Simulation and Performance Evaluation

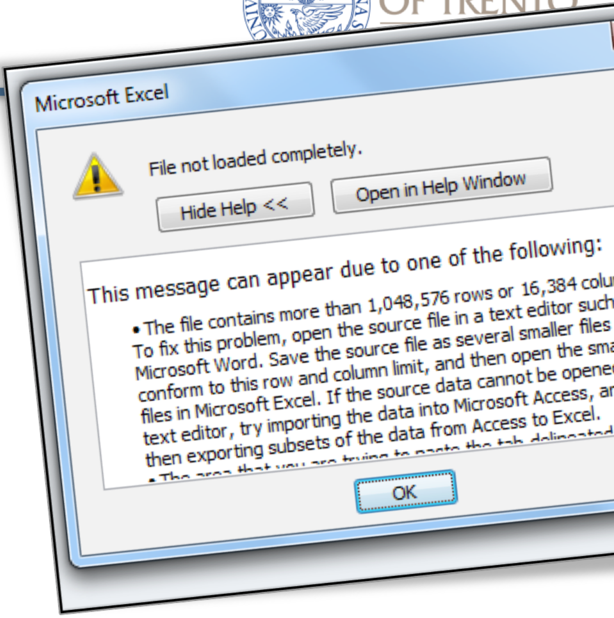
Basic R Usage

Michele Segata, Renato Lo Cigno

Original version taken from the Network Simulation course held at the University of Paderborn, Germany
Special thanks to Falko Dressler and Christoph Sommer

Motivation

- How to analyze large amounts of data?
  - SAS
  - SPSS
  - Stata
  - Matlab
  - SciPy (or Pandas, or …)
  - SQL
  - Excel
  - …
  - R



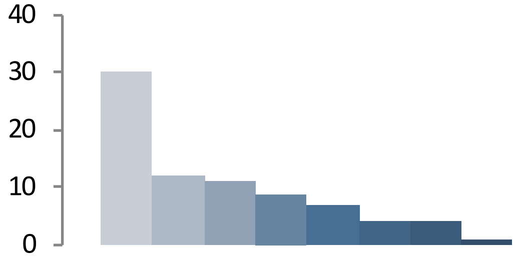
File not loaded completely.

This message can appear due to one of the following:

- The file contains more than 1,048,576 rows or 16,384 colu
To fix this problem, open the source file in a text editor such
Microsoft Word. Save the source file as several smaller files
conform to this row and column limit, and then open the sma
files in Microsoft Excel. If the source data cannot be opene
text editor, try importing the data into Microsoft Access, ar
then exporting subsets of the data from Access to Excel.

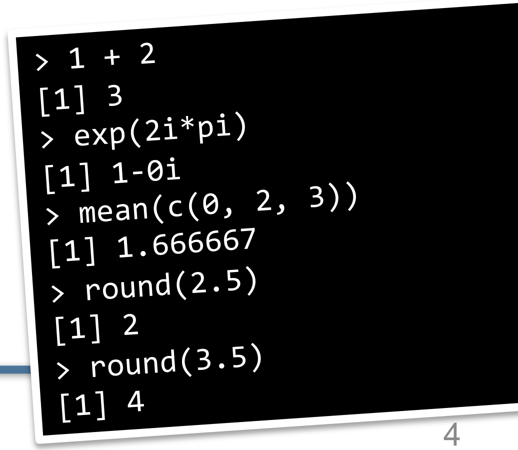
Motivation

- Why R?
  - Oldest
    - Implementation of S, designed at Bell Labs starting 1975
  - Newest
    - Actively maintained by international team
    - Version 3.4.4 released in March 2018
    - Version 3.5.3 available since March 2019
  - Open Source
    - Available for every major OS
  - Popular
    - Consistently ranks among top 3 software tools for data analysis



About

- What is R?
  - the computer language R
  - an interpreter of code written in R
  - a execution environment that contains the R interpreter
- From Wikipedia:
  - "R is an implementation of the S programming language combined with lexical scoping semantics inspired by Scheme. S was created by John Chambers while at Bell Labs. R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team, of which Chambers is a member. R is named partly after the first names of the first two R authors and partly as a play on the name of S."
- R is a (kind of) scripting language
  - Automatic variables, garbage collection
  - Lots of "smart" shortcuts (for better or worse)



```
> 1 + 2
[1] 3
> exp(2i*pi)
[1] 1-0i
> mean(c(0, 2, 3))
[1] 1.666667
> round(2.5)
[1] 2
> round(3.5)
[1] 4
```

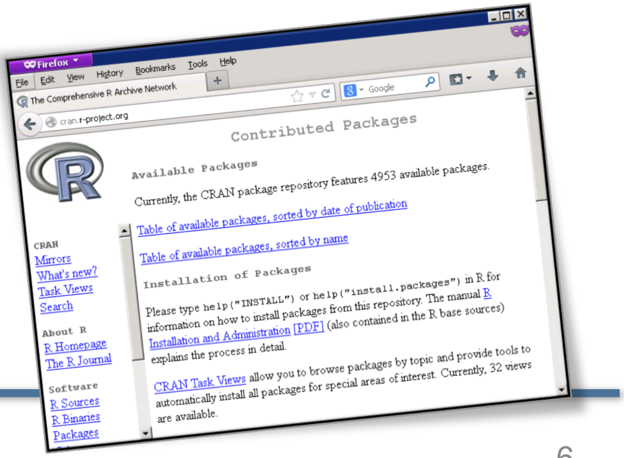
Installing R

- To install R, you can do the following depending on your OS:
  - **Linux:** Just open your terminal and type
    - sudo apt-get install r-base
  - **Mac OS X:** If you have MacPorts installed, open your terminal and type
    - sudo port install R
  - **Mac OS X:** If you don't have MacPorts, you can download R binaries from the official R website (http://www.r-project.org/)
  - **Windows:** Download the binaries from the official R website

Running R

- Frontend
  - Official frontend: www.r-project.org
    - R
    - Rscript (to run R scripts from command line)
  - Or any of
    - RStudio, Revolutions, Tinn-R, Deducer, RKward, R Commander, Vim R, …

- Packages
  - Extend R functionality
  - Written in C, Java, Fortran, or R
  - Official repository: cran.r-project.org
    - Comprehensive R Archive Network (CRAN)
  - Or any of
    - Bioconductor, Crantastic, R-Forge, …



Contributed Packages

Available Packages

Currently, the CRAN package repository features 4953 available packages.

Table of available packages, sorted by date of publication

Table of available packages, sorted by name

Installation of Packages

Please type `help("INSTALL")` or `help("install.packages")` in R for information on how to install packages from this repository. The manual R Installation and Administration [PDF] (also contained in the R base sources) explains the process in detail.

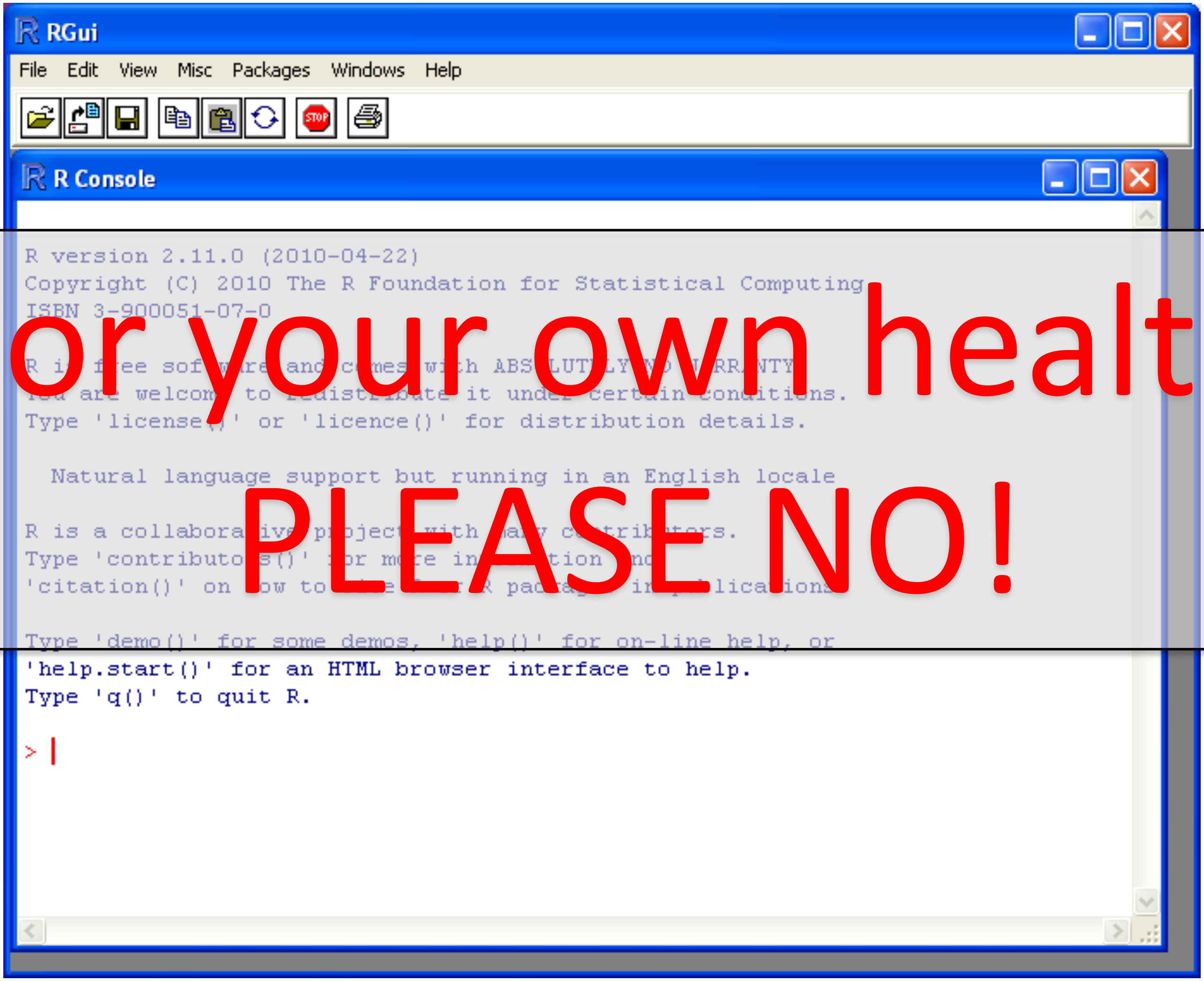
CRAN Task Views allow you to browse packages by topic and provide tools to automatically install all packages for special areas of interest. Currently, 32 views are available.

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages

R Frontends – Standard R Console

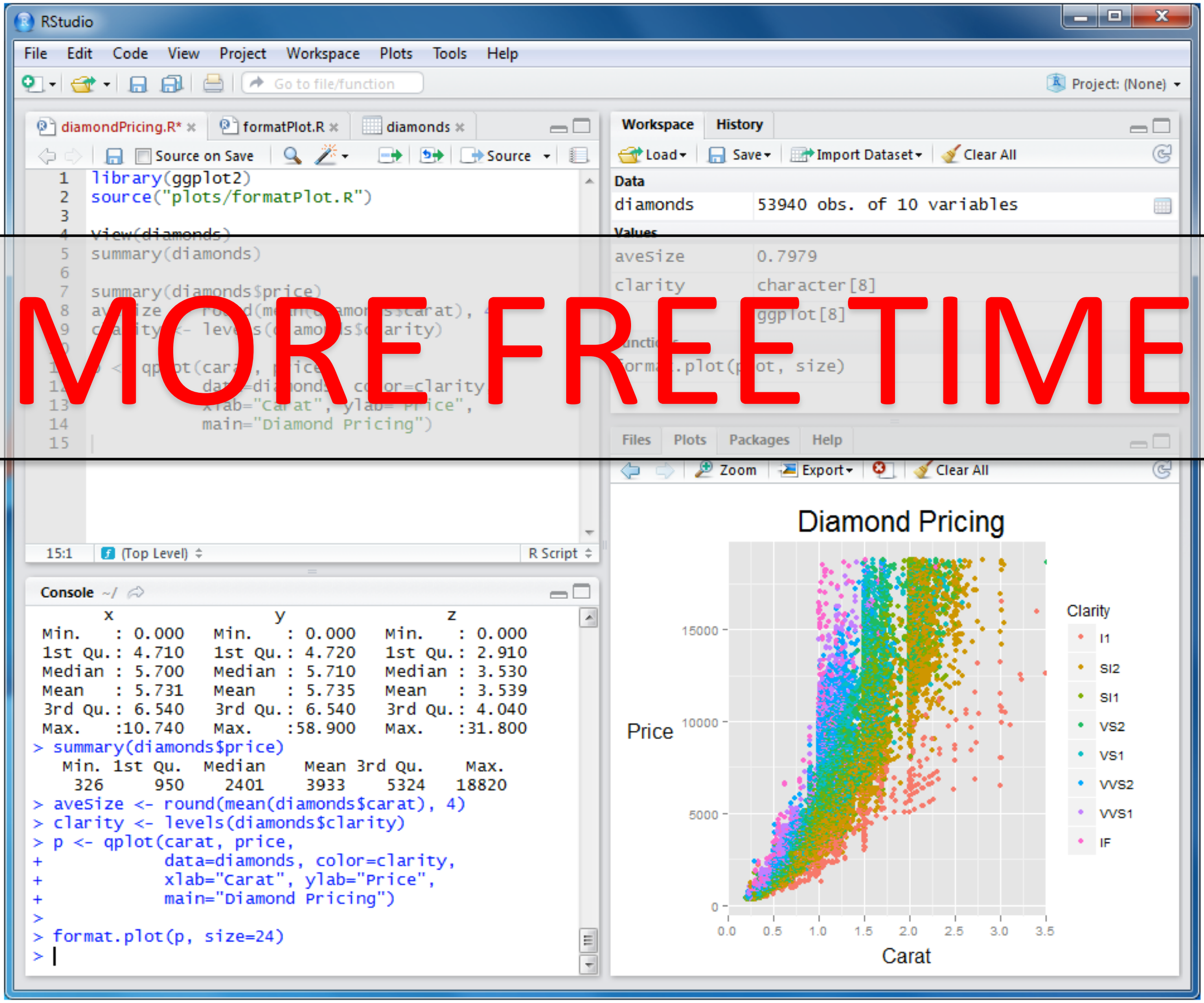


```
R version 2.11.0 (2010-04-22)
Copyright (C) 2010 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

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.

  Natural language support but running in an English locale

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.

> 
```

For your own health:

PLEASE NO!

R Frontends - RStudio



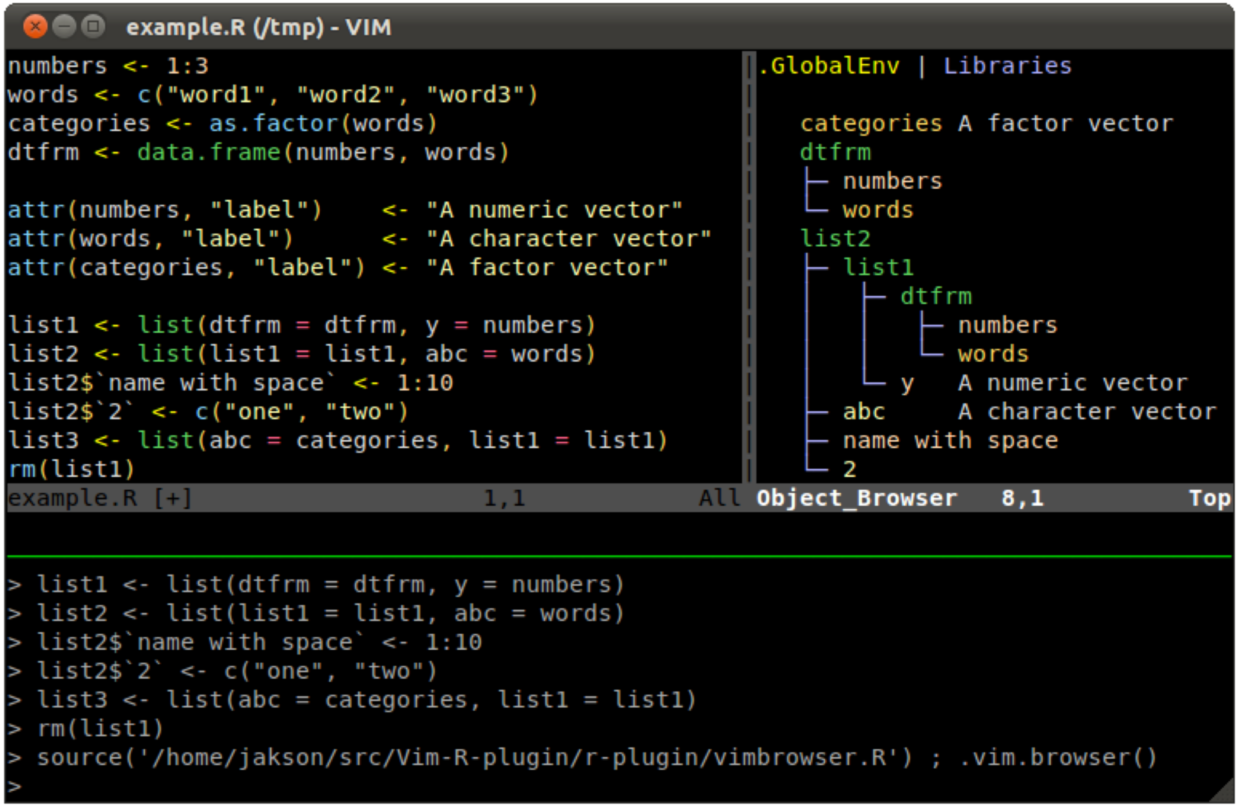
```
library(ggplot2)
source("plots/formatPlot.R")

View(diamonds)
summary(diamonds)

summary(diamonds$price)
aveSize <- round(mean(diamonds$carat), 4)
clarity <- levels(diamonds$clarity)
p <- qplot(carat, price,
           data=diamonds, color=clarity,
           xlab="Carat", ylab="Price",
           main="Diamond Pricing")
```

```
Data
diamonds     53940 obs. of 10 variables
Values
aveSize      0.7979
clarity      character[8]
p            ggplot[8]
Functions
format.plot(plot, size)
```

```
          x                   y                   z
 Min.   : 0.000     Min.   : 0.000     Min.   : 0.000
 1st Qu.: 4.710     1st Qu.: 4.720     1st Qu.: 2.910
 Median : 5.700     Median : 5.710     Median : 3.530
 Mean   : 5.731     Mean   : 5.735     Mean   : 3.539
 3rd Qu.: 6.540     3rd Qu.: 6.540     3rd Qu.: 4.040
 Max.   :10.740     Max.   :58.900     Max.   :31.800
> summary(diamonds$price)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    326     950    2401    3933    5324   18820
> aveSize <- round(mean(diamonds$carat), 4)
> clarity <- levels(diamonds$clarity)
> p <- qplot(carat, price,
+            data=diamonds, color=clarity,
+            xlab="Carat", ylab="Price",
+            main="Diamond Pricing")
>
> format.plot(p, size=24)
> 
```

MORE FREE TIME!

R Frontends – Vim R Plugin



```
numbers <- 1:3
words <- c("word1", "word2", "word3")
categories <- as.factor(words)
dtfrm <- data.frame(numbers, words)

attr(numbers, "label")    <- "A numeric vector"
attr(words, "label")      <- "A character vector"
attr(categories, "label") <- "A factor vector"

list1 <- list(dtfrm = dtfrm, y = numbers)
list2 <- list(list1 = list1, abc = words)
list2$`name with space` <- 1:10
list2$`2` <- c("one", "two")
list3 <- list(abc = categories, list1 = list1)
rm(list1)
```

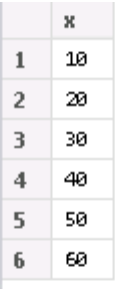
```
.GlobalEnv | Libraries

   categories A factor vector
   dtfrm
   ├─ numbers
   └─ words
   list2
   ├─ list1
   │   ├─ dtfrm
   │   │   ├─ numbers
   │   │   └─ words
   │   └─ y    A numeric vector
   ├─ abc      A character vector
   ├─ name with space
   └─ 2
```

```
example.R [+]                  1,1           All Object_Browser    8,1           Top
```

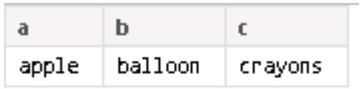
```
> list1 <- list(dtfrm = dtfrm, y = numbers)
> list2 <- list(list1 = list1, abc = words)
> list2$`name with space` <- 1:10
> list2$`2` <- c("one", "two")
> list3 <- list(abc = categories, list1 = list1)
> rm(list1)
> source('/home/jakson/src/Vim-R-plugin/r-plugin/vimbrowser.R') ; .vim.browser()
>
```

Data in R

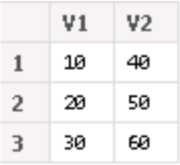
- Assigning data
  - x <- 1
  - x = 1 also works (not always)
  - x <- (1+2)*NA*3 ⇨ x <- NA
- Creating data of specific type and class
  - v <- c(10, 20, 30, 40, 50, 60)
  - l <- list(a="apple", b="balloon", c="crayons")
  - m <- matrix(v, nrow=3)
  - d <- data.frame(student=c("alice", "bob", "carol"), grade=c(1,2,3))



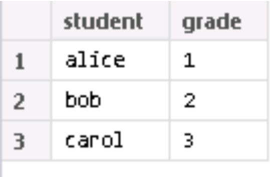
	x
1	10
2	20
3	30
4	40
5	50
6	60



a	b	c
apple	balloon	crayons



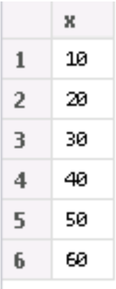
	V1	V2
1	10	40
2	20	50
3	30	60



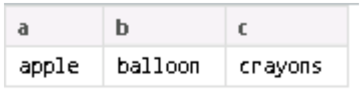
	student	grade
1	alice	1
2	bob	2
3	carol	3

Data in R

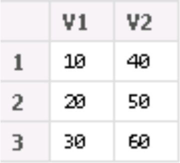
- Printing
  - print(v) ⇨ [1] 10 20 30 40 50 60
  - summary(v) ⇨ Minimum: 10.0   1st Quartile: 32.5   Median: 45.0 …
  - …



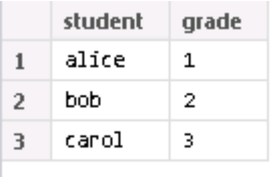
	x
1	10
2	20
3	30
4	40
5	50
6	60



a	b	c
apple	balloon	crayons



	V1	V2
1	10	40
2	20	50
3	30	60



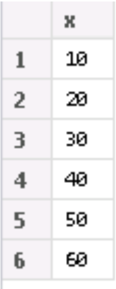
	student	grade
1	alice	1
2	bob	2
3	carol	3

Data in R

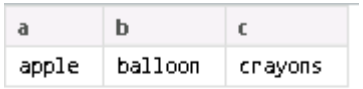
- At its core, an R object is defined by three properties
  - Type
    - homogeneous vector types (logical, integer, double, …)
    - heterogeneous list type (list)
    - miscellaneous types (function, expression, …)
  - Length
    - Only really sensible for vectors and lists
  - Attributes
    - class (matrix, factor, data.frame, …)
      - determines how generic functions interoperate
    - dim (dimensions of matrix)
    - levels (possible values of factor)
    - names, dimnames, rownames, colnames, …
    - comment
    - …

Interacting with Data

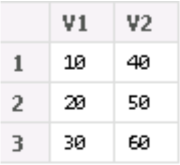
- Subset operator (attn: R is one-based, not zero-based)
  - print(v[c(2, 3, 4)]) ⇨ [1] 20 30 40
  - print(v[2:4]) ⇨ [1] 20 30 40
  - print(v[c(T,F,F,F,F,T)]) ⇨ [1] 10 60
  - print(v[-2]) ⇨ [1] 10 30 40 50 60
  - v[2] <- 99; print(v) ⇨ [1] 10 99 30 40 50 60
  - print(m[1,2]) ⇨ [1] 40
  - print(m[1,]) ⇨ [1] 10 40
  - print(m[,2]) ⇨ [1] 40 50 60



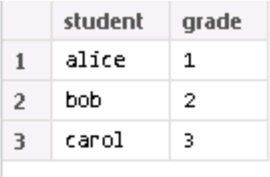
	x
1	10
2	20
3	30
4	40
5	50
6	60



a	b	c
apple	balloon	crayons



	V1	V2
1	10	40
2	20	50
3	30	60

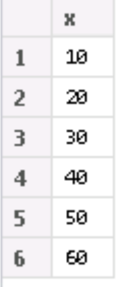


	student	grade
1	alice	1
2	bob	2
3	carol	3

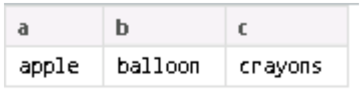
Interacting with Data

- Element operator
  - print(l[["a"]]) ⇨ [1] "apple"
  - print(d[["grade"]]) ⇨ [1] 1 2 3
- CAREFUL: d[[”grade”]] IS DIFFERENT FROM d[”grade”]

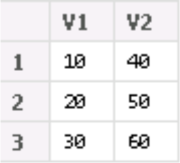
- Name operator
  - print(l$a) ⇨ [1] "apple"
  - print(d$grade) ⇨ [1] 1 2 3



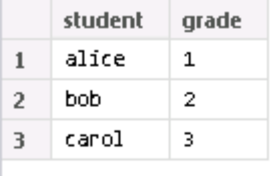
	x
1	10
2	20
3	30
4	40
5	50
6	60



a	b	c
apple	balloon	crayons



	V1	V2
1	10	40
2	20	50
3	30	60



	student	grade
1	alice	1
2	bob	2
3	carol	3

Functions

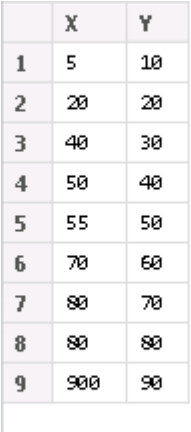
- Defining functions
  - f <- function(x, y=1, offset=0) { offset + x * y }
  - print(f) ⇨ function (x, y = 1, offset = 0) { offset + x * y }

- Calling functions
  - f(3, 7, 1000) ⇨ [1] 1021
  - f(3, 7) ⇨ [1] 21
  - f(offset=1000, x=3, y=7) ⇨ [1] 1021
  - f(3, offset=1000) ⇨ [1] 1003

Built-ins

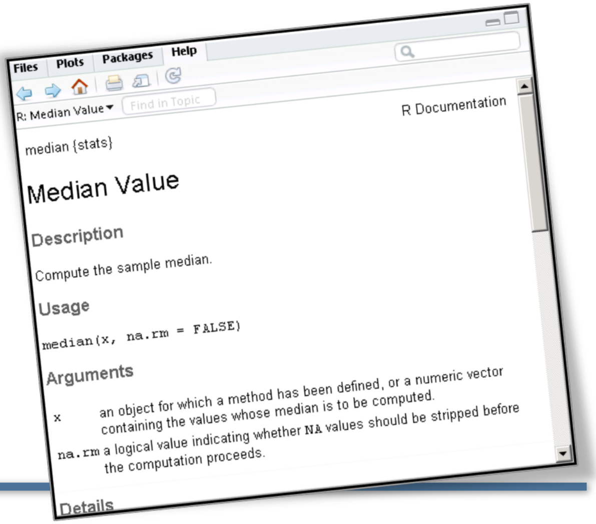
- Basic statistics
  - X <- c(5, 20, 40, 50, 55, 70, 80, 80, 900)
  - Y <- c(10, 20, 30, 40, 50, 60, 70, 80, 90)

  - mean(X) ⇨ 144.4444
  - median(X) ⇨ 55
  - quantile(X, 0.25) ⇨ 40
  - glm.fit(1:9, Y, intercept=F)$coefficients ⇨ 10
  - wilcox.test(X, Y)$p.value ⇨ 0.72



	X	Y
1	5	10
2	20	20
3	40	30
4	50	40
5	55	50
6	70	60
7	80	70
8	80	80
9	900	90

- Help and more information
  - help('mean')
  - ?mean
  - citation('vioplot')



R Documentation

median {stats}

Median Value

Description

Compute the sample median.

Usage

```
median(x, na.rm = FALSE)
```

Arguments

x an object for which a method has been defined, or a numeric vector containing the values whose median is to be computed.

na.rm a logical value indicating whether NA values should be stripped before the computation proceeds.

Details

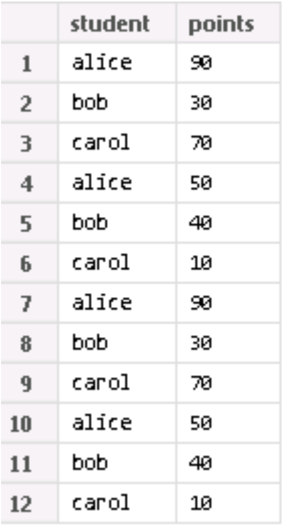
Working with Data Frames

- Creating

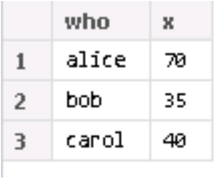
  -
    ```
    d <- data.frame(
      student=rep(c("alice", "bob", "carol"),4),
      points=rep(c(90,30,70,50,40,10), 2)
    )
    ```
- Grouping averages

  -
    ```
    a <- aggregate(
      d$points,
      by=list(who=d$student),
      FUN=mean
    )
    ```
  - or use ddply by plyr (see example)
- Adding a new column

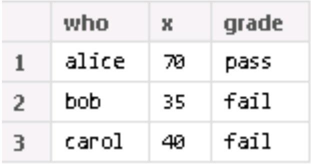
  -
    ```
    a$grade <- ifelse(a$x >= 50, "pass", "fail")
    ```



	student	points
1	alice	90
2	bob	30
3	carol	70
4	alice	50
5	bob	40
6	carol	10
7	alice	90
8	bob	30
9	carol	70
10	alice	50
11	bob	40
12	carol	10



	who	x
1	alice	70
2	bob	35
3	carol	40



	who	x	grade
1	alice	70	pass
2	bob	35	fail
3	carol	40	fail

Installing External Packages

- Syntax:
  - install.packages(“package name”) or
  - install.packages(vector of packages names)
  - e.g.: install.packages(c(“plyr”, “moments”))

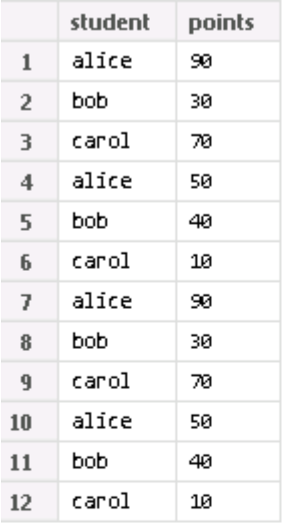
You don't need to manually download, compile, and install packages!

- R will ask you which mirror to use, download, and install the packages
  - with dependencies

- To load your library:
  - library(“plyr”) or require(“plyr”)

Getting Data into R

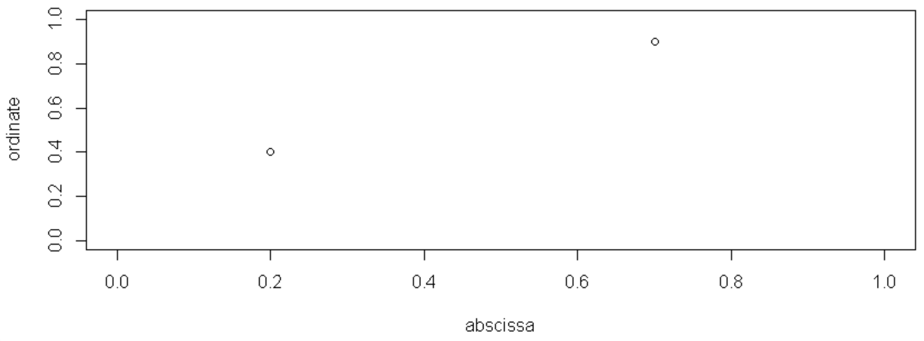
- Read data frame from text file
  - d <- read.csv('data.csv’)
  - d will be a data.frame where column names matches csv column names



	student	points
1	alice	90
2	bob	30
3	carol	70
4	alice	50
5	bob	40
6	carol	10
7	alice	90
8	bob	30
9	carol	70
10	alice	50
11	bob	40
12	carol	10

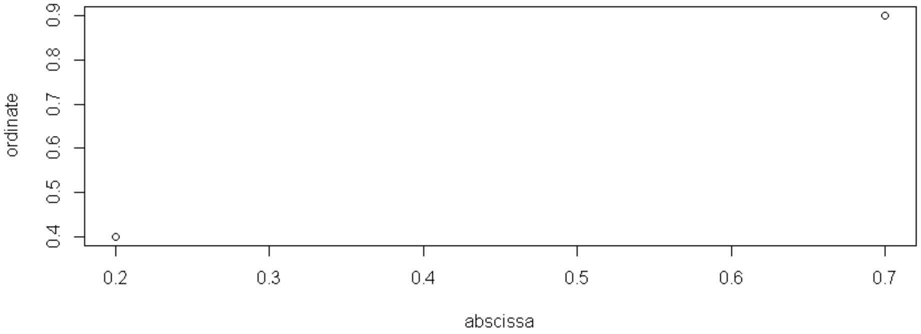
Plotting Data

- R can do plots, too
- Low level plotting commands
  - plot.new()
  - plot.window(xlim=c(0,1), ylim=c(0,1))
  - points(c(0.2, 0.7), c(0.4, 0.9))
  - box()
  - axis(1)
  - axis(2)
  - title(xlab="abscissa", ylab="ordinate")



Plotting Data

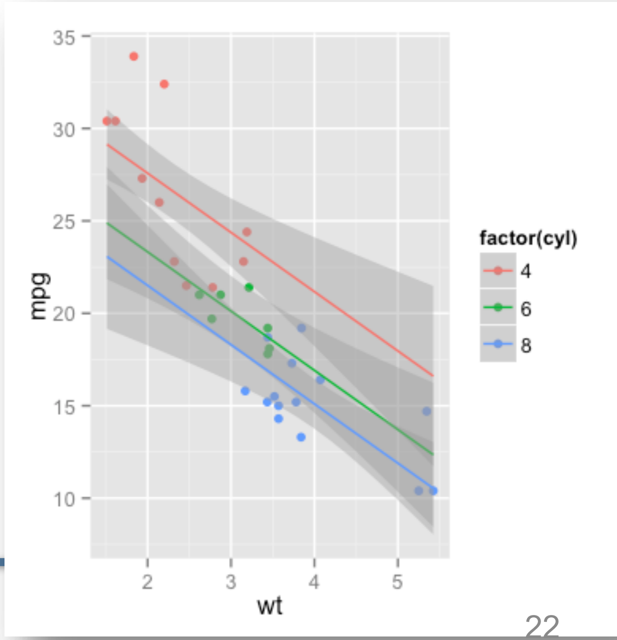
- High level plotting commands
  - generic “plot” command guesses good plot type
  - plot(c(0.2, 0.7), c(0.4, 0.9), xlab="abscissa", ylab="ordinate")
- Can mix high level and low level plotting commands
  - e.g., annotations, lines, points, abline, …



Add-on Packages

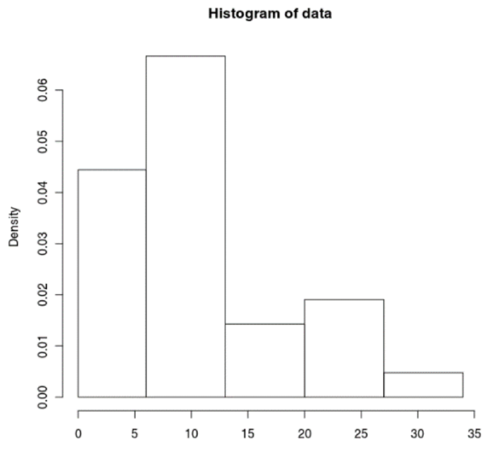
- Tons of packages for basically EVERYTHING
- **Lattice**
  - Good for *conditioning* types of plots (a vs. b depending on c…)
  - Good for assembling many plots into one
- **ggplot2**
  - Grammar-based approach
  - Good for data exploration

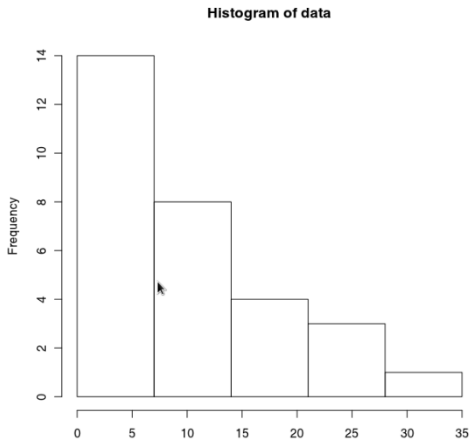
```
ggplot(d, aes(x=wt, y=mpg, colour=cyl))
+ geom_line()
+ geom_point(…)
+ geom_smooth(…)
+ facet_grid(model ~ make)
+ …
```



Histogram

- Illustrates frequencies of discrete intervals (bins)
- Used to plot density of data
- Bin size important
  - May hide data when using inappropriate bin sizes
- data <- c(25, 11, 2, 13, 1, 22, 15, 3, 1, 7, 8, 10, 32, 7, 4, 9, 18, 21, 7, 7, 16, 6, 4, 12, 5, 27, 7, 9, 10, 7)





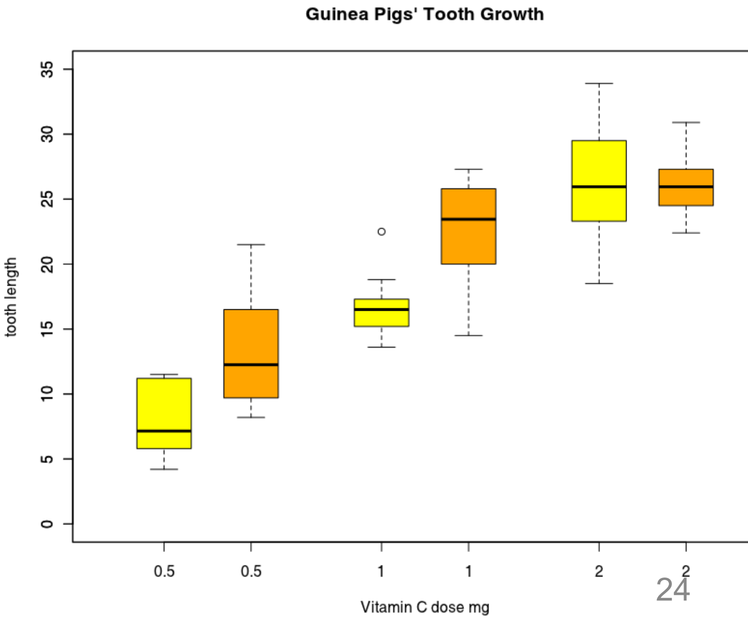
- hist(data, breaks=c(0,6,13,20,27,34)) hist(data, breaks=c(0,7,14,21,28,35))

Boxplot

- Give a quick overview of the following values:
  - Min, 1st quantile, median, 3rd quantile, max
  - Whiskers:
    - extend to the most extreme data point which is no more than 1.5 times the interquartile range from the box
    - absolute minimum and maximum
- Outliers are displayed separately
  - if the minimum or maximum are not within the 1.5 IQR

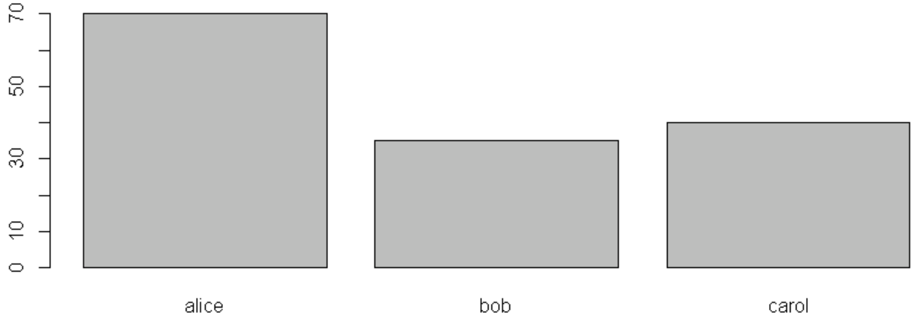
- Advantages:
  - Allows to compare different result sets quickly by using multiple box plots

- Disadvantages:
  - Bimodal distributions cannot be spotted

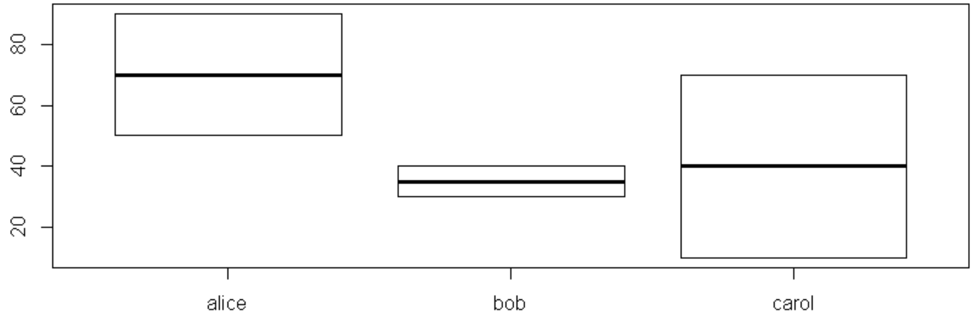


Plotting with R

- Barplot
  - barplot(a$x, names.arg=a$who)

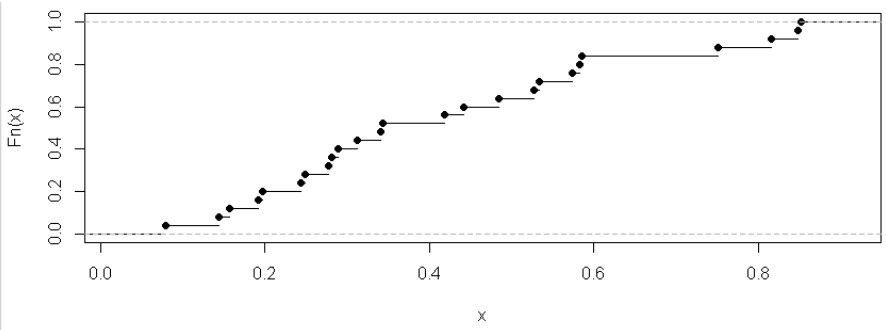


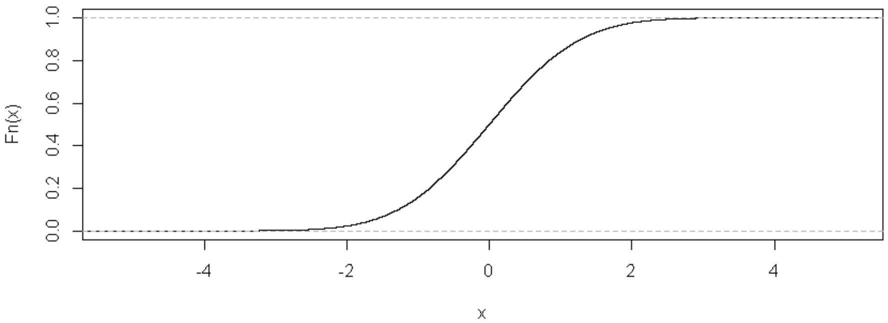
- Boxplot
  - boxplot(d$points ~ d$student)



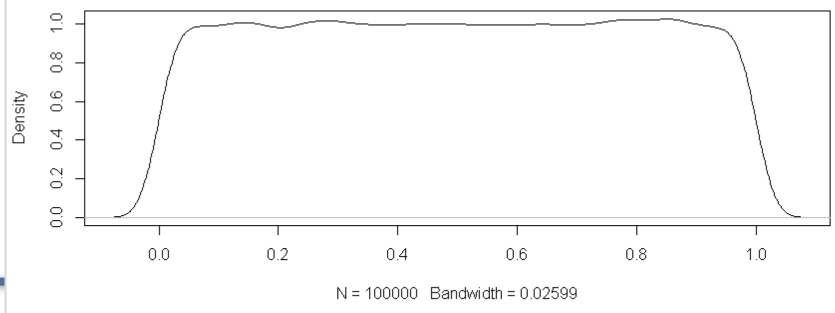
Plotting with R

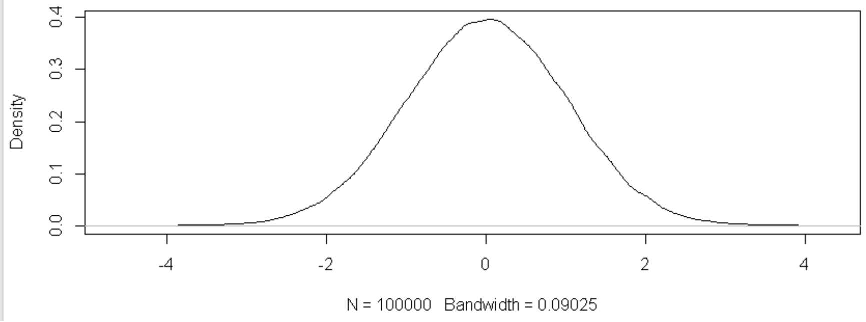
- eCDF
  - plot(ecdf(runif(25, min=0, max=1)))
  - plot(ecdf(rnorm(1e5)))

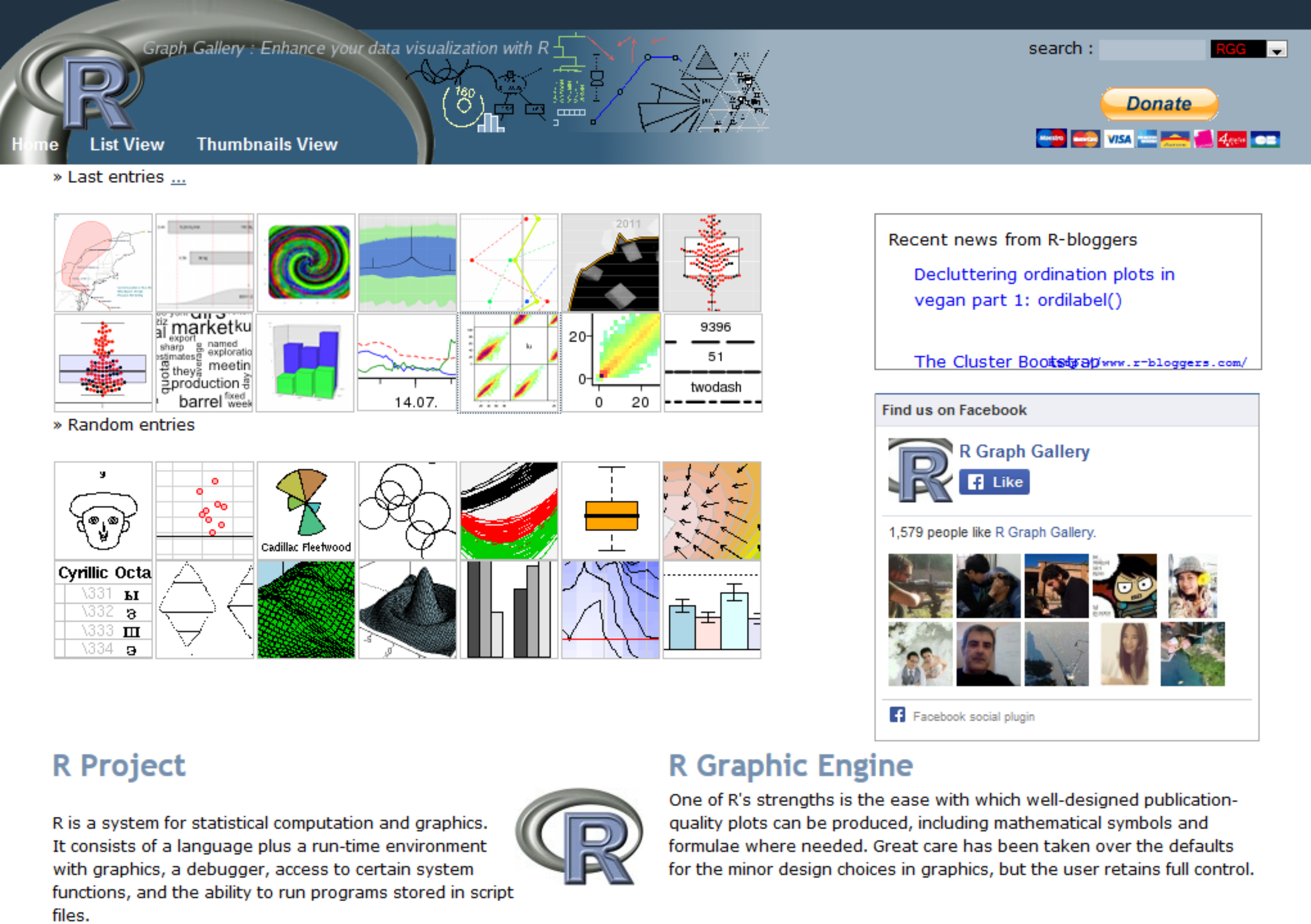




- Kernel Density Estimate
  - plot(density(runif(1e5, min=0, max=1)))
  - plot(density(rnorm(1e5)))







Graph Gallery : Enhance your data visualization with R

search : RGG

Donate

Home List View Thumbnails View

» Last entries ...

» Random entries

Recent news from R-bloggers

Decluttering ordination plots in vegan part 1: ordilabel()

The Cluster Bootstrap http://www.r-bloggers.com/

Find us on Facebook

R Graph Gallery

Like

1,579 people like R Graph Gallery.

Facebook social plugin

R Project

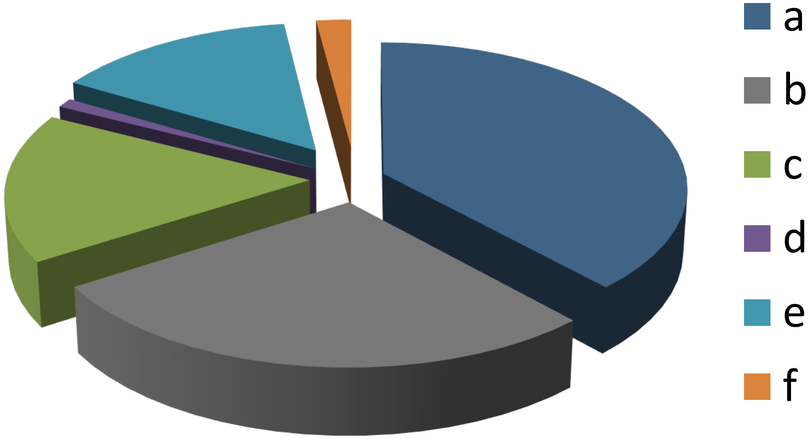
R is a system for statistical computation and graphics. It consists of a language plus a run-time environment with graphics, a debugger, access to certain system functions, and the ability to run programs stored in script files.

R Graphic Engine

One of R's strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed. Great care has been taken over the defaults for the minor design choices in graphics, but the user retains full control.

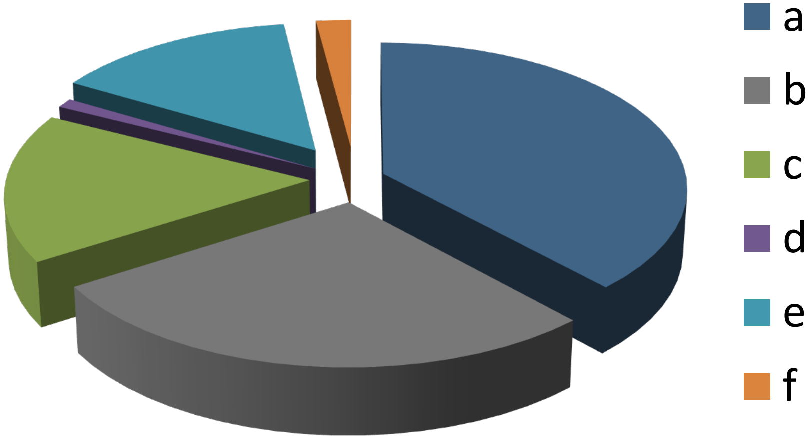
Plotting Data – Bad graphs

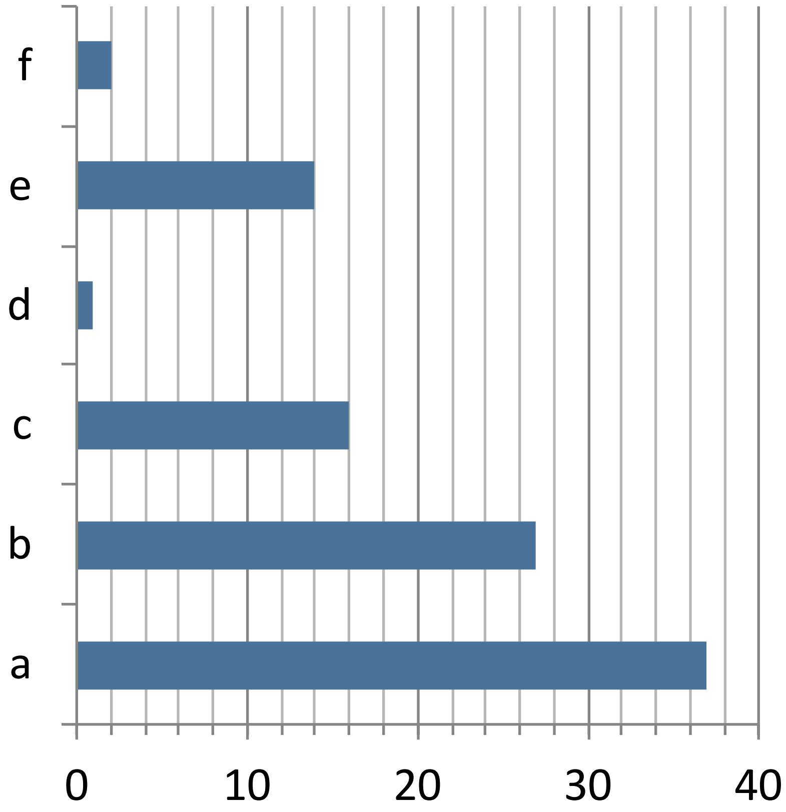
- Pie charts are almost never a good idea
  - Humans cannot perceive area, angle or perimeter properly
  - Hence comparison of quantities is difficult



Plotting Data – Bad graphs

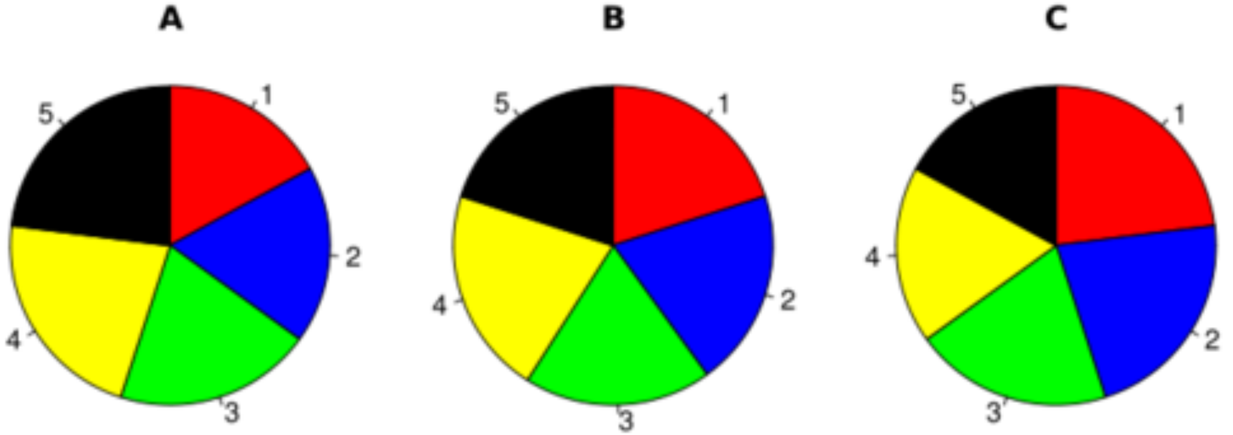
- Compare to bar charts:





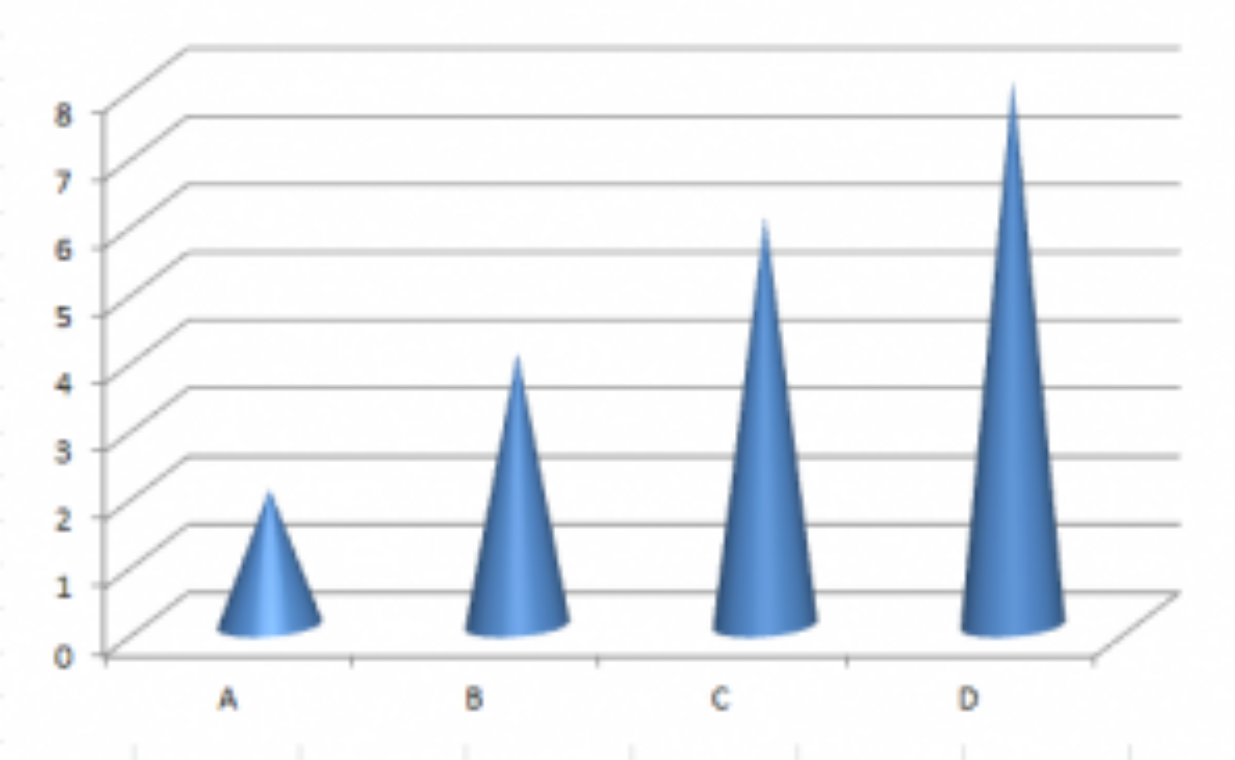
Plotting Data – Bad graphs

- Spot the differences!



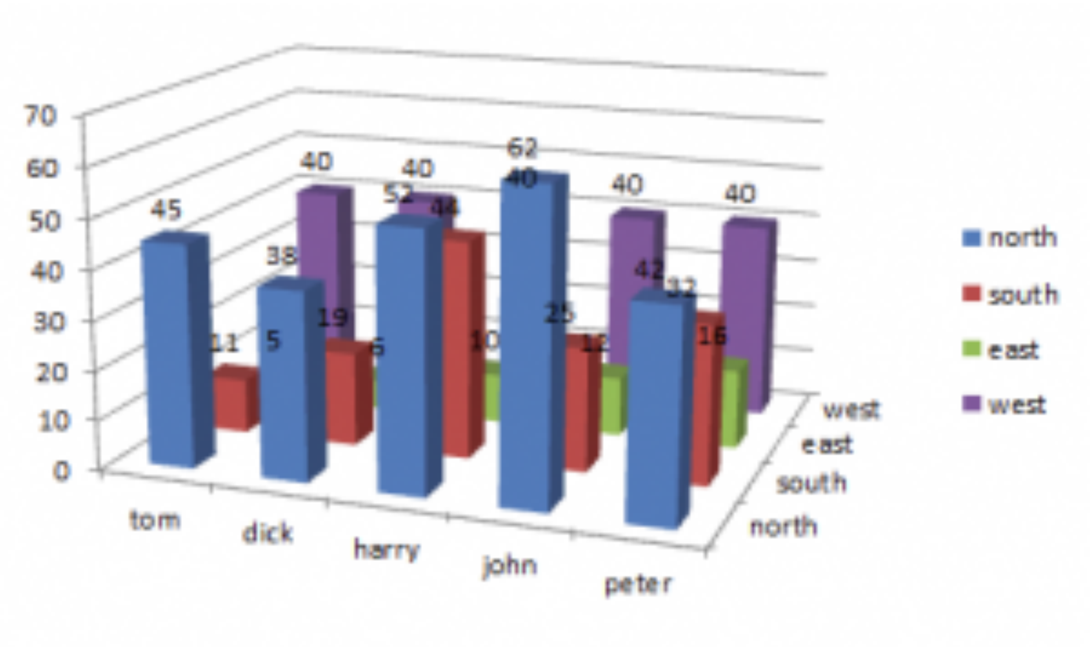
http://blog.revolutionanalytics.com/2009/08/how-pie-charts-fail.html

Bad Graphs



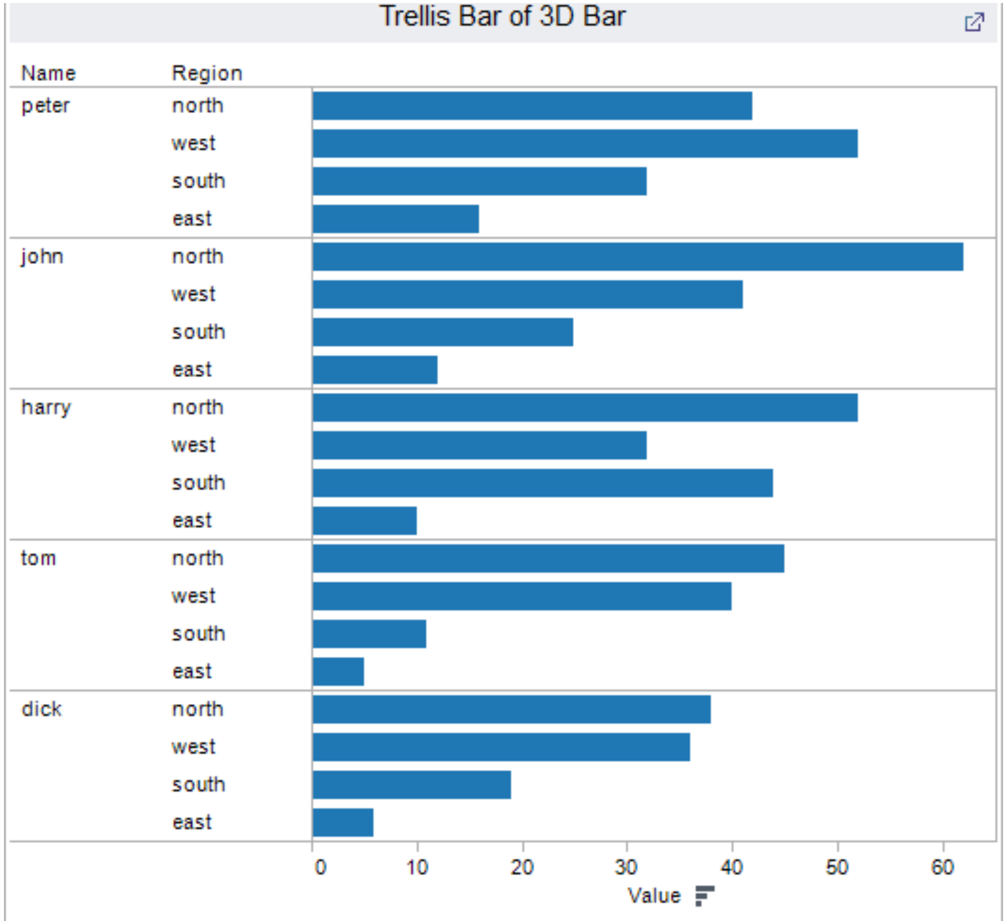
http://www.forbes.com/sites/naomirobbins/2012/05/30/winner-of-the-bad-graph-contest-announced-2/

Bad Graphs



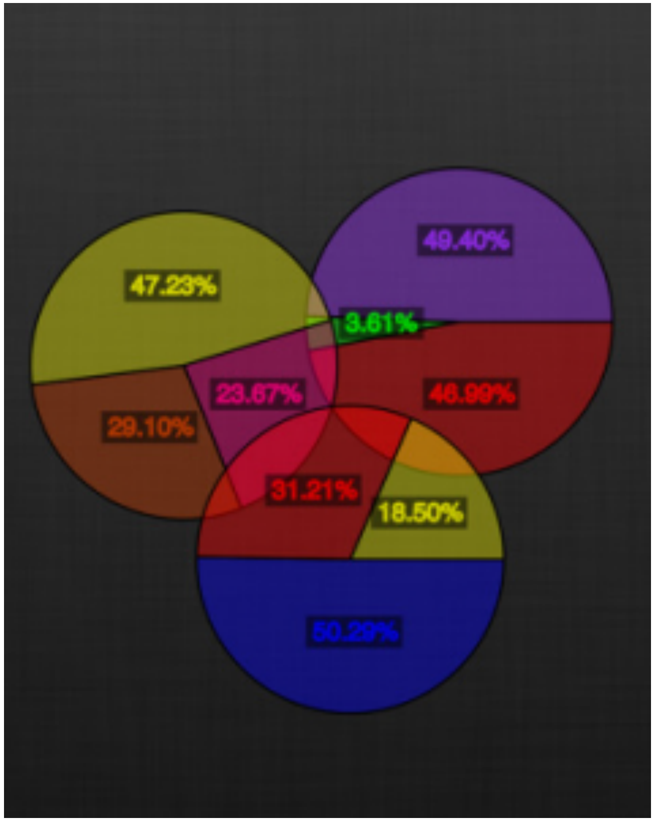
http://www.forbes.com/sites/naomirobbins/2012/05/30/winner-of-the-bad-graph-contest-announced-2/

Bad Graphs



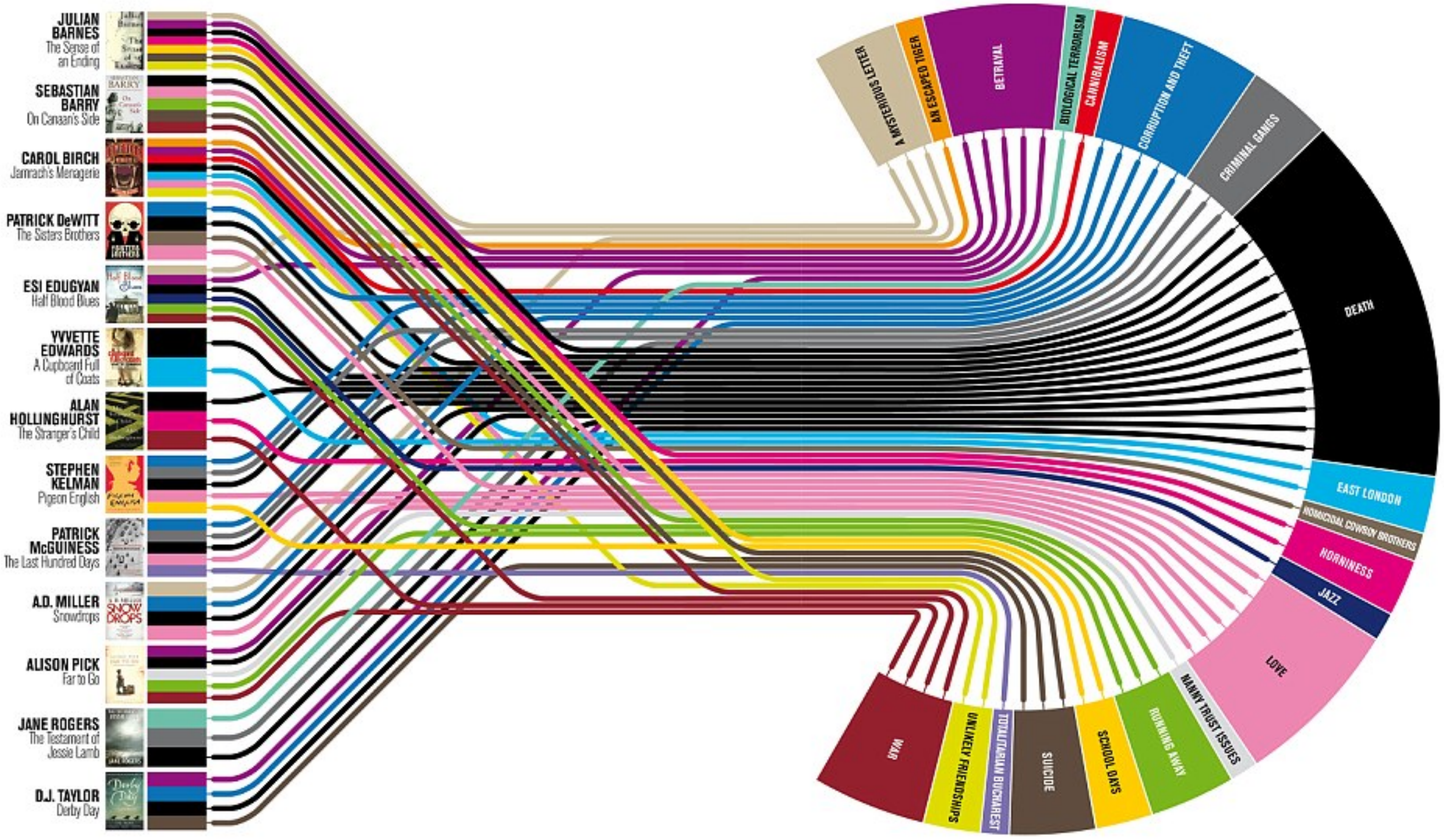
http://www.forbes.com/sites/naomirobbins/2012/06/07/trellis-plot-alternative-to-three-dimensional-bar-charts/

Bad Graphs



http://www.forbes.com/sites/naomirobbins/2012/05/30/winner-of-the-bad-graph-contest-announced-2/

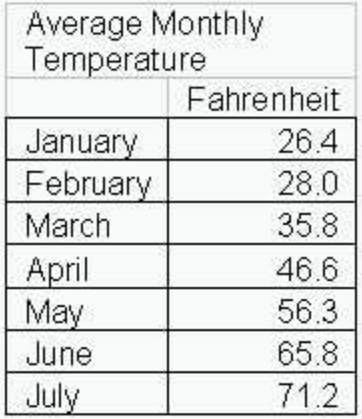
Bad Graphs



http://junkcharts.typepad.com/junk_charts/2012/10/can-information-be-beautiful-when-information-doesnt-exist.html

Bad Graphs

- Global Warming!!!



Average Monthly Temperature	
	Fahrenheit
January	26.4
February	28.0
March	35.8
April	46.6
May	56.3
June	65.8
July	71.2

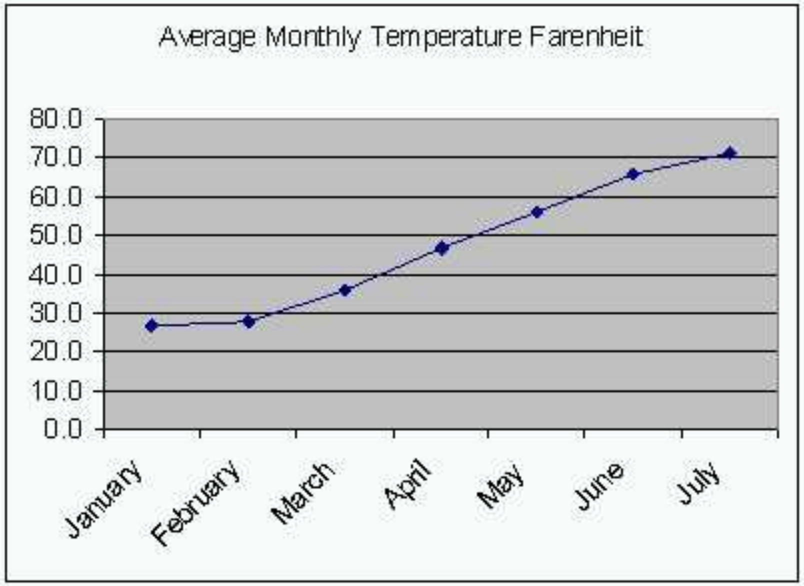
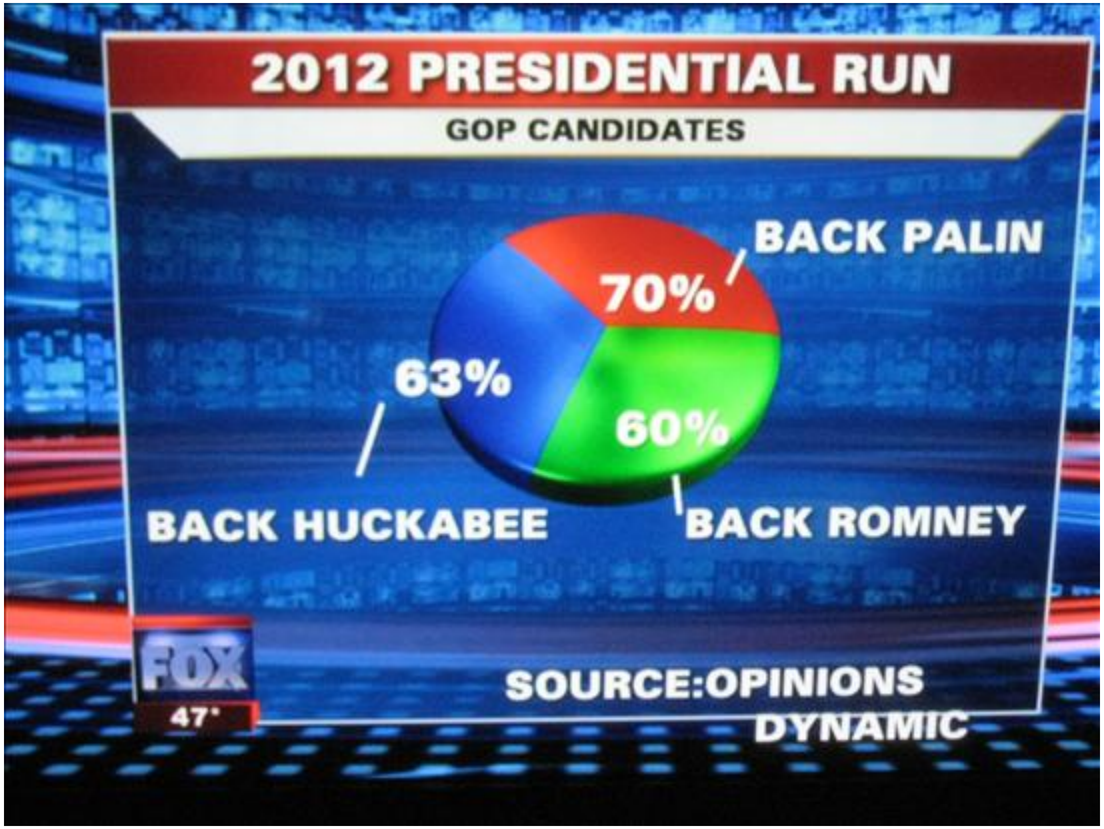


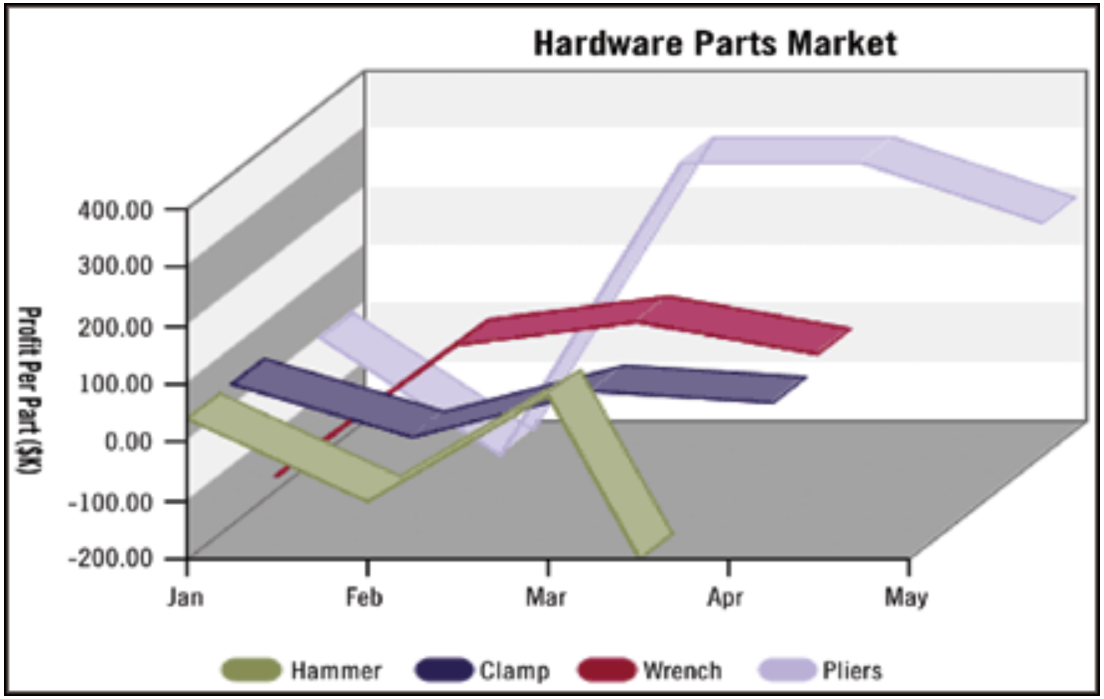
Figure 2

Bad Graphs

- For a total of… ???



Bad Graphs


Hardware Parts Market
Profit Per Part ($K)
400.00
300.00
200.00
100.00
0.00
-100.00
-200.00
Jan
Feb
Mar
Apr
May
Hammer
Clamp
Wrench
Pliers

Saving you plots

- With R you can quickly plot a graph: good

- Result is satisfying, how do I save it?
  - Open a graphic driver with pdf(), png(), svg(), jpeg(), …, even LaTeX

  - n <- rnorm(1000)
  - pdf('myplot.pdf')
  - plot(n, type='l')
  - dev.off()

Plotting hints

- **CHECKLIST FOR PLOTS BEFORE HANDING IN THE ASSIGNMENT**
- Make plots readable by
  - Not showing too much data, plot should be readable in seconds!
  - Using the “right” plot for visualization purpose
  - Do not rely on **COLOR** alone
    - plot *will* be printed out in greyscale, or black-and-white
  - Make it easy to compare the quantities
  - **AXIS**: what’s on abscissa? what’s on ordinate?
  - **UNITS**: without units values makes no sense
    - meters? millimeters? light years? bananas? potatoes?
  - Check your plot in the final document. **PRINT IT OUT**
    - a document on your screen might not be as good in a double column document
  - Use **PDF** as output format!!! **NO HORRIBLE RASTER GRAPHICS!!!**
    - **BE CAREFUL:** consider the data size as well. In case, go for high-res raster

Exercises – At your will

- Perform some simple data analysis / plotting with R
  - download and install R
  - download and install a front end (R Studio might be the the quickest option)
  - download the meteo dataset from classroom (taken from www.soda-is.com)

- First steps
  - load the dataset in R (function read.csv()) (see next slide for field description)
  - what is inside the dataset? (function names(), or head())
  - which cities are included? (function unique())
  - can you get min, max, quartiles, and average for the average temperatures? (function summary())
  - can you split the previous stats by city? (unique(), for loop, subset())
    - for the braves: install the plyr package and to this with ddply()

Exercises – CSV fields

- Month: Value for each month.
- City: Guess what?
- IrradianceWm2: Monthly mean of irradiance (W/m2)
- IrradianceKLyYear: Monthly mean of daily irradiance (kLy/year)
- IrradiationJcm2: Monthly mean of daily irradiation (J/cm2)
- TempMin: Monthly mean of daily min. of air temperature (C)
- TempMax: Monthly mean of daily max. of air temperature (C)
- TempMean: Monthly mean of air temperature (C)
- RelHumMin: Monthly mean of daily min. of relative humidity (%)
- RelHumMax: Monthly mean of daily max. of relative humidity (%)
- RelHumMean: Monthly mean of relative humidity (%)

Exercises – Some plots

- Make some plots:
  - mean temperature trend for Trento in a year (high level plot commands)
  - mean temperature trend for all cities in a year (one line each: use low level plot commands)
  - add a new column to the data.frame labeling the month (JAN, FEB, MAR…). Can you do it in a smart way? Then replot the previous graph with the name instead of the number
  - boxplots of mean temperature for different cities (look at the documentation for boxplot(), parameter “formula”). What can you infer from this plot?
  - add a column for the season (again, smart way) and plot the boxplots of mean temperatures of Trento for the different seasons
  - use ddply to compute the total irradiation for a city in a year (W/m2) and make a barplot of such values. Would you go selling solar panels in Eureka? (assume each month has 30 days)
  - bonus: use low level plotting functions for the previous plot and make the y axis nicer (kWh/cm2)