IOSR Journal of Mathematics (IOSR-JM)
e-ISSN: 2278-5728, p-ISSN: 2319-765X. Volume 13, Issue 4 Ver. II (Jul. – Aug. 2017), PP 76-84
www.iosrjournals.org

Data Analysis and Interpretation of the Problem

*S MD Riyaz Ahmad
Research Scholar, Department of Mathematics, Rayalaseema University, A.P, India
Corresponding Author: S MD Riyaz Ahmad

Abstract: *To analyze single data set-based benchmark experiments using R. Exploratory and inferential methods are used to compare the distributions and to finally set up mathematical (order) relations between the algorithms. I reviewed the theoretical framework of the current work for inference problems in benchmark experiments. Benchmarking UCI and Grasshopper domains presented two domain-based benchmark experiments. A large number of regression diagnostic tests showing the problems and solution of diagnostics and procedures and Outlier diagnostics and procedures have been proposed in the econometrics literature. MATLAB and Gauss code for implementing these methods can be found on many sources.*
Keywords: *Benchmarking, R, Grasshopper and UCI domain, Regression, Matlab, Outlier diagnostics.*

Date of Submission: 08-08-2017 Date of acceptance: 25-08-2017

I. Introduction

Benchmarking UCI and Grasshopper domains using R**,** I presented two domain-based benchmark experiments one for each application scenario we sketch in the introduction. The UCI domain serves as a domain for the scenario when comparing a newly developed algorithm with other well-known algorithms on a well-known domain. The Grasshopper domain serves as domain where we want to find the best candidate algorithm for predicting whether a grasshopper species is present or absent in a specific territory. The algorithm is then used as a prediction component in an enterprise application software system in R [1].

UCI domain:

The UCI domain is defined by 21 data sets binary classification problems available from Asuncion and Newman (2007). I am interested in the behavior of the six common learning algorithms linear discriminant analysis (lda, purple),k-nearest neighbor classifiers, (knn, yellow), classification trees (rpart,red), support vector machines (svm, blue), neural networks (nnet, green), and random forests (rf, orange); see all, for example, Hastie et al. (2009). The benchmark experiment is defined with B = 250 replications, bootstrapping as resampling scheme to generate the learning samples $\mathfrak{L}^b$ and the out-of-bag scheme for the corresponding validation samples $\mathfrak{T}^b$. Misclassification on the validation samples is the performance measure of interest[6]. The results are 21 fi 6 fi 1 estimated performance distributions $\hat{P}^j_{mk}$, the corresponding pair-wise comparisons based on mixed effects models and test decisions for a given fi = 0:05, and the resulting preference relations $\mathcal{R} = \{R_1, \ldots, R_{21}\}$. Note that we present selected results, the complete results are available in the supplemental material.

Grasshopper domain:

In this application example we are interested in finding the best algorithm among the candidate algorithm as a prediction component of an enterprise application software system. The domain is the domain of grasshopper species in Bavaria (Germany), the task is to learn whether a species is present or absent in a specific territory. The quadrants where a species is present are positively classified; as negatively classified quadrants we draw random samples from the remaining ones. If enough remaining quadrants are available we create a balanced classification problem, otherwise we use all remaining quadrants. I only used data sets with more than 300 positively classified quadrants {so, the Grasshopper domain is finally defined by 33 data sets[3-4].

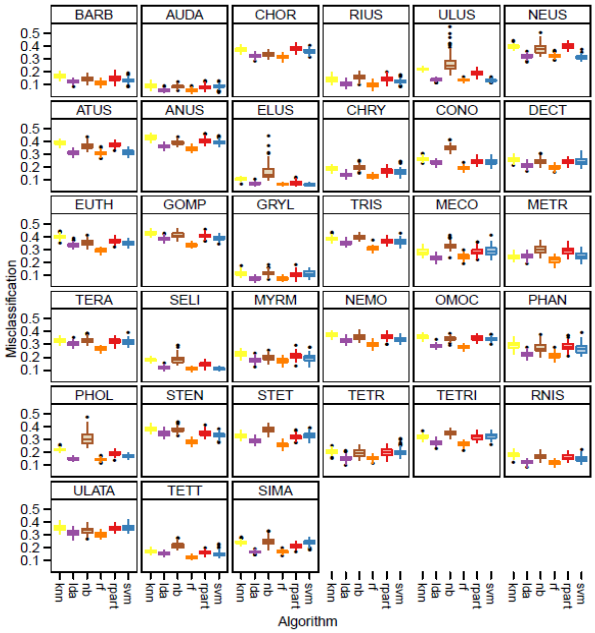

Figure 1: Trellis graphic with box plot of the candidate algorithms' misclassification error on the Grasshopper domain. Each data set is one grasshopper species.

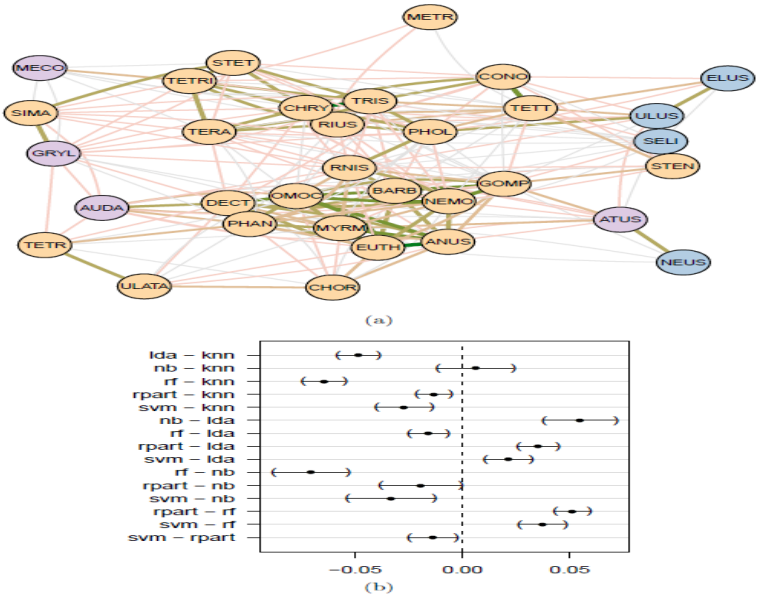

Figure 2: (a) The Grasshopper domain's benchmark summary graph; the color of the nodes indicate the algorithm with the minimum median misclassification error. (b) The Grasshopper domain's simultaneous 95% confidence intervals for multiple significant comparisons for a fitted linear mixed-effects model on the algorithms' misclassification errors.

The benchmark experiment is defined with B = 100 replications, bootstrapping as re-sampling scheme to generate the learning samples $\mathfrak{L}^b$, and the out-of-bag scheme for the corresponding validation samples $\mathfrak{T}^b$. Misclassification on the validation samples is the performance measure of interest. Note that I presented selected results; the complete results are available in the supplemental material. Above figure shows the Trellis plot with box plots for the six algorithms' mis- classification errors. I see that for most data sets the relative order of the candidate algorithms seems to be similar, but that the individual datasets are differently \hard" to solve. The locally computed preference relations $\mathcal{R} = \{R_1, \ldots, R_{33}\}$ contains no transitive relations; therefore, a visualization using the benchmark summary plot is not possible. Now, one possibility is to plot the asymmetric part of the transitive reduction for each of the 33 relations in a Trellis plot. However, such a plot is very hard to read and the benchmark summary graph provides a simpler visualization (albeit with less information).Figure 4.9a shows the bsgraph with the six smallest distance levels visible[7].

II. Regression Diagnostics and procedures

Simply stated, the collinearity problem is that near linear relations among the explanatory variable vectors tends to degrade the precision of the estimated parameters. Degraded precision refers to a large variance associated with the parameter estimates. Why should this be a matter of concern? Quite often the motivation for estimating an economic model is hypothesis testing to draw inferences about values of the model parameters. A large variance, or a lack of precision, may inhibit our ability to draw inferences from the hypothesis tests [9-10].

III. Suggestion of the problem

One way to illustrate the increase in dispersion of the least-squares estimates is with a Monte Carlo experiment. We generate a collection of y vectors from a model wherever the informative variables are moderately orthogonal, involving no close to linear dependencies. Different sets of y vectors are then generated from a model wherever the informative variables become more and more collinear. An examination of the variances of the $\hat{\beta}$ estimates from these experimental models should illustrate the increase in dispersion of the estimates arising from increasing the severity of the collinear relations between the explanatory variables. The specific experiment concerned victimization three informative variables during a model shown in (1).

$$Y = \alpha + \beta X_1 + \gamma X_2 + \theta X_3 + \in \qquad (1)$$

Initially, the three explanatory variables *X*1,*X*2,*X*3, were generated as random numbers from a uniform distribution. This ensures that they will be reasonably orthogonal or independent, not involved in any near linear dependencies. We followed a typical Monte Carlo procedure, producing 1000 different *y* vectors by adding a normally distributed random *"* vector to the *same* three fixed *X*'s multiplied times the parameters β, γ, θ, whose values were set to unity[13].

After estimating the parameters for the 1000 data sets we find the mean and variance of the distribution of the 1000 sets of estimates. The mean and variance of each parameter will characterize the distribution of outcomes from parameter estimation for a model that obeys the Gauss-Markov assumptions and contains no collinearity problems. This also provides a

benchmark against which to judge the impact on the distribution of estimates from near linear dependencies that we introduce next [14].

To create collinear relations we used the scheme shown in (2) where we no longer generate the *X*2 and *X*3 vectors independently.

$$X2 = X3 + u \qquad (2)$$

Instead, we generate the *X*2 vector from the *X*3 vector with an added random error vector *u*. Equation (2) represents *X*2 as a near linear combination of *X*3 where the strength of the linear dependency is determined by the size of the *u* vector[7-8]. To generate information sets with an increasing quantity of collinearity between X2 and X3, we tend to adopted the subsequent strategy:

1. first set the variance of the random normal error vector u at 1.0 and generate the X2 vector from the X3 vector.
2. Use the 3 vectors X1,X2,X3 to get a group of 1000 Y vectors by adding the precise same " vector that we have a tendency to utilized in the benchmark generation to those three mounted X's. The virtue of victimization the " vector from the benchmark is that, we have a tendency to hold the noise within the information generation process constant. This should provide a ceteris paribus experiment where the only change between these 1000 *Y* vectors and those from the benchmark generation is the collinear nature of the *X*2 and *X*3 vectors.
3. Two additional sets of 1000 *Y* vectors were generated in the same manner based on the same *X*3 and *X*1 vectors, but with two new versions of the *X*2 vector generated from *X*3. The new *X*2 vectors were produced by decreasing the variance of the random vector *u* to0.5 and 0.1, respectively. In summary, we have four sets of 1000 *Y* vectors, one benchmark set, where the three explanatory variables are reasonably independent, and three sets where the collinear relation between the *X*2 and *X*3 vectors becomes increasingly severe. The MATLAB code to produce this experiment is:

Application 1 Collinearity experiment

```
% ----- Application 1 Collinearity experiment
n=100; k=4; u1 = randn(n,1); u2=u1*0.5; u3 = u1*0.1;
x1 = [ones(n,1) rand(n,k-1)]; % orthogonal x's
x2 = [x1(:,1:2) x1(:,4)+u1 x1(:,4)]; % collinear set 1
x3 = [x1(:,1:2) x1(:,4)+u2 x1(:,4)]; % collinear set 2
x4 = [x1(:,1:2) x1(:,4)+u3 x1(:,4)]; % collinear set 3
ndraws = 1000; beta = ones(k,1);
bsave1 = zeros(ndraws,k); bsave2 = zeros(ndraws,k);
bsave3 = zeros(ndraws,k); bsave3 = zeros(ndraws,k);
```

```
for i=1:ndraws; % do 1000 experiments
e = randn(n,1);
y = x1*beta + e; res = ols(y,x1); b1save(i,:) = res.beta';
y = x2*beta + e; res = ols(y,x2); b2save(i,:) = res.beta';
y = x3*beta + e; res = ols(y,x3); b3save(i,:) = res.beta';
y = x4*beta + e; res = ols(y,x4); b4save(i,:) = res.beta';
end;
% compute means and std deviations for betas
mtable = zeros(4,k); stable = zeros(4,k);
mtable(1,:) = mean(b1save); mtable(2,:) = mean(b2save);
mtable(3,:) = mean(b3save); mtable(4,:) = mean(b4save);
stable(1,:) = std(b1save); stable(2,:) = std(b2save);
stable(3,:) = std(b3save); stable(4,:) = std(b4save);
% print tables
in.cnames = strvcat('alpha','beta','gamma','theta');
in.rnames = strvcat('beta means','bench','sigu=1.0','sigu=0.5','sigu=0.1');
in.fmt = '%10.4f';
mprint(mtable,in);
in.rnames = strvcat('stand dev','bench','sigu=1.0','sigu=0.5','sigu=0.1');
mprint(stable,in);
```

IV. The results of the experiment

It showing both the means and standard deviations from the distribution of estimates are:

beta means	alpha	beta		gamma	theta
benchmark	1.0033	1.0027	1.0047	0.9903	
sigu=1.0	1.0055	1.0027	1.0003	0.9899	
sigu=0.5	1.0055	1.0027	1.0007	0.9896	
sigu=0.1	1.0055	1.0027	1.0034	0.9868	
standard_dev	alpha	beta		gamma	theta
benchmark	0.3158	0.3285	0.3512	0.3512	
sigu=1.0	0.2697	0.3286	0.1025	0.3753	
sigu=0.5	0.2697	0.3286	0.2049	0.4225	
sigu=0.1	0.2697	0.3286	1.0247	1.1115	

Table 1. The Distribution of Estimates

A first point to note about the experimental outcomes is that the means of the estimates are unaffected by the collinearity problem. Collinearity creates problems with regard to the variance of the distribution of the estimates, not the mean. A second point is that the benchmark data set produced precise estimates, with standard deviations for the distribution of outcomes around 0.33. These standard deviations would result in t-statistics around 3, allowing us to infer that the true parameters are significantly different from zero. Turning attention to the standard deviations from the three collinear data sets we see a clear illustration that increasing the severity of the near linear combination between *X*2 and *X*3 produces an increase in the standard deviation of the resulting distribution for the γ and θ estimates associated with *X*2 and *X*3. The increase is about three-fold for the worse case where $\sigma_u^2 = 0.1$ and the strength of the collinear relation between *X*2 and *X*3 is the greatest.

A diagnostic technique presented of *Regression Diagnostics* by Belsley, Kuh, and Welsch (1980) is implemented in the function **bkw**. The diagnostic is capable of determining the number of near linear dependencies in a given data matrix *X*, and the diagnostic identifies which variables are involved in each linear dependency. This diagnostic technique is based on the Singular Value Decomposition that decomposes a matrix $X = UDV^1$, where *U* contains the eigenvectors of *X* and *D* is a diagonal matrix containing eigenvalues. For diagnostic purposes the singular value decomposition is applied to the variance-covariance matrix of the least-squares estimates and rearranged to form a table of *variance-decomposition proportions[11]*. The procedure for a *k* variable least-squares model is described in the following. The variance of the estimate $\hat{\beta}_k$ can be expressed as shown in (3).

$$\text{var}(\hat{\beta}_k) = \sigma_U^2 \sum_{J=1}^{K} V_{KJ}^2 / \lambda_J^2 \qquad (3)$$

The diagnostic value of this expression lies in the fact that it decomposes var($\hat{\beta}_k$) into a sum of components, each associated with one of the *k* singular values ,2*j* that appear in the denominator. Expression (4) expands the summation in (3) to show this more clearly.

$$\text{var}(\hat{\beta}_k) = \sigma_\epsilon^2 \{(V_{11}^2 / \lambda_1) + (V_{12}^2 / \lambda_2) + (V_{13}^2 / \lambda_3) + ... + (V_{1K}^2 / \lambda_K)\} \qquad (4)$$

Since small ,*j* are associated with near linear dependencies, an unusually large proportion of the variance of the coefficients of variables involved in the linear dependency will be concentrated in the components associated with the small λ_j . The table is formed by defining the terms φ and π shown in (5) and (6).

$$\phi_{ij} = \left(V_{ij}^2 / \lambda_j^2\right) \qquad (5)$$

$$\phi_i = \sum_{j=1}^{k} \phi_{ij}, \quad i = 1, ..., k$$

$$\pi_{ji} = \left(\phi_{ij} / \phi_i\right), \quad i, j = 1, ..., k \qquad (6)$$

The term π_{ji} is called a *variance-decomposition proportion*. These magnitudes are placed in a table as shown in Table 1.

Condition index	$\text{var}(\hat{\beta}_1)$	$var(\hat{\beta}_2)$	...	$var(\hat{\beta}_k)$
$\lambda_{max}/\lambda_{max}$	π_{11}	π_{12}	...	π_{1k}
λ_{max}/λ_2	π_{21}	π_{22}	...	π_{2k}
⋮	⋮	⋮	⋱	⋮
$\lambda_{max}/\lambda_{min}$	π_{k1}	π_{k2}	...	π_{kk}

Table 2: Variance-decomposition proportions table

It is shown in Belsley, Kuh and Welsch (1980) that a large value of the *condition index*, k(*X*) = λ_{max}/λ_i is associated with each near linear dependency, and the variates involved in the dependency are those with large proportions of their variance associated with large *k*(*X*) magnitudes. Empirical tests performed of Belsley, Kuh, and Welsch (1980) determined that variance-decomposition proportions in excess of 0.5 indicate the variates involved in specific linear dependencies. The joint condition of magnitudes for *k*(*X*) >30, and π_{ij} values >0.5 diagnose the presence of strong collinear relations as well as determining the variants involved. Table 3 shows an example of how the variance-decomposition proportions table might look for an actual data set. The example in the table would indicate that there exists one condition index, *k*(*X*), of 87 and another of 98. For these two condition indices we examine the variance proportions looking for those that exceed 0.5. We find that for *k*(*X*) = 87, two variance-proportions exceed 0.5 pointing towards a near linear relationship between the *x*1 and *x*5 variable. The *k*(*X*) = 98 also contains two variance-proportions that exceed 0.5 indicating the presence of a collinear relation involving *x*2 and *x*6. From Table 3 then we would conclude that two possible near linear relations existed in the data set.

$\kappa(X)$	x_1	x_2	x_3	x_4	x_5	x_6
1	.00	.00	.01	.01	.00	.00
28	.02	.03	.07	.03	.04	.03
58	.04	.00	.78	.02	.28	.01
60	.01	.02	.03	.76	.02	.22
87	.58	.28	.04	.10	.55	.07
98	.36	.66	.07	.09	.11	.67

Table 3: BKW collinearity diagnostics

The function allows a variable name vector and format as optional inputs. As a convenience, either a variable name vector with names for the variables in the data matrix *X* or one that includes a variable name for *y* as well as the variables in *X* can be used. This is because the **bkw** function is often called in the context of regression modeling, so we need only construct a single variable name string vector that can be used for printing regression results as well as labeling variables in the **bkw** output. As an example of using the **bkw** function to carry out tests for collinearity, the program below generates a collinear data set and uses the **bkw** function to test for near linear relationships.

Application 2 Using the bkw() function

```
% ----- Application 2 Using the bkw() function
n = 100; k = 5;
x = randn(n,k);
% generate collinear data
x(:,1) = x(:,2) + x(:,4) + randn(n,1)*0.1;
bkw(x);
```

V. The results of the program

They notice the close to linear relationship between variables one, two and four that we have a tendency to generated within the information matrix X. Belsley, Kuh, Welsch Variance-decomposition

K(x)	var1	var2	var3	var4	var5
1	0	0	0	0	0
15	0	0	0.26	0	0.4
17	0	0	0.24	0	0
20	0	0	0.47	0	0.59
31	1	0.99	0.03	0.99	0.01

Table 4. To Linear relationship between variables

A common corrective procedure for this problem is ridge regression, which is implemented by the function **ridge**. Ridge regression attacks the problem of small eigenvalues in the $X'X$ matrix by augmenting or inflating the smallest values to create larger magnitudes. The increase in small eigenvalues is accomplished by adding a diagonal matrix $\gamma I_k$to the $X'X$ matrix before inversion. The scalar term γis called the `ridge' parameter. The ridge regression formula is shown in (7).

$$\hat{\beta}_R = \left(X'X + \gamma I_K\right)^{-1} X'y \qquad (7)$$

Recall that $X'X$ is of dimension (*k* x *k*), where *k* is the number of explanatory variables in the model. Of the *k* eigenvalues associated with $X'X$, the smallest are presumably quite small as a result of the collinear relationship between some of the explanatory variables. To see how addition of the diagonal matrix γI_k to the $X'X$ matrix increases the smallest eigenvalues, consider using the singular value decomposition of $X'X$. This allows us to rewrite (7) as:

$$\hat{\beta}_R = \left(V'DV + \gamma I_K\right)^{-1} X'y \qquad (8)$$

Since γI_k is a diagonal matrix, containing zeros on the off-diagonal elements, adding this to the $V'DV$ matrices will only affect the elements of the diagonal matrix *D*. Noting this, we find that the ridge estimation formula can be written as in (9)

$$\hat{\beta}_R = \left(V'(DV + \gamma I_K)V\right)^{-1} X'y \qquad (9)$$

he diagonal matrix *D* from the Singular Value Decomposition contains the eigenvalues of the $X'X$ matrix, and equation (9) shows that the ridge parameter γis added to each and every eigenvalue on the diagonal of the matrix *D*. An expansion of the matrix ($D + \gamma I_k$) shown in (10) should make this clear.

$$(D + \gamma I_k) = \begin{pmatrix} \lambda_1 + \gamma & 0 & 0 & \dots & 0 \\ 0 & \lambda_2 + \gamma & 0 & \dots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & & & \lambda_k + \gamma \end{pmatrix} \qquad (10)$$

To illustrate how addition of the γparameter to the eigenvalues impacts the estimates, consider the following numerical example. As an example of using this function, consider the following program where we recover the ridge parameter determined using the Hoerl and Ken-nard formula, double it and produce a trace plot using values between θ= 0 and 2 θ. A value of θ= 0 represents least-squares estimates, and 2 θis twice the value we found using the Hoerl-Kennard formula.

Application 3 Using the rtrace() function

```
% ----- Application 3 Using the rtrace() function
n = 100; k = 5;
x = randn(n,k); e = randn(n,1); b = ones(k,1);
% generate collinear data
x(:,1) = x(:,2) + x(:,4) + randn(n,1)*0.1;
y = x*b + e;
% ridge regression
res = ridge(y,x);
theta = res.theta;
tmax = 2*theta;
ntheta = 50;
vnames = strvcat('y','x1','x2','x3','x4','x5');
rtrace(y,x,tmax,ntheta,vnames);
```

VI. The results from Ridge trace plot

To the extent that the parameter values vary greatly from those associated with values of θ= 0, we can infer that a great deal of bias has been introduced by the ridge regression. A graph produced by **rtrace**is shown in Figure 3, indicating a fair amount of bias associated with 3 of the 5 parameters in this example.

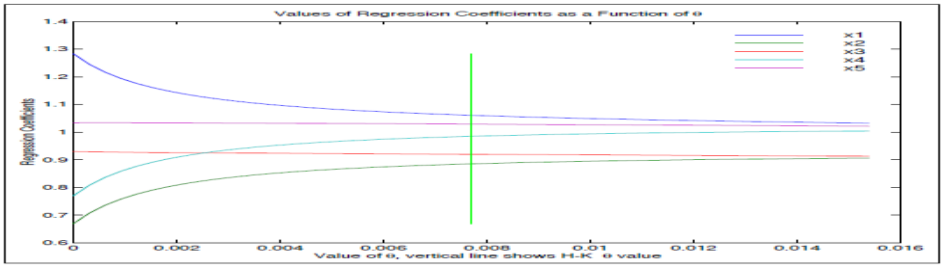

Figure 3: Ridge trace plot

The user can specify subjective prior information in the form of a normal prior for the parameters βin the model. Theil and Goldberger showed that this prior information can be expressed as stochastic linear restrictions taking the form:

$$c = R\beta + u, \qquad (11)$$

These matrices are used as additional *dummy or fake* data observations in the estimation process. The original least-squares model in matrix form can be rewritten as in (12) to show the role of the matrices defined above.

$$\begin{bmatrix} y \\ \dots \\ c \end{bmatrix} = \begin{bmatrix} X \\ \dots \\ R \end{bmatrix} \beta + \begin{bmatrix} \varepsilon \\ \dots \\ u \end{bmatrix} \qquad (12)$$

The partitioning symbol, (…), in the matrices and vectors of (12) designates that we are adding the matrix *R* and the vectors *c* and *u* to the original matrix *X* and vectors *y* and ε. These additional observations make it clear we are augmenting the *weak* sample data with our prior information.

VII. Outlier diagnostics and procedures

Outlier observations are best-known to adversely impact least-squares estimates because the aberrant observations generate large errors. The least-squares criterion is such that observations associated with large errors receive more weight or exhibit more influence in determining the estimates of β. A number of procedures have been proposed to diagnose the presence of outliers and numerous alternative estimation procedures exist that attempt to "robustify" or down weight aberrant or outlying observations. Function **dfbeta** produces a set of diagnostics discussed in Belsley, Kuh and Welsch (1980). They suggest examining the change in least-squares estimates $\hat{\beta}$ that arise when we omit each observation from the regression sample sequentially. The basic idea is

that eliminating an influential observation will produce a large change in the $\hat{\beta}$ estimates, allowing us to detect these observations graphically.

An example where we generate a data set and then artificially create two outliers at observations #50 and #70 is shown below. The graphical output from **pltdfb** in Figure shows a graph of the change in $\hat{\beta}$ associated with omitting each observation. We see evidence of the outliers at observations #50 and #70 in the plot.

Application 5 Using the dfbeta() function

```
% ----- Application 5 Using the dfbeta() function
n = 100; k = 4;
x = randn(n,k); e = randn(n,1); b = ones(k,1);
y = x*b + e;
% now add a few outliers
y(50,1) = 10.0; y(70,1) = -10.0;
vnames = strvcat('y','x1','x2','x3','x4');
res = dfbeta(y,x);
plt_dfb(res,vnames);
pause;
plt_dff(res);
```

VIII. The results from the regression data set

The figure 1.3 shows another diagnostic `dffits' produced by the function **dfbeta** that indicates how the fitted values change when we sequentially eliminate each observation from the regression data set. A similar function **diagnose** computes many of the traditional statistics from the regression diagnostics literature and prints this information for candidate outlier observations.

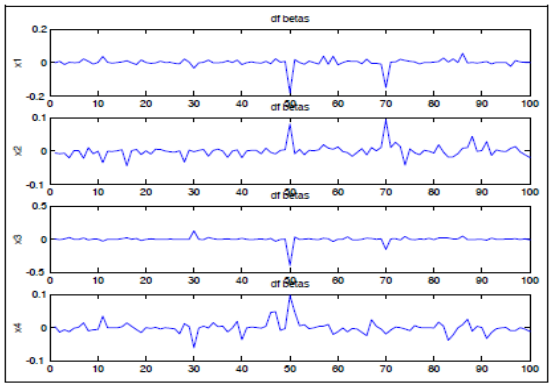

Figure 4. Dfbeta plots

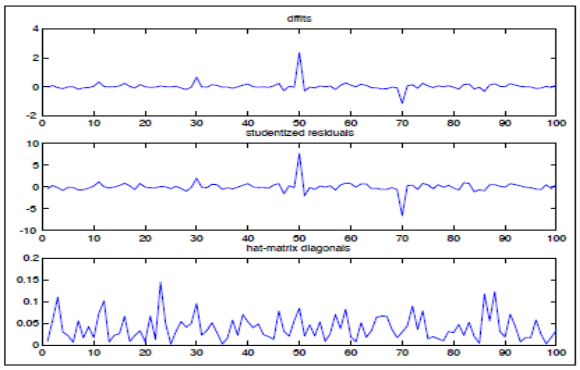

Figure 5. Dffits plots

The function incorporates four alternative weighting schemes that have been proposed in the literature on iteratively re-weighted regression methods. The routine **olst** performs regression based on an assumption that the errors are *t*-distributed rather than normal, which allows for \fat-tailed" error distributions.

Application 6 Using the pairs() function

```
% ----- Application 6 Using the pairs() function
n = 100;
y1 = randn(n,1);
y2 = 2*y1 + randn(n,1);
y3 = -2*y2 + randn(n,1);
y4 = randn(n,1); % uncorrelated with y1,y2,y3
y5 = randn(n,1).^4; % leptokurtic variable
y = [y1 y2 y3 y4 y5];
vnames = strvcat('apples','oranges','pairs','ha ha','leptoku');
pairs(y,vnames);
```

IX. The Results Pairwise Scatterplot

The documentation was altered to conform to that for other functions in the *Econometrics Toolbox* and a variable names capability was added to the function. The following program illustrates use of this function and Figure 6 shows the resulting pairwise scatterplot.

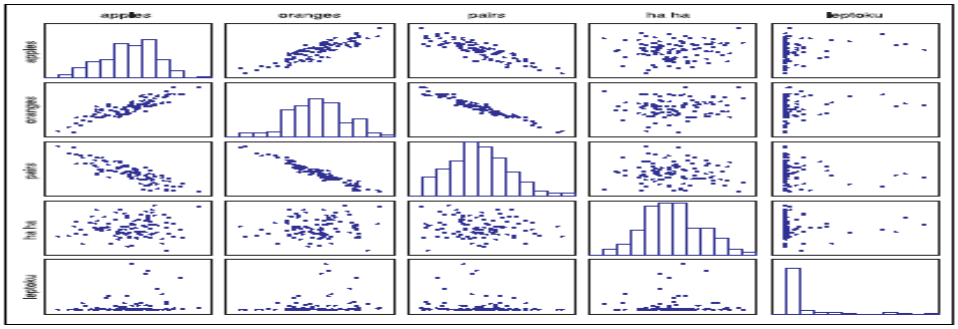

Figure6. Pairwise scatter plots

X. Conclusion

These methods to use on analyze single data set-based benchmark experiments. Exploratory and inferential methods are used to compare the distributions and to finally set up mathematical (order) relations between the algorithms. The UCI domain's simultaneous 95% confidence intervals for multiple Significant and relevant comparisons for a fitted linear mixed-effects model on the algorithms' misclassification errors. Archetypes of the domain present a different approach to analyze the performances pmbkj of a set of candidate algorithms ak on a domain D. Benchmarking UCI and Grasshopper domains presented two domain-based benchmark experiments. A large number of regression diagnostic tests showing the problems and solution of diagnostics and procedures and Outlier diagnostics and procedures have been proposed in the econometrics literature. MATLAB and Gauss code for implementing these methods can be found on many sources. The *Econometrics Toolbox* design allows these routines to be implemented and documented in a way that provides a consistent user interface for printing and plotting the diagnostics.

References

[1]. Gilks, W.R., S. Richardson and D.J. Spiegelhalter. 1996. Markov Chain Monte Carlo in Practice, (London: Chapman & Hall).
[2]. Goldfeld S.M. and R. E. Quandt 1973. \A Markov model for switching regressions", Journal of Econometrics, Vol. 1, pp. 3-16.
[3]. Hastings, W. K. 1970. \Monte Carlo sampling methods using Markov chains and their applications," Biometrika, Vol. 57, pp. 97-109.
[4]. Hoerl A.E., and R.W. Kennard, 1970. \Ridge Regression: Biased Estimation and Applications for Nonorthogonal Problems," Technometrics,Vol. 12, pp. 55-82.
[5]. Hoerl A.E., R.W. Kennard and K.F. Baldwin, 1975. "Ridge Regression: Some Simulations," Communications in Statistics, A, Vol. 4, pp.105-23.
[6]. Shoesmith, Gary L. 1995. \Multiple Cointegrating Vectors, error Correction, and Litterman's Model" International Journal of Forecasting,Vol. 11, pp. 557-567.
[7]. Simon, S.D., and J.P. LeSage, 1988a. \The Impact of Collinearity Involving the Intercept Term on the Numerical Accuracy of Regression, "Computational Statistics in Economics and Management Science, Vol.1 no. 2, pp. 137-152.
[8]. Simon, S.D., and J.P. LeSage, 1988b. \Benchmarking Numerical Accuracy of Statistical Algorithms," with Stephen D. Simon, Computational Statistics and Data Analysis, Vol. 7, pp. 197-209.
[9]. Kenneth J. Arrow. Social Choice and Individual Values. Yale University Press, second edition, 1963. ISBN 0300013647.
[10]. A. Asuncion and D.J. Newman. UCI machine learning repository, 2007. URL http://www.ics.uci.edu/~mlearn/MLRepository.html.
[11]. Douglas Bates and Martin Maechler. lme4: Linear Mixed-Effects Models using S4Classes, 2010. URL http://CRAN.R-project.org/package=lme4. R packageversion 0.999375-33.
[12]. Richard A. Becker, William S. Cleveland, and Ming-Jen Shyu. The visual design and control of Trellis display. Journal of Computational and Graphical Statistics, 5(2):123{155, 1996. doi: 10.2307/1390777.
[13]. Christopher M. Bishop. Neural Networks for Pattern Recognition. Oxford University Press, 1995. ISBN 0198538642.
[14]. Becker, R. A., Cleveland, W. S., and Shyu, M.-J. (1996). The visual design and control of Trellis display. Journal of Computational and Graphical Statistics, 5(2), 123–155.

S MD Riyaz Ahmad. "Data Analysis and Interpretation of the Problem ." IOSR Journal of Mathematics (IOSR-JM) , vol. 13, no. 4, 2017, pp. 76–84.