LinRegOutliers: A Julia package for detecting outliers in linear regression

Mehmet Hakan Satman¹, Shreesh Adiga², Guillermo Angeris³, and Emre Akadal⁴

1 Department of Econometrics, Istanbul University, Istanbul, Turkey 2 Department of Electronics and Communication Engineering, RV College of Engineering, Bengaluru, India 3 Department of Electrical Engineering, Stanford University, Stanford, California, USA 4 Department of Informatics, Istanbul University, Istanbul, Turkey

Summary

LinRegOutliers is a Julia package that implements a number of outlier detection algorithms for linear regression. The package also implements robust covariance matrix estimation and graphing functions which can be used to visualize the regression residuals and distances between observations, with many possible metrics (e.g., the Euclidean or Mahalanobis distances with either given or estimated covariance matrices). Our package implements many algorithms and diagnostics for model fitting with outliers under a single interface, which allows users to quickly try many different methods with reasonable default settings, while also providing a good starting framework for researchers who may want to extend the package with novel methods.

State of the field

In linear regression, we are given a number of data points (say, \( n \)) where each data point is represented by a vector \( x_i \), with \( p \) entries, and a dependent variable that corresponds to each of these data points, represented by the scalar \( y_i \), for \( i = 1, 2, \ldots, n \). We then seek to find a linear model which best describes the data (up to some error term, \( \epsilon_i \)):

\[
y_i = \beta_1(x_i)_1 + \cdots + \beta_p(x_i)_p + \epsilon_i,
\]

where \( \beta_1, \ldots, \beta_p \) are the \( p \) unknown parameters. We will assume that the \( \epsilon_i \) are independent and identically-distributed (i.i.d.) error terms with zero mean. Note that, if \( (x_i)_1 = 1 \) for all \( i = 1, \ldots, n \), this is equivalent to having an intercept term given by \( \beta_1 \).

We can write this more conveniently by letting \( X \) be the design matrix of size \( n \times p \), whose \( i \)th row is given by the vectors \( x_i \) (where \( (x_i)_1 = 1 \) if the model has an intercept), while \( y \) is an \( n \)-vector of observations, whose entries are \( y_i \), and similarly for \( \epsilon \):

\[
y = X\beta + \epsilon.
\]

The usual approach to finding an estimate for \( \beta \), which we call \( \hat{\beta} \), is the Ordinary Least Squares (OLS) estimator given by

\[
\hat{\beta} = (X^TX)^{-1}X^Ty,
\]

which is efficient and has good statistical properties when the error terms are all of roughly the same magnitude (i.e., there are no outliers). On the other hand, the OLS estimator is very sensitive to outliers: even if a
single observation lies far from the regression hyperplane, OLS will often fail to find a good estimate for the parameters, \( \beta \).

To solve this problem, a number of methods have been developed in the literature. These methods can be roughly placed in one or more of the five following categories: diagnostics, direct methods, robust methods, multivariate methods, and visual methods. Diagnostics are methods which attempt to find points that significantly affect the fit of a model (often, such points can be labeled as outliers). Diagnostics can then be used to initialize direct methods, which fit a (usually non-robust) model to a subset of points suspected to be clear of outliers; remaining points which are not outliers with respect to this fit are continually added to this subset until all points not in the subset are deemed outliers. Robust methods, on the other hand, find a best-fit model by approximately minimizing a loss function that is resistant to outliers. Some of the proposed methods are also multivariate methods, which can accommodate obtaining robust location and scale measures of multivariate data. Visual methods generally work on the principle of visualizing the statistics obtained from these mentioned methods. As an example, the method \texttt{mveltsplot} constructs a 2-D plot using robust distances and scaled residuals obtained from \texttt{mve} and \texttt{lts} which are multivariate data and robust regression methods, respectively. Many direct and robust methods for regression select an initial basic or clean subset of observations using the results of diagnostics and methods for multivariate data. This is why methods that are not directly related to regression are included in the package.

**Statement of need**

In practice, many of the proposed methods have reasonable performance and yield similar results for most datasets, but sometimes differ widely in specific circumstances by means of masking and swamping ratios. Additionally, some of the methods are relatively complicated and, if canonical implementations are available, they are often out of date or only found in specific languages of the author’s choice, making it difficult for researchers to compare the performance of these algorithms on their datasets.

We have reimplemented many of the algorithms available in the literature in Julia (Bezanson et al., 2017), an open-source, high performance programming language designed primarily for scientific computing. Our package, LinRegOutliers, is a comprehensive and simple-to-use Julia package that includes many of the algorithms in the literature for detecting outliers in linear regression. The implemented Julia methods for diagnostics, direct methods, robust methods, multivariate methods, and visual diagnostics are shown in Table 1, Table 2, Table 3, Table 4, and Table 5, respectively.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Reference</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hadi Measure</td>
<td>(Chatterjee &amp; Hadi, 2015)</td>
<td>hadimeasure</td>
</tr>
<tr>
<td>Covariance Ratio</td>
<td>(Belsley et al., 2005)</td>
<td>covratio</td>
</tr>
<tr>
<td>DFBET A</td>
<td>(Belsley et al., 2005)</td>
<td>dfbeta</td>
</tr>
<tr>
<td>DFFIT</td>
<td>(Belsley et al., 2005)</td>
<td>dffit</td>
</tr>
<tr>
<td>Mahalanobis Distances</td>
<td>(Mahalanobis, 1930)</td>
<td>mahalanobisSquaredMatrix</td>
</tr>
<tr>
<td>Cook Distances</td>
<td>(Cook, 1977)</td>
<td>cooks</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Reference</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ransac</td>
<td>(Fischler &amp; Bolles, 1987)</td>
<td>ransac</td>
</tr>
</tbody>
</table>

### Table 3: Robust Methods

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Reference</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Least Absolute Deviations</td>
<td>(Nobakhti et al., 2009)</td>
<td>lad</td>
</tr>
<tr>
<td>Least Absolute Trimmed Deviations</td>
<td>(Hawkins &amp; Olive, 1999)</td>
<td>lta</td>
</tr>
<tr>
<td>Least Median of Squares</td>
<td>(Rousseeuw, 1984)</td>
<td>lms</td>
</tr>
<tr>
<td>Least Trimmed Squares</td>
<td>(Rousseeuw &amp; Van Driessen, 2000)</td>
<td>lts</td>
</tr>
<tr>
<td>CM-97</td>
<td>(Chatterjee &amp; Mächler, 1997)</td>
<td>cm97</td>
</tr>
<tr>
<td>ga-lts</td>
<td>(Satman, 2012)</td>
<td>galts</td>
</tr>
<tr>
<td>Satman-2013</td>
<td>(Satman, 2013)</td>
<td>satman2013</td>
</tr>
<tr>
<td>Satman-2015</td>
<td>(Satman, 2015)</td>
<td>satman2015</td>
</tr>
<tr>
<td>CCF</td>
<td>(Barratt et al., 2020)</td>
<td>ccf</td>
</tr>
</tbody>
</table>

### Table 4: Multivariate Methods

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Reference</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hadi-1992</td>
<td>(Hadi, 1992)</td>
<td>hadi1992</td>
</tr>
<tr>
<td>Hadi-1994</td>
<td>(Hadi, 1994)</td>
<td>hadi1994</td>
</tr>
<tr>
<td>Minimum Volume Ellipsoid</td>
<td>(Van Aelst &amp; Rousseeuw, 2009)</td>
<td>mve</td>
</tr>
<tr>
<td>Minimum Covariance Determinant</td>
<td>(Rousseeuw &amp; Driessen, 1999)</td>
<td>mcd</td>
</tr>
</tbody>
</table>

### Table 5: Visual Methods

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Reference</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCH Plot</td>
<td>(Billor et al., 2006)</td>
<td>bchplot</td>
</tr>
<tr>
<td>MVE-LTS Plot</td>
<td>(Van Aelst &amp; Rousseeuw, 2009)</td>
<td>mveltsplot</td>
</tr>
<tr>
<td>Data Images</td>
<td>(Marchette &amp; Solka, 2003)</td>
<td>dataimage</td>
</tr>
<tr>
<td>Stalactite Plot</td>
<td>(Atkinson, 1994)</td>
<td>atkinsonstalactiteplot</td>
</tr>
</tbody>
</table>

### Installation and basic usage

LinRegOutliers can be downloaded and installed using the Julia package manager by typing

```
 julia> using Pkg
 julia> Pkg.add("LinRegOutliers")
```
in the Julia console. The regression methods follow a uniform call convention. For instance, a user can type

```
julia> setting = createRegressionSetting(@formula(calls ~ year), phones);
julia> smr98(setting)
Dict{String,Array{Int64,1}} with 1 entry:
  "outliers" => [15, 16, 17, 18, 19, 20, 21, 22, 23, 24]
```
or

```
julia> X = hcat(ones(24), phones[:, "year"]);
julia> y = phones[:, "calls"];
julia> smr98(X, y)
Dict{String,Array{Int64,1}} with 1 entry:
  "outliers" => [15, 16, 17, 18, 19, 20, 21, 22, 23, 24]
```
to apply `smr98` (Sebert et al., 1998) on the Telephone dataset (Rousseeuw, 1984), where `X` is the design matrix with ones in its first column. In this case, observations 15 to 24 are reported as outliers by the method. Some methods may also return additional information specific to the method which is passed back in a `Dict` object. For example, the `ccf` function returns a `Dict` object containing `betas`, `outliers`, `lambdas`, and `residuals`:

```
julia> ccf(X, y)
Dict{Any,Any} with 4 entries:
  "betas" => [-63.4816, 1.30406]
  "outliers" => [15, 16, 17, 18, 19, 20]
  "lambdas" => [1.0, 1.0, 1.0, 1.0, 1.0, ...]
  "residuals" => [-2.67878, -1.67473, -0.37067, -0.266613, ...]
```
Indices of outliers can be accessed using standard `Dict` operations like

```
julia> result = ccf(X, y)
julia> result["outliers"]
6-element Array{Int64,1}:
  15
  16
  17
  18
  19
  20
```

**Acknowledgements**

Guillermo Angeris is supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1656518.

**References**


