timeseries analysis : normality, q-q plots, distr estimation

Andreas Hadjiprocopis

February, 2018

```
#!/usr/bin/env Rscript
```

```
set.seed(1234)

source('lib/UTIL.R');
source('lib/DATA.R');
source('lib/IO.R');
source('lib/TS.R');
source('lib/MC.R');
source('lib/SEASON.R');
source('lib/MIXTURES.R');

infile='cleaned_data/dat1.eliminateNA.csv'

dat1 <- data.frame(read_data(
    filename=infile
))
```

```
## read_data(): data read from file 'cleaned_data/dat1.eliminateNA.csv'.
```

```
if( is.null(dat1) ){
    cat("call to read_data() has failed for file '",infile,"'.\n", sep='')
    quit(status=1)
}
dummy <- remove_columns(inp=dat1, colnames_to_remove=c('id'))
dat1_noid <- dummy[['retained']]
dat1_id <- dummy[['removed']]
dat1_detrended_id <- list(c(dat1_id[['id']][2:length(dat1_id[['id']])]))
names(dat1_detrended_id) <- c('id')
# detrend the data
dat1_detrended <- detrend_dataset(inp=dat1_noid,times=1)
```

Check for normality: do data come from a Gaussian distribution process For the Anderson-Darling and Shapiro-Wilk normality test the p-value refers to the null hypothesis being that data DOES not come from a Gaussian distribution process (i.e. high p-value means 'a high probability that data comes from Gaussian distribution process') Data (detrended or not) do not seem to come from a Gaussian distribution process because the p-values of the test are very very small. First let's test a normally distributed sample just to see what kind of p-values to expect.

```
library(nortest)
print(ad.test(rnorm(n=1000, mean=5, sd=3)))
```

```
##
##  Anderson-Darling normality test
##
## data:  rnorm(n = 1000, mean = 5, sd = 3)
## A = 0.57857, p-value = 0.1323
```

```
print(ad.test(rnorm(n=1000, mean=5, sd=1)))
```

```
##
##  Anderson-Darling normality test
##
## data:  rnorm(n = 1000, mean = 5, sd = 1)
## A = 0.18795, p-value = 0.9025
```

```
print(shapiro.test(rnorm(n=1000, mean=5, sd=3)))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  rnorm(n = 1000, mean = 5, sd = 3)
## W = 0.99866, p-value = 0.6587
```

```
print(shapiro.test(rnorm(n=1000, mean=5, sd=1)))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  rnorm(n = 1000, mean = 5, sd = 1)
## W = 0.99902, p-value = 0.8817
```

it looks like if p-value is not extremely small then we can assume it passes the test of normality.

```
for(acol in names(dat1_noid)){
    cat("original data, col", acol, "\n", sep='')
    x<-ad.test(dat1_noid[[acol]])
    print(x)
}
```

```
## original data, colA
##
##  Anderson-Darling normality test
##
## data:  dat1_noid[[acol]]
## A = 75.256, p-value < 2.2e-16
##
## original data, colB
##
##  Anderson-Darling normality test
##
## data:  dat1_noid[[acol]]
## A = 1147.2, p-value < 2.2e-16
##
## original data, colC
##
##  Anderson-Darling normality test
##
## data:  dat1_noid[[acol]]
## A = 2030.8, p-value < 2.2e-16
##
## original data, colD
##
##  Anderson-Darling normality test
```

```
##
## data:  dat1_noid[[acol]]
## A = 138.07, p-value < 2.2e-16
##
## original data, colE
##
##  Anderson-Darling normality test
##
## data:  dat1_noid[[acol]]
## A = 7884.2, p-value < 2.2e-16
##
## original data, colF
##
##  Anderson-Darling normality test
##
## data:  dat1_noid[[acol]]
## A = 10508, p-value < 2.2e-16
```

```
for(acol in names(dat1_detrended)){
    cat("detrended data, col", acol, "\n", sep='')
    x<-ad.test(dat1_detrended[[acol]])
    print(x)
}
```

```
## detrended data, colA
##
##  Anderson-Darling normality test
##
## data:  dat1_detrended[[acol]]
## A = 3369.7, p-value < 2.2e-16
##
## detrended data, colB
##
##  Anderson-Darling normality test
##
## data:  dat1_detrended[[acol]]
## A = 3424, p-value < 2.2e-16
##
## detrended data, colC
##
##  Anderson-Darling normality test
##
## data:  dat1_detrended[[acol]]
## A = 1491.1, p-value < 2.2e-16
##
## detrended data, colD
##
##  Anderson-Darling normality test
##
## data:  dat1_detrended[[acol]]
## A = 10395, p-value < 2.2e-16
##
## detrended data, colE
##
##  Anderson-Darling normality test
```

```
##
## data:  dat1_detrended[[acol]]
## A = 11831, p-value < 2.2e-16
##
## detrended data, colF
##
##  Anderson-Darling normality test
##
## data:  dat1_detrended[[acol]]
## A = 12326, p-value < 2.2e-16
```

Conclusion: data do not pass the Normality test. This means that non-parameteric statstical tools should be used for assessing the data. For example Pearson's require bivariate normality while Spearman's does not. print summary statistics for original data:

```
for(acol in names(dat1_noid)){
    x <- dat1_noid[[acol]]
    cat("Original data, column ", acol, ", mean=", mean(x), ", sd=", sd(x), "\n", sep='')
}
```

```
## Original data, column A, mean=49.39303, sd=9.499579
## Original data, column B, mean=80.99835, sd=11.71468
## Original data, column C, mean=5.08511, sd=5.262529
## Original data, column D, mean=29.81647, sd=0.3481571
## Original data, column E, mean=83.66461, sd=159.3282
## Original data, column F, mean=0.3703302, sd=0.8356359
```

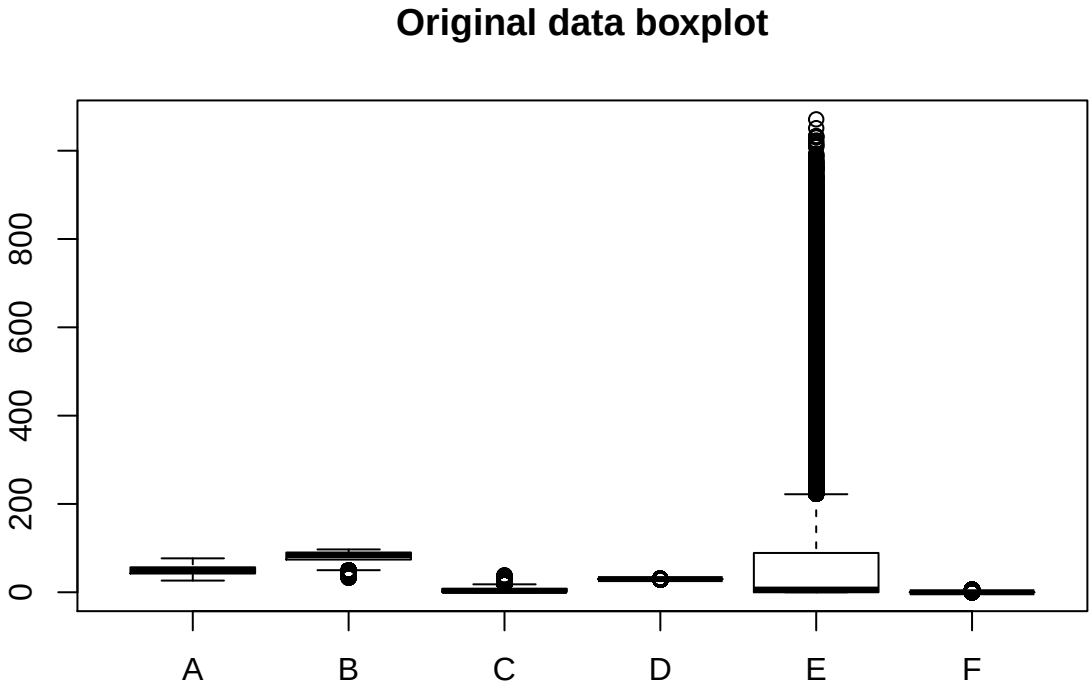
print summary statistics for detrended data:

```
for(acol in names(dat1_detrended)){
    x <- dat1_detrended[[acol]]
    cat("Detrended data, column ", acol, ", mean=", mean(x), ", sd=", sd(x), "\n", sep='')
}
```

```
## Detrended data, column A, mean=-1.025213e-05, sd=0.2437584
## Detrended data, column B, mean=0.0002468104, sd=1.291171
## Detrended data, column C, mean=0, sd=2.358147
## Detrended data, column D, mean=-2.018302e-06, sd=0.00484357
## Detrended data, column E, mean=0, sd=57.02373
## Detrended data, column F, mean=-9.7284e-21, sd=0.1546259
```

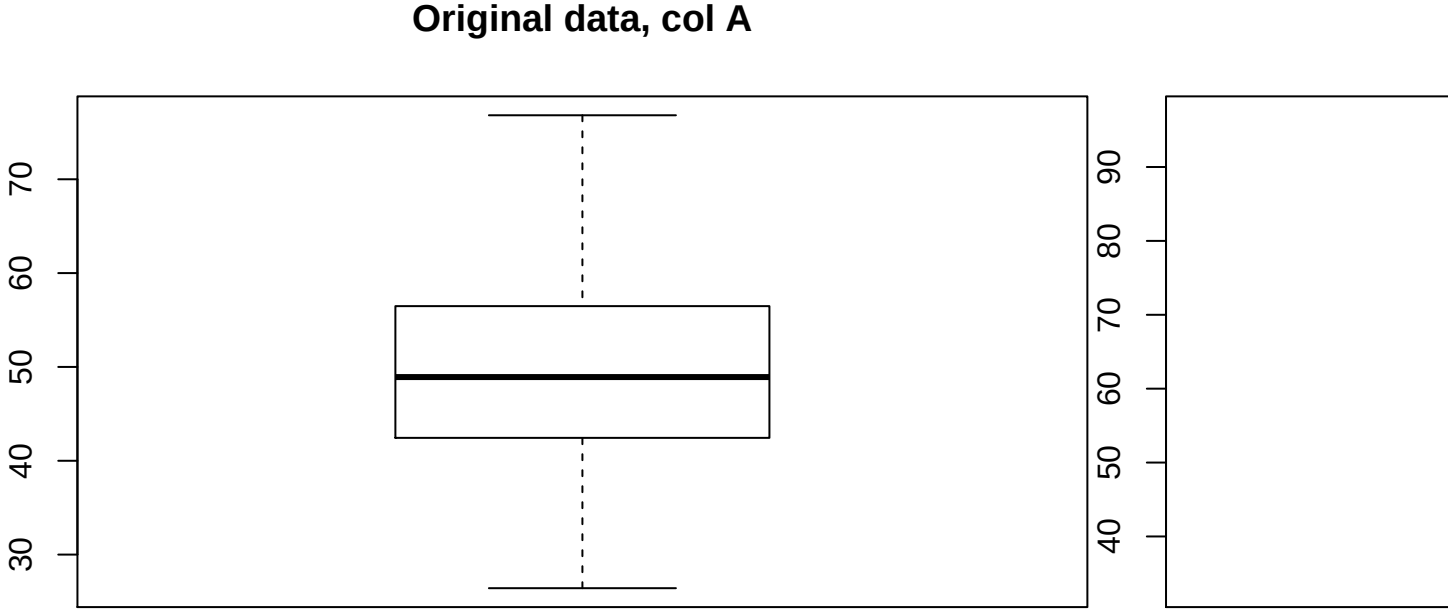
Boxplot of all of original data's columns in one page for comparison Note: a boxplot shows the minimum and maximum values in the whiskers The interquartile range (between lower and upper quartiles) is the rectangle's width. The median is the indicator line inside the rectangle.

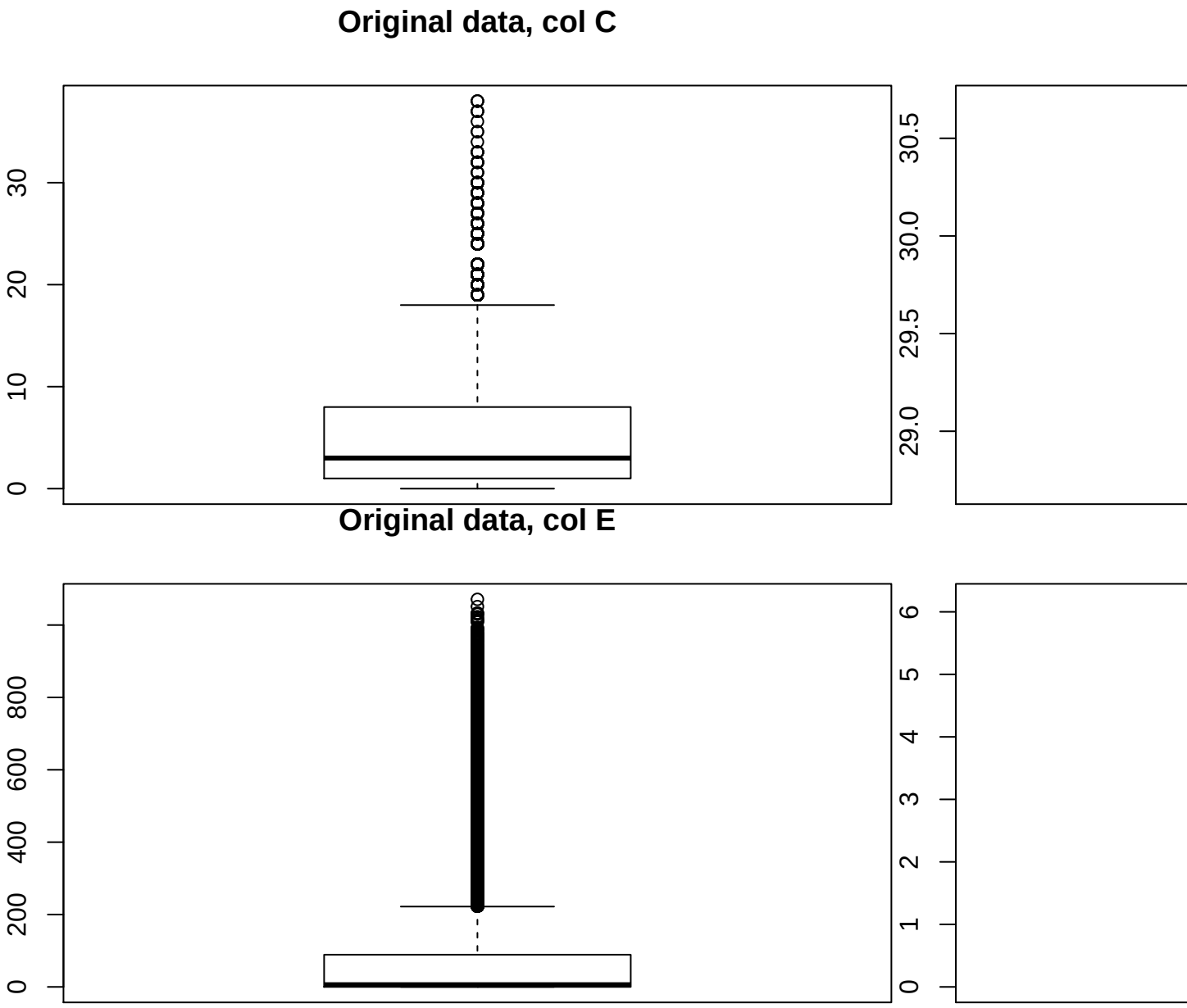
```
boxplot(
    dat1_noid,
    main=paste0('Original data boxplot', sep='')
)
```



boxplot of original data per column:

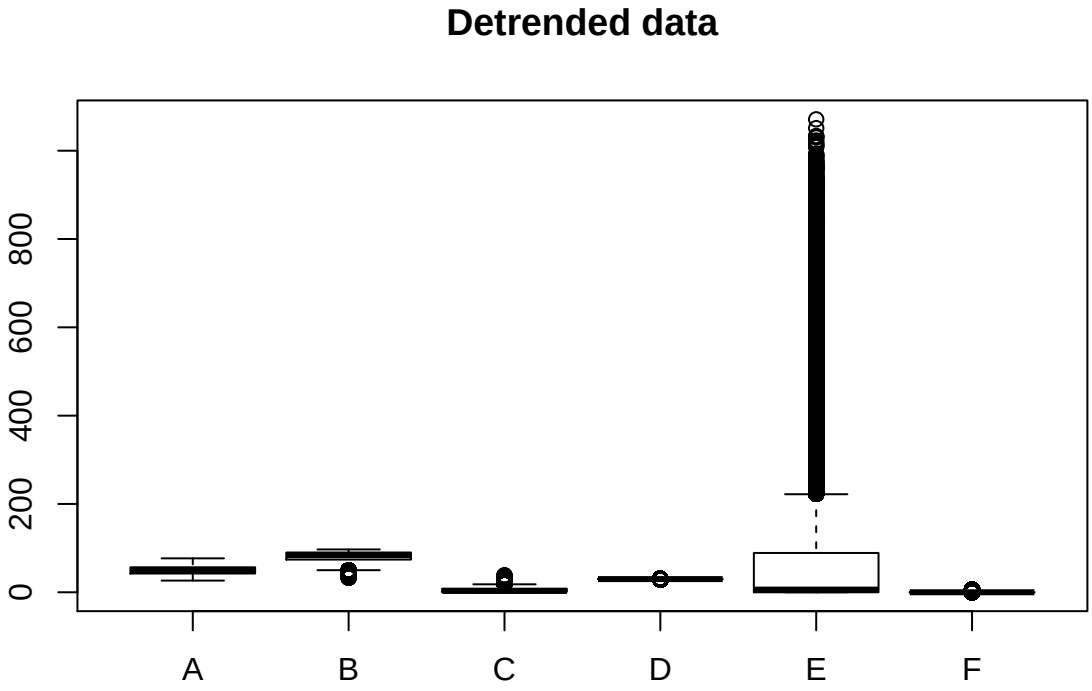
```
for(acol in names(dat1_noid)){
    x <- dat1_noid[[acol]]
    boxplot(
        x,
        main=paste0('Original data, col ', acol, sep='')
    )
}
```





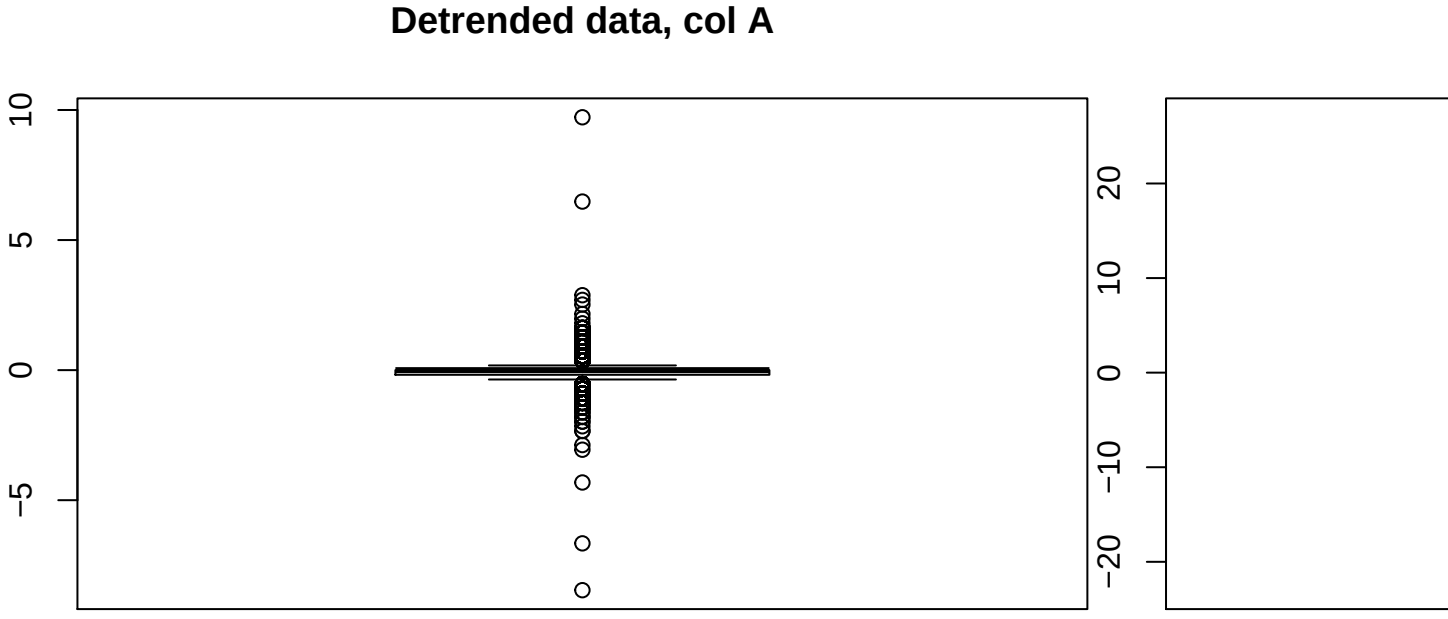
boxplot of all of detrended data's columns in one page for comparison

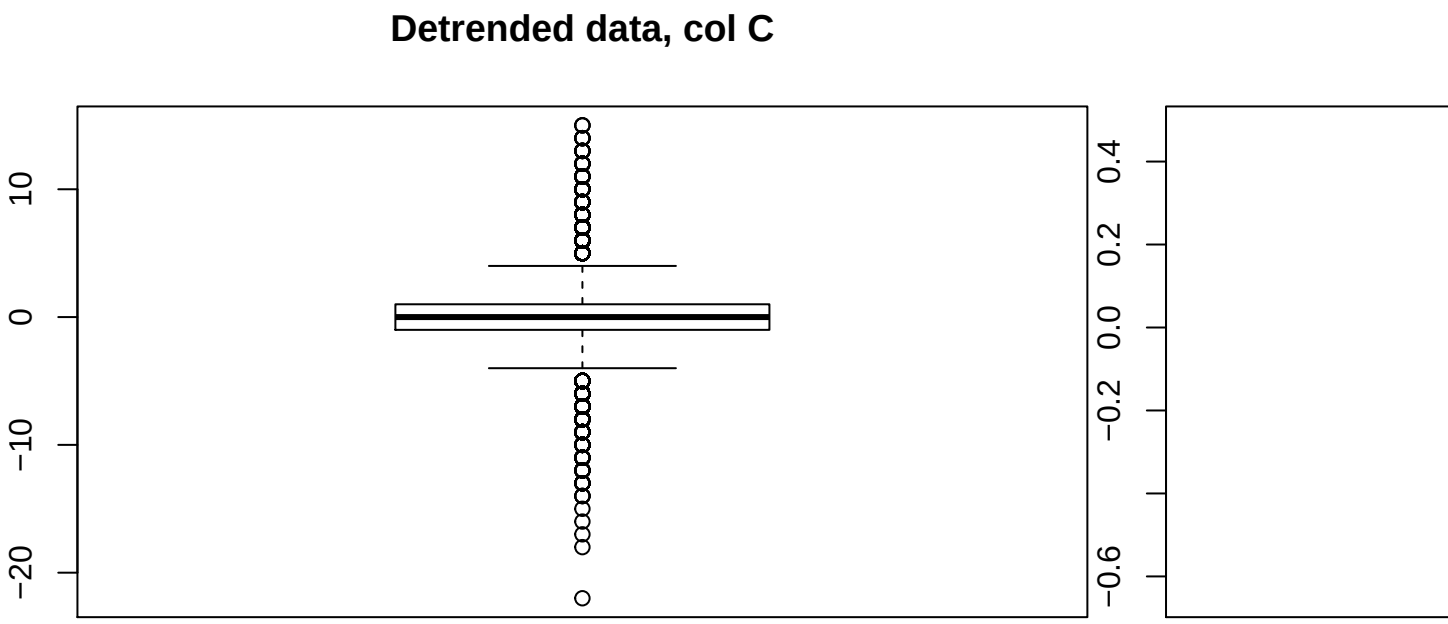
```
boxplot(
    dat1_noid,
    main=paste0('Detrended data', sep='')
)
```

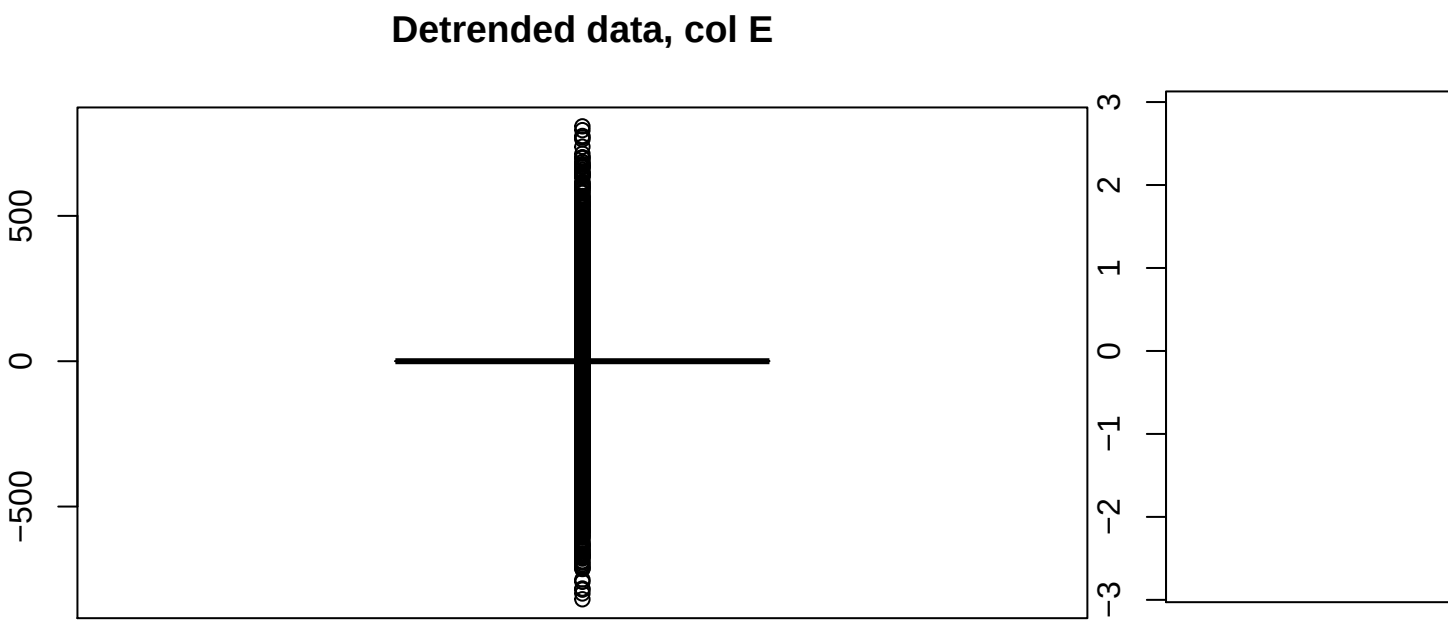


boxplot of detrended data per column:

```
for(acol in names(dat1_noid)){
    x <- dat1_detrended[[acol]]
    boxplot(
        x,
        main=paste0('Detrended data, col ', acol, sep='')
    )
}
```







Do a scatter plot of pairs of variables with a regression line added for the original data

```
library(ggpubr)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: magrittr
```

```
adf <- list2dataframe(dat1_noid)
cnames <- names(adf)
Ncnames <- length(cnames)
for(i in 1:Ncnames){
    acol1 = cnames[i]
    for(j in (i+1):Ncnames){
        acol2 = cnames[j]
        ggscatter(
```

```
            adf,
            x=acol1,
            y=acol2,
            add="reg.line",
            conf.int=TRUE,
            cor.coef=TRUE,
            cor.method = "spearman",
            title=paste0('Scatter plot of original data, columns ', acol1, ' and ', acol2,sep='')
        )
    }
}
```

Do a scatter plot of pairs of variables with a regression line added for the detrended data

```
library(ggpubr)
adf <- list2dataframe(dat1_detrended)
cnames <- names(adf)
Ncnames <- length(cnames)
for(i in 1:Ncnames){
    acol1 = cnames[i]
    for(j in (i+1):Ncnames){
        if( j > Ncnames ){ next }
        acol2 = cnames[j]
        ggscatter(
            adf,
            x=acol1,
            y=acol2,
            add="reg.line",
            conf.int=TRUE,
            cor.coef=TRUE,
            cor.method = "spearman",
            title=paste0('Scatter plot of detrended data, columns ', acol1, ' and ', acol2,sep='')
        )
    }
}
```

Print and plot (Spearman) correlation matrix of original data (pairwise). Correlation tells us whether two variables move together in the same or opposite direction (correlations of +1 and -1) or are their movement is not related at all (correlation of 0). Correlation is related to covariance in that it is essentially covariance normalised over the product of the standard deviations of the two variables.

```
cnames <- names(dat1_noid)
Ncnames <- length(cnames)
amatr <- matrix(unlist(dat1_noid), ncol=Ncnames, byrow=T)
colnames(amatr) <- cnames
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
cor_matrix <- cor(amatr, method='spearman')
cor_pvalues <- cor.mtest(amatr)
```

the correlation matrix of original columns set:

```
print("correlation matrix of original data:")
```

```
## [1] "correlation matrix of original data:"
```

```
print(cor_matrix)
```

```
##           A         B         C         D         E         F
## A 1.0000000 0.9953697 0.9926465 0.9903925 0.9882165 0.9860988
## B 0.9953697 1.0000000 0.9952715 0.9925822 0.9904287 0.9881424
## C 0.9926465 0.9952715 1.0000000 0.9955148 0.9928157 0.9904968
## D 0.9903925 0.9925822 0.9955148 1.0000000 0.9952246 0.9925382
## E 0.9882165 0.9904287 0.9928157 0.9952246 1.0000000 0.9953121
## F 0.9860988 0.9881424 0.9904968 0.9925382 0.9953121 1.0000000
```

statistical significance of the correlation matrix

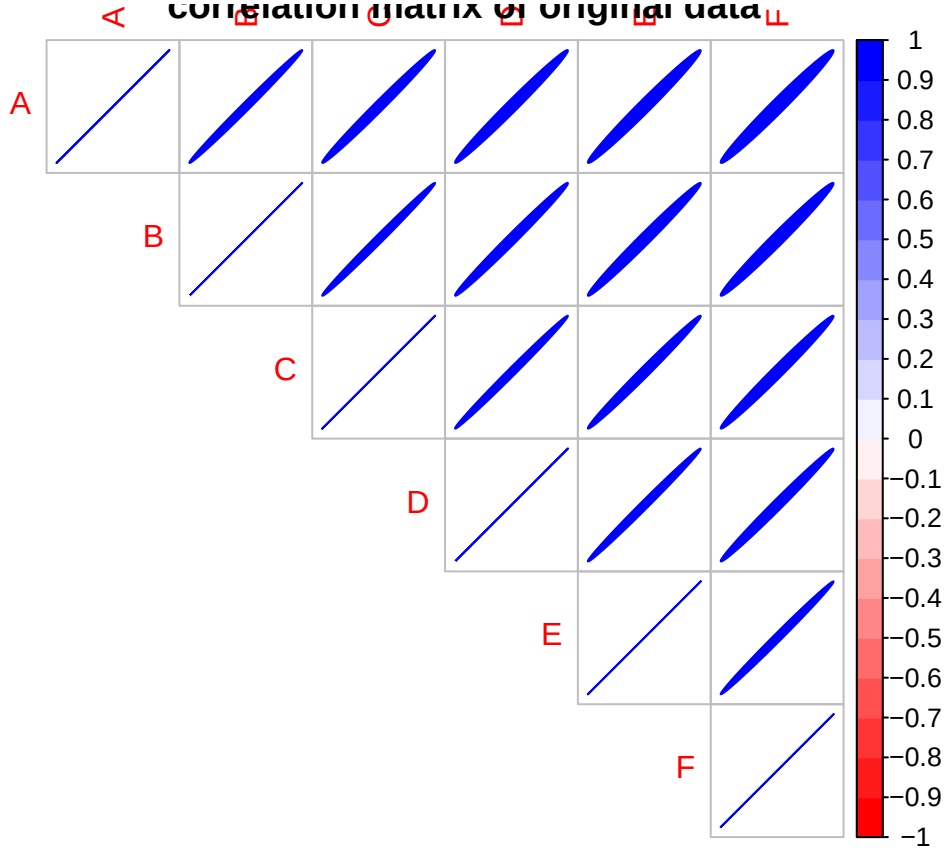
```
print("and its statistical significance:")
```

```
## [1] "and its statistical significance:"
```

```
colnames(cor_pvalues$p) <- cnames
rownames(cor_pvalues$p) <- cnames
print(cor_pvalues$p)
```

```
##   A B C D E F
## A 0 0 0 0 0 0
## B 0 0 0 0 0 0
## C 0 0 0 0 0 0
## D 0 0 0 0 0 0
## E 0 0 0 0 0 0
## F 0 0 0 0 0 0
```

```
acolor <- colorRampPalette(c("red", "white", "blue"))(20)
corrplot(
    cor_matrix,
    p.mat=cor_pvalues$p,
    method="ellipse",
    order="original",
    col=acolor,
    type='upper',
    sig.level=0.0,
    insig="p-value",
    title='correlation matrix of original data'
)
```



plot correlation matrix of detrended data (pairwise)

```
cnames <- names(dat1_detrended)
Ncnames <- length(cnames)
amatr <- matrix(unlist(dat1_detrended), ncol=Ncnames, byrow=T)
colnames(amatr) <- cnames
library(corrplot)
cor_matrix <- cor(amatr, method='spearman')
cor_pvalues <- cor.mtest(amatr)
```

the correlation matrix of detrended columns set:

```
print("correlation matrix of original data:")
```

```
## [1] "correlation matrix of original data:"
```

```
print(cor_matrix)
```

```
##            A          B          C          D          E          F
## A 1.00000000 0.01846806 0.08630014 0.10118207 0.07398528 0.06682168
## B 0.01846806 1.00000000 0.01899633 0.09033755 0.10640717 0.08220713
## C 0.08630014 0.01899633 1.00000000 0.02234015 0.09594644 0.10275620
## D 0.10118207 0.09033755 0.02234015 1.00000000 0.02221574 0.08849447
## E 0.07398528 0.10640717 0.09594644 0.02221574 1.00000000 0.01224202
## F 0.06682168 0.08220713 0.10275620 0.08849447 0.01224202 1.00000000
```

statistical significance of the correlation matrix

```
colnames(cor_pvalues$p) <- cnames
rownames(cor_pvalues$p) <- cnames
```

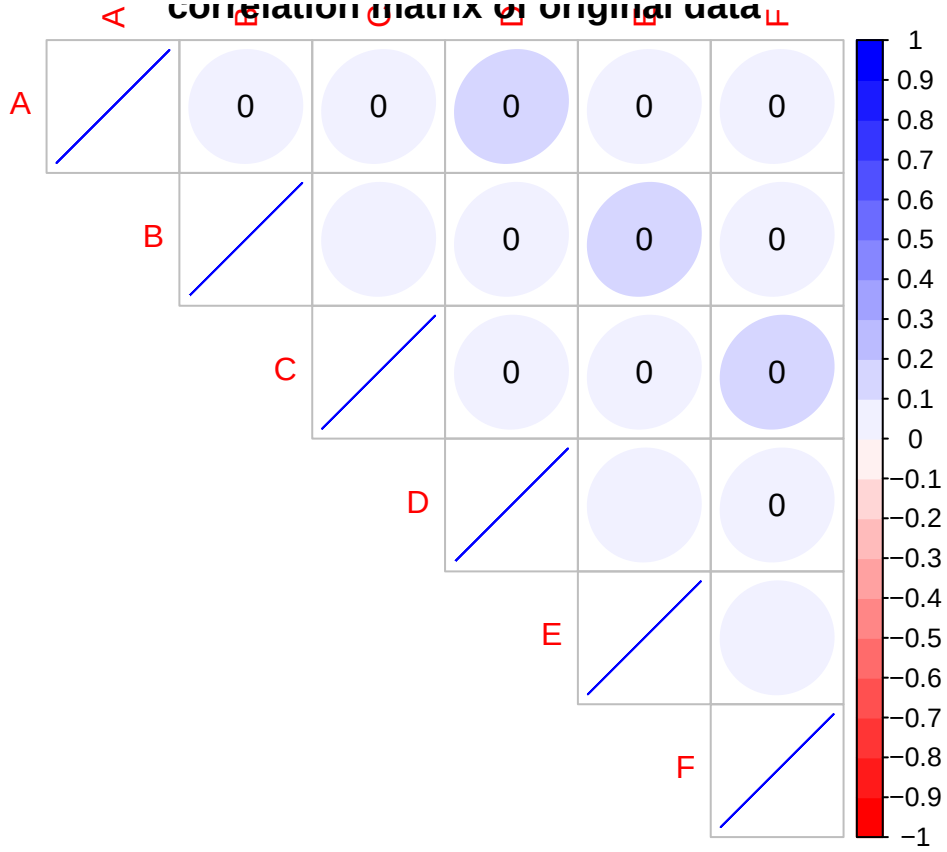
```r
print("and its statistical significance:")
```

```
## [1] "and its statistical significance:"
```

```r
print(cor_pvalues$p)
```

```
##                A             B             C             D             E
## A  0.000000e+00 1.662902e-283  4.154338e-79  1.760060e-34  6.462174e-94
## B 1.662902e-283  0.000000e+00  0.000000e+00 3.566732e-226  6.802040e-04
## C  4.154338e-79  0.000000e+00  0.000000e+00 4.933289e-299 1.153638e-198
## D  1.760060e-34 3.566732e-226 4.933289e-299  0.000000e+00  0.000000e+00
## E  6.462174e-94  6.802040e-04 1.153638e-198  0.000000e+00  0.000000e+00
## F  7.410103e-09  1.417765e-45  5.378597e-38  1.399490e-54  0.000000e+00
##              F
## A 7.410103e-09
## B 1.417765e-45
## C 5.378597e-38
## D 1.399490e-54
## E 0.000000e+00
## F 0.000000e+00
```

```r
acolor <- colorRampPalette(c("red", "white", "blue"))(20)
corrplot(
    cor_matrix,
    p.mat=cor_pvalues$p,
    method="ellipse",
    order="original",
    col=acolor,
    type='upper',
    sig.level=0.0,
    insig="p-value",
    title='correlation matrix of original data'
)
```



Correlation summary of original data:

```
library(PerformanceAnalytics)
```

```
## Loading required package: xts
## Loading required package: zoo
##
## Attaching package: 'zoo'
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
## Attaching package: 'PerformanceAnalytics'
## The following object is masked from 'package:graphics':
##
##     legend
```

```
chart.Correlation(dat1_noid, histogram=TRUE, pch=19)
```

```
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
```

```
## Warning in title(...): "method" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "method" is
## not a graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "method" is
## not a graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "method" is
## not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "method" is
## not a graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "method" is
## not a graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter
```

```
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "method" is
## not a graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "method" is
## not a graphical parameter
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
```

```
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
```

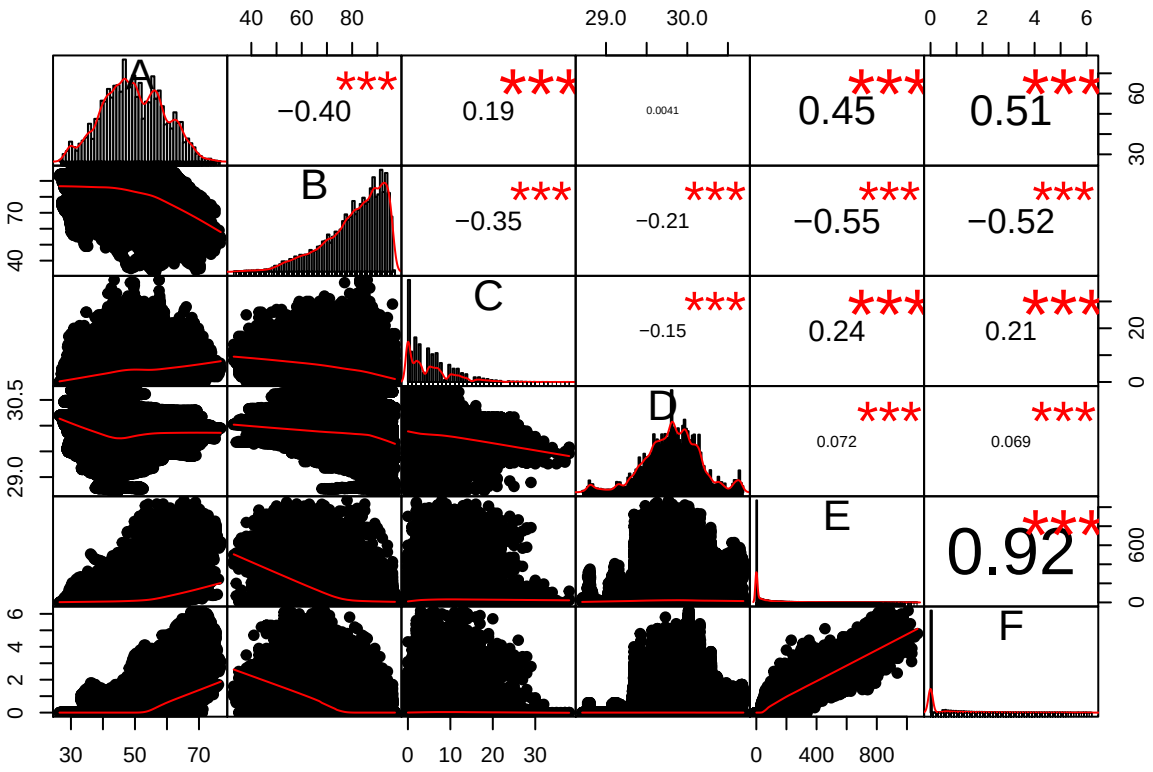
```
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "method" is
## not a graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "method" is
## not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "method" is
## not a graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "method" is
## not a graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "method" is
## not a graphical parameter
```

```
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter
```

```
## Warning in plot.window(...): "method" is not a graphical parameter
```

```
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
```

```
## Warning in title(...): "method" is not a graphical parameter
```



Correlation summary of detrended data:

```
library(PerformanceAnalytics)
chart.Correlation(dat1_noid, histogram=TRUE, pch=19)
```

```
## Warning in plot.window(...): "method" is not a graphical parameter
```

```
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
```

```
## Warning in title(...): "method" is not a graphical parameter
```

```
## Warning in plot.window(...): "method" is not a graphical parameter
```

```
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
```

```
## Warning in title(...): "method" is not a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "method" is
## not a graphical parameter
```

```
## Warning in plot.window(...): "method" is not a graphical parameter
```

```
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
```

```
## Warning in title(...): "method" is not a graphical parameter
```

```
## Warning in plot.window(...): "method" is not a graphical parameter
```

```
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
```

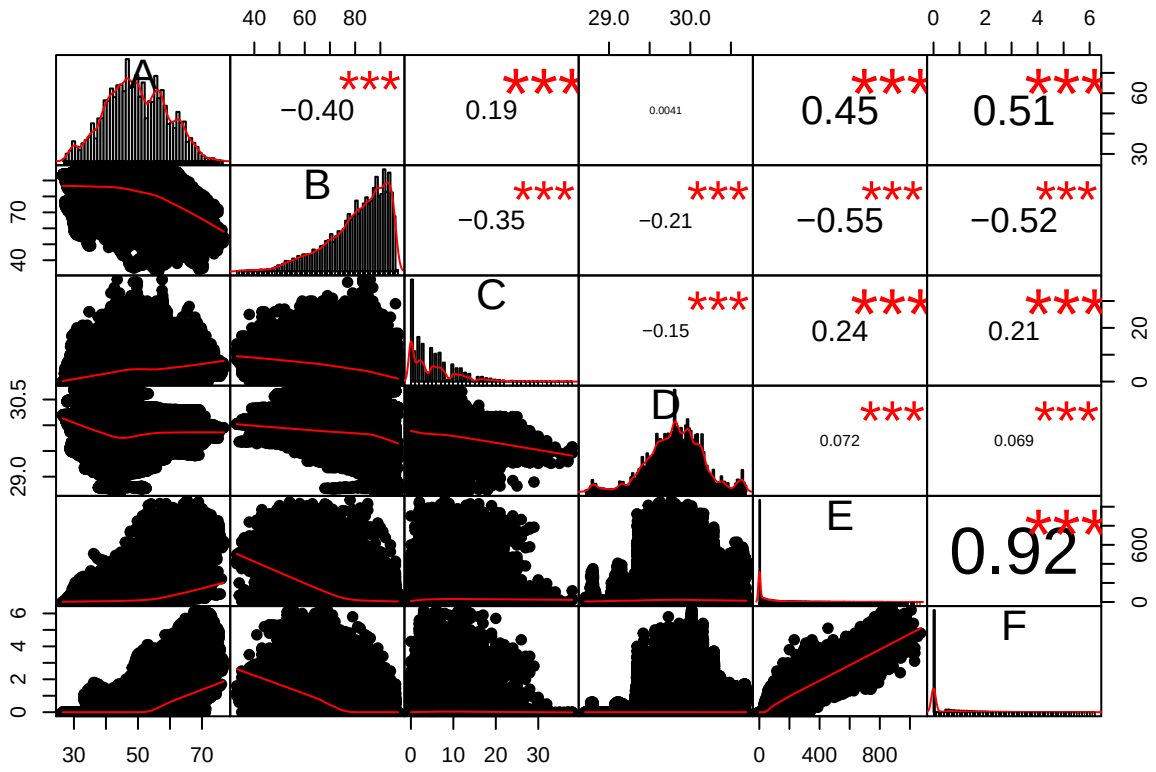
```
## Warning in title(...): "method" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "method" is
## not a graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "method" is
## not a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "method" is
## not a graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "method" is
## not a graphical parameter
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
```

```
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "method" is
## not a graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "method" is
## not a graphical parameter
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter
## Warning in plot.window(...): "method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter
## Warning in title(...): "method" is not a graphical parameter
```

```
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "method" is
## not a graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "method" is
## not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "method" is
## not a graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "method" is
## not a graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "method" is
## not a graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "method" is not a
## graphical parameter

## Warning in plot.window(...): "method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "method" is not a graphical parameter

## Warning in title(...): "method" is not a graphical parameter
```



plot covariance matrix of original data (pairwise)

```
cnames <- names(dat1_noid)
Ncnames <- length(cnames)
amatr <- matrix(unlist(dat1_noid), ncol=Ncnames, byrow=T)
colnames(amatr) <- cnames
cov_matrix <- cov(amatr)
```

the covariance matrix of original columns set:

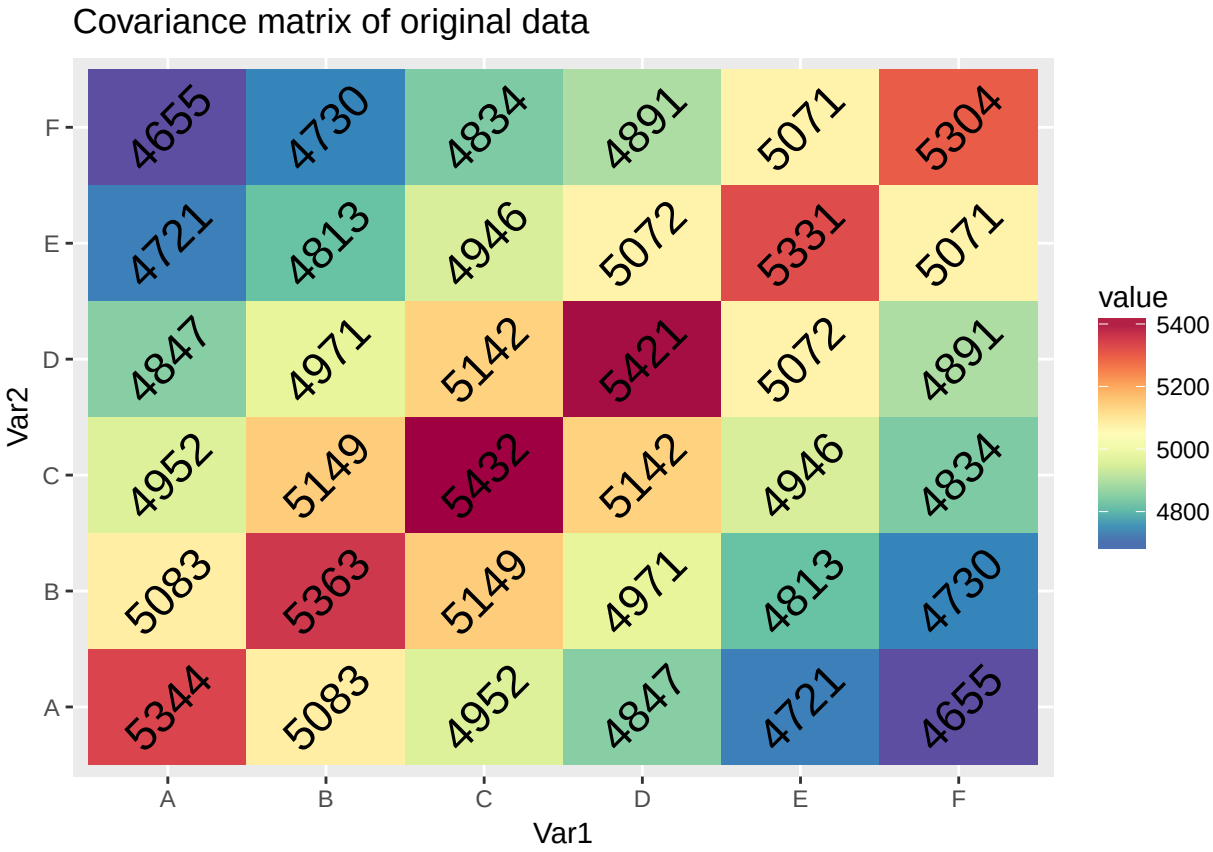
```
print("covariance matrix of original data:")
```

```
## [1] "covariance matrix of original data:"
```

```
print(cov_matrix)
```

```
##          A        B        C        D        E        F
## A 5344.077 5082.750 4952.304 4846.738 4720.588 4654.519
## B 5082.750 5362.658 5149.259 4971.203 4812.842 4729.721
## C 4952.304 5149.259 5432.363 5141.714 4946.124 4834.244
## D 4846.738 4971.203 5141.714 5421.468 5071.708 4890.835
## E 4720.588 4812.842 4946.124 5071.708 5331.006 5071.083
## F 4654.519 4729.721 4834.244 4890.835 5071.083 5304.439
```

```
p <- ggplot_heatmap(
    inp=cov_matrix,
    plot.title='Covariance matrix of original data'
)
print(p)
```



plot covariance matrix of detrended data (pairwise) Covariance is similar to variance. The latter applies to a single variable Covariance applies to two variables and is the sum of the product of each data-point's deviation from the mean

$$\sum_{i=1}^{N} \frac{(x_1 - \bar{(}x_1))(x_2 - \bar{(}x_2))}{N-1}$$

```
cnames <- names(dat1_detrended)
Ncnames <- length(cnames)
amatr <- matrix(unlist(dat1_detrended), ncol=Ncnames, byrow=T)
colnames(amatr) <- cnames
cov_matrix <- cov(amatr)
```

the covariance matrix of detrended columns set:

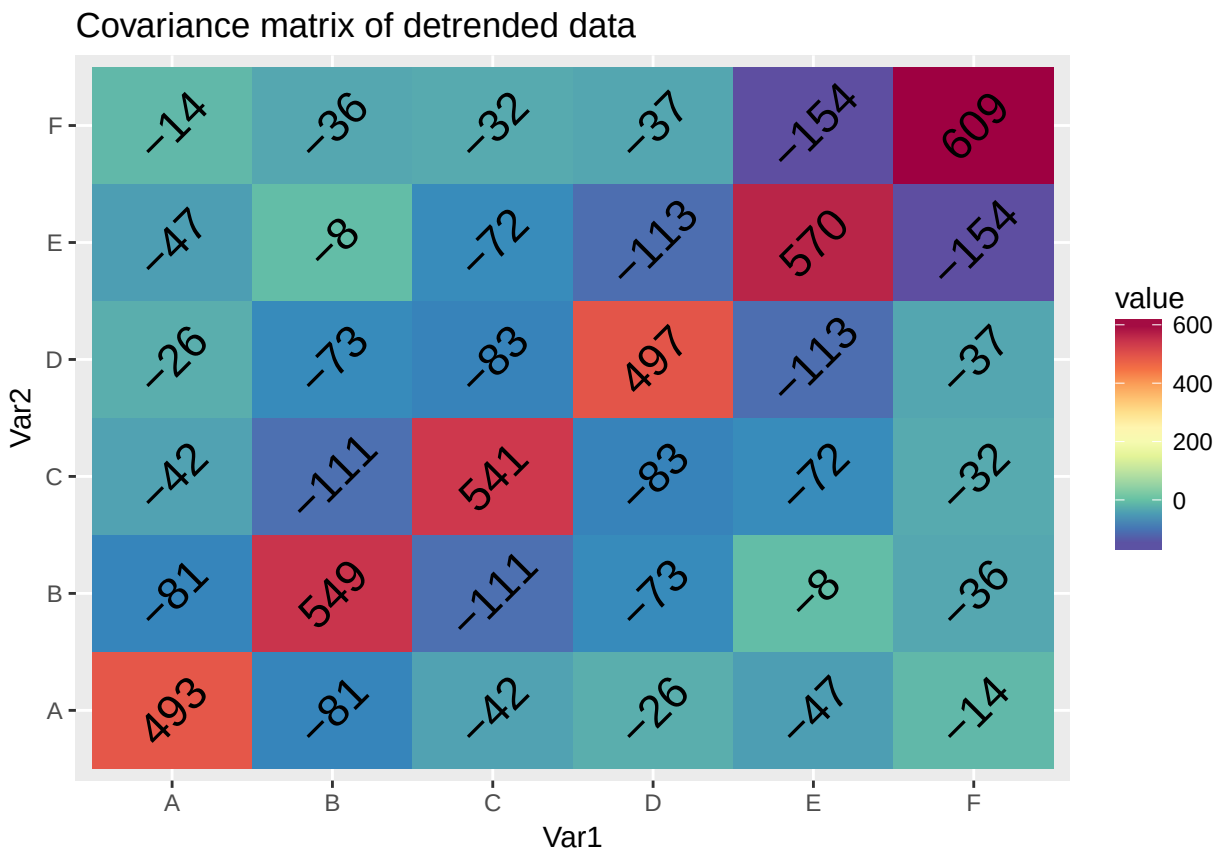
```
print("covariance matrix of detrended data:")
```

```
## [1] "covariance matrix of detrended data:"
```

```
print(cov_matrix)
```

```
##           A           B          C          D           E          F
## A 493.24296  -81.060207  -42.32725  -26.38814  -47.418538  -13.80354
## B -81.06021  548.650378 -111.09997  -72.66176   -8.280808  -35.65198
## C -42.32725 -111.099974  541.27997  -82.95156  -72.493255  -32.21031
## D -26.38814  -72.661759  -82.95156  496.51207 -112.577544  -37.23274
## E -47.41854   -8.280808  -72.49326 -112.57754  570.366279 -154.18776
## F -13.80354  -35.651978  -32.21031  -37.23274 -154.187763  608.91611
```

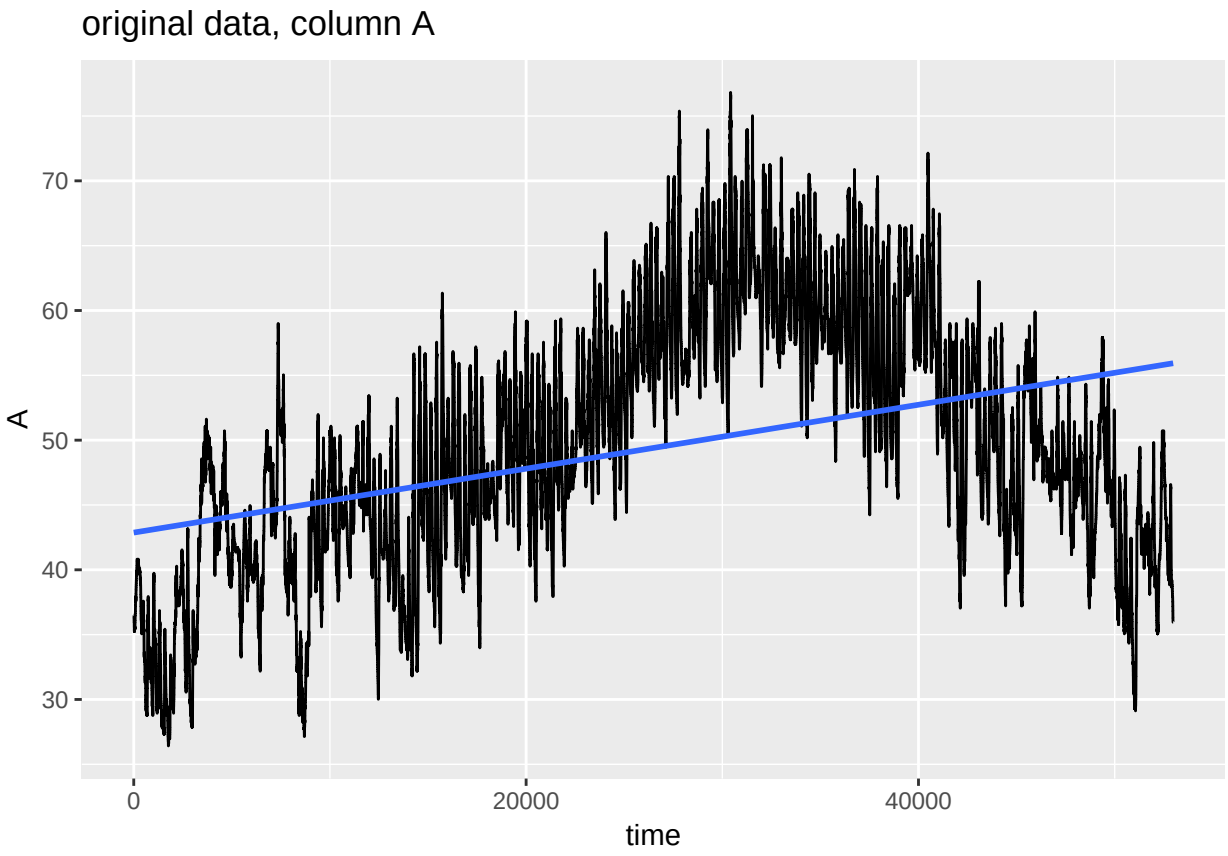
```
p <- ggplot_heatmap(
    inp=cov_matrix,
    plot.title='Covariance matrix of detrended data'
)
print(p)
```



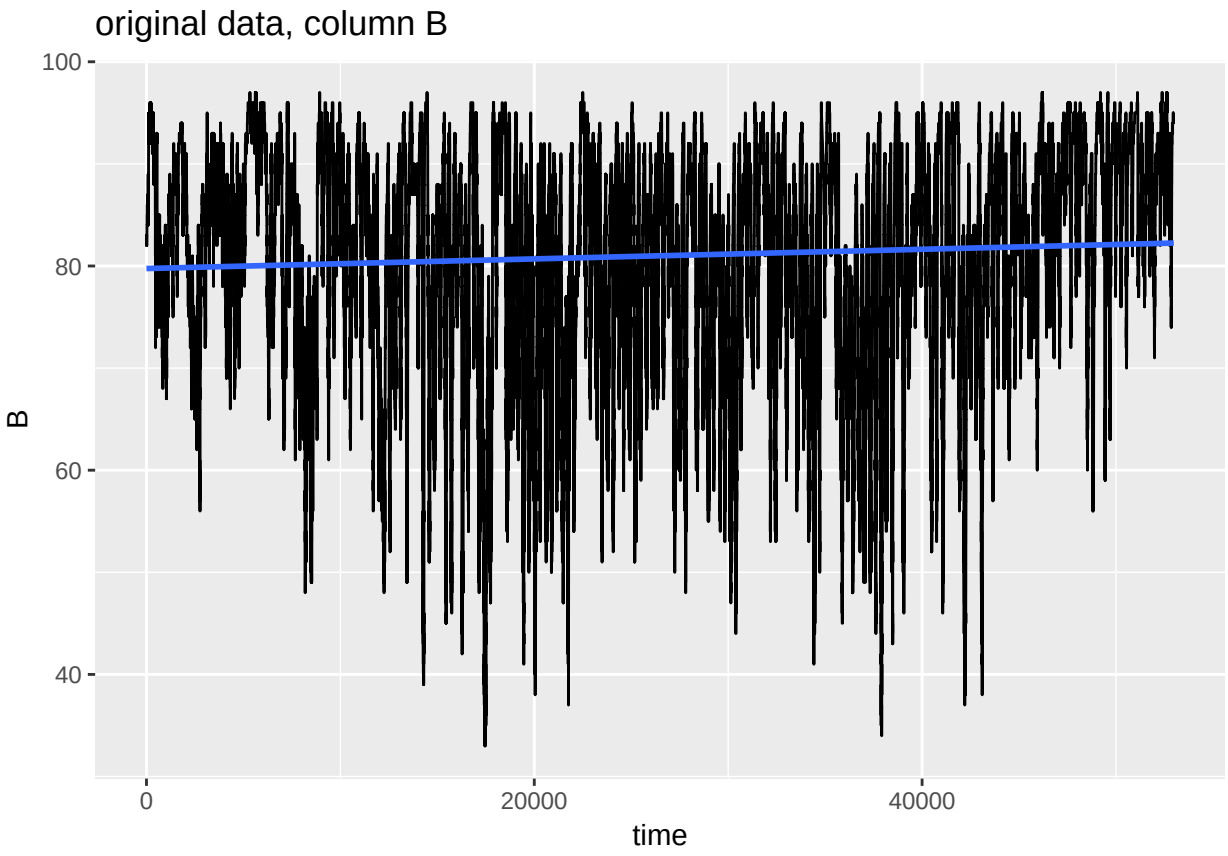
plot the original data

```
library(ggplot2)
library(reshape2)
adat = dat1_noid; adat[['id']] = dat1_id[['id']]
for(acol in names(dat1_noid)){
    atitle <- paste0("original data, column ", acol)
    cat("plot of ", atitle, "\n", sep='')
    print(
        ggplot(adat, aes_string(x='id',y=acol))+
        geom_line()+
        ggtitle(atitle)+
        xlab('time')+
        geom_smooth(method='gam')
    )
}
```

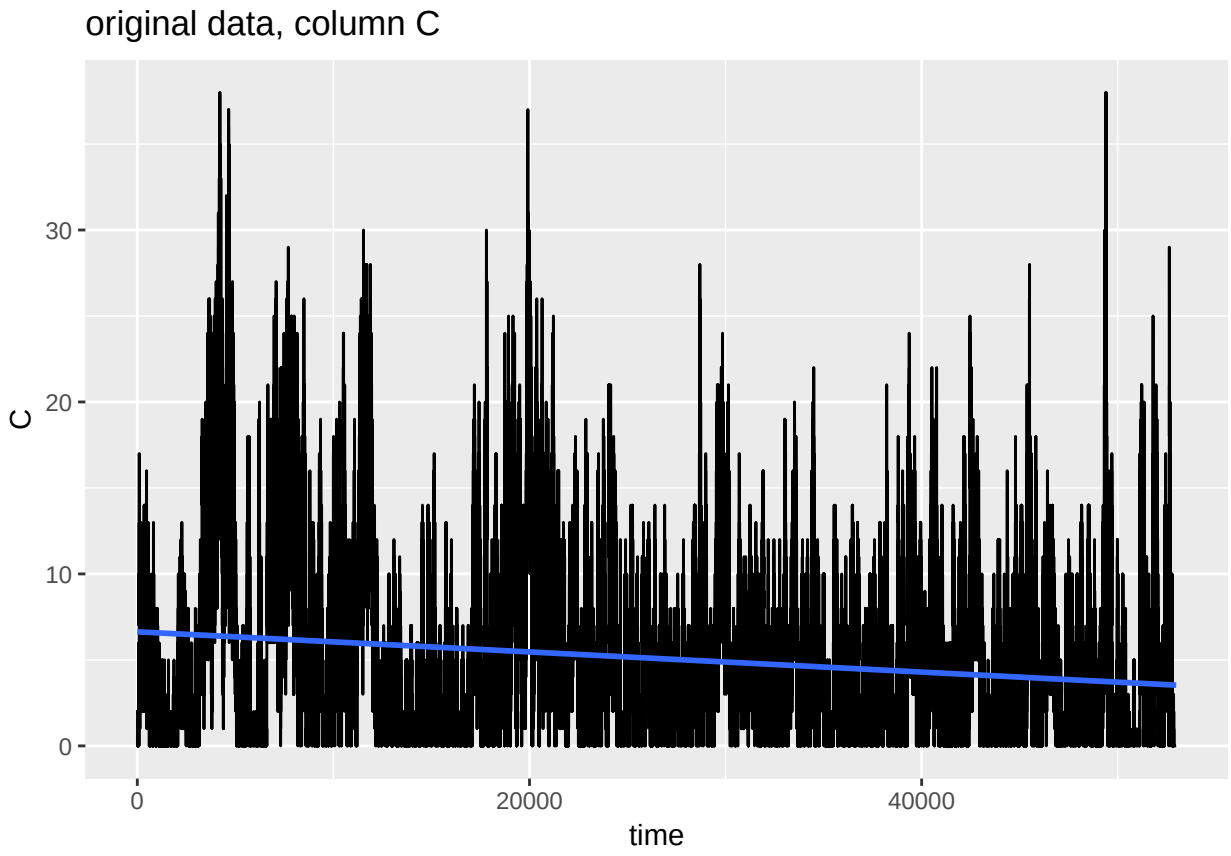
```
## plot of original data, column A
```



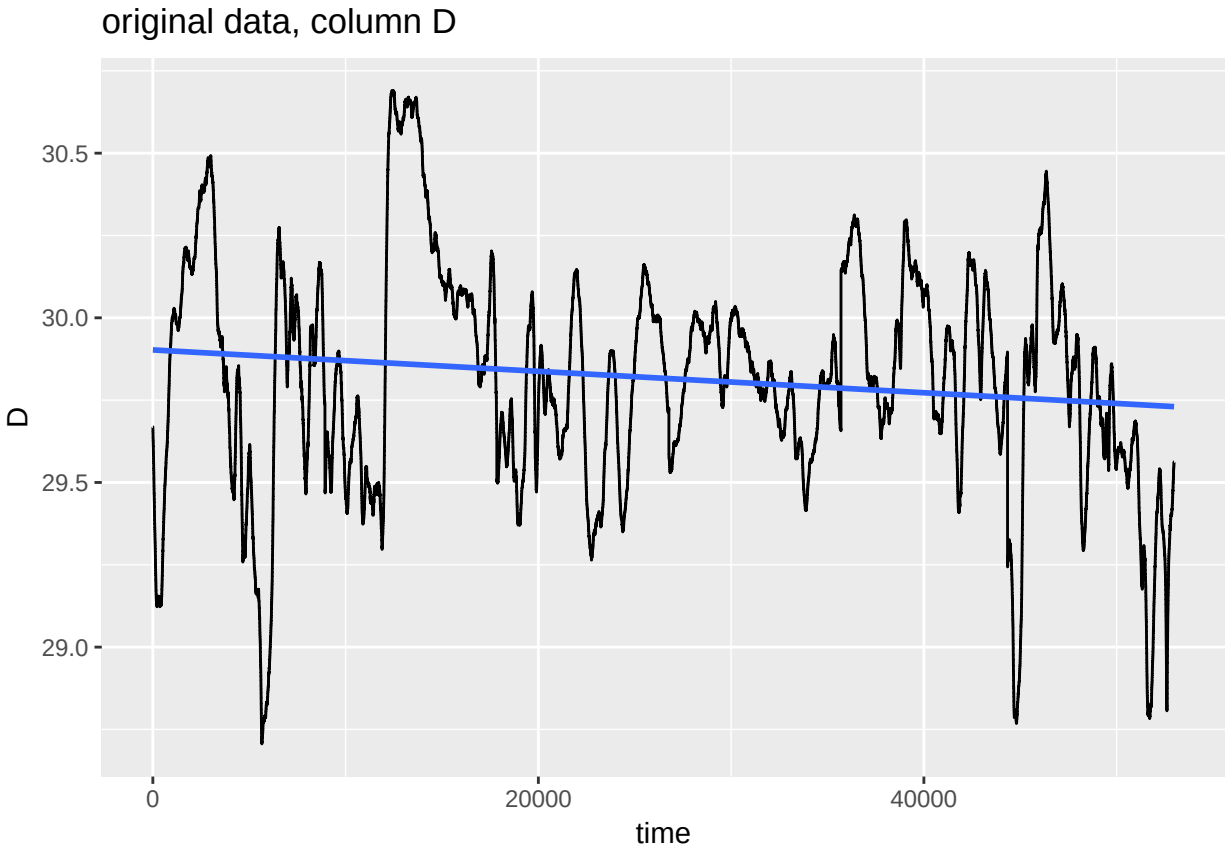

```
## plot of original data, column B
```



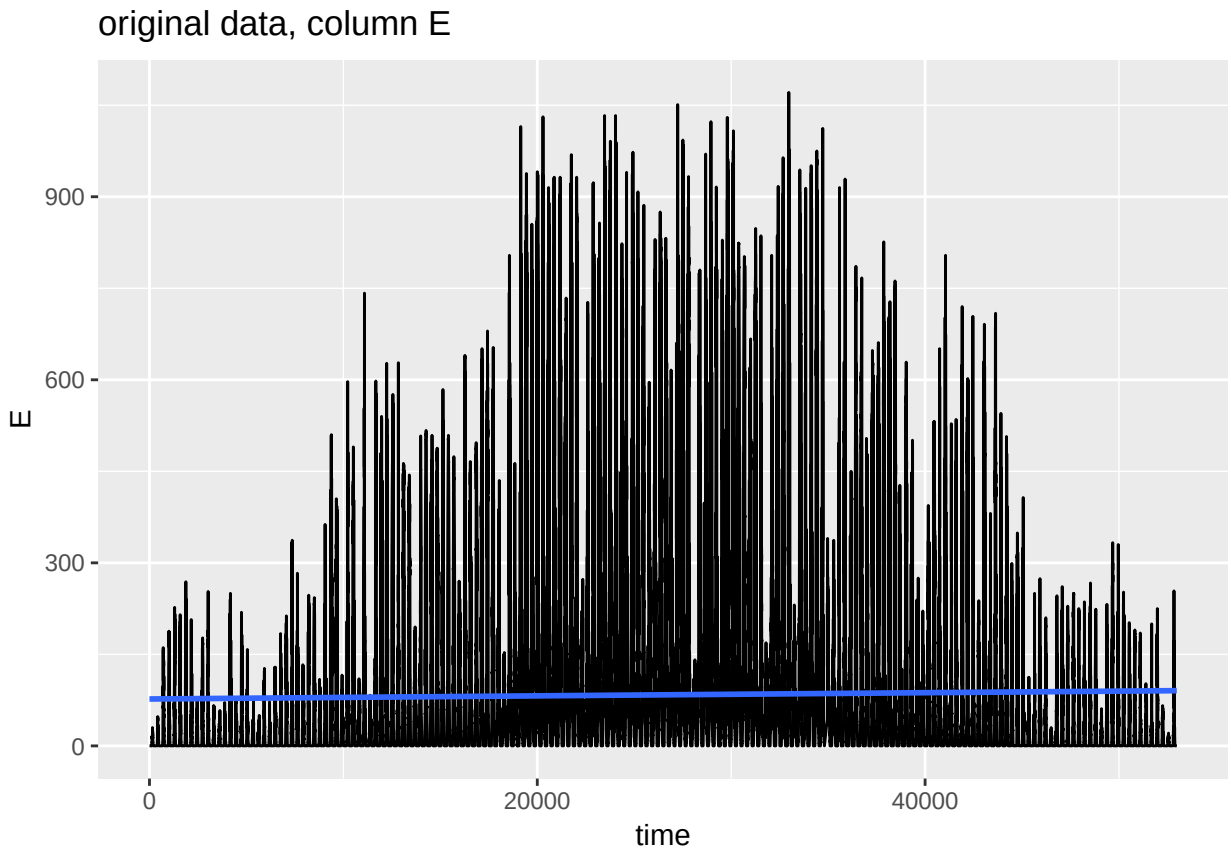

```
## plot of original data, column C
```



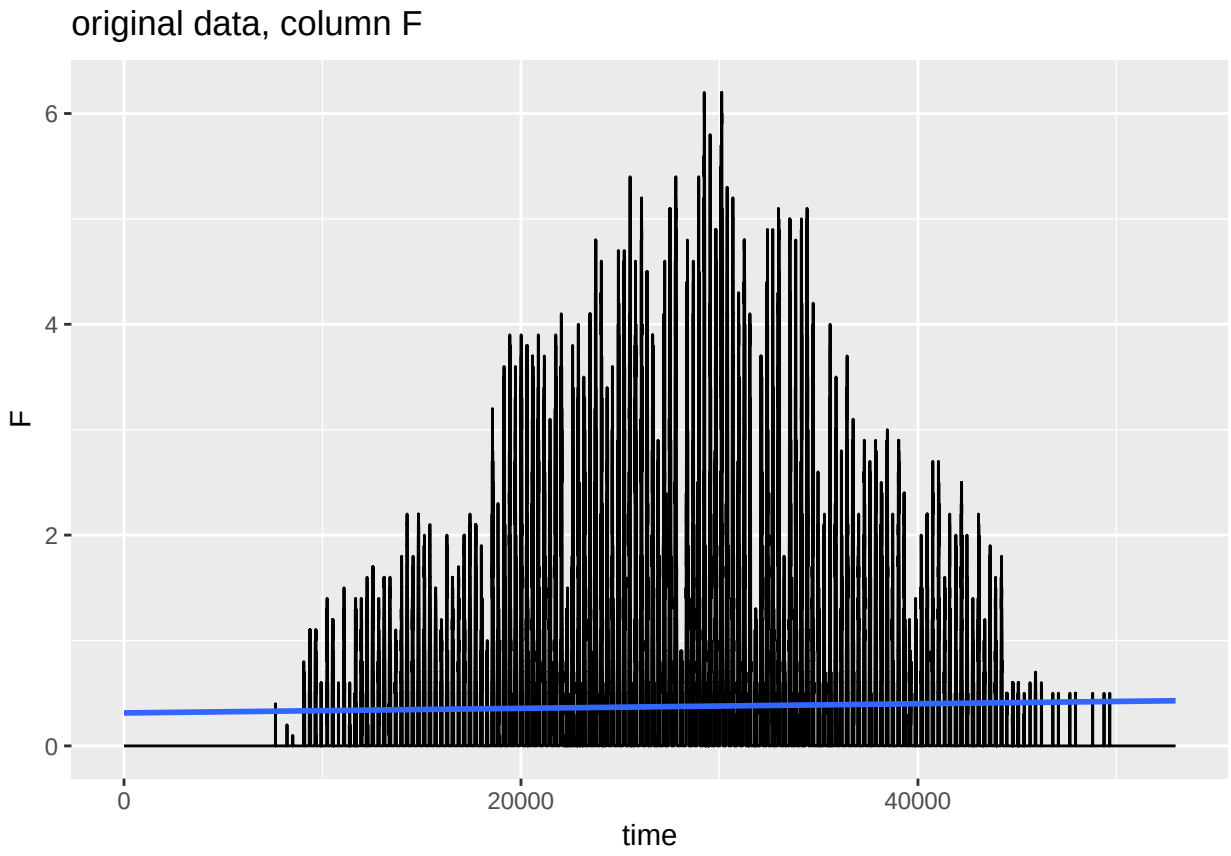

```
## plot of original data, column D
```




```
## plot of original data, column E
```



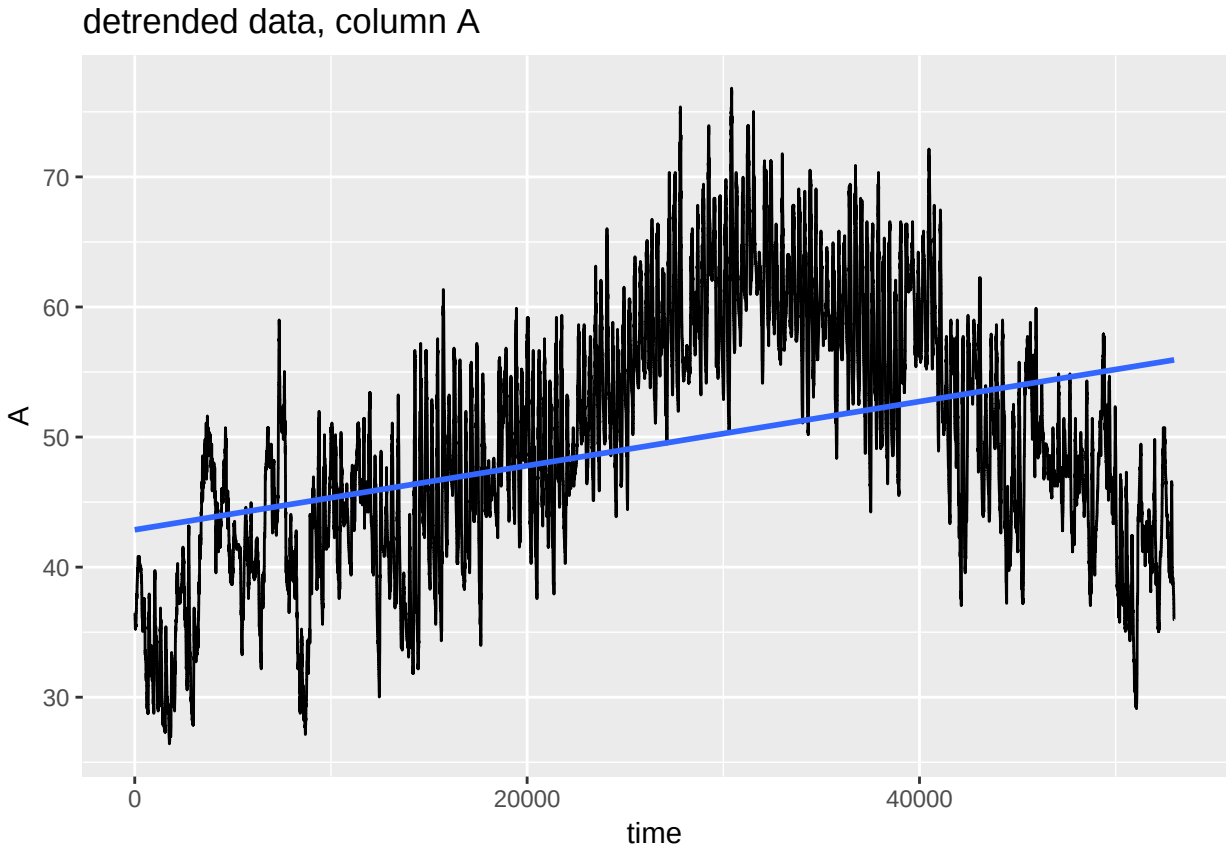

```
## plot of original data, column F
```



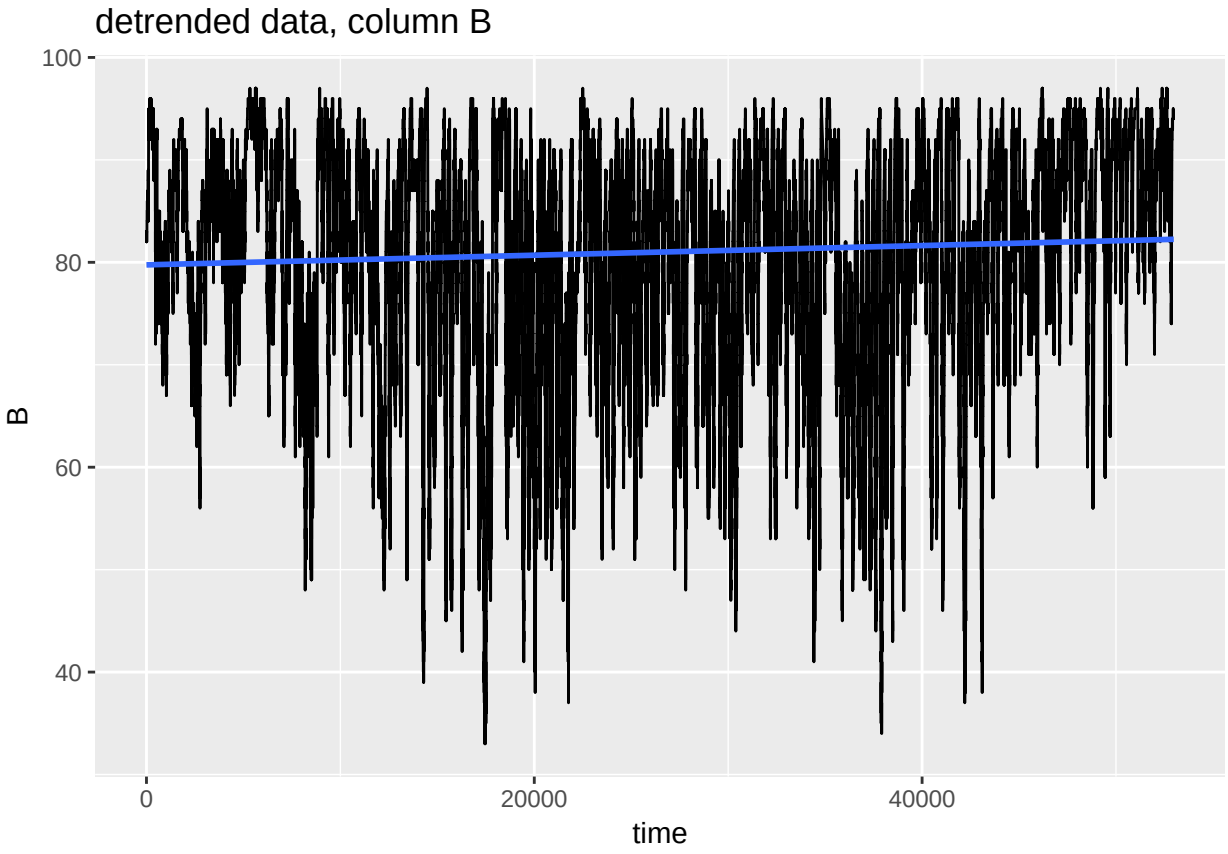
plot detrended data

```
adf <- data.frame(dat1_detrended); adf$id <- dat1_detrended_id[['id']]
for(acol in names(dat1_detrended)){
    atitle <- paste0("detrended data, column ", acol)
    cat("plot of ", atitle, "\n", sep='')
    print(
        ggplot(dat1, aes_string(x='id',y=acol))+
        geom_line()+
        ggtitle(atitle)+
        xlab('time')+
        geom_smooth(method='gam')
    )
}
```

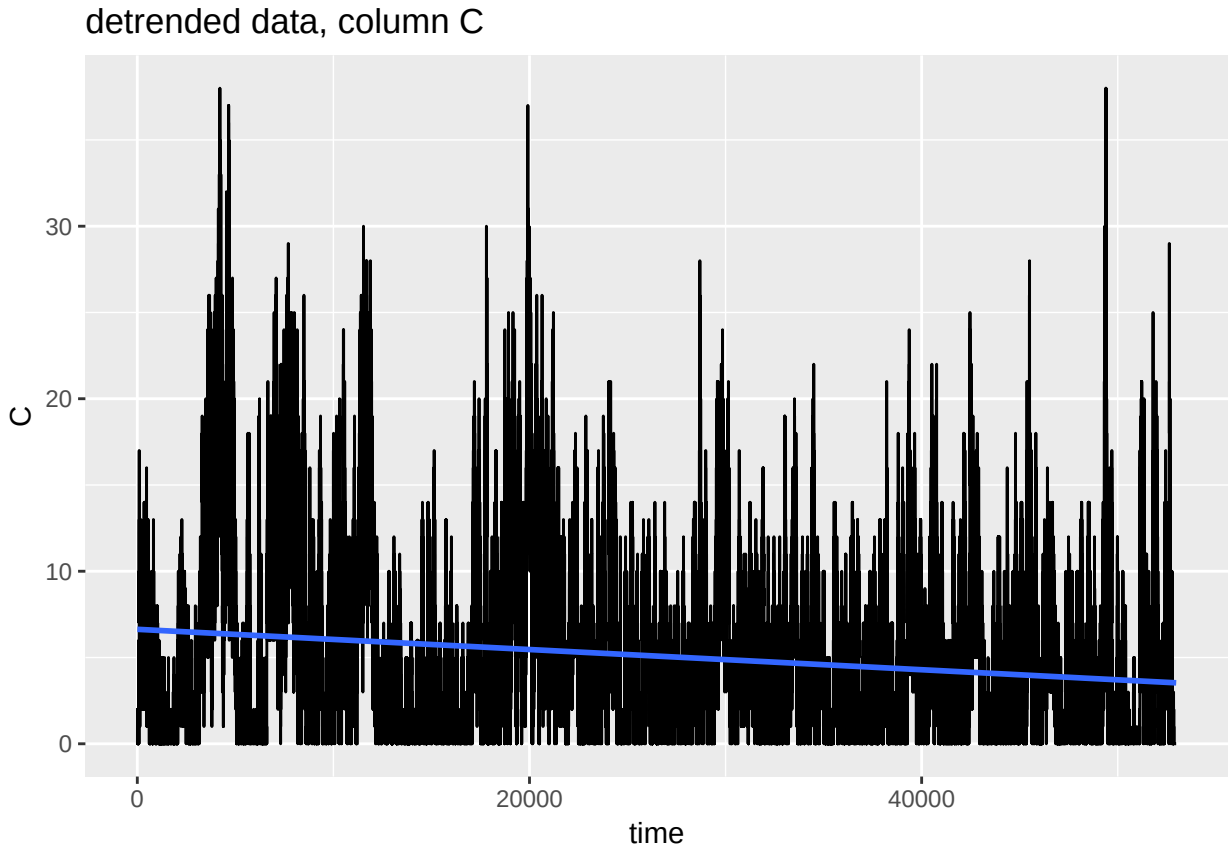
```
## plot of detrended data, column A
```



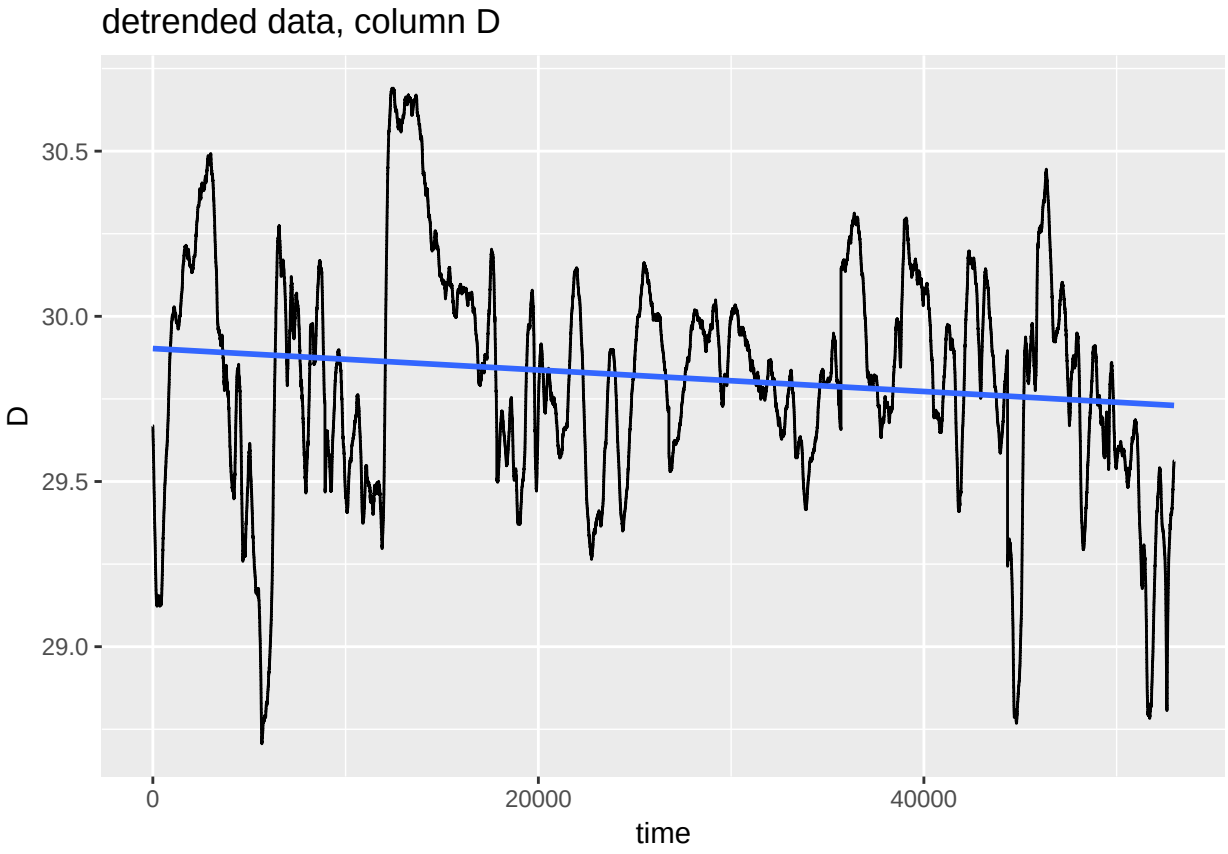

```
## plot of detrended data, column B
```



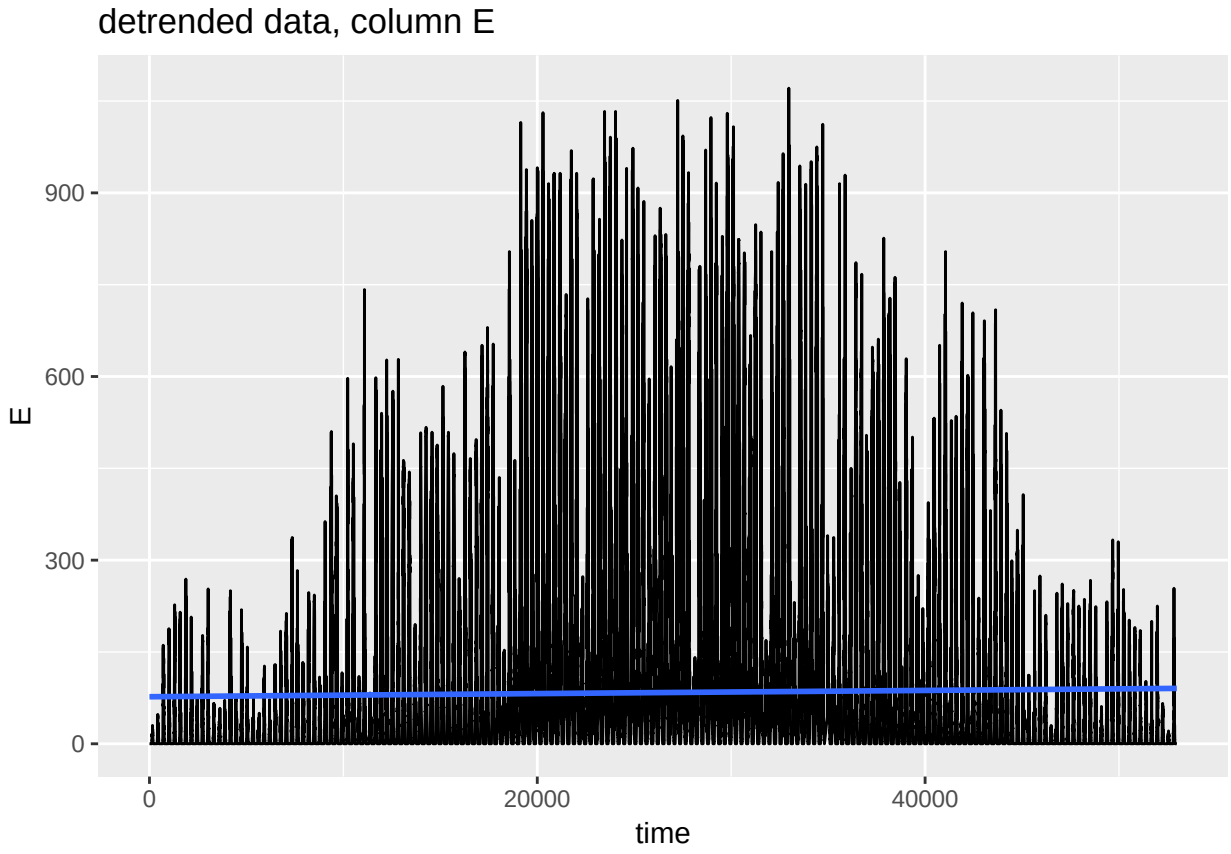

```
## plot of detrended data, column C
```



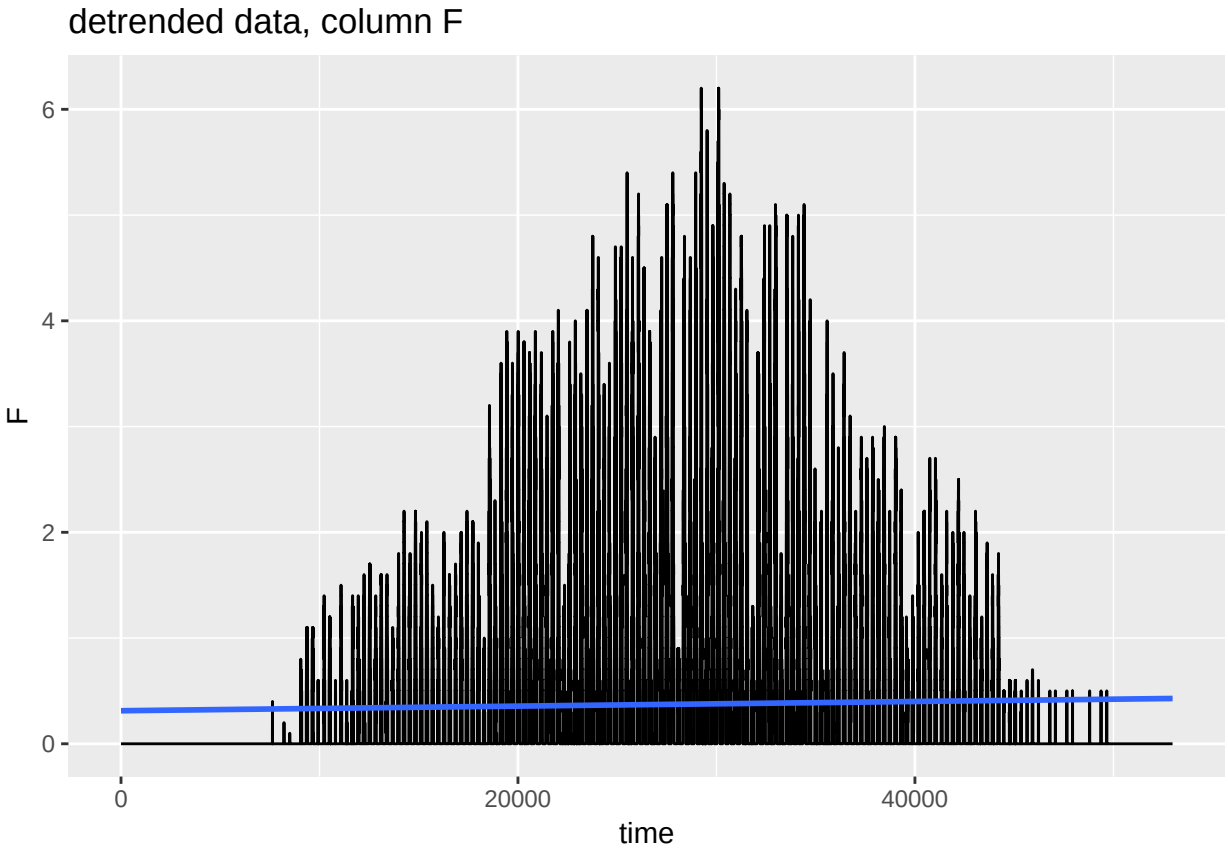

```
## plot of detrended data, column D
```




```
## plot of detrended data, column E
```




```
## plot of detrended data, column F
```



discretise the data to five levels dat1_discrete <- discretise(dat1_noid, levels=5) plot discrete data

```
if( FALSE ){
for(acol in names(dat1_discrete)){
    atitle <- paste0('Original data, discretised to 5 levels')
    adf <- data.frame(data=dat1_discrete[[acol]]$discretised)
    adf$id <- seq.int(nrow(adf))
    p <- ggplot(
        adf,
        aes_string(x='id',y='data')
    )+geom_line()+ggtitle(atitle)
    print(p)
}
}
```

discretise the detrended data dat1_detrended_discrete <- discretise(dat1_detrended, levels=5) plot detrended discrete data

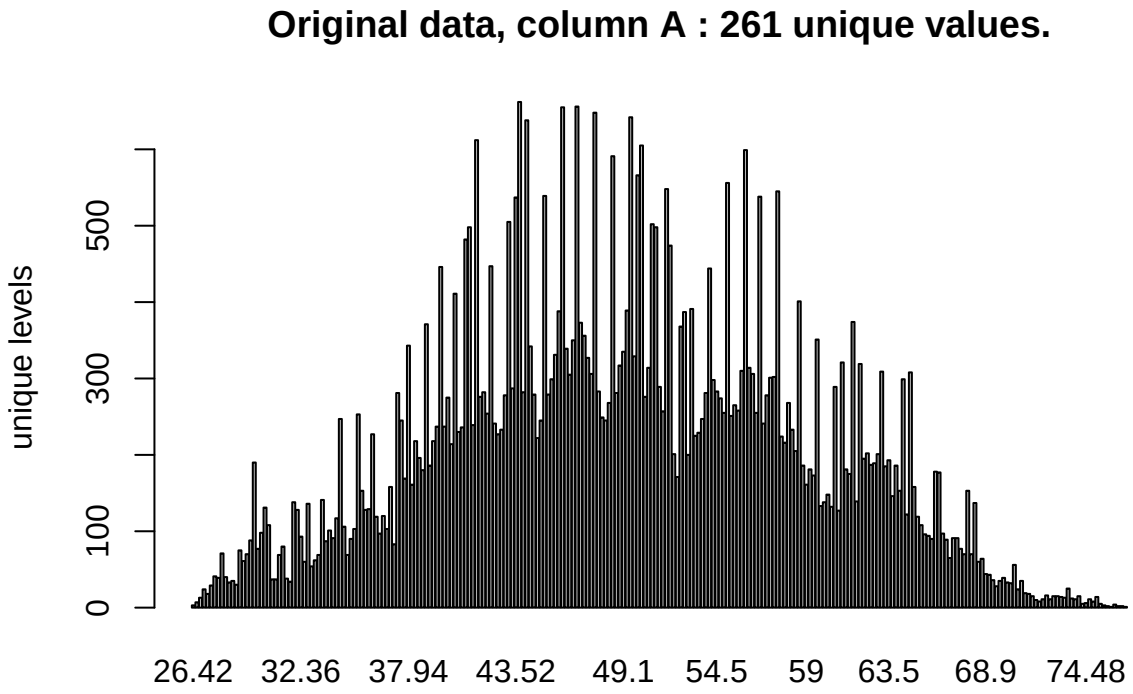
```
if( FALSE ){
for(acol in names(dat1_detrended_discrete)){
    atitle <- paste0('Detrended data, discretised to 5 levels')
    adf <- data.frame(data=dat1_detrended_discrete[[acol]]$discretised); adf$id <- seq.int(nrow(adf))
    p <- ggplot(
        adf,
        aes_string(x='id',y='data')
    )+geom_line()+ggtitle(atitle)
    print(p)
}
```

```
} # if( FALSE )
```

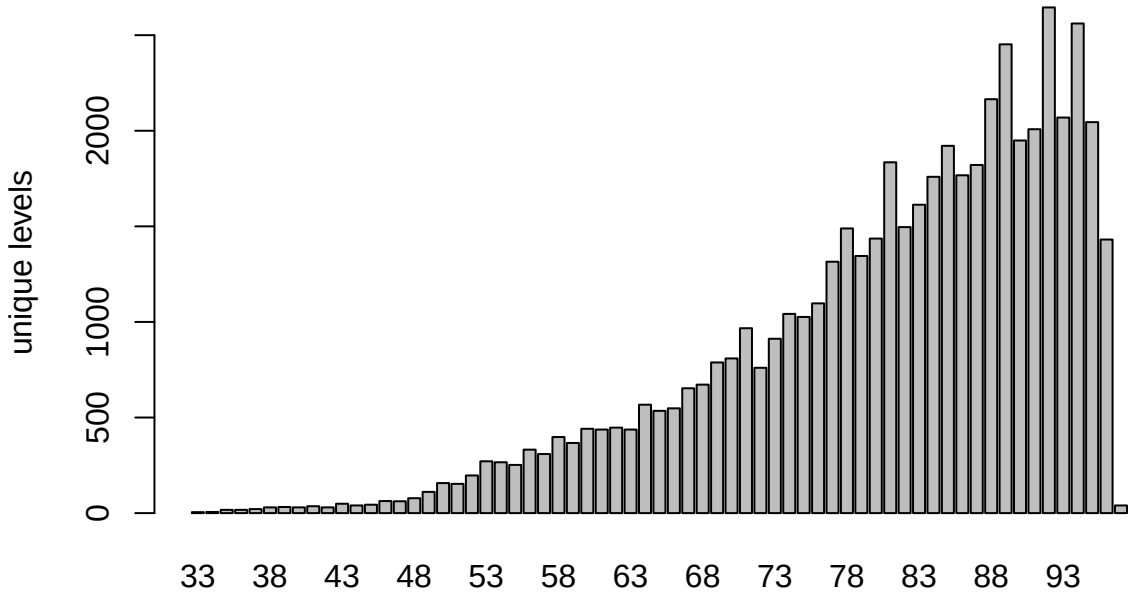
find unique values

```
uniq_vals <- unique_values(inp=dat1_noid)
histo_dat1 <- histo(inp=dat1_noid,unique_vals=uniq_vals)
for(acol in names(dat1_noid)){
    cat("found ", uniq_vals$num_unique_values[1,acol], " unique values in the original data, column ", a
    barplot(
        table(dat1_noid[[acol]]),
        main=paste0('Original data, column ', acol, ' : ', uniq_vals$num_unique_values[1,acol], ' unique
        ylab='unique levels'
    )
}
```

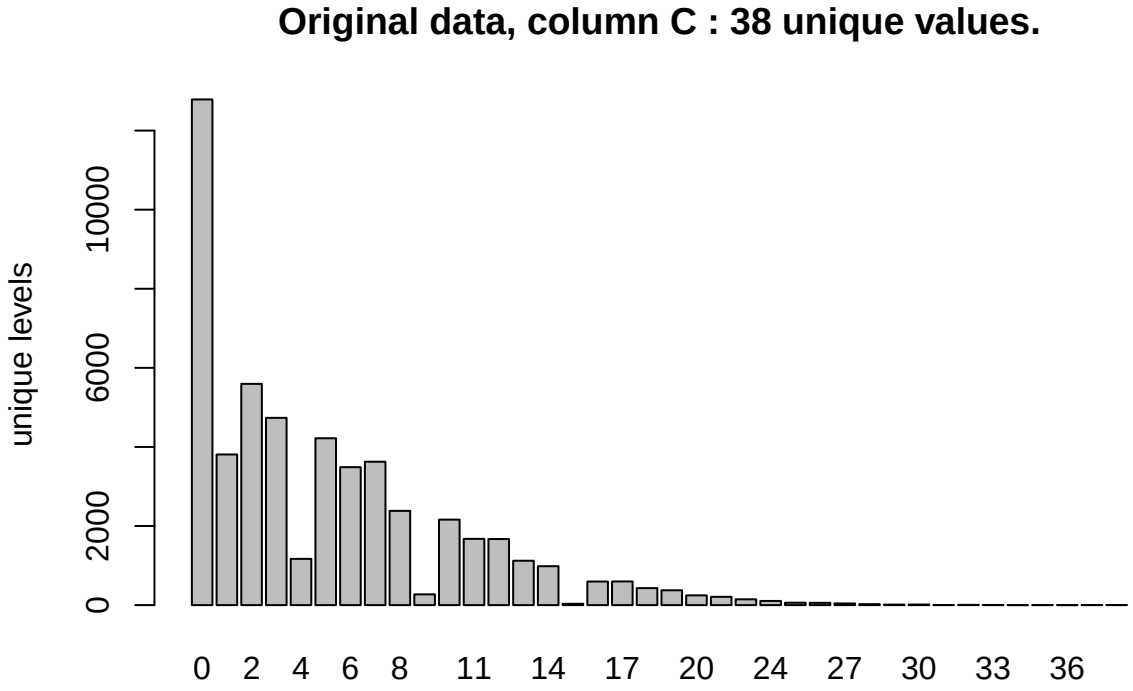
```
## found 261 unique values in the original data, column A
```



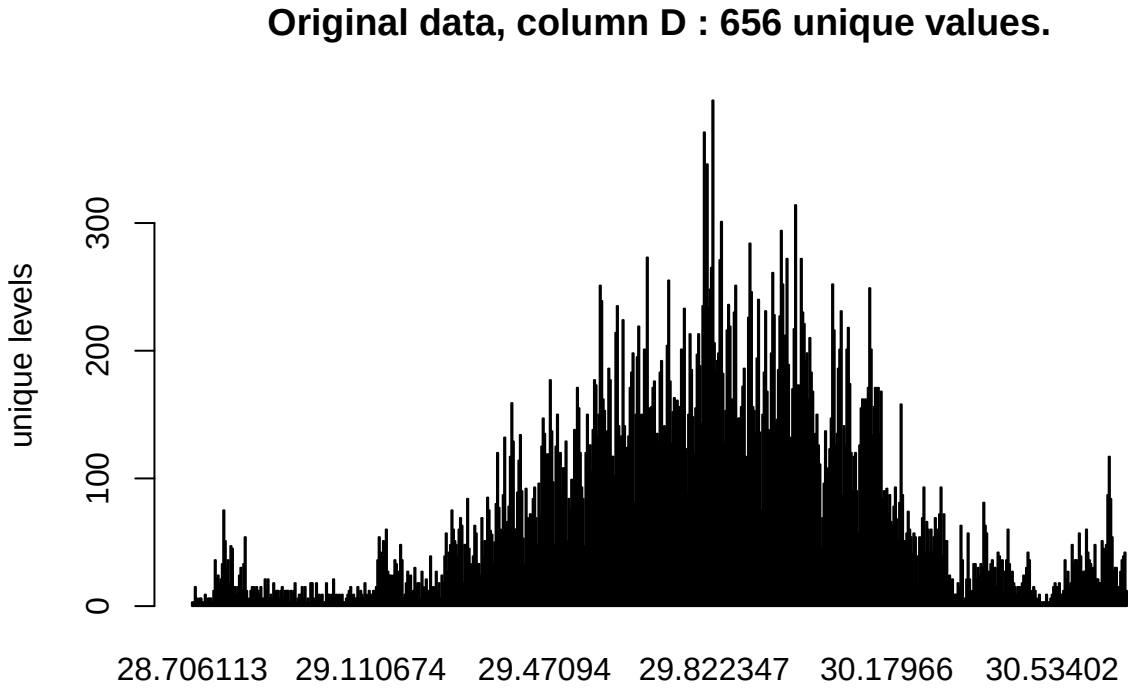

```
## found 65 unique values in the original data, column B
```



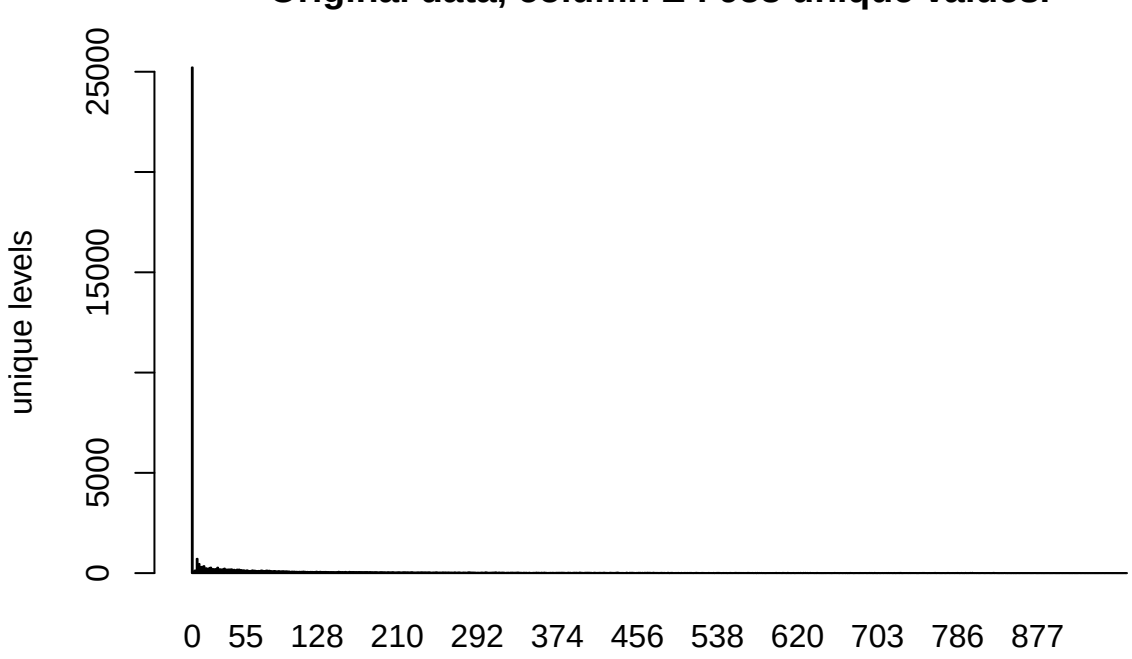

```
## found 38 unique values in the original data, column C
```



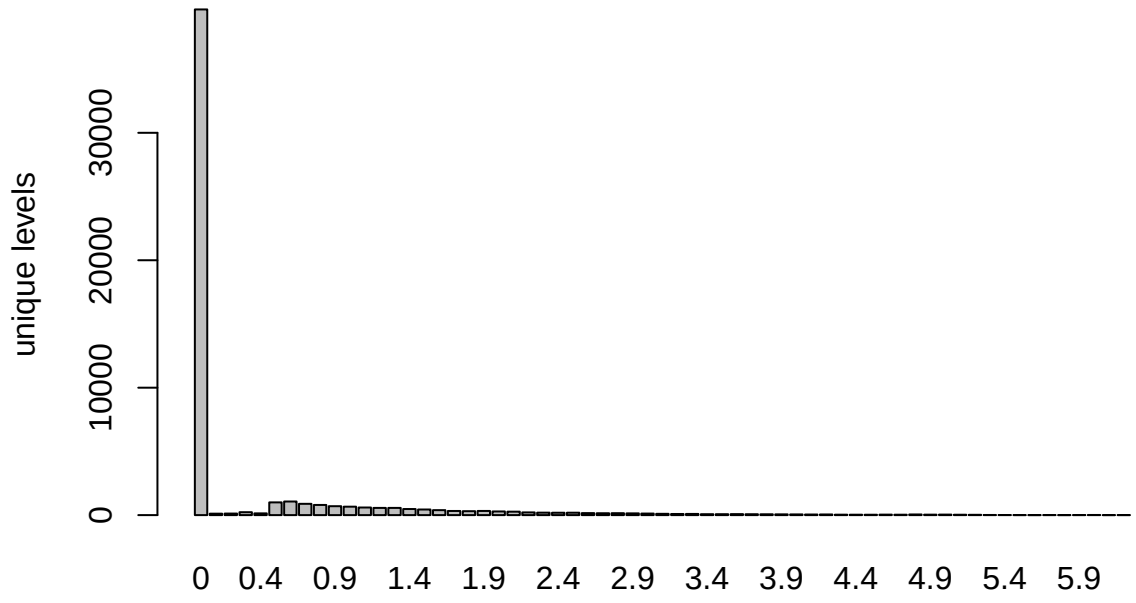

```
## found 656 unique values in the original data, column D
```




```
## found 958 unique values in the original data, column E
```




```
## found 63 unique values in the original data, column F
```

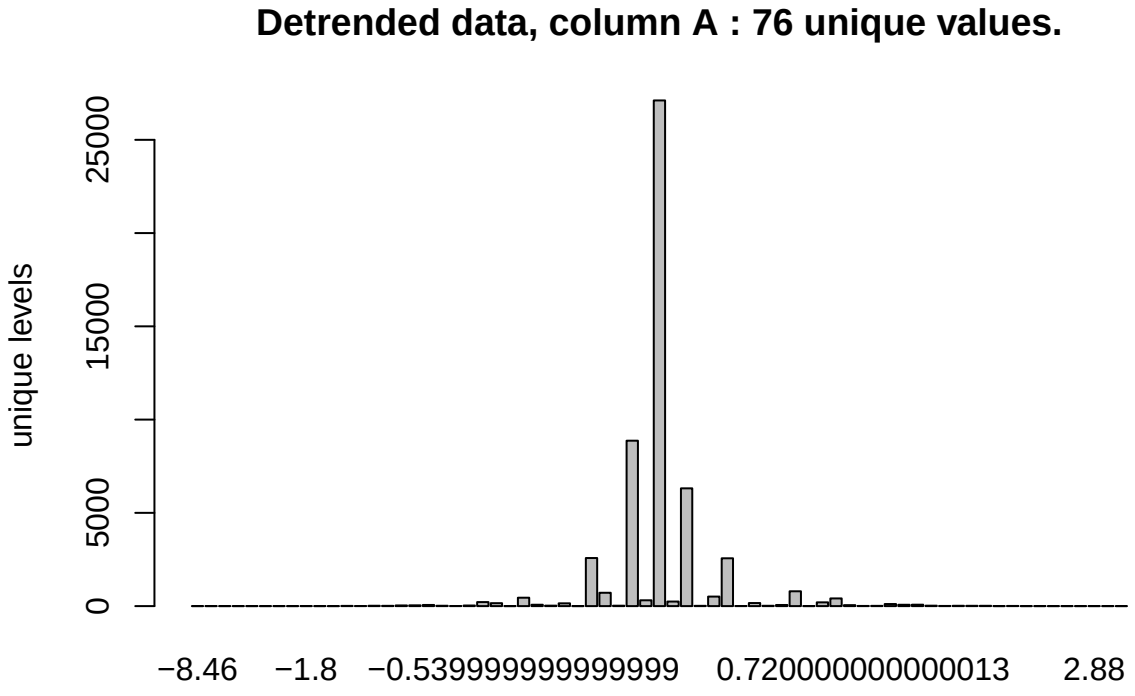



```
#print(histo_dat1)
```

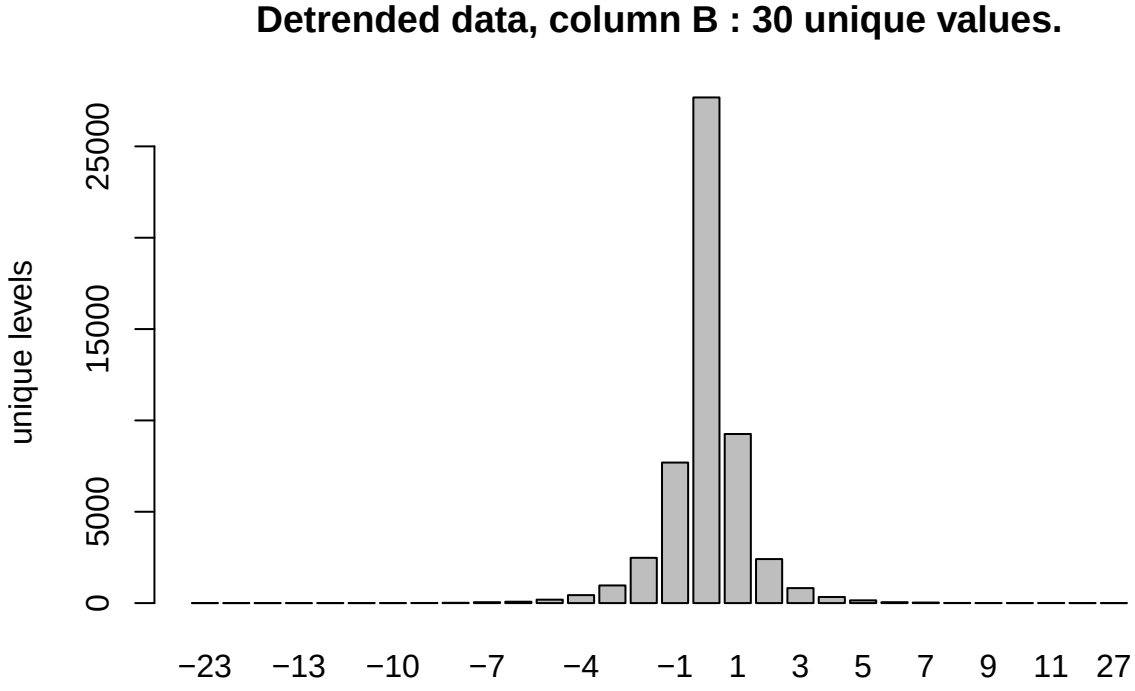
find unique values in the detrended data

```
uniq_vals_detrended <- unique_values(inp=dat1_detrended)
histo_dat1_detrended <- histo(inp=dat1_detrended,unique_vals=uniq_vals_detrended)
for(acol in names(dat1_detrended)){
    cat("found ", uniq_vals_detrended$num_unique_values[1,acol], " unique values in the detrended data,
    barplot(
        table(dat1_detrended[[acol]]),
        main=paste0('Detrended data, column ', acol, ' : ', uniq_vals_detrended$num_unique_values[1,acol
        ylab='unique levels'
    )
}
```

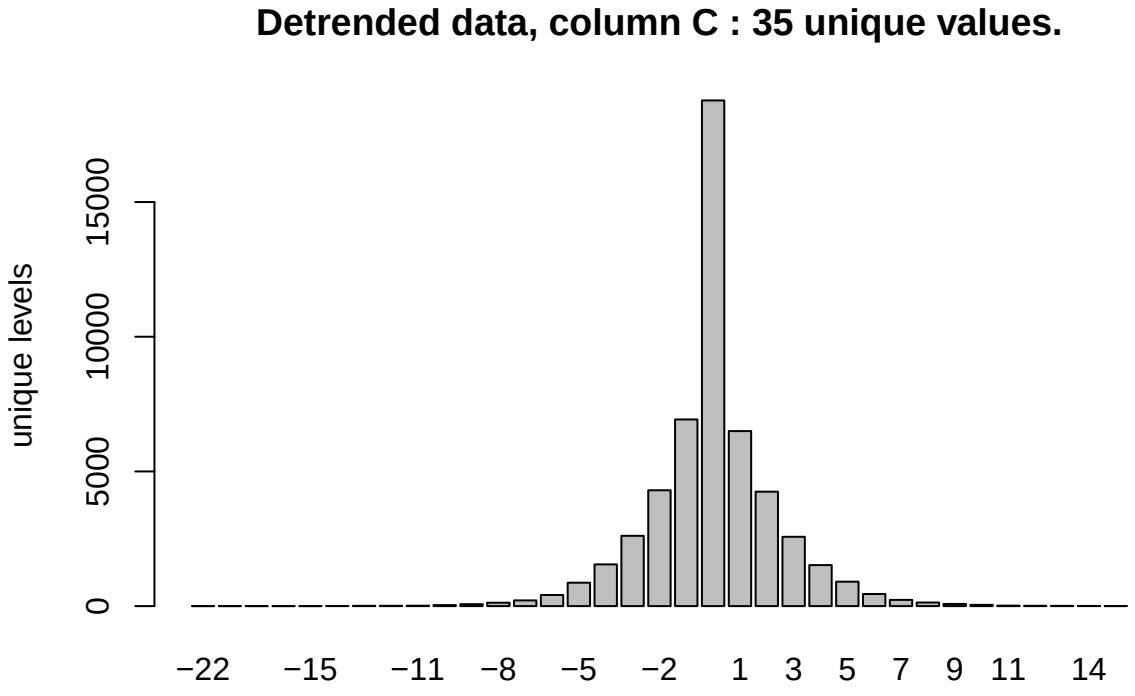
```
## found 76 unique values in the detrended data, column A
```



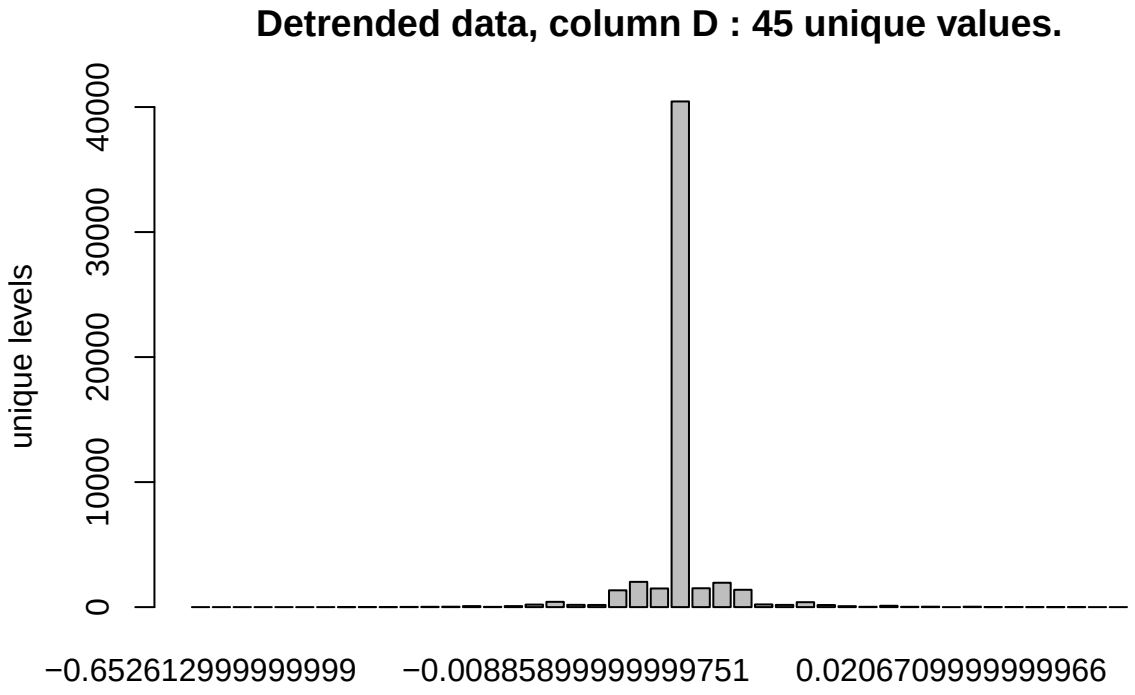

```
## found 30 unique values in the detrended data, column B
```



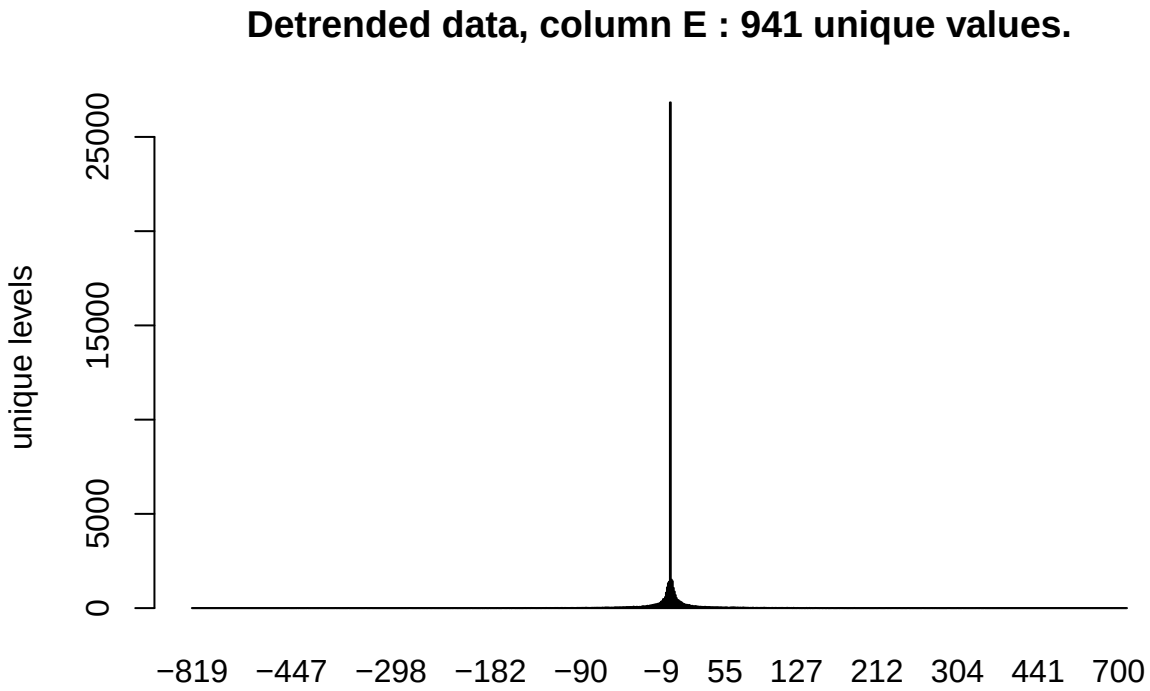

```
## found 35 unique values in the detrended data, column C
```



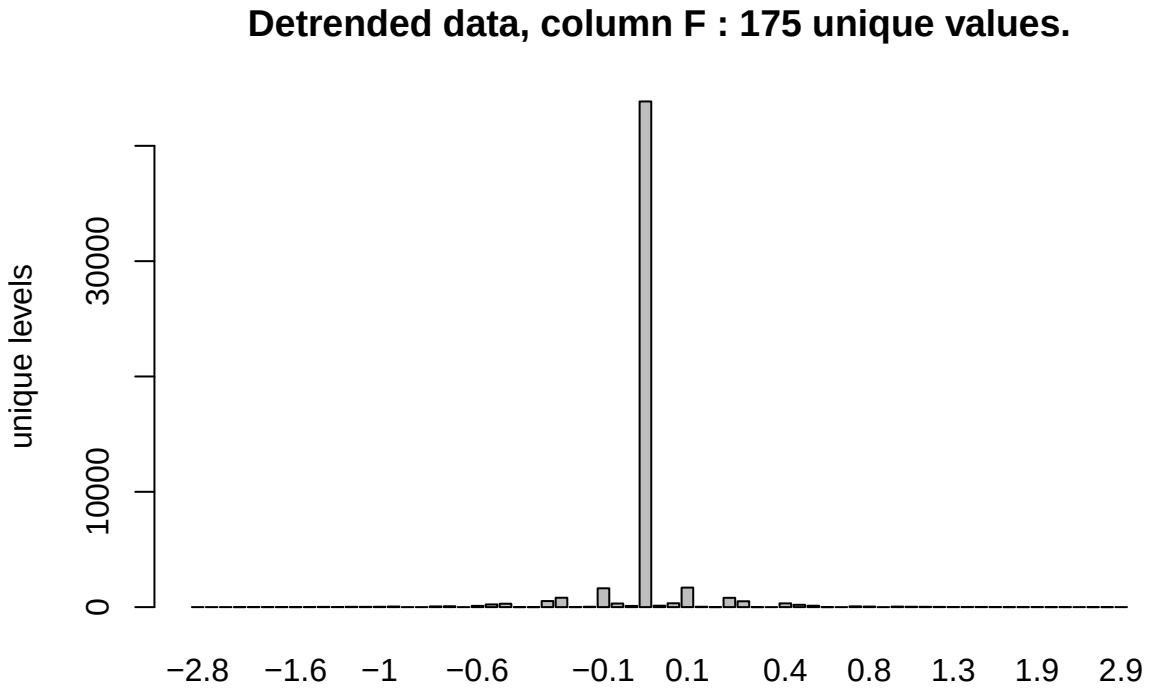

```
## found 45 unique values in the detrended data, column D
```




```
## found 941 unique values in the detrended data, column E
```




```
## found 175 unique values in the detrended data, column F
```

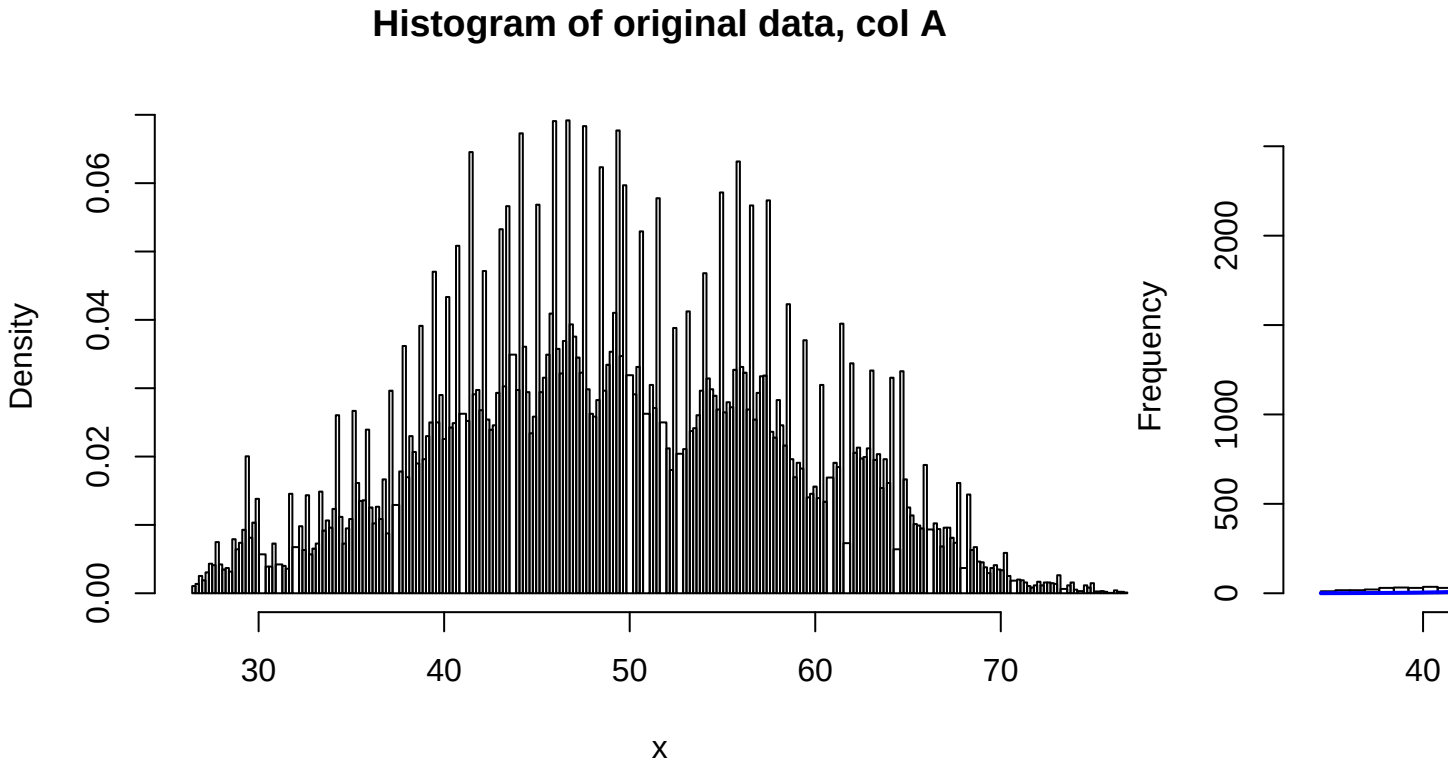


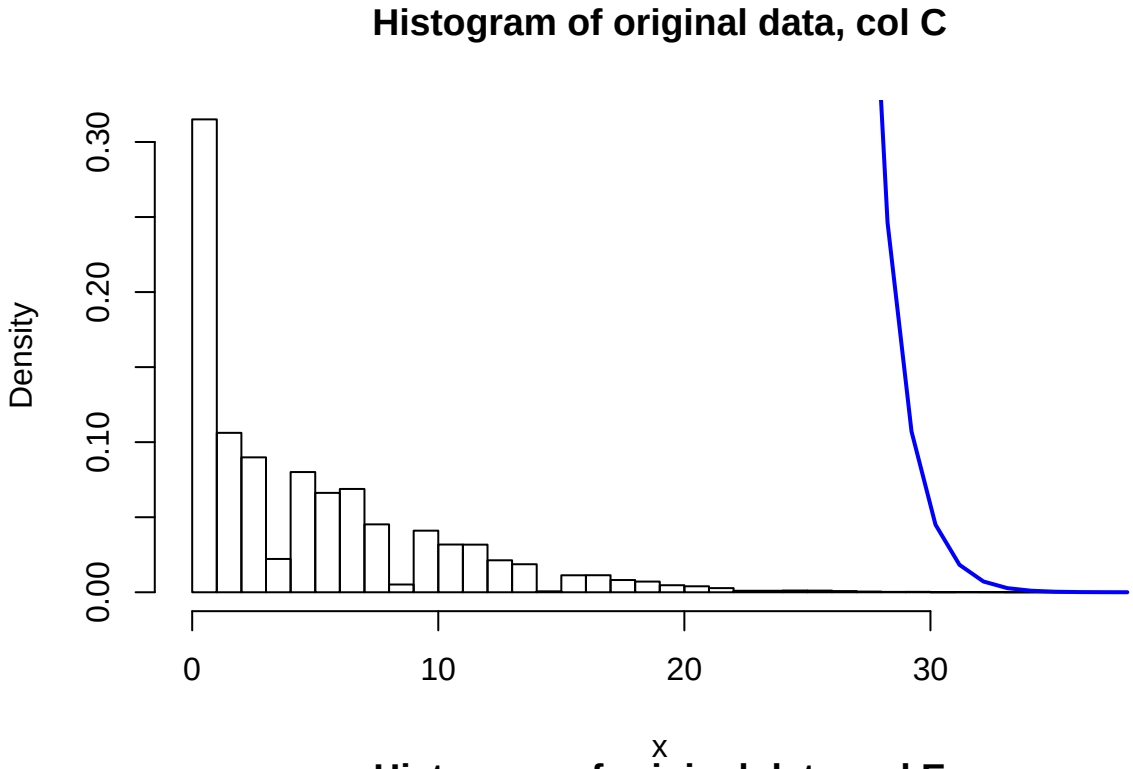

```
#print(histo_dat1_detrended)
```

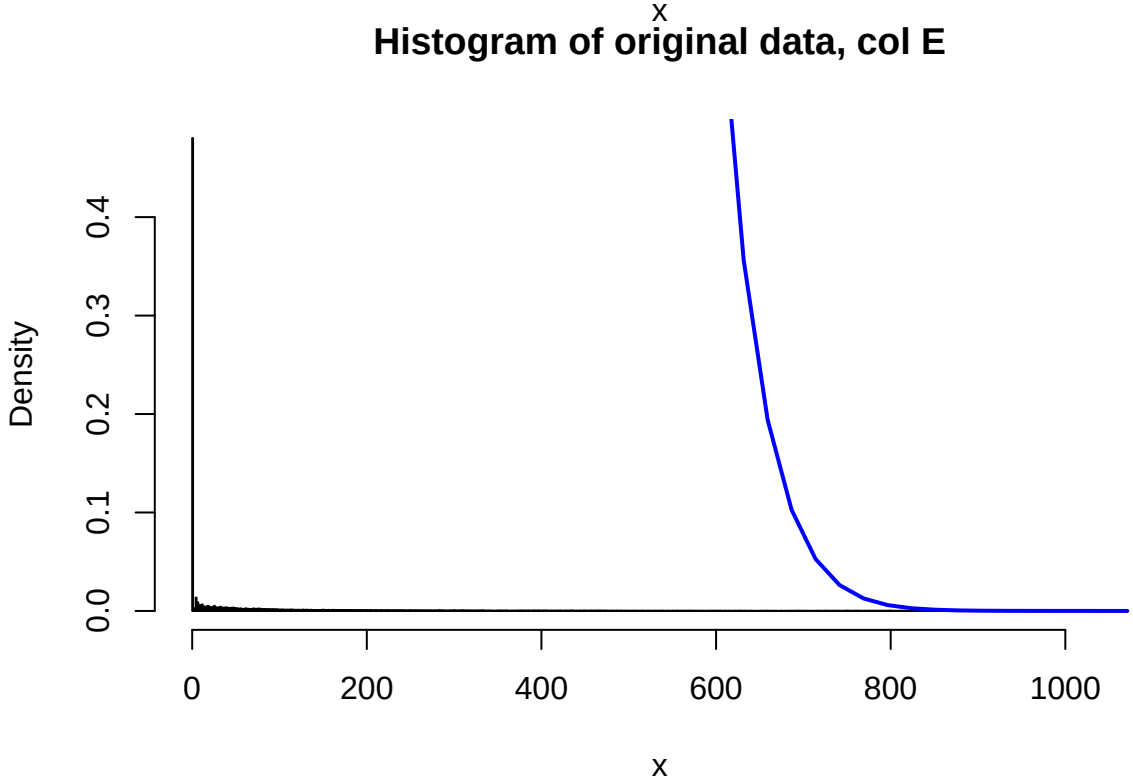
plot the histogram of each column

```
for(acol in names(dat1_noid)){
    x <- dat1_noid[[acol]]
    h <- hist(
        x=x,
        main=paste0("Histogram of original data, col ", acol, sep=''),
```

```
        breaks=uniq_vals$histograms[[acol]]$breaks
    )
    #' and plot a normal density (if it was coming from that), from https://www.statmethods.net/graphs/
    xfit<-seq(min(x),max(x),length=40)
    yfit<-dnorm(xfit,mean=mean(x),sd=sd(x))
    yfit <- yfit*diff(h$mids[1:2])*length(x)
    lines(xfit, yfit, col="blue", lwd=2)
}
```

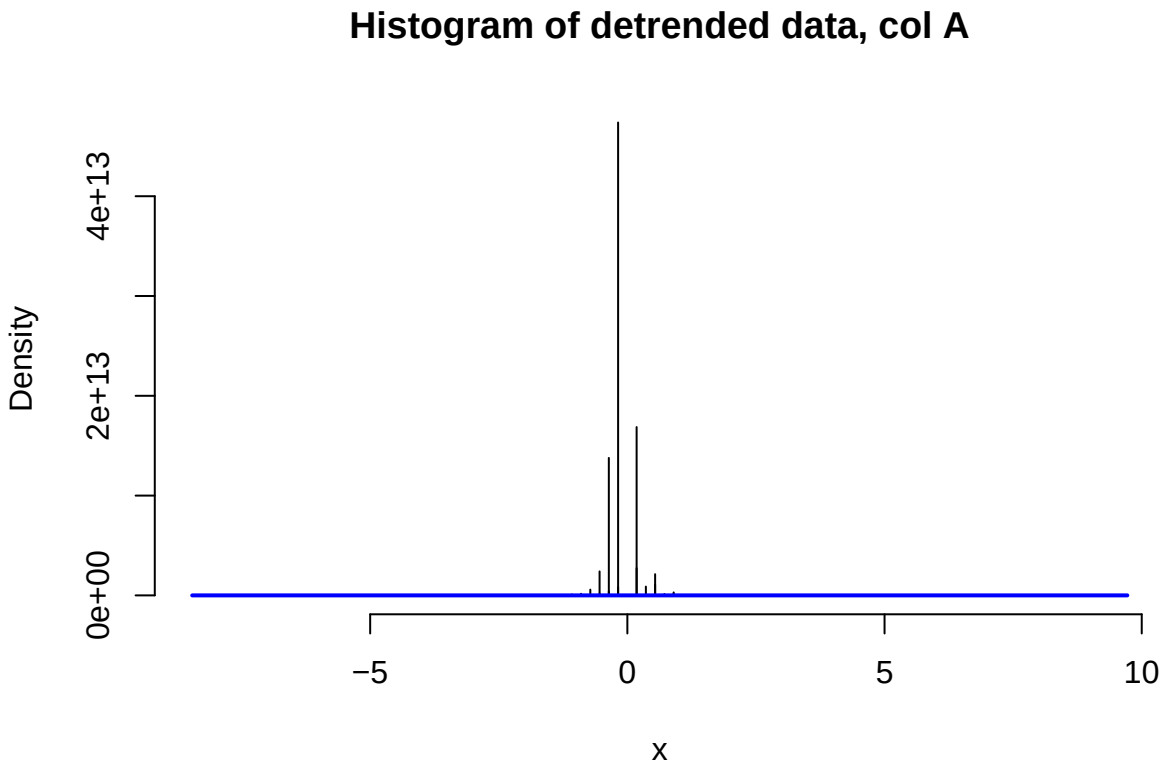


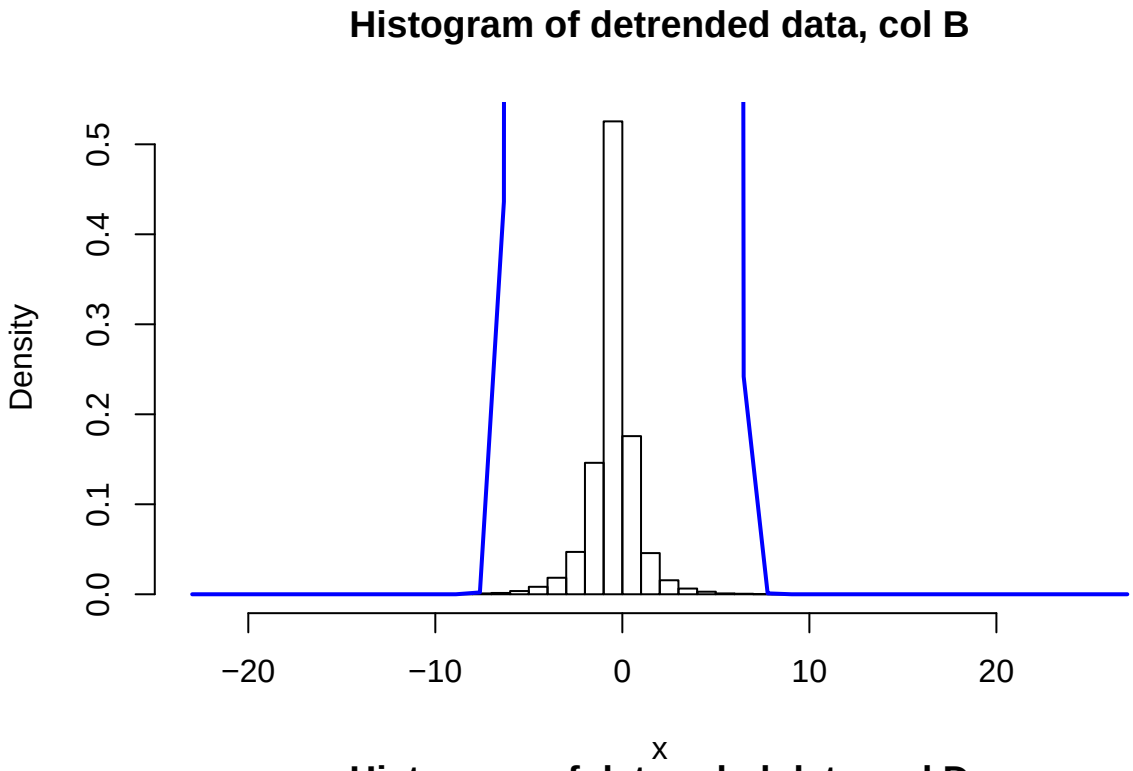


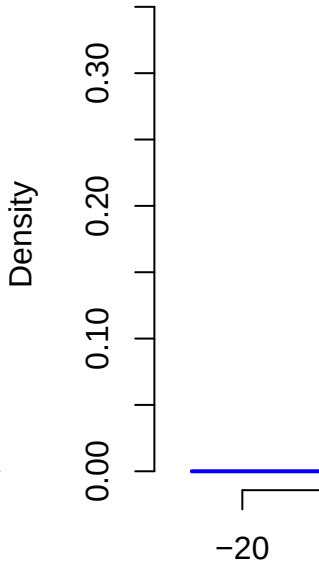


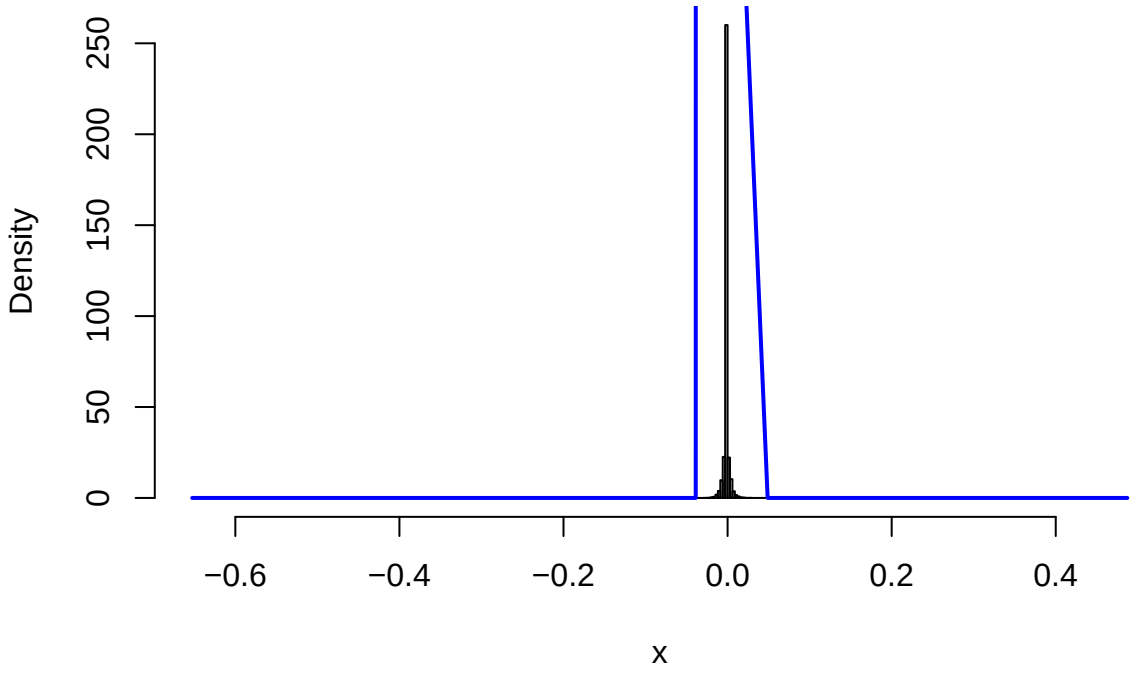

```
for(acol in names(dat1_detrended)){
    x <- dat1_detrended[[acol]]
    h <- hist(
        x=x,
        main=paste0("Histogram of detrended data, col ", acol, sep=''),
        breaks=uniq_vals_detrended$histograms[[acol]]$breaks
```

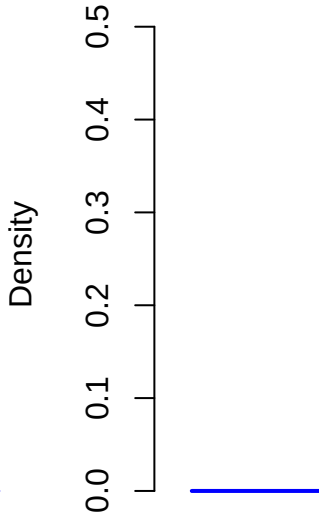
```
    )
    #' and plot a normal density (if it was coming from that), from https://www.statmethods.net/graphs/
    xfit<-seq(min(x),max(x),length=40)
    yfit<-dnorm(xfit,mean=mean(x),sd=sd(x))
    yfit <- yfit*diff(h$mids[1:2])*length(x)
    lines(xfit, yfit, col="blue", lwd=2)
}
```

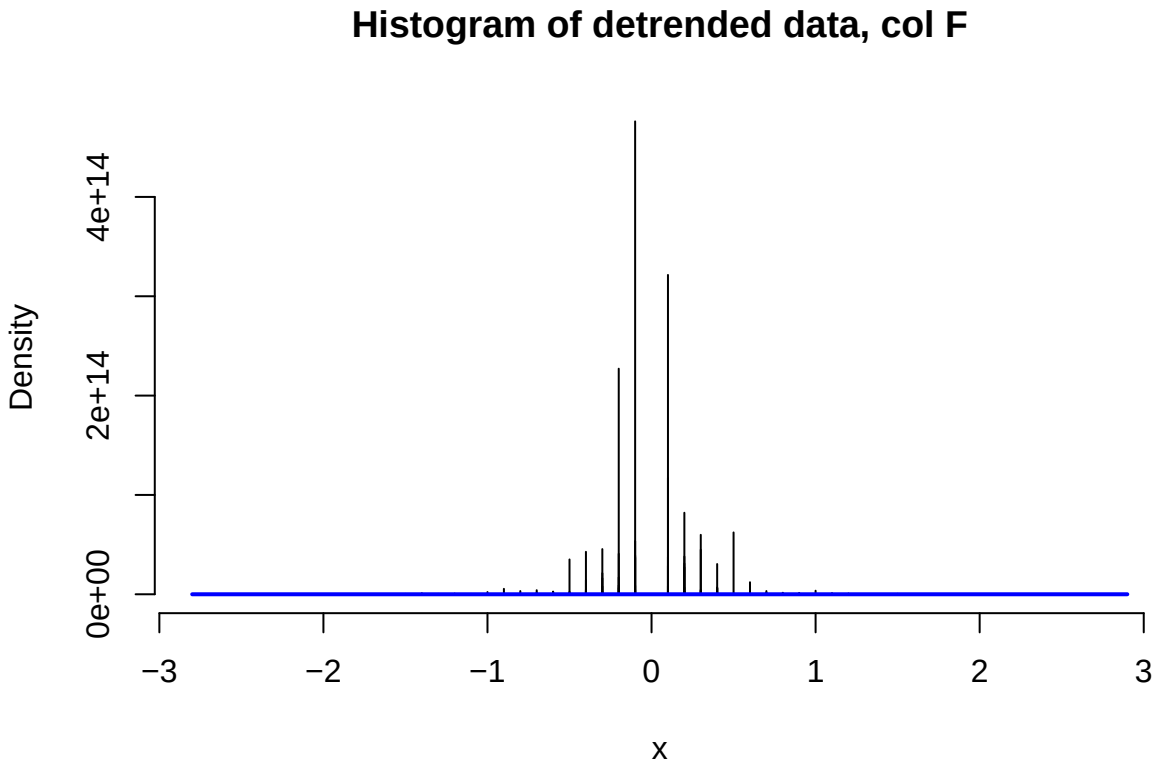



Histogram of detrended data, col B
Density
0.0
0.1
0.2
0.3
0.4
0.5
−20
−10
0
10
20
x


Density
0.00
0.10
0.20
0.30
−20

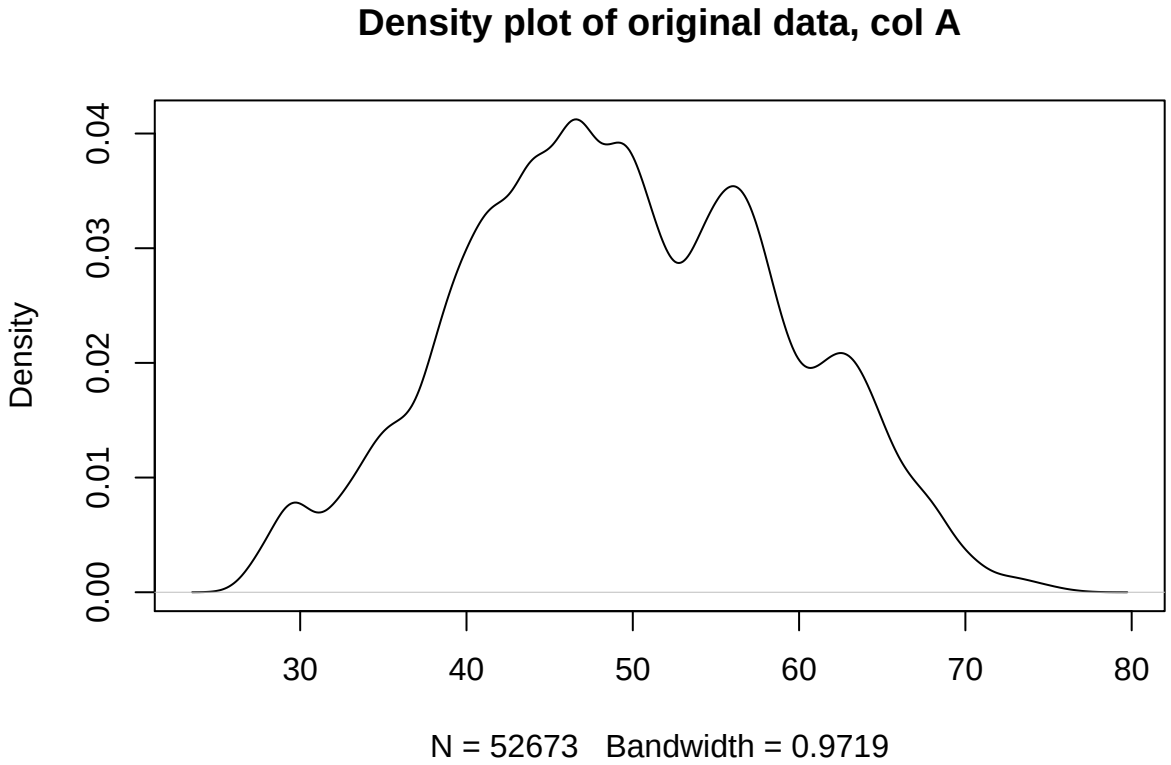

Histogram of detrended data, col D
Density
0
50
100
150
200
250
−0.6
−0.4
−0.2
0.0
0.2
0.4
x

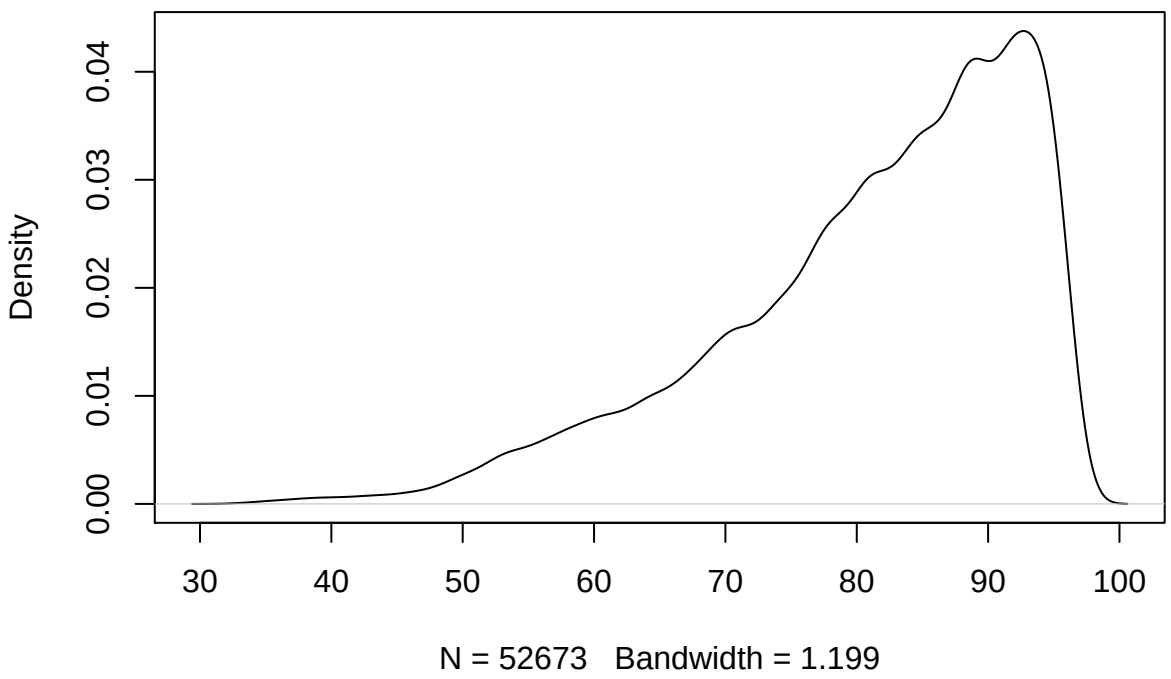

Density
0.0
0.1
0.2
0.3
0.4
0.5

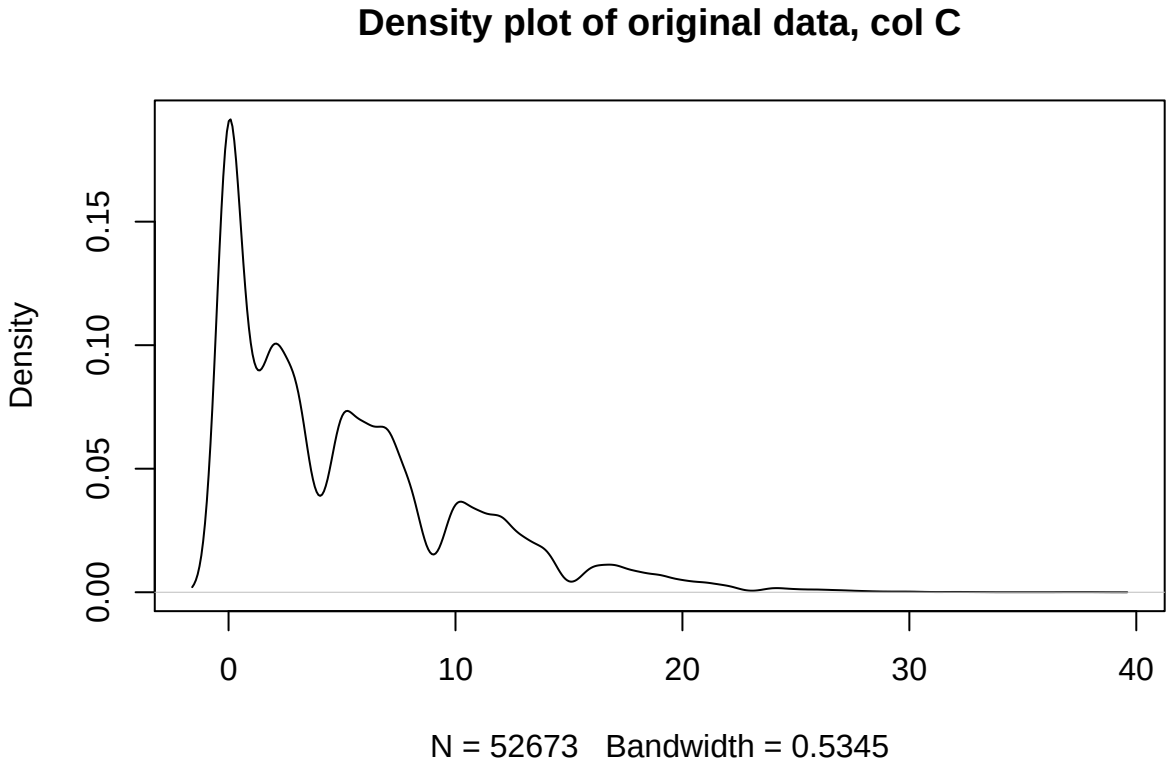


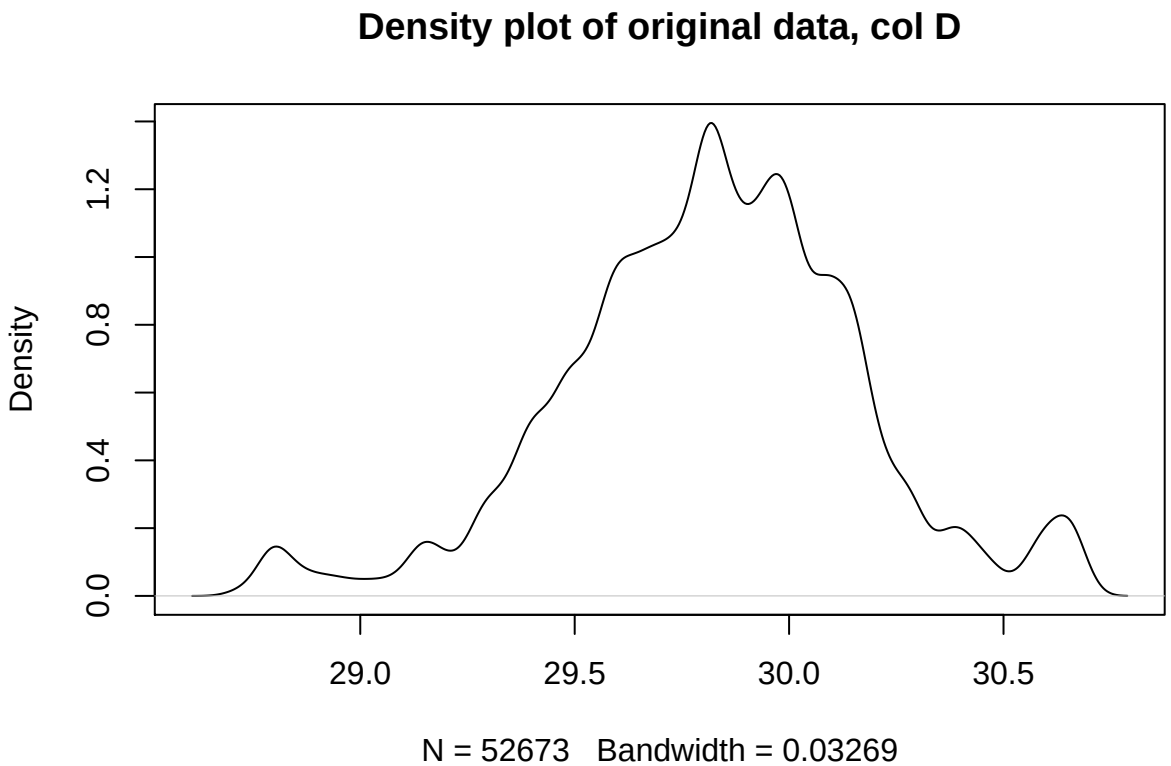
Density plots:

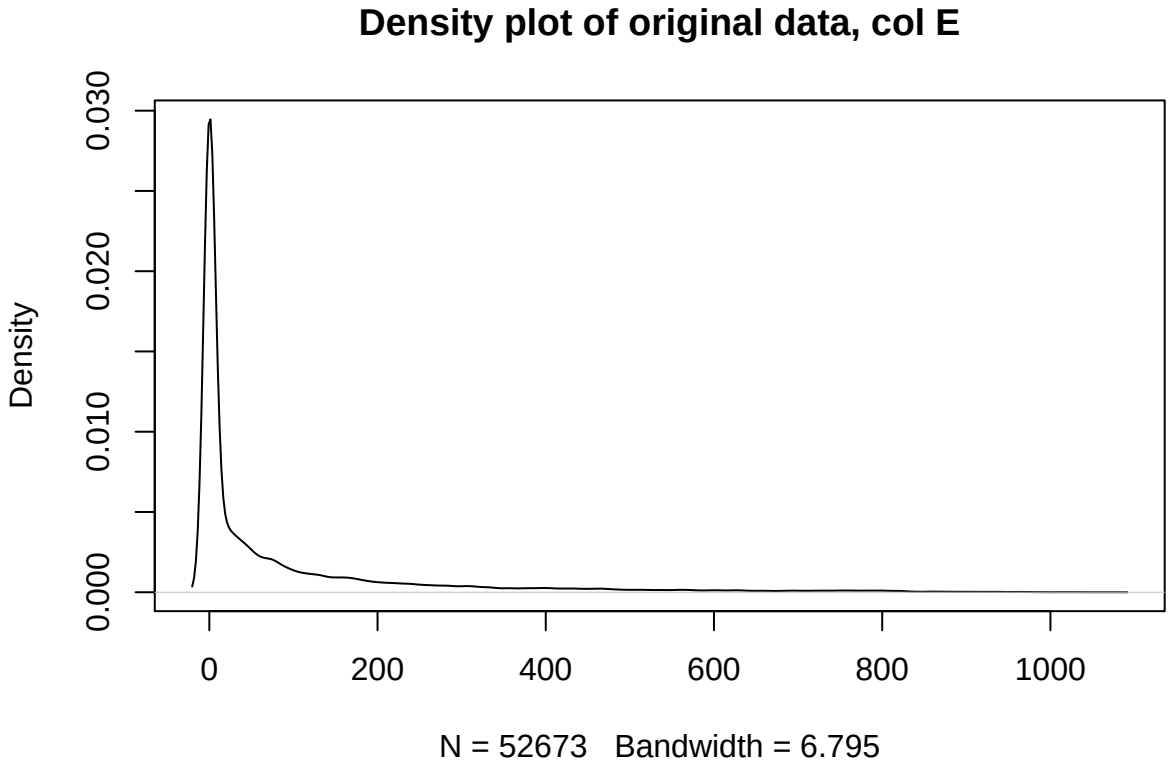
```
for(acol in names(dat1_noid)){
    plot(
        density(dat1_noid[[acol]]),
        main=paste0("Density plot of original data, col ", acol, sep='')
    )
}
```

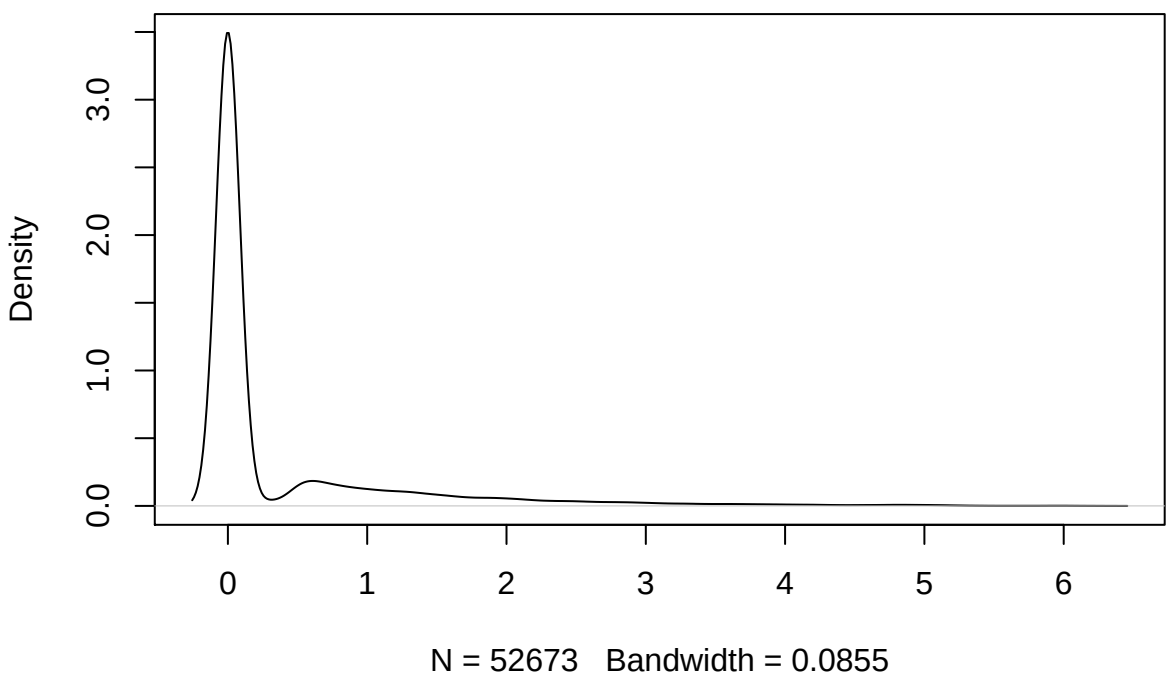

Density plot of original data, col A
Density
0.00
0.01
0.02
0.03
0.04
30
40
50
60
70
80
N = 52673 Bandwidth = 0.9719


Density plot of original data, col B
Density
0.00
0.01
0.02
0.03
0.04
30
40
50
60
70
80
90
100
N = 52673 Bandwidth = 1.199


Density plot of original data, col C
Density
0.00
0.05
0.10
0.15
0
10
20
30
40
N = 52673 Bandwidth = 0.5345

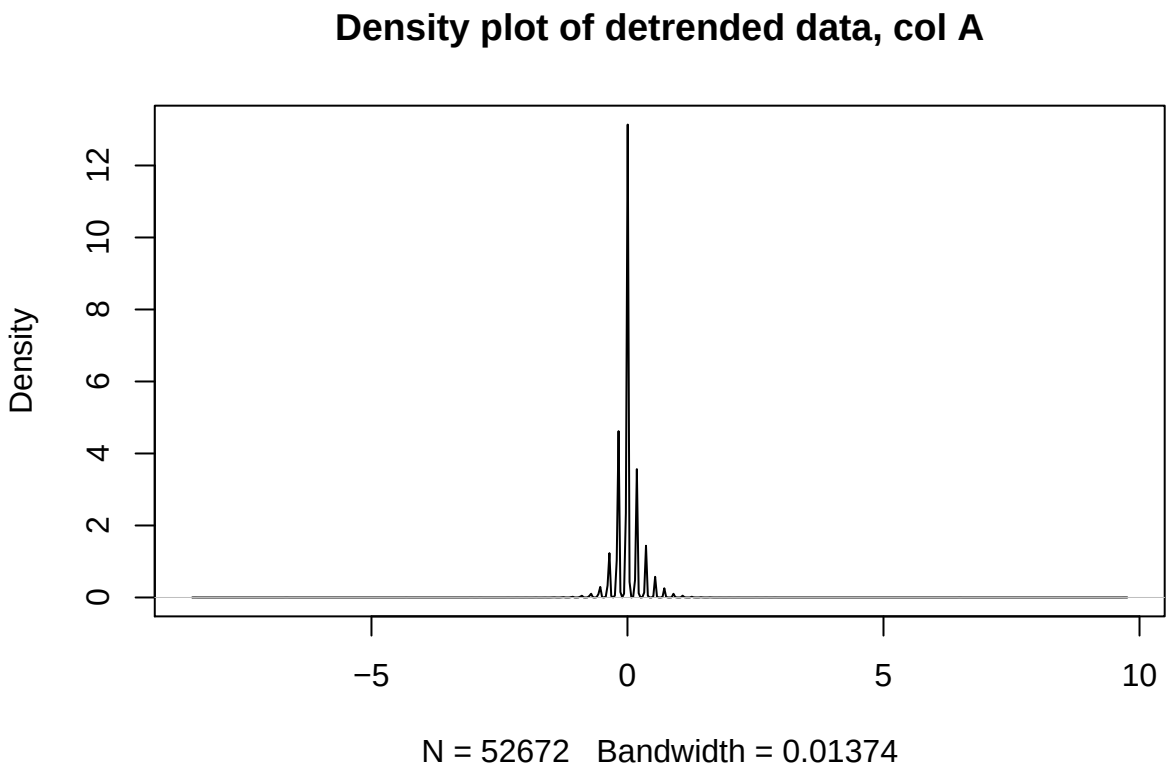

Density plot of original data, col D
Density
0.0
0.4
0.8
1.2
29.0
29.5
30.0
30.5
N = 52673 Bandwidth = 0.03269

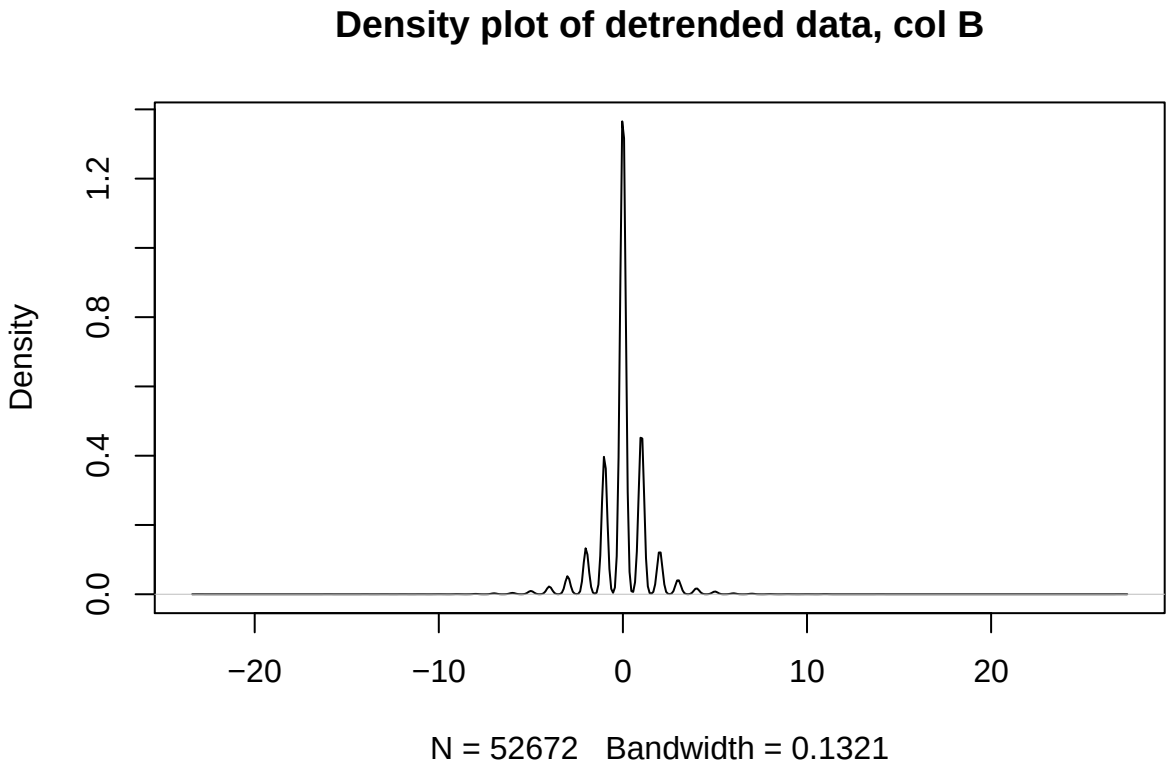


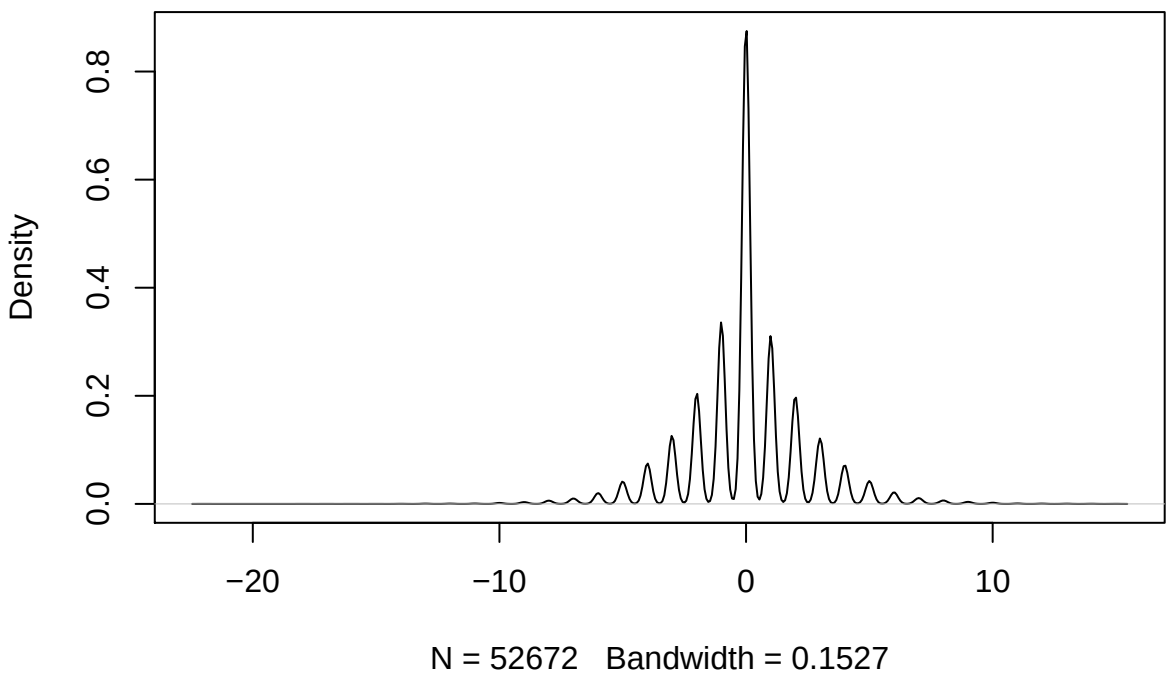


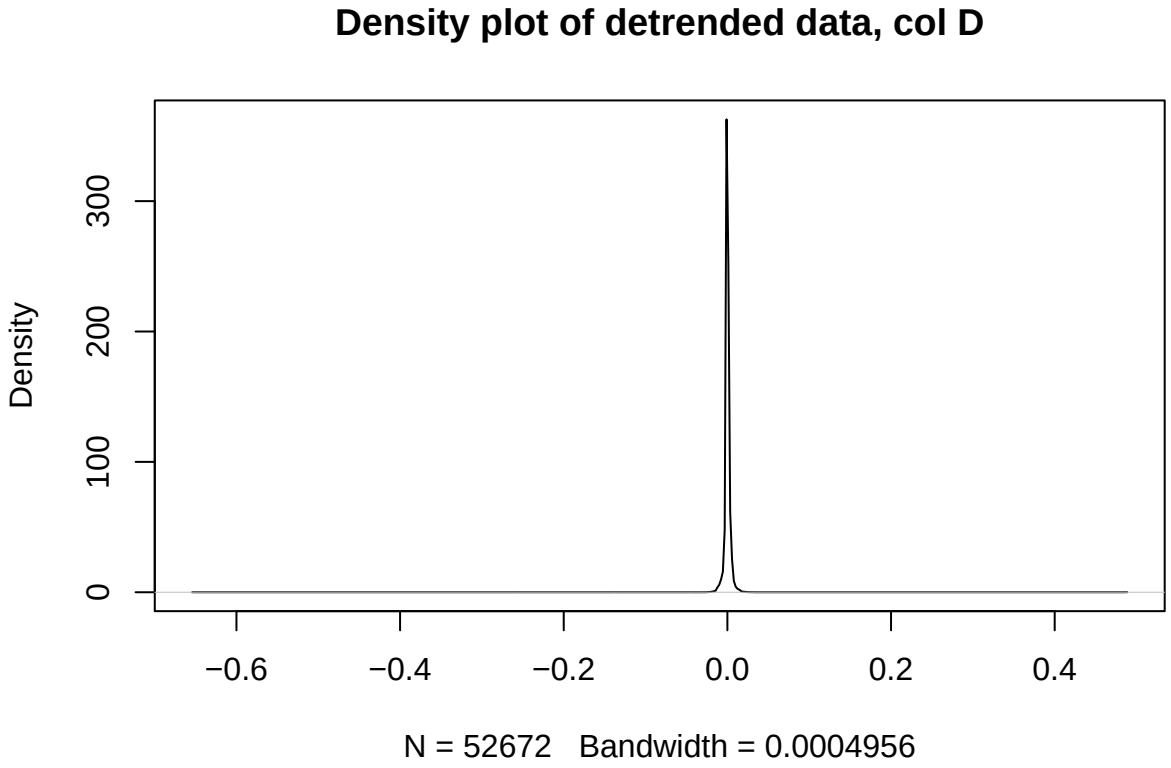
comments on the distribution of the generating process of the original data: A : looks like a mixture of Gaussians (maybe 4) B : a lognormal/weibull distrbution C : a mixture of lognormal/weibull distributions or 1 lognormal/weibull and some normal distributions D : a mixture of normal distributions E : a lognormal/weibull (shape ~1) F : a mixture of lognormal/weibull and 1 normal(?)

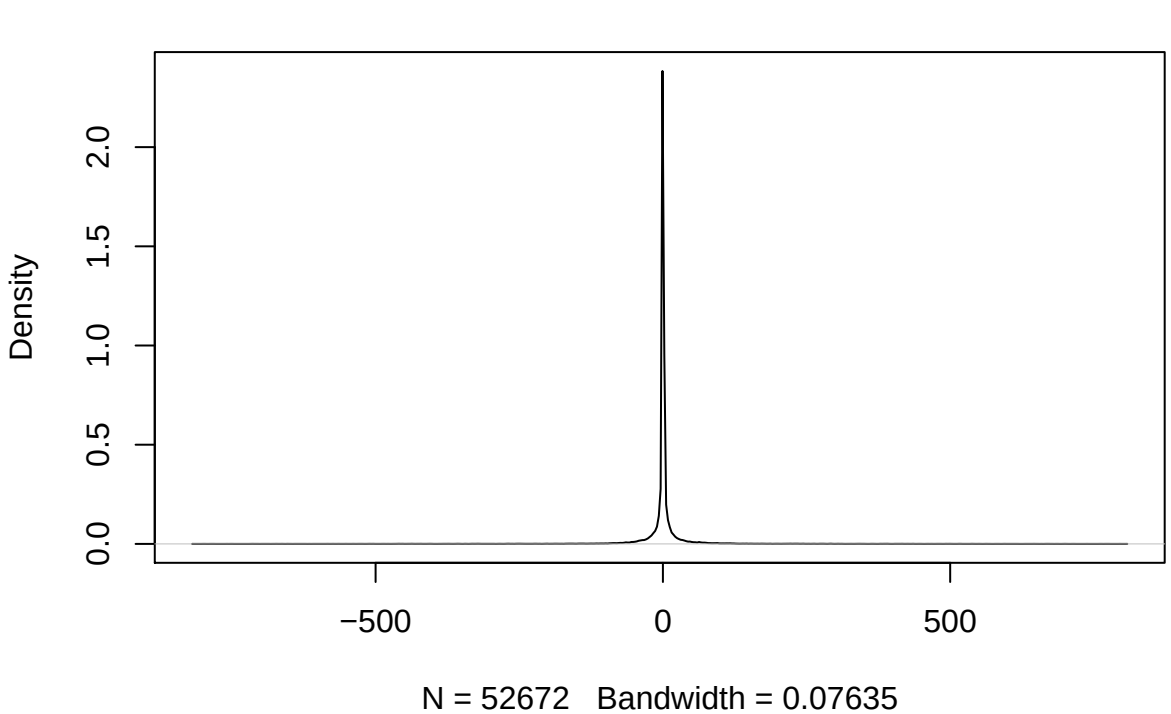
```r
for(acol in names(dat1_detrended)){
    plot(
        density(dat1_detrended[[acol]]),
        main=paste0("Density plot of detrended data, col ", acol, sep='')
    )
}
```

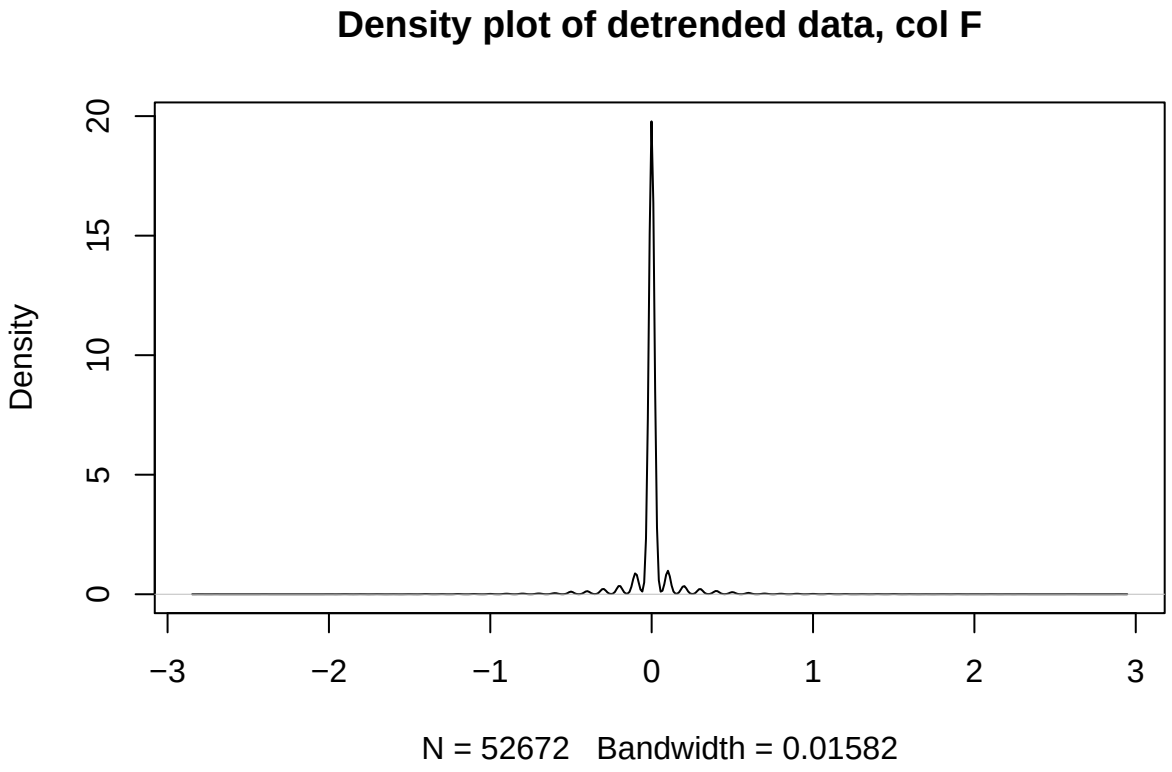



Density plot of detrended data, col B
Density
0.0
0.4
0.8
1.2
−20
−10
0
10
20
N = 52672 Bandwidth = 0.1321


Density plot of detrended data, col C
Density
0.0
0.2
0.4
0.6
0.8
−20
−10
0
10
N = 52672 Bandwidth = 0.1527


Density plot of detrended data, col D
Density
0
100
200
300
−0.6
−0.4
−0.2
0.0
0.2
0.4
N = 52672 Bandwidth = 0.0004956


Density plot of detrended data, col E
Density
0.0
0.5
1.0
1.5
2.0
−500
0
500
N = 52672 Bandwidth = 0.07635



comments on the distribution of the generating process of the original data: A,B,C,F : mixture of normal distributions D,E : pure normal distributions with very light tails, see below on fitting a distribution on them Let's also do a q-q plot (quantile-quantile plot) comparing with the Normal distribution. A q-q plot is a plot of the quantiles of test data coming e.g. from the Normal distribution against the corresponding quantiles of our data. 'the data in the x% quantile' means the first x% of the data sorted in ascending order. The more the plot coincides with the 45-degree line the stronger the assumption that our data derives from specific distribution is. Other distributions can be tested. note: flat regions indicate agreemenet, curved regions disagreement for example, flat in the middle but curved on the sides means that it probably comes from a Normal distribution but has heavy tails, more data at the tails than Normal.

```
library(ggpubr)
for(acol in names(dat1_noid)){
#   qqnorm( y=dat1_noid[[acol]],
#       main=paste0("Q-Q plot of original data, col ", acol, sep='')
#   ); qqline(dat1_noid[[acol]]);
    ggqqplot(
        dat1_noid[[acol]],
        main=paste0("Q-Q plot of original data, col ", acol, sep='')
    )
}
for(acol in names(dat1_detrended)){
#   qqnorm( y=dat1_detrended[[acol]],
#       main=paste0("Q-Q plot of detrended data, col ", acol, sep='')
#   ); qqline(dat1_detrended[[acol]])
    ggqqplot(
        dat1_detrended[[acol]],
        main=paste0("Q-Q plot of original data, col ", acol, sep='')
    )
}
```

One can also try to fit a distribution to our data for example, in the detrended column D (which appears to have a pure normal distribution)

```
library(fitdistrplus)
```

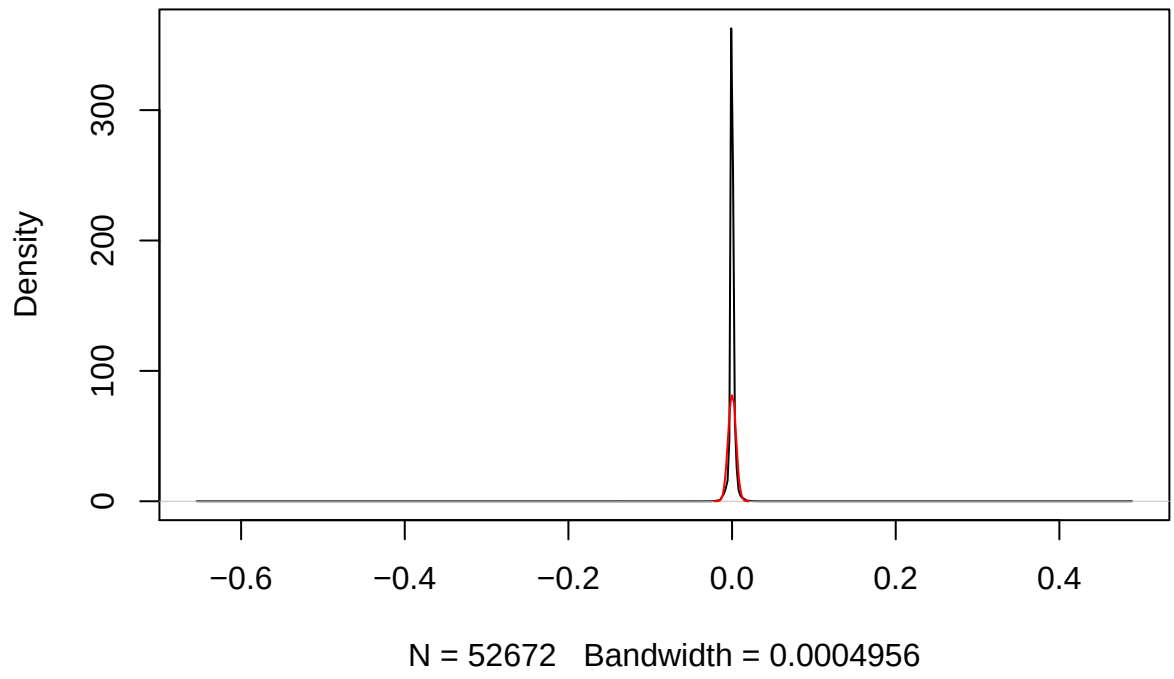
```
## Loading required package: MASS
```

```
## Loading required package: survival
```

```
acol='D'
x <- dat1_detrended[[acol]]
afit <- fitdistr(x, 'normal')
print(afit)
```

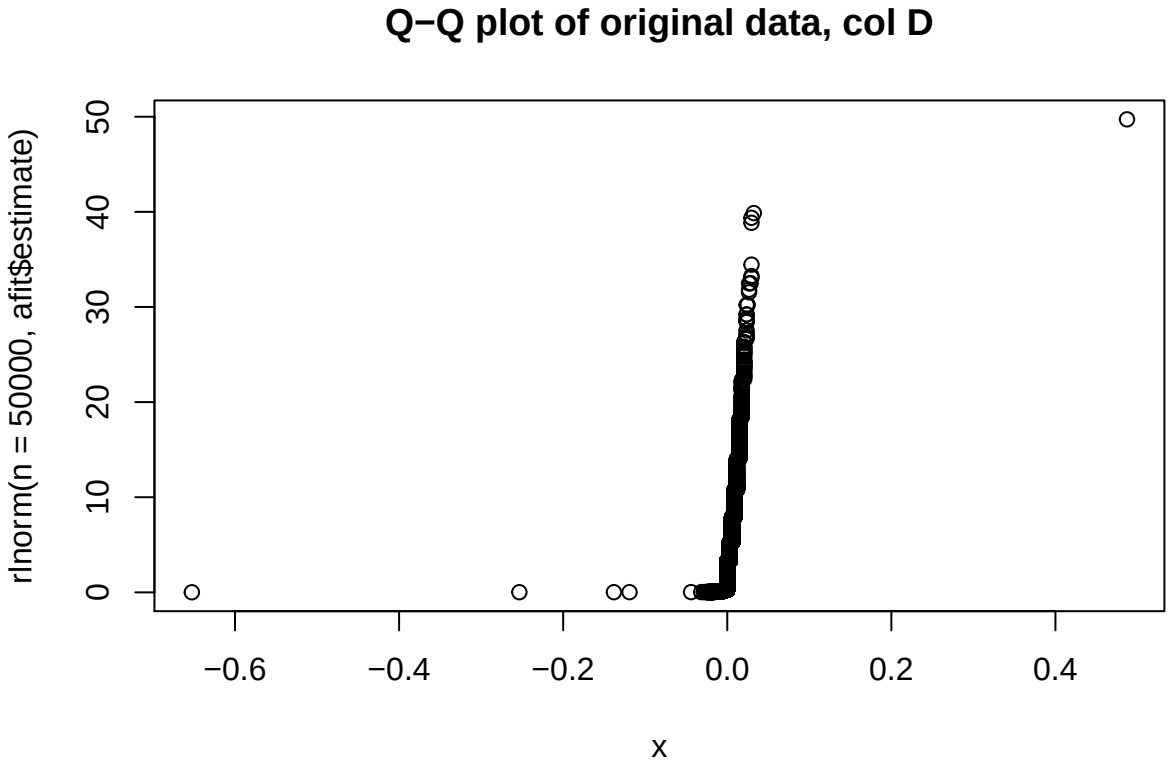
```
##        mean             sd
##   -2.018302e-06    4.843524e-03
##  ( 2.110433e-05) ( 1.492302e-05)
```

```
plot(
    density(x),
    main=paste0('Density plot of detrended data, col ', acol, ' superimposed with fitted Normal dist.',
)
lines(
    density(
        rnorm(
            n=50000,
            mean=afit$estimate['mean'],
            sd=afit$estimate['sd']
        )
    ),
    col='red',
    pch=22
)
```



the q-q plot

```
qqplot(
    x=x,
    y=rlnorm(n=50000, afit$estimate),
    main=paste0('Q-Q plot of original data, col ', acol, sep='')
)
```



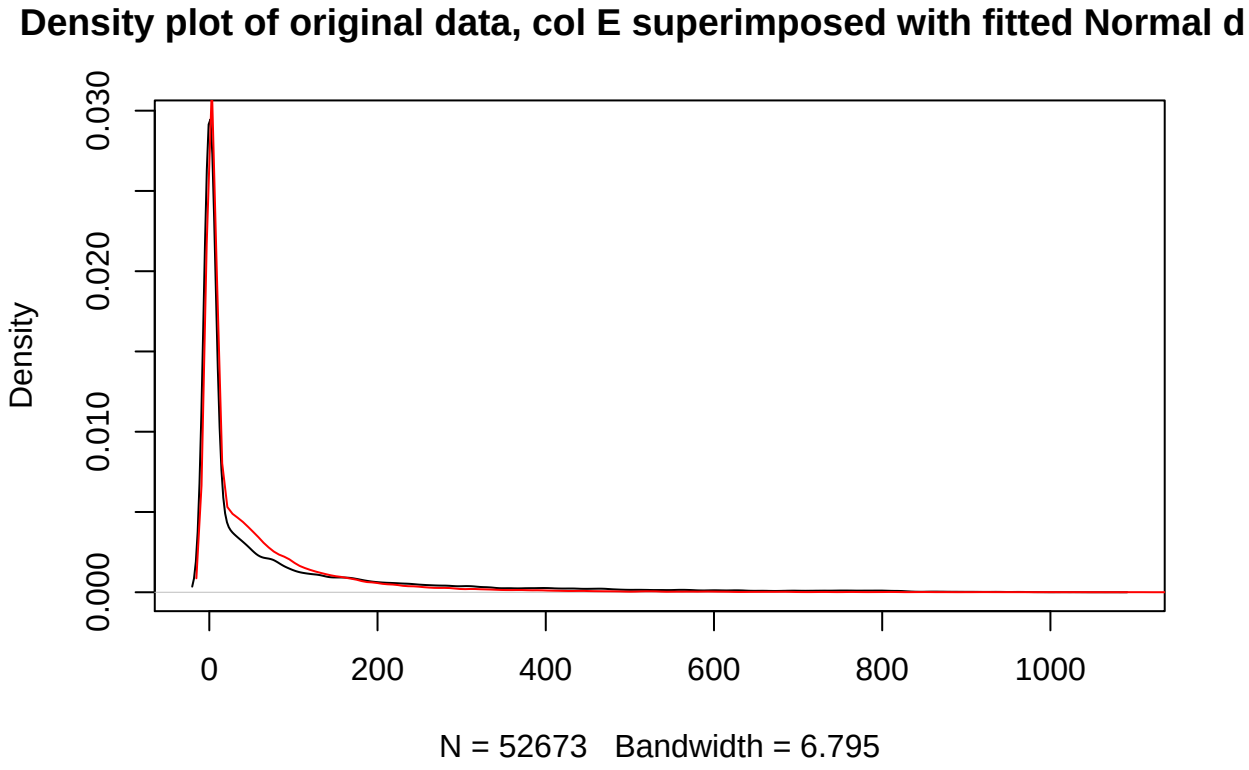
In this case (column D), we can see the peak of the data is much higher than that which comes from the fitted distribution Here is column E (original) which appears as coming from lognormal distribution

```
acol='E'
x <- dat1_noid[[acol]]+0.02 #' must be positive!
afit <- fitdistr(x, 'lognormal')
print(afit)
```

```
##     meanlog       sdlog
##   0.38862519   4.23427355
##  (0.01844951) (0.01304578)
```

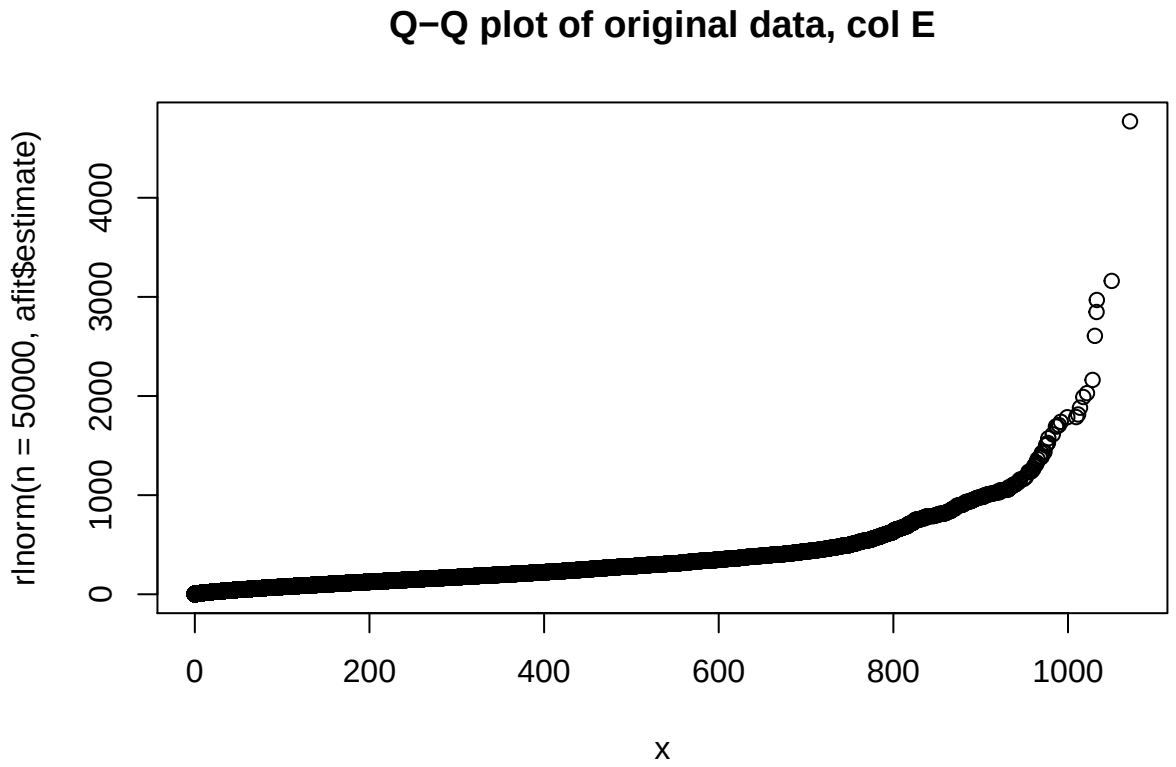
and plot

```
plot(
    density(x),
    main=paste0('Density plot of original data, col ', acol, ' superimposed with fitted Normal dist.', s
)
lines(density(rlnorm(n=50000, afit$estimate)), col='red', pch=22)
```



the q-q plot

```
qqplot(
    x=x,
    y=rlnorm(n=50000, afit$estimate),
    main=paste0('Q-Q plot of original data, col ', acol, sep='')
)
```

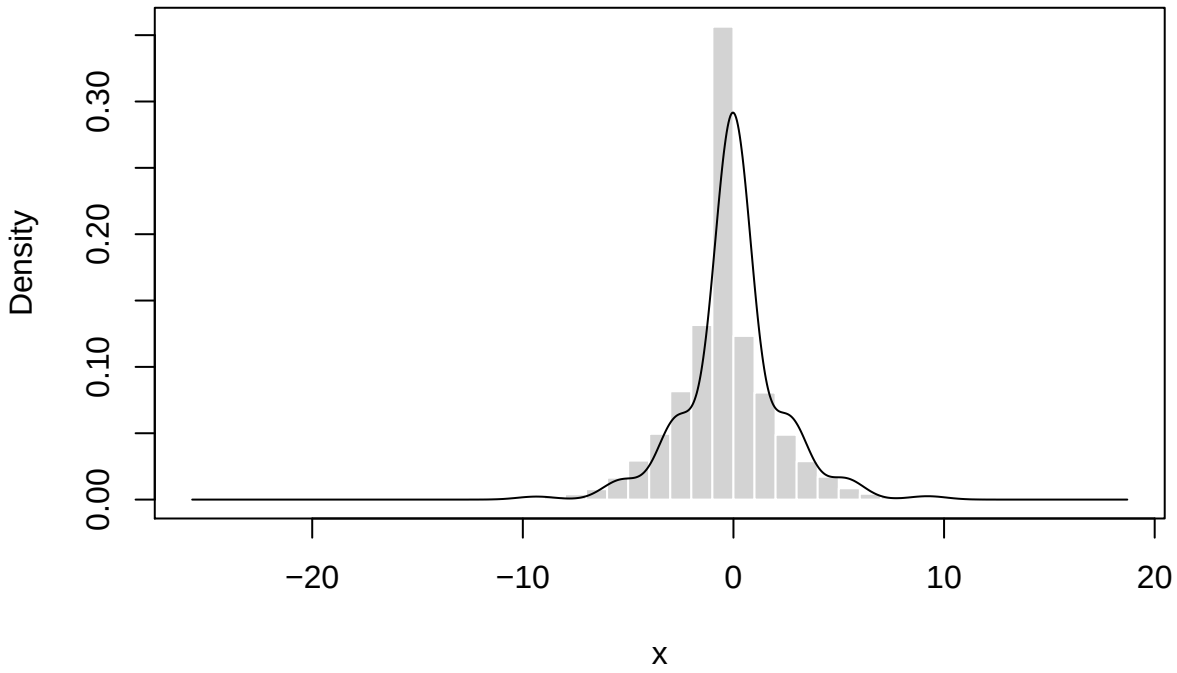


which is not bad except the tail is too heavy! For those columns that I think derive from mixture processes we can use a mixture model estimator

```
library(mclust)
```

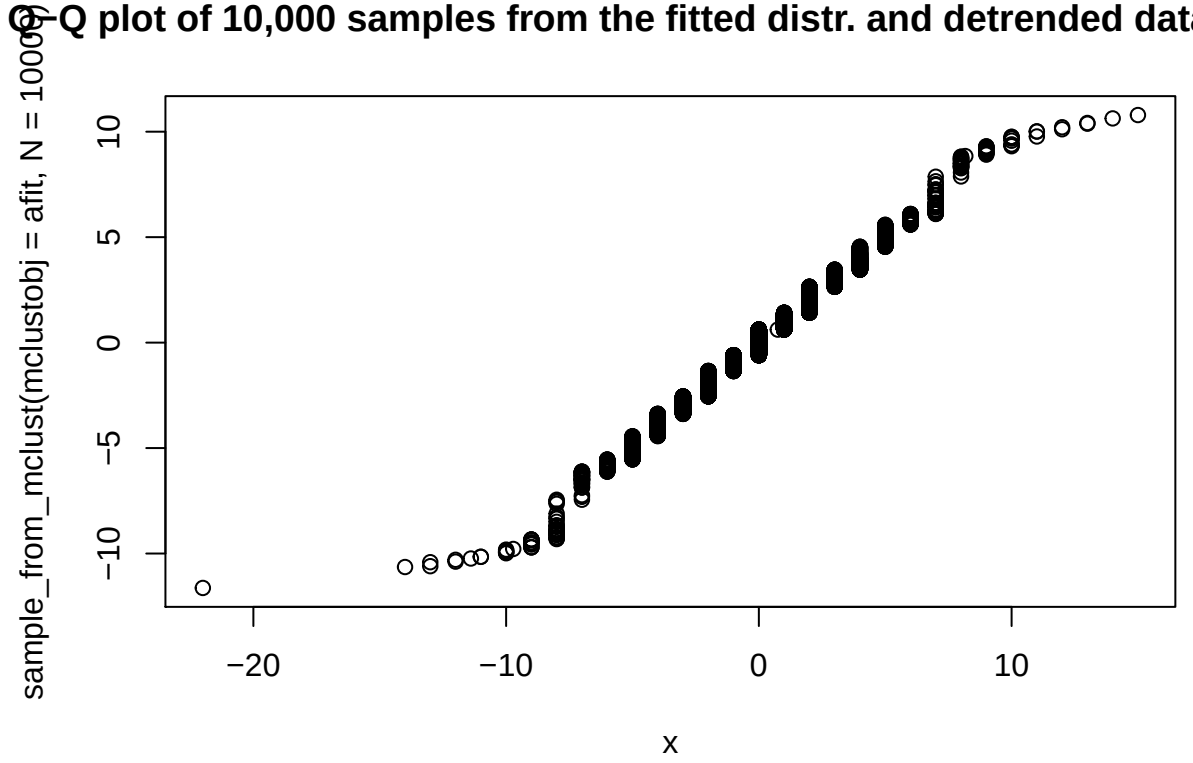
```
## Package 'mclust' version 5.4
## Type 'citation("mclust")' for citing this R package in publications.
```

```
acol='C'
x <- dat1_detrended[[acol]]
afit <- densityMclust(x)
plot(
    afit,
    what="density",
    data=x,
    breaks=50,
    main=paste0('Fitting a mixture of Gaussian distributions to detrended data, col, ', acol, sep='')
)
```



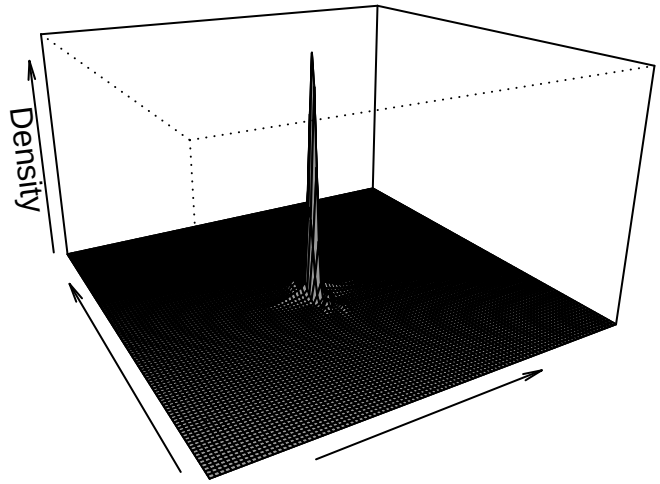
and here is a q-q plot of original and fitted distribution notice that drawing samples from mclust fit is handled by function sample_from_mclust() in library lib/MIXTURES.R

```
qqplot(
    x,
    sample_from_mclust(mclustobj=afit, N=10000),
    main=paste0('Q-Q plot of 10,000 samples from the fitted distr. and detrended data, col, ', acol, sep
)
```



Finally do a multi-variate fit using mclust. We used only 2 columns as it is better to visualise. All variables could have been used.

```
acol1='A'; acol2='B'
dat2 = cbind(
    dat1_detrended[[acol1]],
    dat1_detrended[[acol2]]
)
afit <- densityMclust(dat2)
plot(
    afit,
    what = "density",
    type = "persp",
    main=paste0('Multi-variate mixture fit for detrended data, cols ', acol1, ',', acol2, '.', sep='')
)
```



A mixture model for detrended column C with 6 Gaussians using package Rmixmod.

```
library(Rmixmod)
```

```
## Loading required package: Rcpp
```

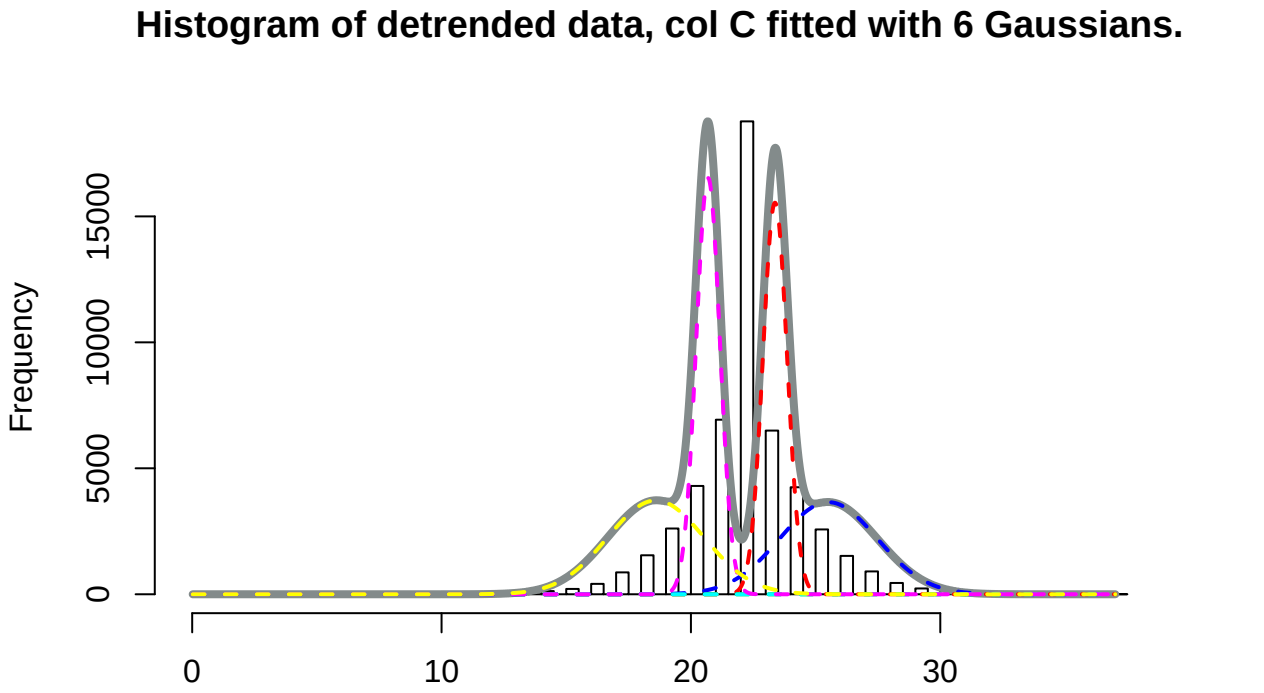
```
## Rmixmod version 2.1.1 loaded
## R package of mixmodLib version 3.2.2
##
## Condition of use
## ----------------
## Copyright (C)  MIXMOD Team - 2001-2013
##
## MIXMOD is publicly available under the GPL license (see www.gnu.org/copyleft/gpl.html)
## You can redistribute it and/or modify it under the terms of the GPL-3 license.
## Please understand that there may still be bugs and errors. Use it at your own risk.
## We take no responsibility for any errors or omissions in this package or for any misfortune that may
##
## Please report bugs at: http://www.mixmod.org/article.php3?id_article=23
##
## More information on : www.mixmod.org
```

```
acol='C'
nGaussians=6
x <- data.frame(data=(dat1_detrended[[acol]] - min(dat1_detrended[[acol]]) + 0.02))
```

```r
afit <- mixmodCluster(x, nbCluster=nGaussians)
summary(afit)
```

```
## ************************************************************
## * Number of samples    =  52672
## * Problem dimension    =  1
## ************************************************************
## *       Number of cluster =  6
## *               Model Type =  Gaussian_pk_Lk_C
## *                Criterion =  BIC(-680511.0009)
## *               Parameters =  list by cluster
## *                   Cluster  1 :
##                           Proportion =  0.1633
##                                Means =  23.3716
##                            Variances =  0.2362
## *                   Cluster  2 :
##                           Proportion =  0.1782
##                                Means =  22.0200
##                            Variances =  0.0000
## *                   Cluster  3 :
##                           Proportion =  0.1543
##                                Means =  25.5165
##                            Variances =  3.8131
## *                   Cluster  4 :
##                           Proportion =  0.1782
##                                Means =  22.0200
##                            Variances =  0.0000
## *                   Cluster  5 :
##                           Proportion =  0.1704
##                                Means =  20.6881
##                            Variances =  0.2275
## *                   Cluster  6 :
##                           Proportion =  0.1556
##                                Means =  18.5940
##                            Variances =  3.7390
## *          Log-likelihood =  340347.9111
## ************************************************************
```

```r
histCluster(
    x=afit["bestResult"],
    data=x,
    breaks=100,
    # CHECK THIS:
    main=c(paste0('Histogram of detrended data, col ', acol, ' fitted with ', nGaussians, ' Gaussians.'
)
```



Here we estimate a mixture of *different* distribution families For example Gaussian and Gamma for original data, column C

```
library(flexmix)
```

```
## Loading required package: lattice
```

```
adat <- list()
acol='C'
adat[[acol]] = dat1_noid[[acol]]-min(dat1_noid[[acol]])+10
```

we will a mix of 3 Gaussians and 1 Gamma

```
mymodels <- list(
    FLXMRglm(family = "gaussian"),
    FLXMRglm(family = "gaussian"),
    FLXMRglm(family = "gaussian"),
    FLXMRglm(family = "gaussian")
)
```

use the data from column C (original) for k=6 components

```
st<- system.time(
    afit <- flexmix(
        C ~ 1,
        data=adat,
        k=6,
        model=mymodels
    )
)
cat("done, model estimated in ", st[2], " seconds.\n", sep='')
```

```
## done, model estimated in 3.061 seconds.
```

```
print(parameters(afit))
```

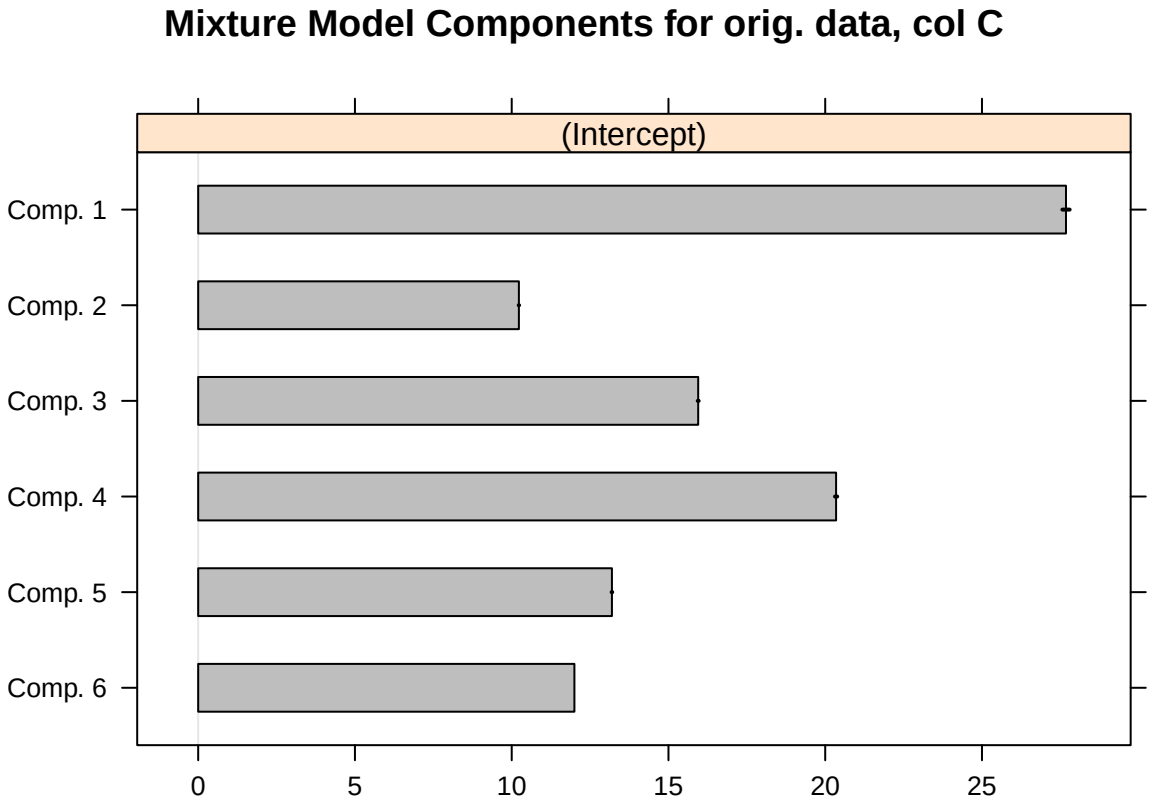
```
## [[1]]
##                      Comp.1     Comp.2     Comp.3    Comp.4     Comp.5
## coef.(Intercept) 27.680767 10.2295131 15.9485114 20.347395 13.1970149
## sigma             3.602226  0.4205237  0.8338333  1.704873  0.3977475
##                      Comp.6
## coef.(Intercept) 1.200000e+01
## sigma            1.198945e-12
##
## [[2]]
##                      Comp.1     Comp.2     Comp.3    Comp.4     Comp.5
## coef.(Intercept) 27.680767 10.2295131 15.9485114 20.347395 13.1970149
## sigma             3.602226  0.4205237  0.8338333  1.704873  0.3977475
##                      Comp.6
## coef.(Intercept) 1.200000e+01
## sigma            1.198945e-12
##
## [[3]]
##                      Comp.1     Comp.2     Comp.3    Comp.4     Comp.5
## coef.(Intercept) 27.680767 10.2295131 15.9485114 20.347395 13.1970149
## sigma             3.602226  0.4205237  0.8338333  1.704873  0.3977475
##                      Comp.6
## coef.(Intercept) 1.200000e+01
## sigma            1.198945e-12
##
## [[4]]
##                      Comp.1     Comp.2     Comp.3    Comp.4     Comp.5
## coef.(Intercept) 27.680767 10.2295131 15.9485114 20.347395 13.1970149
## sigma             3.602226  0.4205237  0.8338333  1.704873  0.3977475
##                      Comp.6
## coef.(Intercept) 1.200000e+01
## sigma            1.198945e-12
```

obtain additional information on the estimated model using refit()

```
arfit <- refit(afit)
```

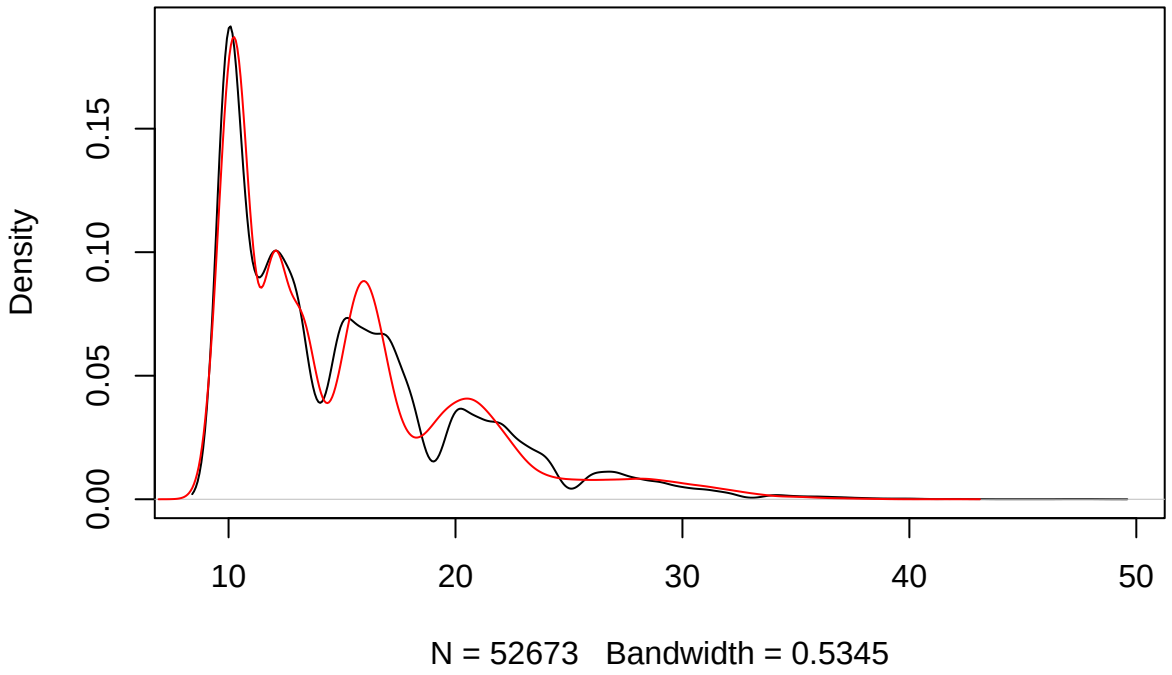
```
## Warning in sqrt(diag(z@vcov)[indices]): NaNs produced

## Warning in sqrt(diag(z@vcov)[indices]): NaNs produced

## Warning in sqrt(diag(z@vcov)[indices]): NaNs produced

## Warning in sqrt(diag(z@vcov)[indices]): NaNs produced
```

```
plot(
    arfit,
    bycluster=F,
    main=paste0('Mixture Model Components for orig. data, col ', acol, sep='')
)
```



compare densities, sample from our estimated model using rflexmix() :

```
sam <- rflexmix(afit)
plot(
    density(adat[[1]]),
    main=paste0('Fitting a mixture of different distributions to detr. data, col ', acol, sep='')
)
lines(density(sam$y[[1]]), col='red', pch=22)
```



not a bad fit at all!