

# Functional normalization: reproducible report

Jean-Philippe Fortin, Kasper Daniel Hansen

June 30, 2014

## 1 Introduction

---

[https://github.com/Jfortin1/funnorm\\_repro](https://github.com/Jfortin1/funnorm_repro)

```
funnormDir <- "/Users/Jean-Philippe/funnorm_repro"  
setwd("/Users/Jean-Philippe/funnorm_repro/")
```

### Scripts directory

### Cluster jobs file

The shell script `jobs.sh` contain all the ordered jobs that were submitted to the cluster.

### Extraction of the data

#### Metadata

The folder `/metadata` contains the mappings file from TCGA for the AML and KIRC datasets, for both 27k and 450k platforms. These `.csv` files contain the names of the samples, the plate information, sample tissue and histology of the samples (used as the phenotype for the subsequent analysis).

The file `clinical_patient_laml-1.txt` contains the clinical leukemia tumor subtype defined by *fixme*: *FAB*.

The file `link27kto450k.Rda` pairs the AML 27k samples to their corresponding 450k samples.

### Experimental designs

The folder `/designs` contain all the design information to construct the discovery and validation datasets used throughout the analysis. For instance, for the KIRC dataset, the design file is called

`design_kirc.Rda` and contains the sample names, the phenotypic group and the discovery/validation set identification as follows:

```
setwd(file.path(funnormDir, "designs"))
load("design_kirc.Rda")
head(design_kirc)
```

##		sampleName	group	set
##	6042316009_R02C02	6042316009_R02C02	Tumor	Validation
##	6042316009_R03C01	6042316009_R03C01	Tumor	Validation
##	6042316009_R04C01	6042316009_R04C01	Tumor	Validation
##	6042316009_R04C02	6042316009_R04C02	Tumor	Validation
##	6042316009_R06C01	6042316009_R06C01	Tumor	Validation
##	6042316009_R06C02	6042316009_R06C02	Tumor	Validation

The file `create.design.aml.R` extracts the AML tumor subtype from the file `clinical_patient_laml-1.txt` for the AML samples.

## Plate information

The directory `/plate_info` contains the physical plate information of the samples for each dataset. This information is used in the ComBat method `[[]]` and in the plate adjustment model used for the 27k data.

```
setwd(file.path(funnormDir, "plate_info"))
load("kirc_plate_450k.Rda")
head(kirc_plate_450k)
```

##		sampleName	plate
##	6042324009_R01C01	6042324009_R01C01	90
##	6042324009_R02C01	6042324009_R02C01	90
##	6042324009_R04C01	6042324009_R04C01	90
##	6042324009_R05C01	6042324009_R05C01	90
##	6042324009_R02C02	6042324009_R02C02	90
##	6042324009_R03C02	6042324009_R03C02	90

The companion script `create.plate.info.R` was used to extract the physical plate information of the samples.

## Loading the data in R

For each dataset, we read the data into *R* using the packages [minfi](#) and [methylumi](#). The first package produces an `RGChannelSet` for each dataset, while the latter package produces a `MethyLumiSet`. The reason why we use both packages to read the data is that the *noob* normalization method is implemented in the [methylumi](#) package, while all other normalization methods are compatible with [minfi](#). For

each dataset, the files are saved in the `/raw_datasets` directory. For instance, `rgset_kirc.Rda` and `methylumi_kirc.Rda` correspond to the `RGChannelSet` and `methylumiSet` for the KIRC dataset, respectively.

## Production of the discovery and validation datasets

The folder `dis_val_datasets` contains the code to create the discovery and validation datasets objects. For instance, the script `create.dis.val.methylumi.R` will produce the `MethylumiSet` object for each discovery and validation subsets of each datasets, and these objects will be saved in this directory under the name `methylumi_dis_XXX` and `methylumi_val_XXX` where `XXX` is the name of the dataset, for instance `kirc`.

## 2 Normalization

---

All the scripts to produce the normalized datasets can be found in the folder `/norm_datasets`.

### Normalization for raw, funnorm, quantile, SWAN and dasen methods

The file `create.norm.R` is used to produce the normalized datasets for the raw, funnorm, quantile, SWAN and dasen methods. It will produce files of the form `norm_dis_XXX.Rda` and `norm_val_XXX.Rda` in the same directory that contains the normalized data.

For example, the object stored in `norm_dis_kirc.Rda` is a list of 5 matrices corresponding to the normalized Beta values for each aforementioned method, for the KIRC discovery dataset.

### Normalization for noob method

Since the *noob* method requires a `MethylumiObject`, we use a different script to produce the normalized datasets. The script `create.norm.noob.R` is used to produce these datasets, and the normalized datasets are saved under the name `noob_dis_XXX.Rda` and `noob_val_XXX.Rda`.

### Normalization for funnorm + noob method

The script `create.funnorm.noob.R` generates the normalized data for the funnorm method that includes the background correction as implemented in the [methylumi](#) (noob). The normalized data can be found in the files `funnorm_noob_dis_XXX.Rda` and `funnorm_noob_val_XXX.Rda`.

## Normalization for BMIQ

Because BMIQ is extremely slow and no parallel method is implemented yet, the normalization of the data with BMIQ requires a special treatment.

The script `split.bmiq.R` will read in it `RGChannelSet`, and will split the data into individual samples, and resave the samples separately in the folder `/funnorm_repro/raw_datasets/all_samples`. This folder contains the 1460 individual samples saved in `.Rda` files (for instance, `6042308157_R06C01.Rda`).

Since BMIQ does not perform between-array normalization, we can normalize all these samples in an embarrassingly parallel way; that's what the script `create.norm.bmiq.R` does. It uses the version 1.3 of BMIQ that can be found in the `/scripts` folder.

The normalized samples will be saved in the folder `/funnorm_dir/norm_datasets/all_samples_bmiq`.

The script `merge.norm.bmiq.R` will merge back the normalized samples and will create the files `bmiq_dis_XXX.Rda` and `bmiq_val_XXX.Rda` for each dataset `XXX`.

## Merging all the normalized data

For each dataset, the script `merge.all.norm.R` will merge all the normalized data above in files called `all_norm_dis_XXX.Rda` and `all_norm_val_XXX.Rda`.

## 3 Creation of the DMPs

---

For each dataset, we create a list of differentially methylated positions (DMPs) with the corresponding dataset phenotype. The script `create.dmps.R` located in the directory `/dmps` is used to create those lists. For each dataset, the results are stored in files `dmps_dis_XXX.Rda` and `dmps_val_XXX.Rda`. Each file is a list (which each entry corresponding to a normalization method) of data frames containing the results. For instance, the top DMPs from the funnorm method for the KIRC discovery dataset can be accessed by

```
setwd(file.path(funnormDir, "dmps"))
load("dmps_dis_kirc.Rda")
head(dmps$funnorm)
```

The row names correspond to the 450k probes. The DMPs are produced with the function `dmpFinder` from the *minfi* package.

## 4 SVA analysis

---

## SVA alone

The folder `/sva_results` contain the code to run SVA *fixme: citation* on each dataset. The script `create.sva.results.R` (which calls the script `/scripts/returnDmpsFromSVA.R`) will output the results from SVA in files called `sva_results_dis_XXX.Rda` and `sva_results_val_XXX.Rda` for each dataset XXX.

The script `create.sva.dmps.R` will format the SVA results in a similar manner to the DMPs previously produced by the normalization methods so that subsequent code can be applied generally. The DMPs are saved in files called `sva_dmps_dis_XXX.Rda` and `sva_dmps_val_XXX.Rda` for each dataset XXX.

For instance, the SVA DMPs for the KIRC discovery dataset can be obtained by

```
setwd(file.path(funnormDir,"sva_results"))
load("sva_dmps_dis_kirc.Rda")
head(dmps)
```

The file `number_pcs.txt` described the number of surrogate variables found by SVA for each dataset.

## Funnorm + SVA

We also look at the combination Funnorm and SVA. The code to produce the results can be found in the directory `/sva_funnorm_results`. The scripts `create.sva.funnorm.results.R` and `create.sva.funnorm.dmps.R` and the text file `number_pcs.txt` are similar to the scripts used for SVA above.

## Noob + Funnorm + SVA

We also look at the combination Noob + Funnorm and SVA. The code to produce the results can be found in the directory `/sva_funnorm_noob_results`. The scripts `create.sva.funnorm.noob.results.R` and `create.sva.funnorm.noob.dmps.R` and the text file `number_pcs.txt` are similar to the scripts used for SVA above.

## 5 RUV analysis

---

For RUV *fixme: citation*, we used RUV-2. The scripts to run RUV can be found in `/scripts/RUV.functions`.

### RUV alone

### Funnorm + RUV

### Noob + Funnorm + RUV

## 6 ComBat analysis

---

Because ComBat cannot be applied when there is perfect confounding between batch and phenotype, we only applied the ComBat method to the following datasets: KIRC Discovery, KIRC validation and AML.

The folder `/combat_results` contain the code to produce the results from ComBat. Specifically, the script `create.combat.results.R` will output the results of ComBat in the files `combat_results_dis_kirc.Rda`, `combat_results_dis_val.Rda` and `combat_results_aml.Rda`.

Similar to the SVA script, the script `create.combat.dmps.R` will produce the DMPs list for each dataset.

## 7 Creation of the DMPs data for the 27K data

---

### KIRC data

In the folder `/kirc_27k`, the script `create.dmps.R` will produce the DMPs for the 27K samples for the KIRC dataset with a plate adjustment using the package [limma](#).

The results will be stored in the file `dmps_27k_kirc_plate_adjusted.Rda`.

### AML data

In the folder `/aml_27k`, the script `create.dmps.R` will produce the DMPs for the 27K samples for the AML dataset with a plate adjustment using the package [limma](#).

The results will be stored in the file `dmps_27k_aml_plate_adjusted.Rda`.

## 8 Creation of the Discovery/Validation ROC data

---

In the folder `/roc_data`, the scripts `create.roc.data.XXX.Rda`, where XXX is either blank, `sva`, `sva.funnorm`, `rub` or `ruv.funnorm` will produce the discovery/validation ROC data for the EBV, Blood and KIRC datasets for the normalized data, the SVA results, the funnorm + SVA results, the RUV results and the funnorm + RUV results respectively.

The results will be stored in files called `XXX_rocData_YYY_ZZZ.Rda` where XXX depends on the method, and where YYY and ZZZ depends on the dataset. For instance, `ruv_funnorm_rocData_100K_kirc.Rda` stores the ROC data from the funnorm + RUV method, with 100K loci used as the truth, for the KIRC dataset.

## 9 Creation of the concordance data

---

In the folder `/roc_data`, the script `create.overlap.data.R` will create the concordance curves data between the discovery and validation subjects for the EBV, Blood and KIRC datasets, for each method.

The results will be stored in the files `overlapData_XXX_YYY.Rda` where XXX is either 100 or 1000 (step to calculate the concordance) and YYY is the name of the dataset. For instance, the file `overlapData_100_kirc.Rda` contains the concordance curves data for the KIRC dataset, where the concordance was calculated at every 100 loci.

## 10 External validation with 27k data

---

The directory `/external_validations` contains the script to create the overlap and ROC data of the KIRC and AML datasets when the 27K data DMPs results are used as the truth.

The scripts `external.validation.kirc.R` and `external.validation.aml.R` create the concordance curves and ROC curves data for the KIRC and AML datasets, respectively.

For the concordance curves, the results are stored in the files `overlapData1K_XXX.Rda` where XXX is either `kirc_dis`, `kirc_val` or `aml`.

For the ROC curves data, the results are stored in the files `rocdData_27k_XXX` where XXX is either `kirc_dis`, `kirc_val` or `aml`.

## 11 External validation with the WGBS data

---

To assess the quality of the top DMPs for the Ontario-EBV dataset, we use an external dataset from whole-genome bisulfite sequencing as described in the manuscript. The blocks and DMPs data from the WGBS can be found at `/external_validations` under the names `blocks_ebvs_null.rda` and `dmrs_ebv_null.rda` respectively. The script `external.validation.ebv.R` will produce the overlap/replication percentages for each dataset and will save the results in the file `data.wgbs.replication.Rda`.

## 12 Sex analysis

---

### Creation of the probes X-inactivation status

The file `nature03479-s9.xls`, which was downloaded from the supplemental material at <http://www.nature.com/nature/journal/v434/n7031/full/nature03479.html>, contains the X-inactivation status that was used throughout the present paper. The script `create.sex.probes.R` extracts the information of the file `nature03479-s9.xls` to assign an X-inactivation status for each of the X-chromosome probes of the 450K array. The status is saved in the file `x.probes.status.Rda`; the

intermediate file `x.inactivation.data.Rda` is a representation of `nature03479-s9.xls` in a format readable by the R software. The script `create.x.y.probes.R` extracts the probe names for the probes mapping to the X and Y chromosome and save them in the files `chrY.Rda` and `chrX.Rda`.

### Creation of the ROC data for the Ontario-Gender dataset

The directory `/roc_data_gender` contains the script `create.roc.data.gender.R` which generates the file `rocDataGender.Rda` that contains the ROC curves data for the sex prediction using the X-inactivation status described above.

## 13 Sensitivity analysis

---

We have run `Funnorm` for  $k = 1, \dots, 10$  number of principal components. The directory `/sensitivity_analysis` contains the different scripts to produce the sensitivity analysis. The script `sensitivity.norm.R` creates the normalized dataset for each  $k$ ; the normalized data will be saved in `/sensitivity_analysis/norm_datasets`. The scripts `sensitivity.dmps.R` and `sensitivity.roc.R` create the DMPs and ROC data for each value of  $k$ , for each dataset.

## 14 Sample size simulation

---

To assess the performance of `Funnorm` for small sample sizes, we devised the following simulation scheme for the Ontario-EBV dataset. First, we kept the discovery dataset intact to make sure to have a reasonable gold standard in our discovery-validation ROC curves; we only simulated different sample sizes for the validation subset. For different sample sizes  $n \in \{10, 20, 30, 50, 80\}$ , we randomly chose half of the samples from the EBV-transformed samples, and the other half from the lymphocyte samples. For instance, for  $n = 10$ , we randomly picked 5 samples from each of the treatment groups; we repeated this subsampling  $B = 100$  times, which generated 100 ROC discovery-validation ROC curves for each  $n$ , for a total of 500 ROC curves.

For a fixed  $n$ , we took the mean of the  $B = 100$  ROC curves as well as the 0.025 and 0.975 quantiles to mimic a 95% confidence interval.

The different scripts are located in the directory `simulation_samplesize`.

The script `create.subsamples.matrices.R` creates a matrix of subsampling ids for each of  $n \in \{10, 20, 30, 50, 80\}$ , and saved the results in the file `subsamples.matrices.Rda`.

The file `create.norm.R` creates the 500 simulated validation datasets for the data normalized with `Funnorm`. The file `create.raw.R` creates the 500 simulated validation datasets for the raw data. The file `create.noob.R` creates the 500 simulated validation datasets for the data normalized with `noob`. The file `create.bmiq.R` creates the 500 simulated validation datasets for the data normalized with



BMIQ. The file `create.norm.other.R` creates the 500 simulated validation datasets for the remaining normalization methods.

The subfolders *fixme: names of the folders* contain the simulated datasets for the corresponding datasets.

## 15 Reproducible plots

---

### Main results

```
source(file.path(funnormDir, "paper_figures/generate.main.results.plots.R"));
create.main.ebv(print=FALSE)
create.main.kirc(print=FALSE)
create.main.blood(print=FALSE)
create.main.aml(print=FALSE)
```

### SVA,RUV and ComBat results

```
source(file.path(funnormDir, "paper_figures/generate.sva.ruv.results.plots.R"));
create.sva.ruv.ebv(print=FALSE)
create.sva.ruv.kirc(print=FALSE)
create.sva.ruv.blood(print=FALSE)
create.sva.ruv.aml(print=FALSE)
```

### Sex results

```
source(file.path(funnormDir, "paper_figures/generate.roc.gender.plots.R"));
create.roc.gender(print=FALSE)
```

### Additional Results

```
source(file.path(funnormDir, "paper_figures/generate.add.results.plots.R"));
create.add.ebv(print=FALSE)
create.add.kirc(print=FALSE)
create.add.blood(print=FALSE)
create.add.aml(print=FALSE)
```

### Dotplot for WGBS validation

```
source(file.path(funnormDir, "paper_figures/generate.dot.plot.R"));  
create.dot.plot(print=FALSE)
```

## ROC curves for sample size simulation

```
source(file.path(funnormDir, "paper_figures/generate.sample.size.simulation.plots.R"));  
create.sample.size.sim.figures(print=FALSE)
```

## Curves for sensitivity analysis

```
source(file.path(funnormDir, "paper_figures/generate.sensitivity.analysis.plots.R"));  
create.sens.analysis(print=FALSE)
```