# ELL Software Documentation

Version 1.0
June 18, 2021

Takintayo Akinbiyi[1], Mary Sara McPeek[2]

Department of Biostatistics[1]
Yale University, New Haven CT 06510
Departments of Statistics and Human Genetics[2]
The University of Chicago, Chicago, IL 60637, USA

ELL
A Python program for global testing with application to Trans eQTL detection

Homepage: http://www.stat.uchicago.edu/~mcpeek/software/index.html
Release 1.0 January 15, 2021



We request that use of this software be cited in publications as follows:
Akinbiyi T., McPeek M. S. (2021).

# Contents

# 1 Overview of ELL

ELL.py contains python code to compute the ELL statistic ([1]). ELL.py is a python file that is run by command line with a configuration file, whose location is specified on the command line. The following sections detail the usage of the command line utility. First, however, we give a brief overview of the theory. For a full treatment see ([1]).

## 1.1 The ELL statistic

Suppose we observe $Z$-scores $Z_{r,d},\ r = 1, ..., R,\ d = 1, ..., D$ with corresponding p-values $\pi_{r,d} = 2\Phi(-|Z_{r,d}|),\ r = 1, ..., R,\ d = 1, ..., D$. We assume for each $r$ that the vector $\mathbf{Z}_r = [Z_{r,1} \cdots\ Z_{r,D}] \sim MVN(\mu_r,\ \Omega)$. For example, each $Z_{r,d}$ could correspond to a test statistic for the individual null hypothesis of no association between the $r$th SNP in a genome and the $d$th phenotype. We are interested in finding pairs $(r, d)$ such that $\mu_{r,d} \equiv E_0[Z_{r,d}] \neq 0$.

The ELL method proceeds in two stages. The first stage is to test for each $r$, the global null hypotheses $\mu_r = 0$ vs the alternative that $\mu_{r,d} \neq 0$ for some $d$. The second stage will determine for each $r$ for which the null is rejected, the set of $d$ such that $\mu_{r,d} \neq 0$. This two stage approach is appropriate in scenarios where for most $r$, $\mu_r = 0$.

The first stage involves performing $R$ simultaneous tests and some method of multiple testing correction must be employed. Typically we would find a threshold for rejecting each of the $R$ first stage tests that guarantees the desired joint type 1 error.

## 1.2 Identifying which SNPs are associated with at least one expression trait

Let $\Omega_r = V_0(\boldsymbol{Z}_r)$ be the correlation matrix for the vector $\boldsymbol{Z_r} = (Z_{r,1}, ..., Z_{r,D})$. We assume that for all $r$, $\Omega_r$ is constant. For simplicity, from here on we drop the $r$ from the notation and describe the test statistic and global null test for $r = 1$ as it is identical for other $r$.

Suppose the p-values are ordered as $\pi_{(1)} \leq \cdots \leq \pi_{(D)}$. Smaller values of some of $(\pi_{(1)}, ..., \pi_{(D)})$ than would be expected under the global null would be evidence against the global null. Suppose we have cutoff values $h_1, ..., h_D$ and we reject the global null if $\pi_{(d)} \leq h_d$ for any $d$.

When $\Omega = I$, Berk-Jones [3] introduced the following statistic:

$$T = \min_{1 \leq d \leq \delta D} P_0 \left[ \pi_{(d)} \leq \pi_{(d)}^{observed} \mid \Omega = I \right]$$

$\delta$ is chosen to focus the statistic on deviations from the null among the smaller p-values. In their paper they use $\delta = 0.5$. McPeek [4] derived analytic formulas for the p-value of $T$. When $\Omega \neq I$,

analytic formulas don't exist we can use the following idea.

$$\begin{aligned} P_0\left[\pi_{(d)} \leq x\right] &= P_0\left[|Z|_{(D+1-d)} \geq -\Phi^{-1}(x/2)\right] \\ &= P_0\left[\left(\sum_{i=1}^{D} I\left\{|Z_i| \geq -\Phi^{-1}(x/2)\right\}\right) \geq d\right] \\ &= 1 - P_0\left[S\left(-\Phi^{-1}(x/2)\right) \leq d-1\right] \end{aligned}$$

$S(c) = \sum_{d=1}^{D} I\left\{|Z_d| \geq c\right\}$, $I\{\cdot\}$ is the indicator, and $\Phi$ be the CDF of N(0,1).

Following Sun & Lin [5] we approximate the distribution of $S(-\Phi^{-1}(x/2))$ with a Beta Binomial distribution BB$(\lambda, \gamma)$ matched by method of moments. Let $F\left[\ \cdot\ ;\ \lambda(x), \gamma(x)\right]$ be the matched BB$(\lambda, \gamma)$ CDF.

$$P_0\left[\pi_{(d)} \leq x\right] \approx 1 - F\left[\ d-1\ ;\ \lambda\left(x\right), \gamma\left(x\right)\right]$$

Our statistic is then:

$$\begin{aligned} T &= \min_{1 \leq d \leq \delta D} P_0\left[\pi_{(d)} \leq \pi_{(d)}^{observed} \mid \Omega\right] \\ &= \min_{1 \leq d \leq \delta D}\left(1 - F\left[\ d-1\ ;\ \lambda\left(\pi_{(d)}^{observed}\right), \gamma\left(\pi_{(d)}^{observed}\right)\right]\right) \end{aligned}$$

Given observed $T^*$, to approximate its pvalue we repeatedly sample $\mathbf{Z} \sim MVN(0, \Omega)$ calculate $T$ for each and find the quantile of $T^*$ relative to the sample of $T$.

The pvalue of $T^*$ would then be compared to a genome wide threshold to maintain an joint upper bound on the type I error of the global null hypotheses.

Typically, we estimate $\Omega$ from the sample correlation of $\mathbf{Z}_1, ..., \mathbf{Z}_M$ from our $M$ observed snps. In ELL.py, for each $d$, we pre-compute $1 - F\left[\ d-1\ ;\ \lambda\left(x\right), \gamma\left(x\right)\right]$ for $x$ on a fine grid and then interpolate between the grid points.

### 1.3 Identifying Which Expression Traits are Associated with a Significant SNP

We use a methodology derived from Peterson et. Al. 2016 [6]. Given a set of $M$ SNPs to be considered for eQTL mapping and a genome wide cutoff, we calculate ELL p-values for each. Let $m$ (possibly 0) be the number of SNPs whose ELL pvalue is below the genome wide cutoff. For each SNP for which the ELL p-value reaches the genomewide significance threshold: take the set of marginal association p-values for that SNP and apply FDR with target false discovery rate $0.05 * \frac{m}{M}$.

### 1.4 A Naive Statistic For Comparison

For comparison ELL.py can also output a naive statistic counter-part of the ELL statistic. Again we take the case of $r = 1$ as the other values of $r$ are analogous. The Naive Test Statistic is

$$D \times \left(\min_{d=1}^{D} \pi_d\right)$$

In ELL.py we can compute p-values of this statistic using the same Monte Carlo approach as with the ELL statistic. However, it is also useful to compare the ELL p-value for a snp to the Naive statistic value itself not just the Naive statistic's pvalue.

## 2 Installing ELL

ELL requires python 3.8 and is only tested to work in Linux currently. The following python packages and the version of each package ELL was tested on are listed below:

| Package | Version |
|---|---|
| matplotlib | 3.3.1 |
| numpy | 1.19.1 |
| statsmodels | 0.12.0 |
| scikit-learn | 0.23.2 |
| rpy2 | 3.3.6 |
| pandas | 1.1.3 |

To install ELL, proceed with the following steps:

1. Install Python version 3.8 or later. This can be downloaded from https://www.python.org/downloads/. Currently, ELL.py has only been tested on linux. Follow the directions therein to install.
2. Download ELL.py can from https://github.com/tayoakinbiyi/ELL. You can also download this document.
3. Install R version 3.6 or later from https://cloud.r-project.org/. Follow the directions therein to install.
4. Check that the python executable's location is in the linux PATH environmental variable

   ```
   python –V
   ```

5. If the OS says the command can't be found then you need to add the location it was installed into to the system PATH
6. Verify that R is accessible

   ```
   which R
   ```

7. Check that the necessary packages are installed in your python environment.

   ```
   python list
   ```

   will list all python packages installed, whereas to check for a specifc package *pkg* use

   ```
   python list | grep pkg
   ```

# 3 Using ELL

Place ELL.py in the current directory. One can execute ELL using the following command:

```
python ./ELL.py path/to/config/file
```

The configuration file format is described below. For example, if the config file and ELL.py are in the current directory, then one would execute:

```
python ./ELL.py ./config
```

## 3.1 Config File

The config file should have one line per parameter and should have two total columns that are comma separated. The first column in each row is the name of a parameter, and the second column is the parameter's value. An example config file is included in this document below. The order of rows does not matter but all rows included in the example config file must be included.

The last 10 lines in the example file correspond to modules that can be run on each instance of running ELL.py. To indicate that a module should be run, the value of the second column on any row where the first column is the name of a module should be True. If the value is anything other than True the module will not be run. The order of running modules - when more than one module has True for its second column value - is the same as the order that they are described below. Any combination of modules can be run on any execution of ELL.

One parameter is particularly important for I/O with ELL.py that is *folder*. This parameter gives the main folder. Once ELL.py begins, the current directory is switches to the one in the folder parameter and then all file paths are assumed relative to that folder.

### 3.1.1 Example Config File

| | |
|---|---|
| delta, | 0.2 |
| totalSnps, | 522000 |
| numCandidates, | 29506 |
| numCores, | 36 |
| minEta, | 1e-140 |
| maxEta, | 1e-1 |
| maxIters, | 1e4 |
| numVzEigenValsToKeep, | 194 |
| numHermite, | 2e2 |
| numLam, | 2e5 |
| mcReps, | 1e5 |
| folder, | ellTest |
| globalPvalThresh, | 8.06e-3 |
| FDR, | 0.05 |
| clusterDistance, | 0.8 |
| compute, | True |
| monteCarlo, | False |
| monteCarloCombine, | False |
| score, | False |
| associatedTraits, | False |
| cluster, | False |
| plotEllPvals, | False |
| plotEllStats, | False |
| plotOverlap, | False |

## 3.2 Modules

There are 9 modules ELL can execute named: *compute*, *monteCarlo*, *monteCarloCombine*, *score*, *associatedTraits*, *cluster*, *plotEllPvals*, *plotEllStats*, *plotOverlap*. These are described in detail below. They must be executed in the listed order. Whenever two or modules are specified to be executed in the config file, their order of execution matches their listed order below.

- *compute* The first step in calculating the ELL statistic is the precomputation phase as described in ([1]). The precomputation involves precomputing the ELL statistic on a grid of possible input values so that given actual data, the ELL statistic can be computed by interpolating the closest precomputed ELL value.

- *monteCarlo* The monteCarlo module simulates a sample from the null distribution of the ELL and naive statistics for the set of traits in the file *input/traitLabels*. The covariance matrix in *input/vZ* is decomposed to form $vZ = P\Lambda P^T$. We form $\tilde{\Lambda}$ from $\Lambda$ by setting all but the top *numVzEigenValsToKeep* eigen values of $\Lambda$ to 0. Multiple samples of $Z$ are drawn independently from the null distribution of $MVN(0, P\tilde{\Lambda}P^T)$. An ELL statistic is computed for each sample, and that forms the empirical null distribution.

  If x samples are created then the smallest ELL pvalue that can be calculated relative to that sample is 1/(x+1). This module will create separate files in *intermediate/ellRef* and *intermediate/naiveRef* each of which contains a simulated sample of the null distribution of the ELL

statistic. Since this module is the longest running one, the user may want to only simulate a portion of the desired x samples at one sitting and repeat the module subsequently to expand the number of samples to total x over multiple runs. For example, if the desired final x is 1000 then the user can run *monteCarlo* 10 times, each time setting the parameter *mcReps* - see description below - to 100.

The simulated samples are stored in *intermediate/ellRef* and *intermediate/naiveRef*

- *monteCarloCombine* This module concatenates different simulated samples of the null distribution of the ELL and naive statistics. This must be run even if *moteCarlo* is run only once and with *numCores* equal to 1. If *numCores* is greater than 1 when *monteCarlo* is run then there will necessarily be more than one Monte Carlo sample of the null distribution of the ELL and naive statistics. *monteCarloCombine* is necessary to combine those separate files into one sorted file.

- *score* The *score* module will calculate for each snp - that is each row in *input/Z* - an ELL statistic, a p-value for each ELL statistic formed by comparing the statistic to the simulated null distribution sample created by *monteCarlo*, a naive statistic, and a pvalue for each naive statistic formed by comparing the statistic to the simulated null distribution sample created by *monteCarlo*. This data will be written to a file *output/score*. Subsequent modules add columns to the file *output/score* and those columns are described in the description of *output/score* in the file structure section below.

- *associatedTraits* This module determines which traits each candidate snps - i.e snp whose ELL pvalue is below *globalPvalThresh* - is associated with. This is described in further detail in the description of the file *output/associatedTraits* in the File Structure section below.

  The module then runs the R module hclust [2] which create a hiearchical clustering tree of the candidate snps (i.e. those rows in *input/Z* having an ELL pvalue below *globalPvalThresh*). The clustering tree connects such candidate snps together into clusters. To do this hclust requires a matrix giving the distances of each candidate snp to each of the other candidate snps. See the description of *output/score* in the file structure section below for the details of the distance metric.

- *cluster* This module clusters the set of candidate snps using the R module cutree [2]

  The cutree function requires a height to cut the hierarchical clustering tree created by hclust. This must be supplied to the cluster module as the parameter *clusterDistance* in the config file.

- *plotEllPvals* This module creates a manhattan plot of the -log10 of the ELL pvalues for the rows (i.e. snps) of *input/Z* (see file structure below). The x axis for this plot is the label given to each snp in the file *input/snpLabels* described below in file structures. An attempt is made to convert these labels to float values and if not successful, the index of each snp out of the set is used. I.e. the ith snp - i.e. ith row - in *input/Z* (corresponds to the label i-1). The -log10 of the p-values of the naive statistic for each snp is also plotted.

- *plotEllStats* This module creates a manhattan plot of the -log10 of the ELL statistics for the rows (i.e. snps) of *input/Z* (see file structure below). The x axis for this plot is the label

given to each snp in the file *input/snpLabels* described below in file structures. An attempt is made to convert these labels to float values and if not successful, the index of each snp out of the set is used. I.e. the ith snp - i.e. ith row - in *input/Z* (corresponds to the label i-1). The -log10 of the naive statistic for each snp is also plotted against the right axis.

- *plotOverlap* This module plots a heatmap representation of the level of overlap between candidate snps in the traits that each is associated with. The top triangle (i.e. the portion above the diagonal line from bottom left to top right) of the heatmap indicates on a blueness scale - where white is 0 and darkest blue is 1 - this overlap. The color of the dot at the ith row (starting from the top) and the jth column (starting from the left) indicates the fraction of overlap in associated traits between the ith and jth candidate snps. See the description of *output/associatedTraits* for details on overlap calculations. The bottom triangle has only dark blue or pure white colors. Dark blue in the grid for row i and column j indicates that the ith and jth candidate snps were placed in the same cluster, and white indicates they are in different clusters.

### 3.3 Main Parameters

- **delta** This the fraction $\delta$ described in section (1).
- **numVzEigenValsToKeep** The number of eigenvalues of *input/vZ* to keep. vZ is spectrally decomposed, the top *numVzEigenValsToKeep* eigenvalues are kept and the rest set to 0.
- **minEta** The smallest ELL value that the user needs the code to be able to produce. Since the ELL.py code precomputes the ELL statistic on a grid of possible input values, the smaller *minEta* the smaller the possible ELL statistic that the code can output. In the paper a vaue of 1e-140 was used. The user may check the output file output/score after running all modules up to and including scoere, to see if there are any value of the column ellStats in *output/score* which are equal to minEta. If there are then the execution of ELL.py from the compute model onwards should be redone with a smaller value of minEta. Smaller values of *minEta* will also cause the compute module to take longer and will cause the intermediate files produced by ELL.py to be larger. Suggested default 1e-140.
- **folder** The location of the main folder to be used by ELL.py to read input data and deposit output files. This folder is referred to in this document as the main folder.
- **totalSnps** The total number of snps genome-wide that were evaluated by ELL.py. If the user breaks the genome wide set of snps into smaller sets - for example separating them by chromosome - and runs ELL.py on each separately, then *totalSnps* should be the total across all sets. *totalSnps* is equal to $M$ in section (1).
- **numCandidates** The number of snps - out of the snps corresponding to the rows of *input/Z* - which have an ELL p-value below globalPvalThresh. The module score outputs this number by printing it to screen upon completion. If the genome-wide total set of snps is split into smaller sets - for example by chromosome - and the user runs ELL.py on each set separately, then *numCandidates* is the total number of candidates across all sets of snps. *numCandidates* is equal to $m$ in section (1).
- **numCores** The number of cpu cores that ELL.py should use. Several modules are able to internally run in parallel using multiple processes. ELL.py must be run on python 3.8 or later which allows the use of the shared_memory module of the multiprocessing package. During

internal parallel processing, files are created in /dev/shm which are shared between multiple ELL.py processes. If problems with multiple processes occur during running, one can reduce *numCores* to see if that helps.

The larger *numCores*, the more memory that will in general be occupied during running ELL.py. Particularly, the *monteCarlo* module of ELL.py will use roughly twice as much memory if *numCores* is roughly doubled.

- **mcReps** The *monteCarlo* module simulates a sample of ELL statistics under the Null using the covariance matrix in *input/vZ* as the true between trait covariance matrix of the association statistics between each snp and the traits. This is typically the longest running module of ELL.py as a large sample is typically required. If x samples are simulated then the smallest ELL p-value possible would be 1/(x+1). On any run of *monteCarlo* in ELL.py, *mcReps* is the number of samples that are simulated.

  If the user is running ELL.py separately on different subsets of the genome-wide set of snps - for example subsets based on chromosome - then a different sample of the null distribution of the ELL stat will need to be simulated for each subset of snps.

  The the user can opt to have ELL.py do the simulation of null draws of ELL statistics in batches. In this case the user must make sure that across different runs of *monteCarlo* module, the sum of *mcReps* equals the total number of samples, x, desired so that 1/(x+1) is small enough. The larger *mcReps*, the more memory that will be used while running *monteCarlo*. So the user may opt to run *monteCarlo* several times with *mcReps* set to a fraction of the desired final value.

- **globalPvalThresh** This parameter is the value that a snp's ELL p-value must be below in order to be a 'candidate' snp. This should be a genome wide quantity, meaning it should be valid for simultaneous comparison to ELL p-values of all snp's on the genome being evaluated.

- **FDR** This parameter determines the global false discovery rate for which traits are deemed to be associated with each snp determined (by having an ELL p-value *globalPvalThresh*) to be an eQTL with at least one trait. See the above description of *output/associatedTraits*.

- **clusterDistance** The height at which to cut the hierarchical clustering tree created during the module *eqtlDistance*. Higher values will result in less clusters of snps which have less overlap in traits they are associated with.

### 3.4 Ancillary Parameters

- **numHermite** The ELL.py code uses a BetaBinomial approximation which involves a statician's hermite polynomial sequence. *numHermite* is the number of terms in that sequence to calculate. Higher values lead to higher accuracy but slower computation. Suggested default 200.

- **numLam** The number of grid points used in the pre-computation. Higher values lead to tighter approximation but also slower results and larger intermediate files in the intermediate files folder. Suggested default 1e5.

- **maxIters** The maximum number of iterations the pre-computation code will attempt when trying to find the appropriate boundaries for the pre-computed grid of ELL values. Suggested default 1e4

# 4 Input

ELL.py reads files and output files to the path specified in the main folder in the config file. The main folder should initially contain just one sub-folder *input*. The user should place in this input folder four files: Z, traitLabels, snpLabels and vZ. These four files are described below. The files must have these names and no other file will be read.

Two other sub-folders are created by ELL.py in the main folder, *intermediate* and *output*. Intermediate contains files that some modules create to be used by successive modules. The output sub-folder, contains files which give the results of running many modules of ELL.py such as score and the plotting modules.

ELL.py will not create the main folder if it does not exist. The user must create it because the user must put certain files into the a sub-folder of the main folder called 'input'. These input files are detailed below in terms of format and content. ELL.py will create two other sub folders of the main folder: *intermediate* and *output*. The user can effectively ignore intermediate.

- **input/Z** This file contains the association statistics (z-scores) between snps and each trait. When the score module is run, ELL.py will calculate one ELL statistic and p-value for each row of Z. The file should be a text file, comma separated with D columns, no header, no index column, and the same number of rows as the file *input/snpLabels* and the same number of columns as the number of rows in *input/traitLabels* file. Order of rows in Z should match order of rows in *input/snpLabels* and order of columns in Z should match order of rows in *input/traitLabels*.

  add example

- **input/vZ** The covariance matrix of the association statistic z scores. The file should be comma delimited, no header, and no index column. Each row of this file should have D floats comma separated. The file should have D rows and should be symmetric. Each line is ended with \n.
  Here is an example where D=4

  ```
  1.04,   2.2,    -2,         1.1
  2.2,    0.97,   1.8,        -0.51
  -2,     1.8,    -1.099,     1
  1.1,    -0.51,  1,          -0.91
  ```

  vZ_i,j (the element of vZ at row i, column j) should be the covariance between the assocation statistic for a snp and the ith trait and athe association statistic for a snp and the jth trait. ELL.py assumes that this covariance is the same across the set of snps correspnding to rows in input/Z.

- **input/snpLabels** The ith row contains the label of the ith snp. No header, no index column. One string label per row, and the file should have the same number of rows as the file Z below. Example with 5 snps, here the label is the chromosome and location of the snp on that chromosome in Mbp

```
chr1-2.456
chr1-2.48
chr1-2.8
chr1-3.1
chr1-3.656
```

- **input/traitLabels** The ith row contains the label of the ith trait. There should be D rows, no header, and no index column. One string label per row.

  Example where D=4, here the label is the chromosome and trait name

  ```
  Xkr4-chr2
  Mettl13-chr2
  Vamp4-chr2
  Myoc-chr2
  ```

# 5 output

ELL will create a folder called output in the main folder. This sub-folder will contain all of the output files ELL produces that are to be used by the user.

Note that some modules add columns to existing files output by previous modules so one should not change the contents of any file in the *output* sub-folder.

- **output/associatedTraits** Matrix of snp,trait pairs where each pair represents a snp and one trait it is deemed by ELL.py to be associated with. This association is based on both the ELL p-value of the snp and the individual marginal association statistic p-values of the snp and each trait. File has one snp,trait pair per row, comma separated and no header or index column. If a snp is not associated with any trait then it will not appear in the *output/associatedTraits* file. If a snp has an ELL p-value above the global threshold the user provides (see parameter *globalPvalThresh*) then it will not appear in the *output/associatedTraits* file. The snps and traits will be labelled by their label in *input/snpLabels* and *input/traitLabels* user input files.

  The algorithm for associated traits works as follows. Suppose a snp has an ELL p-value below *globalPvalThresh*. Then it's marginal association statistics (i.e. z scores between the snp and each trait) are ordered from largest to smallest, each is converted to a p-value, and an FDR analysis is performed with the FDR cutoff set to 0.05**numCandidates/totalSnps* (see parameters *numCandidates* and *totalSnps*). It is mathematically possible for a snp to have an ELL p-value below the *globalPvalThresh* parameter value but not have any traits associated.

- **output/clusterOverlapMatrix** Matrix indicating the cluster memberships of each marker and the prportion of overlap in associated traits between each pair of markers. For $1 \leq i \leq j \leq D$ (where $D$ is defined as in 1), the entry of **output/clusterOverlapMatrix** at row $i$, column $j$ will be 1 minus the proportion overlap in associated traits between the $i$th and $j$th markers (where the proportion is as defined in *output/associatedTraits*.

For $1 \leq i > j \leq D$, the entry of **output/clusterOverlapMatrix** at row $i$, column $j$ will be 1 if thr $i$th and $j$th markers are placed in the same cluster and 0 otherwise.

- **output/score** This file contains the ELL statistics and p-values output by the ELL.py code. The file is first created by the module score. However, columns are subsequently added by the modules *associatedTraits* and *cluster*. When all modules have been run, the file will have the following columns: snpLabels, ellStats, ellPvals, naiveStats, naivePvals, numAssociatedTraits and cluster. There is no header or index column and the file is comma separated. There is one row for each row in the input file *input/Z*.

  The snpLabels column contains the labels in the input file *input/snpLabels*. The ellStats column contains the ELL statistic evaluate for each snp by the score module. The ellPvals column contains the Monte Carlo based p-value of the ELL statistic for each snp as calculated by the score module. The naiveStats column contains the naive statistic for each snp.

  For example, if there are 4 traits and a snp has the following row in *input/Z* 1.1,-0.8,3.5,2.2 then the naive statistic is 0.0009305163161421001. This columns is added by the module score. The column naivePvals is calculated for each naiveStatistic by comparing each naiveStatistic value to a Mont eCarlo simulated sample of the null distribution of the navieStatistic. This column is added by the module *score*. The column numAssociatedTraits contains the number of traits each snp is assocated with. If a snp has an ELL p-value above *globalPvalThresh* then it will automatically have 0 associated traits. The column cluster contains the cluster to which the module *cluster* assigns each snp with ¿0 associatedTraits. If a snp has 0 associated traits then it is nominally given cluster 0 but its cluster is meaningless.

  To determine which cluster a snp belongs to, the R module hclust is used. The distance between two snps is 1 minus the ratio of the number of traits associated with both snps to the smaller of the number of traits each of the two snps is associated with. For example, if snp 1 is associated with traits 1,2 and 4 and snp 2 is associated with traits 1, 2, 5, 6 and 16 then the distance between the two snps is 1-2/4=0.5. The higher the fraction of associated traits two snps share, the less distance between them. The R module cutree is used to find a particular clustering based on the user supplied parameter *clusterDistance*. The higher *clusterDistance* the smaller the number of clusters and the less overlap in associated traits between snps in the same cluster.

# 6 Acknowledgments

We gratefully acknowledge:

- GEMMA. This code was tested using output from GEMMA.
- The hclust package. We used code from it for performing hierarchical clustering of candidate snps
- rpy2 package. We used the rpy2 package to call R routines from python.

# 7 References


[1] Akinbiyi, Takintayo. "Equal Local Levels: A Global Testing Approach with Application to Trans eQTL Detection." Order No. 28024590, The University of Chicago, 2020. http://proxy.uchicago.edu/login?url=https://www-proquest-com.proxy.uchicago.edu/dissertations-theses/equal-local-levels-global-testing-approach-with/docview/2472183090/se-2?accountid=14657

[2] https://stat.ethz.ch/R-manual/R-devel/library/stats/html/hclust.html

[3] Berk, R.H., Jones, D.H. Goodness-of-fit test statistics that dominate the Kolmogorov statistics. Z. Wahrscheinlichkeitstheorie verw Gebiete 47, 47–59 (1979). https://doi.org/10.1007/BF00533250

[4] Weine, E and McPeek, M S. Efficient Calculation of Alternative Agnostic Testing Bands for QQ-Plots. pre-print

[5] Sun, Ryan, and Xihong Lin. "Set-Based Tests for Genetic Association Using the Generalized Berk-Jones Statistic." ArXiv:1710.02469 [Stat], Oct. 2017. arXiv.org, http://arxiv.org/abs/1710.02469.

[6] Christine B. Peterson, Marina Bogomolov, Yoav Benjamini, andChiara Sabatti.Many phenotypes with many false discoveries: error controllingstrategies for multitrait association studies.Genetic Epidemiology, 40:45–56, 2016.