Independent Component Analysis (ICA) for Blind Source Separation

---

A Comprehensive Evaluation Report
Presented to
The Statistics Faculty
Amherst College

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Arts
in
Statistics

Sara J. Culhane

September 17, 2017

Acknowledgements

I would like to thank my advisor Professor Amy Wagaman for her assistance with concepts and ideas throughout my Comprehensive Project process as well as her dedication to excellent teaching of Statistics. I would also like to thank Professor Horton for his commitment to the growth Amherst Statistics program and his passion for data science. Many thanks extended to Professor Susan Wang, Professor Jordan Crouser at Smith and Professor Albert Kim for their invaluable courses on Statistics and Machine Learning that impacted by studies greatly.

Table of Contents

List of Tables

List of Figures

Abstract

Most of statistics as well as aspects of machine learning rely on linear models or Gaussian distributions in order to successfully model and separate signal from noise. While our focus throughout our coursework at Amherst has mostly been involved analysis based on known or applied variables and distributions, Independent Component Analysis (ICA) and other methods of Blind Source Separation (BSS) call for the decomposition of their components with much more limited knowledge of their distribution than we have seen previously. By identifying independent, non-Gaussian components of a mixture with ICA, the desired/original source can be uncovered and hidden information about the data can be revealed. R packages like fastICA,JADE and ICA Infomax make its application to numerous fields more efficient and usable.

Introduction

0.1 Overview : What is Blind Source Separation (BSS)?

Blind Source Separation is a set of various methods in multivariate data analysis used to recompose unknown original sources from known data. These methods utilize information from the known observed sources with consideration of a set of unknown weights, which are then applied to the known observations to recover the original sources desired.

In other words, there are always two unknowns and one known (each source is its own vector within the matrix).

On a fundamental level, BSS attempts to solve a system of linear equations in which:

$$x = A \cdot s$$

Where the solution to this system will be in the form:

$$s = A^{-1} \cdot x$$

However, these s_i are not known values but instead a vector of distinct source signals, defined by their relationship to an also unknown operator matrix A. The x_i are values that can be directly observed from the signal/data without decomposition.

In general the basic components are:

• N unknown sources s_j (eg. the original sources to recover)

• One unknown operator **A** (an "un-mixing" or "de-mixing" matrix)

• P observed signals x_i with some relation:

$$x = A(s)$$

With this, the chosen BSS method takes the P known observations, performs a separation technique (also known as un-mixing or de-mixing) and outputs the unknown N sources that have been obscured by some unobserved process (i.e. the data collection process, random noise or limits of a set of data in most cases).

These y_k outputs should effectively estimate the original sources that make up the s vector(s).

BSS possesses a large range of applications but can effectively uncover factors in data that are previously unknown or underlying the known data through this demixing of source signals.

(Puigt, 2011)

0.2 ICA Generally

0.2.1 Definition: Independent Component Analysis (ICA)

Independent Component Analysis (ICA) provides a specific, efficient methodology to the general BSS framework described above and has been proven a useful application in various settings with complex data. This process can be referred to a generative model, as ICA determines how the observed data are constructed through the mixing process. (pg. 151, Aapo Hyvärinen & Oja, 2001)

While ICA follows the same basic matrix model as described above, it can also often also be written as:

$$x = \sum_{i=1}^{n} a_i s_i$$

0.2.2 Assumptions of ICA Model

The two key characteristics and model requirements differentiate ICA from standard BSS and other methods are the following:

1. Assumption of statistical independence amongst the unknown source components
By definition, no component provides information on any of the other components.
(p 3, Aapo Hyvärinen & Oja, 2001)

2. Assumption of Non-Gaussianity:

- These independent components are also assumed to have strictly Non-Gaussian distributions (or at most one independent component with a Gaussian distribution)
- Distinguishes from the similar structure method of Factor Analysis, which instead requires Gaussianity.

(p. 2, Hyvärinen & Oja, 2000)

Note-

• The unknown mixing matrix will be denoted as a square for ease of calculation (not always the case)

In this scenario, the number of IC will equal the number of observed mixtures (pg. 153, Aapo Hyvärinen & Oja, 2001)

0.2.3 Ambigities of ICA Model

Because of the two unknown factors (source signals and mixing matrix) :

• No reliable method of calculating variance of ICs exists.
Solution : Set the RV to have unit variance or $E(s_i^2) = 1$

• No method of determining the order of ICs
(pg. 154, Aapo Hyvärinen & Oja, 2001)

0.3 Primary Features of ICA (from Assumptions)

0.3.1 Independence

Since independence is a fundamental condition of ICA for the construction of ICs , it is important to define it conceptually for clarity.

Looking at two random variables (RV) denoted y_1 and y_2 in general, they will be independent if and only if their joint probability density function (pdf) can be rewritten as:

$$p(y_1, y_2) = p_1(y_1)p_2(y_2)$$

This can be extended to the expectation of two functions applied to these RV's and is denoted:

$$E(h_1(y_1)h_2(y_2)) = E(h_1(y_1))E(h_2(y_2))$$

(p. 3-4, Hyvärinen & Oja, 2000)

Note on Correlation and Independence

By definition, two RV's are uncorrelated if and only if:

$$E(y_1y_2) - E(y_1)E(y_2) = 0$$

While uncorrelatedness does not necessarily imply independence, independent RV's will always be uncorrelated. For ICA modeling purposes, this property helps simplify the estimation to only uncorrelated values.

(p. 5, Hyvärinen & Oja, 2000)

0.3.2 Non-Gaussianity and Measurement

Though the Central Limit Theorem asserts that the sum of two or more independent RV's will tend towards a Gaussian distribution, the process of un-mixing actually produces non-Gaussian signals.

(Puigt, 2011)

ICA requires non-Gaussianity for the distributions of all source signals, as accurate estimation of the mixing matrix A is not even possible without meeting this condition.

To show this, assume that the known mixing matrix A is orthogonal (eg. $A^TA = AA^T = I$) and that all of the s_i are Gaussian.

Then, x_1 and x_2 have joint density:

$$p(x_1, x_2) = \frac{1}{2\pi} exp(\frac{x_1^2 + x_2^2}{2})$$

The symmetry of this joint density function makes it impossible to know the directional signals of the columns of matrix A, and therefore, the matrix cannot be estimated, rendering ICA ineffective in the Gaussian case (although one Gaussian IC is technically allowed).

(p. 5, Hyvärinen & Oja, 2000)

Kurtosis as a Measure of Non-gaussianity

Kurtosis is defined as the 4th order cumulant:

$$kurt(y) = E(y^4) - 3(E(y^2))^2$$

Since y= 1, it can be simplified to:

$$kurt(y) = E(y^4) - 3$$

Which is equivalent to the 4th moment of y

In a Gaussian Distribution, the 4th moment equals $3(E(y^2))^2$
(p. 6, Hyvärinen & Oja, 2000)

Thus,

$$kurt(y) = 3(E(y^2))^2 - 3(E(y^2))^2 = 0$$

All Gaussian distributions have kurtosis equal to 0, and therefore, it is expected that most non-Gaussian distributions will have a nonzero Kurtosis with some rare exceptions. Beyond this, Kurtosis essentially measures the deviation of RV from Gaussianity. (Puigt, 2011)

It's use in ICA is popular due to the ease of estimation as well as its linearity property. This holds two properties, assuming that x_1 and x_2 are independent:

$$kurt(x_1 + x_2) = kurt(x_1) + kurt(x_2)$$
$$kurt(\alpha x_1) = \alpha^4 kurt(x_1)$$

Issues with Kurtosis - Highly sensitive to outliers as just a few extreme values can dramatically impact its calculation. Therefore, its application to skewed data or data with outliers that strong influence may not be accurate.

(p. 7, Hyvärinen & Oja, 2000)

Given this and the preference for another method in R packages to be utilized later, kurtosis will not be discussed much further in this paper, and more robust methods of measuring non-Gaussianity will be explored instead.

Negentropy

Since Kurtosis cannot always successfully determine Non-gaussianity, the use of negentropy as a alternative method of estimation and is typically preferred.

Negentropy derives from the theoretical "differential" entropy or level of randomness of a given variable. In general, Gaussian variables tend to possess the largest entropy given control for an equal variance condition. (8, Hyvärinen & Oja, 2000)

Theoretical entropy is defined for discrete and continuous respectively as:

$$H(Y) = -\sum_{i=1} P(Y = a_i) log P(Y = a_i)$$

$$H(\mathbf{y}) = -\int f(\mathbf{y}) log f(\mathbf{y}) dy$$

To obtain the desired negative differential entropy, the calculation modifies to:

$$J(\mathbf{y}) = H(\mathbf{y}_{gauss}) - H(\mathbf{y})$$

Critically in the above equation, $\mathbf{y}_{gauss}$ is the Gaussian RV of with identical covariance as $\mathbf{y}$.

It is always:

- Non-negative and will be zero if and only if $\mathbf{y}$ is Gaussian.

- Invariant for invertible linear transformations.

In most cases, negentropy tends to be the best performing estimator of non-gaussianity but can be difficult or expensive to compute. However, the ease of computation has improved over time with technology, and thus the fastICA package relies on the estimator in its calculation.

(p. 8, Hyvärinen & Oja, 2000)

Minimization of Mutual Information

Though negentropy will be primarily used in subsequent examples and applications in this paper through fastICA, other forms of ICA estimation are still important to note for thoroughness purposes. A third commonly used estimator is the minimization of mutual information technique.

Mutual Information - The MI between m scalar variables $y_i, i = 1, ..m$ is indicated by:

$$I(y_1, y_2, ..., y_m) = \sum_{i=1}^{m} H(y_i) - H(\mathbf{y})$$

This can be rewritten as the following to descibe its relationship to negentropy:

$$I(y_1, y_2, ..., y_m) = C - \sum_{i=1} J(y_i)$$

MI measures the dependence between RV's and will be non-negative and zero if independent. Critically, this allows it to be used as a criteria for ICA transformation by minimizing mutual information of transformed components. ICA can be viewed as the minimization of mutual information.

(p. 9-10, Hyvärinen & Oja, 2000)

Maximum Likelihood Estimation (MLE)

MLE serves as a similar estimation approach to the mutual information method for the ICA model.

Particularly, the likelihood of the noise-free model can be formulated then used to find an estimate with MLE.

Log-likelihood for ICA:

Denote:

$$\mathbf{W} = (\mathbf{w}_1, ...\mathbf{w}_n) = \mathbf{A}^{-1}$$

$$L = \sum_{t=1}^{T} \sum_{i=1}^{n} log f_j(\mathbf{w}_i^T \mathbf{x}(t)) + T log|det\mathbf{W}|$$

Further simplified, this becomes:

$$\frac{1}{T} log L(\mathbf{W}) = E\{\sum_{i=1}^{n} log p_i w_i^T x\}) + log|det\mathbf{W}|$$

(p. 10 Hyvärinen & Oja, 2000)

In this, the f_i signify the density functions of the s_i, which would generally be unknown but are known here.

$$\mathbf{x}(t), t = 1, ..., N$$

are the observations of x

$$log|det\mathbf{W}|$$

is linear transformation of an RV and its density

(p. 11 Hyvärinen & Oja, 2000)

Connect to Infomax

A similar principle that has been used to develop ICA algorithms previously is Infomax (eg. the maximization of output entropy).

Given a neural network with :

$$y_i = \phi_i(\mathbf{w}_i^T x) + \mathbf{n}$$

Apply some function *H()* and maximize:

$$H(\mathbf{y}) = H(\phi_1(\mathbf{w}_1^T x), ..., \phi_n(\mathbf{w}_n^T x))$$

(pg. 211, Aapo Hyvärinen & Oja, 2001)

Maximization of mutual information will be equal to maximization of output entropy, for which the transformation can be written as:

$$H(\phi_1(\mathbf{w}_1^T x), ..., \phi_n(\mathbf{w}_n^T x)) = H(x) + E\{log|det\frac{\partial F}{\partial W}(x)|\}$$

Where:

$$\mathbf{F}(x) = (\phi_1(\mathbf{b}_1^T x), ..., \phi_n(\mathbf{b}_n^T x))$$

And therefore:

$$E\{log|det\frac{\partial F}{\partial W}(x)|\} = \sum_i E\{log\phi_i'\mathbf{w}_i^T x\} + log|det\mathbf{W}|$$

Thus, output entropy equals likelihood and that infomax and MLE are effectively equal methods.

(pg. 213, Aapo Hyvärinen & Oja, 2001)

ICA and Projection Pursuit

It is important to note the relationship between ICA and Projection Pursuit due to some significant similarities in their processes. Particularly, projection pursuit also looks at multidimensional data with the intent to find "interesting" projections. Much of its key applications are in the data exploration phase, but often projection pursuit directions can successfully visualize nuanced distribution of the data.

Beyond this, projection pursuit's goal of finding interesting data projections ties it directly to non-Gaussian distributions and how they can produce powerful visualizations of clustered data. Thus, most projection pursuit projections will be performed by determining the most non-Gaussian, and therefore, ICA components can be considered projection pursuit *indicies* However, projection pursuit does not follow a set model with rigorous definition like ICA.

If the specified ICA model holds, then non-Gaussainity optimization will generate independent components. If not, the result model can be viewed as a set of projection pursuit directions.

(p 197-198, Aapo Hyvärinen & Oja, 2001)

0.4 Pre-proccessing in ICA

0.4.1 Centering

Practically speaking, subtracting the expectation/mean vector $\mathbf{m} = E(\mathbf{x})$ will *center* the observations and make x zero-mean.

Centering data simplifies algorithmic processes and calculations.
(p. 12, Hyvärinen & Oja, 2000)

0.4.2 Whitening

As a concept, whitening serves to take the original data and generate vectors with components that are both uncorrelated and equal variance with theoretical notation for the covariance matrix of a whiten vector as stated below:

$$E(\tilde{x}\tilde{x}^T) = \mathbf{I}$$

(p. 12, Hyvärinen & Oja, 2000)

Whitening, also known as *Sphering*, removes the scale and correlation structure from the **X** dat matrix. In general, the process has been criticized for manipulating the distribution too far but still remains standard practice for pre-processing data for ICA. (p. 5, Izenman, 2003)

0.5 fastICA and Other Methods

0.5.1 Quick Look at fastICA

The next chapter will dig deeper in demonstrating the power of the fastICA R package in generating ICA models, but the key features should be pointed out before their application to two toy examples later.

Key Details

fastICA takes the data matrix X of the model and attempts to un-mix components. Explicitly, this matrix is a linear combination of matrices S (original source signals) and A (mixing matrix) by generating matrix W that maximized non-Gaussianity, which is $\mathbf{WX} = \mathbf{S}$

This non-Gaussian approximation is based on negentropy calculations discussed in prior sections, and it is used over kurtosis because of efficiency according to the authors of the package.

Specfically for the fastICA algorithm, the $H(\mathbf{y})$ function options are:

- $G(u) = \frac{1}{\alpha} logcosh(\alpha u)$
- $G(u) = -exp(u^2/2)$

(see J L Marchini <marchini@stats.ox.ac.uk> & <ripley@stats.ox.ac.uk>, 2017)

0.5.2 fastICA Algorithm

"Basic" - One Component/Unit Alogrithm

1. Choose weight vector $\mathbf{w}$
2. Let $\mathbf{w}^+$ =E{$\mathbf{w}$g($\mathbf{w}$ˆT$\mathbf{x}$)} -E{$\mathbf{w}$g'($\mathbf{w}$ˆT$\mathbf{x}$)}$\mathbf{w}$
3. Let $\mathbf{w} = \mathbf{w}^+/||\mathbf{w}^+||$
4. Repeat until converged (eg. old and new values of vector are same direction)

(p 14, Hyvärinen & Oja, 2000)

Several Units

When determining multiple independent components, the vector $\mathbf{w}$ becomes a matrix $\mathbf{W}$ composed of vectors ($\mathbf{w}_1$,...,$\mathbf{w}_n$)

Outputs must be decorrelated in this scenario, else they may converge to the same value. This is done by iteratively subtracting $\mathbf{w}_{p+1}$ with each additional projection, then normalizing that same vector.

1. Let $\mathbf{w}_{p+1} = \mathbf{w}_{p+1} - \sum_{j=1}^{p} \mathbf{w}^T\mathbf{w}_j\mathbf{w}_j$
2. Let $\mathbf{w}_{p+1} = \mathbf{w}_{p+1}/\sqrt{\mathbf{w}_{p+1}^T\mathbf{w}_{p+1}}$

This can also be written as:

1. Let $\mathbf{W} = \mathbf{W}/\sqrt{||\mathbf{W}\mathbf{W}^T||}$
2. Let $\mathbf{W}$ = (3/2)*$\mathbf{W}$-(1/2)*$\mathbf{W}\mathbf{W}^T\mathbf{W}$

(Hyvärinen & Oja, 2000, p. 15)

0.5.3 Key Properties of fastICA

1. Cubic/quadratic convergence leads to faster convergences than linear methods
2. No step size parameters
3. No estimate distribution needs to be chosen
4. Can choose suitable non-linearity
5. Components can be done one by one, reducing computational expense
6. Neural algorithm: parallel, distributed, computationally simple, and requires little memory

(p. 16, Hyvärinen & Oja, 2000)

0.5.4 Arguments in the fastICA function

fastICA (R implementation) employs several different technical arguments in its function that must be understood in order to interrupt the results of its output rigorously.

X simply the data matrix of interest
n.comp Number of components to be extracted
alg.typ :

1. if == "parallel" - components are extracted simultaneously(default)
2. if == "deflation" - components are extracted one at a time

fun Function form of G to use (see above)
alpha Constant range[1,2] used in approx. to neg-entropy when fun == "logcosh"
method :

1 if == "R" then computations are done exclusively in R
2. if == "C" then C code is used to perform computations (runs faster)

row.norm logical value indicating whether rows of X should be standardized
maxit Maximum number of iterations
tol A positive scalar giving tolerance at which the un-mix is considered to have converged
verbose level of output
w.init Initialized un-mixing matrix of dim(n.comp,n.comp) if Null matrix of RV's used

0.5.5 JADE and Infomax

Although fastICA has largely become the primary R packages for ICA modeling, JADE and Infomax methods are also available to perform the same task. In subsequent sections, their effectiveness will be evaluated in relation to fastICA.

JADE -
Uses *Joint Approximate Diagonalization of Eigenmatrices* (JADE) derived by Cardoso and Souloumiac in 1993. This approach involves finding a orthogonal rotation matrix R that diagonalizes the cumulant array of source signals. (Helwig, 2015)

Particularly, the original Cardoso paper describes the algorithm in 4 steps:

1 Create sample covariance $\hat{R}_x$ and compute whitening matrix $\hat{W}$

2 Create sample of 4th-order cumulants of whitening; compute significant eigenpairs

3 Jointly diagonialize the set with unitary matrix

4 Estimate A with $\hat{A} = \hat{W}\hat{U}$

(p.366, Cardoso & Souloumiac, 1993)

Infomax -

Also utilizes the orthogonal rotation matrix R that maximizes the joint entropy of a non-linear function of the estimated source signals and is derived from Bell and Sejnowski's 1995 paper *An Information-Maximization Approach to Blind Separation and Blind Decon*
(Helwig, 2015)

Chapter 1

Chapter 1: Toy Examples and Simulation

1.0.1 Simple Example: Two Random uniform Distributions

To demonstrate effectiveness of ICA at decomposing source signals and to develop a better understanding of the process conceptually,a simple toy example will be used. The"unknown" source components in this model are just two random uniform variables in the matrix S, distorted by some arbitrary "unknown."

Here the mixing matrix is A

$$A = \begin{bmatrix} 1 & 1 \\ -1 & 3 \end{bmatrix}$$

.

The known observations is the product of these two matrices, or matrix X.

To test ICA, apply fastICA to this simple model and generate 5 plots:

1. The pre-processed data

2. Principle Component Analysis (PCA - for comparison to a different component based model)

• **$ICA components**

3. fastICA
4. JADE method
5. Infomax
With the Random Uniform case, the distribution of signals is known to be two random uniform vectors, so it is desired to show that the ICA method can unmix the mixture matrix back into the original, known S. Therefore, the plot ICA components should look very similar to the plot of the two random uniforms.

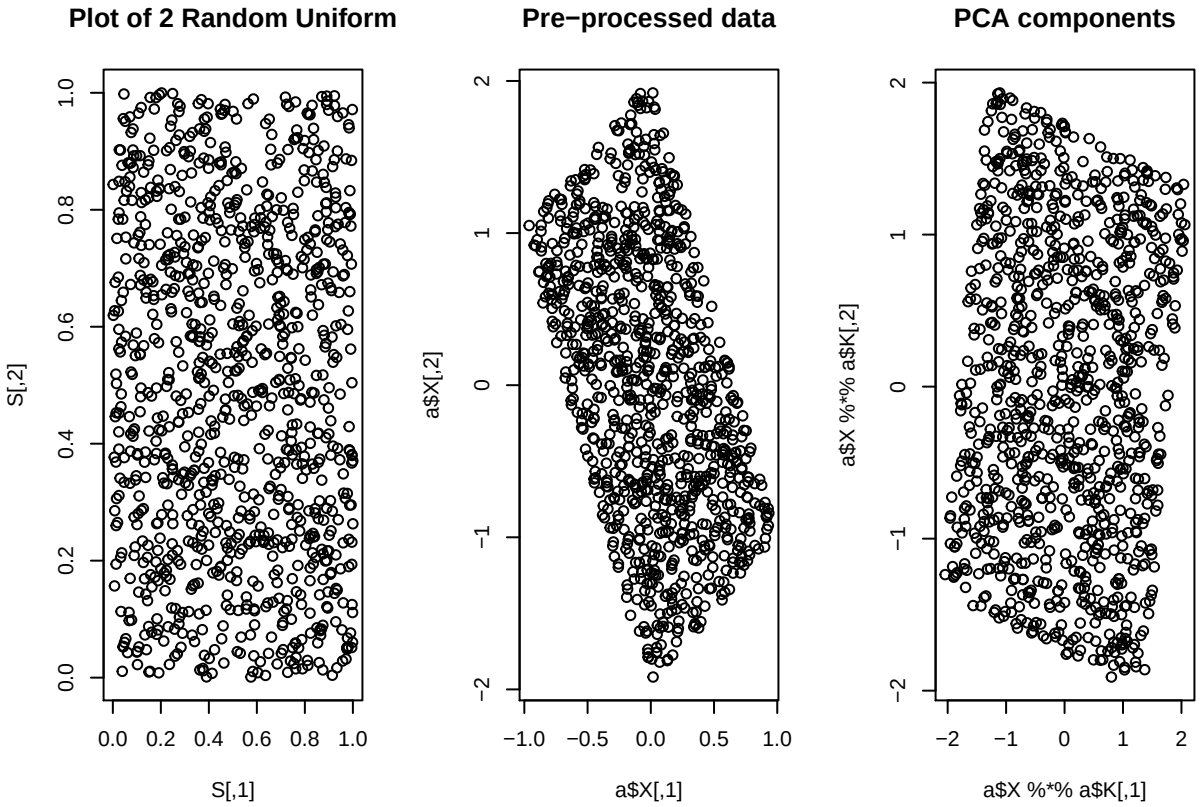
(This test case has been adapted from the fastICA package Example 1)
(see J L Marchini <marchini@stats.ox.ac.uk> & <ripley@stats.ox.ac.uk>, 2017)

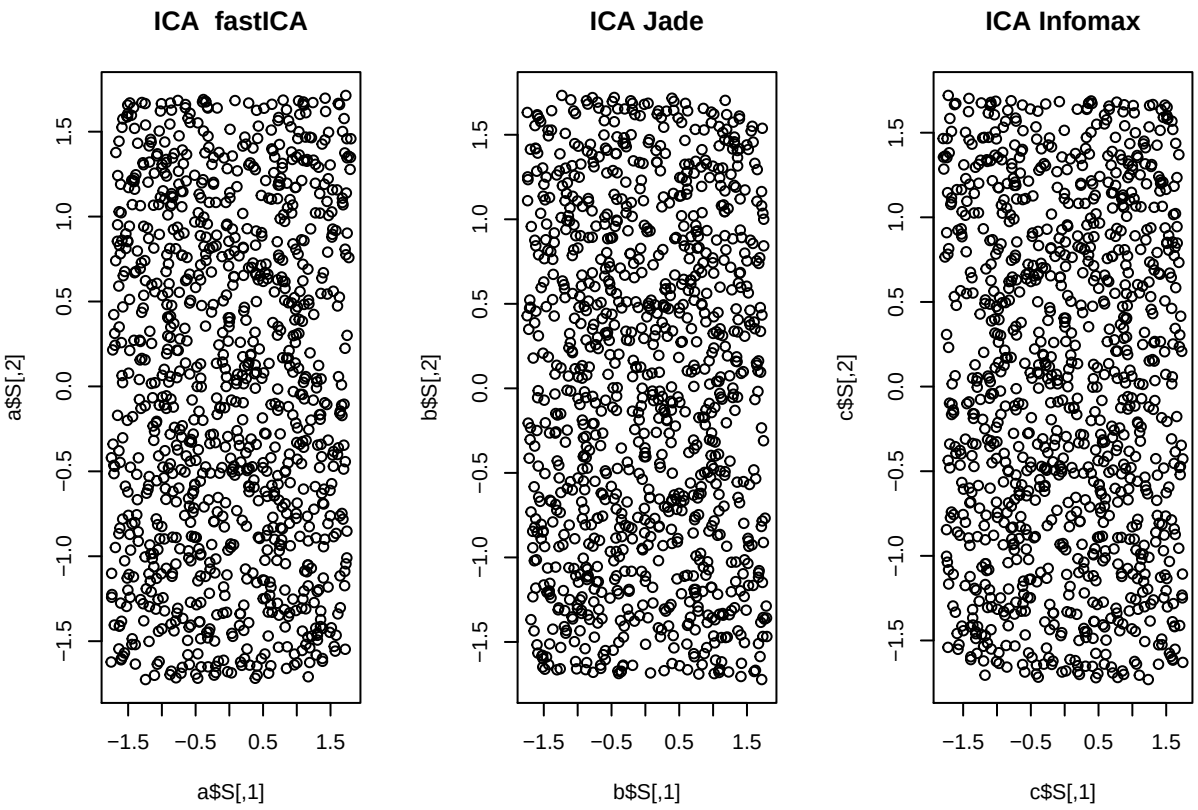
```
set.seed(10)
# Original source signals (usually unknown)
S <- matrix(runif(10000), 1000, 2)
# Mixing matrix (usually unknown)
A <- matrix(c(1, 1, -1, 3), 2, 2, byrow = TRUE)
# "Observed" mixture - always known
X <- S %*% A

#ICA
a <- fastICA(X, 2, alg.typ = "parallel", fun = "logcosh", alpha = 1,
             method = "C", row.norm = FALSE, maxit = 200,
             tol = 0.0001, verbose = TRUE)

#JADE
b <- icajade(X,2,center=TRUE,maxit=200,tol=0.0001)

#Infomax
c <- icaimax(X,nc=2,center=TRUE,maxit=200,tol=0.0001,alg="newton",fun="log")
```





Plot Interpretation

Looking at these plots, the pre-processed data (plot of the constructed X) is rotated and no signals are clear from it.

PCA appears to have flattened the data slightly and creates almost a mirror of the original data, but the true signals from the distributions still do not appear completely evident. PCA does not effectively separate sources given how it rotates data.

However, looking at the ICA transformation,the shape of the original uniform distribution clearly forms after the decomposition. ICA was able to recover the original signals quite easily in this simple simulation. The results for all 3 ICA packages (JADE,Infomax and fastICA) appear to produce approximately the same plots for this example.

Congurence Coefficient of Source Signals

To get an idea of the underlying differences between these methods, the absolute maximum column value of the congruence coefficient of each un-mixing matrix can be calculated.

Tucker's Congruency Coefficient compares uncentered correlation between the columns of two matrices of the same dimensions. The absolute value of max column of this coefficient is then taken.

```
colMax <- function(data) sapply(data, max, na.rm = TRUE)[1] #Function to find colMax
Fast1 <- colMax(abs(congru(S,a$S)))
JADE1 <- colMax(abs(congru(S,b$S)))
Info1 <- colMax(abs(congru(S,c$S)))
```

```
   Method Absolute Max Congruence Coef
1 fastICA                   0.007352430
2    JADE                   0.010033080
3 Infomax                   0.009346984
```

With just one dataset in this toy example,the results from the absolute maximum congruence coefficient test do not indicate accurately which algorithm will perform best overall. A higher value will indicate more similarities between the actual source matrix and the S generated by an ICA method. Here JADE and Infomax seem to perform better, but this value needs to be replicated and tested with more noise to determine if fastICA is truly less effective on multiple datasets. This computation will be done with the later simulation.

1.0.2 (More) Complex Example: The Cocktail Party Problem

The most common example used to explain and motivate the use of ICA (or any type of BSS) is the "Cocktail Party Problem."

Two (or more) microphones are placed in a room where to different people are speaking at the same time. Each microphone signal is defined as $x_1(t)$ and $x_2(t)$, with some observed (i.e. known) amplitudes x_1 and x_2 with time as an index. The unknown speech signals of the speakers are denoted $s_1(t)$ and $s_2(t)$ to make up the following system of equations:

$$x_1(t) = a_{11}s_1 + a_{12}s_2$$

$$x_2(t) = a_{21}s_1 + a_{22}s_2$$

This can also be rewritten as:

$$x_i = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$s_i = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}$$

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

If the weighted a_{ij} was known, this problem could be easily solved but because it is not, ICA becomes a necessary tool

Example with Cocktail Party Data

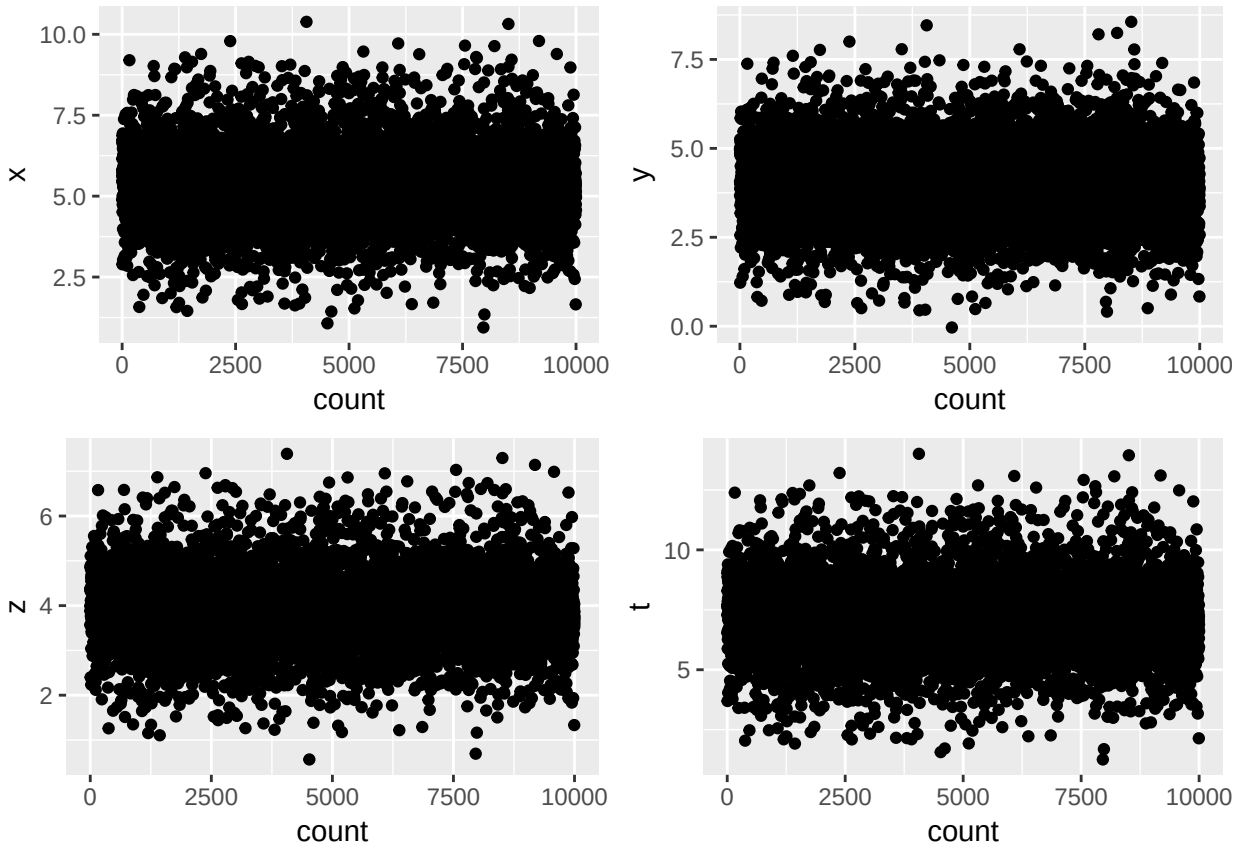
A test example and simulation can be generated using the CPPdata from the JADE package that contains 50,000 observations from 4 different microphones. (Klaus Nordhausen, 2017)

A smaller sample of 10,000 is take here for graphical purposes.

```
set.seed(40)
dat <- CPPdata[sample(10000),]
dat <- mutate(dat, count= 1:nrow(dat))
colnames(dat) <- c("x","y","z","t", "count") # sample to reduce noise
dat <- mutate(dat, count= 1:nrow(dat))
```

Original Source Signals

Unlike with the Random Uniform example, the true source signal matrix is unknown and the only information provided is the observed data. Because of this, it makes sense to look at each column (different microphone) separately as a starting point.

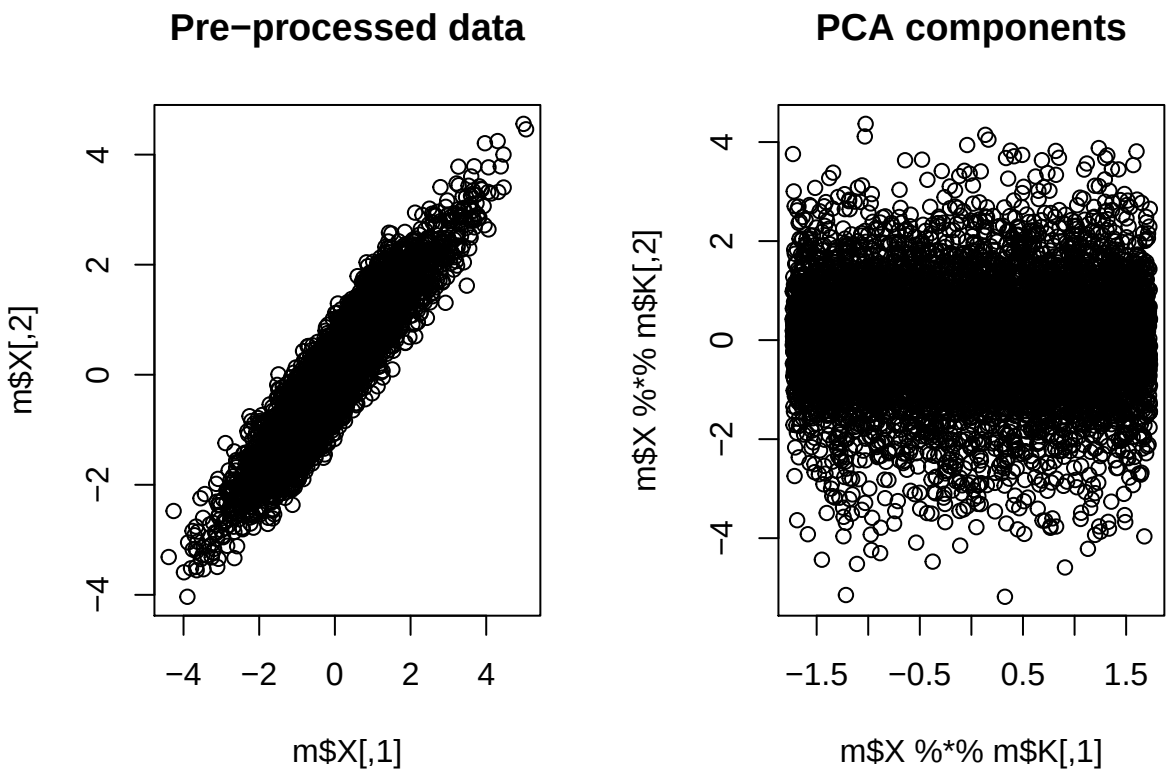


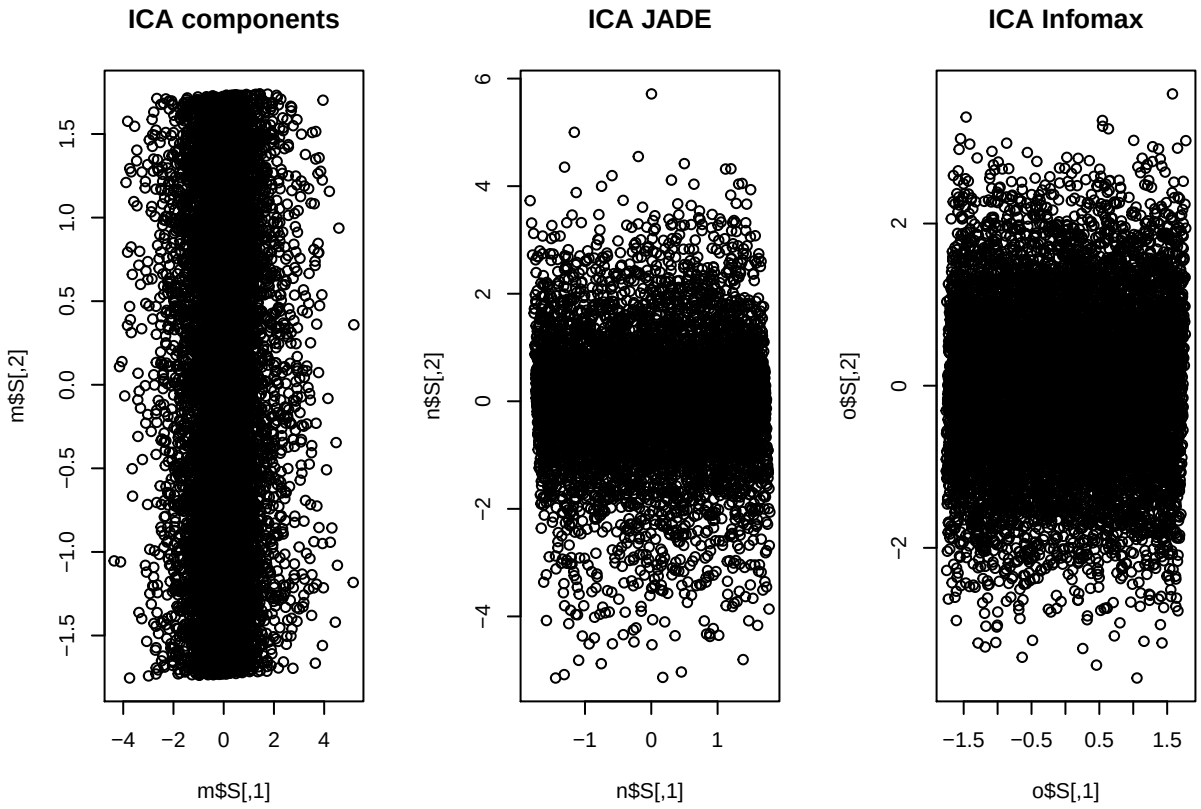
Each microphone has unique signal even to the point of quite variable means.

```
m <- fastICA(dat,2,alg.typ = "parallel", fun = "logcosh", alpha = 1,
             method = "C", row.norm = FALSE, maxit = 200,
             tol = 0.0001, verbose = TRUE)
```

```
Centering
Whitening
Symmetric FastICA using logcosh approx. to neg-entropy function
Iteration 1 tol=0.005382
Iteration 2 tol=0.000005
```

```
n <- icajade(dat,4,center=TRUE,maxit=200,tol=0.0001)
o <- icaimax(dat,nc=4,center=TRUE,maxit=200,tol=0.0001,alg="newton",fun="log")
```





Plot Interpretation

Again, the data appears to have a highly rotated set of pre-processed data and a PCA rotation that only transforms the data enough to separate the mixture ever so slightly.

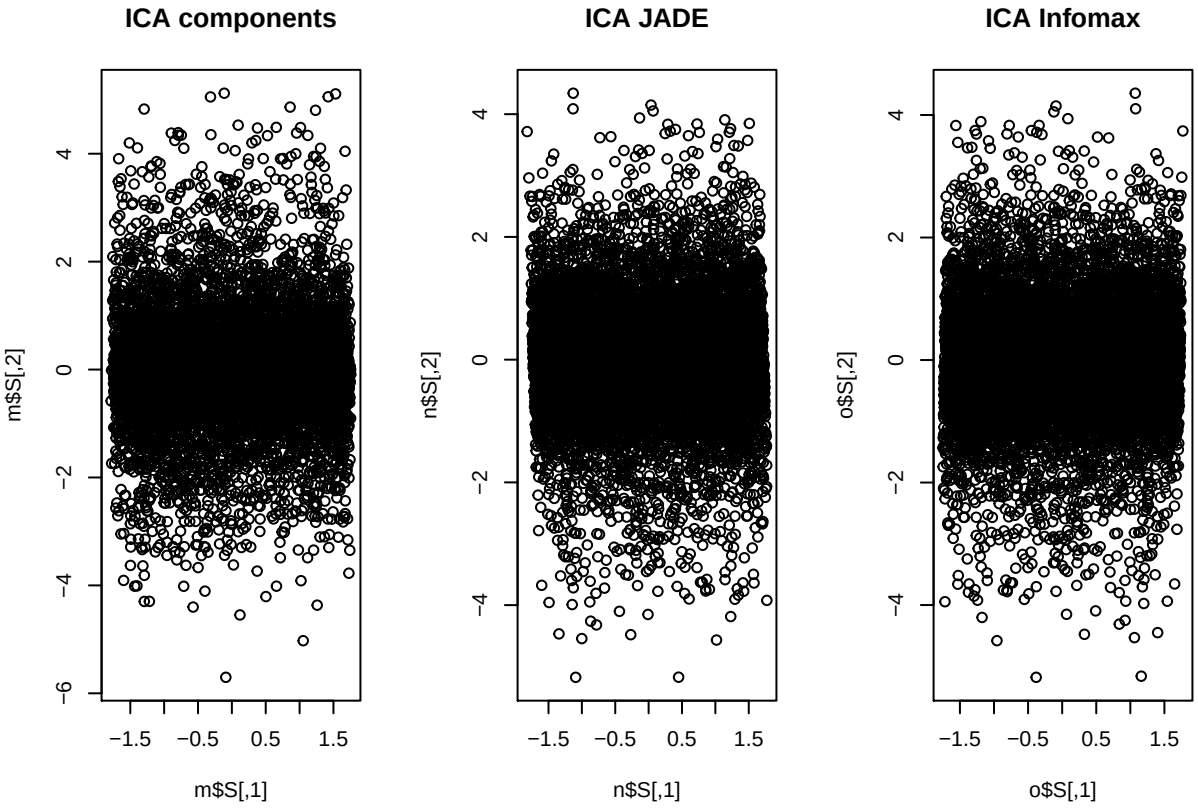
With ICA, a full rotation and a much cleaner shape begin to form, almost diamond-like. Like with the random uniform example, the superior performance of any one of the three ICA methods is not identifiable just from these primitive plots of the model.

Of note with the ICA algorithms example is the "nc" (number of components) input of the function for each of these different methods. Particularly, both the JADE and Infomax produce similar plots at $nc = 4$, which makes sense as the dataset $CPPdata$ has observations from 4 microphones. However, to generate a similar plot for fastICA, $nc = 2$ was used. The results for $nc = 4$ with ICA appear as below (JADE and Infomax shown at $nc = 2$) for reverse comparison.

```
S <- matrix(runif(10000), 1000, 2)
A <- matrix(c(1, 1, -1, 3), 2, 2, byrow = TRUE)
X <- S %*% A
m <- fastICA(dat,4,alg.typ = "parallel", fun = "logcosh", alpha = 1,
             method = "C", row.norm = FALSE, maxit = 200,
             tol = 0.0001, verbose = TRUE)
n <- icajade(dat,2,center=TRUE,maxit=200,tol=0.0001)

o <- icaimax(dat,nc=2,center=TRUE,maxit=200,tol=0.0001,alg="newton",fun="log")
```

```
par(mfrow = c(1,3))
plot(m$S, main = "ICA components")
plot(n$S, main = "ICA JADE")
plot(o$S, main ="ICA Infomax")
```



While JADE and Infomax at $nc = 2$ do not differ greatly from $nc = 4$, the fastICA plot appears more similar to the pre-processed data when the number of components is set at 4.

Congruence Coefficient of Source Signals

```
   Method        Coeff
1 fastICA 0.0018272806
2    JADE 0.0032751124
3 Infomax 0.0009626886
```

For the CPP data, the congruence coefficient compares the this value between the observed data and generated source data, as the true S is not known like in the Random Uniform case. Thus, a lower value for our absolute max congruence coefficient should be optimal. Again, running the ICA methods on more than one dataset will provide more robust results.

1.1 Simulation

Given the simplistic nature of both toy examples, a simulation can provide more robust insight into the effectiveness of ICA in extracting original source components.

1.1.1 Random Uniform Distribution Simulation

Adding noise to the Random Uniform example will be done by generating a noise vector from a random exponential distribution with $\lambda = 1$. This should inform how fastICA, JADE and Infomax perform when the Random Uniform data has been obscured further beyond the mixing matrix A.

```
n <- 1000
set.seed(10)
sim_S <- function(n,noise) {
  U <- runif(n, 0,1)
  samples <- rep(0,n)
# Add noise from a Random Exponential Distribution with lambda = 1
  X <- rexp(n,1)
  S <- runif(10000)
  for(i in 1:n) {
# Higher the noise value, the more we sample from noise vector
    if(U[i]<noise) {
      samples[i] = X[i]

    }else {
      samples[i] = S[i]}

  }
    m.samples <- matrix(samples,N,2,byrow=T)
  return(m.samples)

}
```

```
N <- 1000

# Create 3 empty vectors for each ICA algorithm
itF <- rep(NA,1,N)
itJ <- rep(NA,1,N)
itM <- rep(NA,1,N)
JF <- rep(NA, 1,N)
A <- matrix(c(1, 1, -1, 3), 2, 2, byrow = TRUE)

#Function to find colMax of the generated congruencey coefficients
```

```
colMax <- function(data) sapply(data, max, na.rm = TRUE)[1]

gen <- function(N,S, noise)  {
#Inputs: number of interations,amount of noise,
# and source signal matrix

  for (i in 1:N) {
    X[i] <- sim_S(N,noise) %*% A
    a <- fastICA(X, 2, alg.typ = "parallel", fun = "logcosh", alpha = 1,
                 method = "C", row.norm = FALSE, maxit = 200,
                 tol = 0.0001, verbose = TRUE)
    b<- icajade(X,2,center=TRUE,maxit=200,tol=0.0001)

    c<- icaimax(X,nc=2,center=TRUE,maxit=200,tol=0.0001,
                alg="newton",fun="log")

# Store a Absolute Maximum Congruence Coefficient for
# each N iteration of the ICA model
    itF[i] <- colMax(abs(congru(S,a$S)))
    itJ[i] <- colMax(abs(congru(S,b$S)))
    itM[i] <- colMax(abs(congru(S,c$S)))

# This compares Absolute Maximum Congruence Coefficient between fastICA and JADE

    JF[i] <- colMax(abs(congru(a$S,b$S)))

  }
  # Return a list of values from each method
  return(list(itF,itJ,itM,JF))
}

# Generate Coefficients for N datasets at multiple noise levels

I1 <- gen(N,S,0.05) # Noise = 0.05  Norm values
I2 <- gen(N,S,0.10) # Noise = 0.10 Norm values
I3 <- gen(N,S,0.15) # Noise = 0.15 Norm values
NoNoise <- gen(N,S,0)  # Noise = 0 / No Noise
```

1.1.2 Noise = 0 Results

```
    column    n      mean         sd    median   trimmed        mad
1  fastICA 1000 0.1638213 0.13109493 0.1375945 0.1484611 0.11741881
2     JADE 1000 0.1024091 0.04922798 0.1167199 0.1073506 0.05138732
3  Infomax 1000 0.1003643 0.05153871 0.1155706 0.1050809 0.05374063
```

```
4 JadeFast 1000 0.5240766 0.47524293 0.6889716 0.5287862 0.46111155
           min       max     range        skew   kurtosis          se
1 1.660327e-04 0.4877045 0.4875384  0.94597205 -0.1645861 0.004145586
2 1.049017e-03 0.1526784 0.1516294 -0.55175316 -1.1498735 0.001556725
3 9.359263e-06 0.1535549 0.1535456 -0.48932480 -1.2989490 0.001629797
4 4.176865e-03 0.9999916 0.9958147 -0.02324014 -1.9847584 0.015028501
```

The expectation for the Absolute Maximum Congruence Coefficient for the noiseless model was for it to be high (eg. close to 1) since the true S and the generated S by the algorithms should be similar. However, this was not the case, particularly for JADE and Infomax, as their means were below 0.10 with SD of around 0.044. fastICA's mean was larger, but the SD was high at around 0.125.

Thus, the Absolute Maximum Congruence Coefficient between fastICA and JADE was also calculated, and was found to be close to 0.51 but also had a high standard deviation. Interestingly, the median was actually close 1 at 0.9988695, suggesting that the algorithms performed similarly.

1.1.3 Noise = 0.05 Results

```
    column    n      mean         sd    median   trimmed        mad
1  fastICA 1000 0.1489319 0.09964877 0.1346393 0.1376030 0.04651286
2     JADE 1000 0.1285576 0.02007665 0.1330783 0.1300112 0.02079712
3  Infomax 1000 0.1243085 0.02718760 0.1318008 0.1279910 0.02393317
4 JadeFast 1000 0.5399478 0.48143559 0.9563555 0.5495371 0.06470635
           min       max     range       skew  kurtosis           se
1 3.334330e-04 0.4877898 0.4874563  1.1591724  1.246994 0.0031511709
2 3.785129e-03 0.1555764 0.1517913 -2.1263283 10.416212 0.0006348793
3 4.905307e-03 0.1521648 0.1472594 -1.9849818  5.459700 0.0008597475
4 5.344373e-05 1.0000000 0.9999466 -0.1250572 -1.957931 0.0152243302
```

Unsurprisingly, the results when a small amount of noise was added to our data was essentially the same, although it seems strange that the JADE and Infomax algorithms behaved more similar to the model with some noise.

Again, comparing fastICA and JADE as an alternative approach, they appeared to perform about equally on this noisier model.

1.1.4 Noise = 0.10 Results

```
    column    n      mean         sd    median   trimmed        mad
1  fastICA 1000 0.1615841 0.11899359 0.1408028 0.1471446 0.06225168
2     JADE 1000 0.1295651 0.03908884 0.1446667 0.1396517 0.01037763
3  Infomax 1000 0.1188921 0.04516841 0.1380394 0.1289802 0.01988052
4 JadeFast 1000 0.5257030 0.46904189 0.3589025 0.5301044 0.52473593
           min       max     range        skew   kurtosis          se
1 2.655194e-04 0.4877920 0.4875265  1.06081461  0.6014137 0.003762908
```

```
2 2.422218e-05 0.1529682 0.1529439 -2.28344998  4.1430261 0.001236098
3 1.105514e-03 0.1522820 0.1511765 -1.73089978  1.6937317 0.001428351
4 3.400405e-03 0.9999930 0.9965926 -0.01229774 -1.9723536 0.014832407
```

Both JADE and Infomax "improved" at the 0.10 noise level, and fastICA variability only fell marginally. The fastICA JADE did not demonstrate the same success as the median value is only 0.24, suggesting increased dissimilarity in results from the two methods.

1.1.5 Noise = 0.15 Results

```
    column    n      mean         sd    median   trimmed         mad
1  fastICA 1000 0.1517650 0.08694443 0.1464803 0.1454231 0.025503600
2     JADE 1000 0.1582890 0.01587534 0.1562030 0.1583514 0.007266911
3  Infomax 1000 0.1469350 0.01670913 0.1486793 0.1487019 0.002560352
4 JadeFast 1000 0.5447331 0.45285642 0.9042250 0.5489777 0.141995965
           min       max     range        skew  kurtosis           se
1 0.0002573546 0.4876775 0.4874202  0.93470598  1.746629 0.0027494243
2 0.0062724214 0.1889853 0.1827129 -5.86205706 54.876948 0.0005020225
3 0.0071266111 0.1660601 0.1589335 -7.19399671 53.212491 0.0005283891
4 0.0044797527 1.0000000 0.9955202 -0.02841594 -1.981007 0.0143205773
```

Oddly enough, all three algorithms performed their best on the 0.15 noise model, perhaps this indicates a need to run more iterations in order to recover more of the uniform distributions. Additionally, the random uniform may just have a high level of randomness for each dataset that does not allow the Absolute Maximum Congruence Coefficient to perform well.

Again, the JADE and fastICA comparison did not demonstrate that the performed roughly identically under these conditions.

1.2 CPP Simulation:

1.2.1 Adding noise from Random Uniform

```
x <- data(CPPdata)
set.seed(40)
dat <- CPPdata[sample(10000),]
mix <-matrix(runif(10000),10000,4)
N <- 10000
t <- rep(NA,4)

sim_CPP <- function(N,noise) {
  samples <- rep(0,N)
  t <- 1:4
```

```
  U <- runif(N,0,1)
  for(i in 1:N) {
      if(U[i]<noise) {
          samples[i] = runif(N,0,10)[i]

      }else {
          samples[i] = CPPdata[sample(N),sample(t)[1]]
      }
  }
    m.samples <- matrix(samples,N,4,byrow=T)
  return(m.samples)
}
```

```
N <- 500
C_S <- as.matrix(dat)[sample(N),]
itCF <- rep(NA,1,N)
itCJ <- rep(NA,1,N)
itCM <- rep(NA,1,N)
JF2 <- rep(NA,1,N)

gen_CPP <- function(N,S, noise)  {
  for (i in 1:N) {
    X <- sim_CPP(N,noise)
    m <- fastICA(X,2,alg.typ = "parallel", fun = "logcosh", alpha = 1,
             method = "C", row.norm = FALSE, maxit = 200,
             tol = 0.0001, verbose = TRUE)
    n <- icajade(X,4,center=TRUE,maxit=200,tol=0.0001)

    o <- icaimax(X,nc=4,center=TRUE,maxit=200,tol=0.0001,
                 alg="newton",fun="log")

    itCF[i] <- colMax(abs(congru(C_S,m$S)))
    itCJ[i] <- colMax(abs(congru(C_S,n$S)))
    itCM[i] <- colMax(abs(congru(C_S,o$S)))
    JF2[i] <- colMax(abs(congru(n$S,m$S)))

  }
  return(list(itCF,itCJ,itCM,JF2))
}
IC1 <- gen_CPP(N,C_S,0.05) # Noise = 0.05
IC2 <- gen_CPP(N,C_S,0.10) # Noise = 0.10
IC3 <- gen_CPP(N,C_S,0.15)  # Noise = 0.15
NN2 <- gen_CPP(N,C_S,0)
```

1.2.2 Noise = 0 Results

```
    column   n        mean          sd      median     trimmed         mad
1  fastICA 500 0.005989154 0.004695226 0.004878922 0.005444689 0.004256639
2     JADE 500 0.005697663 0.004477445 0.004750534 0.005169665 0.004408381
3  Infomax 500 0.005716690 0.004693025 0.004435234 0.005149401 0.004381271
4 JadeFast 500 0.505330809 0.356041955 0.539719093 0.507151751 0.536848127
            min        max      range        skew   kurtosis           se
1 3.618137e-05 0.02605655 0.02602036  1.05563779  0.8880875 0.0002099769
2 2.355080e-05 0.02295155 0.02292799  0.97077381  0.5105231 0.0002002374
3 6.902722e-06 0.02250676 0.02249986  0.99808237  0.4336876 0.0002098785
4 1.624218e-03 0.99955303 0.99792881 -0.04310171 -1.6485584 0.0159226803
```

The comparison by Absolute Maximum Congruence Coefficient for the CPP proves slightly more difficult since the true S is not known (comparison of congruence here is done between the data and generated S as mentioned earlier). Regardless, each algorithm performed very similarly (means within 0.0003 of each other) and had low standard deviation, thus the noiseless model seems to be relatively indifferent between the models.

1.2.3 Noise = 0.05 Results

```
    column   n        mean          sd      median     trimmed         mad
1  fastICA 500 0.005839718 0.004641555 0.004927834 0.005268833 0.004494524
2     JADE 500 0.005781619 0.004465908 0.004901945 0.005283814 0.004393377
3  Infomax 500 0.005776980 0.004360168 0.004914277 0.005325630 0.004420861
4 JadeFast 500 0.536839683 0.325429353 0.591683862 0.545462039 0.440532969
            min        max      range       skew   kurtosis           se
1 1.914060e-05 0.02685490 0.02683576  1.0472508  0.9069099 0.0002075766
2 5.288971e-05 0.02347204 0.02341915  1.0056448  0.7555682 0.0001997215
3 2.482226e-06 0.02175965 0.02175717  0.8992272  0.4414992 0.0001949926
4 1.216242e-03 0.99456224 0.99334599 -0.1838815 -1.4800576 0.0145536431
```

Results for the lowest level of noise are even closer than in the noiseless model.

1.2.4 Noise = 0.10 Results

```
    column   n        mean          sd      median     trimmed         mad
1  fastICA 500 0.005925766 0.004351024 0.005267391 0.005483064 0.004477307
2     JADE 500 0.005902257 0.004513415 0.004923009 0.005409258 0.004317833
3  Infomax 500 0.005940126 0.004409482 0.005133846 0.005482307 0.004196482
4 JadeFast 500 0.551310470 0.313737413 0.622876286 0.564327966 0.379089057
            min        max      range       skew   kurtosis           se
1 4.174936e-06 0.02147595 0.02147177  0.8480802  0.2542811 0.0001945837
2 1.530885e-05 0.02659126 0.02657596  1.0098493  0.8982511 0.0002018460
```

```
3 1.674796e-06 0.02677447 0.02677279  0.9745771  0.7996612 0.0001971980
4 1.357246e-03 0.99994625 0.99858900 -0.3430005 -1.2804338 0.0140307636
```

Once again, all 3 algorithms produce almost identical means.

1.2.5 Noise = 0.15 Results

```
    column   n        mean          sd      median     trimmed         mad
1  fastICA 500 0.005923584 0.004366387 0.005129439 0.005461611 0.004364728
2     JADE 500 0.006243437 0.004499542 0.005326893 0.005755261 0.004323296
3  Infomax 500 0.006003017 0.004569875 0.004994865 0.005499254 0.004515031
4 JadeFast 500 0.537741988 0.313622884 0.575066472 0.547251464 0.418546575
           min        max      range       skew   kurtosis           se
1 7.304603e-06 0.02073405 0.02072675  0.8870981  0.3354806 0.0001952708
2 8.044330e-06 0.02356652 0.02355847  0.9295177  0.4837393 0.0002012256
3 2.844015e-05 0.02439330 0.02436486  0.9784172  0.6836353 0.0002043710
4 2.052682e-05 0.98760350 0.98758297 -0.2043053 -1.3882923 0.0140256418
```

The highest Noise threshold behaved most similarly to the noiseless, in terms of slightly more variability between algorithms, with JADE computing a marginally higher mean.

1.2.6 Method Effectiveness

In conclusion, differentiating between the various ICA model algorithms may not be particular important in analysis aside from wanting a use a particular non-Gaussian estimator or because of the need to control a certain parameter within the function. Notably, JADE has fewer inputs than Infomax or fastICA, thus has more limited flexibility. For the purposes of the rest of this paper, only the fastICA algorithm will be utilized.

Chapter 2

Computation and Application to Dataset

2.1 Dataset 1: Walmart Store Data from Kaggle

From a 2014 Kaggle competition with a goal of forecasting Walmart Sales, the Walmart Store dataset provides information on sales for $n = 45$ located in different regions of the United States. Each store has Weekly Sales data from $2010 - 02 - 05$ to $2012 - 11 - 01$, information by department and a binary IsHoliday variable. Evidently, this data has fairly limited number of predictors but was one of few available retail datasets, thus was used as a warm up application before applying to a larger dataset.

(Kaggle/Walmart, 2017)

2.1.1 Prior Analysis : Kimmo Kiviluoto and Erkki Oja

In 1998, Kiviluoto and Erkki Oja used ICA on parallel Financial Time Series from a retail chain of 40 stores. They claim in their paper *Independent Component Analysis for Parallel Financial Time Series* that by removing the fundamental factors of the data, they were able to see the impact of management decisions of a particular store more clearly. (pg. 2, **???**)

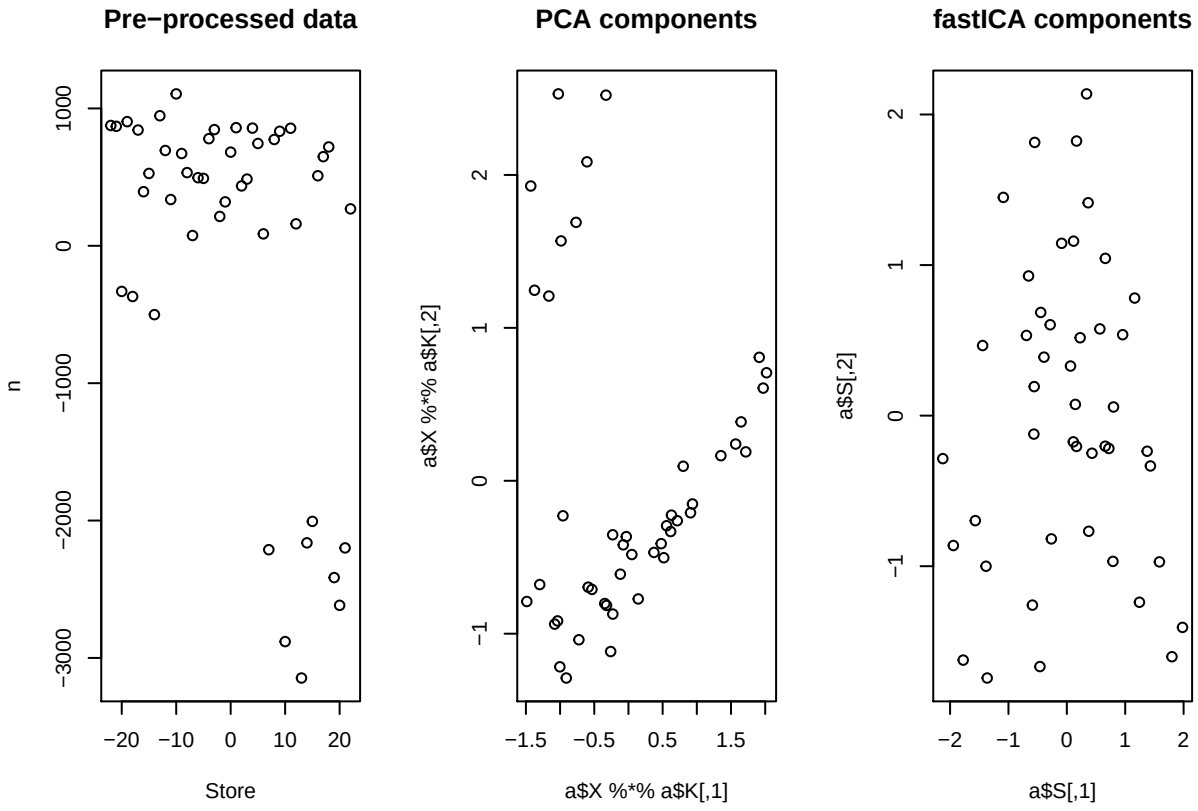
With the Walmart store data, an attempt to show a similar, interesting result using retail data will be made. However, this will look at Weekly Sales not Cash-flow data, which will probably not generate as robust results since cash-flow provides more information about financial activities of a store than just sales.

2.1.2 Exploration

2.2 FastICA Application

```
# Set nc= 3 based on dimensions
a <-fastICA(w2,3, alg.typ="parallel",fun= "logcosh",row.norm=FALSE,maxit=5,tol= 0.
```

```
par(mfrow = c(1, 3))
plot(a$X, main = "Pre-processed data")
plot(a$X %*% a$K, main = "PCA components" )
plot(a$S, main = "fastICA components")
```



From these plots, there appears to be some clustering uncovered by the fastICA algorithm. One possibility is that the cluster of 8 stores at towards the top represents the most productive in terms of sales but this could just be random noise. In general,the dataset appears to be too small in terms of both dimensions and observations to come to any explicit conclusions about underlying influences of Weekly Sales and top performing stores.

2.2.1 Dataset 2: Sample Superstore data

The Superstore Sample data appears in the December Tableau User Group presentation.

Since this data has higher dimensionality and more variety of variables, the ICA model should be more effective here. Unlike with the prior simulations, only the fastICA algorithm will be used. (Tableau, 2017)

Note: Geographic locations have been altered to include Canadian locations numerically (provinces / regions).

2.2.2 Select Variables of Interest

Sample of Variables

```
# A tibble: 21 x 1
                  x
              <chr>
 1         Row ID
 2       Order ID
 3     Order Date
 4 Order Priority
 5 Order Quantity
 6          Sales
 7       Discount
 8      Ship Mode
 9         Profit
10     Unit Price
# ... with 11 more rows
```

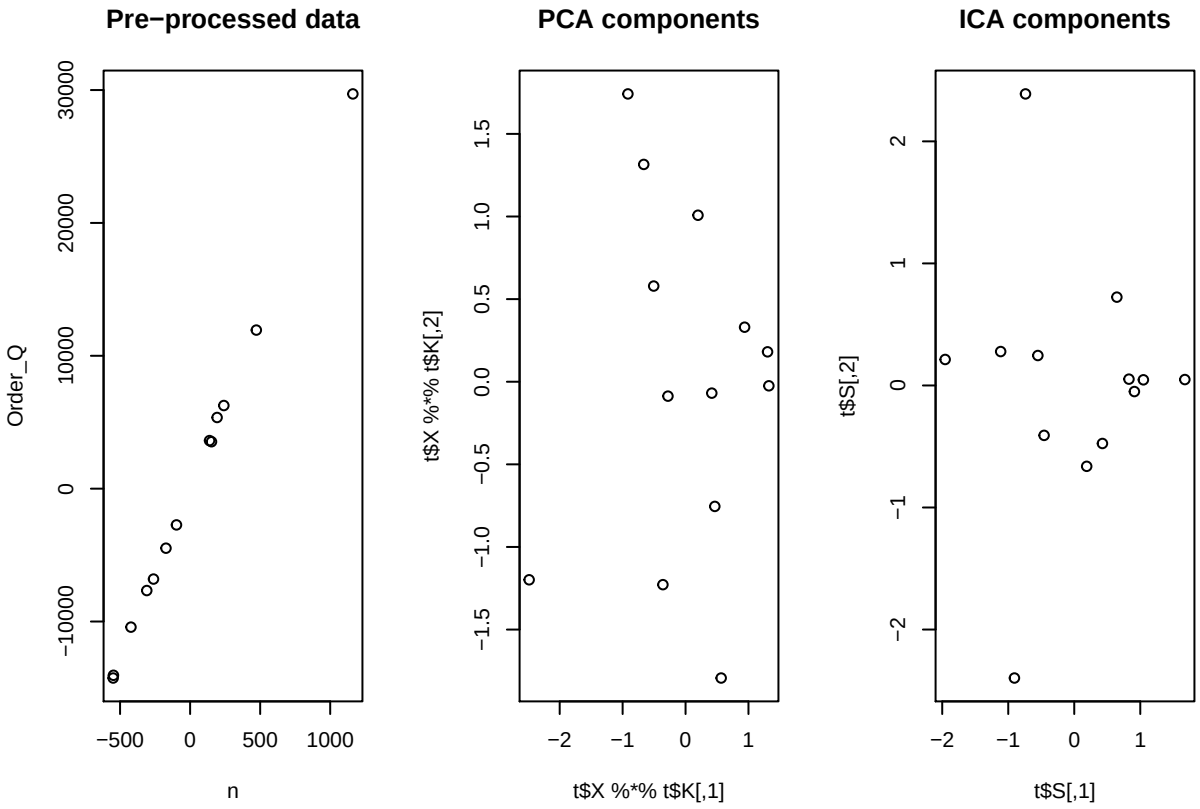
By Province

```
s <- store %>%
  select(`Order ID`,`Row ID`,`Order Date`,`Order Quantity`,Sales, Discount, Profit
  group_by(Province) %>% summarise(n = n(),
    Order_Q = sum(`Order Quantity`),
    Sales = sum(Sales),
    Discount = sum(Discount),
    Profit = sum(Profit),
    U_Price = sum(`Unit Price`),
    Ship_C = sum(`Shipping Cost`))

x <- s[,-1] # Remove Province variables to fit ICA model (non-numeric)
```

```
t <- fastICA(x,4, alg.typ="parallel",fun= "logcosh",row.norm=FALSE,maxit=5,tol= 0.
```

```
par(mfrow = c(1, 3))
plot(t$X, main = "Pre-processed data")
plot(t$X %*% t$K, main = "PCA components" )
plot(t$S, main = "ICA components")
```



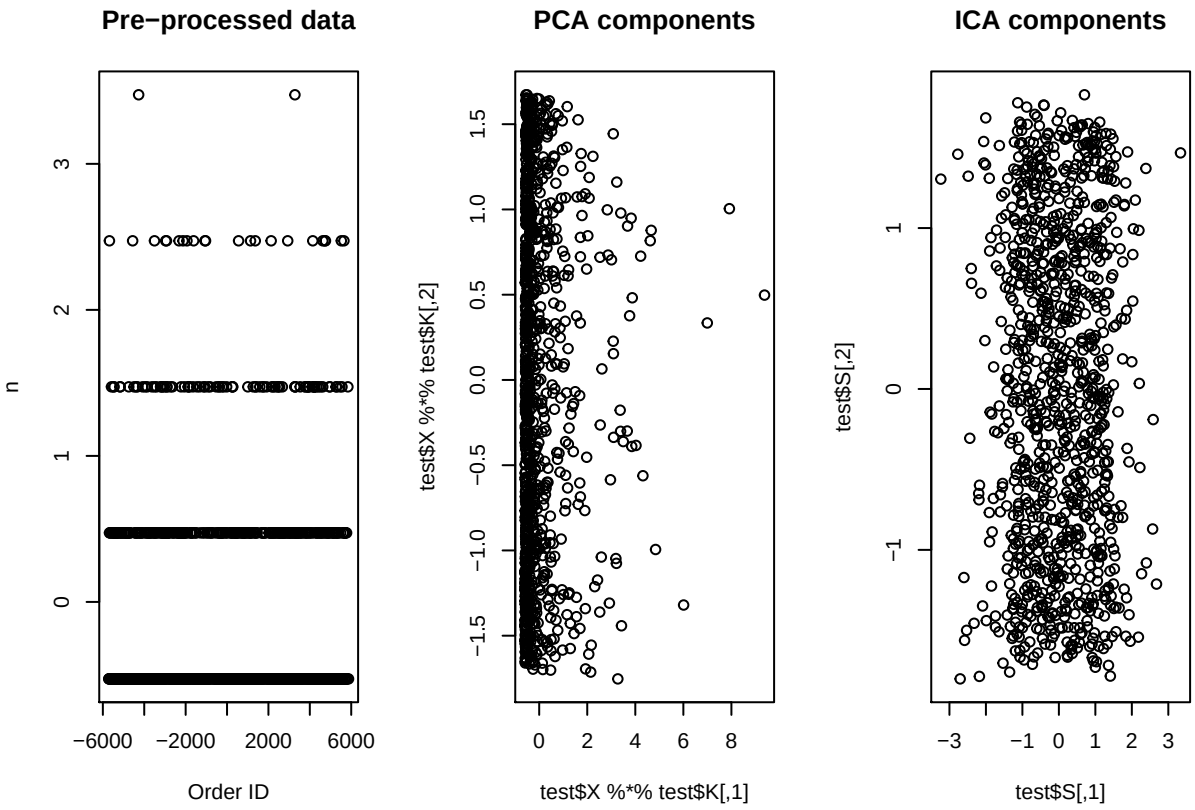
Looking at the superstore data by Province, some limited clustering exists that could potentially indicate nuances in store performance by Province or more volume of orders. Unfortunately, this view does not seem to be robust enough for any substantial conclusions.

By Order ID

```
z <- store %>%
  select(`Order ID`,`Row ID`,`Order Date`,`Order Quantity`,Sales, Discount, Profit,`Unit
  group_by(`Order ID`) %>% summarise(
  n = n(),
    Order_Q = sum(`Order Quantity`),
    Sales = sum(Sales),
    Discount = sum(Discount),
    Profit = sum(Profit),
    U_Price = sum(`Unit Price`),
    Ship_C = sum(`Shipping Cost`))
z1 <- z[sample(1000),] #take sub sample
```

```
test <- fastICA(z1,8, alg.typ="parallel",fun= "logcosh",row.norm=FALSE,maxit=5,tol= 0.00
par(mfrow = c(1, 3))
plot(test$X, main = "Pre-processed data")
```

```
plot(test$X %*% test$K, main = "PCA components" )
plot(test$S, main = "ICA components")
```



While the PCA model provides minimal information, the ICA model for the superstore data by order ID appears to have separated some particular ID's from a massive cluster of orders. These orders could be the most atypical in terms of quantity or price, but this separation indicates the effectiveness of the ICA model on the data in general.

2.2.3 Run 1000 fastICA Iterations

```
set.seed(10)
N <- 1000
itF <- rep(NA,1,N)

colMax <- function(data) sapply(data, max, na.rm = TRUE)[1]
gen_s <- function(N,S)  {
  for (i in 1:N) {
    test <- fastICA(z1,8, alg.typ="parallel",fun= "logcosh",row.norm=FALSE,maxit=5
```

```
    itF[i] <- colMax(abs(congru(S,test$S)))


  }
  return(itF)
}
```

2.2.4 Absolute Maxmimum Congruency Coefficient

```
kable(favstats(reps))
```

min	Q1	median	Q3	max	mean	sd	n	missing
4.17e-05	0.0069342	0.0172902	0.0355768	0.5157638	0.074891	0.1511403	1000	0

Like with the CPP simulation, the superstore data had to be compared to the observations as opposed to the true S. Unsurprisingly, this produces a very low coefficient with high standard deviation so does not seem inform the analysis in a particularly useful way.

Largely, these two dataset applications may have not been the most compelling examples of the power of ICA due to their limited dimensionality and lack of complexity to their data structure as a whole.

Conclusion

4.1 Concluding Thoughts

While Independent Component Analysis can be highly effective on a wide range of datasets, its ambiguities regarding particular results and relatively difficulty of interpretation explain the lack of widespread usage by the Statistical community at large. Even if the key model criteria (independence and non-Gaussianity) are met, choosing the correct number of components and making sense of the output requires a deep understanding of the data. In their study on parallel time series financial data, Kiviluoto and Oja note how the method successfully revealed underlying factors in the retail chain analyzed but also how ICA modeling calls for "inspection from a domain expert." (pg. 3, **???**)

Perhaps the best example of this need for experts to analyze results would be the work done with the ICA model on brain imaging data like EEG and MEG. In many ways, ICA has proven to be a valuable tool for neuroscience researchers, but this application would not be possible with analysis by statisticians alone. Although successful understanding of results in this case may always be limited to experts, new ICA algorithms with more explicit outputs could potentially be developed to improve breadth of its application.

Appendix A

This first appendix includes all of the R chunks of code that were hidden throughout the document (using the `include = FALSE` chunk tag) to help with readability and/or setup.

In the main Rmd file:

```
# This chunk ensures that the acstats package is
# installed and loaded. This acstats package includes
# the template files for the thesis and also two functions
# used for labeling and referencing
if(!require(devtools))
  install.packages("devtools", repos = "http://cran.rstudio.com")
if(!require(acstats)){
  library(devtools)
  devtools::install_github("Amherst-Statistics/acstats")
}
library(acstats)
library(knitr)
library(readr)
library(fastICA)
library(ica)
library(gridExtra)
```

In :

Appendix B

The Second Appendix, for Fun

References

Aapo Hyvärinen, Juha Karhunen, & Oja, E. (2001). *Independent component analysis*. New York, New York: A Wiley Interscience Publication.

Cardoso, J. F., & Souloumiac, A. (1993). Blind beamforming for non-gaussian signals. *IEE Proceedings F - Radar and Signal Processing*, *140*(6), 362–370. http://doi.org/10.1049/ip-f-2.1993.0054

Helwig, N. E. (2015, August). Independent component analysis. *"Proceedings of" SIGGRAPH 2000*. Retrieved from https://cran.r-project.org/web/packages/ica/ica.pdf

Hyvärinen, A., & Oja, E. (2000). Independent component analysis: Algorithms and applications. *"'Independent Component Analysis: Algorithms and Applications' Neural Netw. 200*, 411–30.

Izenman, A. J. (2003). What is independent component analysis?

J L Marchini <marchini@stats.ox.ac.uk>, C. H. <. r, & <ripley@stats.ox.ac.uk>, B. D. R. (2017, June). FastICA algorithms to perform ica and projection pursuit. Retrieved from https://cran.r-project.org/web/packages/fastICA/fastICA.pdf

Kaggle/Walmart. (2017, September). Walmart recruiting - store sales forecasting. Retrieved from https://www.kaggle.com/c/walmart-recruiting-store-sales-forecasting

Klaus Nordhausen, J. M., Jean-Francois Cardoso. (2017, January). Blind source separation methods based on joint diagonalization and some bss performance criteria. *N/A*. Retrieved from https://cran.r-project.org/web/packages/JADE/JADE.pdf

Puigt, M. (2011). *A very short introduction to blind source separation*. Heraklion, Crete, Greece: Foundation for Research; Technology – Hellas Institute of Computer Science.

Tableau. (2017, September). Sample - superstore sales. Retrieved from https://community.tableau.com/docs/DOC-1236