Robust Location and Scatter Estimators for Multivariate Data Analysis {robustbase}, {rrcov}

Valentin Todorov

valentin.todorov@chello.at

Outline

- Background and Motivation
- Computing the Robust Estimates
  - Definition and computation
    - MCD, OGK, S, M
  - Object model for robust estimation
  - Comparison to other implementations
- Applications
  - Hotelling T^2
  - Robust Linear Discriminant Analysis
- Conclusions and future work

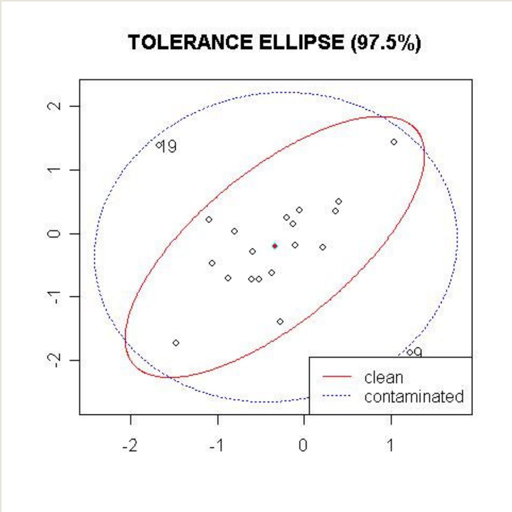
Multivariate location and scatter

- *Location*: coordinate-wise mean
- *Scatter*: covariance matrix
  - Variances of the variables on the diagonal
  - Covariance of two variables as off-diagonal elements

- Optimally estimated by the sample mean and sample covariance matrix at any multivariate normal model
- Essential to a number of multivariate data analyses methods

- But extremely sensitive to outlying observations

Example



- Marona & Yohai (1998)
- **rrcov**: data set **maryo**
- A bivariate data set with:

$$n = 20, \mu = \begin{pmatrix} 0 & 0 \end{pmatrix}$$

$$S = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

- sample correlation: 0.81
- interchange the largest and smallest value in the first coordinate
- the sample correlation becomes 0.05

Software for robust estimation of multivariate location and scatter

- **S-Plus** – *covRob* in the *Robust* library
- **Matlab** – *mcdcov* in the toolbox *LIBRA*
- **SAS/IML** – *MCD call*
- **R** – *cov.rob* and *cov.mcd* in *MASS*

- **R** – *covMcd* in *{robustbase}*
- ***R*** – *CovMcd, CovOgk, CovMest {rrcov}*

Motivation

- **R** 2.3.1: *cov.rob* (*cov*.mcd) in MASS, but
  - Implements *C-Step* similar to the one in Rousseeuw & Van Driessen (1999) but no partitioning and no nesting -> very slow for larger data sets
  - No small sample corrections
  - No generic functions *print/show, summary, plot*
  - No graphical and diagnostic tools

rrcov ➔ robustbase

\- Port of the Fortran code for FAST-MCD and FAST-LTS of Rousseeuw and Van Driessen

\- Functions *covMcd*, *ltsReg* and the corresponding help files

\+ Datasets - Rousseeuw and Leroy (1987), Milk - Daudin (1988), etc.

\+ Generic functions *print* and *summary* for *covMcd*

\+ Graphical and diagnostic tools based on the robust and classical Mahalanobis distances - *plot.mcd*

\+ Formula interface and generic functions *print*, *summary* and *predict* for *ltsReg*

\+ Graphical and diagnostic tools based on the residual - *plot.lts*

rrcov

\+ Constrained M-estimates of location and covariance - Rocke (1996)

\+ Orthogonalized Gnanadesikan-Kettering (OGK) – Maronna and Zamar (2002)

\+ S4 object model

  \+ CovMcd

  \+ CovOgk

  \+ CovMest

rrcov

- CovSest: S estimates - FAST-S Salibian & Yohai (2005)
- Trellis style graphics
- Hotelling T^2
- Robust Linear Discriminant Analysis with option for Stepwise selection of variables
- More data sets

Outline

- Background and Motivation
- Computing the Robust Estimates
  - Definition and computation
    - MCD, OGK, S, M
  - Object model for robust estimation
  - Comparison to other implementations
- Applications
  - Hotelling T^2
  - Robust Linear Discriminant Analysis
- Conclusions and future work

Minimum Covariance Determinant Estimator

Given a p dimensional data set $\mathbf{X}=\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$

- The **MCD** estimator (Rousseeuw, 84) is defined by
  - the subset of h observations out of n whose classical covariance matrix has a smallest determinant
  - the MCD location estimator **T** is defined by the mean of that subset
  - the MCD scatter estimator C is a multiple of its covariance matrix
  - $n/2 <= h < n$; $h=[(n+p+1)/2]$ yields maximal BDP

Computing of MCD: FAST-MCD

- Consists of three phases: basic *C-step* iteration, partitioning and nesting
- ***C-step***: move from one approximation $(\boldsymbol{T_1},\boldsymbol{C_1})$ of MCD of a data set $\boldsymbol{X}=\{\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}\}$ to a new one $(\boldsymbol{T_2},\boldsymbol{C_2})$ with possibly lower determinant by computing the distances relative to $(\boldsymbol{T_1},\boldsymbol{C_1})$ and then computing $(\boldsymbol{T_2},\boldsymbol{C_2})$ for the h observations with smallest distances.
- ***C-step* iteration**:
  - Repeat a number of times (say 500) {
    - start from a trial subset of h points and perform several *C-step*s
    - keep the 10 best solutions

    }
  - From each of these solutions carry out *C-step*s until convergence and select the best result

Computing of MCD: FAST-MCD

- ***Partitioning***: If the data set is large (e.g. > 600) it is partitioned into (five) disjoint subsets
  - Carry out *C-step*s iterations for each of the subsets
  - Use the best (50) solutions as starting points for *C-step*s on the entire data set and again keep the best 10 solutions
  - Iterate these 10 solutions to convergence
- ***Nesting***: If the data set is larger then (say 1500)
  - draw a random subset and apply the partitioning procedure to it
  - use the 10 best solutions from the partitioning phase for iterations on the entire data set
- The number of solutions used and the number of *C-step*s performed on the entire data set depend on its size

Compound Estimators

- MVE and MCD - a first stage procedure
- Rousseeuw and Leroy 87, Rousseeuw and van Zomeren 91 - one step re-weighting
- One-step M-estimates using Huber or Hampel function
- Woodruff and Rocke 93, 96 - use MCD as a starting point for S-estimation or constraint M-estimation

Using the estimators: Example

Delivery Time Data

- Rousseeuw and Leroy (1987), page 155, table 23 (Montgomery and Peck (1982)).
- 25 observations in 3 variables
  - X1 Number of Products
  - X2 Distance
  - Y Delivery time
- The aim is to explain the time required to service a vending machine (Y) by means of the number of products stocked (X1) and the distance walked by the route driver (X2).
- *delivery.x* – the X-part of the data set


```
>library(rrcov)
Loading required package: robustbase
Loading required package: MASS
Scalable Robust Estimators with High Breakdown Point (version 0.3-03)

>data(delivery)
>delivery.x <- as.matrix(delivery[, 1:2])
>mcd <- CovMcd(delivery.x)
>mcd

Call:
CovMcd(x = delivery.x)

Robust Estimate of Location:
  n.prod  distance
   5.895   268.053

Robust Estimate of Covariance:
          n.prod    distance
n.prod      12.30     232.98
distance   232.98   56158.36
```

```
> summary(mcd)
Call:
CovMcd(x = delivery.x)

Robust Estimate of Location:
  n.prod  distance
   5.895   268.053

Robust Estimate of Covariance:
          n.prod     distance
n.prod       12.30     232.98
distance    232.98   56158.36

Eigenvalues of covariance matrix:
[1]  56159.32     11.34

Robust Distances:
[1] 1.51872   0.68199   0.99165   0.73930   0.27939   0.13181   1.37029
[8] 0.21985  57.68290   2.48532   9.30993   1.70046   0.30187   0.71296
… …
```


The *CovMcd* object

- *CovMcd()* returns an S4 object of class *CovMcd*

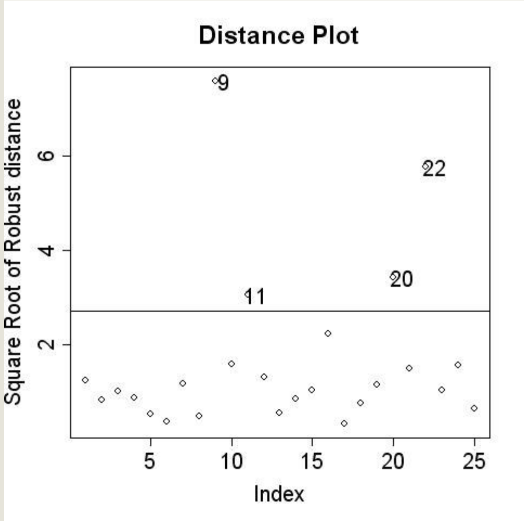
```
> data.class(mcd)
[1] "CovMcd"
```

- Input parameters used for controlling the estimation algorithm: *alpha, quan, method, n.obs*, etc.
- Raw MCD estimates: *crit, best, raw.center, raw.cov, raw.mah, raw.wt*
- Final (re-weighted) estimates – *center, cov, mah, wt*

The *CovMcd* object (cont.)

- *show(mcd)*
- *summary(mcd)* – additionally prints the eigenvalues of the covariance and the robust distances.
- *plot(mcd)* - shows the Mahalanobis distances based on the robust and classical estimates of the location and the scatter matrix in different plots.
  - distance plot
  - distance-distance plot
  - chi-Square plot
  - tolerance ellipses
  - scree plot

Plot of the Robust Distances



- The Mahalanobis distances based on the robust estimates – the outliers have large ***Rd$_i$***
- A line is drown at

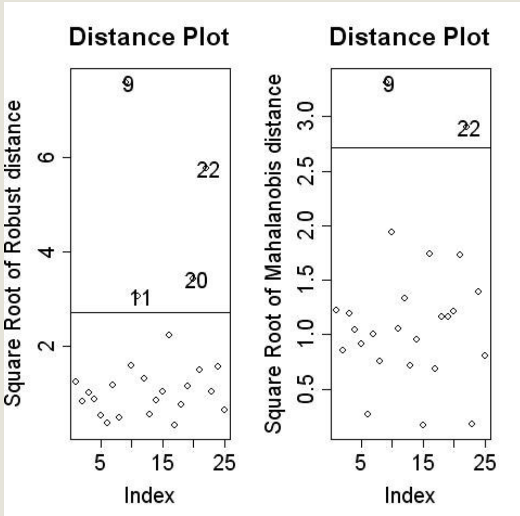
$$y = cutoff = \sqrt{\chi^2_{p,0.975}}$$

- The observations with

$$RD_i \geq cutoff = \sqrt{\chi^2_{p,0.975}}$$

are identified by their subscript

Plot of the Robust and Classical Distances

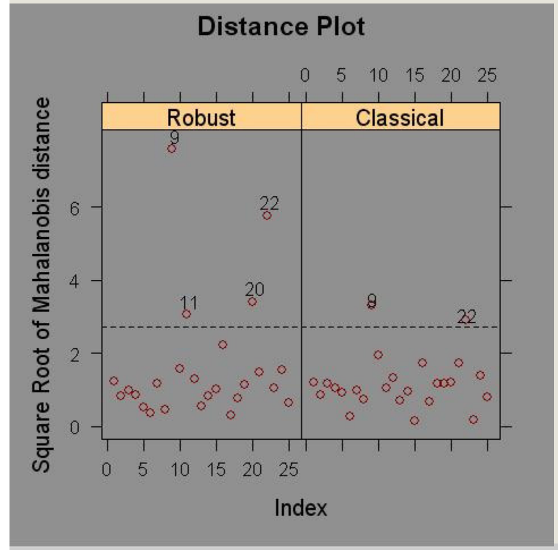


- With the option

  class=TRUE

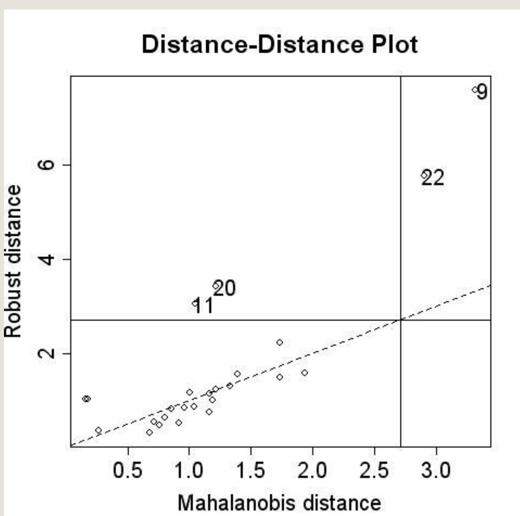
  *b*oth the robust and classical distances are shown in parallel panels
- The horizontal scales are different for the two displays

Plot of the Robust and Classical Distances – trellis style



- using R *lattice* package
- using function *xyplot* instead of *plot*
- functions *ltext* instead of *text*, *panel.abline* instead of *abline*, etc.
- the plot must be completed in a single function call – all the actions are described in a panel function and this function is passed as a *panel* argument

Robust distances vs. Mahalanobis distances



- Robust distances versus Mahalanobis distances
- The dashed line -

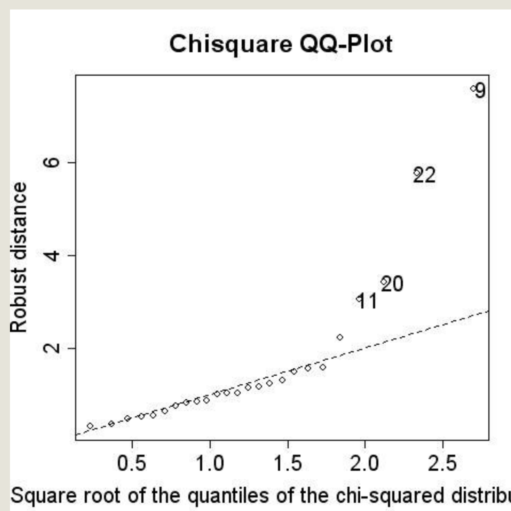
$$RD_i = MD_i$$

- The horizontal and vertical lines:

$$y = \sqrt{\chi^2_{p,0.975}}$$

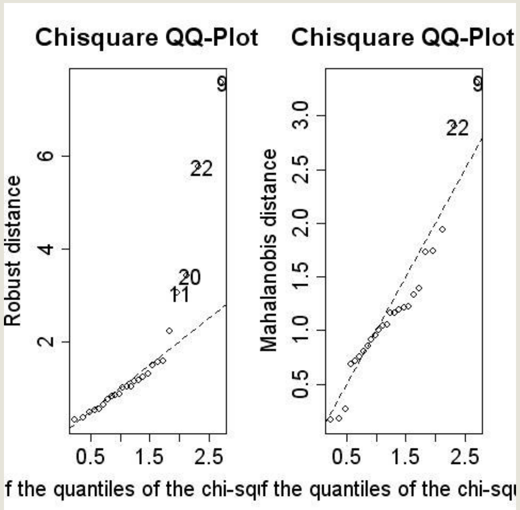
$$x = \sqrt{\chi^2_{p,0.975}}$$

Chi-Square QQ-Plot



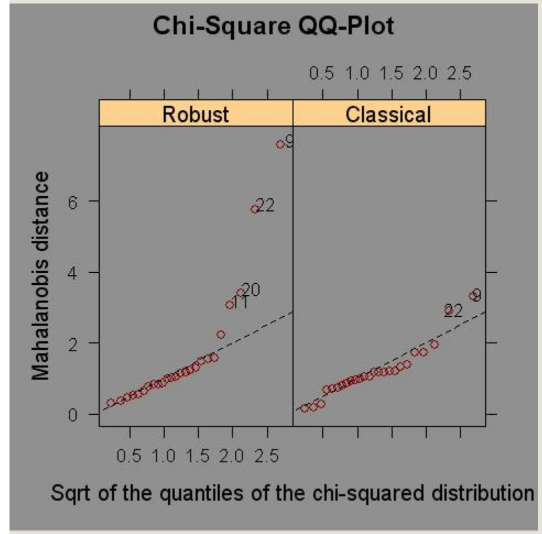
- A Quantile-Quantile comparison plot of the Robust distances versus the square root of the quantiles of the chi-squared distribution.

Chi-Square QQ-Plot



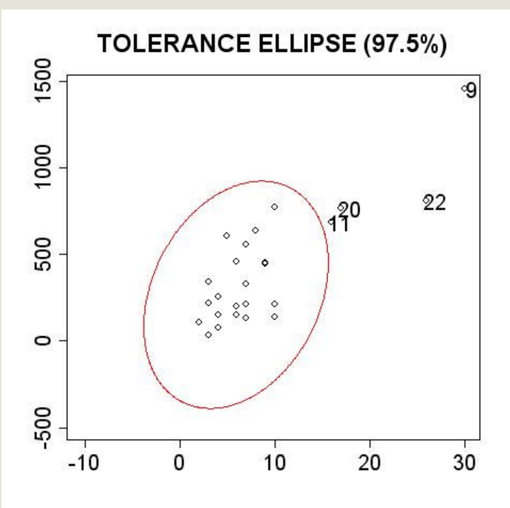
- A Quantile-Quantile comparison plot of the Robust distances and the Mahalanobis distances versus the square root of the quantiles of the chi-squared distribution.

Chi-Square QQ-Plot– trellis style



- using R *lattice* package

Robust Tolerance Ellipse



- Scatter plot of the data
- Superimposed is the 97.5% robust confidence ellipse defined by the set of points with robust distances

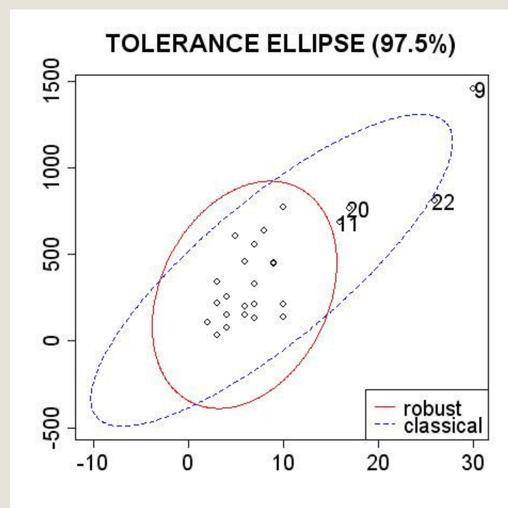
$$RD_i = \sqrt{\chi^2_{p,0.975}}$$

- Only in case of bivariate data
- The observations with

$$RD_i \geq cutoff = \sqrt{\chi^2_{p,0.975}}$$

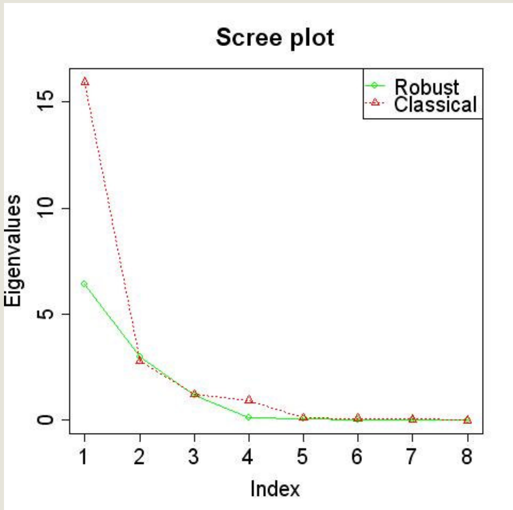
are identified by their subscript

Robust and Classical Tolerance Ellipses



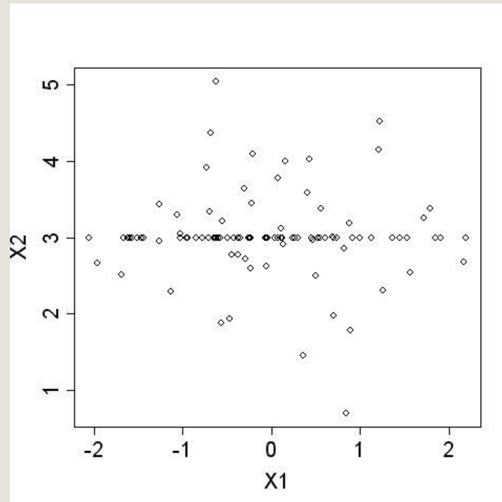
- Scatter plot of the data
- Superimposed are the 97.5% robust and classical confidence ellipses
- Only in case of bivariate data

Eigenvalues Plot



- Eigenvalues comparison plot for the Milk data set – Daudin (1988)
- Find out if there is much difference between the classical and robust covariance (or correlation) estimates.

Handling exact fits



- More than *h* observations lie on a hyperplane
- Although **C** is singular, the algorithm yields an MCD estimate of **T** and **C** from which the equation of the hyperplane can be computed.

Handling exact fits (cont.)

```
The covariance matrix has become singular during the
iterations of the MCD algorithm. There are 55 observations
in the entire dataset of 100 observations that lie on the
line with equation
    0 (x_i1-m_1)+ -1 (x_i2-m_2)=0
with (m_1,m_2) the mean of these observations.
Call: covMcd(x = xx)
Center:
     X1       X2
-0.2661   3.0000
Covariance Matrix:
           X1          X2
X1   3.617e+00  -4.410e-16
X2  -4.410e-16   0.000e+00
```

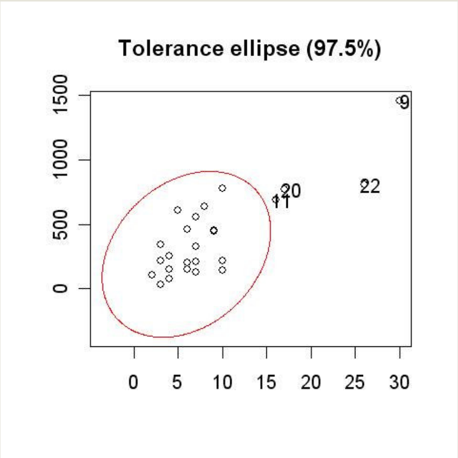

Orthogonalized Gnanadesikan-Kettering (OGK)

- *CovOgk(x, niter = 2, beta = 0.9, control)*
- Pairwise covariance estimator, Maronna and Zamar (2002)
- The pairwise covariances are computed using the estimator proposed by Gnanadesikan and Kettering (1972), but other estimators can be used too
- Adjustment is applied to ensure that the obtained covariance matrix is positive definite
- To improve efficiency the OGK estimates are re-weighted in a similar way as the MCD ones
- The returned S4 object *CovMest* inherits from *CovRobust*, so all methods of *CovRobust* can be used

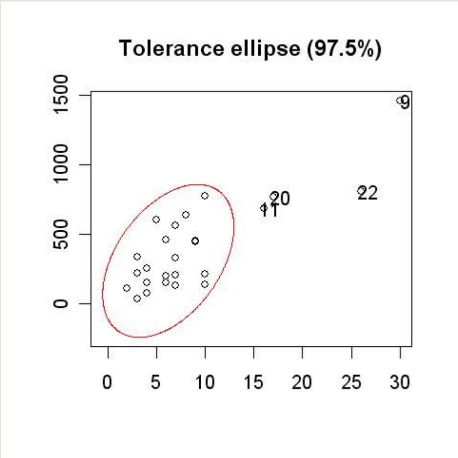
```
>ogk <- CovOgk(delivery.x)
>ogk

Call:
CovOgk(x = delivery.x)

Robust Estimate of Location:
  n.prod  distance
    6.19    309.71

Robust Estimate of Covariance:
          n.prod     distance
n.prod        6.154    222.769
distance    222.769  40826.776

> data.class(ogk)
[1] "CovOgk"
```


Delivery Data

MCD



OGK



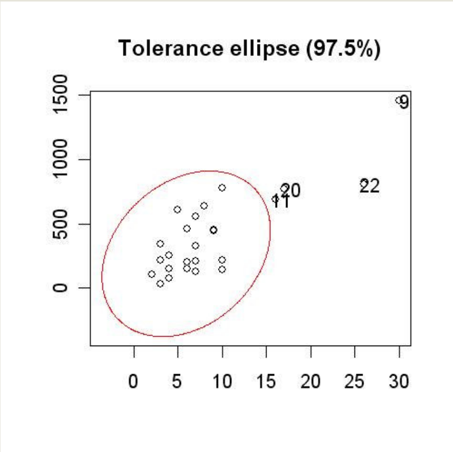
M-estimates of location and scatter

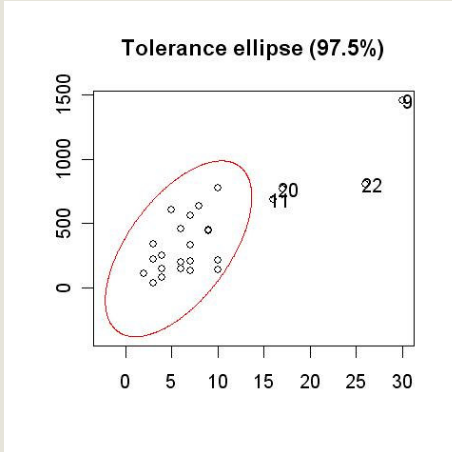
- *CovMest(x, r = 0.45, arp = 0.05, eps=1e-3, maxiter=120, control, t0, S0)*
- Constrained M-estimates, Rocke (1996)
- Starts with highly robust initial estimate ($\mathbf{t}^0$,$\mathbf{S}^0$) – MCD
- M iterations using translated biweight function
- Two parameters – *c* and *M* – specify the desired breakdown point and asymptotic rejection probability
- *(t0, S0)* are the initial robust estimates. If omitted, *CovMcd* will be called to compute it
- The returned S4 object *CovMest* inherits from *CovRobust*, so all methods of *CovRobust* can be used


```
>mest <- CovMest(delivery.x)
>mest

Call:
CovMest(x = delivery.x)

Robust Estimate of Location:
  n.prod  distance
   5.737   305.112

Robust Estimate of Covariance:
          n.prod     distance
n.prod        8.541    434.224
distance    434.224  63421.639

> data.class(mest)
[1] "CovMest"
```

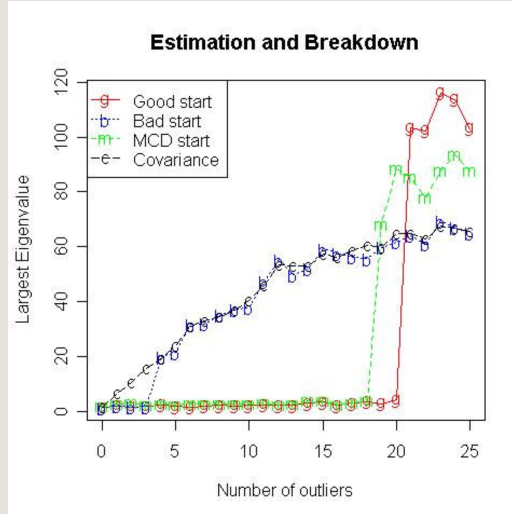
Delivery Data

MCD **M**





Breakdown of the constrained M-estimates



- Rocke (1996)
- Generate a data set:
  $n = 50, p = 10, N(0, I_p)$
- shift *i* observations to distance sqrt(250) from the origin
- Iterate from:
  - *Good start* – mean and covariance of the good portion of the data
  - *Bad start* – mean and covariance of all the data
  - *MCD start* – the MCD estimates
- Measure of quality: the largest eigenvalue of the cov.matrix

S-estimates of location and scatter

- *CovSest(x, nsamp=20, seed=0, control)*

- S-estimates: introduced by Rowsseeuw and Leroy (1987) and further studied by Davies (1987), Lopuhaä (1989)
- Fast-S algorithm based on the one for regression proposed by Salibian and Yohai (2005)
- Similar to FAST-MCD (C-step, partitioning, nesting)
- Ideas from Ruppert's SURREAL (1992)
- The returned S4 object *CovSest* inherits from *CovRobust*, so all methods of *CovRobust* can be used

Outline

- Background and Motivation
- Computing the Robust Estimates
  - Definition and computation
    - MCD, OGK, S, M
  - **Object model for robust estimation**
  - Comparison to other implementations
- Applications
  - Hotelling T^2
  - Robust Linear Discriminant Analysis
- Conclusions and future work

The object model: (simple) naming convention

- There is no agreed naming convention (coding rules) in R.
- These are several simple rules, following the recommended Java/Sun style (see also Bengtsson 2005):
  - Class, function, method and variable names are alphanumeric, do not contain "_" or "." but rather use interchanging lower and upper case
  - Class names start with an uppercase letter
  - Methods, functions, and variables start with a lowercase letter
  - Exception are functions returning an object of a given class (i.e. constructors) – they have the same name as the class
  - Variables and methods which are not intended to be seen by the user – i.e. private members - start with "."
  - Violate this rules whenever necessary to maintain compatibility

The object model: accessor methods

- Encapsulation and information hiding
- Accessor methods: methods used to examine or modify the members of a class
- Accessors in R (same name as the slot):

  cc <- a(obj)

  a(obj) <- cc
- Accessors in *rrcov* - *getXXX() and setXXX()*

  cc <- getA(obj)

  setA(obj, cc)
- Examples:
  - *getCov(), getCenter()*
  - *getMah()* – on demand computation
  - *getCorr()* – non existing slots

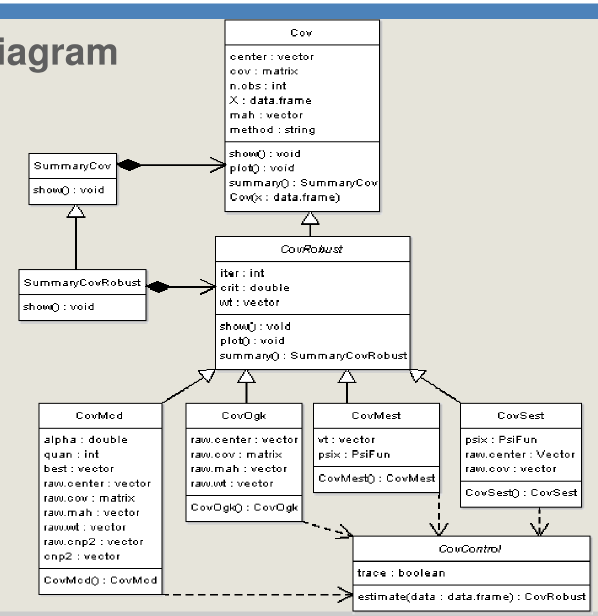
The object model: coexistence of S3 and S4

- A common problem when porting S3 classes and functions to S4 is what names to choose for the new classes and functions
- In rrcov the Java approach is used:
  - Choose freely names for the new S4 classes and corresponding functions
  - Leave the old S3 classes and functions but mark them as "**deprecated**": i.e. going to be made invalid or obsolete in future versions. The deprecated functions issue an warning when called:

    Warning: [deprecation] covMcd in robustbase has been deprecated

  - Add a package-wide variable which can be used to suppress these warnings

Class Diagram



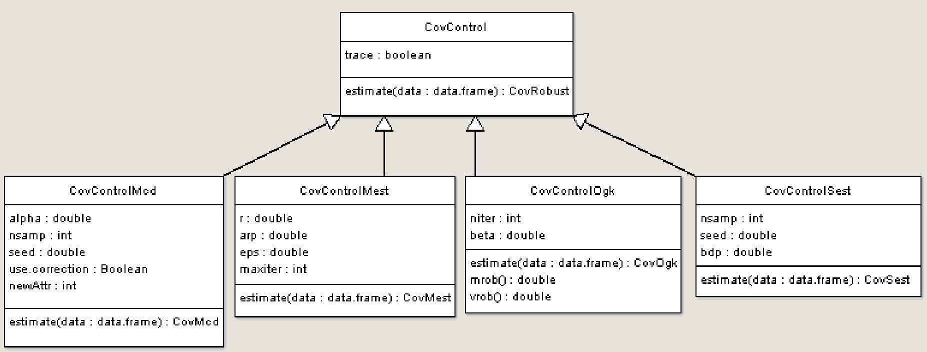
Controlling the estimation options

- MCD
  - *nsamp* – number of trial subsamples (500)
  - *alpha* – controls the size of the subsets over which the determinant is minimized. Possible values between 0.5 and 1, default 0.5
  - *seed* – seed for the Fortran random generator (0)
  - *trace* – intermediate output (FALSE)
- M
  - *r* – required breakdown point (0.45)
  - *arp* – asympthotic rejection point, i.e. the fraction of points receiving zero weight (0.05)
  - *eps* – a numeric value specifying the required relative precision of the solution of the M-estimate (1e-3)
  - *maxiter* – maximum number of iterations allowed in the computation of the M-estimate (120)

Controlling the estimation options (cont.)

- OGK
  - *niter* – number of iterations, usually 1 or 2
  - *beta* – coverage parameter for the final re-weighted estimate
  - *mrob* – function for computing the robust univariate location and dispersion - defaults to the 'tau scale' defined in Yohai and Zamar (1998)
  - *vrob* – function for computing robust estimate of covariance between two random vectors - defaults the one proposed by Gnanadesikan and Kettenring (1972)

Class Diagram: Control objects



Using the Control structure

```
>mcd <- CovMcd(delivery.x)
```

or

```
>ctrl <- CovControlMcd(alpha=0.75)
>mcd <- CovMcd(delivery.x, control=ctrl)
```

or

```
>mcd <- CovMcd(delivery.x, control=CovControlMcd(alpha=0.75))
```

or use the generic *estimate()*

```
>ctrl <- CovControlMcd(alpha=0.75)
>mcd <- estimate(ctrl, delivery.x)
```

or

```
>mcd <- estimate(CovControlMcd(alpha=0.75), delivery.x)
>ogk <- estimate(CovControlOgk(), delivery.x)
>mest <- estimate(CovControlMest(), delivery.x)
```

Using the Control structure (cont.)

- Let R choose a suitable estimation method

```
>cov <- estimate(CovControl(), delivery.x)
```

or

```
>cov <- estimate(x=delivery.x)
>getMethod(cov)
[1] "Minimum Covariance Determinant Estimator"

>cov

Call:
CovMcd(x = x)

Robust Estimate of Location:
  n.prod  distance
   5.895   268.053
… …
```


Robust Location and Scatter Estimation

Using the Control structure (cont.)

- Loop over different estimation methods

```
>cc <- list(CovControlMcd(), CovControlMest(), CovControlOgk())
>clist <- sapply(cc, estimate, x=delivery.x)
>sapply(clist, data.class)
[1] "CovMcd"  "CovMest" "CovOgk"

>sapply(clist, getMethod)

[1] "Minimum Covariance Determinant Estimator"
[2] "M-Estimates"
[3] "Orthogonalized Gnanadesikan-Kettenring Estimator"

>clist <- estimate(cc, delivery.x)

>sapply(clist, data.class)
[1] "CovMcd"  "CovMest" "CovOgk"
```


Robust Location and Scatter Estimation

Outline

- Background and Motivation
- Computing the Robust Estimates
  - Definition and computation
    - MCD, OGK, S, M
  - Object model for robust estimation
  - **Comparison to other implementations**
- Applications
  - Hotelling T^2
  - Robust Linear Discriminant Analysis
- Conclusions and future work

Robust Location and Scatter Estimation

Comparison to other implementations

- **R** 2.3.1 – *cov.rob* (*cov.*mcd) in MASS
  - No access to the “raw” MCD estimates, no small sample corrections
  - Implemented as native code in C using the memory management and other facilities of R
  - Implements *C-Step* similar to the one in Rousseeuw & Van Driessen (1999) but no partitioning and no nesting
  - No generic functions *print, summary, plot*
- **Matlab** 7.0 (R 14) - *mcdcov* in the toolbox *LIBRA* – Verboven and Hubert (2005)
  - Raw MCD estimates and re-weighted estimates, small sample corrections not used
  - Pure *Matlab* code
  - Diagnostic graphics

Comparison to other implementations (cont.)

- **S-PLUS** 6.2 – function *covRob* in the *Robust* library which implements several HBDP covariance estimates. The user can choose one of
  - (i) Donoho-Stahel projection based estimator,
  - (ii) Fast MCD algorithm of Rousseeuw and Van Driessen,
  - (iii) quadrant correlation based pairwise estimator or Gnanadesikan-Kettenring pairwise estimator (Maronna and Zamar (2002)
  - (iv) *auto* – let the program select an estimate based on the size of the problem
- **SAS/IML** – MCD call

Time comparison

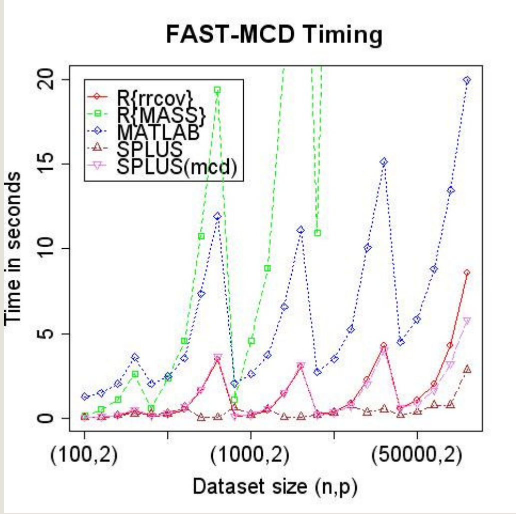
- Large data sets:
  - n=100, 500, 1000, 10000, 50000 and
  - p=2, 5, 10, 20, 30
- Shift outliers: $(1-\varepsilon)N_p(\mathbf{0},\mathbf{I}_p)+\varepsilon N_p(\mathbf{b},\mathbf{I}_p)$
  - with $\mathbf{b}=(10,\ldots,10)$ and $\varepsilon<0.5$
- Default options *nsamp=500* and *alpha=0.5*
- Average over 100 runs

Time comparison (cont.)

- PC 3 Ghz, 1Gb Memory, Windows XP Profesional

- R 2.3.1
- rrcov 0.3-3
- Matlab 7.0 (R12)
- S-PLUS 6.2

Time comparison (cont.)



- ***S-PLUS*** – uniformly fastest because of the use of the pairwise algorithms
- ***rrcov*** *and S-PLUS* with option *mcd* coincide
- ***Matlab*** – uniformly slower than *rrcov* and *S-PLUS* because of the interpreted code
- ***MASS*** – slowest because of not using partitioning and nesting

Outline

- Background and Motivation
- Computing the Robust Estimates
  - Definition and computation
    - MCD, OGK, S, M
  - Object model for robust estimation
  - Comparison to other implementations
- Applications
  - Hotelling T2
  - Robust Linear Discriminant Analysis
- Conclusions and future work

Robust Hotelling test

- *HotellingTsq(x, mu0, alpha=0.05, control)*
- Performs one sample hypothesis test for the center based on a robust version of the Hotelling T^2 statistic – Willems et al (2001)
- *U*ses the re-weighted MCD estimates
- T^2-statistic, p-value and cutoff value for the specified *alpha*
- Simultaneous confidence intervals for the components of the mean vector are also computed
- Returns an S4 object of class *HotellingTsq*
- Methods:
  - *show*

Robust Linear Discriminant Analysis

- *Linda(x, grouping, prior = proportions, step=FALSE, control)*
- *Linda(formula, data, prior = proportions, step=FALSE, control)*

- Uses one of the available robust location and scatter estimators
- Several ways to compute the within-group covariance matrix – Todorov (1990), He (2000), Hubert and Van Driessen (2004)
- Stepwise selection of the variables – Todorov (2005)
- Returns an S4 object of class *Linda*
- Methods *show, summary, predict* and *plot*

Outline

- Background and Motivation
- Computing the Robust Estimates
  - Definition and computation
    - MCD, OGK, S, M
  - Object model for robust estimation
  - Comparison to other implementations
- Applications
  - Hotelling T2
  - Robust Linear Discriminant Analysis
- **Conclusions and future work**

Future work

- Finalize and release the already implemented features:
  - CovSest: S estimates - FAST-S Salibian & Yohai (2005)
  - Hotelling T^2
  - Robust Linear Discriminant Analysis with option for Stepwise selection of variables
- More data sets
- Trellis style graphics