Handling data with the datamart package

Karsten Weinert
Query the web with S4

Karsten Weinert
Albacete, 11. Jul 2013
Strategic Consulting
- ca. 130 employees, ca. 40 employees in energy (policy, industry)
- based in Basel (Switzerland), further locations in Berlin, Düsseldorf, Bremen, München (Germany)

Software we are using
Excel (everyone) Mathematica (10) R (4) Stata (2)

A typical data flow
Collect → Save locally → Analyze / Visualize

Various sources
sometimes behind paywall
sometimes self-hosted online surveys
Example 1: Gas Prices at NetConnect Germany

XML based web service for natural gas prices at various hubs

- many rights reserved

Example query with S4 object

```r
> res <- query(ncg, "BasicPriceData", from="2010-10-01", to="2011-09-30", verbose=T)
```

trying to construct URL ..

Downloading http://datenservice.net-connect-germany.de/XmlInterface/getXML.ashx?ReportId=BasicPriceData&Start=01-10-2010&End=30-09-2011 ...

Transforming data ...

```r
> head(res)

   Gasday NCG_one_Day_Ahead_Settl_Price GASPOOL_one_Day_Ahead_Settl_Price TTF Zeebruegge     Unit
1 2010-10-01 06:00:00          1.898         1.912 1.8858 1.9035 Cent/kWh
2 2010-10-02 06:00:00          1.850         1.883 1.8858 1.9035 Cent/kWh
3 2010-10-03 06:00:00          1.850         1.883 1.8858 1.9035 Cent/kWh
4 2010-10-04 06:00:00          1.860         1.895 1.8205 1.8528 Cent/kWh
```
Example 1: How to construct the ncg object

Constructor urldata3 in package datamart (upcoming version)

```r
> ncg <- urldata3(
    resource = "BasicPriceData",
    template = "http://datenservice.net-connect-germany.de/
                XmlInterface/getXML.ashx?ReportId=BasicPriceData&
                Start=$\{from\}&End=$\{to\}",
    from = function(x) strftime(as.Date(x), "%d-%m-%Y"),
    to = function(x) strftime(as.Date(x), "%d-%m-%Y"),
    extract.fct = xmlParse,
    transform.fct = function(dat) {
        namespaces <- c(sqlrowset="urn:schemas-microsoft-com:sql:SqlRowSet1",
                        xsd="http://www.w3.org/2001/XMLSchema-instance",
        nodes <- getNodeSet(dat, "/sqlrowset:Price", namespaces=namespaces)
        dat <- sapply(nodes, function(n) c(
            xmlValue(n["Gasday"]), xmlValue(n["NCG_one_Day_Ahead_Settl_Price"]),
            xmlValue(n["Unit"])))
        dat <- data.frame(Gasday=strptime(dat[1,], "%Y-%m-%dT%H:%M:%S"),
                           NCG=as.numeric(dat[2,]),
                           Unit=dat[3,], stringsAsFactors=FALSE)
        return(dat)
    }
)`
Example 2: Eurostat Bulk download

Direct access to all Eurostat datasets as tab-separated gzipped files

- uses codes for statistics, country names, product names, etc

Example query with S4 object

```r
> res <- query(eurostat, "nrg_103m", verbose=T)
```
Example 2: How to construct the eurostat object

Constructor urldata3 in package datamart (upcoming version)

```r
> eurostat <- urldata3(resource=„nrg_103m“, 
  template="http://epp.eurostat.ec.europa.eu/NavTree_prod/ 
  everybody/BulkDownloadListing?file= 
  data%2Fnrg_103m.tsv.gz“, 

extract.fct=function(uri) { 
  tf <- tempfile() 
  content <- getBinaryURL(uri) 
  writeBin(content, tf) 
  return(tf) 
},

transform.fct=function(x) { 
  res <- read.csv(gzfile(x), na.strings=c(" ", "p"), stringsAsFactor=FALSE, sep="\t") 
  unlink(x) 
  code <- strsplit(strhead(colnames(res)[[1]], -5) , "\.")[[1]] 
  tm <- as.Date(paste(substring(tail(colnames(res),-1),2),1,1,sep="-0")) 
  nm <- gsub("", ".", res[,1]) 
  res <- res[,2:ncol(res)] 
  res <- as.numeric(gsub("( s|,)", "", as.matrix(res))) 
  res <- matrix(res, nrow=length(tm), ncol=length(nm), byrow=TRUE) 
  return(res) 
}
```
Advantages of the (still experimental) query generic

Simple and versatile, easy to remember interface to various web resources
- it’s always query(obj, resource, …)
- queries(obj) lists all defined resources

Extensible: can be inherited for alternative data transformations
- for instance:
  setClass(Class=“NcgWeb“, contains=“UrlData3“)
  ncg <- urldata3(clss=“NcgWeb“, … # rest as above)
  setMethod(f=“query“, signature=c(“NcgWeb“, resource(“WeeklyLogReturns“),
                                 definition=function(self, resource, …) { # code goes here })

Extensible: several objects can be mashed up
- for instance:
  gas_web <- mashup(
    ncg =urldata3(…# as above),
    eurostat_nrg103m=urldata3(… # as above),
    … # further objects
  )

> queries(gas_web)
[2] „nrg_103m“, „BasicPriceData“
Karsten Weinert
Consultant

prognos | Goethestr. 85 | D-10623 Berlin

Tel: +49 30 52 00 59 266
Fax: +49 30 52 00 59 288 266

E-Mail: karsten.weinert@prognos.com
Wir geben Orientierung.

Prognos AG – Europäisches Zentrum für Wirtschaftsforschung und Strategieberatung.
Visualizing correlation matrices with R

Jan Graffelman
Visualizing correlation matrices with R

Jan Graffelman$^1$

$^1$Department of Statistics and Operations Research
Universitat Politècnica de Catalunya
Barcelona, Spain

jan.graffelman@upc.edu

The R User Conference
July 10-12 2013, University of Castilla-La Mancha, Albacete, Spain
Outline

1. Representing correlations
2. Methods: PCA, PFA and Correlograms
3. Example in R
Introduction

- We live a high-tech era with automated data-collection and ever larger data matrices.
- The correlation matrix is often used to summarize pairwise relationships.
- Good graphical tools for visualizing a correlation matrix are needed.
Ways to represent correlations

- There are (infinitely) many ways to represent correlations graphically.

Correlations can be represented by:

- the amount of scatter around a regression line.
- an angle between two vectors in plot.
- a scalar product in a biplot.
- a distance between two points.
- the area of a triangle.
- ...
Six ways to depict $r(x, y) = 0.5$
Three methods for approximating the correlation matrix $\mathbf{R}$

- Principal Component Analysis (PCA) biplot
- Principal Factor Analysis (PFA) biplot
- Correlogram (CRG)

We use the $\text{RMSE}$ as a measure of goodness-of-fit. We have the error matrix $\mathbf{E} = \mathbf{R} - \hat{\mathbf{R}}$.

\[
\text{RMSE} = \sqrt{\frac{1}{2} p(p - 1) \sum_{i<j} e_{ij}^2},
\]
Based on the eigenvalue-eigenvector decomposition of $R$:

$$R = VD\lambda V'$$

$R$ approximated in 2D by plotting first two columns of $G = VD^{1/2}\lambda$

$$\hat{R} = G_{(2)} G_{(2)}' = V_{(2)} D_{(2,2)} V_{(2)}'$$

$\hat{R}$ is the best approximation to $R$ in the least squares sense.
The classical tool: PCA (2/3)

- The approximation of $\hat{R}$ by $G_{(2)}G_{(2)}'$ is by scalar product.
- Generally, correlations are inferred from biplots by looking at angles.
- A well-known PCA property is that correlations are approximated by cosines of angles.
- The latter approximation is obtained by
  $$\hat{R} = D_g^{-1}G_{(2)}G_{(2)}'D_g^{-1}$$
  with $D_g = \text{diag}(G_{(2)}G_{(2)}')^{1/2}$
- The approximation of $R$ with cosines of angles is suboptimal.
A least-squares approximation to $R$ will also try to approximate the ones on the diagonal.

This is unnecessary, as we know $r(x_i, x_i) = 1$.

Principal Factor Analysis (PFA) avoids fitting the ones on the diagonal.

The fit to the off-diagonal elements of $R$ obtained by PFA is superior to PCA.

The fit obtained by PFA is also by scalar product.
Correlogram

- The correlogram (Trosset, 2005) optimizes the fit to $\mathbf{R}$ by using cosines of angles.
- Numerically minimize

$$f(\theta) = \| \mathbf{R} - \mathbf{C}(\theta) \|^2$$

with

$$\mathbf{C}(\theta) = (\cos(\theta_j - \theta_k)) \quad \text{and} \quad \theta = (0, \theta_2, \ldots, \theta_p)$$

- Can be done with R's function `nlminb`.
- All vectors constrained to be of norm 1.
Example: Mardia’s grades for students

<table>
<thead>
<tr>
<th>mechanics (C)</th>
<th>vectors (C)</th>
<th>algebra (O)</th>
<th>analysis (O)</th>
<th>statistics (O)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>77</td>
<td>82</td>
<td>67</td>
<td>67</td>
</tr>
<tr>
<td>2</td>
<td>63</td>
<td>78</td>
<td>80</td>
<td>70</td>
</tr>
<tr>
<td>3</td>
<td>75</td>
<td>73</td>
<td>71</td>
<td>66</td>
</tr>
<tr>
<td>4</td>
<td>55</td>
<td>72</td>
<td>63</td>
<td>70</td>
</tr>
<tr>
<td>5</td>
<td>63</td>
<td>63</td>
<td>65</td>
<td>70</td>
</tr>
<tr>
<td>6</td>
<td>53</td>
<td>61</td>
<td>72</td>
<td>64</td>
</tr>
<tr>
<td>7</td>
<td>51</td>
<td>67</td>
<td>65</td>
<td>65</td>
</tr>
<tr>
<td>8</td>
<td>59</td>
<td>70</td>
<td>68</td>
<td>62</td>
</tr>
<tr>
<td>9</td>
<td>62</td>
<td>60</td>
<td>58</td>
<td>62</td>
</tr>
<tr>
<td>10</td>
<td>64</td>
<td>72</td>
<td>60</td>
<td>62</td>
</tr>
<tr>
<td>11</td>
<td>52</td>
<td>64</td>
<td>60</td>
<td>63</td>
</tr>
<tr>
<td>12</td>
<td>55</td>
<td>67</td>
<td>59</td>
<td>62</td>
</tr>
<tr>
<td>13</td>
<td>50</td>
<td>50</td>
<td>64</td>
<td>55</td>
</tr>
<tr>
<td>14</td>
<td>65</td>
<td>63</td>
<td>58</td>
<td>56</td>
</tr>
<tr>
<td>15</td>
<td>31</td>
<td>55</td>
<td>60</td>
<td>57</td>
</tr>
<tr>
<td>16</td>
<td>60</td>
<td>64</td>
<td>56</td>
<td>54</td>
</tr>
<tr>
<td>17</td>
<td>44</td>
<td>69</td>
<td>53</td>
<td>53</td>
</tr>
<tr>
<td>18</td>
<td>42</td>
<td>69</td>
<td>61</td>
<td>55</td>
</tr>
<tr>
<td>19</td>
<td>62</td>
<td>46</td>
<td>61</td>
<td>57</td>
</tr>
<tr>
<td>20</td>
<td>31</td>
<td>49</td>
<td>62</td>
<td>63</td>
</tr>
<tr>
<td>84</td>
<td>15</td>
<td>38</td>
<td>39</td>
<td>28</td>
</tr>
<tr>
<td>85</td>
<td>5</td>
<td>30</td>
<td>44</td>
<td>36</td>
</tr>
<tr>
<td>86</td>
<td>12</td>
<td>30</td>
<td>32</td>
<td>35</td>
</tr>
<tr>
<td>87</td>
<td>5</td>
<td>26</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>88</td>
<td>0</td>
<td>40</td>
<td>21</td>
<td>9</td>
</tr>
</tbody>
</table>
Scatterplot matrix
install.packages("Correlplot")
library("Correlplot")
data(students)
R <- cor(students)
out.pca <- pca(students)
out.pfa <- pfa(students)
correlogram(R)
Miniature example code (2/2)

(b) PFA biplot (scp: 0.004 cos: 0.33)

(d) correlogram (cos: 0.22)
References

- R-package Correlplot, will be made available on CRAN, and can make PCA and PFA biplots and the various correlograms.
Rcpp is ready

Romain Francois
Rcpp is ready

Romain François

@romain_francois

jeudi 11 juillet 13
Rcpp is ready

Romain François

@romain_francois
0.10.4
acer, ALKr, Amelia, apcluster, BayesComm, bcp, bfa, bfp, bifactorial, blockcluster, ccaPP, cda, cladoRcpp, classify, climdex.pcic, clogitL1, clusteval, ConConPiWiFun, cxxfunplus, ddgraph, Delaporte, DESeq2, devtools, diversitree, ecp, EpiContactTrace, ExactNumCI, fastGHQuad, FastPCS, FBFsearch, fdaMixed, flowWorkspace, forecast, fugeR, geiger, GeneNetworkBuilder, GeneticTools, gMWT, GOSemSim, GOSummaries, gRbase, GRENITS, gRim, growcurves, GSE, GUTS, HLMdiag, httpuv, hyperSpec, inline, jaatha, KernSmoothIRT, Kmisc, knitr, LaF, marked, maxent, mets, minqa, mirt, miscF, mosaics, MPTinR, msgl, multmod, mvabund, MVB, mzR, NetworkAnalysis, ngsSpatial, oem, openair, orQA, pcaMethods, phom, phylobase, planar, PReMiuM, psgp, quadrupen, RcppArmadillo, RcppBDT, rcppbugs, RcppClassic, RcppClassicExamples, RcppCNPy, RcppDE, RcppEigen, RcppExamples, RcppGSL, RcppOctave, RcppProgress, RcppRoll, RcppSMC, RcppXts, Rdisop, rexpokit, rforensicbatwing, rgam, RInside, Risa, Rmalschains, RMessenger, rmgarch, Rmixmod, robustgam, robustHD, rococo, RProtoBuf, RQuantLib, RSNNS, RSofia, rTANDEM, rugarch, RVowpalWabbit, SBSA, sdcMicro, sdcTable, sequences, simFrame, sirt, spacodiR, sparseHessianFD, sparseLTSEigen, SpatialTools, stochvol, stream, surveillance, survSNP, tagcloud, TAM, tbart, termstrc, tmg, transmission, transnet, trustOptim, unmarked, VIM, waffect, WideLM, wordcloud, zic
```cpp
#include <Rcpp.h>

// [[Rcpp::export]]
int add( int a, int b){
    return a + b ;
}

> sourceCpp( "add.cpp" )
> add( 1, 2 )
[1] 3
```
cppFunction( 'int add( int a, int b){
    return a + b ;
}' )
add( 1, 2 )
# [1] 3
C++ Classes

class Account {
private:
  double balance;

global:
  Account( ) : balance(0){}
  double get_balance(){
    return balance;
  }
  void withdraw(double x){
    balance -= x;
  }
  void deposit(double x){
    balance += x;
  }
}

RCPP_MODULE(BankAccount){
class_<Account>( "Account"
  .constructor()
  .property( "balance", Account::get_balance )
  .method( "deposit", Account::deposit )
  .method( "withdraw", Account::withdraw )
}
Seamless R and C++ Integration with Rcpp

rcpp.org
gallery.rcpp.org
Challenge :

API Documentation
jeudi 11 juillet 13
jeudi 11 juillet 13
RStudio

develtools

jeudi 11 juillet 13
Questions

romain@r-enthusiasts.com

@romain_francois
Mapping half a million petition signatures using R

Andy South
Mapping half a million petition signatures using R

Andy South
southandy@gmail.com
@southmapr
La plataforma de peticiones
¿Qué quieres cambiar?

La unidad de cardiólogía infantil en Las Palmas no se cierra
Ganada con 46.788 firmas
£53/wk?
Prove it IDS

Petitioning Iain Duncan Smith
Iain Duncan Smith: Iain Duncan Smith to live on £53 a week.
API

*application programming interface
Barry Rowlingson

https://github.com/barryrowlingson/changer
Geocode*

*finding geographic coordinates from other data
Javier Bardem, Penelope Cruz, Julio Medem, Burnley, Doncaster, Norwich

Bus Stop

coordinates

Northampton: 68,18
Norwich: 76,24
Nottingham: 56,32
symbols( x = dFcity$Easting, 
         y = dFcity$Northing, 
         circles = dFcity$numSignatures )

First 1,000 signatures in 33 mins
How many signed the IDS petition in your town?

Town starting with:

2nd Letter:

Choose a town:

London

54292 people from London signed.

Map  Rank Table  Bars  Timeline (slow)  About
We call on Iain Duncan Smith to prove his claim to be able to live on £7.57 a day, or £53 a week, by doing so for one year.

438,000 of us (and counting)

From every corner of Great Britain

Made using R by
@southmpr
@carborneelr
@geospacedman

www.andysouth.co.uk
southandy@gmail.com
@southmapr

Share this and go to change.org to help make it 500,000...
Using R for the parallelized analysis of big genomics data

Ramon Diaz-Uriarte
Using R for the parallelized analysis of big genomic data

Ramon Diaz-Uriarte

useR! 2013, Albacete, Spain
Chromosomes

aCGH: example of data
Large data sets

- Millions of data points ("genes", "probes") per subject (hundreds of thousands of data points by chromosome).
- Hundreds or thousands of subjects.
Why parallelize?

- Code available for many procedures, but not parallelized.
- Many computations embarrassingly parallelizable:
  - subjects
  - subjects by chromosomes
Parallelizing code

- Parallelize existing code. parallel and snow packages.
  - Clusters of workstations: MPI, PVM, sockets, etc.
  - Single multicore machine: forking.
Forking

Multicore (forking) vs. multiple nodes (MPI)
- No need to explicitly pass objects.
- Avoid communication overheads.
- Child processes share memory with parent.
Using `ff`

- Millions of probes.
- Hundreds or thousands of subjects.
- No need to hold everything in memory at once.
- Load only needed data, leave the rest on disk.

Package `ff`: “memory-efficient storage of large data on disk and fast access functions.”
Reading files

- Each column a single file: allow parallelized reading.
- Convert to \( \mathbb{F} \) objects.
$ff$, parallelization, shared storage

![Diagram showing parallelization and shared storage in genomics data analysis]

**Data** → $R_{\text{master}}$ → $ff_{\text{out}}$ → Results

$ff_{\text{in}}$ connects $R_1$, $R_2$, $R_n$ to $ff_1$, $ff_2$, $ff_n$.

Read only: $R_1$, $R_2$, $R_n$

Write: $ff_{\text{out}}$
ff, parallelization, shared storage (II)

Data → $R_{master}$

$ff_{in}$ → $ff_{out}$ → Results

$R_1$ → $ff_1$ → Fig.1

$R_2$ → $ff_2$ → Fig.2

$R_n$ → $ff_n$ → Fig.n

write
read only
Benchmarks

- Each column: 6,067,433 probes (or rows).
- Size RData with 1000 columns: 46 GB.
- Machines:
  - 64-core, 384 GB RAM.
  - 12-core, 64 GB RAM.
Parallelized analysis (HaarSeg)

<table>
<thead>
<tr>
<th>Fork/ NP</th>
<th>Cores</th>
<th>ff/ RAM</th>
<th>Arrays</th>
<th>Wall time (minutes)</th>
<th>Max. Σ mem. (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>-</td>
<td>RAM</td>
<td>1000</td>
<td>198.3 <strong>Cannot allocate memory</strong></td>
<td></td>
</tr>
<tr>
<td>Fork</td>
<td>64</td>
<td>ff</td>
<td>1000</td>
<td>6.4</td>
<td>26.9</td>
</tr>
</tbody>
</table>
## Parallelized analysis (HaarSeg)

<table>
<thead>
<tr>
<th>Fork/ NP</th>
<th>Cores</th>
<th>ff/ RAM</th>
<th>Arrays</th>
<th>Wall time (minutes)</th>
<th>Max. $\Sigma$ mem. (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fork</td>
<td>12</td>
<td>ff</td>
<td>2000</td>
<td>57.4</td>
<td>10.1</td>
</tr>
<tr>
<td>Fork</td>
<td>64</td>
<td>ff</td>
<td>2000</td>
<td>11.9</td>
<td>28.4</td>
</tr>
</tbody>
</table>
Conclusions

- Possible to deal with realistically large data sets in reasonable time.
- Make good usage of hardware (multicore).
- Package ADaCGH2 in BioConductor.

Acknowledgements

- O. M. Rueda, D. Rico.
- Funding: Fundación de Investigación Médica Mutua Madrileña, Project TIC2003-09331-C02-02 of the Spanish MEC and BIO2009-12458 of the Spanish MICINN.
- CNIO (Spanish National Cancer Research Center), IIB (UAM-CSIC).
RStudio Debugging Features Preview

Jonathan McPherson
RStudio Debugging Features

jonathan@rstudio.com
```r
# first function
first <- function() {
  second()
}

second <- function() {
  a <- 1
  browser()
  b <- 2
}
```
```r
first <- function() {
  second()
}

second <- function() {
  a <- 1
  browser()
  b <- 2
}
```

```
> first()
Called from: second()
Browse[1]
```
```r
first <- function()
{
  second()
}

second <- function()
{
  a <- 1
  browser()
  b <- 2
}
```

---

```r
> first()
Called from: second()
Browse[1]>
```

---

```
second() at hello.R:7
first() at hello.R:2
```
```r
# hello.R

first <- function() {
  second()
}

second <- function() {
  a <- 1
  browser()
  b <- 2
}

first()
```
```r
# hello.R

first <- function() {
  second()
}

second <- function() {
  a <- 1
  browser()
  b <- 2
}
```

```
> first()
Called from: second()
Browse[1]>
```

Traceback:
- `second() at hello.R:7`
- `first() at hello.R:2`
```r
first <- function() {
  second()
}

second <- function() {
  a <- 1
  browser()
  b <- 2
  c <- 3
}
```

Values

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>2</td>
</tr>
</tbody>
</table>

Traceback

```
second() at hello.R:7
first() at hello.R:2
```
```
first <- function() {
  second()
}

second <- function() {
  a <- 1
  b <- 2
  c <- 3
}
```

```
> debug(second)
> first()
debugging in: second()
```

```
Traceback
second() at hello.R:5
first() at hello.R:2
```
```r
fac <- function(n) {
  if (n == 1) stop(n)
  fac(n-1)
}
```

```
> fac(7)
Error in fac(n - 1): 1
>  
```
```r
fac <- function(n)
{
  if (n == 1) stop(n)
  fac(n-1)
}
```

```
> options(error=browser)
> fac(7)
Error in fac(n - 1) : 1
Called from: stop(n)
Browse[1]>
```

```
fac(7) at factorial.R:4
fac(6) at factorial.R:4
fac(5) at factorial.R:4
fac(4) at factorial.R:4
fac(3) at factorial.R:4
fac(2) at factorial.R:4
fac(1) at factorial.R:3
```
Function: devtools::load_all (source unavailable or out of sync)

```r
function(pkg = ".", reset = TRUE, recompile = FALSE,
         export_all = TRUE, quiet = FALSE) {
  if (is.package(pkg)) {
    create_description(pkg)
    pkg <- as.package(pkg)
  }
  if (!quiet) message("Loading ", pkg$package)

  # Reloading devtools is a special case. Normally, objects in the
  # namespace become inaccessible if the namespace is unloaded before the
  # the object has been accessed. This is kind of a hack - using as.list
  # on the namespace accesses each object, making the objects accessible
  # later, after the namespace is unloaded.
}
```

Traceback
```
devtools::load_all() at load.r:86
```
Available Today

http://www.rstudio.com/ide/download/preview
```r
# greeting functions

# hello function
hello <- function(person) {
  paste("Hello, ", person)
}

# goodbye function
goodbye <- function(person) {
  paste("Goodbye, ", person)
}

# rapidgreet function
rapidgreet <- function(person) {
  greeting <- hello(person)
  parting <- goodbye(person)
  greeting <- paste(greeting, parting)
}

# greet function
greet <- function(person) {
  rapidgreet(person)
}
```

```
```r
# hello.R

```
Using mixed models for quality control of citizen science data

Thierry Onkelinx
Using mixed models for quality control of citizen science data

Finding anomalies in large datasets

useR!2013, Albacete, Spain
July 11th 2013
Thierry Onkelinx, Koen Devos, Glenn Vermeersch, Paul Quataert
Quality issues

- Type of errors
  - Determination errors
  - Counting errors
  - Coding errors

- Manual validation of each record is tedious

  **Waterfowl Counts** 40 species, 1000 sites, 6 records per winter, 23 winters

  **Common Breeding Birds** 100 species, 900 sites, 18 records per spring, 7 years

- Automatically setting all records to 'validated' is plain wrong

- We need a tool to find anomalies
Anomaly

- Deviation from the 'normal' pattern
- Due to
  - Errors
  - Unusual circumstances
    - flooding
    - weather conditions
    - ...
  - Misspecification of the 'normal' pattern
Define the ’normal’ pattern

- Start by looking at the design
  - Sites are repeatedly visited within a winter and during several winters
- Convert the design into a suitable model
  - Winter as fixed factor
  - Month as fixed factor
  - Site as random intercept
- More complexity can be added
  - Interaction between winter and month as fixed factor
  - Random walk along winter per site as random effect
  - Relevant environmental covariates (if available)
Model fitting

- Bird counts
  - generalized linear mixed model
- Highly variable counts
  - overdispersion in likely
  - poisson distribution with observation level random effect
  - negative binomial distribution
- Species not or rarely observed in some sites
  - zero inflation?
  - remove irrelevant sites
Model fitting: the packages

\textbf{lme4}  
- Overdispersion: Poisson $+$ OLRE  
- No zero inflation  
- Syntax is more R-ish and straightforward  
- Frequentist

\textbf{INLA}  
- Overdispersion: negative binomial  
- Constant zero inflation available  
- Complex syntax  
- Bayesian
Define anomaly

- Visual inspection on fixed effects
- \(x\) highest and \(x\) lowest residuals
- \(x\) highest and \(x\) lowest random intercept
- \(x\) strongest random slopes
- ... 

- \(x\) can be relative small (e.g. 10 or 20)
- All observations linked to an anomaly should be checked
Example: European golden plover (*Pluvialis apricaria*)
Fixed effect parameters

Expected count in average site in January

Multiplicative effect of month

Winter

0
20
40
60
2000 2005 2010

October
November
December
January
February
March

Month

9 / 15
A site with some strong residuals
Site random intercept
Extreme random walks along winter

Random walk

Winter


1010301 1010802 1011004 1011201
1011404 1012501 1014201 1014401
1020107 1020701 1030204 1033303
Workflow

- validation status is multinomial instead of binomial
  - Not checked (default)
  - Anomaly
  - Validated
  - Rejected
  - ...
- run model(s)
- status 'Not checked' $\Rightarrow$ 'Anomaly' when relevant
- check observations flagged as 'Anomaly'
  - Correct coding errors
  - status: 'Validated', 'Rejected', ...
- repeat until no more changes from 'Not checked' to 'Anomaly'
Documentation and reproducibility

- Split code in clear functions
  - One task = one function
  - Use subfunctions in case of complex tasks

- Store everything in package

- Use version control (e.g. svn)

- Run analysis on remote machine
  1. Check out svn repository
  2. Compile package
  3. Run analysis

- Store output and metadata in database

- Generate standard reports using Sweave
Conclusions

- Mixed models are a useful tool for scanning large anomalies
- Give relevant output to field experts to evaluate the anomalies
- Automatisation is a must for ongoing data collection
- R packages are a great tool for documentation

Special thanks to all the volunteers for collection of the data
Transcatheter Heart Valve Registry Data Explorer - a Shiny Use Case

Alexander Meyer
Transcatheter Heart Valve Registry Data Explorer - a Shiny Use Case

Alexander Meyer, M.D.
Kerckhoff HeartCenter, Department Cardiac Surgery, Bad Nauheim, Germany
Tricuspid valve
Pulmonary veins
Mitral valve
Right atrium
Pulmonary valve
Left atrium

KERCKHOFF HERZ- UND THORAXZENTRUM
The Problem(s)
The Problem(s)
The solution – TAVI-WebApp
Shiny shines easily!!

No HTML

or JavaScript

or Event Handling or …

expertise necessary!!

Just pure R!
Year Selection

2009 2012 2013 2014

Misc. Settings

☑ Plotting Options
  Categorical aggregates shown as
    ⬜ Total Percentages
    ○ Marginal Percentages (for Mortality & Complication Variables)
    ○ Counts
    □ Flip Axis
    □ Use Log Scale

☑ Advanced Filter
  age > 65
Descriptive Statistics:

Selected: 349 cases of the year/s 2012 2013

For status = TA-AVI
obs. mean median s.d. min. max.
162 7.99 6.55 5.548 1.5 34.1

For status = TT-AVI
obs. mean median s.d. min. max.
184 6.45 5.6 4.32 1.71 36.1

Unadjusted Inferential Statistics: (EXPERIMENTAL)

Overall Statistics among Groups/Exposures/Treatments:
obs. mean median s.d. min. max.
246 7.17 6 4.99 1.5 36.1
For Exposure = TA-AVI
Anticipated Useage

Broad Data Overview

Hypothesis Generation

- Observational Studies
- Prospective Studies
Thank you R community!!!
10 Reasons to Use Revolution R Enterprise - Free to Academics

Mario Inchiosa
10 Reasons to Use Revolution R Enterprise – Free to Academics

Presented by:
Mario E. Inchiosa, Ph.D.
US Chief Scientist

useR! 2013
Albacete, Spain
July 11, 2013
R is open source and drives analytic innovation but has some limitations

<table>
<thead>
<tr>
<th>Memory Bound</th>
<th>Big Data</th>
<th>Bigger data sizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Threaded</td>
<td>Scale out, parallel processing, high speed</td>
<td>Speed of analysis</td>
</tr>
<tr>
<td>Community Support</td>
<td>Commercial production support</td>
<td>Production support</td>
</tr>
<tr>
<td>Innovative – 5000+ packages, exponential growth</td>
<td>Combines with open source R packages where needed</td>
<td>Innovation and scale</td>
</tr>
</tbody>
</table>
Revolution R Enterprise
High Performance, Multi-Platform Analytics Platform

Revolution R Enterprise

DeployR
Web Services
Software Development Kit

DevelopR
Integrated Development Environment

ConnectR
High Speed & Direct Connectors
Teradata, Hadoop (HDFS, HBase), SAS, SPSS, CSV, OBDC

ScaleR
High Performance Big Data Analytics
Platform LSF, MS HPC Server, MS Azure Burst, SMP Servers

DistributedR
Distributed Computing Framework
Platform LSF, MS HPC Server, MS Azure Burst

RevoR
Performance Enhanced Open Source R + CRAN packages
IBM PureData (Netezza), Platform LSF, MS HPC Server, MS Azure Burst,
Cloudera, Hortonworks, IBM Big Insights, Intel Hadoop, SMP servers

Revolution Analytics Value-Add Components
Providing Power and Scale to Open Source R

Open Source R Plus
Revolution Analytics performance enhancements
## Performance: Multi-threaded Math

<table>
<thead>
<tr>
<th>Computation (4-core laptop)</th>
<th>Open Source R</th>
<th>Revolution R</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Linear Algebra</strong>¹</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Matrix Multiply</td>
<td>327 sec</td>
<td>13.4 sec</td>
<td>23x</td>
</tr>
<tr>
<td>Cholesky Factorization</td>
<td>31.3 sec</td>
<td>1.8 sec</td>
<td>17x</td>
</tr>
<tr>
<td>Linear Discriminant Analysis</td>
<td>216 sec</td>
<td>74.6 sec</td>
<td>2x</td>
</tr>
<tr>
<td><strong>General R Benchmarks</strong>²</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R Benchmarks (Matrix Functions)</td>
<td>22 sec</td>
<td>3.5 sec</td>
<td>5x</td>
</tr>
<tr>
<td>R Benchmarks (Program Control)</td>
<td>5.6 sec</td>
<td>5.4 sec</td>
<td>Not appreciable</td>
</tr>
</tbody>
</table>

RevoScaleR Overview

- An R package that adds capabilities to R:
  - Data Import/Clean/Explore/Transform
  - Analytics – Descriptive and Predictive
  - Parallel and distributed computing
  - Visualization

- Scales from small local data to huge distributed data

- Scales from laptop to server to cluster to cloud

- Portable – the same code works on small and big data, and on laptop, server, cluster, Hadoop
ScaleR: High Performance Scalable Parallel External Memory Algorithms

Data Prep, Distillation & Descriptive Analytics

- **R Data Step**
  - Data import – Delimited, Fixed, SAS, SPSS, OBDC
  - Variable creation & transformation
  - Recode variables
  - Factor variables
  - Missing value handling
  - Sort
  - Merge
  - Split
  - Aggregate by category (means, sums)

- **Descriptive Statistics**
  - Min / Max
  - Mean
  - Median (approx.)
  - Quantiles (approx.)
  - Standard Deviation
  - Variance
  - Correlation
  - Covariance
  - Sum of Squares (cross product matrix for set variables)
  - Pairwise Cross tabs
  - Risk Ratio & Odds Ratio
  - Cross-Tabulation of Data (standard tables & long form)
  - Marginal Summaries of Cross Tabulations

- **Statistical Tests**
  - Chi Square Test
  - Kendall Rank Correlation
  - Fisher’s Exact Test
  - Student’s t-Test

- **Sampling**
  - Subsample (observations & variables)
  - Random Sampling
**ScaleR: High Performance Scalable Parallel External Memory Algorithms**

### Predictive Models
- Correlation, Covariance, and Sum of Squares/Cross-product Matrices
- Multiple Linear Regression
- Stepwise Regression
- Logistic Regression
- Generalized Linear Models (GLM)
  - All exponential family distributions: binomial, Gaussian, inverse Gaussian, Poisson, Tweedie. Standard link functions including: cauchit, identity, log, logit, probit. User defined distributions & link functions.
- Classification & Regression Trees
- Predictions/scoring for models
- Residuals for all models

### Statistical Modeling

### Data Visualization
- Histogram
- Line Plot
- Scatter Plot
- Lorenz Curve
- ROC Curves (actual data and predicted values)

### Machine Learning

### Cluster Analysis
- K-Means

### Classification
- Decision Trees

### Simulation
- Monte Carlo
Key ScaleR HPA features

- Handles an arbitrarily large number of rows in a fixed amount of memory
- Scales linearly with the number of rows
- Scales linearly with the number of nodes
- Scales well with the number of cores per node
- Scales well with the number of parameters
- Extremely high performance
Regression comparison using in-memory data: \texttt{lm()} vs \texttt{rxLinMod()}

Linear Model Estimation for In-Memory Data:
Independent Variables: 2 factors (100 and 20 levels) and one continuous

![Graph showing comparison between \texttt{lm()} in R and \texttt{rxLinMod()} in Revolution R]

\textbf{Note}: \texttt{lm} results in memory allocation errors for larger numbers of rows.
GLM comparison using in-memory data: glm() vs rxGlm()
# SAS HPA Benchmarking Comparison

## Logistic Regression

<table>
<thead>
<tr>
<th></th>
<th>SAS</th>
<th>Revolution R</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Rows of data</strong></td>
<td>1 billion</td>
<td>1 billion</td>
</tr>
<tr>
<td><strong>Parameters</strong></td>
<td>“just a few”</td>
<td>7</td>
</tr>
<tr>
<td><strong>Time</strong></td>
<td>80 seconds</td>
<td>44 seconds</td>
</tr>
<tr>
<td><strong>Data location</strong></td>
<td>In memory</td>
<td>On disk</td>
</tr>
<tr>
<td><strong>Nodes</strong></td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td><strong>Cores</strong></td>
<td>384</td>
<td>20</td>
</tr>
<tr>
<td><strong>RAM</strong></td>
<td>1,536 GB</td>
<td>80 GB</td>
</tr>
</tbody>
</table>

Revolution R is faster on the same amount of data, despite using approximately a 20\textsuperscript{th} as many cores, a 20\textsuperscript{th} as much RAM, a 6\textsuperscript{th} as many nodes, and not pre-loading data into RAM.

Revolution R Enterprise Delivers Performance at **2% of the Cost**

*As published by SAS in HPC Wire, April 21, 2011*
Allstate compares SAS, Hadoop and R for Big-Data Insurance Models

<table>
<thead>
<tr>
<th>Approach</th>
<th>Platform</th>
<th>Time to fit</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAS</td>
<td>16-core Sun Server</td>
<td>5 hours</td>
</tr>
<tr>
<td>rmr/MapReduce</td>
<td>10-node 80-core Hadoop Cluster</td>
<td>&gt; 10 hours</td>
</tr>
<tr>
<td>R</td>
<td>250 GB Server</td>
<td>Impossible (&gt; 3 days)</td>
</tr>
<tr>
<td>Revolution R Enterprise</td>
<td>5-node 20-core LSF cluster</td>
<td>5.7 minutes</td>
</tr>
</tbody>
</table>

Generalized linear model, 150 million observations, 70 degrees of freedom
http://blog.revolutionanalytics.com/2012/10/allstate-big-data-glm.html
R Productivity Environment (Windows)

- Script with type ahead and code snippets
- Solutions window for organizing code and data
- Sophisticated debugging with breakpoints, variable values etc.
- Packages installed and loaded
- Objects loaded in the R Environment
- Object details

http://www.revolutionanalytics.com/demos/revolution-productivity-environment/demo.htm
Deployment with Revolution R Enterprise

End User

Application Developer

R Programmer

Desktop Applications (e.g. Excel)

Business Intelligence (e.g. Jaspersoft)

Interactive Web Applications

Client libraries (JavaScript, Java, .NET)

HTTP/HTTPS – JSON/XML

RevoDeployR Web Services

Session Management

Authentication

Data/Script Management

Administration

R

R

R

R
Revolution R Enterprise

www.revolutionanalytics.com

1. High-performance R for multiprocessor systems
2. Modern Integrated Development Environment
3. Statistical Analysis of Terabyte-Class Data Sets
4. In-database R analytics with Hadoop\(^1\) and Netezza
5. Import SAS files without needing a SAS license
6. Deploy & Integrate R Applications via Web Services
7. Telephone and email technical support
8. Training and consulting services
9. Compatible with R packages
10. Free to Academics\(^2\)

\(^1\) Coming Soon \(^2\) One installed workstation per user - support not included
Hints for R faster R code

Wolfgang Raffelsberger
Hints for faster R code

UseR! 2013

W. Raffelsberger @ unistra.fr

LBGI Laboratoire de Bioinformatique et de Génomique Intégratives

igbmc Institut de génétique et de biologie moléculaire et cellulaire

Strasbourg, France
Hints for faster R code

- **R is vector oriented language**
  - Vectors and matrixes allow “parallel” calculation …
  - Avoid nested loops
  - Keep loops short (if you really need loops)

- **Some objects allow faster use/access**
  - e.g. matrix vs data.frame
Hints for faster R code /2

Let's make some toy data

```r
> set.seed(4321)                  # make it reproducible

> nMax <- 90000

> mat1 <- matrix(c(rnorm(nMax, 5), runif(nMax, 1, 10),
                   abs(rcauchy(nMax, loc=5, sca=0.1))), nr=nMax/2, byrow=T)

> head(mat1, n=3)
[1,] 4.573243 4.776388 5.717607 5.841446 4.871643 6.609347
[3,] 3.739015 6.139464 3.778218 6.573316 5.073478 3.824885

>
```
**Hints for faster R code**

### Calculate average across rows

```r
> system.time({
    rowAv <- numeric();
    for(i in 1:nrow(mat1)) rowAv <- c(rowAv, mean(mat1[i,]));
})
# loop over rows, slow !!
user  system elapsed
6.25  0.04  6.30

> system.time({
    rowAv <- rep(NA, nrow(mat1));
    for(i in 1:nrow(mat1)) rowAv[i] <- mean(mat1[i,]);
})
# loop, with indexing, better
user  system elapsed
0.77  0.00  0.77

> system.time(rowAv2 <- apply(mat1, 1, mean))
# (slightly) better
user  system elapsed
0.61  0.00  0.61

> system.time(rowAv2 <- rowMeans(mat1))
# best & simple code !
user  system elapsed
0    0    0

> 
```
Hints for faster R code

Calculate average across rows

```r
> system.time({
    rowAv <- numeric();
    for(i in 1:nrow(mat1)) rowAv <- c(rowAv, mean(mat1[i,]))
})
# loop over rows, slow !!
user  system elapsed
6.25  0.04   6.30

> system.time({
    rowAv <- rep(NA, nrow(mat1));
    for(i in 1:nrow(mat1)) rowAv[i] <- mean(mat1[i,])
})
# loop with indexing, better
user  system elapsed
0.77  0.00   0.77

> system.time(rowAv2 <- apply(mat1, 1, mean))
# (slightly) better
user  system elapsed
0.61  0.00   0.61

> system.time(rowAv2 <- rowMeans(mat1))
# best & simplest code !
user  system elapsed
0     0     0

> identical(rowAv, rowAv2)
# TRUE

> system.time(rowAv <- (mat1[,1]+mat1[,2]+mat1[,3]+mat1[,4]+mat1[,5]+mat1[,6])/6)
user  system elapsed
0     0     0

> system.time({
    rowAv <- rep(NA,nrow(mat1));
    for(i in 1:ncol(mat1)) rowAv <- rowAv + mat1[,i]; rowAv <- rowAv/ncol(mat1)
})
# quite good
user  system elapsed
0.03  0.00   0.04
```

Calculate average across rows
Calculate average across rows

... same, but as `data.frame`

```r
> df1 <- as.data.frame(mat1)
> system.time({
>   rowAv <- numeric();
>   for(i in 1:nrow(df1)) rowAv <- c(rowAv, mean(as.numeric(df1[i,])))
> }) # very slow!
   user  system elapsed
  17.69    0.11   17.84

> system.time({
>   rowAv <- rep(NA, nrow(df1));
>   for(i in 1:nrow(df1)) rowAv[i] <- mean(as.numeric(df1[i,]))
> }) # slow
   user  system elapsed
  11.80    0.04   11.86

> system.time( rowAv3 <- apply(df1, 1, mean) ) # better
   user  system elapsed
  0.73    0.00   0.73

> system.time( rowAv3 <- rowMeans(df1) ) # best!
   user  system elapsed
  0.02    0.00   0.01
```
Hints for faster R code /4

Calculate average across rows

... same, but as \texttt{data.frame}

```r
> df1 <- as.data.frame(mat1)
> system.time({
  rowAv <- numeric();
  for(i in 1:nrow(df1)) rowAv <- c(rowAv, mean(as.numeric(df1[i,]))) })  # very slow!
user  system elapsed
17.69   0.11  17.84

> system.time({
  rowAv <- rep(NA, nrow(df1));
  for(i in 1:nrow(df1)) rowAv[i] <- mean(as.numeric(df1[i,])) })  # slow
user  system elapsed
11.80   0.04  11.86

> system.time( rowAv3 <- apply(df1, 1, mean) )  # better
user  system elapsed
0.73   0.00  0.73

> system.time( rowAv3 <- rowMeans(df1) )  # best!
user  system elapsed
0.02   0.00  0.01

> identical(rowAv, rowAv3)  # TRUE

> system.time({
  rowAv <- rep(NA, nrow(df1));
  for(i in 1:ncol(df1)) rowAv <- rowAv + df1[,i]; rowAv <- rowAv/ncol(df1) })  # short loop, good
user  system elapsed
0.04   0.00  0.03
```
Hints for faster R code /5

Calculate average across rows

... same, but as `data.frame`

```r
> df1 <- as.data.frame(mat1)
> system.time({
    rowAv <- numeric();
    for(i in 1:nrow(df1)) rowAv <- c(rowAv, mean(as.numeric(df1[i,])))
}
user system elapsed
17.69 0.11 17.84
> system.time({
    rowAv <- rep(NA, nrow(df1));
    for(i in 1:nrow(df1)) rowAv[i] <- mean(as.numeric(df1[i,]))
}
user system elapsed
11.80 0.04 11.86
> system.time( rowAv3 <- apply(df1, 1, mean)
user system elapsed
0.73 0.00 0.73
> system.time( rowAv3 <- rowMeans(df1)
user system elapsed
0.02 0.00 0.01
> identical(rowAv, rowAv3)
> system.time({
    rowAv <- rep(NA, nrow(df1));
    for(i in 1:ncol(df1)) rowAv <- rowAv + df1[,i];
    rowAv <- rowAv / ncol(df1)
}
user system elapsed
0.04 0.00 0.03
```

![Diagram showing time of calculating row-means](image)
Hints for faster R code /6

Calculate average across rows

... but with multiple groups

```r
> colnames(mat1) <- colnames(df1) <- paste(rep(LETTERS[1:3],each=2),1:2,sep="")
> useCol <- gl(3,2)
> fMultAv <- function(dat,fact) tapply(dat,fact,mean)  # function for row means
> system.time({ rowAvs <- numeric(); for(i in 1:nrow(mat1)) rowAvs <- c(rowAvs,fMultAv)
          user  system elapsed
2033.18  0.22  2035.97
> system.time({ rowAvs <- matrix(nr=nrow(mat1),nc=3);
           for(i in 1:nrow(mat1)) {
             for(j in 1:length(levels(useCol))) rowAvs[i,j] <- mean(mat1[i,useCol==j])}
          # faster
          user  system elapsed
8.40    0.02   8.96
>```
Hints for faster R code /6

Calculate average across rows

... but with multiple groups

```r
> colnames(mat1) <- colnames(df1) <- paste(rep(LETTERS[1:3],each=2),1:2,sep="")
> useCol <- gl(3,2)
> fMultAv <- function(dat,fact) tapply(dat,fact,mean)        # function for row means
> system.time({ rowAvs <- numeric(); for(i in 1:nrow(mat1)) rowAvs <- c(rowAvs,fMultAv)
user  system elapsed
2033.18  0.22  2035.97
>
> system.time({ rowAvs <- matrix(nr=nrow(mat1),nc=3);
  for(i in 1:nrow(mat1)) {
    for(j in 1:length(levels(useCol))) rowAvs[i,j] <- mean(mat1[i,useCol==j])}
})          # faster
user  system elapsed
 8.40   0.02   8.96
>
> system.time({ rowAvs2 <- matrix(nr=nrow(mat1),nc=3);
  for(j in 1:length(levels(useCol))) rowAvs2[,j] <- rowMeans(mat1[,useCol==j])}
})          # best
user  system elapsed
 0.00   0.00   0.00
>
```
List example

> lst1 <- apply(mat1[,], 1, function(x) x[1:sample(2:ncol(mat1),1)])
> length(lst1)
[1] 45000

... mean of each list-element

> system.time({ lstAv2 <- numeric();
        for(i in 1:length(lst1)) lstAv2[i] <- mean(lst1[[i]]) })
    # loop: slow
    user  system elapsed
     6.34     0.00     6.45

> system.time( lstAv <- sapply(lst1, mean) )
    # faster
    user  system elapsed
      0.52     0.00     0.52

> identical(lstAv, lstAv2)
    # compare results
[1] TRUE

>
List example

get 2\textsuperscript{nd} lowest value of each list-element

```r
secLowestVal <- function(x) sort(x, na.last=T)[2]

# slower
system.time({
  secRank <- numeric();
  for(i in 1:length(lst1)) secRank[i] <- secLowestVal(lst1[[i]])
})

# faster
system.time( secRank <- sapply(lst1,secLowestVal) )
```
List example

get 2nd lowest value of each list-element

> secLowestVal <- function(x) sort(x, na.last=T)[2]
> system.time(
  secRank <- numeric();
  for(i in 1:length(lst1)) secRank[i] <- secLowestVal(lst1[[i]])
) # slower
    user  system elapsed
   8.16    0.08   8.30

> system.time( secRank <- sapply(lst1,secLowestVal) ) # faster
    user  system elapsed
   2.63    0.02   2.64

... with function calling .Internal

> secLowestVal2 <- function(x) .Internal(qsort(x, FALSE))[2]
> system.time(
  secRank <- numeric();
  for(i in 1:length(lst1)) secRank[i] <- secLowestVal2(lst1[[i]])
) # slower
    user  system elapsed
   5.63    0.01   5.65

> system.time( secRank <- sapply(lst1,secLowestVal2) ) # faster
    user  system elapsed
   0.11    0.00   0.11
Hints for faster R code

• **R is vector oriented language**
  - Vectors and matrixes allow “parallel” calculation …
  - Avoid nested loops
  - Keep loops short (if you really need loops)

• **Some objects allow faster use/access**
  - e.g. matrix vs data.frame

• **Indexing**
  - precise indexing faster
    - eg : matr[sel1,] vs matr[which(sel1),]

• **High level commands, .Internal()**
  - differences in degree of checking
    - eg : read.table() and scan()
Hints for faster R code /10

A few References:

- **R-inferno** by Patrick Burns (Burns statistics)

- **Tricks and Traps for Young Players** by Ray D Brownrigg
  http://www.statistik.uni-dortmund.de/useR-2008/slides/Brownrigg.pdf

- **R-help** mailing list

Thank you for listening
R packages with PISA data. Let's learn about education!

Przemyslaw Biecek
R and PISA. Let’s learn about education!

Przemyslaw.Biecek@gmail.com

MIM UW

Outline:

40 sec PISA - what is it?
40 sec PISA - what is measured?
40 sec PISA - how to load it to R?
140 sec PISA - some examples, what you can do with it [plots, plots, plots].
PISA - what is it?

From Wikipedia:

The Programme for International Student Assessment (PISA) is a worldwide study by the Organisation for Economic Co-operation and Development (OECD) in member and non-member nations of 15-year-old school pupils’ scholastic performance on mathematics, science, and reading. It was first performed in 2000 and then repeated every three years. It is done with view to improving education policies and outcomes.

520,000 15-year-old students representing 74 nations and territories participated in PISA 2009.

Raw data, findings and reports are freely available on the OECD website.
PISA - what is it?

From Wikipedia:

The Programme for International Student Assessment (PISA) is a worldwide study by the Organisation for Economic Co-operation and Development (OECD) in member and non-member nations of 15-year-old school pupils’ scholastic performance on mathematics, science, and reading. It was first performed in 2000 and then repeated every three years. It is done with view to improving education policies and outcomes.

520,000 15-year-old students representing 74 nations and territories participated in PISA 2009.

Raw data, findings and reports are freely available on the OECD website.
PISA - what is it?

For each student performance in math, reading and science is measured based on dozens of items/problems to solve.

In addition there are hundreds of questions in the student questionnaire, like:

- Do you enjoy reading about mathematics?
- Do you talk with friends about mathematics?
- Did your teacher explain the concepts well this week?
- In the last two full weeks of school, how many times did you skip some classes?
- How many books are there in your home?
- Occupation of your parents?
PISA - what is it?

A lot of questions for parents, like:

- Do you eat the main meal with your child around a table?
- Do you agree: It is important to have good mathematics knowledge and skills in order to get any good job in today’s world?
- How important it was: The school is at a short distance to home?

A lot of questions for school officials, like:

- Number of boys / girls in your school.
- Does it affects your school: A lack of qualified teachers of other subjects?
- Do you have in your school: Band, orchestra or choir?
- Are students arriving late for school?

There are few R packages that give you access to this data.

- **pisa** (maintained by Jason Bryer, github/jbryer) with data from PISA 2009,

Please note, that these datasets are large [hundreds of MBs after compression].
library(devtools)
install_github("PISA2009lite", "pbiecek")
library(PISA2009lite)

dim(student2009)
## [1] 515958  437

dim(parent2009)
## [1] 106287  90

dim(school2009)
## [1] 18641  247

dim(item2009)
## [1] 515958  273
Math performance and number of books in home
Math performance and number of books in home
How familiar are you with given concept?
How familiar are you with given concept?
Strong and weak sides of students from different countries
Strong and weak sides of students from different countries
Who is taking extra math hours?
Who is taking extra math hours?
Reproducible Presentations with RStudio

JJ Allaire
R work journal

Alex Zolotovitski
R Work Journal

Alex Zolotovitski, www.zolot.us/work/UseR-2013

Medio Inc, www.medio.com
1. **R Work Journal**: `Code2HTML(); MakeRWJournals(); createRWJalbum()`

`Code2HTML()` **Features**:

1. Transforms `.R` file into self-documented `.html` file, containing all R code with output pics, headers, table of contents and gallery.

2. The titles in body and contents are clickable to navigate from contents to body and back.

3. The pics are clickable to resize.

4. The html file has partly R syntax highlighted. It is possible to do the full R syntax highlighting in resulting html, but the result file becomes almost twice heavier.

5. Parts of the result html file could be folded.

6. If you in browser fold TOC, select all, copy and paste from browser to a text editor, you should get the pure original R file.

7. If modify `.R` code, recreate `.html` is fast.

8. It is not replacement of knitr or sweave, because output is not a document to print, but rather an R work journal.
```r
options(help = 'html', digit = NULL) # q:

onWin = Sys.getenv('R_PLATFORM')
if(onWin){root = 'R:/'}
else{root = 'm:'}

source(file.path(root, 'WK')
source("R:\work\R-svn-ass\")
source(file.path(root, '71'))

source('T:\work\UseR-2013/)
source('T:\work\UseR-2013/)
source('T:\work\UseR-2013/)

library(RColorBrewer) # disppalette(c(adjustcolor(cn(')
library(plyr)
```
```r
mtext('isTrans ~ CustomerType+ UA {+ cntD+ dur+ LifeSpan+ MTI}
HttP://lift rf for models isTrans ~ CustomerType+ UA {+ cntD-
# Pic_95. Lift rf for models isTrans ~ CustomerType+ UA {
# 2008-10-07 22:50:11:: Image saved: lo(\'m:\/95_ABC_LTV-2smpAc
cc()  # Code2HTML (theFile)
```
--- init ---

```r
# Init.
```

---

--- Act and Trans Log Files Inventory: analysys of DateTimeStamps (DTS) ---

--- Hadoop files ---

```r
# Plot 61. Activity logs
```

---

```r
# Plot 62. Activity logs
```
1. 95_ABC (file://f)
2. 97_tutorial-demo (www.zolot.us)