Code break vignette for `recountmethylation` Bioc2021 demo

Sean K. Maden, Ph.D. candidate, Oregon Health & Science University

August 04, 2021

Abstract

This vignette walks through all of the examples in the code break sessions for the `recountmethylation` Bioc2021 demo. Covered topics include how to work with DNAm array data from the HM450K or EPIC platform as `RGChannelSets`, how to obtain the database compilation files with `getdb` functions, how to convert between `SummarizedExperiment` and `h5se`/`HDF5-SummarizedExperiment` objects, and how to search in the harmonized metadata for samples of interest. Additional resources have been provided at the demo's GitHub repo.

Contents

1 Code break #1 – Working with HM450K and EPIC DNAm array data as `RGChannelSet` objects

This code break will demonstrate how to work with `RGChannelSet` objects, a subclass of `SummarizedExperiment` container object. Objects will include DNAm array data from either the HM450K or EPIC DNAm array platform. Example datasets will be loaded from `minfiData` and `minfiDataEPIC`, and `minfi` functions will be used to convert and combine the data platforms.

1.1 Install and load the dependencies

If you haven't already installed the dependencies, do this now with `BiocManager::install(c("minfi", "minfiData", "minfiDataEPIC"))`. Next, use `library` to load these:

```
library(minfi)
library(minfiData)
library(minfiDataEPIC)
```

1.2 Access the example `RGChannelSet` objects

We'll use the `data()` function to access the pre-compiled example `RGChannelSet` objects containing HM450K and EPIC DNAm array data. Next, assign these example objects to informative variables.

```
data("RGsetEx") # load the example hm450k data
data("RGsetEPIC") # load the example epic data
rg.hm450k <- RGsetEx
rg.epic <- RGsetEPIC
```

Note when you are working with your own data in the form of the IDAT intensity files, you will read the data in using one of `minfi`'s functions for this task, such as `read.metharray()` or `read.metharray.exp()`.

1.3 Inspect the `RGChannelSet` objects

View the object classes with `class()`.

```
class(rg.hm450k)
```

```
## [1] "RGChannelSet"
## attr(,"package")
## [1] "minfi"
```

```
class(rg.epic)
```

```
## [1] "RGChannelSet"
## attr(,"package")
## [1] "minfi"
```

These are both `RGChannelSet` objects, a type of `SummarizedExperiment` container used to handle DNAm array data.

You can view the annotation strings with `annotation()`.

```
annotation(rg.hm450k)
```

```
##                           array                      annotation
## "IlluminaHumanMethylation450k"                  "ilmn12.hg19"
```

```
annotation(rg.epic)
```

```
##                           array                      annotation
## "IlluminaHumanMethylationEPIC"                 "ilm10b2.hg19"
```

These annotation strings will associate the contained assays with feature annotations from the manifest, which we'll access using `getAnnotation()` below.

Inspect the dimensions of the assay data matrix with `dim()`.

```
dim(rg.hm450k)
```

```
## [1] 622399      6
```

```
dim(rg.epic)
```

```
## [1] 1052641       3
```

Note, the first dimension returned, or the number of rows, reflects the number of BeadArray assays read from the IDATs. Because more than one such assay can inform a single array probe measurement, these rows exceed the total number of array probes. Also note the column count reflects the number of samples in the assays matrix.

Next, we'll access the green channel intensities data contained in the example objects using the `getGreen()` function.

```
gm.hm450k <- getGreen(rg.hm450k)
gm.epic <- getGreen(rg.epic)
dim(gm.hm450k)
```

```
## [1] 622399      6
```

```
dim(gm.epic)
```

```
## [1] 1052641       3
```

```
class(gm.hm450k)
```

```
## [1] "matrix" "array"
```

```
class(gm.epic)
```

```
## [1] "matrix" "array"
```

The red channel intensities data can be similarly accessed using `getRed()`. Note the dimensions in the green color channel-specific intensities matrices returned by `getGreen()` reflect the total BeadArray assays as returned by `dim()`.

Next, view the array annotation, or manifest, with `getAnnotation()`.

```
anno.hm450k <- getAnnotation(rg.hm450k)
anno.epic <- getAnnotation(rg.epic)
dim(anno.hm450k)
```

```
## [1] 485512     33
```

```
dim(anno.epic)
```

```
## [1] 866836     46
```

The number of rows in these annotations reflects the total probes on each of the array platforms.

You can view the mappings of the assay measures to the probes by accessing the columns `AddressA` and `AddressB` in the annotations. These addresses reflect the assay measures, and they can be viewed in the assays matrices using `rownames()`.

```
rownames.gm <- rownames(getGreen(rg.hm450k))
head(rownames.gm)
```

```
## [1] "10600313" "10600322" "10600328" "10600336" "10600345" "10600353"
```

```
anno.filt <- anno.hm450k$AddressA %in% rownames.gm
anno.filt <- anno.filt | anno.hm450k$AddressB %in% rownames.gm
dim(anno.hm450k[anno.filt,])
```

```
## [1] 485512     33
```

1.4 Converting between platforms

When analyzing public DNAm array data, you may identify similar samples run on both the HM450K and EPIC platforms. Since 93% of HM450K probes overlap EPIC, one experimental approach is to harmonize the arrays together before analysis, to boost sample size and power.

Convert the `rg.epic` data to the HM450K platform with the `convertPlatform()` `minfi` function, and inspect its contents.

```
# declare a valid annotation string
anno.str <- "IlluminaHumanMethylation450k"
# convert epic to hm450k data
rg.converted <- convertArray(rg.epic, anno.str)
```

```
## [convertArray] Casting as IlluminaHumanMethylation450k
```

```
# inspect rg.converted
class(rg.converted)
```

```
## [1] "RGChannelSet"
## attr(,"package")
```

```
## [1] "minfi"
```

```
annotation(rg.converted)
```

```
##                           array                      annotation
## "IlluminaHumanMethylation450k"                   "ilmn12.hg19"
```

```
dim(rg.converted)
```

```
## [1] 575721      3
```

Note the data annotation was chanaged to reflect HM450K, and that the assay dimensions, or number of rows from `dim()`, are slightly lower than those in `rg.hm450k`. This reflects the fact that not all of HM450K's probes are available on the EPIC platform.

1.5 Combining data from different platforms

We can combine arrays from the same or different platforms with the `combineArrays()` `minfi` function. If one of the platforms doesn't match the platform specified by the `outType` argument, it will be converted prior to being combined.

Let's combine `rg.hm450k` and `rg.epic` as a single `RGChannelSet` set on the HM450K platform. We'll use the same annotation string as `anno.str` above.

```
rg.combined <- combineArrays(rg.epic, rg.hm450k, outType = anno.str)
```

```
## [convertArray] Casting as IlluminaHumanMethylation450k
```

```
dim(rg.combined)
```

```
## [1] 575719      9
```

The combined `RGChannelSet` now includes a subset of assays found on the HM450K platform, and the total samples, or number of columns from `dim()` is now 9.

2 Code break #2 – Obtaining, converting, and using `h5se` objects

This code break will demonstrate how to work with `h5se/HDF5-SummarizedExperiment` objects, including details about available datasets to download, and how to convert from `SummarizedExperiment` and back.

2.1 Obtaining the database compilation files with the `getdb` functions

First, load the `recountmethylation` library.

```
library(recountmethylation)
```

Next, inspect the docstrings for the `getdb` functions with `?getdb`. Note there are functions corresponding to each class of database file available, including `getdb_h5se_gr` for the `h5se/GenomicRatioSet` data, `getdb_h5se_rg` for the `h5se/RGChannelSet` data, etc.

The first argument for these functions is the platform, which can be either "hm450k" or "epic" ("hm450k" is the default). The second argument is the path to the directory to search for similarly named dataset files, and where the new download will be located. If a simliarly named file is detected, there will be prompts for whether to use the existing file, and whether to overwrite the existing file.

When the `getdb` functions run, they will display the download progress in you console. After successfully completing the download, an attempt to load the downloaded object will be made.

For this example, let's download a small example dataset.

```
getdb_h5se_test(platform = "hm450k", dfp = ".")
```

```
## Downloading database...
```

```
## Download completed.
```

```
## Loading database...
```

```
## Database file loaded.
```

```
## class: GenomicRatioSet
## dim: 8552 2
## metadata(0):
## assays(1): Beta
## rownames(8552): cg00017461 cg00077299 ... ch.22.47579720R
##   ch.22.48274842R
## rowData names(0):
## colnames(2): GSM1038308.1548799666.hlink.GSM1038308_5958154021_R01C01
##   GSM1038309.1548799666.hlink.GSM1038309_5958154021_R02C01
## colData names(19): gsm gsm_title ... predcell.Gran storage
## Annotation
##   array: IlluminaHumanMethylation450k
##   annotation: ilmn12.hg19
## Preprocessing
##   Method: NA
##   minfi version: NA
##   Manifest version: NA
```

Having successfully download this file, we can load and inspect it.

```
h5se.fname <- "remethdb-h5se_gr-test_0-0-1_1590090412"
gr.h5se.test <- loadHDF5SummarizedExperiment(h5se.fname)
dim(gr.h5se.test)
```

```
## [1] 8552    2
```

```
class(getBeta(gr.h5se.test))
```

```
## [1] "DelayedMatrix"
## attr(,"package")
## [1] "DelayedArray"
```

Note the assays, or rownames from `dim()`, reflect probes in `GenomicRatioSet` objects. Further, note

the class of the DNAm fractions (a.k.a. "Beta-values") matrix, returned by `getBeta()`, is a type of `DelayedArray` matrix.

2.2 Convert from `RGChannelSet` to `h5se`

We can convert a `SummarizedExperiment` object to a `DelayedArray`-backed `h5se` object with the `saveHDF5SummarizedExperiment()` function from the `HDF5Array` library.

Let's do this using the example HM450k `rg.hm450k` data now. We'll save this as an `h5se` object with the sub-directory name `rg_h5se`.

```
rg <- RGsetEx
h5se.dir <- "rg_h5se"
saveHDF5SummarizedExperiment(x = rg, dir = h5se.dir)
```

Note the `h5se` object is actually a directory of two files: a larger "assays.h5" file, and a smaller "se.rds" file.

```
list.files(h5se.dir)
```

```
## [1] "assays.h5" "se.rds"
```

To load the new `h5se` object, we point to the directory and use `HDF5Array`'s `loadHDF5SummarizedExperiment` function.

```
rg.h5se <- loadHDF5SummarizedExperiment(h5se.dir)
```

Inspecting `rg.h5se` shows it's still an `RGChannelSet` object, but accessing its component assay matrices returns `DelayedArray` matrices.

```
class(rg.h5se)
```

```
## [1] "RGChannelSet"
## attr(,"package")
## [1] "minfi"
```

```
class(getBeta(rg.h5se))
```

```
## Warning: 'colGrid' is deprecated.
## Use 'colAutoGrid' instead.
## See help("Deprecated")

## Warning: 'colGrid' is deprecated.
## Use 'colAutoGrid' instead.
## See help("Deprecated")
```

```
## [1] "DelayedMatrix"
## attr(,"package")
## [1] "DelayedArray"
```

```
class(getGreen(rg.h5se))
```

```
## [1] "DelayedMatrix"
```

```
## attr(,"package")
## [1] "DelayedArray"
```

2.3 Troubleshooting h5se downloads

Downloading the full h5se database files will take longer than shown in our example above. If we try to access the h5se data subdirectories too early (e.g. before both files have downlaoded), or if one of the requisite files is corrupted, we'll see an error. We can simulate this by removing one of the requisite files for rg.h5se using unlink.

```
unlink(file.path(h5se.dir, "se.rds"))
rg.h5se <- loadHDF5SummarizedExperiment(h5se.dir)
```

```
## Error in stop_if_bad_dir(dir, prefix): directory "rg_h5se" does not seem to contain an HDF5-b
##   SummarizedExperiment object previously saved with
##   saveHDF5SummarizedExperiment()
```

Let's now resave to the h5se.dir data directory with both requisite files. We'll do this by passing replace = T to the save function, since the directory already exists.

```
saveHDF5SummarizedExperiment(x = rg, dir = h5se.dir, replace = T)
```

2.4 Rapidly update sample metadata without resaving assays

For very large h5se objects, you may find you want to update the stored sample metadata without overwriting all the assays data. We can access and modify the sample metadata with the colData() function, and then update the stored metadata using HDF5Array's quickResaveHDF5SummarizedExperiment() function. Note you only need to pass the object in memory, and it will update the target h5se data directory automatically.

```
coldata <- colData(rg.h5se)
colnames(coldata)
```

```
##  [1] "Sample_Name"  "Sample_Well"  "Sample_Plate" "Sample_Group" "Pool_ID"
##  [6] "person"       "age"          "sex"          "status"       "Array"
## [11] "Slide"        "Basename"     "filenames"
```

```
colData(rg.h5se)$newvar <- "newvalues"
quickResaveHDF5SummarizedExperiment(rg.h5se)
```

Reloading the rg.h5se shows the new variable newvar was stored and successfully reloaded.

```
rg.h5se <- loadHDF5SummarizedExperiment(h5se.dir)
rg.h5se$newvar
```

```
## [1] "newvalues" "newvalues" "newvalues" "newvalues" "newvalues" "newvalues"
```

2.5 Recasting an h5se object as a regular SummarizedExperiment object

Occassionally, you will need to run a function that doesn't support the DelayedArray backend found in h5se datasets. For example, let's try to run minfi's preprocessFunnorm() function on the rg.h5se dataset.

```
minfi::preprocessFunnorm(rg.h5se)
```

```
## Error: 'preprocessFunnorm()' only supports matrix-backed minfi objects.
```

We see that an error is thrown relating to the type of data passed to the function.

In these cases, you can recast the h5se dataset, or revert it back, into a normal SummarizedExperiment type container. When working with the full-sized h5se objects, you may need to recast the data in chunks to save memory.

For this example, recast the entire rg.h5se dataset in one go. We'll isolate the basic RGChannelSet components for the Green, Red, and annotation slots in the new object.

```
gm <- as.matrix(getGreen(rg.h5se))
rm <- as.matrix(getRed(rg.h5se))
anno <- annotation(rg.h5se)
```

Now we make a new object with RGChannelSet() and store this in the rg.new variable.

```
rg.new <- RGChannelSet(Green = gm, Red = rm, annotation = anno)
```

Running the preprocess function on rg.new will now complete successfully.

```
gr.new <- minfi::preprocessFunnorm(rg.new)
```

```
## [preprocessFunnorm] Background and dye bias correction with noob
## [preprocessFunnorm] Mapping to genome
## [preprocessFunnorm] Quantile extraction
## [preprocessFunnorm] Normalization
```

Inspecting gr.new, we see this is a GenomicRatioSet that contains normalized probe-level measurements.

```
class(gr.new)
```

```
## [1] "GenomicRatioSet"
## attr(,"package")
## [1] "minfi"
```

```
annotation(gr.new)
```

```
##                          array                     annotation
## "IlluminaHumanMethylation450k"                 "ilmn12.hg19"
```

```
dim(gr.new)
```

```
## [1] 485512      6
```

```
dim(getBeta(gr.new))
```

```
## [1] 485512      6
```

This can be further recast as an h5se object using the same strategy as shown above.

3 Code break #3 – Querying and filtering samples of interest using harmonized metadata

This code break covers how to summarize, query, and filter the harmonized sample metadata found in recountmethylation's database compilation files.

3.1 Load a full-sized h5se object containing EPIC DNAm array data

This example will use an h5se dataset containing a recent compilation of EPIC DNAm array data with available IDATs from GEO. A version of this dataset will likely appear on the server as part of an upcoming paper.

Let's first load the HDF5Array library.

```
library(HDF5Array)
```

Next, let's note the information in the name of the h5se object's data directory: "remethdb_h5se_rg_epic-hm850k_merged_1621537799-1589820348_0-0-3". We can see a lot of details about the contained data. For instance, this h5se object includes the rg/RGChannelSet for the "epic" platform, including data that was merged between two files with distinct timestamps, "1621537799" and "1589820348". Informative file titles can simplify your work as you produce new compilations.

With the test h5se data directory in the current working directory, let's load and inspect the dataset.

```
h5se.fname <- "remethdb_h5se_rg_epic-hm850k_merged_1621537799-1589820348_0-0-3"
rg <- loadHDF5SummarizedExperiment(h5se.fname)
class(rg)
```

```
## [1] "RGChannelSet"
## attr(,"package")
## [1] "minfi"
```

```
annotation(rg)
```

```
##                          array                     annotation
## "IlluminaHumanMethylationEPIC"                "ilm10b2.hg19"
```

```
dim(rg)
```

```
## [1] 1052641   13835
```

We can see the assay quantity, or number of rows from dim(), reflects the total bead assay measures for the EPIC platform.

3.2 Summarize the harmonized metadata

Let's now view some of the available sample metadata in this object with `colData()`.

```
cd <- colData(rg)
colnames(cd)[1:10]
```

```
##  [1] "gsm"         "gse"         "gsm_title"   "storageinfo" "sex"
##  [6] "age"         "disease"     "tissue"      "sampletype"  "predage"
```

We can see there are columns for the sample/gsm ID ("gsm"), study/gse ID ("gse"), the sample title string ("gsm_title"), the mined storage info ("storageinfo"), the mined demographic variables for "sex" and "age", the learned tissue terms ("tissue"), the learned disease and experiment condition terms ("disease"), the most likely sample type prediction from the MetaSRA-pipeline ("sampletype"), and the predicted age ("predage").

Let's inspect the non-NA learned terms under the "tissue" variable.

```
head(cd$tissue[!cd$tissue == "NA"])
```

```
## [1] "prostate_cancer;cancer;cell_line;prostate"
## [2] "prostate_cancer;cancer;cell_line;prostate"
## [3] "cells;prostate;epithelial"
## [4] "cells;prostate;epithelial"
## [5] "fibroblast"
## [6] "fibroblast"
```

Here, we can see the uniform formatting for the learned terms: underscores instead of spaces; all lower case letters; and terms separated by semicolons. Also note the extent of detail in these learned terms varies considerably by sample.

Let's now view the most frequent terms under "tissue". We can get the vector of all unique learned terms using `unlist` on `strsplit` where we split terms on their separating semicolons. Finally, we coerce a frequency table to a data frame and sort it.

```
tissuev <- unlist(strsplit(cd$tissue, ";"))
df.tissue <- as.data.frame(table(tissuev))
df.tissue <- df.tissue[rev(order(df.tissue[,2])),]
head(df.tissue)
```

```
##             tissuev Freq
## 13            blood 4758
## 75               NA 2797
## 22           cancer 2515
## 92       peripheral 2186
## 93 peripheral_blood 2078
## 21       buffy_coat 1480
```

Viewing the top most frequent terms under "tissue", we see these mainly relate to blood and cancer samples.

3.3 Query the metadata for colon samples

Next, let's use regular expressions to precisely match a query. Here, we'll search for samples with the learned term "colon". We'll ensure this term occurs in its entirety, either by itself or embedded with other terms flanked by ";". Thus, we use the regular expression pattern "(^|;)colon($|;)" in the `grepl()` function for this search.

```
tx.query.str <- "colon"
tx.query.str.format <- paste0("(^|;)",tx.query.str,"($|;)")
tx.filt <- grepl(tx.query.str.format, cd$tissue)
cdf <- cd[tx.filt,]
dim(cdf)
```

```
## [1] 321  54
```

We see that 321 sample records remain after filtering on "colon" under the tissue variable. Next, let's summarize the studies and samples per study in this filtered sample set.

```
df.gse <- as.data.frame(table(cdf$gse))
df.gse <- df.gse[rev(order(df.gse[,2])),]
df.gse
```

```
##        Var1 Freq
## 4 GSE132804  206
## 8 GSE166212   45
## 7 GSE149282   24
## 3 GSE123367   16
## 6 GSE144213   11
## 2 GSE122126    7
## 9  GSE98990    4
## 5 GSE135802    4
## 1 GSE116699    4
```

```
dim(df.gse)
```

```
## [1] 9 2
```

We see there are 9 total unique study records, or gse IDs, which contribute between 4 and 206 samples.

3.4 Filter studies on their sample counts

We can now apply a minimum samples filter. This can be useful if you intend to perform a study-wise bias correction, and want to ensure enough samples are contributed to yield an accurate study bias estimate.

Let's now use a filter to retain samples from studies having at least 20 colon samples.

```
min.gsm <- 20
dff.gse <- df.gse[df.gse[,2] >= min.gsm,]
dff.gse
```

```
##        Var1 Freq
## 4 GSE132804  206
## 8 GSE166212   45
## 7 GSE149282   24
```

```
sum(dff.gse[,2])
```

```
## [1] 275
```

```
cdff <- cdf[cdf$gse %in% dff.gse[,1],]
dim(cdff)
```

```
## [1] 275  54
```

We're left with 3 studies and 275 samples. Let's now summarize terms learned for the "disease" variable for this subset of colon samples.

```
dx.df <- as.data.frame(table(unlist(strsplit(cdff$disease, ";"))))
dx.df <- dx.df[rev(order(dx.df[,2])),]
dx.df
```

```
##                Var1 Freq
## 4            normal  218
## 1            cancer   52
## 2 colorectal_cancer   12
## 3                NA    5
```

We can see terms for cancer and colorectal cancer, indicating that some of these samples may be from cancer patients or tumor samples.

3.5 Annotate studies and samples by consulting the GEO records

We can now start to clarify the types of colon samples we've identified. A reasonable and time-effective way to do this is to inspect the study records in GEO from your browser, and to begin a custom annotation of the studies themselves. This is more rapid than reviewing all samples individually, as you'll often find characteristics at the study level which also apply to each of its samples.

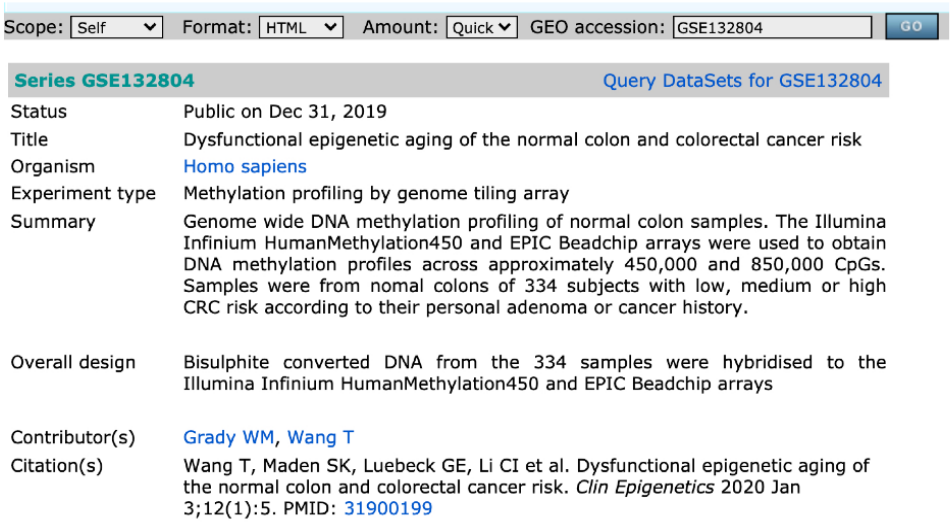
An example record for the study with gse ID GSE132804 is:

From the record, we can note this was a study of dysfuntional epigenetic aging in normal colon samples from patients with low to high colorectal cancer risk, based on their clinical record and history of cancer or adenoma (full disclosure: I was the second author on this study :^]).

If we're studying normal colon, we may or may not care if the sample came from a patient with current or prior cancer, or current or prior adenoma. These factors can be important for deciding on samples to include and exclude as you analyze the publicly available datasets.

An example study-level annotation is shown:

```
lgse <- list()
lgse[["GSE132804"]] <- list(title = "Dysfunctional epigenetic aging of the normal colon and colo
                            notes = "Subjects include low to high CRC risk, with possible adenom
```


Scope: Self Format: HTML Amount: Quick GEO accession: GSE132804 GO

Series GSE132804 Query DataSets for GSE132804

Status	Public on Dec 31, 2019
Title	Dysfunctional epigenetic aging of the normal colon and colorectal cancer risk
Organism	Homo sapiens
Experiment type	Methylation profiling by genome tiling array
Summary	Genome wide DNA methylation profiling of normal colon samples. The Illumina Infinium HumanMethylation450 and EPIC Beadchip arrays were used to obtain DNA methylation profiles across approximately 450,000 and 850,000 CpGs. Samples were from nomal colons of 334 subjects with low, medium or high CRC risk according to their personal adenoma or cancer history.
Overall design	Bisulphite converted DNA from the 334 samples were hybridised to the Illumina Infinium HumanMethylation450 and EPIC Beadchip arrays
Contributor(s)	Grady WM, Wang T
Citation(s)	Wang T, Maden SK, Luebeck GE, Li CI et al. Dysfunctional epigenetic aging of the normal colon and colorectal cancer risk. *Clin Epigenetics* 2020 Jan 3;12(1):5. PMID: 31900199

Figure 1: GSE record for GSE132804

```
lgse[["GSE149282"]] <- list(title = "Genome-wide open chromatin methylome profiles in colorectal
                            notes = "Study in CRC patients, includes cancer tissues and normal t
lgse[["GSE166212"]] <- list(title = "Colorectal Cancer DNA Methylation",
                            notes = "Study in CRC tumors, includes demographics and tumor type i
```

3.6 Compare a sample's harmonized metadata with its online record in GEO

With GEO opened in a browser, we can also review a selection of samples and check the learned and harmonized metadata against data available in the online record. Notice for the sample with gsm ID GSM5065983, the GEO record looks like:

```
gsm.row <- cdff[cdff$gsm == "GSM5065983",]
gsm.row$tissue
```

```
## [1] "cancer;tumor;colorectal;intestine;colon;rectum"
```

```
gsm.row$disease
```

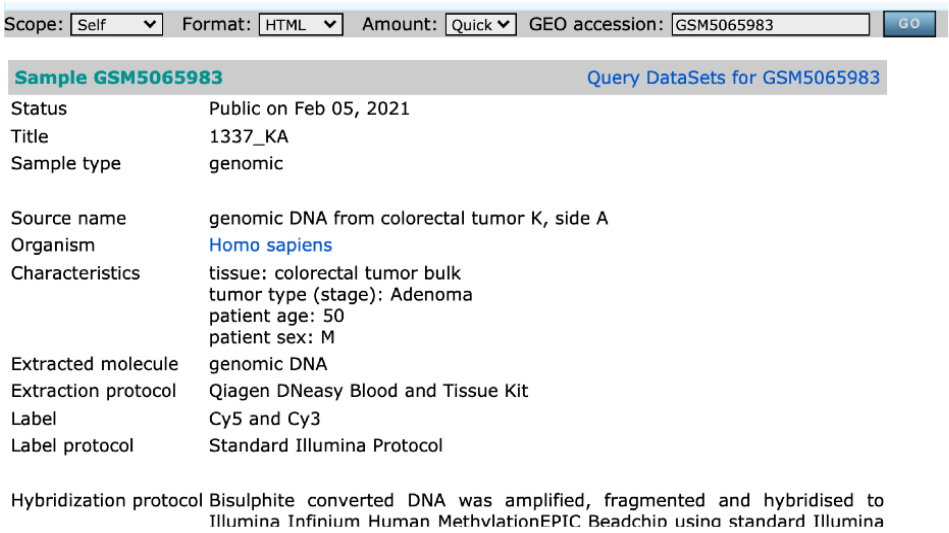
```
## [1] "cancer"
```

```
gsm.row$sex
```

```
## [1] "M"
```

```
gsm.row$age
```

```
## [1] "age_val:50;age_info:NA"
```


Scope: Self Format: HTML Amount: Quick GEO accession: GSM5065983 GO

Sample GSM5065983 Query DataSets for GSM5065983

Status	Public on Feb 05, 2021
Title	1337_KA
Sample type	genomic
Source name	genomic DNA from colorectal tumor K, side A
Organism	Homo sapiens
Characteristics	tissue: colorectal tumor bulk tumor type (stage): Adenoma patient age: 50 patient sex: M
Extracted molecule	genomic DNA
Extraction protocol	Qiagen DNeasy Blood and Tissue Kit
Label	Cy5 and Cy3
Label protocol	Standard Illumina Protocol
Hybridization protocol	Bisulphite converted DNA was amplified, fragmented and hybridised to Illumina Infinium Human MethylationEPIC Beadchip using standard Illumina

Figure 2: Example GSM record for GSM5065983

Comparing the learned versus stored metadata, we see terms for sex and age appear accurate, and there's an "NA" value for "age_info", reflecting that the numeric age units couldn't be mined for this sample. Learned terms in the "tissue" and "disease" fields further convey this is a colorectal cancer sample, although whether it precisely originates from colon or rectum is unclear.

3.7 Filter samples according to the presence of the "cancer" term

Finally, we can reuse the above approach exclude samples with "cancer" learned under the "tissue" variable. This leaves us with 223 colon samples.

```
tx.normal.filt <- !grepl("(^|;)cancer($|;)", cdff$tissue)
cdfff <- cdff[tx.normal.filt,]
dim(cdfff)
```

```
## [1] 223  54
```

4 Next steps

For additional help with analysis of DNAm array data in the R environment with Bioconductor libraries, you can consult the documentation for the `minfi`, `HDF5Array`, `DelayedArray`, and `recountmethylation` libraries. Additional resources have been provided in the wiki at the GitHub repo for this software demo.

5 Citations

If you use `recountmethylation` in published work, please cite the following paper:

Sean K Maden, Reid F Thompson, Kasper D Hansen, Abhinav Nellore, Human methylome variation across Infinium 450K data on the Gene Expression Omnibus, NAR Genomics and Bioinformatics, Volume 3, Issue 2, June 2021, lqab025, https://doi.org/10.1093/nargab/lqab025

This document uses the pdf template available at github.com/svmiller/svm-r-markdown-templates/

Credit is further due to the authors of the Bioconductor libraries featured above, including `minfi` and `HDF5Array`. Their citations are viewable using the `citation()` function.

```
citation("minfi")
```

```
##
## The minfi package is described in:
##
## Aryee MJ, Jaffe AE, Corrada-Bravo H, Ladd-Acosta C, Feinberg AP, Hansen
## KD, Irizarry RA (2014). "Minfi: A flexible and comprehensive
## Bioconductor package for the analysis of Infinium DNA Methylation
## microarrays." _Bioinformatics_, *30*(10), 1363-1369. doi:
## 10.1093/bioinformatics/btu049 (URL:
## https://doi.org/10.1093/bioinformatics/btu049).
##
## SWAN normalization (preprocessSWAN) is described in:
##
## Maksimovic J, Gordon L, Oshlack A (2012). "SWAN: Subset quantile
## Within-Array Normalization for Illumina Infinium HumanMethylation450
## BeadChips." _Genome Biology_, *13*(6), R44. doi:
## 10.1186/gb-2012-13-6-r44 (URL:
## https://doi.org/10.1186/gb-2012-13-6-r44).
##
## Funnorm normalization (preprocesFunnorm) is described in:
##
## Fortin J, Labbe A, Lemire M, Zanke BW, Hudson TJ, Fertig EJ, Greenwood
## CM, Hansen KD (2014). "Functional normalization of 450k methylation
## array data improves replication in large cancer studies." _Genome
## Biology_, *15*(12), 503. doi: 10.1186/s13059-014-0503-2 (URL:
## https://doi.org/10.1186/s13059-014-0503-2).
##
## noob background correction (preprocessNoob) is described in:
##
## Triche TJ, Weisenberger DJ, Van Den Berg D, Laird PW, Siegmund KD
## (2013). "Low-level processing of Illumina Infinium DNA Methylation
## BeadArrays." _Nucleic Acids Research_, *41*(7), e90. doi:
## 10.1093/nar/gkt090 (URL: https://doi.org/10.1093/nar/gkt090).
##
## Reconstruction of A/B compartments (compartments, extractAB) is
```

```
## described in:
##
## Fortin J, Hansen KD (2015). "Reconstructing A/B compartments as
## revealed by Hi-C using long-range correlations in epigenetic data."
## _Genome Biology_, *16*, 180. doi: 10.1186/s13059-015-0741-y (URL:
## https://doi.org/10.1186/s13059-015-0741-y).
##
## Gap hunting is described in:
##
## Andrews SV, Ladd-Acosta C, Feinberg AP, Hansen KD, Fallin MD (2016).
## "'Gap hunting' to characterize clustered probe signals in Illumina
## methylation array data." _Epigenetics & Chromatin_, *9*, 56. doi:
## 10.1186/s13072-016-0107-z (URL:
## https://doi.org/10.1186/s13072-016-0107-z).
##
## Extending minfi to support EPIC arrays is described in:
##
## Fortin J, Triche TJ, Hansen KD (2017). "Preprocessing, normalization
## and integration of the Illumina HumanMethylationEPIC array with minfi."
## _Bioinformatics_, *33*(4). doi: 10.1093/bioinformatics/btw691 (URL:
## https://doi.org/10.1093/bioinformatics/btw691).
##
## To see these entries in BibTeX format, use 'print(<citation>,
## bibtex=TRUE)', 'toBibtex(.)', or set
## 'options(citation.bibtex.max=999)'.
```

```
citation("HDF5Array")
```

```
##
## To cite package 'HDF5Array' in publications use:
##
##   Hervé Pagès (2021). HDF5Array: HDF5 backend for DelayedArray objects.
##   R package version 1.20.0. https://bioconductor.org/packages/HDF5Array
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {HDF5Array: HDF5 backend for DelayedArray objects},
##     author = {Hervé Pagès},
##     year = {2021},
##     note = {R package version 1.20.0},
##     url = {https://bioconductor.org/packages/HDF5Array},
##   }
##
## ATTENTION: This citation information has been auto-generated from the
## package DESCRIPTION file and may need manual editing, see
## 'help("citation")'.
```

6 Session info

The session info shows the versions of R and various packages used in this vignette. Showing this is an important part of reproducible research. View this as follows:

```
sessionInfo()
```

```
## R version 4.1.0 (2021-05-18)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Mojave 10.14.6
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRblas.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats4    parallel  stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] HDF5Array_1.20.0
##  [2] rhdf5_2.35.0
##  [3] DelayedArray_0.18.0
##  [4] Matrix_1.3-3
##  [5] minfiDataEPIC_1.18.0
##  [6] IlluminaHumanMethylationEPICanno.ilm10b2.hg19_0.6.0
##  [7] IlluminaHumanMethylationEPICmanifest_0.3.0
##  [8] minfiData_0.38.0
##  [9] IlluminaHumanMethylation450kanno.ilmn12.hg19_0.6.0
## [10] IlluminaHumanMethylation450kmanifest_0.4.0
## [11] minfi_1.37.0
## [12] bumphunter_1.33.0
## [13] locfit_1.5-9.4
## [14] iterators_1.0.13
## [15] foreach_1.5.1
## [16] Biostrings_2.59.2
## [17] XVector_0.31.1
## [18] SummarizedExperiment_1.21.1
## [19] Biobase_2.51.0
## [20] MatrixGenerics_1.3.0
## [21] matrixStats_0.57.0
## [22] GenomicRanges_1.43.3
## [23] GenomeInfoDb_1.27.3
## [24] IRanges_2.25.6
## [25] S4Vectors_0.29.6
```

```
## [26] BiocGenerics_0.37.0
## [27] recountmethylation_1.2.0
##
## loaded via a namespace (and not attached):
##   [1] rjson_0.2.20              ellipsis_0.3.2
##   [3] siggenes_1.65.0           mclust_5.4.7
##   [5] base64_2.0                bit64_4.0.5
##   [7] AnnotationDbi_1.54.1      fansi_0.4.2
##   [9] xml2_1.3.2                codetools_0.2-18
##  [11] splines_4.1.0             sparseMatrixStats_1.3.2
##  [13] cachem_1.0.5              scrime_1.3.5
##  [15] knitr_1.33                Rsamtools_2.7.0
##  [17] annotate_1.69.0           dbplyr_2.0.0
##  [19] png_0.1-7                 readr_1.4.0
##  [21] compiler_4.1.0            httr_1.4.2
##  [23] assertthat_0.2.1          fastmap_1.0.1
##  [25] limma_3.47.3              htmltools_0.5.1
##  [27] prettyunits_1.1.1         tools_4.1.0
##  [29] glue_1.4.2                GenomeInfoDbData_1.2.4
##  [31] dplyr_1.0.6               rappdirs_0.3.1
##  [33] doRNG_1.8.2               Rcpp_1.0.6
##  [35] vctrs_0.3.8               rhdf5filters_1.3.3
##  [37] multtest_2.47.0           preprocessCore_1.53.1
##  [39] nlme_3.1-152              rtracklayer_1.51.4
##  [41] DelayedMatrixStats_1.13.3 xfun_0.23
##  [43] stringr_1.4.0             lifecycle_1.0.0
##  [45] restfulr_0.0.13           rngtools_1.5
##  [47] XML_3.99-0.5              beanplot_1.2
##  [49] zlibbioc_1.37.0           MASS_7.3-54
##  [51] hms_1.0.0                 GEOquery_2.59.0
##  [53] RColorBrewer_1.1-2        yaml_2.2.1
##  [55] curl_4.3                  memoise_2.0.0
##  [57] biomaRt_2.47.1            reshape_0.8.8
##  [59] stringi_1.6.2             RSQLite_2.2.2
##  [61] genefilter_1.74.0         BiocIO_1.1.2
##  [63] GenomicFeatures_1.43.3    filelock_1.0.2
##  [65] BiocParallel_1.25.2       rlang_0.4.11
##  [67] pkgconfig_2.0.3           bitops_1.0-6
##  [69] nor1mix_1.3-0             evaluate_0.14
##  [71] lattice_0.20-44           purrr_0.3.4
##  [73] Rhdf5lib_1.13.0           GenomicAlignments_1.27.2
##  [75] bit_4.0.4                 tidyselect_1.1.1
##  [77] plyr_1.8.6                magrittr_2.0.1
##  [79] R6_2.5.0                  generics_0.1.0
##  [81] DBI_1.1.1                 pillar_1.6.1
##  [83] survival_3.2-11           KEGGREST_1.32.0
##  [85] RCurl_1.98-1.2            tibble_3.1.2
##  [87] crayon_1.4.1              utf8_1.2.1
```

```
##  [89] BiocFileCache_1.15.1      rmarkdown_2.9
##  [91] progress_1.2.2            grid_4.1.0
##  [93] data.table_1.13.6         blob_1.2.1
##  [95] digest_0.6.27             xtable_1.8-4
##  [97] tidyr_1.1.2               illuminaio_0.33.0
##  [99] openssl_1.4.3             quadprog_1.5-8
## [101] askpass_1.1
```