FastLSU - A more practical approach for the Benjamini-Hochberg FDR controlling procedure for huge-scale testing problems.

Vered Madar*Department of Biostatistics,

University of North Carolina, Chapel Hill, NC 27599, USA,

and

Sandra Batista**Department of Computer Science,

Princeton University, Princeton, NJ 08540, USA

January 22, 2015

Abstract

We address a common problem in large-scale data analysis, and especially the field of genetics, the huge-scale testing problem, where millions to billions of hypotheses are tested together creating a computational challenge to perform multiple hypotheses testing procedures. As a solution we propose an alternative algorithm to the well used Linear Step Up procedure of Benjamini and Hochberg (1995). Our algorithm requires linear time and does not require any p-value ordering. It permits separating huge-scale testing problems arbitrarily into computationally feasible sets or *chunks*. Results from the *chunks* are combined by our algorithm to produce the same results as the controlling procedure on the entire set of tests, thus controlling the global false discovery rate even when p-values are arbitrarily divided. The practical memory usage may also be determined arbitrarily by the size of available memory.

keywords: Adjusted p-values, chunks of p-values, expression quantitative trait loci, false discovery rate, false discovery proportion, family-wise error rate, huge-scale testing problems, large-scale testing problems, linear step-up;

1 The problem of huge-scale data and separating hypotheses tests

In many fields the substantially increased scale of data available has resulted in a significant increase in size of multiple hypotheses testing problems. In genetics in particular, new advances have led to typical GWAS studies consisting of 10^6 SNPs [11] and eQTL studies usually consisting of 10^9 tests [16, 15]. These testing problems are of *huge-scale* as opposed to *large-scale* used by Efron [8] to describe studies consisting hundreds to thousands hypotheses. These huge-scale multiple hypotheses testing problems create numerous computational challenges when many tests, say of the order 10^6, are performed with all of the p-values of more or less equal importance. As a result some simpler testing procedures such as rigid p-value thresholds may be used that sacrifice power and correctness.

Alternatively tests may be separated or chunked into smaller sets or chunks that are more computationally feasible. Efron [9] notes that the problem of separating hypotheses tests has not received great attention and warns of some pitfalls in chunking p-values, but focuses on grouping tests that share a biological property rather than arbitrary, computationally feasible chunks. Cai and Sun [6] and later Benjamini and Bogomolov [5] propose alternative solutions to Efron's grouping problem but do not address the problem of arbitrary, computationally feasible chunking. We confront the computationally feasible chunking problem for the Benjamini-Hochberg false discovery rate [1]. We show on data from Stranger's HapMap study [14, 16] that if results from separate tests are not combined correctly, there is considerable inflation of type I error, offer an explication for this occurrence, and propose our algorithm as a solution.

Consider a huge-scale testing problem of size m where our goal is to select exactly $R \geq 0$ significant tests. Of the R significant discoveries, exactly $V \geq 0$ tests will be false discoveries (i.e., truly non-significant tests that are declared significant). A common approach in multiple comparison testing is to control the family-wise error rate, $FWER = \Pr(V \geq 1)$, the probability to make at least one false discovery. Alternatives are to control the false discovery rate (FDR), the averaged value of the false discovery proportion, $FDP = V/min(R, 1)$, the proportion of truly false tests among the significant R, or the positive FDR, $pFDR = E(V/R|R > 0)$, the average of the FDP when significant tests are selected. For a further discussion about $FWER$, FDR, $pFDR$, and their variations refer to Farcomeni [10].

In a huge-scale testing, when the m p-values are partitioned into chunks, it is challenging to control any of the above error rates ($FWER$, $pFDR$ or FDR) over the entire collection of m p-values. Controlling these error rates on a per chunk basis, if not done correctly, may interfere with the overall results by introducing more false discoveries. Consider for instance an example of $FWER$ control, the Bonferroni approach that for a chunk of size m collects all p-values less than α/m Applying Bonferroni in chunks of size m_i will tend to select more significant results than applying it over the entire set of $m = \sum m_i$ p-values since α/m is less than α/m_i. In the case of $FWER$ control using a fixed bound of α/m for all the chunks is preferred. A stricter constant cut-off on all sets of tests is suggested by Dudbridge and Gusnanto for GWAS [7].

It is preferable to control the false discovery proportion rather than the number of false positives for a huge-scale testing problem. Therefore the FDR or the $pFDR$ approaches are favored and both tend to offer larger, more powerful sets of results than those that might be offered by the conservative $FWER$ control.

1.1 The Benjamini-Hochberg linear step up procedure for controlling the false discovery rate

The Benjamini-Hochberg Linear Step Up(LSU) procedure [1] is designed to control the False Discovery Rate, the expected value of the false discovery proportion, i.e., $FDR = E(FDP)$. The FDR is at most equal to $FWER$ and $FDR = FWER$ when all tests are non-significant in truth. As a result for huge scale multiple hypotheses tests of equal importance, controlling a proportion of false discoveries, especially on the average, has increased power over procedures that control the FWER such as Bonferroni or a rigid cut-off bound such as 5×10^{-8} suggested for GWAS [7]. The larger the multiple hypotheses testing problem is the more powerful the LSU is over procedures that control the $FWER$.

While the LSU procedure [1] is still one of the most cited procedures, its application had required sorting all p-values in decreasing order to look for the largest p-value that satisfies a simple condition. In face of a huge scale testing problem rather than apply LSU, some researchers had preferred to use harsh p-value cut-offs as mentioned above or to divide their huge-scale set of tests into computationally feasible smaller chunks and apply the multiple hypotheses testing procedure on each chunk selecting as the final significant results the union of the results in each of the chunks. Efron [9] warns of the danger in such aggregation from the perspective of $pFDR$, pointing out that some chunks might have a larger proportion of significant results than others, and aggregating the significant results can yield misleading estimates. Moreover different chunking of tests might yield different sets of significant results. Analysis done by different groups of populations or chromosomes may not give the same number of significant tests as analysis that is applied on equal sized subsets; for example, it is well known that chromosome 6 has a higher proportion of significant HLA SNPs than other chromosomes. We shall show that sorting the p-values, arbitrary thresholds, and arbitrary aggregation of results are not necessary and do not improve computational space or time efficiency compared to our algorithm.

1.2 A Faster Algorithm for LSU

Our alternative algorithm to the LSU, **FastLSU**, performs linear scans instead of sorting p-values, but takes into account the overall size of the testing problem. FastLSU tiles the LSU procedure to give one global set of results that does not differ from applying LSU to the entire set of tests. Our algorithm addresses the same objective function as the original. Our approach is provably faster than the conventional approach that relies on sorting p-values. It may also be used on arbitrary chunks of arbitrary size with an arbitrary space constraint in order to return the same set of significant results as those from applying LSU to the entire set of tests.

In the following section we address the difference between grouping and chunking tests and the difficulty in arbitrarily chunking tests by giving examples of inflation of type I error. In section 3 we present FastLSU on a single set of tests and prove its equivalence to LSU, time efficiency, and space efficiency. We present FastLSU on arbitrary chunks and also show its correctness and efficiency in Section 4. We offer suggestions for finalizing the report of significant tests in Section 5 and conclude with discussion in Section 6. Code for an implementation of the algorithm are given in R and SAS in the Appendix A.

2 An exercise of three HapMap groups and their chunking for Stranger's cis-eQTL study

FastLSU controls the global FDR, not a family-based bound or a group-based bound. Our final results are not affected by the procedure of separating tests. This is an important distinction because chunking is arbitrary and based primarily on computational efficiency without any consideration for relationships between the hypotheses being tested. This is in contrast to group-testing procedures where for example, hypotheses may be grouped based on experimental knowledge such as all the tests from the same chromosome, the back and front parts of the brain [9], or different population groups in HapMap [14]. In Section 3.2 we will show how the FastLSU algorithm can even improve the efficiency of a group-based controlling procedure. FastLSU is particularly suited to the current applications in genetics because we typically seek the significant set of SNPs or genes, and the family structure is usually less importance than managing the computational burden. Genetic family structures typically do not require any correction for family selection because each genetic family tends to contribute significant results of its own. (This is the case for the following example of 3 families of HapMap.) For this reason in the remainder of this section, we will give motivating examples of applying FastLSU on the three groups of approximately 14×10^6 cis-eQTLs of Stranger's HapMap study [14, 16] in order to demonstrate the problems with arbitrary chunking without combining the results as FastLSU does.

Stranger [14] presents an eQTL study over 4 HapMap population samples: 30 Central Europeans(CEU), 45 Chinese (CHB), 45 Japanese(JPT) and 30 trios from Nigerians(YRI). To increase power each group is analyzed separately. We follow the recommendations of the SeeQTL website [16] and consider the CHB and JPT together. We define cis-eQTLs as within 1 Mb upstream or downstream of a gene. Each populations has a set of approximately 14 million p-values that were split into chunks in sizes: $1M$, $900K$, $800K$, $700K$, $600K$, $500K$, $250K$, $100K$, $50K$, $25K$ to $10K$ and $5K$. We contrast the differences in the number of significant results when the results are combined by taking the union of the results of LSU on each chunk versus those selected by FastLSU Algorithm 2 in Figure 1.

Applying the FastLSU yields $9,228$ significant results for CEU, $6,497$ for YRI and $33,507$ for the CHB & JPT group. Most notable is that the results for FastLSU Algorithm 2 do not change across the chunk sizes whereas the alternative of taking the union of the results on each chunk increases not only the number of significant tests selected, but also the maximal p-value reported as the chunk size decreases. For example, applying FastLSU at level 10% for the complete chunk of approximately $14M$ p-values yields $9,228$ discoveries for the CEU; however, applying LSU on $1,370$ chunks of $10K$ yields $16,925$ significances which is equivalent to an overall FDR control of 23.75%. If we assess the magnitude of the significance level required to get the same number of significances for $10K$ sized chunks in the above exercise, we see that the CHB-JPT group $41,587$ requires $\alpha = 15.58\%$ and for the YRI group $\alpha = 21.1\%$ will give $10,330$ significant p-values. This implies that performing the analysis using $1,300$ to $1,400$ chunks of size 10K rather than a single chunk inflates the type I error by 50%. While this is compelling experimental evidence, a more compelling explanation is that the union of partial orderings on subsets of a set is in general not equal to the partial ordering on the entire set. Moreover when the size of the chunks decreases, the significance interval is being divided into larger sub-intervals more likely each containing more p-values spanning a greater range of values, so that the set of selected significant p-values will more likely contain a greater maximal p-value. The max p-value we obtain for the case of single chunk is 6.7×10^{-5} for CEU, 4.4×10^{-5} for YRI, and 2.5×10^{-4} for CHB-JPT. For chunks of the size $100K$ (about 13 to 14 chunks), the max p-values are slightly higher especially for the CHB-JPT group: 1.7×10^{-4} for CEU, 3.5×10^{-4} for YRI and 1.2×10^{-3} for CHB-JPT. For chunks of $10K$ size (about 130 to 140 chunks) the maximal p-values are considerably higher still: 0.0063 for CEU, 0.0034 for YRI and 0.011 for CHB-JPT. As we mentioned, there is no need to apply any correction for group of family selection as required by Benjamini and Bogomolov [5] since all three groups in the HapMap example give significant results.

3 The FastLSU algorithm

The usual way to apply the LSU [1] at level $0 \leq \alpha \leq 1$ is to sort the p-values in ascending order, $p_{(m)} \geq p_{(m-1)} \geq \cdots \geq p_{(1)}$. Starting from the largest p-value to the smaller we need to look for the largest p-value that satisfy $p_{(k)} < k\alpha/m$. One may view the LSU algorithm [1] as a search algorithm for

$$r = argmax\{i : \ p_{(i)} < i\alpha/m\}. \tag{1}$$

This observation motivates the following Fast Linear Step UP (FastLSU) algorithm that controls the FDR at level α, for $0 < \alpha < 1$:

Algorithm 1 ***FastLSU for a single batch - without sorting the p-values***

```
Step 1. Start with $r_0 = m$ and count all p-values $< r_0\alpha/m$; let $r_1$ be their count.
Step k. Do for $k = 2, \ldots, m$ Count all p-values $< r_{k-1}\alpha/m$; let $r_k$ be their count.
   End. Repeat step k for $k = k + 1$ until $r_k = r_{k+1}$ or $k = m$.
```

3.1 The equivalence to LSU and computational efficiency

Theorem 1 *For a significance level α, the Algorithm 1 maximizes the same objective function 1 that is used by the LSU. Therefore, the FastLSU controls the FDR at level α and gives the same selected set of significant results as would be obtained by applying the Benjamini-Hochberg [1] LSU FDR controlling procedure.*

Proof. Consider a batch of p-values which we will denote as $\mathcal{C}$. Let $0 \leq t \leq 1$ and define $\mathcal{S}(t : \mathcal{C}) = \{p < t \, : \, p \text{ is p-value} \in \mathcal{C}\}$ to be the number of p-values from $\mathcal{C}$ smaller than t. The set $\mathcal{S}\left(\frac{i\alpha}{m}\right)$ consists of all p-values from $\mathcal{C}$ that are smaller than $\frac{i\alpha}{m}$. Hence, for the case of a single batch, it is possible to verify that the search for

$$r = argmax\{i : \; i = \#\mathcal{S}\left(i\alpha/m\right)\} \tag{2}$$

is equivalent to looking for the largest rth p-value satisfying $p_{(r)} \leq \frac{r\alpha}{m}$, as requested by (1).

¶

Theorem 2 *The FastLSU Algorithm 1 requires $O(m)$ time and $O(m)$ space where m is the number of p-values to be considered.*

The proof to Theorem 2 is given in the Appendix A with accompanying pseudocode for procedural language implementations.

Example 1 *To demonstrate this equivalence we apply FastLSU on the example appearing in the original 1995 FDR paper [1] with $\alpha = 0.05$. Instead of sorting all 15 p-values we apply Algorithm 1 using 5 linear scans and find 4 significant p-values. Consider the 15 p-values from the example given at LSU paper [1]:*

$$0.6528, 0.7590, 0.0298, 0.4262, 0.0459, 0.0278, 0.0001, 0.0019,$$

$$0.0004, 0.0201, 1.0000, 0.5719, 0.3240, 0.0095, 0.0344$$

Scanning the p-values, for the first time, we find 9 p-values < 0.05, the second $r_2 = 7$ p-values $< 9 \cdot 0.05/15$, and the third $r_3 = 5$ p-values $< 7 \cdot 0.05/15$. The fifth scan finds exactly $r_4 = 4$ p-values that satisfy $< 4 \cdot 0.05/15$. The selected p-values are $0.0001, 0.0019, 0.0004, 0.0095$, and the reader can check that these 4 p-values are exactly the ones selected using the original LSU p-value sorting algorithm.

Example 2 (Example of huge-scale run) *We apply the Algorithm 1 over the group of* $13,755,172$ *Central Europeans(CEU) HapMap p-values of Stranger [14] that was described in Section . For this example, the FastLSU makes* 11 *linear scans to achieve the* $9,228$ *significant p-values, and the running time is of several seconds.*

3.2 Applying FastLSU over families or groups of p-values of equal relevance

Efron [9] distinguishes between the groups of imaging p-values that arise from the front brain and the back brain, and develops an empirical Bayes set-up to combine the significant p-values from the two families. Benjamini and Bogomolov [5] extended Efron's idea of groups into the voxel families of MRI where each set of p-values from a specific voxel of a certain location are treated as a separate, homogeneous group of relevance. For the first step each voxel group is analyzed by applying the LSU at level α. For the second step the number of significant groups is collected and LSU is performed again with another significance level α^* to correct for the selection of groups. If, for example, S groups out of G are shown to have at least one significant p-value for LSU of level α, α^* is set to $\frac{S}{G}\alpha$ for each of the groups [5].

Given the equivalence between LSU and FastLSU, FastLSU may be used for each step in the approach of Benjamini and Bogomolov. The second step need only be done for groups that have at least one significant result. Since only significant p-values for each group need to be considered, FastLSU may be applied to each group by using $\alpha^{**} = \frac{Sr_g(\alpha)}{Gm_g}\alpha$ where each group g ($g = 1, 2, \ldots, G$) contains m_g p-values of which $r_g(\alpha)$ were declared significant of level α. Since FastLSU achieves efficiency by the simple recalling and correction of the constants m (or m_g) for the number of tests, a similar approach also works for Storey-Efron's positive FDR approach [8, 13]. To be more precise, the complete grid of λ values should be collected in each step and then the q-values should be corrected accordingly.

4 Applying LSU on arbitrary chunks of p-values

Algorithm 1 may be extended for when p-values are divided into an arbitrary number of chunks of arbitrary size. The size of the chunks may be determined by practical memory constraints or any other criteria. The iterative steps are applied on each chunk. The significant results from each chunk are combined using another iterative step to form the final set of significant results.

Algorithm 2 ***FastLSU for the case of two or more chunks of p-values*** *Given a set of m tests and p-values divided into n separate chunks of sizes m_{c_i} p-values for $i = 1, ..., n$ such that $m = \sum_{i=1}^{n} m_{c_i}$:*

- ***Step*** 1***.*** *On each of the chunks, count the number of p-values less than α. Denote this count as $r_{c_i}^{(1)}$.*
- ***Step*** $k+1$***.*** *On each of the chunks (starting from $k = 1$) count the number of p-values less than $(\sum_{i=1}^{n} r_{c_i}^{(k)})\alpha/m$, and let $r_{c_i}^{(k+1)}$ be the count for the chunk at this step.*
- ***Repeat Step*** $k+1$ *until $k+1 = m$ or $\sum_{i=1}^{n} r_{c_i}^{(k+1)} = \sum_{i=1}^{n} r_{c_i}^{(k)}$. Mark these p-values as significant under LSU.*

An example of Algorithm 2 applied to the example appearing in the original 1995 FDR paper [1] is given in the Appendix.

There are many ways one can alter Algorithm 2, either by flagging or dropping p-values that do not satisfy the condition in the iterative step that preserves the LSU. Alternatively Algorithm 1 may be applied to each chunk independently starting with the initial value for $r_0 = m - m_{c_i} + r_{c_1}^{(1)}$ and still tiling by the total number of tests, m. (The advantage of this is that this may be done in parallel if desired.) The results of the chunks may then be combined into a single set, if space permits, and Algorithm 1 applied to this set to return the final set of significant results. If space does not permit the results of the chunks to be combined, they may be combined up to the space limit. Then the iterative step of Algorithm 2 above may be applied to the resultant chunks starting by checking for p-values less than $m'\alpha/m$ where m' is the size of the union of the results.

Theorem 3 *For a significance level α, and a collection of c chunks the Algorithm 2 gives the same selected set of significant results as would be obtained by applying the Benjamini-Hochberg [1] LSU FDR controlling procedure over the single chunk consisting the union of all p-values from the c chunks.*

Proof. [Proof for Algorithm 2] Suppose we have exactly c disjoint chunks of p-values $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_c$, with sizes $m_{\mathcal{C}_i}$ $(i = 1, 2, \ldots, c)$ such that $\mathcal{C} = \cup_{i=1}^{c} \mathcal{C}_i$ and $|\mathcal{C}| = m = \sum_{i=1}^{c} m_{\mathcal{C}_i}$ p-values. Let $r_{\mathcal{C}}$ be the number of selected significant p-values that satisfying the LSU objective (1):

$$r_{\mathcal{C}} = argmax \left\{ r : \ r = \#\mathcal{S}\left(\frac{r\alpha}{m} \ : \mathcal{C} \right) \right\}. \tag{3}$$

Algorithm 2 can be applied to each c chunks and the last step of Algorithm 2 is to finalize the selection. Accordingly for chunk $\mathcal{C}_i$ we search for the largest $r^*_{\mathcal{C}_i}$ that satisfy the objective:

$$r^*_{\mathcal{C}_i} = argmax \left\{ r : \ r = \#\mathcal{S}\left(\frac{(r + m - m_{\mathcal{C}_i})\alpha}{m} \ : \mathcal{C}_i \right) \right\}. \tag{4}$$

We claim that

$$r_{\mathcal{C}} \leq r^*_{\mathcal{C}_i} + m - m_{\mathcal{C}_i}, \quad \text{for any } i = 1, \ldots, c \tag{5}$$

This implies that each of the p-values selected significant from the search for (3) within chunk $\mathcal{C}_i$ must be selected while searching for argmax $r^*_{\mathcal{C}_i}$ as in (4) since $p \leq r_{\mathcal{C}} \cdot \alpha/m$ implies $p \leq (r^*_{\mathcal{C}_i} + m - m_{\mathcal{C}_i})\alpha/m$.

It is possible to verify that $p_{(k:\mathcal{C}_i)}$ the kth p-value in chunk $\mathcal{C}_i$ cannot be larger than, $p_{(k+m-m_{\mathcal{C}_i})}$, the $(k + m - m_{\mathcal{C}_i})$th p-value in the union $\mathcal{C}$:

$$p_{(k:\mathcal{C}_i)} \leq p_{(k+m-m_{\mathcal{C}_i})} \qquad \text{for all } k = 1, \ldots, m_{\mathcal{C}_i}. \tag{6}$$

In particular, by the condition (4), let $p_{(r^*_{\mathcal{C}_i})}$ be the largest p-value in $\mathcal{C}_i$ that satisfies $p_{(r^*_{\mathcal{C}_i}:\mathcal{C}_i)} < (r^*_{\mathcal{C}_i} + m - m_{\mathcal{C}_i})\alpha/m$. When $m_{\mathcal{C}_i} > r^*_{\mathcal{C}_i}$, it follows that

$$(r^*_{\mathcal{C}_i} + k + m - m_{\mathcal{C}_i})\alpha/m < p_{(r^*_{\mathcal{C}_i}+k:\mathcal{C}_i)} \leq p_{(r^*_{\mathcal{C}_i}+k+m-m_{\mathcal{C}_i})} \qquad \text{for all } k = 1, \ldots, m_{\mathcal{C}_i} - r^*_{\mathcal{C}_i}. \quad (7)$$

(When $m_{\mathcal{C}_i} = r^*_{\mathcal{C}_i}$ (5) holds trivially since at most the entire set of tests may be selected as significant.) To prove (5) by contradiction, let us compare between $p_{(r_\mathcal{C})}$ to $p_{(r^*_{\mathcal{C}_i}+k+m-m_{\mathcal{C}_i})}$. If we assume that $r_\mathcal{C} > r^*_{\mathcal{C}_i} + m - m_{\mathcal{C}_i}$, then we can define a positive integer $k = r_\mathcal{C} - (r^*_{\mathcal{C}_i} + m - m_{\mathcal{C}_i})$ for which following the inequality in (7) holds,

$$r_\mathcal{C} \cdot \alpha/m = (r^*_{\mathcal{C}_i} + k + m - m_{\mathcal{C}_i})\alpha/m < p_{(r^*_{\mathcal{C}_i}+k:\,\mathcal{C}_i)} \leq p_{(r^*_{\mathcal{C}_i}+k+m-m_{\mathcal{C}_i})} = p_{(r_\mathcal{C})},$$

However, this is in contradiction to the condition (3) that provides $p_{(r_\mathcal{C})} < r_\mathcal{C} \cdot \alpha/m$.

In conclusion we show that applying FastLSU over the global union of p-values give the exact same selection of significant p-values. We search for $r^*_\mathcal{C}$ over the union of the results on each chunk of size $\sum_{i=1}^{c} r^*_{\mathcal{C}_i}$

$$r^*_\mathcal{C} = \#\mathcal{S}\left(\frac{r^*_\mathcal{C} \cdot \alpha}{m} : \mathcal{S}(r^*_{\mathcal{C}_1} : \mathcal{C}_1) \cup \cdots \cup \mathcal{S}(r^*_{\mathcal{C}_c} : \mathcal{C}_c)\right).$$

Since the inequality (5) ensures that p-values that were not selected under the condition (4) would have not been selected under the condition (3), we conclude that $r^*_\mathcal{C} = r_\mathcal{C}$. ¶

Theorem 4 *The FastLSU Algorithm 2 requires $O(m)$ time and $O(m)$ space where m is the number of p-values to be considered. The practical memory usage may be restricted to an arbitrary limit for the largest chunk size, $m*$.*

Proof for the running time and space limitations are shown in the Appendix A.

5 Useful tips for finalizing the report of significant p-values

If we assume that the R tests were declared significant by applying either the LSU or FastLSU, we offer tips for finalizing the set of significant p-values. First it is possible to put a bound over the expected value of significant results when applying LSU. Suppose the set of p-values of interest consists of m_0 true null and $m - m_0$, then $E(R) \leq \alpha m_0 + m_1$. In the remainder of this section will explain how to protect the FDR control of FastLSU against dependency structures and when such correction is actually needed. We will also explain how to compute q-values (adjusted p-values) for the final results without keeping the entire set of p-values and show how to add a set of simultaneous confidence intervals for the significant test statistics while accounting for the selection effect.

5.1 Correcting against general case of dependence

The LSU procedure is conservative under the general type of positive regression on subsets(PRDS) [2], so applying the LSU at significance level of α always ensures $FDR \leq \alpha$ for PRDS p-values. The PRDS class contains a larger set of structures, among them, the independent case and any positively associated p-values such as p-values obtained from a two-sided t tests. Since the LSU and FastLSU are equivalent, applying the FastLSU at level α on PRDS p-values will control the FDR at level-α, as well.

For other types of non PRDS dependency, such as in the case of pairwise comparisons, it is recommended to use the Benjamini-Yekutieli procedure [2] that applies LSU using $\alpha^* = \frac{\alpha}{c}$ instead of α where $c = \sum_{k=1}^{m} \frac{1}{k} \approx \ln m + 0.5772$, where 0.5772 is the *Euler-Mascheroni constant*. By the same argument the FastLSU under non PRDS will have $FDR \leq \alpha$ when applied under $\alpha^* = \frac{\alpha}{\ln m + 0.5772}$. As a matter of practice, we suggest to first perform FastLSU using the significance level α. Then, if the p-values are non PRDS, correct the R selected p-values by applying FastLSU again over the single batch consisting of $m^* = R$ p-values using $\alpha^{**} = \frac{R\alpha^*}{m}$. This approach provides $FDR \leq \alpha$ since $\alpha^* < \alpha$ for $m \geq 2$.

5.2 How to compute q-values to a selected subset of significant tests

When it is preferable to report q-values or adjusted p-values, we suggest how this may be done more efficiently. If we assume that R tests were selected as significant, let the p-value $p_{(R)}$ be the largest p-value satisfying $p_{(R)}m/R < \alpha$. All p-values larger than $p_{(R)}$ were not selected as significant since $p_{(i)}m/i > \alpha$ for $i > R$ and have q-values $> \alpha$. Therefore it is sufficient to consider only the set of selected R p-values. The q-value for the p-value, $p_{(i)}$, under the LSU has the form [17]

$$q_{(R)} = p_{(R)}m/R \quad \text{ and } q_{(i)} = \min_{j=R,R-1\ldots,i+1}\{p_{(i)}m/i,\, q_{(j)}\}, \quad \text{ for } i < R. \tag{8}$$

From this we can see that the algorithms presented by Yekutieli and Benjamini [17] and Storey [13] are $O(R \log R)$. One needs only sort the R selected p-values in descending order and then beginning from largest p-value assign the corresponding q-value in a final linear scan recording the minimum q-value assigned thus far. The q-values will also be descending assigned in this way and there is no need to compare previous values except the minimum q-value thus far. The value for the q-value for $q_{(i)}$ will only change when it is less than the minimum seen thus far and the minimum q-values will span a range of p-values until either a sufficient number of p-values have been covered or a sufficient range in p-values have been covered. Alternatively, one can use existing procedures(such as p.adjust in R software or proc multitest in SAS) to compute adjusted p-values for the set of R selected p-values and multiply the results by R/m to correct them.

5.3 Confidence intervals for selected subset of significant results.

A less common approach in genomic studies is to report the significant test results by constructing a set of confidence intervals for the tests statistics. While p-value is a merely measure of the magnitude of the test statistic, a confidence interval may offer the additional information about the dispersion of that magnitude. The selection adjusted confidence intervals of Benjamini and Yekutieli [3] offer an appropriate construction that corrects against the false coverage effect of selection. For a useful example see Jung *et al* [12] usage for the significant log-fold changes of RNA Microarrays.

6 Discussion

We presented an efficient algorithm to apply correctly the Benjamini-Hochberg Linear Step Up FDR controlling procedure in a huge-scale testing problem. Since we have shown that our algorithm requires only linear time, we can claim that is provably not any more computationally burdensome than even using a rigid Bonferroni cut-off for control. However, unlike the rigid Bonferroni cut-off, our approach ensures the FDR control at level α and this is a more powerful alternative to controlling the $FWER$, especially when the multiple testing problem is of huge-scale. Our approach is also scalable to any subsetting or chunking of the overall subset of p-values.

In addition we offered tips for computational efficiency while performing the LSU over a huge-scale multiple hypotheses testing problem, such as showing how to correct for dependency or how to compute q-values directly from the subset of LSU significant results. We hope that these algorithms and tips offer a better insight into the huge-scale testing problem rather just a black-box of solutions. We encourage altering the steps in performing Algorithm 2, for instance, either by applying it sequentially or in parallel or a mixture of both.

We were, indeed, surprised by the amount of inflated type I error we observed during the exercise on different chunk sizes of the HapMap populations. This strongly suggests the need for greater diligence in correctly separating hypotheses tests, whether the objective is to control the $FWER$, the FDR, or even Efron-Storey's $pFDR$. This observations was also reported by Efron [9], but a full investigation on real data with decreasing chunk sizes was not performed. Our suggested approach solves this for the case of the Benjamini-Hochberg FDR. As we have alluded a similar approach can be adopted to control correctly the Efron-Storey's $pFDR$ and this remains open for further research. In addition the severe phenomena of inflated type I error we observed suggests more care may be required in reporting and interpreting results in genetics literature especially in the case of GWAS and eQTL studies. Namely to properly understand the significance results there is a need for consistent consideration of the algorithms or software used for controlling and separating the hypotheses tests and for record keeping of the chunk sizes used for a study.

A Proofs and codes

A.1 Proof of Theorem 2

Proof. [Proof of Theorem 2] We present the algorithm here in three steps with running time for each step and accompanying pseudocode.

Bin. Classify all p-values into the bins of the interval, $[0, \alpha]$ each of size $\frac{1}{m}$ and keep a total of all p-values with value less than α, m^*. Label each p-value with a value k such that $k = \lceil \frac{mp^*}{\alpha} \rceil$ or $k = k + 1$ if $\frac{mp^*}{\alpha}$ equals $\lceil \frac{mp^*}{\alpha} \rceil$ where α is the significance level and p^* is the given p-value being labeled. If p^* is labeled with bin k less than or equal to

m, increment the count for bin k and increment current p-value count, m^*. If a p^* has label k greater than m, it may be filtered, so no counts need to be incremented for such p-values (although they can be labeled with arbitrarily large values for k). *Running time and memory:* Labeling each p-value and incrementing the labeled bin count requires only constant time and a single pass through the p-values. In addition to storing the p-values, the p-value labels and bin counts also need to be stored, also requiring $O(m)$ space each, and a variable for the current p-value count, m^*.

Accumulate. In this step we find the r^* the significant bin to return all p-values in bins less than or equal to this bin as significant. To do so, starting from the highest labeled bin's count, i.e,. for m, keep a partial sum of the total number of p-values in the bins thus far. If the current bin's has a non-zero bin count and its value is equal to m^* minus then the current partial sum, then return the current bin as r^* the significant bin. *Running time and memory*: This step can be done in a single scan of the bin counts and only requires additional variables for the significant bin, r^*, and the partial sum.

Return. Return as significant all the p-values that were labeled with a k less than or equal to r^*. *Running time and memory* This requires only a single scan of the p-value labels and no additional space.

¶

A.2 Pseudocode for Proof of Theorem 2

```
Let p_vals[m] be an array of p-values,
p_val_labels[m] be array of bin labels for p-values,
count_bins[min] be an array of bin counts,
m the total number of p-values, and alpha the significance level.
Initialize p_val_labels and count_bins to 0.
```

```
sig_pvals = 0; #number of p-vals with small enough significance
for i = 1 to m {
   p* = p_vals[i];
   k = ceil(mp*/alpha); #Bin
   if (  ceil(mp*/alpha) equals mp*/alpha) increment k;
   p_val_labels[i] = k;
   if (k less than or equal to m) {
     increment count_bins[k];
     increment sig_pvals;
   }
}
r* = 0; #significant bin
total_p_vals = 0; # partial sum of p_values
for i = m to 1 {
  add count_bins[i] to total_p_vals]; #Accumulate
  if (sig_pvals - total_p_vals equals i) {
       r* = i and break;
  }
}
for i = 1 to m {
     if (p_val_labels[i] less than or equal to r*) #Return
        return p_vals[i] as signficant;
}
```

A.3 Example for Algorithm 2

Example 3 *To demonstrate the Algorithm 2 consider the* 15 *p-values appearing in Example 1 and also in the original LSU paper [1]. Further assume that for some reason the* 15 *p-values are divided into two chunks. The first chunk, say* $\mathcal{C}_1$*, consists of the first* 8 *p-values:*

$$\mathcal{C}_1 = \{0.6528, 0.7590, 0.0298, 0.4262, 0.0459, 0.0278, 0.0001, 0.0019\},$$

and the second chunk, $\mathcal{C}_2$

$$\mathcal{C}_2 = \{0.0004, 0.0201, 1.0000, 0.5719, 0.3240, 0.0095, 0.0344\}.$$

Algorithm 2 applied for $\alpha = 0.05$ *has three steps: In Step 1, we scan the* 8 *p-values in* $\mathcal{C}_1$ *and seek for the largest p-value,* $p_{(i:\mathcal{C}_1)}$ *satisfying* $< (i+15-8)\alpha/15$. *This can be done in a similar manner to applying Algorithm 1. First scan gives* 5 *p-values* $< (8+15-8)0.05/15 = 0.05$. *Second scan gives* 4 *p-values* $< (5+15-8)0.05/15 = 0.04$ *and stops with the* 4 *p-values* $< (4+15-8)0.05/15 = 0.0367$. *The selected p-values for the first step are*

$$\mathcal{S}(0.0367 : \mathcal{C}_1) = \{0.0298, 0.0278, 0.0001, 0.0019\},$$

In Step 2, we scan the 7 *p-values in* $\mathcal{C}_2$ *and seek for the largest* $p_{(i:\mathcal{C}_2)} < (i+15-7)0.05/15$. *First scan gives* 4 *p-values* $< (7+15-7)0,05/15 = 0.05$ *and immediately stops with these* 4 *p-values* $< (4+15-7)0.05/15 = 0.04$. *The selected p-values are*

$$\mathcal{S}(0.04 : \mathcal{C}_2) = \{0.0004, 0.0201, 0.0095, 0.0344\}.$$

Next, we follow the last step of Algorithm 2 which is a combination step and applies on the collection of 8 *p-values selected at the former steps:*

$$\mathcal{S}(0.0367 : \mathcal{C}_1) \cup \mathcal{S}(0.04 : \mathcal{C}_2) = \{0.0298, 0.0278, 0.0001, 0.0019, 0.0004, 0.0201, 0.0095, 0.0344\}.$$

All 8 *selected p-values are clearly* < 0.05. *Second,* 5 *p-values*$< 8 \cdot 0.05/15 = 0.0267$,*and third scan finds out* 4 *p-values* $< 5 \cdot 0.05/15 = 0.0167$ *that also satisfy* $< 4 \cdot 0.05/15$. *The selected p-values are, again,* $0.0001, 0.0019, 0.0004, 0.0095$.

A.4 Proof of Theorem 4

Proof. [Proof of Theorem 4.] For m p-values arbitrarily divided into n chunks of size m_c for $c = 1, ..., n$ such that the maximum chunk size is m^*, we show that the algorithm is still linear in m and never uses more than m^* space.

Bin. This step is as for Algorithm 1. It is important to note that the binning is done relative to m and not the size of the chunk. The only important difference is that bin sums are not maintained because of the m^* space limitation. Labels need not be stored either. This step also counts the sum, m', of all p-values across all groups that are less than α. *Running time and memory:* This requires a linear scan. A count variable can be kept for each group in order to get m'.

Accumulate. For each group, find the bin sums for the largest m^* partition of bins not covered yet, i.e., find bin sums for bins $m' - jm^* + 1$ to $m - (j-1)m^*$ for $j = 1, ..., n$ and we do both of the following before incrementing j. Accumulate bin sums across the chunks. (This can be done in a linear scan of the chunks and a single array accumulating bin sums across chunks.) Finding $r*$ is as in Algorithm 1 Step 2. However, now only m^* bins may be checked at a pass before needed to increment j. The subtotal of p-values counted thus far is maintained after j is incremented. *Running time and memory*: This step must be repeated at most n times and requires a linear scan of the data. At any point at most m^* space plus several count variables are used.

Return. This step is as in Algorithm 1.

¶

A.5 R script for the FastLSU with examples

```
fast.LSU = function(alpha=0.1,pvls,mtc,mtg){
# alpha: the desired significance level;
# mtg: the global number of p-values in the problem (the sum of all p-values in all chunks);
# mcg: total number of p-values in the chunk (<=mtg);
# pvls: pvalues;
# ------------------------------------------------------------------------------------;
  m = length(pvls); #no of actual p-values at the input chunk (might be < mtc, e.g., step c+1 in algorithm 2);
  print(paste('FastLSU for group of',mtg,'p-values.'))
  if(m<mtc){
    print(paste(m,'p-values were detected at the input chunk. Original chunk size is assumed to be',mtc));
    print('Make sure your chunk size is correct.');
    print('Input chunk size can smaller than the assumed chunk size at the final step c+1 of Algorithm 2.');
  }
  if(mtc<mtg){print(paste('Input chunk size is',mtc,'. Group size is', mtg,'p-values.'))}
  sig    = rep(1,m); #flag, for significance, 1 is default;
  r.new  = m;
  r.old  = m-1;
  diff.r = 1;
  stp = 0; #follow steps;
  while(diff.r>0){
       stp = stp+1
       sig[pvls>=(r.new+mtg-mtc)*alpha/mtg] = 0; #set the sig flag of large enough p-values to zero;
       r.old = r.new;
       r.new = sum(sig);
      print(paste('On step',stp,':',sum(sig),'p-values were selected for candidance of significant.'))
     diff.r = r.old - r.new
 }
 return(list(pvalues.c=pvls[sig==1],r=r.new,sig=sig))
}

#simulate p-values for example;
pvals.sim = runif(30000); # random U[0,1] (null) p-values;
Bsig = 1-rbinom(30000,1, .02); Bsig[Bsig==0] = .0001;
pvals.sim = pvals.sim*Bsig; #add 2% of small p-values to be significant;

# Example of a single chunk;
results.allchunks = fast.LSU(alpha=0.1,pvls=pvals.sim,mtc=length(pvals.sim),mtg=length(pvals.sim));

# Example of 2 chunks of p-values (1st of 10,000, 2nd of 20,000);
# chunking;
pv.chunk1 = pvals.sim[1:10000];
pv.chunk2 = pvals.sim[10001:30000];

# Step 1:
results.chunk1 = fast.LSU(alpha=0.1,pvls=pv.chunk1,mtc=length(pv.chunk1),mtg=length(pvals.sim));

# Step 2:
results.chunk2 = fast.LSU(alpha=0.1,pvls=pv.chunk2,mtc=length(pv.chunk2),mtg=length(pvals.sim));
```

```
# Step 3: # get final candidancy;

cand.pvals = t(cbind(t(results.chunk1$pvalues.c),t(results.chunk2$pvalues.c)));

# keep mtc equal to mtg, the fast.LSU function already considers length(cand.pvals).

results.final = fast.LSU(alpha=0.1,pvls=cand.pvals,mtc=length(pvals.sim),mtg=length(pvals.sim));
```

A.6 SAS Macro

```
%macro calc_T_BH(alpha=0.1,pvaldata,pvals,mtg,mtc);

* pvaldata: the name of SAS data set that contains the p-values;

* pvalue: the SAS variable name for the p-values at the SAS data set;

* alpha: the desired significance level;

* mtg: the global number of p-values in the problem;

* mcg: the total number of p-values in the chunk;


data ctemp; set &pvaldata.;

* m is the actual number of p-values at the input chunk (m <= mtc);

* For example: see Step c+1 at Algorithm 2 (m is the number of p-values canditates);

call symput("m",_N_); run;


*set starting values;

%let rtcest = &m.;

%let rold   = %eval(&rtcest.+1);

%let k      = 0;

data ctemp; set ctemp; if &pvalue. < &alpha.; run;


%do %while(&rtcest.< &rold.);

    %let k    = %eval(&k.+1);

    %let rold = &rtcest.;

    *apply the steps of fastLSU;

   data ctemp; set ctemp;

      if &pvalue. < (&rtcest.+ &mtg.-&mtc.) / &mtg. * &alpha.;

   run;


* recompute mtc estimate for fast BH;

data _NULL_; set ctemp; call symput("rtcest",_N_); run;

%let rtcest=&rtcest.;

%end; *end while loop;
```

```
data sig; set ctemp; run;
%mend;


%calc_T_BH(alpha=0.1,pvaldata=StrangerCEU,pvalue=pvalue,mtg=13755172,mcg=13755172);
```

References

[1] Benjamini, Y., and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *J. Roy. Statist. Soc. Ser. B.* **57**, 289300.

[2] Benjamini, Y., and Yekutieli, D. (2001). The control of the false discovery rate in multiple testing under dependency. *Ann. Statist.* **29**, 11651188.

[3] Benjamini, Y., and Yekutieli, D. (2005). False Discovery RateAdjusted Multiple Confidence Intervals for Selected Parameters. *J. Amer. Statist. Assoc.* **100**, 7193.

[4] Benjamini, Y. and Hochberg, Y. (2000). The adaptive control of the false discovery rate in multiple hypotheses testing. *J. Behav. Educ. Statist.* **25**, 60-83.

[5] Benjamini, Y., and Bogomolov, M. (2014). Selective inference on multiple families of hypotheses. *J. Roy. Statist. Soc. Ser. B.*, **76**, 297-318

[6] Cai, T.T., and Sun, W. (2009). Simultaneous Testing of Grouped Hypotheses: Finding Needles in Multiple Haystacks. *J. Amer. Statist. Assoc.* **104**, 1467-1481.

[7] Dudbridge, F., and Gusnanto, A. (2008). P-value less than say 5×10^{-8} can be regarded as convincingly significant. *Genetic Epidemiology* **32**, 227234.

[8] Efron, B. (2004). Large-scale simultaneous hypothesis testing: the choice of a null hypothesis. *J. Amer. Statist. Assoc.* **99**, 96-104.

[9] Efron, B. (2008). Simultaneous inference: When should hypothesis testing problems be combined? *Ann. Appl. Stat.* **2**, 197-223.

[10] Farcomeni, A. Multiple Testing Procedures under Dependence, with Applications. VDM Verlag, 2009

[11] Hindorff, L.A., MacArthur, J., Morales, J., Junkins, H.A., Hall, P.N., Klemm, A.K., and Manolio, T.A. *A Catalog of Published Genome-Wide Association Studies.* Available at: www.genome.gov/gwastudies. Accessed January 11, 2015.

[12] Jung K., Friede, T., and Beissbarth, T. (2011). Reporting FDR analogous confidence intervals for the log fold change of differentially expressed genes. *BMC Bioinformatics* **12**, 288.

[13] Storey (2002). A direct approach to false discovery rates. *J. Roy. Statist. Soc. Ser. B.* **64**, 479-498.

[14] Stranger, B.E., Nica, A.C., Forrest, M.S., Dimas, A., Bird, C.P., Beazley, C., Ingle. C.E., Dunning, M., Flicek, P., Montgomery, S., Tavaré, S., Deloukas, P. and Dermitzakis, E.T. (2007). Population genomics of human gene expression. *Nature Genetics* **39**,1217-1224.

[15] Wright, F.A, Sullivan, P.F., Brooks, A.I., Zou, F., Sun, W., Xia, K., Madar, V., Jansen, R., Chung, W., Zhou, Y.H., Abdellaoui, A., Batista, S., Butler, C., Chen, G., Chen, T.H., D'Ambrosio, D., Gallins, P., Ha, M.J., Hottenga, J.J., Huang, S., Kattenberg, M., Kochar, J., Middeldorp, C.M., Qu, A., Shabalin, A., Tischfield, J., Todd, L., Tzeng, J.Y., van Grootheest, G., Vink, J.M., Wang, Q., Wang, W., Wang, W., Willemsen, G., Smit, J.H., de Geus, E,J., Yin, Z., Penninx, B.W., Boomsma, D.I. (2014). Heritability and genomics of gene expression in peripheral blood. *Nature Genetics.* **46**, 430-437.

[16] Xia, K., Shabalin, A., et. al. (2012). seeQTL: A searchable database for human eQTLs. *Bioinformatics* **28**, 451-452.

[17] Yekutieli, D., and Benjamini, Y. (1999). Resampling-based false discovery rate controlling multiple test procedures for correlated test statistics. *Journal of Statistical Planning and Inference*, **82**, 171-196.

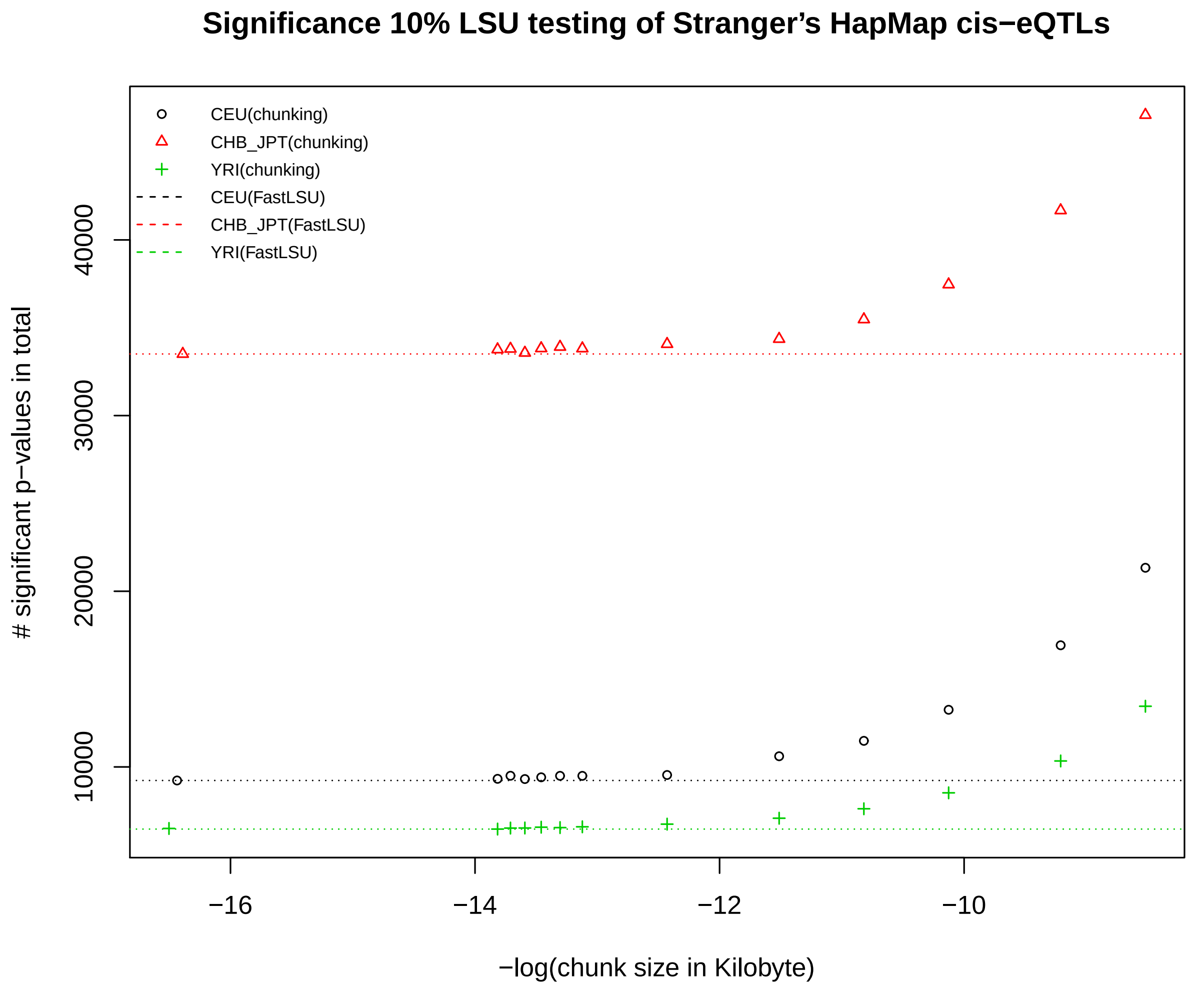

Figure 1: Varying chunk size over Stranger's HapMap populations. Solid lines represent results of applying the LSU under 10% Dashed lines show the consistent result of applying the FastLSU as in Algorithm 2.