Evaluation of BIG-SIR and Cross-Method Comparison in SDR

Minju Lee

Abstract

SIR has been an effective method for dimension reduction, but when dealing with big data sets SIR have had issue with increase in computational time. There have been several papers aiming to solve this issue, one of which is the BIG-SIR method. As simple as BIG-SIR method is, there are no R package or many experiments in evaluating the effectiveness of BIG-SIR. Thus, we focused on [1)] making a R function of BIG-SIR ourselves and [2)] evaluating the effectiveness of BIG-SIR by changing the subset size and data size. Our goal was to see if BIG-SIR is really effective in reducing time and maintaining the quality of SIR and also to [3)] find somewhat of a criterion for choosing the right subset size that could reduce the most amount of time. Moreover, we also tried this [4)] BIG-SIR process on different SDR methods (SAVE, SIRII) that satisfies the assumptions that BIG-SIR (or BIG algorithm) requires. Results showed that BIG-SIR was able to significantly reduce the computational time, but due to the big dataset of each chunk, SIR did not show consistency in direction estimation. The ideal subset size preferred was to not exceed 1.7Mb for BIG-SIR (foreach package). BIG-SAVE and BIG-SIRII had a different result to BIG-SIR, as the measurement quality was robust and identical to the original SDR methods, but the computational time did not show significant reduction. Though BIG-SAVE seemed to show that SAVE exceeded the computational time once the data set reached 419Mb, BIG-SIRII remained almost similar to SIRII. This shows that additional evaluation on measurements by applying different models for BIG-SIR and creating an efficient algorithm that reduces the computational time for BIG-SAVE and BIG-SIRII would be necessary. Moreover, experiments on how to apply BIG-SIR in real datasets should be studied, as satisfying the regression linkage assumption in BIG-SIR is extremely hard in real life datasets. Thus, alternative methods that does not require regression linkage assumption, or can transform real datasets to satisfy such conditions would also need to be experimented.

1. Introduction

SIR is a rather fast SDR procedure that is computationally simple when having the right amount of data. However, imputing SIR has its limits. For big datasets that exceed 400Mb, the computational time for SIR reaches up to 10 second, and when the CPU doubles (800Mb), the system slows down up to 70 seconds and reaches its memory limits. To solve this problem M. Chavent[1] and. B. Liquet[2] introduced a new method called BIG-SIR, where each big dataset is divided into G subsets and SIR procedure is applied in each G subsets. Afterwords, their subsets are reunited according to their proximity to real directions.

In this research, our goal is to provide an R function to the BIG-SIR algorithm that is explained by B. Liquet, see in what situations BIG-SIR works best, and provide somewhat of a guideline to finding the right amount of subset size for BIG-SIR procedure. Since the R code provided by B, Liquet 「BIG-SIR a Sliced Inverse Regression approach for massive data」 relies heavily on R packages, some of which have expired, we will create our own R function for BIG-SIR. Moreover, we will further explore to see if this "divide-and-conquer" method is applicable towards other SDR methods and evaluate with both the measurement quality and the computational time.

2. BIG-SIR

2.1 Basic Information of SIRs

Assumption 1. Linear Condition (LC):

$$E[X'\theta|X'\beta] \; is \; linear \; in \; X'\beta, \quad \forall\theta\epsilon \; \mathbb{R}^P$$

This assumption is satisfied when X has an elliptical distribution, concluding that the projected x | y subspace remains a linear relation with linkage to β.

$$\mathrm{E}[\mathrm{x}-\mathrm{E}(\mathrm{x})\big|\beta^{\mathrm{T}}\mathrm{x}] = \; \mathrm{P}^{\mathrm{T}}{}_{\beta}(\Sigma)[\mathrm{x}-\mathrm{E}(\mathrm{x})]$$

Linear Condition assumption is also important in BIG-SIR, since when calculating the proximity of each subspace of EDR directions need to resemble the linear relations of the original data. BIG-SIR takes this assumption a bit further and adds Assumption 2, thus makes sure that both Y and projected subspace x | y resemble the same $X'\beta$ relations.

This assumption also highlights an important characteristic of SIR, which is that when the data set increases, the projected results can change (meaning it is not robust to even the same dataset). This is due to the fact that big datasets have more diversity in their patterns, leading to a variety in the projected directions as well. On top of that, when there are noises (errors) in certain functions, the estimated SIR directions could be affected easily.

2.2 BIG-SIR Algorithm

Assumption 2. Linear Regression:

$$Y = \; f(X'\beta, \varepsilon)$$

where Y depends on X through β. This assumption is important as it preserves the relations of response variable Y and X even after the data has been divided into subsets. This is an additional

[1] M. Chavent., 'A sliced inverse regression approach for data stream'
[2] B. Liquet 'BIG-SIR a Sliced Inverse Regression approach for massive data'

assumption required other than the LC assumption for SIR, where this time the full data of y and x needs to have a linear combination.

Thus, there will be two assumptions necessary for BIG-SIR, which is that the predictor and response variable need to have a linear relationship (Linear Regression) with parameter $\beta \in \mathbb{R}^P$ and that the linear condition of $E[X'\theta|X'\beta]$ also needs to have a linear relationship.

BIG-SIR algorithm proceeds the following step:

(1) Divide the dataset into G subsets.
(2) Use the SIR estimator for each subset.
(3) Combine the EDR directions from each subset that are most collinear.

The process is quite simple, but the importance of this algorithm relies heavily on step 3, the method of how you combine the divided subsets. For the criteria on how to combine each subset that matches with the real direction, B. Liquet and M. Chavent discusses an idealistic and practical solution to step 3.

Theorem 1.

$$\max_{a \in \mathcal{R}^P} \sum_{g=1}^{G} w_g \, m(b_g, a) \qquad s.t \, ||a|| = 1$$

where $m(b_g, a) = \cos^2(b_g, a) = (b_g{}^T a)^2$, $\sum_{g=1}^{G} w_g = 1$ $(Weight)$. This function is to find the subspace that has the maximum collinearity (proximity) between standardized direction 'a' (also could be interpreted as the real eigenvectors of the reduced direction) and SDR direction of each subspace 'b_g'. Though this calculation would be the most idealistic approach to find the best SDR direction within each subset, M. Chavent introduced a simpler way of combining subsets.

Theorem 2.

$$\widehat{M_G} = \sum_{g=1}^{G} w_g \, \hat{b}_g \, \hat{b}_g{}^T$$

Since $w_g \, \hat{b}_g \, \hat{b}_g{}^T$ returns p x p dimension ($\hat{b}_g$ has dimension of p x r) for each subset, adding all G subsets could return the eigenvectors of all p subsets. After that, all we have to do is select r columns, which will be the estimated SDR direction. A problem that comes up with Theorem 2 is that when adding G subsets into one matrix, due to the fact that $\hat{b}_g$ gives standardized results, when the number of subsets G gets bigger, $\widehat{M_G}$ matrix gives a value near 0. Because this paper will evaluate the quality of each SDR with the distance between true beta subspace $\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 0 & 0 \end{pmatrix}$, the distance of BIG-SIR direction and SIR will not be in the same scale.

Thus, we introduce an alternative way of using Theorem 1 in a more efficient way.

Theorem 3.

$$\hat{b}_g\ [\arg\max_g(\ w_g\ \hat{b}_g\ \hat{b}_g{}^T)] \approx \max_{a\in\mathcal{R}^p} \sum_{g=1}^{G} w_g\ m(b_g, a)$$

Due to Assumption 2 (Linear Regression), the relations of Y and X will be dependent in a linear relation through β. Thus, even when the data of X and Y are randomly divided, Yg and Xg will still remain the same relations of the full data, thus make it okay to separately do SIR. The results of each subset will reduce time (when the right amount of G is selected), and therefore will not only find SIR results similar to SIR, but also reduce computational time for calculation.

There are three types of methods introduced by B. Liquet that explains how to make the BIG-SIR function in R. The only difference between these methods is how to divide and combine each EDR directions to get the final estimator. The first one is simply using the for-loop provided in R. Though for-loops are convenient and intuitive for interpretations, as subset G expands, the for-loop has a hard time iterating the process over and over again. To solve this problem, using the foreach package to run loops parallelly is introduced as a solution. The last method of combining bigmemory and foreach is also introduced, however, due to the fact that bigmemory package does not exist anymore, we will use for-loop and foreach package and compare the results.[3]

2.3 Application to Multi-indices Model

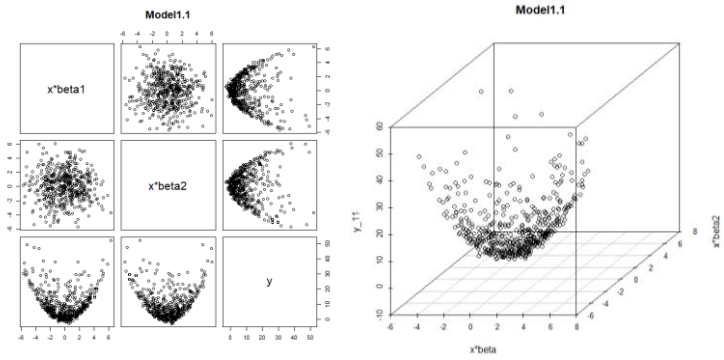
Model 1.1:

$$Y = (x^T\beta_1)^2 + (x^T\beta_2)^2 + \ \varepsilon$$

where $X \sim N_p(0_p, \Sigma)$ $\varepsilon \sim N(0,\ 0.2^2)$

$$\beta_1 = (0, \dots, 0, 1, -1, 2, -2)/\sqrt{10},$$

$$\beta_2 = (1, -, 1, 2, -, 2, 0 \dots 0)/\sqrt{10}$$



Model 1.1 follows a muti-indices model that has a symmetric dependence with $(x^T\beta)$ and Y. In this situation, SIR method would not perform well, rather second-order methods such as SAVE or SIRII would be more helpful in finding the right estimator.

2.3.1 Quality (Distance) and Computational Time of Model 1.1 Small Dataset (~4.2 Mb, n≦50,000)

[3] R Code (last page)

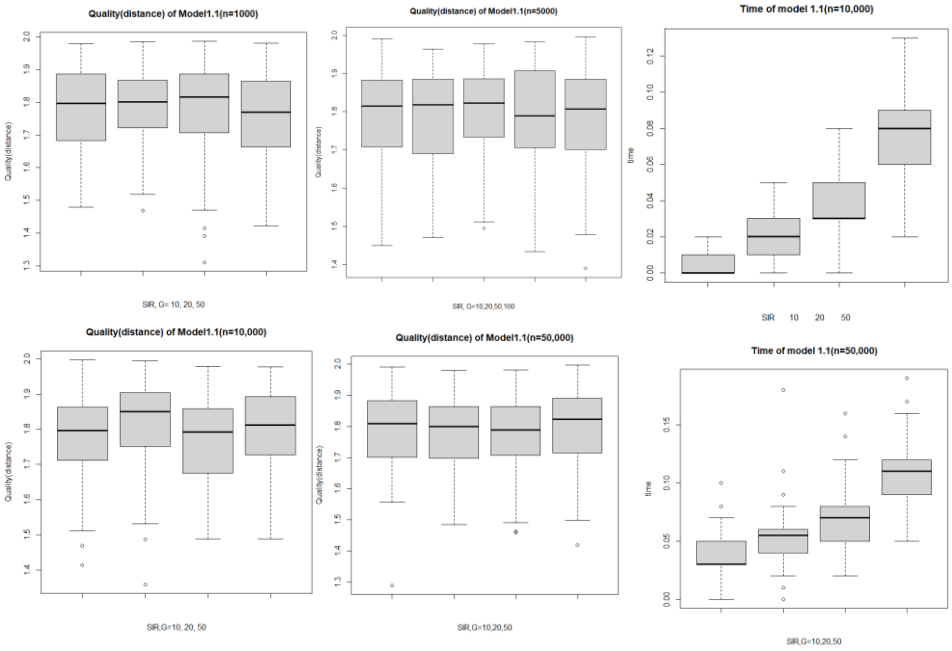


Figure 1. Boxplot of Quality (Distance) and Time difference between SIR and BIG-SIR for different Subset Size(G) and Data Size(n). Iteration was conducted 100 times to see the Boxplot.

Figure 1 shows that in small datasets, where even for the biggest data, SIR process takes nearly 0 seconds to give its estimators. Thus, having the additional process of dividing the data into subsets and selecting the one with the most collinearity would be an unneccessary process. Overall, for the quality measures, the difference between SIR and BIG-SIR is not shown, thus could say that quality for SIR and BIG-SIR is similar when the data size is small.

2.3.2 Quality (Distance) and Computational Time of Model 1.1 Big Dataset n= $10^{7,}$ 839.2Mb)

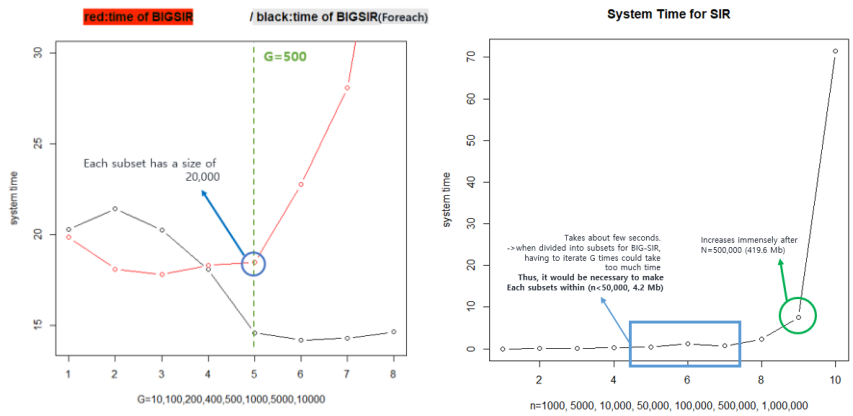


Figure 2. System time for Big Dataset

Figure 2 shows the difference in computational time for SIR, BIG-SIR (for loop), BIG-SIR (foreach). The plot on the right shows that normally the computational time for SIR is near 0 until it reaches approximately 4Mb. Though the calculation time is small (less then 5 seconds), having the G subset procedure make the total computational time increase immensely. This is also a good indicator that when deciding the size for G (subsets), it is important to keep each chunk size less than 4Mb, so that the computational time doesn't go overboard. In this case, the computational time for SIR exceeded

massively when CPU went over 400 Mb (n=500,000).

In the left plot, it shows the computational time between BIG-SIR (for loop), BIG-SIR (foreach) when n=1,000,000 is fixed and the size of G changes. It also shows that for BIG-SIR (for-loop) time increases massively when G size increases up to 500. On the other hand, the foreach BIG-SIR decreases as G>500. It clearly shows that foreach's parallel loop is not affected to G (subset) size, and thus can be advantageous when dividing massive data into small chunks and iterating each subset G times to find the right SIR estimator.

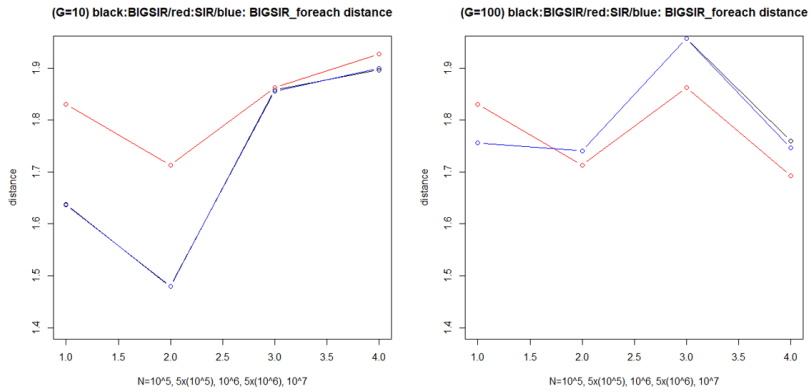


Figure 3. Quality(distance) difference for SIR, BIG-SIR, BIG-SIR (foreach)

Figure 3 shows the difference in distance between SIR, BIG-SIR, BIG-SIR (foreach) and true beta subspace $\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 0 & 0 \end{pmatrix}$. While both subset size shows that BIG-SIR and BIG-SIR foreach gives similar results, G=10 seems to give better results when subset size is smaller, as having big chunk size to execute estimated direction will not have good quality. On the other hand, G=100 seems to show that when n size is small, SIR has worse results than BIG-SIR. On the other hand, when the data increases, it shows that BIG-SIR results worsen.

However, this plot should be taken with a grain of salt because SIR is not a robust method (this is shown in Figure 1 with the high variance and commented in assumption 1), and when the size of the data increases, it is likely to give different results each time SIR process is repeated.

3. Applying BIG-SIR Algorithm to other SDR Methods

3.1 BIG-SAVE

3.1.1 Brief Review of SAVE

Assumption 3 Constant Conditional Variance (CCV):

$$Var(X|\beta' X) \; is \; a \; nonrandom \; matrix$$

SAVE requires two assumptions, one is LC (assumption 2) and the second is CCV. Having CCV assumption satisfied gives the advantage of $\text{var}(X|\beta' X) = \Sigma Q_\beta(\Sigma) = \Sigma(1 - P_\beta(\Sigma))$. Finding SDR directions by using the orthogonal direction of Var(x|y) could be advantageous in several ways. One is that this method can effectively find directions in datasets that have a symmetric relation with x and y. Moreover, finding the SDR direction with this method will give a more robust result, as it only considers direction of x|y, a concentrated subspace that is less effected to outlier.

3.1.2 BIG-SAVE algorithm and its application to Model 1.1

Since Model 1.1 seems to have constant variance in x and shows symmetric relations with x and y, SAVE would be a better method for SDR. Thus, in this section, we will apply SAVE, BIG-SAVE, BIG-SAVE (foreach) method and assess them based on their performance.

BIG-SAVE algorithm has the same process as BIG-SIR, but only this time SAVE estimator is used to evaluate each subset. Similar to BIG-SIR, we will use Theorem 3 ($\hat{b}_g\ [\arg\max_g(w_g\ \hat{b}_g\ \hat{b}_g{}^T)]$) to recombine the subsets and return the BIG-SAVE estimator.

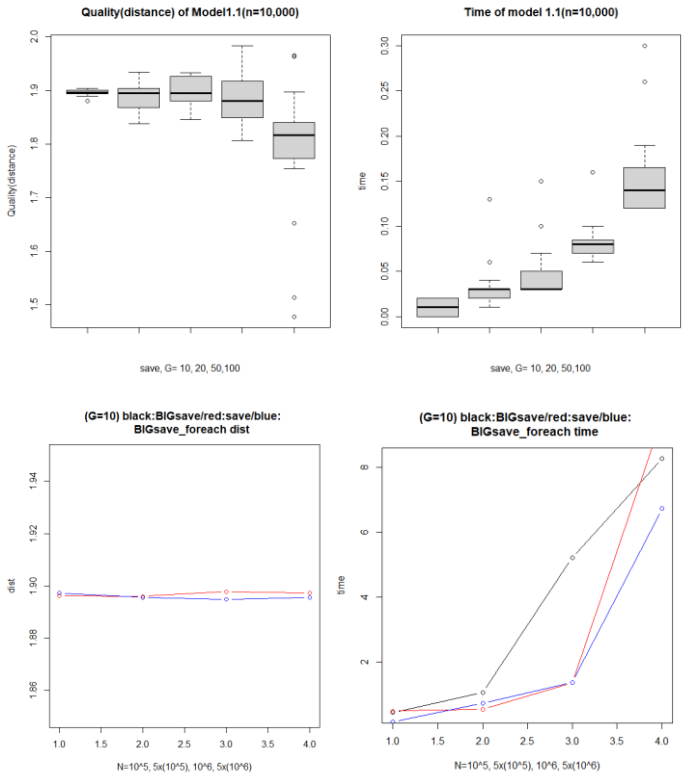


Figure 4. Quality and Time measurement for Small (top two boxplot) and Big (bottom two boxplot) dataset.

Figure 4 shows both the quality and time change for big and small data. The two plots on the top measured the performance of a data with (n=10,000) with SAVE and BIG-SAVE with different subset size. The process was iterated 20 times. It shows that overall, compared to SIR, the SAVE method is much robust than SIR and gives a constant result. On the other hand, BIG-SIR seems to lose its robustness when the subset size increases. This is due to the selection of $\hat{b}_g\ [\arg\max_g(w_g\ \hat{b}_g\ \hat{b}_g{}^T)]$ altering when there are more options to choose from (when subset G increases). The positive effect to that is the quality of measurements increasing, giving better estimators then the original SAVE. Similar to BIG-SIR, in small datasets, having too much iteration will only make the computation time increase. However, since the quality of measurement seems to increase, implementing BIG-SIR with big subsets could be an option for better measurement.

The bottom two plots represent the performance measurement of big data sets (8.4~839.2Mb). The quality measurement for BIG-SAVE for loop and foreach seems identical, and the performance of BIG-SAVE and SAVE is also very similar when dealing with big datasets. The computational time, on the other hand, shows that as data size increases, SAVE performance seems to get worse, and BIG-SAVE foreach seems to take the least amount of computing time.

3.2 BIG-SIRII

3.2.1 Brief Introduction to SIRII

Theorem 4 SIRII matrix:

$$M = E\{(V - E(V))\Sigma^{-1}(\mathrm{V} - \mathrm{E(V)})'\}$$

where V = V(X|β′X). SIRII also assumes LC and CCV is satisfied. SIR-II also focuses on direction of V(X|β′X), thus can say that SIR-II will be effective estimator for symmetric dataset.

3.1.2 BIG-SIRII algorithm and its application to Model 1.1

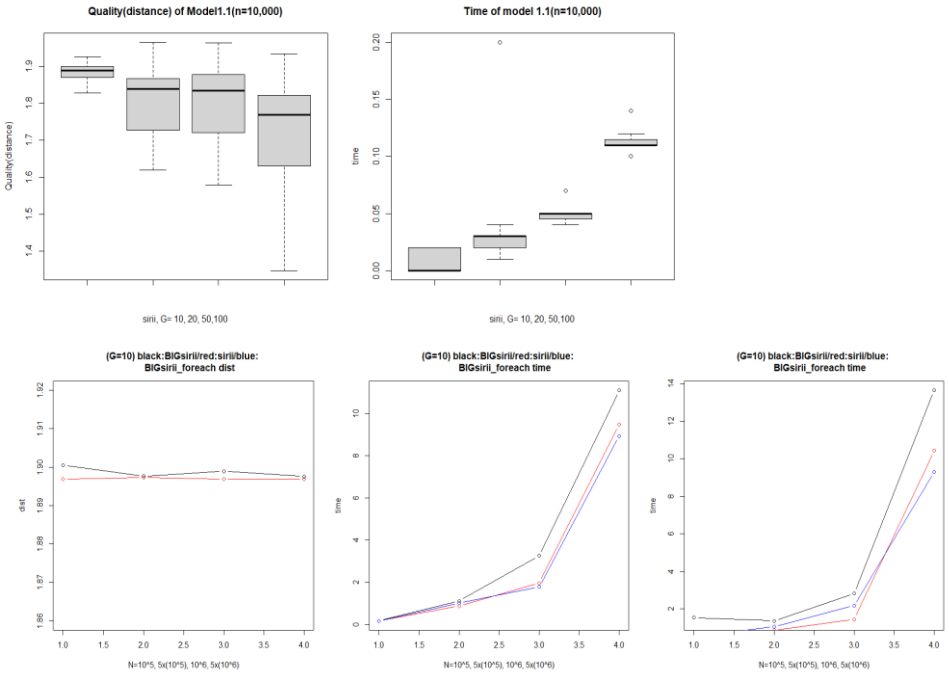


Figure 5. Quality and Time measurement for Small (top two boxplot) and Big (bottom two boxplot) dataset.

Figure 5 shows the quality and computational time for both small and big datasets. Similar results with BIG-SIR and BIG-SAVE, having big subsets (G) for small dataset (~4.2Mb) increases the computational time. Similar with BIG-SAVE, SIRII shows robust quality regardless of subset size, but also shows an increase in quality when the size of subset increases. In a big dataset, the distance of BIG-SIRII and BIG-SIRII (foreach) gives identical estimators and both BIG-SIRII and SIRII gives robust distance near 1.9. On the other hand, the computational time for BIG-SIRII (for-loop) does not seem better than SIRII, and both BIG-SIRII methods do not show dramatic improvement in system time compared to SIRII.

4. Discussion

The results showed that BIG-SIR was highly affected by the iteration, and thus BIG-SIR (using foreach package) was preferred when each subset size did not exceed 20,000 (approximately 1.7Mb). It also showed that BIG-SIR (using foreach package) was not affected by subset size, thus was advantageous to reduce the subset size as much as possible for reduction in computational time. However, the quality measurement of BIG-SIR was a tough issue, as SIR does not give robust estimators due to the big data set size and outliers. Thus, we believe that additional evaluation on

measurements by applying different models for BIG-SIR might be necessary in the future.

BIG-SAVE and BIG-SIRII had a complete opposite result from BIG-SIR, where the quality of measurements was robust, as BIG-SAVE and BIG-SIRII both gave exact results to SAVE and SIRII. However, though the computational time did decrease when using the foreach package, it was not as dramatic as BIG-SIR. Though BIG-SAVE showed some hope, as SAVE computational time exceeded both BIG-SAVE and BIG-SAVE (foreach) when it reached 419Mb ($n=5*10^6$), BIG-SIRII (foreach) did not get significantly better than SIRII. Thus, more studies on creating an algorithm that can reduce the computational time for BIG-SAVE and BIG-SIRII and also not harm the quality that BIG-SAVE and BIR-SIRII already has would be necessary.

Moreover, more experiments on how to apply BIG-SIR in real datasets would also need to be studied. One of the hardest parts of applying BIG-SIR is satisfying the regression linkage (Assumption 2), which is a difficult condition to satisfy in real life datasets. Thus, alternative methods that does not require Assumption 2, or can transform real datasets to satisfy such conditions could be an interesting topic to research in the future.

R code

BIG-SIR (for loop)

```
bigsir = function(x,y,h,r,ytype,G){
  chunk.x=list(); chunk.y=list()
  Wg=c(); Mg=c(); beta.BIG=c()
  #Step 1. split the data into G subsets
  n1 =nrow(x)
  chunk_size1 = ceiling(n1/G)
  indices1= rep(1:G,  each = chunk_size1)
  chunk.x = split(x,indices1)
  chunk.x =lapply(chunk.x, function(chunk) {
   matrix(as.vector(chunk), nrow=chunk_size1,
          ncol=p, byrow = FALSE)
  })
  chunk.y = split(y, indices1)

  #step 2. apply SIR in each subsets
   for (i in 1:G){
      beta.BIG[[i]] = sir(chunk.x[[i]],chunk.y[[i]],h,r,ytype)
      Wg[i] = nrow(chunk.x[[i]])/n
      Mg[i] = sum(Wg[i]*t(as.matrix(beta.BIG[[i]])) %*%
                    as.matrix(beta.BIG[[i]]))
   }

    #Step 3. return the EDR with most proximity
    return(beta.BIG[[which.max(Mg)]])
}
```

BIG-SIR (foreach)

```
bigsir_for = function(x,y,h,r,ytype,G){
  chunk.x=list(); chunk.y=list()
  Wg=c(); Mg=c(); beta.BIG=c()
  #Step 1. split the data into G subsets
  n1 =nrow(x)
  chunk_size1 = ceiling(n1/G)
  indices1= rep(1:G,  each = chunk_size1)
  chunk.x = split(x,indices1)
  chunk.x =lapply(chunk.x, function(chunk) {
    matrix(as.vector(chunk), nrow=chunk_size1, ncol=p, byrow = FALSE)})
  chunk.y = split(y, indices1)

  #Step 2. apply SIR in each subsets (foreach)
  Wg =c();Mg=c();beta.BIG = list()
  results =   foreach(i=1:G, .combine=c) %dopar% {
      beta.BIG = sir(chunk.x[[i]], chunk.y[[i]], h, r, ytype)
      Wg = nrow(chunk.x[[i]])/n1
      Mg = sum(Wg*t(as.matrix(beta.BIG)) %*%
                 as.matrix(beta.BIG))
      list(beta.BIG = beta.BIG, Wg=Wg, Mg= Mg)
  }
  beta.BIG = results[1+3*(0:(G-1))]
  Wg = results[2+3*(0:(G-1))]
  Mg = results[3*(1:G)]

  #Step 3. return the EDR with most proximity
  return(beta.BIG[[which.max(Mg)]])
}
```

sir() function could be changed to do.sir(chunk.x[[i]],chunk.y[[i]],ndim=2,h=h) (Rdimtools package) and will return same results. Instead of using sir, save() or sirii() could be replaced to make BIG-

SAVE, BIG-SIRII.