

Data Visualization in R

1. Overview

Michael Friendly
SCS Short Course
Sep/Oct, 2018

<http://datavis.ca/courses/RGraphics/>



Course outline

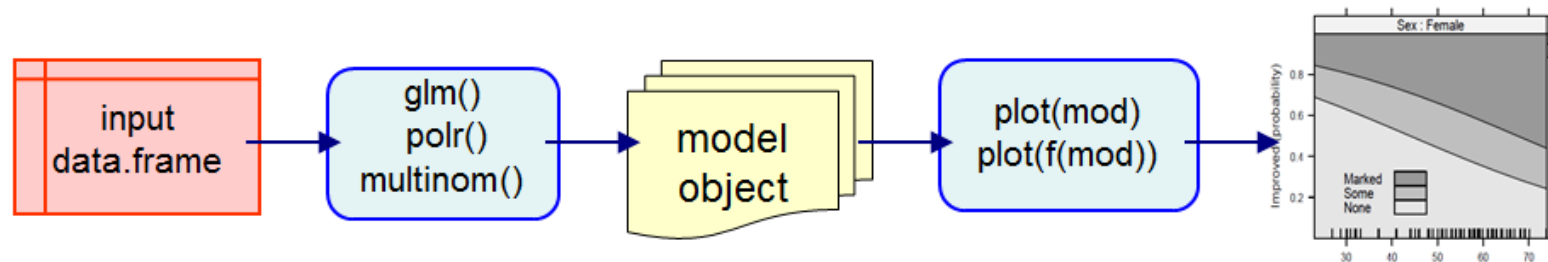
1. Overview of R graphics
2. Standard graphics in R
- ~~3. Grid & lattice graphics~~
4. ggplot2

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

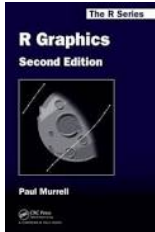
- Session 3: Grid & lattice graphics
 - Another, more powerful “graphics engine”
 - All standard plots, with more pleasing defaults
 - Easily compose collections (“small multiples”) from subsets of data
 - vcd and vcdExtra packages: mosaic plots and others for categorical data

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

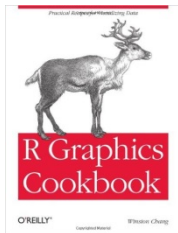
Resources: Books



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.

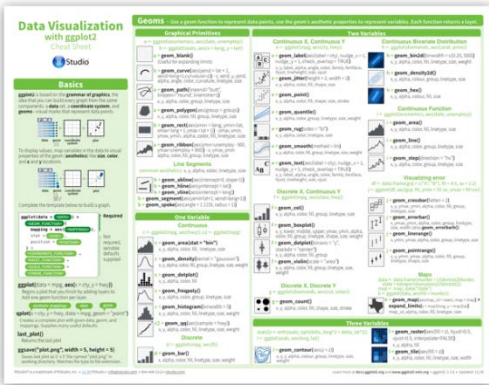
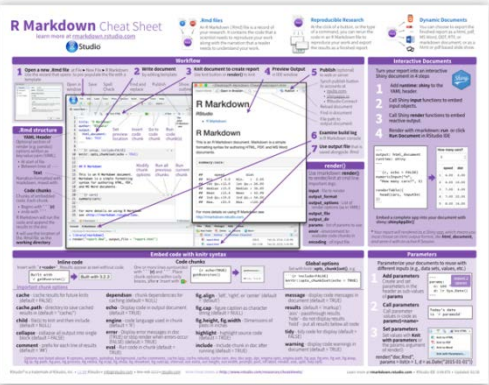
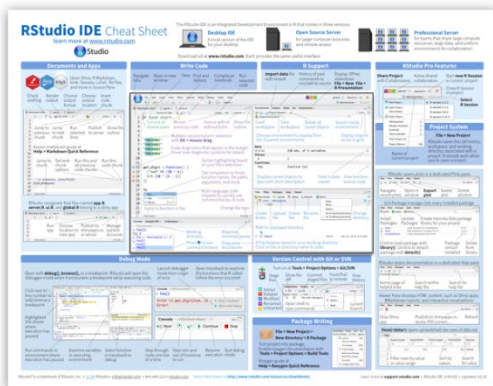
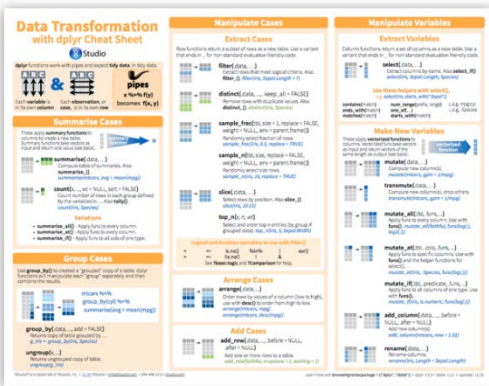
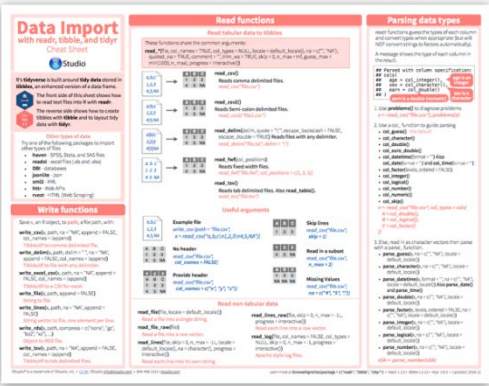
1st Ed: Online, <http://ggplot2.org/book/>

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/>



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"](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)

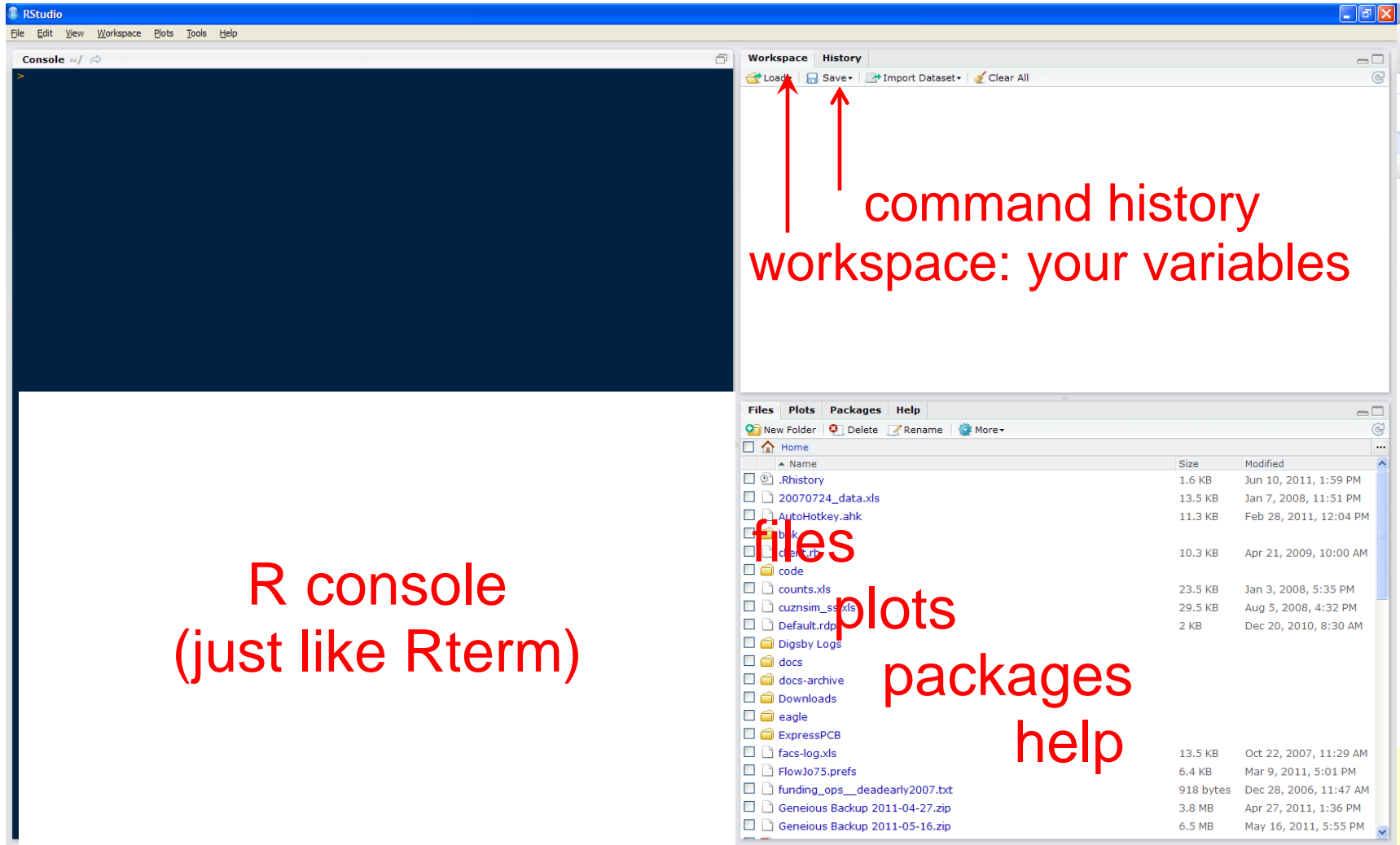


Publish: A variety of R packages make it easy to write and publish research reports and slide presentations in various formats (HTML, Word, LaTeX, ...), all within R Studio



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

Getting started: R Studio



R Studio navigation

R folder navigation commands:

- Where am I?

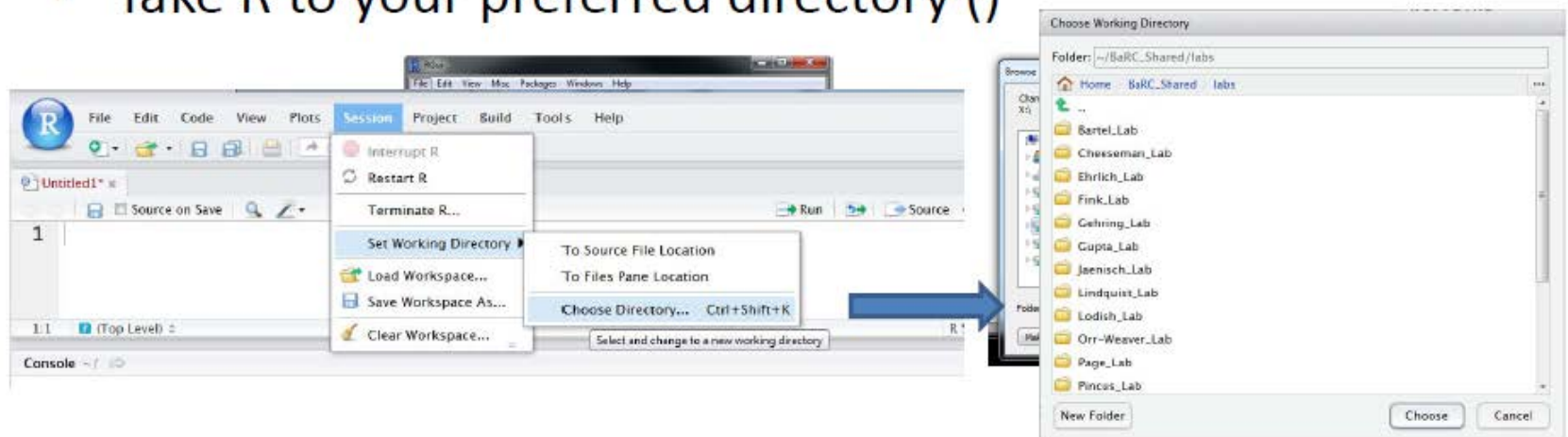
```
> getwd()  
[1] "C:/Dropbox/Documents/6135"
```

- Go somewhere:

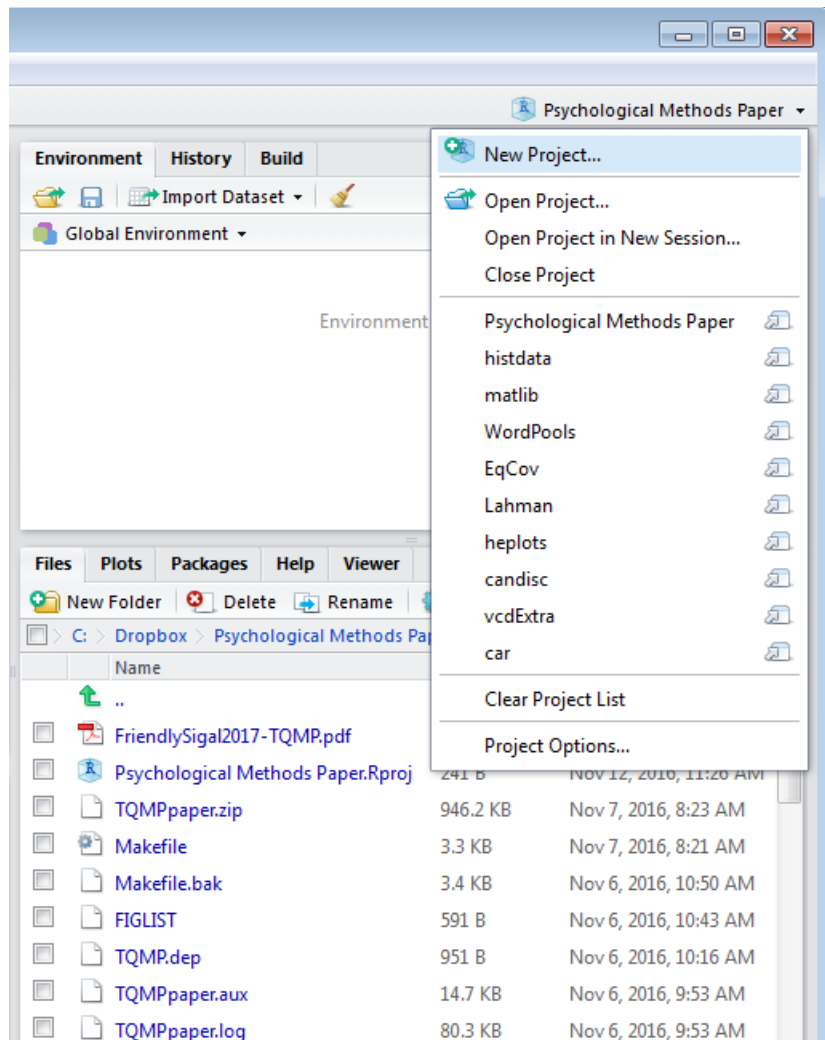
```
> setwd("C:/Dropbox")  
> setwd(file.choose())
```

R Studio GUI

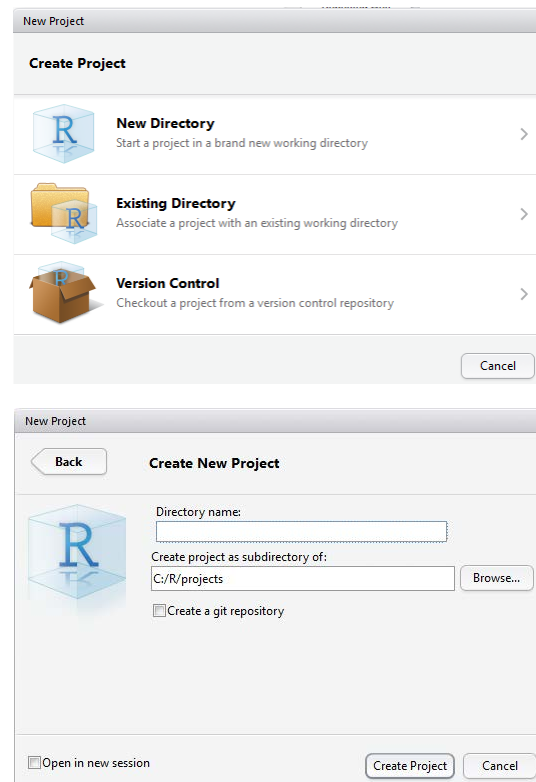
- Take R to your preferred directory ()



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 displays the R Studio environment with a project titled "Psychological Methods Paper". The main editor shows an R Markdown file named "TQMPpaper.Rmd" with the following content:

```
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     id: 1
15     institution: York University
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
```

The console at the bottom shows the R startup message and the license information:

```
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

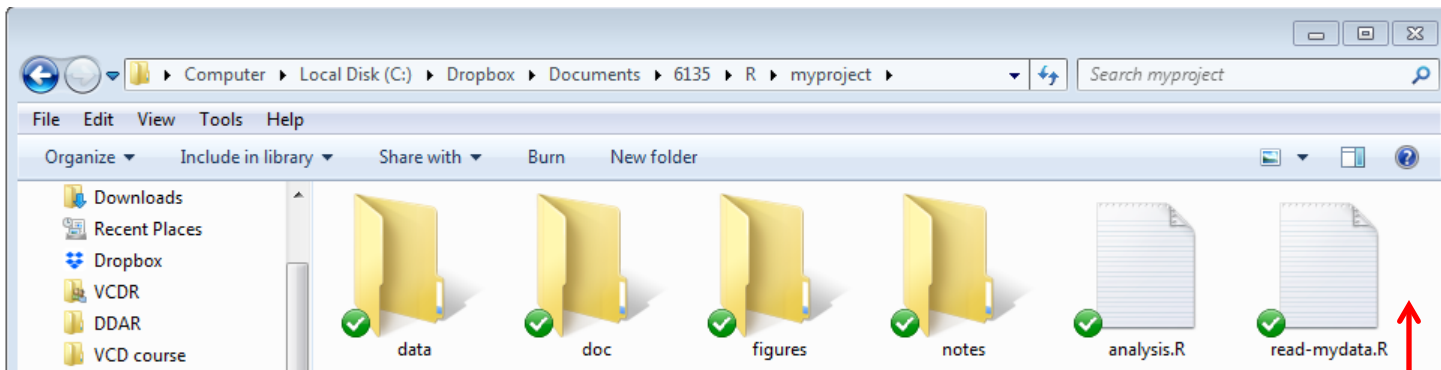
> |
```

The right-hand pane shows the "Files" tab with a list of files in the project directory:

Name	Size	Modified
..		
FriendlySigal2017-TQMP.pdf		
Psychological Methods Paper.Rproj		
TQMPpaper.zip	946.2 KB	Nov 7, 2016, 8:23 AM
Makefile	3.3 KB	Nov 7, 2016, 8:21 AM
Makefile.bak	3.4 KB	Nov 6, 2016, 10:50 AM
FIGLIST	591 B	Nov 6, 2016, 10:43 AM
TQMP.dep	951 B	Nov 6, 2016, 10:16 AM
TQMPpaper.aux	14.7 KB	Nov 6, 2016, 9:53 AM
TQMPpaper.log	80.3 KB	Nov 6, 2016, 9:53 AM
TQMPpaper.out	6 KB	Nov 6, 2016, 9:53 AM
TQMPpaper.pdf	819.8 KB	Nov 6, 2016, 9:53 AM
TQMPpaper.synctex.gz	286.1 KB	Nov 6, 2016, 9:53 AM
.Rhistory	13.5 KB	Nov 3, 2016, 9:35 AM

Organizing an R project

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



data files:

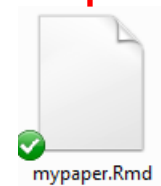
- raw data (.csv)
- saved R data (.Rdata)

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")

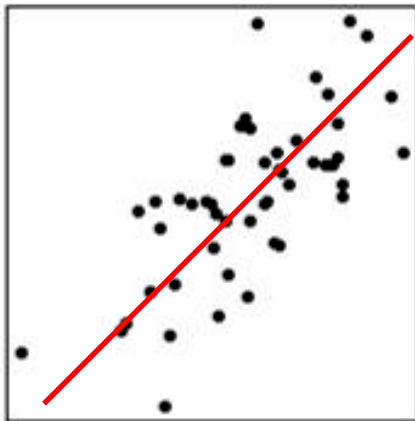
# do the analysis – exploratory plots
plot(mydata)

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

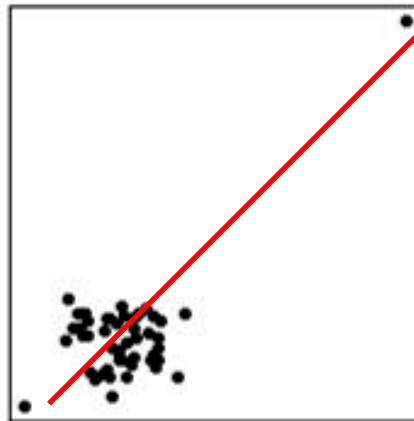
# plot models, extract model summaries
plot(mymod.1)
summary(mymod.1)
```

Graphics: Why plot your data?

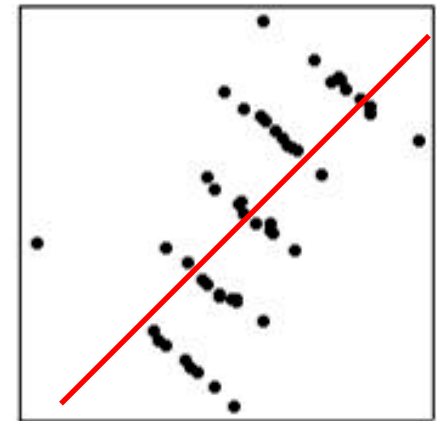
- Three data sets with exactly the same bivariate summary statistics:
 - Same correlations, linear regression lines, etc
 - Indistinguishable from standard printed output



Standard data



$r=0$ but + 2 outliers



Lurking variable?

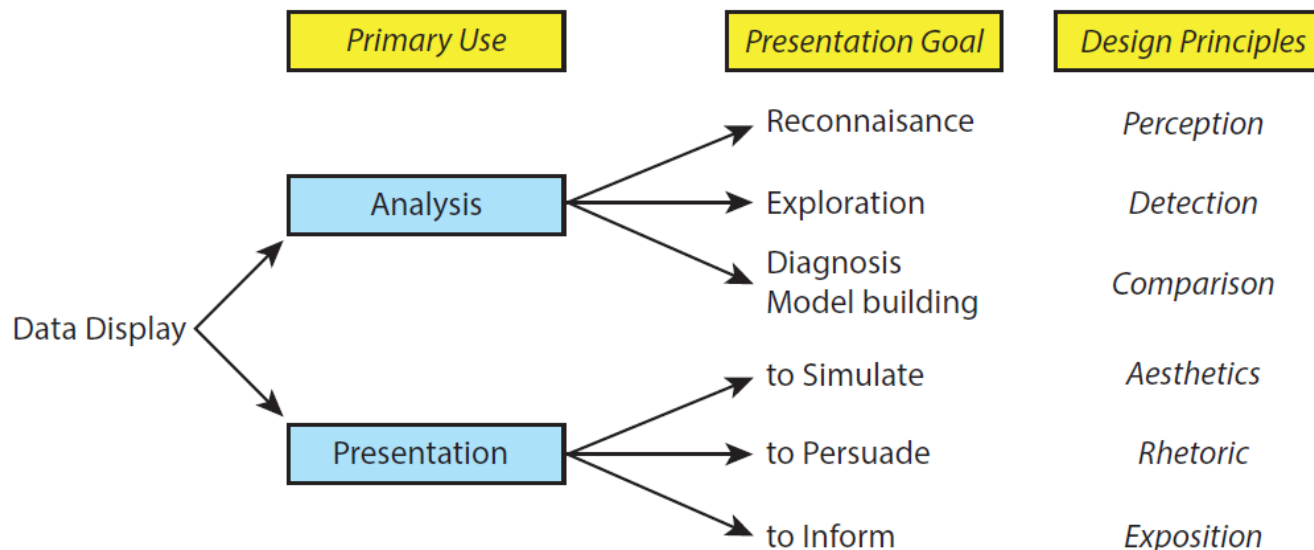
Roles of graphics in data analysis

- 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

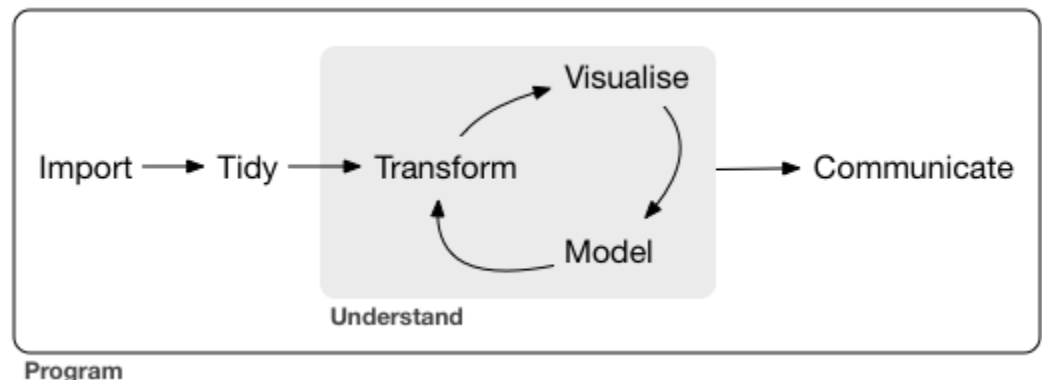
Basic functions of data display



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”

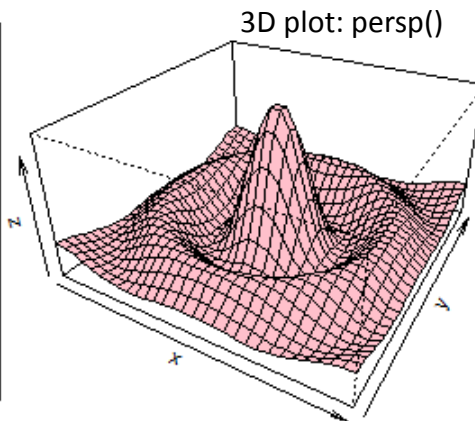
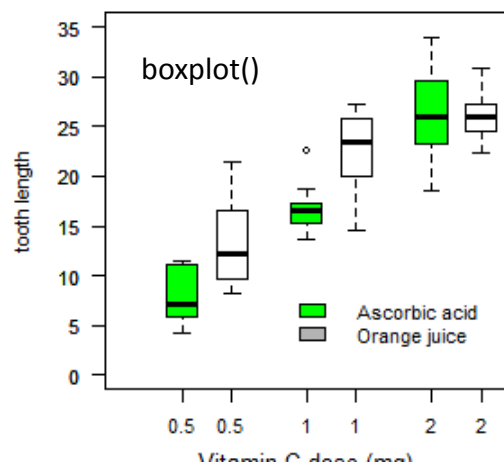
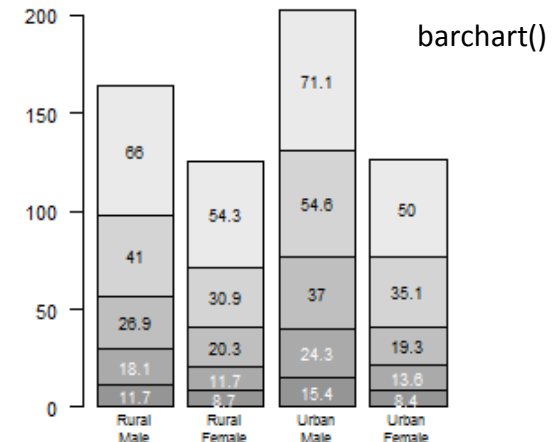
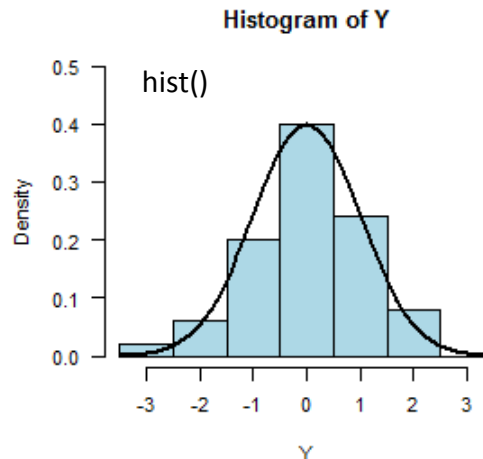
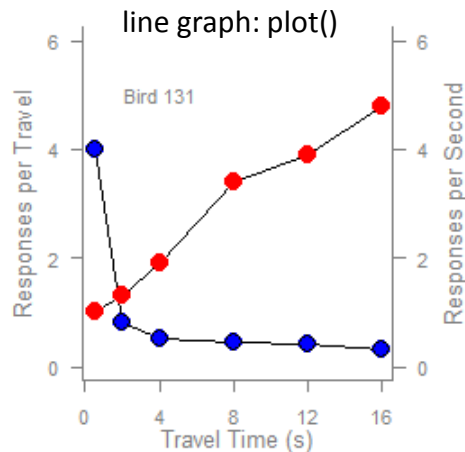


The 80-20 rule: Graphics

- **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)

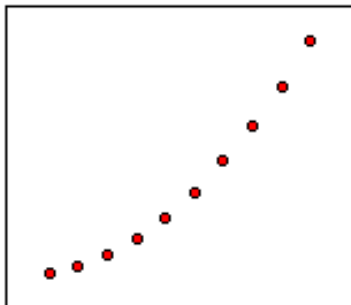


Bivariate plots

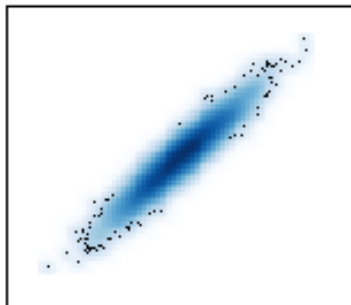
R base graphics provide a wide variety of different plot types for bivariate data

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

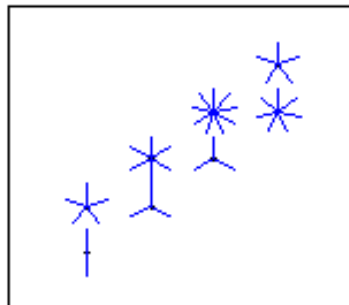
`plot(num, num)`



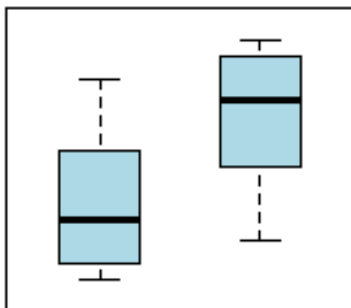
`smoothScatter()`



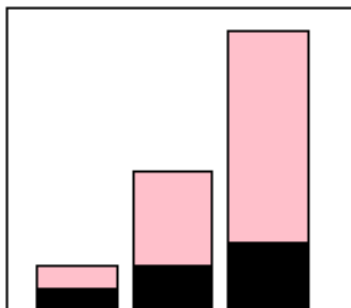
`sunflowerplot()`



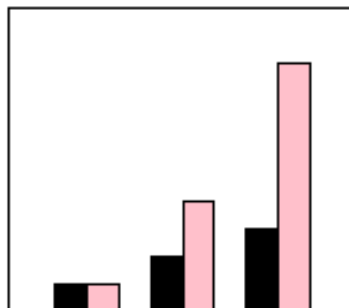
`boxplot(list)`
`plot(fac, num)`



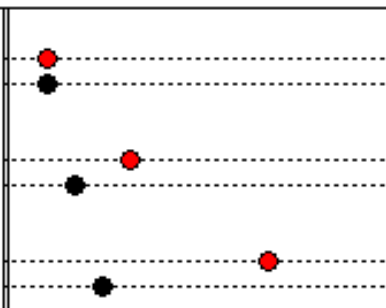
`barplot(matrix)`



`barplot(matrix)`



`dotchart(matrix)`



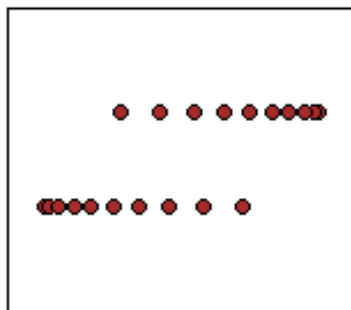
Some plotting functions take a **matrix** argument & plot all columns

Bivariate plots

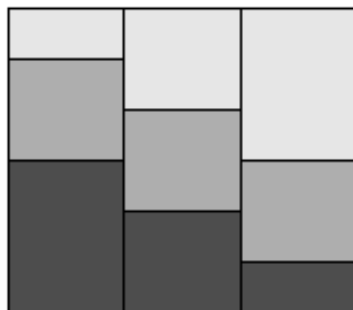
A number of specialized plot types are also available in base R graphics

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

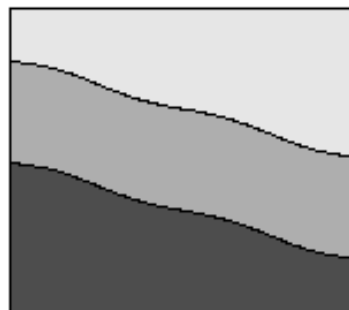
`stripchart(list)`
`plot(num, fac)`



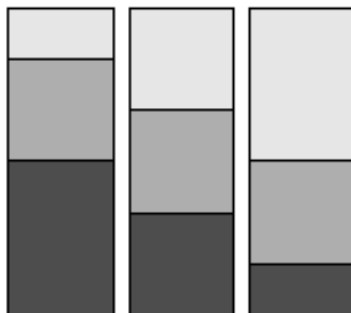
`spineplot(num, fac)`



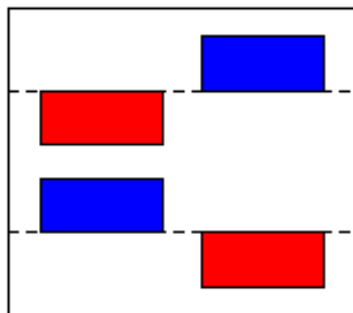
`cdplot()`



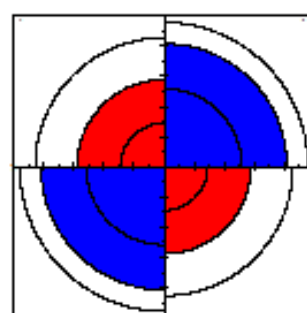
`spineplot(fac, fac)`
`plot(fac, fac)`



`assocplot()`

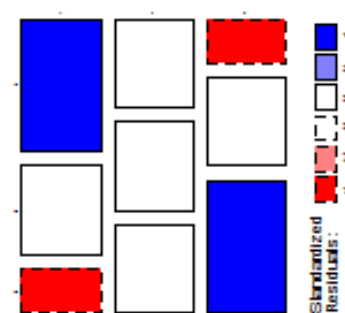


`fourfoldplot()`



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

`mosaicplot()`
`plot(table)`



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
```

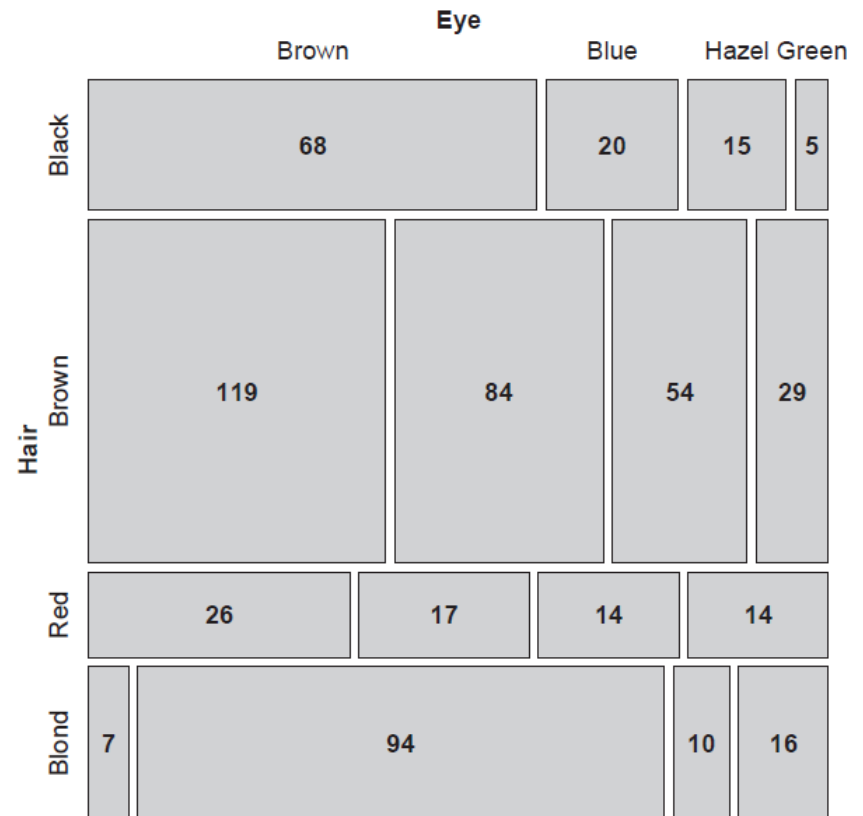
	Eye			
Hair	Brown	Blue	Hazel	Green
Black	68	20	15	5
Brown	119	84	54	29
Red	26	17	14	14
Blond	7	94	10	16

```
> 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?

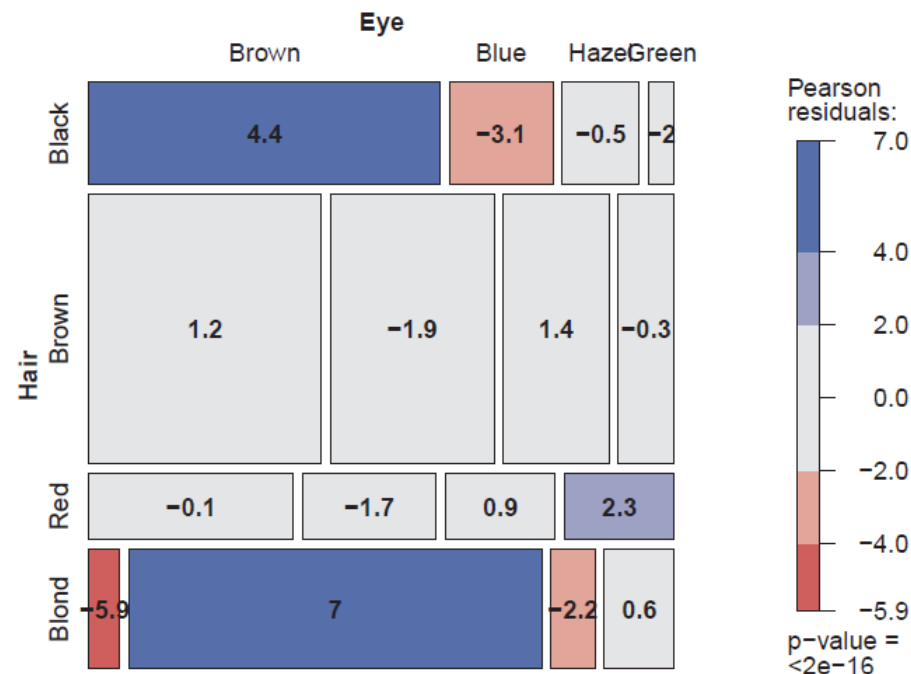
Mosaic plots

Shade each tile in relation to the contribution to the Pearson χ^2 statistic

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

```
> round(residuals(chisq.test(HEC)), 2)
```

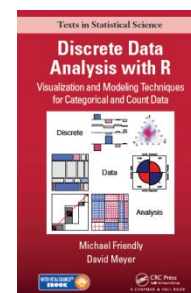
	Eye			
Hair	Brown	Blue	Hazel	Green
Black	4.40	-3.07	-0.48	-1.95
Brown	1.23	-1.95	1.35	-0.35
Red	-0.07	-1.73	0.85	2.28
Blond	-5.85	7.05	-2.23	0.61



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/>



Follow along

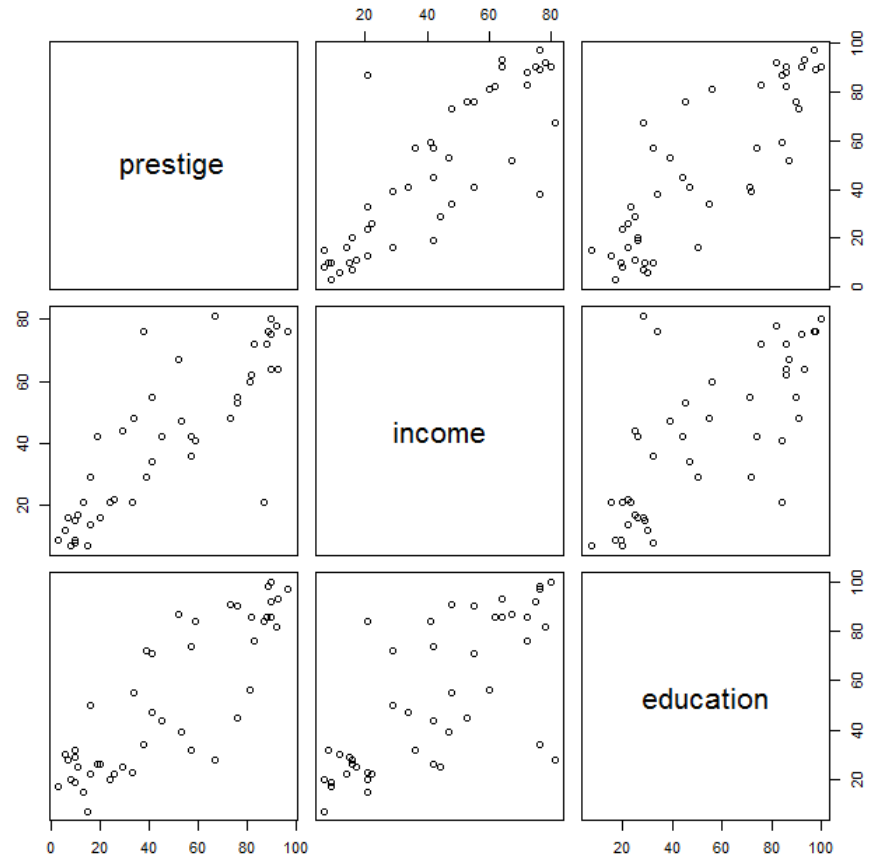
- From the course web page, click on the script duncan-plots.R,
<http://www.datavis.ca/courses/RGraphics/R/duncan-plots.R>
- Select all (ctrl+A) and copy (ctrl+C) to the clipboard
- In R Studio, open a new R script file (ctrl+shift+N)
- Paste the contents (ctrl+V)
- Run the lines (ctrl+Enter) along with me

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)
```

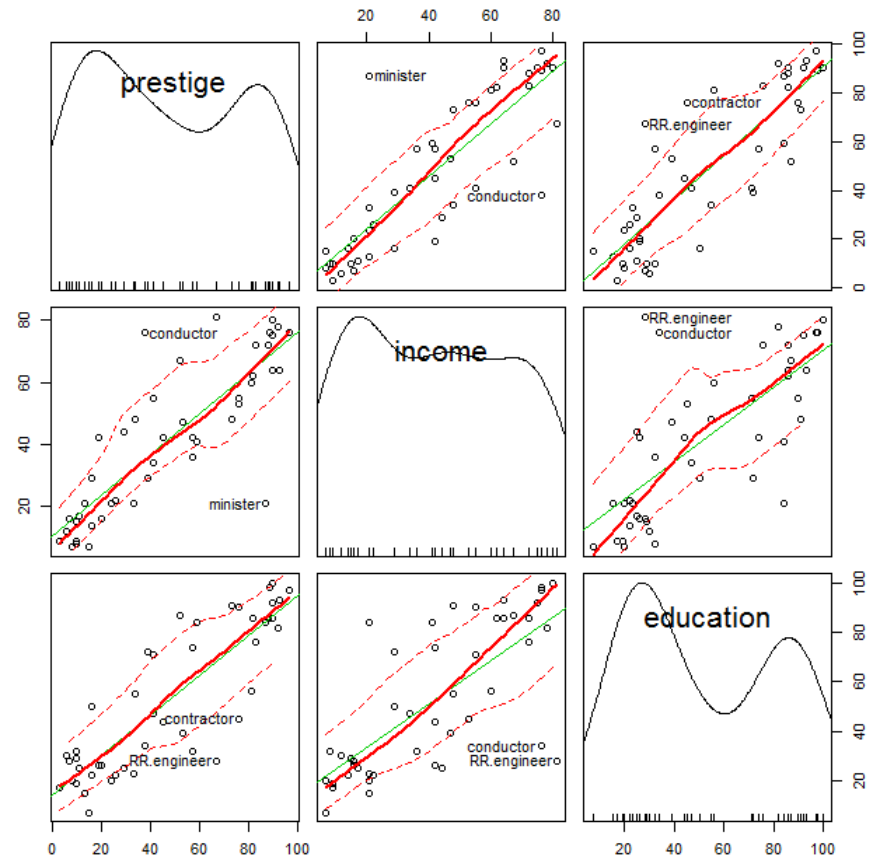


Multivariate plots

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=`)



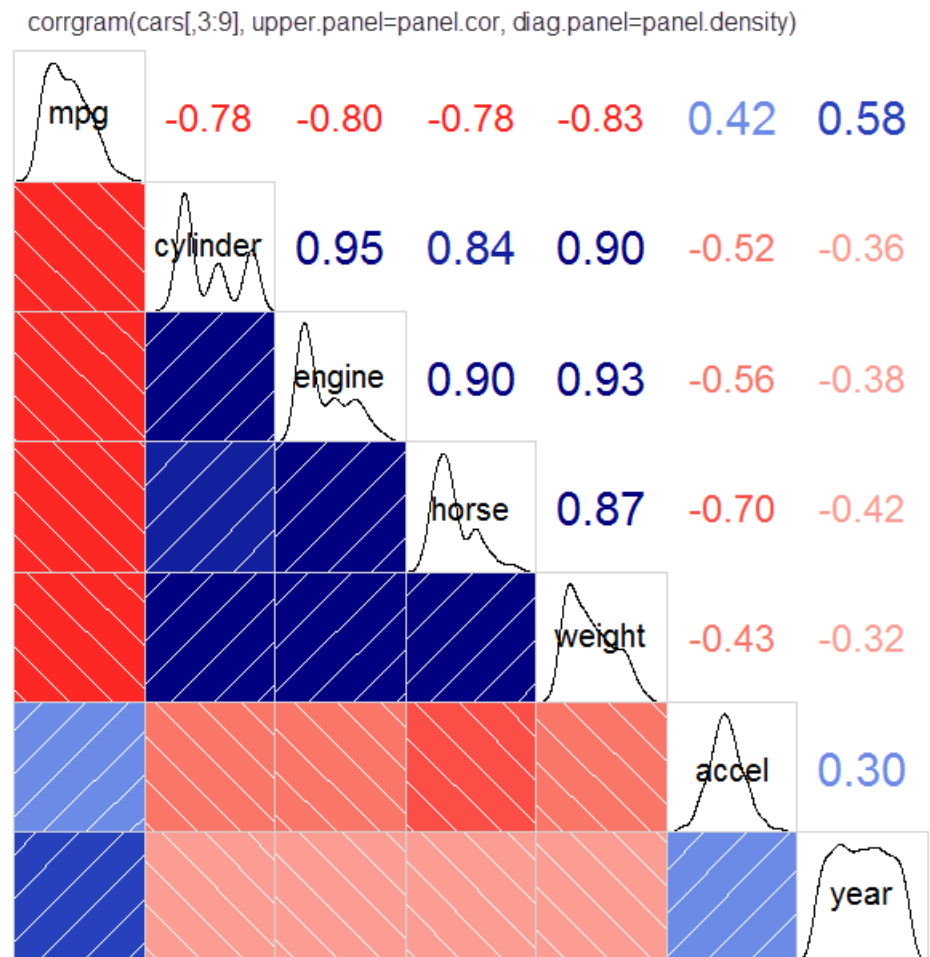
```
library(car)
scatterplotMatrix(~prestige + income + education, data=Duncan, id.n=2)
```

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

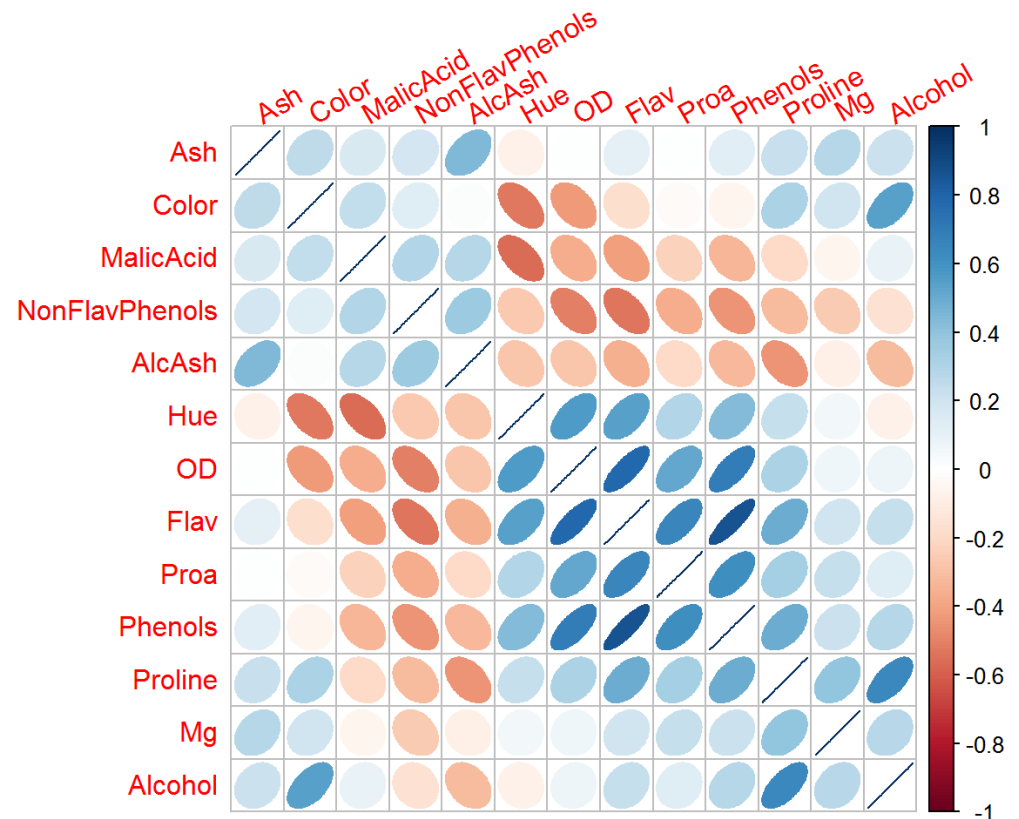


Multivariate plots: corrgrams

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.

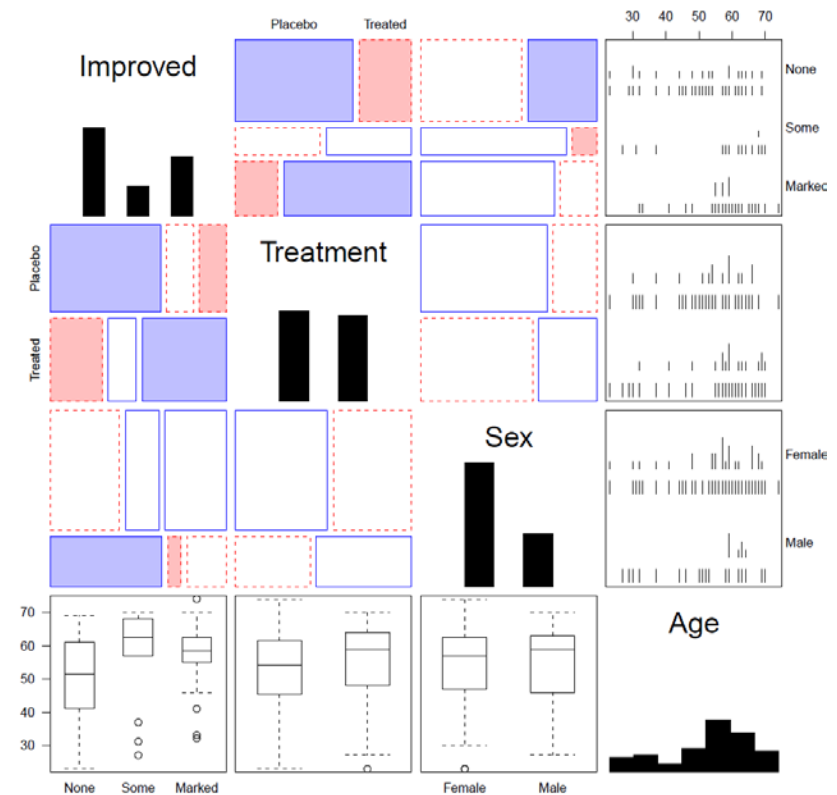


```
library(corrplot)
corrplot(cor(wine), tl.srt=30, method="ellipse", order="AOE")
```

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



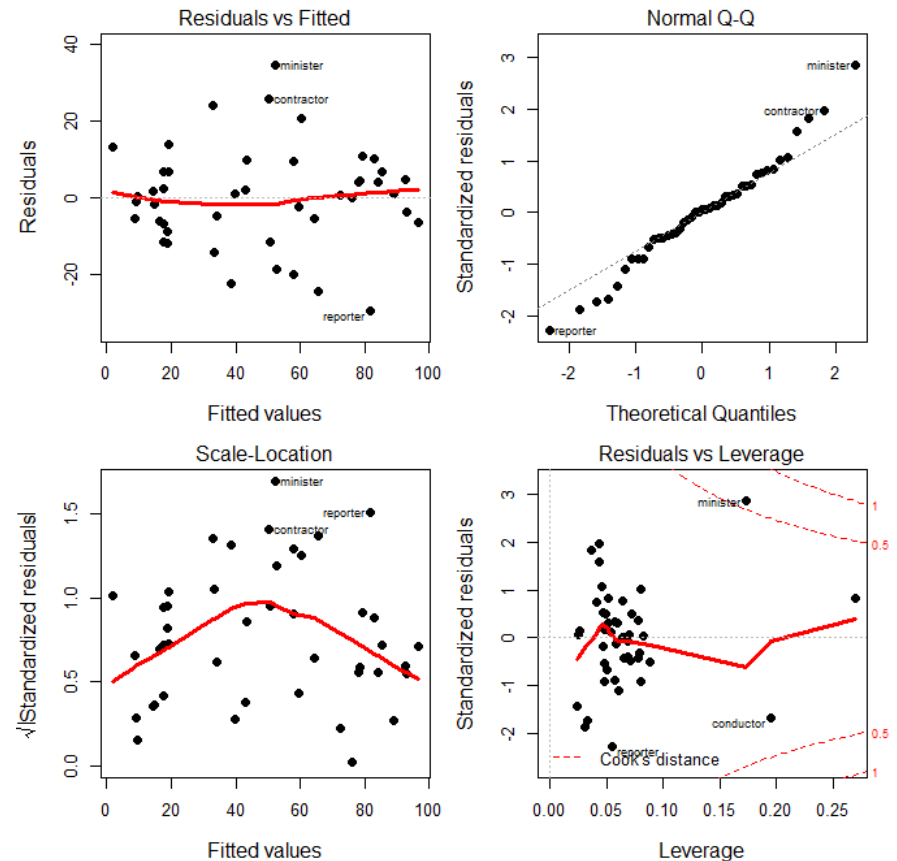
```
library(gpairs)
data(Arthritis)
gpairs(Arthritis[, c(5, 2:5)], ...)
```


Models: diagnostic plots

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

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?

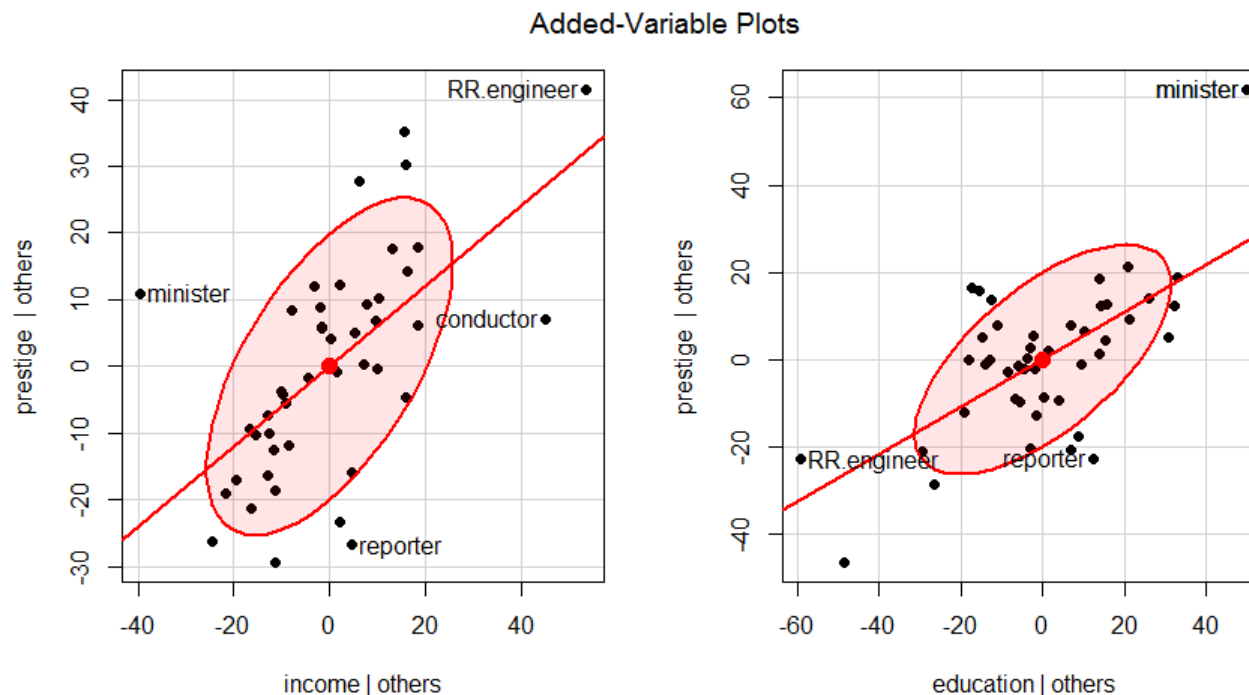


```
duncan.mod <- lm(prestige ~ income + education, data=Duncan)
plot(duncan.mod)
```

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, β_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

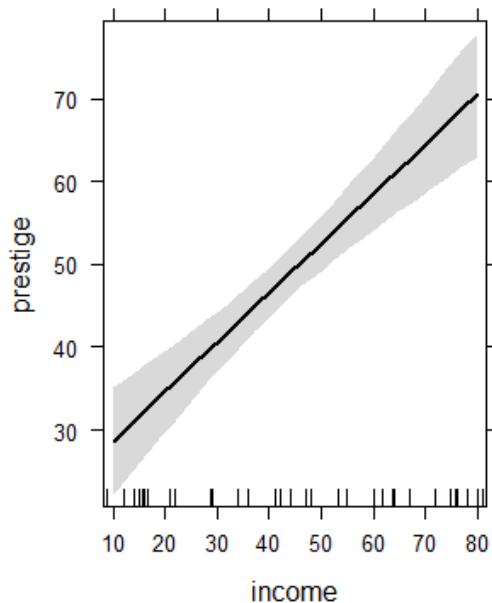
How to understand
effect of each
predictor?

Models: Effect plots

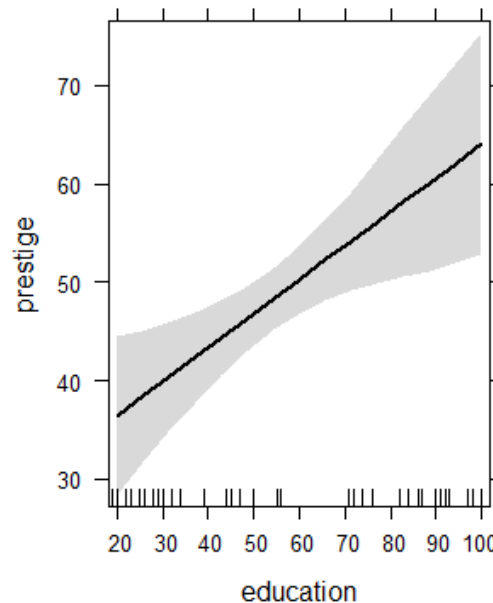
Fitted models are more easily interpreted by plotting the predicted values. Effect plots do this nicely, making plots for each **high-order term**, controlling for others

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

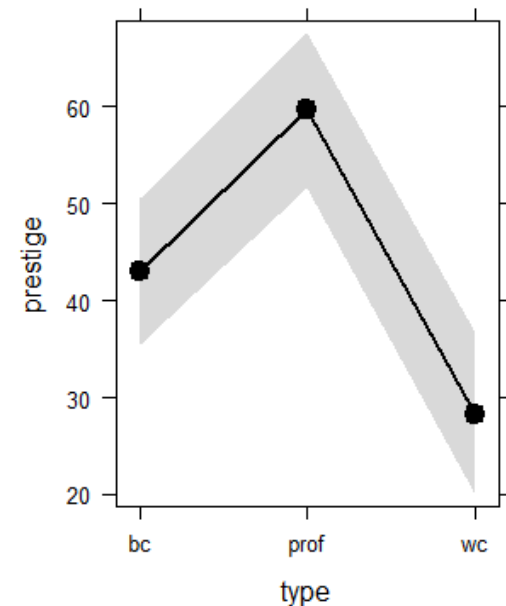
income effect plot



education effect plot



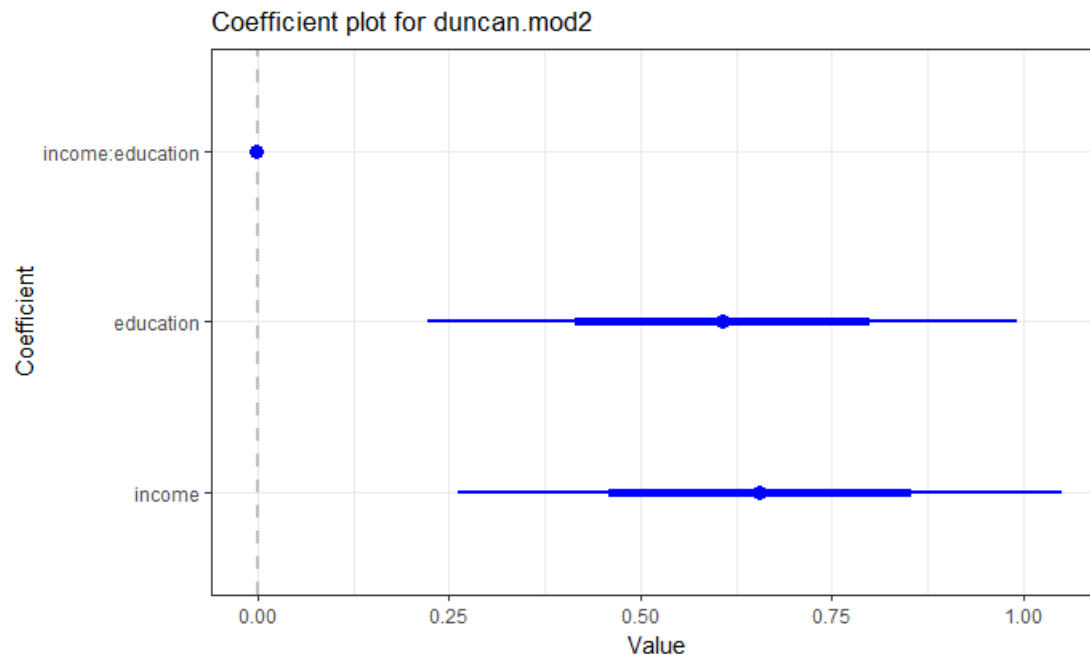
type effect plot



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")
```



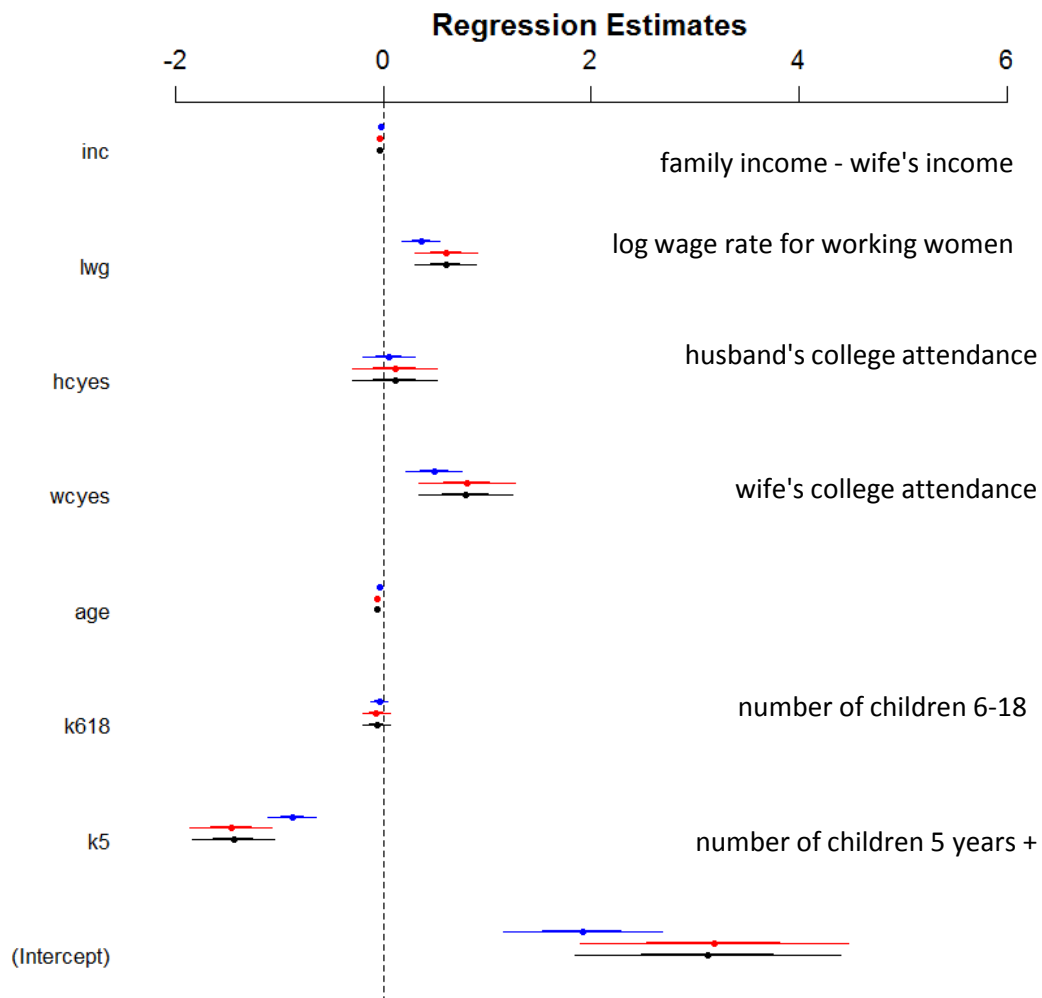
Coefficient plots become increasingly useful as:

- (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

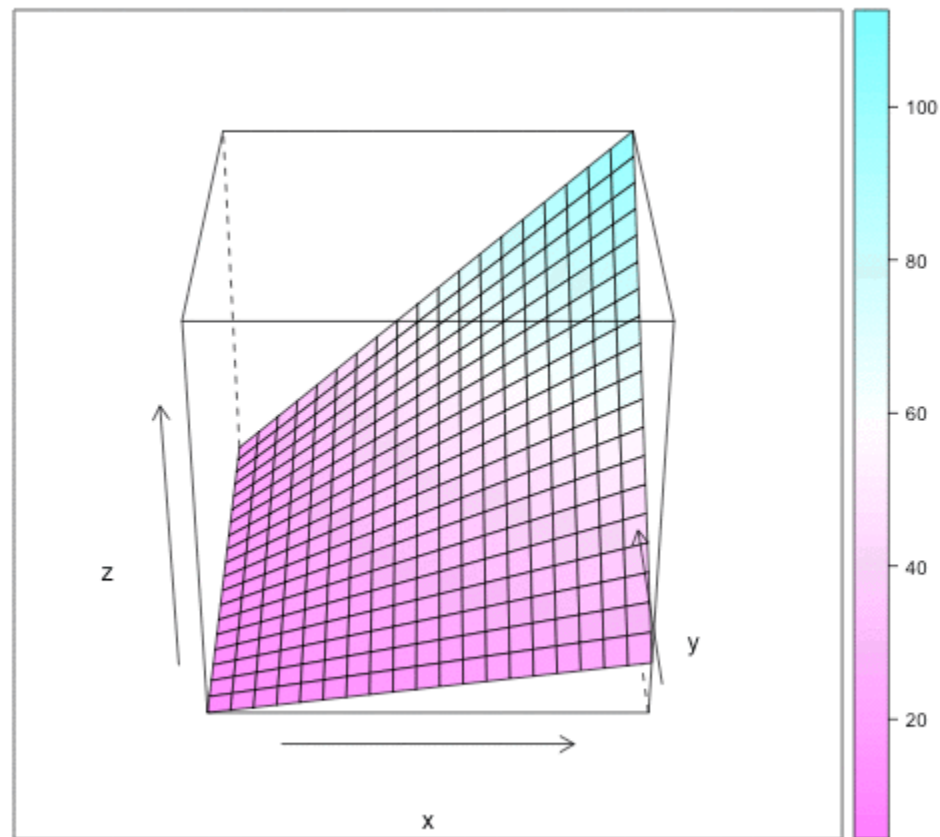
This example illustrates the concept of an [interaction](#) between predictors in a linear regression model

It uses:

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

The basic plot is “printed” 36 times rotated 10° 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()
```

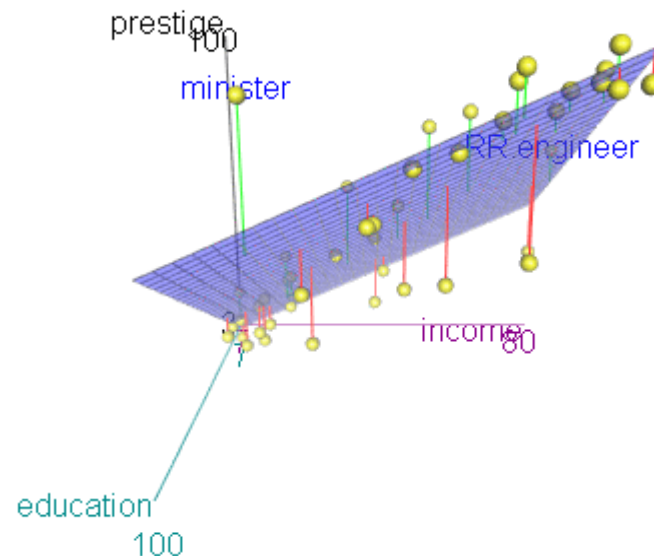
4. Convert PNGs to GIF using ImageMagik

```
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)
```

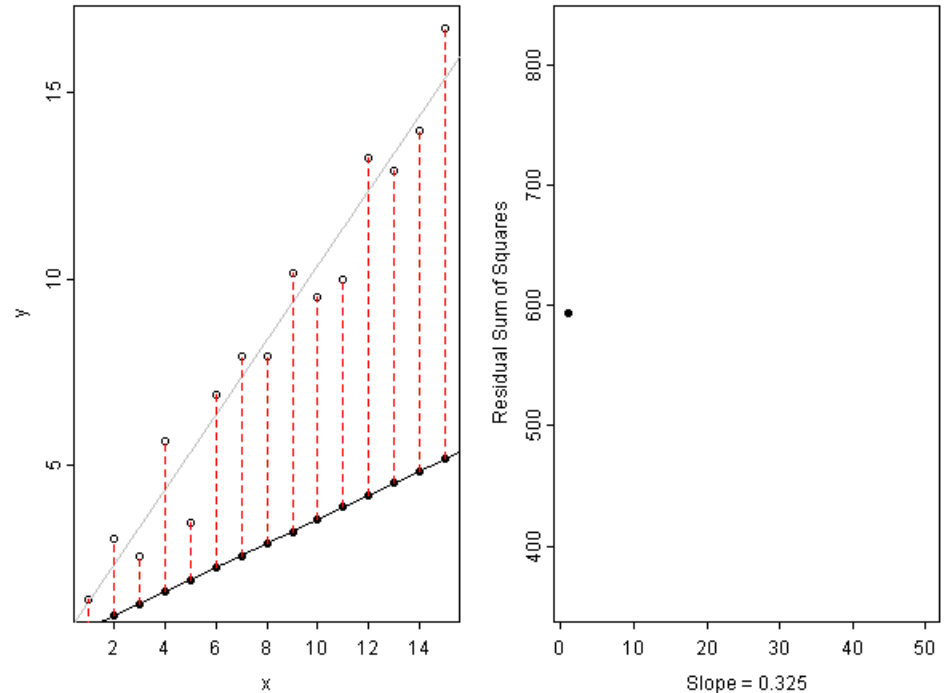
Statistical animations

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

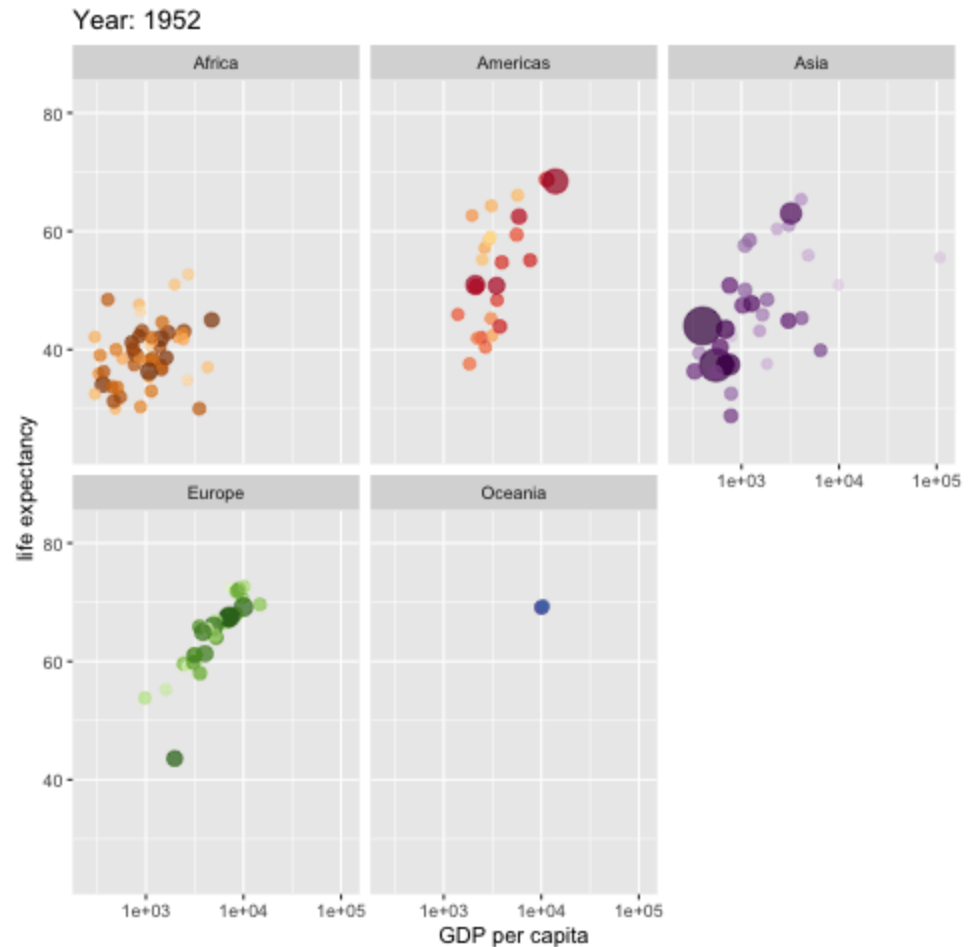


Data animations

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



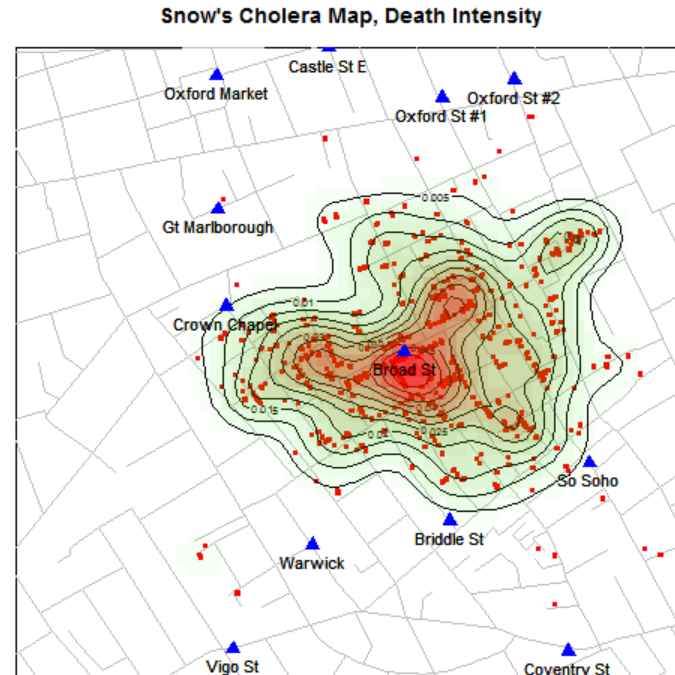
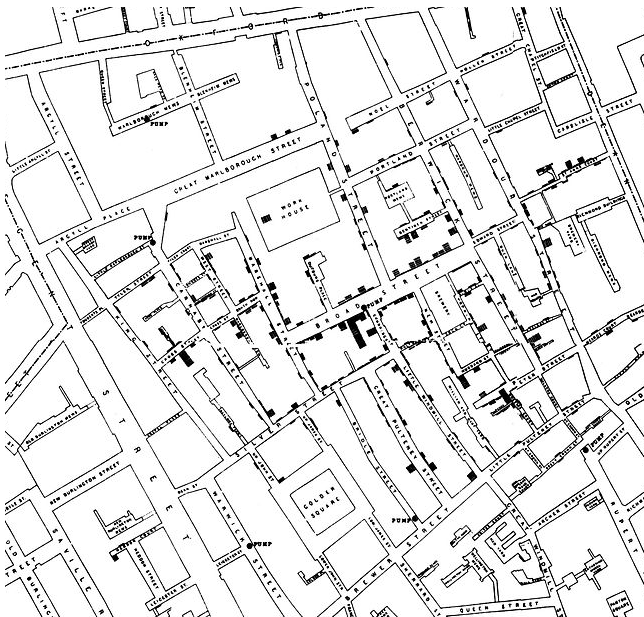
Murrell, Fig. 14.5

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")
```

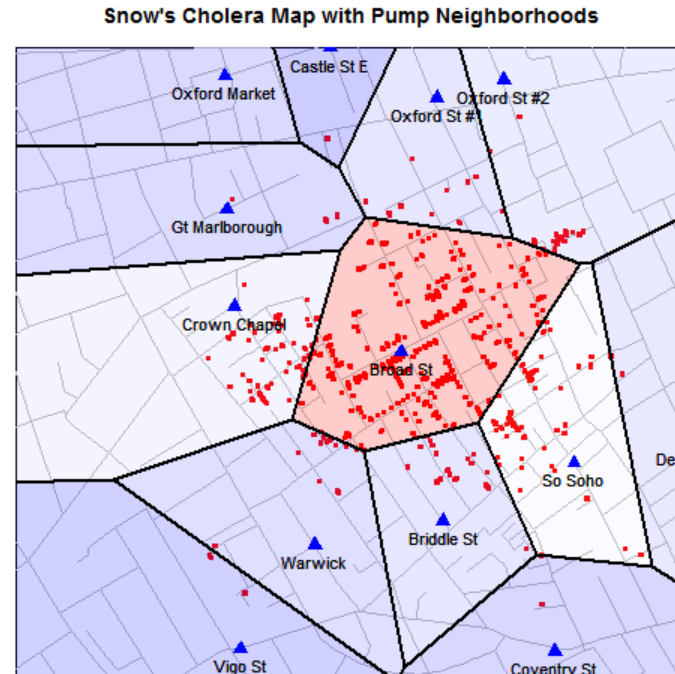
Contours of death densities are calculated using a 2d binned kernel density estimate, [bkde2D\(\)](#) from the [KernSmooth](#) package

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.

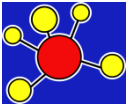


```
SnowMap(density=TRUE,  
main="Snow's Cholera Map with Pump Neighborhoods")
```

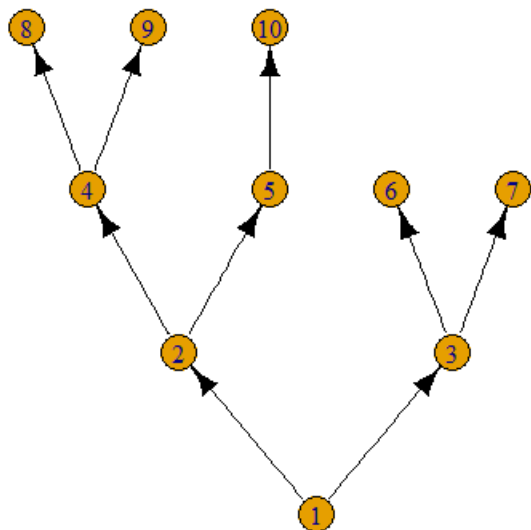
Neighborhoods are the Voronoi polygons of the map closest to each pump, calculated using the [deldir](#) package.

Diagrams: Trees & Graphs

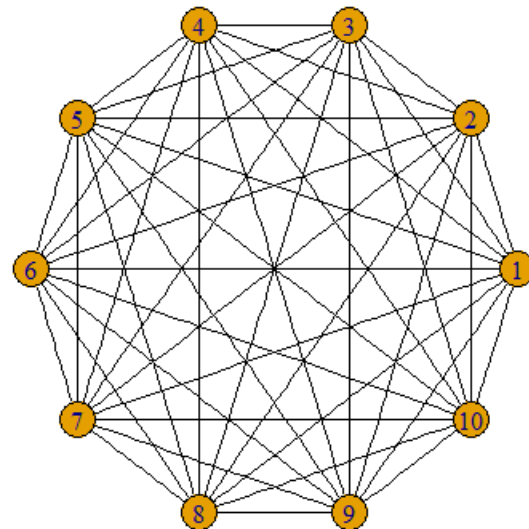
A number of R packages are specialized to draw particular types of diagrams.
igraph is designed for network diagrams of nodes and edges



```
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)
```



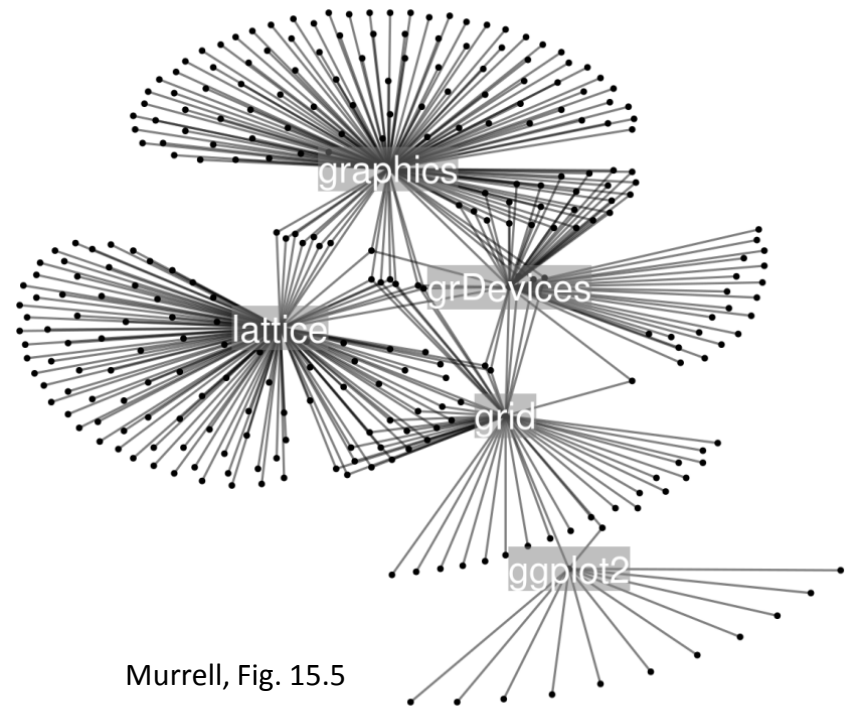
Diagrams: Network diagrams

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

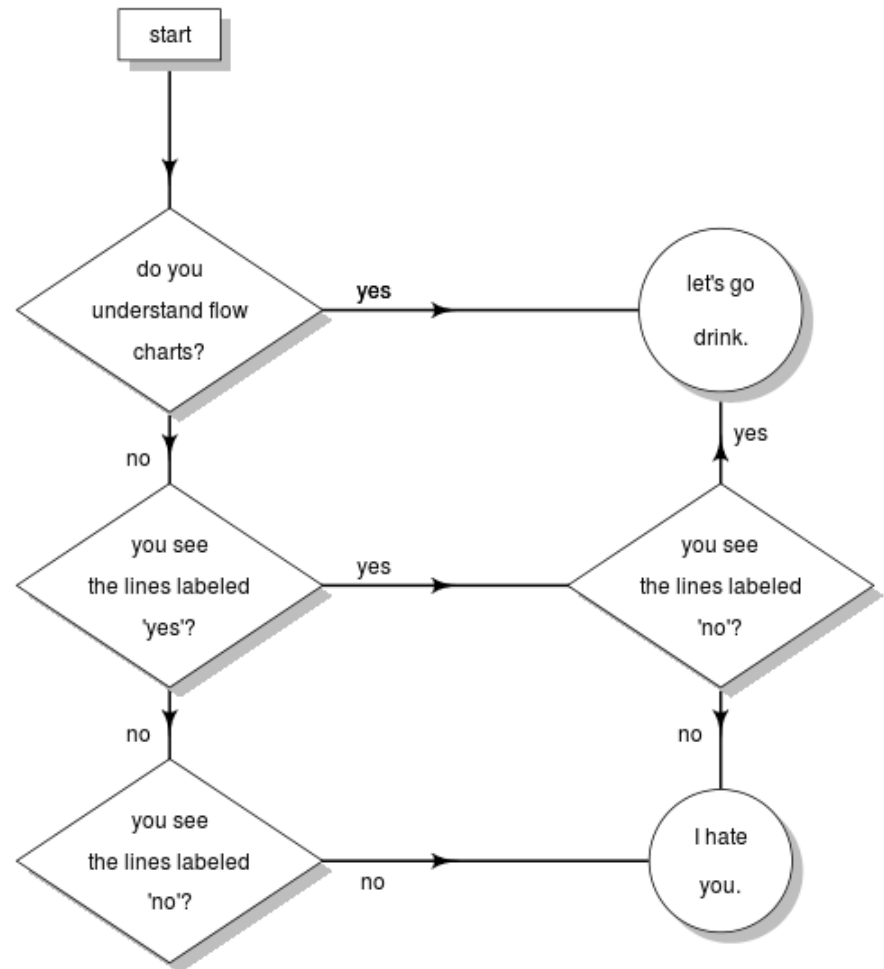


Murrell, Fig. 15.5

Diagrams: Flow charts

The [diagram](#) package:

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

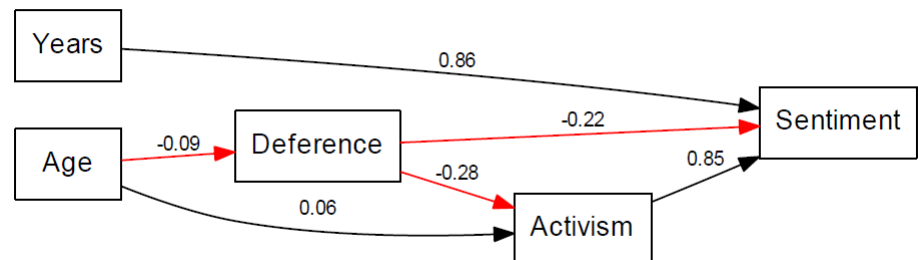


Flow chart about understanding flow charts (after <http://xkcd.com/518>). From: Murrell, Fig 15.10

Path diagrams: structural equation models

Similar diagrams are used to display structural equation models as “path diagrams”
The `sem` and `laavan` 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

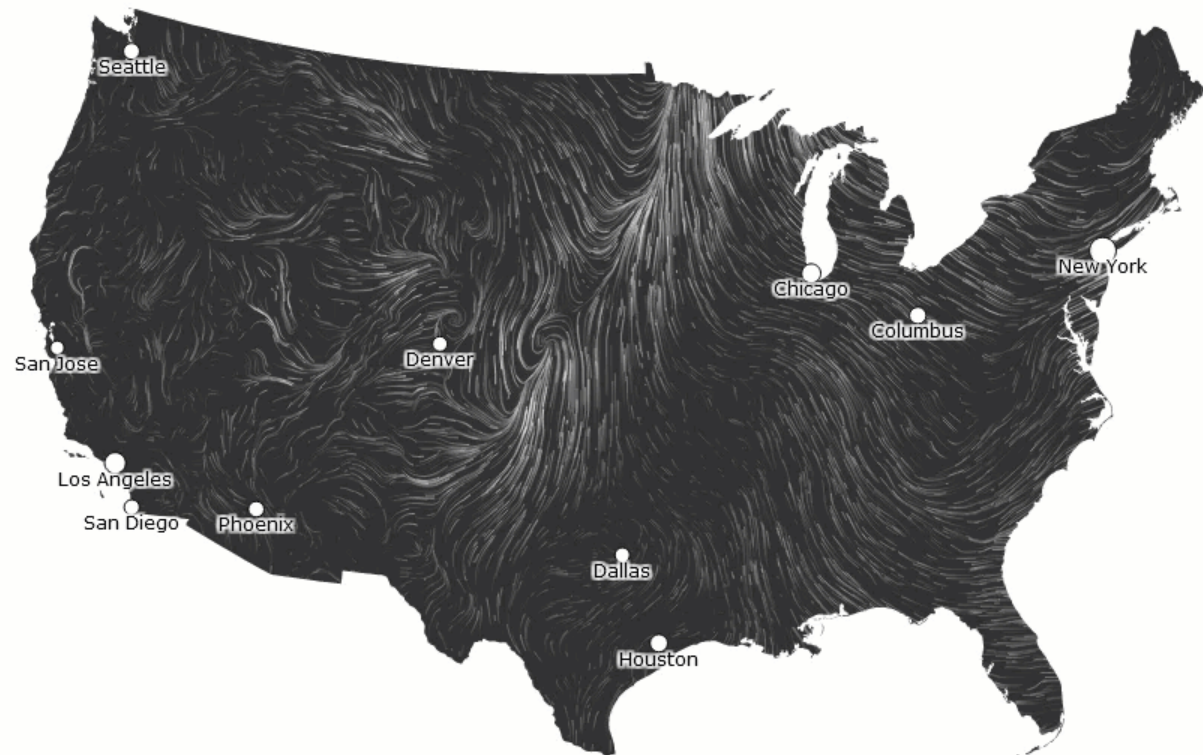
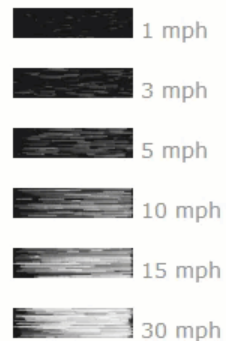
Sept. 24, 2018

11:22 am EST

(time of forecast download)

top speed: **31.3 mph**

average: **7.9 mph**



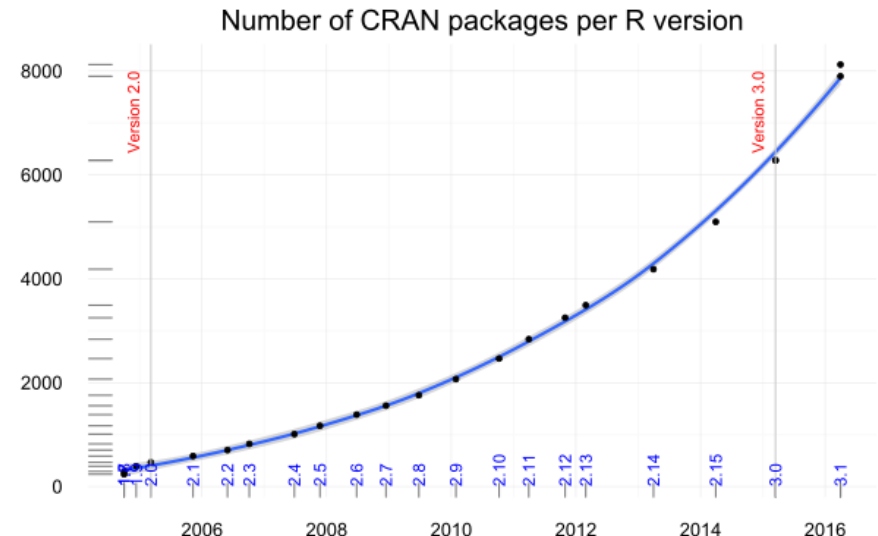
Web scraping: CRAN package history

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

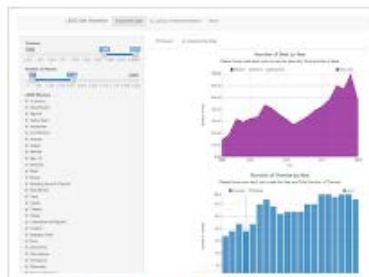
The [Shiny User Showcase](#) contains an inspiring set of sophisticated apps developed and contributed by Shiny users.



Genome browser



Papr



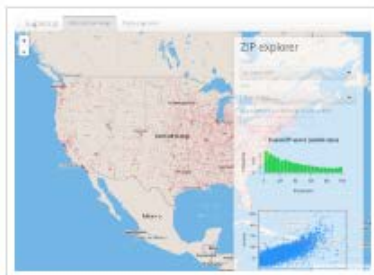
Lego Set Database Explorer



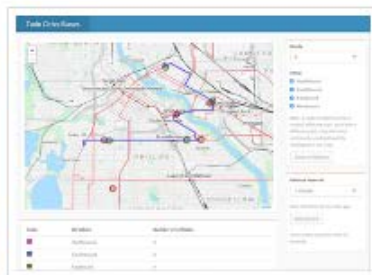
See more

Interactive visualizations

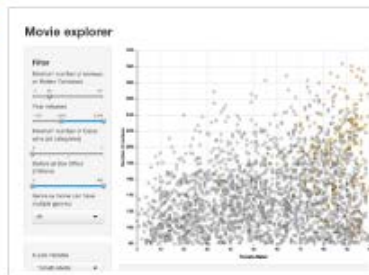
Shiny is designed for fully interactive visualization, using JavaScript libraries like [d3](#), [Leaflet](#), and [Google Charts](#).



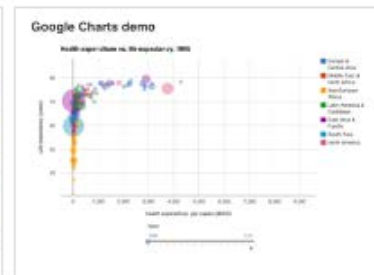
SuperZip example



Bus dashboard



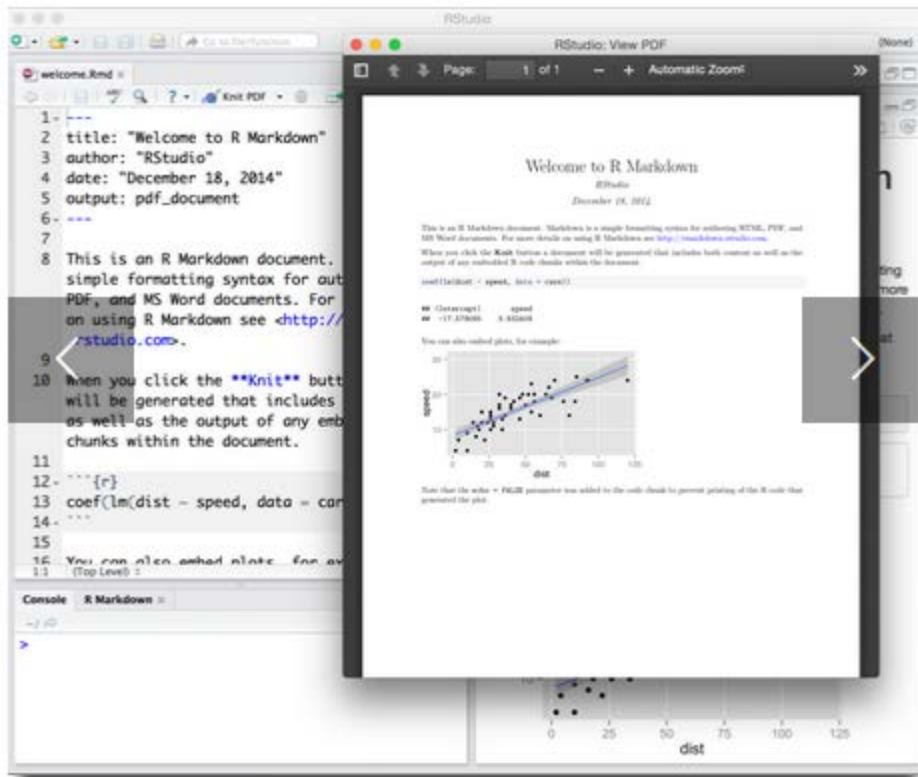
Movie explorer



Google Charts

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

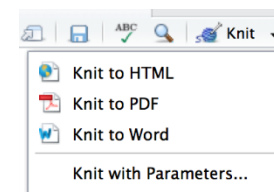
Reproducible analysis & reporting



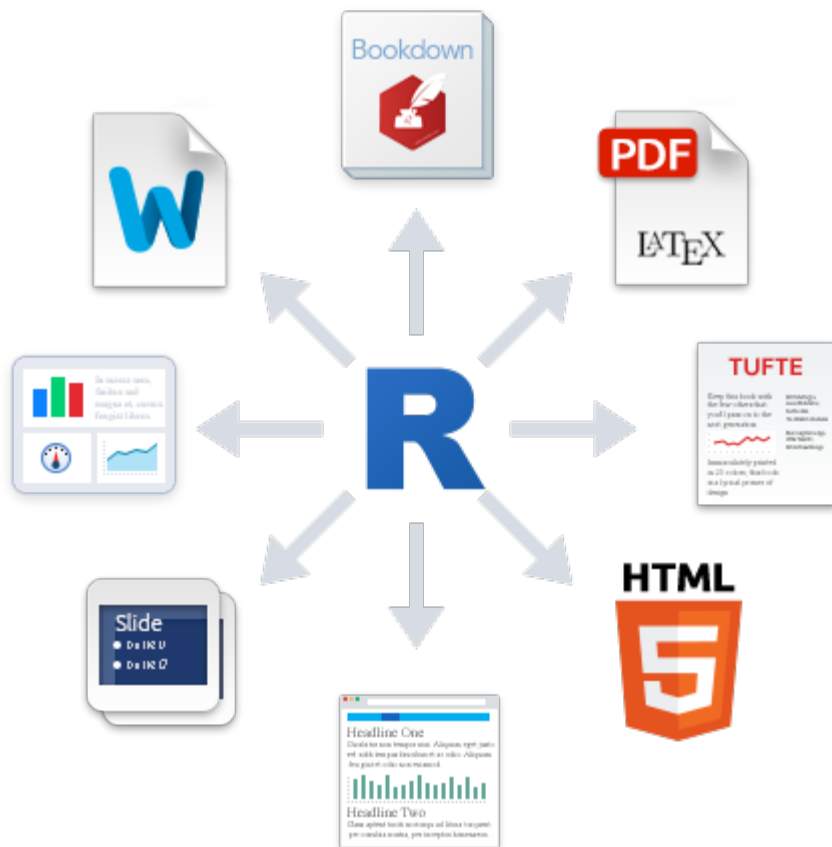
R Studio, together with the knitr and rmarkdown packages provide an easy way to combine writing, analysis, and R output into complete documents

.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:

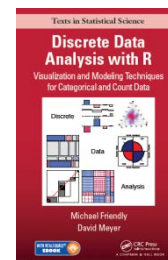


Output formats and templates



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.

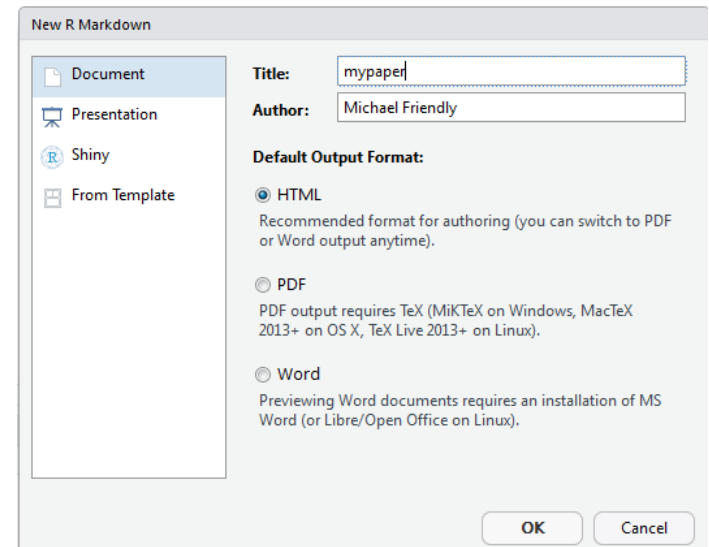
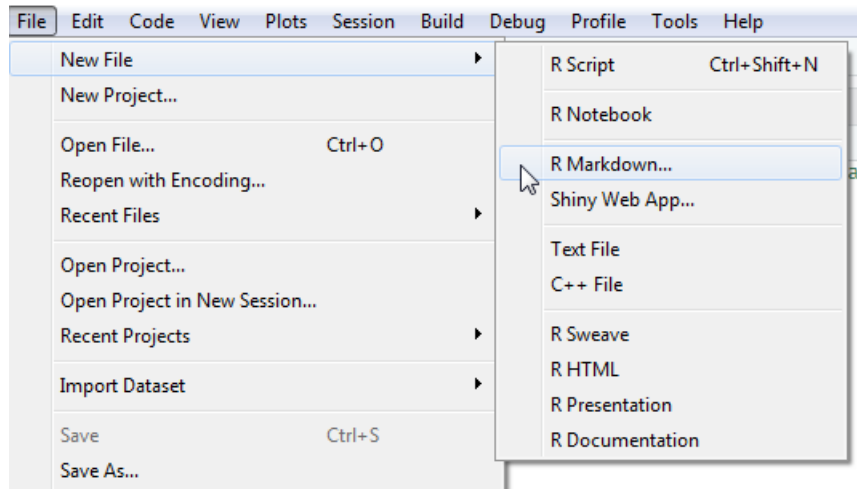


The [bookdown](#) package makes it easier to manage a book-length project – TOC, fig/table #s, cross-references, etc.

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

```
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
15 details on using R Markdown see <http://rmarkdown.rstudio.com>.
16
17 when you click the Knit button a document will be generated
18 R code chunks within the document. You can embed an R code chunk
19
20 ```{r cars}
21 summary(cars)
22 ```
23
24 ## Including Plots
25
26 You can also embed plots, for example:
27
28 ```{r pressure, echo=FALSE}
29 plot(pressure)
30 ```
```

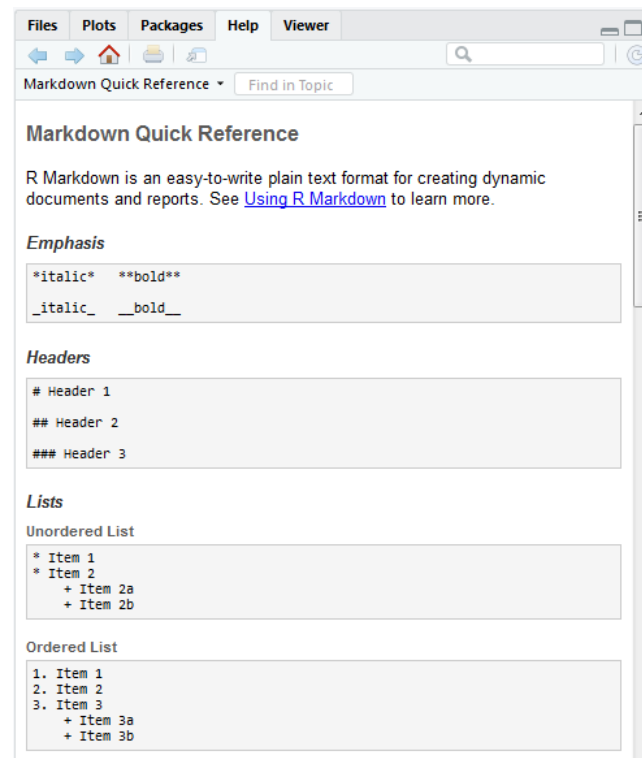
yaml header

Header 2

output code chunk

plot code chunk

Help -> Markdown quick reference



rmarkdown basics

rmarkdown uses simple markdown formatting for all standard document elements

The image displays two windows side-by-side, illustrating the conversion of R Markdown source code to a rendered HTML document.

Left Window (Source Code): The file is named `example.Rmd`. The code includes:

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

Right Window (Rendered HTML): The file is named `example.html`. The rendered output shows:

- A large heading: **Header 1**
- Paragraph: `This is an R Markdown document. Markdown is a simple formatting syntax for authoring web pages.`
- Text: `Use an asterisk mark to provide emphasis, such as italics or bold.`
- Text: `Create lists with a dash:`
- List:
 - Item 1
 - Item 2
 - Item 3
- Code block (preformatted text):

```
Use back ticks to
create a block of code
```
- Text: `Embed LaTeX or MathML equations, $\frac{1}{n} \sum_{i=1}^n x_i$`
- Text: `Or even footnotes, citations, and a bibliography. 1`
- Footnote:

1. Markdown is great.↩

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 'chunks.Rmd' file is open, displaying an R code chunk. The chunk starts with a title 'R Code Chunks' and a description. The R code includes a comment, a library call for 'ggplot2', a summary of the 'cars' dataset, and a plot using 'qplot' with 'geom_smooth()'. A red arrow points from the code line 'qplot(speed, dist, data=cars) + geom_smooth()' to the rendered output in the preview window. The preview window, titled 'RStudio: Preview HTML', shows the rendered HTML. It includes the title 'R Code Chunks', the description, the R code, the summary of the 'cars' dataset, and the plot. The plot is a scatter plot of 'dist' vs 'speed' with a blue smoothing line and a grey confidence interval. A red arrow points from the code line 'qplot(speed, dist, data = cars) + geom_smooth()' to the plot.

```
1 R Code Chunks
2 =====
3
4 With R Markdown, you can insert R code
5 chunks including plots:
6
7 ```{r qplot, fig.width=4, fig.height=3,
8   message=FALSE}
9 # quick summary and plot
10 library(ggplot2)
11 summary(cars)
12 qplot(speed, dist, data=cars) +
13   geom_smooth()
14 ```
```

R Code Chunks

With R Markdown, you can insert R code chunks including plots:

```
# quick summary and plot
library(ggplot2)
summary(cars)
```

##	speed	dist
##	Min. : 4.0	Min. : 2
##	1st Qu.: 12.0	1st Qu.: 26
##	Median : 15.0	Median : 36
##	Mean : 15.4	Mean : 43
##	3rd Qu.: 19.0	3rd Qu.: 56
##	Max. : 25.0	Max. : 120

```
qplot(speed, dist, data = cars) + geom_smooth()
```

dist

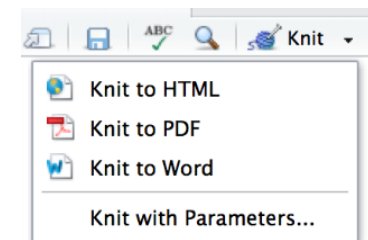
speed

An R chunk:

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

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

.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 rerun 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.

Workflow

- 1 Open a new .Rmd file: at File > New File > R Markdown. Use the wizard that opens to pre-populate the file with a template.
- 2 Write document by editing template.
- 3 Knit document to create report. Use knit button or `render()` to knit.
- 4 Preview Output in IDE window.
- 5 Publish (optional) to web or server. Sync publish button to accounts at:
 - GitHub.com
 - shinyapps.io
 - RStudio Connect
 Reload document. Find in document. File path to output document.
- 6 Examine build log in R Markdown console.
- 7 Use output file that is saved alongside .Rmd.

.Rmd structure

Optional section of header in g. pandoc options written as key-value pairs (YAML).

- At start of file
- Between lines of

Text

Narration formatted with markdown, mixed with

Code chunks

Chunks of embedded code. Each chunk

- Begins with ````{r}`
- Ends with `````

R Markdown will run the code and append the results to the doc.

It will use the location of the .Rmd file as the working directory.

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.

```

output: html_document
runtime: shiny
    
```

How many cars?

speed dist

| speed | dist |
|-------|-------|
| 4.40 | 2.88 |
| 4.60 | 10.00 |
| 4.80 | 4.40 |
| 4.80 | 22.00 |
| 5.00 | 16.00 |

Embed a complete app into your document with `shiny::shinyAppDir()`

* Your report will rendered as a Shiny app, which means you must choose an html output format, save `html_document`, and serve it with an active R Session.

Embed code with knitr syntax

Inline code

Insert with `<code>`. Results appear as text without code.

Built with `<code>getVersion()`

Built with 3.2.3

Code chunks

One or more lines surrounded with ````{r}` and `````. Place chunk options within curly braces, after `<code>` insert with

Global options

Set with knitr `opts_chunk$set()` e.g.

```

{r knitr=FALSE}
knitr::opts_chunk$set(cache = TRUE)
    
```

Important chunk options

- cache** - cache results for future knits (default = FALSE)
- cache.path** - directory to save cached results in (default = "cache/")
- child** - files to knit and then include (default = NULL)
- collapse** - collapse all output into single block (default = FALSE)
- comment** - preface for each line of results (default = "##")
- dependson** - chunk dependencies for caching (default = NULL)
- echo** - Display code in output document (default = TRUE)
- engine** - code language used in chunk (default = "R")
- error** - Display error messages in doc (TRUE) or stop render when errors occur (FALSE) (default = TRUE)
- eval** - Run code in chunk (default = TRUE)
- fig.align** - 'left', 'right', or 'center' (default = 'center')
- fig.cap** - figure caption as character string (default = NULL)
- fig.height**, **fig.width** - Dimensions of plots in inches
- highlight** - highlight source code (default = TRUE)
- include** - include chunk in doc after running (default = TRUE)
- message** - display code messages in document (default = TRUE)
- results** - 'asis' - passthrough results; 'hide' - do not display results; 'hold' - put all results below all code (default = 'markup')
- tidy** - tidy code for display (default = FALSE)
- warning** - display code warnings in document (default = TRUE)

Options not listed above: R options, asis, autolink, background, cache.comments, cache.lazy, cache.rebuild, cache.args, dir, div.args, div, engine.opts, engine.path, fig.asp, fig.env, fig.out, fig.keep, fig.on, fig.path, fig.prio, fig.preview, fig.retina, fig.size, fig.show, fig.showtext, fig.width, internal, last.extra, and knitr.out.width, knit.opts, knit.opts.reload, knit.opts, knit.opts, knit.opts

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()`.

```

params:
  as: LAG
  d: Sys.Date()
    
```

Today's date is: `params$d`

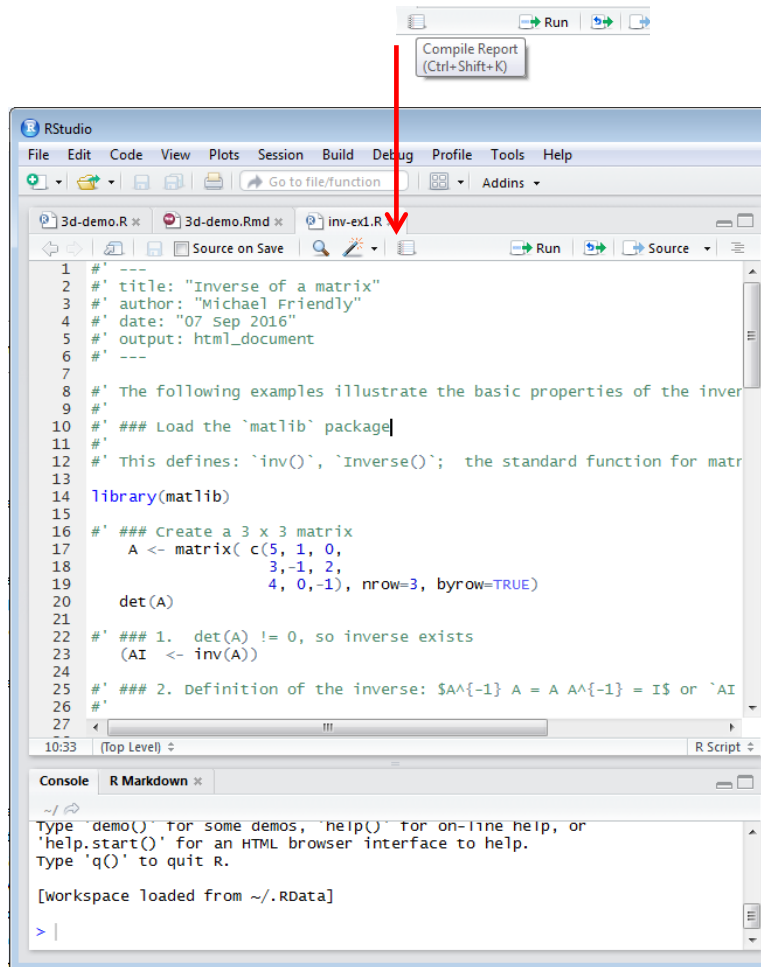
Render with Parameters:

```

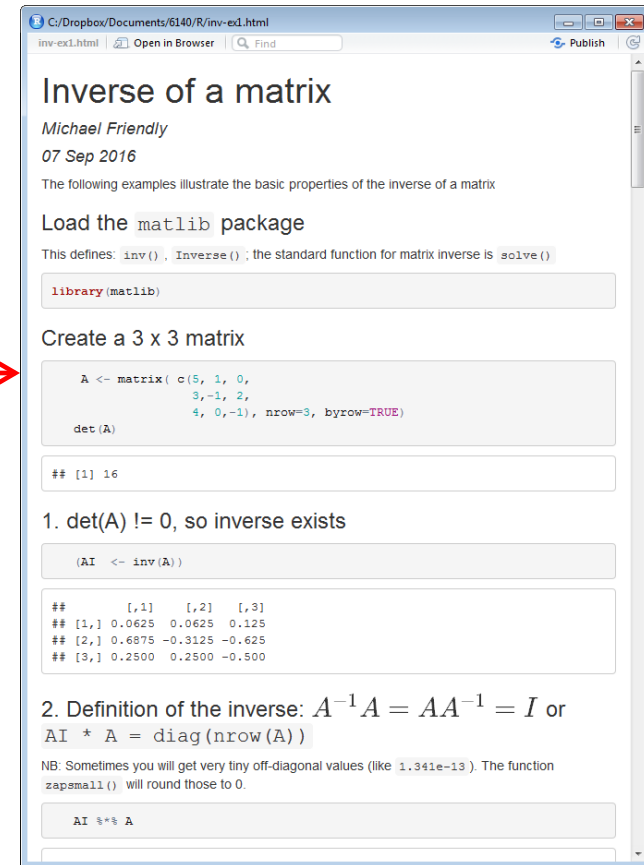
render("doc.Rmd",
  params = list(n = L, d = as.Date("2015-01-01"))
    
```

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



```
1 #' ---
2 #' title: "Inverse of a matrix"
3 #' author: "Michael Friendly"
4 #' date: "07 Sep 2016"
5 #' output: html_document
6 #' ---
7
8 #' The following examples illustrate the basic properties of the inverse
9 #'
10 #' ### Load the `matlib` package
11 #'
12 #' This defines: `inv()`, `Inverse()`; the standard function for matrix inverse is `solve()`
13
14 library(matlib)
15
16 #' ### Create a 3 x 3 matrix
17 A <- matrix(c(5, 1, 0,
18             3, -1, 2,
19             4, 0, -1), nrow=3, byrow=TRUE)
20
21 det(A)
22
23 #' ### 1. det(A) != 0, so inverse exists
24 (AI <- inv(A))
25
26 #' ### 2. Definition of the inverse:  $AA^{-1} = A^{-1}A = I$  or  $AI = I$  or  $AI = I$ 
27
```



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 $AI = I$ or $AI = I$

```
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?