## Contents

Abstract iv  
Signed Statement vi  
Acknowledgements vii  

### 1 Introduction  
1.1 Expectation and Variance .......................... 1  
1.2 Estimator ........................................... 2  
1.3 The Cramér Rao Lower bound ....................... 3  
1.4 Generating Data and Fisher Consistency .......... 3  
1.5 The Normal Distribution .......................... 4  
1.6 Generating data from Tukey’s Model ............... 5  
1.6.1 Estimation ..................................... 6  
1.7 Univariate Outlier Detection ....................... 6  
1.8 Law of Large Numbers ............................ 8  
1.9 Relative Efficiency ............................... 10  
1.10 Onewsample relative to the Sample Mean ........ 11  
1.10.1 Simulation Results ............................ 11  
1.10.2 Discussion .................................. 11  

### 2 Multivariate Outlier Detection  
2.1 An Introduction to Multivariate Data ............... 13  
2.2 The Multivariate Normal Distribution ............... 13  
2.2.1 Departures from Normality ..................... 14  
2.3 The Mahalanobis Distance .......................... 14  
2.4 Swamping and Masking ............................ 16  
2.5 The Forward Search ............................... 16  
2.6 The Multivariate Adaptive Trimmed Likelihood Algorithm ... 19  
2.7 The Adaptive Trimmed Likelihood Algorithm in Action .... 22  
2.8 Generating Multivariate Data ....................... 23  
2.9 Size and Power of ATLA ......................... 25
2.10 The Cramér Rao Lower bound - Multivariate Case .......................... 26
2.11 Relative Efficiency of Multivariate Estimates .............................. 27
2.12 ATLA relative to the Sample Mean ........................................... 28
2.13 Student’s t-Distribution ......................................................... 30
2.14 Multivariate Student’s t-Distribution ....................................... 30
  2.14.1 Simulation Results .......................................................... 32
  2.14.2 Discussion ................................................................. 32
2.15 ATLA relative to the Forward Search ........................................ 35
  2.15.1 Simulation Results .......................................................... 35
  2.15.2 Discussion ................................................................. 37
2.16 ATLA relative to BACON Method ............................................. 38
  2.16.1 Simulation Results .......................................................... 38
  2.16.2 Discussion ................................................................. 41
  2.16.3 Simulation Results for Large Sample Sizes ............................ 41
  2.16.4 Discussion ................................................................. 44
2.17 ATLA relative to FSM ........................................................... 45
  2.17.1 Simulation Results .......................................................... 46
  2.17.2 Discussion ................................................................. 46
2.18 Computational Expense ......................................................... 47
  2.18.1 R Simulation Results ........................................................ 47
  2.18.2 Discussion ................................................................. 48
  2.18.3 MATLAB Simulation Results ............................................. 49
  2.18.4 Discussion ................................................................. 50

3 Conclusion ................................................................................. 51

A R and MATLAB Programs ....................................................... 53
  A.1 R Program Simulations .......................................................... 53
  .......................... Demonstration of Law of Large Numbers (Figure 1.1) .. 53
  A.1.2 Onesample relative to Sample Mean (Figure 1.2) ......................... 55
  A.1.3 ATLA Size and Power Calculations (Table 2.1) ............................ 58
  A.1.4 ATLA relative to Sample Mean (Figure 2.4) ............................... 60
  A.1.5 ATLA relative to Hadi’s FS (Tables 2.3 and 2.4) .......................... 63
  A.1.6 ATLA relative to BACON (Tables 2.5 and 2.6) ............................ 68
  A.1.7 ATLA relative to BACON for Large n (Tables 2.7 and 2.8) ............. 73
  A.1.8 R Computing Times (Figures 2.6) .......................................... 78
  A.2 MATLAB Program Simulations ................................................ 83
  A.2.1 ATLA relative to T-distribution (Table 2.2) ................................. 83
  A.2.2 ATLA relative to T-distribution (Table 2.5) ................................. 85
  A.2.3 ATLA relative to FSM (Table 2.9) .......................................... 89
  A.2.4 MATLAB Computing Times (Figures 2.7) ................................. 91
A.3 Supplementary R Functions ...................................... 93
A.3.1 ATLA Function Modified for Planted Outliers ............. 93
A.3.2 ATLA Function Modified for Location Output ............. 98
A.3.3 Original Forward Search by Hadi (1994) adapted to R .... 103
A.4 Supplementary MATLAB Functions .............................. 106
A.4.1 ATLA Function ............................................. 106

B Supplementary Simulation Results ............................... 115
B.1 Supplementary Tables ........................................... 115
B.1.1 ATLA relative to Hadi’s FS (Table 2.4) ..................... 115
B.1.2 ATLA relative to BACON (Table 2.6) ....................... 116
B.1.3 ATLA relative to BACON for Large $n$ (Table 2.8) ........ 116
B.1.4 R Computing Times (Figures 2.6) .......................... 117
B.1.5 MATLAB Computing Times (Figures 2.7) .................. 119
Abstract

This thesis aims to make a comparison between the efficacy of three prominent multivariate outlier detection algorithms, and in addition compare the efficiency of the methods (methods with outliers removed) with common robust estimation methods. This is done by comparing estimates of their relative efficiency based on estimates of multivariate mean in Monte Carlo simulation. The number of samples generated for these simulations is chosen to ensure reliable estimates yet avoid excessive computing times. Efficacy in comparison of multivariate outlier detection is judged by examining performance in terms of power in finding outliers in simulated data from contaminated distributions, as well as size for uncontaminated distributions, as well as looking at power in real-world data sets with known or planted outliers. Comparisons of time elapsed for various routines are briefly investigated, albeit on an ad hoc basis.

A particular motivation for this study is the focus on a specific adaptive method known as the adaptive trimmed likelihood algorithm (ATLA). ATLA is the multivariate version of a method which developed out of adaptive univariate location estimation first explored in Clarke (1994) and later related in terms of asymptotic theory in Bednarski and Clarke (2002). The asymptotic theory for the trimmed likelihood estimator was countenanced in Bednarski and Clarke (1993). A numerical routine using what is termed forward search and various comparisons made using ATLA are also described in Schubert (2005). The routine was later slightly modified by Robert Hammarstrand for completeness and is available in the supplementary materials of Clarke (2018) at the website:


Also available at that website is an algorithm called Onesample written by Brenton R Clarke and Betty Mouchel that implements the initial algorithm used to evaluate the estimator described in Clarke (1994) in the case of univariate estimation. For multivariate estimation, ATLA serves as an outlier detection method based on the use of the minimum covariance determinant (MCD) (Rousseeuw; 1983) used in an adaptive way.

Simulations and resulting output given in this thesis are presented with software in R and MATLAB, calling on new and pre-established functions available in downloadable packages. The code is presented in the appendix, together with supporting information that details the respective tasks of the as-
associated functions along with the various functions and/or packages that are required.
Signed Statement

This work contains no material which has been accepted for the award of any other degree or diploma in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text.

I give consent to this copy of my thesis, when deposited in the University Library, being available for loan and photocopying.

Signed: Andrew Grose        Date: December 19, 2019
I would like to thank my supervisor Dr Brenton Clarke for his continued guidance and support over both my undergraduate and honours courses. Brenton’s counsel was one of the main motivating factors behind undertaking honours as he has provided me with the original project suggestion as well as all the relevant material necessary to complete this project.

I would also like to acknowledge Dr. Daniel Schubert who passed away in 2007 shortly after completing his PhD. This thesis would not have been possible if it was not for the legacy of the multivariate adaptive trimmed likelihood algorithm left by Daniel.

I am also grateful for the guidance and advice from Dr. Mark Lukas in his role as Chair of the Mathematics and Statistics Honours Subcommittee.

Thanks are also due to Dr Ali Hadi who’s personal correspondence has assisted in the implementation of his original forward search procedure within R.

In addition I give thanks to Dr Marco Riani for his recommendation and assistance with regard to his procedure for finding multivariate outliers.

Lastly I would like to thank my family for their ongoing moral support and encouragement over the past few years.
Chapter 1

Introduction

Classically the problem in robust estimation is epitomised by the simple problem of estimating location from a symmetric distribution. Writers on robust estimation took the line of Tukey (1960) and compared allegedly robust estimators such as the trimmed mean and medians against the average or sample mean for data that were generated by a normal distribution and also contaminated normal distributions. See also Huber (1977, pp. 1–3) and Hampel et al. (1986, p. 29). The average or sample mean is supposedly the most efficient estimator when the data are normal. This is borne out by what is known as the Cramér Rao Lower bound.

Such an inequality exists in various forms relating to the number of parameters involved (singular or multivariate) as well as whether or not such estimators are biased or unbiased.

1.1 Expectation and Variance

Both the cases of the expectation and variance of a random variable are exceedingly important in various contexts throughout statistics and probability theory and consequently it is essential to introduce these terms preceding our exploration into the topic of estimation.

The theory of expectation exists in multiple different forms, however, here it is assumed that our random variable will have a continuous density, which is the case when the cumulative probability distribution is absolutely continuous, and then the expectation of a random variable $X$ involves taking a Lebesgue Integral. That is, it is possible to assert a mean value or expectation of a continuous random variable $X$ by taking the following integral assuming it
exists,

$$E[X] = \int_{-\infty}^{\infty} xf(x)dx$$  \hspace{1cm} (1.1)

where $f(x)$ is the probability density function (pdf). Note here that there will be instances where this mean is denoted $\mu = E[X]$, as is seen when introducing variance.

There may also be cases which require finding the expectation of a function of a continuous random variable, say $g(x)$. The function $g$ of a continuous random variable $X$ is in itself a random variable and by taking its expectation it follows that its mean value using the following formula given that $X$ has the pdf $f(x)$, is

$$E[g(X)] = \int_{-\infty}^{\infty} g(x)f(x)dx,$$  \hspace{1cm} (1.2)

again assuming it exists. Consequently the expectation can also be extended into forms which involve a measure of the dispersion of a random variable about the expected value which may formally be known as the variance. As a result it is common to see the variance expressed in the form of two terms of expectation as follows, $\text{Var}[X] = E[X^2] - E[X]^2$. Extending the above expectation results into our definition of the variance of a continuous random variable $X$, it naturally follows that

$$\text{Var}[X] = E[(X - \mu)^2] = \int_{-\infty}^{\infty} (x - \mu)^2 f(x)dx$$
$$\hspace{1cm} = \int_{-\infty}^{\infty} x^2 f(x)dx - \mu^2.$$

Often the notation $\sigma^2 = \text{Var}(X)$ is used for brevity. Both concepts of the expectation and variance play an important role in a range of applications involving estimation.

### 1.2 Estimator

Suppose interest is in estimating a certain unknown parameter, $\theta$ from a sample which is assumed to be a simple random sample from the population. The quality of this estimator, denoted say by $\hat{\theta}$ is able to be measured by the deviation or error between the estimate and the true parameter, that is $|\hat{\theta} - \theta|$.

In fact by taking the mean of the squared errors based on a random sample, $X = \{X_1, X_2, \ldots, X_n\}$ then we get what is called the mean squared error (MSE). The significance of which can be explained in the following interpretation,

$$\text{MSE} = E[(\hat{\theta} - \theta)^2]$$
$$\hspace{1cm} = \text{Var}(\hat{\theta}) + (E[\hat{\theta}] - \theta)^2.$$  \hspace{1cm} (1.3)
The first component in the above Equation (1.3) measures the variability of the estimator, \( \hat{\theta} \) in this case, while the second term is what is known as the bias of the estimator. That is,

\[
\text{Bias}[\hat{\theta}] = E[\hat{\theta}] - \theta.
\]

In consequence an estimator is said to be unbiased whenever \( \text{Bias} = 0 \) and thus \( E[\hat{\theta}] = \theta \) for all possible \( \theta \) in the parameter space, or in other words the mean of the statistic’s sampling distribution is equal to the population’s parameter. There are a number of variations in the notation of an estimator however for the duration of this thesis it is assumed that an estimator of a given parameter \( \theta \) is denoted by \( T \). We chose to use \( \hat{\theta} \) above for simplicity.

1.3 The Cramér Rao Lower bound

In the case of location estimation it is usual to be concerned with the form of the Cramér Rao inequality under mild regularity conditions that involve a singular unbiased estimator, \( T \) of \( \theta \), as follows

\[
\text{Var}(T) \geq \frac{1}{E \left[ \left( \frac{\partial}{\partial \theta} \log f \left( X | \theta \right) \right)^2 \right]}
\]

where \( f \left( X | \theta \right) \) denotes the probability density function for the sample \( X \). Here for example, \( E \) denotes the expectation with respect to the assumed parametric distribution that defines the data.

The denominator of the above equation is denoted Fisher Information after the renowned statistician Sir Ronald Aylmer Fisher.

1.4 Generating Data and Fisher Consistency

In order to test the efficacy of more than one statistic typically data are generated and compared using Monte Carlo techniques. For instance to represent an independent identically distributed normal sample \( X_1, \ldots, X_n \) where \( n \) is the sample size and each variable is having distribution \( F \), on generating a single sample \( x_1, \ldots, x_n \), most statistics can be represented in terms of the ordered sample \( y_1 \leq y_2 \leq \cdots \leq y_n \). There is a one to one correspondence between this ordered sample and what is known as the empirical cumulative distribution function,

\[
F_n(x) = \frac{\#x_i \leq x}{n}.
\]
In fact it is true that,

\[ y_i = F_n^{-1}(i/n). \]

Thus knowing the order statistics is equivalent to knowing the empirical distribution function. For this reason writing a statistic \( T_n(x_1, \ldots, x_n) \) can be equivalently written as \( T[F_n] \).

Through the use of a computer one is able to generate \( N \) samples of size \( n \), effectively giving us \( N \) empirical distributions \( \{F_n^{(j)}\}_{j=1}^N \), and equivalently \( N \) statistics \( T[F_n^{(j)}] \) which are all estimating the underlying statistic \( T[F] \).

In the parametric distribution set up where the underlying distribution is depending on some parameter \( \theta \in \Theta \) it is typically required that the estimator satisfies Fisher Consistency. That is \( T[F_\theta] = \theta \) for all \( \theta \in \Theta \). That is, knowing the population one knows the parameter \( \theta \) that describes the population.

### 1.5 The Normal Distribution

One of the most prominent families of distributions in all of statistics is what is referred to as the Normal or Gaussian Distribution, a symmetric continuous probability distribution which has a number of useful results that make it a fundamental component in the field of estimation theory.

The normal distribution has the following probability density function:

\[
f(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{1.4}\]

where \( \mu \) denotes the mean and \( \sigma^2 \), the variance. Often one will use the notation \( \mathcal{N}(\mu, \sigma^2) \) to denote a random variable with this distribution for given location and scale parameters \( \mu \) and \( \sigma^2 \) respectively.

Now consider the classic situation of generating data from the normal distribution with mean \( \mu \) and standard deviation \( \sigma = 1 \). Here,

\[
T_n(X_1, \ldots, X_n) = \frac{1}{n} \sum_{i=1}^n X_i = \int xdF_n(x)
\]

where \( F_n(x) \) is the empirical distribution of the random sample. The underlying distribution is

\[
\Phi(x - \mu) = \int_{-\infty}^{x} \phi(u - \mu)du
\]

where \( \phi \) corresponds to the standard normal distribution pdf, \( \phi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \) and \( \Phi(x) = \int_{-\infty}^{x} \phi(y) dy \).

Clearly the statistic,
\[ T[\Phi(x - \mu)] = \int_{-\infty}^{\infty} x\phi(x - u)dx = \mu, \]
whereby the statistic is Fisher Consistent.

As an estimate of the variance of \( T[F_n] \) it is seen that a useful estimate can be gained by looking at the sample variance from the Monte Carlo which would be \( \text{Var}\{T[F_n^{(1)}], \ldots, T[F_n^{(N)}]\} \) which should approximate what we know as \( 1/n \) which is the variance of the sample mean when the standard deviation of the data is one.

However as many practising statisticians will be quick to point out, data that are normally distributed is seldom the case in many real-world instances, albeit that the Central Limit Theorem sees that the normal distribution is often nominally achieved in practice when observing variables generated by the addition of many small increments over time. There of course may be a source of contamination that can be modelled as in the next chapter.

### 1.6 Generating data from Tukey’s Model

Consider a set \( \{X_1, \ldots, X_n\} \) of independent and identically distributed \( (i.i.d) \) random variables and let us define an epsilon contaminated normal to be one having a distribution,

\[
f_{\mu,\epsilon,\sigma_c}(x) = (1 - \epsilon) \phi(x - \mu) + \frac{\epsilon}{\sigma_c} \phi \left( \frac{x - \mu}{\sigma_c} \right)
\]

where

\[
F_{\mu,\epsilon}(x) = \int_{-\infty}^{x} f_{\mu,\epsilon}(y) dy \\
= (1 - \epsilon) \Phi(x - \mu) + \epsilon \Phi \left( \frac{x - \mu}{\sigma_c} \right).
\]

This model was creatively used to describe robustness or the lack thereof for the sample mean in comparison to say the 5% trimmed mean in Tukey (1960). In Tukey’s Model the parameter \( \sigma_c = 3 \) was used, i.e. \( \sigma_c^2 = 9 \).

The choice in the proportion of contamination, \( \epsilon \) is very much down to the discretion of the statistician, however Donoho and Huber (1983) argue that in real word circumstances it is possible for data to contain up to 50% contamination. While Hampel et al. (1986) found that anything up to 10% contamination is considered “realistic” as instances that consist of data that are in excess of this value tend to make it difficult to discern any real statistical information and consequently construct any valid inferences.
1.6.1 Estimation

Now using Tukey’s model presented above and the R programming environment it is possible to generate \( N = 10,000 \) samples of size \( n = 10, 20, 50 \) and 100 respectively, with \( \mu = 0 \), \( \sigma_c = 3 \) and with various contamination values, \( \epsilon \). From any individual sample it is possible to calculate an estimate for the sample mean,

\[
\bar{X} = \frac{1}{n} \sum_{i=1}^{n} X_i.
\]

On the other hand alternative estimates can be used. Here follows a description of an adaptive estimator investigated first in Clarke (1994). Using the function \textit{Onesample} created by Clarke and Mouchel (see supplementary materials in Clarke (2018)), it is possible to produce individual estimates for the Adaptive Trimmed Likelihood (ATLA) class of estimators. It can be noted that for the special case of trimming approximately one half of the data, the trimmed likelihood estimator becomes the Least Trimmed Squares estimator for location of Rousseeuw (1983).

Given an observed sample \( \{x_1, \ldots, x_n\} \), and letting residuals \( \epsilon(x - \mu) = (x - \mu) \), the Least Trimmed Squares estimator (LTS) is that \( \hat{\mu} \) that satisfies

\[
\sum_{i=1}^{h} (e^2(\hat{\mu}))_{i:n} = \min \sum_{i=1}^{h} (x_i - \mu)^2_{i:n}
\]

where \( (e^2)_{1:n} \leq \cdots \leq (e^2)_{n:n} \) are the ordered squared residuals and in the case of Rousseeuw (1983), \( h = \lfloor n/2 \rfloor + \lfloor (p + 1)/2 \rfloor \) which is chosen to maximise the finite sample breakdown point. The \textit{breakdown point} can be thought of as the point in which the level of contamination reaches a particular threshold/percentage before the estimator is unable to produce consistently correct results when it is subject to arbitrary contamination and thus it "breaks down". The \( p \) here corresponds to the amount of free parameters/independent variables, so for the case of univariate data, \( p = 1 \) and \( h = \lfloor n/2 \rfloor + 1 \).

A more detailed description of such as well as the adaptive trimmed likelihood estimator and its advantages can be found in Clarke (1994), Bednarski and Clarke (2002) and more recently in Clarke (2018, pp. 160–163).

1.7 Univariate Outlier Detection

Here we will briefly summarise the calculation/computation of the trimmed likelihood estimator at the normal distribution. For a general trimming proportion \( 0 \leq \alpha < \frac{1}{2} \) so that the potential number of observations trimmed is
Then for each individual subsample one would produce $n - h + 1$ means,

$$
g^{(1)} = \frac{1}{n} \sum_{i=1}^{h} y_i, \quad \ldots, \quad g^{(n-h+1)} = \frac{1}{n} \sum_{i=n-h+1}^{n} y_i
$$

and corresponding sums of squares,

$$
SQ^{(i)} = \sum_{i=1}^{h} \{y_i - g^{(i)}\}^2, \quad \ldots, \quad SQ^{(n-h+1)} = \sum_{i=n-h+1}^{n} \{y_i - g^{(n-h+1)}\}^2.
$$

Now the trimmed likelihood estimator pair $(\hat{g}, F_n)$, for which $SQ^{(j)}$ is the smallest are denoted as $(T_{\alpha}[F_n], \hat{\sigma}_n[F_n])$. See Clarke (2018, p. 156). The choice of $\alpha$ is done adaptively by choosing to trim a proportion of observations $\alpha = \hat{g}/n$ so that an estimator of asymptotic variance is minimised. For example,

$$
\nu(\hat{g}, F_n) = \min_{0 \leq \gamma \leq G^*(n)} \nu(\gamma/n, F_n).
$$

where $G^*(n)$ is typically chosen to be $[n/2]$ or something less. The functions $\nu$ touted are estimates of the asymptotic variance of the estimators so that

$$
\nu(\hat{\alpha}, F_n) = \frac{\hat{\alpha}^2 [F_n]}{\{1 - \alpha - \sqrt{2/\pi}z_{\alpha/2}\exp(-\frac{1}{2}z_{\alpha/2}^2)\}^2}. \quad (1.7)
$$

or, basing the estimate on the large sample consistent estimate of the asymptotic variance of the location estimator, denoted $V(\alpha, F)$ in Clarke (2018, pp. 157–158), a quantity minimise instead is

$$
\nu^*(\hat{\alpha}, F_n) = \frac{(1 - \alpha)\hat{\alpha}^2 [F_n]}{\{1 - \alpha - \sqrt{2/\pi}z_{\alpha/2}\exp(-\frac{1}{2}z_{\alpha/2}^2)\}^2}. \quad (1.8)
$$

Here $z_{\alpha/2}$ corresponds to the critical point of the standard normal distribution that is, $\Phi(z_{\alpha/2}) = 1 - \alpha/2$. The choice of $\nu$ in Equation (1.7) is used in the algorithm Onesample. The potential outliers are identified as those order statistics that are not in the span or optimal $(1 - \alpha)$-shorth leading to the trimmed likelihood estimator for $\alpha = \hat{\alpha}$. The actual observations leading to those order statistics are deemed outliers.

From the simulations using data generated from the standard normal distribution, in order to produce estimates for the variances of say the sample mean and/or the adaptive trimmed likelihood estimator, given by the statistic Onesample, we first note that through arguments of symmetry, that for each statistic $T$ we have $E[T] = 0$, that is assuming without loss of generality that $\mu = 0$. Consequently the variance of each statistic is equal to the expectation of the statistic squared, that is,
\[ \text{Var}[T] = E[T^2] - E[T]^2 = E[T^2]. \]

Each of the \( N = 10,000 \) statistics \( T[F_n]_l^2, \ l = 1, \ldots, N \) estimate \( E[T[F_n]]^2 \). It therefore follows by the law of large numbers (see below) that an estimate of the variance of each statistic is the average over all samples of all the squared values of the statistic.

Before continuing it is appropriate in this instance to briefly introduce a formal definition of the law of large numbers (LLN) due to its expansive importance in probability theory and statistics and its use in this thesis.

### 1.8 Law of Large Numbers

The theorem of law of large numbers essentially explains the influence of performing repeated experiments and the effect this has on a particular outcome or collective outcomes. As the name suggests it implies that the sample average, \( \bar{Y} \) of these outcomes obtained from a large number of experiments, \( N \) should be close to the expected value, \( \mu \) and converge to this value as \( N \to \infty \), that is,

\[
\bar{Y}_N \to \mu \quad \text{as} \quad N \to \infty
\]

Such a theorem exists in two different forms relating to this convergence; the strong law of large numbers (SLLN) and the weak law of large numbers (WLLN). The latter states that no matter how small a non-zero deviation bounded by a margin say, \( \epsilon \) from the mean there is a high probability so long as the sample size, \( N \) is sufficiently large, that the sample average, \( \bar{Y}_N \) is close to that of its expected value. That is,

\[
\lim_{N \to \infty} P \left( |\bar{Y}_N - \mu| > \epsilon \right) = 0.
\]

This is equivalent to

\[
\bar{Y}_N \xrightarrow{p} \mu \quad \text{as} \quad N \to \infty,
\]

where the \( p \) above the arrow indicates convergence of probability. While the strong law implies that as the sample size \( N \) tends to infinity the probability that this sample average converges to the expected value is 1 i.e.

\[
P \left( \lim_{N \to \infty} \bar{Y}_N = \mu \right) = 1
\]
or equivalently,

\[ \bar{Y}_N \overset{a.s.}{\longrightarrow} \mu \quad \text{as} \quad N \to \infty. \]

where the letters \( a.s \) denotes convergence almost surely. It is important to note that it is possible to show that the strong law of large numbers implies that of the weaker law/form.

As an example to demonstrate this convergence in action, the following entails producing a graphical plot, in the form of a line graph, which shows the influence of the number of trials on the average. The graph refers to coin flips or die rolls due to their probabilistic simplicity but of course such a law can be applied to much more complex problems.

![Law of Large Numbers for Rolls of a Dice](image)

**Figure 1.1:** Demonstration of the law of large numbers and its effect on the average proportion of sixes achieved as the number of dice rolls (i.e. trials) increase. Note that the average converges to the expected value of 1/6 indicated as a dashed line.

Now returning back to our example of the previous section, it follows by the law of large numbers (LLN) that \( \frac{1}{N} \sum_{i=1}^{N} T[F_n^{(i)}]_2 \) converges to \( E[T^2] = \text{Var}[T] \) as \( N \to \infty. \) This is illustrated using the LLN with \( Y_1 = T[F_n^{(1)}]_2, \ldots, Y_N = T[F_n^{(N)}]_2. \)

Now with \( N = 10,000 \) in this case it is possible to produce simplified estimates for the variances by the LLN, For instance \( \frac{1}{N} \sum_{i=1}^{N} Y_i \) converges to \( E[Y_i] = E[T^2] \) which in the above examples is the variance of the statistic \( T. \) Thus whether we choose as our estimate \( T_1[F_n] \equiv \bar{X} \) or the statistic \( T_2[F_n] \equiv \text{OneSample}[F_n] \) an estimate of the variance of that corresponding statistic is based on
\[
\frac{1}{N} \sum_{l=1}^{N} T[F_n^{(l)}]^2.
\] (1.9)

In both cases the subscript \( l \) refers to the evaluation of the statistic for the \( l^{th} \) sample of which there are \( N = 10,000 \) samples generated.

### 1.9 Relative Efficiency

In order to continue in our discussion it is informative to first introduce the idea of efficiency in regards to estimation theory. It is used essentially as a criterion of the quality or accuracy of an estimator.

We are able to calculate the efficiency of an unbiased estimator, \( T \) using the following formula;

\[
\text{eff}(T) = \frac{1}{\mathcal{I} (\theta)} \frac{\text{var}(T)}{\text{var}(T)}
\] (1.10)

where, \( \mathcal{I}(\cdot) \) denotes the *Fisher Information* which depends on the parametric distribution defined by the parameter, \( \theta \). See any classic text, for example Berry and Lindgren (1996), Hogg and Craig (1995), Mendenhall et al. (1986) or Rasch (2018).

Since this function involves that variance of an estimator \( T \), it is possible to show by the Cramér Rao Lower bound that the efficiency of an estimator is less than or equal to 1. In fact when an estimator has an efficiency of 1 then it is said to be *efficient*. While an estimator with efficiency 0.95 or 95% efficiency is said to be highly efficient, for example.

Additionally using this efficiency formula it is possible to produce a function for the *relative efficiency* which can be used to compare the efficiency of two estimators, \( T_i \) for \( i = 1, 2 \) using the ratio as follows

\[
\text{RellEff} = \frac{\text{eff}(T_1, T_2)}{\text{eff}(T_1)} = \frac{\text{eff}(T_1)}{\text{eff}(T_2)}.
\] (1.11)

Hence for the duration of this thesis it will be assumed that the wording \( T_1 \) “relative to” \( T_2 \) to denote the particular order shown above.

Using the estimate of variances and the efficiency formula above, it is possible to produce a function for the estimated Relative Efficiency of the *Onesample* estimator, or ATLA, relative to the mean. The use of the word function here is
because the estimated relative efficiency will depend on the value of $\epsilon$ which details the average proportion of contamination in the model (1.5). Though neither estimator, the sample average or ATLA, is efficient for positive $\epsilon$ it is possible to examine their relative efficiency, without resort to calculating the Fisher Information at the Tukey Model (1.5). Since by continuation of (1.11),

$$RellEff_\epsilon = \left( \frac{1/I(\theta)}{\text{var}(T_2)} \right) = \frac{\text{var}(T_2)}{\text{var}(T_1)}.$$ 

### 1.10 Onesample relative to the Sample Mean

#### 1.10.1 Simulation Results

Reverting back to our intended simulation, in order to investigate the influence of contamination it proves convenient to produce estimated relative efficiencies for the contamination values $\epsilon = 0, \ldots, 0.2$ (0 − 20%) initially in increments of 0.05 (5%). This will then be followed by smaller increments of 0.002 (0.2%) to investigate the behaviour of the estimates graphically.

*Table 1.1:* Estimated relative efficiencies of the *Onesample* or the ATLA statistic relative to the sample mean based off of a simulation of $N = 10,000$ univariate samples of size $n$ generated from Tukey’s Model with varying contamination values, $\epsilon$.

<table>
<thead>
<tr>
<th>$\epsilon$</th>
<th>0</th>
<th>0.05</th>
<th>0.1</th>
<th>0.15</th>
<th>0.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n=5$</td>
<td>0.8580</td>
<td>0.9964</td>
<td>1.0795</td>
<td>1.0974</td>
<td>1.1545</td>
</tr>
<tr>
<td>$n=10$</td>
<td>0.9658</td>
<td>1.1015</td>
<td>1.1624</td>
<td>1.1619</td>
<td>1.1710</td>
</tr>
<tr>
<td>$n=20$</td>
<td>0.9945</td>
<td>1.1389</td>
<td>1.1772</td>
<td>1.2080</td>
<td>1.1899</td>
</tr>
<tr>
<td>$n=50$</td>
<td>0.9989</td>
<td>1.1365</td>
<td>1.1982</td>
<td>1.2100</td>
<td>1.1798</td>
</tr>
<tr>
<td>$n=100$</td>
<td>0.9995</td>
<td>1.1472</td>
<td>1.1934</td>
<td>1.2085</td>
<td>1.1561</td>
</tr>
</tbody>
</table>

#### 1.10.2 Discussion

Looking at our results produced in Table 1.1 for contamination $\epsilon = 0$ it is possible to see that as the sample sizes, $n$ increase our relative efficiency estimates appear to converge to 1. In fact this is yet another clear example of the law of large numbers in action as it appears to be affecting the variance estimates (1.9) for each of the associated statistics. If one were to observe the variance estimates as the sample size $n \to \infty$, we would see that the variances gradually become closer to each other as they tend to zero and by definition the estimated
Figure 1.2: Plot of estimated relative efficiencies of the Onesample statistic relative to the sample mean based against various contamination values, $\epsilon$, from a simulation of $N = 10,000$ samples of univariate data of size $n$ from Tukey’s contamination model.

relative efficiency will also converge to 1. This concurs with the simulations in Clarke (1994) where it was noticed that for large $n$ and the model normal distribution the proportion of trimming any data set tended to zero.

One might also notice that as the proportion of contamination, $\epsilon$, increases the efficiency of our Onesample statistic quickly betters that of the sample mean (evident by a relative efficiency greater than 1). In fact so long as the samples sizes are of sufficient quantity the Onesample statistic out performs its counterpart almost as soon as contamination is introduced.

Perhaps highlighted more clearly in the form of graphical plots as presented in Figure 1.2, it can be seen that as the proportion of contamination, $\epsilon$ increases so does the relative efficiency only at a higher rate depending on how large the sample sizes, $n$, are. For larger $n$ the relative efficiency reaches larger values.

It also shows that for relatively large $n = 100$ the estimated relative efficiency appears to reach its maximum of approximately 120% around $0.11 < \epsilon < 0.14$ contamination before slightly decreasing for larger $\epsilon$ up to 20%. Furthermore the influence of the sample size on the relative efficiency appears to diminish for higher proportions of contamination.

Now that we have introduced basic concepts including the adaptive trimmed likelihood class of estimators and relative efficiency it would be appropriate to begin an exploration into multivariate outlier detection.
Chapter 2

Multivariate Outlier Detection

2.1 An Introduction to Multivariate Data

Multivariate data can be expressed in the form of an \( n \times p \) data matrix, \( n \) representing the number of observations and \( p \) being the number of variables or dimension. This data matrix can represented in the simplified form \( X = (X_1, X_2, \ldots, X_n)^T \) where \( X_i^T \) denotes a row vector of length \( p \) corresponding to observation \( i \) which is assumed to be a random variable.

2.2 The Multivariate Normal Distribution

Just like for one-dimensional (univariate) data that are assumed to follow a Normal Distribution with mean \( \mu \) and variance \( \sigma^2 \) there is a multi-dimensional generalisation of this called the Multivariate Normal Distribution with mean vector \( \mu \) to denote location and covariance matrix \( \Sigma \) for dispersion. The multivariate normal distribution can be expressed by the notation \( \mathcal{N}_p(\mu, \Sigma) \) for \( p \)-dimensional data.

Here the form of the mean vector is simply, \( \mu = (\mu_1, \mu_2, \ldots, \mu_p)^T \) where \( \mu_i \) are the component locations corresponding to the \( i \)-th variate while the \((i,j)\)-th element of the covariance matrix is calculated as follows

\[
\sigma_{ij} = \text{Cov}(X_i, X_j) = \mathbb{E} \left[ (X_i - \mathbb{E}[X_i]) (X_j - \mathbb{E}[X_j]) \right]
\]

where each \( X_k \) is assumed to be random variables with finite variance and expected value.
This symmetric matrix can also be expressed in the form,

\[
\Sigma = \begin{bmatrix}
\sigma_1^2 & \sigma_{12} & \sigma_{13} & \cdots & \sigma_{1p} \\
\sigma_{21} & \sigma_2^2 & \sigma_{23} & \cdots & \sigma_{2p} \\
\sigma_{31} & \sigma_{32} & \sigma_3^2 & \cdots & \sigma_{3p} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\sigma_{p1} & \sigma_{p2} & \sigma_{p3} & \cdots & \sigma_p^2
\end{bmatrix}
\]

where \( \sigma_i \) denotes the standard deviation of variable \( i \). For the purpose of simulating multivariate data this particular form of the covariance matrix is utilised through ease of assigning values to each of the \( \sigma_i \)'s.

### 2.2.1 Departures from Normality

As it is often the case in a lot of real-world scenarios, datasets may not necessarily be perfectly normally distributed. However, it is possible to break up departures from this model into four main groups:

- The presence of a single outlier;
- The presence of a group of outliers;
- The presence of two or more distinct groups; and
- General non-normality of the data (possibly requiring some sort of transformation/standardisation).

For the simulation component of this thesis is mostly concerned with the first two possibilities however later on there is a brief exploration a particular real-world dataset namely the Old Faithful geyser dataset which demonstrates the third potential scenario, that is, it consists of two distinct groups. In addition use is made of the Brain and Weight data which considers the log transformed brain and body weights of different animal species; three of which correspond to dinosaurs which are known to be outliers.

### 2.3 The Mahalanobis Distance

The Mahalanobis Distance which was first developed in 1930 by Mahalanobis, is a very important statistical measure in the context of multivariate analysis. In particular it is a very important application in multivariate outlier detection and consequently plays an integral part in a number of outlier detection methods including the ATLA. It can be thought of as a measure of distance between a point, \( x_i \in \mathcal{E}^p \) where \( \mathcal{E}^p \) is Euclidean-\( p \) space from a location \( C \) in the scale metric governed by \( S \) yielding,
Figure 2.1: When plotted [a)] the log transformed Brain and Weight data of various species appear to be correlated with the exception of three outliers shown in red. However evidence of masking is apparent when considering the magnitude of the associated mahalanobis distances in relation to the rest of the data [b]). The dashed line corresponds to the critical value of $(\chi^2_{2, 0.975})^{1/2} = 2.72$ for an $\alpha = 0.05$.

$$D_i(C, S) = \sqrt{(x_i - C)^T S^{-1} (x_i - C)}.$$ (2.1)

Note that it is common to use the notation $\hat{\mu}$ and $\hat{\Sigma}$ in lieu of $C$ and $S$ to denote the respective location and scale estimates. However for reasons that will become clear use is made the above notation in Equation (2.1) for our explanation of the forward search algorithm. Traditionally one would produce some sort of robust location and scale estimates obtained from the desired dataset which then assist in the calculation of mahalanobis distances for all or some of the individual observations $x_i$. It is here where these distances are either compared to one another either graphically evaluated or through a predetermined criteria or using a fixed threshold, so that one can determine which observations are deemed as outlying.

There are a number of different fixed thresholds that have been formulated in the previous literature, however the simplest form involves calculating the critical value from the $\chi^2_p$-distribution on $p$ degrees of freedom subject to an appropriate significance level, $\alpha$. It is worth noting that this is an approximate distribution and hence this is why a fixed threshold should not be taken as a strict criteria for deeming observations to be outliers. An example of this can be applied to the Mahalanobis distance plot seen in Figure 2.1b) above whereby an approximated critical region of $(\chi^2_{2, 0.975})^{1/2} = 2.72$ for an $\alpha = 0.05$ in this case falls quite closely to two separate observations.
Although in theory the mahalanobis distance may serve as a good quantitative measure of whether a certain observation may be outlying there may be several instances where it may not be as effective. For example there may be instances where certain observations result in large mahalanobis distances yet are not necessarily outlying and conversely certain observations with small mahalanobis distances may be outlying. These issues can be categorised into two main phenomena, namely *swamping* and *masking*.

### 2.4 Swamping and Masking

*Swamping* occurs when a small cluster of outliers may be inflating the estimate for the covariance matrix, and consequently attracting the location estimate away from the centroid of the majority data thus resulting in large mahalanobis distance values for observations that are within this group. This essentially results in good observations being incorrectly identified as outliers.

On the other hand when the presence of one outlier masks the appearance of another than this is known as *Masking*. Masking problems usually result in certain outlying observations yielding smaller valued mahalanobis distances and hence it might not be correctly deemed as an outlier. An example of this phenomenon can be seen in Figure 2.1a) for example, whereby three dinosaurs species appear to be clearly outlying. However such a statement would come under scrutiny when erroneously considering their masked mahalanobis distances in Figure 2.1b). One might look to the line of Hadi (1992) whom details the importance of acquiring more robust location and scale estimates to be used in the calculation of mahalanobis distances.

This is where we introduce the Forward Search which was originally developed to overcome these issues.

### 2.5 The Forward Search

The forward search was first developed by Hadi (1992, 1994) in a proposed procedure that involves the detection of multiple outliers in multivariate data. It involves a simple algorithmic procedure which incrementally cycles through a number of different possible subsets in order to obtain robust estimates for the location and scale subject to a pre-specified stopping criterion. Consider now an explanation of the forward search that then leads us into a discussion of a complimentary method presented by Schubert (2005), and Clarke and Schubert (2006). It must be emphasised that the information presented in this thesis regarding these methods is a simple exposition of the previous literature.
but is helpful in understanding the mechanism of the methods involved.

Hadi’s original algorithm proposed in Hadi (1992) begins by initially rearranging the \( n \) observations according to a suitably chosen robust distance of the form,

\[
D_i(C_R, S_R) = \sqrt{(x_i - C_R)^T S_R^{-1} (x_i - C_R)}, \quad \text{for } i = 1, \ldots, n. \tag{2.2}
\]

These robust estimates for location and scale, \( C_R \) and \( S_R \) are obtained by first calculating median-based location and scale estimates, \( D_i(C_M, S_M) \) where \( C_M \) is a vector with the coordinate-wise medians and,

\[
S_M = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - C_M)(x_i - C_M)^T.
\]

Now according to these estimates \( D_i(C_M, S_M) \) the observations are then rearranged in ascending order and using the weight function

\[
v_i = \begin{cases} 
1, & \text{if } i \leq \lfloor (n + p + 1)/2 \rfloor \text{ or,} \\
0, & \text{otherwise.}
\end{cases}
\]

We then compute distances based on Equation (2.2) by substituting \( C_R = C_V \) and \( S_R = S_V \) where,

\[
C_V = \frac{\sum_{i=1}^{n} v_i x_i}{\sum_{i=1}^{n} v_i} \quad \text{and} \quad S_V = \frac{\sum_{i=1}^{n} v_i (x_i - C_V)(x_i - C_V)^T}{\sum_{i=1}^{n} v_i - 1}.
\]

Once the observations have been rearranged then they are then divided into two subsets; the \textit{basic} subset containing the \( p + 1 \) observations with the smallest valued distances and the \textit{non-basic} subset containing \( n - p - 1 \) observations with the largest distances.

It is here where estimates for location \( C_b \) and scale \( S_b \) are calculated solely from observations that are within this \textit{basic} subset. These then assist in the calculation of the distances,

\[
D_i(C_b, S_b) = \sqrt{(x_i - C_b)^T S_b^{-1} (x_i - C_b)}, \quad \text{for } i = 1, \ldots, n.
\]

Note here that there may be instances where the basic subset covariance matrix is not of full rank, \( p \). In this situation then its inverse must be expressed in the form \( S_b^{-1} = V_b W_b V_b^T \). This involves computing the eigenvalues of \( S_b \) that is \( \lambda_1 \geq \cdots \geq \lambda_p = 0 \) and then \( V_b \) is a matrix containing the corresponding
set of normalised eigenvectors. Here $W_b$ is a diagonal matrix with diagonal elements,

$$w_j = \frac{1}{\max\{\lambda_j, \lambda_s\}}, \quad j = 1, \ldots, n,$$

and $\lambda_s$ denotes the smallest non-zero eigenvalue of $S_b$

Based off the distances $D_i(C_b, S_b)$, the observations are rearranged into ascending order similar to the initial step. The observations are then partitioned into two similar basic and non-basic subsets only with $r + 1$ and $n - r - 1$ observations respectively, where $r$ represents the number of observations in the previous basic subset.

These steps are repeated until either of the following stopping criterion are satisfied:

a) When $\min\{D_i(C_b, S_b); i \in \text{non-basic subset}\} \geq c_\alpha$ or,

b) When the basic subset contains $h$ observations; where $h = n - k$ if the number of outliers, $k$, are known otherwise $h = \lfloor (n + p + 1)/2 \rfloor$.

The choice of $h = \lfloor (n + p + 1)/2 \rfloor$ here is in line with Rousseeuw and van Zomeren (1990) which is based off of the proportion of good observations beyond which the estimator is known to breakdown. The critical value $c_\alpha$ is chosen such that $\mathbb{P}[\min\{D_i(C_b, S_b); i \in \text{non-basic subset}\} \geq c_\alpha | X \text{ contains no outliers}] = 1 - \alpha$. Note that since $c_\alpha$ depends on knowing the distribution of $D_i(C_b, S_b)$ which is difficult to derive, therefore until $c_\alpha$ is known we adopt criterion b).

It is important to note that it is possible for some observations from the basic subset to interchange with certain complement members due to the reordering. This also implies that there may be observations that appear in the original basic subset but may no longer be a member of this subset in later steps.

Once either of the stopping criterion has been met then the final robust distances are calculated,

$$D_i(C_b, S_b) = \sqrt{(x_i - C_b)^T (c_b S_b)^{-1} (x_i - C_b)}, \quad \text{for } i = 1, \ldots, n$$

where $c_b = c_{npr} m_j/\chi^2_{p:0.50}$ is the correction factor to obtain consistency when the data comes from a multivariate distribution. The $m_j$ represents the $100(h/n)$-th percentile of the $n$ values in the above equation and the small sample correction factor, $c_{npr} = \{1 + r/(n - p)\}^2$ where $r$ is the number of observations in the final basic subset.
Using these final robust distances, observations with large values are declared outliers either through a subjective evaluation based on a graphical plot or through a predefined critical point, $\sqrt{\chi_{p,1-\alpha/2}^2}$ for a chosen significance level, $\alpha$.

Figure 2.2 demonstrates the final robust distances of the log transformed Brain and Weight data obtained by a similar procedure to the one detailed above. As one can see the distances and resulting outliers are more representative of the underlying distribution.

![Final Robust Distances](image)

Figure 2.2: Final robust distances of the log transformed Brain and Weight data calculated through the procedure by Hadi (1994). The dashed line corresponds to the critical value of $\chi^2_{p, \alpha/n} = 12.65$ for $\alpha = 0.05$.

In a later publication Hadi (1994) proposed a slightly modified version of the above algorithm which improves upon its performance, the $R$ function for which can be seen in Appendix A.3.3. The original Forward Search algorithm is detailed here due to its significance in the field of multivariate outlier detection and as can be seen it has inspired a number of modifications including the multivariate ATLA.

### 2.6 The Multivariate Adaptive Trimmed Likelihood Algorithm

The adaptive algorithm proposed by Clarke and Schubert (2006) for the identification of multivariate outliers is introduced here. Those authors refer to the most important multivariate outlier detection techniques developed before which rely on the Minimum Covariance Determinant (MCD) (See Rousseeuw (1983) and Butler et al. (1993)). The adaptive trimmed likelihood algorithm (ATLA) utilises the minimum covariance determinant (MCD) in the estima-
tion of the multivariate location parameter and covariance matrix. Due to its propensity to identify outlying observations combined with its highly robust nature (which can be implied by its high breakdown point qualities as seen in later discussion) makes it the ideal estimator for use in identifying outliers in multivariate data. Although its calculation can seem quite complex we will demonstrate a roughly simplified version of its computation in our discussion of ATLA.

(See Rousseeuw (1983) and Rousseeuw and Leroy (1987) for a more detailed explanation of the MCD estimator).

One might notice here that the ATLA for multivariate data shares a lot of similarities to that of the \textit{Onesample} described in Section 1.7. This is because it is the multivariate extension of the univariate estimator.

The summary given here in order to define the ATLA can be found essentially in the book of Clarke (2018, pp. 176–178), the fuller details of which are given in Clarke and Schubert (2006).

Now suppose that there is a set of i.i.d observations \( \{X_i\}_{i=1}^n \) in \( p \)-dimensional Euclidean space \( \mathcal{E}^p \). Now let \( S_n \) be an arbitrary subset taken from this set of observations with size \( h_n = \lfloor n\gamma \rfloor \) where \( \gamma = 1 - \alpha \), for some \( \alpha \) corresponding to the amount of trimming required subject to \( 0.5 < \gamma \leq 1 \). It is from these subsets \( S_n \) that it is possible to define,

\[
\hat{\mathcal{M}}(S_n) = \frac{1}{h_n} \sum_{i \in S_n} X_i \quad \text{and} \quad \hat{\mathcal{S}}(S_n) = \frac{1}{h_n} \sum_{i \in S_n} (X_i - \hat{\mathcal{M}}(S_n))(X_i - \hat{\mathcal{M}}(S_n))^T.
\]

Which then leads to the ability to define a particular subset, \( \hat{S}_n \) for which attains the minimum value of the determinant of \( \hat{\mathcal{S}}(S_n) \), \( |\hat{\mathcal{S}}(S_n)| \), over all possible subsets, \( S_n \) of size \( h_n \). It is for this corresponding subset for which \( \hat{\mathcal{M}}(\hat{S}_n) \) yields the MCD estimator.

As in previous sections the notation \( T[F_n] \) is used to denote the estimate for location in the case of univariate estimation, in continuation of this theme we will use \( T[F_n] \) to estimate location for the multivariate case noting here the bold \( T \) denotes the vector functional. When dealing with multivariate data it is common to use elliptical probabilities of the form,

\[
\frac{1}{|\Sigma|^{1/2}} h^*(\mathbf{x} - \mu) \Sigma^{-1} (\mathbf{x} - \mu)
\]

where \( h^* \) such that \( \mathcal{E}^+ \rightarrow \mathcal{E}^+ \) is assumed to be non-increasing, yielding a unimodal density with mean vector, \( \mu \) and covariance matrix \( \Sigma \), the determinant
for which is assumed to be non-zero.

Now supposing that the data follows a multivariate normal distribution with mean zero and covariance matrix $I$, the identity of size $p$, then using the MCD estimator the sample covariance matrix converges with probability one ($wp1$) such that

$$\hat{\Sigma}_n[F_n] \xrightarrow{wp1} \rho(\gamma)I_p.$$  

Here $\rho(\gamma)I_p$ represents the expression for the asymptotic covariance matrix,

$$\rho(\gamma)I_p = \frac{1}{\gamma} \int_E (x - \mu)(x - \mu)^T dF(x)$$

a more detailed expression of which can be found in Clarke and Schubert (2006). Now referring back to the sample estimate for the multivariate mean $T[F_n]$ it follows that,

$$\sqrt{n}(T[F_n] - \mu) \overset{d}{\to} N(0, \kappa(\gamma)I_p)$$

As demonstrated in Clarke and Schubert (2006) the form of $\kappa(\gamma)$ reduces to,

$$\kappa(\gamma) = \frac{\rho(\gamma)}{\left(\frac{4\pi^{p/2}}{\Gamma(p/2)} \int_0^{r_{\gamma}} r^{p+1} \phi'(r^2) dr\right)^2}$$

where, $r_{\gamma} = \sqrt{\chi^2_{p,1-\alpha}}$, can be used to estimate the asymptotic variance of $T[F_n]$ assuming the data is taken from multivariate normal distribution. Here $\Gamma$ corresponds to the gamma function, $\Gamma(z) = \int_0^{\infty} x^{z-1} e^{-x} dx$, Re($z$) > 0 and $\phi(x) = (1/(2\pi))^{p/2} e^{-x/2}$.

Similar to minimising Equation (1.8) for univariate, in the case of multivariate data choosing $\gamma$ to minimise the function

$$\left|\kappa(\gamma)\hat{\Sigma}_n[F_n]\right| = \frac{|\hat{\Sigma}_n[F_n]|}{\left(\frac{4\pi^{p/2}}{\Gamma(p/2)} \int_0^{r_{\gamma}} r^{p+1} \phi'(r^2) dr\right)^{2p}}$$

which is referred to as Type 1 or $T1$ in Clarke and Schubert (2006).

Now by a result from Bednarski and Clarke (1993) it is possible to produce a Fisher consistent estimate for $\Sigma$ and again by choosing $\gamma$ to minimise the function
\[
\left| \frac{\kappa(\gamma)}{\rho'(\gamma)} (1 - \alpha) \hat{\Sigma}_\alpha[F_n] \right| = \frac{|(1 - \alpha) \hat{\Sigma}_\alpha[F_n]|}{\left( \frac{4\pi^{p/2}}{p!} \frac{1}{\Gamma(p/2)} \right) \int_0^\gamma r^{p+1} \phi(r^2) dr} \]  

(2.4)

for which is called \( T_2 \). One of these two objective function proposals \( T_1 \) or \( T_2 \) will then be used as a type of stopping criterion in the subsequent forward search, a decision for which is based on the sample size, \( n \). A guideline for this decision was one based on a series of Monte Carlo simulations to establish the efficacy of either objective function which can be found in the original literature, Schubert (2005) and Clarke and Schubert (2006). It was ultimately decided that the \( T_1 \) objective function would be used for data sets of size \( n < 30 \) and \( T_2 \) otherwise.

Following the evaluation of MCD estimates for location and scale, the algorithm then executes a Forward Search. Although this forward search algorithm does not have a particular stopping criterion unlike many of its counterparts it uses successive subsample assessments and optimisation of the associated statistic in order to arrive to an appropriate decision on the final sample.

Again it helps to relay the description of the forward search procedure applied to this particular algorithm as outlined in Clarke and Schubert (2006, pp. 357–358) to give an understanding of how outliers are identified.

### 2.7 The Adaptive Trimmed Likelihood Algorithm in Action

Before commencing our comparison of various multivariate outlier detection algorithms it would be appropriate to first demonstrate it’s ability on a real-world data set. Take for example a particular dataset available from the MASS package in \( R \) (Venables and Ripley; 2002) which considers waiting time between eruptions and the duration of the eruption for the Old Faithful geyser in Yellowstone National Park. Due to reasons which will become known it serves as a good reference dataset to test various outlier detection procedures such as the adaptive trimmed likelihood algorithm.

To demonstrate the results one would expect when performing the ATLA we will produce a graphical plot of the old faithful data highlighting those outliers that have been identified by the algorithm and compare its performance relative to a figure reproduced from Cerioli et al. (2018) which explores a similar estimation method. The method used here is referred to as S-estimation which will not be formally defined here instead one is recommended to look to Cerioli et al. (2018) for a detailed explanation.
As seen in Figure 2.3 separation of the groups appears to be better for the ATLA for multivariate data compared to that of the S-estimation for 50% breakdown point as the ATLA has identified 98 outliers compared with 82 using S-estimation. As well as exploring different choices in breakdown point Cerioli et al. (2018) incorporate a number of different methods including MM-estimation and hard trimmed estimates such as the minimum covariance determinant (MCD). This brings us to one of the key advantages of the ATLA for multivariate data and this is that one is not plagued with having to choose a particular value or threshold such as for the efficiency or subset size. An essential choice for which a small deviation/perturbation in said value might mean the difference between flagging an observation as an outlier when in actual fact it is an observation within the uncontaminated component population.

2.8 Generating Multivariate Data

Generating data from a particular given model serves as an essential step in the testing of certain statistical procedures, estimators or algorithms such as the adaptive trimmed likelihood outlined in the previous section. Similar to how we generated univariate contaminated data through multiple simulations of Tukey’s Model in Section (1.4) it is essential we do the same for the case of multivariate data. There are a number of different contamination model frameworks that exist for the generation of multivariate contaminated data. The multivariate Tukey-Huber Model is one such framework which is actually derived from the Tukey-Huber contamination model for univariate loca-
tion/scale as explored in previous sections. As a consequence it shares a lot of similarities including the form of equation,

$$X = (I - B)Y + BZ$$

the only difference being that for multivariate data it involves a series of $d$-dimensional vectors, $X, Y$ and $Z$ noting here the similarities with Equation (1.5) when $p = 1$. The matrix $B$ here is called the contaminated indicator matrix which is a diagonal matrix with elements $\{B_1, B_2, \ldots, B_d\}$ that satisfy Bernoulli random variables such that $\mathcal{P}(B_i = 1) = \epsilon_i$ and the $Z$ matrix is an arbitrary outlier generating distribution.

Supposing that the random vector $Y$ has density,

$$f_Y(y) = h((y - \mu_0)^T\Sigma_0^{-1}(y - \mu_0))$$

where $\mu_0$ represents the multivariate location vector that is to be estimated and $\Sigma_0$ is the covariance matrix.

There may be instances for which $Y, B$ and $Z$ are dependent however we will suppose that they are independent for the time being. There are various choices of the joint probability distribution one can impose on the elements of $B$ which give rise to different contamination models. For the sake of simplicity we will start with the most basic framework that is the fully independent contaminated model (FICM) which implies that $B_1, B_2, \ldots, B_p$ are fully independent i.e. $\mathcal{P}(B_1 = 1) = \cdots = \mathcal{P}(B_p = 1) = \epsilon_i$. The other case which we will briefly consider is the fully dependent contaminated model (FDCM) where $B_1, B_2, \ldots, B_p$ are fully dependent, that is $\mathcal{P}(B_1 = B_2 = \cdots = B_p) = 1$. This matrix form of the multivariate Tukey-Huber model, can be expressed in the more easy to follow form:

$$f(\mu, \Sigma, \epsilon) = (1 - \epsilon) \cdot N_p(\mu, \Sigma) + \epsilon \cdot N_p(\mu, \Sigma \cdot \sigma^2)$$

where $\mu$ denotes the mean vector, $\Sigma$ the covariance matrix generated from a multivariate normal distribution of dimension $p$ and $\epsilon$, the contamination. Here $\sigma^2$ denotes the contaminating covariance multiplier which is set as $\sigma = 3$ for consistency with the univariate Tukey-Huber Model. The procedure for generating data from this model will be similar to how univariate data is generated in Section (1.4). That is, a uniform random sample of size $n$ is generated from $U(0, 1)$ which we will denote $U_i$ then for $i = 1, \ldots, n$:

**Generate** observation $i$ from the distribution $N_p(\mu, \Sigma)$
Otherwise if $U_i \geq (1 - \epsilon)$:

**Generate** observation $i$ from the distribution $\mathcal{N}_p(\mu, \Sigma \cdot \sigma^2)$.

For our FDCM and FICM frameworks, without loss of generality, $\mu = 0$ since this simplifies the relative efficiency calculations.

Through our simulations that are explored in later sections use is made of the *mvrnorm* function in the MASS package within R (Venables and Ripley; 2002).

In addition to the above generating frameworks a method explored by Hadi (1994) called the *mean slippage model* is further investigated. This model involves planting $k$ outliers generated from the distribution $\mathcal{N}_p(cJ, I_p)$ together with the majority of $n - k$ observations from $\mathcal{N}_p(0J, I_p)$ where $J$ represents a vectors of one’s of length $p$ and $c$, an arbitrary integer.

### 2.9 Size and Power of ATLA

Before beginning our in-depth comparison of various outlier detection algorithms and how they perform consideration is given here to a first look at the individual performance of ATLA. One way to do this at least for any outlier detection algorithm is evaluate the probability of receiving a type I error. That is, identifying an outlier when in actual fact the observation is an observation generated from the uncontaminated parent component distribution.

In particular for hypothesis testing, the probability of receiving a type I error or false positive can sometimes be referred to as the *size* of a test.

By simulating data using the multivariate Tukey-Huber contamination model introduced in the previous section one can calculate the approximate power ($\epsilon > 0$) and size ($\epsilon = 0$) of ATLA for various sample sizes $n$ and variables $p$.

For simplicity set the mean vector $\mu = 0$ and covariance matrix $\Sigma = I$ which corresponds to the *FICM*.

The power and size estimates of the ATLA will be calculated using the following probability formula,

$$\text{Size/Power} = \mathcal{P}(\text{at least one outlier is identified})$$

$$= \left( \frac{\text{#s of times at least one outlier is identified}}{N} \right)$$

where $N$ denotes the number of samples generated which in this instance will be $N = 50,000$ to ensure accurate approximations.
Table 2.1: Simulation of $N = 50,000$ samples of size $n$ and variables $p$ generated from Multivariate $\epsilon$-contaminated Tukey-Huber Model with $\mu = 0J$, $\Sigma = I_{p}$ and $\sigma = 3$, for the Size ($\epsilon = 0$) and Power ($\epsilon > 0$) of ATLA.

<table>
<thead>
<tr>
<th></th>
<th>Size Calculation (ATLA)</th>
<th>Power Calculation (ATLA)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$n=25$</td>
<td>$n=50$</td>
</tr>
<tr>
<td>$\epsilon = 0$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p=2$</td>
<td>0.027</td>
<td>0.011</td>
</tr>
<tr>
<td>$p=5$</td>
<td>0.027</td>
<td>0.004</td>
</tr>
<tr>
<td>$\epsilon = 0.1$</td>
<td>0.387</td>
<td>0.693</td>
</tr>
<tr>
<td>$p=2$</td>
<td>0.490</td>
<td>0.904</td>
</tr>
<tr>
<td>$p=5$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\epsilon = 0.2$</td>
<td>0.516</td>
<td>0.834</td>
</tr>
<tr>
<td>$p=2$</td>
<td>0.647</td>
<td>0.985</td>
</tr>
<tr>
<td>$p=5$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

As evident in the first row corresponding to $\epsilon = 0$ in Table 2.1 one can see that the ATLA performs quite well across all of the imposed conditions in terms of the approximated sizes with the largest value of only 2.7% occurring for $n = 25$.

Once contamination is introduced into the samples these probabilities increase dramatically which is to be expected. One can also see the influence the sample size has on both the power and size calculations which is intuitively reasonable considering the larger number of observations result in more accurate location and scale estimate due to the LLN. It would also be reasonable to attribute this to the choice in objective function indicated by a slight change in performance that can be seen when comparing $n = 25$ and $n = 50$.

Interestingly one can see that samples of higher dimension $p$ quite consistently result in smaller values for the size approximations whilst the opposite occurs for the power calculations once contamination is introduced into the data. Perhaps this may be simply due to the fact that additional variables and the information these contain are leading to more robust location/scale estimates and hence leading to less occurrences of type I errors like in the case of larger sample sizes.

2.10 The Cramér Rao Lower bound - Multivariate Case

For the case of multivariate location estimation one would be concerned with the form of the Cramér Rao inequality that involves an unbiased estimator vector, $T$ of $X$, however one must first define the matrix form of Fisher information for multiple parameters. Given the parameter column vector $\theta = [\theta_1 \theta_2 \cdots \theta_p]^T$ and $f(x|\theta)$ denotes the probability density function for the sample $x$ then the Fisher information matrix has element-$(i,j)$ of the
form,

$$[\mathcal{I}(\theta)]_{i,j} = E \left[ \left( \frac{\partial}{\partial \theta_i} \log f(x|\theta) \right) \left( \frac{\partial}{\partial \theta_j} \log f(x|\theta) \right) \right] \bigg| \theta.$$ 

Now defining the Cramér Rao lower bound in matrix form under some regularity conditions as,

$$\text{Cov}_{\theta}(T[X]) \geq \frac{\partial \psi(\theta)}{\partial \theta} \left[ \mathcal{I}(\theta) \right]^{-1} \left( \frac{\partial \psi(\theta)}{\partial \theta} \right)^T$$

(2.7)

where $\psi(\theta)$ is the expectation vector $E[T(X)] = [E[T_1], E[T_2], \ldots, E[T_p]]^T$, which is in fact a vector that depends on the underlying parameter vector $\theta$. Here $\text{Cov}_{\theta}(T[X])$ corresponds to the covariance matrix of $T[X]$ which is denoted $\Sigma(T, \theta)$ for the time being and setting $J(\theta)$ to be the Jacobian matrix, $\frac{\partial \psi(\theta)}{\partial \theta}$.

Then we have the following;

$$\Sigma(T, \theta) \geq J(\theta)[\mathcal{I}(\theta)]^{-1}J(\theta)^T$$

which implies that the matrix, $\Sigma(T, \theta) - [\mathcal{I}(\theta)]^{-1}$ is positive semi-definite. As a result of this positive semi-definiteness it follows that,

$$|\Sigma(T, \theta)| \geq |J(\theta)[\mathcal{I}(\theta)]^{-1}J(\theta)^T|,$$

where $|\Sigma(T, \theta)|$ is what is referred to as the generalised variance of the estimators. A much formal proof may be found in Rao (1973, pp. 326–328) and in particular noting the discussion of generalised variance.

By continuation of Equation (2.7) in the case that the estimator is unbiased this implies $\psi(\theta) = \theta$ and thus $J(\theta) = I$ which leads to a simplification to the equation,

$$\Sigma(T, \theta) \geq [\mathcal{I}(\theta)]^{-1}.$$

### 2.11 Relative Efficiency of Multivariate Estimates

One might recall our discussion of the use of the relative efficiency in comparing the performance of two univariate estimates of locations from back in Section (1.9). For the sake of consistency we will introduce the concept of relative efficiency specifically for the case of multivariate location estimation as
this plays a significant part in assessing the performance of an outlier detection algorithm.

The calculation follows the same form as in formula (1.10) whereby as a single measure of efficiency of a multivariate estimator say $T$, one could propose,

$$\text{eff}(T) = \frac{|I(\theta)^{-1}|}{\text{Var}(T)}$$

(2.8)

Noting here that $\text{Var}(T)$ corresponds to the generalised variance which in this case is the determinant of the covariance matrix, of the estimator $T$, call it $|\Sigma_T|$.

Hence if one were to compare the efficiencies of two multivariate location estimates, $T_1$ and $T_2$ for example then this would involve the following calculation,

$$\text{RelEff} = \frac{\text{eff}(T_2)}{\text{eff}(T_1)} = \frac{|\hat{\Sigma}_{T_1}|}{|\hat{\Sigma}_{T_2}|}$$

(2.9)

where $|\cdot|$ denotes the determinant of the associated covariance matrix.

### 2.12 ATLA relative to the Sample Mean

To demonstrate the concept of relative efficiency for multivariate location estimation one can assess the optimal performance of the ATLA through calculations for the estimated relative efficiency of the trimmed sample mean relative to the original sample mean without trimming.

For this situation one can use the $FDCM$ with mean vector $\mu = 0$ and covariance matrix,

$$\Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix},$$

corresponding to setting $\sigma_1 = \sigma_2 = 1$ and $\rho = 0.5$ and here $p = 2$. From this simulation of $N = 5,000$ bivariate samples of size $n = 100$ one is able to produce $N$ individual estimates for the sample mean,

$$\bar{X}_1, \ldots, \bar{X}_N = \left(\bar{X}_1 \bar{X}_2\right)_{1,\ldots,N}$$
and the ATLA-trimmed sample mean respectively. Using these it is possible to produce estimates for the variance-covariance matrix of the sample means,

$$\sum_0 = \text{var} [\bar{X}] = \text{Cov} (\bar{X}_1, \ldots, \bar{X}_N)$$

as well as for the ATLA trimmed sample means which we will denote $\Sigma_{ATLA}$. These assist in the calculation of the estimated relative efficiency,

$$\text{RelEff}_\epsilon = \frac{|\sum_0|}{|\sum_{ATLA}|}$$

where $|\cdot|$ denotes the determinant of the associated covariance matrix. It is a natural thing to therefore compare multivariate estimators by looking at the determinant of the covariance matrix which corresponds to the generalised variance.

Looking at Figure 2.4 one can see that for $\epsilon = 0$ the relative efficiency starts at approximately 100% which is to be expected seeing as there would be minimal cases of trimming. Then as the contamination $\epsilon$ increases the relative efficiency does so also in an approximately linear fashion before reaching approximately 180% at $\epsilon = 0.10$. 

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{Figure2.4.png}
\caption{Estimated Relative efficiency of the ATLA trimmed sample mean relative to the original sample mean without trimming based on 5,000 simulations of bivariate ($p = 2$) samples of size $n = 100$.}
\end{figure}
2.13 Student’s t-Distribution

The Student’s t-Distribution is an extremely prominent statistical measure in the context of univariate location estimation particularly for situations involving small samples which have been taken from a normally distributed population with unknown standard deviation. For the generalised univariate case it has the following probability density function:

\[
p(x|\nu, \mu, \sigma^2) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)\sqrt{\pi\nu\sigma}} \left(1 + \frac{1}{\nu} \frac{(x - \mu)^2}{\sigma^2}\right)^{-\frac{\nu+1}{2}}
\]  \hspace{2cm} (2.10)

where \(\nu\) represents the degrees of freedom and \(\mu\) and \(\sigma^2\), the location and scale parameters respectively and \(\Gamma\) here corresponds to the Gamma Function defined previously (see Section 2.6). Using the density 2.10 one can establish the following results,

\[
\begin{align*}
E(X) &= \mu &\text{for } \nu > 1, \\
\text{Var}(X) &= \sigma^2 \nu \left(\nu - 2\right)^{-1} &\text{for } \nu > 2.
\end{align*}
\]

The t-distribution is very similar to the Normal Distribution in terms of its symmetry and the overall bell shape only it tends to have wider tails. The degrees of freedom parameter \(\nu\), determines the kurtosis of the distribution and for increasing \(\nu\) the resulting distribution tends to the Normal Distribution with mean \(\mu = 0\) and variance \(\sigma^2 = 1\), that is

\[\text{As } \nu \uparrow \infty, \ t_\nu \rightarrow \mathcal{N}(0, 1)\]

hence why it is often referred to as the normality parameter. Convergence here is in distribution, a discussion discussed at length in Billingsley (1968).

Now just as we detailed the existence of a multivariate generalisation of the Normal Distribution in Section (2.2), there also exists a similar result in terms of the multivariate Student’s t-distribution.

2.14 Multivariate Student’s t-Distribution

Now suppose that we have a \(p\)-dimensional random vector \(X = (X_1, X_2, \ldots, X_p)^T\) which is said to follow a multivariate t-distribution with mean vector \(\mu = (\mu_1, \mu_2, \ldots, \mu_p)^T\) and variance-covariance matrix \(\Sigma\), denoted by \(T_{\nu,p}(\mu, \Sigma)\) for a predefined degrees of freedom \(\nu > 2\) and \(p\) being the di-
Now the probability density function is as follows,

\[
p(x|\nu, \mu, \Sigma) = \frac{\Gamma\left(\frac{\nu+p}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right) (\pi\nu)^{\frac{p}{2}}} \left| \Sigma \right|^{-\frac{1}{2}} \left(1 + \frac{(x - \mu)^T \Sigma^{-1} (x - \mu)}{\nu} \right)^{-\frac{\nu+p}{2}}, \quad \nu > 2.
\]

Using this density it is possible to establish the following results,

\[
\begin{align*}
\mathbb{E}(X) &= \mu & \text{for } \nu > 1, \\
\text{Var}(X) &= \Sigma \nu (\nu - 2)^{-1} & \text{for } \nu > 2.
\end{align*}
\]

Note that for \( p = 1 \) the distribution \( T_{\nu,1}(\mu, \Sigma) \) is equivalent to the univariate Student’s t-distribution established in the previous section. One might also notice the similarities between the densities (2.10) and (2.11). Furthermore the same result in the univariate case applies to that of the multivariate and that is,

\[
\text{As } \nu \uparrow \infty, \; T_{\nu,p}(\mu, \Sigma) \to \mathcal{N}_p(0, I_p).
\]

Here again convergence is in distribution as detailed in Billingsley (1968). Now as alluded to previously, the presence of wider tails and the ability to control this kurtosis through the degrees of freedom makes the t-distribution ideal for estimation particularly in situations involving the presence of outliers. This is because the kurtosis allows for the exclusion and/or downweighting of these outlying values and thus in theory produce a more accurate location and scale estimate.

Now just like for any parametric modelling procedure there exists different types of estimation methods which one could use for the multivariate t-distribution. Perhaps the most well known and the one that we will be considering is finding the Maximum Likelihood Estimator (MLE) through the use of an expectation-maximisation or EM-algorithm.

It is worth noting is that it is possible to produce estimates not only for location and scale but also the degrees of freedom. It is generally accepted that the degrees of freedom \( \nu \) should be fixed for small data sets and estimated for larger ones however for the sake of computational simplicity and consistency the degrees of freedom will be fixed to an appropriate low value.

Now the next question that can be asked is, is what degrees of freedom should one use in the distribution in order to achieve the most consistently accurate location estimates? For smaller samples Lange et al. (1989) found that the value \( \nu = 4 \) to work well in numerous applications whilst Venables and Ripley (2002) suggests \( \nu = 5 \) degrees of freedom.
2.14.1 Simulation Results

In order to test the performance of the t-distribution in terms of location estimation it is possible to conduct the same simulations as in Section (2.12) which now produce estimated relative efficiencies against ATLA-estimates. On the other hand, using the method of planting \( k \) outliers generated from \( \mathcal{N}_p(5\mathbf{J}, I_p) \) and \( n \) observations from \( \mathcal{N}_p(0\mathbf{J}, I_p) \), where \( \mathbf{J} \) is a vector of ones, this does not result in common zero mean vector unlike the Tukey-Huber contamination model used previously. So instead the Mean Squared Error (MSE) is used by simply taking the determinant of the covariance matrix of estimates \( (\hat{\mu} - \mu) \) where \( \mu \) is the expected value of the combined contaminated sample which is \( (0.95 \times 0 + 0.05 \times 5\mathbf{J}) = 0.05\mathbf{J} \).

The MATLAB function `fitt_fixnu.m` within the `tdistfit` package will be used here which is provided by Ince (2013) on GitHub.

Table 2.2: Estimated Relative efficiency of the ATLA trimmed sample mean relative to the location estimate from a t-distribution fit on \( \nu = 4 \) degrees of freedom using an EM Algorithm based on a simulation of \( N = 1000 \), samples of size \( n \) and \( p \)-variables, consisting of \( k \) planted outliers. The \( n - k \) observations have been generated from the Multivariate distribution \( \mathcal{N}_p(0\mathbf{J}, I_p) \) and the \( k \) planted outliers from \( \mathcal{N}_p(5\mathbf{J}, I_p) \).

<table>
<thead>
<tr>
<th></th>
<th>( n=25 )</th>
<th>( n=50 )</th>
<th>( n=100 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( p=2 )</td>
<td>( p=5 )</td>
<td>( p=2 )</td>
</tr>
<tr>
<td>( k=0 )</td>
<td>1.200</td>
<td>1.842</td>
<td>1.210</td>
</tr>
<tr>
<td>( k=1 )</td>
<td>1.017</td>
<td>1.562</td>
<td>1.178</td>
</tr>
<tr>
<td>( k=2 )</td>
<td>0.887</td>
<td>1.565</td>
<td>1.196</td>
</tr>
<tr>
<td>( k=5 )</td>
<td>0.468</td>
<td>0.991</td>
<td>0.992</td>
</tr>
<tr>
<td>( k=10 )</td>
<td><strong>0.040</strong></td>
<td><strong>6.110</strong></td>
<td><strong>0.792</strong></td>
</tr>
</tbody>
</table>

2.14.2 Discussion

Looking at the simulation results in Table 2.2 in most circumstances the ATLA performs better than the EM-algorithm when there are no planted outliers that is when \( k = 0 \). This to be expected since the main advantage of the t-distribution is that it downweights outlying values and so with no planted outliers then it essentially downweights non-outliers leading to it’s inefficiency. One might also notice in almost all situations the efficiency significantly improves going from \( p = 2 \) to \( p = 5 \) variables. In fact ATLA is more efficient for almost all combinations of \( n \) and \( p \). This could be a reflection of one of
the strengths of ATLA, a deficiency of the t-distribution or a combination of both.

Furthermore it is seen that the EM-algorithm outperforms the ATLA in situations involving smaller sample sizes with planted outliers which may be due to the choice in degrees of freedom. As mentioned previously the choice of \( \nu = 4 \) degrees of freedom was not without reason as it is known to be particularly versatile yet optimal in situations involving small sample sizes.

One might also notice the strange phenomenon whereby the ATLA appears to fail quite significantly for \( p = 2, n = 25 \) and \( k = 10 \). Although one would expect a decrease in performance since it is equivalent to 40% of contamination, however such a significant decrease in performance is unforeseen particularly when it appears to perform normally if not significantly better than the t-distribution for \( p = 5 \). Whether this comes as a result of a breakdown in the MCD estimator is up for conjecture, however further investigation would be warranted.

Although this framework of planting \( k \) outliers generated from \( \mathcal{N}_p(5J, I_p) \) may seem to be adequate in testing the performance of algorithms over a broad range of conditions, we can’t help but criticise any simulation results produced off of such a framework. One might argue that it is not a proper distribution to be generating outliers from and in addition the choice of mean vector \( 5J \) although in theory seems reasonable, can be regarded as somewhat arbitrary.

![Figure 2.5: Estimated Relative efficiency of the ATLA trimmed sample mean relative to the location estimate from a t-distribution fit on \( \nu = 4 \) degrees of freedom using an EM Algorithm based on a simulation of \( N = 1000 \) samples of size \( n \) and variables \( p = 2 \) [a] and \( p = 5 \) [b] respectively. Samples have been generated from the Tukey-Huber \( \epsilon \)-contamination model, that is \((1-\epsilon) \cdot \mathcal{N}_p(0J, I_p) + \epsilon \cdot \mathcal{N}_p(0, I_p \cdot 9)\)](image)
For the time being judgement of the performance of the Multivariate $t$-distribution over the ATLA from samples generated through the Tukey-Huber $\epsilon$-contamination multivariate model framework is reserved. Looking at both plots in Figure 2.5 one can initially see that it follows the general trend of a decrease in efficiency for increasing contamination which is to be expected.

One can also notice that in most circumstances the larger sample sizes, $n$, resulted in higher relative efficiencies or equivalently the better performance of the ATLA, especially so going from $n = 25$ to $n = 50$ for $p = 5$. However it is apparent that this case can not be generalised for large sample sizes evident by the small to no performance increase going from $n = 50$ to $n = 100$.

For the bivariate case shown in Figure 2.5a) it appears that for $n = 25$ the ATLA is the more efficient estimator up until approximately $3 - 4\%$ contamination before the $t$-distribution appears to surpass the performance of ATLA. This is also the case for $n = 25$ and $p = 5$ where ATLA initially performs significantly better before the $t$-distribution exceeds it only at a much more significant rate than the bivariate case as evident by the relative efficiency reaching approximately $40\%$ at $10\%$ contamination. This result is somewhat in line with the phenomenon was found occurring when using the mean slippage model however the significant drop in relative efficiency was seen going from $k = 2$ to $k = 5$ which is equivalent to $8\%$ and $20\%$ contamination respectively. Perhaps it would be appropriate explore contamination in excess of $10\%$ to see whether it results in the same phenomenon to confirm whether it is as a result of the actual algorithm or a by-product of the data and how it is simulated.

One can see in situations involving $0\%$ contamination that samples of size $n = 50$ achieve the respective maximum relative efficiencies of $127\%$ and $190\%$ respectively. This is somewhat surprising as one would expect to see this maximum occur at the largest sample size of $n = 100$, however this could be as a result of reaching a sufficiently large sample size for the algorithm to perform properly yet not excessively large to minimise the chance of flagging an outlier or simply inflating the location estimate.

From our two brief simulations results it is quite evident that for the case of estimating location the multivariate $t$-distribution on $\nu = 4$ degrees of freedom performs optimally at least comparatively for smaller sample sizes, $n$ with less variables, $p$ and for higher levels of contamination, $\epsilon$. These observations are tempered by the fact that the algorithms are inherently solving different objectives, the multivariate $t$-distribution seeking to find efficient estimates in data having heavy tails, and ATLA being used for outlier identification.
2.15 ATLA relative to the Forward Search

Following the discussion of the original Forward Search algorithm proposed by Hadi (1992) in Section (2.5) it is appropriate to see how it compares to that of the ATLA. However it is important to emphasise that Hadi’s original procedure and the ATLA are essentially different algorithm’s meaning each have their respective advantages and disadvantages so one should not make any gross generalisations based off of our limited comparisons.

2.15.1 Simulation Results

The modified algorithm of the forward search is considered here together with the mean slippage model and in addition the same proposed measures of performance as explored in Hadi (1994) are utilised. To evaluate each algorithm five measures of performance that are calculated follows;

\[ p_1 = \Pr(\text{at least one observation is identified}) \]
\[ p_2 = \Pr(\text{at least one planted outlier is successfully identified}) \]
\[ p_3 = \Pr(\text{exactly correct identification of all planted outliers}) \]
\[ p_4 = \text{Proportion of masked observations} \]
\[ = \frac{\text{(total number of masked observations)}}{(n \times N)} \]
\[ p_5 = \text{Proportion of swamped observations} \]
\[ = \frac{\text{(total number of swamped observations)}}{(n \times N)}. \]

For the probabilities \( p_1, p_2 \) and \( p_3 \) larger resulting values are indicative of good performance for the respective algorithm whilst the opposite can be said for \( p_4 \) and \( p_5 \).

Table 2.3: Size approximations of the ATLA against Hadi’s original forward search (1994) based on a simulation of \( N = 1000 \), samples of \( p \)-variables of size \( n \) generated from the multivariate distribution \( N_p(0, I_p) \).

<table>
<thead>
<tr>
<th>Simulation results ((p_1)) for ( k=0 )</th>
<th>( n=25 )</th>
<th>( n=50 )</th>
<th>( n=100 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p=2 ) ( p=5 ) ( p=2 ) ( p=5 ) ( p=2 ) ( p=5 )</td>
<td>( p=2 ) ( p=5 ) ( p=2 ) ( p=5 ) ( p=2 ) ( p=5 )</td>
<td>( p=2 ) ( p=5 ) ( p=2 ) ( p=5 ) ( p=2 ) ( p=5 )</td>
<td></td>
</tr>
<tr>
<td>Hadi</td>
<td>0.045 0.040 0.052 0.043 0.040 0.037</td>
<td>0.045 0.040 0.052 0.043 0.040 0.037</td>
<td></td>
</tr>
<tr>
<td>ATLA</td>
<td>0.026 0.026 0.008 0.001 0.002 0.000</td>
<td>0.026 0.026 0.008 0.001 0.002 0.000</td>
<td></td>
</tr>
</tbody>
</table>
Table 2.4: Performance measures of the ATLA against Hadi’s original forward search (1994) based on a simulation of $N = 1000$, samples of size $n$ and $p$-variables, consisting of $k$ planted outliers. The $n - k$ observations have been generated from the multivariate distribution $\mathcal{N}_p(0J, I_p)$ and the $k$ planted outliers from $\mathcal{N}_p(5J, I_p)$.

<table>
<thead>
<tr>
<th></th>
<th>n=25</th>
<th>n=50</th>
<th>n=100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ATLA</td>
<td>Hadi</td>
<td>ATLA</td>
</tr>
<tr>
<td>$p_1$</td>
<td>0.921</td>
<td>0.995</td>
<td>0.976</td>
</tr>
<tr>
<td>$p_2$</td>
<td>0.921</td>
<td>0.995</td>
<td>0.976</td>
</tr>
<tr>
<td>$p_3$</td>
<td>0.897</td>
<td>0.943</td>
<td>0.948</td>
</tr>
<tr>
<td>$p_4$</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>$p_5$</td>
<td>0.002</td>
<td>0.007</td>
<td>0.000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>n=25</th>
<th>n=50</th>
<th>n=100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ATLA</td>
<td>Hadi</td>
<td>ATLA</td>
</tr>
<tr>
<td>$p_1$</td>
<td>0.981</td>
<td>0.965</td>
<td>1.000</td>
</tr>
<tr>
<td>$p_2$</td>
<td>0.981</td>
<td>0.964</td>
<td>1.000</td>
</tr>
<tr>
<td>$p_3$</td>
<td>0.894</td>
<td>0.881</td>
<td>0.856</td>
</tr>
<tr>
<td>$p_4$</td>
<td>0.001</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>$p_5$</td>
<td>0.004</td>
<td>0.006</td>
<td>0.003</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>n=25</th>
<th>n=50</th>
<th>n=100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ATLA</td>
<td>Hadi</td>
<td>ATLA</td>
</tr>
<tr>
<td>$p_1$</td>
<td>0.928</td>
<td>0.840</td>
<td>1.000</td>
</tr>
<tr>
<td>$p_2$</td>
<td>0.928</td>
<td>0.836</td>
<td>1.000</td>
</tr>
<tr>
<td>$p_3$</td>
<td>0.812</td>
<td>0.768</td>
<td>0.753</td>
</tr>
<tr>
<td>$p_4$</td>
<td>0.001</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>$p_5$</td>
<td>0.004</td>
<td>0.003</td>
<td>0.007</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>n=25</th>
<th>n=50</th>
<th>n=100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ATLA</td>
<td>Hadi</td>
<td>ATLA</td>
</tr>
<tr>
<td>$p_1$</td>
<td>0.620</td>
<td>0.290</td>
<td>1.000</td>
</tr>
<tr>
<td>$p_2$</td>
<td>0.620</td>
<td>0.287</td>
<td>1.000</td>
</tr>
<tr>
<td>$p_3$</td>
<td>0.607</td>
<td>0.257</td>
<td>0.855</td>
</tr>
<tr>
<td>$p_4$</td>
<td>0.001</td>
<td>0.001</td>
<td>0.000</td>
</tr>
<tr>
<td>$p_5$</td>
<td>0.000</td>
<td>0.006</td>
<td>0.003</td>
</tr>
</tbody>
</table>

36
2.15.2 Discussion

Starting with the null case \((k = 0)\) corresponding to the size approximations as seen in Table 2.3 Hadi’s procedure seems to produce results which are consistent with, if not better than, the pre-specified significance level, \(a = 0.05\). When comparing procedures it is evident that it does not perform as well as the ATLA which brings us to one of the main benefits of this algorithm and that is that it does not rely on a user defined significance level/fixed threshold. Although there may be some instances where having the option to impose a certain significance level is beneficial, one would argue that not having this requirement makes the ATLA much more versatile and consequently ideal for automation. For example there may be situations that require assessing multiple datasets for which some are known to consist of outliers and some are not and hence consistent performance with a small probability of type I error makes the ATLA the preferable method.

Now comparing the performance measures for outlier contaminated samples in Tables 2.4a) and b) it shows that Hadi’s procedure performs very well particularly for samples with fewer planted outliers or equivalently less contamination. One can assume this has something to do with the sizes discussed in the previous paragraph implying that Hadi’s method is much more sensitive to flagging outliers and hence leading to comparatively higher \(p_1, p_2\) and \(p_3\) probabilities than the ATLA. Although it does seem to be slightly more susceptible to swamping effects than to that of the ATLA at least for \(k = 1\) shown in Table 2.4a). However with such low proportions of masking and swamping, \(p_4\) and \(p_5\) one can not be over reliant on such estimates even when it is based on \(N = 1,000\) samples.

It is when more outliers are planted and higher contamination levels are reached that the performance of ATLA equals or exceeds that of Hadi’s procedure yet there are still several instances where Hadi’s method produces a higher probability of the exact correct outlier identification, \(p_3\).

However as was assumed in a previous section, the main strength of ATLA in terms of its performance are in large samples of \(p > 2\) which seem to be consistent with the results from this particular simulation. Note we have chosen to omit these tables for brevity however they are available in Appendix B.1.1.

There is one particular instance which shows a significant failure in the performance of Hadi’s algorithm for \(n = 25, k = 10\) and \(p = 5\) indicated in red in Table 2.4d). This comes under similar circumstances to what is found happening to the ATLA whereby it performs appropriately for one variable condition.
yet erroneous for another for a fixed contamination and sample size. Again one can only assume that this comes as a results of a breakdown, however further investigation is warranted.

### 2.16 ATLA relative to BACON Method

The *Blocked Adaptive Computationally-Efficient Outlier Nominators* (BACON) method is a general approach that was proposed in order to overcome such issues as the computational cost that continually plagues a lot of the multivariate outlier detection methods that precede it. This procedure was based on the original forward search procedures of Hadi (1992, 1994) and as a consequence shares a lot of similarities with such, as well as multivariate ATLA. One such similarity is the requirement of specifying a significance level, $\alpha$ for the cut-off value $\chi^2$ used to define the next iterations basic subset. The BACON algorithm also allows the user to define the size of the initial basic subset, $m = cp$ based on a decision of the integer $c$. This choice is subject to the discretion of the analyst however through simulation results Nedret et al. (2000) recommend $c = 4$ or 5 for adequate performance.

This discussion does not go into too much detail regarding the explanation of the BACON method unlike previous methods however if one were to be so inclined it is recommended to see Nedret et al. (2000).

This method operates under the same premise of the construction of an initial basic (and non-basic) subset that is assumed to be outlier-free followed by a rapid inflation of the basic subset and testing of criterion all while incorporating blocks of observations at each step. The employment of blocks rather than gradual inflations results in a significant increase in computational efficiency due to requiring less covariance matrix evaluations and inversions as well as eliminating the need for continual re-orderings.

#### 2.16.1 Simulation Results

Nedret et al. (2000) acknowledges this improvement with claims that the BACON method is able to be applied to very large data sets of size $n > 100,000$ whilst also producing results that are comparable in performance with that of Hadi (1994) as well as MCD, minimum volume ellipsoid (MVE) and least mean squares (LMS) based algorithm’s. Hence through $N = 1000$ simulations of the mean slippage model given here is an investigation of how it compares to the ATLA in terms of the five measures of performance for smaller samples of $n = 25$, 50 and 100 initially. To remain consistent with Nedret et al. (2000)
use is made of a significance level of $\alpha = 0.05$, integer $c = 4$ and $k$ outliers to be generated from $\mathcal{N}(4\mathbf{J}, I_p)$ as opposed to mean vector $5\mathbf{J}$ in the previous section.

Here we will be using the `mvBACON` function in the `robustX` R package (Stahel and Maechler; 2019).

Table 2.5: Size approximations of the ATLA against the BACON method based on a simulation of $N = 1000$, samples of $p$-variables of size $n$ generated from the multivariate distribution $\mathcal{N}_p(0, I_p)$.

<table>
<thead>
<tr>
<th></th>
<th>n=25</th>
<th>n=50</th>
<th>n=100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>p=2</td>
<td>p=5</td>
<td>p=2</td>
</tr>
<tr>
<td><strong>BACON</strong></td>
<td>0.041</td>
<td>0.017</td>
<td>0.049</td>
</tr>
<tr>
<td><strong>ATLA</strong></td>
<td>0.026</td>
<td>0.026</td>
<td>0.008</td>
</tr>
</tbody>
</table>
Table 2.6: Performance measures of the ATLA against the BACON method based on a simulation of $N = 1000$, samples of size $n$ and $p$-variables, consisting of $k$ planted outliers. The $n - k$ observations have been generated from the Multivariate distribution $N_p(0, I_p)$ and the $k$ planted outliers from $N_p(4J, I_p)$.

### a) Simulation results for $k=1$ and $p=2$

<table>
<thead>
<tr>
<th></th>
<th>n=25</th>
<th>n=50</th>
<th>n=100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ATLA</td>
<td>BACON</td>
<td>ATLA</td>
</tr>
<tr>
<td>$p_1$</td>
<td>0.613</td>
<td>0.894</td>
<td>0.760</td>
</tr>
<tr>
<td>$p_2$</td>
<td>0.613</td>
<td>0.893</td>
<td>0.760</td>
</tr>
<tr>
<td>$p_3$</td>
<td>0.581</td>
<td>0.851</td>
<td>0.733</td>
</tr>
<tr>
<td>$p_4$</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>$p_5$</td>
<td>0.005</td>
<td>0.005</td>
<td>0.001</td>
</tr>
</tbody>
</table>

### b) Simulation results for $k=5$ and $p=2$

<table>
<thead>
<tr>
<th></th>
<th>n=25</th>
<th>n=50</th>
<th>n=100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ATLA</td>
<td>BACON</td>
<td>ATLA</td>
</tr>
<tr>
<td>$p_1$</td>
<td>0.772</td>
<td>0.619</td>
<td>0.963</td>
</tr>
<tr>
<td>$p_2$</td>
<td>0.772</td>
<td>0.615</td>
<td>0.963</td>
</tr>
<tr>
<td>$p_3$</td>
<td>0.657</td>
<td>0.514</td>
<td>0.738</td>
</tr>
<tr>
<td>$p_4$</td>
<td>0.002</td>
<td>0.003</td>
<td>0.002</td>
</tr>
<tr>
<td>$p_5$</td>
<td>0.005</td>
<td>0.006</td>
<td>0.003</td>
</tr>
</tbody>
</table>

### c) Simulation results for $k=10$ and $p=2$

<table>
<thead>
<tr>
<th></th>
<th>n=25</th>
<th>n=50</th>
<th>n=100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ATLA</td>
<td>BACON</td>
<td>ATLA</td>
</tr>
<tr>
<td>$p_1$</td>
<td>0.620</td>
<td>0.078</td>
<td>0.960</td>
</tr>
<tr>
<td>$p_2$</td>
<td>0.620</td>
<td>0.068</td>
<td>0.960</td>
</tr>
<tr>
<td>$p_3$</td>
<td>0.513</td>
<td>0.042</td>
<td>0.638</td>
</tr>
<tr>
<td>$p_4$</td>
<td>0.001</td>
<td>0.004</td>
<td>0.002</td>
</tr>
<tr>
<td>$p_5$</td>
<td>0.004</td>
<td>0.001</td>
<td>0.007</td>
</tr>
</tbody>
</table>

### d) Simulation results for $k=10$ and $p=5$

<table>
<thead>
<tr>
<th></th>
<th>n=25</th>
<th>n=50</th>
<th>n=100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ATLA</td>
<td>BACON</td>
<td>ATLA</td>
</tr>
<tr>
<td>$p_1$</td>
<td>0.395</td>
<td>0.010</td>
<td>1.000</td>
</tr>
<tr>
<td>$p_2$</td>
<td>0.395</td>
<td>0.005</td>
<td>1.000</td>
</tr>
<tr>
<td>$p_3$</td>
<td>0.379</td>
<td>0.000</td>
<td>0.855</td>
</tr>
<tr>
<td>$p_4$</td>
<td>0.001</td>
<td>0.001</td>
<td>0.000</td>
</tr>
<tr>
<td>$p_5$</td>
<td>0.000</td>
<td>0.000</td>
<td>0.003</td>
</tr>
</tbody>
</table>
2.16.2 Discussion

Beginning with the simulation results of $p_1$ for $k = 0$ corresponding to the size approximations as seen in Table 2.5 it is evident that the BACON algorithm performs slightly below the pre defined significance level, $\alpha = 0.05$ as expected. Again it is evident that the ATLA consistently displays a significantly smaller size.

When considering the performance measures associated with the probabilities $p_1$, $p_3$ and $p_2$ in Tables 2.6a) the BACON algorithm performs quite effectively even in situations involving smaller sample sizes as it consistently outperforms the measures of ATLA. However once more outliers are introduced and higher contamination is achieved then it is seen that the ATLA leads to better measures of performance like in the case of Tables 2.6b), c) and d). Interestingly comparing these measure with previous comparisons in Tables 2.4 where outliers were generated from mean vector 5J it transpires that there is significant drop in performance of up to 30% particularly for smaller samples of size $n < 100$.

On consideration of the probability of correctly identifying all planted outliers, $p_3$ it is evident that the block inflations are also leading to good observations being flagged as outlying which is consistent with what was found in Table 2.5. This can be confirmed by the fact that the BACON algorithm consistently produces a higher proportion of swamped observations, $p_5$ in comparison to ATLA. Interestingly it is found that the same phenomenon is occurring with the BACON algorithm in situations involving high contamination and more variables whereby the performance dramatically decreases as shown in red in Tables 2.6 c) and d). One can only attribute this to the finite sample breakdown point which in the case of BACON is claimed to be approximately 20% (Nedret et al.; 2000). A measure one can assume would also be influenced by the number of variables $p$.

2.16.3 Simulation Results for Large Sample Sizes

As alluded to previously due to the way BACON incorporate blocks of observations in the inflations of the basic subsets this allows it to be effective in situations involving larger samples sizes. Now consider the exploration of these same performance measure only with samples of size $n = 250$, 500 and 1000 respectively with a proportional number of planted outliers to ensure consistent contamination levels. This will not only give us an idea of how the ATLA performs for larger $n$ but also whether there is a significant improvement in the BACON method’s performance. Again due to the way BACON was designed
one cannot ensure the validity of this comparison however it does gauge an idea of how each performs in each others "optimal" conditions.

Table 2.7: Size approximations of the ATLA against the BACON method based on a simulation of $N = 1000$, samples of $p$-variables of size $n > 100$ generated from the multivariate distribution $\mathcal{N}_p(0J, I_p)$.

<table>
<thead>
<tr>
<th></th>
<th>n=200</th>
<th>n=500</th>
<th>n=1000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>p=2</td>
<td>p=5</td>
<td>p=2</td>
</tr>
<tr>
<td>BACON</td>
<td>0.052</td>
<td>0.045</td>
<td>0.061</td>
</tr>
<tr>
<td>ATLA</td>
<td>0.000</td>
<td>0.000</td>
<td>0.001</td>
</tr>
</tbody>
</table>
Table 2.8: Performance measures of ATLA against BACON method based on a simulation of $N = 1000$, samples of size $n > 100$ and $p$-variables, consisting of $k$ planted outliers. The $n-k$ observations have been generated from the Multivariate distribution $\mathcal{N}_p(0_J, I_p)$ and the $k$ planted outliers from $\mathcal{N}_p(4J, I_p)$.

a) Simulation results for $k=20$ and $p=2$

<table>
<thead>
<tr>
<th></th>
<th>n=200</th>
<th>n=500</th>
<th>n=1000</th>
<th>n=200</th>
<th>n=500</th>
<th>n=1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$</td>
<td>ATLA</td>
<td>BACON</td>
<td>ATLA</td>
<td>BACON</td>
<td>ATLA</td>
<td>BACON</td>
</tr>
<tr>
<td></td>
<td>0.998</td>
<td>0.871</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>$p_2$</td>
<td>0.998</td>
<td>0.869</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>$p_3$</td>
<td>0.501</td>
<td>0.327</td>
<td>0.442</td>
<td>0.175</td>
<td>0.318</td>
<td>0.132</td>
</tr>
<tr>
<td>$p_4$</td>
<td>0.002</td>
<td>0.008</td>
<td>0.001</td>
<td>0.004</td>
<td>0.001</td>
<td>0.002</td>
</tr>
<tr>
<td>$p_5$</td>
<td>0.002</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

b) Simulation results for $k=50$ and $p=2$

<table>
<thead>
<tr>
<th></th>
<th>n=200</th>
<th>n=500</th>
<th>n=1000</th>
<th>n=200</th>
<th>n=500</th>
<th>n=1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$</td>
<td>ATLA</td>
<td>BACON</td>
<td>ATLA</td>
<td>BACON</td>
<td>ATLA</td>
<td>BACON</td>
</tr>
<tr>
<td></td>
<td>0.998</td>
<td>0.187</td>
<td>1.000</td>
<td>0.557</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>$p_2$</td>
<td>0.998</td>
<td>0.184</td>
<td>1.000</td>
<td>0.553</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>$p_3$</td>
<td>0.328</td>
<td>0.052</td>
<td>0.299</td>
<td>0.026</td>
<td>0.279</td>
<td>0.005</td>
</tr>
<tr>
<td>$p_4$</td>
<td>0.002</td>
<td>0.003</td>
<td>0.001</td>
<td>0.012</td>
<td>0.001</td>
<td>0.010</td>
</tr>
<tr>
<td>$p_5$</td>
<td>0.005</td>
<td>0.000</td>
<td>0.001</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

c) Simulation results for $k=100$ and $p=2$

<table>
<thead>
<tr>
<th></th>
<th>n=200</th>
<th>n=500</th>
<th>n=1000</th>
<th>n=200</th>
<th>n=500</th>
<th>n=1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$</td>
<td>ATLA</td>
<td>BACON</td>
<td>ATLA</td>
<td>BACON</td>
<td>ATLA</td>
<td>BACON</td>
</tr>
<tr>
<td></td>
<td>0.856</td>
<td>0.020</td>
<td>1.000</td>
<td>0.050</td>
<td>1.000</td>
<td>0.203</td>
</tr>
<tr>
<td>$p_2$</td>
<td>0.856</td>
<td>0.008</td>
<td>1.000</td>
<td>0.040</td>
<td>1.000</td>
<td>0.196</td>
</tr>
<tr>
<td>$p_3$</td>
<td>0.000</td>
<td>0.000</td>
<td>0.173</td>
<td>0.002</td>
<td>0.190</td>
<td>0.000</td>
</tr>
<tr>
<td>$p_4$</td>
<td>0.005</td>
<td>0.004</td>
<td>0.000</td>
<td>0.003</td>
<td>0.001</td>
<td>0.014</td>
</tr>
<tr>
<td>$p_5$</td>
<td>0.000</td>
<td>0.000</td>
<td>0.003</td>
<td>0.000</td>
<td>0.001</td>
<td>0.000</td>
</tr>
</tbody>
</table>

d) Simulation results for $k=100$ and $p=5$

<table>
<thead>
<tr>
<th></th>
<th>n=200</th>
<th>n=500</th>
<th>n=1000</th>
<th>n=200</th>
<th>n=500</th>
<th>n=1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$</td>
<td>ATLA</td>
<td>BACON</td>
<td>ATLA</td>
<td>BACON</td>
<td>ATLA</td>
<td>BACON</td>
</tr>
<tr>
<td></td>
<td>0.002</td>
<td>0.026</td>
<td>1.000</td>
<td>0.840</td>
<td>1.000</td>
<td>0.996</td>
</tr>
<tr>
<td>$p_2$</td>
<td>0.002</td>
<td>0.014</td>
<td>1.000</td>
<td>0.838</td>
<td>1.000</td>
<td>0.996</td>
</tr>
<tr>
<td>$p_3$</td>
<td>0.000</td>
<td>0.000</td>
<td>0.495</td>
<td>0.794</td>
<td>0.571</td>
<td>0.940</td>
</tr>
<tr>
<td>$p_4$</td>
<td>0.000</td>
<td>0.007</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>$p_5$</td>
<td>0.000</td>
<td>0.000</td>
<td>0.001</td>
<td>0.000</td>
<td>0.001</td>
<td>0.000</td>
</tr>
</tbody>
</table>
2.16.4 Discussion

The size approximations of BACON for large $n$ shown in Table 2.7 appear to be consistent with smaller sized samples if not slightly in excess of the assigned significance value, $\alpha = 0.05$. In contrast the ATLA seems to produce much lower sizes of $< 0.001$ which could partly be attributed to the effect of the LLN.

Now considering the performance measures $p_1$ and $p_2$ for $k = 20$ outlier-contaminated samples of size $n > 100$ in Figure 2.8a) one can see both algorithm’s appear to be consistently identifying at least one of the planted outliers to a much higher degree compared to previous situations involving smaller sample sizes. However evidenced by the probability $p_3$, one finds that it is less likely for all the planted outliers to be correctly identified. This may be due to two reasons; either the larger amount of planted outliers are finding there way into more basic subsets and perhaps leading to more instances of masking and swamping. One can investigate this first possibility by considering the proportions $p_4$ and $p_5$ corresponding to masking and swamping respectively. In some situations the BACON appears to be effected by higher proportions of masking however it is still quite difficult to discern whether these are random errors or legitimate representations of the true proportions. Another possibility is that the proportionally higher amount of $n - k$ "good" observations are leading to less robust location and scale estimates. It may just be the case that with more possible subsets it is less likely for each algorithm’s forward search to consider those subsets which consist solely of "good" observations. Nevertheless one can see these probabilities $p_3$ decrease with increase sample size $n$ which may give us an indication as to why this is happening.

Now continuing on in our discussion of the simulation results one can see with more planted outliers and proportionally higher levels of contamination this leads to instances of significant breakdown in performance as shown in red. This is first seen in Table 2.8b) where the BACON algorithm appears to fail with $k = 50$, $n = 200$ and $p = 2$ which is in line with it’s quoted breakdown point of approximately 20% contamination. One can also see that the same occurs at $k = 100$, $p = 2$ and $n = 1000$ in Table 2.8c) which is equivalent to 10% contamination a figure that puts into perspective the effect the number of variables $p$ has on this breakdown point when comparing this with the $p = 5$ case in the following Table 2.8d).

It is also apparent by comparing Tables 2.8c) and d) that the performance of both algorithms significant improved for higher dimension $p$ which is consistent to what was found in previous comparisons. In fact if one were interested in seeing the performance of the BACON algorithm in samples of higher di-
mension may look to Nedret et al. (2000) for detailed simulation results. Also discernible from our simulation results is the level of contamination the ATLA is able to handle before showing evidence of breakdown with it consistently providing high probabilities of successfully identifying at least one planted outlier even with equivalent contamination levels of up to 50% at least for \( p = 2 \). However for the case of \( p = 5 \) one can finally see a significant drop in probabilities something that is consistent with the \( n - h \) outliers it is able to handle, where \( h = \lfloor (n + p + 1)/2 \rfloor \). This figure would also help in explaining why it was resulting in such a low probability of the exact identification of all planted outliers \( p_3 \) despite comparatively higher \( p_2 \) in Table 2.8c). Nevertheless these results account for the great advantage of utilising an adaptive trimming procedure.

### 2.17 ATLA relative to FSM

The procedure proposed by Riani et al. (2009) is one that also involves yet another adaptation of the forward search only it incorporates a two step process. The first step involves the search through the data for what is referred to as a signal that is a particular observation \( m^\dagger \) and therefore succeeding ones may be outliers subject to a value of a statistic that passes through a defined threshold. The following step then involves superimposing envelopes for each \( n \) from this particular point, \( m^\dagger \) until the initial instance of an introduction of a particular observation then implies that it is recognised as an outlier. Not too much detail of the procedure of this method is given here as the reader is encouraged to look to the discussions of Riani et al. (2009) and Cerioli et al. (2014).

As for previous procedures consideration for this particular method is versatile to a range of conditions in that it allows the user to define certain constraints such as the initial basic subset size as well as the criterion to be used to initialise the search. However, interestingly it does not require the user to define the significance level. Instead it chooses to set this at a relatively small value of \( \alpha = 0.01 \), supposedly to ensure a good size combined with a high power (Riani et al.; 2009).

Using the FSM.m function available in the Flexible Statistics and Data Analysis (FSDA) toolbox (Riani et al.; 2012) for MATLAB comparisons are made of its performance against ATLA. From here we will refer to this automatic outlier detection procedure by Riani et al. (2009) as simply, FSM, which represents a Forward Search in Multivariate data.
2.17.1 Simulation Results

Here samples of size $n = 50$, 100 and 200 and dimension $p = 5$ and $p = 10$ respectively are explored in order to remain consistent with the imposed conditions used in the literature from whence it was originally developed. This is also so that one can investigate how the ATLA performs in wider range of conditions to what was explored in previous simulations. As established in previous comparisons it is seen that the ATLA and some of it’s counterparts tended to show better performance for a larger number of variables, $p$. However this increase in dimension comes at the cost of computational expense particularly for the case of the ATLA.

Table 2.9: Estimated Relative efficiency of the ATLA trimmed sample mean relative to the FSM trimmed sample mean based on a simulation of $N = 1000$, samples of size $n$ and $p$-variables, consisting of $k$ planted outliers. The $n - k$ observations have been generated from the Multivariate distribution $\mathcal{N}_p(0J, I_p)$ and the $k$ planted outliers from $\mathcal{N}_p(5J, I_p)$.

<table>
<thead>
<tr>
<th></th>
<th>n=50</th>
<th></th>
<th>n=100</th>
<th></th>
<th>n=200</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>p=5</td>
<td>p=10</td>
<td>p=5</td>
<td>p=10</td>
<td>p=5</td>
</tr>
<tr>
<td>$k$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1.036</td>
<td>1.669</td>
<td>1.025</td>
<td>1.107</td>
<td>1.003</td>
</tr>
<tr>
<td>1</td>
<td>1.083</td>
<td>1.504</td>
<td>1.050</td>
<td>1.169</td>
<td>1.000</td>
</tr>
<tr>
<td>2</td>
<td>1.040</td>
<td>1.542</td>
<td>1.092</td>
<td>1.150</td>
<td>1.001</td>
</tr>
<tr>
<td>5</td>
<td>1.031</td>
<td>1.212</td>
<td>0.870</td>
<td>1.084</td>
<td>1.001</td>
</tr>
<tr>
<td>10</td>
<td>1.042</td>
<td>1.551</td>
<td>0.847</td>
<td>1.100</td>
<td>0.421</td>
</tr>
</tbody>
</table>

2.17.2 Discussion

By tabulating the relative efficiencies as seen in Table 2.9 one can establish that the ATLA tends to be the more efficient location estimator than that of FSM in the majority of cases albeit often by a small margin of a few percent. This is especially the case when considering samples of higher dimension $p = 10$ which consistently lead to higher relative efficiencies compared with its lower dimensional counterpart, $p = 5$. There are some instances however which indicate FSM to be more efficient than the ATLA particularly for cases of higher contamination with larger samples of dimension $p = 5$. For example in the case of $k = 10$ and $n = 200$ it produces a relative efficiency of 42% corresponding to its reciprocal of 237% which is quite a considerable performance improvement over the ATLA.

Focusing on the null case for the moment, that is for $k = 0$, one sees again the
advantage of the ATLA’s ability to produce low size values whereby the relative efficiency reaches its optimal value of 166% for $n = 50$ and $p = 10$.

\section*{2.18 Computational Expense}

Now as alluded to previously one disadvantage of the ATLA is that samples of higher dimension, $p$ lead to an increase in it’s computational expense and when combining this with a large sample size, $n$, this exacerbates the computational problem. It would be intuitively reasonable to assume this is partly due to the calculation and inversion of numerous covariance matrices as well as the subsequent matrix multiplications that is required for the calculation of mahalanobis distances within the forward search. Also when considering that it does not rely on stopping criterion one would expect a further penalty in terms of computational efficiency.

The requirement to obtain initial robust location and scales estimates preceding the forward search also would result in additional computational expense. For example when searching for the initial subset for the MCD algorithm the ATLA must consider,

$$
\left\lfloor \log \left( \frac{\log(0.05)}{\log \left( 1 - \frac{n}{p+1} \right)} \right) \right\rfloor
$$

different samples of size $p + 1$ in order to ensure a 95% chance of a non-contaminated starting sample and thus ensuring the estimate achieves maximum breakdown. For example in the simulation above for $p = 10$ and $n = 200$ the ATLA must consider 4,662 different samples where as for the more extreme case of say, $p = 15$, this number increases to 116,869 and thus leads to a substantial increase in the time it takes to complete the algorithm.

However such a problem is not inherent to the ATLA, in fact it is common for most multivariate outlier detection procedures and in particular those that utilise the forward search, to experience computational inefficiency for higher dimensional data, if not at a more significant degree to what was detailed here. A discussion in Schubert (2005, p. 21) details the rationale behind achieving the MCD estimate for the ATLA.

\subsection*{2.18.1 R Simulation Results}

To give an idea of the approximate computing times required for the execution of the ATLA and BACON algorithms we will utilise the `rbenchmark` package within R (Eddelbuettel; 2012).
By performing $N = 1000$ replications of the respective functions performed on simulated data of various sizes $n$ and variables $p$ it is possible to establish the average elapsed execution times (in seconds). In this instance data is randomly generated from $\mathcal{N}_p(0I_p)$ with no contamination in order to ensure the entire forward search is performed. Also note that these figures and supplementary CPU time information are provided in the appendix.

![Figure 2.6: Average Elapsed times over $N = 1000$ replications of the R ATLA [a)] and BACON [b)] functions performed on randomly generated samples from $\mathcal{N}_p(0I_p)$. For BACON $c = 4$ and $\alpha = 0.05$ was used and since $m \leq n$, $n = 50$ and $p = 15$ is not applicable. Note the vertical axes are logarithmic and are on inconsistent scales.](image)

### 2.18.2 Discussion

By comparing average elapsed times and with consideration of the inconsistent y-axis scales in Figures 2.6 it is clearly evident that the BACON algorithm is significantly faster than the ATLA. The execution times of the BACON function are almost instantaneous which is consistent with its claim of being computationally efficient.

We can also confirm that both algorithm’s appear to be similarly influenced by the sample size and variables which is to be expected seeing as each method works under the same premise. Although the computing times of the ATLA appears to be slightly more influenced by the number of variables which could be due to the initial MCD estimate. There is one instance where a larger sample size resulted in a faster elapsed time for $p = 15$ however this is not necessarily the case when considering the system CPU times (available in the appendix).

It would seem reasonable to assume that the block inflations of the BACON al-
algorithm would alleviate the influence of the sample size. However this would also result in fewer overall mahalanobis distance evaluations, a computation for which is predominantly dependent on the number of variables, $p$. Perhaps this might explain why the influence of the sample size is smaller for the BA-CON method a trait which is highlighted more clearly in supplementary tables in the appendix.

It is worth noting here that the ATLA functions within MATLAB and R are yet to be optimised according to their respective programming languages so times presented here may not be truly representative of its computational efficiency. Moreover due to the inability to control various hardware and software variables these figures should not be considered representative of the true computational complexity of the underlying algorithms.

The main purpose of presenting these figures is to demonstrate the influence of the sample size and variables and highlight the relative computing times to be expected of an algorithm regarded as computationally efficient.

2.18.3 MATLAB Simulation Results

Now comparing the MATLAB implementations of the ATLA and FSM functions we will be using the \texttt{timeit} function which calls the respective function multiple times to establish the median of the execution times.

Data will again be generated randomly from $\mathcal{N}_p(0, I_p)$ however due to inefficiencies of the ATLA function we will limit the simulation to a maximum size of $n = 500$ as well as omit the case for $p = 15$ and $n = 500$. Again raw figures from this simulation are presented in the appendix.
2.18.4 Discussion

Presenting these computing times in Figures 2.7 and with consideration of the inconsistent scales it is apparent that the FSM function is significantly faster than the MATLAB implementation of ATLA.

It is also shows that results found in the R simulation seem to corroborate with those found in this instance. However it appears that the the FSM function is less influenced by the sample size, albeit in cases involving smaller sample sizes. In addition it seems as though it may be influenced more so by the number of variables which may not be as a result of the underlying algorithm but how it has been implemented within MATLAB.

It must be stressed that the times presented here should not be compared with those presented in previous R simulations since each are programmed differently and are subject to variations in processing expenses. Again, the main intention is to highlight the relative computing times to expect of a routine that has been optimised according to the MATLAB programming language.

These comparisons demonstrate the potential for further enhancement and perhaps serve as motivation for the optimisation of the implementations of ATLA within MATLAB and R.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure2_7.png}
\caption{Median Computing times of the MATLAB ATLA [a)] and FSM [b)] functions performed on randomly generated samples from $N_p(0J, I_p)$. Note the vertical axis in a) is logarithmic and each are on differing scales.}
\end{figure}
Chapter 3

Conclusion

The act of comparing outlier detection methods can be considered a particular exhaustive one when considering the variety of established and recent methodologies and estimation techniques. Moreover the ability to impose endless combinations of generating frameworks, variable and sample size conditions and performance indicators makes it difficult to come to a concise conclusion.

However, as evidenced by our comparisons one can be confident in acknowledging that our goal of assessing the performance of various outlier detection methods including the adaptive trimmed likelihood algorithm has been realised. Through the results from our simulations and subsequent discussions one is able to establish key advantages of the ATLA including its ability to consistently produce low size calculations yet still remain sensitive once contamination is introduced into the data. A characteristic which we can say improves for increasing $n$ and a proportionally large number of variables, $p$ provided at a slight detriment to computational efficiency.

The work of Hadi et. al. regarding the early formulation (1992, 1994) of the forward search and further exploration in (2000) has paved the way in terms of outlier detection procedures and subsequently has lead to the adoption and influence of a number of the methods some of which have been explored in this thesis, and ATLA being one of these.

The rise of new procedures and subsequent comparisons remains increasingly important particularly in the field of multivariate analysis. This comes at a time of the technological era which has facilitated the demand of new and robust procedures that are able to handle the never-ending supply of data that has become synonymous with it. Although in today’s era a focus may be on computational efficiency which points to such algorithms as the BACON method proposed by Nedret et al. (2000) there still exists a significant requirement for
versatile and adaptable methods like that of the ATLA or FSM. These methods are able to achieve sufficiently low size and high power without the requirement of the user having to define a certain significance level, initial subset size or some additional input that might result in inconsistent performance or extra computational expense.

The multivariate t-distribution was briefly considered due to its ability to produce robust location and scale estimates by enabling the control of the kurtosis of the distribution facilitated by the choice of the degrees of freedom. Of course there also exists a range of robust multivariate estimation methods for which we unfortunately did not cover for this thesis for reasons involving brevity. However an effort was made to address as many methods that were relevant to the subject area while remaining consistent with the methods detailed in the associated literature.
Appendix A

R and MATLAB Programs

A.1 R Program Simulations

The following programs were built/modified under version 3.5.2 of the R package (R Core Team; 2014).

A.1.1 Demonstration of Law of Large Numbers (Figure 1.1)

The following R program simulates the average proportion of "sixes" achieved when rolling a dice $n = 1000$ times.

```R
# CoinTossLLN.R
n <- 1000 # The number of rolls of the dice
mnt <- vector("numeric",length=n) # Initialise vector

for (i in 1:n) {
  t <- rbinom(1,prob=1/6,1)
  mnt[i] <- mean(t)
  t <- -t
}
toss <- seq(1:n)
```

53
plot(toss,mnt,type="l",xlab="Number of Dice Rolls",ylab="Average", main="Average")
abline(h=1/6)
A.1.2 Onesample relative to Sample Mean (Figure 1.2):

The following R program calculates relative efficiencies of the Onesample Statistic relative to the sample mean by generating random univariate data from Tukey's Model. This program requires the function Onesample.R by Betty Mouchel & Dr Brenton Clarke (see supplementary materials in Clarke (2018)).

```
# ThesisChapter1ComputingAll.R

rm(list=ls()) # Clears objects/variables from workspace

set.seed(1); # Sets the seed so each random sample is reproducible

### Source Onesample Function ###
source("Onesample.R")

### Variables ###
nseq<-c(5,10,20,50,100) # Sample Sizes
N=10000 # Number of samples to generate
sigma=3 # Sigma_c
elmax=0.2 # Maximum epsilon to plot
elstep=0.002 # Step of epsilon's
elseq=seq(0,elmax,elstep) # Define epsilon sequence
eltotal=length(elseq) # Number of epsilons

### Initialize Permanant Variables for each sample of size N ###
xbar<-vector("numeric",length=N)
mutilde<-vector("numeric",length=N)
varxbar<-vector("numeric",length=eltotal)
varmutilde<-vector("numeric",length=eltotal)
releff<-vector("numeric",length=eltotal)
varxbarmat<-matrix(ncol=eltotal,nrow=length(nseq))
varmutildemat<-matrix(ncol=eltotal,nrow=length(nseq))
releffmat<-matrix(ncol=eltotal,nrow=length(nseq))

### Main Program ###
for (l in 1:length(nseq)) {
    n=nseq[l]
    xtemp<-vector("numeric",length=n)
    x<-vector("numeric",length=n)
    U<-vector("numeric",length=n)
```
```
varxbar<-vector("numeric",length=eltotal)
varmutilde<-vector("numeric",length=eltotal)
releff<-vector("numeric",length=eltotal)
for (k in 1:eltotal) {
  epsilon=elseq[k]
xbar<-vector("numeric",length=N)
mutilde<-vector("numeric",length=N)
avprop<-vector("numeric",length=eltotal)
  for (j in 1:N) {
    U=runif(n) # Generate n uniform random variables on U(0,1)
xtemp=rnorm(n,0,1) # Generate n normal random variables from N(0,1)
      for (i in 1:n) {
        if (U[i] < (1-epsilon)) {
          x[i]=xtemp[i] } else {
          x[i]=xtemp[i]*sigma # Generates n normal r.v's from N(0,sigma^2)
        }
      }
xbar[j]=mean(x) # Calculates x bar for N samples of size n
mutilde[j]=Onesample(data=x)$TF_n #Calculates mu tilda using OneSample
x<-rep(0,n)
}
varxbar[k]=(1/N)*sum((xbar)^2) # Calculate variance of xbars
varmutilde[k]=(1/N)*sum((mutilde)^2)
releff[k]=varxbar[k]/varmutilde[k]
}
varxbarmat[1,]<-varxbar
varmutildemat[1,]<-varmutilde
releffmat[1,]<-releff
table<-releffmat[,c(1,26,51,76,elmax)]
fullreleffmat<-releffmat

### Combined into One plot ###
plot(elseq,fullreleffmat[1,]*100,col=1,xlab=expression(paste("Epsilon (","epsilon"," )")),pch=20,main="",ylab="% Relative Efficiency",xlim=c(0,elmax), ylim=c(86, 124))
legend(0.17,100,legend=c("n = 5", "n = 10", "n = 20", "n = 50", "n = 100" ),col = 1:5,pch = 20,cex = 0.8,title="Sample Size")
```
points(elseq,fullreleffmat[2,]*100,col=2,pch=20)
points(elseq,fullreleffmat[3,]*100,col=3,pch=20)
points(elseq,fullreleffmat[4,]*100,col=4,pch=20)
points(elseq,fullreleffmat[5,]*100,col=5,pch=20)

### Individual Plots ###
plot(elseq,fullreleffmat[1,]*100,xlab=expression(paste("Epsilon (", epsilon,")")),ylim=c(85,125),lwd=2,ylab="% Relative Efficiency ",main="n=5")
plot(elseq,fullreleffmat[2,]*100,xlab=expression(paste("Epsilon (", epsilon,")")),ylim=c(85,125),pch=16,ylab="% Relative Efficiency ",main="n=10")
plot(elseq,fullreleffmat[3,]*100,xlab=expression(paste("Epsilon (", epsilon,")")),ylim=c(85,125),pch=16,ylab="% Relative Efficiency ",main="n=20")
plot(elseq,fullreleffmat[4,]*100,xlab=expression(paste("Epsilon (", epsilon,")")),ylim=c(85,125),pch=16,ylab="% Relative Efficiency ",main="n=50")
plot(elseq,fullreleffmat[5,]*100,xlab=expression(paste("Epsilon (", epsilon,")")),ylim=c(85,125),pch=16,ylab="% Relative Efficiency ",main="n=100")
A.1.3 ATLA Size and Power Calculations (Table 2.1)

The following R program calculates the power and size of the ATLA using the Tukey-Huber $\epsilon$-contamination model. It requires the function \texttt{ATLAFunc.R} provided in Appendix A.3.1 which is a small extension of \texttt{ATLA_for_Multivariate_Data.R} by Daniel Schubert & Dr Brenton Clarke (see supplementary materials in Clarke (2018)).

\begin{verbatim}
ATLAPowerContamination.R

rm(list=ls()) # Clears objects/variables from workspace

# The following packages are required for ATLAFunc.R #
library(robustbase) # Required for covMcd()
library(Hmisc)
library(survival) # Pre-installed
library(stats) # Pre-installed
library(graphics) # Pre-installed
library(splines) # Pre-installed

source("ATLAFunc.R")

library(MASS) # Required for mvrnorm
library(robustX) # Required for mvBACON

N=50000 # Number of Samples to Generate
pseq<-c(2,5) # Number of Variables
nseq<-c(25,50,100) # Sample Sizes
maxeps=0.2 # Maximum epsilon to plot
stepeps=0.1 # Step of epsilon's
epsseq=seq(0,maxeps,stepeps) # Define epsilon sequence
sig=3 # Sigma for contamination

# Initialize Vectors #
k0atlap1vec<-matrix(nrow=length(epsseq),ncol=(length(nseq)*length(pseq)),
dimnames=list(c("eps=0","eps=0.1","eps=0.2"),c("n=25","n=50","n=100","n=25","n=50","n=100")))

ptm <- proc.time() # Begin Timer

#### For loop ####

##### For loop #####

58
\end{verbatim}
for (i1 in 1:length(epsseq)) { # Epsilon Contaminaton
  eps<-epsseq[i1]
  j=1
  for (i2 in 1:length(pseq)) { # Variables
    p<-pseq[i2]
    for (i3 in 1:length(nseq)) { # Sample sizes
      n<-nseq[i3]
      nout=0
      for (l in 1:N) { # Number of Samples
        set.seed(i) # Sets seed due to ATLA resetting seed each time
        tempsamp=mvrnorm(n,rep(0,p),diag(p)) # Generate a single normal
          # random variables from N(mu,Sigma)
        sample<-matrix(ncol=p,nrow=n)
        U=runif(n) # Generate n=100 uniform random variables on U(0,1)
        for (i4 in 1:n) { # Generates each sample of size n=100
          ifelse(U[i4]<(1-eps),sample[i4,]<-tempsamp[i4,],sample[i4,]
            <-tempsamp[i4,]*sig)
        }
        ATLAout<-matrix(ATLAfunc(sample),ncol=(p)) #find ATLA outliers
        if (nrow(ATLAout)==0) {nout=nout+1} # Counts how many times
          # no outliers are identified for ATLA
        i=i+1
      }
      k0atlap1vec[i1,j]<-c((N-nout)/N)
    }
    i=i+1
  }
  j=j+1
i=i+1
}

proc.time() - ptm
round(k0atlap1vec,3)
A.1.4 ATLA relative to Sample Mean (Figure 2.4)

The following $R$ program calculates the relative efficiencies of the ATLA relative to the sample mean through data generated through the Tukey-Huber $\epsilon$-contamination model. It requires the function $ATLAmulti.R$ provided in Appendix A.3.2 which is a small extension of $ATLA_{\text{for Multivariate Data}}.R$ by Daniel Schubert & Dr Brenton Clarke (see supplementary materials in Clarke (2018)) only it outputs the mean after the flagged outliers have been trimmed.

```
Multivariate Simulation.R

rm(list=ls()) # Clears objects/variables from workspace

# The following packages are required for ATLAmulti.R #
library(robustbase) # Required for covMcd()
library(Hmisc)
library(survival) # Pre-installed
library(stats) # Pre-installed
library(graphics) # Pre-installed
library(splines) # Pre-installed

source("ATLAmulti.R") #

set.seed(1); # Sets the seed so each random sample is reproducible

library(MASS) # Required for mvrnorm

#### Variables ####
p=2 # Dimension (bivariate=2)
n=100 # Sample Size
N=5000 # Number of samples to generate
rho=0.5 # rho in Sigma off-diagonal
mu=c(rep(0,p)) # Mu_0 vector
sig=3 # sigma=sig^2 for which to multiply with Sigma
Sigma=diag(p) # Specifies diagonal elements of Sigma
Sigma[upper.tri(Sigma)]<-c(rep(rho,p*(p-1)/2))) # Specifies # upper diag elements of Sigma
Sigma[lower.tri(Sigma)]<-c(rep(rho,p*(p-1)/2))) # Specifies # lower diag elements of Sigma
elmax1=0.1 # Maximum epsilon to plot
elstep1=0.001 # Step of epsilon’s
```
elseq1=seq(0,elmax1,elstep1) # Define epsilon sequence
eltotal1=length(elseq1) # Number of epsilons

#### Initialize Permanant Variables for each sample of size N ####
detvarxbar<-vector("numeric",length=eltotal1)
detvarmutilde<-vector("numeric",length=eltotal1)
releff1<-vector("numeric",length=eltotal1)

#### Main Program ####
x<-matrix(ncol=d,nrow=n)
U<-vector("numeric",length=n)

for (k in 1:eltotal1) { # Epsilon = 0, 0.05, 0.1
  epsilon=elseq1[k] # Specifies epsilon
  xbar1<-matrix(ncol=d,nrow=N) # Intialize vectors
  mutilde1<-matrix(ncol=d,nrow=N)
  for (j in 1:N) { # Generates N=10,000 samples
    set.seed(j)
    xtemp1=mvrnorm(n,mu,Sigma) # Generate a single normal random
    # variables from N(mu,Sigma)
    x<-matrix(ncol=d,nrow=n)
    U=runif(n) # Generate n=100 uniform random variables on U(0,1)
    for (i in 1:n) { # Generates each sample of size n=100
      ifelse(U[i]<(1-epsilon),x[i,]<-xtemp1[i,],x[i,]<-xtemp1[i,]*sig)
    }
    xbar1[j,]=apply(x,2,mean) # Calculates x bar for N=10,000 samples
    # of size n=100
    mutilde1[j,]<-ATLAmulti(as.data.frame(x)) # Calculates mu tilde
    # for N=10,000 samples
    # of size n=100 using the function ATLAmulti
  }
  detvarxbar[k]= det(cov(xbar1)) # Calculate the DETERMINANT
  # of the variance of xbars
  detvarmutilde[k]=det(cov(mutilde1)) # Calculate the DETERMINANT
  # of the variance of xbars
}
releff1=detvarxbar/detvarmutilde

#### Plot ####
plot(elseq1,releff1*100,xlab=expression(paste("Epsilon (",epsilon, ")"))
\textbf{62}

\textbf{Relative Efficiency},\textbf{ylim=c(100,180)},\textbf{main=""})
A.1.5 ATLA relative to Hadi’s FS (Tables 2.3 and 2.4)

The following R program calculates the performance measures of the ATLA against Hadi’s (1994) procedure through data generated through the mean slippage model. It requires the function \textit{ATLA\_func.R} provided in Appendix A.3.1 which is a small extension of \textit{ATLA\_for\_Multivariate\_Data.R} by Daniel Schubert & Dr Brenton Clarke (see supplementary materials in Clarke (2018). It also requires the function \textit{Hadi1994funcIndout.R} which is available in Appendix A.3.3.

\begin{verbatim}
ATLAvsHadi1994.R

rm(list=ls()) # Clears objects/variables from workspace

# The following packages are required for ATLAfunc.R#
library(robustbase) # Required for covMcd()
library(Hmisc)
library(survival) # Pre-installed
library(stats) # Pre-installed
library(graphics) # Pre-installed
library(splines) # Pre-installed

source('Hadi1994funcIndout.R')
source('ATLAfunc.R')

library(MASS) # Required for mvrnorm

N=1000 # Number of Samples to Generate
kseq<-c(1,5,10) # Number of Outliers
pseq<-c(2,5) # Number of Variables
nseq<-c(25,50,100) # Sample Sizes
jmult=5 # Covariance multiplier

# Intialize Vectors#
heads<-list(c("k=1","k=5","k=10"),c("n=25","n=50","n=100","n=25","n=50","n=100"))
heads2<-list(c("k=0"),c("n=25","n=50","n=100","n=25","n=50","n=100"))
atlap1vec<-matrix(nrow=length(kseq),ncol=(length(nseq)*length(pseq)),
dimnames=heads)
atlap2vec<-atlap1vec
atlap3vec<-atlap1vec
\end{verbatim}
atlap4vec<-atlap1vec
atlap5vec<-atlap1vec
hadip1vec<-atlap1vec
hadip2vec<-atlap1vec
hadip3vec<-atlap1vec
hadip4vec<-atlap1vec
hadip5vec<-atlap1vec
k0atlap1vec<-matrix(nrow=1,ncol=(length(nseq)*length(pseq)),dimnames=heads2)
k0hadip1vec<-matrix(nrow=1,ncol=(length(nseq)*length(pseq)),dimnames=heads2)

i=1;j=1;
i1=1; i2=1; i3=1
for (i1 in 1:length(kseq)) { # Outliers
  k<-kseq[i1]
  j=1
  for (i2 in 1:length(pseq)) { # Variables
    p<-pseq[i2]

    for (i3 in 1:length(nseq)) { # Sample sizes
      n<-nseq[i3]
      nout=0
      correctout=0
      incorrectout=0
      missedout=0
      nout1=0
      correctout1=0
      incorrectout1=0
      missedout1=0

      for (l in 1:N) { # Number of Samples
        set.seed(i) # Sets seed due to ATLA reseting seed to 1 each time
        invec<-mvrnorm(n-k,rep(0,p),diag(p)) # Generates n-k observations
        outvec<-mvrnorm(k,jmult*rep(1,p),diag(p)) # Generates k outliers
        sample<-rbind(cbind(invec,rep(0,n-k)),cbind(matrix(outvec,ncol=p), rep(1,k))) # Joins outliers and normal observations and a dummy variable
# to signify whether it is a planted outlier
ATLAout<-matrix(ATLAfunc(sample),ncol=(p+1)) # find ATLA outliers
hadiout<-Hadi1994funcIndout(sample,0.05) # find outliers using Hadi 1994

if (nrow(ATLAout)==0) {nout=nout+1} else { # Counts how many times no outliers are identified
  if (sum(ATLAout[,,(p+1)])>0) {correctout=correctout+1} # Counts how many times planted outliers are successfully identified
  if (sum(ATLAout[,,(p+1)])==k & mean(ATLAout[,,(p+1)])==1) {allcorrectout=allcorrectout+1} else { # Counts how many times all planted outliers are successfully identified
    if (any(ATLAout[,,(p+1)]==0)) {incorrectout=incorrectout+sum(ATLAout[,,(p+1)]==0)} else { # Counts how many times non-planted outliers are identified
      missedout=missedout+(k-sum(ATLAout[,,(p+1)])) # Counts how many times planted outliers are not identified
    }
  }
}

if (is.character(hadiout)) {nout1=nout1+1} else { # Counts how many times no outliers are identified
  if (sum(hadiout[,,(p+1)])>0) {correctout1=correctout1+1} # Counts how many times planted outliers are successfully identified
  if (sum(hadiout[,,(p+1)])==k & mean(hadiout[,,(p+1)])==1) {allcorrectout1=allcorrectout1+1} else { # Counts how many times all planted outliers are successfully identified
    if (any(hadiout[,,(p+1)]==0)) {incorrectout1=incorrectout1+sum(hadiout[,,(p+1)]==0)} else { # Counts how many times non-planted outliers are identified
      missedout1=missedout1+(k-sum(hadiout[,,(p+1)])) # Counts how many times planted outliers are not identified
    }
  }
}
i=i+1

atlap1vec[i1,j]<-c((N-nout)/N)
atlap2vec[i1,j]<-c(correctout/N)
atlap3vec[i1,j]<-c(allcorrectout/N)
atlap4vec[i1,j]<-c(missedout/(N*n))
atlap5vec[i1,j]<-c(incorrectout/(N*n))

hadip1vec[i1,j]<-c((N-nout1)/N)
hadip2vec[i1,j]<-c(correctout1/N)
hadip3vec[i1,j]<-c(allcorrectout1/N)
hadip4vec[i1,j]<-c(missedout1/(N*n))
hadip5vec[i1,j]<-c(incorrectout1/(N*n))
j=j+1
i=i+1

#### For loop for k=0 i.e. no outliers ####
k<-0; i=1; j=1; i1=1; i2=1; i3=1
for (i2 in 1:length(pseq)) { # Variables
  p<-pseq[i2]
  for (i3 in 1:length(nseq)) { # Sample sizes
    n<-nseq[i3]
    nout=0
    nout1=0
    for (l in 1:N) { # Number of Samples
      set.seed(i) # Sets seed due to ATLA reseting seed to 1 each time
      sample<-mvrnorm(n-k,rep(0,p),diag(p)) # Generates n-k observations
      ATLAout<-matrix(ATLAfunc(sample),ncol=(p)) # find ATLA outliers
      hadiout<-Hadi1994funcIndout(sample,0.05) # find outliers using # Hadi 1994
      if (nrow(ATLAout)==0) {nout=nout+1} # Counts how many times no
      # outliers are identified for ATLA
      if (is.character(hadiout)) {nout1=nout1+1} # Counts how many times
      # no outliers are identified for Hadi 1994
      i=i+1
    }
k0atlap1vec[i1,j]<-c((N-nout)/N)
k0hadip1vec[i1,j]<-c((N-nout1)/N)
j=j+1
i = i + 1
}
A.1.6 ATLA relative to BACON (Tables 2.5 and 2.6)

The following $R$ program calculates the relative efficiencies of ATLA trimmed sample mean against the BACON method for significance level $\alpha = 0.05$ and subset size $m = cp$ where $c = 4$. It requires the function `ATLAfunc.R` provided in Appendix A.3.1.

```r
ATLAvsBACON.R

rm(list=ls()) # Clears objects/variables from workspace

# The following packages are required for ATLAfunc.R #
library(robustbase) # Required for covMcd()
library(Hmisc)
library(survival) # Pre-installed
library(stats) # Pre-installed
library(graphics) # Pre-installed
library(splines) # Pre-installed

source('ATLAfunc.R')

library(MASS) # Required for mvrnorm
library(robustX) # Required for mvBACON

N=1000 # Number of Samples to Generate
kseq<-c(1,5,10) # Number of Outliers (1,5,10)
pseq<-c(2,5) # Number of Variables
nseq<-c(25,50,100) # Sample Sizes (25,50,100)
jmult=4 # Covariance multiplier
c=4 # Integer for size of initials basic subset in Bacon

# Intialize Vectors #
heads<-list(c("k=1","k=5","k=10"),c("n=25","n=50","n=100","n=25","n=50","n=100"))
heads2<-list(c("k=0"),c("n=25","n=50","n=100","n=25","n=50","n=100"))
atlap1vec<-matrix(nrow=length(kseq),ncol=(length(nseq)*length(pseq)),dimnames=heads)
atlap2vec<-atlap1vec
atlap3vec<-atlap1vec
atlap4vec<-atlap1vec
atlap5vec<-atlap1vec
```

68
baconp1vec<-atlap1vec
bacn0p2vec<-atlap1vec
baconp3vec<-atlap1vec
baconp4vec<-atlap1vec
baconp5vec<-atlap1vec
k0atlap1vec<-matrix(nrow=1,ncol=(length(nseq)*length(pseq)),
dimnames=heads2)
k0baconp1vec<-matrix(nrow=1,ncol=(length(nseq)*length(pseq)),
dimnames=heads2)

i=1; j=1;
i1=1; i2=1; i3=1
for (i1 in 1:length(kseq)) { # Outliers
  k<-kseq[i1]
  j=1
  for (i2 in 1:length(pseq)) { # Variables
    p<-pseq[i2]

    for (i3 in 1:length(nseq)) { # Sample sizes
      n<-nseq[i3]
      nout=0
      correctout=0
      incorrectout=0
      missedout=0
      nout1=0
      correctout1=0
      allcorrectout=0
      incorrectout1=0
      missedout1=0

      for (l in 1:N) { # Number of Samples
        set.seed(i) # Sets seed due to ATLA resetting seed to 1 each time
        invec<-mvrnorm(n-k,rep(0,p),diag(p)) # Generates n-k observations
        outvec<-mvrnorm(k,jmult*rep(1,p),diag(p)) # Generates k outliers
        sample<-rbind(cbind(invec,rep(0,n-k)),cbind(matrix(outvec,ncol=p),
                      rep(1,k)))) # Joins outliers and normal observations and a dummy
        # variable to signify whether it is a planted outlier
        ATLAout<-matrix(ATLAfunc(sample),ncol=(p+1)) # find ATLA outliers
      }
    }
  }
}
bacon<-mvBACON(sample[,-(p+1)],alpha=0.05,m=(c*p),verbose=FALSE)
baconout<-matrix(sample[!bacon$subset],ncol=(p+1)) # find outliers using BACON

if (nrow(ATLAout)==0) {nout=nout+1} else { # Counts how many times no outliers are identified
  if (sum(ATLAout[,,(p+1)])>0) {correctout=correctout+1} # Counts how many times planted outliers are successfully identified
  if (sum(ATLAout[,,(p+1)])==k & mean(ATLAout[,,(p+1)])==1) {allcorrectout=allcorrectout+1} else { # Counts how many times all planted outliers are successfully identified
    if (any(ATLAout[,,(p+1)]==0)) {incorrectout=incorrectout+sum(ATLAout[,,(p+1)]==0)} else { # Counts how many times non-planted outliers are identified
      missedout=missedout+(k-sum(ATLAout[,,(p+1)]))) # Counts how many times planted outliers are not identified
  }
}

if (nrow(baconout)==0) {nout1=nout1+1} else { # Counts how many times no outliers are identified
  if (sum(baconout[,,(p+1)])>0) {correctout1=correctout1+1} # Counts how many times planted outliers are successfully identified
  if (sum(baconout[,,(p+1)])==k & mean(baconout[,,(p+1)])==1) {allcorrectout1=allcorrectout1+1} else { # Counts how many times all planted outliers are successfully identified
    if (any(baconout[,,(p+1)]==0)) {incorrectout1=incorrectout1+sum(baconout[,,(p+1)]==0)} else { # Counts how many times non-planted outliers are identified
      missedout1=missedout1+(k-sum(baconout[,,(p+1)]))) # Counts how many times planted outliers are not identified
  }
}
i=i+1

atlap1vec[i1,j]<-c((N-nout)/N)
atlap2vec[i1,j]<-c(correctout/N)
atlap3vec[i1,j]<-c(allcorrectout/N)
atlap4vec[i1,j]<-c(missedout/(N*n))
atlap5vec[i1,j]<-c(incorrectout/(N*n))
baconp1vec[i1,j]<-c((N-nout1)/N)
baconp2vec[i1,j]<-c(correctout1/N)
baconp3vec[i1,j]<-c(allcorrectout1/N)
baconp4vec[i1,j]<-c(missedout1/(N*n))
baconp5vec[i1,j]<-c(incorrectout1/(N*n))
j=j+1
i=i+1
}
i=i+1
}
i=i+1
}

#### For loop for k=0 i.e. no outliers ####
k<-0; i=1; j=1; i1=1; i2=1; i3=1
for (i2 in 1:length(pseq)) { # Variables
  p<-pseq[i2]
  for (i3 in 1:length(nseq)) { # Sample sizes
    n<-nseq[i3]
    nout=0
    nout1=0
    for (l in 1:N) { # Number of Samples
      set.seed(i) # Sets seed due to ATLA resetting seed to 1 each time
      sample<-mvnorm(n-k,rep(0,p),diag(p)) # Generates n-k observations
      ATLAout<-matrix(ATLAfunc(sample),ncol=(p)) # find ATLA outliers
      bacon<-mvBACON(sample[,-(p+1)],alpha=0.05,m=(c*p),
                    verbose=FALSE)
      baconout<-matrix(sample[!bacon$subset],ncol=(p)) # find outliers
      # using BACON
      if (nrow(ATLAout)==0) {nout=nout+1} # Counts how many times
      # no outliers are identified for ATLA
      if (nrow(baconout)==0) {nout1=nout1+1} # Counts how many
      # times no outliers are identified for BACON
      i=i+1
    }
    k0atlap1vec[i1,j]<-c((N-nout)/N)
k0baconp1vec[i1,j]<-c((N-nout1)/N)
  j=j+1
}
i=i+1
A.1.7 ATLA relative to BACON for Large $n$ (Tables 2.7 and 2.8)

The following $R$ program calculates the relative efficiencies of ATLA trimmed sample mean against the BACON method for significance level $\alpha = 0.05$ and subset size $m = cp$ where $c = 4$. It requires the function `ATLAfunc.R` provided in Appendix A.3.1.

```
ATLAvsBACONLargeN.R

rm(list=ls()) # Clears objects/variables from workspace

# The following packages are required for ATLAfunc.R#
library(robustbase) # Required for covMcd()
library(Hmisc)
library(survival) # Pre-installed
library(stats) # Pre-installed
library(graphics) # Pre-installed
library(splines) # Pre-installed

source('ATLAfunc.R')

library(MASS) # Required for mvrnorm
library(robustX) # Required for mvBACON

N=1000 # Number of Samples to Generate
kseq<-c(20,50,100) # Number of Outliers (1,5,10)
pseq<-c(2,5) # Number of Variables
nseq<-c(200,500,1000) # Sample Sizes (25,50,100)
jmult=4 # Covariance multiplier
c=4 # Integer for size of initials basic subset in Bacon

# Intialize Vectors#
heads<-list(c("k=1","k=5","k=10"),c("n=25","n=50","n=100","n=25","n=50","n=100", "n=100"))
heads2<-list(c("k=0"),c("n=25","n=50","n=100","n=25","n=50","n=100"))
atlap1vec<-matrix(nrow=length(kseq),ncol=(length(nseq)*length(pseq)),
dimnames=heads)
atlap2vec<-atlap1vec
atlap3vec<-atlap1vec
atlap4vec<-atlap1vec
```

73
atlap5vec<-atlap1vec
baconp1vec<-atlap1vec
baconp2vec<-atlap1vec
baconp3vec<-atlap1vec
baconp4vec<-atlap1vec
baconp5vec<-atlap1vec
k0atlap1vec<-matrix(nrow=1,ncol=(length(nseq)*length(pseq)),
dimnames=heads2)
k0baconp1vec<-matrix(nrow=1,ncol=(length(nseq)*length(pseq)),
dimnames=heads2)

i=1; j=1;
i1=1; i2=1; i3=1
for (i1 in 1:length(kseq)) { # Outliers
  k<-kseq[i1]
  j=1
  for (i2 in 1:length(pseq)) { # Variables
    p<-pseq[i2]

    for (i3 in 1:length(nseq)) { # Sample sizes
      n<-nseq[i3]
      nout=0
      correctout=0
      allcorrectout=0
      incorrectout=0
      missedout=0
      nout1=0
      correctout1=0
      allcorrectout1=0
      incorrectout1=0
      missedout1=0

      for (l in 1:N) { # Number of Samples
        set.seed(i) # Sets seed due to ATLA reseting seed to 1 each time
        invec<-mvrnorm(n-k,rep(0,p),diag(p)) # Generates n-k observations
        outvec<-mvrnorm(k,jmult*rep(1,p),diag(p)) # Generates k outliers
        sample<-rbind(cbind(invec,rep(0,n-k)),cbind(matrix(outvec,ncol=p)
          ,rep(1,k))) # Joins outliers and normal observations and a dummy
        # variable to signify whether it is a planted outlier

      } # Number of Samples
    } # Sample sizes
  } # Variables
}
} # Outliers
ATLAout <- matrix(ATLAFunc(sample), ncol=(p+1)) # find ATLA outliers
bacon <- mvBACON(sample[,-(p+1)], alpha=0.05, m=(c*p),
verbose=FALSE)
baconout <- matrix(sample[!bacon$subset], ncol=(p+1)) # find
# outliers using BACON

if (nrow(ATLAout)==0) {nout=nout+1} else { # Counts how many
# times no outliers are identified
  if (sum(ATLAout[, (p+1)])>0) {correctout=correctout+1} #
  # Counts how many times planted outliers are succesfully identified
  if (sum(ATLAout[, (p+1)])==k & mean(ATLAout[, (p+1)])==1) {
    allcorrectout=allcorrectout+1}
else {
  # Counts how many times all
  # planted outliers are succesfully identified
    if (any(ATLAout[, (p+1)]==0)) {incorrectout=incorrectout
+sum(ATLAout[, (p+1)]==0)} else { # Counts how many times non-planted
  # outliers are identified
          missedout=missedout+(k-sum(ATLAout[, (p+1)]))} # Counts how
  # many times planted outliers are not identified
  }
  }

if (nrow(baconout)==0) {nout1=nout1+1} else { # Counts how many
  # times no outliers are identified
    if (sum(baconout[, (p+1)])>0) {correctout1=correctout1+1} #
    #Counts how many times planted outliers are succesfully identified
    if (sum(baconout[, (p+1)])==k & mean(baconout[, (p+1)])==1) {
      allcorrectout1=allcorrectout1+1}
  else {
    # Counts how many times all
    # planted outliers are succesfully identified
      if (any(baconout[, (p+1)]==0)) {incorrectout1=incorrectout1+
      sum(baconout[, (p+1)]==0)} else { # Counts how many times non-planted
    # outliers are identified
            missedout1=missedout1+(k-sum(baconout[, (p+1)]))} # Counts
  # how many times planted outliers are not identified
  }
  }
  i=i+1
atlap1vec[i1,j]<-c((N-nout)/N)
atlap2vec[i1,j]<-c(correctout/N)
atlap3vec[i1,j]<-c(allcorrectout/N)
atlap4vec[i1,j]<-c(missedout/(N*n))
atlap5vec[i1,j]<-c(incorrectout/(N*n))
baconp1vec[i1,j]<-c((N-nout1)/N)
baconp2vec[i1,j]<-c(correctout1/N)
baconp3vec[i1,j]<-c(allcorrectout1/N)
baconp4vec[i1,j]<-c(missedout1/(N*n))
baconp5vec[i1,j]<-c(incorrectout1/(N*n))
j=j+1
i=i+1
}
i=i+1
}

#### For loop for k=0 i.e. no outliers ####
k<-0; i=1; j=1; i1=1; i2=1; i3=1
for (i2 in 1:length(pseq)) { # Variables
p<-pseq[i2]
for (i3 in 1:length(nseq)) { # Sample sizes
n<-nseq[i3]
nout=0
nout1=0
for (l in 1:N) { # Number of Samples
set.seed(i) # Sets seed due to ATLA resetting seed to 1 each time
sample<-mvrnorm(n-k,rep(0,p),diag(p)) # Generates n-k observations
ATLAout<-matrix(ATLAfunc(sample),ncol=(p)) # find ATLA outliers
bacon<-mvBACON(sample[,-(p+1)],alpha=0.05,m=(c*p),
verbose=FALSE)
baconout<-matrix(sample[!bacon$subset],ncol=(p)) # find outliers
# using BACON
if (nrow(ATLAout)==0) {nout=nout+1} # Counts how many times
# no outliers are identified for ATLA
if (nrow(baconout)==0) {nout1=nout1+1} # Counts how many
# times no outliers are identified for BACON
i=i+1
}
\[ \text{k0atlap1vec}[i1,j] <- \text{c}((N - \text{nout})/N) \]
\[ \text{k0baconp1vec}[i1,j] <- \text{c}((N - \text{nout1})/N) \]
\[ j = j + 1 \]
\}
\}

77
A.1.8  R Computing Times (Figures 2.6)

The following R program times the execution of the ATLA and BACON functions over \( N = 1000 \) replications. It requires the function \( ATLAfunc.R \) provided in Appendix A.3.1.

---

*ATLAvsBACONBenchmark.R*

```r
rm(list=ls())  # Clears objects/variables from workspace

# The following packages are required for ATLAfunc.R #
library(robustbase)  # Required for covMcd()
library(Hmisc)
library(survival)  # Pre-installed
library(stats)  # Pre-installed
library(graphics)  # Pre-installed
library(splines)  # Pre-installed

source('ATLAfunc.R')

library(MASS)  # Required for mvrnorm
library(robustX)  # Required for mvBACON

library(rbenchmark)  # Required for benchmark function

c=4  # Used in mvBacon() to specify size of subsets

p=100  # Arbitrary value to avoid deleting of actual data
N=1000
pvec<-c(5,10,15)
nvec<-c(50,100,200,500,1000)

# ONLY ATLA

set.seed(1)
bench<-benchmark("ATLAp5n50" = {
  ATLAfunc(mvrnorm(nvec[1],rep(0,pvec[1]),diag(pvec[1]))),
},
"ATLAp10n50" = {
  ATLAfunc(mvrnorm(nvec[1],rep(0,pvec[2]),diag(pvec[2]))),
},
"ATLAp15n50" = {

78```
ATLAfunc(mvrnorm(nvec[1],rep(0,pvec[3]),diag(pvec[3])))

"ATLAp5n100" = {
    ATLAfunc(mvrnorm(nvec[2],rep(0,pvec[1]),diag(pvec[1])))
},
"ATLAp10n100" = {
    ATLAfunc(mvrnorm(nvec[2],rep(0,pvec[2]),diag(pvec[2])))
},
"ATLAp15n100" = {
    ATLAfunc(mvrnorm(nvec[2],rep(0,pvec[3]),diag(pvec[3])))
},
"ATLAp5n200" = {
    ATLAfunc(mvrnorm(nvec[3],rep(0,pvec[1]),diag(pvec[1])))
},
"ATLAp10n200" = {
    ATLAfunc(mvrnorm(nvec[3],rep(0,pvec[2]),diag(pvec[2])))
},
"ATLAp15n200" = {
    ATLAfunc(mvrnorm(nvec[3],rep(0,pvec[3]),diag(pvec[3])))
},
"ATLAp5n500" = {
    ATLAfunc(mvrnorm(nvec[4],rep(0,pvec[1]),diag(pvec[1])))
},
"ATLAp10n500" = {
    ATLAfunc(mvrnorm(nvec[4],rep(0,pvec[2]),diag(pvec[2])))
},
"ATLAp15n500" = {
    ATLAfunc(mvrnorm(nvec[4],rep(0,pvec[3]),diag(pvec[3])))
},
"ATLAp5n1000" = {
    ATLAfunc(mvrnorm(nvec[5],rep(0,pvec[1]),diag(pvec[1])))
},
"ATLAp10n1000" = {
    ATLAfunc(mvrnorm(nvec[5],rep(0,pvec[2]),diag(pvec[2])))
},
"ATLAp15n1000" = {
    ATLAfunc(mvrnorm(nvec[5],rep(0,pvec[3]),diag(pvec[3])))
},
replications = N,
order="sys.self",
pvar<-rep(seq(5,15,5),length(nvec))
nvar<-c(rep(nvec[1],length(pvec)),rep(nvec[2],length(pvec)),rep(nvec[3],length(pvec)),rep(nvec[4],length(pvec)),rep(nvec[5],length(pvec)))
benchord<-bench[order(as.numeric(row.names(bench))),]

bardata<-matrix(benchord$elapsed,ncol=3,byrow=T)
rownames(bardata)<-c("p=5","p=10","p=15")
colnames(bardata)<-c("n=50","n=100","n=200","n=500","n=1000")

barplot(bardata/N,beside=TRUE,log="y",col=brewer.pal(5, "Greens"),
legend = rownames(bardata),args.legend = list(x="topleft"),xlab="Variables",ylab="Average Elapsed Time (in sec)",main="ATLA")

# ONLY BACON #
set.seed(1)
BACONbench<-benchmark(
  "BACONp5n50" = {mvBACON(mvrnorm(nvec[1],rep(0,pvec[1])),diag(pvec[1])),alpha=0.05, m=(c*pvec[1]),verbose=FALSE},
  "BACONp10n50" = {mvBACON(mvrnorm(nvec[1],rep(0,pvec[2])),diag(pvec[2])),alpha=0.05, m=(c*pvec[1]),verbose=FALSE},
  "BACONp5n100" = {mvBACON(mvrnorm(nvec[2],rep(0,pvec[1])),diag(pvec[1])),alpha=0.05, m=(c*pvec[1]),verbose=FALSE},
  "BACONp10n100" = {mvBACON(mvrnorm(nvec[2],rep(0,pvec[2])),diag(pvec[2])),alpha=0.05, m=(c*pvec[2]),verbose=FALSE},
  "BACONp15n100" = {mvBACON(mvrnorm(nvec[2],rep(0,pvec[3])),diag(pvec[3])),alpha=0.05, m=(c*pvec[3]),verbose=FALSE})
BACONp5n200 = {
    mvBACON(mvrnorm(nvec[3], rep(0, pvec[1]), diag(pvec[1])), alpha=0.05,
    m=(c*pvec[1]), verbose=FALSE)
},
BACONp10n200 = {
    mvBACON(mvrnorm(nvec[3], rep(0, pvec[2]), diag(pvec[2])), alpha=0.05,
    m=(c*pvec[2]), verbose=FALSE)
},
BACONp15n200 = {
    mvBACON(mvrnorm(nvec[3], rep(0, pvec[3]), diag(pvec[3])), alpha=0.05,
    m=(c*pvec[3]), verbose=FALSE)
},
BACONp5n500 = {
    mvBACON(mvrnorm(nvec[4], rep(0, pvec[1]), diag(pvec[1])), alpha=0.05,
    m=(c*pvec[1]), verbose=FALSE)
},
BACONp10n500 = {
    mvBACON(mvrnorm(nvec[4], rep(0, pvec[2]), diag(pvec[2])), alpha=0.05,
    m=(c*pvec[2]), verbose=FALSE)
},
BACONp15n500 = {
    mvBACON(mvrnorm(nvec[4], rep(0, pvec[3]), diag(pvec[3])), alpha=0.05,
    m=(c*pvec[3]), verbose=FALSE)
},
BACONp5n1000 = {
    mvBACON(mvrnorm(nvec[5], rep(0, pvec[1]), diag(pvec[1])), alpha=0.05,
    m=(c*pvec[1]), verbose=FALSE)
},
BACONp10n1000 = {
    mvBACON(mvrnorm(nvec[5], rep(0, pvec[2]), diag(pvec[2])), alpha=0.05,
    m=(c*pvec[2]), verbose=FALSE)
},
BACONp15n1000 = {
    mvBACON(mvrnorm(nvec[5], rep(0, pvec[3]), diag(pvec[3])), alpha=0.05,
    m=(c*pvec[3]), verbose=FALSE)
},
replications = N,
order="relative",
columns = c("test", "replications", "elapsed",

81
BACONbench1ord<-BACONbench[order(as.numeric(row.names(BACONbench))),]
pvar<-rep(seq(5,15,5),length(nvec))
pvar<-pvar[-3]
nvar<-c(rep(nvec[1],length(pvec)),rep(nvec[2],length(pvec)),rep(nvec[3],
length(pvec)),rep(nvec[4],length(pvec)),rep(nvec[5],length(pvec)))
nvar<-nvar[-3]

bar2data<-matrix(c(BACONbench1ord$elapsed[1:2],0,BACONbench1ord$elapsed[3:14]),ncol=3,byrow=T)
colnames(bar2data)<-c("p=5","p=10","p=15")
rownames(bar2data)<-c("n=50","n=100","n=200","n=500","n=1000")

barplot(bar2data/N,beside=TRUE,log="y",col=brewer.pal(5, "Blues"),legend=
rownames(bar2data),args.legend = list(x="topleft"),xlab="Variables",
ylab="Average Elapsed Time (in sec)",main="BACON")
A.2 MATLAB Program Simulations

The following programs were built/modified under MATLAB version R2018b (MATLAB; 2018).

A.2.1 ATLA relative to T-distribution (Table 2.2)

The following MATLAB program calculates the relative efficiencies of ATLA trimmed sample mean relative to the t-distribution on $\nu = 4$ degrees of freedom by use of an EM algorithm. It requires the function \texttt{fitt\_fixnu.m} available from the \texttt{tdistfit} package available on GitHub provided by Ince (2013) as well as the \texttt{ATLAfuncNew.m} function provided in Appendix A.4.1.

\[
TdistvsATLAdraft.m
\]

```matlab
clear
pvec=[2 5];
N=1000;
nvec=[25 50 100];
kvec=[0 1 2 5 10];
jmult=5; sig=3;
df=4;  % Degrees of Freedom for the fitted t-distribution using
% the EM-algorithm

tcovdet = zeros(length(pvec)*length(nvec)*length(kvec),1);
atlacovdet = zeros(length(pvec)*length(nvec)*length(kvec),1);
l=1;
% set.seed(1)

for o=1:length(kvec)
    k=kvec(o);
    for i=1:length(pvec)
        p=pvec(i);
        mu = ones(p,1);
        sigma = eye(p);
        for j=1:length(nvec)
            n=nvec(j);
            tmu = zeros(N,p);  \%\%
            atlamu = zeros(N,p);
            for m=1:N
                Y1 = mvnrnd(mu*0,sigma,n-k);
```
if k > 0
    Y1out = mvnrnd(mu*jmult, sigma, k);
    Y1 = [Y1; Y1out];
end
atlaout = ATLAfuncNew(Y1);
Y2 = Y1;
if isnan(atlaout(1)) == 0
    Y1(atlaout,:) = []; % Trims atlaout from Y1
end
tmu(m,:) = fitt_fixnu(Y2, df)-(k/n)*jmult*mu';
atlamu(m,:) = mean(Y1)-(k/n)*jmult*mu';
end
tcovdet(l) = det(cov(tmu));
atlacovdet(l) = det(cov(atlamu));
l=1+l;
end
end

k0detcovatla = atlacovdet(1:6,:);
k0detcovt = tcovdet(1:6,:);
releffmat1 = tcovdet./atlacovdet;
releffmat2 = atlacovdet./tcovdet;
A.2.2 ATLA relative to T-distribution (Table 2.5)

The following MATLAB program calculates the relative efficiencies of ATLA trimmed sample mean relative to the t-distribution on \( \nu = 4 \) degrees of freedom by use of an EM algorithm. For this particular program data has been generated from the Tukey-Huber \( \epsilon \)-contamination model. It requires the function \texttt{fitt\_fixnu} available from the \texttt{tdistfit} package available on GitHub provided by Ince (2013) as well as the \texttt{ATLAfuncNew} function provided in Appendix A.4.1.

\begin{verbatim}
TdistvsATLAContamination.m

clear
N=1000;
nvec=[25 50 100];
sig=3;
df=4;  % Degrees of Freedom for the fitted t-distribution using the  
  % EM-algorithm
epsmax=0.1;  % Maximum level of contamination
epstep=0.01;  % Contaminaton increments for graph

epsvec=0:epstep:epsmax;
tcovdetr2 = zeros(length(nvec)*length(epsvec),1);
atlacovdetr2 = zeros(length(nvec)*length(epsvec),1);
tcovdetr5 = zeros(length(nvec)*length(epsvec),1);
atlacovdetr5 = zeros(length(nvec)*length(epsvec),1);
l=1;


p=2;
for j=1:length(nvec)
    n=nvec(j);
    tmu = zeros(N,p);  \texttt{%%%}
    atlamu = zeros(N,p);
    for o=1:length(epsvec)
        eps=epsvec(o);
        for m=1:N
            U = rand(n,1);  \texttt{%%%}
            Y1temp = mvnrnd(zeros(p,1),eye(p),n);
            \texttt{%%%}
            Y1 = zeros(n,p);
        end
    end
end
\end{verbatim}
\[
\text{logind} = \text{ones}(n,1) + \left( (U \geq (1-\text{eps})) \right) \times (\text{sig}-1);
\]
\[
\text{Y1} = \text{Y1temp} \times \text{logind}; \quad \% \text{This is a simplified version}
\]
\%

% of the \%code below inorder to improve efficiency
\%

\%for i=1:n
\% if U(i) < (1-eps)
\% \quad Y1(i,:) = Y1temp(i,:);
\% else
\% \quad Y1(i,:) = Y1temp(i,:)\times\text{sig};
\% end
\%end

atlaout = \text{ATLAfuncNew}(Y1);
Y2 = Y1;
if isnan(atlaout(1)) == 0
\quad Y1(atlaout,:) = [];
end
tmu(m,:) = \text{fitt\_fixnu}(Y2, df);
atlamu(m,:) = \text{mean}(Y1);
end
tcovdetp2(l) = \text{det}(\text{cov}(\text{tmu}));
atlacovdetp2(l) = \text{det}(\text{cov}(\text{atlamu}));
l=l+1;
end
end

p=5; l=1;
for j=1:length(nvec)
\quad n=nvec(j);
\quad tmu = \text{zeros}(N,p); \quad \%%\%%
\quad atlamu = \text{zeros}(N,p);
\quad for o=1:length(epsvec)
\quad \quad \quad eps=epsvec(o);
\quad \quad \quad for m=1:N
\quad \quad \quad \quad U = \text{rand}(n,1); \quad \% \text{Generate n random varables between 0 and 1}
\quad \quad \quad \quad \text{Y1temp} = \text{mvnrnd}(\text{zeros}(p,1),\text{eye}(p),n);
\quad \quad \quad \quad \%Y1 = \text{zeros}(n,p);
\quad \quad \quad \quad \quad \text{logind} = \text{ones}(n,1) + \left( (U \geq (1-\text{eps})) \right) \times (\text{sig}-1);
\quad \quad \quad \quad \text{Y1} = \text{Y1temp} \times \text{logind}; \quad \% \text{This is a simplified version}
% of the code below in order to improve efficiency

\texttt{%for i=1:n}
\texttt{% if U(i) < (1-\texttt{eps})
\texttt{Y1(i,:) = Y1temp(i,:);}
\texttt{% else
\texttt{Y1(i,:) = Y1temp(i,:)*sig;}
\texttt{% end
\texttt{%end

\texttt{atlaout = ATLAfuncNew(Y1);}
\texttt{Y2 = Y1;}
\texttt{if isnan(atlaout(1)) == 0}
\texttt{\hspace{1cm}Y1(atlaout,:) = [];
\texttt{end

\texttt{tmu(m,:) = fitt\_fixnu(Y2, df);}
\texttt{atlamu(m,:) = mean(Y1);

end

\texttt{tcovdtep5(l) = det(cov(tmu));}
\texttt{atlacovdtep5(l) = det(cov(atlamu));}
\texttt{l=l+1;
end

\texttt{releffmatp2 = tcovdtep2/atlacovdtep2;
releffmatp5 = tcovdtep5/atlacovdtep5;

%%% Figures %%%

\texttt{colvec=['r','m','b'];}
\texttt{pvec=['2' '5'];

for k=1:2

\hspace{1cm}if k==1
\hspace{1cm}releff = releffmatp2;
\hspace{1cm}else
\hspace{1cm}releff = releffmatp5;
\hspace{1cm}end
\hspace{1cm}subplot(1,2,k);
\hspace{1cm}l=1;
\hspace{1cm}for i=1:length(nvec)
plot(epsvar(l:i*length(epsvec)),releff(l:i*length(epsvec)),
    [' -o' colvec(i)], 'MarkerFaceColor', colvec(i))
    if i==1
        hold on
    end
    l=l+length(epsvec);
end
xlabel('Contamination (%)');
ylabel('Relative Efficiency');
title(['p = ' pvec(k)]);
legend(['n=25', 'n=50', 'n=100']);
hold off
end
A.2.3 ATLA relative to FSM (Table 2.9)

The following MATLAB function calculates the relative efficiency of the ATLA against the FSM function through data generated through the mean slippage model. It requires the FSM.m function which is available in the FSDA toolbox by Riani et al. (2012) as well as the ATLAfuncNew.m function provided in Appendix A.4.1.

```matlab
ATLAvsFSM.m

clear
pvec=[5, 10];
N=1000;
nvec=[50 100 200];
kvec=[0 1 2 5 10];
jmult=5; sig=3;

fsmcovdet = zeros(length(pvec)*length(nvec)*length(kvec),1);
atlacovdet = zeros(length(pvec)*length(nvec)*length(kvec),1);
l=1;

for o=1:length(kvec)
    k=kvec(o);
    for i=1:length(pvec)
        p=pvec(i);
        mu = ones(p,1);
        sigma = eye(p);
        for j=1:length(nvec)
            n=nvec(j);
            fsmmu = zeros(N,p);
            atlamu = zeros(N,p);
            for m=1:N
                Y1 = mvnrnd(mu*0,sigma,n-k);
                if k > 0
                    Y1out = mvnrnd(mu*jmult,sigma,k);
                    Y1 = [Y1; Y1out];
                end
                Y2 = Y1;
                fsmout = getfield(FSM(Y1,'plots',0,'msg',0),'outliers');
                atlaout = ATLAfuncNew(Y2);
                if isnan(fsmout(1)) == 0
```
Y1(fsmout,:) = [];
end
if isnan(atlaout(1)) == 0
    Y2(atlaout,:) = [];
end
fsmmu(m,:) = mean(Y1)-(k/n)*jmult*mu';
atlamu(m,:) = mean(Y2)-(k/n)*jmult*mu';
end
fsmcovdet(l) = det(cov(fsmmu));
atlacovdet(l) = det(cov(atlamu));
l=l+1;
end
end
end
k0detcovatla = atlacovdet(1:6,:);
k0detcovfsm = fsmcovdet(1:6,:);
releffmat1 = fsmcovdet./atlacovdet;
releffmat2 = atlacovdet./fsmcovdet;
A.2.4 *MATLAB* Computing Times (Figures 2.7)

The following *MATLAB* program calculates the median computing times of the ATLA and FSM functions over multiple runs. It requires the *FSM.m* function which is available in the FSDA toolbox by Riani et al. (2012) as well as the *ATLAFuncNew.m* function provided in Appendix A.4.1.

```
MATLABBenchmark.m

clear

nvec = [50 100 200 500];
pvec = [5 10 15];

% Initialize Vectors 
q=0;
for i=1:length(nvec)
    n = nvec(i);
    for j=1:length(pvec)
        p = pvec(j);
        X = mvnrnd(zeros(p,1),eye(p),n);
        fsmf = @(X) FSM(X,'plots',0,'msg',0)
        t2(j,i) = timeit(fsmf,1);
        if n==500&&p==15
            break
        end
        atlaf = @(X) ATLAfuncNew(X);
        t1(j,i) = timeit(atlaf,1);
    end
end

format bank

sumcol=summer(4);
autcol=autumn(4);

figure
```
hold on
b1=bar(pvec,t1,'BarWidth', 1,'FaceColor','flat')
for k=1:4
    b1(k).FaceColor = sumcol(5-k,:);
end
set(gca,'YScale','log')
hold off
legend({'n=50','n=100','n=200','n=500'},'Location','northwest')
title('ATLA')
xlabel('Variables (p)')
ylabel('Median Execution Time (in sec)')
xticks([5 10 15])

figure
hold on
b2=bar(pvec,t2,'BarWidth', 1,'FaceColor','flat')
for k=1:4
    b2(k).FaceColor = autcol(5-k,:);
end
set(gca,'YScale')
hold off
legend({'n=50','n=100','n=200','n=500'},'Location','northwest')
title('FSM')
xlabel('Variables (p)')
ylabel('Median Execution Time (in sec)')
xticks([5 10 15])

92
A.3 Supplementary R Functions

The following programs were built/modified under version 3.5.2 of the R package (R Core Team; 2014).

A.3.1 ATLA Function Modified for Planted Outliers

The following R function is the original ATLA R function \(ATLA\_\text{for}\_\text{Multivariate}\_\text{Data.R}\) by Daniel Schubert (2005) and later modified by Robert Hammarstrand (see supplementary materials in Clarke (2018)) only it has been modified to allow the translation of an additional indicator variable for whether or not the observation was a planted outlier. This function requires the packages \texttt{robustbase}, \texttt{survival}, \texttt{stats}, \texttt{graphics}, \texttt{Hmisc} and \texttt{splines} in order to run.

\begin{verbatim}
ATLAfunc <- function(y)
{
  dataset <- y
  filename <- dataset[,-(p+1)]
  numberrow <- nrow(filename)
  numbercol <- ncol(filename)
  datasize <- numberrow
  dim <- numbercol
  b <- floor((datasize+dim+1)/2)
  h <- b
  trim <- (datasize-b)
  bsample <- floor((datasize+dim)/2)
  jsample <- dim+1
  nchoosek <- function(n,k) exp(lgamma(n+1)-(lgamma(k+1)+lgamma(n-k+1))))
  samples <- log(0.05)/log(1-nchoosek(bsample,jsample)/
    nchoosek(datasize,jsample))
  samples <- ceiling(samples)
  sample <- filename
  kapparho <- rep(0,(datasize+1-b))
}
\end{verbatim}
minimumlocal <- rep(0, (trim + 1))

tick <- 0
for(i in b:(datasize - 1))
{
    tick <- (tick + 1)
    e <- i / datasize

    y <- sqrt(qchisq(e, dim))
    k <- dim
    integrand <- function(x) {x^(k + 1)*(-1/2)*1/((2*pi)^(k/2))*
        exp(-1/2*x^2)}
    intanswer <- integrate(integrand, lower = 0, upper = y)
    intvalue <- intanswer$value
    kappa <- 1/((4*pi^(k/2))/(k*gamma(k/2))*intvalue)^2
    kappanew <- kappa

    kapparho[tick] <- (kappanew)^dim
}

kapparho[tick + 1] <- 1

ksize <- length(kapparho)

alp <- alimit / datasize

set.seed(1)
minimumcovariancedeterminant1 <- covMcd(filename, cor = FALSE,
alpha = alp, nsamp = samples)

centre1 <- minimumcovariancedeterminant1$center

bestmcd1 <- minimumcovariancedeterminant1$best
lengthbestmcd1 <- length(bestmcd1)

S1 <- minimumcovariancedeterminant1$cov

temp <- list()

newdf <- filename[bestmcd1,]
temp[[1]] <- newdf

newobject <- rep(1, ksize)

for (i in 1:ksize)
{
  centre1 <- colMeans(newdf)

  if (datasize < 31)
  {
    newcov <- cov(newdf)
  }
  else
  {
    newcov <- (i + (datasize - ksize))/datasize*cov(newdf)
  }
  newdet <- det(newcov)

  newobject[i] <- kapparho[i]*newdet

  if (i < ksize)
  {
    dsquared <- mahalanobis(filename, centre1, cov(newdf),
                            inverted=FALSE, tol.inv = 1e-70)
    names(dsquared) <- 1:length(dsquared)
    sortdsquared <- sort(dsquared)
    deletedsquared <- sortdsquared[-(1:length(bestmcd1))]
    keptdsquared <- sortdsquared[1:length(bestmcd1)]

    kept <- t(as.numeric(names(keptdsquared)))
  }
newdf <- filename[,]

bigdsquared <- deletedsquared
inflate<-matrix(c(as.numeric(names(bigdsquared)),bigdsquared), ncol=2)

extractinflate <- inflate[inflate[,2]==min(inflate[,2]),]

extinf<- extractinflate[1]

newdf <- rbind(newdf,filename[extinf,])

centre1<-colMeans(newdf)
bestmcd1 <- c(bestmcd1,extinf)

dsquared <- mahalanobis(filename,centre1,cov(newdf),inverted=FALSE, tol.inv = 1e-70)

names(dsquared)<-1:length(dsquared)
sortdsquared <- sort(dsquared)

keptdsquared <- sortdsquared[1:length(bestmcd1)]

kept <- t(as.numeric(names(keptdsquared)))

newdf <- filename[kept,]

temp[[i+1]]<-newdf

}

countpicked <- 0
picked <- 0
i<-(ksize+1)
while(i > 2)
{
  i <- i-1
  if(newobject[i] > newobject[i-1])
  {
    
  }


countpicked <- countpicked + 1

if(countpicked == 1)
{
    picked <- (i-1)
}

if(countpicked > 1)
{
    if(newobject[picked] > newobject[i-1]) {picked <- (i-1)}
}

if(picked == 0){picked <- ksize}
newdf<- data.frame(temp[[picked]])
finalsamplerownames <- c(as.numeric(row.names(newdf)))
outliers <- dataset[-finalsamplerownames,]
outliers
A.3.2 ATLA Function Modified for Location Output

The following R function is the original ATLA R function (ATLA_for_Multivariate_Data.R) by Daniel Schubert (2005) and later modified by Robert Hammarstrand (see supplementary materials in Clarke (2018)) only it has been extended to output the location estimate after the flagged outliers have been trimmed. This function requires the packages robustbase, survival, stats, graphics, Hmisc and splines in order to run.

`ATLAmulti.R`

```r
ATLAmulti <- function(y) {
  dataset <- y
  filename<- y
  numberrow <- nrow(filename)
  numbercol <- ncol(filename)
  datasize <- numberrow
  dim <- numbercol
  b <- floor((datasize+dim+1)/2)
  h <- b
  trim <- (datasize-b)
  bsample <- floor((datasize+dim)/2)
  jsample <- dim+1
  nchoosek <- function(n,k) exp(lgamma(n+1)-(lgamma(k+1)+lgamma(n-k+1)))
  samples <- log(0.05)/log(1-nchoosek(bsample,jsample)/nchoosek(datasize,jsample))
  samples <- ceiling(samples)
  sample <- filename
  kapparho <- rep(0,(datasize+1-b))
  minimumlocal <- rep(0,(trim+1))
  tick <- 0
  for(i in b:(datasize-1)) {
  ...}
}
```

98
tick <- (tick+1)

e <- i/datasize

y <- sqrt(qchisq(e,dim))
k <- dim

integrand <- function(x) {x^(k+1)*(-1/2)*1/((2*pi)^(k/2))*exp(-1/2*x^2)}

intanswer <- integrate(integrand,lower=0,upper=y)

intvalue <- intanswer$value

kappa <- 1/((4*pi^(k/2)/(k*gamma(k/2))*intvalue)^2)

kappanew <- kappa

kapparho[tick] <- (kappanew)^dim

kapparho[tick+1] <- 1

ksize <- length(kapparho)

alp <- alimit/datasize

set.seed(1)

minimumcovariancedeterminant1 <- covMcd(filename, cor=FALSE, alpha=alp, nsamp=samples)

centre1 <- minimumcovariancedeterminant1$center

bestmcd1 <- minimumcovariancedeterminant1$best

lengthbestmcd1<- length(bestmcd1)

S1 <- minimumcovariancedeterminant1$cov

temp<-list()
newdf <- filename[bestmcd1,]
temp[[1]]<-newdf

newobject<-rep(1, ksize)
for(i in 1:ksize)
{
    centre1<-colMeans(newdf)

    if(datasize<31)
    {
        newcov <- cov(newdf)
    }else
    {
        newcov <- (i+(datasize-ksize))/datasize*cov(newdf)
    }
    newdet <- det(newcov)

    newobject[i] <- kapparho[i]*newdet

    if(i<ksize)
    {
        dsquared <- mahalanobis(filename,centre1,cov(newdf),inverted=FALSE, tol.inv = 1e-70)
        names(dsquared)<-1:length(dsquared)
        sortdsquared <- sort(dsquared)
        deletedsquared <- sortdsquared[-(1:length(bestmcd1))]
        keptdsquared <- sortdsquared[1:length(bestmcd1)]
        kept <- t(as.numeric(names(keptdsquared)))
        newdf <- filename[kept,]
        bigdsquared <- deletedsquared
        inflate<-matrix(c(as.numeric(names(bigdsquared)),bigdsquared), ncol=2)
extractinflate <- inflate[inflate[,2]==min(inflate[,2])]

extinf<- extractinflate[1]

newdf <- rbind(newdf, filename[extinf,])

centre1<-colMeans(newdf)
bestmcd1 <- c(bestmcd1, extinf)

d squared <- mahalanobis(filename, centre1, cov(newdf), inverted=FALSE, tol.inv = 1e-70)

names(dsquared)<-1:length(dsquared)
sortedsquared <- sort(dsquared)

keptsquared <- sortedsquared[1:length(bestmcd1)]

kept <- t(as.numeric(names(keptsquared)))

newdf <- filename[kept,]

temp[[i+1]]<-newdf

}
}

countpicked <- 0
picked <- 0
i<- (ksize+1)
while (i > 2) {
  i <- i-1
  if (newobject[i] > newobject[i-1])
    {
      countpicked <- countpicked+1

      if (countpicked == 1)
        {
          picked <- (i-1)
        }
if(countpicked > 1) {
    if(newobject[picked] > newobject[i-1]) {picked <- (i-1)}
}

}

if(picked == 0) {picked <- ksize}
newdf<- data.frame(temp[[picked]])
meantrimmed<- apply(newdf,2,mean) 
meantrimmed
}
A.3.3 Original Forward Search by Hadi (1994) adapted to R

The following R function is the original Forward Search procedure developed by Hadi (1994) that has been adapted to R. It requires the input of the dataset and significance level and outputs the flagged outliers.

```
Hadi1994funcIndout <- function(y,z)
{
    origdataset<-as.data.frame(y)
    dataset1<-origdataset[,,-c(p+1)]
    alpha<-z
    p=ncol(dataset1)
    n=nrow(dataset1)
    h=floor((n+p+1)/2)
    r<-p
    tol<-max(10^--(p+5), 10^-12) # Tolerance for Singular matrix
    cnp=(1+2/(n-1-3*p)+(p+1)/(n-p))^2

    ### Step 0: Initial Ordering ###
    ### Compute Cm and Sm ###
    cm<-apply(dataset1,2,median)
    dumat<-as.matrix(dataset1-matrix(cm,nrow=n,ncol=p,byrow = T))
    summat=matrix(0,nrow = p,ncol=p)
    for (i in 1:n) {
        summat2=dumat[i,]%*%t(dumat[i,])
        summat=summat+summat2
    }
    sm<-(1/(n-1))*summat
    mahdist<-mahalanobis(dataset1,cm,sm)
    order1<-sort.list(mahdist)

    ### Compute Cv and Sv ####
    repeat {
        basic<-dataset1[order1[1:h],]
        cv<-apply(basic,2,mean)
        sv=cov(basic)
        if (det(sv) > tol) {
            mahdist<-mahalanobis(dataset1,cv,sv)
            order1<-sort.list(mahdist)
        }
    }
}
```
### Step 1 ###
repeat {
  r<-r+1
  if (h<r) {
    break
  }
  basic<-dataset1[order1[1:r],]
  sb<-cov(basic)
  cb<-apply(basic,2,mean)
  if (det(sb) > tol) {
    mahdist<-mahalanobis(dataset1,cb,sb)
    order1<-sort.list(mahdist)
  }
}

### Step 3 ###
repeat {
  basic<-dataset1[order1[1:h],]
  sb<-cov(basic)
  cb<-apply(basic,2,mean)
  if (det(sb) > tol) {
    mahdist<-mahalanobis(dataset1,cb,sb)
    order1<-sort.list(mahdist)
    if (mahdist[order1[h+1]]>=(cnp*qchisq(alpha/n,df=p,lower.tail = F))){
      flaggedout<-order1[c(h+1):n]
      break
    } else {
      h<-h+1
      if (n<=h) {
        return("No outliers")
      }
    }
  }
}
} 
} 
else {
    h<-h+1
    if (n<=h) {
        return("No outliers")
    } 
} 

origdataset[flaggedout,]
A.4 Supplementary MATLAB Functions

The following programs were built/modified under MATLAB version R2018b (MATLAB; 2018).

A.4.1 ATLA Function

The following MATLAB function is the original ATLA MATLAB routine (ATLAmultivariate.m) by Daniel Schubert (2005) and later modified by Robert Hammarstrand only it has been adapted to a function. Also note the function has been modified to include the T1 objective function (for $n < 30$), a feature which was absent in the original code.

$ATLAsfuncNew.m$

function [ATLAout]=ATLAsfuncNew(Y)

format long

filename = Y;
syms x
% obtaining sample size and dimension
[nrow,ncol]=size(filename);
datasize=nrow;
dim=ncol;

b=floor((datasize+dim+1)/2);
h=b;
trim=datasize-b;

% establishing number of samples of size=dimension+1 required to ensure
% 95% chance of non-contaminated starting sample for MCD algorithm

bsample=floor((datasize+dim)/2);
jsample=dim+1;
samples=log(0.05)/(log(1-(nchoosek(bsample,jsample)/nchoosek(datasize,
jsample))));
samples=ceil(samples);

sample =filename;
%initializing minimum
for i=1:(trim+1)
    minimumlocal(i)=0;
end;

%calculating each 1/zeta^2 (Butler et al 1993) see
%An Adaptive Trimmed Likelihood Algorithm for Identification
%of Multivariate Outliers

tick=0;
for i=b:(datasize-1)
    tick=tick+1;
    e=i/datasize;
    k=dim;
    y=sqrt(chi2inv(e,dim));
    kappa=1/((4*pi^(k/2))/(k*gamma(k/2))*int(x^(k+1)*(-1/2)*1/((2*pi)^(k/2))*exp(-1/2*x^2),0,y)^2);
    kappanew=eval(kappa);
    kapparho(tick)=(kappanew)^dim;
end;
kapparho(tick+1)=1;

while p<samples
    j=0;
    %finding initial sample of size (dim+1) as preliminary to C-steps in
    %MCD estimate see Woodruff and Rocke 1993 for details
    while j<1
        j=j+1;
        samplechange = sample;
        %...
clear samplechoice

for i=1:(dim+1)
    r=ceil((datasize-i+1)*rand);
    samplechoice(i,:)=samplechange(r,:);
    samplechange(r,:)=[];
end;

cdet=det(cov(samplechoice));

if cdet==0
    j=0;
end;
end;

%C-steps until H_3

k=0;
while k<3
    d=mahal(sample,samplechoice);
    sortd=sort(d);
    for i=1:h
        for j=1:datasize
            if d(j,:)==sortd(i,:)
                samplechoice(i,:)=sample(j,:);
            end;
        end;
    end;
    k=k+1;
end;
p=p+1;

newmu(p,:)= mean(samplechoice);
mcdobject(p)=det(cov(samplechoice));
selectmatrix=['M',int2str(p),'=samplechoice'];
evalc(selectmatrix);
evalc(['M',int2str(p)]);

d=selecting 10 minimum determinants from p samplechoices
mindet=sort(mcdobject);

for i=1:10
    for j=1:samples
        if mcdobject(j)==mindet(i)
            select(i)=j;
        end;
    end;
end;

%C-steps for each of the above 10 chosen samples above until
% convergence
q=0;
while q<10
    q=q+1;
    mu=newmu(select(q),:);
    samplechoice=eval(['M',int2str(select(q))]);
    cmatrix=cov(samplechoice);
    k=0;
    while k<1
        dmat=det(cmatrix);
        d=mahal(sample,samplechoice);
sortd=sort(d);

for i=1:h
    for j=1:datasize
        if d(j,:)==sortd(i,:)
            samplechoice(i,:)=sample(j,:);
        end;
    end;
end;

mu=mean(samplechoice);
cmatrix=cov(samplechoice);
dnew=det(cmatrix);

if dnew==dmat
    k=2;
end;
end;

lastmu(q,:)=mu;
dchoice(q)=dnew;
selectmatrix=['M',int2str(q),',='samplechoice'];
evalc(selectmatrix);
evalc(['M',int2str(q)]);
end;

% determining minimum determinant of 10 converged samples
k=0;
mindet=min(dchoice);

while k<10
    k=k+1;
    if dchoice(k)==mindet
        mind=k;
        k=10;
    end;
end;
mcdmu=lastmu(mind,:);
samplechoice=eval(['M',int2str((mind))]);
cmatrix=cov(samplechoice);

% using mcd estimate obtained above begin trimming see Clarke
% & Schubert (2006)
mu=mcdmu;

rsqr=mahal(sample,samplechoice);

ordering whole sample here
for i=1:datasize

    for j=1:datasize
        if sorting2(i)==rsqr(j)
            orderedsample(i,:)=sample(j,:);
        end;
    end;
end;

unchanged=orderedsample;

j=floor((datasize+dim+1)/2);
count=0;found=1;
for i=(j):datasize
    count=count+1;

    newchanged=unchanged(1:i,:);
    newrsqr=mahal(unchanged,newchanged);

end;
newsort=sort(newrsqr);

% ordering whole sample here
for i2=1:datasize
    for j2=1:datasize
        if newsort(i2)==newrsqr(j2)
            orderedunchanged(i2,:)=unchanged(j2,:);
        end;
    end;
end;

unchanged=orderedunchanged;

haschanged=orderedunchanged(1:i,:);

leftoverchanged=orderedunchanged((i+1):datasize,:);

% Original
% sigma=(i/datasize)*cov(haschanged);

% Added
if datasize < 30
    sigma=cov(haschanged);
else
    sigma=(i/datasize)*cov(haschanged);
end;
%

mcd(count)=det(sigma);

selectsecondmatrix=['M',int2str(count),'=haschanged'];
evalc(selectsecondmatrix);
evalc(['M',int2str(count)]);

selectsecondmatrix=['N',int2str(count),'=leftoverchanged'];
evalc(selectsecondmatrix);
evalc(['N',int2str(count)]);

end;%for i=size:-1:j

%calculating objective function for each subset ie Sigma*1/zeta^2

for i=1:(trim+1)
    newobject(i)=mcd(i)*kapparho(i);
end;
w=(newobject);

%local minima
i=1;
while i < trim
    i=i+1;
    if w(i-1) > w(i)
        if w(i+1) > w(i)
            minimumlocal(i)=1;
        end;
    end;
end;

%checking if there were any minima for \alpha>0
for i = 1: length(w)
    check(i)=w(i)*minimumlocal(i);
end;
summingminima= sum(minimumlocal);

%if there was at least one minima for \alpha>0
if summingminima > 0

[minimum,trimming]=min(-check);

% matrix of outliers by value
trimmatrix=eval(['N',int2str(trimming)]);

sample = filename;

% locating outlier by observation number
for i=1:datasize
    for j=1:(length(w)-trimming)
        if sample(i,:)== trimmatrix(j,:)
            outliers(j)=i;
        end;
    end;
end;

ATLAout=outliers;

else
    ATLAout=NaN;
end;
Appendix B

Supplementary Simulation Results

B.1 Supplementary Tables

B.1.1 ATLA relative to Hadi’s FS (Table 2.4)

e) Simulation results for k=1 and p=5

<table>
<thead>
<tr>
<th></th>
<th>ATLA</th>
<th>Hadi</th>
<th>ATLA</th>
<th>Hadi</th>
<th>ATLA</th>
<th>Hadi</th>
</tr>
</thead>
<tbody>
<tr>
<td>n=25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p1</td>
<td>0.983</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p2</td>
<td>0.983</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p3</td>
<td>0.981</td>
<td>0.958</td>
<td>0.996</td>
<td>0.971</td>
<td>0.998</td>
<td>0.965</td>
</tr>
<tr>
<td>p4</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>p5</td>
<td>0.000</td>
<td>0.008</td>
<td>0.000</td>
<td>0.002</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

f) Simulation results for k=2 and p=5

<table>
<thead>
<tr>
<th></th>
<th>ATLA</th>
<th>Hadi</th>
<th>ATLA</th>
<th>Hadi</th>
<th>ATLA</th>
<th>Hadi</th>
</tr>
</thead>
<tbody>
<tr>
<td>n=25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p1</td>
<td>0.997</td>
<td>0.976</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p2</td>
<td>0.997</td>
<td>0.976</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p3</td>
<td>0.957</td>
<td>0.869</td>
<td>0.944</td>
<td>0.965</td>
<td>0.963</td>
<td>0.949</td>
</tr>
<tr>
<td>p4</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>p5</td>
<td>0.002</td>
<td>0.011</td>
<td>0.001</td>
<td>0.001</td>
<td>0.000</td>
<td>0.001</td>
</tr>
</tbody>
</table>
### B.1.2 ATLA relative to BACON (Table 2.6)

#### e) Simulation results for k=1 and p=5

<table>
<thead>
<tr>
<th></th>
<th>ATLA</th>
<th>BACON</th>
<th>ATLA</th>
<th>BACON</th>
<th>ATLA</th>
<th>BACON</th>
</tr>
</thead>
<tbody>
<tr>
<td>n=25</td>
<td>0.812</td>
<td>0.983</td>
<td>0.986</td>
<td>0.999</td>
<td>0.994</td>
<td>1.000</td>
</tr>
<tr>
<td>p1</td>
<td>0.812</td>
<td>0.983</td>
<td>0.986</td>
<td>0.999</td>
<td>0.994</td>
<td>1.000</td>
</tr>
<tr>
<td>p2</td>
<td>0.805</td>
<td>0.967</td>
<td>0.982</td>
<td>0.974</td>
<td>0.992</td>
<td>0.967</td>
</tr>
<tr>
<td>p3</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>p4</td>
<td>0.002</td>
<td>0.001</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

#### f) Simulation results for k=5 and p=5

<table>
<thead>
<tr>
<th></th>
<th>ATLA</th>
<th>BACON</th>
<th>ATLA</th>
<th>BACON</th>
<th>ATLA</th>
<th>BACON</th>
</tr>
</thead>
<tbody>
<tr>
<td>n=25</td>
<td>0.923</td>
<td>0.029</td>
<td>1.000</td>
<td>0.999</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>p1</td>
<td>0.923</td>
<td>0.018</td>
<td>1.000</td>
<td>0.999</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>p2</td>
<td>0.873</td>
<td>0.004</td>
<td>0.944</td>
<td>0.965</td>
<td>0.962</td>
<td>0.950</td>
</tr>
<tr>
<td>p3</td>
<td>0.000</td>
<td>0.002</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>p4</td>
<td>0.003</td>
<td>0.000</td>
<td>0.001</td>
<td>0.001</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

### B.1.3 ATLA relative to BACON for Large n (Table 2.8)

#### e) Simulation results for k=20 and p=5

<table>
<thead>
<tr>
<th></th>
<th>ATLA</th>
<th>BACON</th>
<th>ATLA</th>
<th>BACON</th>
<th>ATLA</th>
<th>BACON</th>
</tr>
</thead>
<tbody>
<tr>
<td>n=200</td>
<td>1.000</td>
<td>0.992</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>p1</td>
<td>1.000</td>
<td>0.992</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>p2</td>
<td>0.874</td>
<td>0.947</td>
<td>0.936</td>
<td>0.955</td>
<td>0.958</td>
<td>0.939</td>
</tr>
<tr>
<td>p3</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>p4</td>
<td>0.001</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

#### f) Simulation results for k=50 and p=5

<table>
<thead>
<tr>
<th></th>
<th>ATLA</th>
<th>BACON</th>
<th>ATLA</th>
<th>BACON</th>
<th>ATLA</th>
<th>BACON</th>
</tr>
</thead>
<tbody>
<tr>
<td>n=200</td>
<td>1.000</td>
<td>0.821</td>
<td>1.000</td>
<td>0.991</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>p1</td>
<td>1.000</td>
<td>0.818</td>
<td>1.000</td>
<td>0.991</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>p2</td>
<td>0.673</td>
<td>0.770</td>
<td>0.742</td>
<td>0.934</td>
<td>0.816</td>
<td>0.956</td>
</tr>
<tr>
<td>p3</td>
<td>0.000</td>
<td>0.001</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>p4</td>
<td>0.002</td>
<td>0.000</td>
<td>0.001</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>
## B.1.4 R Computing Times (Figures 2.6)

Table B.1: The computing times (in seconds) of $N = 1000$ replications of the R ATLA function performed on randomly generated samples from $\mathcal{N}_p(0, I_p)$. (For a detailed explanation of the different time measurements refer to the help section of the `proc.time` function within R.)

<table>
<thead>
<tr>
<th>$p$</th>
<th>$n$</th>
<th>CPU Time (sec)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>User</td>
<td>System</td>
<td>Elapsed</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>50</td>
<td>27.32</td>
<td>0.82</td>
<td>28.97</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>44.66</td>
<td>0.98</td>
<td>45.64</td>
<td></td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>105.88</td>
<td>1.53</td>
<td>107.47</td>
<td></td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>400.44</td>
<td>3.75</td>
<td>404.29</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>1302.91</td>
<td>8.29</td>
<td>1312.2</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>50</td>
<td>130.79</td>
<td>5.25</td>
<td>137.36</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>335.75</td>
<td>9.80</td>
<td>346.08</td>
<td></td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>656.94</td>
<td>11.75</td>
<td>668.89</td>
<td></td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>2091.25</td>
<td>16.05</td>
<td>2108.17</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>3197.08</td>
<td>24.66</td>
<td>3257.95</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>50</td>
<td>2344.08</td>
<td>57.25</td>
<td>2406.33</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>9657.45</td>
<td>146.62</td>
<td>9811.76</td>
<td></td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>29545.95</td>
<td>258.99</td>
<td>29823.28</td>
<td></td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>105383.21</td>
<td>418.42</td>
<td>105963.62</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>80149.39</td>
<td>450.72</td>
<td>80761.63</td>
<td></td>
</tr>
</tbody>
</table>
Table B.2: The computing times (in seconds) of \( N = 1000 \) replications of the R BACON function (with \( c = 4 \) and \( \alpha = 0.05 \)) performed on randomly generated samples from \( \mathcal{N}_p(0J, I_p) \). (For a detailed explanation of the different time measurements refer to the help section of the `proc.time` function within R.)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>CPU Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>User</td>
<td>System</td>
</tr>
<tr>
<td>( p=5 )</td>
<td>( n=50 )</td>
<td>1.11</td>
</tr>
<tr>
<td></td>
<td>( n=100 )</td>
<td>1.14</td>
</tr>
<tr>
<td></td>
<td>( n=200 )</td>
<td>1.31</td>
</tr>
<tr>
<td></td>
<td>( n=500 )</td>
<td>1.99</td>
</tr>
<tr>
<td></td>
<td>( n=1000 )</td>
<td>3.20</td>
</tr>
<tr>
<td>( p=10 )</td>
<td>( n=50 )</td>
<td>1.03</td>
</tr>
<tr>
<td></td>
<td>( n=100 )</td>
<td>1.32</td>
</tr>
<tr>
<td></td>
<td>( n=200 )</td>
<td>1.86</td>
</tr>
<tr>
<td></td>
<td>( n=500 )</td>
<td>2.88</td>
</tr>
<tr>
<td></td>
<td>( n=1000 )</td>
<td>4.80</td>
</tr>
<tr>
<td>( p=15 )</td>
<td>( n=50 )</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>( n=100 )</td>
<td>1.61</td>
</tr>
<tr>
<td></td>
<td>( n=200 )</td>
<td>2.22</td>
</tr>
<tr>
<td></td>
<td>( n=500 )</td>
<td>3.75</td>
</tr>
<tr>
<td></td>
<td>( n=1000 )</td>
<td>6.69</td>
</tr>
</tbody>
</table>
MATLAB Computing Times (Figures 2.7)

Table B.3: The median computing time (in seconds) of the MATLAB ATLA and FSM functions performed on randomly generated samples from $\mathcal{N}_p(0, I_p)$. Note that the category $n = 500$ and $p = 15$ has been omitted for the ATLA due to excessive computing times.

<table>
<thead>
<tr>
<th></th>
<th>ATLA</th>
<th>FSM</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p=5$</td>
<td>n=50</td>
<td>0.4288</td>
</tr>
<tr>
<td></td>
<td>n=100</td>
<td>0.7494</td>
</tr>
<tr>
<td></td>
<td>n=200</td>
<td>1.4310</td>
</tr>
<tr>
<td></td>
<td>n=500</td>
<td>4.4915</td>
</tr>
<tr>
<td>$p=10$</td>
<td>n=50</td>
<td>4.1271</td>
</tr>
<tr>
<td></td>
<td>n=100</td>
<td>9.0658</td>
</tr>
<tr>
<td></td>
<td>n=200</td>
<td>17.8902</td>
</tr>
<tr>
<td></td>
<td>n=500</td>
<td>48.6238</td>
</tr>
<tr>
<td>$p=15$</td>
<td>n=50</td>
<td>60.3811</td>
</tr>
<tr>
<td></td>
<td>n=100</td>
<td>266.9633</td>
</tr>
<tr>
<td></td>
<td>n=200</td>
<td>2586.8143</td>
</tr>
<tr>
<td></td>
<td>n=500</td>
<td>n/a</td>
</tr>
</tbody>
</table>
Bibliography


**URL:** [https://cran.r-project.org/web/packages/rbenchmark/index.html](https://cran.r-project.org/web/packages/rbenchmark/index.html)


**URL:** [http://www.R-project.org/](http://www.R-project.org/)


URL: https://doi.org/10.1016%2Fj.chemolab.2012.03.017


URL: https://cran.r-project.org/web/packages/robustX/index.html


URL: http://www.stats.ox.ac.uk/pub/MASS4