Preliminary Outlier Detection in Missing Values Free data

February 8, 2022

Problems : Valid estimation of a composition of rocks.

1. composition of major elements with outliers
2. composition of major elements with outliers and missing values
3. : composition of major elements with outliers and missing values
4. : composition of all elements with outliers and missing values

An initial goal of this work is to provide robust estimates of the composixtion of each rock. This problem is made difficult by the presence of both missing values (laboratories did not measure any composition) and outliers (the measures reported are extreme).

Setting aside the problem of missing values for now, one may look at a subcomposition $\mathbf{x} \in \mathbb{S}^D$ of oxides of major elements.

One begin by importing a dataset, for example GeoPT48, a monzonite.

```
setwd("/home/max/Documents/MStatistics/MA2/Thesis/Repository/")
data <- read_csv("data/raw/GeoPT48 -84Ra.csv")

##
## - Column specification ----------------------------
## cols(
##   .default = col_double(),
##   Laboratory = col_character(),
##   Au = col_logical(),
##   N = col_logical(),
##   Os = col_logical()
## )
## i Use 'spec()' for the full column specifications.
```

Then, one looks at the subcomposition of major elements :

```
sel <-c("SiO2","TiO2","Al2O3","Fe2O3T","MnO","MgO","CaO","Na2O",
        "K2O","P2O5")
df.majors <- select(data,all_of(sel))
# closure operation
df.majors <- data.frame(clo(df.majors))
```

Then missing values are imputed by the column geometric mean

```
geomean.v <- sapply(rbind(df.majors),geomean)
for (i in 1:ncol(df.majors)){
  df.majors[,i][is.na(df.majors[,i])] <- geomean.v[i]
}
```

One then transform the dataset using the CLR transformation which is a mapping $\mathbb{S}^D - > \mathbb{U}^D$:

$$\mathbf{z} = clr(\mathbf{x} = [log(x_1/g(x)), .., log(x_D/g(x))] \qquad (1)$$

Where $\mathbb{U}^D$ is an hyperplane of $\mathbb{R}^D$ defined as :

$$U^D = \left\{ [u_1, .., u_D] : \sum_{i=1}^{D} = 0 \right\}$$

```
cr.df <- data.frame()
clr.df <- data.frame()
# cr.df, divide each entries in a column by
# the geometric mean of this column
cr.df <- sweep(df.majors,MARGIN = 2,FUN="/",STATS = geomean.v)
# clr.df is the natural logarithm of cr.df.
# Now this dataframe contains clr components
clr.df <- log(cr.df)
```

Then principal component analysis is conducted. Z denotes the mean-centered data matrix X :

$$z_{ij} = x_{ij} - \mu_j$$

Where μ_j denotes the arithmetic mean of the j-th column. Recall that here using the arithmetic mean is justified because X now lives in a subspace of $\mathbb{R}^D$ which is no longer constrained by the unit sum.

Perform Singular Value Decomposition on clr.df :

$$Z = UDW^T = (UD)W^T = Z^* W^T, Z \in \mathbb{R}^{n\times(d-1)} U \in \mathbb{R}^{n\times p}, D \in \mathbb{R}^{p\times p}, W \in \mathbb{R}^{(d-1)\times p} \quad (2)$$

Z^* denotes the projection of the mean-centered data matrix on a space of dimension p. Hopefully, Z^* contains enough meaningful information about Z while having a much lower number of dimensions. To evaluate how good Z^* approximates Z, one looks at the proportion of variance explained by each of the components and more specifically, the cumulative proportion of variance explained by each of the components when these components are ranked from most to less important.

```
pca.clr <- prcomp(clr.df,scale = T,rank. = ncol(clr.df)-1 )
summary(pca.clr)

## Importance of first k=9 (out of 10) components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.5272 1.1288 1.0464 0.72257 0.59620 0.48113 0.23209
## Proportion of Variance 0.6387 0.1274 0.1095 0.05221 0.03555 0.02315 0.00539
## Cumulative Proportion  0.6387 0.7661 0.8756 0.92779 0.96334 0.98649 0.99187
##                            PC8     PC9
## Standard deviation     0.18604 0.17566
## Proportion of Variance 0.00346 0.00309
## Cumulative Proportion  0.99533 0.99842
```

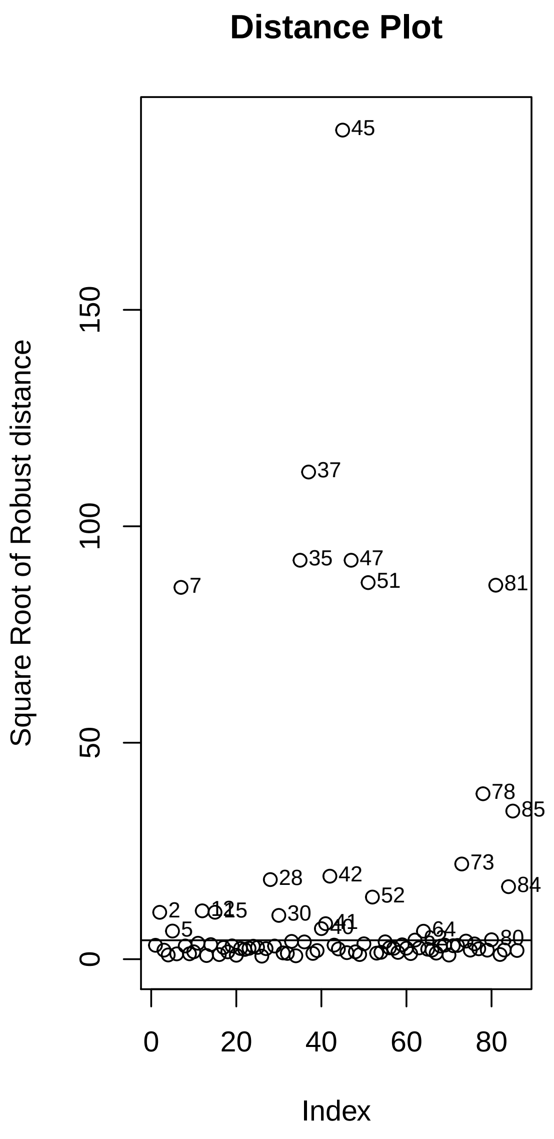
The importance of a component, in terms of proportion of variance explained, is directly related to the D matrix whose eigenvalues squared are directly related to the proportion of variance explained through the relationship :

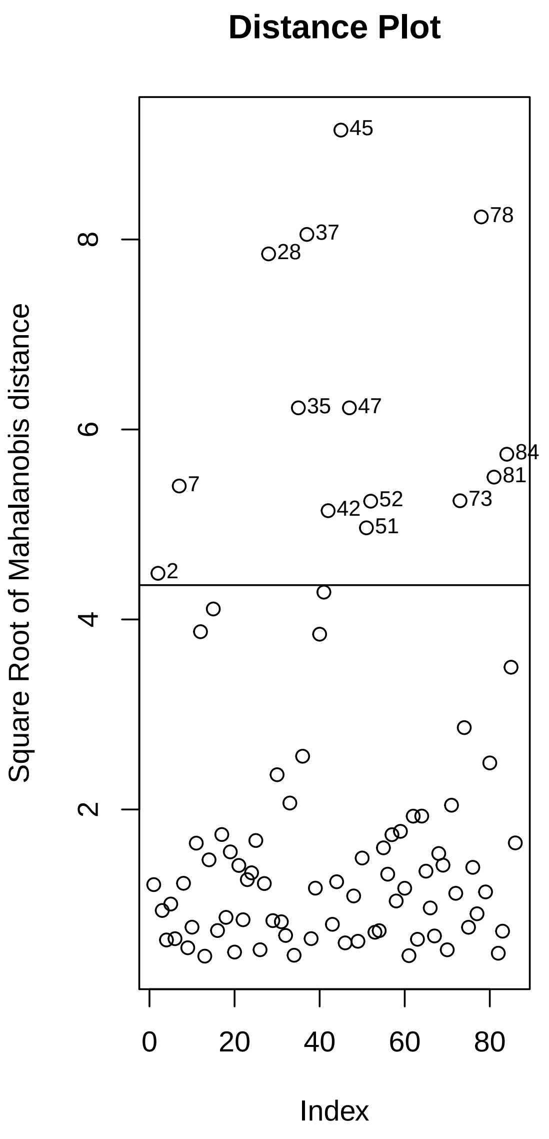
$$\lambda_i = d_i^2/(n-1) \tag{3}$$

Where d_i denotes the i-th diagonal element in the square matrix D of the singular values and n is the number of principal components which is equal to the dimensionality of the original data matrix X.

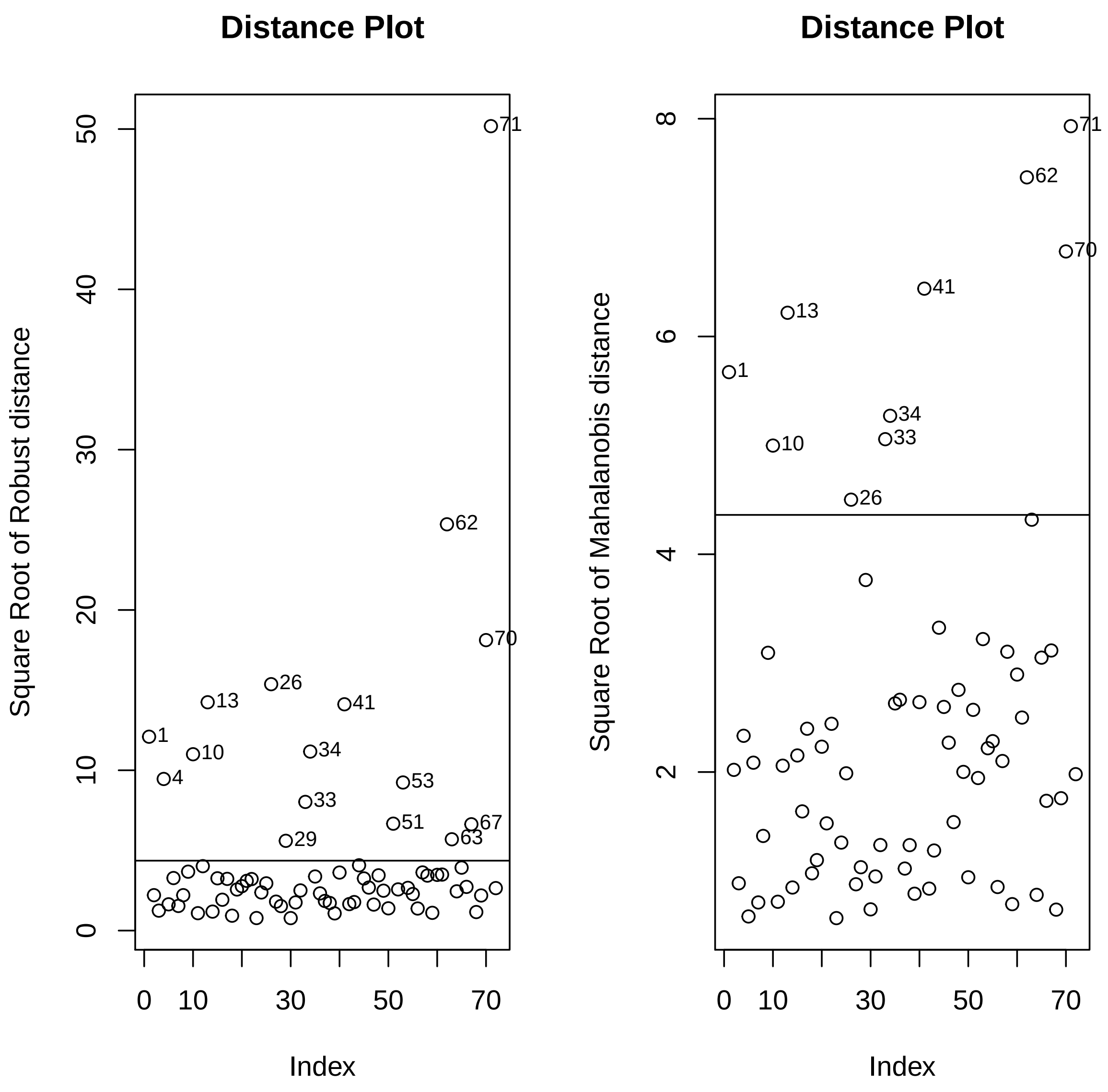
Now, one looks at the rank-two approximation of Z (as the biplot does) :

```r
# autoplot
autoplot(pca.clr,loadings=T,loadings.label=T)+theme_bw()
```



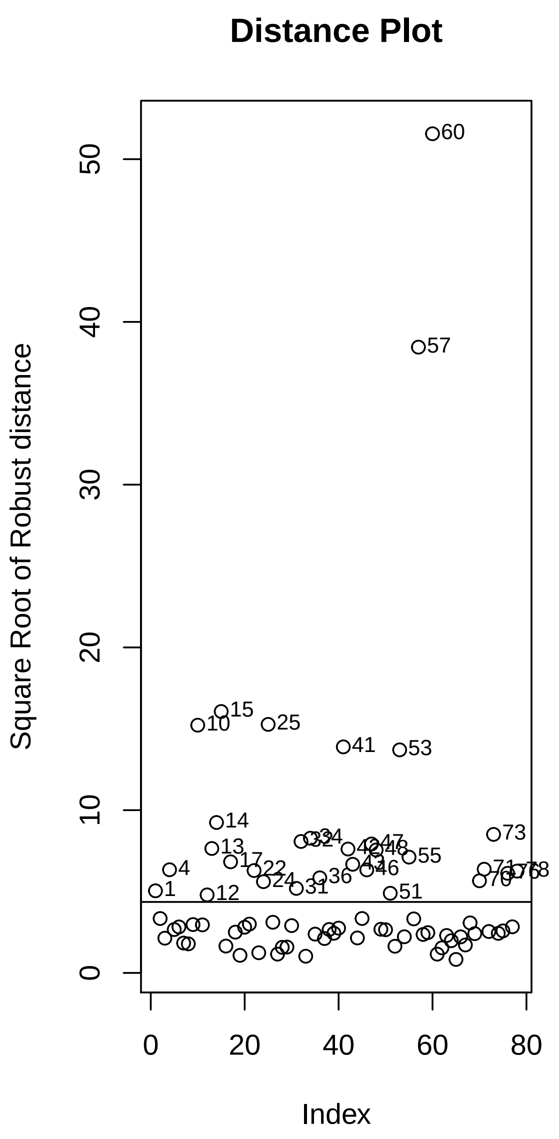


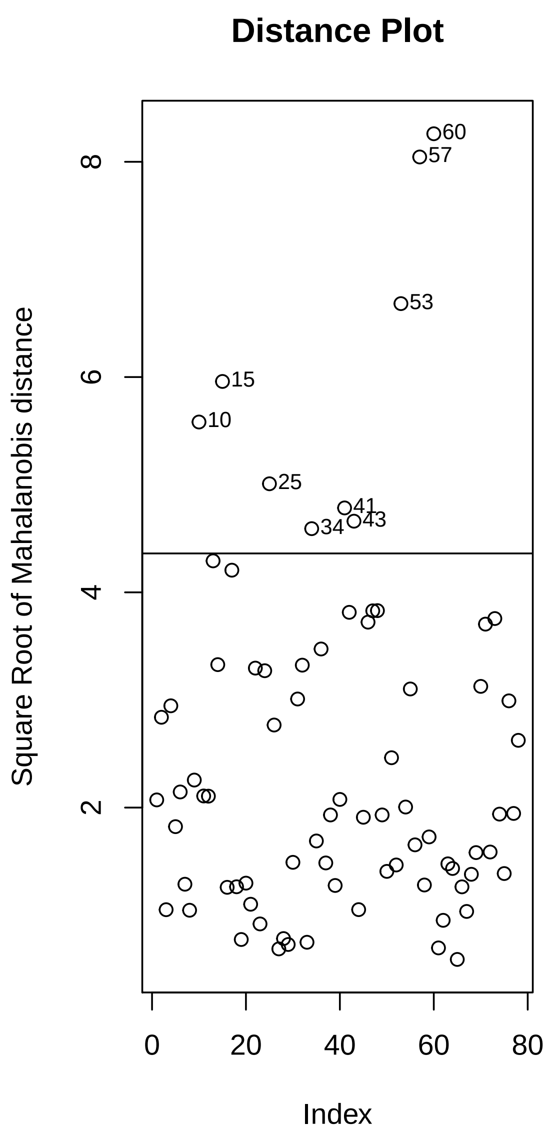
```r
# manually extracting the rank 2 approximation of Z
Z.approx <- data.frame(pca.clr$x[,c(1,2)])
colnames(Z.approx) = c("PC1","PC2")
ggplot(aes(x=PC1,y=PC2),data=Z.approx)+theme_bw()+geom_point()
```



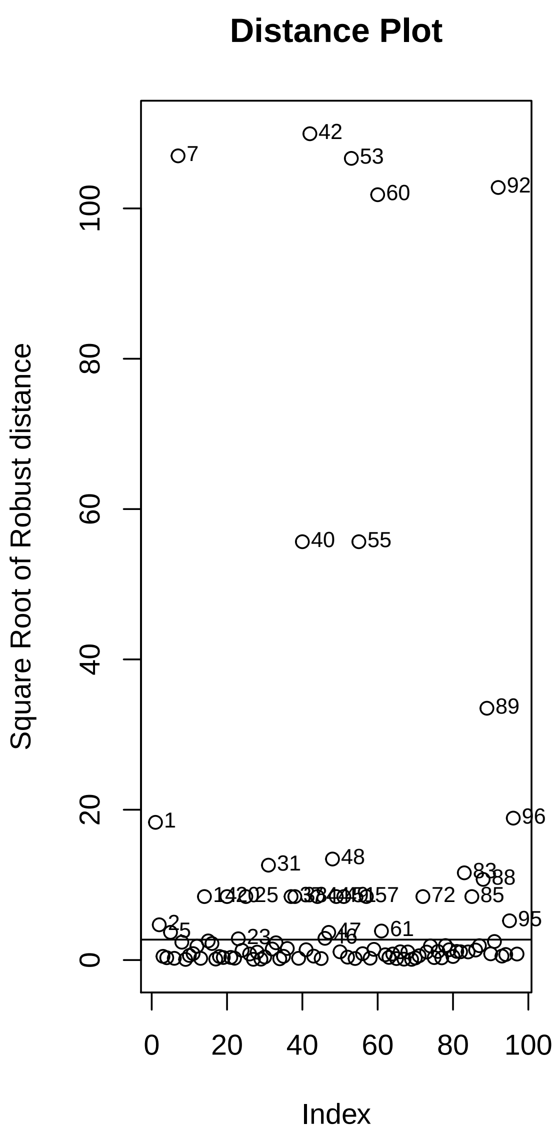
Flagging the outliers, remarkable discrepancy between robust method (MCD estimator of the location, MCD is affine equivariant and has a high breakdown value) vs classical method (sample covariance matrix).

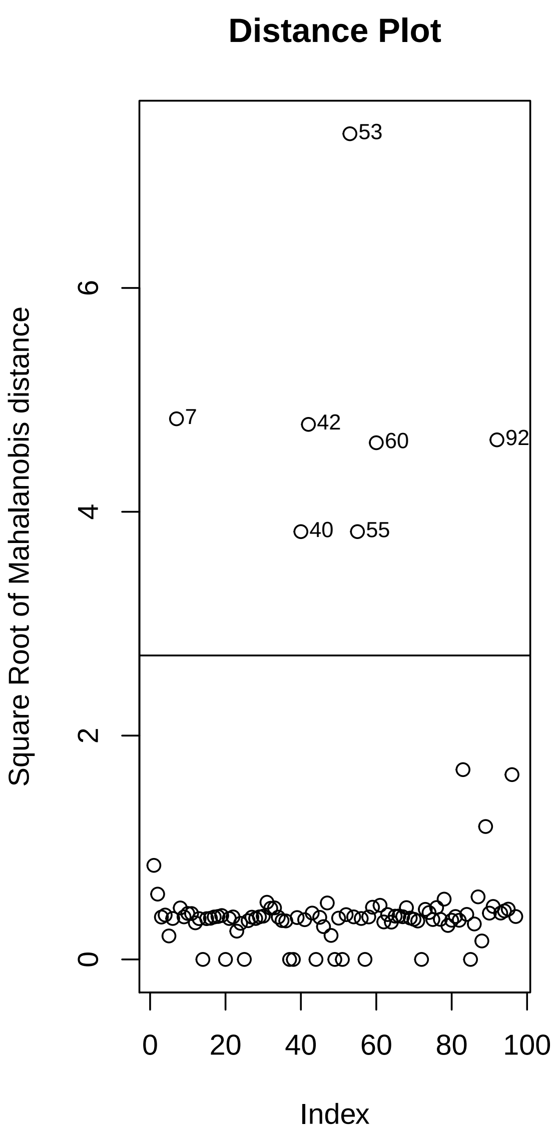
```r
Z2.mcd <- covMcd(Z.approx,)
tolEllipsePlot(Z.approx,classic = T)
```



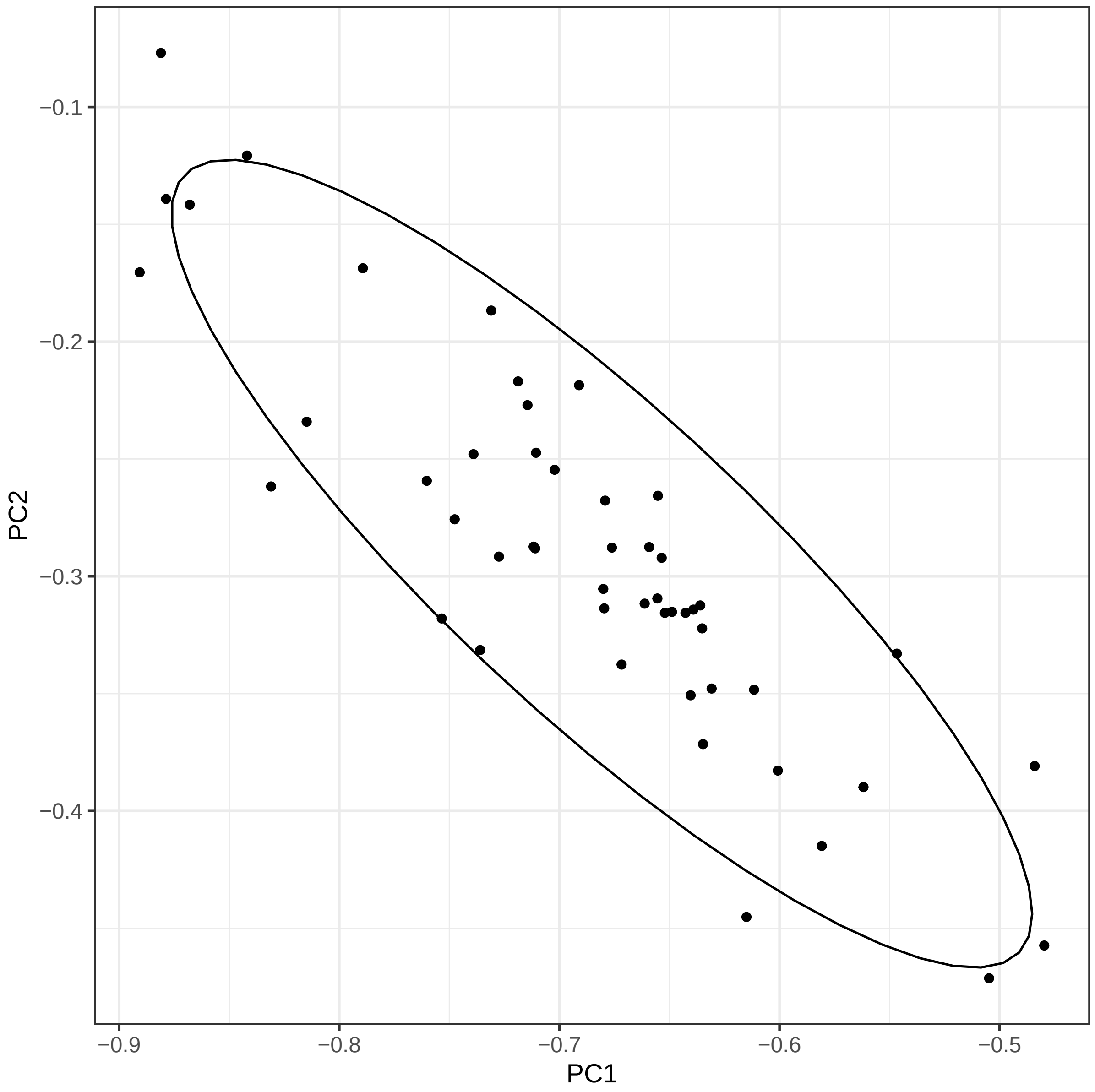


```
plot(Z2.mcd,which = c("distance"),classic = TRUE)
```





```
Z2.outfree <- Z.approx[Z2.mcd$best,]
ggplot(aes(x=PC1,y=PC2),data=Z2.outfree)+theme_bw()+geom_point()+stat_ellipse()
```



This approach has a pitfall. The estimation of the PC is itself not robust. Robcompositions package provides a robust PCA estimation method :

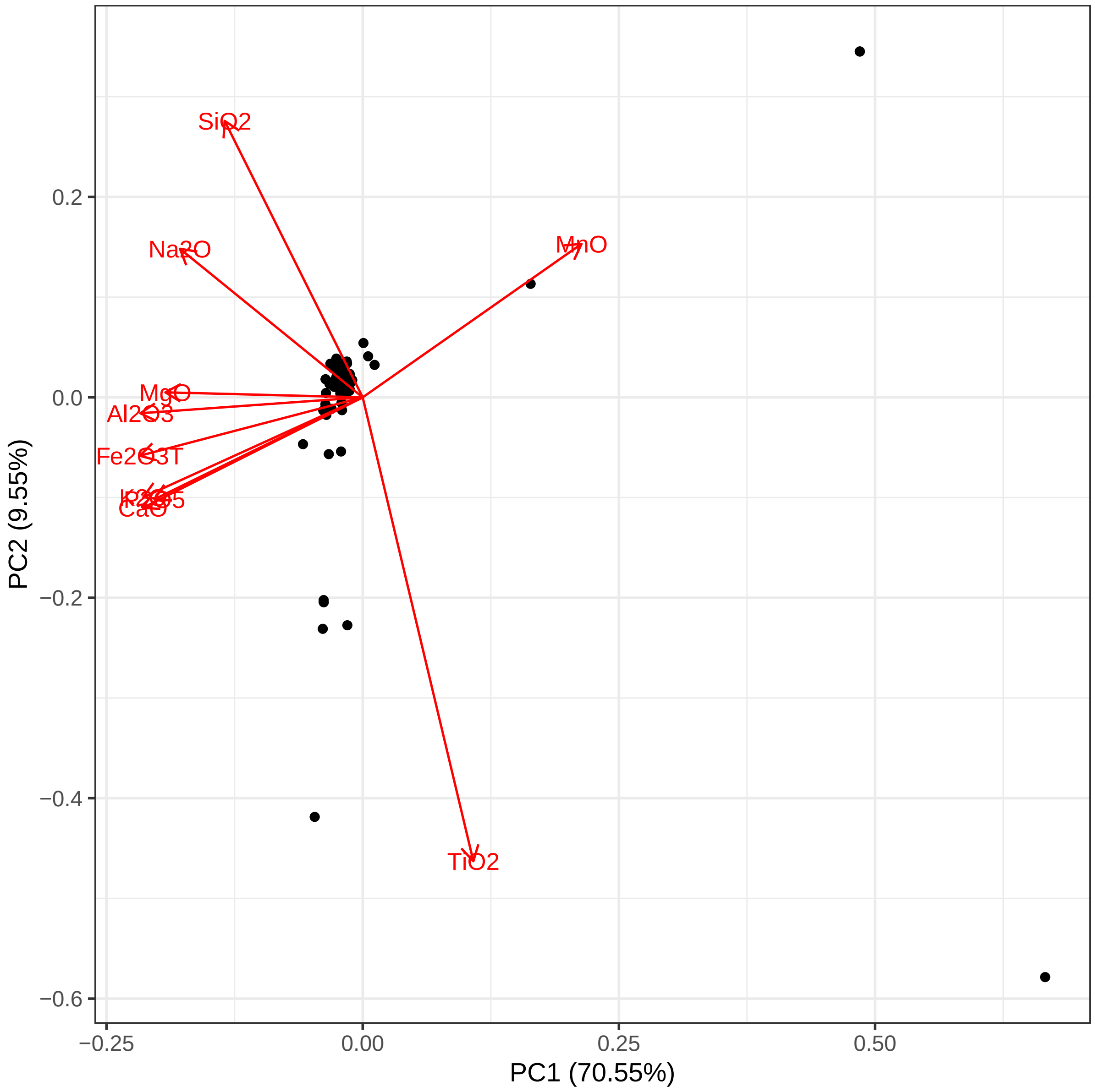
> The compositional data set is expressed in isometric logratio coordinates. Afterwards, robust principal component analysis is performed. Resulting loadings and scores are back-transformed to the clr space where the compositional biplot can be shown. CITE ROBCOMP R package

```
rob.pca.clr <- pcaCoDa(df.majors)
summary(rob.pca.clr)

## Importance of components:
##                           Comp.1     Comp.2     Comp.3     Comp.4     Comp.5
## Standard deviation     0.1253479 0.03149760 0.02812230 0.02340062 0.01896927
## Proportion of Variance 0.8331335 0.05260606 0.04193556 0.02903592 0.01908016
## Cumulative Proportion  0.8331335 0.88573954 0.92767510 0.95671102 0.97579118
##                              Comp.6      Comp.7       Comp.8       Comp.9
## Standard deviation     0.01647891 0.01044713 0.007836675 0.0038005573
## Proportion of Variance 0.01439918 0.00578728 0.003256451 0.0007659058
## Cumulative Proportion  0.99019036 0.99597764 0.999234094 1.0000000000
```

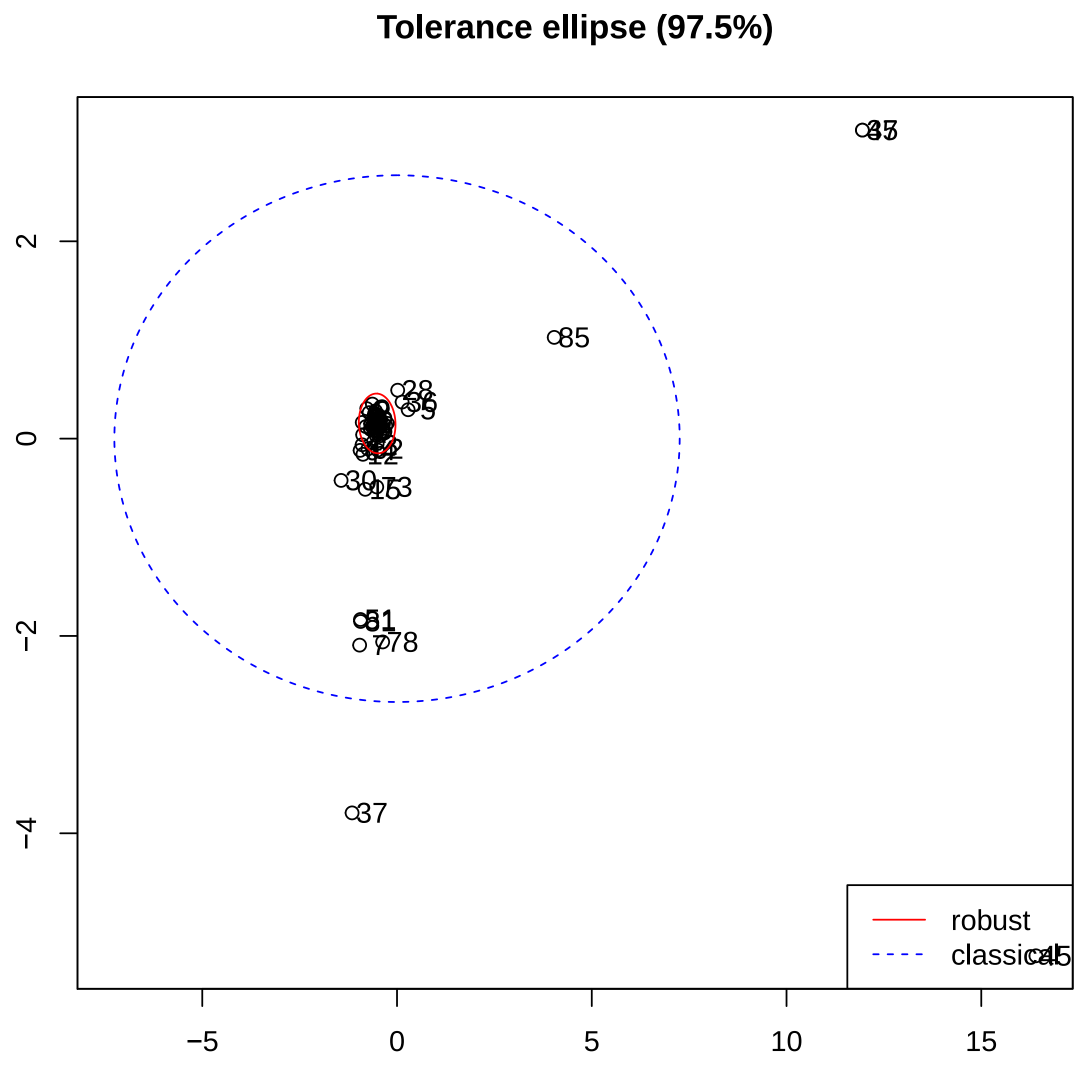
One sees that the first two PC's found by using the robust method explains (89 %) whereas in the classical method, the first two PC's explained only 76 %. The outlier detection is repeated in the first 2 robust PC's subspace.

```
Z.approx.rob <- data.frame(rob.pca.clr$scores[,c(1,2)])
colnames(Z.approx.rob) = c("PC1","PC2")
ggplot(aes(x=PC1,y=PC2),data=Z.approx.rob)+theme_bw()+geom_point()
```

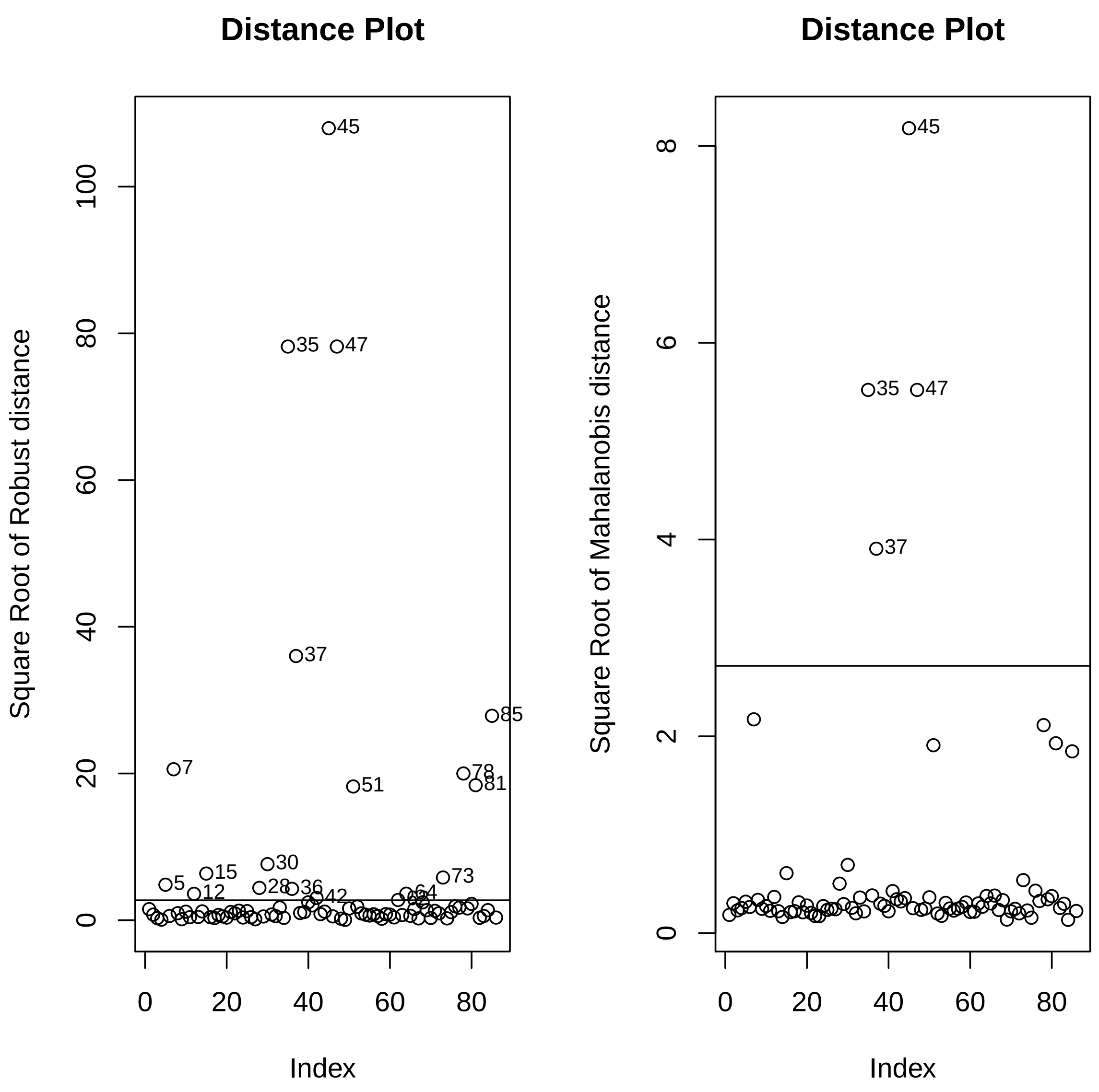


At first glance, there seem to be less outliers in the robust first two PC's space. This is checked by using diagnostic plots :

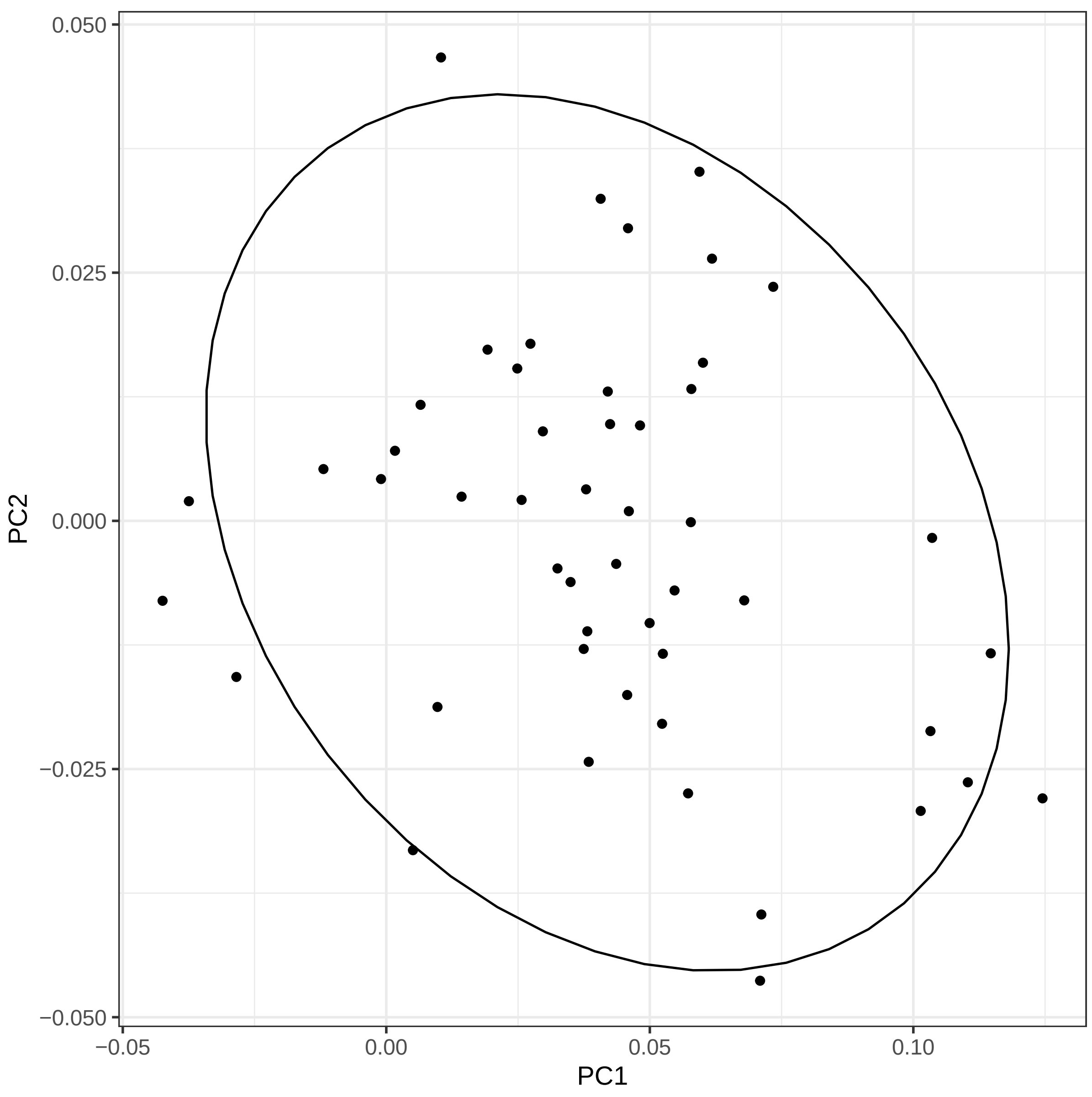
```r
Z2.rob.mcd <- covMcd(Z.approx.rob,)
tolEllipsePlot(Z.approx.rob,classic = T)
```



```r
plot(Z2.rob.mcd,which = c("distance"),classic = TRUE)
```



```
Z2.rob.outfree <- Z.approx.rob[Z2.rob.mcd$best,]
ggplot(aes(x=PC1,y=PC2),data=Z2.rob.outfree)+theme_bw()+geom_point()+stat_ellipse()
```



There seems to be no association between PC1 and PC2, which makes much more sense. Eventually, an estimation for rock 1 composition is

```
df.majors.out.free <- df.majors[Z2.rob.mcd$best,]
df.majors.out.free.clr <- data.frame(clr(df.majors.out.free))
clr.mean <- colMeans(df.majors.out.free.clr)
clr.cov <- Cov(df.majors.out.free.clr)

mean.estimate <- as.vector(clrInv(clr.mean))
cov.estimate <- clrInv(as.matrix(clr.cov$cov))
cov.estimate

##         SiO2         TiO2         Al2O3        Fe2O3T       MnO
## SiO2    "0.10001128" "0.10000217" "0.09999462" "0.10000478" "0.09999436"
## TiO2    "0.10000216" "0.10003559" "0.09999296" "0.09999844" "0.10000563"
## Al2O3   "0.09999461" "0.09999295" "0.10007133" "0.09997229" "0.09995735"
## Fe2O3T "0.10000478" "0.09999844" "0.09997229" "0.10004305" "0.10000978"
## MnO    "0.09999435" "0.10000562" "0.09995734" "0.10000978" "0.10012407"
## MgO    "0.09999526" "0.09997199" "0.09998626" "0.09999430" "0.09998264"
## CaO    "0.09999876" "0.10000198" "0.10001290" "0.09999585" "0.09998321"
## Na2O   "0.09999735" "0.09998972" "0.09998627" "0.09998342" "0.09999514"
## K2O    "0.10000055" "0.09999952" "0.10002800" "0.09999778" "0.09997054"
## P2O5   "0.10000086" "0.10000198" "0.09999805" "0.10000030" "0.09997734"
##        MgO          CaO          Na2O         K2O          P2O5
## SiO2   "0.09999527" "0.09999876" "0.09999735" "0.10000056" "0.10000086"
## TiO2   "0.09997200" "0.10000198" "0.09998972" "0.09999952" "0.10000198"
## Al2O3  "0.09998626" "0.10001289" "0.09998627" "0.10002800" "0.09999804"
## Fe2O3T "0.09999431" "0.09999585" "0.09998342" "0.09999778" "0.10000030"
## MnO    "0.09998264" "0.09998320" "0.09999513" "0.09997053" "0.09997733"
## MgO    "0.10011263" "0.09999039" "0.10000429" "0.09996256" "0.09999967"
## CaO    "0.09999040" "0.10002174" "0.09999453" "0.10000190" "0.09999873"
## Na2O   "0.10000429" "0.09999453" "0.10007487" "0.09998931" "0.09998510"
## K2O    "0.09996257" "0.10000190" "0.09998931" "0.10004847" "0.10000135"
## P2O5   "0.09999968" "0.09999873" "0.09998511" "0.10000135" "0.10003660"
## attr(,"class")
## [1] "acomp"
```

How does it compare with the naive estimates of GeoPT ?

```
mean.naive <- colMeans(clo(df.majors))
od <- outCoDa(df.majors, quantile = 0.975, method = "robust", alpha = 0.9, coda = TRUE
df.major.wo.outliers <- df.majors[od$outlierIndex,]
mean.naive.wo.outlier <- colMeans(clo(df.major.wo.outliers))
```

Not much difference for SiO2 *but* TiO2 concentration is twice as high in the naive way, 5 times higher in the naive way when removing outliers. MnO concentration is 10 times higher in the naive way.

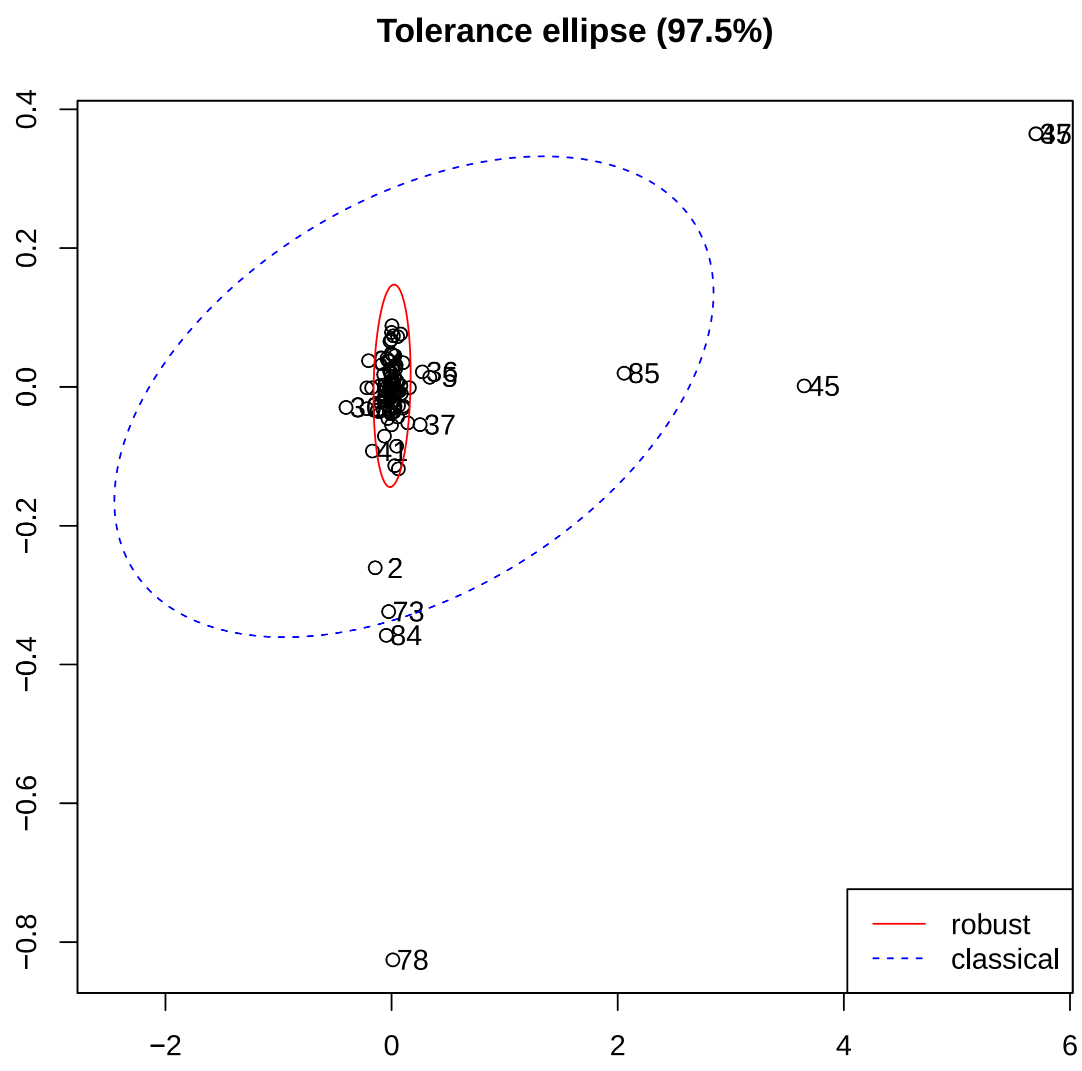
```
df <- data.frame(mean.estimate,mean.naive,mean.naive.wo.outlier)
df

##        mean.estimate  mean.naive mean.naive.wo.outlier
## SiO2     0.581220398 0.563604249           0.512617236
## TiO2     0.010824249 0.015611874           0.029696364
## Al2O3    0.197571359 0.198331406           0.200360748
## Fe2O3T   0.048093886 0.048458236           0.048884754
## MnO      0.001403392 0.012352326           0.045442617
## MgO      0.011311008 0.011289539           0.011110850
## CaO      0.037648453 0.038427579           0.040367724
## Na2O     0.067546763 0.066829195           0.064474391
## K2O      0.038348247 0.038910243           0.040419032
## P2O5     0.006032244 0.006185351           0.006626284
```

```
df$element <- row.names(df)
df.plot <- df %>% pivot_longer(cols=starts_with("mean"))
df.plot

## # A tibble: 30 x 3
##    element name                    value
##    <chr>   <chr>                   <dbl>
##  1 SiO2    mean.estimate          0.581
##  2 SiO2    mean.naive             0.564
##  3 SiO2    mean.naive.wo.outlier 0.513
##  4 TiO2    mean.estimate          0.0108
##  5 TiO2    mean.naive             0.0156
##  6 TiO2    mean.naive.wo.outlier 0.0297
##  7 Al2O3   mean.estimate          0.198
##  8 Al2O3   mean.naive             0.198
##  9 Al2O3   mean.naive.wo.outlier 0.200
## 10 Fe2O3T  mean.estimate          0.0481
## # ... with 20 more rows

df.plot$group <- ifelse(df.plot$element %in% c("SiO2","Al2O3"),yes = "large","small")
ggplot(aes(x=element,y=value),data = df.plot)+geom_col(aes(fill=name),position = "dodg
```



Preliminary conclusion : naive without outlier downplay the concentration of elements present in large quantities and blow the concentration of elements present in small quantities.