

# Data Visualization in R

## 1. Overview

**PREPARED FOR**  
**Engineering Students**  
**All Engineering College**

**PREPARED BY: MS. SHWETA TIWARI**  
**Published On: March 12, 2022**

# Course outline

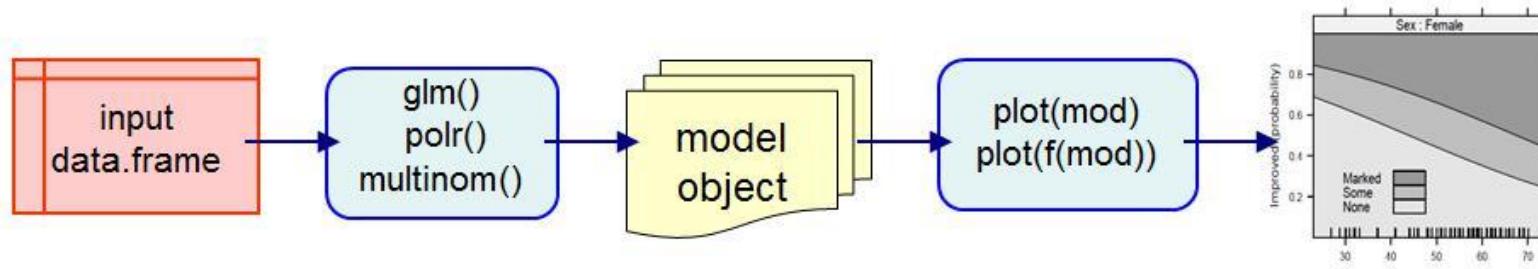
1. Overview of R graphics
2. Standard graphics in R
3. ~~Grid & lattice graphics~~
4. ggplot2

# Outline: Session 1

- Session 1: Overview of R graphics, the big picture
  - Getting started: R, R Studio, R package tools
  - Roles of graphics in data analysis
    - Exploration, analysis, presentation
  - What can I do with R graphics?
    - Anything you can think of!
    - Standard data graphs, maps, dynamic, interactive graphics
      - we'll see a sampler of these
    - R packages: many application-specific graphs
  - Reproducible analysis and reporting
    - knitr, R markdown
    - R Studio

# Outline: Session 2

- Session 2: Standard graphics in R
  - R object-oriented design



- Tweaking graphs: control graphic parameters
  - Colors, point symbols, line styles
  - Labels and titles
- Annotating graphs
  - Add fitted lines, confidence envelopes

# Outline: Session 3

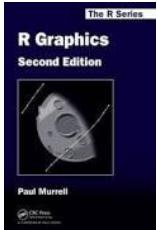
- Session 3: Grid & lattice graphics
  - Another, more powerful “graphics engine”
  - All standard plots, with more pleasing defaults
  - Easily compose collections (“small multiples”) from subsets of data
  - vcd and vcdExtra packages: mosaic plots and others for categorical data

Lecture notes for this session are available on the web page

# Outline: Session 4

- Session 4: ggplot2
  - Most powerful approach to statistical graphs, based on the “Grammar of Graphics”
  - A graphics language, composed of layers, “geoms” (points, lines, regions), each with graphical “aesthetics” (color, size, shape)
  - part of a workflow for “tidy” data manipulation and graphics

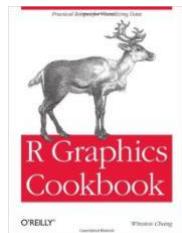
# Resources: Books



## Paul Murrell, *R Graphics*, 2nd Ed.

Covers everything: traditional (base) graphics, lattice, ggplot2, grid graphics, maps, network diagrams, ...

R code for all figures: <https://www.stat.auckland.ac.nz/~paul/RG2e/>



## Winston Chang, *R Graphics Cookbook: Practical Recipes for Visualizing Data*

Cookbook format, covering common graphing tasks; the main focus is on ggplot2

R code from book: <http://www.cookbook-r.com/Graphs/>

Download from: <http://ase.tufts.edu/bugs/guide/assets/R%20Graphics%20Cookbook.pdf>



## Deepayan Sarkar, *Lattice: Multivariate Visualization with R*

R code for all figures: <http://lmdvr.r-forge.r-project.org/>



## Hadley Wickham, *ggplot2: Elegant graphics for data analysis*, 2nd Ed.

1st Ed: Online, <http://ggplot2.org/book/>

ggplot2 Quick Reference: <http://sape.inf.usi.ch/quick-reference/ggplot2/>

Complete ggplot2 documentation: <http://docs.ggplot2.org/current/>

# Resources: cheat sheets

R Studio provides a variety of handy cheat sheets for aspects of data analysis & graphics See: <https://www.rstudio.com/resources/cheatsheets/>

The image displays nine RStudio cheat sheets arranged in a 3x3 grid:

- Data Import**: Shows functions for reading various file formats like CSV, Excel, and databases.
- Read functions**: Details for reading data from files, databases, and URLs.
- Parsing data types**: How to parse different data types including JSON, XML, and YAML.
- Data Transformation with dplyr Cheat Sheet**: A comprehensive guide to dplyr's data manipulation functions.
- Manipulate Cases**: Functions for creating, selecting, and summarizing cases.
- Manipulate Variables**: Functions for creating, selecting, and summarizing variables.
- RStudio IDE Cheat Sheet**: A detailed overview of the RStudio interface and its features.
- R Markdown Cheat Sheet**: Instructions for creating, editing, and publishing R Markdown documents.
- Data Visualization with ggplot2**: A guide to ggplot2's capabilities for creating various types of plots.

Download, laminate,  
paste them on your  
fridge

# Getting started: Tools

- To profit best from this course, you need to install both R and R Studio on your computer



The basic R system: R console (GUI) & packages

Download: <http://cran.us.r-project.org/>

**Add** my recommended packages:

source("http://datavis.ca/courses/RGraphics/R/install-pkgs.R")



The R Studio IDE: analyze, write, publish

Download:

<https://www.rstudio.com/products/rstudio/download/>

**Add**: R Studio-related packages, as useful



# R package tools



**Data prep:** Tidy data makes analysis and graphing much easier.

Packages: [tidyverse](#), comprised of: [tidyr](#), [dplyr](#), [lubridate](#), ...

## The tidyverse

### Components



**R graphics:** general frameworks for making standard and custom graphics

Graphics frameworks: base graphics, [lattice](#), [ggplot2](#), [rgl](#) (3D)

Application packages: [car](#) (linear models), [vcd](#) (categorical data analysis), [heplots](#) (multivariate linear models)



**Publish:** A variety of R packages make it easy to write and publish research reports and slide presentations in various formats (HTML, Word, LaTeX, ...), all within R Studio



**Web apps:** R now has several powerful connections to preparing dynamic, web-based data display and analysis applications.



# Getting started: R Studio

The image shows the R Studio interface with several red annotations:

- R console**: A large red box covers the left pane, which contains the R console.
- files**: A red box covers the bottom-right pane, which displays a file browser.
- plots**: A red box covers the top-right pane, which displays a command history.
- packages**: A red box covers the middle-right pane, which displays a workspace containing variables.
- help**: A red box covers the bottom-right pane, which displays help documentation.

**R Studio**

File Edit View Workspace Plots Tools Help

Console

Workspace History

Load Save Import Dataset Clear All

command history

workspace: your variables

files

plots

packages

help

Files Plots Packages Help

New Folder Delete Rename More

| Name                           | Size    | Modified               |
|--------------------------------|---------|------------------------|
| .Rhistory                      | 1.6 KB  | Jun 10, 2011, 1:59 PM  |
| 20070724_data.xls              | 13.5 KB | Jan 7, 2008, 11:51 PM  |
| AutoHotkey.ahk                 | 11.3 KB | Feb 28, 2011, 12:04 PM |
| blk                            | 10.3 KB | Apr 21, 2009, 10:00 AM |
| code                           | 23.5 KB | Jan 3, 2008, 5:35 PM   |
| counts.xls                     | 29.5 KB | Aug 5, 2008, 4:32 PM   |
| cuznsim_ss.xls                 | 2 KB    | Dec 20, 2010, 8:30 AM  |
| Default.rdp                    |         |                        |
| Digsby Log                     |         |                        |
| docs                           |         |                        |
| docs-archive                   |         |                        |
| Downloads                      |         |                        |
| eagle                          |         |                        |
| ExpressPCB                     |         |                        |
| facs-log.xls                   |         |                        |
| FlowJo75.prefs                 |         |                        |
| funding_ops_deadearly2007.txt  |         |                        |
| Geneious Backup 2011-04-27.zip |         |                        |
| Geneious Backup 2011-05-16.zip |         |                        |

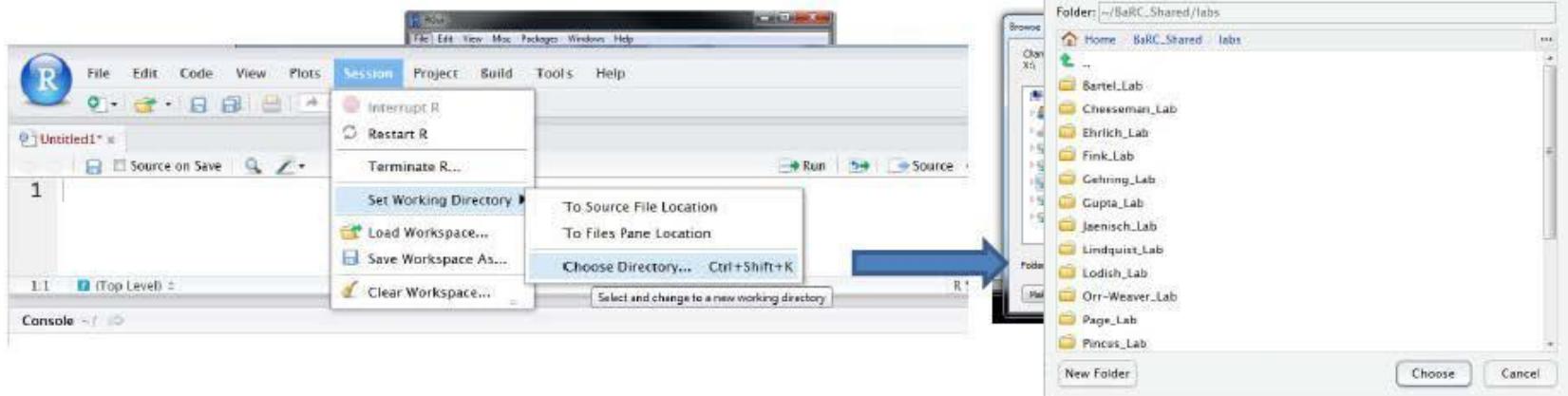
# R Studio navigation

## R folder navigation commands:

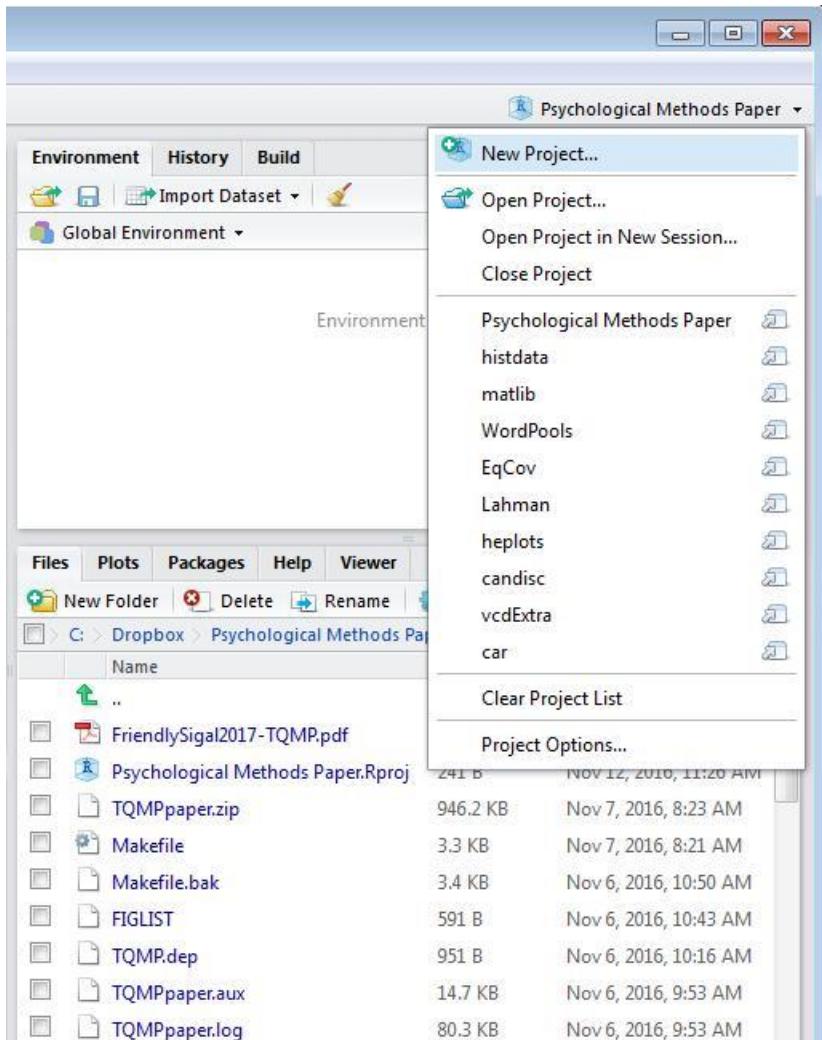
- Where am I? > `getwd()`  
[1] "C:/Dropbox/Documents/6135"
- Go somewhere:  
> `setwd("C:/Dropbox")`  
> `setwd(file.choose())`

## R Studio GUI

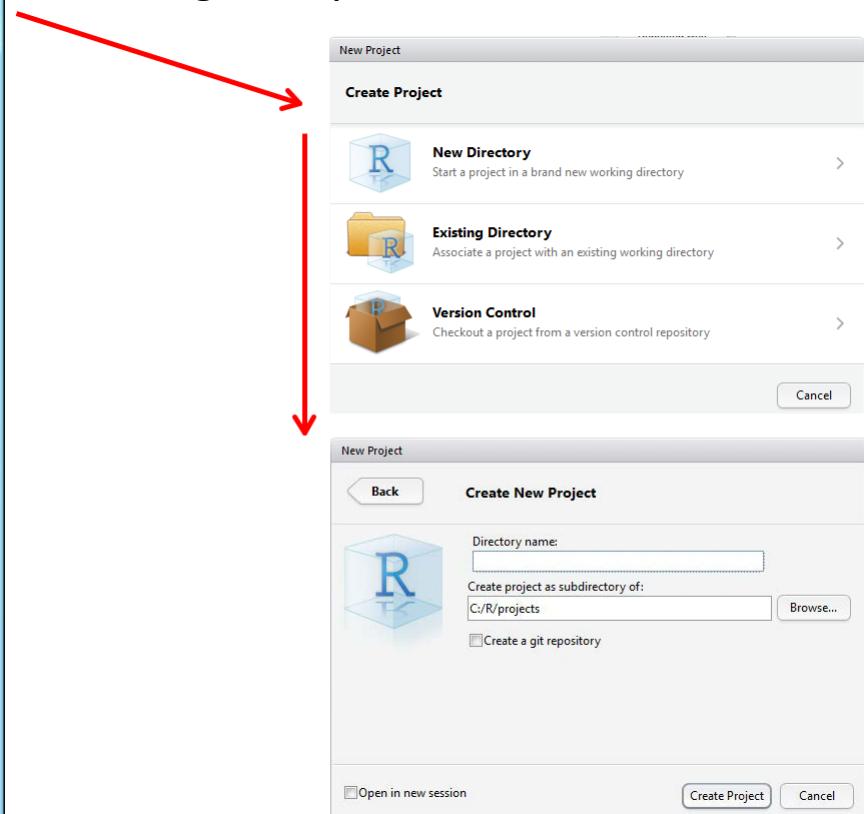
- Take R to your preferred directory ()



# R Studio projects



R Studio projects are a handy way  
to organize your work



# R Studio projects

An R Studio project for a research paper: R files (scripts), Rmd files (text, R “chunks”)

The screenshot shows the R Studio interface with a project titled "Psychological Methods Paper".

**File Explorer:** Shows the project structure under "C:/Dropbox/Psychological Methods Paper".

| Name                              | Type    | Date        | Size     |
|-----------------------------------|---------|-------------|----------|
| FriendlySigal2017-TQMP.pdf        | PDF     | Nov 7, 2016 | 8:23 AM  |
| Psychological Methods Paper.Rproj | Project | Nov 7, 2016 | 8:21 AM  |
| TQMPpaper.zip                     | Zip     | Nov 7, 2016 | 3.3 KB   |
| Makefile                          | Text    | Nov 6, 2016 | 10:50 AM |
| Makefile.bak                      | Text    | Nov 6, 2016 | 3.4 KB   |
| FIGLIST                           | Text    | Nov 6, 2016 | 10:43 AM |
| TQMP.dep                          | Text    | Nov 6, 2016 | 10:16 AM |
| TQMPpaper.aux                     | Text    | Nov 6, 2016 | 9:53 AM  |
| TQMPpaper.log                     | Text    | Nov 6, 2016 | 9:53 AM  |
| TQMPpaper.out                     | Text    | Nov 6, 2016 | 9:53 AM  |
| TQMPpaper.pdf                     | PDF     | Nov 6, 2016 | 819.8 KB |
| TQMPpaper.synctex.gz              | Text    | Nov 6, 2016 | 286.1 KB |
| .Rhistory                         | Text    | Nov 3, 2016 | 13.5 KB  |

**Code Editor:** Displays an Rmd file (TQMPpaper.Rmd) containing the following code:

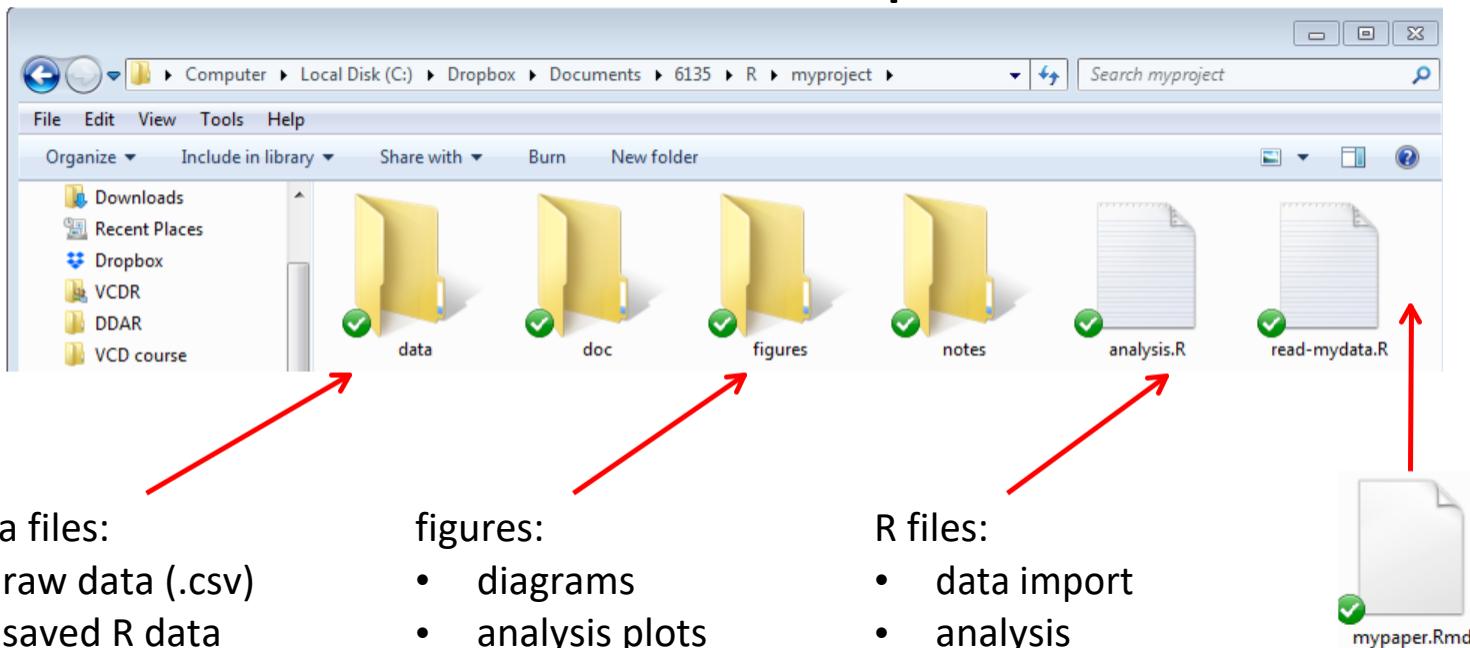
```
1: ---  
2: title: "Graphical Methods for Multivariate Linear Models in  
3: Psychological Research: An R Tutorial"  
4: shorttitle: "Graphical Methods for MLMs"  
5: author:  
6: - name: Michael Friendly  
7:   affiliation: 1  
8:   corresponding: yes # Define only one corresponding author  
9:   address: Psychology Department, York University, Toronto, Ontario,  
10:  Canada, M3J1P3  
11:  email: friendly@yorku.ca  
12:  - name: Matthew Sigal  
13:    affiliation: 1  
14:    institution: York University  
15:  
16: abstract: |  
17:   This paper is designed as a tutorial to highlight some  
18:   recent developments for visualizing the relationships among  
19:   response and predictor variables in multivariate linear  
20:  
21: Graphical Methods for Multivariate Linear Models in Psychological Research: An R Tutorial
```

**Console:** Displays the R startup message and information about the R version and platform.

**Help:** A sidebar menu provides access to various R documentation and tools.

# Organizing an R project

- Use a separate folder for each project
- Use sub-folders for various parts



**data files:**

- raw data (.csv)
- saved R data (.Rdata)

**figures:**

- diagrams
- analysis plots

**R files:**

- data import
- analysis

Write up files will go here (.Rmd, .docx, .pdf)

# Organizing an R project

- Use separate R files for different steps:
  - Data import, data cleaning, ... → save as an RData file
  - Analysis: load RData, ...

read-mydata.R

```
# read the data; better yet: use RStudio File -> Import Dataset ...
mydata <- read.csv("data/mydata.csv")
```

```
# data cleaning ....
```

```
# save the current state
save("data/mydata.RData")
```

# Organizing an R project

- Use separate R files for different steps:
  - Data import, data cleaning, ... → save as an RData file
  - Analysis: load RData, ...

analyse.R

```
# analysis
load("data/mydata.RData")

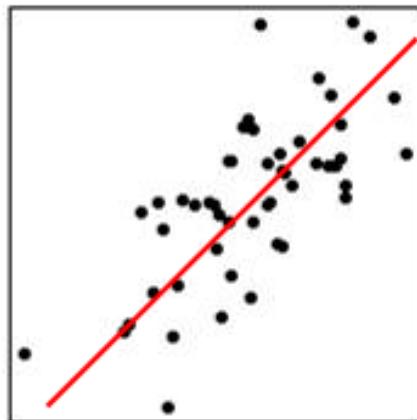
# do the analysis – exploratory
plots plot(mydata)

# fit models
mymod.1 <- lm(y ~ X1 + X2 + X3, data=mydata)

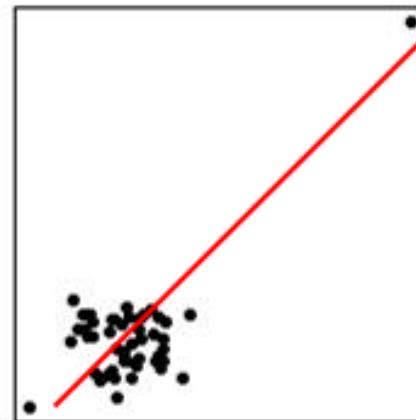
# plot models, extract model
summaries plot(mymod.1)
summary(mymod.1)
```

# Graphics: Why plot your data?

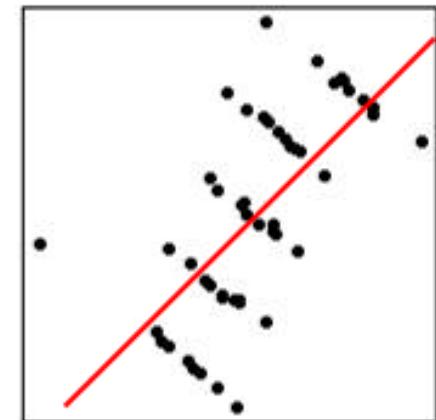
- Three data sets with exactly the same bivariate summary statistics:
  - Same correlations, linear regression lines, etc
  - Indistinguishable from standard printed output



Standard data



$r=0$  but + 2 outliers



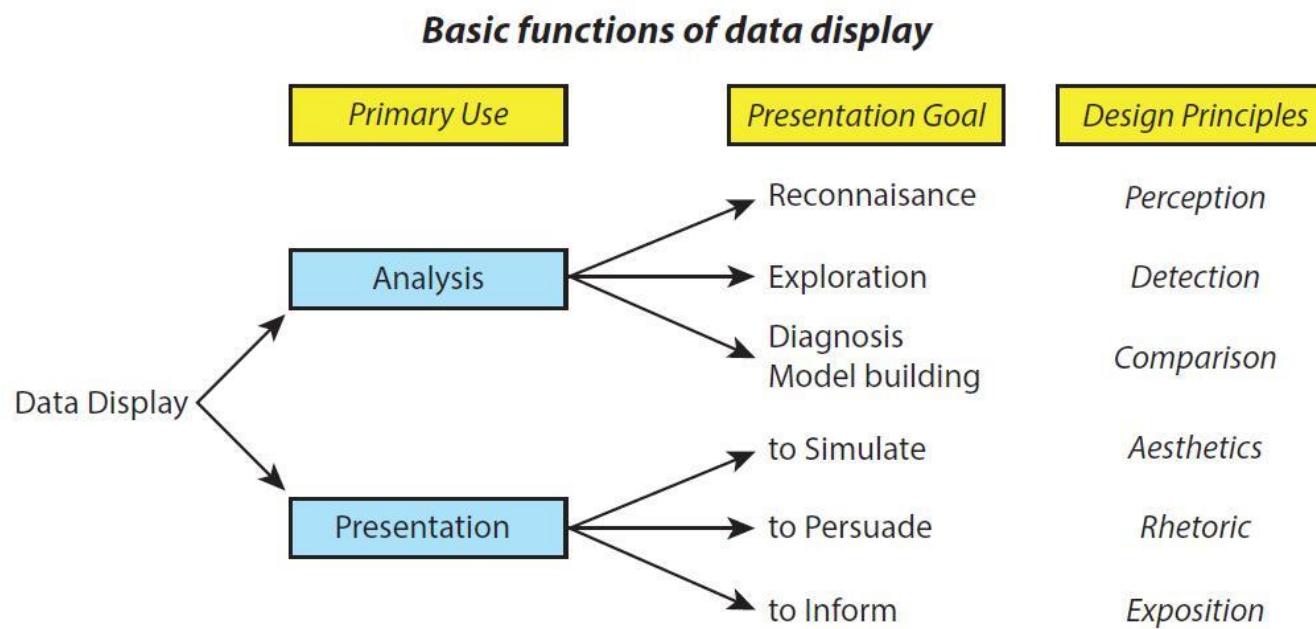
Lurking variable?

# Roles of graphics in data analysis

- Graphs (& tables) are forms of communication:
  - What is the audience?
  - What is the message?

**Analysis graphs:** design to see patterns, trends, aid the process of data description, interpretation

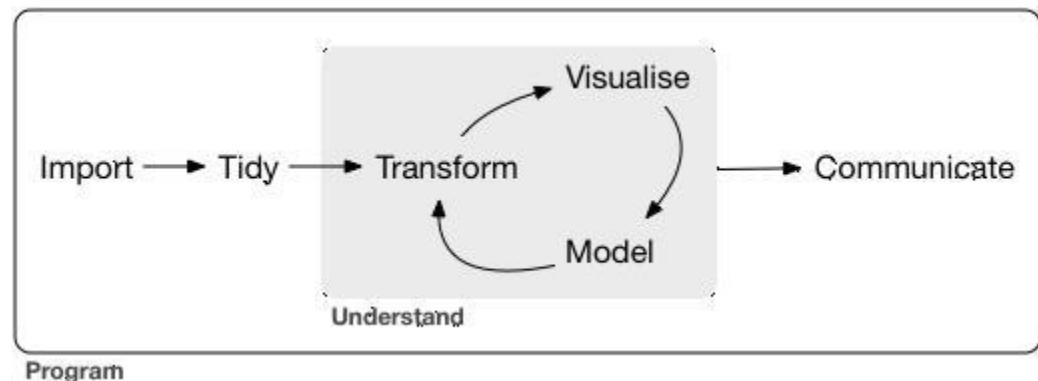
**Presentation graphs:** design to attract attention, make a point, illustrate a conclusion



# The 80-20 rule: Data analysis

- Often ~80% of data analysis time is spent on data preparation and data cleaning
  1. data entry, importing data set to R, assigning factor labels,
  2. data screening: checking for errors, outliers, ...
  3. Fitting models & diagnostics: whoops! Something wrong, go back to step 1
- Whatever you can do to reduce this, gives more time for:
  - Thoughtful analysis,
  - Comparing models,
  - Insightful graphics,
  - Telling the story of your results and conclusions

This view of data analysis, statistics and data vis is now rebranded as “data science”

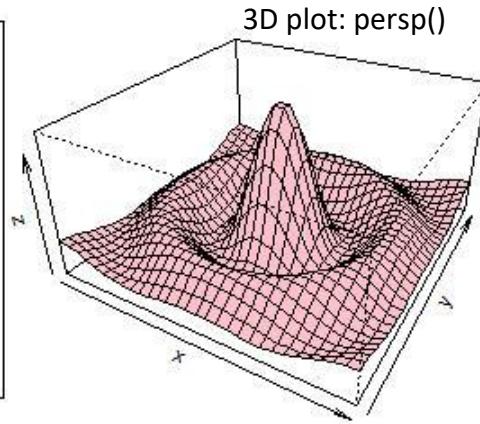
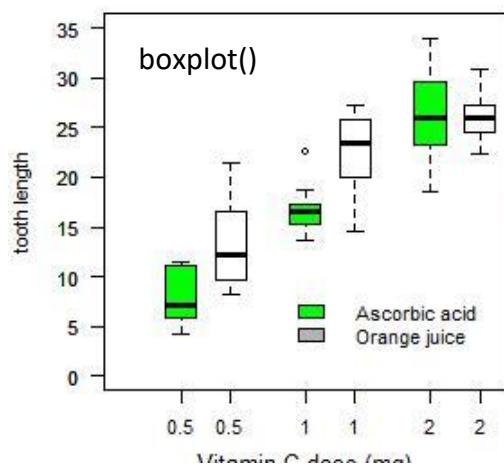
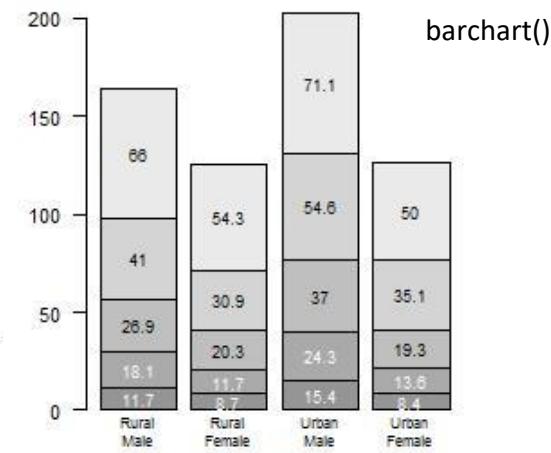
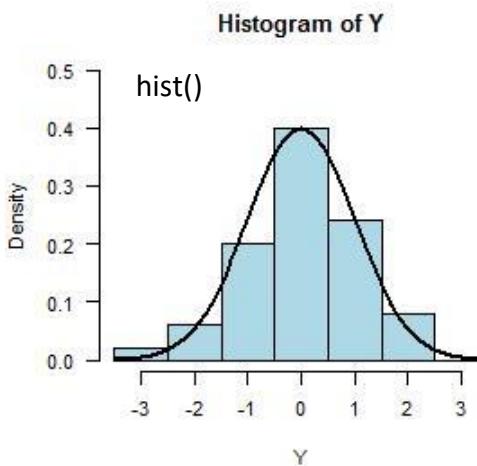
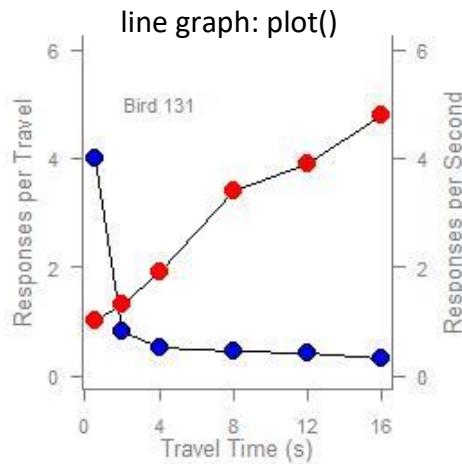


# The 80-20 rule: Graphics

- **Analysis graphs:** Happily, 20% of effort can give 80% of a desired result
  - Default settings for plots often give something reasonable
  - 90-10 rule: Plot annotations (regression lines, smoothed curves, data ellipses, ...) add additional information to help understand patterns, trends and unusual features, with only 10% more effort
- **Presentation graphs:** Sadly, 80% of total effort may be required to give the remaining 20% of your final graph
  - Graph title, axis and value labels: should be directly readable
  - Grouping attributes: visually distinct, allowing for BW vs color
    - color, shape, size of point symbols;
    - color, line style, line width of lines
  - Legends: Connect the data in the graph to interpretation
  - Aspect ratio: need to consider the H x V size and shape

# What can I do with R graphics?

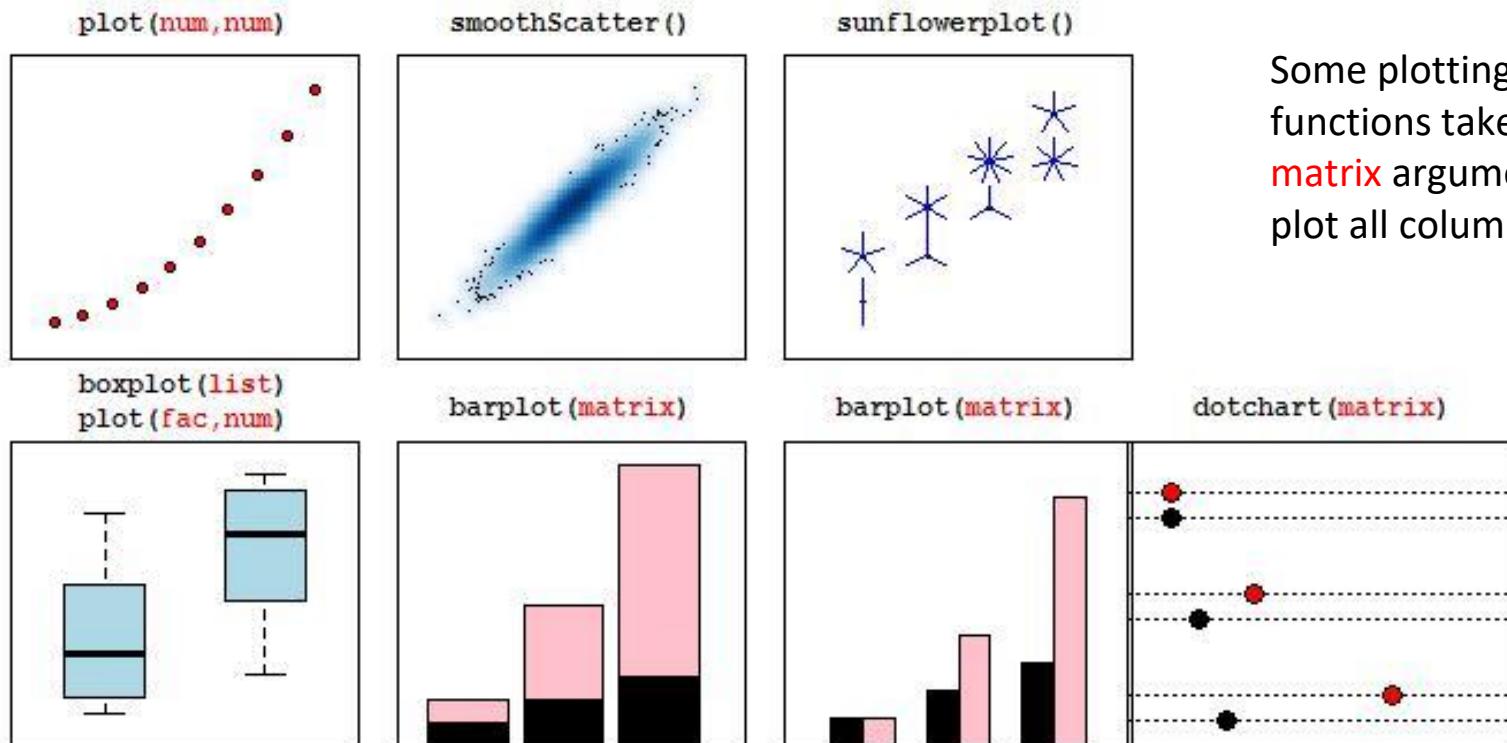
A wide variety of standard plots (customized)



# Bivariate plots

R base graphics provide a wide variety of different plot types for bivariate data

The function `plot(x, y)` is generic. It produces different kinds of plots depending on whether x and y are **numeric** or **factors**.



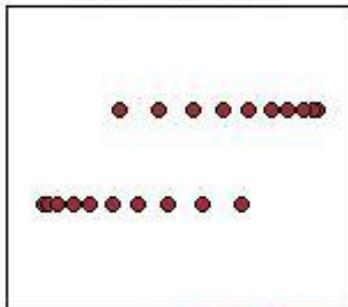
Some plotting functions take a **matrix** argument & plot all columns

# Bivariate plots

A number of specialized plot types are also available in base R graphics

Plot methods for **factors** and **tables** are designed to show the association between categorical variables

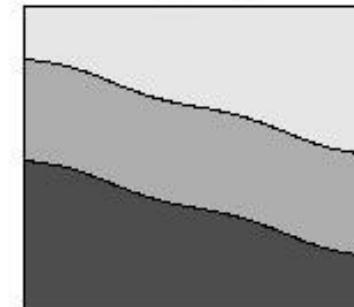
`stripchart(list  
plot(num,fac))`



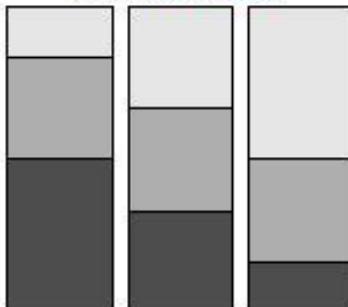
`spineplot(num,fac)`



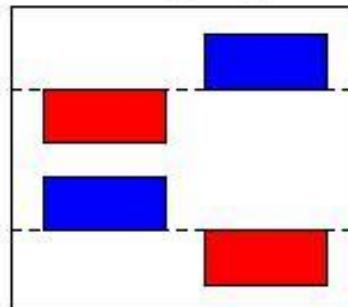
`cdplot()`



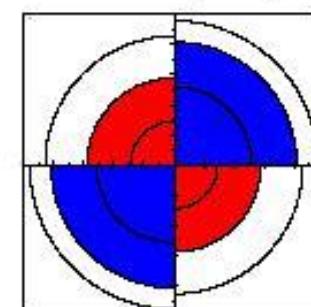
`spineplot(fac,fac  
plot(fac,fac))`



`assocplot()`

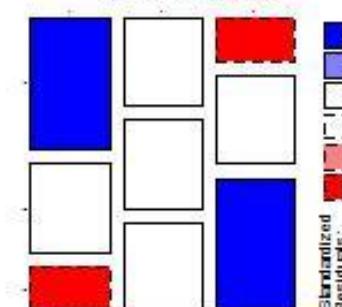


`fourfoldplot()`



The **vcd** & **vcdExtra** packages provide more and better plots for categorical data

`mosaicplot()  
plot(table)`



Standardized Residuals:

# Mosaic plots

Similar to a grouped bar chart

Shows a frequency table with tiles,  
area ~ frequency

```
> data(HairEyeColor)
> HEC <- margin.table(HairEyeColor, 1:2)
> HEC
```

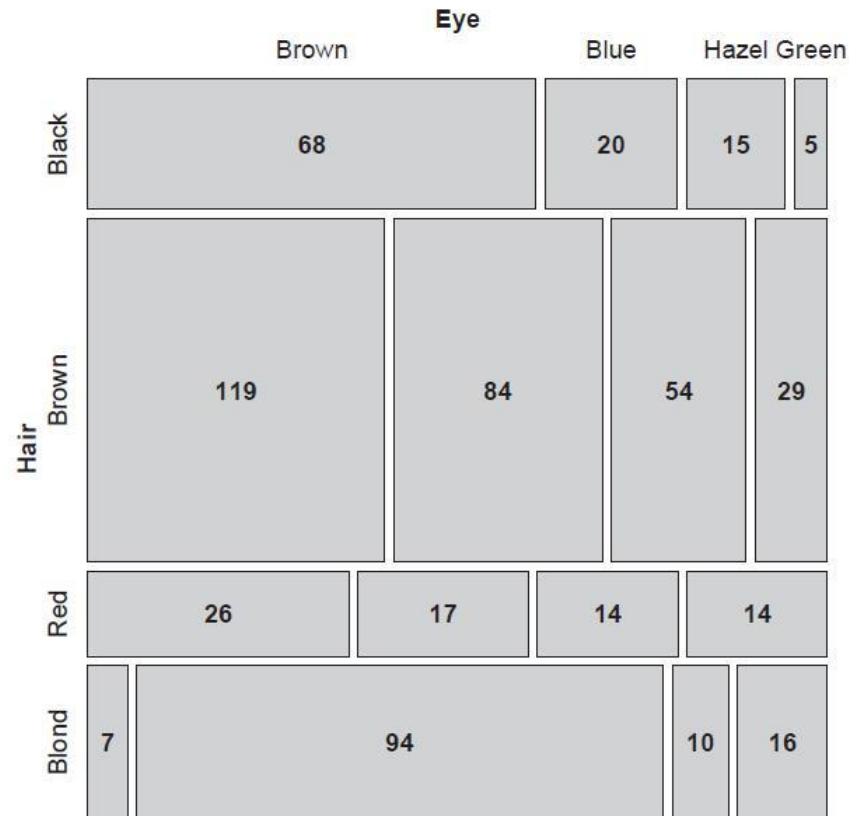
| Hair  | Eye   |      |       |       |
|-------|-------|------|-------|-------|
|       | Brown | Blue | Hazel | Green |
| Black | 68    | 20   | 15    | 5     |
| Brown | 119   | 84   | 54    | 29    |
| Red   | 26    | 17   | 14    | 14    |
| Blond | 7     | 94   | 10    | 16    |

```
> chisq.test(HEC)
```

Pearson's Chi-squared test

```
data: HEC
```

```
X-squared = 140, df = 9, p-value <2e-16
```



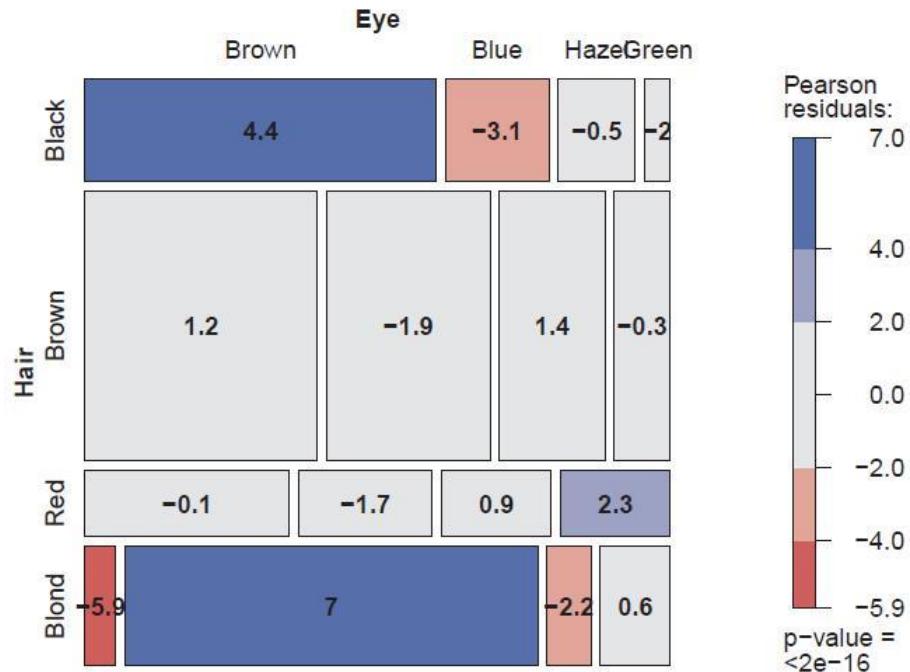
How to understand the association  
between hair color and eye color?

# Mosaic plots

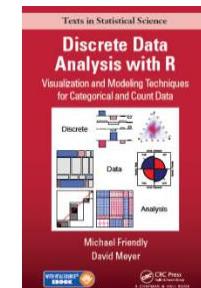
Shade each tile in relation to the contribution to the Pearson  $\chi^2$  statistic

$$\chi^2 = \sum r_{ij} = \sum \frac{(o_{ij} - e_{ij})^2}{e_{ij}}$$

```
> round(residuals(chisq.test(HEC)), 2)
   Eye
Hair    Brown Blue Hazel Green
Black   4.40 -3.07 -0.48 -1.95
Brown   1.23 -1.95  1.35 -0.35
Red     -0.07 -1.73  0.85  2.28
Blond  -5.85  7.05 -2.23  0.61
```



Mosaic plots extend readily to 3-way + tables  
They are intimately connected with loglinear models  
See: Friendly & Meyer (2016), Discrete Data Analysis with R, <http://ddar.datavis.ca/>

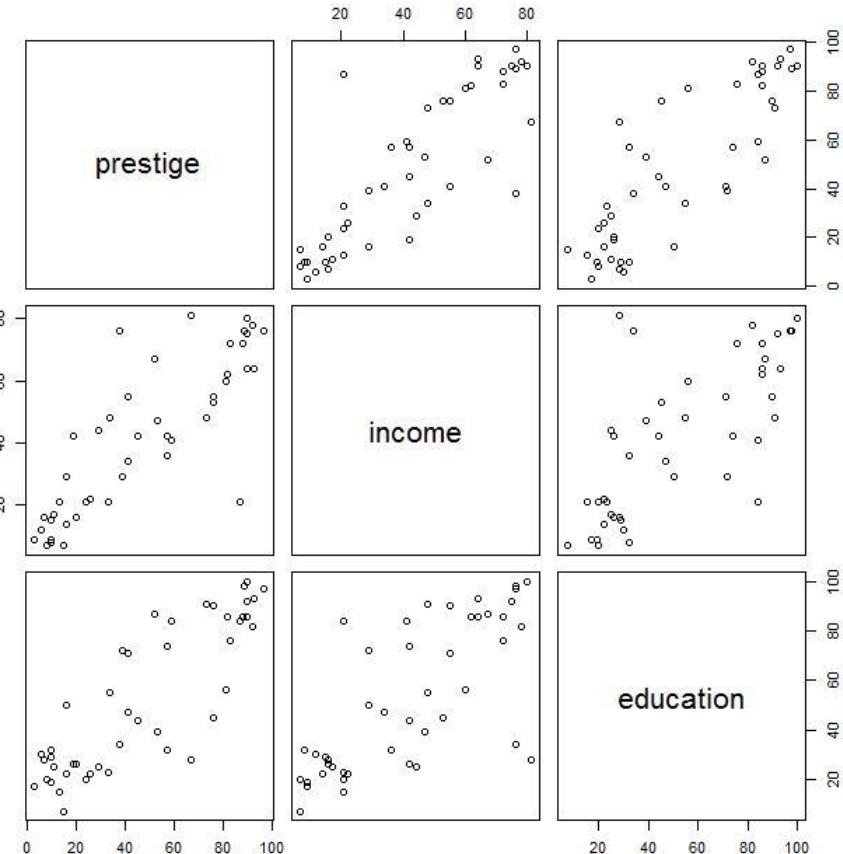


# Multivariate plots

The simplest case of multivariate plots is a **scatterplot matrix** – all pairs of bivariate plots

In R, the generic functions **plot()** and **pairs()** have specific methods for data frames

```
data(Duncan, package="car")
plot(~ prestige + income + education,
     data=Duncan)
pairs(~ prestige + income + education,
      data=Duncan)
```

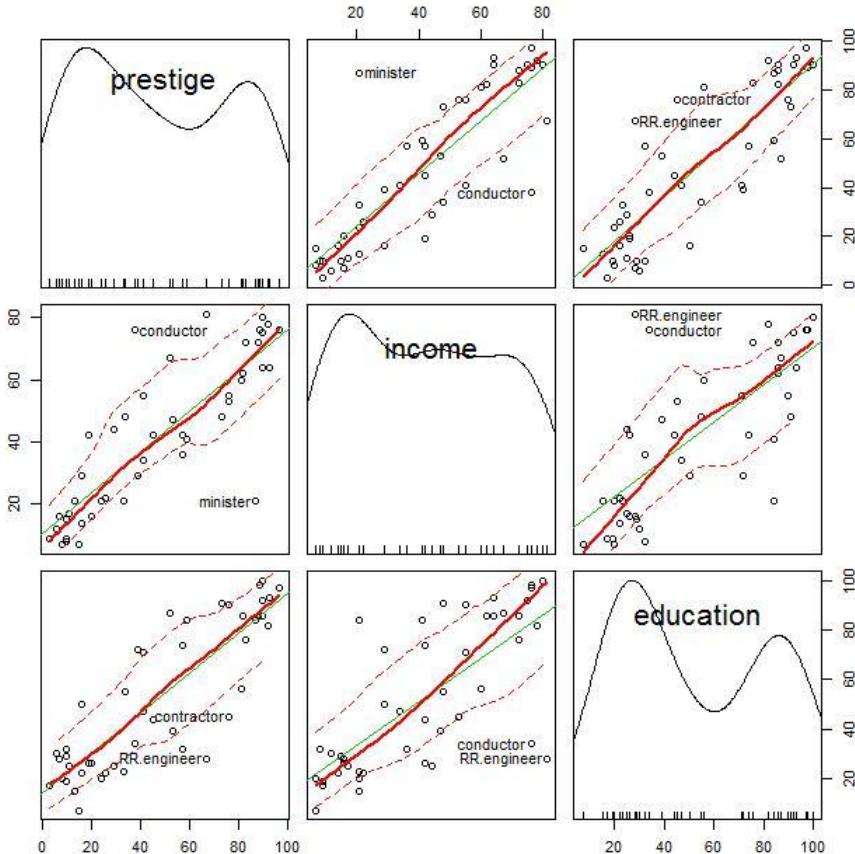


# Multivariate plots

These basic plots can be enhanced in many ways to be more informative.

The function `scatterplotMatrix()` in the `car` package provides

- univariate plots for each variable
- linear **regression lines** and loess **smoothed curves** for each pair
- automatic labeling of noteworthy observations (`id.n=`)



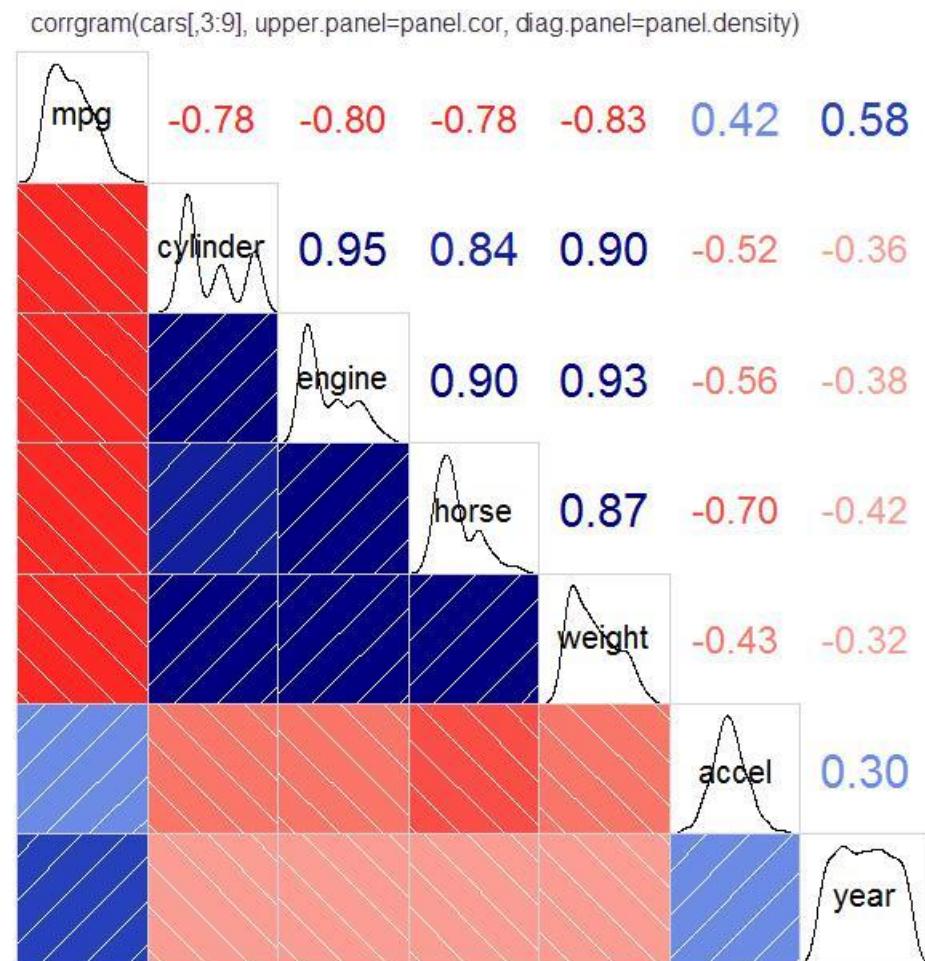
```
library(car)
scatterplotMatrix(~prestige + income + education, data=Duncan, id.n=2)
```

# Multivariate plots: corrgrams

For larger data sets, visual summaries are often more useful than direct plots of the raw data

A corrgram (“correlation diagram”) allows the data to be rendered in a variety of ways, specified by panel functions.

Here the main goal is to see how mpg is related to the other variables



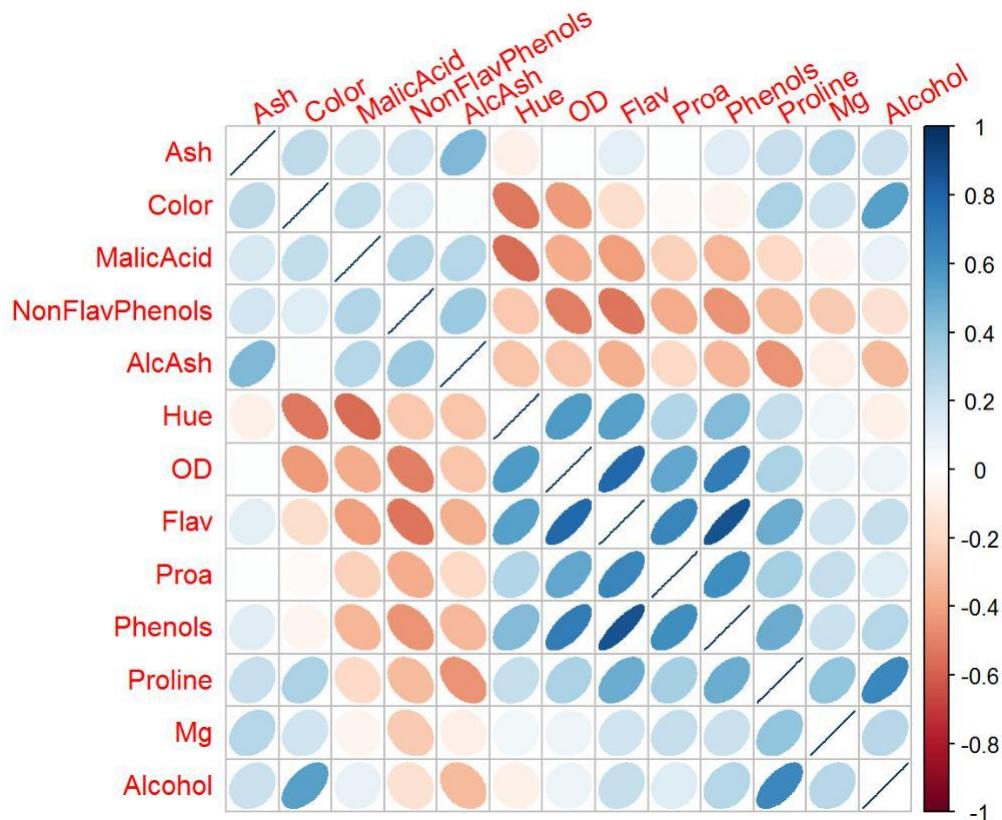
See: Friendly, M. Corrgrams: Exploratory displays for correlation matrices. *The American Statistician*, 2002, 56, 316-324

# Multivariate plots: corrgrams

For even larger data sets, more abstract visual summaries are necessary to see the patterns of relationships.

This example uses schematic ellipses to show the strength and direction of correlations among variables on a large collection of Italian wines.

Here the main goal is to see how the variables are related to each other.



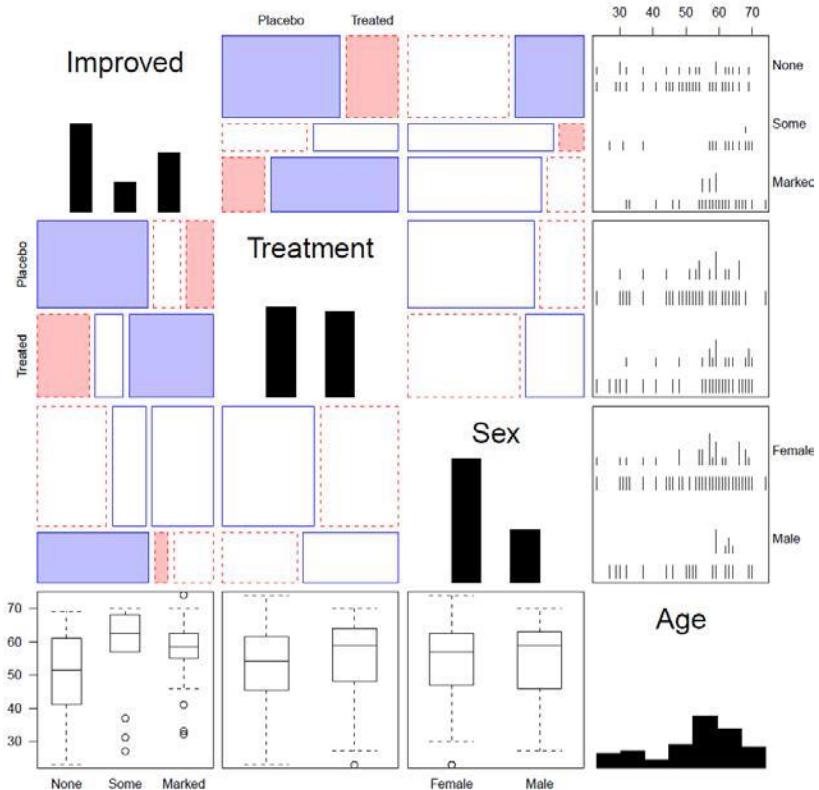
```
library(corrplot)
corrplot(cor(wine), tl.srt=30, method="ellipse", order="AOE")
```

See: Friendly, M. Corrgrams: Exploratory displays for correlation matrices. *The American Statistician*, 2002, 56, 316-324

# Generalized pairs plots

Generalized pairs plots from the `gpairs` package handle both categorical (**C**) and quantitative (**Q**) variables in sensible ways

| x | y | plot        |
|---|---|-------------|
| Q | Q | scatterplot |
| C | Q | boxplot     |
| Q | C | barcode     |
| C | C | mosaic      |



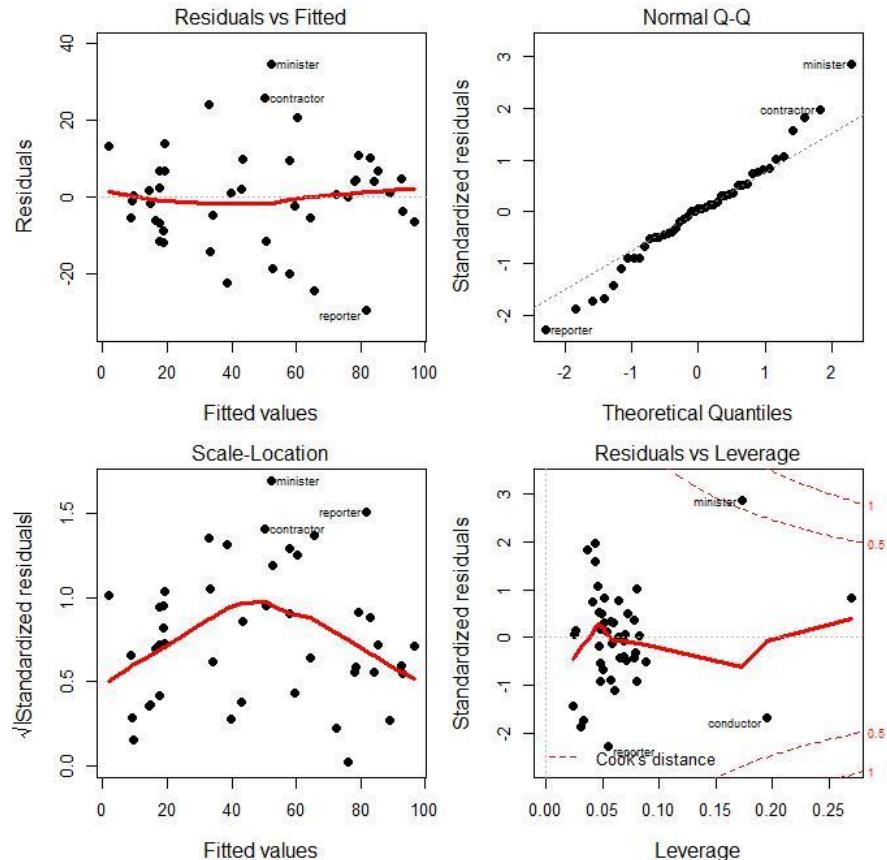
```
library(gpairs)
data(Arthritis)
gpairs(Arthritis[, c(5, 2:5)], ...)
```

# Models: diagnostic plots

Linear statistical models (ANOVA, regression),  $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$ , require some assumptions:  $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \sigma^2)$

For a fitted model object, the **plot()** method gives some useful diagnostic plots:

- residuals vs. fitted: any pattern?
- Normal QQ: are residuals normal?
- scale-location: constant variance?
- residual-leverage: outliers?

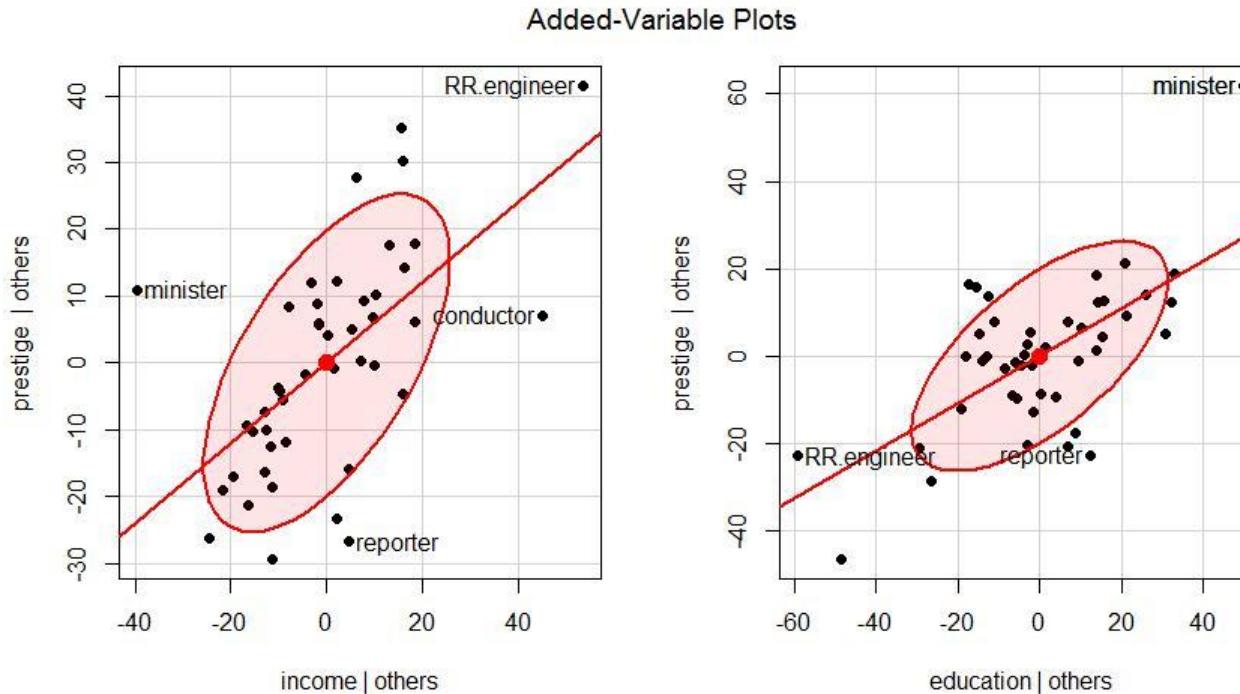


```
duncan.mod <- lm(prestige ~ income + education, data=Duncan)
plot(duncan.mod)
```

# Models: Added variable plots

The `car` package has many more functions for plotting linear model objects  
Among these, added variable plots show the partial relations of  $y$  to each  $x$ , holding **all other predictors constant**.

```
library(car)  
avPlots(duncan.mod, id.n=2, ellipse=TRUE, ...)
```



Each plot shows:  
partial slope,  $\beta_j$   
influential obs.

# Models: Interpretation

Fitted models are often difficult to interpret from tables of coefficients

```
# add term for type of job  
duncan.mod1 <- update(duncan.mod, . ~ . + type)  
summary(duncan.mod1)
```

Call:

```
lm(formula = prestige ~ income + education + type, data = Duncan)
```

Coefficients:

|             | Estimate  | Std. Error | t value | Pr(> t )     |
|-------------|-----------|------------|---------|--------------|
| (Intercept) | -0.18503  | 3.71377    | -0.050  | 0.96051      |
| income      | 0.59755   | 0.08936    | 6.687   | 5.12e-08 *** |
| education   | 0.34532   | 0.11361    | 3.040   | 0.00416 **   |
| typeprof    | 16.65751  | 6.99301    | 2.382   | 0.02206 *    |
| typewc      | -14.66113 | 6.10877    | -2.400  | 0.02114 *    |

---

Signif. codes: 0 '\*\*\*\*' 0.001 '\*\*\*' 0.01 '\*\*' 0.05 '\*' 0.1 '.' 1

Residual standard error: 9.744 on 40 degrees of freedom

Multiple R-squared: 0.9131, Adjusted R-squared: 0.9044

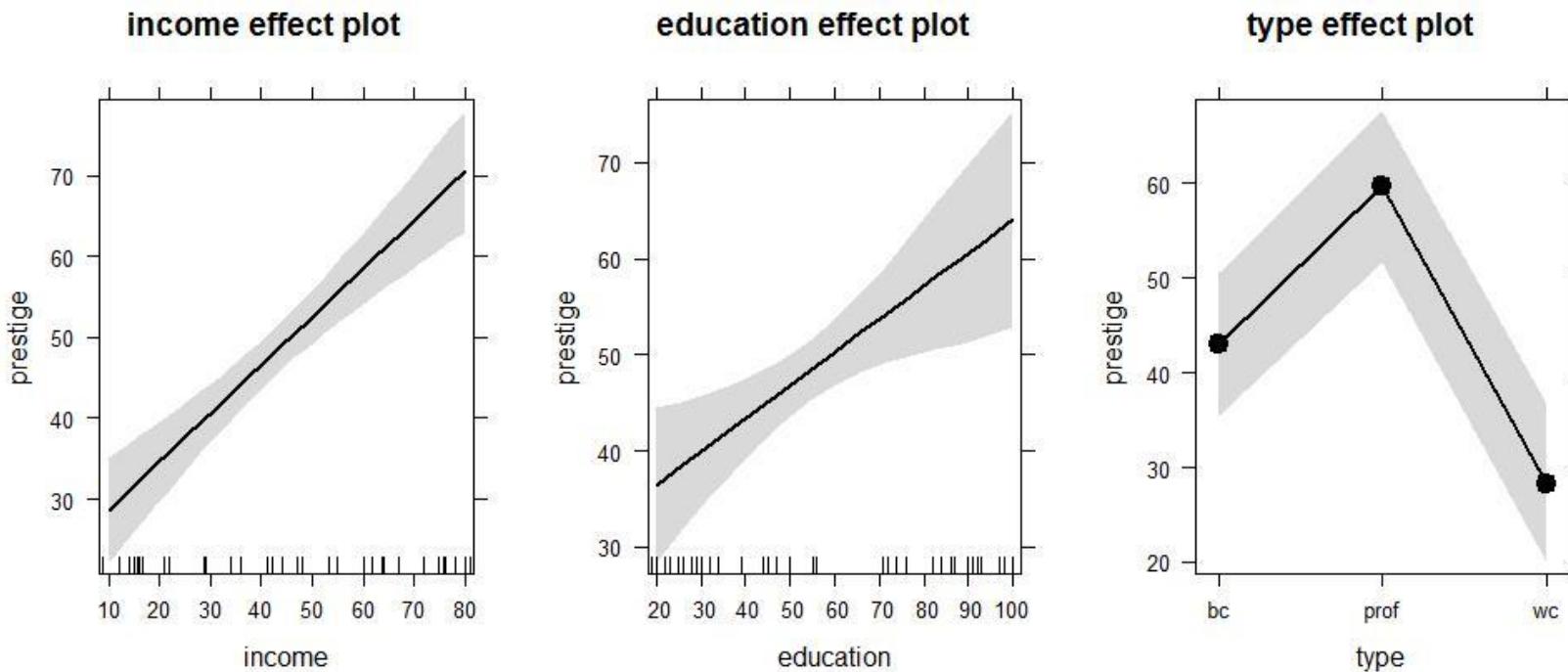
F-statistic: 105 on 4 and 40 DF, p-value: < 2.2e-16

How to  
understand effect  
of each predictor?

# Models: Effect plots

Fitted models are more easily interpreted by plotting the predicted values. Effect plots do this nicely, making plots for each **high-order term**, controlling for others

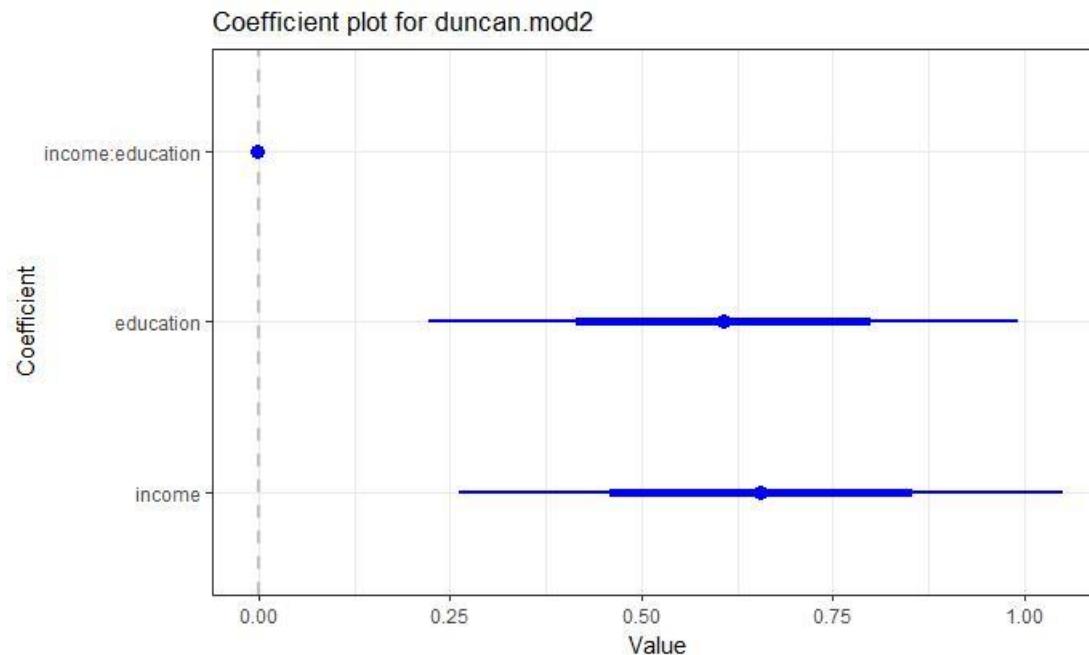
```
library(effects)
duncan.eff1 <- allEffects(duncan.mod1)
plot(duncan.eff1)
```



# Models: Coefficient plots

Sometimes you need to report or display the coefficients from a fitted model.  
A plot of coefficients with CIs is sometimes more effective than a table.

```
library(coefplot)
duncan.mod2 <- lm(prestige ~ income * education, data=Duncan)
coefplot(duncan.mod2, intercept=FALSE, lwdInner=2, lwdOuter=1,
           title="Coefficient plot for duncan.mod2")
```



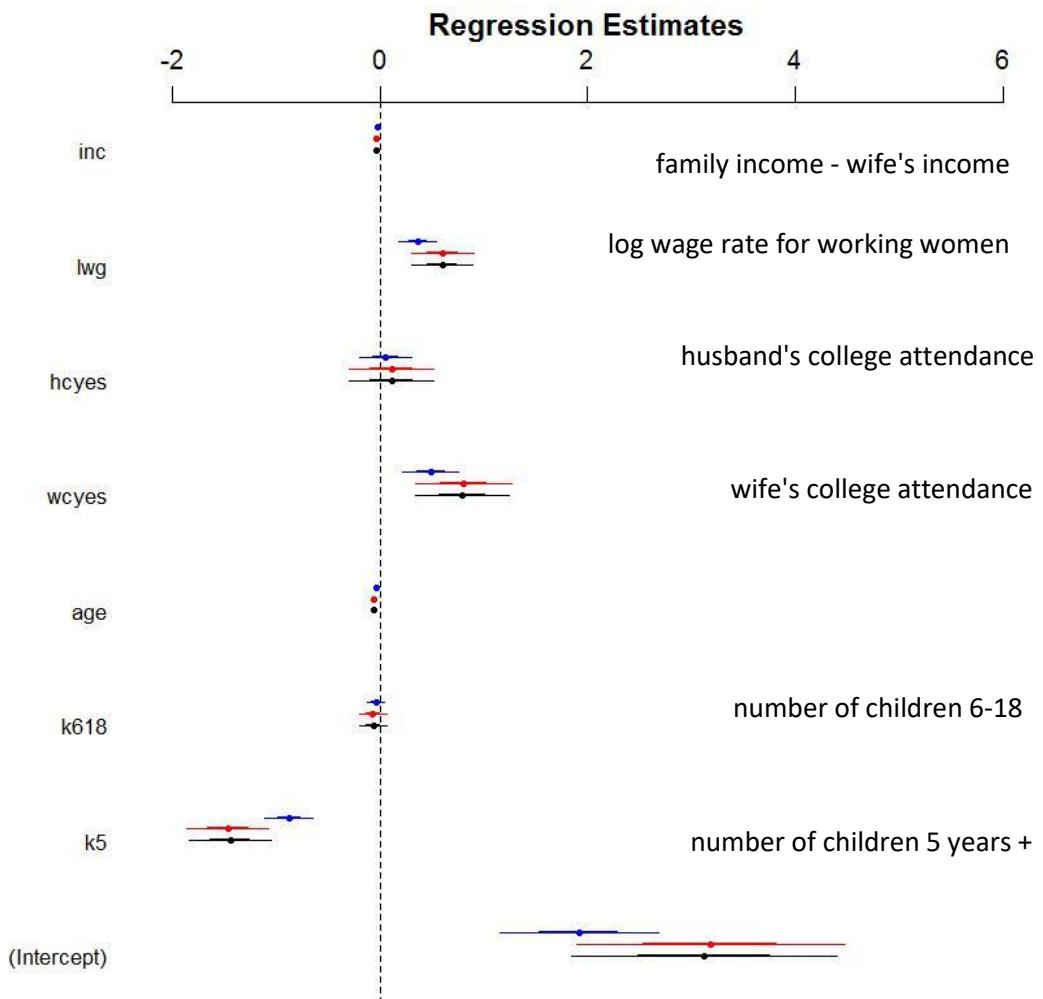
Coefficient plots become increasingly useful as:

- (a) models become more complex
- (b) we have several models to compare

This plot compares three different models for women's labor force participation fit to data from Mroz (1987) in the car package

This makes it relatively easy to see

- (a) which terms are important
- (b) how models differ



# 3D graphics

R has a wide variety of features and packages that support 3D graphics

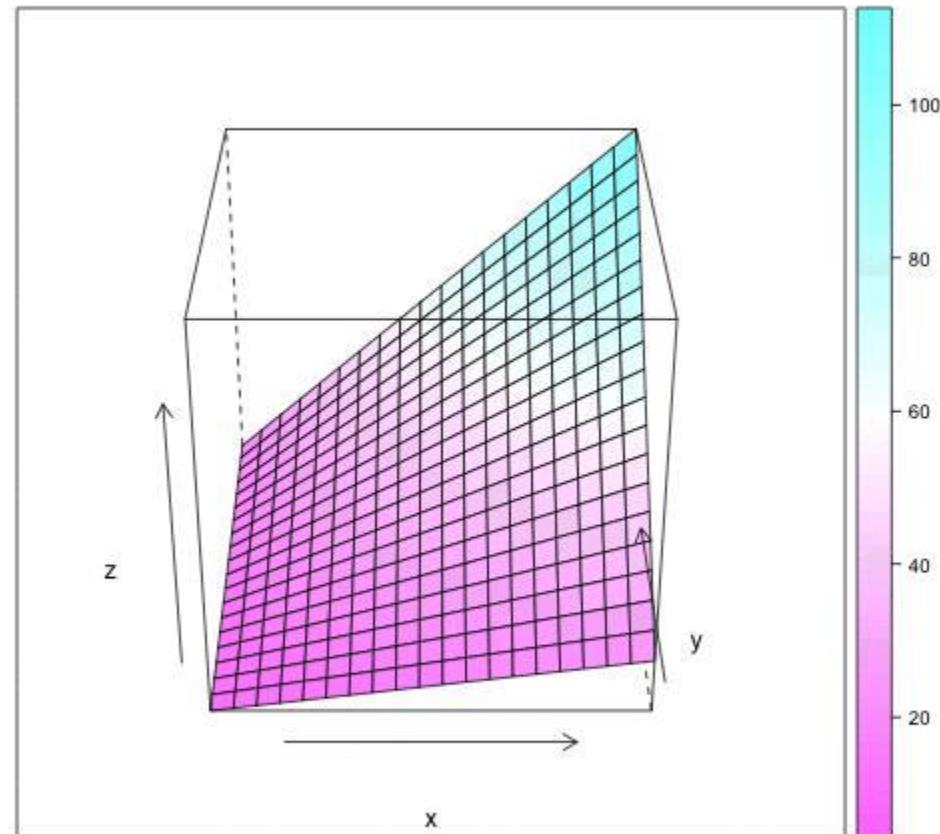
This example illustrates the concept of an [interaction](#) between predictors in a linear regression model

It uses:

`lattice::wireframe(z ~ x + y, ...)`

The basic plot is “printed” 36 times rotated  $10^{\circ}$  about the z axis to produce 36 PNG images.

The ImageMagick utility is used to convert these to an animated GIF graphic



$$z = 10 + .5x + .3y + .2 x*y$$

# 3D graphics: code

## 1. Generate data for the model $z = 10 + .5x + .3y + .2 x*y$

```
b0 <- 10      # intercept
b1 <- .5       # x coefficient
b2 <- .3       # y coefficient
int12 <- .2    # x*y coefficient
g <- expand.grid(x = 1:20, y = 1:20)
g$z <- b0 + b1*g$x + b2*g$y + int12*g$x*g$y
```

## 2. Make one 3D plot

```
library(lattice)
wireframe(z ~ x * y, data = g)
```

## 3. Create a set of PNG images, rotating around the z axis

```
png(file="example%03d.png", width=480, height=480)
for (i in seq(0, 350 ,10)){
  print(wireframe(z ~ x * y, data = g,
                  screen = list(z = i, x = -60), drape=TRUE) )
}
dev.off()
```

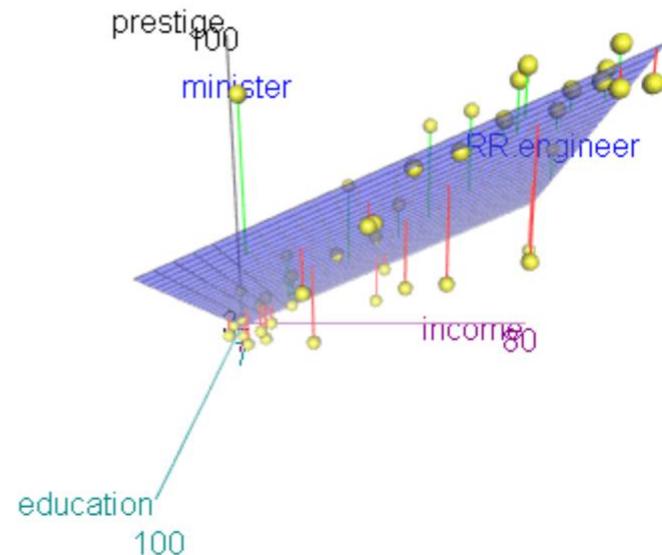
## 4. Convert PNGs to GIF using ImageMagik

```
system("convert -delay 40 example*.png animated_3D_plot.gif")
```

# 3D graphics

The `rgl` package is the most general for drawing 3D graphs in R.  
Other R packages use this for 3D statistical graphs

This example uses `car::scatter3d()` to show the data and fitted response surface for the multiple regression model for the Duncan data



```
scatter3d(prestige ~ income + education,  
          data=Duncan, id.n=2, revolutions=2)
```

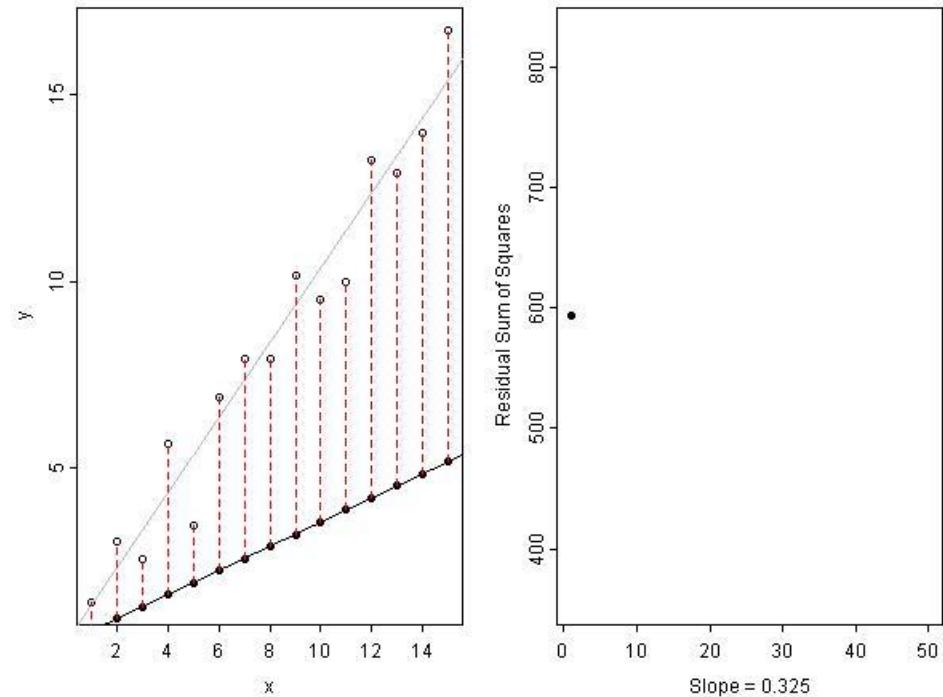
# Statistical animations

Statistical concepts can often be illustrated in a dynamic plot of some process.

This example illustrates the idea of least squares fitting of a regression line.

As the slope of the line is varied, the right panel shows the residual sum of squares.

This plot was done using the [animate](#) package

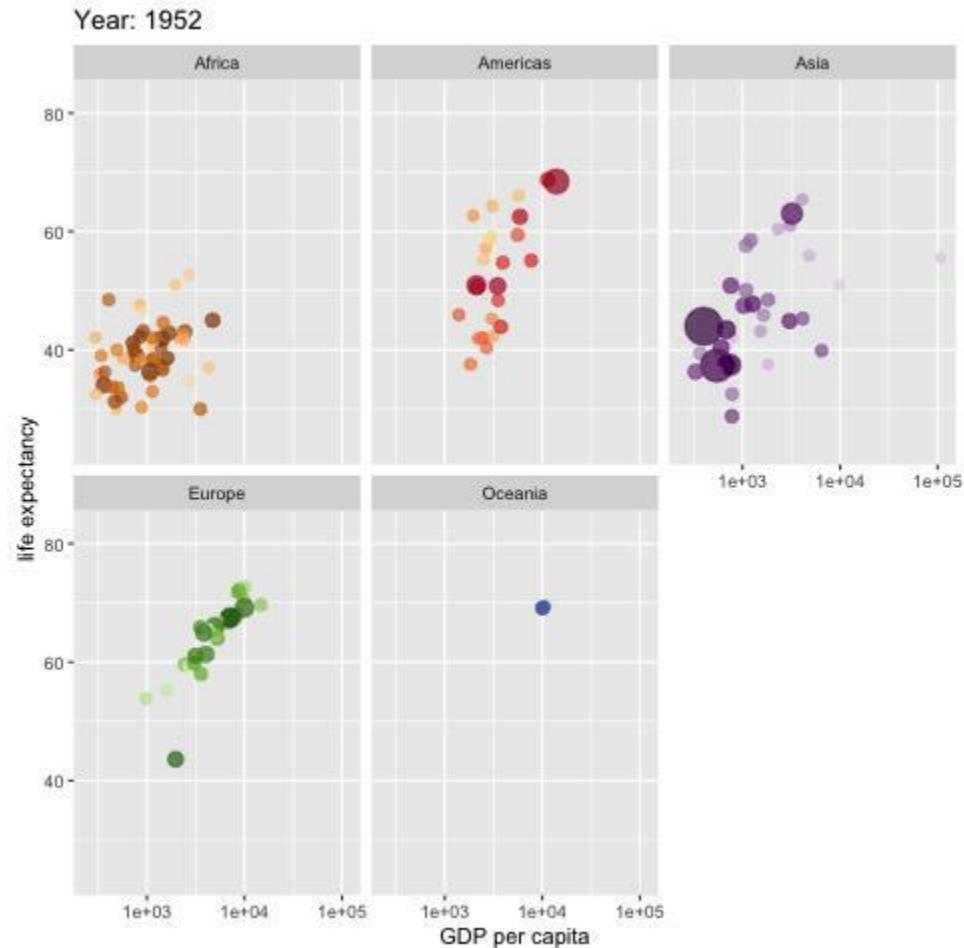


# Data animations

Time-series data are often plotted against time on an X axis.

Complex relations over time can often be made simpler by animating change – liberating the X axis to show something else

This example from the [tweenr](#) package (using [gganimate](#))



See: <https://github.com/thomasp85/tweenr> for some simple examples

# Maps and spatial visualizations

Spatial visualization in R, combines map data sets, statistical models for spatial data, and a growing number of R packages for map-based display

This example, from Paul Murrell's *R Graphics* book shows a basic map of Brazil, with provinces and their capitals, shaded by region of the country.

Data-based maps can show spatial variation of some variable of interest



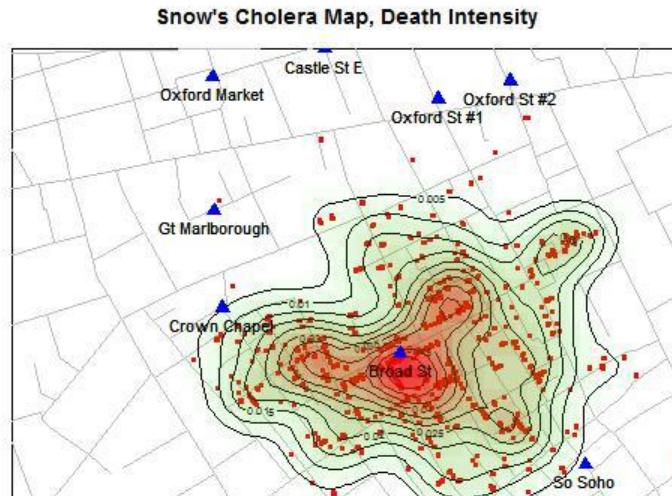
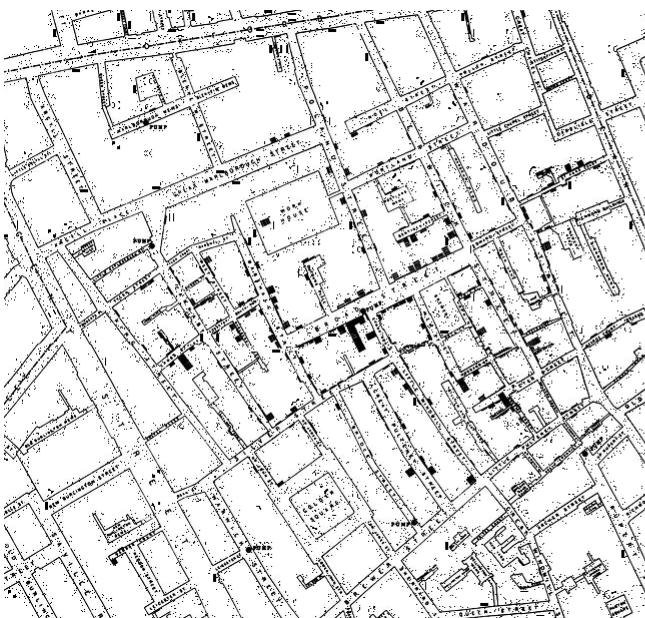
Murrell, Fig. 14.5

# Maps and spatial visualizations

Dr. John Snow's map of cholera in London, 1854

Enhanced in R in the [HistData](#) package to make Snow's point

Portion of Snow's map:



```
library(HistData)
SnowMap(density=TRUE,
        main="Snow's Cholera Map, Death Intensity")
```

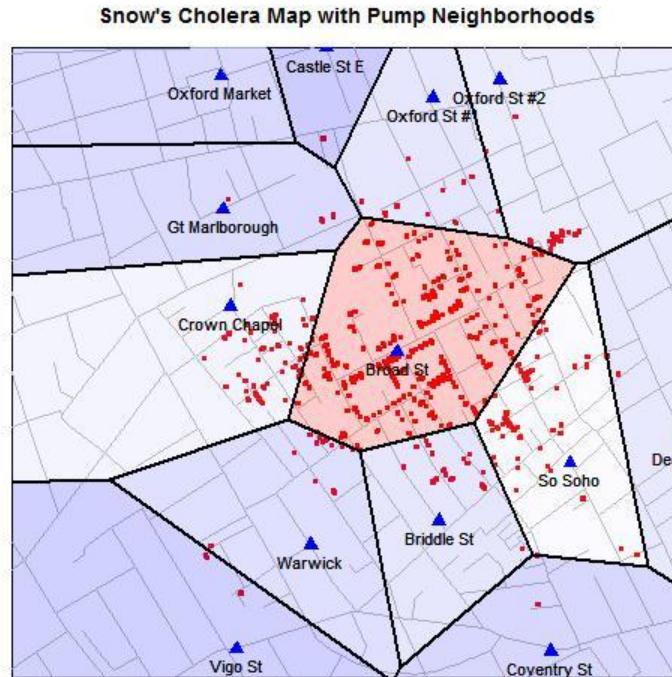
Contours of death densities are calculated using a 2d binned kernel density estimate, [bkde2D\(\)](#) from the [KernSmooth](#) package

# Maps and spatial visualizations

Dr. John Snow's map of cholera in London, 1854

Enhanced in R in the [HistData](#) package to make Snow's point

These and other historical examples come from Friendly & Wainer, *The Origin of Graphical Species*, Harvard Univ. Press, in progress.

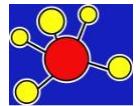


```
SnowMap(density=TRUE,  
       main="Snow's Cholera Map with Pump Neighborhoods")
```

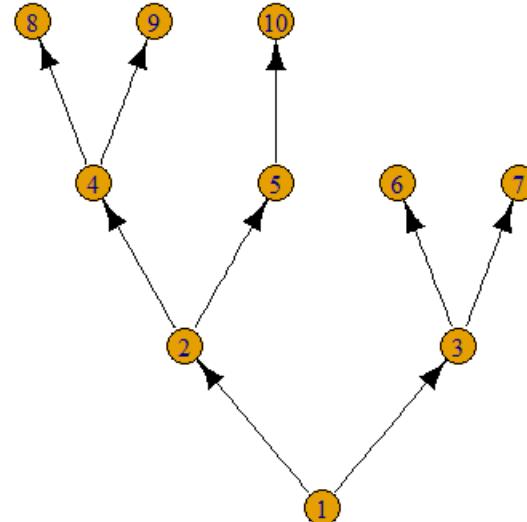
Neighborhoods are the Voronoi polygons of the map closest to each pump, calculated using the [deldir](#) package.

# Diagrams: Trees & Graphs

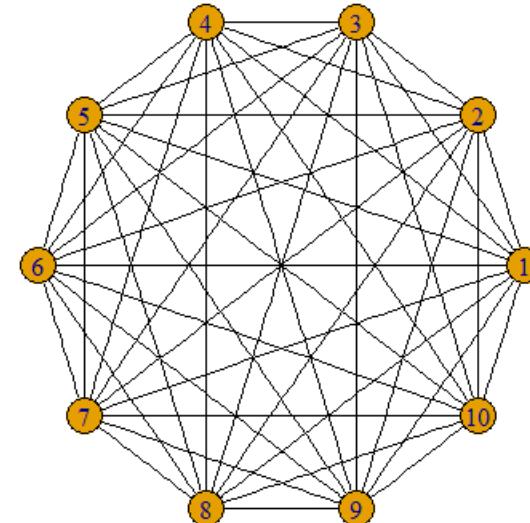
A number of R packages are specialized to draw particular types of diagrams. [igraph](#) is designed for network diagrams of nodes and edges



```
library(igraph)
tree <- graph.tree(10)
tree <- set.edge.attribute(tree, "color", value="black")
plot(tree,
      layout=layout.reingold.tilford(tree,
                                     root=1, flip.y=FALSE))
```



```
full <- graph.full(10)
fullgraph <- set.edge.attribute(full, "color",
                                 value="black")
plot(full, layout=layout.circle)
```



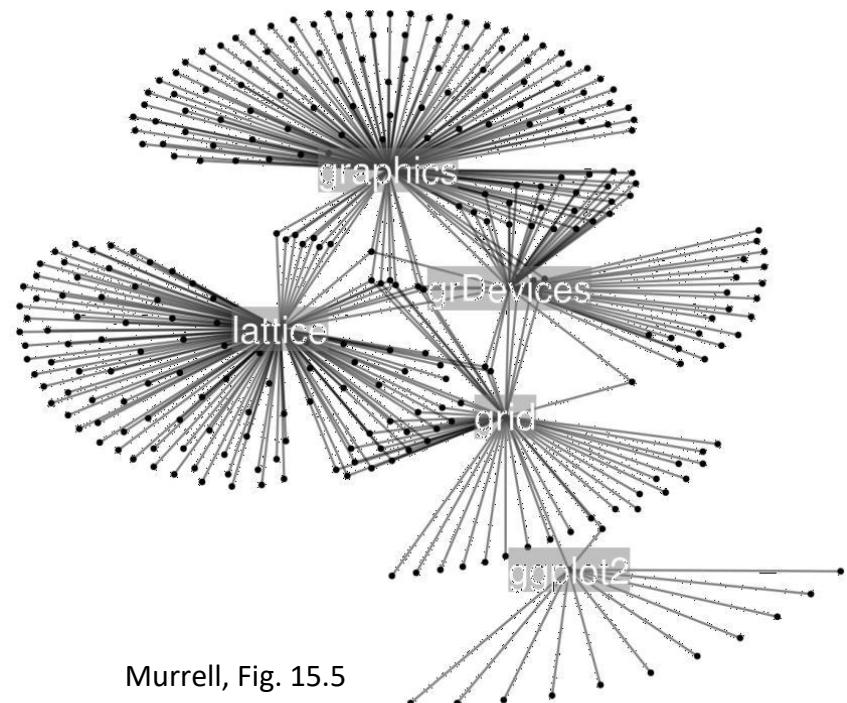
# Diagrams: Network diagrams

graphviz (<http://www.graphviz.org/>) is a comprehensive program for drawing network diagrams and abstract graphs. It uses a simple notation to describe nodes and edges.

The [Rgraphviz](#) package (from Bioconductor) provides an R interface

This example, from Murrell's *R Graphics* book, shows a node for each package that directly depends on the main R graphics packages.

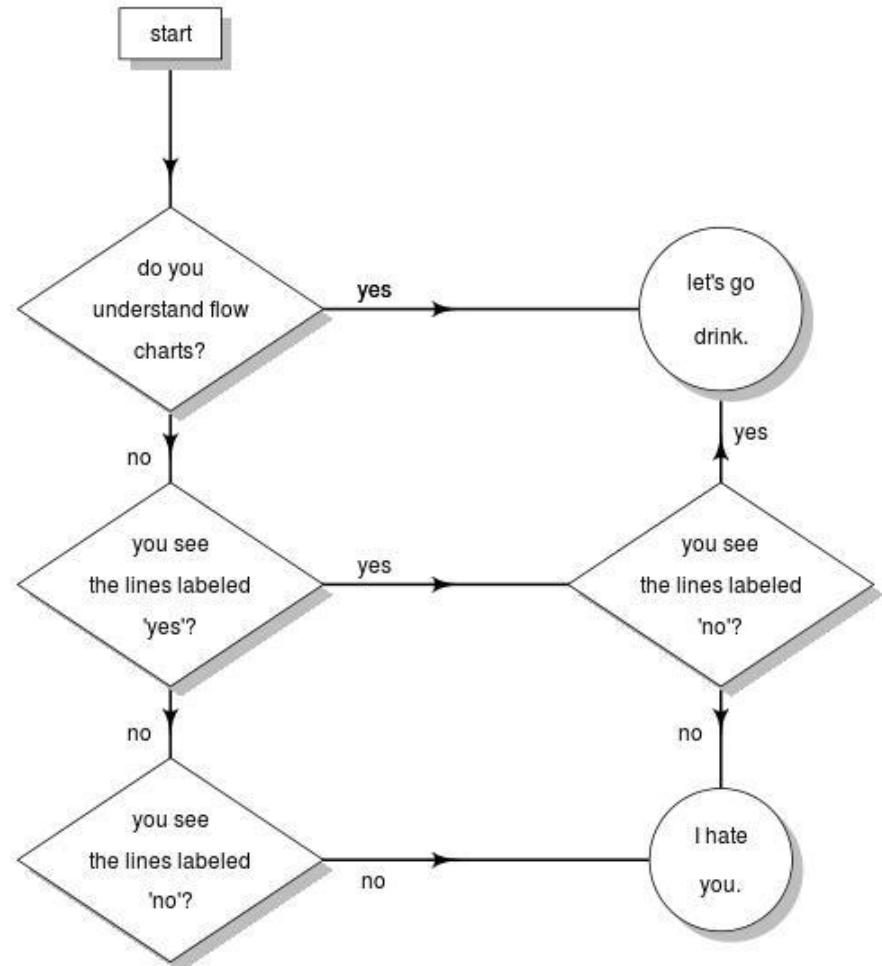
An interactive version could provide "tool tips", allowing exploring the relationships among packages



# Diagrams: Flow charts

The [diagram](#) package:

Functions for drawing diagrams  
with various shapes, lines/arrows,  
text boxes, etc.



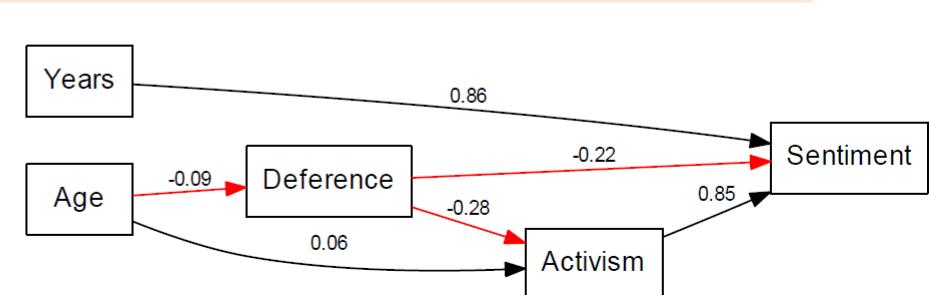
Flow chart about understanding flow charts (after <http://xkcd.com/518>). From: Murrell, Fig 15.10

# Path diagrams: structural equation models

Similar diagrams are used to display structural equation models as “path diagrams” The `sem` and `lavaan` packages have `pathDiagram()` functions to draw a proposed or fitted model.

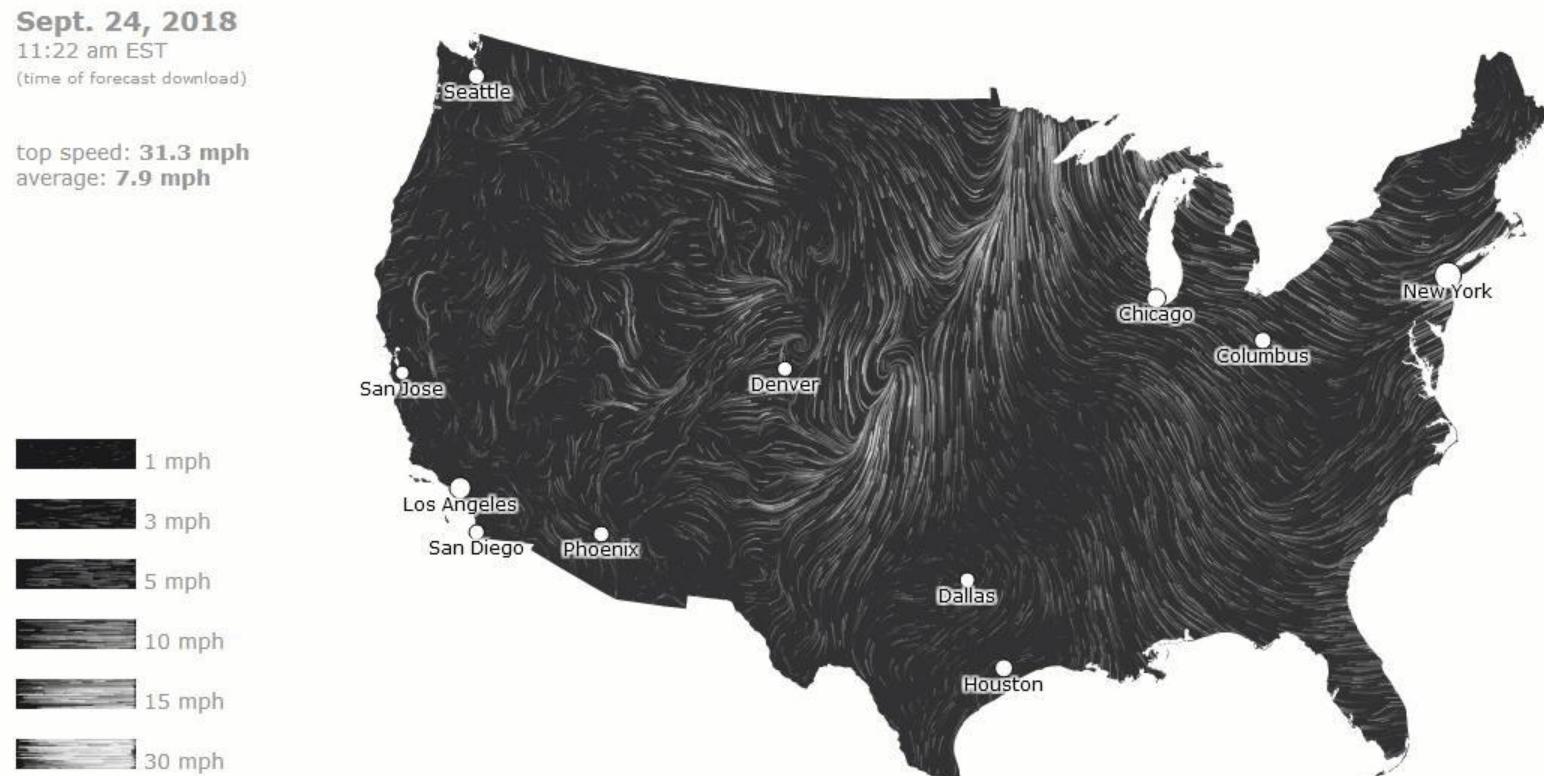
They use the `DiagrammeR` package to do the drawing.

```
library(sem)
union.mod <- specifyEquations(covs="x1, x2", text="
  y1 = gam12*x2
  y2 = beta21*y1 + gam22*x2
  y3 = beta31*y1 + beta32*y2 + gam31*x1
")
union.sem <- sem(union.mod, union, N=173)
pathDiagram(union.sem,
  edge.labels="values",
  file="union-sem1",
  min.rank=c("x1", "x2"))
```



# Dynamically updated data visualizations

The wind map app, <http://hint.fm/wind/> is one of a growing number of R-based applications that harvests data from standard sources, and presents a visualization



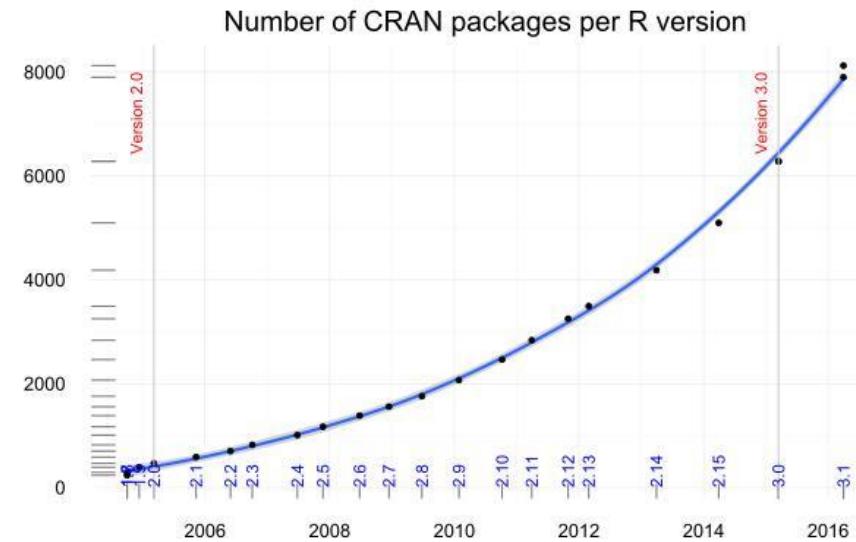
# Web scraping: CRAN package history

R has extensive facilities for extracting and processing information obtained from web pages. The [XML](#) package is one useful tool for this purpose.

This example:

- downloads information about all R packages from the CRAN web site,
- finds & counts all of those available for each R version,
- plots the counts with [ggplot2](#), adding a smoothed curve, and plot annotations

On Jan. 27, 2017, the number of R packages on CRAN reached 10,000



Code from: <https://git.io/vy4wS>

# shiny: Interactive R applications



shiny, from R Studio, makes it easier to develop interactive applications

## Shiny User Showcase

The Shiny User Showcase contains an inspiring set of sophisticated apps developed and contributed by Shiny users.



Genome browser



Paper



Lego Set Database Explorer



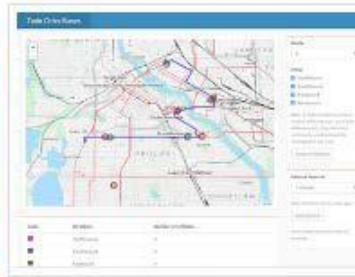
See more

## Interactive visualizations

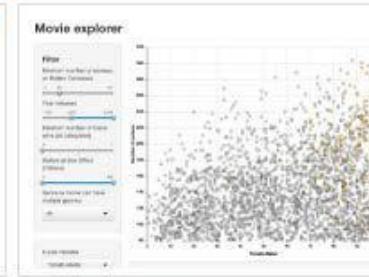
Shiny is designed for fully interactive visualization, using JavaScript libraries like d3, Leaflet, and Google Charts.



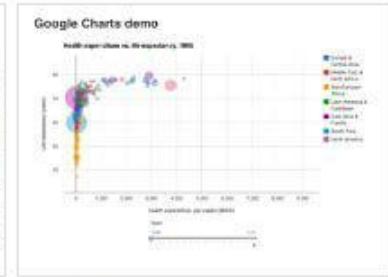
SuperZip example



Bus dashboard



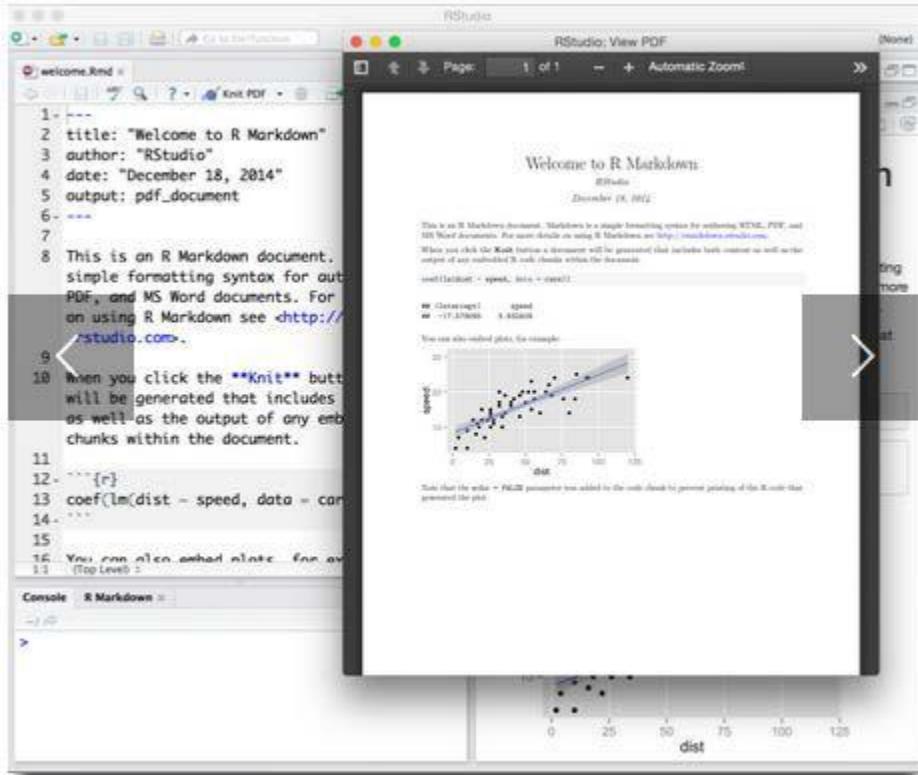
Movie explorer



Google Charts

Many examples at <https://shiny.rstudio.com/gallery/>

# Reproducible analysis & reporting



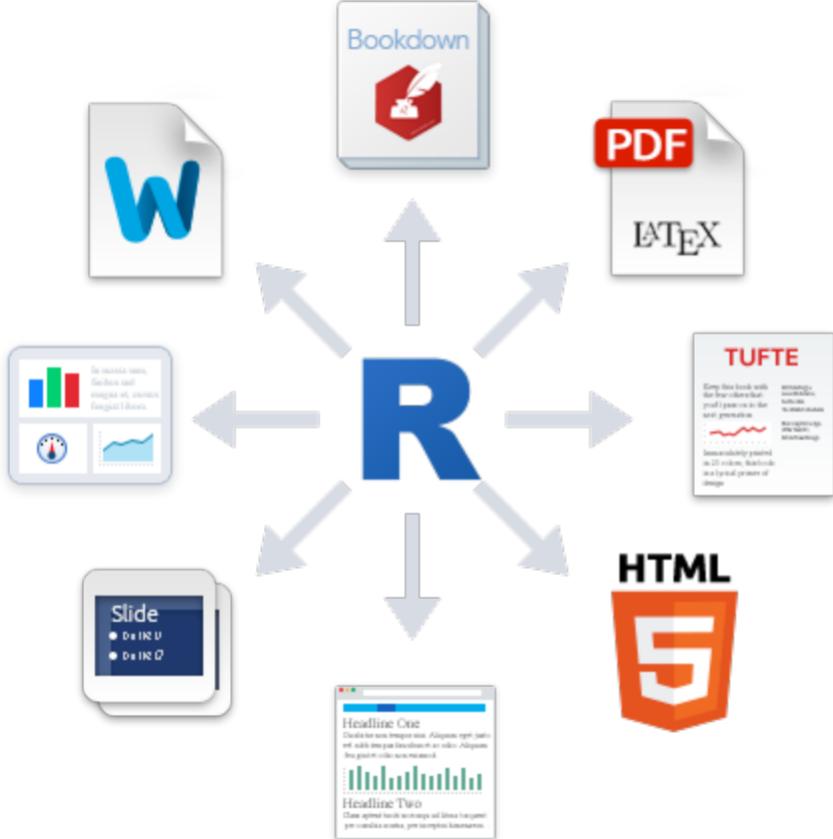
R Studio, together with the knitr and rmarkdown packages provide an easy way to combine writing, analysis, and R output into complete documents

.Rmd files are just text files, using rmarkdown markup and knitr to run R on “code chunks”

A given document can be rendered in different output formats:

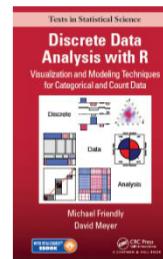


# Output formats and templates



Templates are available for APA papers, slides, handouts, entire web sites, etc.

The integration of R, R Studio, knitr, rmarkdown and other tools is now highly advanced.



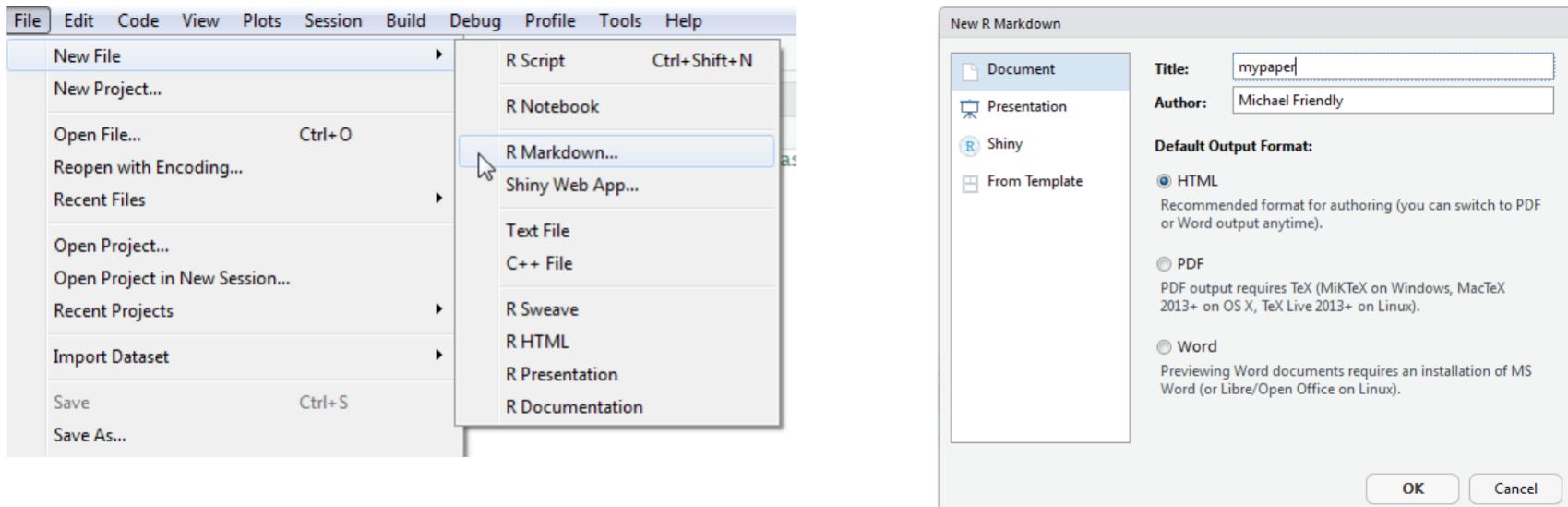
My last book was written entirely in R Studio, using .Rnw syntax → LaTeX → PDF → camera ready copy

The ggplot2 book was written using .Rmd format.

The **bookdown** package makes it easier to manage a book-length project – TOC, fig/table #s, cross-references, etc.

# Writing it up

- In R Studio, create a .Rmd file to use R Markdown for your write-up
  - lots of options: HTML, Word, PDF (needs LaTeX)



# Writing it up

- Use simple Markdown to write text
- Include code chunks for analysis & graphs

mypaper.Rmd, created from a template

```
analysis.R x read-mydata.R x mypaper.Rmd x
[...]
1 ---  
2 title: "mypaper"  
3 author: "Michael Friendly"  
4 date: "January 29, 2018"  
5 output: html_document  
6 ---  
7  
8 ```{r setup, include=FALSE}  
9 knitr::opts_chunk$set(echo = TRUE)  
10  
11  
12 ## R Markdown  
13  
14 This is an R Markdown document. Markdown is a simple formatting  
details on using R Markdown see <http://rmarkdown.rstudio.com>.  
15  
16 when you click the **Knit** button a document will be generated  
R code chunks within the document. You can embed an R code chunk  
17  
18 ```{r cars}  
19 summary(cars)  
20  
21  
22 ## Including Plots  
23  
24 You can also embed plots, for example:  
25  
26 ```{r pressure, echo=FALSE}  
27 plot(pressure)  
28
```

yaml header

Header 2

output code chunk

plot code chunk

Help -> Markdown quick reference

Files Plots Packages Help Viewer

Markdown Quick Reference Find in Topic

### Markdown Quick Reference

R Markdown is an easy-to-write plain text format for creating dynamic documents and reports. See [Using R Markdown](#) to learn more.

#### Emphasis

\*italic\* \*\*bold\*\*  
italic bold

#### Headers

# Header 1  
## Header 2  
### Header 3

#### Lists

##### Unordered List

\* Item 1  
\* Item 2  
+ Item 2a  
+ Item 2b

##### Ordered List

1. Item 1  
2. Item 2  
3. Item 3  
+ Item 3a  
+ Item 3b

# rmarkdown basics

rmarkdown uses simple markdown formatting for all standard document elements

The image shows a comparison between an R Markdown source file and its generated HTML output. On the left, the R Markdown file 'example.Rmd' is open in RStudio. It contains the following content:

```
1 # Header 1
2
3 This is an R Markdown document. Markdown is a
4 simple formatting syntax for authoring webpages.
5
6 Use an asterisk mark to provide emphasis, such
7 as *italics* or **bold**.
8
9 Create lists with a dash:
10 - Item 1
11 - Item 2
12 - Item 3
13
14 Use back ticks to
15 create a block of code
16
17
18 Embed LaTeX or MathML equations,
19 $\frac{1}{n} \sum_{i=1}^n x_i$
20
21 Or even footnotes, citations, and a
22 bibliography. [^1]
23
24 [^1]: Markdown is great.
```

On the right, the resulting HTML document 'example.html' is shown in a browser window. The content is identical to the R Markdown file, but formatted according to the specified markdown rules. Red arrows point from specific lines in the R Markdown code to their corresponding rendered elements in the HTML output. For example, the header '# Header 1' is rendered as 

# Header 1

, and the list '- Item 1' is rendered as 

- Item 1
- Item 2
- Item 3

.

# R code chunks

R code chunks are run by [knitr](#), and the results are inserted in the output document

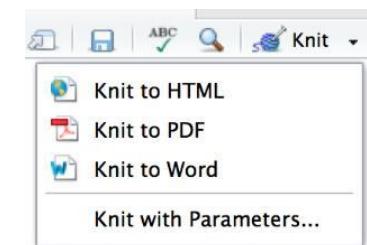
The screenshot shows the RStudio interface. On the left, the R Markdown file 'chunks.Rmd' is open, displaying R code chunks. A red box highlights the first chunk, with a red arrow pointing from it to a callout box containing the code: 

```
```{r name, options}
# R code here
```
```

. The main preview window on the right shows the resulting HTML output. The title 'R Code Chunks' is present, followed by the explanatory text 'With R Markdown, you can insert R code chunks including plots:' and the R code for a plot. Below the code is the generated output: a summary table of the 'cars' dataset and a scatter plot of 'dist' vs 'speed'. The scatter plot includes a blue regression line and a grey shaded area representing the confidence interval.

There are many options for controlling the details of chunk output – numbers, tables, graphs

Choose the output format:



# The R Markdown Cheat Sheet provides most of the details

<https://www.rstudio.com/wp-content/uploads/2016/03/rmarkdown-cheatsheet-2.0.pdf>

**R Markdown Cheat Sheet**  
learn more at [rmarkdown.rstudio.com](http://rmarkdown.rstudio.com)

**Studio**

**Workflow**

- 1 Open a new .Rmd file
- 2 Write document by editing template
- 3 Knit document to create report
- 4 Preview Output in IDE window
- 5 Publish (optional) its web or server
- 6 Examine build log in R Markdown console
- 7 Use output file that is saved alongside .Rmd

**Rmd files**  
An R Markdown (.Rmd) file is a record of your research. It contains the code that a scientist needs to reproduce your work along with the narration that a reader needs to understand your work.

**Reproducible Research**  
At the click of a button, or the type of a command, you can run the code in an R Markdown file to reproduce your work and export the results as a finished report.

**Dynamic Documents**  
You can choose to export the finished report as a html, pdf, MS Word, ODT, RTF, or Markdown document, or as a html or pdf based slide show.

**Interactive Documents**  
Turn your report into an interactive Shiny document in 4 steps.

- 1 Add runtime: shiny to the YAML header.
- 2 Call Shiny input functions to embed input objects.
- 3 Call Shiny render functions to embed reactive output.
- 4 Render with `markdown::run`, or click Run Document in RStudio IDE.

**Embed code with knitr syntax**

**Global options**  
Set with `knitr::opts_chunk$set(...)`

**Parameters**  
Parameterize your documents to reuse with different inputs (e.g., data sets, values, etc.)

**1 Add parameters**  
Create and set parameters in the header as sub-values of `params`.

**2 Call parameters**  
Call parameter values in code as `params$<name>`.

**3 Set parameters**  
Set values with `Knit with parameters` or the `params` argument of `render`.

**render("doc.Rmd", params = list(n = 1, d = as.Date("2015-01-01")))**

# R notebooks

Often, you just want to “compile” an R script, and get the output embedded in the result, in HTML, Word, or PDF. Just type Ctrl-Shift-K or tap the Compile Report button

The image shows two screenshots of RStudio. On the left, the RStudio interface is displayed with several tabs open: '3d-demo.R', '3d-demo.Rmd', and 'inv-ex1.R'. A red arrow points from the 'inv-ex1.R' tab to the toolbar above, specifically to the 'Compile Report (Ctrl+Shift+K)' button, which is highlighted with a red box. On the right, the resulting HTML document 'inv-ex1.html' is shown in a browser window. The document title is 'Inverse of a matrix' by Michael Friendly on 07 Sep 2016. It contains R code for creating a matrix and calculating its inverse, along with the resulting output.

**Inverse of a matrix**  
Michael Friendly  
07 Sep 2016

The following examples illustrate the basic properties of the inverse of a matrix

Load the `matlib` package

This defines: `inv()`, `Inverse()`; the standard function for matrix inverse is `solve()`

```
library(matlib)
```

Create a 3 x 3 matrix

```
A <- matrix(c(5, 1, 0,  
            3,-1, 2,  
            4, 0,-1), nrow=3, byrow=TRUE)  
det(A)
```

## [1] 16

1.  $\det(A) \neq 0$ , so inverse exists

```
(AI <- inv(A))
```

```
##          [,1]     [,2]     [,3]  
## [1,] 0.0625  0.0625  0.125  
## [2,] 0.6875 -0.3125 -0.625  
## [3,] 0.2500  0.2500 -0.500
```

2. Definition of the inverse:  $A^{-1}A = AA^{-1} = I$  or  
 $AI * A = diag(nrow(A))$

NB: Sometimes you will get very tiny off-diagonal values (like  $1.341e-13$ ). The function `zapsmall()` will round those to 0.

```
AI %*% A
```

# Summary & Homework

- Today has been mostly about an overview of R graphics, but with emphasis on:
  - R, R Studio, R package tools
  - Roles of graphics in data analysis,
  - A small gallery of examples of different kinds of graphic applications in R; only small samples of R code
  - Work flow: How to use R productively in analysis & reporting
- Next week: start on skills with traditional graphics
- Homework:
  - Install R & R Studio
  - Find one or more examples of data graphs from your research area
    - What are the graphic elements: points, lines, areas, regions, text, labels, ???
    - How could they be “described” to software such as R?
    - How could they be improved?