The Stata Journal (2010)
10, Number 3, pp. 331–338

bacon: An effective way to detect outliers in multivariate data using Stata (and Mata)

Sylvain Weber
University of Geneva
Department of Economics
Geneva, Switzerland
sylvain.weber@unige.ch

Abstract. Identifying outliers in multivariate data is computationally intensive. The `bacon` command, presented in this article, allows one to quickly identify outliers, even on large datasets of tens of thousands of observations. `bacon` constitutes an attractive alternative to `hadimvo`, the only other command available in Stata for the detection of outliers.

Keywords: st0197, bacon, hadimvo, outliers detection, multivariate outliers

1 Introduction

The literature on outliers is abundant, as proved by Barnett and Lewis's (1994) bibliography of almost 1,000 articles. Despite this considerable research by the statistical community, knowledge apparently fails to spill over, so proper methods for detecting and handling outliers are seldom used by practitioners in other fields.

The reason is likely that algorithms implemented for the detection of outliers are sparse. Moreover, the few algorithms available are so time-consuming that using them may be discouraging. Until now, `hadimvo` was the only command in Stata available for identifying outliers. Anyone who has tried to use `hadimvo` on large datasets, however, knows it may take hours or even days to obtain a mere dummy variable indicating which observations should be considered as outliers.

The new `bacon` command, presented in this article, provides a more efficient way to detect outliers in multivariate data. It is named for the blocked adaptive computationally efficient outlier nominators (BACON) algorithm proposed by Billor, Hadi, and Velleman (2000). `bacon` is a simple modification of the methodology proposed by Hadi (1992, 1994) and implemented in `hadimvo`, but `bacon` is much less computationally intensive. As a result, `bacon` runs many times faster than `hadimvo`, even though both commands end up with similar sets of outliers. Identifying multivariate outliers thus becomes fast and easy in Stata, even with large datasets of tens of thousands of observations.

2 The BACON algorithm

The BACON algorithm was proposed by Billor, Hadi, and Velleman (2000). The reader who is interested in details is referred to that original article, because only a brief presentation is provided here.

In step 1, an initial subset of m outlier-free observations has to be identified out of a sample of n observations and over p variables. Any of several distance measures could be used as a criterion, and the Mahalanobis distance seems especially well adapted. It possesses the desirable property of being scale-invariant—a great advantage when dealing with variables of different magnitudes or with different units. The Mahalanobis distance of a p-dimensional vector $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^T$ from a group of values with mean $\overline{x} = (\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_p)^T$ and covariance matrix S is defined as

$$d_i(\overline{x}, S) = \sqrt{(x_i - \overline{x})^T S^{-1} (x_i - \overline{x})} \ , \qquad i = 1, 2, \ldots, n$$

The initial basic subset is given by the m observations with the smallest Mahalanobis distances from the whole sample. The subset size m is given by the product of the number of variables p and a parameter chosen by the analyst.

Billor, Hadi, and Velleman (2000) also proposed using distances from the medians for this first step. This second version of the algorithm is also implemented in `bacon`. Distances from the medians are not scale-invariant, so they should be used carefully if the variables analyzed are of different magnitudes.

In step 2, Mahalanobis distances from the basic subset are computed:

$$d_i(\overline{x}_b, S_b) = \sqrt{(x_i - \overline{x}_b)^T S_b^{-1} (x_i - \overline{x}_b)} \ , \qquad i = 1, 2, \ldots, n \tag{1}$$

In step 3, all observations with a distance smaller than some threshold—a corrected percentile of a χ^2 distribution—are added to the basic subset.

Steps 2 and 3 are iterated until the basic subset no longer changes. Observations excluded from the final basic subset are nominated as outliers, whereas those inside the final basic subset are nonoutliers.

The difference in the algorithm proposed by Hadi (1992, 1994) is that observations are added by blocks in the basic subset instead of observation by observation. Thus some time is spared through a reduction of the number of iterations. Nevertheless, it is important to note that the performance of the algorithm is not altered, as Billor, Hadi, and Velleman (2000) and section 5 of this article show.

The reduction in the number of iterations is not the only source of efficiency gain. Another major improvement lies in the way `bacon` is coded. When `hadimvo` was implemented, Mata did not exist. Now, though, Mata provides significant speed enhancements to many computationally intensive tasks, like the calculation of Mahalanobis distances. I therefore coded `bacon` so that it benefits from Mata's power.

3 Why Mata matters for bacon

The `bacon` command uses Mata, the matrix programming language available in Stata since version 9. I explain here how Mata allows `bacon` to run very fast. This section draws heavily on Baum (2008), who offers a general overview of Mata's capabilities.

The BACON algorithm requires creating matrices from data, computing the distances using (1), and converting the new matrix-containing distances back into the data. Operations that convert Stata variables into matrices (or vice versa) require at least twice the memory needed for that set of variables, so it stands to reason that using Stata's matrices would consume a lot of memory. On the other hand, Mata's matrices are only views of, not copies of, data. Hence, using Mata's virtual matrices instead of Stata's matrices in `bacon` spares memory that can be used to run the computations faster.

Moreover, Stata's matrices are unsuited for holding large amounts of data, their maximal size being 11,000 × 11,000. Using Stata, it would not be possible to create a matrix $\mathbf{X} = (x_1, x_2, \ldots, x_i, \ldots, x_n)^T$ containing all observations of the database if the n were larger than 11,000. One would thus have to cut the $\mathbf{X}$ matrix into pieces to compute the distances in (1), which is obviously inconvenient. Mata circumvents the limitations of Stata's traditional matrix commands, thus allowing the creation of virtually infinite matrices (over 2 billion rows and columns). Thanks to Mata, I am thus able to create a single matrix $\mathbf{X}$ containing all observations to whatever n. I then use the powerful element-by-element operations available to compute the distances.

Mata is indeed efficient for handling element-by-element operations, whereas Stata ado-file code written in the matrix language with explicit subscript references is slow. Because the distances in (1) have to be computed for each individual at each iteration of the algorithm, this feature of Mata provides another important efficiency gain.

4 The bacon command

4.1 Syntax

The syntax of `bacon` is as follows:

`bacon` *varlist* [*if*] [*in*] `, generate(`*newvar1* [*newvar2*]`)` [`replace` `percentile(`#`)` `version(1|2)` `c(`#`)`]

4.2 Options

`generate(`*newvar1* [*newvar2*]`)` is required; it identifies the new variable(s) that will be created. Whether you specify two variables or one, however, is optional. *newvar2*, if specified, will contain the distances from the final basic subset. That is, specifying `generate(out)` creates a dummy variable `out` containing `1` if the observation is an outlier in the BACON sense and `0` otherwise. Specifying `generate(out dist)`

additionally creates a variable `dist` containing the distances from the final basic subset.

`replace` specifies that the variables *newvar1* and *newvar2* be replaced if they already exist in the database. This option makes it easier to run `bacon` several times on the same data. It should be used cautiously because it might definitively drop some data.

`percentile(#)` determines the $1 - \#$ percentile of the chi-squared distribution to be used as a threshold to separate outliers from nonoutliers. A larger # identifies a larger proportion of the sample as outliers. The default is `percentile(0.15)`. If # is specified greater than 1, it is interpreted as a percent; thus `percentile(15)` is the same as `percentile(0.15)`.

`version(1 | 2)` specifies which version of the BACON algorithm must be used to identify the initial basic subset in multivariate data. `version(1)`, the default, identifies the initial subset selected based on Mahalanobis distances. `version(2)` identifies the initial subset selected based on distances from the medians. In the case of `version(2)`, *varlist* must not contain missing values, and you must install the `moremata` command before running `bacon`.

`c(#)` is the parameter that determines the size of the initial basic subset, which is given by the product of # and the number of variables in *varlist*. # must be an integer. `c(4)` is used by default as proposed by Billor, Hadi, and Velleman (2000, 285).

4.3 Saved results

`bacon` saves the following results in `r()`:

Scalars

`r(outlier)`	number of outliers	`r(iter)`	number of iterations
`r(corr)`	correction factor	`r(chi2)`	percentile of the χ^2 distribution

5 bacon versus hadimvo

Let us now compare `bacon` and `hadimvo` considering two criteria: i) the set of observations identified as outliers and ii) the speed. We will see that both commands lead to similar outcomes (providing some tuning of the cutoff parameters) but that `hadimvo` is terribly slower. `bacon` thus outperforms `hadimvo` and should be preferred in any case.

First, let us use `auto.dta` to illustrate the similarity of the results obtained through both commands:

```
. webuse auto
(1978 Automobile Data)
. hadimvo weight length, generate(outhadi) p(0.05)
  (output omitted)
. bacon weight length, generate(outbacon) percentile(0.15)
  (output omitted)
. tabulate outhadi outbacon
```

Hadi outlier (p=.05)	BACON outlier (p=.15) 0	1	Total
0	72	0	72
1	0	2	2
Total	72	2	74

Both commands have identified the same two observations as outliers. The parameter (in the `p()` and the `percentile()` options) was set higher in `bacon` than in `hadimvo`. With a parameter of 5%, `bacon` would not have identified any observation as an outlier. It is the role of the researcher to choose the parameter that is best adapted for each dataset, but the default `percentile(0.15)` appears to bring sensible outcomes in any case and could always be used as a first benchmark.

With two-dimensional data, it is helpful to draw a scatterplot such as figure 1 that allows us to see where outliers are located:

```
. scatter weight length, ml(outbacon) ms(i) note("0 = nonoutlier, 1 = outlier")
```

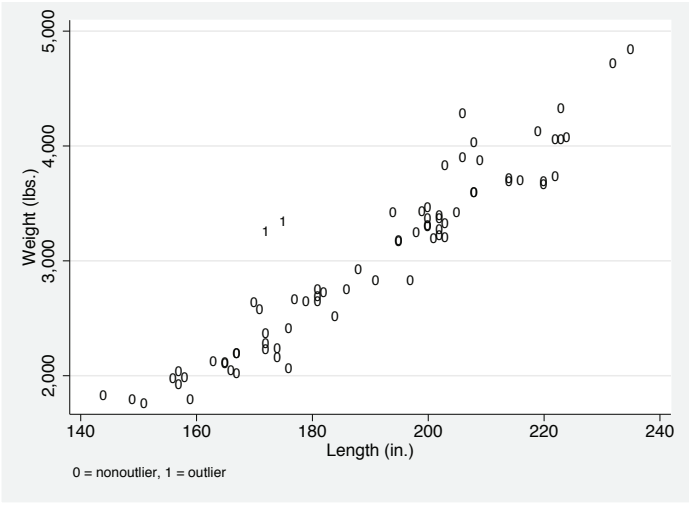

Figure 1. Scatterplot locating the observations identified as outliers

To compare the speeds of `bacon` and `hadimvo`, let us now use a larger dataset. Containing about 28,000 observations, `nlswork.dta` is sufficiently large to illustrate the point. Suppose we want to identify outliers with respect to the variables `ln_wage`, `age`, and `tenure`. If we did not have `bacon`, we would type

```
. webuse nlswork, clear
(National Longitudinal Survey. Young Women 14-26 years of age in 1968)
. hadimvo ln_wage age tenure, generate(outhadi) p(0.05)
Beginning number of observations:         28101
              Initially accepted:             4
             Expand to (n+k+1)/2:
```

At this point, your screen will remain idle. You might become worried and think your computer crashed, but in fact `hadimvo` is simply going to take some long minutes to run its many iterations. Remember, there are "only" 28,000 observations in this dataset. If you are patient enough, Stata will at last show you the outcome:

```
. hadimvo ln_wage age tenure, generate(outhadi) p(0.05)
Beginning number of observations:         28101
              Initially accepted:             4
             Expand to (n+k+1)/2:         14052
                Expand,  p = .05:         28081
              Outliers remaining:            20
```

Thanks to `bacon`, you now have a faster alternative. If you type

```
. bacon ln_wage age tenure, generate(outbacon) percentile(0.15)
Total number of observations:       28101
   BACON outliers (p = 0.15):          29
      Non-outliers remaining:       28072
```

the solution appears in only a few seconds! Again we can check that the set of identified outliers is pretty much the same in the two cases:

```
. tabulate outhadi outbacon
```

Hadi outlier (p=.05)	BACON outlier (p=.15) 0	1	Total
0	28,072	9	28,081
1	0	20	20
Total	28,072	29	28,101

Given the time `hadimvo` needs and the similarities between the outcomes, it seems clear that `bacon` is preferable.

Because there is no rule for the choice of `percentile()`, the practitioner might legitimately be willing to test several values and decide after several trials which set of observations to nominate as outliers. With `hadimvo`, such an iterative process is almost impracticable, unless you are particularly patient and have enough time in front of you. With `bacon`, on the other hand, completing the iterative process becomes readily feasible.

bacon has a replace option precisely to give the possibility of running the algorithm several times without having to add a new variable at each iteration. For the user wanting to try several percentile() values, replace will prove convenient:

```
. bacon ln_wage age tenure, generate(outbacon) percentile(0.1)
outbacon already defined
r(110);

. bacon ln_wage age tenure, generate(outbacon) percentile(0.1) replace
Total number of observations:       28101
   BACON outliers (p = 0.10):           6
      Non-outliers remaining:       28095

. bacon ln_wage age tenure, generate(outbacon) percentile(0.2) replace
Total number of observations:       28101
   BACON outliers (p = 0.20):         160
      Non-outliers remaining:       27941
```

6 Conclusion

"The two big questions about outliers are 'how do you find them?' and 'what do you do about them?'" (Ord 1996). The bacon command presented here provides an answer to the first of these questions. The answer to the second is beyond the scope of this article and is left to the consideration of the researcher.

No doubt, bacon renders the process of detecting outliers in multivariate data easier. Compared with hadimvo, the only other command devoted to this task in Stata, bacon appears to identify a similar set of observations as outliers. In terms of speed, bacon proves to be far faster. Hence, there is no apparent reason to use hadimvo instead of bacon.

Even though the bacon command provides a fast and easy way to identify potential outliers, a certain amount of judgment is always needed when deciding which cases to nominate as outliers and what to do with those observations. Most researchers simply discard outliers, but before you do so, keep in mind that something new and useful can often be learned by looking at the nominated cases.

7 References

Barnett, V., and T. Lewis. 1994. *Outliers in Statistical Data*. 3rd ed. Chichester, UK: Wiley.

Baum, C. F. 2008. Using Mata to work more effectively with Stata: A tutorial. UK Stata Users Group meeting proceedings.
http://ideas.repec.org/p/boc/usug08/11.html.

Billor, N., A. S. Hadi, and P. F. Velleman. 2000. BACON: Blocked adaptive computationally efficient outlier nominators. *Computational Statistics & Data Analysis* 34: 279–298.

Hadi, A. S. 1992. Identifying multiple outliers in multivariate data. *Journal of the Royal Statistical Society, Series B* 54: 761–771.

———. 1994. A modification of a method for the detection of outliers in multivariate samples. *Journal of the Royal Statistical Society, Series B* 56: 393–396.

Ord, K. 1996. Review of *Outliers in Statistical Data*, 3rd ed., by V. Barnett and T. Lewis. *International Journal of Forecasting* 12: 175–176.

About the author

Sylvain Weber is working as a teaching assistant in the Department of Economics at the University of Geneva in Switzerland. He is pursuing a PhD in the field of human capital depreciation, wage growth over the career, and job stability.