Statistical Computing and Data Visualization in R

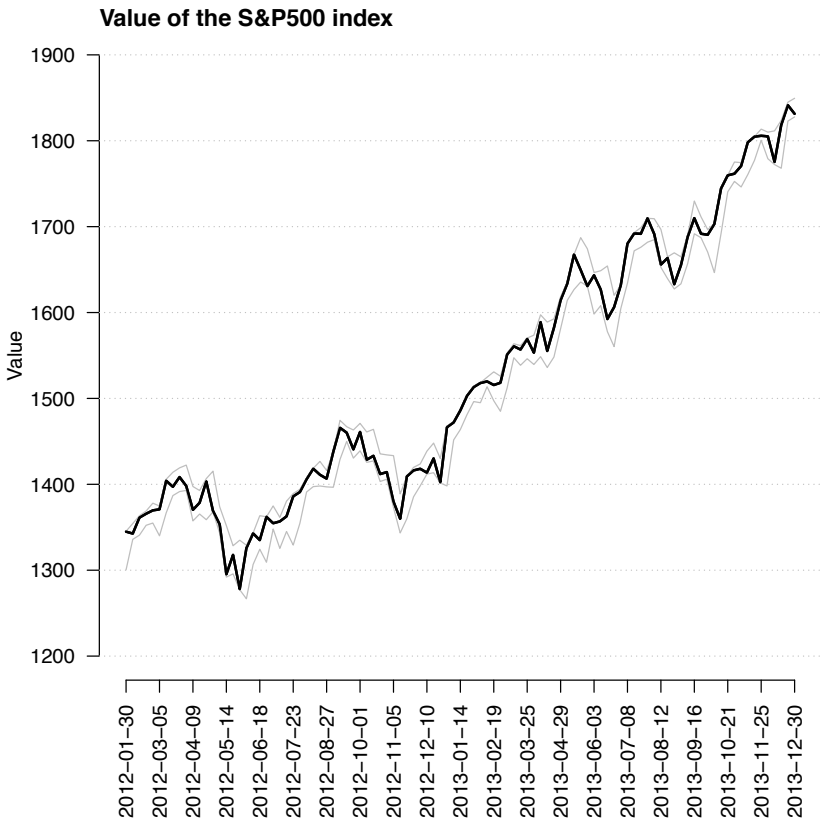
Lecture 8

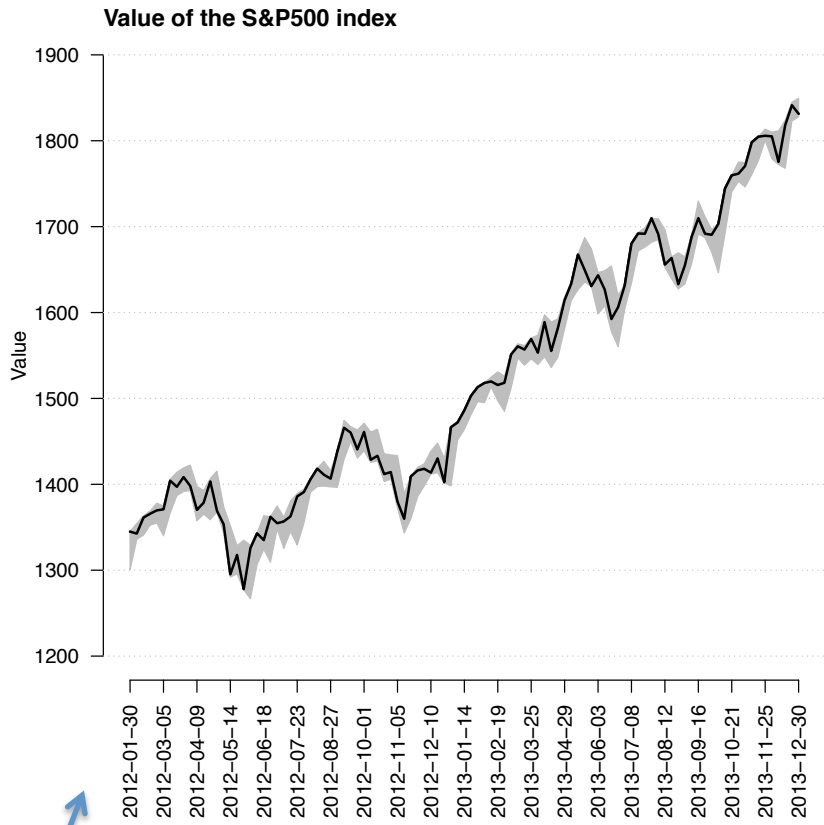
More Case Studies

Uncertainty bands

- A key concern in statistics is representing uncertainty in estimates/data.
- Depending on the context this needs to be treated as context in the graph, so it needs to clear but not overwhelming.
- We consider two situations:
  - Uncertainty in line plots/functional estimates.
  - Confidence intervals.

Uncertainty bands in line plots





Using a shaded area conveys a clearer message than the lines!
Using grey for the shading helps in keeping it secondary!

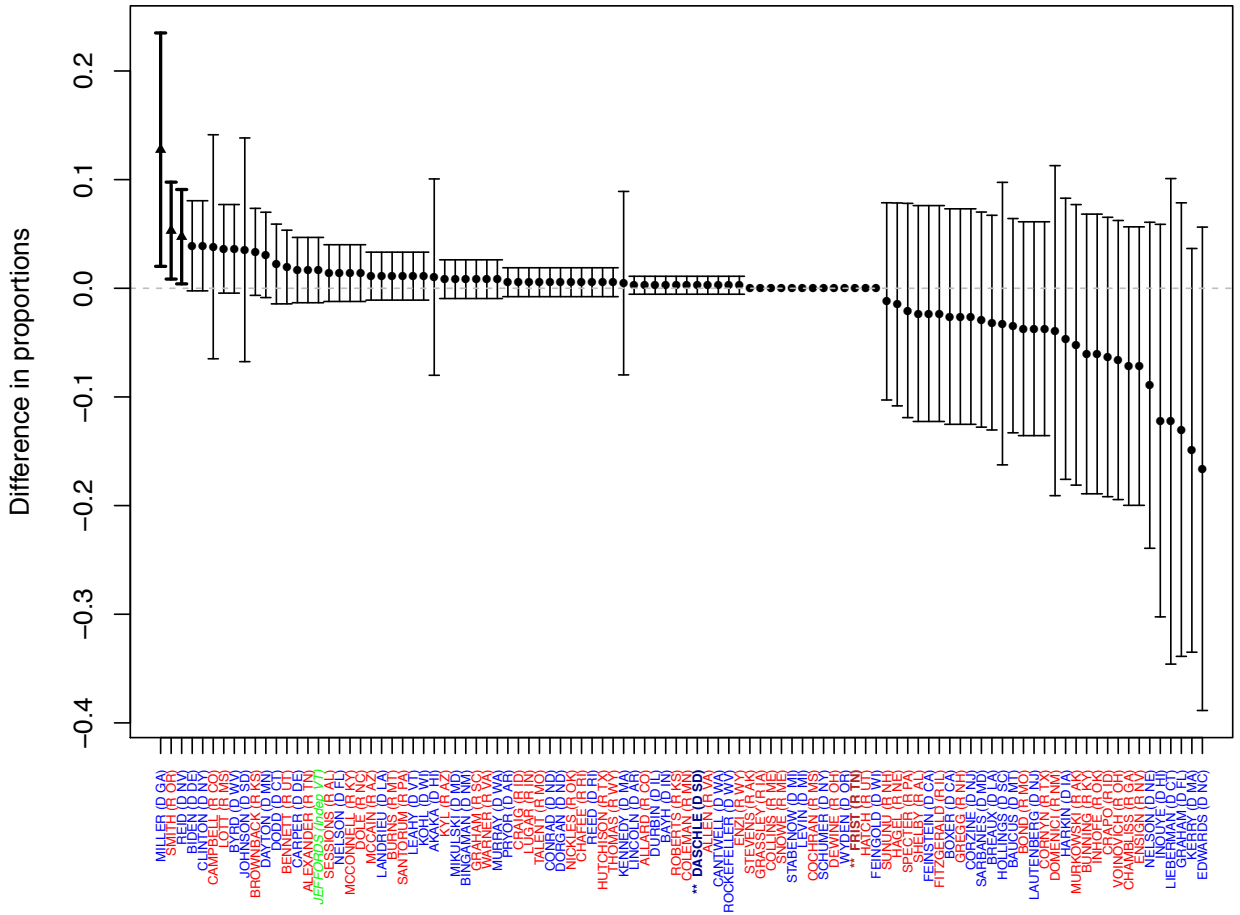
Uncertainty bars in line plots

```
dat = read.table(file="sp500historicalprices.csv", header=T, sep=",")
dat = dat[seq(1,101),]
n   = dim(dat)[1]
subs = seq(1,n, by=5)

quartz()
par(mar=c(7,4,1,1)+0.1)
plot(seq(1,n), rev(dat$Close), type="l", axes=F, ylab="Value", xlab="",
ylim=c(1200,1900), lwd=2)
axis(2, las=1, at=seq(1200,1900,by=100), labels=seq(1200,1900,by=100))
axis(1, at=subs, labels=rev(dat$Date[subs]),las=2)
mtext(side=1, line=6, at=3000, text="Source:  Yahoo Finance", las=1, cex=0.8)
title(main="Value of the S&P500 index", adj=0)
abline(h=seq(1100,1900,by=100), col="grey", lty=3, lwd=1)
polygon(c(seq(1,n),rev(seq(1,n))), y = c(rev(dat$Low), dat$High), col="grey",
border="grey")
lines(seq(1,n), rev(dat$Close), lwd=2)
```

Creates the shaded area, do it before the main line!

Plotting confidence intervals



Plotting confidence intervals

```
load(file="rawvotes108th.Rdata")
library(plotrix)
partycolors       = c("blue","red","green")
confidenceint.f = matrix(0, nrow=100, ncol=2)
pointestimate.f = rep(0, 100)
pvalue.f        = rep(0, 100)
for(i in 1:100){
  y1       = as.numeric(is.na(vote.data[i,libbills]))
  y2       = as.numeric(is.na(vote.data[i,conbills]))
  test.res = prop.test(x=c(sum(y1), sum(y2)), n=c(length(y1),length(y2)), conf.level = 1-0.05, correct=TRUE)
  confidenceint.f[i,] = test.res$conf.int
  pointestimate.f[i]  = test.res$estimate[1] - test.res$estimate[2]
  pvalue.f[i]         = test.res$p.value
}
significant.f = as.numeric(confidenceint.f[,1] > 0 | confidenceint.f[,2] < 0)+1
ord           = order(pointestimate.f,decreasing=T)
quartz(height=6,width=8)
par(mar=c(5.7,4,1,1)+0.1)
linewidthscheme = c(1,2)
pchscheme = c(16,17)
plotCI(x=seq(1,100), y=pointestimate.f[ord], li=confidenceint.f[ord,1], ui=confidenceint.f[ord,2], xlab="", ylab="Difference in
proportions", lwd=linewidthscheme[significant.f[ord]], pch=pchscheme[significant.f[ord]], pt.bg=par("bg"), cex=0.7, sfrac=0.004,
xlim=c(2,99), axes=F)
abline(h=0, lty=2, col="grey")
axis(2)
axis(1,at=seq(1,II,by=1),labels=FALSE)
posleaders            = c(which(sennames[ord]=="DASCHLE (D SD)"), which(sennames[ord]=="FRIST (R TN)"))

mtext(side=1, text=(sennames[ord])[partylines[ord]==1 & sennames[ord]!="DASCHLE (D SD)"], at=seq(1,II)[partylines[ord]==1 &
sennames[ord]!="DASCHLE (D SD)"], las=2, cex=.46, col=partycolors[1], line=1, font=1)
mtext(side=1, text=(sennames[ord])[partylines[ord]==2 & sennames[ord]!="FRIST (R TN)"],   at=seq(1,II)[partylines[ord]==2 &
sennames[ord]!="FRIST (R TN)"],   las=2, cex=.46, col=partycolors[2], line=1, font=1)
mtext(side=1, text=(sennames[ord])[partylines[ord]==3], at=seq(1,II)[partylines[ord]==3], las=2, cex=.46, col=partycolors[3],
line=1, font=3)
mtext(side=1, text=paste("** ", sennames[ord[posleaders[1]]]), at=posleaders[1], las=2, cex=.46, col="darkblue",  line = 1, font=2)
mtext(side=1, text=paste("** ", sennames[ord[posleaders[2]]]), at=posleaders[2], las=2, cex=.46, col="firebrick", line = 1, font=2)
box()
```

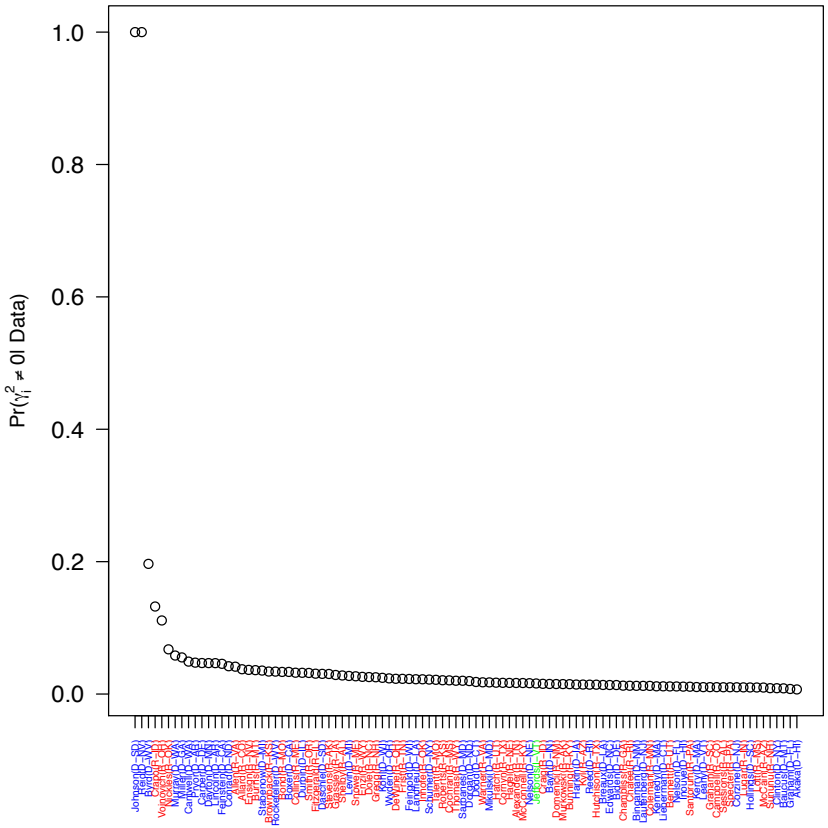
Used to plot the confidence intervals

Greek letters and math notation in R

- The function `expression()` can be used to create Greek symbols (LaTex-like syntax).
- Sometimes it might be easier to edit the image in Inkscape.

```
load(file="108thUSCongress.Rdata")
ord = order(gama.prob0)
partycolors = c("blue","red","green")
quartz()
par(mar=c(4.5,4.3,1,1)+0.1)
plot(seq(1,II), 1-gama.prob0[ord],
xlab="",ylab = expression(paste("Pr(",
gamma[i]^2!=0, "| Data)")), axes=F)
axis(1,at=seq(1,II,by=1),labels=FALSE)
mtext(side=1, text=sennames[ord],
at=seq(1,II), line =1, las=2, cex=.5,
col=partycolors[partylines[ord]])
axis(2,cex.axis=1.2, las=2)
box()
```

Adds names of Senators in color!

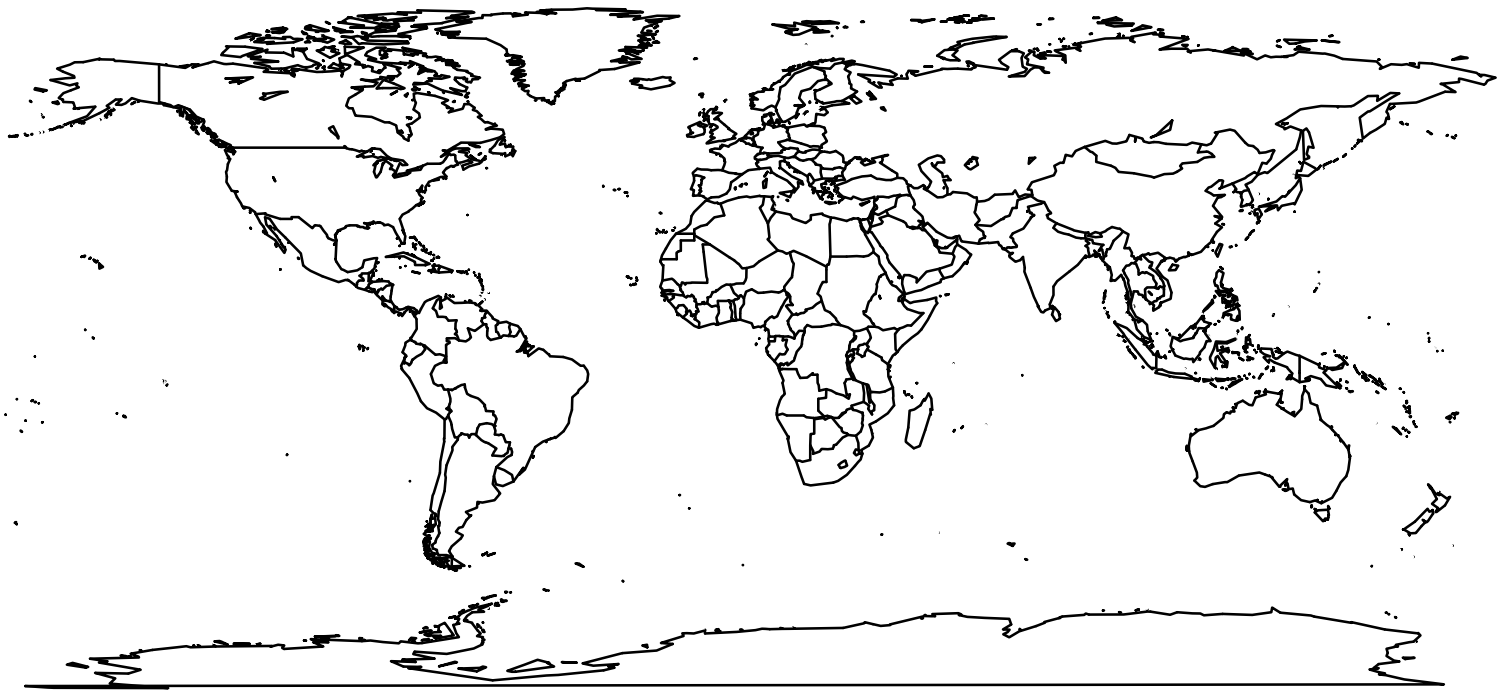


It would have been better to plot only the top 20 or 50 Senators.

Drawing Maps in R

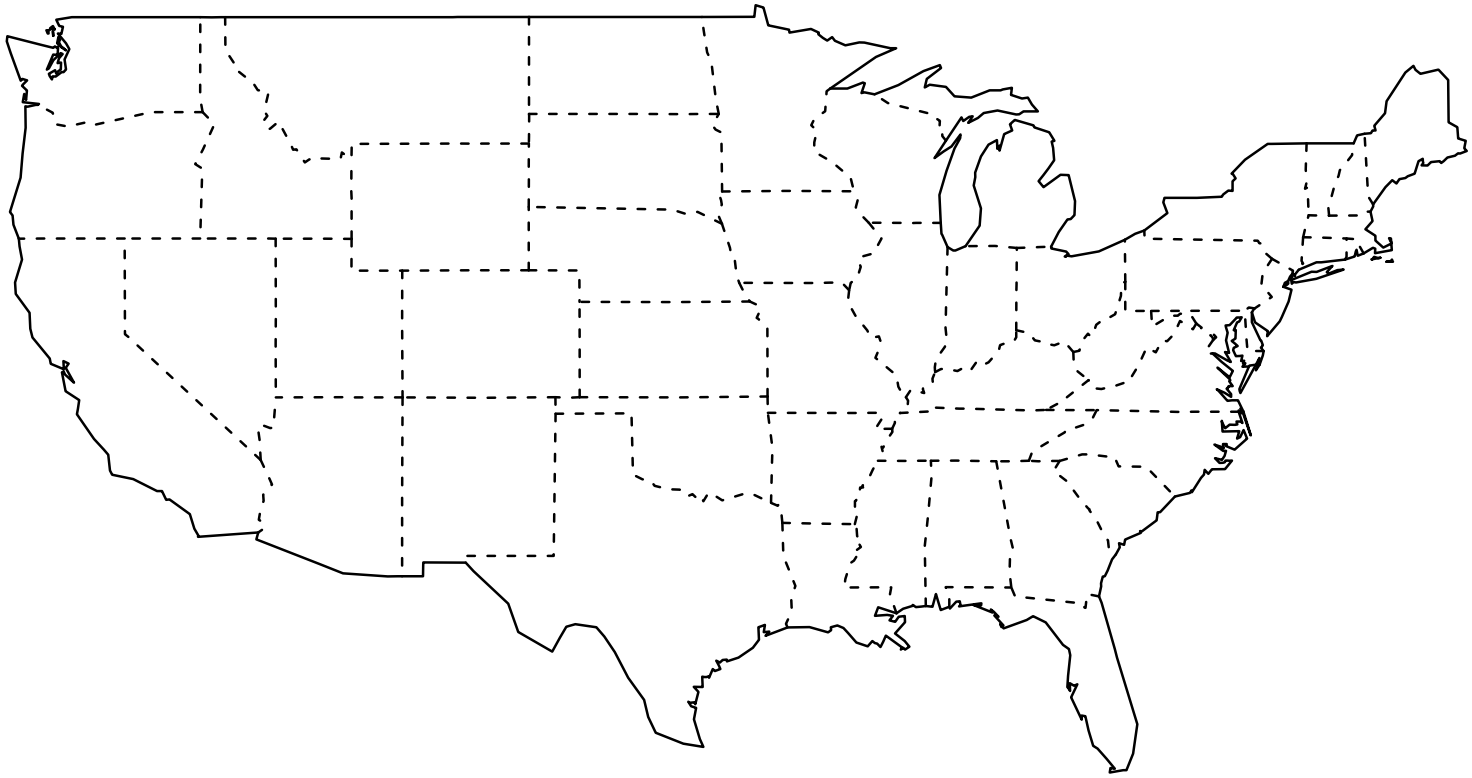
- To draw maps in R you need to install the library `maps`.
- The library contains a database of maps that includes US counties and states, as well as a world map that includes countries. It also includes detailed maps for some other countries (Italy, France, New Zealand).
- By default it uses a rectangular projection with the aspect ratio chosen so that longitude and latitude scales are equivalent at the center.
- The library also can locate "main" cities.

World map



```
library(maps)
map()
```

United States map



```
map("state", interior = FALSE)
map("state", boundary = FALSE, lty = 2, add = TRUE)
```

United States map (polyconic projection)

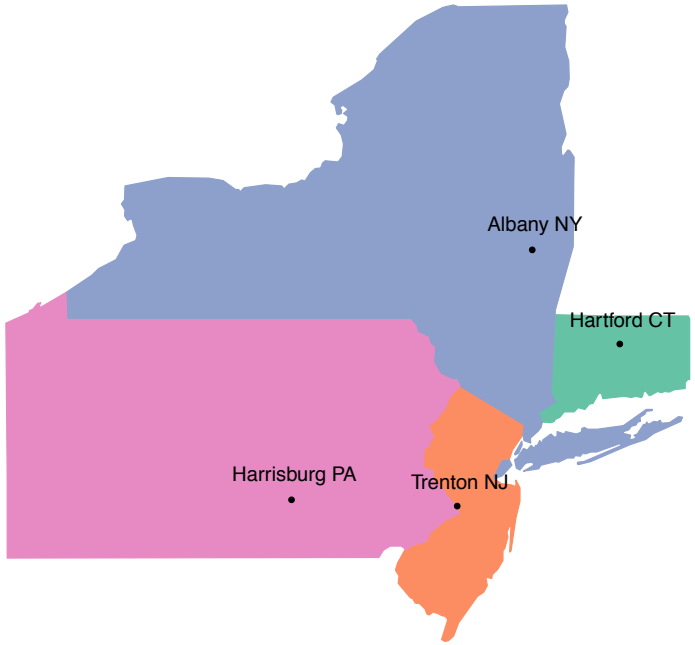
Often used for US maps in the mid 1900's. Vintage look!

```
map("state", projection = "polyconic")
```

California map with cities


```
map("county", region="california", interior=F, col="darkgrey")
map("county", region="california", boundary=F, lty=2, add=T, col="darkgrey")
map.cities(us.cities, country="CA", minpop=400000, pch=20, pos = 4, cex=1.1)
```

A map of four north-eastern states

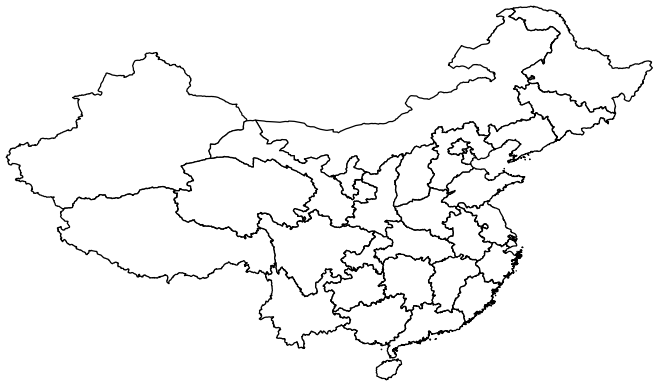



```
quartz()
colorscale = brewer.pal(4, "Set2")
colstates  =
c(colorscale[1],colorscale[2],rep(colorscale[
3],4),colorscale[4])
map("state", region = c("new york", "new
jersey", "penn", "connecticut"), fill=T,
col=colstates, border=colstates)
map.cities(us.cities, country=c("NY"),
pch=20, capital=2, cex=0.9)
map.cities(us.cities, country=c("NJ"),
pch=20, capital=2, cex=0.9)
map.cities(us.cities, country=c("PA"),
pch=20, capital=2, cex=0.9)
map.cities(us.cities, country=c("CT"),
pch=20, capital=2, cex=0.9)
```

Drawing Maps in R

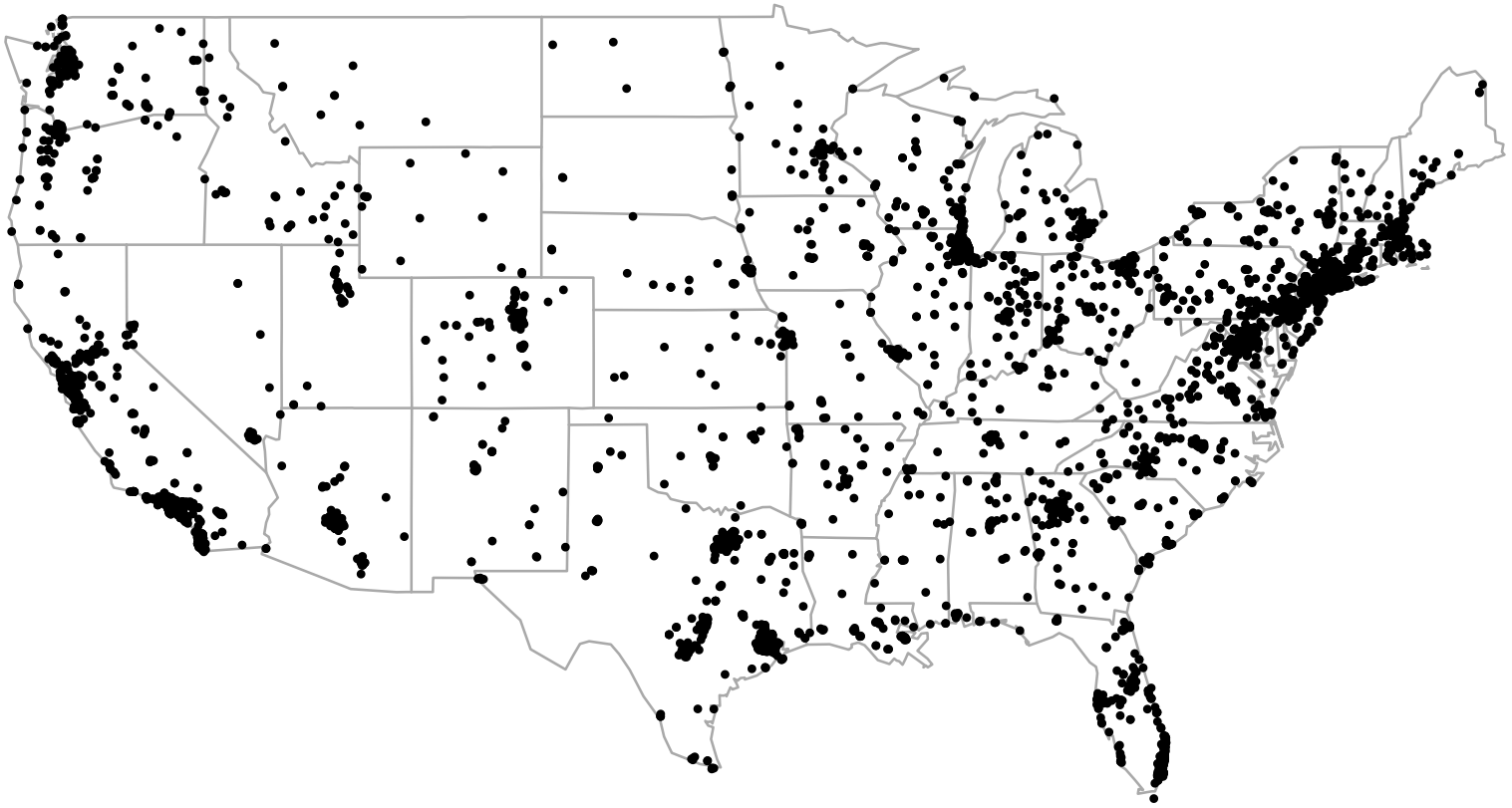
- You can load other maps using functions in the maptools library.

```
library(maptools)
xx = readShapePoly(fn="map_china”)
quartz()
par(mar=c(1,1,1,)+0.1)
plot(xx, col="white", axes=F,
las=1)
```



- Many .shp files can be download for free in the internet.

Point process maps: Yellow fever vaccination centers

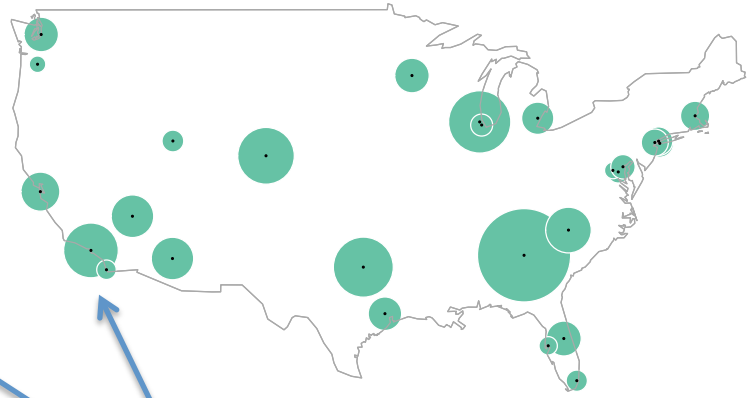


```
yellowfever = read.table(file="yellow_fever_vaccination2.csv", sep=",")
map("state", col="darkgrey")
points(yellowfever[,2], yellowfever[,1], pch=20, cex=0.5)
```

Proportional symbol maps: Major US airports and arriving passengers

```
airport =
read.table(file="mainairport.csv",
header=T, sep=",")
colorscale = brewer.pal(8, "Set2")

quartz()
map("usa", col="darkgrey")
symbols(-airport[,3], airport[,2],
circles=airport$arrivingpass/11000,
inches=F, add=T, fg="white",
bg=colorscale[1])
map("usa", col="darkgrey", add=T)
points(-airport[,3], airport[,2],
pch=20, cex=0.3)
```

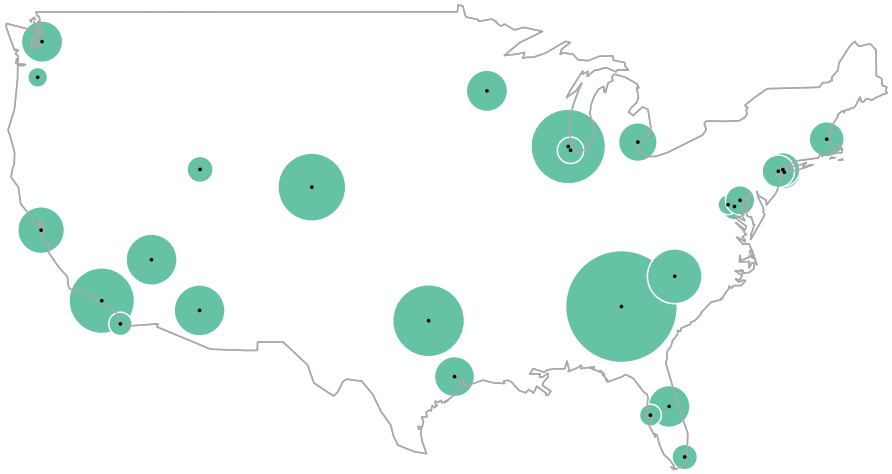


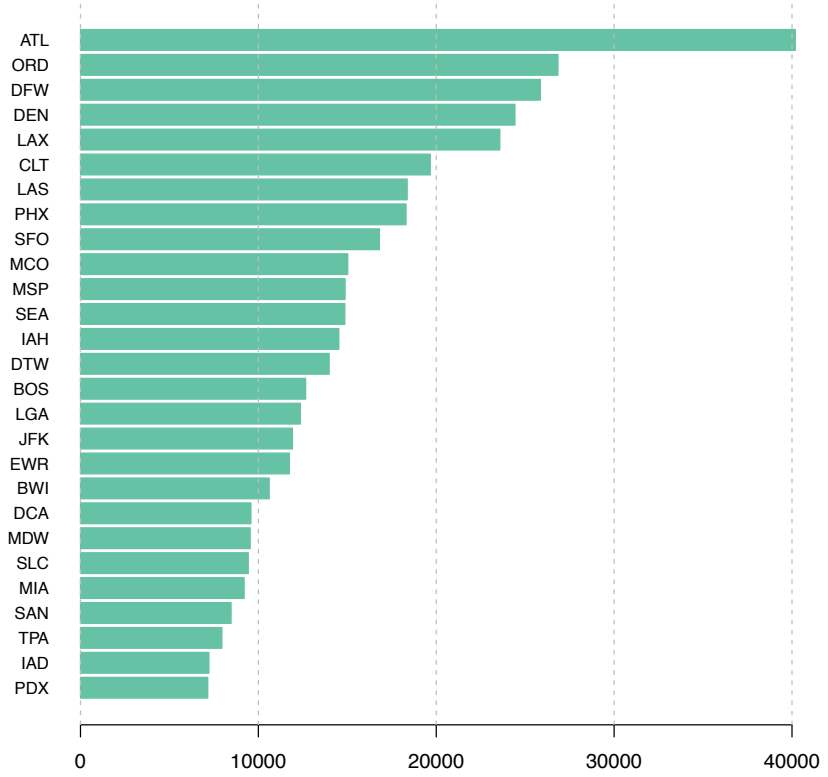
Note that a white boundary helps separate the bubbles of nearby airports!!

Location of major US airports and number of arriving passengers in 2013

- Additional information can be encoded:
  - Part to whole relationships (e.g., how many passengers are from each airline) could be depicted by converting the bubbles into pie charts.
  - Other quantitative or qualitative variables could be encoded using the color of the bubble.
- Note that this graph is good at showing you the big picture (in particular, how geography seems to affect passenger volumes), but does not allow for accurate comparisons.
- It would be useful to complement this map with a bar plot!

Location of major US airports and number of arriving passengers in 2013






```
quartz()
par(mar=c(4,4,1,1)+0.1)
ord = order(airport$arrivingpass, decreasing=F)
barplot(airport$arrivingpass[ord], names.arg=airport$name[ord],las=1,
col=colorscale[1], border=colorscale[1], cex.names=0.85, horiz=T)
abline(v=seq(0,40000,by=10000), col="grey", lty=2)
```

Choropleth maps

- A choropleth map is a thematic map in which areas are shaded or patterned in proportion to the measurement of the statistical variable being displayed on the map (e.g., population density or per-capita income).
- It is important to represent normalized values (rates, densities) in choropleth maps.
  - The eye naturally integrates over areas of the same color, giving undue prominence to larger polygons of moderate magnitude and minimizing the significance of smaller polygons with high magnitudes

Choropleth maps

- It is easy to construct choropleth maps in R
  - Just use the options `col` and `fill` of map to provide the colors.
- A few things to keep in mind:
  - It is important to represent normalized values (rates, densities) in choropleth maps.
  - You need to check the order of the regions!
  - Some regions involve more than one polygon (e.g., New York)!
  - Selecting an appropriate color scale is important!
    - Categorical scales (based on quartiles of the data).
    - Continuous scale (like we used in heatmaps).

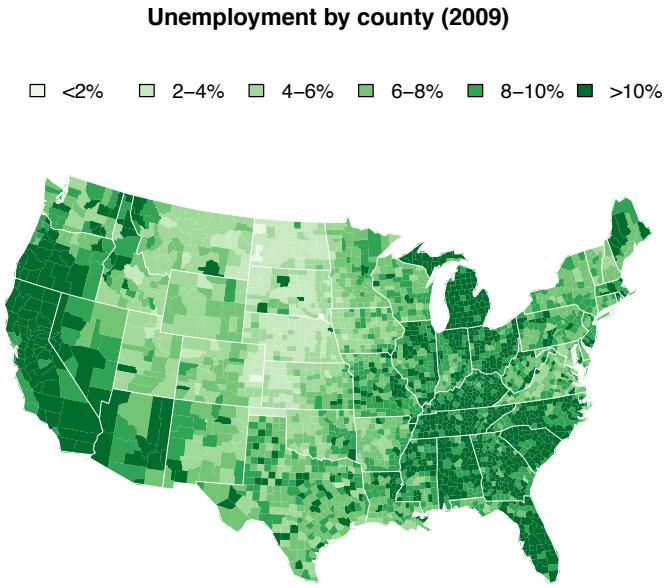
Example: Unemployment rates

```
data(unemp)
data(county.fips)

# Define color buckets
colors = brewer.pal(6, "Greens")
unemp$colorBuckets <- as.numeric(cut(unemp$unemp, c(0, 2, 4, 6, 8, 10, 100)))
leg.txt <- c("<2%", "2-4%", "4-6%", "6-8%", "8-10%", ">10%")

# Align data with map definitions by matching FIPS codes
# Works much better than trying to match the state, county names
# Which also include multiple polygons for some counties
colorsmatched <- unemp$colorBuckets
[match(county.fips$fips, unemp$fips)]

# Draw map
map("county", col = colors[colorsmatched], fill = TRUE, resolution = 0, lty = 0, projection = "polyconic")
map("state", col = "white", fill = FALSE, add = TRUE, lty = 1, lwd = 0.2, projection="polyconic")
title("Unemployment by county (2009)")
legend("topright", leg.txt, horiz = TRUE, fill = colors, bty="n")
```



Proportional symbol maps

- Bubble maps have two potential advantages over choropleth maps:
  - It allows us to represent more than one variable at a time (you can use area and color scheme).
  - Because areas are higher in the cognitive hierarchy than color, they allow for slightly more accurate comparisons.
- In spite of this, I still tend to prefer choropleth maps:
  - Areas do not allow for very accurate comparisons (you will need to supplement the map anyway).
  - Packing too much information in a single graph can be counterproductive …
  - Bubbles on nearby regions that overlap are hard to see and understand.

Cartograms

- Cartograms modify the size of different geographical regions so that they represent a variable other than area.
- Cartograms are highly decorative and amusing and can help emphasize specific arguments!
- There is not way to create cartograms directly in R, but you can create the pieces using the mapping functions and put them together using Inkscape.

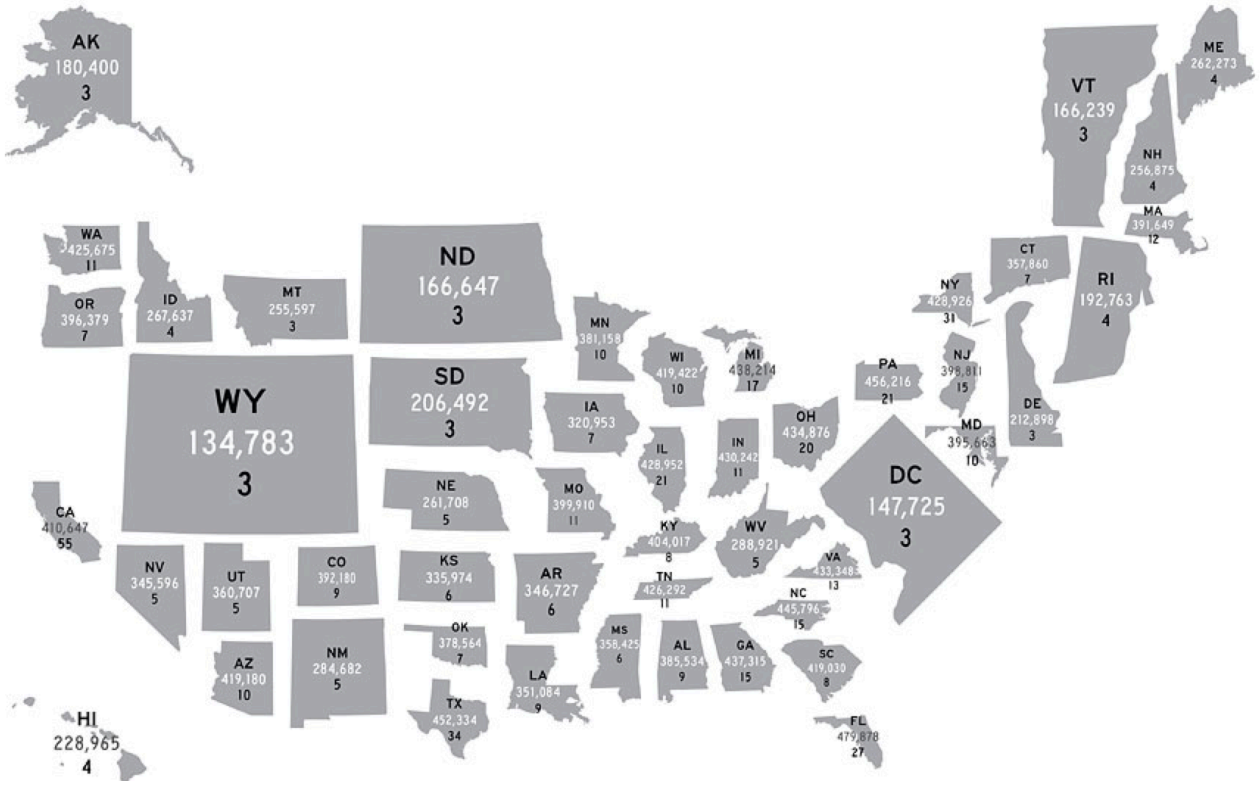
Cartograms – Varying-size

November 2, 2008

E-MAIL | FEEDBACK

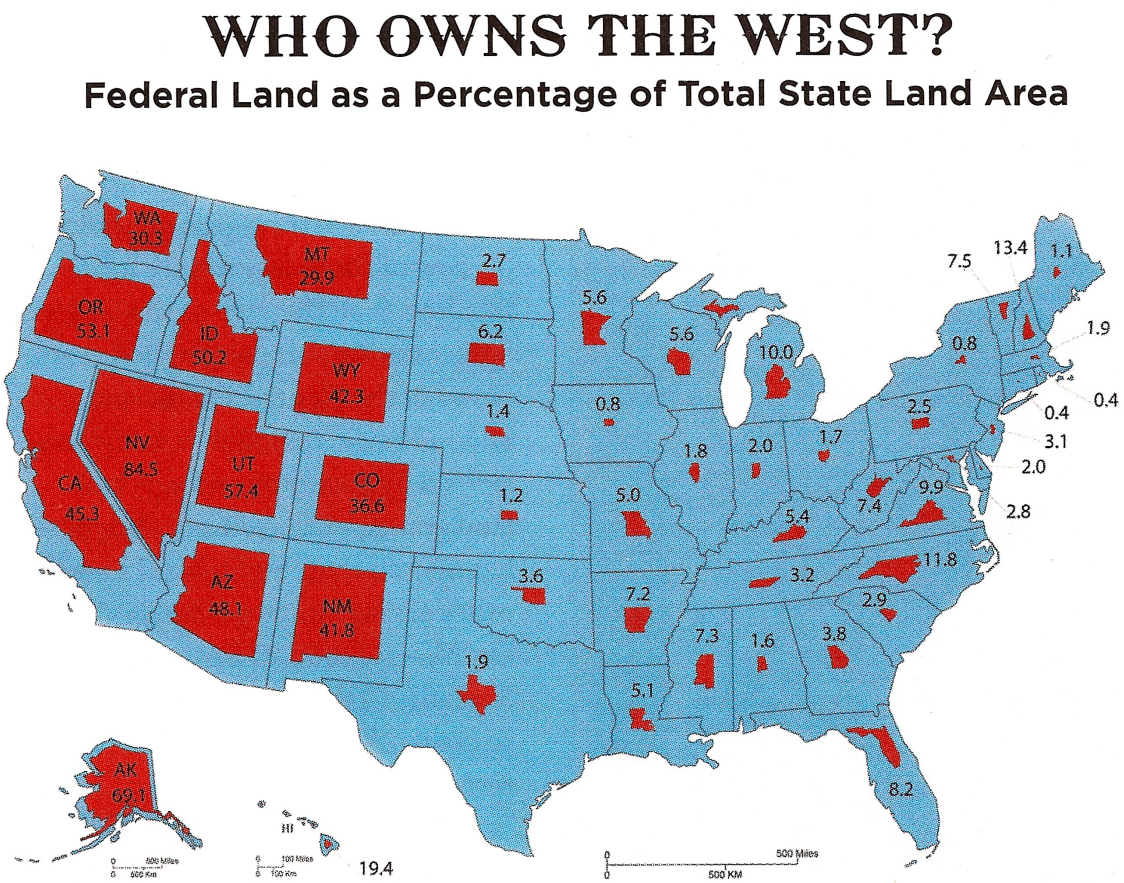
Op-Chart: How Much Is Your Vote Worth?

This map shows each state re-sized in proportion to the relative influence of the individual voters who live there. The numbers indicate the total delegates to the Electoral College from each state, and how many eligible voters a single delegate from each state represents.



http://www.nytimes.com/interactive/2008/11/02/opinion/20081102_OPCHART.html

Cartograms – Varying-size

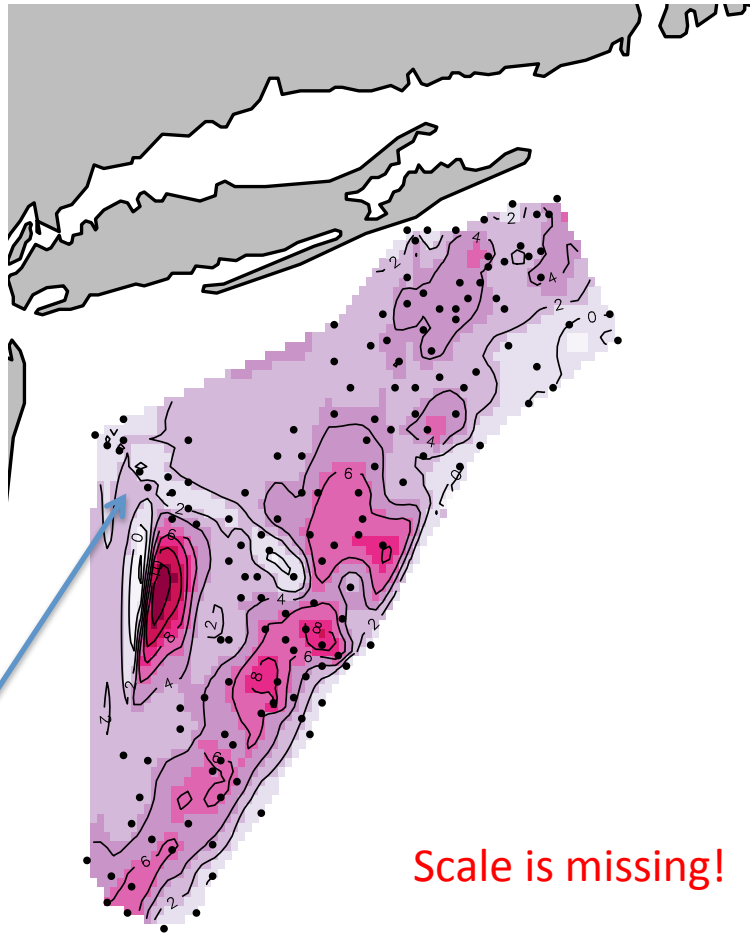


Isopleth maps

- Simple contours plots in R can be created using the `contour` function.
- Color can be added using the `image` function.
  - Remember to carefully choose a color scale that is either sequential or divergent.
- For these two functions to work, you need to interpolate the surface you are plotting so that it lives in a (regular) grid.
  - Use the library `akima`.

Isopleth maps

- Logarithm of the size of the scallop catch outside Long Island/ New Jersey
- Points show locations where fishermen attempted to catch scallops.



Very low level!
Pollution from the Hudson River?

Scale is missing!

Isopleth maps

Function for spatial interpolations

```
library(akima)
scallops              = read.table(file="scallops.txt", header=T)
scallops[,"lgcatch"] = log(scallops$tcatch + 1)
colorscale = brewer.pal(8, "PuRd")
quartz()
map("usa", xlim=c(-74,-71), ylim=c(38.2, 41.5), lwd=2, fill=T, col="grey",
boundary=FALSE, mar=c(1,1,1,1))
scallops.int = interp(x=scallops$long, y=scallops$lat, z=scallops$lgcatch,
xo=seq(min(scallops$long), max(scallops$long), length = 80),
yo=seq(min(scallops$lat), max(scallops$lat), length = 80), linear = F)
image(scallops.int, col=colorscale, axes=F, add=T)
contour(scallops.int, add=T, col="black")
points(scallops$long, scallops$lat, cex=0.75, pch=20)
dev.print(dev=pdf, file="scallops.pdf")
```

Colors

Contour plot

Networks

- Networks summarize data about the relationships between pairs of subjects.
  - How many times has a politician mentioned another one in his/her speeches.
  - Who is a friend of who in Facebook.
  - How many emails one person sent another.
- Network data usually comes in the form of a square matrix where entry (i,j) is a measure of the relationship between subjects i and j.

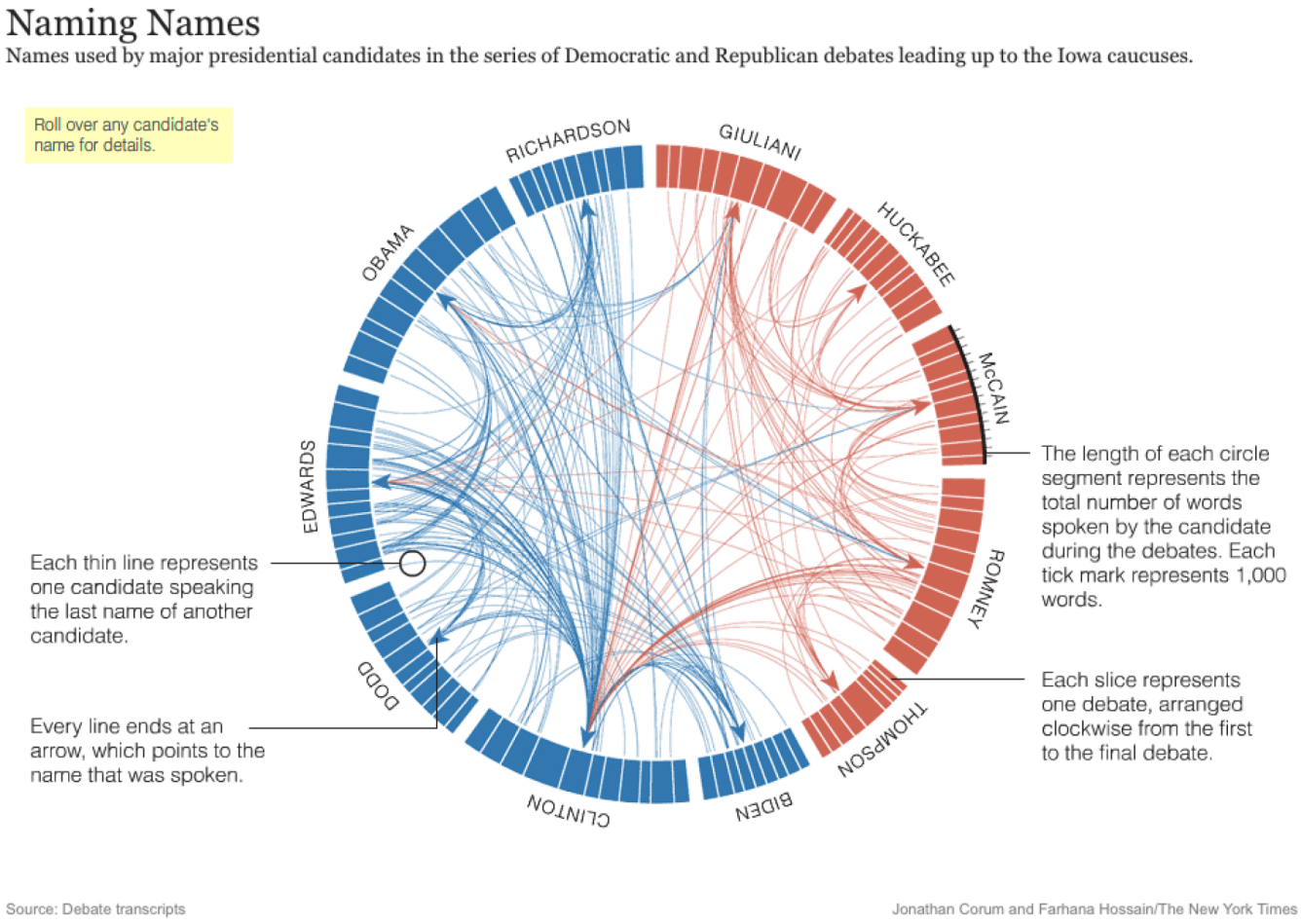
Networks

- One possible classification of networks is:
  - Binary: Only know if they are related or not (e.g., whether they are friends in Facebook or not). Matrix only has zeros and ones.
  - Weighted: We also know how strong the relationship is between individuals (e.g., we know the number of emails that each user sent to all others). Matrix might contain integers or real numbers.
- Usually diagonal elements of the matrix are zeros (no self relation).

Networks

- Another possible classification is:
  - Directed: a link from A to B has a different interpretation than a link from B to A (e.g., receiving an email is different from sending one).
  - Undirected: the relationship we are interested does not have a direction (or we elect to ignore it). For example, Facebook friendships.
- Facebook friendships could be made into a directed network by recording who initiated the link.

Visualizations for networks: circular network diagram



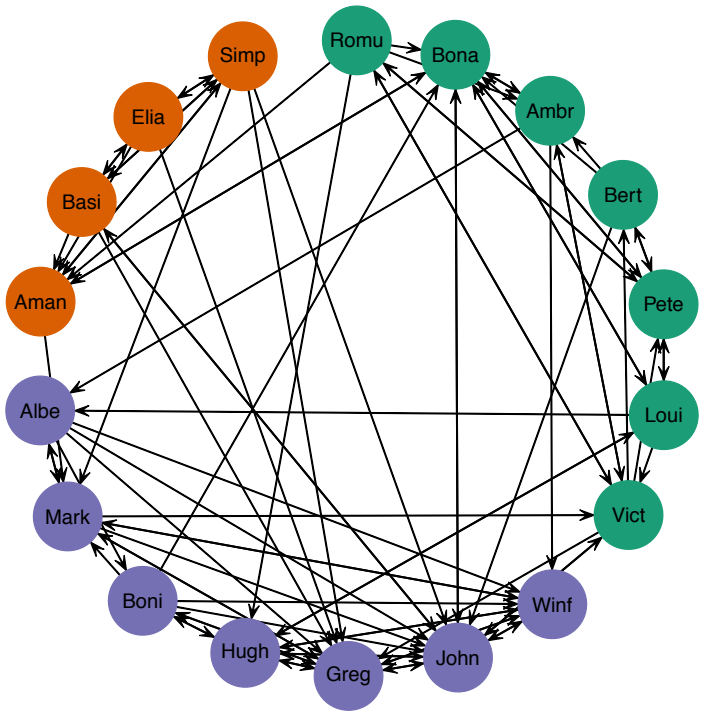
http://www.nytimes.com/interactive/2007/12/15/us/politics/DEBATE.html

Binary Networks

- Alternatively, nodes can be arranged on a circle, polar coordinates actually work well here!
- Nodes have been ordered according to Sampson's groups.

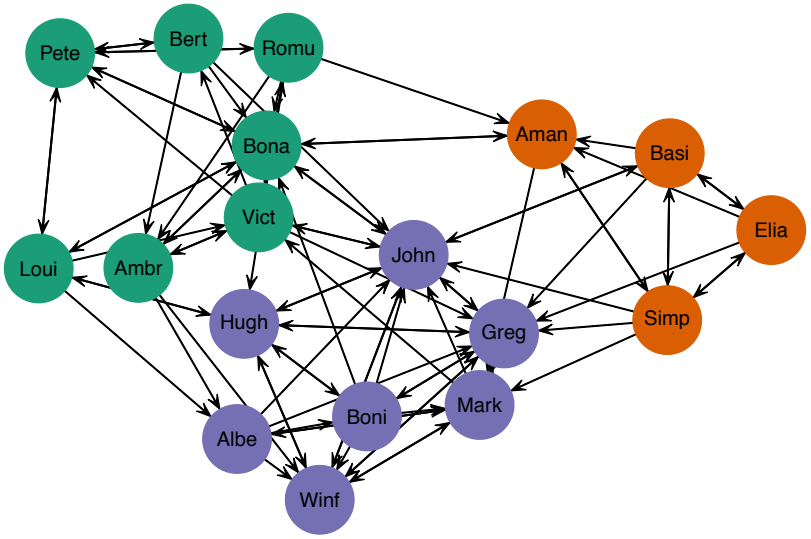
```
library(latentnet)
library(RColorBrewer)
data(sampson)
monknames =
substr(get.vertex.attribute(samplike,
"vertex.names"), 1, 4)
cluster =
as.numeric(as.factor(get.vertex.attribute(samp
like, "group")))
pal = brewer.pal(3,"Dark2")

quartz()
par(mar=c(0.5,0.5,0.5,0.5))
plot.network(samplike, mode="circle",
usearrows=T, label=monknames, label.pos=5,
vertex.cex=4.5, vertex.col=pal[cluster],
vertex.border=pal2[cluster])
```



Option "`circle`" arranges the nodes on a circle

Binary Networks

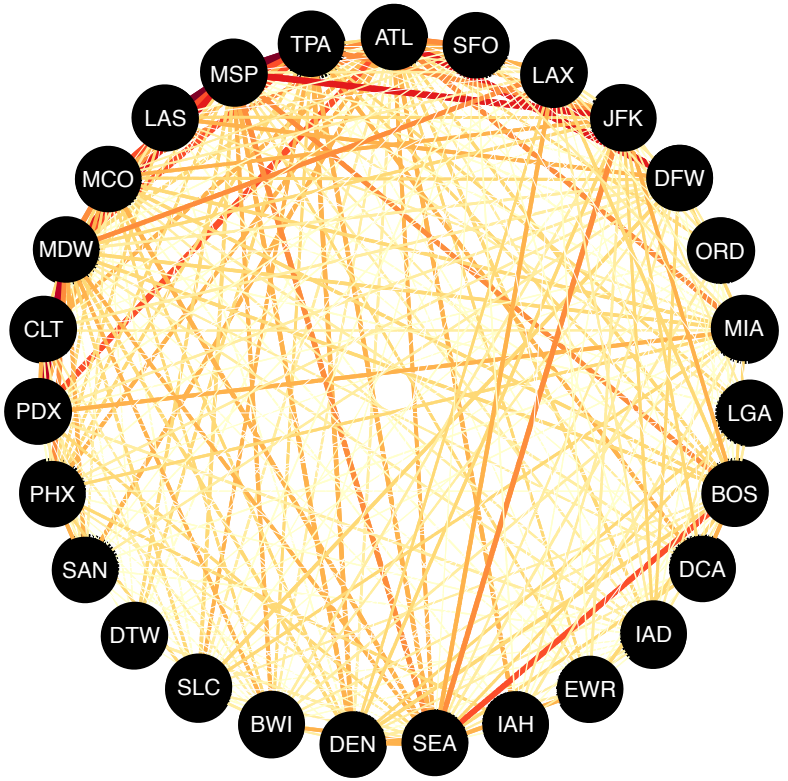


Use option `mode = "fruchtermanreingold"` with `plot.network` to generate this graph

- Circular plots often do not work for large networks.
- Force-directed algorithms assign forces among the set of edges and the set of nodes, based on their relative position and minimize energy/simulate motion:
  - Kamada-Kawai's: use only spring forces between all pairs of vertices, with ideal spring lengths equal to the vertices' graph-theoretic distance.
  - Fruchterman-Reingold's: uses both attractive forces on connected vertices, and repulsive forces on all vertices.

Weighted Networks

- Network of 27 most important US airports.
- Weight and color of the links corresponds to the number of seats available between both airports.
- Shortcomings:
  - No geographical information (maybe reorder the vertices?)
  - Because the network is dense, it is very hard to read!!



Weighted Networks

- A more sophisticated version of the visualization that overlays the network on a map.
- More informative about geography, but still hard to read!

Weighted Networks

```
graphroutes = function(subset.s.x, subset.s.y, usairport.v, usairport.e.n, colorpal){
  n.s.x             = length(subset.s.x)
  n.s.y             = length(subset.s.y)
  quartz()
  map("usa", fill=T, col="black", bg="grey")
  for(i in rev(subset.s.x)){
    for(j in rev(subset.s.y)){
      if(i!=j){
        segments(x0=-usairport.v[i,3], y0=usairport.v[i,2], x1=-usairport.v[j,3],
y1=usairport.v[j,2], lwd=5*usairport.e.n[i,j], col=colorpal[floor(10*usairport.e.n[i,j])
+1])
      }
    }
  }
}

usairport.e   = as.matrix(read.table(file="usairport_extended.csv",sep=",",header=F))
usairport.e.n = usairport.e/max(usairport.e)
usairport.e.n = (usairport.e.n + t(usairport.e.n))/2
usairport.v   = read.table(file="mainairport.csv", header=T, sep=",")
n             = dim(usairport.v)[1]
colorpal      = c("white", brewer.pal(9,"YlOrRd"))
graphroutes(seq(1,n), seq(1,n), usairport.v, usairport.e.n, colorpal)
```

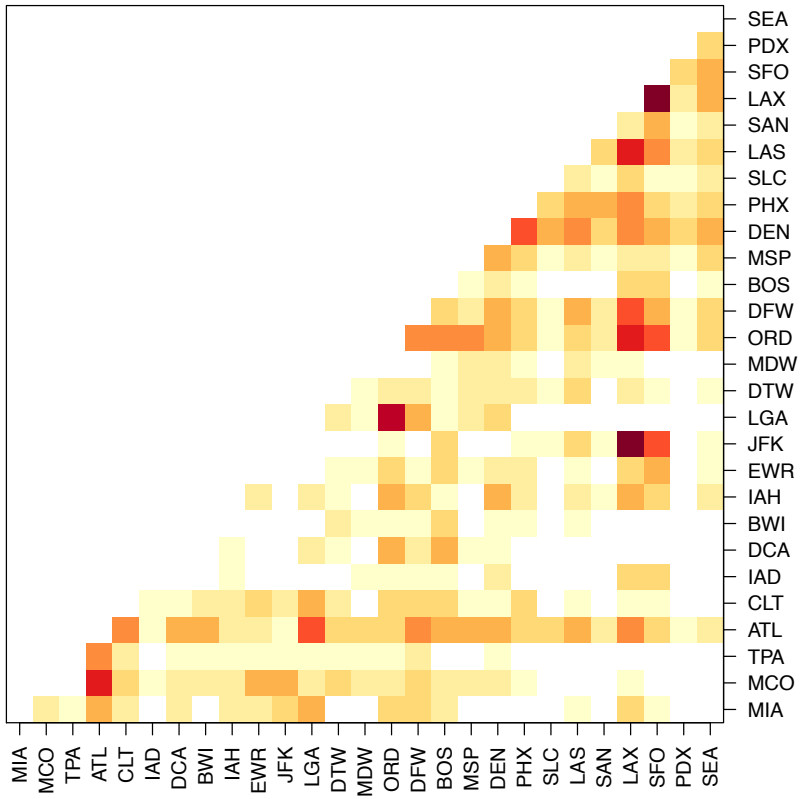
Weighted Networks

```
subset.s.1      = seq(1,n)[usairport.v[,3]> 96]
subset.s.2      = seq(1,n)[usairport.v[,3]<=96]
graphroutes(subset.s.1, subset.s.1, usairport.v, usairport.e.n, colorpal)
graphroutes(subset.s.2, subset.s.2, usairport.v, usairport.e.n, colorpal)
graphroutes(subset.s.1, subset.s.2, usairport.v, usairport.e.n, colorpal)
```

For dense networks like this, an interactive visualization works much better!

Weighted Networks

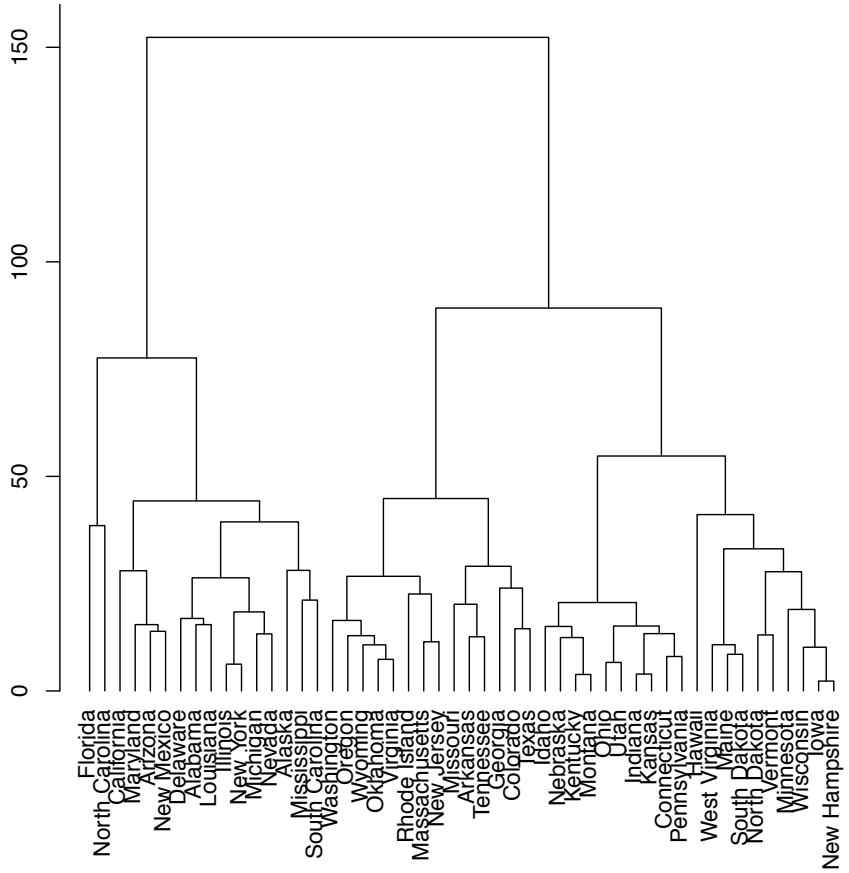
- The previous graphs are a bit less cluttered, but still very hard to read.
- For dense graphs, heatmaps can be the best (if least visually attractive) solution.




```
ind = order(as.matrix(dist(usairport.v[,2:3]))[7,])
quartz()
par(mar=c(4,1,1,4)+0.2)
colorpal      = c("white", brewer.pal(9,"YlOrRd"))
image(seq(1,n), seq(1,n), lower.tri(usairport.e.n[ind,ind])*usairport.e.n[ind,ind],
col=colorpal, axes=F, xlab="", ylab="")
axis(1, at=seq(1,n), label=usairport.v[ind,1], las=2)
axis(4, at=seq(1,n), label=usairport.v[ind,1], las=2)
box()
```

Binary trees

- Binary trees arise often:
  - Clustering (dendrograms).
  - Phylogenetics.
- Dendrograms can be easily generated using a plot method (see right hand side).
- However, this hard to manipulate, e.g., hard to add colors to the tips.
- Also linearly arranging the categories takes a lot of space.

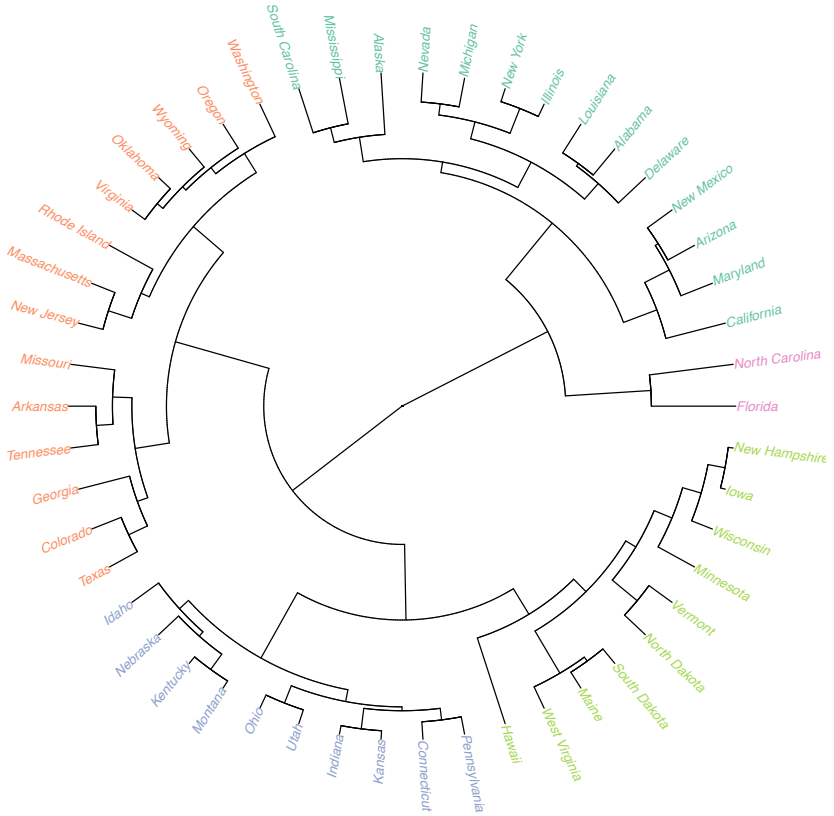



```
hc <- hclust(dist(USArrests), "ave”)
quartz()
par(mar=c(1,4,1,1)+0.2)
plot(hc, hang = -1, xlab="",sub="",main="")
```

Dendrograms

- Binary trees can be represented in circular form very effectively.

```
library(ape)
library(RColorBrewer)
hc <- hclust(dist(USArrests), "ave”)
ind = cutree(hc, k=5)
colscale = brewer.pal(max(ind), "Set2")
quartz()
plot(as.phylo(hc), type = "fan", tip.color
= colscale[ind], label.offset = .1, cex
=0.7, no.margin = T)
```



Word clouds

- Word clouds are very popular on the web:
  - They are good for visualizing the "big picture" (e.g., what are the most important words).
- However, because they use areas to represent proportions, word clouds do not allow for very accurate comparisons!
  - Solution: complement the word cloud with a bar chart associated with the most frequent words in the vocabulary that allows for more accurate comparisons.

Creating word clouds

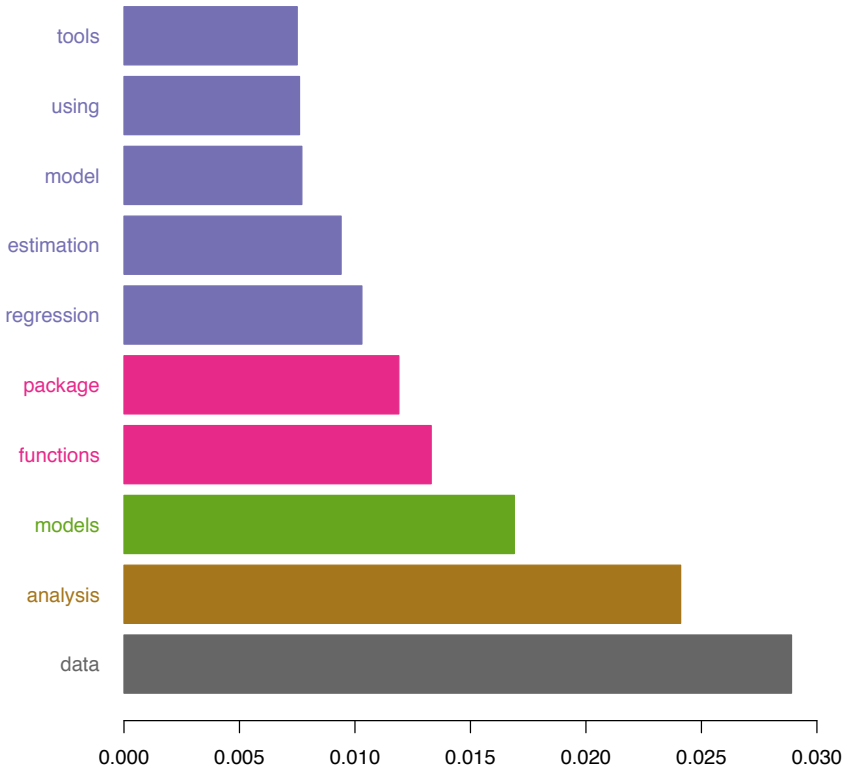
```
require(XML)
require(tm)
require(wordcloud)
require(RColorBrewer)
u = "http://cran.r-project.org/web/packages/
available_packages_by_date.html"
t = readHTMLTable(u)[[1]]
ap.corpus =
Corpus(DataframeSource(data.frame(as.character(t[,
3]))))
ap.corpus = tm_map(ap.corpus, removePunctuation)
ap.corpus = tm_map(ap.corpus, tolower)
ap.corpus = tm_map(ap.corpus, function(x)
removeWords(x, stopwords("english")))
ap.tdm = TermDocumentMatrix(ap.corpus)
ap.m = as.matrix(ap.tdm)
ap.v = sort(rowSums(ap.m),decreasing=TRUE)
ap.d = data.frame(word = names(ap.v),freq=ap.v)
table(ap.d$freq)
pal2 = brewer.pal(8,"Dark2")
quartz()
wordcloud(ap.d$word,ap.d$freq,
scale=c(8,.2),min.freq=3,
max.words=Inf, random.order=FALSE, rot.per=.15,
colors=pal2)
```

Creating word clouds

Barplots allow for more accurate comparisons:

Note that I used the same colors on the left and right to facilitate comparisons

Creating word clouds




```
kk  = 10
ord = order(ap.d$freq, decreasing=T)
pal2 = brewer.pal(8,"Dark2")
colscale = c(pal2[8], pal2[7],
pal2[5], rep(pal2[4],2),
rep(pal2[3],5))
quartz()
par(mar=c(3,5,1,1)+0.2)
barplot(height=round(ap.d$freq[ord[1
:kk]]/sum(ap.d$freq), 4),
names.arg=NULL, horiz=T, las=1,
xlim=c(0,0.03), col=colscale,
border=colscale)
mtext(side = 2, line = 1, text =
ap.d$word[ord[1:kk]], at =
seq(0.7,length=10,by=1.2), las=2,
col=colscale)
```