UseR! 2008 Tutorial — Robust Statistics with R
"Exercises and Demos"

Martin Maechler

ETH Zurich
Switzerland
maechler@R-project.org

UseR! 2008, Dortmund
Aug. 11, 2008

Outline

Preliminary

- Robust Statistics using R: only recently blossoming.
- On CRAN (http://CRAN.R-project.org/), with its more than 1400 R packages, *CRAN Task Views* provide focus on 19 different subject areas, one of which "Robust Methods".
  $\longrightarrow$
  http://CRAN.R-project.org/web/views/Robust.html.
- This tutorial: Just small parts of robustbase and robust

— Part 1 —

Basics: Sensitivity-Curve

The sensitivity curve is the "empirical influence function", i.e., $\mathrm{SC}_n(...) \stackrel{n\to\infty}{\to} IF(...)$

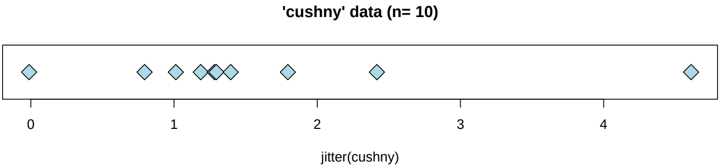
$$\begin{aligned}\mathrm{SC}(x; x_1, \ldots, x_{n-1}; T_n) \quad &:= \quad \frac{T_n(x_1, \ldots, x_{n-1}, x) - T_{n-1}(x_1, \ldots, x_{n-1})}{1/n} \\ &= \quad n \cdot (T_n(x_1, \ldots, x_{n-1}, x) - T_{n-1}(x_1, \ldots, x_{n-1}))\,.\end{aligned}$$

Task: Compute and plot the $\mathrm{SC}()$ for a few location estimators (i.e., the arithmetic mean and robust versions of it).

Cushny Data

As example, we use a historical small example data set from Student (1908):

```
> data(cushny, package="robustbase")
> plot(jitter(cushny), rep(0,10), pch=23, cex=2, bg="light blue"
+       main = "'cushny' data (n= 10)", ylab="", yaxt = "n")
```



and we will "vary" the 10-th observation ($x_{10} = 4.6$), i.e., draw $\mathrm{SC}_9(x, x_1, \ldots, x_9)$.

SC ()

In the accompanying R-script,we define a short function `SC()` to compute the sensitivity curve; it is basically

```
SC <- function(x, x.dat, EST, ...)
{
  # Arguments: x     : varying data point - as vector!
  #            x.dat: the n-1 given x_1 .. x_n
  #            EST  : function(x, ...) {x := "sample"}
  #            ...  : optional further arg.s to EST()

  stopifnot(is.numeric(x), is.numeric(x.dat), is.functio
  n_1 <- length(x.dat)
  n <- n_1 + 1
  # when 'x' is a vector, compute T_n(x[i],...) for each
  Tn <- sapply(x, function(z) EST(c(x.dat, z), ...))
  n*(Tn - EST(x.dat, ...))
}
```

SC (., cushny[-10])

and applied to the Cushny data,

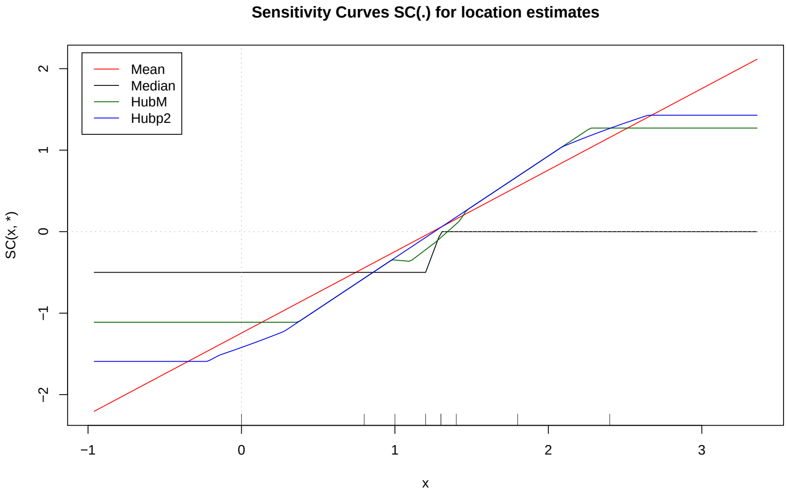
```
> source("R/basics-defs.R")
> x <- -1:6
> SC(x, cushny[-10], mean)

[1] -2.2444 -1.2444 -0.2444  0.7556  1.7556  2.7556  3.7556  4.7556

> SC(x, cushny[-10], median)

[1] -0.5 -0.5 -0.5  0.0  0.0  0.0  0.0  0.0

> SC(x, cushny[-10], mean, trim = 0.20)

[1] -0.9048 -0.9048 -0.5714  0.7619  0.7619  0.7619  0.7619  0.7619
```

we see that the SC() function is linear for the mean and bounded for the median and a trimmed mean.

plot SC (., cushny[-10])

In order to plot these, we use the utility `p.SCs()`,

```
> p.SCs(cushny[-10])
```



plot SC (., cushny[-10]) — 2

`p.SCs(., *)` uses by default the following to versions of Huber location M-estimators, both of which behave remarkably. [1]

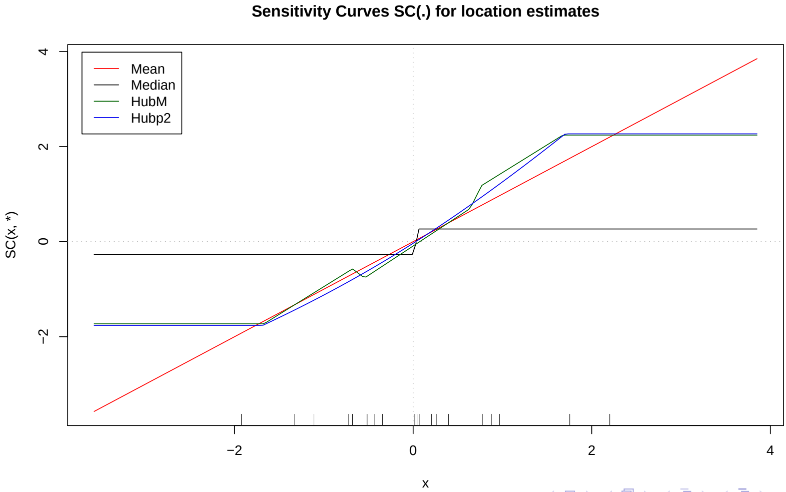
```
## hubers() is in MASS -- computing ''proposal 2''
HubS <- function(x, ...) hubers(x, ...)$mu

## huberM() is in robustbase -- and returns a short l
HubM <- function(x, ...) huberM(x, ...)$mu
```

[1] We need these definitions here because the corresponding functions in `MASS` and `robustbase` return a list structure.

plot SC (rnorm(20)))

Further examples of SC()s for simulated data

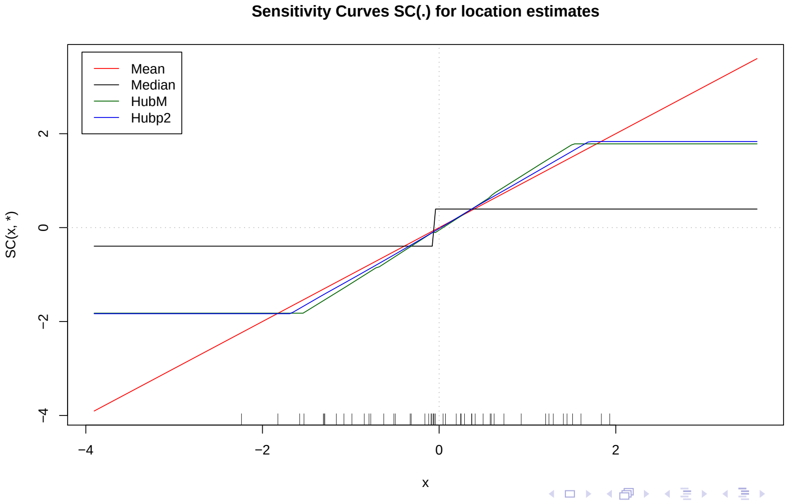
```
> set.seed(12)
> p.SCs(scale(rnorm(20)))
```



plot SC (rnorm(50)))

Further examples of SC()s for simulated data

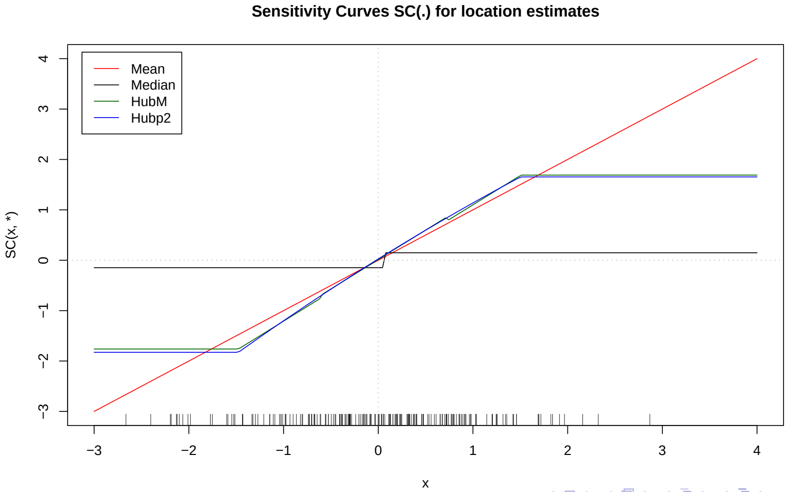
```
> set.seed(21)
> p.SCs(scale(rnorm(50)))
```



plot SC (rnorm(200)))

Further examples of SC()s for simulated data

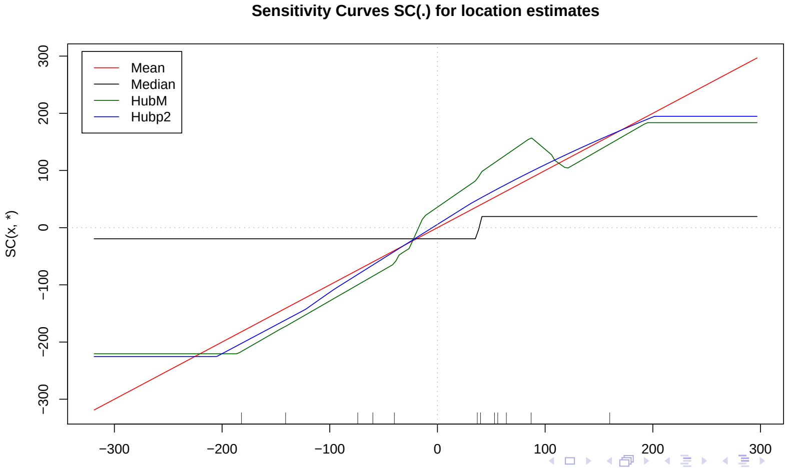
```
> set.seed(1959)
> p.SCs(scale(rnorm(200)), xlim=c(-3,4))
```



plot SC (“bizarre”))

A “bizarre” example (found in about a dozen rnorm(12) trials):

```
> x12 <- c(-182, -141, -74, -60, -40, 37,
+            40, 53, 56, 64, 87, 160)
> p.SCs(x12)
```



Questions on Section 1 — “Basics” ?

— Part 2 —

Linear Models — Robustly

I.e., doing inference about

$$\boldsymbol{y} = \boldsymbol{X} \cdot \boldsymbol{\beta} + \boldsymbol{\epsilon}, \qquad \boldsymbol{X} \in \mathbb{R}^{n \times p}$$

Covering only parts of

1. finding $\hat{\boldsymbol{\beta}}$ *robustly*
2. Testing $H_0 : \beta_j = 0$ (or general $H_0 : \boldsymbol{A} \cdot \boldsymbol{\beta} = \mathbf{0}$) *robustly*
3. Variable selection (model building) *robustly*
4. *robust* (residual) diagnostics

Remember:

$$\mathrm{IF}() = \tilde{\mathrm{IF}}(\mathrm{resid}) \times \tilde{\mathrm{IF}}(\boldsymbol{x})$$

and M-estimators (Huber, including L_1 $(:= \arg\min_{\boldsymbol{\beta}} \sum_i |y_i - \boldsymbol{x}_i^{\mathsf{T}} \boldsymbol{\beta}|)$) only bound the influence of the residuals.

Robust LM with R

R: Standard `lm()` is for classical least squares. "Robust `lm`" in three flavors:

- `rlm()` from `MASS`[1]
- `lmrob()` from `robustbase`
- `lmRob()` from `robust` (Insightful)

[1] MASS = "Recommended" R package: always installed

Robust LM with R– Overview

"Exercise tasks":

1. Get a feeling for robust "simple" regression, $p = 2$, $\boldsymbol{x}_i = (1, x_i) \in \mathbb{R}^2$.
   $\longrightarrow$ interactive demo.
2. Main importance of robust regression is *not* for $p = 2$, but rather $p \approx 10, 20, 50$ or even higher!

Robust **Simple** LM with R

"Simple" regression: $p = 2$:

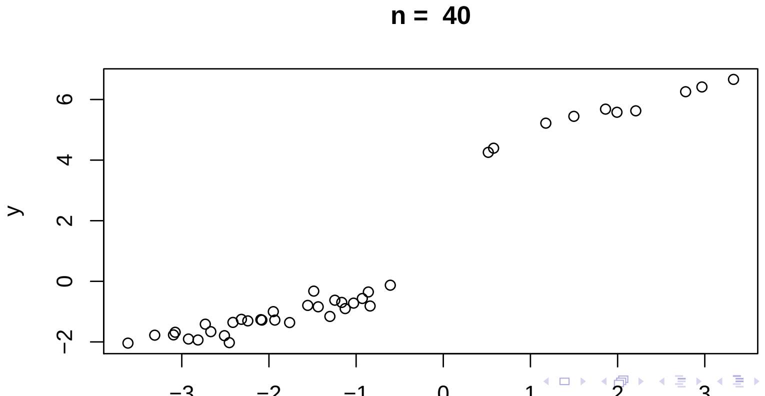
$$y_i = \beta_1 + \beta_2 x_i + \epsilon_i.$$

1. Artificial example (residual plots in lecture notes $\approx$ p.37)
2. interactive "play" and demo

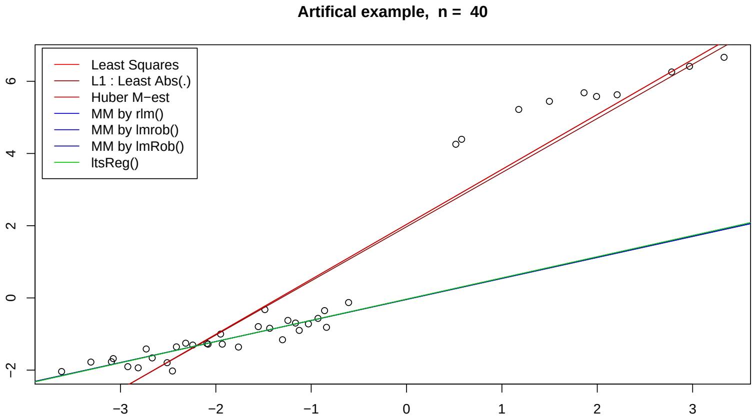
Simple robust LM – 1 –

1) The artificial example from the lecture notes:

```
> set.seed(050808) ## 40 observations in two groups of 30 + 10 :
> x1 <- rnorm(30,-2,1); y1 <-    .6*x1 + rnorm(30)/5
> x2 <- rnorm(10,2,1) ; y2 <- 4+.8*x2 + rnorm(10)/5
> x <- c(x1,x2)        ; y <- c(y1,y2)
> plot(y ~ x, main = paste("n = ", length(x)))
```



```
> ## source("ftp://stat.ethz.ch/......../regr-defs.R")
> plot(y ~ x, ann=FALSE); title(paste("Artifical example,  n = "
> re <- Reg.Estimators(y ~ x)
> p.line.legend(re)
```



There are 7 lines . . . . . . which are which ?

Simple robust LM – 2 –

Interactively drag a point (x_i, y_i) and watch the regression lines changing, using our R script,
ftp://stat.ethz.ch/U/maechler/R/robust-tutorial/regDemo.R:

```
> source("ftp://stat.ethz.ch/..../regDemo.R")
> regDemo(8) ## n =  8  or
> regDemo(20)## n = 20
```

robust LM for stackloss

```
> mSLr <- lmrob(formula = stack.loss ~ ., data = stackloss)
> summary(mSLr)

Call:
lmrob(formula = stack.loss ~ ., data = stackloss)

Weighted Residuals:
     Min       1Q   Median       3Q      Max
-10.5097  -1.4382  -0.0913   1.0250   7.2311

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -41.5246     5.2978   -7.84  4.8e-07 ***
Air.Flow      0.9388     0.1174    7.99  3.7e-07 ***
Water.Temp    0.5796     0.2630    2.20    0.042 *
Acid.Conc.   -0.1129     0.0699   -1.62    0.125
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Robust residual standard error: 1.91
Convergence in 17 IRWLS iterations
```

lmrob(.. stackloss) – 2nd part

```
summary( lmrob(formula = stack.loss   ., ....))
[................................]
```

[continued]

```
Robustness weights:
 observation 21 is an outlier with |weight| = 0 ( < 0.0048);
 2 weights are ~= 1. The remaining 18 ones are summarized as
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.122   0.876   0.943   0.872   0.980   0.998
Algorithmic parameters:
tuning.chi          bb tuning.psi refine.tol    rel.tol
  1.55e+00    5.00e-01    4.69e+00    1.00e-07    1.00e-07
[................................]
```

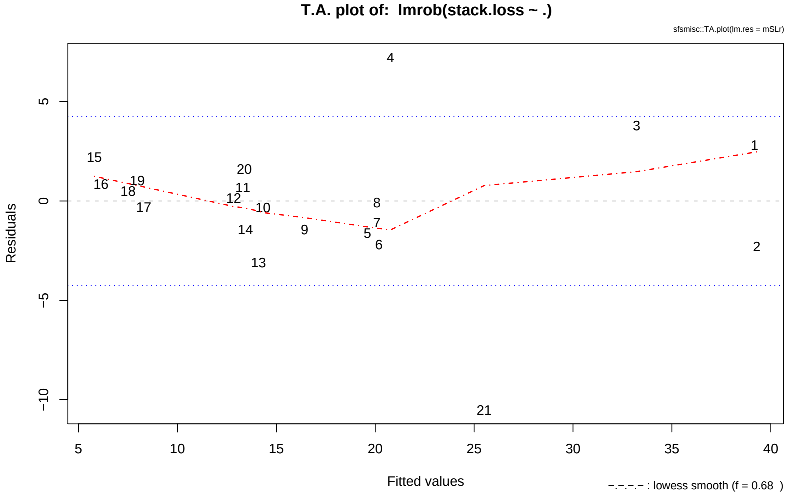
Let us look at the robustness weights more closely:

```
> round(weights(mSLr), 3)

 [1] 0.812 0.873 0.675 0.122 0.936 0.884 0.971 1.000 0.949 0.997 0.988
[13] 0.775 0.949 0.883 0.982 0.998 0.994 0.974 0.936 0.000

> which(weights(mSLr) < 0.2)

[1]  4 21
```

$\longrightarrow$ One clear and one borderline outlier

Plot robust LM for stackloss

```
> sfsmisc::TA.plot(mSLr) ## slightly nicer than plot(mSLr)
```



robust vs. L.S. regression

The `robust` package (from Insightful's S-plus version) fosters idea to *compare* classical and robust fits

```
> fm.SL <- fit.models(list(Robust = "lmRob", LS = "lm"),
+                      stack.loss ~ ., data = stackloss)
> (sfm.SL <- summary(fm.SL))

Calls:
Robust: lmRob(formula = stack.loss ~ ., data = stackloss)
    LS: lm(formula = stack.loss ~ ., data = stackloss)

Residual Statistics:
           Min      1Q  Median    3Q   Max
Robust: -8.630 -0.6713  0.3594 1.151 8.174
    LS: -7.238 -1.7117 -0.4551 2.361 5.698

Coefficients:
                           Value Std. Error t value  Pr(>|t|)
Robust (Intercept) -37.65246    5.00256 -7.5266 8.289e-07
    LS (Intercept) -39.91967   11.89600 -3.3557 3.750e-03
Robust    Air.Flow   0.79769    0.07129 11.1886 2.914e-09
    LS    Air.Flow   0.71564    0.13486  5.3066 5.799e-05
```

robust vs. L.S. regression – 2nd part

`summary(fit.models(...))` continued

```
[.................................]
Coefficients:
                         Value Std. Error t value  Pr(>|t|)
Robust (Intercept) -37.65246    5.00256 -7.5266 8.289e-07
    LS (Intercept) -39.91967   11.89600 -3.3557 3.750e-03
Robust    Air.Flow   0.79769    0.07129 11.1886 2.914e-09
    LS    Air.Flow   0.71564    0.13486  5.3066 5.799e-05
Robust  Water.Temp   0.57734    0.17546  3.2905 4.318e-03
    LS  Water.Temp   1.29529    0.36802  3.5196 2.630e-03
Robust  Acid.Conc.  -0.06706    0.06512 -1.0297 3.176e-01
    LS  Acid.Conc.  -0.15212    0.15629 -0.9733 3.440e-01

Residual Scale Estimates:
Robust: 1.837 on 17 degrees of freedom
    LS: 3.243 on 17 degrees of freedom

Multiple R-Squared:
Robust: 0.6205
    LS: 0.9136
[.................................]
```

Questions on Section 2 — “Linear Models” ?

— Part 3 —

Generalized Linear Models

We will only consider

- Logistic/Binomial regression
- Poisson regression (for count data)

Task: One GLM for each situation, including tests . . . . . .

GLMs - Logistic Regression

Logistic: Binary response $Y = 0$ or 1: occurence of "vaso constriction" reflex (Finney, 1947)

```
> data(vaso)
> ## classical :
> v.cla  <- glm(Y ~ log(Volume) + log(Rate), family=binomial, da
> ## robust :
> v.r <- glmrob(Y ~ log(Volume) + log(Rate), family=binomial, da
> ## quite different:
> cbind(class = coef(v.cla),  robust = coef(v.r))
```

```
             class robust
(Intercept) -2.875 -21.37
log(Volume)  5.179  34.82
log(Rate)    4.562  27.87
```

GLMs - Logistic – 2 –

We can do *inference*: classical and robust

```
> summary(v.cla)# indication of clear effect

Call:
glm(formula = Y ~ log(Volume) + log(Rate), family = binomial,
    data = vaso)

Deviance Residuals:
   Min       1Q   Median       3Q      Max
-1.453  -0.611   0.100   0.618   2.278

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)    -2.88       1.32   -2.18   0.0295 *
log(Volume)     5.18       1.86    2.78   0.0055 **
log(Rate)       4.56       1.84    2.48   0.0131 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 54.040  on 38  degrees of freedom
```

GLMs - Logistic – 3 –

Robust *inference*: summary quite different:

```
> summary(v.r)  # explanatory variables don't predict at all?

Call:  glmrob(formula = Y ~ log(Volume) + log(Rate), family = binomial,


Coefficients:
            Estimate Std. Error z-value Pr(>|z|)
(Intercept)    -21.4       14.1   -1.51     0.13
log(Volume)     34.8       23.6    1.47     0.14
log(Rate)       27.9       18.0    1.55     0.12
Robustness weights w.r * w.x:
 2 observations c(4,18) are outliers with |weight| <= 0.00023 ( < 0.002
 36 weights are ~= 1. The remaining one are
   24
0.695

Number of observations: 39
Fitted by method 'Mqle'  (in 15 iterations)

(Dispersion parameter for binomial family taken to be 1)
```

GLMs - Logistic – 4 –

Robust *inference*: Compare with 0 model:

```
> anova(update(v.r, . ~ 1), v.r)

Robust Wald Test Table

Model 1: Y ~ 1
Model 2: Y ~ log(Volume) + log(Rate)
Models fitted by method 'Mqle'

  pseudoDf Test.Stat Df Pr(>chisq)
1       38
2       36      2.69  2       0.26
```

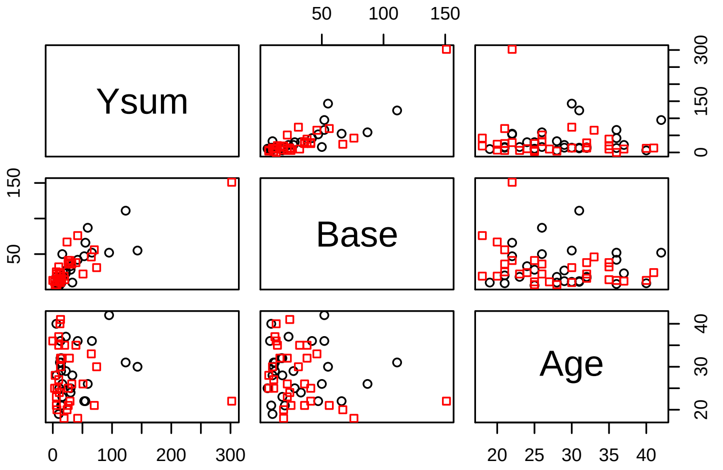
Poisson GLM - Epilepsy Data

```
> data(epilepsy)
> str(epilepsy[,6:9]) ## will only use  (Ysum ~ (Base, Age, Trt)

'data.frame': 59 obs. of  4 variables:
 $ Base: int  11 11 6 8 66 27 12 52 23 10 ...
 $ Age : int  31 30 25 36 22 29 31 42 37 28 ...
 $ Trt : Factor w/ 2 levels "placebo","progabide": 1 1 1 1 1 1 1 1 1 1
 $ Ysum: int  14 14 11 13 55 22 12 95 22 33 ...
```

`Ysum` is the number epileptic attacks of 4 different kinds. They are modeled to depend on a `Base` number, patient `Age` and a treatment `Trt` (drug or placebo).

Epilepsy Data PLot

```
> with(epilepsy, pairs(cbind(Ysum, Base, Age), col= Trt, pch= 20
```



GLMs - Poisson Regression

```
> summary(m1 <- glmrob(Ysum ~ Age + Base*Trt, family=poisson, da

Call:  glmrob(formula = Ysum ~ Age + Base * Trt, family = poisson, data


Coefficients:
                   Estimate Std. Error z-value Pr(>|z|)
(Intercept)         2.04495    0.15217   13.44  < 2e-16 ***
Age                 0.01600    0.00468    3.42  0.00064 ***
Base                0.02124    0.00103   20.64  < 2e-16 ***
Trtprogabide       -0.33278    0.08630   -3.86  0.00012 ***
Base:Trtprogabide  0.00299    0.00123    2.44  0.01462 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Robustness weights w.r * w.x:
 27 weights are ~= 1. The remaining 32 ones are summarized as
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0829  0.3440  0.5620  0.5380  0.7610  0.9640

Number of observations: 59
Fitted by method 'Mqle'  (in 13 iterations)
```

GLMs - Poisson Regression - *Tests*

Is the interaction `Base:Trt` necessary ?

$\longrightarrow$ Test $H_0 : \beta_k = 0$?

```
> ## model withOUT interaction:
> m2 <- glmrob(Ysum ~ Age + Base + Trt, family=poisson, data=epi
> anova(m2, m1, test = "Wald")  # P = .015
```

```
Robust Wald Test Table

Model 1: Ysum ~ Age + Base + Trt
Model 2: Ysum ~ Age + Base * Trt
Models fitted by method 'Mqle'

  pseudoDf Test.Stat Df Pr(>chisq)
1       55
2       54      5.96  1      0.015 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

GLMs - Poisson - *Tests* - 2 -

Is the interaction Base:Trt necessary ?
Quasi-Deviance ("QD") (Cantoni & Ronchetti) test instead of "Wald" suggest a different story again:

```
> anova(m2, m1, test = "QD")

Robust Quasi-Deviance Table

Model 1: Ysum ~ Age + Base + Trt
Model 2: Ysum ~ Age + Base * Trt

. . . . . . . . . . . . . . . . . . .
> ## Compare:
> sapply(c("Wald", "QD", "QDapprox"),
+        function(T) anova(m2, m1, test = T)$P[2])

    Wald        QD  QDapprox
 0.01462   0.06598   0.01462
```

Questions on Section 3 — "Generalized LM's" ?

— Part 4 —

Multivariate Location & Scatter

Estimation "location" and "scatter" in p-dimensional, e.g., estimation of μ and Σ.
Tasks: similar to regression,

1. $p = 2$ is "easy", and nice for visualization
2. For $p \geq 3$, and *"p moderately large"*, robustness is harder to achieve and more important

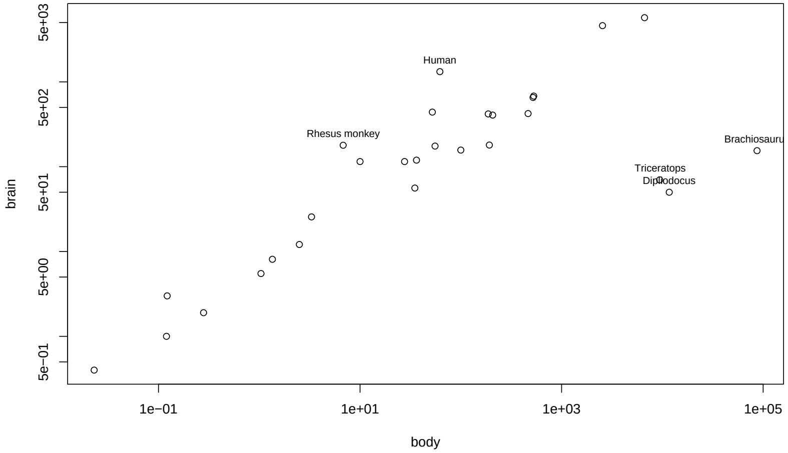
$p = 2$-dimensional Location & Scatter

Using a famous kind of data, body and brain weights of different animal species:

```
> data(Animals, package ="MASS")
> brain <- Animals[c(1:24, 26:25, 27:28),]  #  28 x 2
> head(brain)

                      body brain
Mountain beaver       1.35   8.1
Cow                 465.00 423.0
Grey wolf            36.33 119.5
Goat                 27.66 115.0
Guinea pig            1.04   5.5
Dipliodocus       11700.00  50.0

> cR <- covMcd( log(brain) )
> ## ‘‘the outliers’’
> which(outL <- cR$mcd.wt == 0)

  Dipliodocus           Human   Triceratops Rhesus monkey Brachiosaurus
            6              14            16            17            25
```

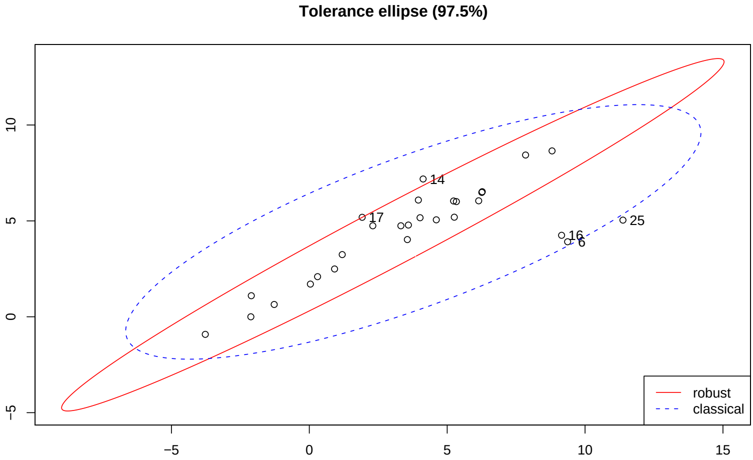
$p = 2$-dimensional Location & Scatter

```
> plot(brain, log="xy")
> text(brain[outL,], rownames(brain)[outL], cex = .75, pos = 3)
```



$p = 2$-dimensional Location & Scatter

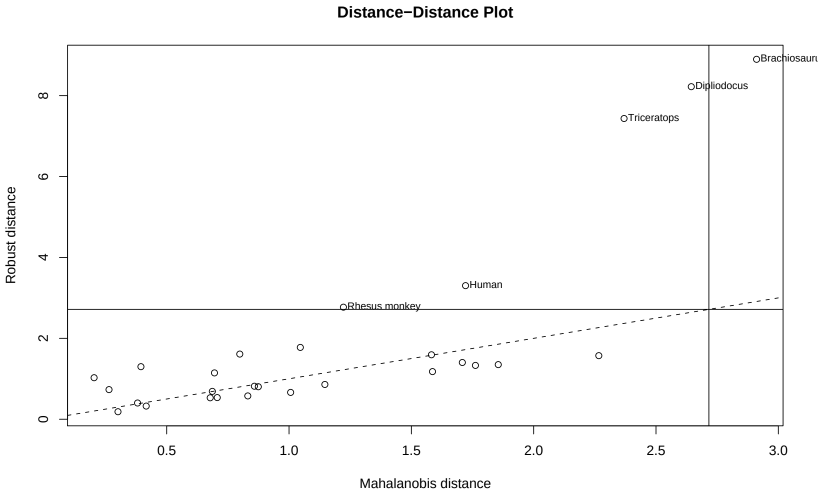
```
> plot( covMcd( log(brain) ), which = "tolEllipsePlot",
+                                 classic = TRUE)
```



Robust vs. classical Mahalanobis Distances

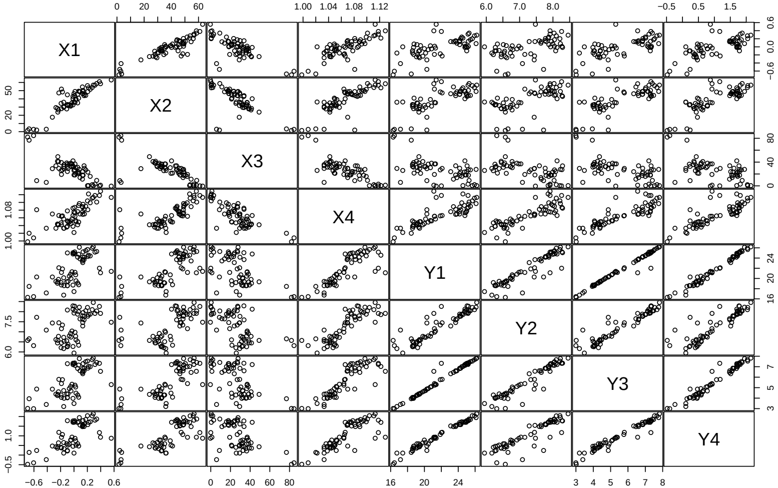
This is *The* plot that also applies to high dim. p :

```
> plot( covMcd( log(brain)), which = "dd")
```



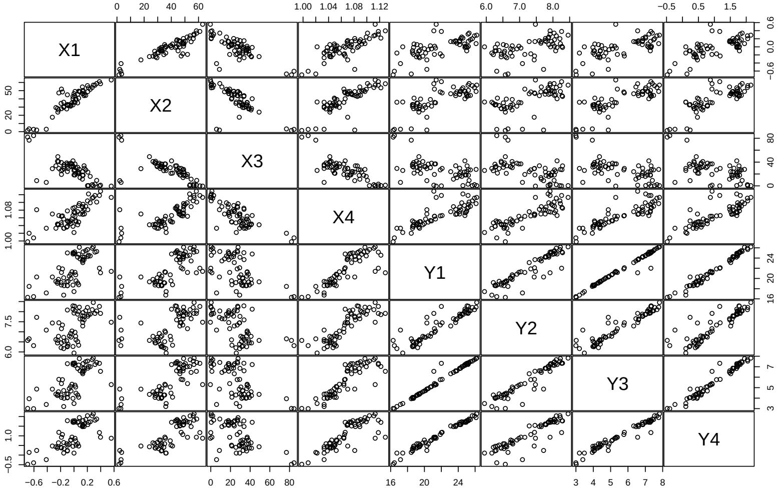
higher-dimensional

```
> data(pulpfiber)
> pairs(pulpfiber, gap=.1) ## 2 blocks of 4 ..
```



higher-dimensional

```
> data(pulpfiber)
> pairs(pulpfiber, gap=.1) ## 2 blocks of 4 ..
```

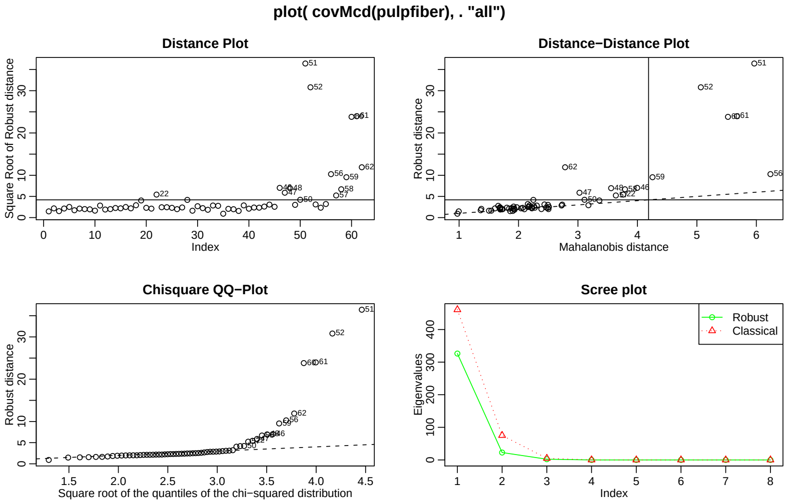


higher-dimensional

```
> c1 <- cov(pulpfiber) ; cR <- covMcd(pulpfiber) ## how differen
> symnum(cov2cor(c1))

   X1 X2 X3 X4 Y1 Y2 Y3 Y4
X1 1
X2 *  1
X3 ,  ,  1
X4 ,  ,  ,  1
Y1 ,  ,  .  +  1
Y2 .  ,  .  +  *  1
Y3 ,  ,  .  +  B  *  1
Y4 ,  ,  .  +  B  +  B  1
attr(,"legend")
[1] 0 ‘ ’ 0.3 ‘.’ 0.6 ‘,’ 0.8 ‘+’ 0.9 ‘*’ 0.95 ‘B’ 1

> symnum(cov2cor(cR$cov))

   X1 X2 X3 X4 Y1 Y2 Y3 Y4
X1 1
X2 +  1
X3 ,  +  1
X4 +  *  ,  1
Y1 ,  +  ,  *  1
```

higher-dimensional

```
> sfsmisc::mult.fig(4, main = "plot( covMcd(pulpfiber), . \"all\
> plot(cR, type = "all")  ;  par(op)
```



Questions on Section 4 — “Multivariate Analysis” ?