Using the fit.models Package with RobStatTM

November 8, 2021

R. Douglas Martin

1 The fit.models Package and Function

The `fit.models` function was originally developed in 2000 by Kjell Konis for the R package robust, where it was accessible via both the command line and graphical user interface. Subsequently Konis developed the fit.models package currently available on CRAN (https://cran.r-project.org/web/packages/fit.models/index.html) where the Reference manual document fit.models.pdf is available. Here we show how to use the function `fit.models` with the RobStatTM package function `lmrobdetMM` to compare least squares and robust regression estimates, and with the function `covRob` to compare classical and robust covariance and correlation matrix estimates.

To get started in using the `fit.models` function, you first need to install fit.models with

```
install.packages("fit.models")
```

and load both the RobStatTM package and the fit.models package with:

```
library(RobStatTM)
library("fit.models")
```

The code for this vignette is provided in the R script `fit.modelsRobStatTM.R` that is included with other example scripts installed in the RobStatTM "scripts" folder, which is one of several RobStatTM package folders. You can find the location of the scripts folder on your computer by using the function `system.file()` as follows:

```
system.file("scripts", package = "RobStatTM")
```

NOTE: Copy/paste of the above line does not typically work, so you should type it in. The result of using this command will depend upon your computer. For example, in the case of a particular computer running Windows 10, the result is:

```
[1] "C:/R/R-4.0.3/library/RobStatTM/scripts"
```

Then you just need to navigate to the `scripts` folder, where you will find the `fit.models.vignette.R` script among all the other example R scripts. You should then copy/paste this script (and any others you might one to try out), to some other location on your computer where you want to run them.

2 Robust Versus Least Squares Fits of Linear Regression Models

The code below uses the functions `lm` and `lmrobdetMM` to compute and compare least squares (LS) and robust fits of the `zinc` data to the `copper` data in the `minerals` data set. In order to use the `fit.models`

function with `lmrobdetMM`, you first have to register `lmrobdetMM` as a function in the fit.models `lmfm` class of functions, where the "`lm`" stands for linear model and the "`fm`" stands for fitted models. You do so with the `fmclass.add.class` function:[1]

```
fmclass.add.class("lmfm", "lmrobdetMM")
```

The following code first creates the `LSfit` and `robfit` fitted model objects, and then the `fmLSrob` fitted models object that contains both of the fitted model objects.[2]

```
LSfit <- lm(zinc ~ copper, data = mineral)
control <- lmrobdet.control(family = "mopt",
    eff = 0.95)
# The choices 'mopt' and 0.95 are
# defaults
robfit <- lmrobdetMM(zinc ~ copper, control = control,
    data = mineral)
fmLSrob <- fit.models(LSfit, robfit)
class(fmLSrob)

## [1] "lmfm"       "fit.models"

names(fmLSrob)

## [1] "LSfit"  "robfit"

round(coef(fmLSrob), 3)

##        (Intercept) copper
## LSfit        7.961  0.135
## robfit      15.195  0.013

summary(fmLSrob)

##
## Calls:
```

[1] See the function `fmclass.register` in the Reference manual provided with the fit.models CRAN package.

[2] The choices "`mopt`" and `0.95` for the `lmrobdet.control` for the `family` and `eff` options are actually the default values, and we show them here just to indicate that you can make other choices as described in the corresponding RobStatTM Reference manual description.

```
##  LSfit: lm(formula = zinc ~ copper, data = mineral)
## robfit: lmrobdetMM(formula = zinc ~ copper, data = mineral, control = control)
##
## Residual Statistics:
##            Min     1Q  Median    3Q    Max
##  LSfit: -41.62 -6.457 -0.3829 5.158  46.86
## robfit: -16.54 -5.421  1.0491 7.919 116.83
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept):  LSfit:  7.96063    2.70191   2.946  0.00484 **
##              robfit: 15.19455    2.25842   6.728 1.46e-08 ***
##
##      copper:  LSfit:  0.13457    0.01983   6.787 1.18e-08 ***
##               robfit:  0.01261    0.02232   0.565  0.57467
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual Scale Estimates:
##  LSfit: 14.66 on 51 degrees of freedom
## robfit: 9.994 on 51 degrees of freedom
##
## Multiple R-squared:
##  LSfit: 0.4746
## robfit: 0.008241
```

Note that the object created by `fit.models` has the specific class `lmfm` of the general `fit.models` class, and also that the names `LSfit` and `robfit` of the `fmLSrob` object are what one would expect. Also, the `coef` extractor function is very handy for a first quick look at the comparative LS and robust coefficient estimates.

When applied to `fmLSrob`, the generic R function `summary` function produces a convenient display of the LS and robust fits results that facilitates easy comparison of the results on an item by item basis. Note that the generic `summary` calls a *method* function appropriate for the object class of its argument, and in this case the method function is `summary.lmfm`. You can view the help file for `summary.lmfm` with the code line:

```
help(summary.lmfm)
```

Likewise, when applied to an object of class `lmfm`, the generic `plot` function in R calls a `plot.lmfm` `method` function, whose behavior is described in the help file obtained with the following code line:

```
help(plot.lmfm)
```

The help file for the `plot.lmfm` method informs you that you can create the following 10 different types of plots:

1. Normal QQ Plot of Residuals
2. Kernel Density Estimate of Residuals
3. Residuals vs. Mahalanobis Distances
4. Residuals vs. Fitted Values
5. Square Root of Residuals vs. Fitted Values
6. Response vs. Fitted Values
7. Residuals vs. Index (Time)
8. Overlaid Normal QQ Plot of Residuals
9. Overlaid Kernel Density Estimate of Residuals
10. Scatter Plot with Overlaid Fits (for simple linear regression models).

Using the the generic function `plot` with argument `fmLSrob` as follows

```
plot(fmLSrob)
```

results in an interactive display in the RStudio console with the line

> Hit <Return> to see next plot:

and you can sequentially display of each of the above 10 plots in the above order. However, if you want to interactively select which plots to view in any order at in the RStudio console, then use:

```
plot(fmLSrob, which.plots = "ask")
```

Also, you can create any one of the 10 plot types by using the `which.plots` argument of `plot` function. For example, you can create a scatterplot of the zinc versus copper variables in the mineral data, with LS and robust straight line fits as shown in Figure 1, by using:

```
plot(fmLSrob, which.plots = 10)
```

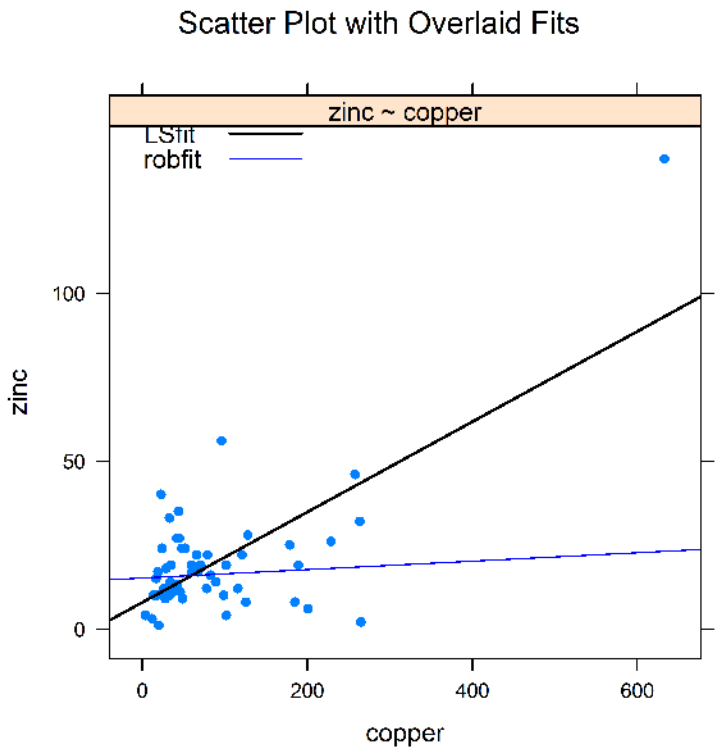

Figure 1: Scatter Plot of Zinc versus Copper with LS and Robust Line Fits

The following code line creates the Residuals versus Index plot shown in Figure 2.

```
plot(fmLSrob, which.plots = 7)
```

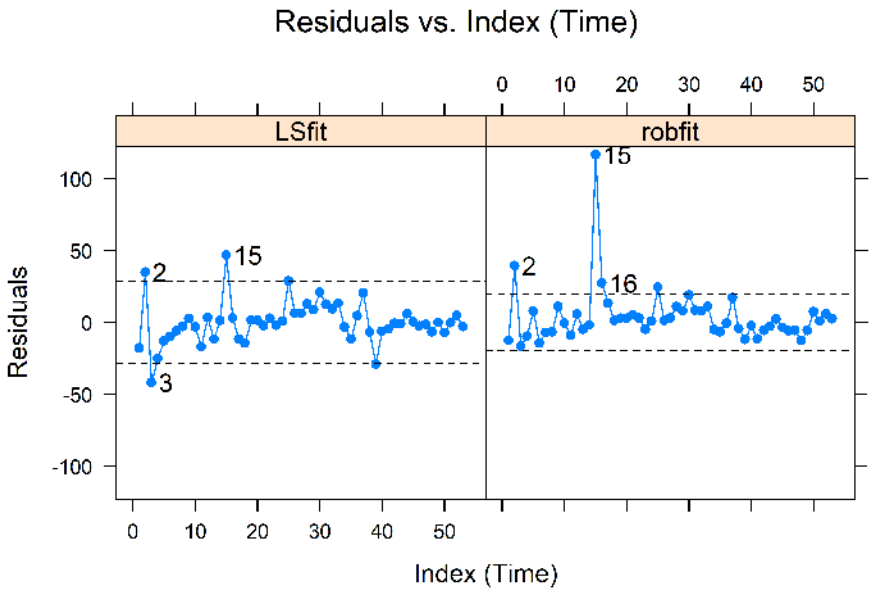

Figure 2: Residuals versus Observation Index

The following code line creates the side-by-side normal QQplots of of the regression residuals shown in Figure 2.

```
plot(fmLSrob, which.plots = 1)
```

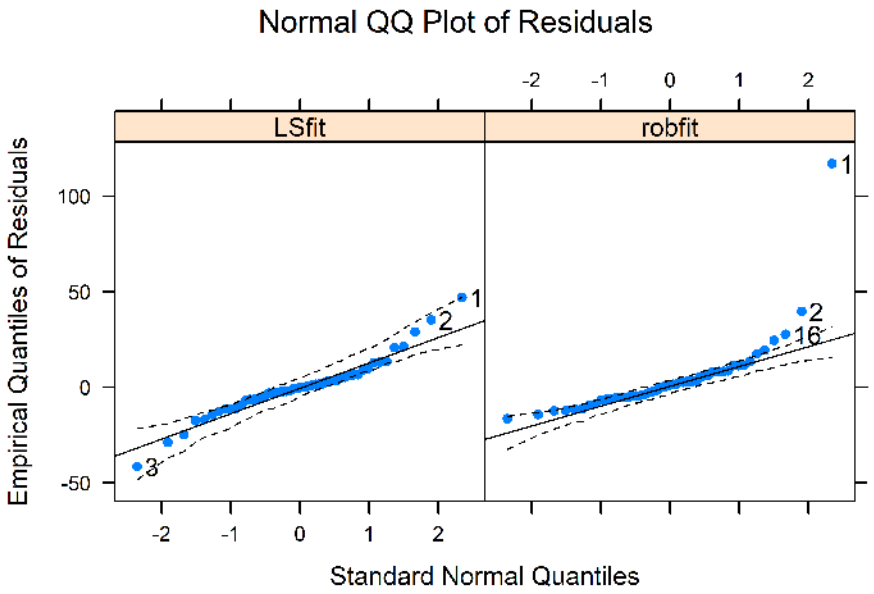

Figure 3: Normal QQ Plots of LS and Robust Fits Residuals

If you want to examine any subset of size twe or larger of the 10 plot types listed above you can easily do so,

for example as follows for the two plot types Residuals vs. Mahalanobis Distances and Residuals vs. Fitted Values:

```
plot(fmLSrob, which.plots = c(3, 4))
```

In this case you step through the two plots interactively in the RStudio console.

It turns out that if you want to create any of the 10 plot types discussed above for just a single fitted model object, you can do so. For example, try the following code lines:

```
fmLSonly <- fit.models(LSfit)
class(fmLSonly)
names(fmLSonly)
coef(fmLSonly)
plot(fmLSonly, which.plots = "ask")
```

3 Robust versus Classical Covariance Matrices

You can also use `fit.models` function for comparing two different covariance matrix estimates, and we show below how to do so for classical and robust covariance matrix estimates, where the classical covariance matrix is computed with the function `covClassic` in RobStatTM and the robust covariance matrix is computed with the function `covRob` in RobStatTM. We note that `covRob` by default automatically chooses `covRobMM` estimate if the the number of variables is less than 10, and chooses the `covRobRocke` estimate if the number of variables is 10 or greater. See the `covRob` help file for more details, including how to set `type = "MM"` or `type = "Rocke"` in case you want to over-ride the automatic choice.[3]

The code below illustrate how to computed the separate classic and robust fits inside the `fit.models` function, rather than prior to use of `fit.models`. And since the number of variables is 3, the choice `"MM"` is automatically made. You see below that the class of `fmCovClassicRob` is `covfm`, and the summary method (`summary.covfm`) prints groups together the elements of the classic and robust covariance matrix elements, the location estimates, and the eigenvalues.

```
data(wine)
wine5 <- wine[, 1:3]
fmCovClassicRob <- fit.models(Classic = covClassic(wine5),
    Robust = covRob(wine5, type = "auto"))
class(fmCovClassicRob)
```

[3] It is also possible to use the functions `covRobMM` and `covRobRocke` directly by name. However, in the current release of `RobStatTM` the ellipses plot below does not work in that case.

```
## [1] "covfm"      "fit.models"

summary(fmCovClassicRob)

##
## Calls:
## Classic: covClassic(data = wine5)
## Robust: covRob(X = wine5, type = "auto")
##
## Comparison of Covariance/Correlation Estimates:
##  (unique correlation terms)
##          [1,1]     [2,1]     [3,1]   [2,2]    [3,2]   [3,3]
## Classic 0.2136 -0.012891 -0.015599 0.47410 0.004101 0.05160
## Robust  0.2622  0.008957 -0.003907 0.04217 0.011486 0.05736
##
## Comparison of center Estimates:
##            V1    V2    V3
## Classic 13.74 2.011 2.456
## Robust  13.77 1.749 2.440
##
## Comparison of Eigenvalues:
##         Eval. 1 Eval. 2 Eval. 3
## Classic  0.4748 0.21438 0.05010
## Robust   0.2626 0.06353 0.03558
```

NOTE: The functions `covClassic` and `covRob` both have an argument `corr,` which is set to the default value `corr = FALSE`. But you can also change it to `corr = TRUE`, and we recommend running the last 4 lines in the above code using `corr = TRUE` and note what happens when the last line is executed.

You can see the names of fmCovClassicRob object, and the names of the Classic and Robust components of the fmCovClassicRob object with the following code lines:

```
names(fmCovClassicRob)

## [1] "Classic" "Robust"

names(fmCovClassicRob$Classic)

## [1] "center" "cov"    "cor"    "dist"   "call"
```

```
names(fmCovClassicRob$Robust)

## [1] "center" "cov"    "cor"    "dist"   "wts"    "call"   "mu"
## [8] "V"
```

In order to see what plot methods are available for a `fmCovClassicRob` object, read the help file displayed with:

```
help(plot.covfm)
```

The help file shows that use of the `plot` function on a covariance matrix fit.models object such as `fmCovClassicRob` creates the following four types of plots based on covClassic and covRob covariance matrix estimates:

1. Mahalanobis Distances
2. Ellipses Matrix
3. Screeplot (eigenvalues of covariance matrix
4. Distance–Distance Plot

Figures 4, 5, 6, 7 below are produced by the code lines preceding them.

```
plot(fmCovClassicRob, which.plot = 1)
```

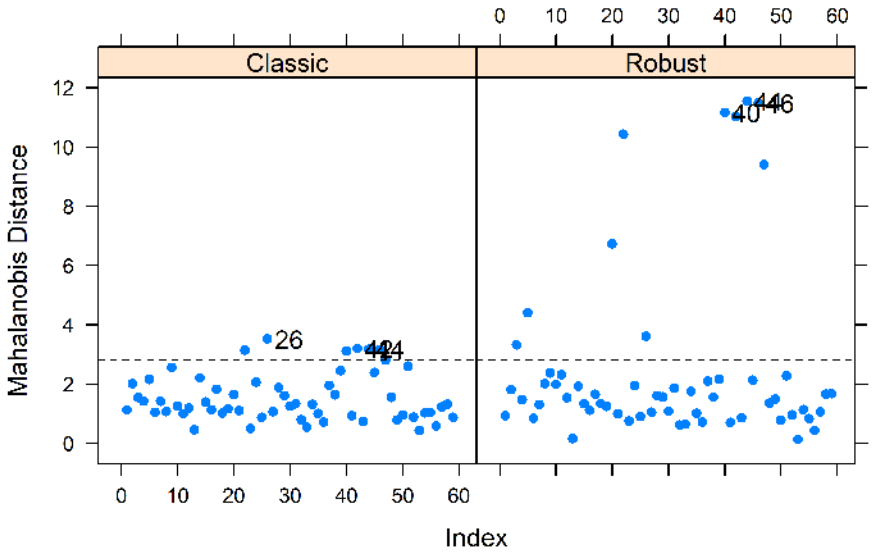

Figure 4: Robust versus Classic Mahalanobis Distances

```
plot(fmCovClassicRob, which.plot = 2)
```

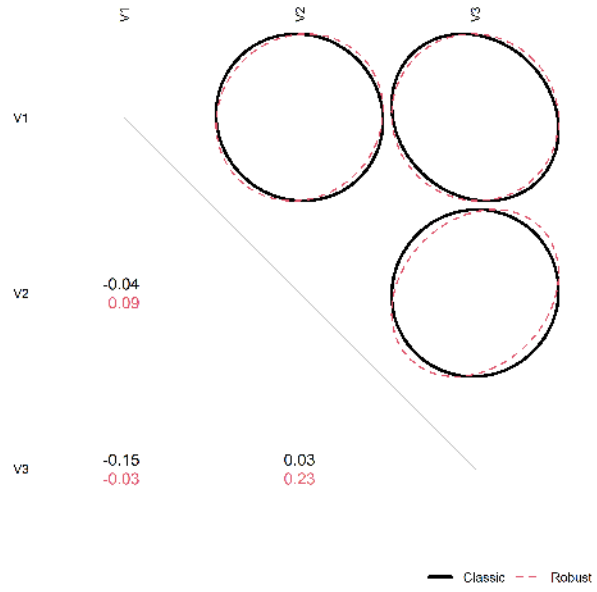

Figure 5: Robust versus Classic Correlations Ellipses

```
plot(fmCovClassicRob, which.plot = 3)
```

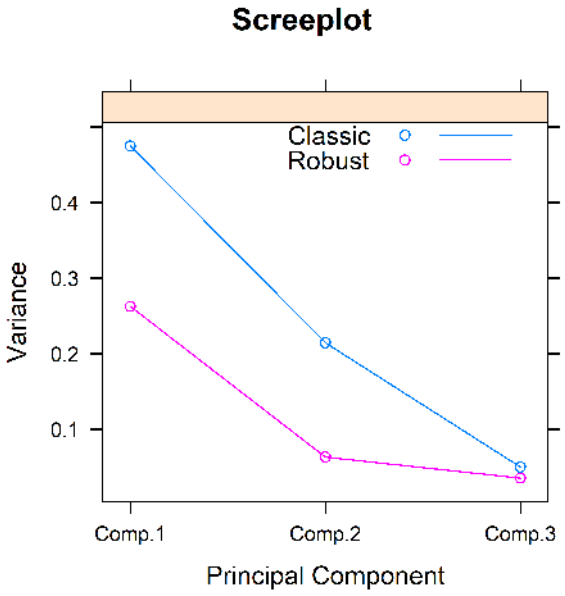

Figure 6: Classic Versus Robust Eigenvalues

```
plot(fmCovClassicRob, which.plot = 4)
```

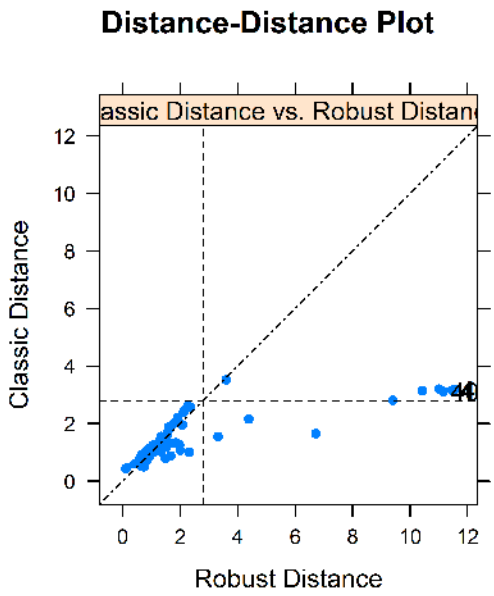

Figure 7: Robust Distances versus Classic Distances

The method function `plot.covfm` also allows you to print a summary and make any one of the four plot types when the covfm object contains just one covariance matrix estimate, as you can confirm by running the following code:

```
fmCovRob <- fit.models(covRob(wine3))
summary(fmCovRob)
plot(fmCovRob, which.plots = 1)
```