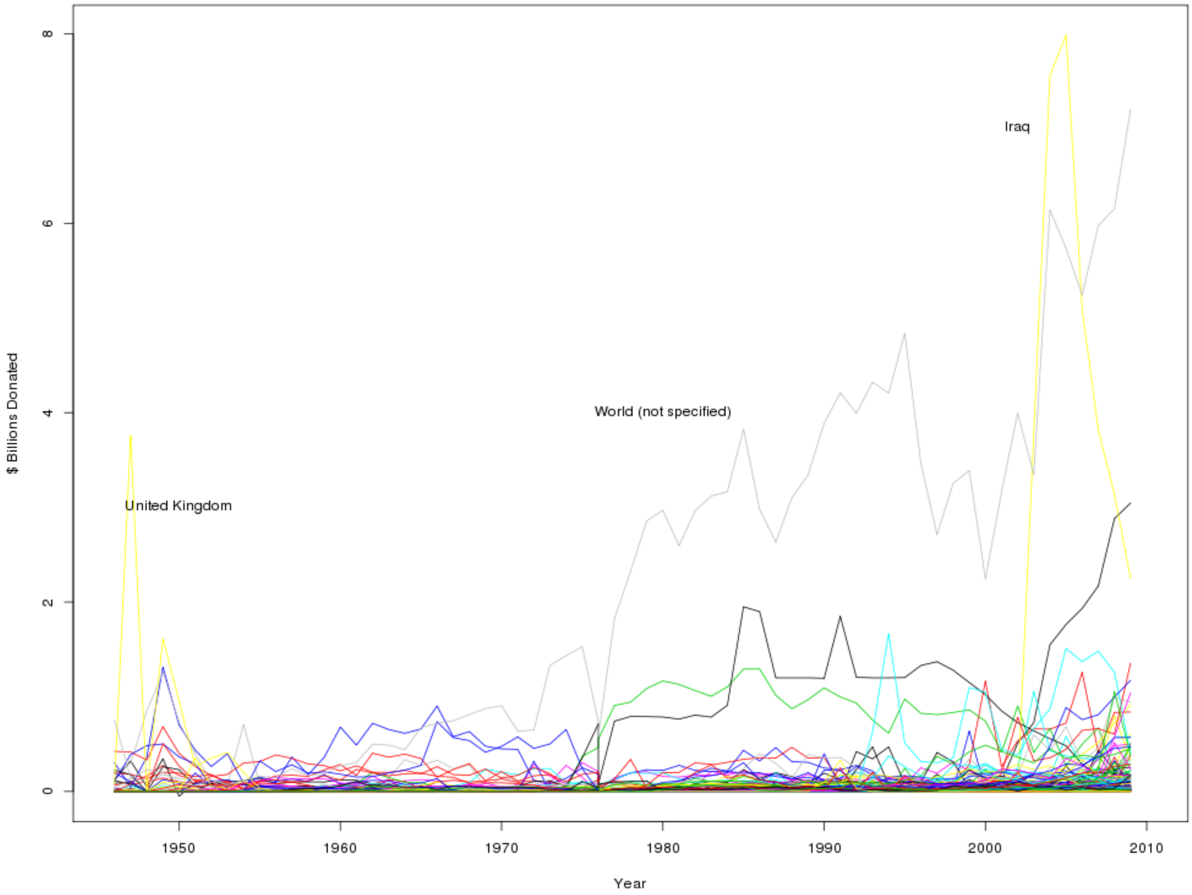
clustering analysis of the external symbols for modules of a specific Linux Kernel, in case you're into that.

complicated and doesn't seem to work well anyways. My suggestion is just be wary of reading too much into trees.

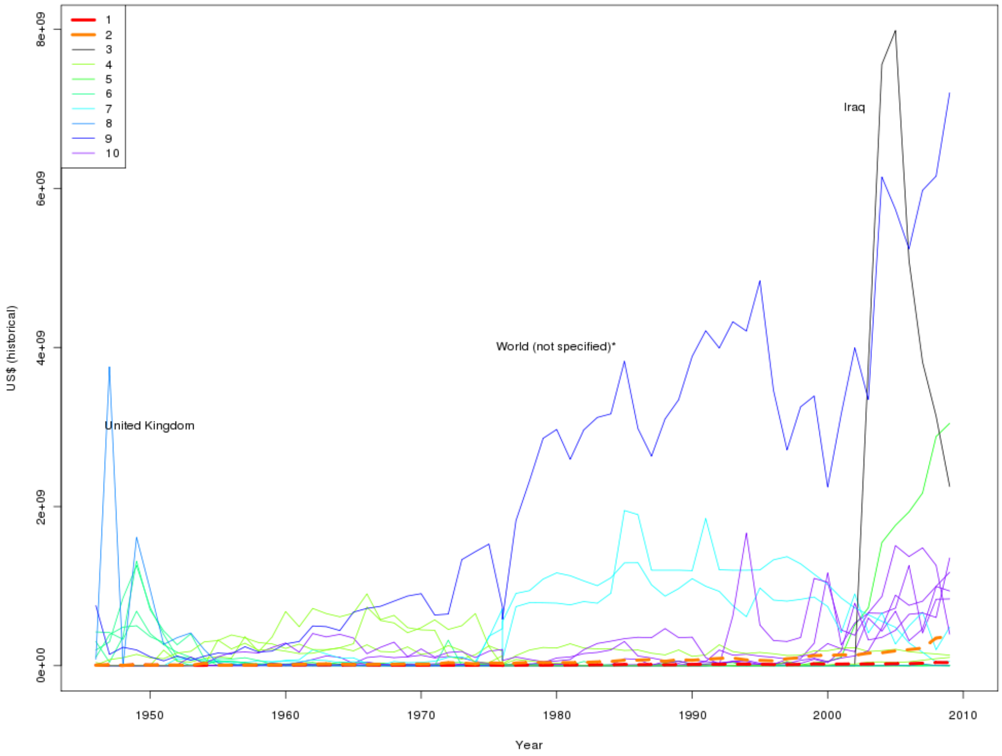
The process:

The first thing we'll do with the Overseas Loans data is aggregate by country. For clustering methods, it is easier to work with only two dimensions, in this case year and country, and at least for now, we'll not care about program information. As a homework assignment, all of the analysis below can be as easily done on program level data, now summing up over countries.

Even so, there are around 300 separate countries, each with 60 years of data. Plotting each individual timeseries, as you can see in figure x, gives you a plot that's way too crowded, both for analysis and presentation. But even here, we can see that most countries line the bottom of the chart - not a lot of aid - with only a few exceptions. Maybe we can find a few groups which summarize the overall structure of the plot without being too crowded?



A standard choice for clustering, or grouping observations is the **k-means algorithm**. Figure x shows the results for an arbitrarily chosen K=10 groups.[2] [3] The plot is nice in that I can see how different groups' timeseries differ from each other. For instance, both the UK and Iraq are unique in that they received large donations for a few years, right after WWII and after September, 2001, respectively. Also, the World (not specified) emerges as the single largest and consistent receiver of aid post 1970. Intrigued, we took a look through the data documentation, and found that the World (not specified) category includes assistance to “international financial institutions, international organizations and global programs.”[4] It turns out that most foreign aid and grants given by the US in the past 40 years are to multilateral organizations such as the International Monetary Fund and the World Bank. There is also a pretty clear increasing trend over time; the y axis is in billions of $.



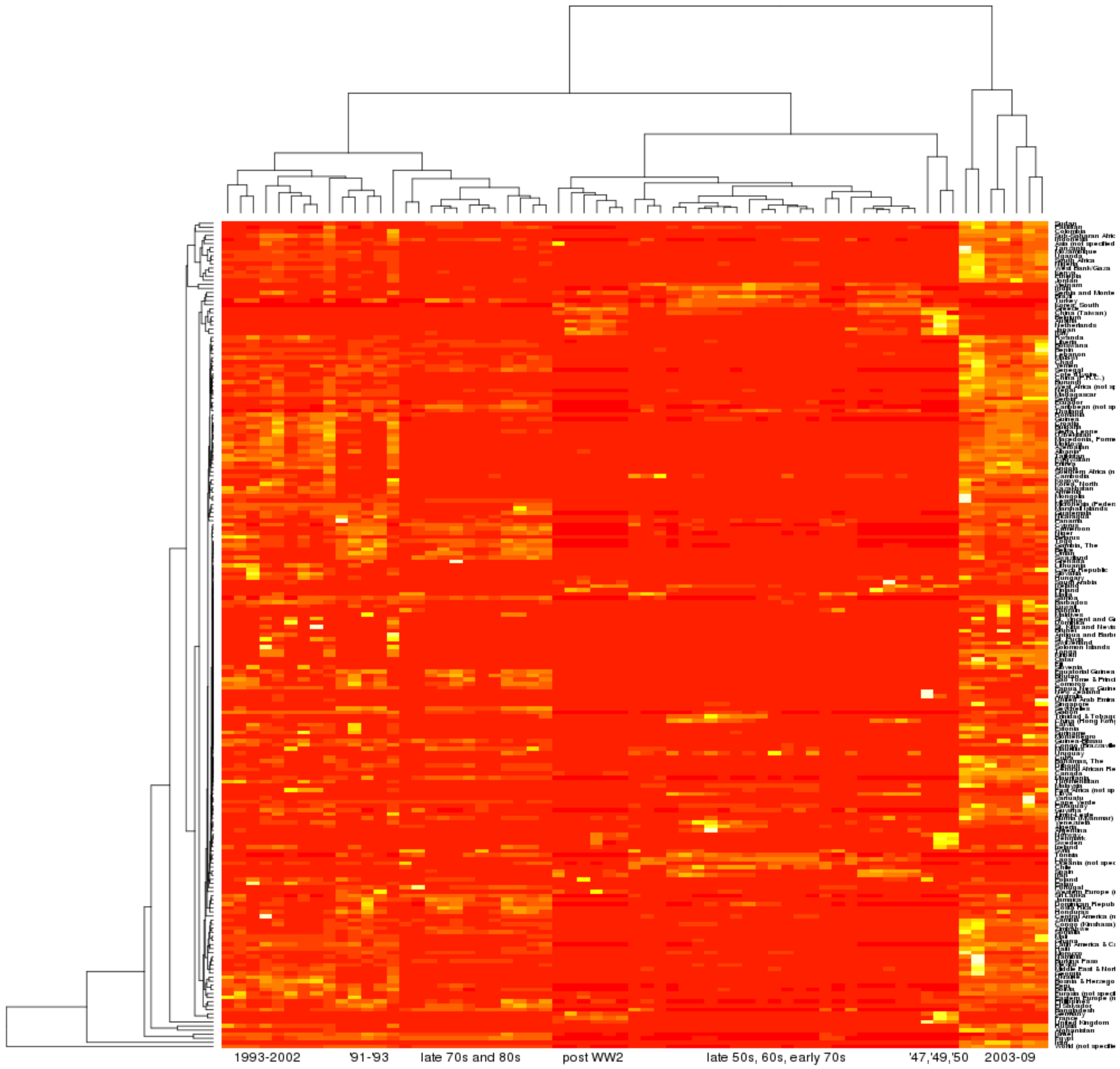
With any plot there are tradeoffs. Some less nice things about figure above are: the choice of 10 groups is arbitrary, I can't drill down on a group to see individual countries, and it's difficult for me to visually compare timeseries past some general comments. The next methods try to address some of these points.

[2] There are, by the way, fancier (read: statistically motivated) methods for choosing the number of clusters K. For example, see the “gap statistic” in ESL sec x.

[3] **Caution**: It can be altogether too easy to find some story that fits any clustering. When your data is random, a new sample can change the clustering enough to tell a different story.

[4] http://gbk.eads.usaidallnet.gov/about/country_notes.html

I started by grouping the countries into clusters, but why stop there? I bet there are some patterns hiding in the years too. One option I have in my statistician's bag of tricks is **bi-clustering**. The idea, as you might have guessed, is to cluster both rows and columns at the same time. A standard representation of this is to use a **heatmap**. Figure x below shows the heatmap for the US aid dataset, with **hierarchical clustering** on both the rows and columns.



Yes, those previous few sentences were loaded with terms. No, you don't absolutely need to understand all of them to use bi-clustering.[5] Here's a rundown of the specifics for interpreting a clustered heatmap.

The trees on the top and left are the output from running heirarchical clustering. Each observation (in this case a country on the rows, or a year on the columns) gets it's own leaf. Chopping a straight horizontal line at any height of the tree gives as many groups as branches the cut passed through. A bonus is you can tell how close different sized groups or leaves are to each other by how high up the tree they're merged. Figure x is an attempt at making this connection clear. The heatmap itself is the collection of colored rectangles in the center, and gives all the country donation-by-year data. Brighter colors = more money donated. With this sort of plot, we can see "important" clusters emerging organically. They're the ones that look like bands running down or across the heatmap. The over interpretation caution on the previous page applies almost doubly here.

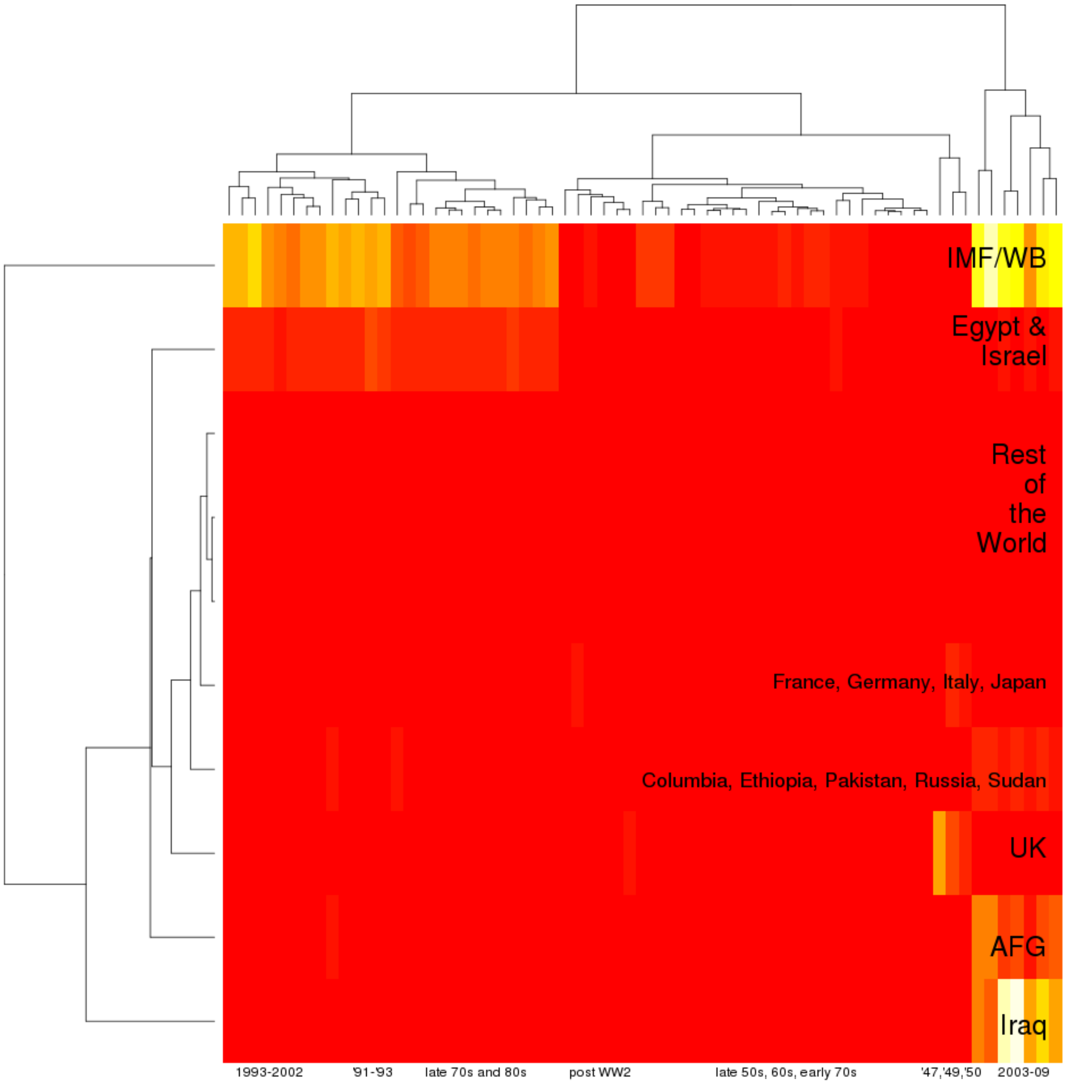
The year columns were mostly clustered by contiguous periods, so I instead wrote the time period the important clusters correspond to. Different periods of economic aid are more similar than others. Not a huge revelation. What is more interesting, however, is that distinct periods of US foreign aid emerge, and we see that more recent presidents were much more diverse in their approach to foreign aid. The years '91-'93 is when Bush Sr. held office, Clinton stepped down in 2001, and Bush Jr. held office through 2009. And it looks like the decade a president held office had a bigger impact on foreign aid than their individual political affiliations.

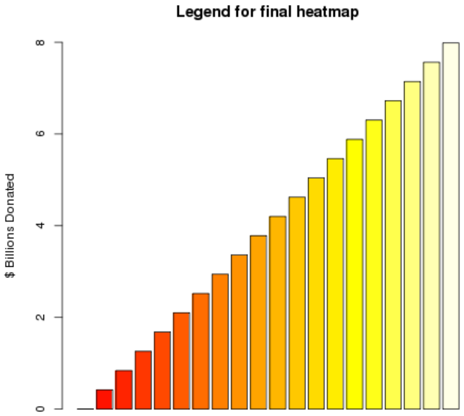
I'll mention now that the colors on the heatmap in figure x don't exactly correspond to the amount of money donated, but rather to the **scaled** amount donated. A lot of times statisticians can find more information from the pattern of deviations from the average, since then one or two countries that always get a lot of aid won't wash out the patterns among the rest. Also the heatmap gets much less interesting if we don't scale. Try it; you'll see what I like to call an "ocean of red."

Will all the countries getting a row, though addressing the drill-down problem, we get a bit crowded along the right. If the purpose was to single out a few specific countries,[6] it could be good to highlight those names in a specific color to show how they compare to the rest. I'll instead stay with a clustering approach and simplify the row-tree down to 10 groups, as in figure x. For consistency, I use the same groups as those found by k-means earlier, but I could have just as easily cut the heirarchical row-tree at a depth of 10 branches. The groups found by both methods will be roughly equal. I also chose not to scale the rows this time.

[5] Though it's my obligation to say, "you should."

[6] If you were writing a piece on foreign aid to Egypt and wanted to contrast with other countries, for example.


IMF/WB
Egypt & Israel
Rest of the World
France, Germany, Italy, Japan
Columbia, Ethiopia, Pakistan, Russia, Sudan
UK
AFG
Iraq
1993-2002
'91-'93
late 70s and 80s
post WW2
late 50s, 60s, early 70s
'47,'49,'50
2003-09


Legend for final heatmap
$ Billions Donated
0
2
4
6
8

My finale is an overview of US foreign aid in a single plot. Multilateral financial groups such as the IMF and World Bank are pretty clear fat cats of foreign aid. Iraq and Afghanistan, but mostly Iraq, received lots of aid post 9/11, and the UK was helped in the reconstruction years post WWII. For the rest you might need to turn off the lights in your office or tilt the monitor, but Egypt and Israel both received similar aid in the 80s and 90s. All these seem fairly in line with general history. The one surprising cluster is that of Columbia, Ethiopia, Pakistan, Russian and Sudan all receiving similar aid amounts and only during the Bush administration, and around $ 1B per year.

Further Readings:

There are certainly other methods out there. For the reader not afraid of some more math and technical jargon, one of my favorites is Principle Component Analysis (PCA), which relies on the ubiquitous Singular Value Decomposition (SVD) to find combinations of countries or years that spread out the data as much as possible. A description of these and others can be found in Chapter 14 of The Elements of Statistical Learning.

One particularly interesting new clustering method is called Prototype Clustering, developed by Jacob Bein at Stanford University. The output of the method is a tree like those from heirarchical clustering, but in addition every branch junction is labeled by the member most describing of the other leaves in the group. Some leaves get all the recognition. An example of what I mean is in figure x. An R library and description of the method can be found at http://cran.r-project.org/web/packages/protoclust/index.html

Appendix (R code):

```r
#r file for economic assistance data mining
#author: Leonid Pekelis

econaid.data = read.csv("us_economic_assistance.csv")

attach(econaid.data)

#####
#aggregate data by country, and plot time-series
#####

N = length(unique(country_name))

country.mat = matrix(0,nrow=N,ncol=65)

for(i in 1:N) {
  country.mat[i,] = colSums(econaid.data[which(country_name ==
unique(country_name)[i]),3:67],na.rm=TRUE)
}

years = c(1946:1976,1976:2009)

colnames(country.mat) = years
rownames(country.mat) = unique(country_name)

#create time-series plot
#use for loop to print 1 line per country
#text command to put labels on plot
png("allcountries.png",height=800,width=1000)
plot(c(1946,2010),c(0,max(country.mat)/(10^9)),type="n",xlab="Year",ylab="$ Billions
Donated")
for(i in 1:N) {
  lines(years,country.mat[i,]/(10^9),col=i)
}
text(1980,4,"World (not specified)")
text(2002,7,"Iraq")
text(1950,3,"United Kingdom")
dev.off()

#####
#k-means clustering
#####

#command to run the k-means algorithm
country.cluster = kmeans(country.mat,center=10)

#gives the names of the countries in each of the 10 clusters
cclust.names = list()
```

```r
for(i in 1:10) {
  cclust.names[[i]] = unique(country_name)[which(country.cluster$cluster == i)]
}

#print the names of countries in each cluster to the screen
for(i in 1:10) {
  print(paste("GROUP",i))
  print(cclust.names[[i]],max.level=0)
}

#plots the average donation time-series per cluster, to reduce clutter in plot
#this is the second time-series plot above
cols = rainbow(12)
cols[3] = "black"
png("cclust_condensed.png",height=800,width=1000)
plot(c(1946,2010),c(0,max(country.mat)),type="n",xlab="Year",ylab="US$ (historical)")
for(i in 1:N) {
  if(country.cluster$cluster[i] > 2) {
  lines(years,country.mat[i,],col=cols[country.cluster$cluster[i]])
}
}
lines(years,country.cluster$centers[1,],col=cols[1],lwd=4,lty=2)
lines(years,country.cluster$centers[2,],col=cols[2],lwd=4,lty=2)
legend("topleft",legend=1:10,col=cols[1:10],lwd=c(4,4,rep(1,8)))
text(1980,4*10^9,"World (not specified)*")
text(2002,7*10^9,"Iraq")
text(1950,3*10^9,"United Kingdom")
dev.off()

#####
#try some biclustering
#####

# the as.dendrogram command gives us the heirarchical clustering that will be above the heatmap
country.hc = as.dendrogram(hclust(dist(country.mat),method="complete"))
year.hc = as.dendrogram(hclust(dist(t(country.mat)),method="complete"))

# plots the 1st heapmap shown, with all countries on the vertical axis
png("countryheat_yrgp.png",width=1000,height=1000)
heatmap(country.mat,Rowv=country.hc,Colv=year.hc,cexCol=1,labCol=NA)
#mtext(c("1990-2002"),side=1,line=1,at=c(.1))
#mtext(seq(0,1,.1),side=1,line=1,at=seq(0,1,.1))
mtext(c("1993-2002","'91-'93","late 70s and 80s","post WW2","late 50s, 60s, early 70s","'47,'49,'50","2003-09"),side=1,line=1,at=c(0.2,.3,.4,.52,.7,.86,.93))
dev.off()

# plots the 2nd heatmap shown, with only the 10 clusters on the vertical axis
# again, condensing removes clutter, allowing general patterns to surface
png("countryheat_gp.png",width=1000,height=1000)
```

```r
heatmap(country.cluster$centers,Colv=year.hc,scale="none",labCol=NA,labRow=NA,col=heat.colors(20))
text(rep(.96,9),c(.80,.73,.70,.6,.57,.54,.51,.2,.1,.02),c("IMF/WB","Egypt &","Israel","Rest","of","the","World","UK","AFG","Iraq"),pos=2,cex=2)
text(rep(.96,2),c(.37,.27),c("France, Germany, Italy, Japan","Columbia, Ethiopia, Pakistan, Russia, Sudan"),pos=2,cex=1.5)
mtext(c("1993-2002","'91-'93","late 70s and 80s","post WW2","late 50s, 60s, early 70s","'47,'49,'50","2003-09"),side=1,line=1,at=c(0.2,.3,.4,.52,.7,.86,.93))
dev.off()

#lets do a legend plot for the final heatmap
png("chtgp_legend.png")
lvls = seq(0,max(country.mat)/(10^9),length.out=20)
barplot(lvls,main="Legend for final heatmap",ylab="$ Billions Donated",xlab="",col=heat.colors(20),ylim=c(0,8))
dev.off()
```