TOPMed Methylation Pipeline CORE Year 3

DNA Methylation Profiling using Illumina EPIC Arrays

Ver. NHLBI.2018.10.007

Appendix E

Minfi Users Guide

Prepare Samples for Automated Batch Processing

1) Move .idat files from /auto/nor-00/mgc/projects/"project ID"/batch####/LEVEL1 to /auto/nor-00/mgc/projects/"project ID"/batch####/LEVEL1/YYYYMMDDA, where YYYYMMDDA is the date of the processing run and an alphabetic counter ("a" – "z") that represents each independent batch on a given date.
2) Place the targets.csv file into the same batch folder created under step 1 above.

Running the Automated Batch Processing

1) Log into hpcc using our dedicated node (hpc-laird.usc.edu).
2) Perform the following commands:
    a. cd /auto/uec-03/djv/dvdb/bin/R/default/bin
    b. ./R
    c. setwd("/auto/nor-00/mgc/projects/"project ID"/batch####/LEVEL1/YYYYMMDDA")
    d. source(epic3.R) #runs the scripts within the epic3.R file (example on page 291-292)
    e. q()
3) Recode the raw data file naming to match original ID and compress created files using the following command:
    a. mgcvalid /auto/nor-00/mgc/projects/"project ID"/batch####/LEVEL1 /YYYYMMDDA

    #This script reads the targets.csv file in /auto/nor-00/mgc/projects/"project ID"/batch####/LEVEL1 /YYYYMMDD to copy the idat files into a subdirectory named "level1-coded" and compresses the data files into "level2" and "level3" subdirectories.

4) Check all controls as described under "Epic Data Processing Workflow" (pgs 293-303).

Manual Batch Processing

Complete "Prepare Samples for Automated Batch Processing" and perform the steps up to 2.c of "Running the Automated Batch Processing". Instead of running the "source(epic3.R)" command, enter the commands in the epic3.R file as indicated on pages 291-292. After completion, quit R ["q()"] and proceed to steps 3 and 4 of the Automated Batch Processing protocol above.

Sharefile Uploading

Using Filezilla, transfer data for LEVEL1, LEVEL2 and LEVEL3 to the corresponding project and batch folder on the sharefile website (USCMCC.sharefile.com)

epic3.R example

```
#Call libraries
library(minfi)

#Read in data and write Raw Files

dat1 = read.csv("targets.csv")
RGset <- read.metharray.exp(targets = dat1)
RGset@annotation = c(array = "IlluminaHumanMethylationEPIC", annotation
   = "ilm10b2.hg19")
MSet.raw <-preprocessRaw(RGset)
Meth.raw <- getMeth(MSet.raw)
write.table(Meth.raw,"Meth.raw.csv", row.names=T, sep=",")
Unmeth.raw <- getUnmeth(MSet.raw)
write.table(Unmeth.raw,"Unmeth.raw.csv", row.names=T, sep=",")
ratioSet <- ratioConvert(MSet.raw, what = "both", keepCN = TRUE)
beta <- getBeta(ratioSet)
write.table(beta,"beta.raw.csv", row.names=T, sep=",")


## Detect P-values
pvalue <- detectionP(RGset, type = "m+u")
write.table(pvalue,"pvalue.csv", row.names=T, sep=",")

## Create pvalue correction matrix
pvalue_mat <- pvalue
pvalue_mat[pvalue_mat > 0.05] <- NA
pvalue_mat[pvalue_mat < 0.05] <- 0

## Preprocess Noob
MSet.noob <- preprocessNoob(RGset, offset = 15, dyeCorr = TRUE, verbose
   = TRUE)

## Generate beta values from preprocess noob
ratioSet.noob <- ratioConvert(MSet.noob, what =  "both", keepCN = TRUE)
beta.noob <- getBeta(ratioSet.noob)
write.table(beta.noob,"beta.noob.csv", row.names=T, sep=",")

## Write methylation and unmethylation values after preprocess noob
Meth.noob <- getMeth(MSet.noob)
Unmeth.noob <- getUnmeth(MSet.noob)

write.table(Meth.noob,"Meth.noob.csv", row.names=T, sep=",")
write.table(Unmeth.noob,"Unmeth.noob.csv", row.names=T, sep=",")


## Overlay correction matrix on the beta values derived from noob

noob_correct <- beta.noob + pvalue_mat
write.table(noob_correct,"noob_correct.csv", row.names=T, sep=",")
```

```
## Get SNP data and write table
rgset_snp <- getSnpBeta(RGset)
write.table(rgset_snp,"rgset_snp.csv", row.names=T, sep=",")

## Dasen normalization from preprocess noob data
d2 <- dasen(MSet.noob, NULL, onetwo, fudge = 100, ret2=FALSE)

## perform correction matrix  post dasen normalization and write table
dasen.noob.correct <- d2 + pvalue_mat
write.table(dasen.noob.correct,"dasen.noob.correct.csv", row.names=T,
sep=",")
```

EPIC Data Processing Workflow

Introduction

The following markdown is to give step by step instruction in processing the EPIC array data. A sample dataset is used to go through the workflow. An R script has been implemented the calls the same commands as described here.

Call minfi and watermelon libraries

Ensure that the most current R version is running and the epic array manifest is installed. Also set the working directory to the location that contains your data.

```
## Call in minfi package
library(wateRmelon)
library(minfi)
setwd("~/Desktop/Omar/USC/1357_Shibata IDATs")
```

Read target file

Read in the target file. Ensure that Basename is in the title.

```
## read in the target files

dat1 = read.csv("targets.csv")
dat1
```

```
##              Basename
## 1 200357150201_R01C01
## 2 200357150201_R02C01
## 3 200357150201_R03C01
## 4 200357150201_R04C01
## 5 200357150201_R05C01
## 6 200357150201_R06C01
## 7 200357150201_R07C01
## 8 200357150201_R08C01
```

Set annotation

Read in the epic arrays and set the appropriate annotation.The epic annotation has been updated.

```
## Make RGset using EPIC array manifest

RGset <- read.450k.exp(targets = dat1)
RGset@annotation = c(array = "IlluminaHumanMethylationEPIC", annotation = "ilmn10b.hg19")
RGset
```

```
## RGChannelSet (storageMode: lockedEnvironment)
## assayData: 1052641 features, 8 samples
##   element names: Green, Red
## An object of class 'AnnotatedDataFrame'
##   sampleNames: 200357150201_R01C01 200357150201_R02C01 ...
##     200357150201_R08C01 (8 total)
##   varLabels: Basename filenames
##   varMetadata: labelDescription
## Annotation
##   array: IlluminaHumanMethylationEPIC
##   annotation: ilmn10b.hg19
```

Generate Raw Values

```
## Preprocess array data to generate raw values
MSet.raw <-preprocessRaw(RGset)
```

```
## Loading required package: IlluminaHumanMethylationEPICmanifest
```

```
## Generate raw methylation values and write table of methylation values
Meth.raw <- getMeth(MSet.raw)
write.table(Meth.raw,"Meth.raw.csv", row.names=T, sep=",")

## Show sample raw methylation values

head(Meth.raw)
```

```
##            200357150201_R01C01 200357150201_R02C01 200357150201_R03C01
## cg18478105                  73                  65                 122
## cg09835024                 527                 420                 525
## cg14361672                2554                3072                5318
## cg01763666                 622                1117                1929
## cg12950382                 964                1509                1794
## cg02115394                 543                 413                 570
##            200357150201_R04C01 200357150201_R05C01 200357150201_R06C01
## cg18478105                 126                 161                 142
## cg09835024                 528                 564                 943
## cg14361672                5024                4379                5639
## cg01763666                2382                2988                3427
## cg12950382                2283                1750                1903
## cg02115394                 536                 560                 668
##            200357150201_R07C01 200357150201_R08C01
## cg18478105                 269                 182
## cg09835024                 744                1280
## cg14361672                3050                2717
## cg01763666                3191                3927
## cg12950382                1352                1316
## cg02115394                 815                1075
```

```
## Generate raw unmethylation values and write table of unmethylation values
Unmeth.raw <- getUnmeth(MSet.raw)
write.table(Unmeth.raw,"Unmeth.raw.csv", row.names=T, sep=",")

## Show sample raw unmethylation values

head(Unmeth.raw)
```

```
##            200357150201_R01C01 200357150201_R02C01 200357150201_R03C01
## cg18478105                1697                2724                3269
## cg09835024                3452                3647                3562
## cg14361672                 887                1078                 597
## cg01763666                 260                 468                 299
## cg12950382                 858                 562                 697
## cg02115394                3754                4312                4602
##            200357150201_R04C01 200357150201_R05C01 200357150201_R06C01
## cg18478105                3779                5048                6258
## cg09835024                4094                3392                3485
## cg14361672                 745                 886                1279
## cg01763666                 259                 258                 353
## cg12950382                 703                 943                1248
## cg02115394                4946                4640                4908
##            200357150201_R07C01 200357150201_R08C01
## cg18478105                8388                8780
## cg09835024                4380                4868
## cg14361672                4254                3602
## cg01763666                 447                 550
## cg12950382                1219                1435
## cg02115394                7030                7937
```

```
## Generate raw beta values and write table of beta values
ratioSet <- ratioConvert(MSet.raw, what = "both", keepCN = TRUE)
beta <- getBeta(ratioSet)
write.table(beta,"beta.raw.csv", row.names=T, sep=",")

## Show sample raw beta values

head(beta)
```

```
##            200357150201_R01C01 200357150201_R02C01 200357150201_R03C01
## cg18478105          0.04124294          0.02330584          0.03597759
## cg09835024          0.13244534          0.10327022          0.12845608
## cg14361672          0.74222610          0.74024096          0.89907016
## cg01763666          0.70521542          0.70473186          0.86579892
## cg12950382          0.52908891          0.72863351          0.72019269
## cg02115394          0.12636723          0.08740741          0.11020882
##            200357150201_R04C01 200357150201_R05C01 200357150201_R06C01
## cg18478105          0.03226633          0.03090804           0.0221875
## cg09835024          0.11423626          0.14256825           0.2129630
## cg14361672          0.87086150          0.83171890           0.8151200
## cg01763666          0.90193109          0.92051756           0.9066138
## cg12950382          0.76456798          0.64983290           0.6039353
## cg02115394          0.09777453          0.10769231           0.1197991
```

```
##            200357150201_R07C01 200357150201_R08C01
## cg18478105          0.03107312          0.02030797
## cg09835024          0.14519906          0.20819779
## cg14361672          0.41757941          0.42997310
## cg01763666          0.87713029          0.87714988
## cg12950382          0.52586542          0.47837150
## cg02115394          0.10388783          0.11928540
```

Create Pvalue Correction Matrix with Noob

Generate beta values with noob and pvalue masking

```
## Detect P-values and write P-value matrix

pvalue <- detectionP(RGset, type = "m+u")
write.table(pvalue,"pvalue.csv", row.names=T, sep=",")


## Show sample P-values
head(pvalue)
```

```
##            200357150201_R01C01 200357150201_R02C01 200357150201_R03C01
## cg18478105        0.000000e+00                   0                   0
## cg09835024        0.000000e+00                   0                   0
## cg14361672        0.000000e+00                   0                   0
## cg01763666        0.000000e+00                   0                   0
## cg12950382        1.321165e-13                   0                   0
## cg02115394        0.000000e+00                   0                   0
##            200357150201_R04C01 200357150201_R05C01 200357150201_R06C01
## cg18478105                   0                   0        0.000000e+00
## cg09835024                   0                   0        0.000000e+00
## cg14361672                   0                   0        0.000000e+00
## cg01763666                   0                   0        0.000000e+00
## cg12950382                   0                   0        8.171241e-14
## cg02115394                   0                   0        0.000000e+00
##            200357150201_R07C01 200357150201_R08C01
## cg18478105        0.000000e+00        0.000000e+00
## cg09835024        0.000000e+00        0.000000e+00
## cg14361672        0.000000e+00        0.000000e+00
## cg01763666        0.000000e+00        0.000000e+00
## cg12950382        1.270222e-10        1.392875e-11
## cg02115394        0.000000e+00        0.000000e+00
```

```
## Create pvalue correction matrix
pvalue_mat <- pvalue
pvalue_mat[pvalue_mat > 0.05] <- NA
pvalue_mat[pvalue_mat < 0.05] <- 0

## Show sample correction matrix

head(pvalue_mat)
```

```
##            200357150201_R01C01 200357150201_R02C01 200357150201_R03C01
## cg18478105                   0                   0                   0
## cg09835024                   0                   0                   0
## cg14361672                   0                   0                   0
## cg01763666                   0                   0                   0
## cg12950382                   0                   0                   0
## cg02115394                   0                   0                   0
##            200357150201_R04C01 200357150201_R05C01 200357150201_R06C01
## cg18478105                   0                   0                   0
## cg09835024                   0                   0                   0
## cg14361672                   0                   0                   0
## cg01763666                   0                   0                   0
## cg12950382                   0                   0                   0
## cg02115394                   0                   0                   0
##            200357150201_R07C01 200357150201_R08C01
## cg18478105                   0                   0
## cg09835024                   0                   0
## cg14361672                   0                   0
## cg01763666                   0                   0
## cg12950382                   0                   0
## cg02115394                   0                   0
```

```
## Preprocess Noob
MSet.noob <- preprocessNoob(RGset, offset = 15, dyeCorr = TRUE, verbose = TRUE)
```

```
## Loading required package: IlluminaHumanMethylationEPICanno.ilmn10b.hg19
```

```
## [preprocessNoob] Using sample number 5 as reference level...
```

```
## Generate beta values from preprocess noob
ratioSet.noob <- ratioConvert(MSet.noob, what =  "both", keepCN = TRUE)
beta.noob <- getBeta(ratioSet.noob)
write.table(beta.noob,"beta.noob.csv", row.names=T, sep=",")

##Show sample noob corrected beta values

head(beta.noob)
```

```
##            200357150201_R01C01 200357150201_R02C01 200357150201_R03C01
## cg18478105          0.01931773          0.01275669          0.01659820
## cg09835024          0.05965761          0.04476500          0.05596220
## cg14361672          0.81503674          0.78850603          0.95866588
## cg01763666          0.76035592          0.73718998          0.91845199
## cg12950382          0.55106923          0.82912676          0.83237527
## cg02115394          0.05695589          0.03679758          0.04587704
##            200357150201_R04C01 200357150201_R05C01 200357150201_R06C01
## cg18478105          0.01493416          0.01352883          0.01100022
## cg09835024          0.04545099          0.06133969          0.11101260
## cg14361672          0.94242005          0.91207333          0.88662651
## cg01763666          0.95060544          0.96735498          0.96417785
## cg12950382          0.87564728          0.74343539          0.67392403
## cg02115394          0.03760538          0.04313641          0.04742438
##            200357150201_R07C01 200357150201_R08C01
```

```
## cg18478105          0.01165002          0.00995885
## cg09835024          0.06132573          0.12849257
## cg14361672          0.39915008          0.41121392
## cg01763666          0.94796822          0.94472488
## cg12950382          0.54994189          0.45939066
## cg02115394          0.04140892          0.05723424
```

```
## Write methylation and unmethylation values after preprocess noob
Meth.noob <- getMeth(MSet.noob)
Unmeth.noob <- getUnmeth(MSet.noob)

#Show sample noob corrected methylation and unmethylation data

head(Meth.noob)
```

```
##            200357150201_R01C01 200357150201_R02C01 200357150201_R03C01
## cg18478105            137.3068            84.22634             94.7517
## cg09835024            342.0624           203.05636            169.2543
## cg14361672           3805.9629          3577.97313           4512.8403
## cg01763666           2297.7126          2513.44099           3208.2031
## cg12950382            998.5666          1525.62590           1186.2028
## cg02115394            357.8441           198.89422            184.4914
##            200357150201_R04C01 200357150201_R05C01 200357150201_R06C01
## cg18478105            71.68803            66.93217            54.78032
## cg09835024           138.50691           183.38674           306.60891
## cg14361672          3667.20677          3778.71890          4316.95403
## cg01763666          2905.55003          2789.07236          2571.79731
## cg12950382          1432.19112          1188.56986          1088.04766
## cg02115394           140.81018           181.94086           183.46793
##            200357150201_R07C01 200357150201_R08C01
## cg18478105            81.97469            64.47596
## cg09835024           228.55466           610.12152
## cg14361672          2245.10898          2019.06510
## cg01763666          2450.63148          2700.66277
## cg12950382           648.17969           643.64092
## cg02115394           258.98679           434.77795
```

```
head(Unmeth.noob)
```

```
##            200357150201_R01C01 200357150201_R02C01 200357150201_R03C01
## cg18478105           6970.5045           6518.2959           5613.8025
## cg09835024           5391.6973           4332.9953           2855.1855
## cg14361672            863.7197            959.6879            194.5770
## cg01763666            724.1783            896.0478            284.8516
## cg12950382            813.4863            314.4135            238.8789
## cg02115394           5924.9844           5206.1949           3836.9400
##            200357150201_R04C01 200357150201_R05C01 200357150201_R06C01
## cg18478105           4728.5830           4880.44187          4925.14700
## cg09835024           2908.8833           2806.30463          2455.32001
## cg14361672            224.0589            364.28009           552.01165
## cg01763666            150.9757             94.12193            95.55012
## cg12950382            203.3888            410.18355           526.44836
## cg02115394           3603.6054           4035.86189          3685.17387
```

```
##            200357150201_R07C01 200357150201_R08C01
## cg18478105           6954.4675           6409.7616
## cg09835024           3498.3422           4138.1803
## cg14361672           3379.6148           2890.9465
## cg01763666            134.5095            158.0137
## cg12950382            530.4534            757.4344
## cg02115394           5995.3858           7161.6877
```

```
write.table(Meth.noob,"Meth.noob.csv", row.names=T, sep=",")
write.table(Unmeth.noob,"Unmeth.noob.csv", row.names=T, sep=",")


## Overlay correction matrix on the beta values derived from noob

noob_correct <- beta.noob + pvalue_mat
write.table(noob_correct,"noob_correct.csv", row.names=T, sep=",")
```

Dasen with Pvalue Matrix and SNP data Using wateRmelon

```
## Dasen normalization from preprocess noob data
d2 <- dasen(MSet.noob, NULL, onetwo, fudge = 100, ret2=FALSE)

## Show sample Dasen normalization
head(d2)
```

```
##            200357150201_R01C01 200357150201_R02C01 200357150201_R03C01
## cg18478105          0.01622481          0.01357231          0.01739815
## cg09835024          0.08440469          0.06858798          0.04917116
## cg14361672          0.75807671          0.73774712          0.81845112
## cg01763666          0.69343980          0.67555925          0.62958055
## cg12950382          0.51038210          0.72619624          0.33515594
## cg02115394          0.08338308          0.05543143          0.04313659
##            200357150201_R04C01 200357150201_R05C01 200357150201_R06C01
## cg18478105          0.01723222          0.01787026          0.01470536
## cg09835024          0.04399250          0.06996798          0.14905644
## cg14361672          0.71969966          0.90461819          0.89735266
## cg01763666          0.74619590          0.92147020          0.91853052
## cg12950382          0.38436865          0.76124414          0.72036210
## cg02115394          0.03806730          0.04279483          0.04398923
##            200357150201_R07C01 200357150201_R08C01
## cg18478105          0.01605487           0.0161077
## cg09835024          0.09089816           0.2364627
## cg14361672          0.48227341           0.5376653
## cg01763666          0.90793098           0.9107254
## cg12950382          0.65902233           0.6474996
## cg02115394          0.06949152           0.1120911
```

```
## perform correction matrix  post dasen normalization and write table
dasen.noob.correct <- d2 + pvalue_mat
write.table(dasen.noob.correct,"dasen.noob.correct.csv", row.names=T, sep=",")
```

```
## Get SNP data and write table
rgset_snp <- getSnpBeta(RGset)
write.table(rgset_snp,"rgset_snp.csv", row.names=T, sep=",")


## Show sample SNP data

head(rgset_snp)
```

```
##           200357150201_R01C01 200357150201_R02C01 200357150201_R03C01
## rs2468330           0.38248130           0.43572216           0.31645771
## rs877309            0.02192658           0.02468856           0.32495512
## rs2857639           0.04991243           0.05555556           0.33714940
## rs798149            0.77090069           0.84028683           0.02227558
## rs939290            0.58162268           0.64504101           0.75619195
## rs9839873           0.72689076           0.80461032           0.80920060
##           200357150201_R04C01 200357150201_R05C01 200357150201_R06C01
## rs2468330            0.2853437           0.05387118            0.0657129
## rs877309             0.4016548           0.87402947            0.8864721
## rs2857639            0.3659469           0.81867846            0.7629131
## rs798149             0.0247578           0.87769784            0.8842497
## rs939290             0.7300310           0.45873418            0.5345129
## rs9839873            0.8510020           0.53830796            0.5725774
##           200357150201_R07C01 200357150201_R08C01
## rs2468330           0.58818645           0.57256795
## rs877309            0.55189277           0.58214528
## rs2857639           0.39756340           0.38603426
## rs798149            0.03865717           0.05204825
## rs939290            0.62479097           0.63945716
## rs9839873           0.45874062           0.46590241
```

Density Plots Showing Normalization Process

```
densityBeanPlot(beta, main="Raw Beta Values", pal="blue")
```

Certified for TOPMed Production by George J. Papanicolaou, PhD
Data Generation Group Leader and Contracting Officer's
Representative for NHLBI's Centralized Omics Resource (CORE)
Agreed to By the University of Southern California and the University of
Washington
Issued Dec 31, 2018