Graphical Methods and Visualization

- There are two kind of graphics used in data analysis:
  - static graphics
  - dynamic, or interactive, graphics

  There is overlap:

  - interactive tools for building static graphs
- Graphics is used for several purposes
  - exploration and understanding
    - of raw data
    - of residuals
    - of other aspects of model fit, misfit
  - displaying and communicating results
- Historically, display and communication usually used static graphics
- Dynamic graphs were used mostly for exploration
- With digital publishing, dynamic graphics are also used for communication:
  - 2014 as hottest year on record on Bloomberg
  - Subway crime on New York Daily News
  - Who was helped by Obamacare on New York Times' Upshot
  - Paths to the White House on Upshot
  - LA Times years in graphics: 2014 and 2015

Historical Graphics

- Easy construction of graphics is highly computational, but a computer isn't necessary.
- Many graphical ideas and elaborate statistical graphs were creates in the 1800s.
- Some classical examples:
  - Playfair's *The Commercial and Political Atlas* and *Statistical Breviary* introduced a number of new graphs including
    * a bar graph
    * a pie chart
  - Minard developed many elaborate graphs, some available as thumbnail images, including an illustration of Napoleon's Russia campaign
  - Florence Nightingale uses a polar area diagram to illustrate causes of death among British troops in the Crimean war.
  - John Snow used a map (higher resolution) to identify the source of the 1854 London cholera epidemic. An enhanced version is available on `http://www.datavis.ca/`. A short movie has recently been produced.
  - Statistical Atlas of the US from the late 1800s shows a number of nice examples. The complete atlases are also available.
  - Project to show modern data in a similar style.
- Some references:
  - Edward Tufte (1983), *The Visual Display of Quantitative Information*.
  - Michael Friendly (2008), "The Golden Age of Statistical Graphics," *Statistical Science* 8(4), 502-535
  - Michael Friendly's Historical Milestones on `http://www.datavis.ca/`
  - A Wikipedia entry

Graphics Software

- Most statistical systems provide software for producing static graphics
- Statistical static graphics software typically provides
  - a variety of standard plots with reasonable default configurations for
    - bin widths
    - axis scaling
    - aspect ratio
  - ability to customize plot attributes
  - ability to add information to plots
    - legends
    - additional points, lines
    - superimposed plots
  - ability to produce new kinds of plots

  Some software is more flexible than others.
- Dynamic graphical software should provide similar flexibility but often does not.
- Non-statistical graph or chart software often emphasizes "chart junk" over content
  - results may look pretty
  - but content is hard to extract
  - graphics in newspapers and magazines and advertising
  - Some newspapers and magazines usually have very good information graphics
    - New York Times
    - Economist
    - Guardian
    - LA Times

- Chart drawing packages can be used to produce good statistical graphs but they may not make it easy.
- They may be useful for editing graphics produced by statistical software. NY Times graphics creators often
  - create initial graphs in R
  - enhance in Adobe Illustrator

Graphics in R and S-PLUS

- Graphics in R almost exclusively static.
- S-PLUS has some minimal dynamic graphics
- R can work with `ggobi`
- Dynamic graphics packages available for R include
  - `rgl` for 3D rendering and viewing
  - `iplots` Java-based dynamic graphics
  - a number of others in various stages of development
- Three mostly static graphics systems are widely used in R:
  - standard graphics (`graphics` base package)
  - `lattice` graphics (trellis in S-PLUS) (a standard recommended package)
  - `ggplot` graphics (available as `ggplot2` from CRAN)

  Minimal interaction is possible via the `locator` command
- Lattice is more structured, designed for managing multiple related graphs
- `ggplot` represents a different approach based on Wilkinson's *Grammar of Graphics*.

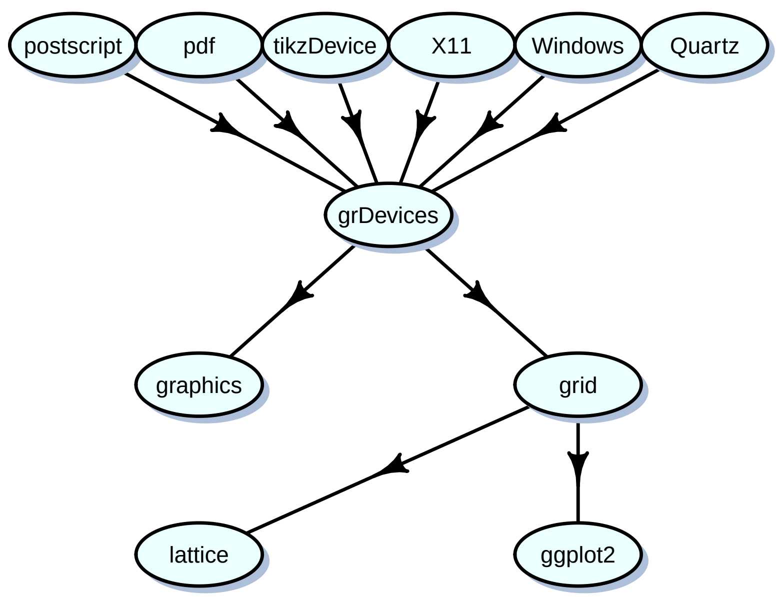
Some References

- Deepayan Sarkar (2008), Lattice: *Multivariate Data Visualization with R*, Springer; has a supporting web page.
- Hadley Wickham (2009), *`ggplot`: Elegant Graphics for Data Analysis*, Springer; has a supporting wep page.
- Paul Murrell (2011), *R Graphics*, 2nd ed., CRC Press; has a supporting web page.
- Josef Fruehwald's introduction to `ggplot`.
- Vincent Zoonekynd's Statistics with R web book; Chapter 3 and Chapter 4 are on graphics.
- Winston Chang (2013), *R Graphics Cookbook*, O'Reilly Media.
- The Graphics task view lists R packages related to graphics.

Some Courses

- Graphics lecture in Thomas Lumley's introductory computing for biostatistics course.
- Ross Ihaka's graduate course on computational data analysis and graphics.
- Ross Ihaka's undergraduate course on information visualization.
- Deborah Nolan's undergraduate course *Concepts in Computing with Data*.
- Hadley Wickham's Data Visualization course

A View of R Graphics



Graphics Examples

- Code for Examples in the remainder of this section is available on line
- Many examples will be from W. S. Cleveland (1993), *Visualizing Data* and N. S. Robbins (2004), *Creating More Effective Graphs*.

Plots for Single Numeric Variables

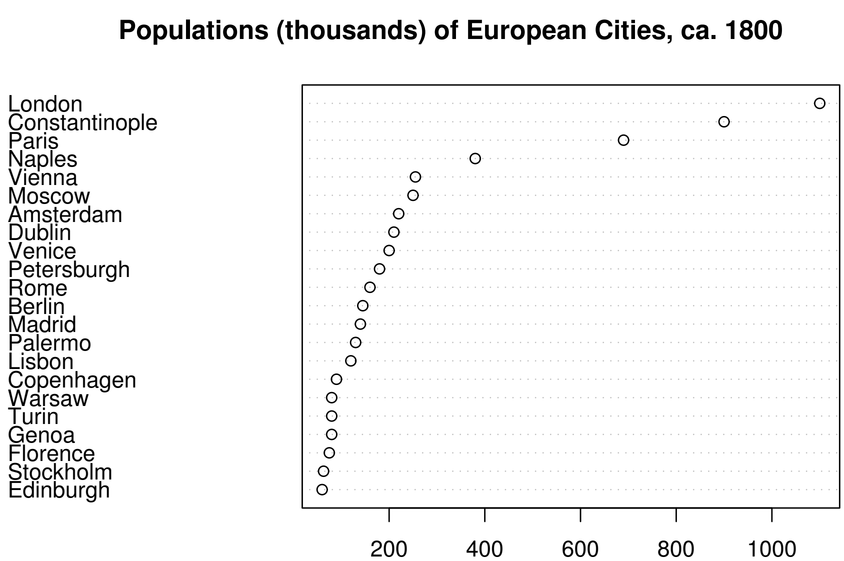
Dot Plots

This uses Playfair's city population data available in the data from Cleveland's *Visualizing Data* book:

```
Playfair <-
  read.table("http://www.stat.uiowa.edu/~luke/classes/STAT7400/examples/Playfair")
```

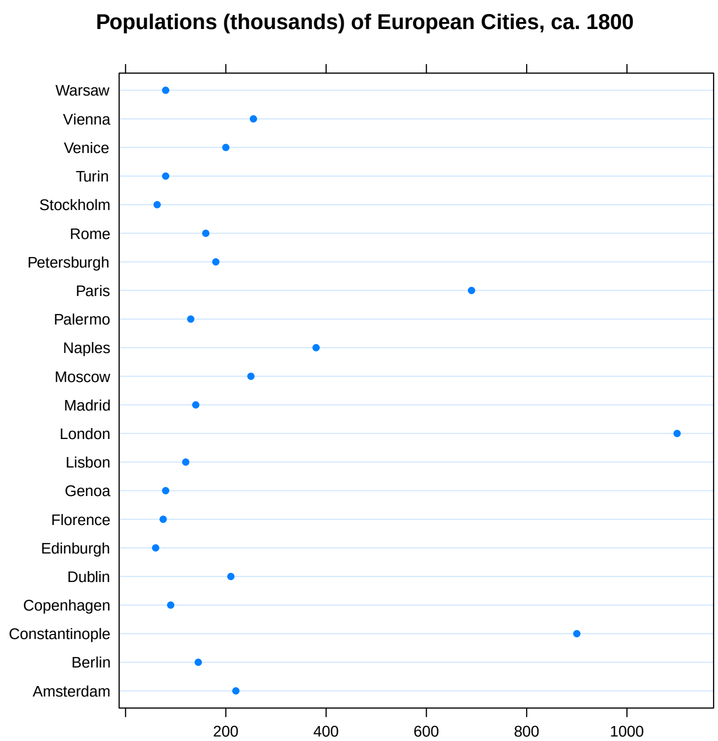
- Useful for modest amounts of data
- Particularly useful for named values.
- Different sorting orders can be useful.
- Standard graphics:

  ```
  dotchart(structure(Playfair[,1],names=rownames(Playfair)))
  title("Populations (thousands) of European Cities, ca. 1800")
  ```



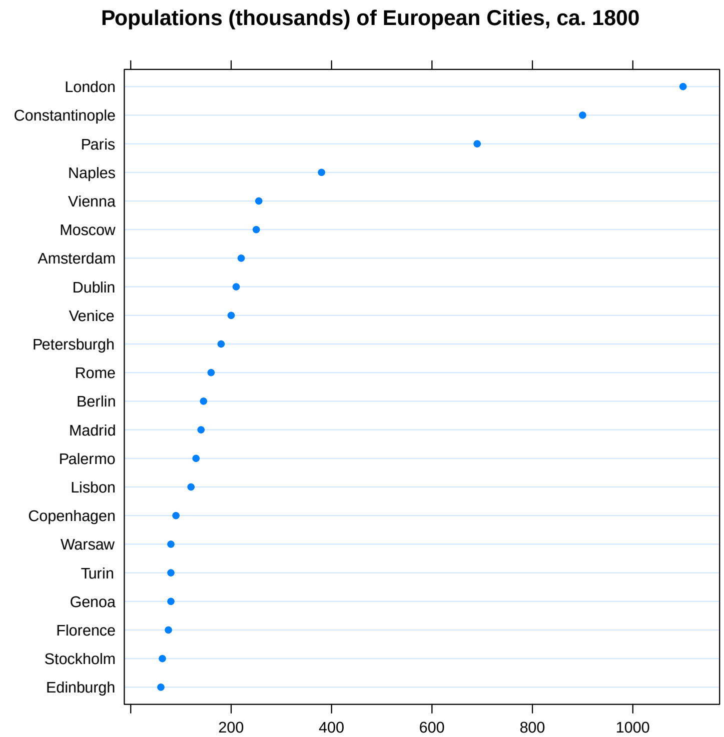
- Lattice uses `dotplot`.

```
library(lattice)
dotplot(rownames(Playfair) ~ Playfair[,1],
        main = "Populations (thousands) of European Cities, ca. 1800",
        xlab = "")
```



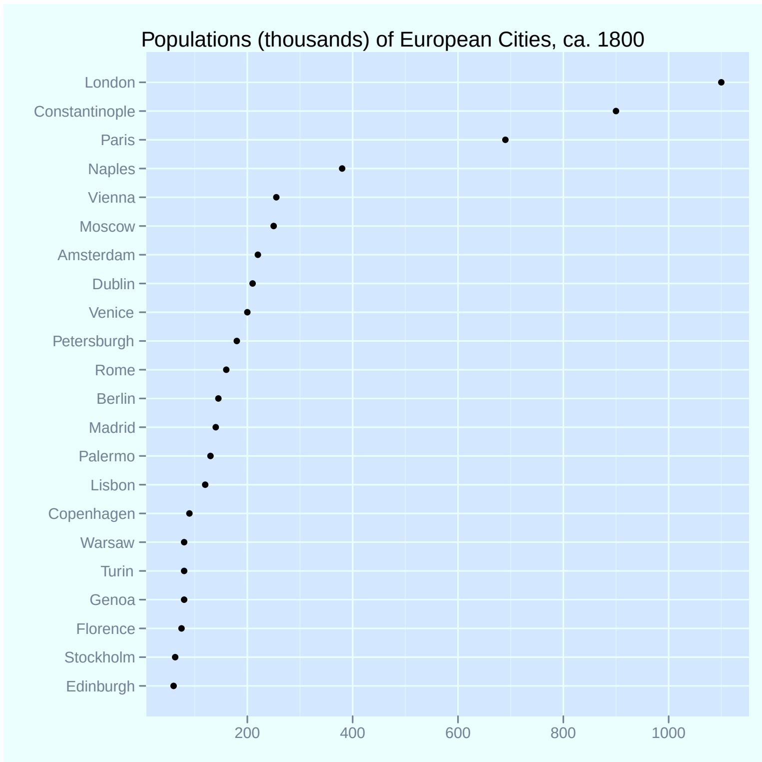
To prevent sorting on names need to convert names to an ordered factor.

```
dotplot(reorder(rownames(Playfair), Playfair[,1]) ~ Playfair[,1],
        main = "Populations (thousands) of European Cities, ca. 1800",
        xlab = "")
```



- `ggplot` graphics

```
library(ggplot2)
qplot(Playfair[,1], reorder(rownames(Playfair), Playfair[,1]),
      main = "Populations (thousands) of European Cities, ca. 1800",
      xlab = "", ylab = "")
```



More Plots for Single Numeric Variables

Bar Charts

An alternative to a dot chart is a bar chart.

- These are more commonly used for categorical data
- They use more “ink” for the same amount of data
- Standard graphics provide `barplot`:

  ```
  barplot(Playfair[,1],names = rownames(Playfair),horiz=TRUE)
  ```

  This doesn’t seem to handle the names very well.

- Lattice graphics use `barchart`:

  ```
  barchart(reorder(rownames(Playfair), Playfair[,1]) ~ Playfair[,1],
           main = "Populations (thousands) of European Cities, ca. 1800",
           xlab = "")
  ```

- `ggplot` graphics:

  ```
  p <- qplot(weight = Playfair[,1],
             x = reorder(rownames(Playfair), Playfair[,1]),
             geom="bar")
  p + coord_flip()
  ```

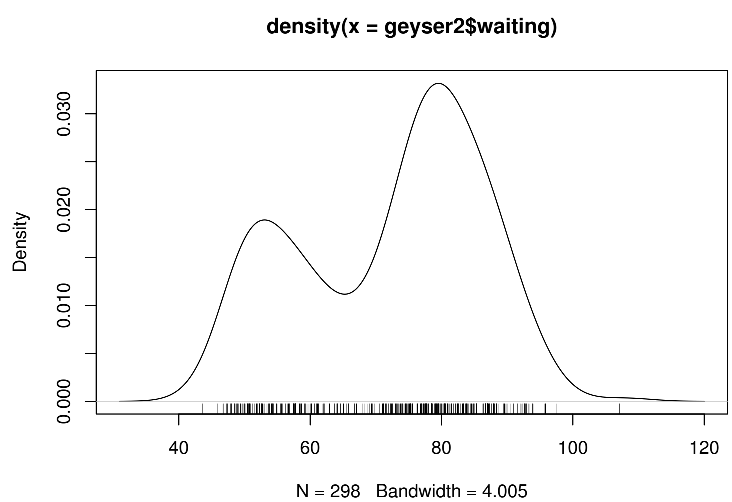
Density Plots

A data set on eruptions of the Old Faithful geyser in Yellowstone:

```
library(MASS)
geyser2 <- data.frame(as.data.frame(geyser[-1,]),
                      pduration=geyser$duration[-299])
```

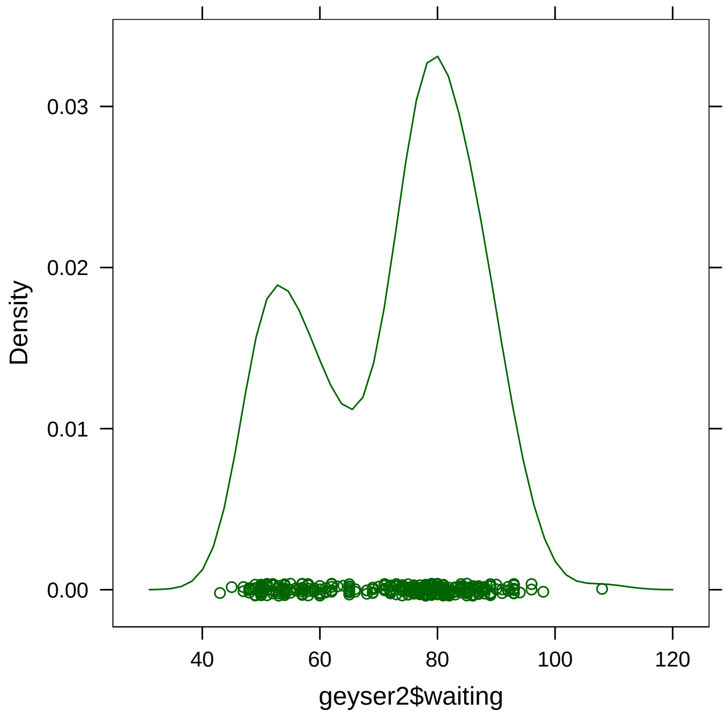
- Standard graphics:

  ```
  plot(density(geyser2$waiting))
  rug(jitter(geyser2$waiting, amount = 1))
  ```



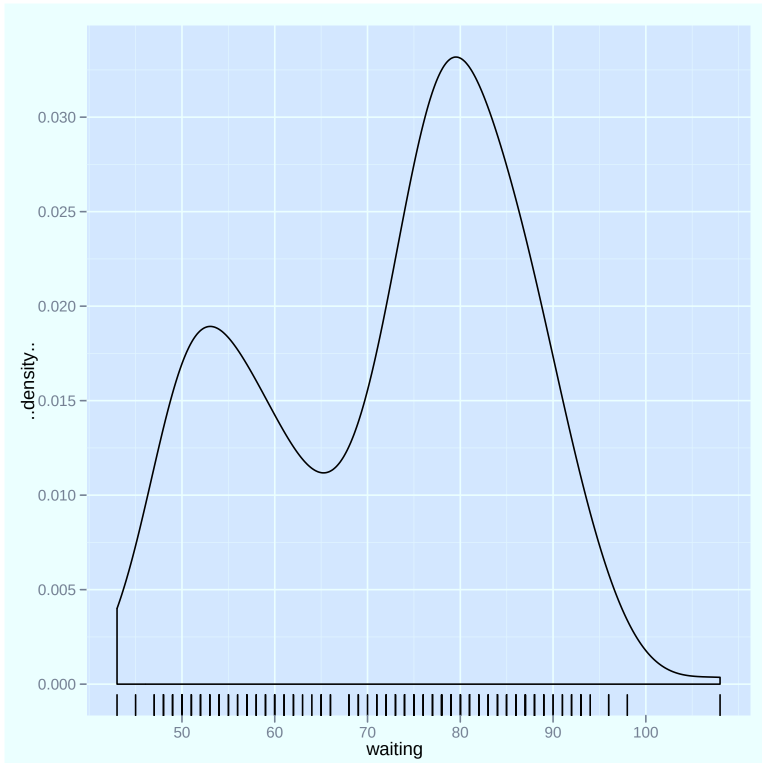
- Lattice graphics:

```
densityplot(geyser2$waiting)
```



- `ggplot2` graphics:

```
qplot(waiting,data=geyser2,geom="density") + geom_rug()
```



Quantile Plots

- Standard graphics

```
data(precip)
qqnorm(precip, ylab = "Precipitation [in/yr] for 70 US cities")
```

- Lattice graphics

```
qqmath(~precip, ylab = "Precipitation [in/yr] for 70 US cities")
```

- `ggplot` graphics

```
qplot(sample = precip, stat="qq")
```

Other Plots

Other options include

- Histograms
- Box plots
- Strip plots; use jittering for larger data sets

Plots for Single Categorical Variables

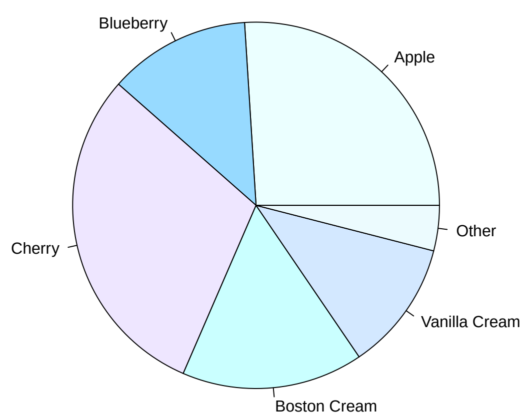
- Categorical data are usually summarized as a contingency table, e.g. using the `table` function.
- A little artificial data set:

```
pie.sales <- c(0.26, 0.125, 0.3, 0.16, 0.115, 0.04)
names(pie.sales) <- c("Apple", "Blueberry", "Cherry",
                      "Boston Cream", "Vanilla Cream",
                      "Other")
```

Pie Charts

- Standard graphics provides the `pie` function:

```
pie(pie.sales)
```



- Lattice does not provide a pie chart, but the Lattice book shows how to define one.
- `ggplot` can create pie charts as stacked bar charts in polar coordinates:

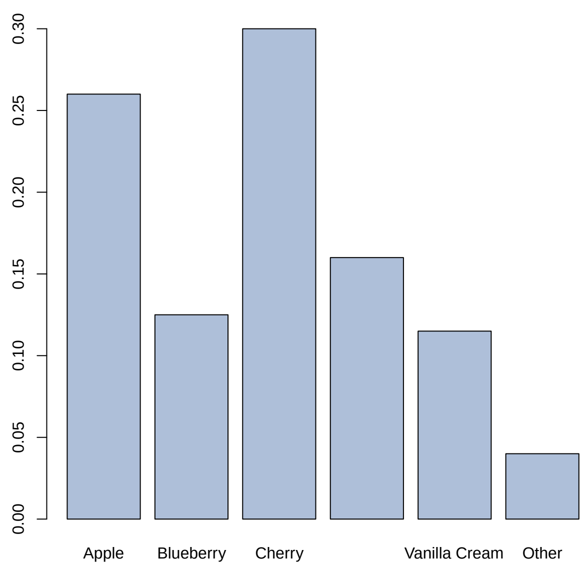
```
qplot(x = "", y = pie.sales, fill = names(pie.sales)) +
    geom_bar(width = 1, stat = "identity") + coord_polar(th

df <- data.frame(sales = as.numeric(pie.sales), pies = name
ggplot(df, aes(x = "", y = sales, fill = pies)) +
    geom_bar(width = 1, stat = "identity") +
    coord_polar(theta = "y")
```

This could use some cleaning up of labels.

Bar Charts

- Standard graphics:

```
barplot(pie.sales)
```



  - One label is skipped to avoid over-printing
  - vertical or rotated text might help.

- Lattice:

```
barchart(pie.sales)
```

- `ggplot`:

```
qplot(x = names(pie.sales), y = pie.sales,
      geom = "bar", stat = "identity")
```

This orders the categories alphabetically.

Plotting Two Numeric Variables

Scatter Plots

- The most important form of plot.
- Not as easy to use as one might think.
- Ability to extract information can depend on aspect ratio.
- Research suggests aspect ratio should be chosen to center absolute slopes of important line segments around 45 degrees.
- A simple example: river flow measurements.

```
river <-
  scan("http://www.stat.uiowa.edu/~luke/classes/STAT7400/examples
plot(river)
xyplot(river~seq_along(river),panel=function(x,y,...) {
    panel.xyplot(x,y,...)
    panel.loess(x,y,...)})
plot(river,asp=4)
plot(river)
lines(seq_along(river),river)
plot(river, type = "b")
```

- Some more Lattice variations

```
xyplot(river~seq_along(river), type=c("p","r"))
xyplot(river~seq_along(river), type=c("p","smooth"))
```

- Some `ggplot` variations

```
qplot(seq_along(river), river)
qplot(seq_along(river), river) + geom_line()
qplot(seq_along(river), river) + geom_line() + stat_smooth(
```

- There is not always a single best aspect ratio.

```
data(co2)
plot(co2)
title("Monthly average CO2 concentrations (ppm) at Mauna Loa Observator
```

Handling Larger Data Sets

An artificial data set:

```
x <- rnorm(10000)
y <- rnorm(10000) + x * (x + 1) / 4
plot(x,y)
```

- Overplotting makes the plot less useful.
- Reducing the size of the plotting symbol can help:

  ```
  plot(x,y, pch=".")
  ```

- Another option is to use translucent colors with *alpha blending*:

  ```
  plot(x,y, col = rgb(0, 0, 1, 0.1, max=1))
  ```

- Hexagonal binning can also be useful:

  ```
  plot(hexbin(x,y))          # standard graphics
  hexbinplot(y ~ x)          # lattice
  qplot(x, y, geom = "hex") # ggplot
  ```

Plotting a Numeric and a Categorical Variable

Strip Charts

- Strip charts can be useful for modest size data sets.

```
stripchart(yield ~ site, data = barley, met)  # standard
stripplot(yield ~ site, data = barley)        # Lattice
qplot(site, yield, data = barley)             # ggplot
```

- *Jittering* can help reduce overplotting.

```
stripchart(yield ~ site, data = barley, method="jitter")
stripplot(yield ~ site, data = barley, jitter.data = TRUE)
qplot(site, yield, data = barley, position = position_jitter(w = 0.1))
```

Box Plots

Box plots are useful for larger data sets:

```
boxplot(yield ~ site, data = barley)                      # standard
bwplot(yield ~ site, data = barley)                       # Lattice
qplot(site, yield, data = barley, geom = "boxplot")  # ggplot
```

Density Plots

- One approach is to show multiple densities in a single plot.
- We would want
  - a separate density for each site
  - different colors for the sites
  - a legend linking site names to colors
  - all densities to fit in the plot
- This can be done with standard graphics but is tedious:

```
with(barley, plot(density(yield[site == "Waseca"])))
with(barley, lines(density(yield[site == "Crookston"]), col = "re
# ...
```

- Lattice makes this easy using the `group` argument:

```
densityplot(~yield, group = site, data = barley)
```

  A legend can be added with `auto.key=TRUE`:

```
densityplot(~yield, group = site, data = barley, auto.key=T
```

- `ggplot` also makes this easy by mapping the site to the `col` aesthetic.

```
qplot(yield, data = barley, geom="density", col = site)
```

- Another approach is to plot each density in a separate plot.
- To allow comparisons these plots should use common axes.
- This is a key feature of Lattice/Trellis graphics:

```
densityplot(~yield | site, data = barley)
```

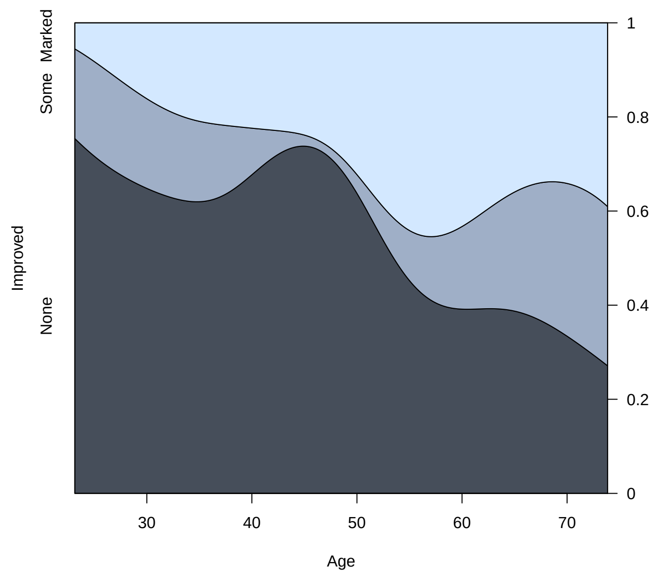
- `ggplot` supports this as *faceting*:

```
qplot(yield, data = barley, geom="density") + facet_wrap(~ site)
```

Categorical Response Variable

Conditional density plots estimate the conditional probabilities of the response categories given the continuous predictor:

```
library(vcd)
data("Arthritis")
cd_plot(Improved ~ Age, data = Arthritis)
```



Plotting Two Categorical Variables

Bar Charts

- Standard graphics:

```
tab <- prop.table(xtabs(~Treatment + Improved, data = Arthr
barplot(t(tab))
barplot(t(tab),beside=TRUE)
```

- Lattice:

```
barchart(tab, auto.key = TRUE)
barchart(tab, stack = FALSE, auto.key = TRUE)
```

  Lattice seems to also require using a frequency table.

- `ggplot`:

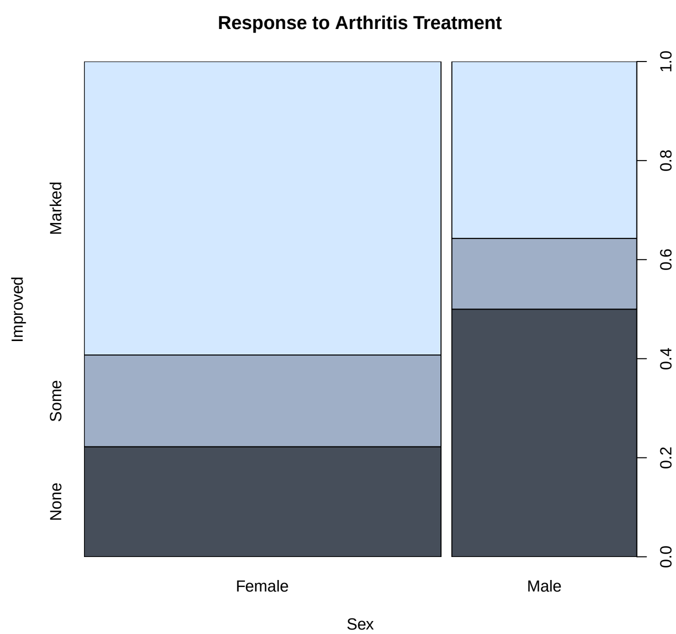
```
qplot(Treatment, geom = "bar", fill = Improved, data = Arth
qplot(Treatment, geom = "bar", fill = Improved,
      position="dodge", data = Arthritis)
qplot(Treatment, geom = "bar", fill = Improved,
      position="dodge", weight = 1/nrow(Arthritis),
      ylab="", data = Arthritis)
```

Plotting Two Categorical Variables

Spine Plots

Spine plots are a variant of stacked bar charts where the relative widths of the bars correspond to the relative frequencies of the categories.

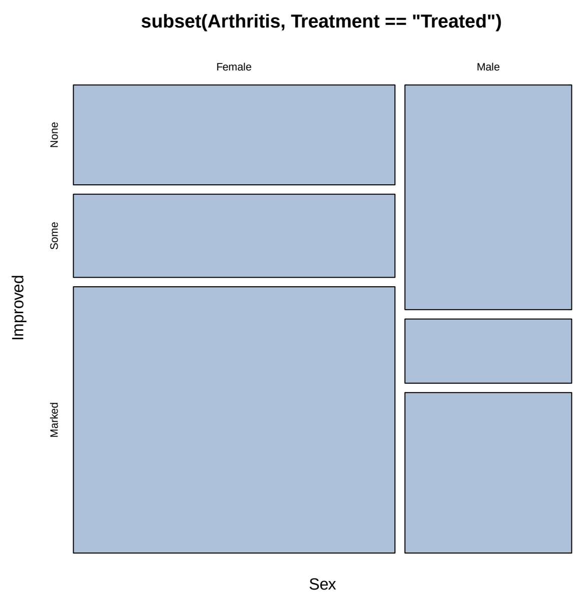
```
spineplot(Improved ~ Sex,
          data = subset(Arthritis, Treatment == "Treated"),
          main = "Response to Arthritis Treatment")
spine(Improved ~ Sex,
      data = subset(Arthritis, Treatment == "Treated"),
      main = "Response to Arthritis Treatment")
```



Mosaic Plots

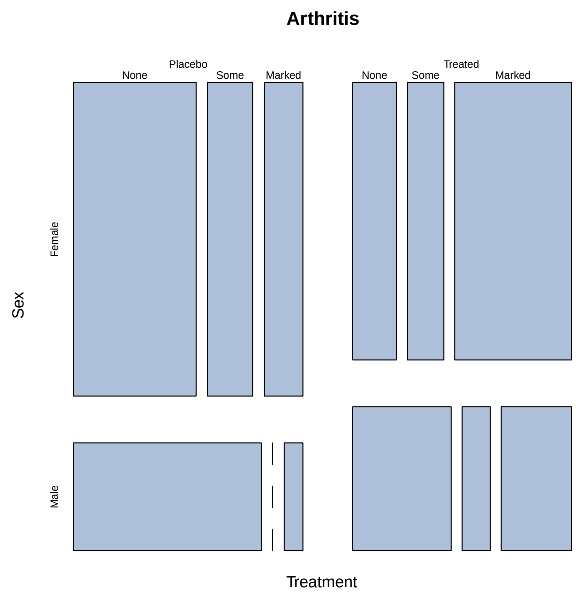
Mosaic plots for two variables are similar to spine plots:

```
mosaicplot(~ Sex + Improved,
           data = subset(Arthritis, Treatment == "Treated"))

mosaic(~ Sex + Improved,
       data = subset(Arthritis, Treatment == "Treated"))
```



Mosaic plots extend to three or more variables:

```
mosaicplot(~ Treatment + Sex + Improved, data = Arthritis)

mosaic(~ Treatment + Sex + Improved, data = Arthritis)
```



Three or More Variables

- Paper and screens are two-dimensional; viewing more than two dimensions requires some trickery
- For three continuous variables we can use intuition about space together with
  - motion
  - perspective
  - shading and lighting
  - stereo
- For categorical variables we can use forms of conditioning
- Some of these ideas carry over to higher dimensions
- For most viewers intuition does not go beyond three dimensions

Some Examples

Soil Resistivity

- Soil resistivity measurements taken on a tract of land.

```
library(lattice)
soilfile <-
    "http://www.stat.uiowa.edu/~luke/classes/STAT7400/examples/soil"
soil <- read.table(soilfile)

p <- cloud(resistivity ~ easting * northing, pch = ".", data = soil)
s <- xyplot(northing ~ easting, pch = ".", aspect = 2.44, data = soil)
print(s, split = c(1, 1, 2, 1), more = TRUE)
print(p, split = c(2, 1, 2, 1))
```

- A loess surface fitted to soil resistivity measurements.

```
eastseq <- seq(.15, 1.410, by = .015)
northseq <- seq(.150, 3.645, by = .015)
soi.grid <- expand.grid(easting = eastseq, northing = northseq)
m <- loess(resistivity ~ easting * northing, span = 0.25,
           degree = 2, data = soil)
soi.fit <- predict(m, soi.grid)
```

- A level/image plot is made with

```
levelplot(soi.fit ~ soi.grid$easting * soi.grid$northing,
          cuts = 9,
          aspect = diff(range(soi.grid$n)) / diff(range(soi.grid$e)),
          xlab = "Easting (km)",
          ylab = "Northing (km)")
```

- An interactive 3D rendered version of the surface:

```
library(rgl)
bg3d(color = "white")
clear3d()
par3d(mouseMode="trackball")
surface3d(eastseq, northseq,
          soi.fit / 100, color = rep("red", length(soi.fit)))
```

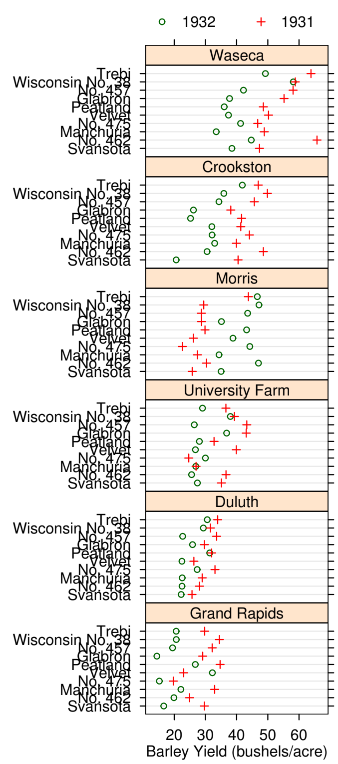
- Partially transparent rendered surface with raw data:

```
clear3d()
points3d(soil$easting, soil$northing, soil$resistivity / 100,
         col = rep("black", nrow(soil)))
surface3d(eastseq, northseq,
          soi.fit / 100, col = rep("red", length(soi.fit)),
          alpha=0.9, front="fill", back="fill")
```

Barley Yields

- Yields of different barley varieties were recorded at several experimental stations in Minnesota in 1931 and 1932
- A dotplot can group on one factor and condition on others:

```
data(barley)
n <- length(levels(barley$year))
dotplot(variety ~ yield | site,
        data = barley,
        groups = year,
        layout = c(1, 6),
        aspect = .5,
        xlab = "Barley Yield (bushels/acre)",
        key = list(points = Rows(trellis.par.get("superpose.symbol"), 1
                   text = list(levels(barley$year)),
                   columns = n))
```



- Cleveland suggests that years for Morris may have been switched.
- A recent article offers another view.

NOx Emissions from Ethanol-Burning Engine

- An experiment examined the relation between nitrous oxide concentration in emissions `NOx` and
  - compression ratio `C`
  - equivalence ratio `E` (richness of air/fuel mixture)
- A scatterplot matrix shows the results

```
data(ethanol)
pairs(ethanol)
splom(ethanol)
```

- Conditioning plots (coplots) can help:

```
with(ethanol, xyplot(NOx ~ E | C))
with(ethanol, {
     Equivalence.Ratio <- equal.count(E, number = 9, overlap = 0.25)
     xyplot(NOx ~ C | Equivalence.Ratio,
            panel = function(x, y) {
                panel.xyplot(x, y)
                panel.loess(x, y, span = 1)
            },
            aspect = 2.5,
            layout = c(5, 2),
            xlab = "Compression Ratio",
            ylab = "NOx (micrograms/J)")
})
```

Three or More Variables

Earth Quakes

- Some measurements on earthquakes recorded near Fiji since 1964
- A scatterplot matrix shows all pairwise distributions:

```
data(quakes)
splom(quakes)
```

- The locations can be related to geographic map data:

```
library(maps)
map("world2",c("Fiji","Tonga","New Zealand"))
with(quakes,points(long,lat,col="red"))
```

- Color can be used to encode depth or magnitude

```
with(quakes,
     points(long,lat,col=heat.colors(nrow(quakes))[rank(depth)]))
```

- Color scale choice has many issues; see `www.colorbrewer.org`
- Conditioning plots can also be used to explore depth:

```
with(quakes,xyplot(lat~long|equal.count(depth)))
```

- Perspective plots are useful in principle but getting the right view can be hard

```
with(quakes,cloud(-depth~long*lat))
library(scatterplot3d)
with(quakes,scatterplot3d(long,lat,-depth))
```

- Interaction with rgl can make this easier:

```
library(rgl)
clear3d()
par3d(mouseMode="trackball")
with(quakes, points3d(long, lat, -depth/50,size=2))
clear3d()
par3d(mouseMode="trackball")
with(quakes, points3d(long, lat, -depth/50,size=2,
                      col=heat.colors(nrow(quakes))[rank(mag)]))
```

Other 3D Options

- Stereograms, stereoscopy.
- Anaglyph 3D using red/cyan glasses.
- Polarized 3D.

Design Notes

- Standard graphics
  - provides a number of basic plots
  - modify plots by drawing explicit elements
- Lattice graphics
  - create an expression that describes the plot
  - basic arguments specify layout vie `group` and conditioning arguments
  - drawing is done by a panel function
  - modify plots by defining new panel functions (usually)
- `ggplot` and Grammar of Graphics
  - create an expression that describes the plot
  - aesthetic elements are associated with specific variables
  - modify plots by adding layers to the specification

Dynamic Graphs

- Some interaction modes:
  - identification/querying of points
  - conditioning by selection and highlighting
  - manual rotation
  - programmatic rotation
- Some systems with dynamic graphics support:
  - S-PLUS, JMP, SAS Insight, ...
  - `ggobi`, `http://www.ggobi.org`
  - `Xmdv`, `http://davis.wpi.edu/~xmdv/`
  - Various, `http://stats.math.uni-augsburg.de/software/`
  - `xlispstat`

Color Issues

Some Issues

- different types of scales, palettes:
  - qualitative
  - sequential
  - diverging
- colors should ideally work in a range of situations
  - CRT display
  - LCD display
  - projection
  - color print
  - gray scale print
  - for color blind viewers
- obvious choices like simple interpolation in RGB space do not work well

Some References

- Harrower, M. A. and Brewer, C. M. (2003). ColorBrewer.org: An online tool for selecting color schemes for maps. *The Cartographic Journal*, 40, 27–37. Available on line. The `RColopBrewer` package provides an R interface.
- Ihaka, R. (2003). Colour for presentation graphics," in K. Hornik, F. Leisch, and A. Zeileis (eds.), *Proceedings of the 3rd International Workshop on Distributed Statistical Computing*, Vienna, Austria. Available on line. See also the R package `colorspace`.
- Lumley, T. (2006). Color coding and color blindness in statistical graphics. *ASA Statistical Computing & Graphics Newsletter*, 17(2), 4–7. Avaivable on line.

- Zeileis, A., Meyer, D. and Hornik, K. (2007). Residual-based shadings for visualizing (conditional) independence. *Journal of Computational and Graphical Statistics*, 16(3), 507–525. See also the R package *vcd*.

- Zeileis, A., Murrell, P. and Hornik, K. (2009). Escaping RGBland: Selecting colors for statistical graphics, *Computational Statistics & Data Analysis*, 53(9), 3259-3270 Available on line.

Perception Issues

- A classic paper:

  William S. Cleveland and Robert McGill (1984), "Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods," *Journal of the American Statistical Association* 79, 531–554.

- The paper shows that accuracy of judgements decreases down this scale:
  - position along a common scale
  - position along non-aligned scales
  - length, direction, angle,
  - area
  - shading, color saturation

- A simple example:

```
x <- seq(0, 2*pi, len = 100)
y <- sin(x)
d <- 0.2 - sin(x+pi/2) * 0.1
plot(x,y,type="l", ylim = c(-1,1.2))
lines(x, y + d, col = "red")
lines(x, d, col = "blue", lty = 2)
```

- Bubble plots
  - An example from Bloomberg.
  - An improved version of the lower row:

```
library(ggplot2)
bankName <- c("Credit Suisse", "Goldman Sachs", "Santander",
              "Citygroup", "JP Morgan", "HSBC")
before <- c(75, 100, 116, 255, 165, 215)
after <- c(27, 35, 64, 19, 85, 92)

d <- data.frame(cap = c(before, after),
                year = factor(rep(c(2007,2009), each=6)),
                bank = rep(reorder(bankName, 1:6), 2))

ggplot(d, aes(x = year, y = bank, size = cap, col = year)) +
    geom_point() +
    scale_size_area(max_size = 20) +
    scale_color_discrete(guide="none")
```

  - A bar chart:

```
ggplot(d, aes(x = bank, y = cap, fill = year)) +
    geom_bar(stat = "identity", position = "dodge") + coord_flip()
```

  - Some dot plots:

```
qplot(cap, bank, col = year, data = d)
qplot(cap, bank, col = year, data = d) + geom_point(size = 4)
do <- transform(d, bank = reorder(bank,rep(cap[1:6],2)))
qplot(cap, bank, col = year, data = do) +
    geom_point(size = 4)
qplot(cap, bank, col = year, data = do) +
    geom_point(size = 4) + theme_bw()
library(ggthemes)
qplot(cap, bank, col = year, data = do) +
    geom_point(size = 4) + theme_economist()
qplot(cap, bank, col = year, data = do) +
    geom_point(size = 4) + theme_wsj()
```

- Our perception can also play tricks, leading to optical illusions.
  - Some examples, some created in R.
  - Some implications for circle and bubble charts.
  - The sine illusion.

Some References

- Cleveland, W. S. (1994), *The Elements of Graphing Data*, Hobart Press.
- Cleveland, W. S. (1993), *Visualizing Data*, Hobart Press.
- Robbins, Naomi S. (2004), *Creating More Effective Graphs*, Wiley; Effective Graphs blog.
- Tufte, Edward (2001), *The Visual Display of Quantitative Information*, 2nd Edition, Graphics Press.
- Wilkinson, Leland (2005), *The Grammar of Graphics*, 2nd Edition, Springer.
- Bertin, Jaques (2010), *Semiology of Graphics: Diagrams, Networks, Maps*, ESRI Press.
- Cairo, Alberto (2012), *The Functional Art: An introduction to information graphics and visualization*, New Riders; The Functional Art blog.
- Few, Stephen (2012), *Show Me the Numbers: Designing Tables and Graphs to Enlighten*, 2nd Edition, Analytics Press; Perceptual Edge blog.

Some Web and Related Technologies

- Google Maps and Earth
  - Mapping earthquakes.
  - Baltimore homicides.
  - Mapping twitter trends.
- SVG/JavaSctipt examples
  - SVG device driver.
  - JavaScript D3 and some R experiments:
    - Contour plots
    - rCharts
- Grammar of Graphics for interactive plots
  - animint package
  - ggvis package; source on github
- Flash, Gapminder, and Google Charts
  - Gapminder: `http://www.gapminder.org/`
  - An example showing wealth and health of nations over time.‘
  - Popularized in a video by Hans Rosling.
  - Google Chart Tools: `https://developers.google.com/chart/`
  - googleVis package.
- Plotly
  - A blog post about an R interface.
- Gif animations
  - Bird migration patterns
- Embedding animations and interactive views in PDF files

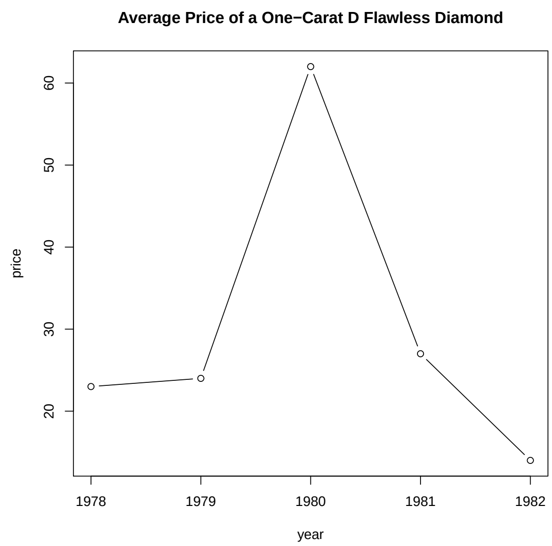
  - Supplemental material to JCGS editorial. (This seems not to be complete; another example is available from my web site.)

- Animations in R
  - `animation` package; has a supporting web site.
  - A simple example is available at the class web site.
  - Rstudio's shiny package.
- Tableau software
  - Tableau Public.

Further References

- Colin Ware (2004), *Information Visualization, Second Edition: Perception for Design*, Morgan Kaufmann.
- Steele, Julie and Iliinsky, Noah (Editors) (2010), *Beautiful Visualization: Looking at Data through the Eyes of Experts*.
- Tufte, Edward (2001), *The Visual Display of Quantitative Information*, 2nd Edition, Graphics Press.
- Tufte, Edward (1990), *Envisioning Information*, Graphics Press.
- Cairo, Alberto (2012), *The Functional Art: An introduction to information graphics and visualization*, New Riders.
- Gelman, Andrew and Unwin, Antony (2013), "Infovis and Statistical Graphics: Different Goals, Different Looks," *JCGS*; links to discussions and rejoinder; slides for a related talk.
- Stephen Few (2011), The Chartjunk Debate A Close Examination of Recent Findings.
- An article in The Guardian.
- Robert Kosara's Eagereyes blog.
- Data Journalism Awards for 2012.
- The Information is Beautiful Awards.

A classic example:



An alternate representation.

Some More References and Links

- Kaiser Fung's Numbers Rule Your World and Junk Charts blogs.
- Nathan Yao's FlowingData blog.
- JSS Special Volume on Spatial Statistics, February 2015.
- An unemployment visualization from the Wall Street Journal.
- A WebGL example from `rgl`

Some Data Technologies

- Data is key to all statistical analyses.
- Data comes in various forms:
  - text files
  - data bases
  - spreadsheets
  - special binary formats
  - embedded in web pages
  - special web formats (XML, JSON, ...)
- Data often need to be cleaned.
- Data sets often need to be reformatted or merged or partitioned.
- Some useful R tools:
  - `read.table`, `read.csv`, and `read.delim` functions.
  - `merge` function for merging columns of two tables based on common keys (data base join operation).
  - The `reshape` function and the `melt` and `cast` functions from the `reshape` or `reshape2` packages for conversion between long and wide formats.

- `tapply` and the `plyr` and `dplyr` packages for
  * partitioning data into groups
  * applying statistical operations to the groups
  * assembling the results
- The `XML` package for reading XML and HTML files.
- The `scrapeR` and `rvest` packages.
- Web Technologies Task View.
- Regular expressions for extracting data from text.

• Some references:

  - Paul Murrell (2009), *Introduction to Data Technologies*, CRC Press; available online at the supporting website,
  - Phil Spector (2008), *Data Manipulation with R*, Springer; available through Springer Link.
  - Deborah Nolan and Duncan Temple Lang (2014), *XML and Web Technologies for Data Sciences with R*, Springer.

Example: Finding the Current Temperature

- A number of web sites provide weather information.
- Some provide web pages intended to be read by humans:
  - Weather Underground.
  - Weather Channel
  - National Weather Service.
- Others provide a web service intended to be accessed by programs:
  - Open Weather Map API.
  - A similar service from Google was shut down in 2012.
  - National Weather Service SOAP API.
  - National Weather Service REST API.
- Historical data is also available, for example from Weather Underground.
- You computer of smart phone uses services like these to display current weather.
- The R package `RWeather` provides access to a number of weather APIs.

- Open Weather Map provides an API for returning weather information in XML format using a URL of the form

  ```
  http://api.openweathermap.org/data/2.5/weather?q=Iowa+
  City,IA&mode=xml&appid=44db6a862fba0b067b1930da0d769e98
  ```

  or

  ```
  http:
  //api.openweathermap.org/data/2.5/weather?lat=41.66&lon=
  -91.53&mode=xml&appid=44db6a862fba0b067b1930da0d769e98
  ```

- Here is a simple function to obtain the current temperature for from Open Weather Map based on latitude and longitude:

  ```
  library(xml2)
  findTempOWM <- function(lat, lon) {
      base <- "http://api.openweathermap.org/data/2.5/weather"
      key <- "44db6a862fba0b067b1930da0d769e98"
      url <- sprintf("%s?lat=%f&lon=%f&mode=xml&units=Imperial&appid=%s",
          base, lat, lon, key)
      page <- read_xml(url)
      as.numeric(xml_text(xml_find_one(page, "//temperature/@value")))
  }
  ```

- For Iowa City you would use

  ```
  findTempOWM(41.7, -91.5)
  ```

- This function should be robust since the format of the response is documented and should not change.
- Using commercial web services should be done with care as there are typically limitations and license terms to be considered.
- They may also come and go: Google's API was shut down in 2012.

Example: Creating a Temperature Map

- The National Weather Service provides a site that produces forecasts in a web page for a URL like this:

```
http://forecast.weather.gov/zipcity.php?inputstring=
                    IowaCity,IA
```

- This function uses the National Weather Service site to find the current temperature:

```
library(xml2)
findTempGov <- function(citystate) {
    url <- paste("http://forecast.weather.gov/zipcity.php?inputstring",
                 url_escape(citystate),
                 sep = "=")
    page <- read_html(url)
    xpath <- "//p[@class=\"myforecast-current-lrg\"]"
    tempNode <- xml_find_one(page, xpath)
    as.numeric(sub("([-+]?[[:digit:]]+).*", "\\1", xml_text(tempNode)))
}
```

- This will need to be revised whenever the format of the page changes, as happened sometime in 2012.
- Murrell's *Data Technologies* book discusses XML, XPATH queries, regular expressions, and how to work with these in R.
- Some other resources for regular expressions:
  - Wikipedia
  - Regular-Expressions.info

- A small selection of Iowa cities

```
places <- c("Ames", "Burlington", "Cedar Rapids", "Clinton",
            "Council Bluffs", "Des Moines", "Dubuque", "Fort Dodge",
            "Iowa City", "Keokuk", "Marshalltown", "Mason City",
            "Newton", "Ottumwa", "Sioux City", "Waterloo")
```

- We can find their current temperatures with

```
temp <- sapply(paste(places, "IA", sep = ", "),
               findTempGov, USE.NAMES = FALSE)
temp
```

- To show these on a map we need their locations. We can optain a file of geocoded cities and read it into R:

```
## download.file("http://www.sujee.net/tech/articles/geocoded/cities.csv.zip",
##               "cities.csv.zip")
download.file("http://www.stat.uiowa.edu/~luke/classes/STAT7400/data/cities.csv.zip",
              "cities.csv.zip")
unzip("cities.csv.zip")
cities <- read.csv("cities.csv", stringsAsFactors=FALSE, header=FALSE)
names(cities) <- c("City", "State", "Lat", "Lon")
head(cities)
```

- Form the temperature data into a data frame and use `merge` to merge in the locations from the cities data frame (a JOIN operation in data base terminology):

```
tframe <- data.frame(City = toupper(places), State = "IA", Temp = temp)
tframe

temploc <- merge(tframe, cities,
                 by.x = c("City", "State"), by.y = c("City", "State"))
temploc
```

- Now use the `map` function from the `maps` package along with the `text` function to show the results:

```
library(maps)
map("state", "iowa")
with(temploc, text(Lon, Lat, Temp, col = "blue"))
```

- To add contours we can use `interp` from the `akima` package and the `contour` function:

```
library(akima)
map("state", "iowa")
surface <- with(temploc, interp(Lon, Lat, Temp, linear = FALSE))
contour(surface, add = TRUE)
with(temploc, text(Lon, Lat, Temp, col = "blue"))
```

- A version using `ggmap`:

```
library(ggmap)
p <- qmplot(Lon, Lat, label = Temp, data = temploc,
            zoom = 7, source = "google") +
    geom_text(color="blue", vjust = -0.5, hjust = -0.3, size = 7)
p
```

- Add contour lines:

```
s <- expand.grid(Lon = surface$x, Lat = surface$y)
s$Temp <- as.vector(surface$z)
s <- s[! is.na(s$Temp),]
p + geom_contour(aes(x = Lon, y = Lat, z = Temp), data = s)
```

Example: 2008 Presidential Election Results

- The New York Times website provides extensive material on the 2008 elections. County by county vote totels and percentages are available, including results for Iowa
- This example shows how to recreate the *choropleth map* shown on the Iowa retults web page.
- The table of results can be extracted using the `XML` package with

  ```
  library(XML)
  url <- "http://elections.nytimes.com/2008/results/states/president/iowa.html"
  tab <- readHTMLTable(url, stringsAsFactors = FALSE)[[1]]
  ```

  Alternatively, using packages `xml2` and `rvest`,

  ```
  library(xml2)
  library(rvest)
  tab <- html_table(read_html(url))[[1]]
  ```

  These results can be formed into a usable data frame with

  ```
  iowa <- data.frame(county = tab[[1]],
                     ObamaPCT = as.numeric(sub("%.*", "", tab[[2]])),
                     ObamaTOT = as.numeric(gsub("votes|,", "", tab[[3]])),
                     McCainPCT = as.numeric(sub("%.*", "", tab[[4]])),
                     McCainTOT = as.numeric(gsub("votes|,", "", tab[[5]])),
                     stringsAsFactors = FALSE)
  head(iowa)
  ```

- We need to match the county data to the county regions. The region names are

  ```
  library(maps)
  cnames <- map("county", "iowa", namesonly = TRUE, plot = FALSE)
  head(cnames)
  ```

- Compare them to the names in the table:

  ```
  which( ! paste("iowa", tolower(iowa$county), sep = ",") == cnames)
  cnames[71]
  iowa$county[71]
  ```

- There is one polygon for each county and they are in alphabetical order, so no elaborate matching is needed.
- An example on the `maps` help page shows how matching on FIPS codes can be done if needed.
- Next, choose cutoffs for the percentage differences and assign codes:

```
cuts <- c(-100, -15, -10, -5, 0, 5, 10, 15, 100)
buckets <- with(iowa, as.numeric(cut(ObamaPCT - McCainPCT, cuts)))
```

- Create a diverging color palette and assign the colors:

```
palette <- colorRampPalette(c("red", "white", "blue"),
                            space = "Lab")(8)
colors <- palette[buckets]
```

- Create the map:

```
map("county", "iowa", col = colors, fill = TRUE)
```

- Versions with no county lines and with the county lines in white:

```
map("county", "iowa", col = colors, fill = TRUE, lty = 0, resolution=0)
map("county", "iowa", col = "white", add = TRUE)
```

- A better pallette:

```
myred <- rgb(0.8, 0.4, 0.4)
myblue <- rgb(0.4, 0.4, 0.8)
palette <- colorRampPalette(c(myred, "white", myblue),
                            space = "Lab")(8)
colors <- palette[buckets]
map("county", "iowa", col = colors, fill = TRUE, lty = 0, resolution=0)
map("county", "iowa", col = "white", add = TRUE)
```

- Some counties have many more total votes than others.
- *Cartograms* are one way to attempt to adjust for this; these have been used to show 2008 and 2012 presidential election results.
- *Tile Grid Maps* are another variation currently in use.
- The New York Times also provides data for 2012 but it seems more difficult to scrape.
- Politoco.com provides results for 2012 that are easier to scrape; the Iowa results are available at

  `http://www.politico.com/2012-election/results/president/iowa/`

ITBS Results for Iowa City Elementary Schools

- The Iowa City Press-Citizen provides data from ITBS results for Iowa City shools.
- Code to read these data is available.
- This code arranges the Standard and Percentile results into a single data frame with additional columns for `Test` and `School`.
- CSV files for the Percentile and Standard results for the elementary schools (except Regina) are also available.
- Read in the Standard results:

```
url <- paste("http://www.stat.uiowa.edu/~luke/classes/STAT7400",
             "examples/ITBS/ICPC-ITBS-Standard.csv", sep = "/")
Standard <- read.csv(url, stringsAsFactors = FALSE, row.names = 1)
names(Standard) <- sub("X", "", names(Standard))
head(Standard)
```

- These data are in *wide* format. To use Lattice or `ggplot` to examine these data we need to convert to *long* format.
- This can be done with the `reshape` function or the function `melt` in the `reshape2` package:

```
library(reshape2)
mS <- melt(Standard, id=c("Grade", "Test", "School"),
           value.name = "Score", variable.name = "Year")
head(mS)
```

- Some Lattice plots:

```
library(lattice)
xyplot(Score ~ Grade | Year, group = Test, type = "l", data = mS,
       auto.key = TRUE)
xyplot(Score ~ Grade | Year, group = Test, type = "l", data = mS,
       subset = School == "Lincoln", auto.key = TRUE)
xyplot(Score ~ Grade | Year, group = Test, type = "l", data = mS,
       subset = Test %in% c("SocialScience", "Composite"),
       auto.key = TRUE)
```

Studying the Web

- Many popular web sites provide information about their use.
- This kind of information is now being actively mined for all sorts of purposes.
- Twitter provides an API for collecting information about "tweets."
  - The R package `twitteR` provides an interface to this API.
  - A simple introduction (deprecated but may still be useful).
  - One example of its use involves mining twitter for airline consumer sentiment.
  - Another example is using twitter activity to detect earthquakes.
- Facebook is another popular framework that provides some programmatic access to its information.
  - The R package `Rfacebook` is available.
  - One blog post shows how to access the data.
  - Another provides a simple illustration.
- Google provides access to a number of services, including
  - Google Maps
  - Google Earth
  - Google Visualization
  - Google Correlate
  - Google Trends

  R packages to connect to some of these and others are available.

- Some other data sites:
  - Iowa Government Data
  - New York Times Data
  - Guardian Data
- Nice summary of a paper on deceptive visualizations.