glint Documentation

Release 1.0.4

Elior Rahmani

Jun 25, 2017

Contents

GLINT is a user-friendly command-line tool for fast analysis of genome-wide DNA methylation data generated using the Illumina human methylation arrays (27K, 450K and 850K/EPIC).

GLINT allows to easily run a pipeline of Epigenome-Wide Association Study (EWAS) under different models while accounting for known confounders in methylation data. Particularly, GLINT provides implementations of ReFACTor and EPISTRUCTURE, for accounting for tissue heterogeneity and capturing ancestry information from methylation data.

CHAPTER 1

Getting GLINT:

Download and installation

The latest version of GLINT and the tutorial files are available on github here. Please read the associated README file on github for details about downloading and installing GLINT.

CHAPTER 2

Quick start tutorial:

Quick start tutorial

This tutorial will quickly walk you through the basic functionality of GLINT. For this tutorial we use subset of a public dataset from GEO (accession ID GSE77716; the dataset is described in details in Rahmani et al.[1]). In order to run this tutorial you will need to download GLINT and get the tutorial files (see here).

The tutorial files include:

- *datafile.txt* - 50,000 sites by 96 samples matrix of methylation levels
- *covariates.txt* - covariates matrix, each column corresponds to one (numeric) covariate
- *phenotypes.txt* - phenotypes matrix, each column corresponds to one phenotype

Bellow is a set of simple commands, together composing a full pipeline of EWAS analysis (after raw data normalization). The commands bellow assume the user downloaded GLINT (see Download and installation) and added the tutorial files into the software's root directory. For more details about any specific argument see the documentation.

Note: Avoid execution errors by using the latest release of GLINT and the tutorial files, and make sure to follow the installation instructions in the README file.

1. **Create GLINT files**

First, we start by saving a binary version of our data: the methylation data file, covariates and phenotypes of interest. This step will allow a substantial speed-up in all following commands. Navigate to the GLINT directory and run the following:

```
python glint.py --datafile datafile.txt --covarfile covariates.txt --phenofile␣
↪phenotypes.txt --gsave
```

[1] Rahmani, Elior, Noah Zaitlen, Yael Baran, Celeste Eng, Donglei Hu, Joshua Galanter, Sam Oh et al. "Sparse PCA corrects for cell type heterogeneity in epigenome-wide association studies." Nature methods 13, no. 5 (2016): 443-445.

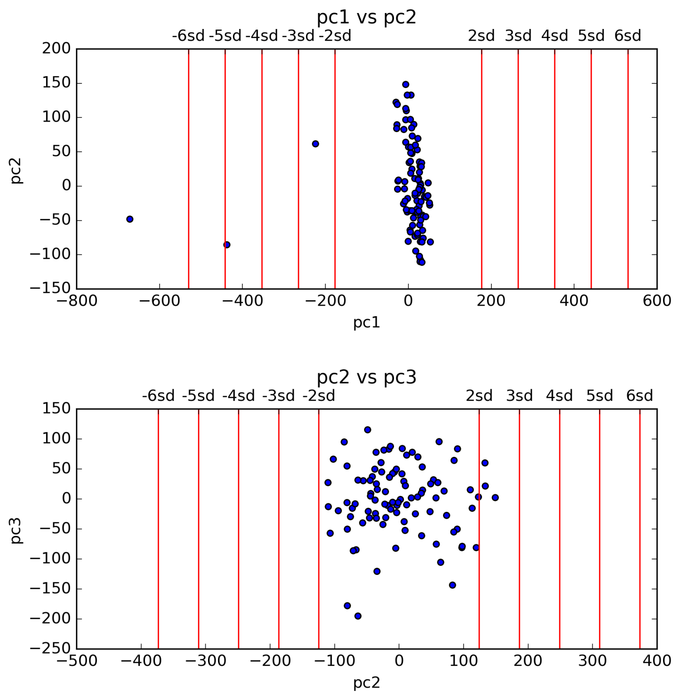
The output of this command is the file *datafile.glint*, a binary version of the methylation data, and two additional files, *datafile.samples.txt* and *datafile.sites.txt*, providing textual information about the samples and sites in the data.

2. **Detect outliers**

We begin the analysis by looking for outliers in the data. We first run:

```
python glint.py --datafile datafile.glint --plot --plotpcs --numpcs 2 --out pcs_plot
```

This command generates a figure titled *pcs_plot.png*, showing scatter plots of the first two principal components (PCs) of the data. Note that this command also generates an additional figure, *pcs_plot.eps*, a publication quality version of the same figure.



Note: For using **‘–plot‘_** when working on a remote server (e.g., via SSH), make sure X11-forwarding is enabled.

3. **Remove outliers**

For this tutorial we consider samples with values more extreme than 4 sandard deviations (SDs) in their first two PCs as outliers. Following that definition, we currently have 2 outliers in the data, as reflected in the top panel of the *pcs_plot.png* figure. We remove these outlier samples by indicating 4 SDs as the maximum level allowed for PC number 1:

```
python glint.py --datafile datafile.glint --maxpcstd 1 4 --gsave --out data_cleaned
```

As before, we use the –gsave argument for generating GLINT files, only this time with outliers excluded. This results in the following files: *data_cleaned.glint*, *data_cleaned.samples.txt* and *data_cleaned.sites.txt* files.

4. **Capture cell type composition**

Since our data were collected from a heterogeneous source (blood tissue), we run ReFACTor in order to account for the cell type composition in the downstream analysis and generate new GLINT files with the results. The resulted ReFACTor components will be used later as covariates in our EWAS analysis, as tissue heterogeneity is a potential confounder in EWAS[2]. In order to boost ReFACTor's performance in capturing the cell composition, we run ReFACTor while adding potential methylation altering factors as covariates. We do that by using the –covar argument which allows us to add covariates by their names (as they appear in the covariates file):

```
python glint.py --datafile data_cleaned.glint --refactor --k 6 --covar age gender␣
↪chip1 chip2 chip3 chip4 chip5 chip6 chip7 chip8 --gsave --out data_cleaned_v2
```

This command creates *data_cleaned_v2.refactor.components.txt* and *data_cleaned_v2.efactor.rankedlist.txt* files (see Tissue heterogeneity for more details), and updated GLINT files: *data_cleaned_v2.glint*, *data_cleaned_v2.samples.txt* and *data_cleaned_v2.sites.txt*. Note that *data_cleaned_v2.samples.txt* includes new covariates: rc1, rc2, ..., rck - these are the ReFACTor components.

5. **Infer population structure**

Since our data were collected from admixed population and we do not have ancestry information available, we estimate the population structure in the data directly from the methylation levels using the EPISTRUCTURE algorithm[3] and generate new GLINT files with the results. In order to boost the performance of EPISTRUCTURE in capturing the ancestry information, we run EPISTRUCTURE while adding strong genome-wide effectors as covariates - in our case we add the estimates of the cell composition. The resulted EPISTRUCTURE PCs will be used later as covariates in our EWAS:

```
python glint.py --datafile data_cleaned_v2.glint --epi --covar rc1 rc2 rc3 rc4 rc5␣
↪rc6 --gsave --out data_final
```

This command results in a file titled *data_final.epistructure.pcs.txt* (see Inferring population structure for more details). In addition, we now have *data_final.glint*, *data_final.samples.txt* and *data_final.sites.txt* files. Note that *data_final.samples.txt* includes a new covariate named *epi1*, which is the first EPISTRUCTURE component (by default –epi outputs one component).

6. **Run EWAS**

We are now ready to run association test for each site. In this tutorial we will run EWAS on a simulated phenotype. The phenotype is selected using the –pheno argument, according to the phenotype's name in the phenotypes file. Since our phenotype is continuous we will use a linear regression model, and in addition to standard age and gender covariates, we will include the ReFACTor components and the first EPISTRUCTURE component in the analysis in order to account for tissue heterogeneity and population structure. In addition, using the –stdth argument we can neglect nearly constant sites having very low variability, and using the arguments –rmxy, –rmns and –rmpoly, we can also neglect X and Y chromosomes sites, cross-reactive sites and polymorphic sites[4].

```
python glint.py --datafile data_final.glint --ewas --linreg --pheno y1 --covar age␣
↪gender rc1 rc2 rc3 rc4 rc5 rc6 epi1 --stdth 0.01 --rmxy --rmns --rmpoly
```

This command outputs a file titled *results.glint.linreg.txt* with the results of the association test. Note that the results are sorted by their association p-value.

7. **Plot results**

Lastly, we would like to plot the results in the *results.glint.linreg.txt* file. GLINT allows to visualize the results by plotting a qq-plot (–qqplot) and a Manahattan plot (–manhattan) as follows.:

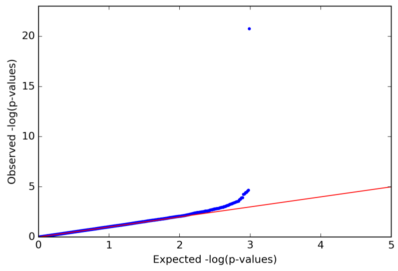
```
python glint.py --plot --qqplot --manhattan --results results.glint.linreg.txt
```

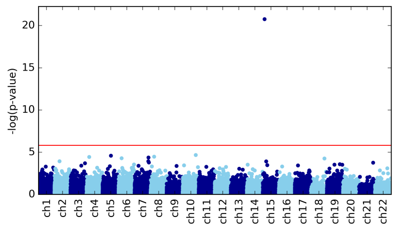
[2] Jaffe, Andrew E., and Rafael A. Irizarry. "Accounting for cellular heterogeneity is critical in epigenome-wide association studies." Genome biology 15, no. 2 (2014): 1.

[3] Rahmani, Elior, Liat Shenhav, Regev Schweiger, Paul Yousefi, Karen Huen, Brenda Eskenazi, Celeste Eng et al. "Genome-wide methylation data mirror ancestry information." bioRxiv (2016): 066340.

[4] Chen, Yi-an, Mathieu Lemire, Sanaa Choufani, Darci T. Butcher, Daria Grafodatskaya, Brent W. Zanke, Steven Gallinger, Thomas J. Hudson, and Rosanna Weksberg. "Discovery of cross-reactive probes and polymorphic CpGs in the Illumina Infinium HumanMethylation450 microarray." Epigenetics 8, no. 2 (2013): 203-209.

This command generates four figures. The first two, *results.glint.qqplot.png* and *results.glint.manhattan.png*, show a qq-plot and a Manhattan plot of the results. The last two are publication quality versions of the same figures (*.eps* files).

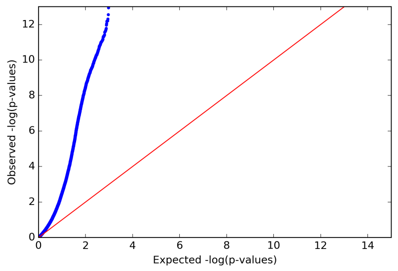


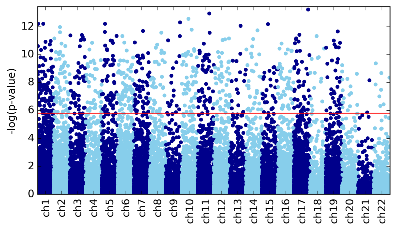


8. **Unadjusted EWAS**

Finally, in our example we found a single significant association in chromosome 15 (cg20510272), as reflected in the qq-plot and in the Manhattan plot. The phenotype we used here was simulated to be correlated with the cell type composition in the data and only one site (cg20510272) was artificially changed to be causal with respect to the phenotype. Since the phenotype is correlated with the cell type composition, performing uncorrected analysis is expected to result in many spurious assocaitions. We can easily see that by running an unadjusted EWAS by simply repeating our EWAS analysis, this time without including the ReFACTor components as covariates. Here, we use a single command for running the EWAS analysis and generating the plots at the same time:

```
python glint.py --datafile data_final.glint --ewas --linreg --pheno y1 --covar age␣
↪gender epi1 --stdth 0.01 --rmxy --rmns --rmpoly --plot --qqplot --manhattan --out␣
↪unadjusted
```




-log(p-value)
0
2
4
6
8
10
12
ch1
ch2
ch3
ch4
ch5
ch6
ch7
ch8
ch9
ch10
ch11
ch12
ch13
ch14
ch15
ch16
ch17
ch18
ch19
ch20
ch21
ch22

CHAPTER 3

Using GLINT:

Input

GLINT accepts as input array methylation data that were generated using the Illumina arrays (27K, 450K and EPIC). The following section describes the arguments that allow to provide data files for GLINT, and the *–gsave* argument that allows to save and work with a binary version of the data (*GLINT files*) for gaining speed-up in computation.

Note: GLINT assumes that input data are given after raw data preprocessing (i.e. after normalization).

Note: GLINT currently does not allow NA values in the data. For users having NA values in their data files see *handling NA values*.

Input files

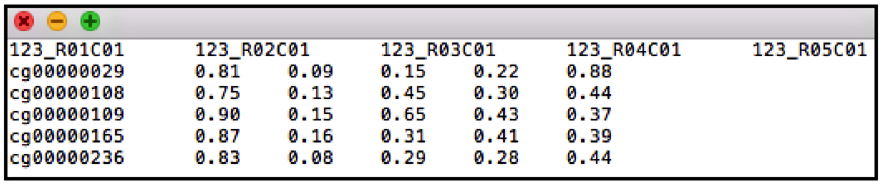
We recommend using the tutorial files as an example for the required file formats. **–datafile:**

Path to a file containing sites by samples matrix of methylation levels. The first row should include sample identifiers and the first column should include CpG identifiers. The first row may include the field “ID” at the beginning. The file can be either tab-delimited, comma-delimited or space-delimited. The matrix entries are not allowed to include quotes.

For example, adding the following to your GLINT command:

```
--datafile datafile.txt
```

will load the methylation data matrix in the *datafile.txt* file. See the tutorial *datafile.txt* file as an example file.



```
123_R01C01      123_R02C01      123_R03C01      123_R04C01      123_R05C01
cg00000029      0.81    0.09    0.15    0.22    0.88
cg00000108      0.75    0.13    0.45    0.30    0.44
cg00000109      0.90    0.15    0.65    0.43    0.37
cg00000165      0.87    0.16    0.31    0.41    0.39
cg00000236      0.83    0.08    0.29    0.28    0.44
```

Fig. 3.1: Figure 1: Example of a data file.

Note: In order to speed-up GLINT, we recommend working with a binary version of the data. See *–gsave* for more details.

Note: For users having *.Rdata* files with methylation data we provide a script for generating files in the required format - see *Convert RData file into GLINT format* for more details.

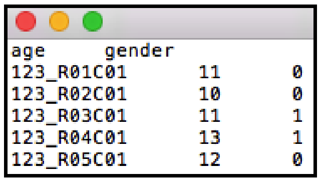
–covarfile

Path to a file containing samples by covariates matrix. The first row may be a row of headers - the names of the covariates, and the first column should include sample identifiers. The first row, if provided headers, may include the field "ID" at the beginning. If a row of headers is not provided then GLINT will automatically generate a name for each covariate. The file can be either tab-delimited, comma-delimited or space-delimited. The matrix entries are not allowed to include quotes, and covariates must be numeric (i.e. categorial covariates should be encoded numerically).

For example, adding the following to your GLINT command:

```
--covarfile covariates.txt
```

will provide the covariates matrix in the *covariates.txt* file. See the tutorial *covariates.txt* file as an example file.



```
age     gender
123_R01C01      11      0
123_R02C01      10      0
123_R03C01      11      1
123_R04C01      13      1
123_R05C01      12      0
```

Fig. 3.2: Figure 2: Example of a covariates file.

Note: More than one covariates file can be provided, e.g. *–covarfile covariates1.txt covariates2.txt*.

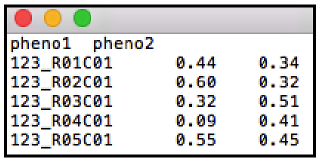
–phenofile

Path to a file containing samples by phenotypes matrix. The first row may be a row of headers - the names of the phenotypes, and the first column should include sample identifiers. The first row, if provided headers, may include the field "ID" at the beginning. If a row of headers is not provided then GLINT will automatically generate a name for each phenotype. The file can be either tab-delimited, comma-delimited or space-delimited. The matrix entries are not allowed to include quotes, and phenotypes must be numeric (i.e. categorial phenotypes should be encoded numerically).

For example, adding the following to your GLINT command:

```
--phenofile phenotypes.txt
```

will provide the phenotypes matrix in the *phenotypes.txt* file. See the tutorial *phenotypes.txt* file as an example file.



```
pheno1  pheno2
123_R01C01      0.44    0.34
123_R02C01      0.60    0.32
123_R03C01      0.32    0.51
123_R04C01      0.09    0.41
123_R05C01      0.55    0.45
```

Fig. 3.3: Figure 3: Example of a phenotypes file.

Note: More than one phenotypes file can be provided, e.g. *–phenofile phenotypes1.txt phenotypes2.txt*.

–out

Allows to change the default titles of output files. Use this argument for changing the prefix of the default title of each argument that produces output files.

For example:

```
python glint.py --datafile datafile.txt --gsave --out newdata
```

will generate GLINT files (see *–gsave*) with *newdata* prefix.

GLINT files

–gsave

Saves GLINT files, including a binary version of the methylation data (*.glint* file) for gaining computation speed-up in the following commands. In addition, this command saves two additional files:

- *datafile.sites.txt* - contains the CpG identifiers of the sites in the data and additional information for each CpG: chromosome, position, nearest gene and genomic category.
- *datafile.samples.txt* - contains the sample identifiers of the samples in the data. If *–covarfile* and *–phenofile* are used then this file also includes the phenotypes and covariates for each sample.

For example:

```
python glint.py --datafile datafile.txt --gsave
```

will save a binary data file titled *datafile.glint* and two additional files titled *datafile.samples.txt* and *datafile.sites.txt*. The following command:

```
python  glint.py --datafile datafile.txt --covarfile covariates.txt --phenofile␣
↪phenotypes.txt --gsave
```

will also include the covariates and phenotypes information found in the *covariates.txt* and *phenotypes.txt* files in the *datafile.samples.txt* file.

Note: Never change the *datafile.samples.txt* and *datafile.site.txt* files manually. Changes can be made using the data management commands.

–txtsave

Allows to save a textual version of the data contained in a binary *.glint* file.

For example:

```
python glint.py --datafile datafile.glint --txtsave
```

will create a file titled *datafile.txt* with a textual version of the methylation matrix in *datafile.glint*.

Note: *–txtsave* can be also used to save a new version of textual format of previous textual files (i.e. *–txtsave* is not restricted to get *.glint* file as an input).

Convert R file to GLINT format

convertToGlintInput.R:

We provide this R script for users having methylation data matrix in *.RData* format. This script gets as an input *.RData* file with sites by samples methylation data matrix saved as a data frame or a matrix variable with CpGs identifiers as row names and sample identifiers as column names. In addition to the *.RData* file name, the script optionally can take two additional arguments:

- varname - if more than a single data frame / matrix variable exists in the *.RData* file then the name of the methylation data variable should be provided. If this argument is not provided then the script automatically attemps to find data frame or a matrix variable.
- transpose - if the methylation data matrix is formatted as samples by sites rather than sites by samples then providing this argument with the value 'true' will transpose the data matrix.

For example:

```
Rscript convertToGlintInput.R datafile.RData X
```

will save a tab-delimited text file containing sites by samples methylation data matrix as appear in the variable X that is saved in the *datafile.RData* file. The resulted file can be then provided as an input to GLINT (using *–datafile*).

Alternatively:

```
Rscript convertToGlintInput.R datafile.RData X true
```

will assume that the information in the variable X is formatted as samples by sites and therefore should be transposed.

Handling NA values

GLINT currently does not allow NA values in the data. For users having NA values in their data we provide an external script *replace_missing_values.py* for a basic imputation of NA values. This script replaces NA values of each site with its mean methylation level (according to all non-NA values of the site), and outputs a new data file with no NA values that can be provided to GLINT as an input.

replace_missing_values.py supports the following arguments:

–datafile - path to a data file (required)

–chr - the symbol (character) indicating missing values in the input file (required)

–maxs - the maximum fraction of missing values allowed per site (required; value between 0 and 1). Sites exceeding this fraction of missing values will be excluded from the output data.

–maxi - the maximum fraction of missing values allowed per sample (required; value between 0 and 1). samples exceeding this fraction of missing values will be excluded from the output data.

–sep - the delimiter in the data file (optional; default value is "\t")

–suffix - the suffix for the output file name (optional; default value is *.no_missing_values*)

For example:

```
python replace_missing_values.py --datafile datafile.txt --chr NA --maxs 0.03 --maxi␣
↪0.03
```

will save a tab-delimited text file titled *datafile.no_missing_values* with imputed values for matrix entries with "NA" values. The resulted file will not include sites and samples having more than 3% missing values.

Data management

The following section describes arguments that allow to perform basic data management and quality control procedures on data.

Note: The example commands described bellow assume that the user generated GLINT files with covariates file and phenotypes file.

Note: Data management commands applied to data do not change the input files. In order to save the changes into a new file use the –gsave or –txtsave commands.

Outliers detection

–maxpcstd

Filters outlier samples using PCA. This argument removes samples with principal components (PCs) above or below a specified number of standard deviations.

For example:

```
python glint.py --datafile datafile.glint --maxpcstd 1 3
```

will remove all samples having extreme values of PC 1 (more than 3 standard deviations).

Note: Use the –plotpcs argument for plotting the first several PCs of the samples in order to determine whether outliers exist in your data.

Note: You can remove outliers based on more than one PC at the same time. For example, for indicating 3 SDs as the maximum level allowed for both PC 1 and PC2 use: *–maxpcstd 1 3 –maxpcstd 2 3*.

Data filtering

–include

Allows to filter sites. This argument gets a text file with a list of CpG identifiers (one CpG identifier per row) and considers only these sites.

For example:

```
python glint.py --datafile datafile.glint --include list.txt
```

will exclude all the sites not indicated in the *list.txt* file. **–exclude**

Allows to filter sites. This argument gets a text file with a list of CpG identifiers (one CpG identifier per row) and considers all sites except for those in the list.

For example:

```
python glint.py --datafile datafile.glint --exclude list.txt
```

will exclude all the sites indicated in the *list.txt* file. **–keep**

Allows to filter sampels. This argument gets a text file with a list of sample identifiers (one sample identifier per row) and considers only these samples.

For example:

```
python glint.py --datafile datafile.glint --keep list.txt
```

will remove all the samples not indicated in the *list.txt* file. **–remove**

Allows to filter sampels. This argument gets a text file with a list of sample identifiers (one sample identifier per row) and considers all samples except for those in the list.

For example:

```
python glint.py --datafile datafile.glint --remove list.txt
```

will remove all the samples indicated in the *list.txt* file. **–minstd**

Filters sites by their variance. This argument removes all sites with standard deviation below the specified value. This command can be used to remove nearly-constant sites that are less likely to result in significant associations in EWAS.

For example:

```
python glint.py --datafile datafile.glint --minstd 0.01
```

will remove all sites with standard deviation bellow 0.01. **–minmean**

Filters sites by their mean methylation levels. This argument removes all sites with mean methylation level below the specified value.

For example:

```
python glint.py --datafile datafile.glint --minmean 0.2
```

will remove all sites with mean methylation level bellow 0.2. **–maxmean**

Filters sites by their mean methylation levels. This argument removes all sites with mean methylation level above the specified value.

For example:

```
python glint.py --datafile datafile.glint --maxmean 0.8
```

will remove all sites with mean methylation level above 0.8. **–rmxy**

Filters out non-autosomal sites (sites in chromsomes X and Y).

For example:

```
python glint.py --datafile datafile.glint --rmxy
```

will remove all non-autosomal sites from the data. **–rmns**

Filters out cross-reactive (non specific) sites according to Chen et al.[1] and McCartney et al.[2].

For example:

```
python glint.py --datafile datafile.glint --rmns
```

will remove all non specific sites from the data. **–rmpoly**

Filters out polymorphic sites according to Chen et al.[1].

For example:

[1] Chen, Yi-an, Mathieu Lemire, Sanaa Choufani, Darci T. Butcher, Daria Grafodatskaya, Brent W. Zanke, Steven Gallinger, Thomas J. Hudson, and Rosanna Weksberg. "Discovery of cross-reactive probes and polymorphic CpGs in the Illumina Infinium HumanMethylation450 microarray." Epigenetics 8, no. 2 (2013): 203-209.

[2] McCartney, Daniel L., Rosie M. Walker, Stewart W. Morris, Andrew M. McIntosh, David J. Porteous, and Kathryn L. Evans. "Identification of polymorphic and off-target probe binding sites on the Illumina Infinium MethylationEPIC BeadChip." Genomics Data 9 (2016): 22-24.

```
python glint.py --datafile datafile.glint --rmpoly
```

will remove all polymorphic sites from the data.

Tissue heterogeneity

When methylation data are collected from an heterogeneous source (e.g. whole blood) the cell type compositions of samples in the data are known to be a major source of variation, and therefore a potential confounder in EWAS. Here we provide an implementation of the ReFACTor algorithm by Rahmani et al. for inferring cell type compostion information in methylation data[1]. In addition, we provide an implementation of the reference-based method by Houseman et al.[2], allowing to estimate cell counts (cell proportion) of the samples in the data.

Note: The example commands described bellow assume that the user generated GLINT files with covariates file and phenotypes file.

ReFACTor

The ReFACTor algorithm calculates components that are correlated with the cell type composition of the samples in the data by applying an unsupervised feature selection step followed by principal component analysis (PCA). These components can be added as covariates in a downstread analysis in order to account for the tissue heterogeneity. Note that ReFACTor calculates linear transformations of the cell type composition rather than estimates of absolute cell count values. **–refactor**

Computes the ReFACTor components and generates two output files:

- *refactor.components.txt* - a text file with the ReFACTor components.
- *refactor.rankedlist.txt* - a text file with the CpGs in the data matrix ranked by their level of information according to ReFACTor.

Note: For best performance of ReFACTor we recommend adding known gneome-wide effectors as covariates using the *–covar* argument.

Note: GLINT computes the ReFACTor components while automatically ignoring X and Y chromosomes sites, polymorphic sites and cross-reactive sites according to Chen et al.[3] for 450K array data and according to McCartney

[1] Rahmani, Elior, Noah Zaitlen, Yael Baran, Celeste Eng, Donglei Hu, Joshua Galanter, Sam Oh et al. “Sparse PCA corrects for cell type heterogeneity in epigenome-wide association studies.” Nature methods 13, no. 5 (2016): 443-445.

[2] Houseman, Eugene Andres, William P. Accomando, Devin C. Koestler, Brock C. Christensen, Carmen J. Marsit, Heather H. Nelson, John K. Wiencke, and Karl T. Kelsey. “DNA methylation arrays as surrogate measures of cell mixture distribution.” BMC bioinformatics 13, no. 1 (2012): 1.

[3] Chen, Yi-an, Mathieu Lemire, Sanaa Choufani, Darci T. Butcher, Daria Grafodatskaya, Brent W. Zanke, Steven Gallinger, Thomas J. Hudson, and Rosanna Weksberg. “Discovery of cross-reactive probes and polymorphic CpGs in the Illumina Infinium HumanMethylation450 microarray.” Epigenetics 8, no. 2 (2013): 203-209.

et al.[4] for 850K (EPIC) array data.

Note: Use –out in order to change the default output name.

Note: Use *–refactor* together with –gsave in order to generate a new version of GLINT files with the computed ReFACTor components (these will be included in the *datafile.samples.txt* file).

–k

The assumed number of cell types in the data. This is the only required argument for ReFACTor.

For example:

```
python glint.py --datafile datafile.glint --refactor --k 6
```

will compute the ReFACTor components under the assumption of 6 cell types in the data. **–covar**

Selects covariates to use in the feature selection step of ReFACTor. Considering genome-wide effectors such as batch information, gender, age, ancestry etc. is expected to improve the correlation of the ReFACTor components with the cell type composition.

For example:

```
python glint.py --datafile datafile.glint --refactor --k 6 --covar c1 c2 c3
```

will compute the ReFACTor components while accounting for the covariates c1, c2 and c3. The names of the covariates are defined by the headers in the *datafile.samples.txt* file associated with the *datafile.glint*. For more details see GLINT files.

Note: Use the argument –covarfile in order to provide covariates that were not included in the *datafile.glint* file or in case where a textual version of the data is used rather than a *.glint* file.

–stdth

Excludes sites with standard deviation lower than a specified value. This argument results in a speed-up of ReFACTor while ignoring sites with low variability that are less likely to contain cell composition information. The default value is 0.02.

For example:

```
python glint.py --datafile datafile.glint --refactor --k 6 --stdth 0.01
```

will remove all sites with standard deviation lower than 0.01 before computing the ReFACTor components. **–t**

The number of sites to use for computing the ReFACTor components. The default value is 500.

For example:

```
python glint.py --datafile datafile.glint --refactor --k 6 --t 1000
```

will compute the ReFACTor components using the top 1000 most informative sites according to ReFACTor. **–numcomp**

[4] McCartney, Daniel L., Rosie M. Walker, Stewart W. Morris, Andrew M. McIntosh, David J. Porteous, and Kathryn L. Evans. "Identification of polymorphic and off-target probe binding sites on the Illumina Infinium MethylationEPIC BeadChip." Genomics Data 9 (2016): 22-24.

The number of ReFACTor components to output. The default value is the same value given with the *–k* argument.

For example:

```
python glint.py --datafile datafile.glint --refactor --k 6 --numcomp 10
```

will output the first 10 ReFACTor components.

Note: While the argument *–k* is the assumed number of cell types in the data, *–numcomp* allows to output more or less ReFACTor components than the number specified by *–k* (in some cases the number of assumed cell types k may be captured by more than k ReFACTor components).

–fs

The type of feature selection procedure to perform in the feature selection step of ReFACTor.

- *normal* (default) - the standard feature selection as described in the ReFACTor paper.
- *controls* - the standard ReFACTor feature selection but based on the control samples only. This option requires the phenotype to be binary (case / control; the controls are assume to be coded as '0'). This option is especially favourable in case where many sites are expected to be assocaited with the phenotype of interest.
- *phenotype* - a continuous version of the *controls* feature selection. This feature selection uses the standard ReFACTor feature selection after adjusting the data for the phenotype of interest, in attempt to avoid capturing true signal of the phenotype that is independent in the cell type composition information in the data. This option is especially favourable in case where many sites are expected to be assocaited with the phenotype of interest.

For example:

```
python glint.py --datafile datafile.glint --refactor --k 6 --fs controls --pheno y1
```

will compute the ReFACTor components using the *controls* feature selection based on the phenotype y1. The names of the phenotypes are defined by the headers in the *datafile.samples.txt* file associated with the *datafile.glint*. For more details see GLINT files.

Note: Use the argument –phenofile in order to provide phenotypes that were not included in the *datafile.glint* file or in case where a textual version of the data is used rather than a *.glint* file.

Houseman

The algorithm by Houseman et al. is a reference-based method for calculating cell count estimates. This method requires reference data of cell type specific mean methylation levels of sorted cell types from the studied tissue. The default reference data is based on whole-blood data by Reinius et al.[5], according to the eature selection proposed by Koestler et al[6].

Note: Reference data currently exist for 7 leukocyte cell types only.

–houseman

[5] Reinius, Lovisa E., Nathalie Acevedo, Maaike Joerink, Göran Pershagen, Sven-Erik Dahlén, Dario Greco, Cilla Söderhäll, Annika Scheynius, and Juha Kere. "Differential DNA methylation in purified human blood cells: implications for cell lineage and studies on disease susceptibility." PloS one 7, no. 7 (2012): e41361.

[6] Koestler, Devin C., Meaghan J. Jones, Joseph Usset, Brock C. Christensen, Rondi A. Butler, Michael S. Kobor, John K. Wiencke, and Karl T. Kelsey. "Improving cell mixture deconvolution by id entifying o ptimal DNA methylation l ibraries (IDOL)." BMC bioinformatics 17, no. 1 (2016): 1.

Computes cell count estimates according to the algorithm by Houseman et al. and generates an output file titled *houseman_estimates.txt*, containing cell count estimates.

For example:

```
python glint.py --datafile datafile.glint --houseman
```

will compute cell count estimates.

Note: Use –out in order to change the default output name.

Note: Use *–houseman* together with –gsave in order to generate a new version of GLINT files with the computed cell count estimates (these will be included in the *datafile.samples.txt* file).

–reference

Allows to include user-supplied reference data. This argument gets path to a file containing sites by cell types matrix of mean methylation levels (for each methylation site in each cell type). The first row should include “ID” followed by cell type names and the first column should include CpG identifiers. The default reference in GLINT contains 7 leukocyte cell types. The file can be either tab-delimited, comma-delimited or space-delimited.

For example:

```
python glint.py --datafile datafile.glint --houseman --reference reference.txt
```

will compute cell count estimates using the reference data in *reference.txt*.

Inferring population structure

Here we provide an implementation of the EPISTRUCTURE algorithm by Rahmani et al.[1] for inferring population structure from methylation without the need for genotyping data. The EPISTRUCTURE algorithm calculates components that are correlated with the ancestry information of the samples in the data by applying principal component analysis (PCA) on a set of pre-defined list of sites that were found to capture a high level of genetic information. The EPISTRUCTURE components can be added as covariates in a downstream analysis.

Note: The example commands described bellow assume that the user generated GLINT files with covariates file and phenotypes file.

Note: The reference list of sites were found based on European individuals.

[1] Rahmani, Elior, Liat Shenhav, Regev Schweiger, Paul Yousefi, Karen Huen, Brenda Eskenazi, Celeste Eng et al. “Genome-wide methylation data mirror ancestry information.” bioRxiv (2016): 066340.

EPISTRUCTURE

–epi:

Computes the EPISTRUCTURE components and generates a file titled *epistructure.pcs.txt* with the output.

For example:

```
python glint.py --datafile datafile.glint --epi
```

will compute the EPISTRUCTURE components of the data.

Note: EPISTRUCTURE leverages polymorphic sites in order to capture the genetic and therefore the ancesty information in the data better. Therefore, we recommend to avoid removing polymorphic sites (–rmpoly) before applying EPISTRUCTURE.

Note: In case of data from heterogeneous source (e.g., blood), we suggest to account for type composition (see *–covar*).

Note: Use *–epi* together with –gsave in order to generate a new version of GLINT files with the computed EPISTRUCTURE components (these will be included in the *datafile.samples.txt* file).

Note: Use –out in order to change the default output name.

–covar:

Selects covariates to use in the calculation of the EPISTRUCTURE components. Considering highly dominant genome-wide effectors such as cell type composition (in case of heterogeneous tissue) is expected to improve the correlation of the EPISTRUCTURE components with the cell type composition.

For example:

```
python glint.py --datafile datafile.glint --epi --covar c1 c2 c3
```

will compute the EPISTRUCTURE components while accounting for the covariates c1, c2 and c3. The names of the covariates are defined by the headers in the *datafile.samples.txt* file associated with the *datafile.glint*. For more details see GLINT files.

Note: Use the argument –covarfile in order to provide covariates that were not included in the *datafile.glint* file or in case where a textual version of the data is used rather than a *.glint* file.

–savepcs

Selectes the number of EPISTRUCTURE components to output (default is 1).

For example:

```
python glint.py --datafile datafile.glint --epi --savepcs 2
```

will compute the first two EPISTRUCTURE components of the data.

EWAS

GLINT allows to perform epigenome-wide association study (EWAS) under several different models, as described bellow. All of the different models require using the general arguments desribed below under “General arguments”.

Note: The example commands described bellow assume that the user generated GLINT files with covariates file and phenotypes file.

General arguments

–ewas

Runs an association test on each site in the data and outputs a results file.

Note: If *–ewas* is used without additional arguments GLINT will use linear regression as a default.

Note: GLINT can produce plots based on the results file. For more details read about the –plot argument.

Note: Use –out in order to change the default output name.

–pheno

Selects a phenotype to use in the association test.

For example:

```
python glint.py --datafile datafile.glint --ewas --linreg --pheno y1
```

will run EWAS using linear regression model with the phenotype y1. The names of the phenotypes are defined by the headers in the *datafile.samples.txt* file associated with the *datafile.glint*. For more details see GLINT files.

Note: Use the argument –phenofile in order to provide phenotypes that were not included in the *datafile.glint* file or in case where a textual version of the data is used rather than a *.glint* file.

–covar

Selects covariates to use in the association test.

For example:

```
python glint.py --datafile datafile.glint --ewas --linreg --covar c1 c2 c3
```

will run EWAS using linear regression model with the covariates c1, c2 and c3. The names of the covariates are defined by the headers in the *datafile.samples.txt* file associated with the *datafile.glint*. For more details see GLINT files.

Alternatively, run:

```
python glint.py --datafile datafile.glint --ewas --linreg --covar
```

without specifying names of covariates in order to include into the model all of the covariates included in the GLINT file.

Note: Use the argument –covarfile in order to provide covariates that were not included in the *datafile.glint* file or in case where a textual version of the data is used rather than a *.glint* file.

–stdth

Excludes sites with standard deviation lower than a specified value for the EWAS analysis. This argument can be used in order to reduce the number of hypotheses by excluding near-constant sites that are not expected to be associated with phenotypes.

For example:

```
python glint.py --datafile datafile.glint --ewas --linreg --pheno y1 --stdth 0.01
```

will consider only sites with standard deviation greater than 0.01 in the EWAS analysis.

Linear regression

–linreg

Performs EWAS on the data using linear regression model. This is the default model for *–ewas*.

The output file titled *results.glint.linreg.txt* includes a list of the sites, sorted by their association p-value. The output file format includes the following columns: ID (CpG identifier), chromosome (chromosome number of the site), MAPINFO (position of the site in the genome), p-value, q-value, intercept , V1 (coefficient of the first covariate),..., Vn (coefficient of the last covaraite, beta (the coefficient of the site under test), statistic (the test statistic), UCSC_RefGene_Name (name of the gene that is closest to this site), Relation_to_UCSC_CpG_Island (category)

For example:

```
python glint.py --datafile datafile.glint --ewas --linreg --pheno y1
```

will run EWAS using linear regression model.

Logistic regression

–logreg

Performs EWAS on the data using logistic regression model. This option requires a binary phenotype (controls are assumed to be coded as ‘0’ and cases as ‘1’).

The output file titled *results.glint.logreg.txt* includes a list of the sites, sorted by their association p-value. The output file format is indentical to the once described under *–linreg*.

For example:

```
python glint.py --datafile datafile.glint --ewas --logreg --pheno y1
```

will run EWAS using logistic regression model.

Wilcoxon rank-sum test

–wilc

Performs EWAS on the data using the non-parameteric Wilcoxon rank-sum text. This option requires a binary phenotype (controls are assumed to be coded as ‘0’ and cases as ‘1’).

The output file titled *results.wilc.logreg.txt* includes a list of the sites, sorted by their association p-value. The output file format includes the following columns: ID (CpG identifier), chromosome (chromosome number of the site), MAPINFO (position of the site in the genome), p-value, q-value, statistic (the test statistic), UCSC_RefGene_Name (name of the gene that is closest to this site), Relation_to_UCSC_CpG_Island (category)

For example:

```
python glint.py --datafile datafile.glint --ewas --wilc --pheno y1
```

will run EWAS using the Wilcoxon rank-sum test.

Linear mixed model (LMM)

–lmm

Performs EWAS on the data using linear mixed model (LMM). This is an implementation of the FaST-LMM algorithm by Lippert et al.[1]

The output file named *results.glint.lmm.txt** includes a list of the sites, sorted by their association p-value. The output file includes the following columns: ID (CpG identifiers), chromosome (chromosome number of the site), MAPINFO (position of the site in the genome), p-value, q-value, intercept , V1 (coefficient of the first covariate),..., Vn (coefficient of the last covaraite, beta (the coefficient of the site under test), statistic (the test statistic), sigma-e (an estimate of sigma_e), sigma-g (an estimate of sigma_g), UCSC_RefGene_Name (name of the gene that is closest to this site), Relation_to_UCSC_CpG_Island (category) **–kinship**

The kinship matrix for modelling the inter-individual similarity in the data that is required for the LMM. GLINT allows two options:

- User-supplied kinship - users can suplly a text file with samples by samples kinship matrix (tab-delimited and with no row or column headers).

[1] Lippert, Christoph, Jennifer Listgarten, Ying Liu, Carl M. Kadie, Robert I. Davidson, and David Heckerman. “FaST linear mixed models for genome-wide association studies.” Nature methods 8, no. 10 (2011): 833-835.

- *refactor* - the ReFACTor algorithm can be used for constructing the kinship matrix. If this option is used then ReFACTor is executed for selecting the top informative sites in the data. The kinship matrix is then constructed by calculatign the empirical covariance matrix of the samples based on the selected sites.

For example:

```
python glint.py --datafile datafile.glint --ewas --lmm --pheno y1 --kinship kinship.
↪txt
```

will run EWAS using LMM with the kinship matrix specified in the *kinship.txt* file. Alternatively:

```
python glint.py --datafile datafile.glint --ewas --lmm --pheno y1 --kinship refactor -
↪-k 6
```

will use the ReFACTor algorithm for constructing the kinship matrix (where 6 is the number of assumed cell types, see the argument –k for more details).

Note: If the *refactor* option is used then all of the arguments available with the –refactor argument are also available here.

–ml

Allows to indicate whether rstricted maximum likelihood estimation (REML) or maximum likelihood estimation (ML) should be used. If this flag is supplied than ML is used, otherwise REML is used (default).

For example:

```
python glint.py --datafile datafile.glint --ewas --lmm --pheno y1 --kinship kinship.
↪txt --ml
```

will perform EWAS on the data using LMM with ML estimation.

–norm

This argument normalizes the covariates (if supplied) before fitting the LMM.

For example:

```
python glint.py --datafile datafile.glint --ewas --lmm --pheno y1 --covar c1 c2 c3 --
↪norm
```

will perform EWAS on the data using LMM after normalizing the covariates c1, c2 and c3.

–oneld

This argument allows to fit the log delta parameter in the Fast-LMM model only once under the null model (instead for each site separately).

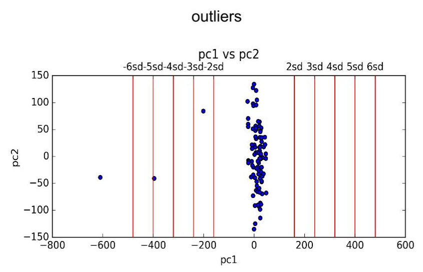
For example:

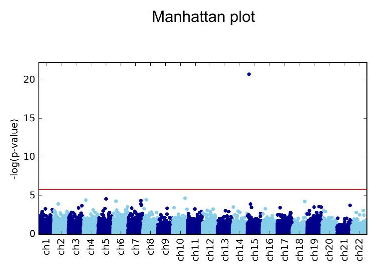
```
python glint.py --datafile datafile.glint --ewas --lmm --pheno y1 --oneld
```

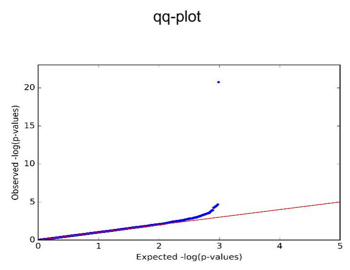
will perform EWAS on the data using LMM with a single value of log detla.

Plots

–plot







This argument allows to plot several types of plots, as desribed bellow.

Note: For using *–plot* when working on a remote server (e.g., via SSH), make sure X11-forwarding is enabled.

Note: The example commands described bellow assume that the user generated GLINT files with covariates file and phenotypes file.

Note: For each plot generated by GLINT two identical figures are saved in two different formats: *.png* file and *.eps* file. The latter is a publication quality figure.

Note: Use –out in order to change the default output name.

Outliers detection

–plotpcs

Generates scatter plots for visualizing the couples of principal components (PCs) of the data. The generated plots help in detecting outliers visualy by marking a grid of standard deviations. This plot can guide the selection of values for the –maxpcstd argument.

For example:

```
python glint.py --datafile datafile.glint --plot --plotpcs
```

will generate a figure titled *pcs_plot.png* with scatter plots of the first couple of PCs of the data (you can change the number of PCs couples to plot with the –numpcs flag as described below).

–numpcs

Allows to select the number of PCs to plot when using the *–plotpcs* argument (default is 2).

For example:

```
python glint.py --datafile datafile.glint --plot --plotpcs --numpcs 3
```

will plot scatter plots of the first 3 PCs of the data

Visualize EWAS results

The following arguments allow to generate plots based on result files generated using the –ewas argument.

Note: The *–plot* argument can be used together with –ewas in order to run EWAS and plot the results in a single command. More than one plot can be selected.

–qqplot

Generates a qq-plot from a results file generated by running –ewas.

For example:

```
python glint.py --datafile datafile.glint --plot --qqplot results.glint.linreg.txt
```

will generate a qq-plot from the results in the *results.glint.linreg.txt* file.

Alternatively, run directly with EWAS:

```
python glint.py --datafile datafile.glint --ewas --linreg --plot --qqplot
```

–manhattan

Generates a Manhattan plot from a results file generated by running –ewas.

For example:

```
python glint.py --datafile datafile.glint --plot --manhattan results.glint.linreg.txt
```

will generate Manhattan plot from the results in the *results.glint.linreg.txt* file.

Alternatively, run directly with EWAS:

```
python glint.py --datafile datafile.glint --ewas --linreg --plot --manhattan
```

CHAPTER 4

Methylation imputation:

Imputation from genotype data

GLINT allows to impute methylation levels from genotype data. Assuming genotype data are available for your samples, GLINT can impute the methylation levels of some methylation sites.

The imputation is based on the EPISTRUCTURE paper by Rahmani et al.[1]. As described in the paper, a linear model was fitted for each methylation site from cis-located SNPs in a large dataset for which both methylation and genotype data were available. A score was then defined for each site, based on the squared linear correlation of the model. Here, we use the coefficients of these linear models in order to predict methylation levels based on the SNPs. Sites having higher scores are expected to be predicted more accurately compared with sites having lower scores.

Note: The example commands described bellow assume that the user generated GLINT files with covariates file and phenotypes file.

Note: We suggest users to apply a full GWAS quality control pipeline on their genotype data before imputing metylation levels, e.g. using PLINK.

Note: The linear models for imputation were fitted based on European individuals.

Note: A/T and C/G SNPs are not used for the imputation, in order to avoid strand misspecification due to different genotyping platforms.

–impute:

[1] Rahmani, Elior, Liat Shenhav, Regev Schweiger, Paul Yousefi, Karen Huen, Brenda Eskenazi, Celeste Eng et al. "Genome-wide methylation data mirror ancestry information." bioRxiv (2016): 066340.

Imputes methylation levels from genotype data. This argument requires using 3 additional arguments for loading the genotype data in an Eigenstrat format. You can find here details about the EIGENSTRAT format.

For example:

```
python glint.py --impute --snp genotypes.snp --geno genotypes.geno --ind genotypes.ind
```

will generate GLINT files with imputed methylation data for a group of methylation sites, based on the *.snp*, *.geno* and *.ind* EIGENSTRAT files of the genotype data.

Note: –impute will automatically generate GLINT files with the imputed methylation levels, therefore there is no need to add the –gsave argument.

Note: Polymorphic CpGs according to Chen et al.[2] are not imputed. In addition, methylation levels are imputed only for CpGs for which at least one of their predicting SNPs exists in the provided genotype data.

Note: Use –out in order to change the default output name.

–score

Controls the number of methylation sites to impute. This argument specifies the minimal score required for a site in order to get imputed. The deafult value is 0.5.

For example:

```
python glint.py --impute --snp genotypes.snp --geno genotypes.geno --ind genotypes.
↪ind --score 0.4
```

will impute methylation levels for every site with a score greater than 0.4. **–maxmiss**

Controls the fraction of missing values allowed for SNPs in the EIGENSTRAT input files. SNPs with fraction of missing values above the specified value will be ignored and will not be used in the imputation. The deafult value is 0.03.

For example:

```
python glint.py --impute --snp genotypes.snp --geno genotypes.geno --ind genotypes.
↪ind --maxmiss 0.01
```

will impute methylation levels for the samples in the data using SNPs with no more than 1% of missing values.

[2] Chen, Yi-an, Mathieu Lemire, Sanaa Choufani, Darci T. Butcher, Daria Grafodatskaya, Brent W. Zanke, Steven Gallinger, Thomas J. Hudson, and Rosanna Weksberg. "Discovery of cross-reactive probes and polymorphic CpGs in the Illumina Infinium HumanMethylation450 microarray." Epigenetics 8, no. 2 (2013): 203-209.

CHAPTER 5

Versions:

Versions

The latest version of GLINT is available on github here. Below are details about each of the versions released so far.

GLINT 1.0.4

June 25, 2017

- Fixed a bug with the –kinship argument when using –lmm

GLINT 1.0.3

February 10, 2017

- Fixed a problem with –rmpoly (the problem was introduced in GLINT 1.0.2 and caused performance issues with –refactor)
- Updated automatic setup of packages for non-Anaconda users

GLINT 1.0.2

December 21, 2016

- Updated support for the Illumina EPIC array
- The following arguments were updated: –rmpoly, –rmns

GLINT 1.0.1

December 8, 2016

- Support for the Illumina EPIC array
- Additional flexibility to the format of the input files
- The following arguments were updated: –refactor, –houseman

GLINT 1.0.0

October 13, 2016

The first release of GLINT!

CHAPTER 6

Citing GLINT:

How to cite GLINT?

If you use GLINT in any published work, please cite the paper describing it:

Rahmani, Elior, Reut Yedidim, Liat Shenhav, Regev Schweiger, Omer Weissbrod, Noah Zaitlen, and Eran Halperin. "GLINT: a user-friendly toolset for the analysis of high-throughput DNA-methylation array data." Bioinformatics 2017; 33 (12): 1870-1872.

In addition:

- If you use the –refactor argument please also cite:

Rahmani, Elior, Noah Zaitlen, Yael Baran, Celeste Eng, Donglei Hu, Joshua Galanter, Sam Oh et al. "Sparse PCA corrects for cell type heterogeneity in epigenome-wide association studies." Nature methods 13, no. 5 (2016): 443-445.

- If you use the –epi argument or the –impute argument please also cite:

Rahmani, Elior, Liat Shenhav, Regev Schweiger, Paul Yousefi, Karen Huen, Brenda Eskenazi, Celeste Eng et al. "Genome-wide methylation data mirror ancestry information." Epigenetics & Chromatin 10, 1 (2017).

- If you use the –rmns argument or the –rmpoly argument for 450K data please also cite:

Chen, Yi-an, Mathieu Lemire, Sanaa Choufani, Darci T. Butcher, Daria Grafodatskaya, Brent W. Zanke, Steven Gallinger, Thomas J. Hudson, and Rosanna Weksberg. "Discovery of cross-reactive probes and polymorphic CpGs in the Illumina Infinium HumanMethylation450 microarray." Epigenetics 8, no. 2 (2013): 203-209.

- If you use the –rmns argument for 850K (EPIC) data please cite:

McCartney, Daniel L., Rosie M. Walker, Stewart W. Morris, Andrew M. McIntosh, David J. Porteous, and Kathryn L. Evans. "Identification of polymorphic and off-target probe binding sites on the Illumina Infinium MethylationEPIC BeadChip." Genomics Data 9 (2016): 22-24.

- If you use the –lmm argument please also cite:

Lippert, Christoph, Jennifer Listgarten, Ying Liu, Carl M. Kadie, Robert I. Davidson, and David Heckerman. "FaST linear mixed models for genome-wide association studies." Nature methods 8, no. 10 (2011): 833-835.

- If you use the –houseman argument please also cite:

Houseman, Eugene Andres, William P. Accomando, Devin C. Koestler, Brock C. Christensen, Carmen J. Marsit, Heather H. Nelson, John K. Wiencke, and Karl T. Kelsey. "DNA methylation arrays as surrogate measures of cell mixture distribution." BMC bioinformatics 13, no. 1 (2012): 1.

and if you use the default reference data please also cite:

Reinius, Lovisa E., Nathalie Acevedo, Maaike Joerink, Göran Pershagen, Sven-Erik Dahlén, Dario Greco, Cilla Söderhäll, Annika Scheynius, and Juha Kere. "Differential DNA methylation in purified human blood cells: implications for cell lineage and studies on disease susceptibility." PloS one 7, no. 7 (2012): e41361.

Koestler, Devin C., Meaghan J. Jones, Joseph Usset, Brock C. Christensen, Rondi A. Butler, Michael S. Kobor, John K. Wiencke, and Karl T. Kelsey. "Improving cell mixture deconvolution by id entifying o ptimal DNA methylation l ibraries (IDOL)." BMC bioinformatics 17, no. 1 (2016): 1.

CHAPTER 7

Contact us:

This software was developed by Reut Yedidim and Elior Rahmani. Code contributions were made by Omer Weissbrod and Dan coster. For any question and for reporting bugs or suggesting new features please send an email to Elior Rahmani at: elior.rahmani@gmail.com