PCA by Projection Pursuit
The Package *pcaPP*

Heinrich Fritz
Vienna University of Technology, Austria

Vienna, Austria

June, 2006

Vienna University of Technology

Joint work with . . .

P. Filzmoser
Department of Statistics and Probability Theory
Vienna University of Technology, Austria

C. Croux
Department of Applied Economics
K.U. Leuven, Belgium

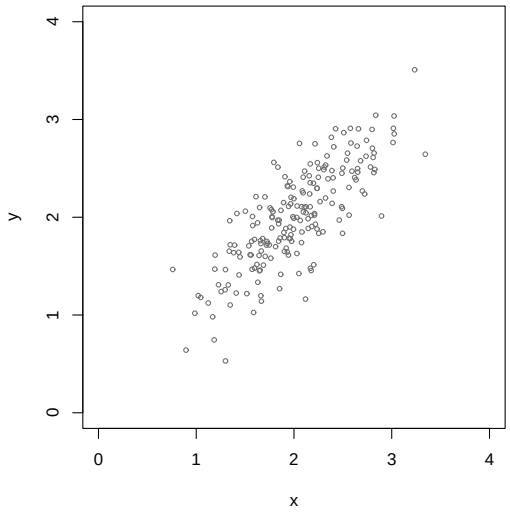
M.R. Oliveira
Department of Mathematics
Instituto Superior Técnico, Lisbon, Portugal

K. Kalcher
Vienna University of Technology, Austria

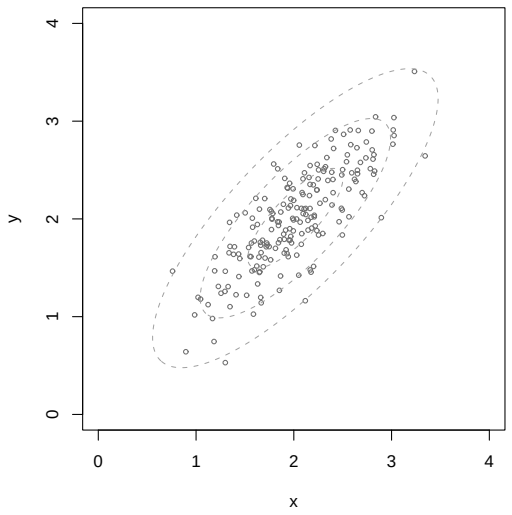
Agenda

- Principal components
- Robust approaches
- The implementation
- Supporting methods
- Covariance estimation by PCAs

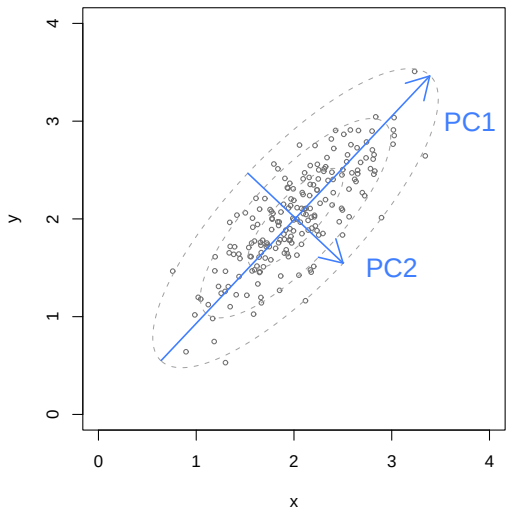
Principal Component Analysis (PCA)



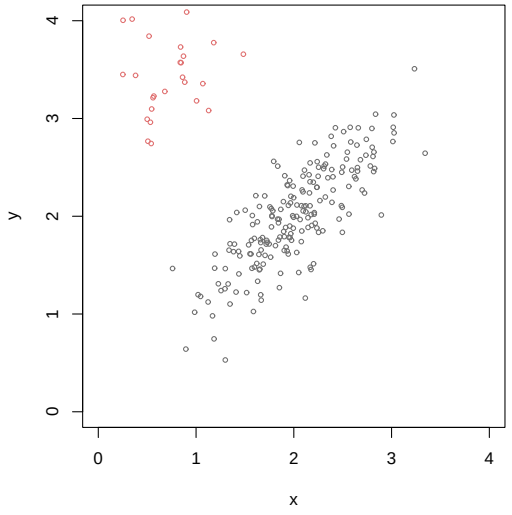
Principal Component Analysis (PCA)



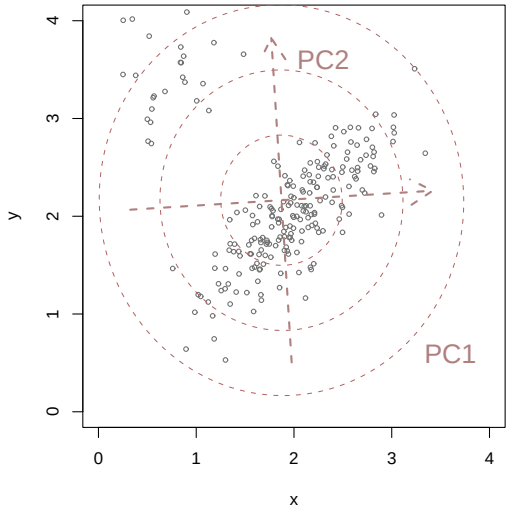
Principal Component Analysis (PCA)



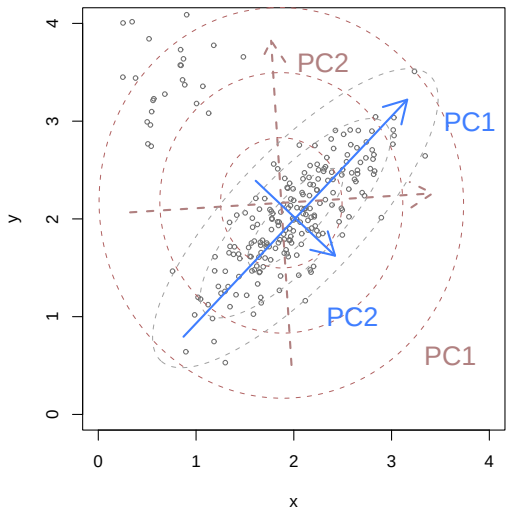
Outliers



Outliers



Outliers



The Classical Approach

- PCA by decomposition of the covariance matrix

$$\hat{\Sigma} = \Gamma \Lambda \Gamma^t$$

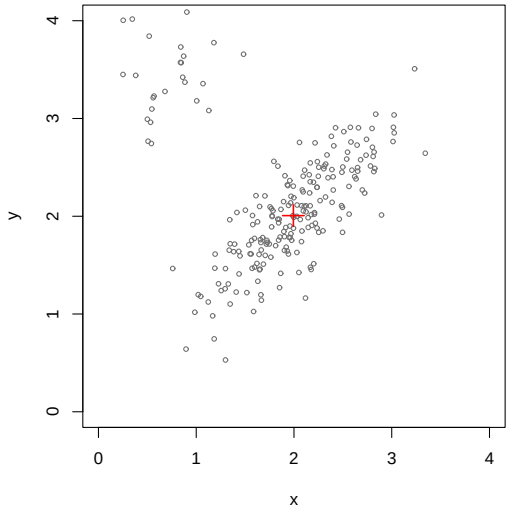
$$Y = (X - 1\bar{x}^t)\,\Gamma$$

- Robustness due to robust covariance estimates.
  - package `rrcov`: `covMCD`, `covMest`
  - package `robustbase`: `covGK`, `covOGK`

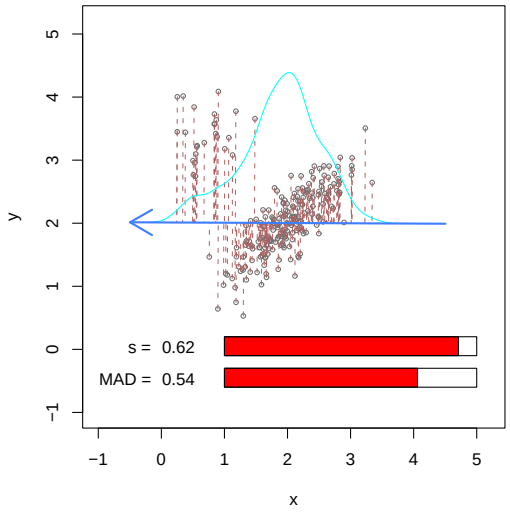
PCA by Projection Pursuit

- No covariance estimation necessary
- Especially for high dimensional data
- Procedure
  - Define a data center (`mean`, `median`, `l1median`, ...)
  - Search for promising directions by maximizing a spread estimation (`sd`, `mad`, `qn`) of the data projected onto these directions
  - Reduce the amount of candidate directions

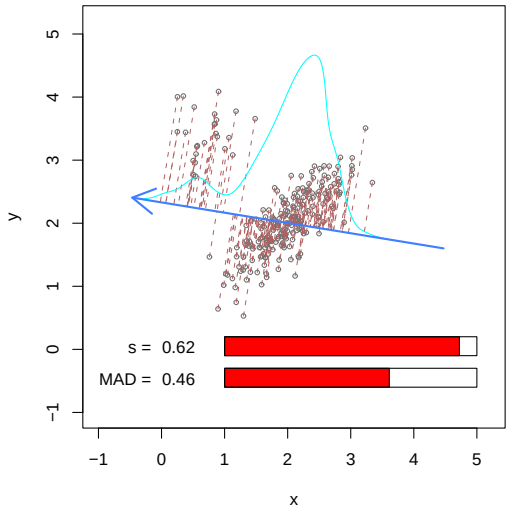
Defining the Data Center



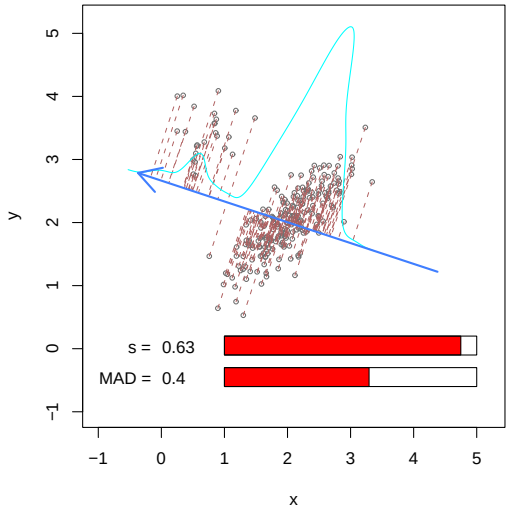
Maximizing Spread



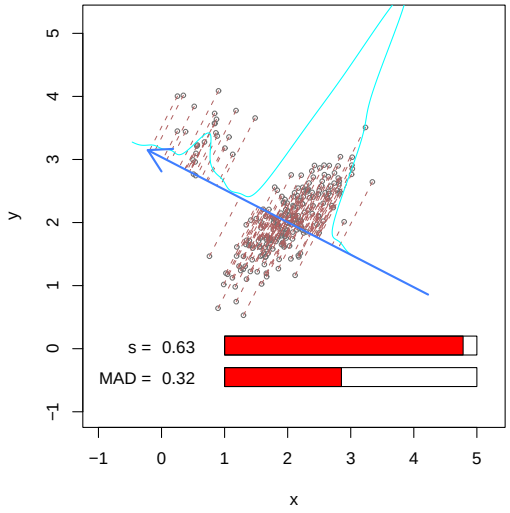
Maximizing Spread



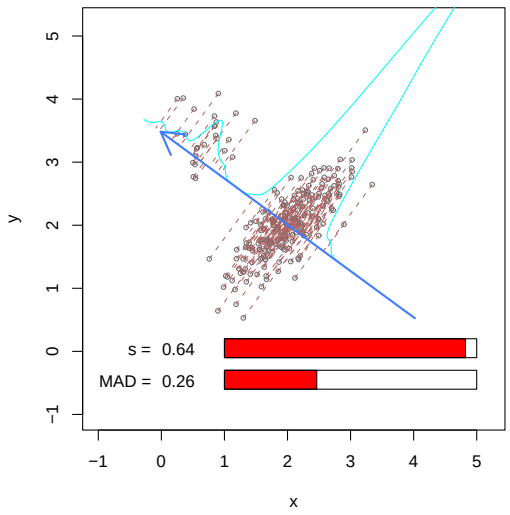
Maximizing Spread



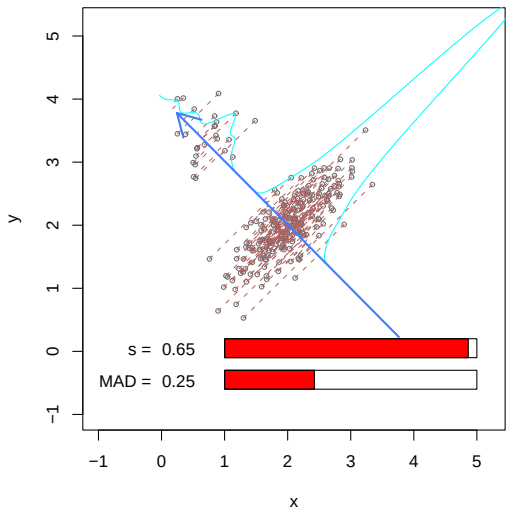
Maximizing Spread



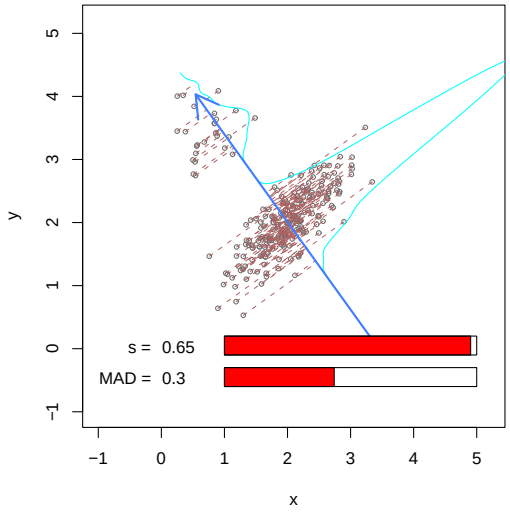
Maximizing Spread



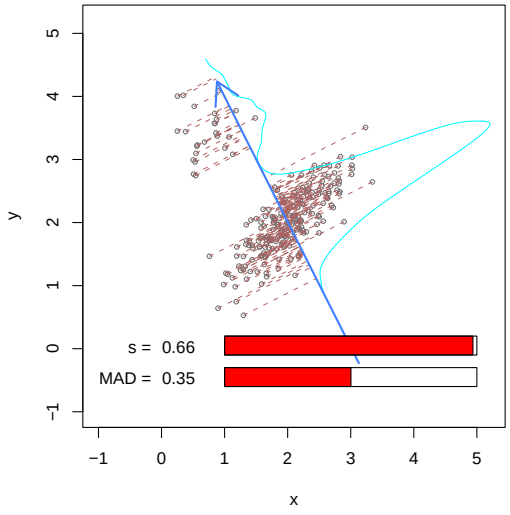
Maximizing Spread



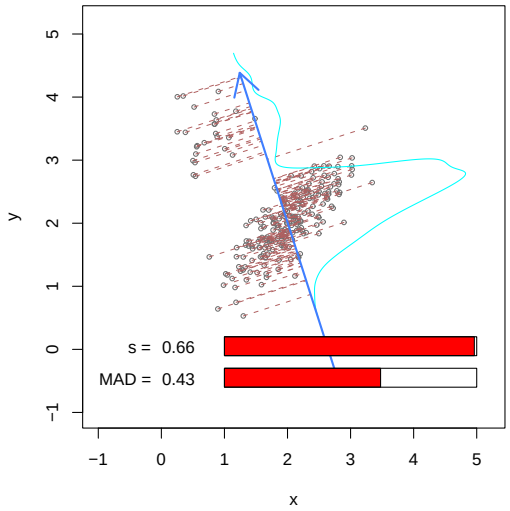
Maximizing Spread



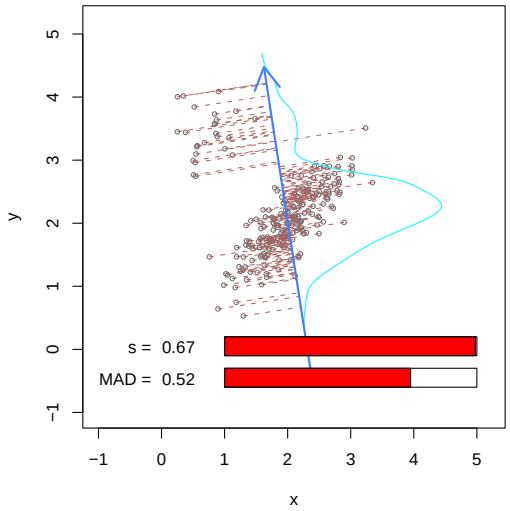
Maximizing Spread



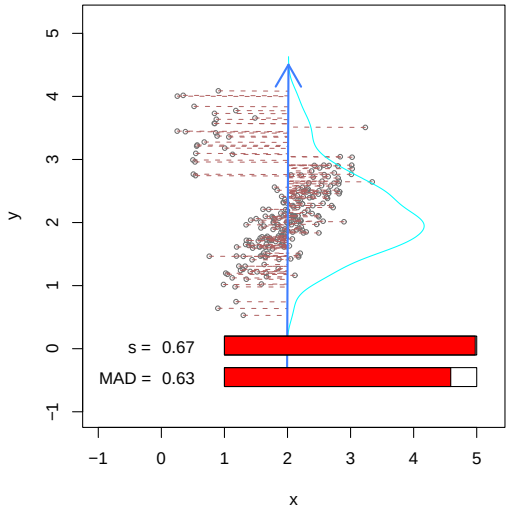
Maximizing Spread



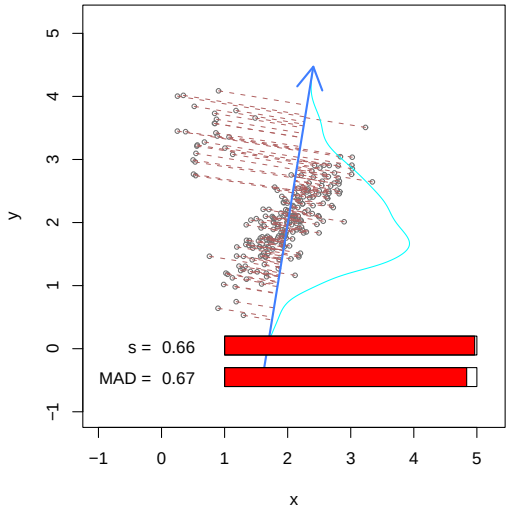
Maximizing Spread



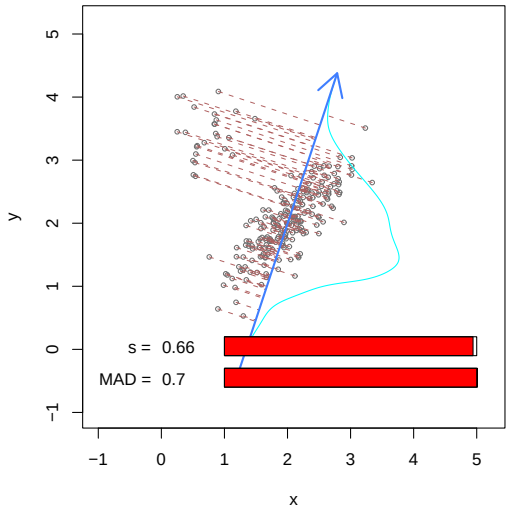
Maximizing Spread



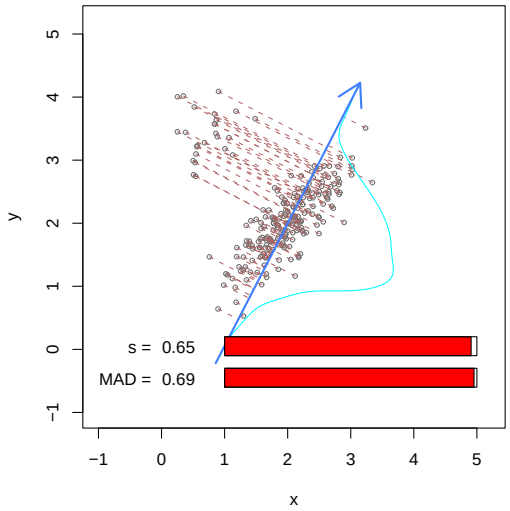
Maximizing Spread



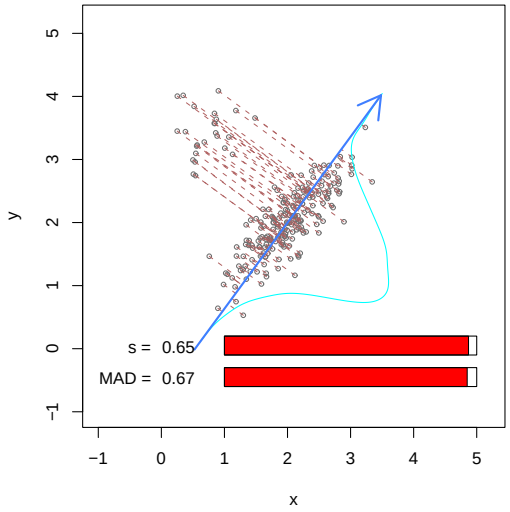
Maximizing Spread



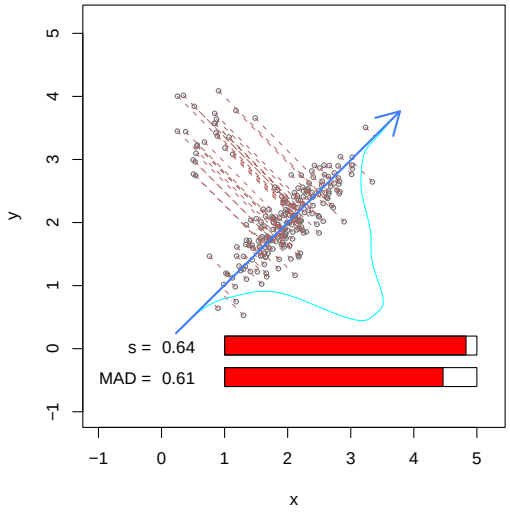
Maximizing Spread



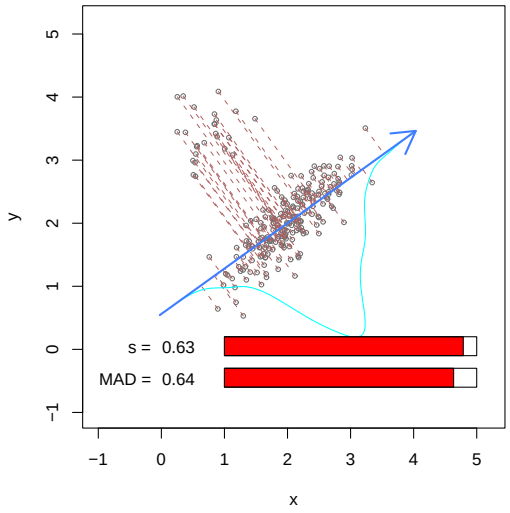
Maximizing Spread



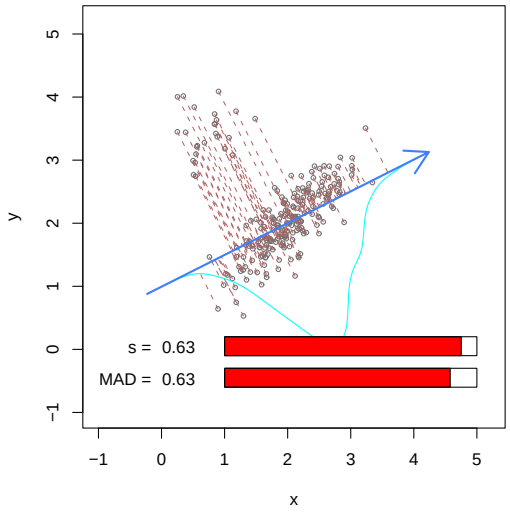
Maximizing Spread



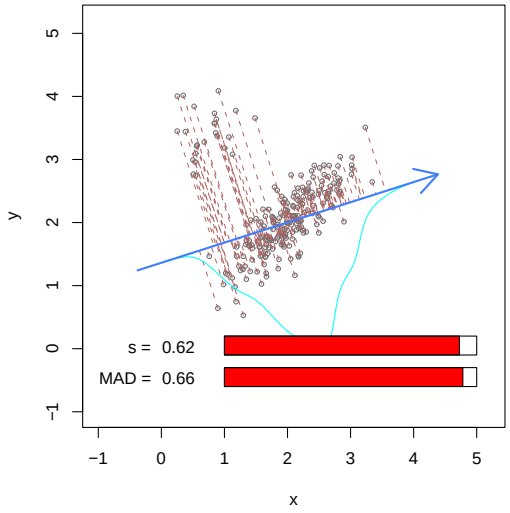
Maximizing Spread



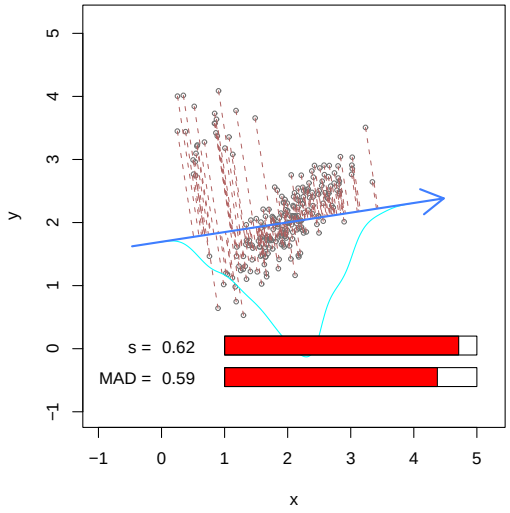
Maximizing Spread



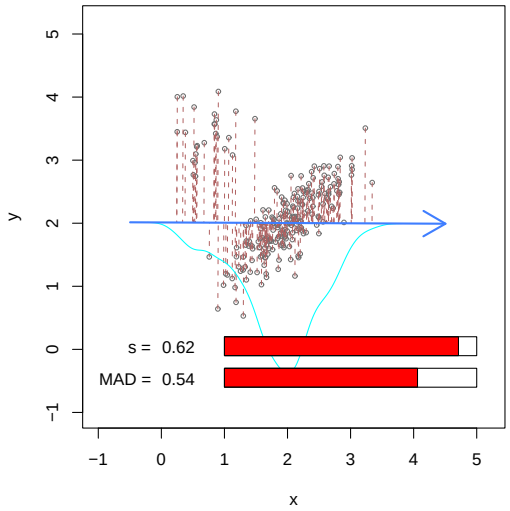
Maximizing Spread



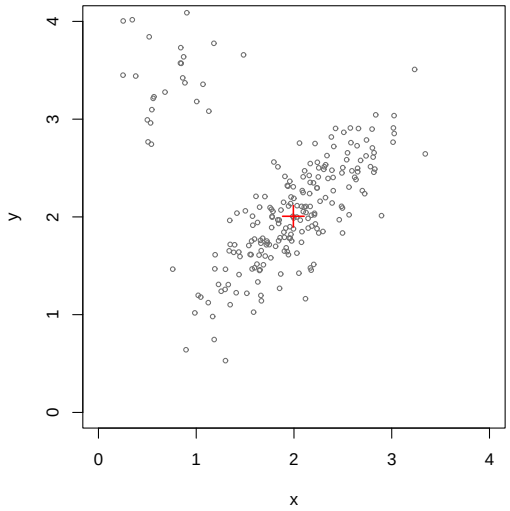
Maximizing Spread



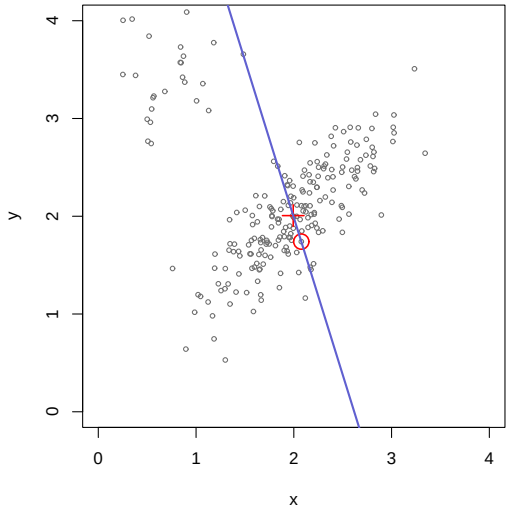
Maximizing Spread



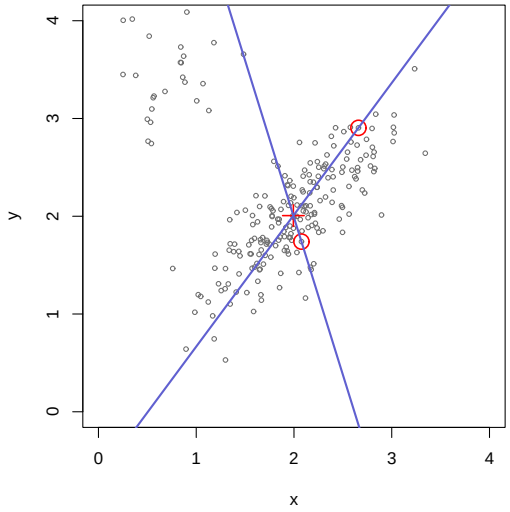
PCAproj



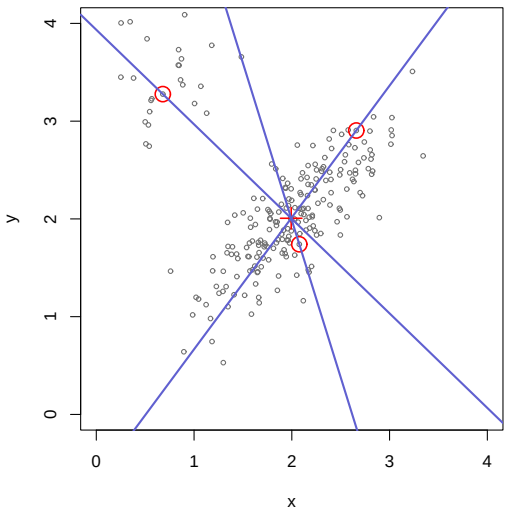
PCAproj



PCAproj



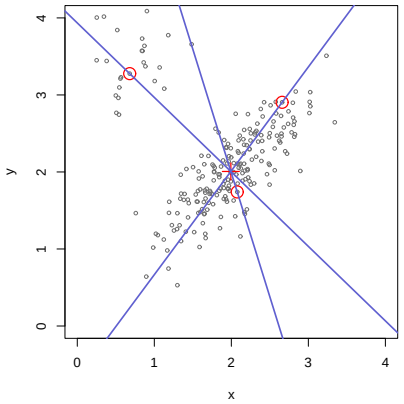
PCAproj



PCAproj

Candidate Directions:

- each data point
- additionally random directions through center
- additional directions by linear combinations of data points
- update algorithm (based on eigenvalues)

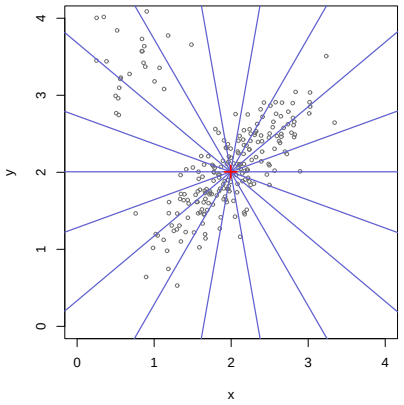


PCAgrid

Grid Algorithm:

Optimization is done on a regular grid in the plane.

- select two variables
- optimization on the grid
- select other variables
- ...



Implementation

- Implementation in C
- Wrapping functions
  - **PCAproj**`(x, k = 2, method = c("sd", "mad", "qn"), CalcMethod = c("eachobs", "lincomb", "sphere"), nmax = 1000, update = TRUE, scores = TRUE, maxit = 5, maxhalf = 5, control, ...)`
  - **PCAgrid**`(x, k = 2, method = c("sd", "mad", "qn"), maxiter = 10, splitcircle = 10, scores = TRUE, anglehalving = TRUE, fact2dim = 10, control, ...)`

Common Parameters

- `x`: Data matrix (data frame)
- `k`: Number of principal components
- `method`: Spread estimator for projection pursuit
- `scores`: Return scores-matrix?
- `control`: Control-structure
- `...` Passed to `ScaleAdv`

PCAgrid - Individual Parameters

- `splitcircle`: Number of directions
- `anglehalving` : Perform anglehalving
- `fact2dim` : Behavior in 2 dimensional case.
- `maxiter`: Maximum number of iterations.

PCAproj - Individual Parameters

- `CalcMethod`: `"eachobs"`,`"lincomb"` or `"sphere"`
- `nmax`: Max directions to search in each step (for `"lincomb"`or `"sphere"`)
- `update`: Perform update steps?
  - `maxhalf`: Maximum number of steps for angle halving
  - `maxit`: Maximum number of iterations

Return Structure

- (S3) class `pcaPP` derived from `princomp`:
  - `sdev`: Spread of principal components
  - `loadings`: Matrix containing the loadings
  - `center`: Center applied to the data matrix
  - `scale`: Scale applied to the data matrix
  - `n.obs`: Number of observations
  - `scores`: Matrix containing the scores
  - `call`: Function call

Additional Functions

- `l1median(X, MaxStep = 200, ItTol = 10^-8)`
  Robust center estimator
- `qn(x)`
  Robust scale estimator
- `ScaleAdv(x, center = mean, scale = sd)`
  Advanced scaling method (takes functions or vectors as input values)

Robust Covariance Estimation

- Robust covariance estimation based on PCs

$$\hat{\Sigma} = \hat{\Gamma}\hat{\Lambda}\hat{\Gamma}^t$$

- `covPCAproj(x, control)`
- `covPCAgrid(x, control)`
- covPC(x, k, method) (under construction ...)

Example

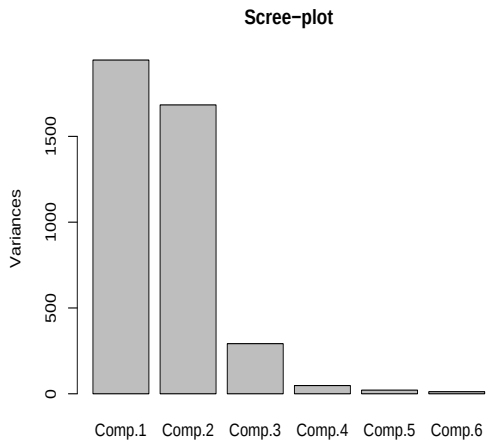
```
> library(pcaPP)
> data(swiss)
> result = PCAproj(swiss, k = 6, method = "mad")

> summary(result)

Importance of components:
                          Comp.1     Comp.2      Comp.3     Comp.4      Comp.5
Standard deviation     44.1005199 41.0302723 17.09114152 6.92022550 4.619893062
Proportion of Variance  0.4859749  0.4206639  0.07299087 0.01196649 0.005333229
Cumulative Proportion   0.4859749  0.9066387  0.97962962 0.99159611 0.996929342
                           Comp.6
Standard deviation     3.505520822
Proportion of Variance 0.003070658
Cumulative Proportion  1.000000000
```

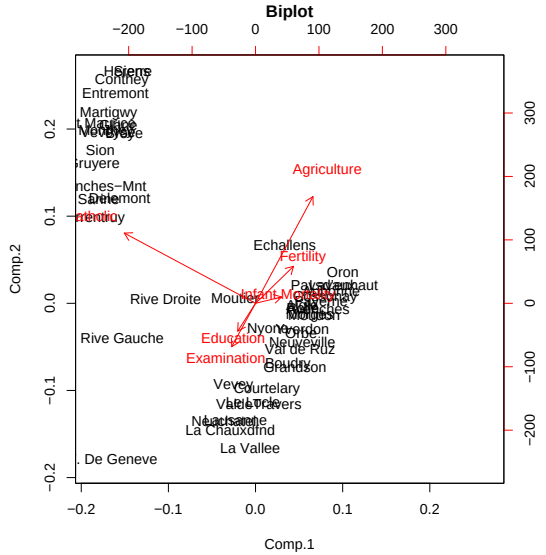
Example

```
screeplot(result)
```

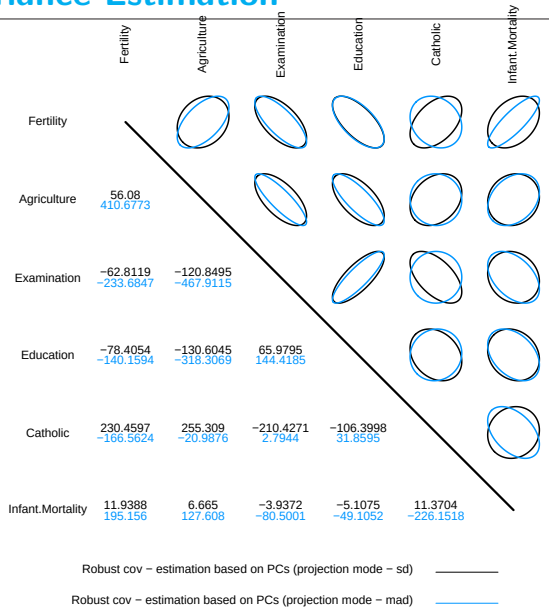


Example

```
biplot(result)
```



Covariance Estimation



Covariance Estimation

```
> library (covrob)
> covswiss.mad <- covrob (swiss, method="covPCAproj", control = list
(k=6,method="mad"))
> covswiss.sd <- covrob (swiss, method="covPCAproj", control = list
(k=6,method="sd"))
> plot (covswiss.mad, covswiss.sd)
```