SAS1760-2016

Outlier Detection Using the Forward Search in SAS/IML® Studio

Jos Polfliet, SAS Institute Inc., Cary, NC

ABSTRACT

In cooperation with the Joint Research Centre (JRC) of the European Commission, we have implemented a number of innovative techniques to detect outliers. In this paper, we show the power of SAS/IML® Studio as an interactive tool for exploring and detecting outliers using customized algorithms that were built from scratch. The JRC uses this for detecting abnormal trade transactions on a large scale. The outliers are detected using the Forward Search, which starts from a (small) central subset in the data and subsequently adds observations that are close to the current subset based on regression (R-student) or multivariate (Mahalanobis distance) output statistics. The implementation of this algorithm and its applications were done in SAS/IML® Studio and converted to a macro for use in *PROC IML* in Base SAS®.

INTRODUCTION

Detection of outliers is an important part of any data mining or predictive modeling exercise to make sure the model fits the population well. Outliers skew the estimated effect size and model parameters. Most modelers agree that having a good model for the majority of the population is better than having a worse model for the whole population. Outliers are often inherently unpredictable and can cause misinterpretation of effect estimates as well as a significant loss of accuracy. An overview of techniques and discussion can be found in the literature about robust statistics, e.g. *Robust Statistics – Theory and Methods (Maronna, Martin, and Yohai – 2006).*

Next to improving model fit, robust outlier detection is also used as the end-result in industries like fraud detection, predictive asset maintenance or server monitoring. Here, finding abnormal patterns is the goal of the analysis.

Special thanks go to Marco Riani, Domenico Perrotta, and Francesca Torti for supporting a larger implementation of which the code presented here is a small part. Additional thanks goes to the JRC, which has made possible this work under its Institutional Research Program.

THE MASKING EFFECT

Traditional approaches for detecting outliers use residual analysis on a model that was fit on all data. Influential observations can have a drastic impact on the parameter estimates of the fitted model. In certain cases, this causes the estimated error variance to increase by a large margin as well as skew the parameter estimates. That in turn causes the responsible outliers to fall within the prediction limits, effectively masking them from detection.

In Figure 1 we see an example of the masking effect. Here, there are 5 observations in the upper left quadrant that are clearly outlying. They skew the regression line to be a lot more horizontal and increase the estimated variability. The traditional way of using studentized residuals fails here since the influential point is in fact spread out over 5 observations. The Forward Search aims to solve these problems and offers a rigorous framework to detect outliers.

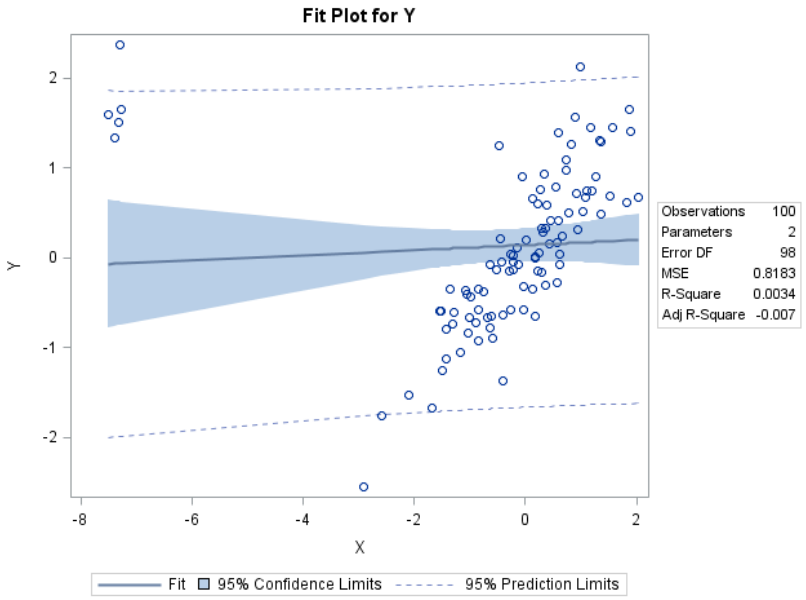

Figure 1: Example of the Masking Effect

INTRODUCTION TO THE FORWARD SEARCH

The forward search algorithm starts from a central subset in the data containing $m0$ observations out of the n original observations. In each step of the forward search, the model parameters are estimated using the m (with $m0 \leq m \leq n$) observations that are currently in the subset. The new subset consists of the $m+1$ values that have the lowest likelihood contribution, and hence are the most central to the current model. Usually, one new observation enters the subset, but it is possible that one or more observations leave the subset and two or more observations enter. This is called an *interchange*. Fit statistics are monitored at every step, and allow us to define which subsets are homogenous and at which step an outlier enters the subset. Because we monitor the parameter estimates and residuals at every step, we have an way to avoid the masking effect.

For more details and rigor we refer to *Atkinson and Riani (2010)* and *Atkinson, Riani and Cerioli (2013).*

OTHER APPROACHES

Statisticians often use a manual backward elimination process for removing outliers by looking at influence statistics and fit statistics like studentized residuals, Cook's distance, DFFits, and DFBeta. Automated approaches using dynamic thresholds for decision making exist as well and are sometimes called the backward search for outliers. These methods typically suffer from the masking effect.

Robust regression and robust statistics are an extensive part of statistical literature. `PROC ROBUSTREG` supports M-estimation, least trimmed squares (LTS), S-estimation and MM-estimation. Furthermore, the least median of squares (LMS) and least trimmed squares (LTS) subroutines in the SAS IML language perform robust regression. These subroutines can detect outliers and perform a least squares regression on the remaining observations. The minimum volume ellipsoid estimation (MVE) and minimum covariance determinant estimation (MCD) subroutines can be used to find a robust location and a robust covariance matrix, respectively, that can be used for detecting multivariate outliers and leverage points as well.

DATA SET USED IN THIS PAPER

The data set we will analyze in this paper is *SASHELP.CARS*, which is available by default in all modern SAS distributions. To read the data into memory, you can use the following code:

```
independent_varnames = {'EngineSize' 'horsepower' 'weight' 'wheelbase'
'length'};
```

```
dependent_varname = 'mpg_highway';
use ("sashelp.cars");
      read all var independent_varnames into x;
      read all var dependent_varname into y;
      read all var "model" into model;
close ("sashelp.cars");
```

This produces a $n \times p$ matrix x containing the data of all explanatory variables, a $n \times 1$ vector y containing the response variable, and a vector *model* which we will use as a unique identifier in the plots. Here, p is the number of explanatory variables in the model.

FORWARD SEARCH FOR OUTLIERS – REGRESSION

INITIAL SUBSET SELECTION

In general there are two strategies for selecting the initial subset to start the forward search, depending on the assumption that is made about the homogeneity of the data. Typically, the initial subset consists of $p+1$ observations. In linear regression, the design matrix should be of full rank, so we need at least $p+1$ observations to build a model using the initial subset.

Homogenous Data

If the data is assumed to be homogenous, the initial subset is selected using a robust regression algorithm to find the subset that is the most likely to be central to the data. We can use LMS to find the initial observations as follows:

```
lxs_options = j(8,1,.);
lxs_options[5] = 10000;

CALL LMS(sc,coef,wgt,lxs_options,y,x);

initial_obs = coef[2,];
```

The code for using LTS is similar.

Non-homogenous Data

When the data is known or suspected to contain multiple clusters, the Forward Search is started with a random sample:

```
initial_obs = sample(1:nrow(y), p+1, "NoReplace");
```

The Forward Search can be repeated multiple times with different seeds to ensure the whole range of clustering is captured. See the chapter about clustering in *Atkinson and Riani (2013)* for more information about this process.

INCREASING THE SIZE OF THE SUBSET

At each step m, we fit a linear regression model using the m observations that are currently in scope. We calculate the parameter estimates $\widehat{\beta(m)}$ and simple residuals for this model by using the matrix inversion capabilities of SAS IML:

```
idx_in = selected[1:m];
idx_out = setdif((1:n),idx_in);

Xin  = x[idx_in,];
Yin  = y[idx_in,];

beta = SOLVE(Xin`*Xin,Xin`*Yin);
```

```
resid  = y - x * beta;                    /* simple residuals */
```

Be mindful that if an intercept is desired, it has to be added manually to the design matrix X in a preprocessing step. If you have categorical variables, these should be dummy encoded before being added to the design matrix. It can sometimes be useful or insightful to monitor the changes of $\widehat{\beta(m)}$ as m increases as well.

The $m+1$ observations with the smallest absolute residual are selected as the subset for the next step. To do that, we use the *sort* function:

```
if m + 1 <= n then do;

      /***  Calculate next subset of size m + 1 at step i ***/

      call sortndx(idx_sorted, abs(resid));

      if n > m + 1 then
            selected[i+1,] = (idx_sorted[1:(m+1)])` || j(1,n-m-1,.);
      else
            selected[i+1,] = 1:n;

      entered_search = rowvec(setdif(selected[i+1,], selected[i,]));

      if ncol(entered_search) > ncol(Un) then do;
            Un = Un || j(nrow(Un), ncol(entered_search) - ncol(Un), .);
      end;

      if m > init then
            Un[i,1:ncol(entered_search)]=entered_search;

end;
```

We store the observations that have entered the search in a $n \times 10$ matrix called *Un* for later analysis and plotting purposes. Typically, one observation enters the search at each step and a row in *Un* contains only 1 index value and missing values otherwise. It can however happen that observations that were previously selected are not selected anymore and are swapped for other observations. The code shown here is abbreviated for clarity and does not deal with the special case where more than 10 observations enter the search.

MONITORING RESIDUALS

To detect if adding additional observations would change the homogeneity of the cluster, we monitor the deletion residual at each step. The $n-m$ deletion residuals of observations not currently in scope at step m are given by

$$r_{i*}(m) = \frac{y_i - x_i^T \widehat{\beta_*(m)}}{\sqrt{s_*^2(m)[1+h_{i*}(m)]}}$$

where $h_{i*}(m)$ is the leverage of observation i at step m and $s_*^2(m)$ is the estimated error variance of the model at step m.

We can calculate the deletion residual using the following code:

```
ssr = ssq(resid[idx_in]);

if (nrow(Xin)-p)=0 then
      s2=ssr/1;
else
      s2=ssr/(m-p);
```

```
iXpX = inv(Xin`*Xin);

Hi = j(n,1,.);
do k = 1 to n;
      z = x[k,];
      Hi[k,] = z * iXpX * z`;       /* external leverage */
end;

DR  = abs(resid/sqrt(s2#(1+Hi)));   /* Deletion Residuals */
```

We are interested in monitoring the smallest deletion residuals of the observations that are outside of the current subset, denoted as the Minimum Deletion Residual or MDR. When the MDR is large, this means that all observations would significantly alter the model when added, which is an indicator that you are adding an outlier to the current set. We calculate the MDR using the convenient >< subscript reduction operator that returns the minimum of a vector:

```
if m < n & ncol(idx_out) > 0 then
      mdr[m,1] = (DR[idx_out])[><];
```

Rules for determining what constitutes and what does not constitute an outlier are out of scope for this paper but are described extensively in *Torti (2009).*

INTERACTIVE EXPLORATION

After the algorithm has finished, we can plot the monitored statistics for every step in the search. Since we have stored the observation indices that have entered the search at each step, we can link the monitored statistics at each step with the observations that were responsible for it. Since multiple observations can enter, and hence will collide on some plots it does require some wizardry to get right. We use a `DataObject` to store our results data since it allows automatic brushing and selection of plots:

```
n_axis = floor((loc(Un)-1)/ncol(Un))+1;
obs_nr = Un[loc(Un)]`;

first_obs = setdif(1:n, obs_nr); /* included before FS began */
if ncol(first_obs) > 0 then
      n_axis = n_axis || j(1,ncol(first_obs), 1); /* pretend these
observations were added in the first step */
obs_nr = obs_nr || first_obs;

declare DataObject dobj;
dobj = DataObject.Create("Forward search data object");
dobj.AddVar("m", "Number of observations in the subset", mm[n_axis]);
dobj.AddVar("model", "Car model", model[n_axis]);
dobj.SetRoleVar(ROLE_LABEL, 'model');
dobj.AddVar("original_obs_nr", "Observation number", obs_nr);
dobj.AddVar("outlier_classification", "Outlier classification",
outlier_classification[obs_nr]);
dobj.AddVar("mdr", "Minimum deletion residual",mdr[mm[n_axis]]);
dobj.SetMarkerShape(loc(outlier_classification[obs_nr]), MARKER_X );

/* Add x and y information of observations that were once MDR */
dobj.AddVar(dependent_varname, dependent_varname, y[obs_nr]);
do k = 1 to ncol(independent_varnames);
      dobj.AddVar(independent_varnames[k], independent_varnames[k],
x[obs_nr,k]);
end;
```

Now that we have all data in one DataObject, it is easy to plot the desired statistics. Let's plot the Minimum Deletion Residual at each step. For reference, we calculated the distributional order statistics and stored these in a matrix *ENV* using a separate IML module called *FSRenvmdr.*

```
run FSRenvmdr(env_quantiles`, init, n, p, ENV);

declare LinePlot plot;
plot = LinePlot.Create( dobj, "m", "mdr", 0);

call FSRmdrplot( dobj, ENV, init, n, md, plot, output_string);
plot.SetWindowPosition( 50, 0, 50, 50 );
```

To investigate the observations that might potentially be outliers, we can plot scatter plots of each explanatory variable with the target variable as well. To do this, we used a slightly modified version of the *CreateFullScatterMatrix*[1] module that is standardly available in SAS/IML STUDIO®:

```
call CreateScatterRow(dobj, independent_varnames, dependent_varname);
```

A screenshot of the consolidated resulting plots can be seen in Figure 2, where we clearly see a number of outliers have been identified.

1 https://support.sas.com/documentation/onlinedoc/imlstudio/WebHelp141/imlplus_module_reference/graphics/modules/createfullscattermatrix.htm

Figure 2. Interactive Plotting with Brushing for Better Exploratory Data Analysis.

FORWARD SEARCH FOR OUTLIERS – OTHER TOPICS

MULTIVARIATE CONTEXT

Similar to the regression context, the forward search can be implemented when there is no target variable. The initial subset selection can be done using Minimum Volume Ellipsoid (MVE) or Minimum Covariance Determinant (MCD) methods that are available in SAS/IML®. Instead of monitoring the residuals at each step, we monitor the Mahalanobis distance, which is a robust and normalized measure of distance.

EXTENSIONS

A comprehensive suite of tools has been programmed in SAS/IML® Studio for the Joint Research Centre in Ispra, Italy. These tools can deal with special cases like mass point contaminations, missing values, singular design matrices, fit problems, and so on. They also provide additional functionality to export data sets, monitor different statistics, use the Forward Search on big (>10,000 observations) data sets, model selection, optimization of the Box-Cox power transformation parameter, repeating the Forward Search from different start points to find clusters, additional methods for selecting the initial subset, and additional plots.

CONCLUSION

We have shown how you can use the Forward Search to detect outliers. The results are more robust and do not suffer from the masking effect that traditional methods have. SAS/IML® Studio is well suited for implementing algorithms, as was shown in the minimal working example of a Forward Search implementation. With modules, it is easy to create structure in your program for complex algorithms. Using the interactive plotting capabilities, detecting outliers and understanding the structure of your data is made easier.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author:

Jos Polfliet

100 SAS Campus Drive

Cary, NC 27513

SAS Institute Inc.

Jos.Polfliet@sas.com

http://www.sas.com

REFERENCES

Piccolo, D., R. Verde, and M. Vichi, eds. 2011. *Classification and Multivariate Analysis for Complex Data Structures*. Springer.

Torti, Francesca. 2009. "Advances in the forward search: methodological and applied contributions." diss, University of Parma and Milano Bicocca.

Atkinson, A. C., M. Riani, and A. Cerioli. 2004. *Exploring Multivariate Data with the Forward Search*. Springer Verlag.

Atkinson, A. C., M. Riani, and A. Cerioli. 2010. "The forward search: theory and data analysis." *Journal of the Korean Statistical Society* 39.2: 117-134.

Atkinson, A. C. and M. Riani, 2000. *Robust diagnostic regression analysis*. Springer Verlag.

Maronna, R., Martin, R.D, Yohai, V. 2006. *Robust Statistics - Theory and Methods.* Wiley.

APPENDIX A: CODE

The full code is also downloadable on GitHub https://github.com/JosPolfliet/FSDA-SAS. The following modules are used in the code but are not strictly necessary

- *FSSignalDetection*: This module is a collection of rules to classify observations as outliers in the Forward Search.
- *FSRmdrplot*: A preconfigured plot containing the Minimum Deletion Residual at each step, as well as the estimated bounds.
- *FSRenvrmdr*: Module to calculate the envelopes of the Minimum Deletion Residual.
- *CreateScatterRow*: Module to create a scatter plot for each combination of y and x_i where x_i are the independent variables. This is heavily based on the standard SAS/IML® module *CreateFullScatterMatrix.* See https://support.sas.com/documentation/onlinedoc/imlstudio/WebHelp141/imlplus_module_reference/graphics/modules/createfullscattermatrix.htm.

```
/********************/
/***  Load data    ***/
/********************/

independent_varnames = {'EngineSize' 'horsepower' 'weight' 'wheelbase'
'length'};
dependent_varname = 'mpg_highway';
use ("sashelp.cars");
      read all var independent_varnames into x;
      read all var dependent_varname into y;
      read all var "model" into model;
close ("sashelp.cars");


/************************************/
/***  Initial subset selection            ***/
/************************************/

ncomb = comb(nrow(x),ncol(x));
if ncomb < 10000 then ncomb = -1; /* use all subsets if the number of
combinations < 10000*/
else ncomb = 10000;
```

```
lxs_options = j(8,1,.);
lxs_options[5] = ncomb;

CALL LMS(sc,coef,wgt,lxs_options,y,x);

initial_obs = coef[2,];

/***********************/
/***  Initalization  ***/
/***********************/

x                 = x || j(nrow(x),1,1);        /* add intercept */


n                 = nrow(x);
p                 = ncol(x);
m0                = ncol(initial_obs);
selected    = j((n-m0)+1,n,.);
selected[1,]=  initial_obs || j(1,n-m0,.);

/*  MDR:      Minimum deletion residuals at each step*/
MDR         = j(n,1,.);

mm                = do(m0, n, 1);

/*  Un:        n x ? Matrix which contains the unit(s) included
               in the subset at each step of the search. */
Un                = j((n-m0)+1, 2, .);

/*  init:          Number of observations in subset when the search
                   is started */
if n<40 then init=p+1;
else init=min(3*p+1,floor(0.5*(n+p+1)));


env_quantiles = {0.99999 0.9999 0.999 0.99 0.50 0.01 };


/*******************************/
/***   Forward search loop   ***/
/*******************************/

DO i= 1 TO (n-m0)+1;
      m = m0+(i-1);                                    /* Number of
observations in the current set */

      idx_in  = selected[i,1:m];
      idx_out = setdif((1:n),idx_in);     /* Indices of current not
selected observations */

      Xin  = x[idx_in,];
      Yin  = y[idx_in,];

      beta = SOLVE(Xin`*Xin,Xin`*Yin);
```

```
      iXpX = inv(Xin`*Xin);
      resid  = y - x * beta;                    /* simple residuals */

      if m >= init then do;

            /***  Do the search: monitor MDR and other statistics  ***/

            ssr = ssq(resid[idx_in]);

            if (nrow(Xin)-p)=0 then
                  s2=ssr/1;
            else
                  s2=ssr/(m-p);

            Hi = j(n,1,.);
            do k = 1 to n;
                  z = x[k,];
                  Hi[k,] = z * iXpX * z`;
                  /* External leverage */
            end;

            DR  = abs(resid/sqrt(s2#(1+Hi)));
            /* Deletion residuals */
            if m < n then
                  if ncol(idx_out) > 0 & m<n then do;
                        mdr_idx = (DR[idx_out])[>:<];
                        mdr[m,1] = (DR[idx_out])[mdr_idx];
            /* Minimal deletion residual of observations not in the
subset */
                        idx_out = setdif(idx_out,idx_out[mdr_idx]);
                  end;
      end;

      if n >= m + 1 then do;

            /***  Calculate next subset of size m + 1  ***/

            call sortndx(idx_sorted, abs(resid));

            if n > m + 1 then
                  selected[i+1,] =  (idx_sorted[1:(m+1)])` || j(1,n-m-
1,.);       /* Select observations */
            else
                  selected[i+1,] =  (idx_sorted[1:(min(m+1,n))])`;
            /* Last step */

            entered_search = rowvec(setdif(selected[i+1,],
selected[i,]));

            if ncol(entered_search) > ncol(Un) then do;
                  Un = Un || j(nrow(Un), ncol(entered_search) -
ncol(Un), .);
            end;

            if m > init then
                  Un[i,1:ncol(entered_search)]=entered_search;
```

```
      end;

end;


/*******************************/
/***   Check for signal      ***/
/*******************************/

CALL FSSignalDetection(
            mdr                     /* output */,
            n                       /* number of observations not
removed */,
            init              /* size of subset when search starts */,
            p                       /* number of variables */,
            0                       /* debug flag */,
            10                      /* h_min */,
            "DELETION"        /* method*/,
            output_string     /* output */,
            md                      /* output */,
            ""                      /* output data set name */,
            x                       /* data */,
            y                       /* data */,
            "x"               /* independent varnames */,
            last_env          /* output */,
            ""                      /* output data set name */,
            "CLASSIFY"        /* options */,
            m0                      /* size of initial subset */,
            1                       /* number of observations added at
each step */,
            selected          /* matrix of which observations where in
subset at each step */,
            1:n               /* valid_observations_index */,
            ""                      /* id_varname */,
            1:n               /* ID to be printed */,
            n                       /* original number of observations
*/,
            outlier_classification /* output */,
            merged_obs_nr     /* point mass contaminations */,
            .                       /* Bonferonni level, use
default*/);


/*******************************/
/***   Plots                 ***/
/*******************************/

n_axis = floor((loc(Un)-1)/ncol(Un))+1;
obs_nr = Un[loc(Un)]`;

first_obs = setdif(1:n, obs_nr); /* included before FS began */
if ncol(first_obs) > 0 then
      n_axis = n_axis || j(1,ncol(first_obs), 1); /* pretend these
observations were added in the first step */
obs_nr = obs_nr || first_obs;

declare DataObject dobj;
```

```
dobj = DataObject.Create("Forward search data object");
dobj.AddVar("m", "Number of observations in the subset", mm[n_axis]);
dobj.AddVar("model", "Car model", model[n_axis]);
dobj.SetRoleVar(ROLE_LABEL, 'model');
dobj.AddVar("original_obs_nr", "Observation number", obs_nr);
dobj.AddVar("outlier_classification", "Outlier classification",
outlier_classification[obs_nr]);
dobj.AddVar("mdr", "Minimum deletion residual",mdr[mm[n_axis]]);
dobj.SetMarkerShape(loc(outlier_classification[obs_nr]), MARKER_X );

/* Add x and y information of observations that were once MDR */
dobj.AddVar(dependent_varname, dependent_varname, y[obs_nr]);
do k = 1 to ncol(independent_varnames);
      dobj.AddVar(independent_varnames[k], independent_varnames[k],
x[obs_nr,k]);
end;

/***   Forward plot fo the minimum deletion residuals of observations
not in the subset   ***/
run FSRenvmdr(env_quantiles`, init, n, p, ENV);

declare LinePlot plot;
plot = LinePlot.Create( dobj, "m", "mdr", 0);

call FSRmdrplot( dobj, ENV, init, n, md, plot, output_string);
plot.SetWindowPosition( 50, 0, 50, 50 );

/***   Scatter Row plot   ***/
call CreateScatterRow(dobj, independent_varnames, dependent_varname);
```