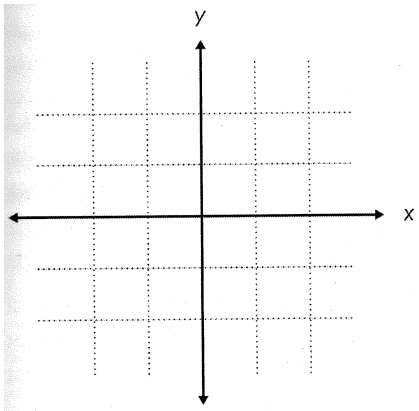
Statistical Computing and Data Visualization in R

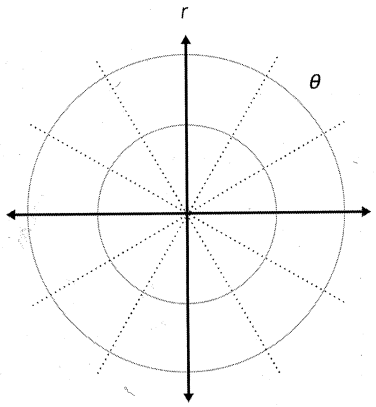
Lecture 6

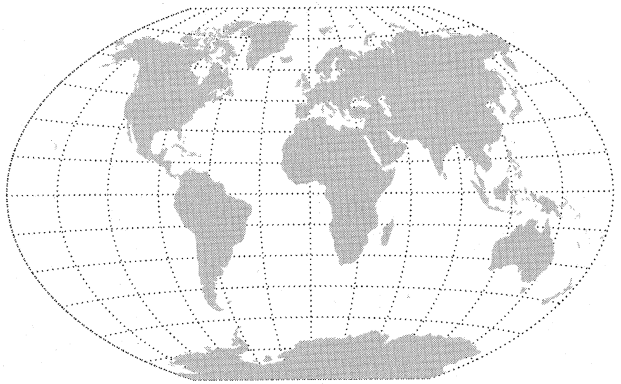
Principles of Data Visualization

Coordinate systems

- The structured space and rules that dictate where shapes, lengths and colors go.
  - Cartesian (Euclidean): Easier to understand (preattentive).
  - Polar: Less intuitive (how do you measure distances?) but more appropriate for certain types of data (e.g., periodic, networks).
  - Geographic: Helps us relate the numbers to specific location in the physical world.



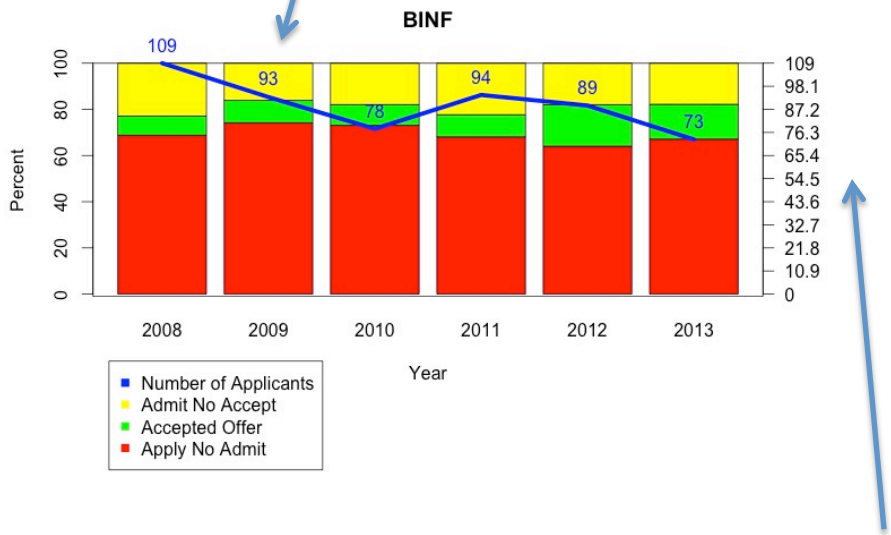




Cartesian coordinate systems

- Sometimes it is useful to include two different sets or coordinates in the same graph.
- Some authors absolutely discourage this practice!
  http://www.perceptualedge.com/articles/visual_business_intelligence/dual-scaled_axes.pdf
- If you do it (and I don't recommend it), make sure you label the two axes clearly to avoid confusions! Context is key here.

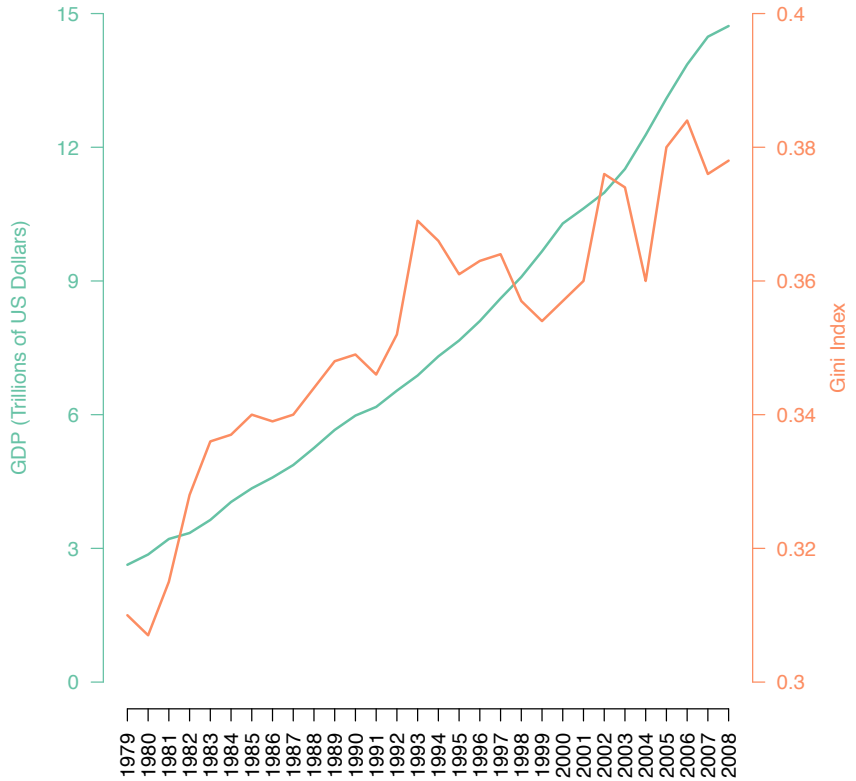
Not the best choice of colors …



- There is no label for this scale, what does it correspond to?
- Furthermore, is this scale associated with the bars or the lines?

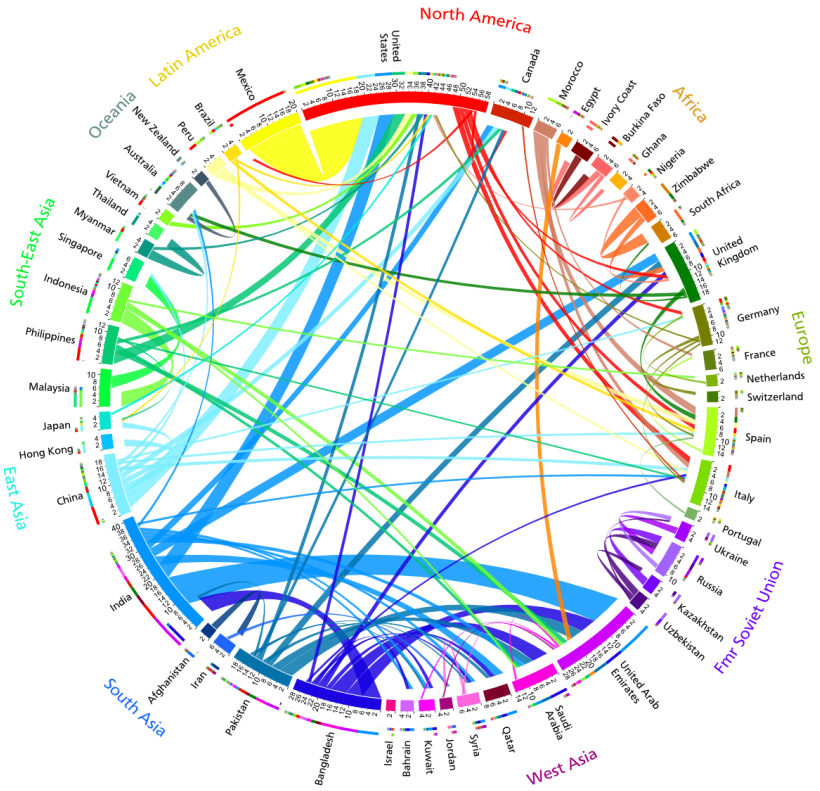
Cartesian coordinate systems

- When using multiple axes, using the same color for the visual cue and the axis often help the reader understand the visualization better.
- In this case I would have used green instead of black for the top axes (reserving black for the context material). Why?

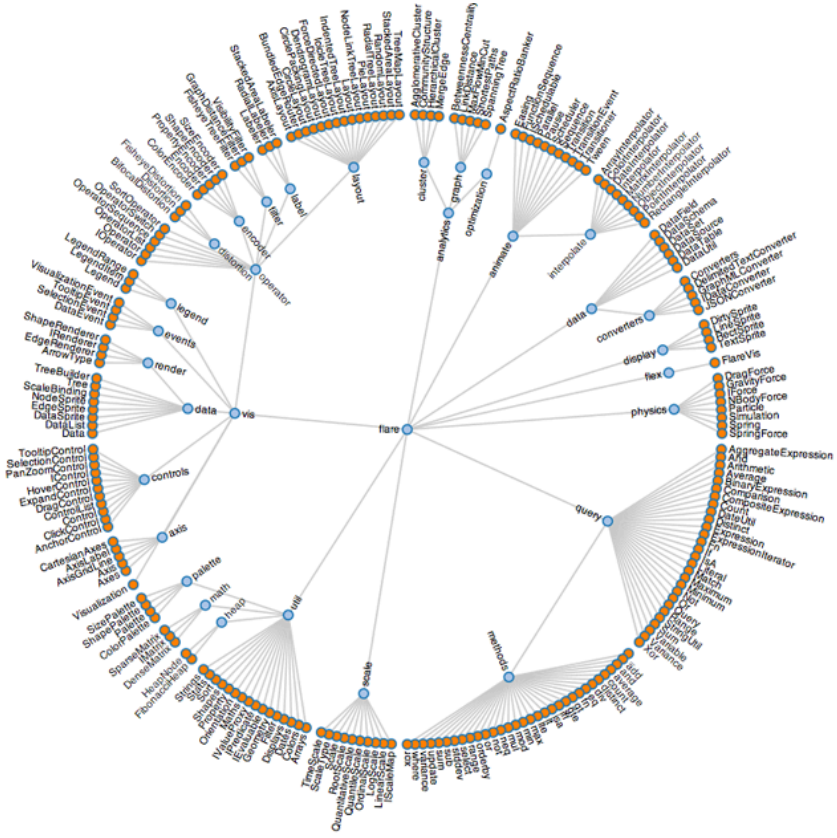


Polar coordinate systems

Networks

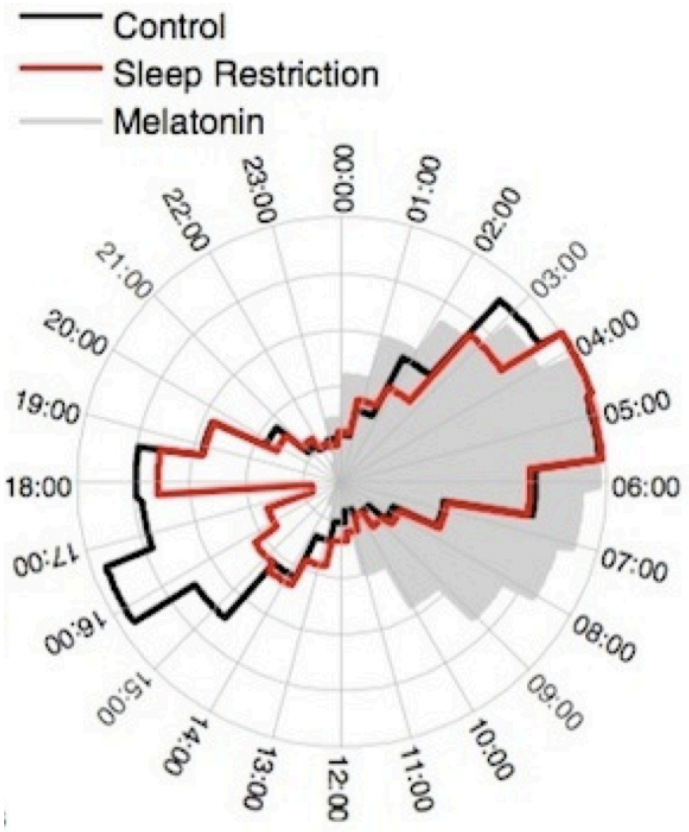


Dendrograms

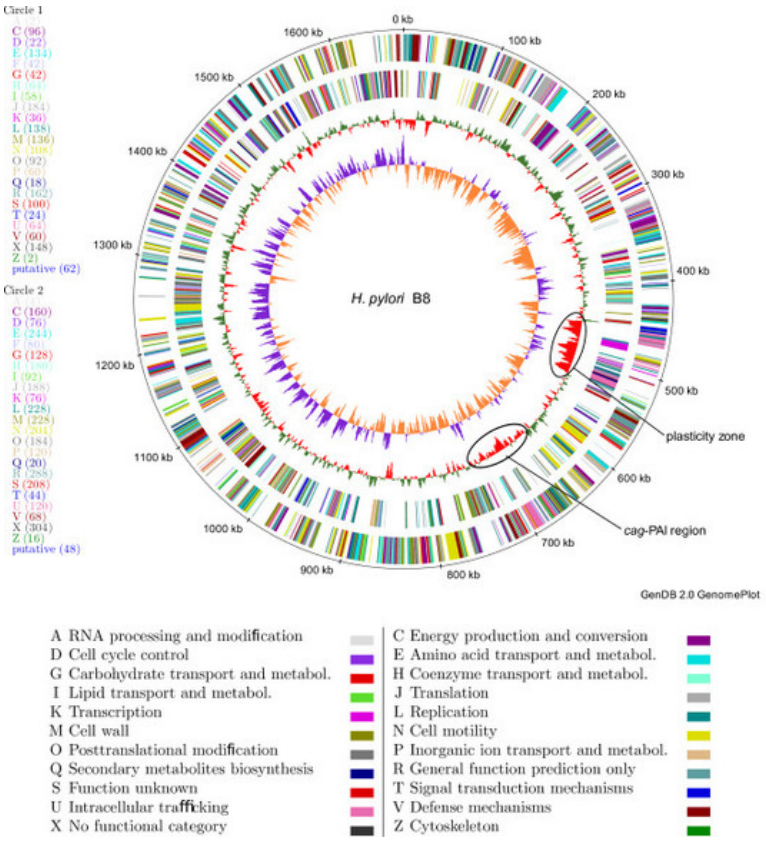


Polar coordinate systems

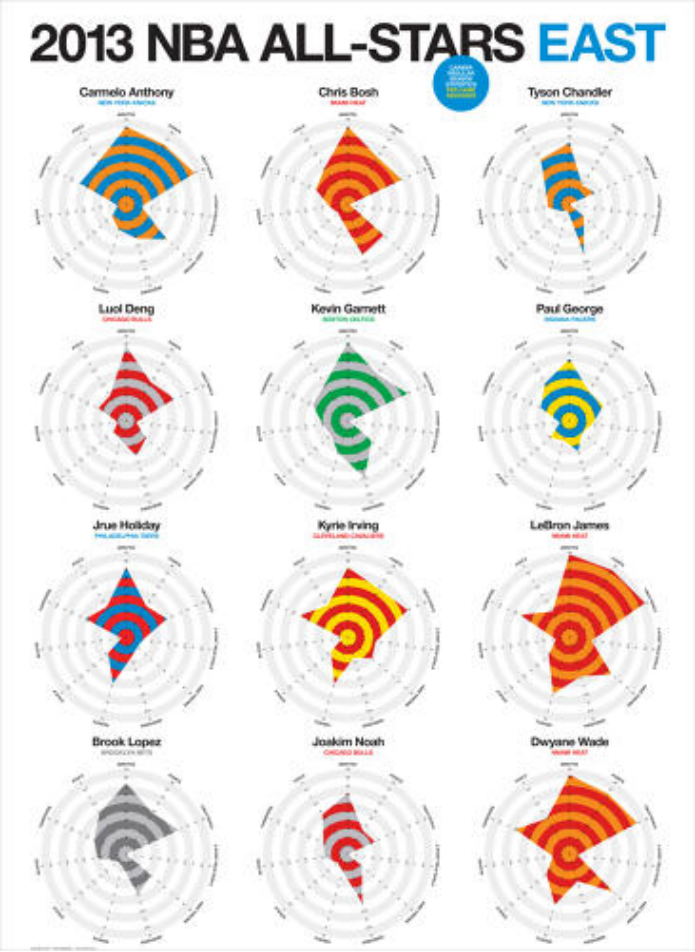
Periodic data



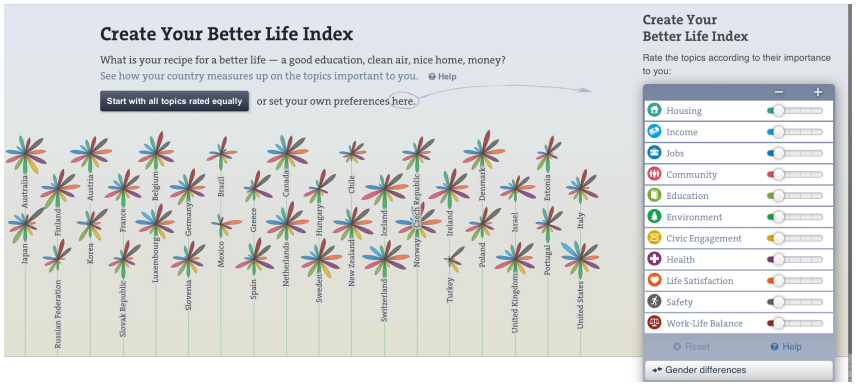
Not a good use …



Star plots



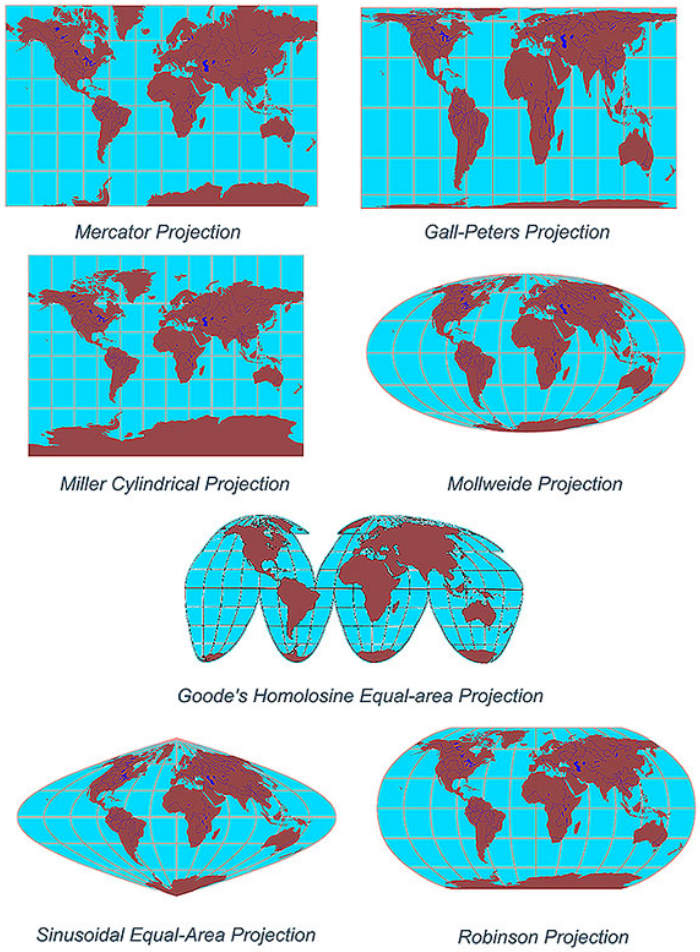
http://e.fastcompany.net/multisite_files/fastcompany/imagecache/slideshow_large/slideshow/2013/12/3023118-slide-1671897-inline-inline-zoom-all-stars-east.jpg



http://www.oecdbetterlifeindex.org

Geographic coordinate systems

- Because the Earth is spherical, to construct a map we need to project onto a plane (which leads to distortions).
- A large number of projections are available, depending on whether you are interested in preserving distances, angles, areas, etc.
- Mercator's and the equicilindridical projections are the most common.
- For small regions the specific projection used does not matter.
- For large areas it is very important, and the choice depends on your message!

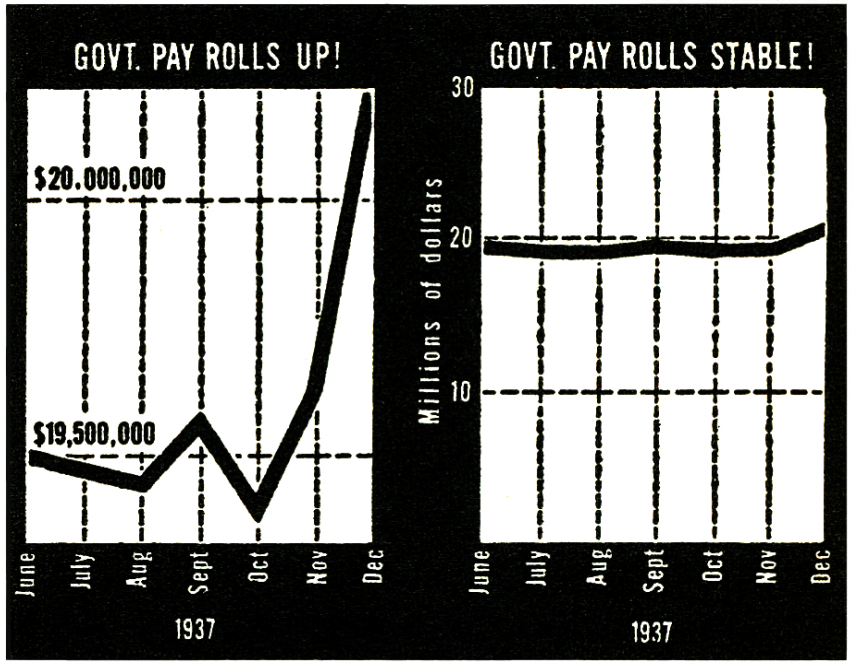


Scale

- While coordinate systems dictate the dimensions of the visualization, scale dictates where in those dimensions your data is placed.
  - Quantitative scales: linear, logarithmic.
  - Categorical: nominal, ordinal.
  - Temporal: time, date, week, year, etc.
- Selecting the right scale is every bit as important as choosing the right visual cues!

Linear scale

- The beginning and end values of your scale can dramatically alter the message transmitted by the graph.
- This is a classical example. The data is the same in both graphs, but the message is very different depending on whether the scale starts at zero or not.



From "How to Lie with Statistics" by Huff

Linear scale

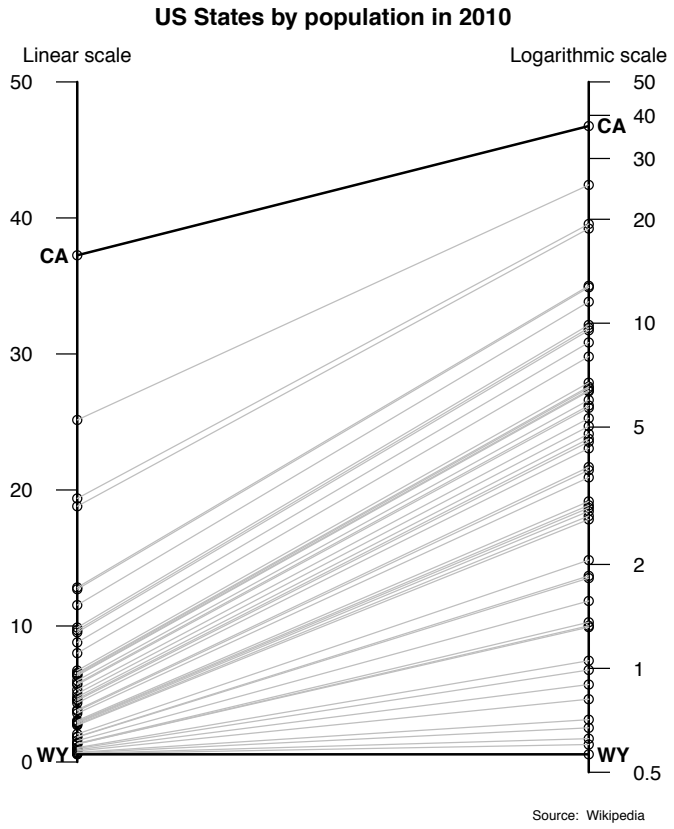
- A commonly cited rule of thumb is that, when the goal is to compare multiple positive values, linear scales should include zero.
  - This is done so that the rate of change (proportions) are maintained.
- More generally, you need to pick a scale that "makes sense" for the problem
  - What are typical values for the variable you are graphing (e.g., historical values, values from other similar entities).

Logarithmic scale

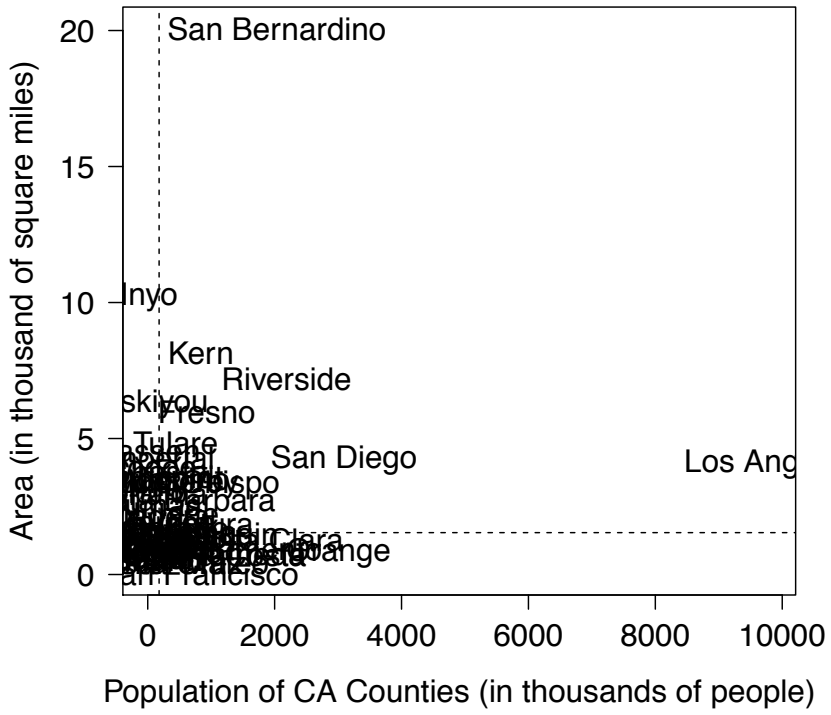
- Unlike the linear scale, in the logarithmic scale the values are not equidistant.
- When using a logarithmic scale, moving one unit on the scale is equivalent to **multiplying** the previous quantity by a constant (rather than **adding** a constant, as in linear scales).
- For this reason, logarithmic scales can be very useful when accurately representing relative (percent) rather than absolute changes is important.
- Some quantities are naturally measured in a logarithmic scale (noise, earthquake strengths).

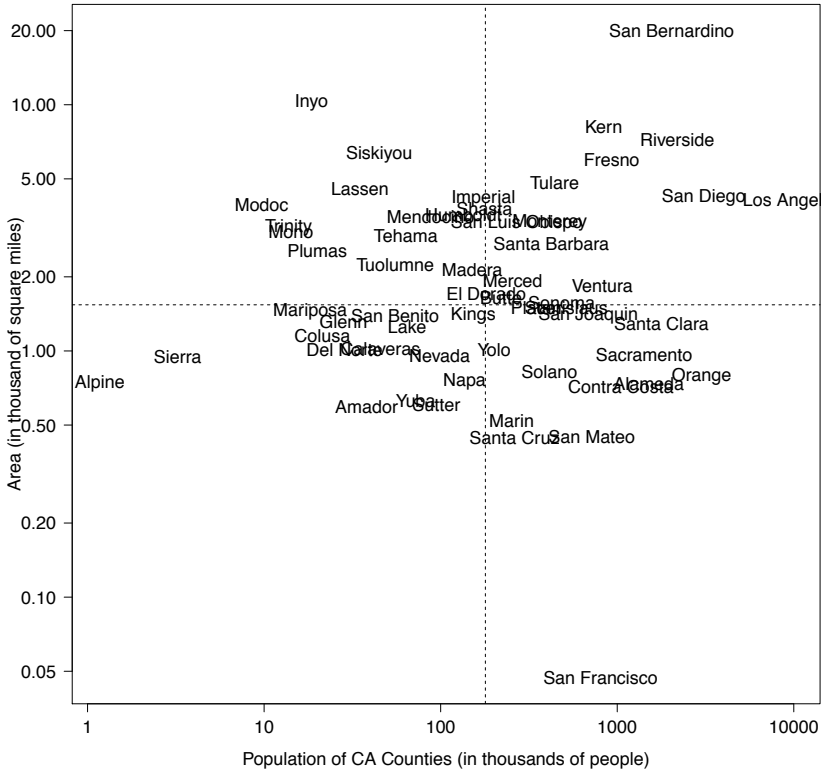
Logarithmic scale

- Logarithmic scales can also be useful when the distribution of the values is very skewed (e.g., income, population).
- Note the scale is "stretched" for small values and shrunk for large ones!

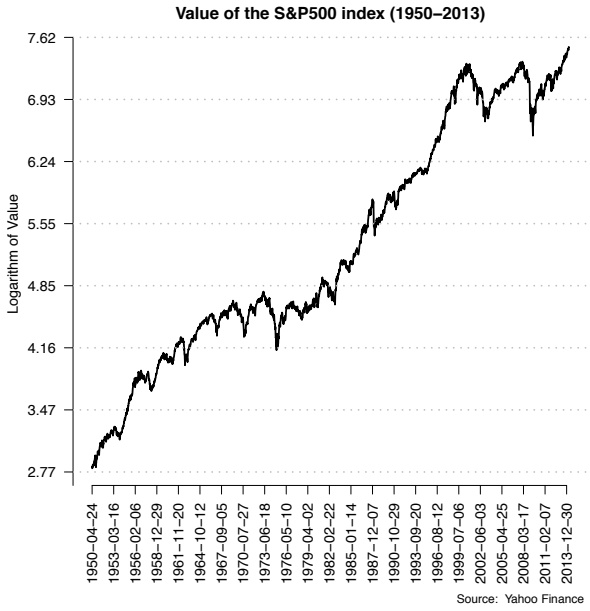


Logarithmic scale

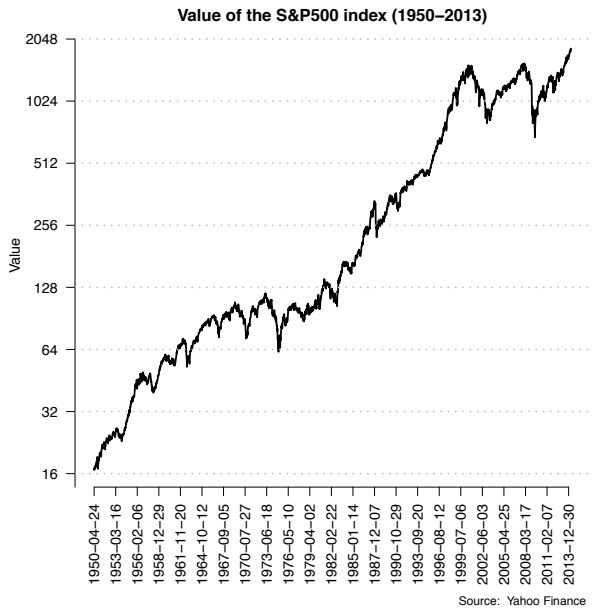




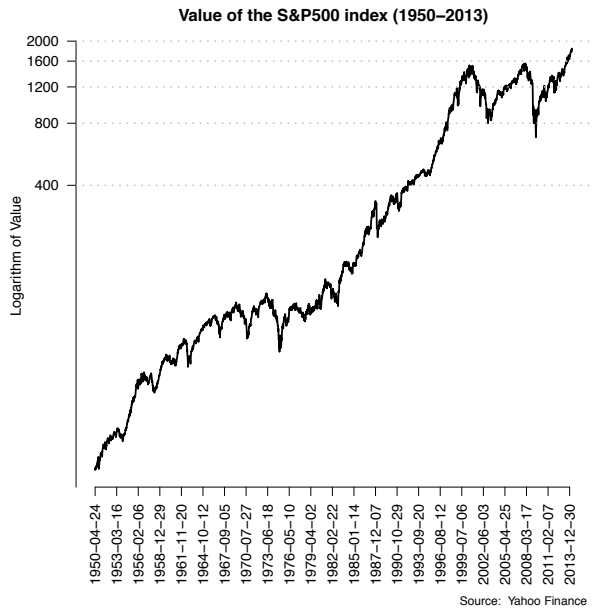
Logarithmic scale



NO!



Most common

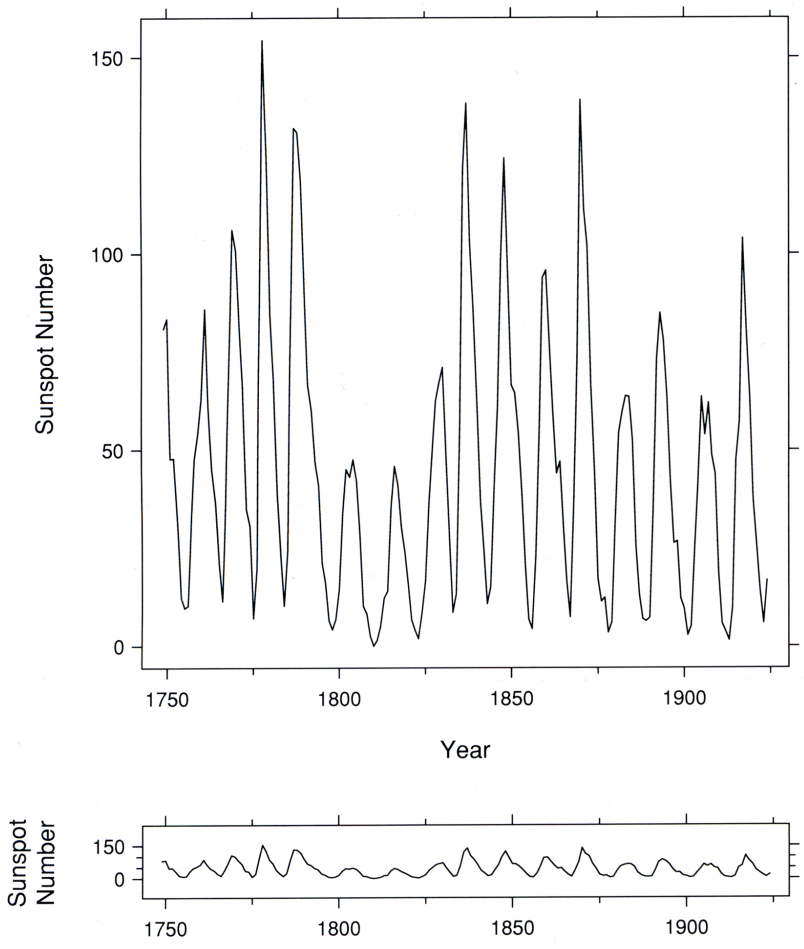


An alternative

- Label axes in the original units!
- Zero cannot be included in the axis!
- Remember that equispaced guidelines correspond to proportional changes

Numerical scales and aspect ratios

- The **aspect ratio** of an image describes the proportional relationship between its width and its height
- If the comparing slopes is important, try to choose a scale and aspect ratio for your plot that makes them close to ±1 (±45°).

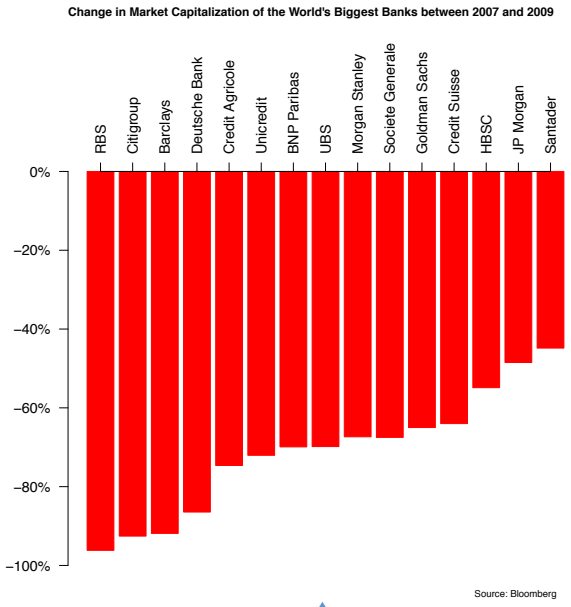


Categorical scales.

- A categorical scale provides visual separation for the different groups.
- Spacing in categorical variables is arbitrary, but should be chosen to improve the legibility of the plot and to allow readers to visually group information.
  - Space between visual cues should be smaller than the visual cues themselves.
  - Don't be cheap! "Negative" (i.e., white) space can make visualizations clearer.
  - White space between bars in a bar plots is important to avoid confusion with a histogram.

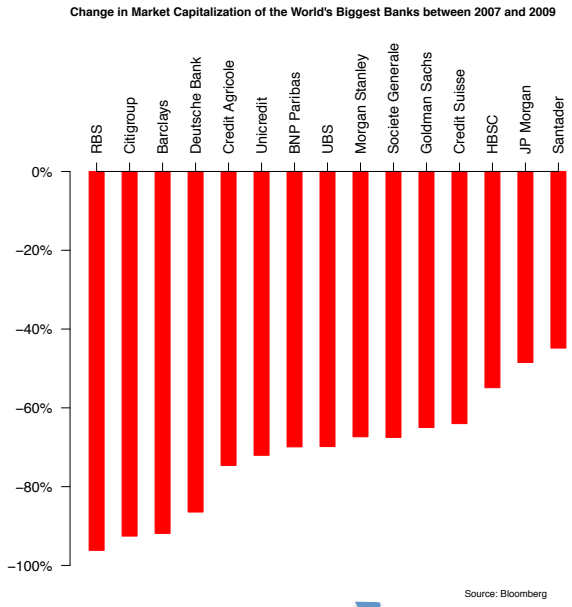
Categorical scales

About the right amount of space between bars

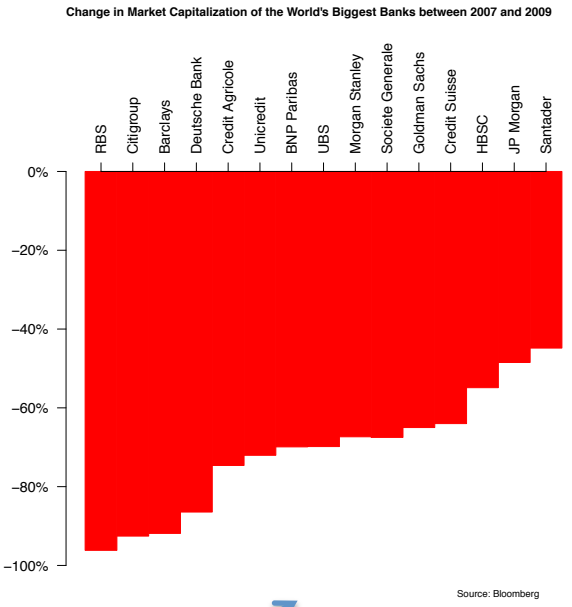


Space in between bars is about 20% of the width of the bar

Too much space between bars



No space between bars

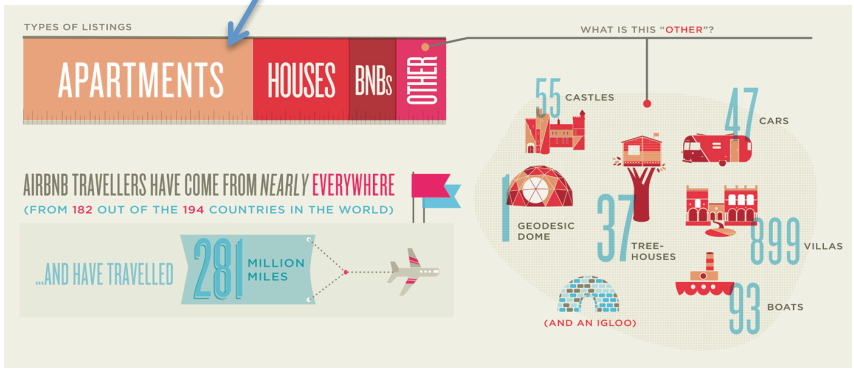


Arguably, less visually appealing

Categorical scales

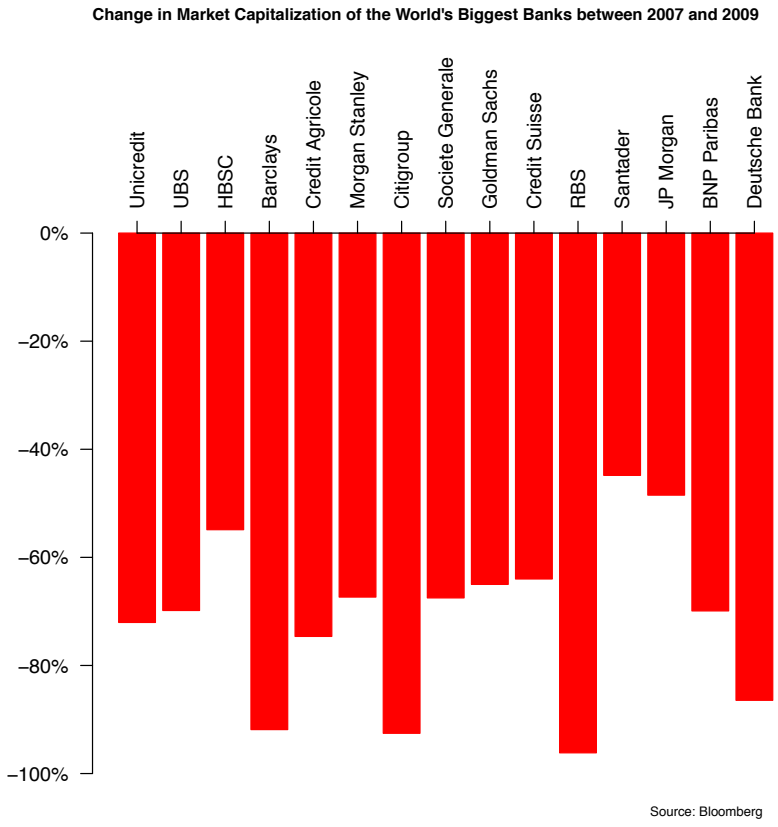
- Too many categories with some having very small counts compared to the rest can make your dot chart or bar plot hard to read.
- Consider combining minor categories into a group called “Others”, which you can explain separately!

Note that there is an axis, but the scale is missing!

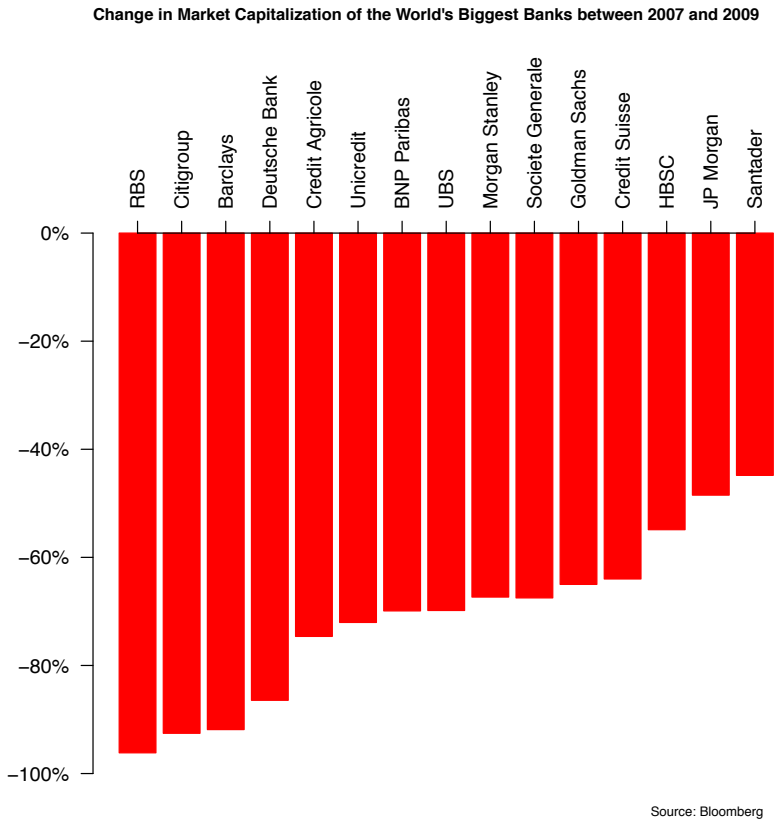


Nominal scales: Diagonalization

Random order for categories

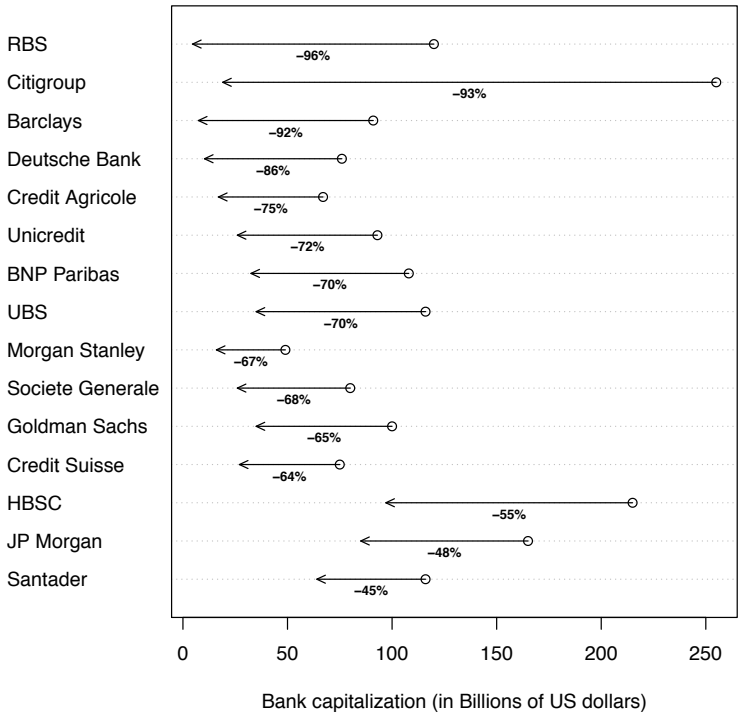


Diagonalized graph

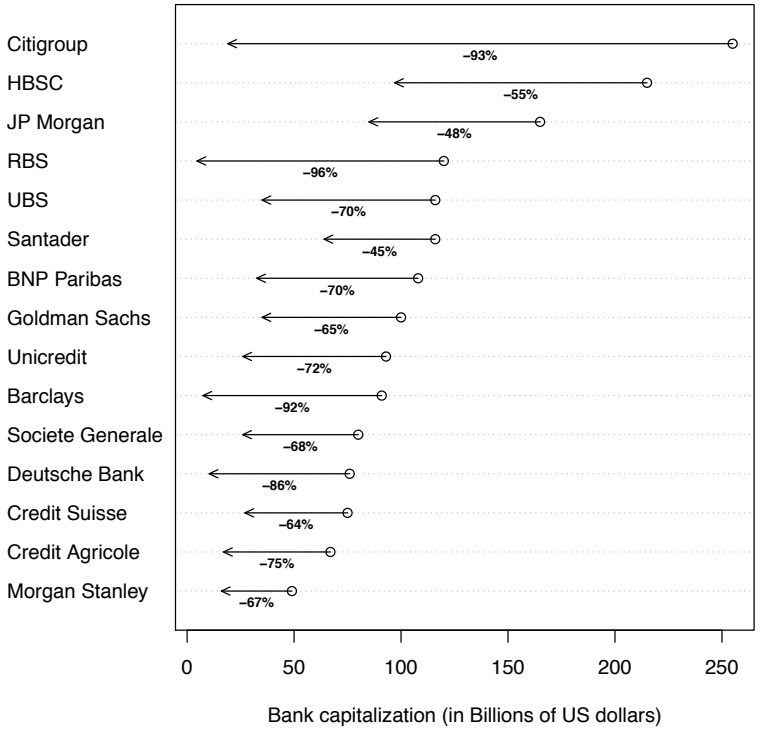


Nominal scales: Diagonalization

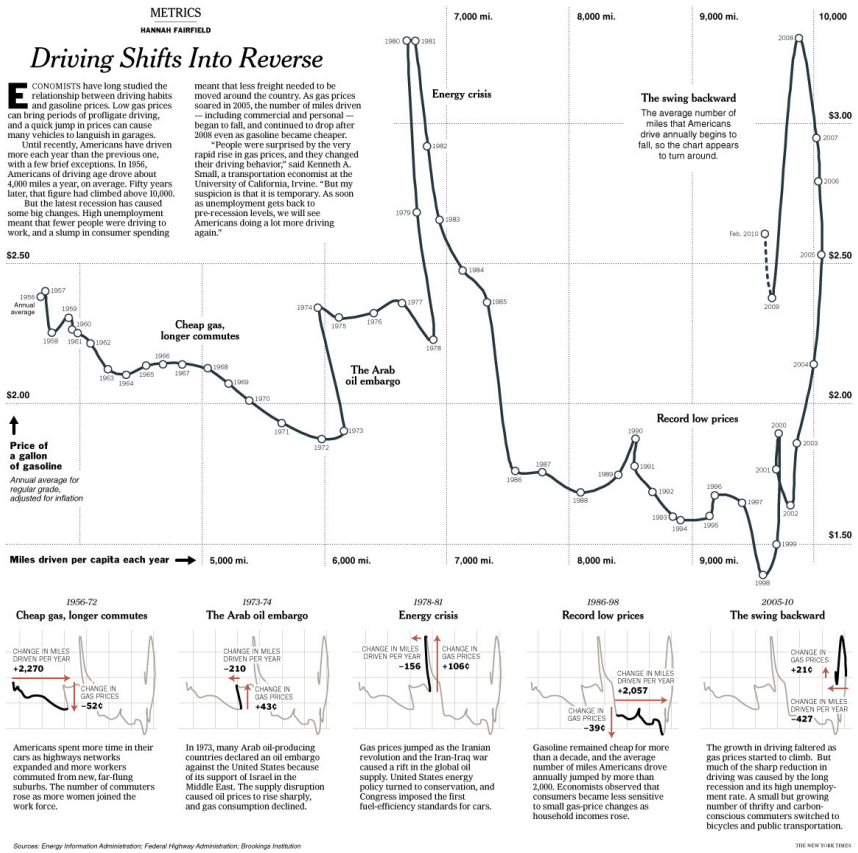
Diagonalized according to % change



Diagonalized according to January 2007 value



Temporal scales



- Temporal scales might be implied rather than explicit.
- In this case, the temporal nature of the data is implied by the line used to connect the points.
- The line is probably generated using a parameterized curve and interpolating splines!

Context

- Title/subtitle should be included
  - Use one as a punch line: “Diamond’s were a girls best friend – Average price of a one-carat D-flawless”.
- Axes labels:
  - As a general rule they should be included, but be alert to situations where they can be dropped.
  - Horizontally oriented labels improve legibility, but they also take more space.
- Legend:
  - Avoid surrounding the legend with a frame.
  - Consider using a horizontally arranged legend.
  - If appropriate, replace the legend with text on the graph.

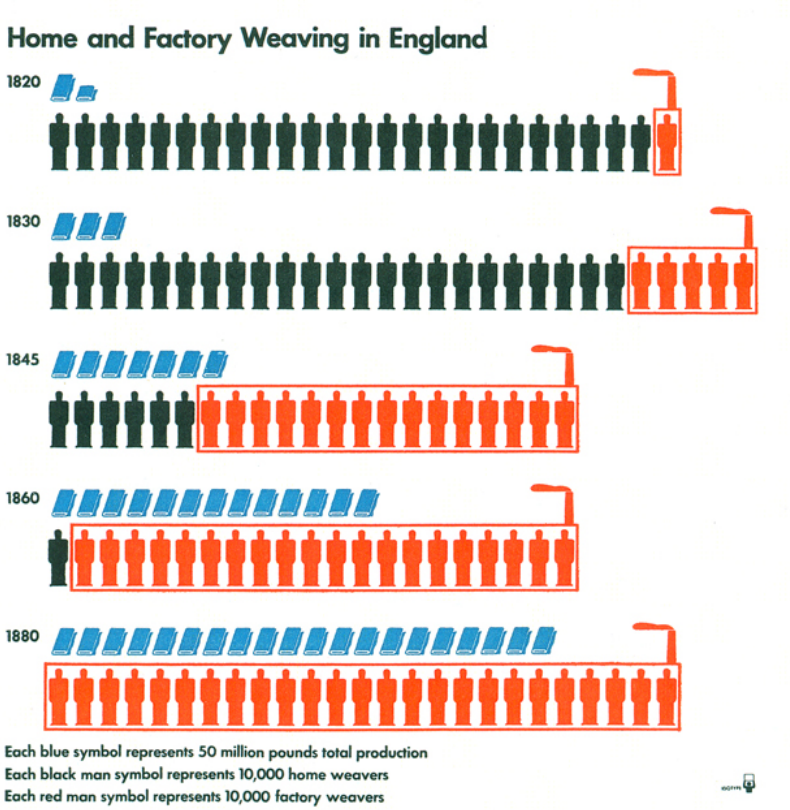
Context

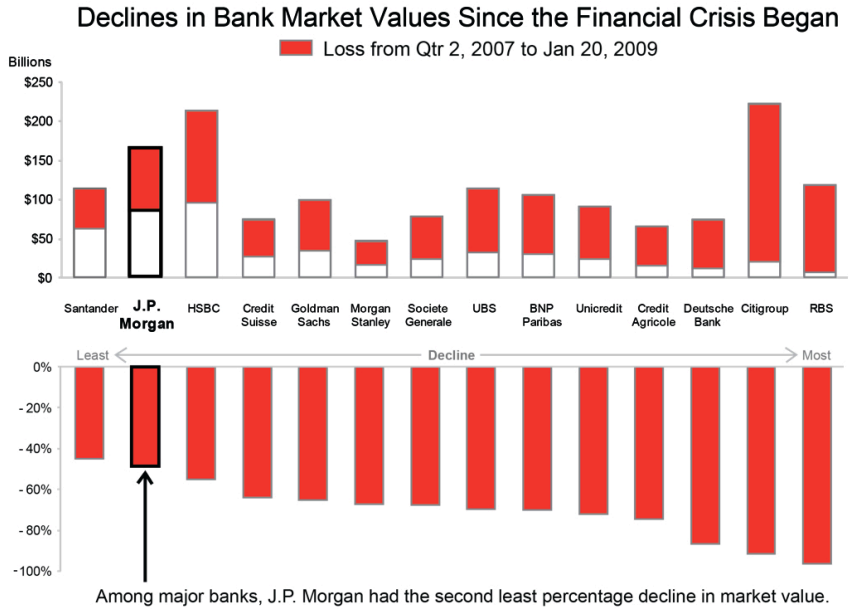
- Grid lines can fulfill three different roles:
  - Enhance the look-up of values and comparison of values.
  - Enhance perception and comparison of localized patterns.
  - Make sure they are not more prominent than the main lines in the graph!
- Highlights:
  - Highlighting certain data points can help make a point.
  - A number different ways to highlight: color (including brighter versions of same hue), enclosings,
  - Elements at the top, the left or the center of the visualization

Context

- Trend and reference lines:
  - A trend line reveals the general direction that a group of points seem to be heading
  - Reference lines/areas provide information about what “normal” values are or mark a point of interest among qualitative variables.
- Other support components:
  - Include raw numbers if possible.
  - Mention the source of your data.
  - Pick font sizes that are readable.
  - Do not forget explicative captions/sidebars are important.
  - I usually avoid borders for the figure, but they can be helpful.

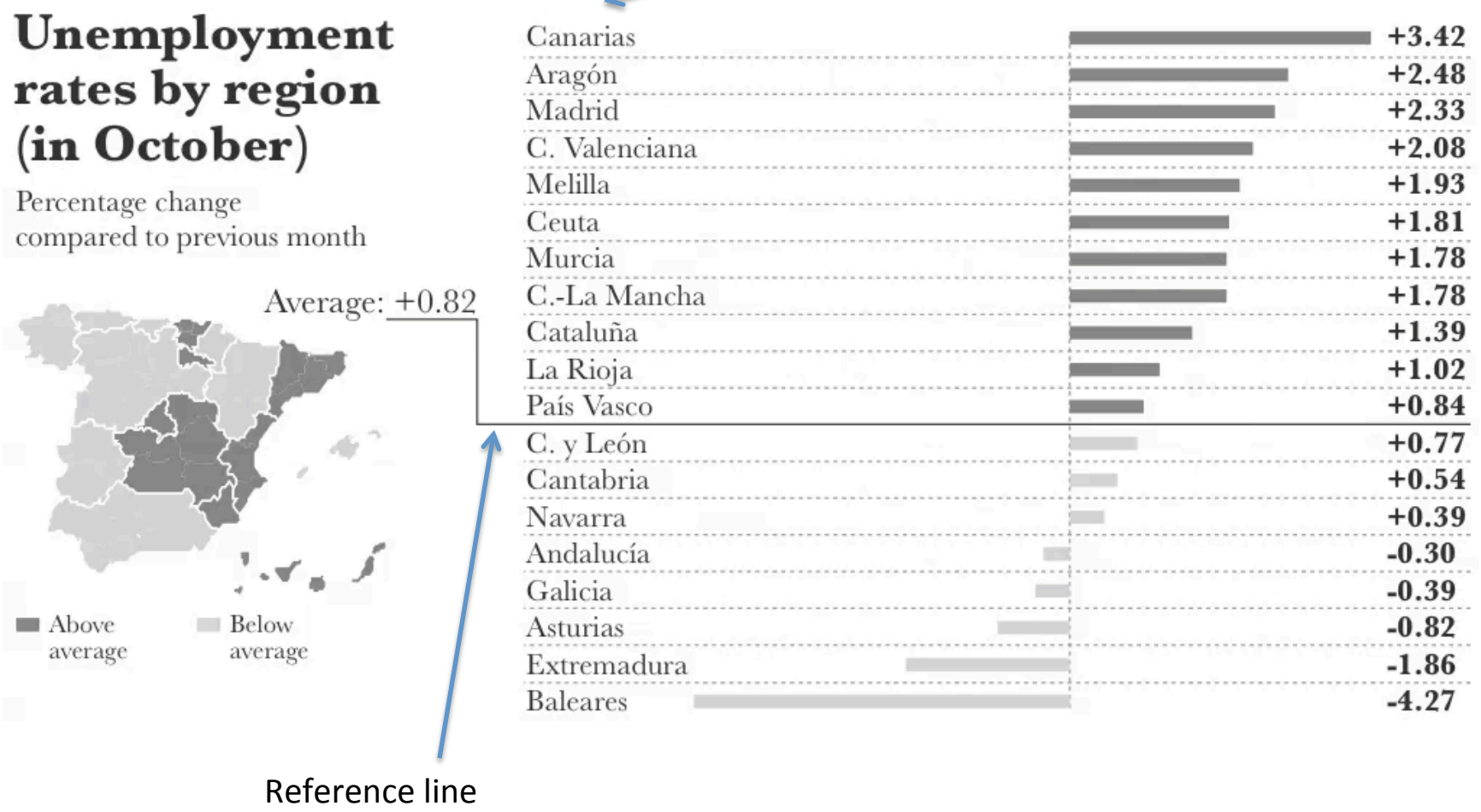
Context


Home and Factory Weaving in England
1820
1830
1845
1860
1880
Each blue symbol represents 50 million pounds total production
Each black man symbol represents 10,000 home weavers
Each red man symbol represents 10,000 factory weavers


Declines in Bank Market Values Since the Financial Crisis Began
Loss from Qtr 2, 2007 to Jan 20, 2009
Billions
$250
$200
$150
$100
$50
$0
Santander
J.P. Morgan
HSBC
Credit Suisse
Goldman Sachs
Morgan Stanley
Societe Generale
UBS
BNP Paribas
Unicredit
Credit Agricole
Deutsche Bank
Citigroup
RBS
Least
Decline
Most
0%
-20%
-40%
-60%
-80%
-100%
Among major banks, J.P. Morgan had the second least percentage decline in market value.

Context

State with the highest unemployment on top



Reference line

Putting it all together

- Group the data: visually (or explicitly!) segment the data into meaningful subsets.
- Prioritize the data: rank the data by importance.
- Sequence the data: give direction for the order in which the data should be read. Storytelling!
- Vertical and horizontal alignment of figures and/or text is important for clear visual flow and to **facilitate comparisons** (particularly across multiple graphs).
- Use the same scale for similar variables on different graphs to **facilitate comparisons**

Basics of plotting in R: Graph displays

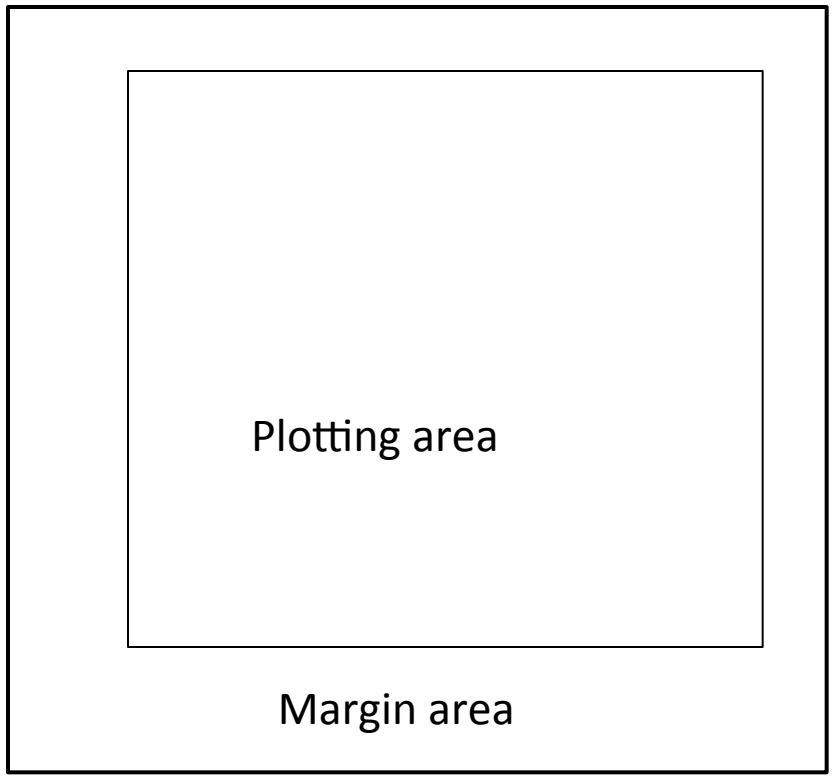
- Creating a new graph display:

Environment	*Command*
Windows	`windows()`
Unix	`x11()`
Mac	`quartz()`

- You can control the size of the box, e.g,

```
quartz(width=5cm,height=5cm)
```

- Other characteristics controlled using `par()`.



Basics of plotting in R: Graph displays

- Margins: usually too big or too small:

```
par(mar=c(4,4,1,1)+0.2)
```

bottom left top right (These are the number of lines of text)

- Partitioning the display into multiple plotting areas:

```
par(mfrow=c(2,2))
```

I typically do not do this, but create each figure individually (more flexibility).

Number of rows Number of columns

- Overlay new graph on top of the previous one

```
par(new=F)
```

Basics of plotting in R: Printing

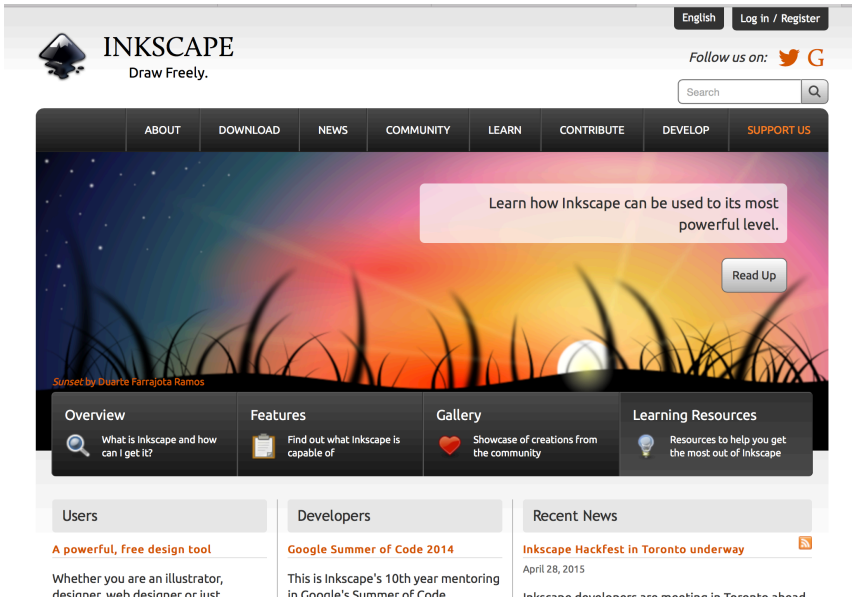
- Most of the options available in `par()` can be passed as parameters to high-level plotting functions.
- Once you have created a graph in a new display you can copy it into a pdf/png/bitmat:

```
dev.print(dev=pdf, file="myplot.pdf")
dev.print(dev=png, file="myplot.png")
dev.print(dev=bmp, file="myplot.bmp")
```

- I recommend that you use formats that correspond to "Scalable Vector Graphics" when saving R graphics.

Scalable Vector Graphics (svg)

- Unlike bitmaps, svg files treat images as a collection of objects that can be manipulated individually.
- Examples of svg formats include .pdf and .eps files!
- Adobe Illustrator is a commercial svg editor. We will use Inkscape, which is free.
- svg editors are a great option to polish or make slight changes to graphs!



https://inkscape.org/en/

Basics of plotting in R: Printing

- If you plan to edit it using Inkscape,

```
dev.print(dev=pdf, file="myplot.pdf", useDingbats=FALSE)
```

- This deals with issues that arise from font availability.
- You can also use pdf/png/bitmat files directly as your device,

```
pdf(file="myplot.pdf", useDingbats=FALSE)
```

but to review what you did you will need to first close the device (`dev.off()`) and then open it using an editor/viewer.

Basics of plotting in R: The `plot` function

- The default plotting function in R is `plot()`
  - If provided with one or two vectors, it creates either a scatterplot (default) or a line plot (using the option `type="l"`).
  - Behavior might be different for other types of objects (this is an overloaded function).
- Empty canvas: I tend to create each of the four components of a graph separately and use `plot` only to set up the graphing region:

```
plot(x, y, type="n", axes=F, xlab="", ylab="")
```

Do not plot Dots or lines

No axes or box around

No x-axis label

No y-axis label

Example: Building a scatterplot

- Area and population of CA counties

Load data

```
cacounties =
read.table(file="california_counties.csv", sep=",")
```

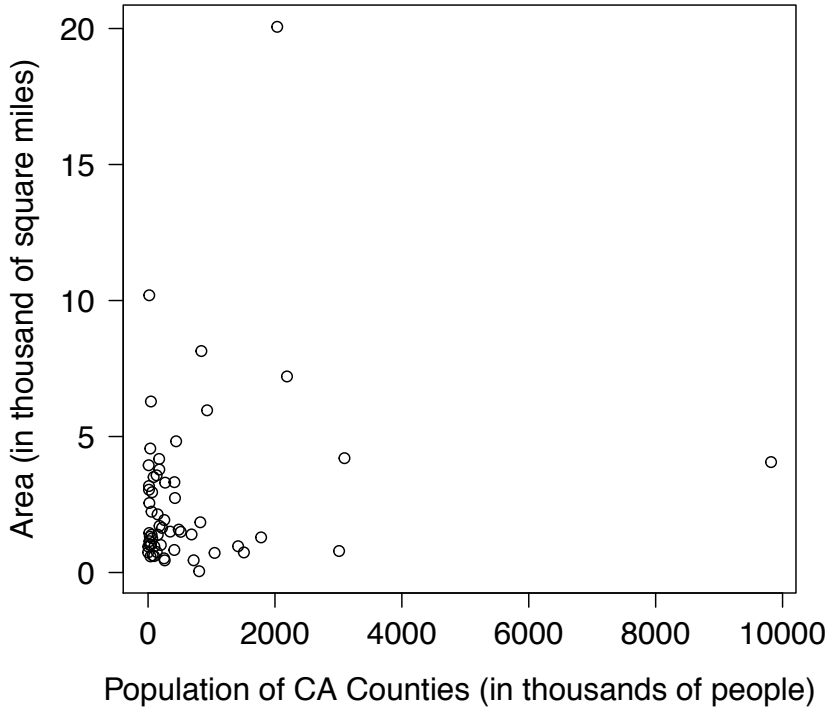
Set graph display

```
quartz()
par(mar=c(4,4,1,1)+0.2)
plot(cacounties[,2]/1000, cacounties[,3]/1000, type="n", xlab="Population of CA Counties (in thousands of people)", ylab="Area (in thousand of square miles)", axes=F)
```

Visual cues + axes

```
points(cacounties[,2]/1000, cacounties[,3]/1000)
box()
axis(1,las=1)
axis(2,las=1)
```

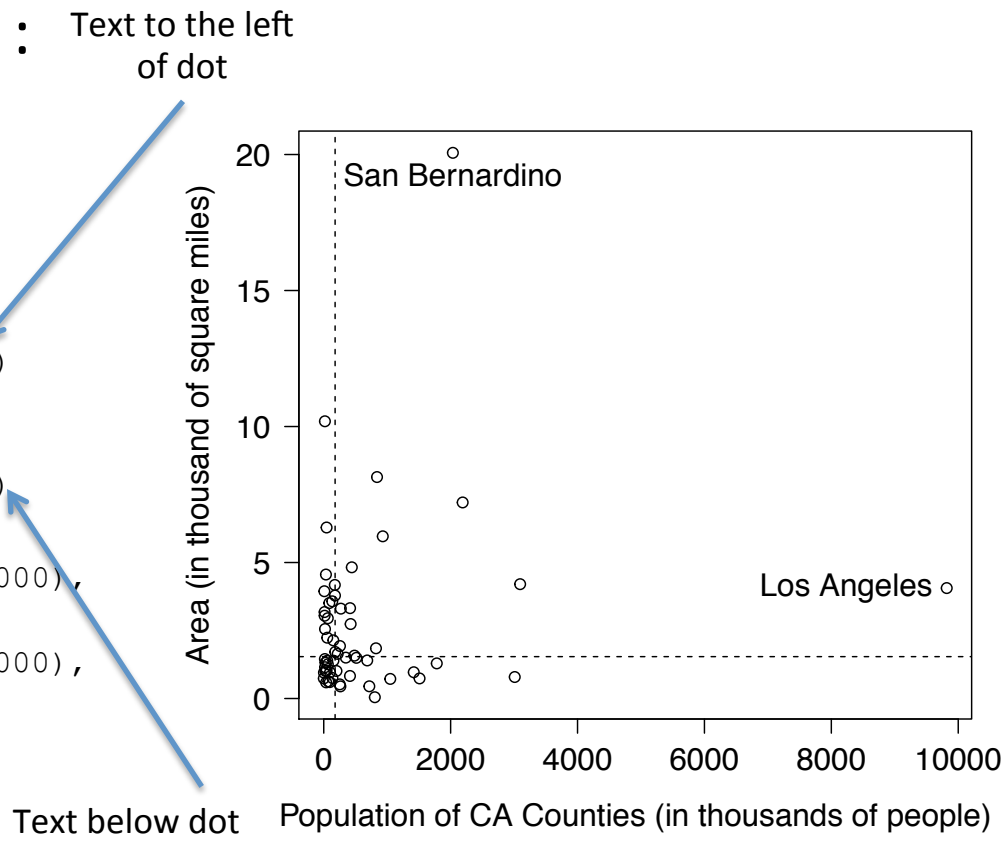
Horizontal tick labels



Building a scatterplot

- You can add text at some locations using `text()`:

```
imax = which.max(cacounties[,2])
jmax = which.max(cacounties[,3])

text(cacounties[imax,2]/1000,
cacounties[imax,3]/1000,
labels=cacounties[imax,1], pos=2)
text(cacounties[jmax,2]/1000,
cacounties[jmax,3]/1000,
labels=cacounties[jmax,1], pos=1)

abline(h=median(cacounties[,3]/1000),
lty=2)
abline(v=median(cacounties[,2]/1000),
lty=2)
```



- `identify()` is an interactive alternative

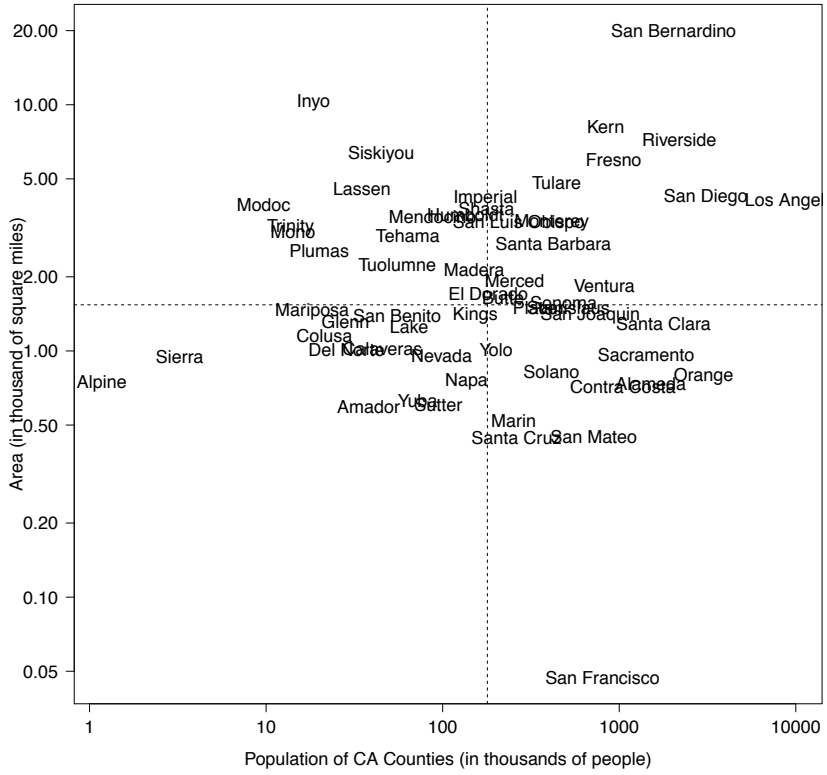
Using a log scale

- You can label all counties and use a logarithmic scale on both axes.

Use logarithmic scale on both axes

```
plot(cacounties[,2]/1000,
cacounties[,3]/1000,
xlab="Population of CA Counties
(in thousands of people)",
ylab="Area (in thousand of
square miles)", las=1, log="xy",
type="n")

abline(h=median(cacounties[,3]/
1000), lty=2)

abline(v=median(cacounties[,2]/
1000), lty=2)

text(cacounties[,2]/1000,
cacounties[,3]/1000, labels=
cacounties[,1])
```

Much more readable, but be careful when interpreting it!

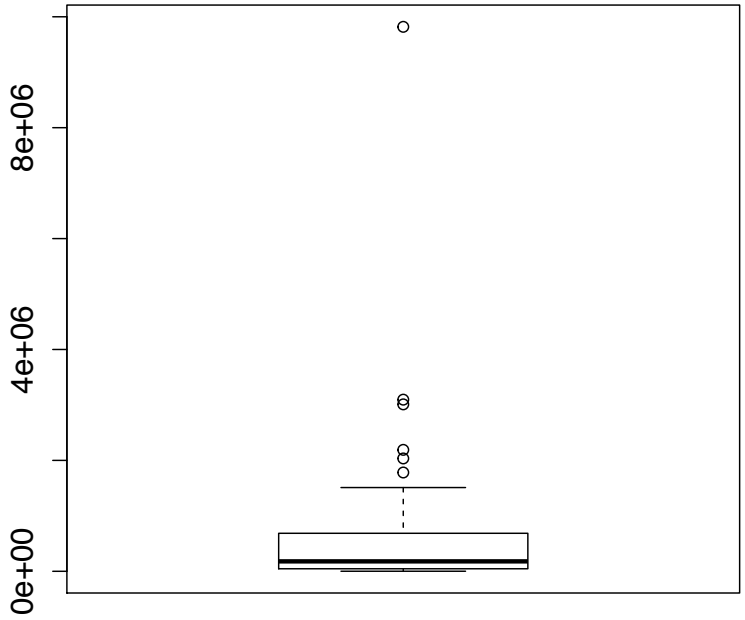


Boxplots

- To create a boxplot of population of California counties use:

```
boxplot(cacounties[,2])
```

- Note that this is not a very pretty image:
  - Labels on y-axes are ugly and unreadable.
  - No labels on x axis.



Building a boxplot

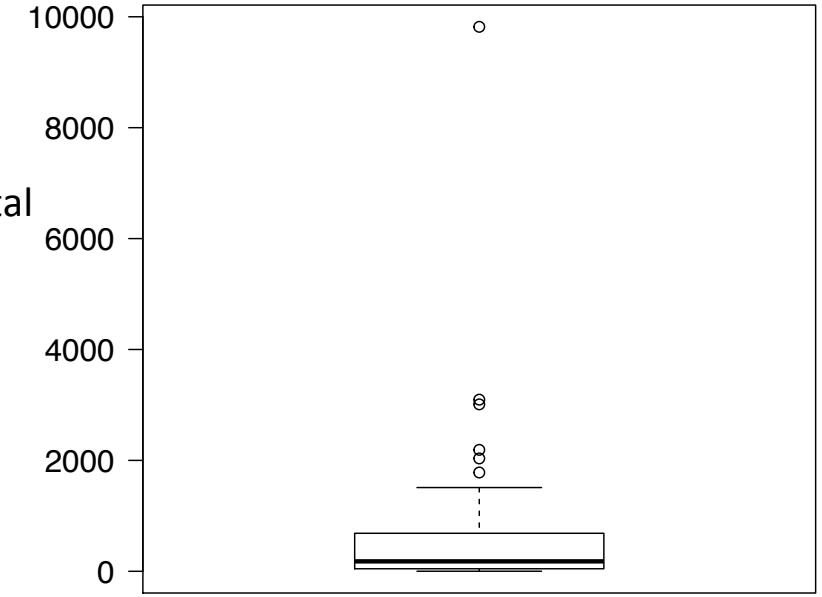
- A slightly better one can be obtained by using some of the options of the function boxplot.

All axis labels are horizontal

Divide by 1000 to get better scale

```
boxplot(cacounties[,2]/1000, las=2,
xlab="Population of CA Counties (in
thousands of people)")
```

Wording of the X-axis label



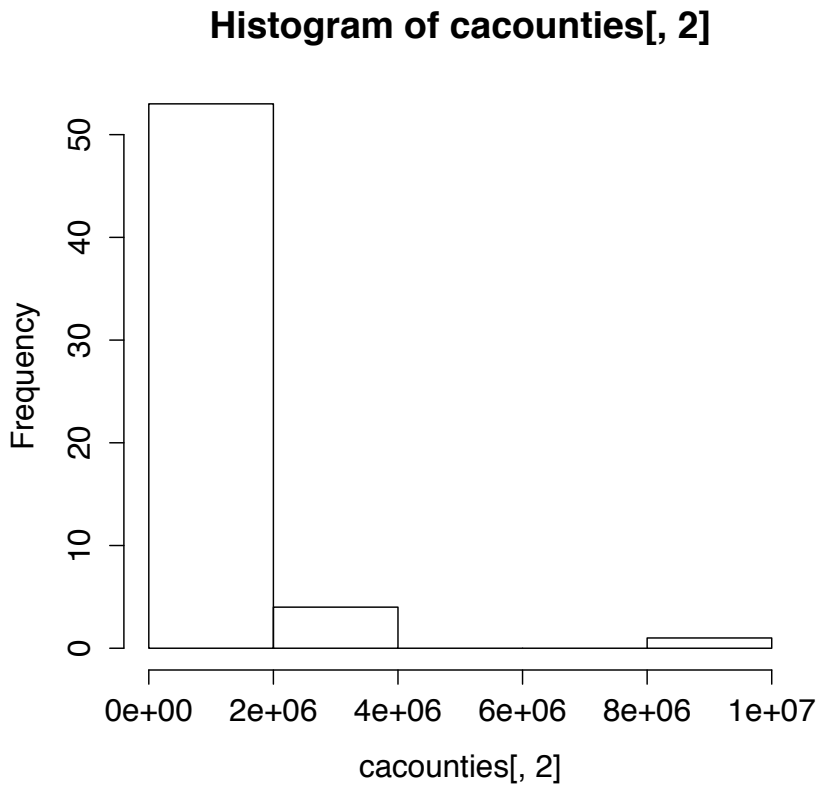
Population of CA Counties (in thousands of people)

Building a histogram

- To create a histogram of population of California counties use:

```
hist(cacounties[,2])
```

- Again, not exactly pretty!



Building a histogram

- A slightly better one can be obtained by using some of the options of the function hist.

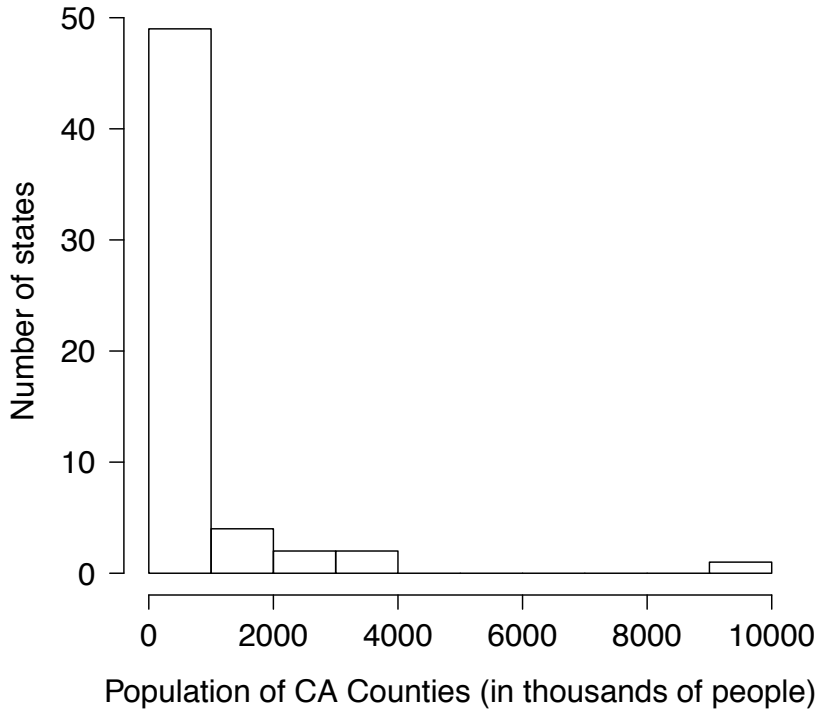
Number of breaks

No main title

```
hist(cacounties[,2]/1000, breaks=9,
main="", xlab="Population of CA
Counties (in thousands of people)",
ylab="Number of states", freq=T, las=1)
```

Y-axis label

Histogram based on frequency



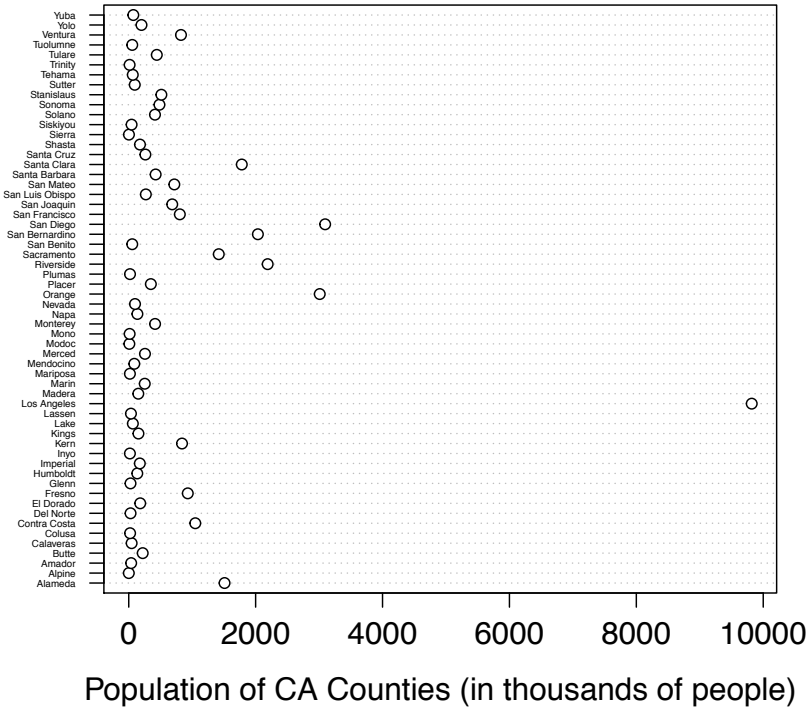
Building a dot chart

- The function `dotchart` can be used to generate dot charts

```
dotchart(cacounties[,2]/
1000, xlab="Population of
CA Counties (in thousands
of people)")
axis(2, at=seq(1,58),
labels=cacounties[,1],
cex.axis=0.3, las=1)
```

Size of font (30% of original size)

- Note that we are adding the labels to the y axes "manually".

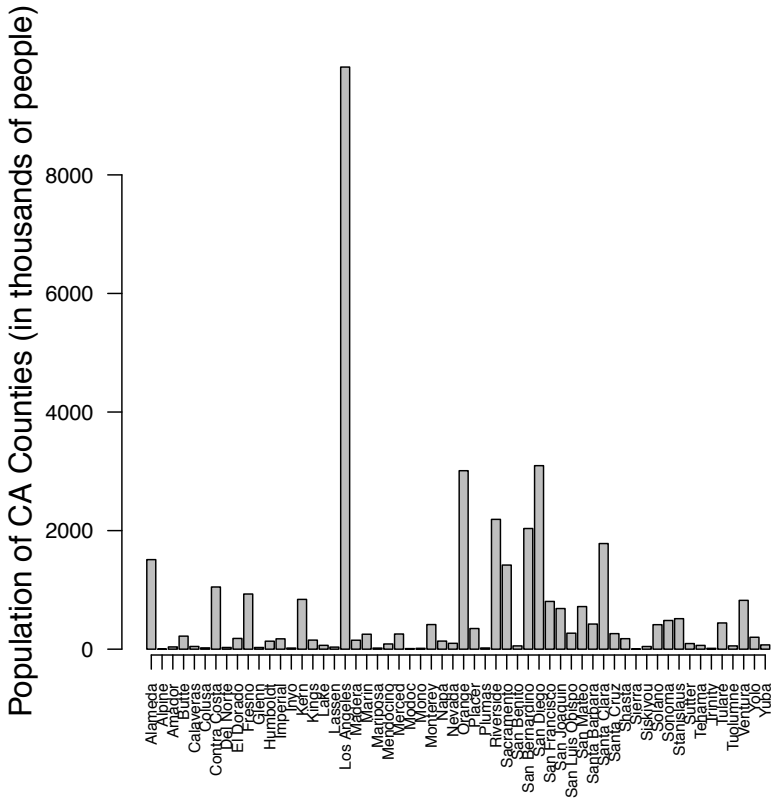


Building a bar plot

- The function barplot can be used to generate bar charts

```
barplot(cacounties[,2]/1000,
ylab="Population of CA Counties
(in thousands of people)",
las=2)
axis(1,
at=seq(0.7,length=58,by=1.2),
labels=cacounties[,1],
cex.axis=0.3, las=2)
```

Width of bars is 1 unit, and separation is .2 units



- Again, we added the x-axis “manually”.

Building a bar plot

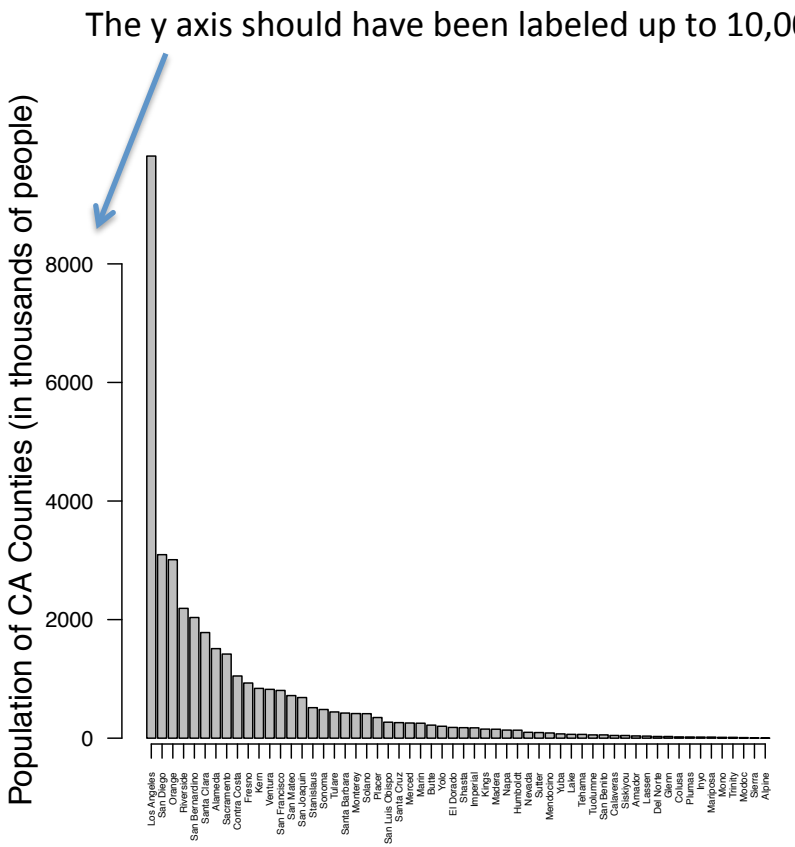
- With a small change, you can order counties by population ("diagonalization")

Organize counties from highest to lower population

```
ord = order(cacounties[,2],
decreasing=T)


barplot(cacounties[ord,2]/1000,
ylab="Population of CA Counties
(in thousands of people)",
las=1)

axis(1,
at=seq(0.7,length=58,by=1.2),
labels=cacounties[ord,1],
cex.axis=0.3, las=2)
```

Use that order for bars and labels

The y axis should have been labeled up to 10,000

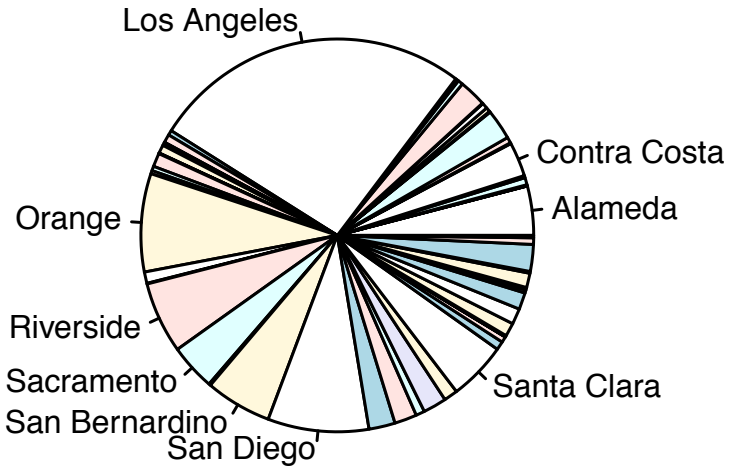


Can you generate a "diagonalized" version of the dot chart?

Building a pie chart

- The function `pie` can be used to generate pie charts.
- We cannot add all names to it: Pick only those of the largest counties!

Not the best option in this case …
Too many categories (color usage and labels).
Hard to make accurate comparisons!




```
labelspiechart =
as.character(cacounties[,1])

labelspiechart[cacounties[,2]<=
quantile(cacounties[,2],0.85)] = ""

pie(cacounties[,2]/1000, labels=
labelspiechart, cex=0.5)
```

For counties with population below the 85% quantile, label is empty!

Using color palettes

- The package RColorBrewer

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

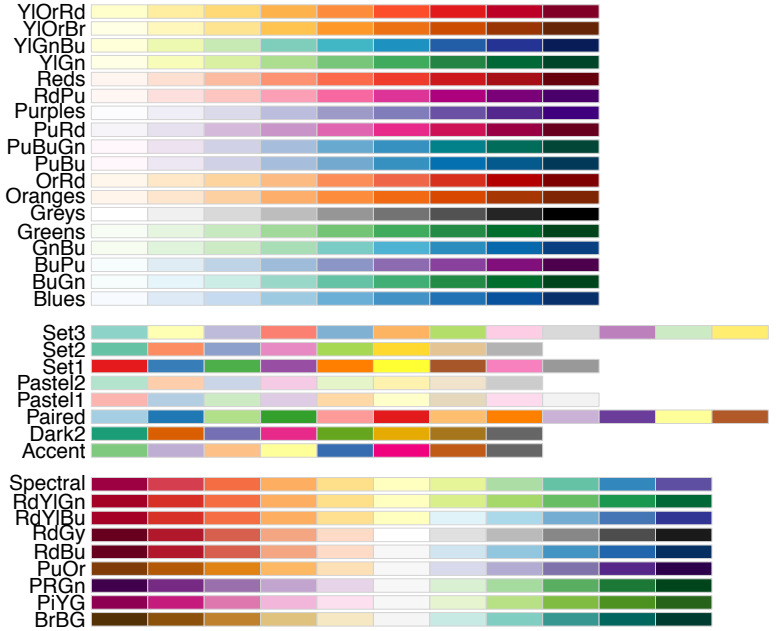
gives you access to the color palettes based on the CIELab scale we discussed before.

- The command

```
display.brewer.all(n=NULL,
type="all", select=NULL,
exact.n=TRUE)
```

Generates the attached palette descriptor.

- The names on the left are used to pick the palette.

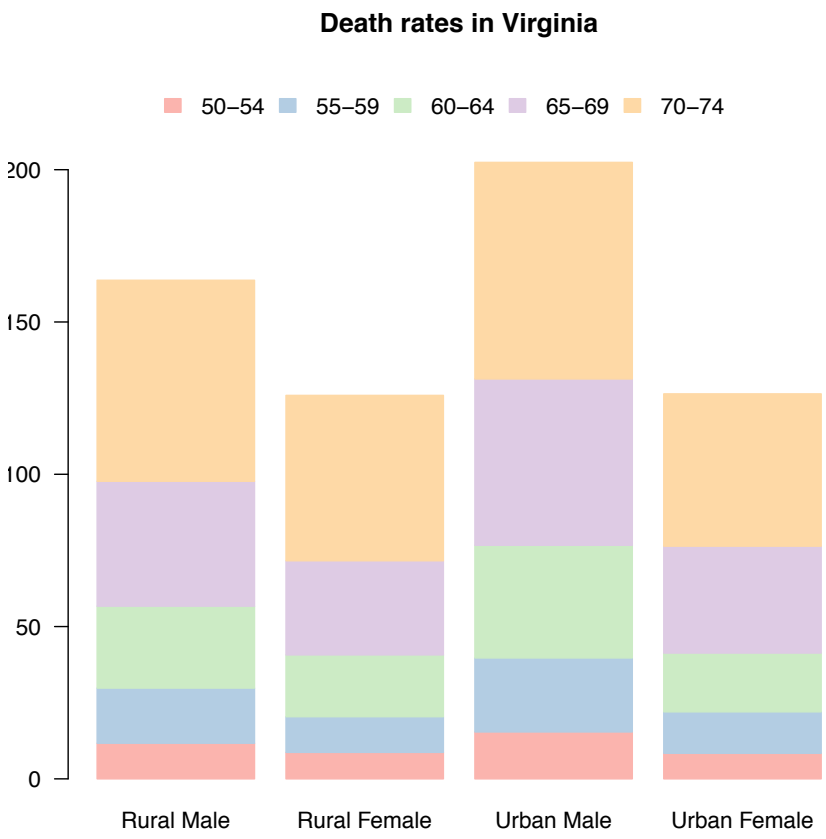


Using color RColorBrewer

Pick the palette to use

```
data(VADeaths)
library(RColorBrewer)
quartz()
colscale=brewer.pal(5,"Pastel1")
barplot(VADeaths, col=colscale,
border=colscale, ylim=c(0, 230), las=1)
title(main = "Death rates in Virginia")
legend(0.5, 230,rownames(VADeaths),
fill=colscale, border=colscale, horiz=T,
bty="n")
```

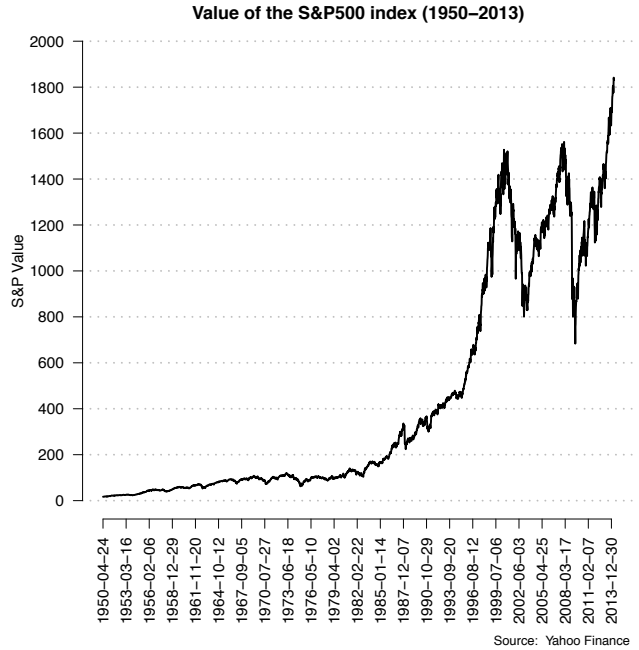
Adds the horizontal legend



Line plot

```
dat = read.table(file="sp500historicalprices.csv", header=T, sep=",")
n   = dim(dat)[1]
subs = seq(1,n, by=151)
quartz()
par(mar=c(7,4,1,1)+0.1)
plot(rev(dat$Close), type="n", axes=F, ylab="S&P Value", xlab="", ylim=c(0,2000))
abline(h=seq(0,2000,by=200), col="grey", lty=3, lwd=2)
lines(rev(dat$Close), lwd=2)
axis(2, las=1, at=seq(0,2000,by=200), labels=seq(0,2000,by=200))
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 (1950-2013)")
```

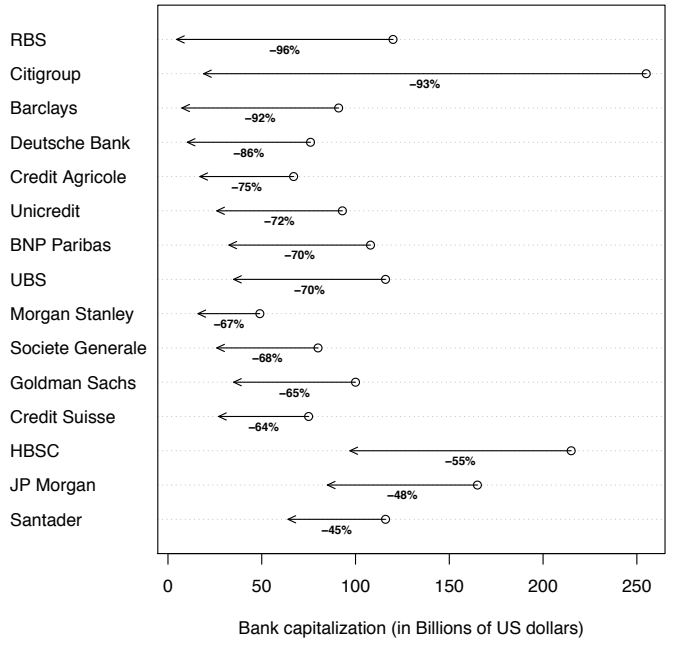
Add text on the margin (in this case, data source)



Fancy dot chart

```
dat         = read.table("marketcapitalizationoftheworldsbiggestbanks.csv", sep=",", header=T, row.names=1)
n           = dim(dat)[1]
dat         = dat[rev(seq(1,n)),]
quartz()
dotchart(dat$January.2007, labels=rownames(dat), xlim=c(min(dat), max(dat)), xlab="Bank capitalization (in Billions of US dollars)")
arrows(dat$January.2007, seq(1,n), dat$January.2009, seq(1,n), col="black", length=0.08, angle=25)
text((dat$January.2007+dat$January.2009)/2, seq(1,n)-0.3, paste(100*round((dat$January.2009 - dat$January.2007)/dat$January.2007, 2), "%", sep=""), cex=0.7, font=2)
```

Adds arrows

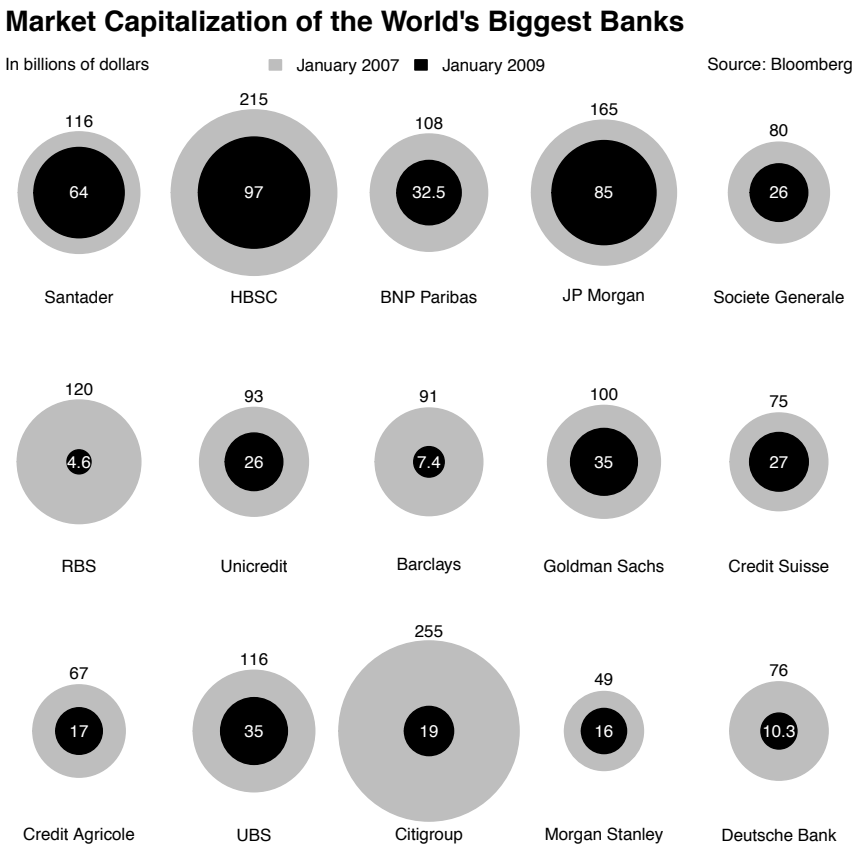


Draws circles
(and other shapes)

Plotting other shapes

```
dat         = read.table("marketcapitalizationoftheworldsbiggestbanks.csv", sep=",", header=T,
row.names=1)

n           = dim(dat)[1]

dat         = dat[sample(seq(1,n), n, replace=F),]

quartz()

par(mar=c(0,0,0,0)+0.1)

plot(seq(1,5), seq(1,5), xlim=c(2.5,27.5), ylim=c(0,13), xlab="",ylab="", type="n", axes=F)

symbols(x = rep(c(seq(5,by=5,length=5),seq(5,by=5,length=5),seq(5,by=5,length=5)),2), y =
rep(c(rep(10,5),rep(6,5),rep(2,5)),2), circles = sqrt(c(dat$January.2007,dat$January.2009)/pi)/
3.5, inches=F, fg=c(rep("grey",15),rep("black",15)), bg=c(rep("grey",15),rep("black",15)),
add=T)

text(x = c(seq(5,by=5,length=5),seq(5,by=5,length=5),seq(5,by=5,length=5)), y =
c(rep(10,5),rep(6,5),rep(2,5))-1.55, labels=rownames(dat), cex=0.75)

legend(10, 12.2, legend=c("January 2007", "January 2009"), fill=c("grey","black"),
border=c("grey","black"), horiz=T, bty="n", cex=0.75)

text(2.5, 11.88, "In billions of dollars", cex=0.75, pos=4)

text(27.5, 11.88, "Source: Bloomberg", cex=0.75, pos=2)

text(2.5, 12.5, "Market Capitalization of the World's Biggest Banks", cex=1.25, pos=4, font=2)

text(x = c(seq(5,by=5,length=5),seq(5,by=5,length=5),seq(5,by=5,length=5)), y =
c(rep(10,5),rep(6,5),rep(2,5)), labels=dat$January.2009, col="white", cex=0.72)

text(x = c(seq(5,by=5,length=5),seq(5,by=5,length=5),seq(5,by=5,length=5)), y =
c(rep(10,5),rep(6,5),rep(2,5)) + sqrt(dat$January.2007/pi)/(2*3.5) + 0.2, labels=dat$January.
2007, cex=0.72)
```

Circles …


Market Capitalization of the World's Biggest Banks
In billions of dollars
January 2007
January 2009
Source: Bloomberg
116
64
Santader
215
97
HBSC
108
32.5
BNP Paribas
165
85
JP Morgan
80
26
Societe Generale
120
4.6
RBS
93
26
Unicredit
91
7.4
Barclays
100
35
Goldman Sachs
75
27
Credit Suisse
67
17
Credit Agricole
116
35
UBS
255
19
Citigroup
49
16
Morgan Stanley
76
10.3
Deutsche Bank

Making it look better in print

- Once you export your graphs into .pdf they might not look exactly the same as they did in R:
  - You might want to make the font in the axes/ labels/titles bigger: (e.g., `cex.lab = 1.2, cex.axis = 1.2`)
  - You might want to use thicker lines (e.g., `lwd=2`).