Killer Whale Methylation Normalization

Caila Kucheravy

2024-01-04

Prep Killer whale normalization code.

Link to data normalization tutorial: https://figshare.com/articles/online_resource/Data_Normalization_Tutorial/13526573/1

The code below was modified from the link above and from code received from Levi Newediuk.

Another helpful link: https://bioconductor.org/packages/devel/bioc/vignettes/minfi/inst/doc/minfi.html

Install libraries and prep workspace:

```r
rm(list=ls())

setwd("~/Documents/Master's/Analysis/Epigenetic Aging/Killer Whales")

# Install packages "sesame" and "minfi" from Bioconductor:
# BiocManager::install("sesame")
# BiocManager::install("minfi")
#install.packages("SeSaMeRpackages/sesame_1.3.0.tar.gz", repos=NULL, type="source")
#install.packages("SeSaMeRpackages/HorvathMammalMethylChip40manifest_0.2.2.tar.gz", repos=NULL, type="s
#install.packages("SeSaMeRpackages/HorvathMammalMethylChip40anno.test.unknown_0.2.2.tar.gz", repos=NULL

library(tidyverse)
library(sesame)
library(minfi)

library(HorvathMammalMethylChip40manifest)
library(HorvathMammalMethylChip40anno.test.unknown)

options(scipen=999)
```

1. Load data and manifest

Load sample sheets and Sesame manifest file:

```r
# Sample list from array run in Meaghan Jones' lab:
sample_file_Jones     <- read.csv("Input/sample_list_whales_updated_Nov2023.csv")

# Sample list from array run in the Horvath lab:
sample_file_Horvath   <-
read.csv("Input/sample_list_whales_HORVATH_updated_Nov2023.csv")

# Manifest file:
manifest_file         <- read.csv("HorvathMammal40.CanonicalManifest.3.2019.sesame.csv")
```

Filter the manifest so that we have the RS sites:

```r
rs_sites <- manifest_file %>%
  filter(grepl('rs', Probe_ID))
```

2. Subset data to obtain only info available in .IDAT files

.idat files are the output data file from the iscan machine when reading a methylation microarray chip. idat stands for Illumina BeadArray Data. These files cannot be opened directly, but must be processed through software.

First, need to remove chip positions without data (failed to read in the iscan). We only had one on this array, in well G6.

List the chip IDs and chip positions for each of the iscan files:

```r
chip.IDs <- substr(list.files('Input/KW_BW IDAT files/', recursive = T, full.names = F), 1, 12)
chip.positions <- substr(list.files(paste0('Input/KW_BW IDAT files/'), recursive = T, full.names = F), 2
```

Each sample file should have both a red (_Red.idat) and a green (_Green.idat) file. Remove files without BOTH a red and a green file:

```r
run_samples <- data.frame(chip.ID.loc = paste(chip.IDs, chip.positions, sep = '_')) %>%
  group_by(chip.ID.loc) %>%
  summarize(n_samples = n()) %>%
  filter(! n_samples == 1)
```

Next, the sample sheet needs a column "Basename" that points to the basename of a two-colour IDAT file (i.e., _Red.idat or _Grn.idat). The Basename column needs to be added with the file path for each sample in the corresponding row. Then, use 'read.metharray.exp' to find the corresponding files using the sample sheet:

Make columns for "Basename", row, column, and lab (1 = Jones, 2 = Horvath)

```r
# Jones lab:
sample_sheet_Jones <- sample_file_Jones %>%
  # Add chip.ID.loc column specifying the chip and the stripe (location of sample on array)
  mutate(chip.ID.loc = paste(chip.id, stripe, sep = '_'),
         # Fix 'row' and 'column' columns:
         row = substr(stripe, 3, 3),
         column = substr(stripe, 6, 6),
         # Add basenames (i.e., file paths for iscan files)
         Basename = paste0('Input/KW_BW IDAT files/', chip.id, '/', chip.id, '_', stripe),
         # Add column for lab:
         lab = 1,
         # Convert chip.id column to character for merging later:
         chip.id = as.character(chip.id)) %>%
  # Filter only iscans with both red and green iscan files
  filter(chip.ID.loc %in% run_samples$chip.ID.loc)

# Horvath lab:
sample_sheet_Horvath <- sample_file_Horvath %>%
  # Add chip.id column & stripe column
  mutate(chip.id = substr(Basename, 1, 12),
         stripe = substr(Basename, 14, 19)) %>%
  # Remove Basename column
  select(-Basename) %>%
  # Add chip.ID.loc column specifying the chip and the stripe (location of sample on array)
  mutate(chip.ID.loc = paste(chip.id, stripe, sep = '_'),
         # Fix 'row' and 'column' columns:
```

```
         row = substr(stripe, 3, 3),
         column = substr(stripe, 6, 6),
         # Add basenames (i.e., file paths for iscan files)
         Basename = paste0('Input/KW IDAT files/', chip.id, '/', chip.id, '_', stripe),
          # Add column for lab:
         lab = 2) %>%
  # Filter only iscans with both red and green iscan files
  filter(chip.ID.loc %in% run_samples$chip.ID.loc)
```

Clean up and join the samples sheets:

```
# Jones lab sample sheet first:
# Select required columns:
sample_sheet_Jones2 <- sample_sheet_Jones %>%
  # Rename first column:
  dplyr::rename(Order = ...1) %>%
  # Select columns:
  select(Order, block, sampleId, Species, Year, Location, Sex,
         lab, plate, chip.id, stripe, row, column, chip.ID.loc, Basename) %>%
  # Filter for killer whales:
  filter(Species == "Orcinus orca")

# Select required columns:
sample_sheet_Horvath2 <- sample_sheet_Horvath %>%
  # rename columns to match Jones sample sheet:
  dplyr::rename(Order = OriginalOrderInBatch, sampleId = ExternalSampleID, block = Block, Species = Spe
  # Change values in 'plate' column to 3
  mutate(plate = 3) %>%
  # Select columns:
  select(Order, block, sampleId, Species, Year, Location, Sex,
         lab, plate, chip.id, stripe, row, column, chip.ID.loc, Basename)

# Combine the sample sheets:
sample_sheet <- rbind(sample_sheet_Jones2, sample_sheet_Horvath2)
```

3. Normalize data using minfi

Command *read.metharray.exp* reads entire methylation array. Create an RGChannelSet object containing the raw red green channel information from the IDAT files:

```
RGset <- minfi::read.metharray.exp(base = NULL, targets = sample_sheet, recursive = TRUE)
```

Annotate the RGset object with probe coordinates. This line is currently a placeholder, as the annotation is empty, but needs to be run anyway.

```
RGset@annotation = c(array='HorvathMammalMethylChip40', annotation="test.unknown")
```

Command *getBeta* calculates the beta from the RGChannelSet data. Beta values are continuous variables between 0 and 1, representing the percentage of methylation (ratio of the intensity of the methylated bead type to the combined locus intensity).

Beta values are computed as: *Beta = Meth / (Meth + Unmeth + offset)* The offset is chosen to avoid dividing with small values - default offset is 100.

Call getBeta on the RGset object to return a dataframe of raw beta values for each CG site - gets the ß values (DNA methylation values) at each CG site.

```
raw_betas_minfi <- as_tibble(minfi::getBeta(RGset), rownames="CGid")
```

The Noob (normal-exponential out-of-band) is a preprocessing background correction method for dye-bias normalization for Illumina Infinium methylation microarrays.

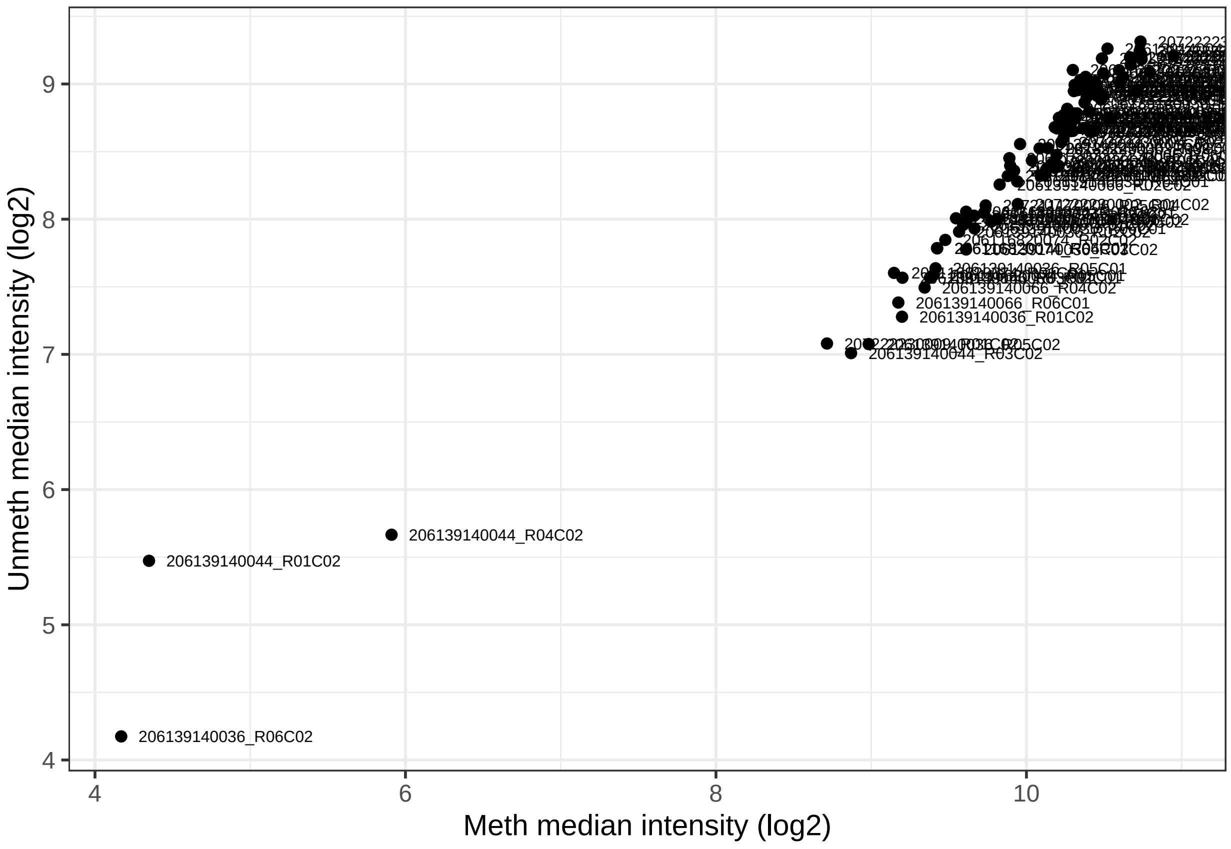
Call preprocessNoob on RGset to return a MethylSet object (with two channels: methylated and unmethylated), which contains normalized beta values along with a few other things:

```
Mset = minfi::preprocessNoob(RGset)
```

Quality Control (QC): use command *getQC* to estimate sample-specific quality control for methylation data. This produces a table with two columns, mMed and uMed (chipwide medians of the Meth and Unmeth channels). Use *plotQC*

Minfi QC plot:

```
qc <- minfi::getQC(Mset)
df_qc <- as.data.frame(qc)

ggplot(data = df_qc, aes(x = mMed, y = uMed, label = rownames(df_qc))) +
  geom_point() +
  geom_text(size = 2, hjust = -0.1) +
  theme_bw() +
  xlab("Meth median intensity (log2)") +
  ylab("Unmeth median intensity (log2)")
```



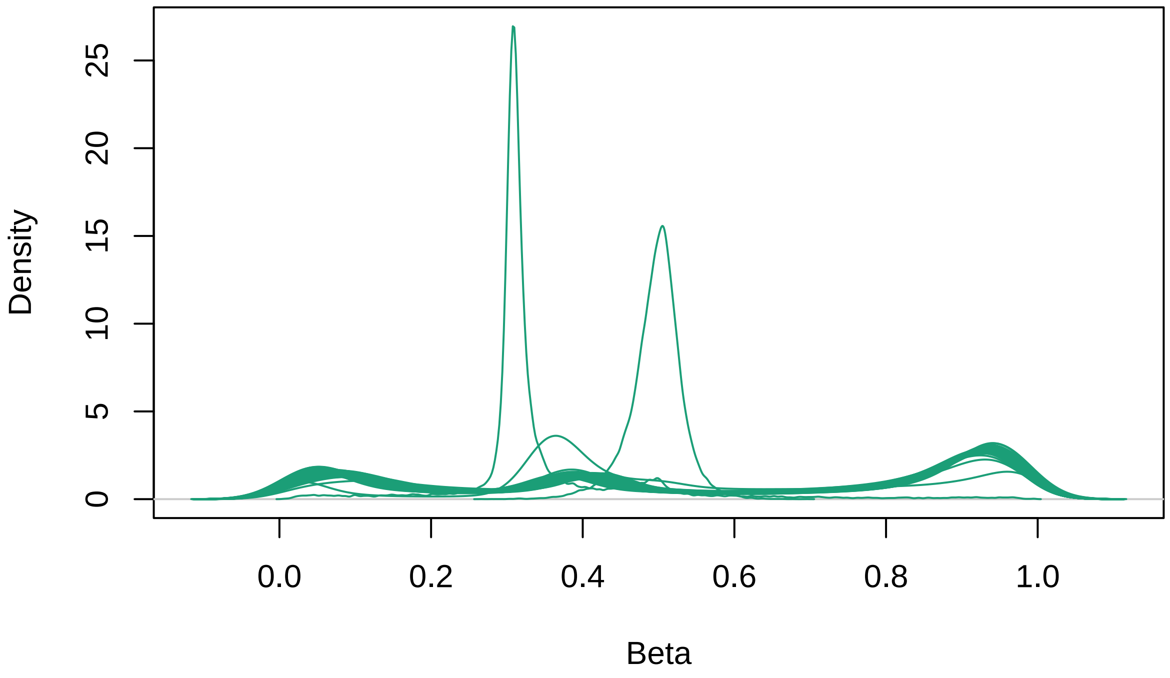
A few outliers here.

206139140044_R04C02 = ARSQ-xx-1379 206139140044_R01C02 = KW-CH-2011 206139140036_R06C02 = KW-CH-2011

All three were stranded whales.

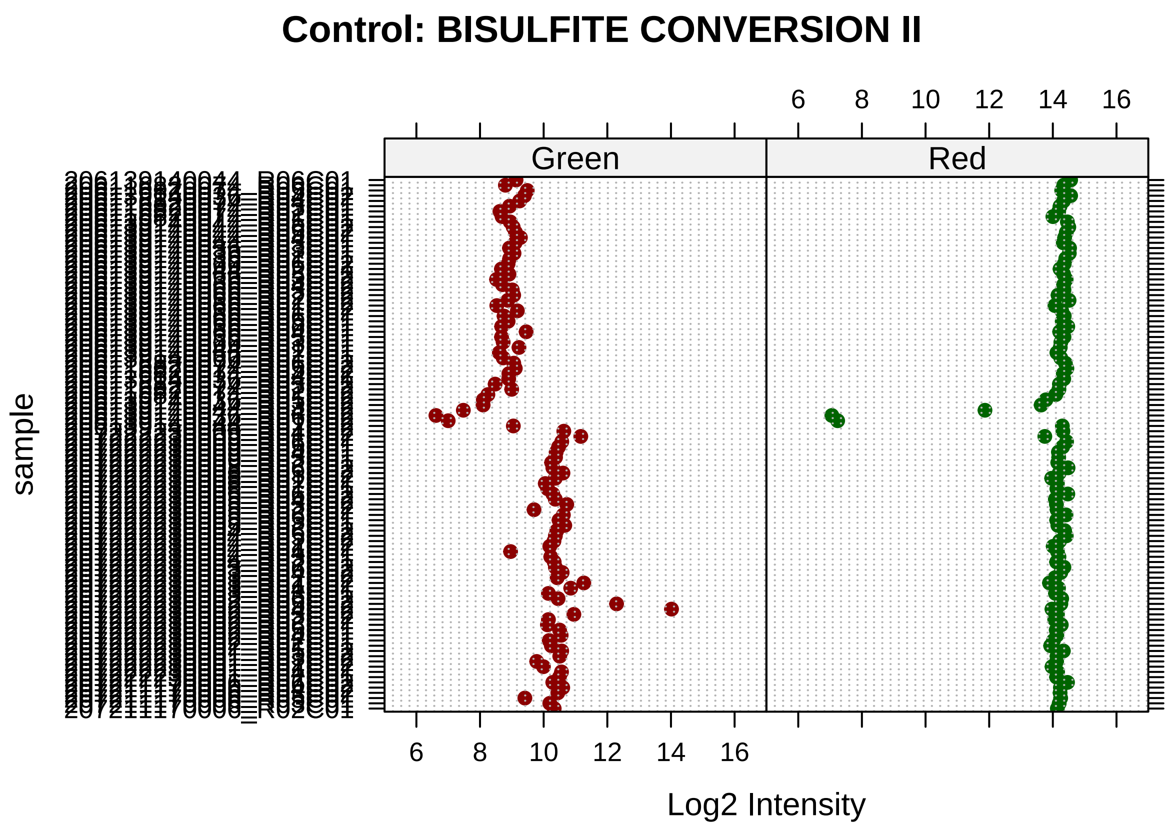
Beta value densities:

```r
minfi::densityPlot(Mset)
```



Control strip plot:

```r
minfi::controlStripPlot(RGset, controls = "BISULFITE CONVERSION II")
```



We see the three outliers in these two plots as well.

Let's remove the outliers and go through the steps again:

```
outliers <- c("206139140044_R04C02", "206139140044_R01C02", "206139140036_R06C02")

sample_sheet_outliersRemoved <- sample_sheet %>%
  filter(!chip.ID.loc %in% outliers)
```

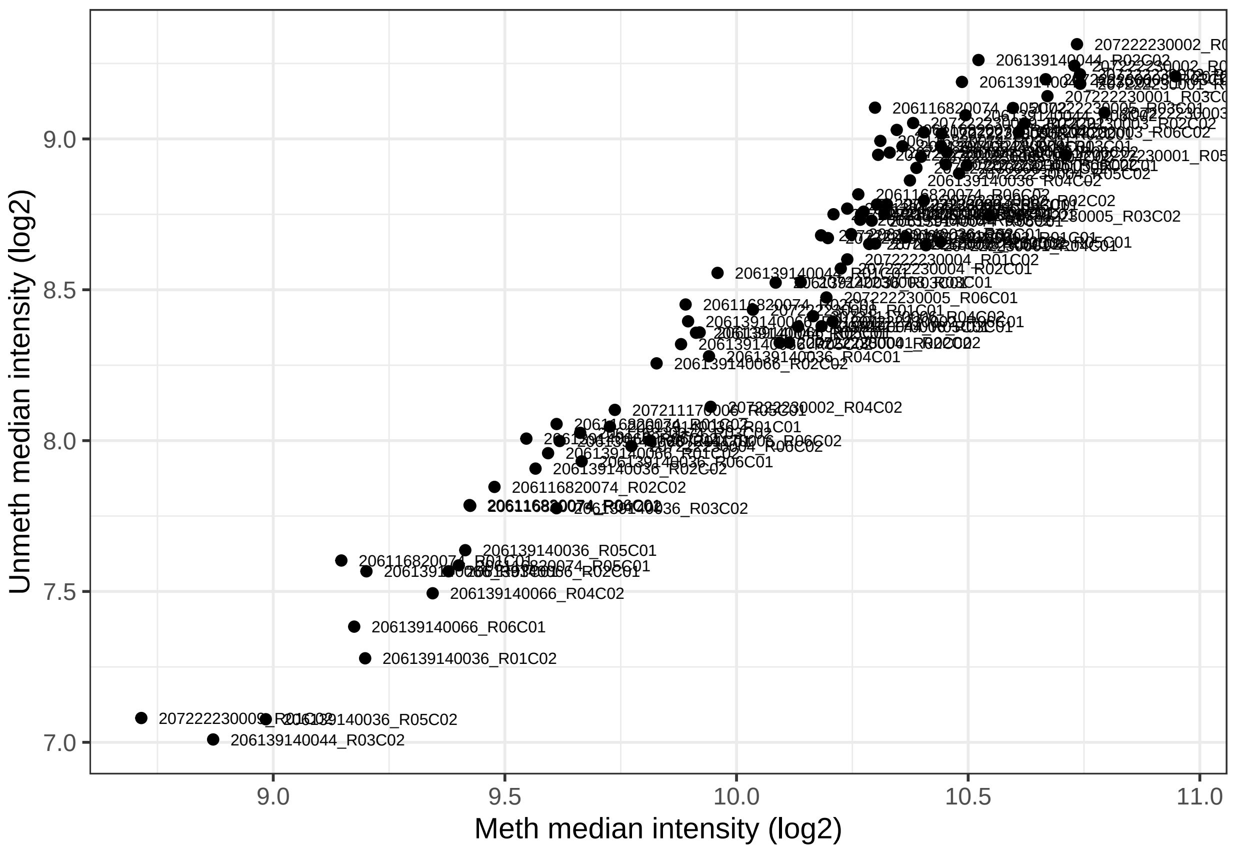
Go through processing steps again:

```
# 1. Create RGset
RGset_outliersRemoved <- minfi::read.metharray.exp(base = NULL, targets = sample_sheet_outliersRemoved,

# 2. Annotation
RGset_outliersRemoved@annotation = c(array='HorvathMammalMethylChip40', annotation="test.unknown")

# 3. Call getBeta
raw_betas_minfi_outliersRemoved <- as_tibble(minfi::getBeta(RGset_outliersRemoved), rownames="CGid")

# 4. pre-process Noob
Mset_outliersRemoved = minfi::preprocessNoob(RGset_outliersRemoved)
```

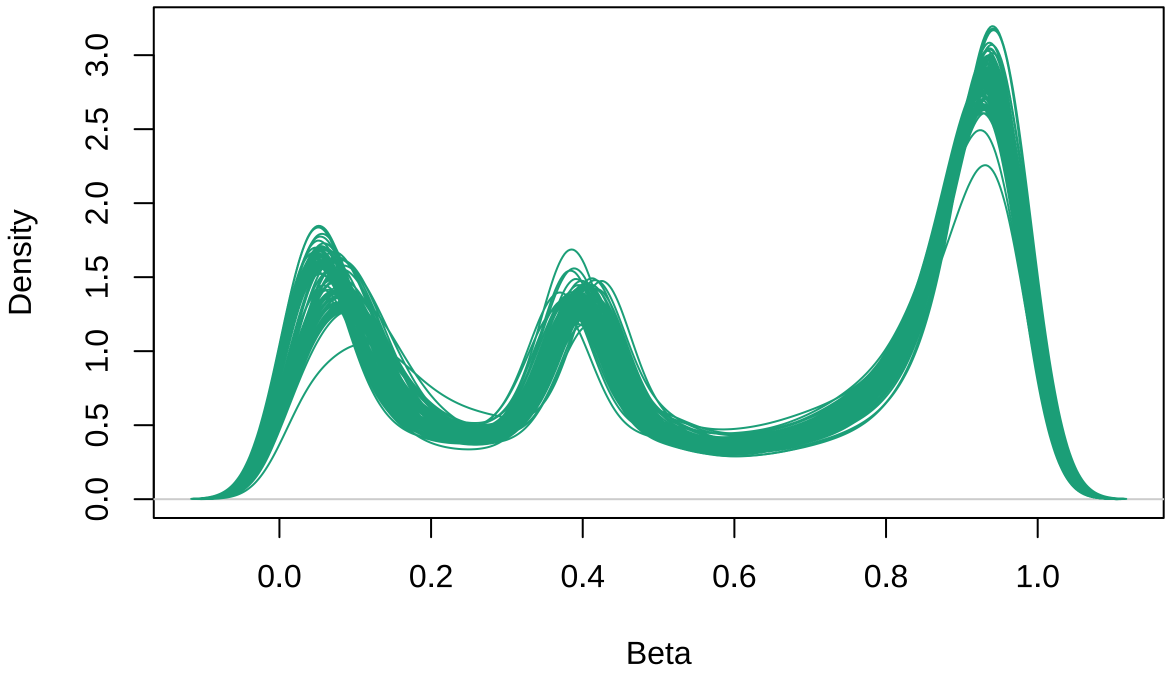
Quality check data with outliers removed:

```
# QC plot:
qc_outliersRemoved <- getQC(Mset_outliersRemoved)
df_qc_outliersRemoved <- as.data.frame(qc_outliersRemoved)

ggplot(data = df_qc_outliersRemoved, aes(x = mMed, y = uMed, label = rownames(df_qc_outliersRemoved))) +
```

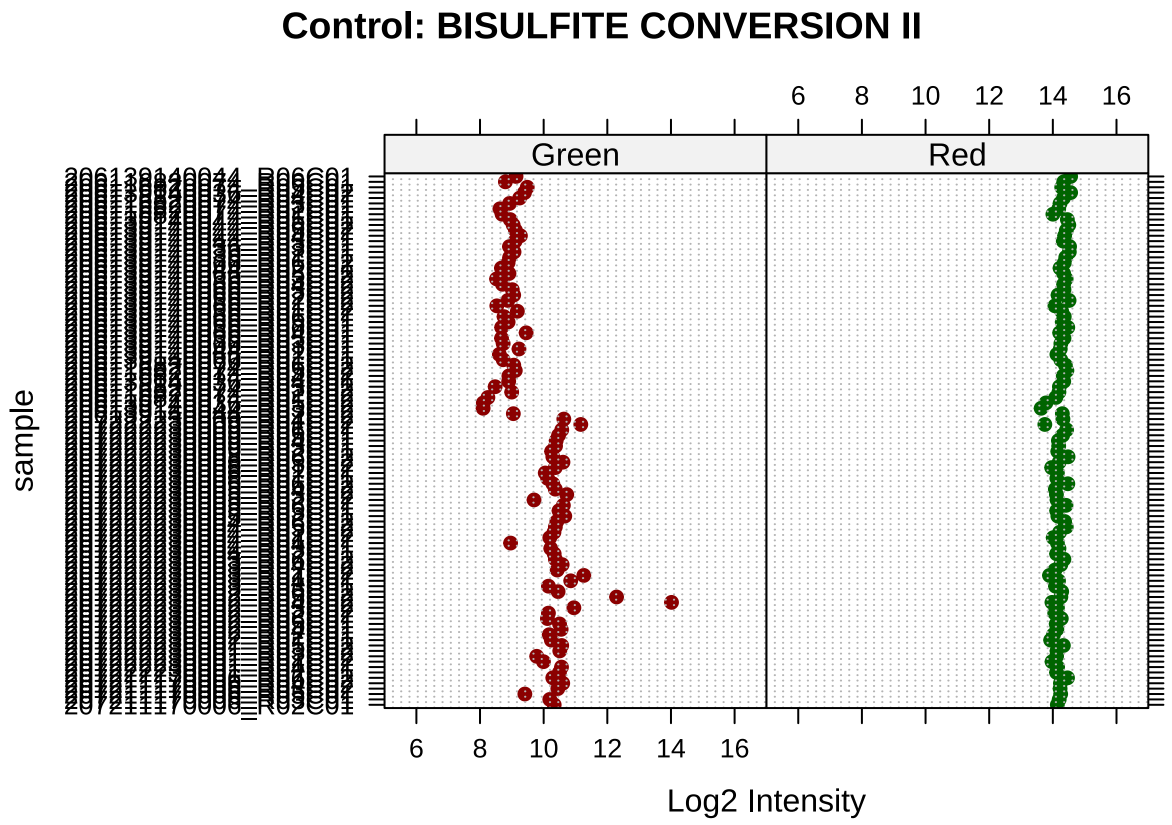
```r
  geom_point() +
  geom_text(size = 2, hjust = -0.1) +
  theme_bw() +
  xlab("Meth median intensity (log2)") + ylab("Unmeth median intensity (log2)")
```



```r
# Beta density plot:
densityPlot(Mset_outliersRemoved)
```



```
# Control strip plot:
controlStripPlot(RGset_outliersRemoved, controls = "BISULFITE CONVERSION II")
```



There are a few hanging out near the bottom still:

206139140044_R03C02 = ARSQ-xx-1397 206139140036_R05C02 = ARSQ-xx-1397 207222230009_R01C02 = ARPI-2019-01

The beta density plot looks better, but can see that there is one a little off, hanging below around beta ~ 0.1. We will leave these samples in for now, but be conscious of how they might affect normalization.

Call getBeta on Mset to return a data frame of normalized beta values for each CG Sse:

```
normalized_betas_minfi = as_tibble(minfi::getBeta(Mset_outliersRemoved), rownames="CGid")

# Make sure that all beta values fall between 0 and 1
max(normalized_betas_minfi[,2:100])
```

```
## [1] 0.9975796
```

```
min(normalized_betas_minfi[,2:100])
```

```
## [1] 0.002418525
```

Save normalized betas:

```
#saveRDS(normalized_betas_minfi, 'output/nbetas_combined_KillerWhale_array.rds')
```

4. Create transposed dataset of beta values:

Transpose matrix after removing grid:

```
n_betas_t <- normalized_betas_minfi %>%
  select(! CGid) %>%
  t() %>%
  as.data.frame()
```

Rename rest of columns to CpG sites:

```
colnames(n_betas_t) <- normalized_betas_minfi$CGid
```

Save transposed betas and update the sample sheet for clock processing.

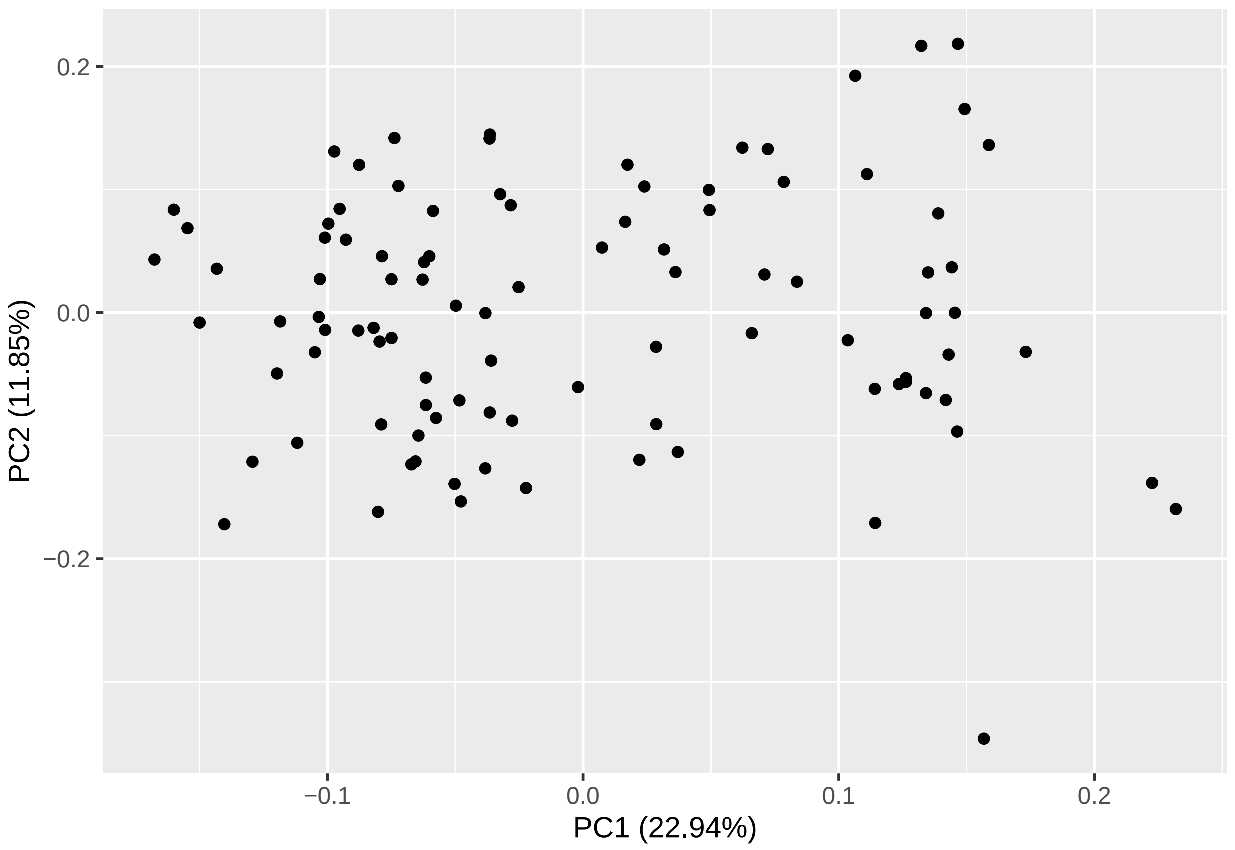
```
#saveRDS(n_betas_t, 'output/tbetas_combined_KillerWhale_array.rds')
```

Save sample sheet:

```
#saveRDS(sample_sheet_outliersRemoved, 'output/updated_sample_sheet_combined_KillerWhale_array.rds')
```

Run PCA to check out the methylation data (exploratory):

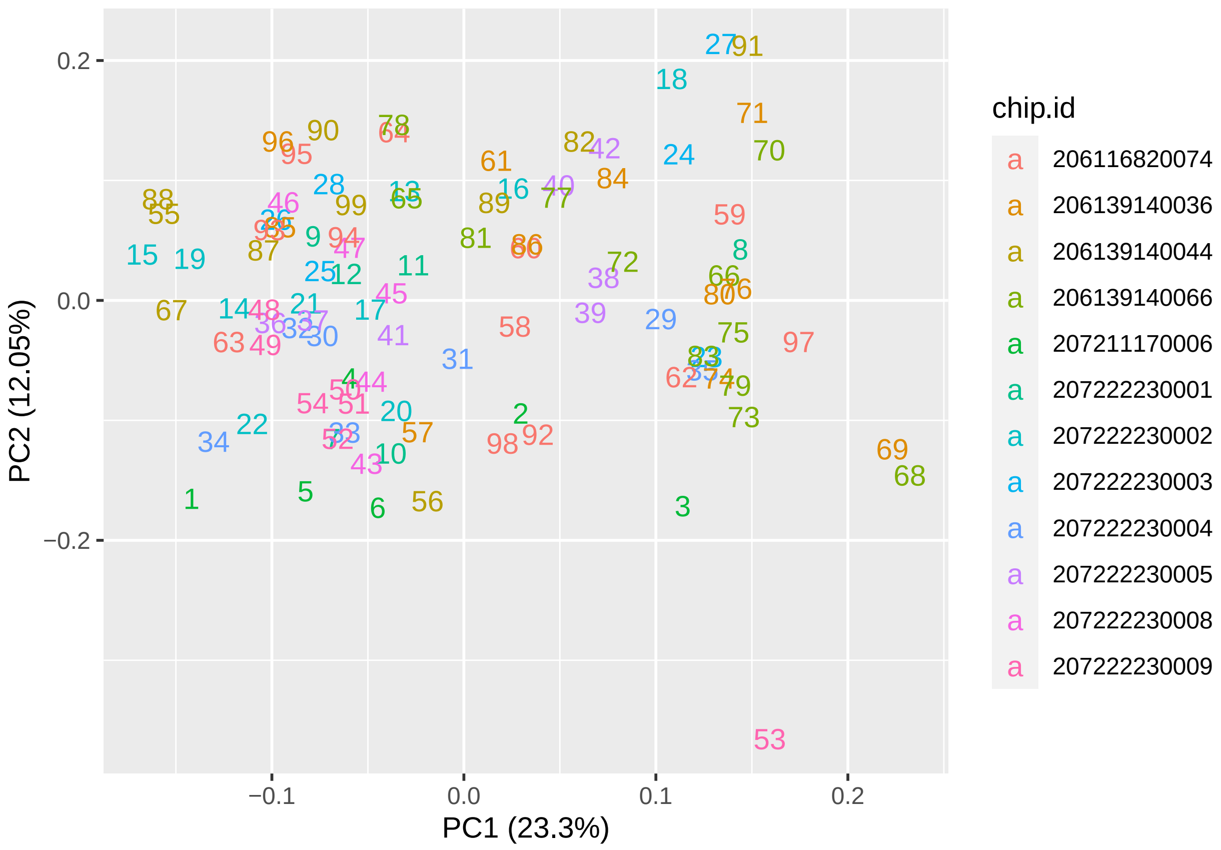
```
library(ggplot2)
library(ggfortify)
library(stats)

pca_methylData <- prcomp(n_betas_t[,1:26126])
autoplot(pca_methylData,
         #dim(label = TRUE
         )
```



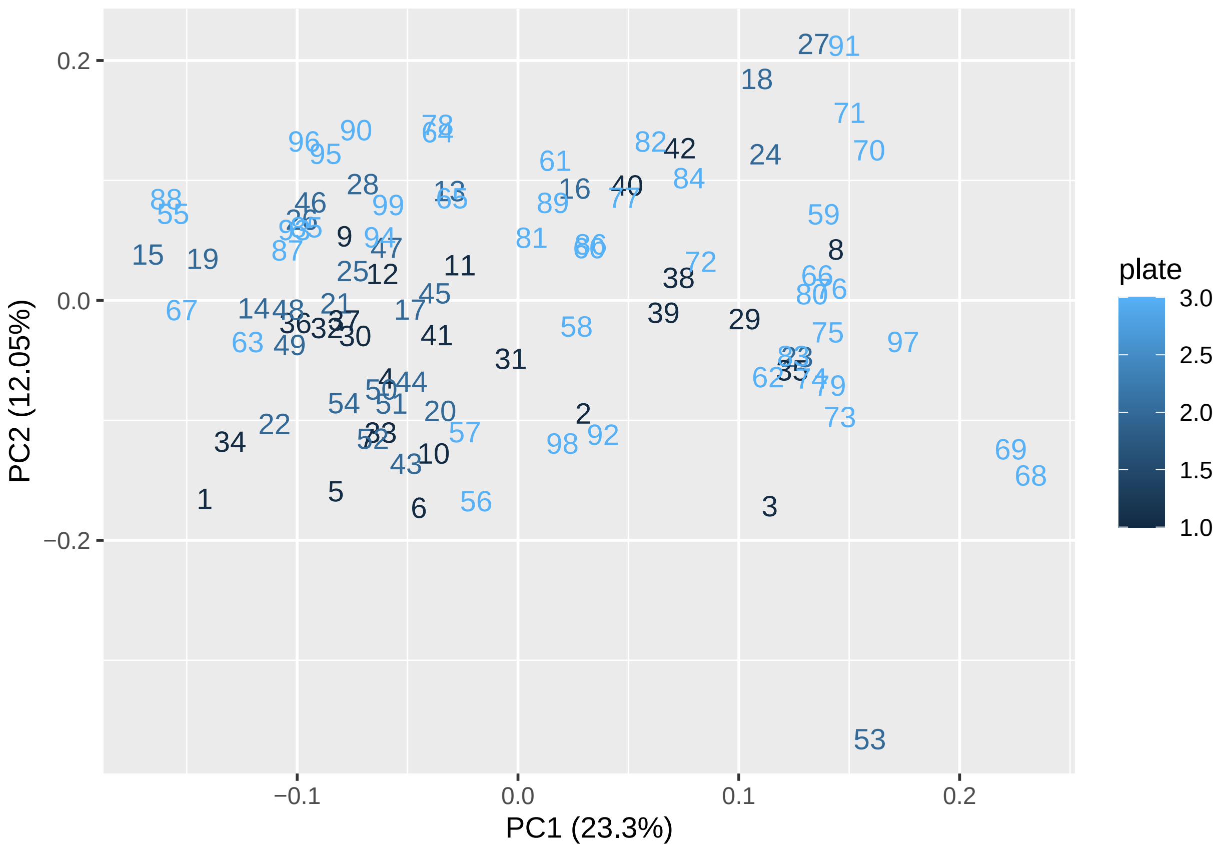
This doesn't look too bad. The outlier is 207222230009_R01C02 = ARPI-2019-01, one of the outliers from before.

but let's take a closer look at specific variables:

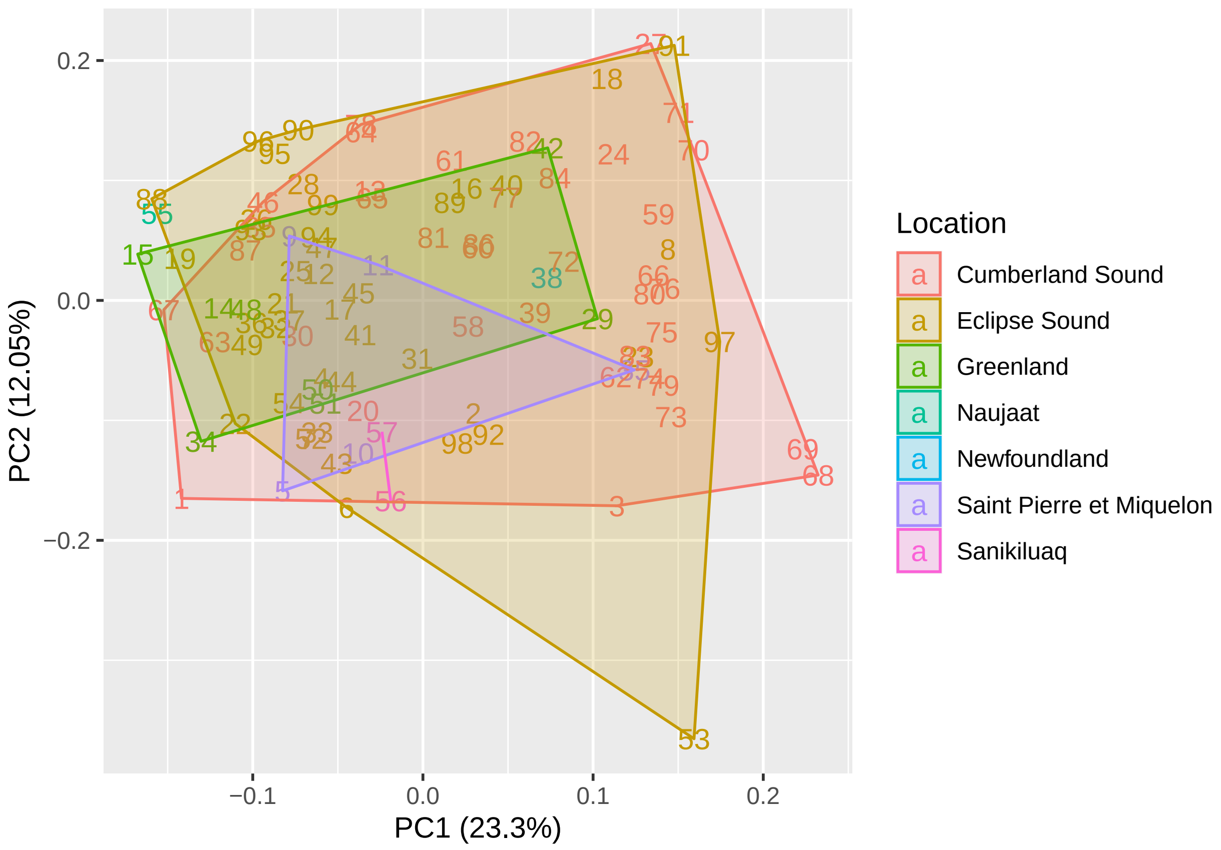
```r
n_betas_t$chip.ID.loc <- rownames(n_betas_t)

chip_kw <- sample_sheet_outliersRemoved %>%
  select(chip.ID.loc, chip.id, Location, Year, Sex, lab, row, column, plate)

kw_pca_data <- n_betas_t %>%
  left_join(chip_kw, by = "chip.ID.loc")

# Run PCA
pca_kw <- prcomp(kw_pca_data[,1:37492])

# Plot with colours for chip number
autoplot(pca_kw, data = kw_pca_data, colour = "chip.id", label = TRUE, shape = FALSE)
```



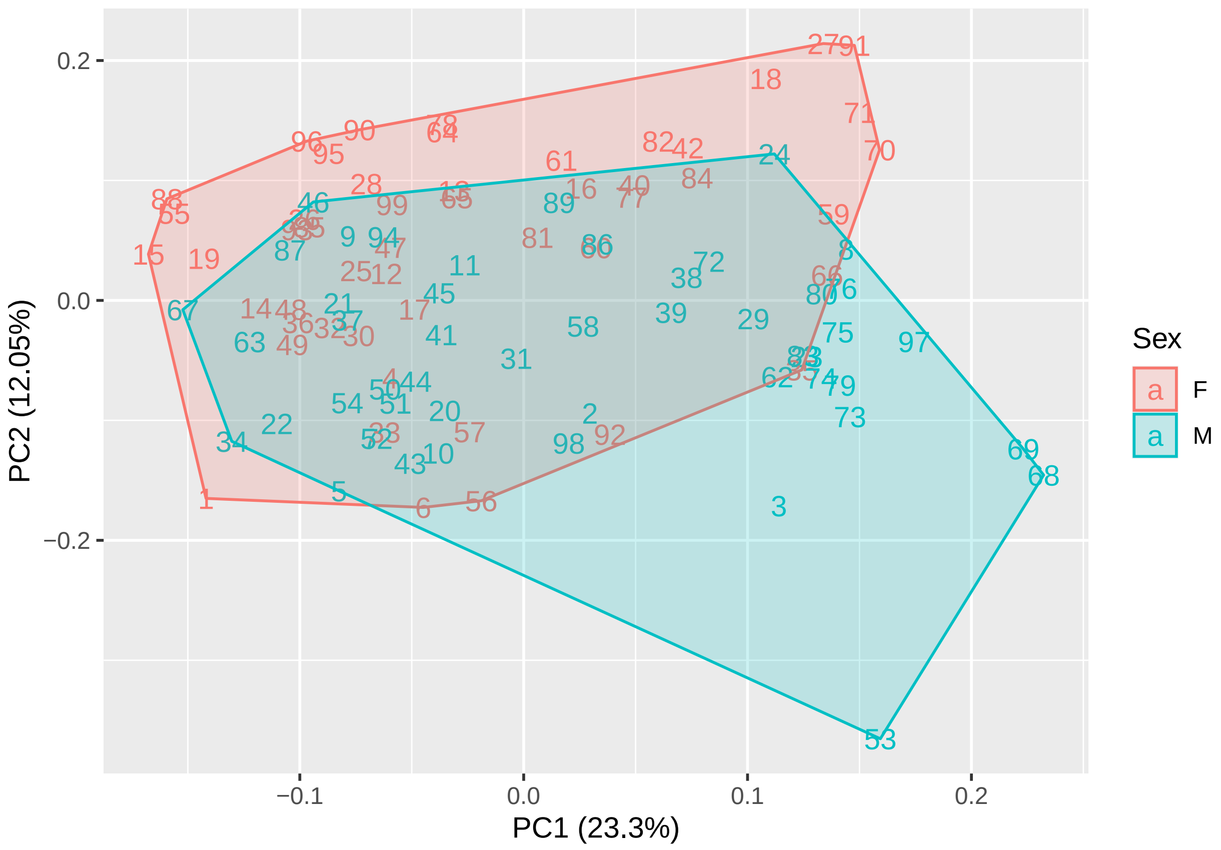
```
# Plot with colours for plate
autoplot(pca_kw, data = kw_pca_data, colour = "plate", label = TRUE, shape = FALSE)
```



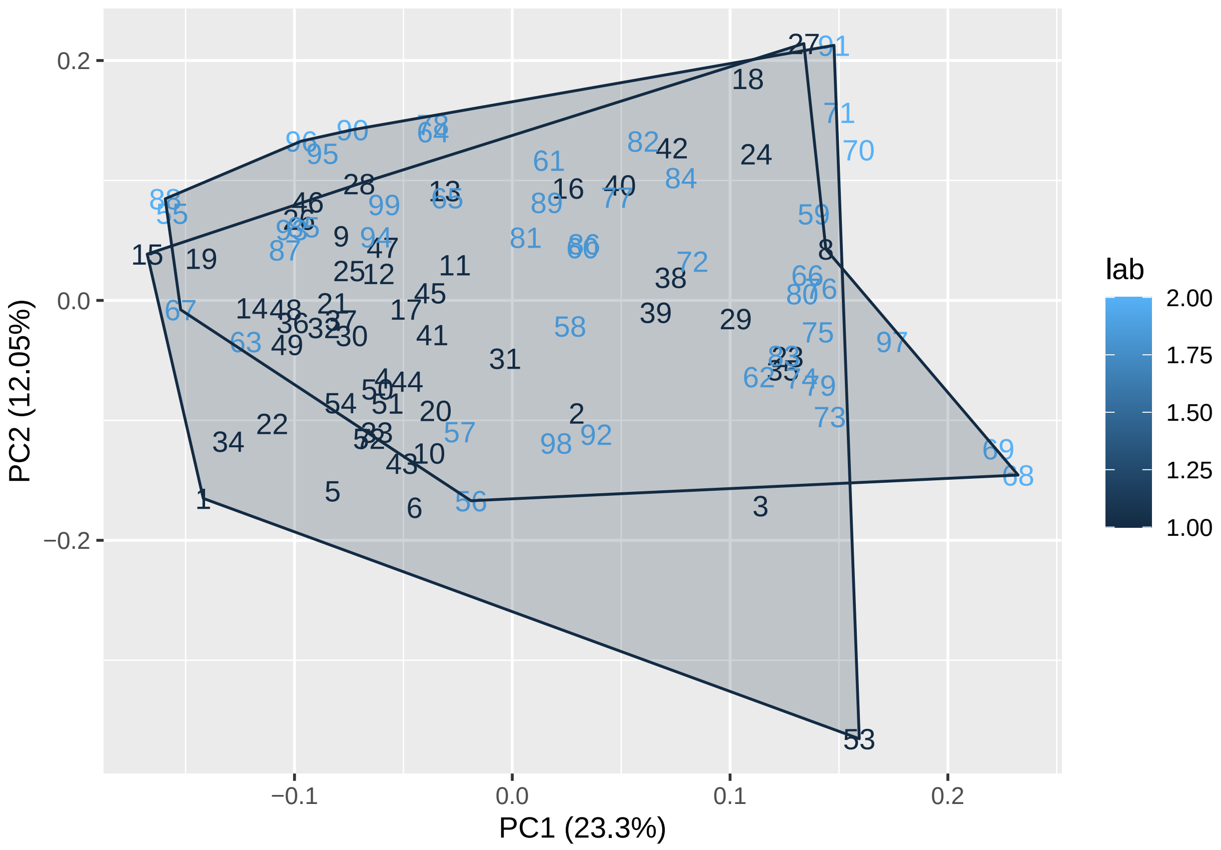
```r
# Plot with colours for location
autoplot(pca_kw, data = kw_pca_data, colour = "Location", label = TRUE, shape = FALSE, frame = TRUE)
```



```r
# Plot with colours for sex
autoplot(pca_kw, data = kw_pca_data, colour = "Sex", label = TRUE, shape = FALSE, frame = TRUE)
```



```r
# Plot with colours for lab
autoplot(pca_kw, data = kw_pca_data, colour = "lab", label = TRUE, shape = FALSE, frame = TRUE)
```



Heat Scree Plot We can use a heat scree plot to look at the influence of specific technical variables on the PCAs:

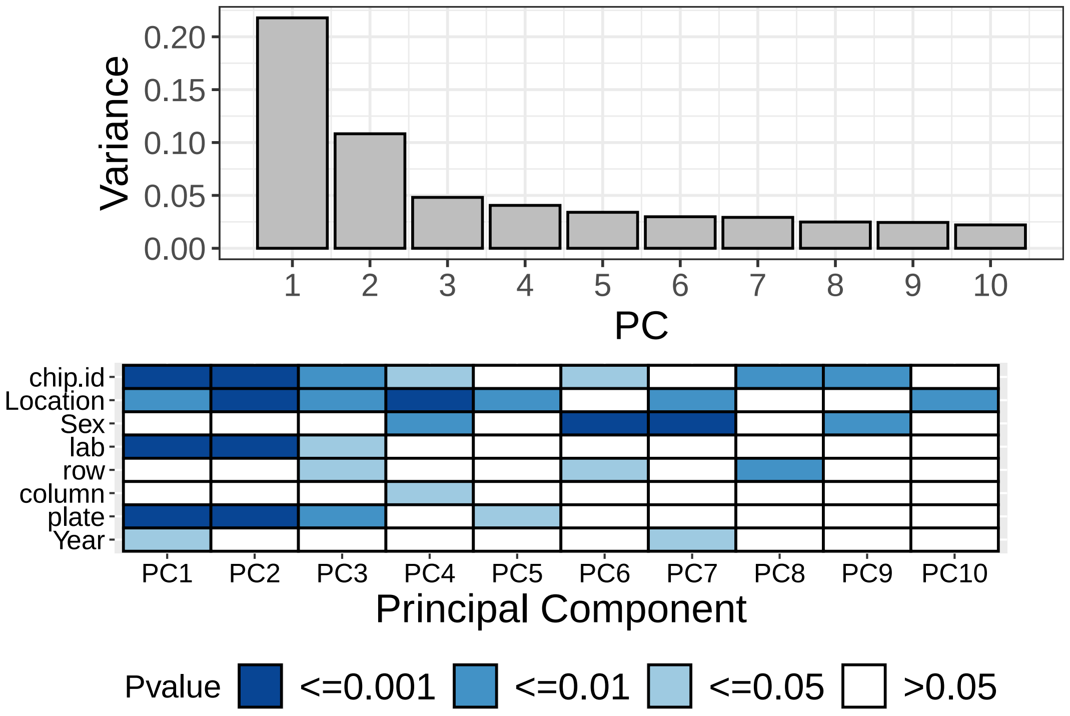
Source the function code (from Sam Lee in Jones Lab) and prep the PCA (adapted from Sam's code: 11_2022-09-27_CHILD_NO2_postnatal_dnam_limma_nameans_nocombat_SVs.Rmd):

```
source('2021-09-14_heat_scree_plot.R')

# PCA
n_betas_pca <- normalized_betas_minfi %>%
  select(! CGid) %>%
  as.data.frame()

dim(n_betas_pca) # 26126    99
```

```
## [1] 37492    99
```

```
# Note that in this dataframe, we have samples as columns and cg sites as rows
# While in the PCA above, we have cg sites as columns and samples as rows

# PCA of scaled DNAm data:
uncor.dat <- t(scale(t(n_betas_pca)))
pca <- princomp(uncor.dat)

# Set variables for heat scree plot function:
Loadings <- as.data.frame(unclass(pca$loadings))
vars <- pca$sdev^2
Importance <- vars/sum(vars)
```

```
#Specify which covariates are categorical and/or continuous
meta_categorical <- kw_pca_data[,c("chip.id", "Location", "Sex", "lab", "row", "column", "plate", "Year

# Specify the number of PCs you want shown
# pick enough to view most of the variance
Num <- 10

# Designate what order you want the variables to appear (continuous variables rbinded to categorical va
Order <- c(1:9)

# Apply function on PCA results of data, pulls in the meta data and beta values from above
# With scaled data
before <- heat_scree_plot(Loadings = Loadings,
                          Importance = Importance,
                          Num = Num,
                          Order = Order,
                          Categorical = meta_categorical,
                          Continuous = NULL)
before
```



Combat batch correction for lab:

```
library(sva)
```

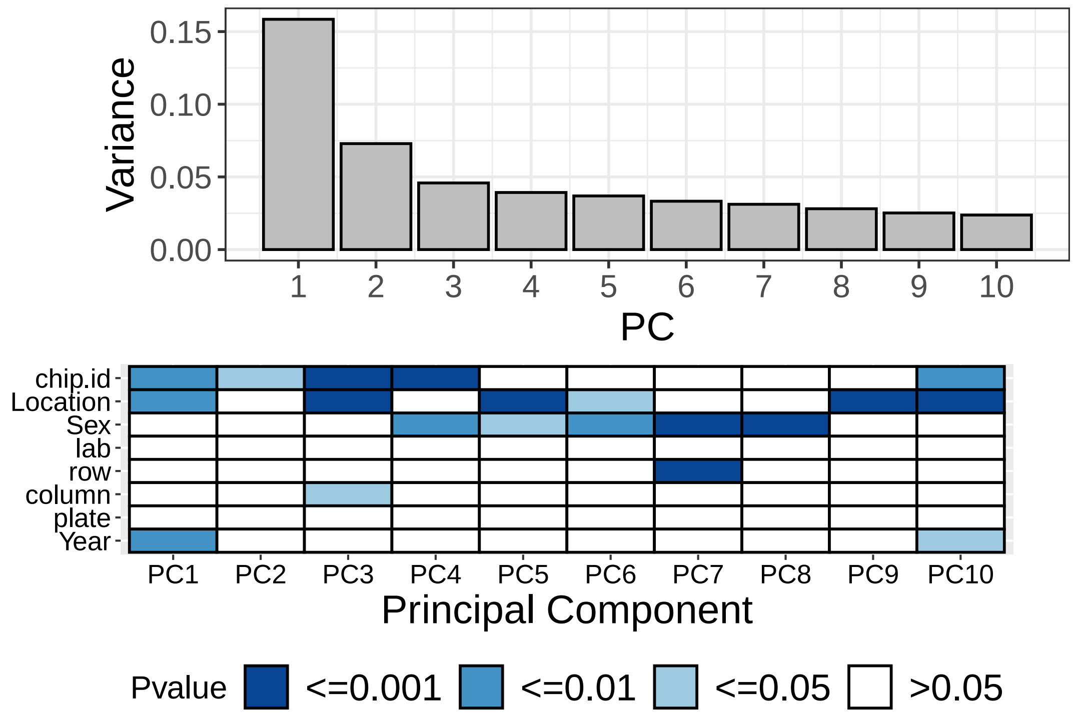
```
## Loading required package: mgcv

## Loading required package: nlme

##
## Attaching package: 'nlme'

## The following object is masked from 'package:Biostrings':
##
```

```
##     collapse
## The following object is masked from 'package:IRanges':
##
##     collapse
## The following object is masked from 'package:dplyr':
##
##     collapse
## This is mgcv 1.8-42. For overview type 'help("mgcv-package")'.
## Loading required package: genefilter
##
## Attaching package: 'genefilter'
## The following objects are masked from 'package:MatrixGenerics':
##
##     rowSds, rowVars
## The following objects are masked from 'package:matrixStats':
##
##     rowSds, rowVars
## The following object is masked from 'package:readr':
##
##     spec
## Loading required package: BiocParallel
```

```
combat_lab <- ComBat(dat = n_betas_pca,
                     batch = kw_pca_data$lab)
```

```
## Found2batches
## Adjusting for0covariate(s) or covariate level(s)
## Standardizing Data across genes
## Fitting L/S model and finding priors
## Finding parametric adjustments
## Adjusting the Data
```

```
# need a model matrix here?

# Now re-run heat scree plot:
# PCA of scaled DNAm data:
uncor_dat_lab <- t(scale(t(combat_lab)))
pca_lab <- princomp(uncor_dat_lab)

# Set variables for heat scree plot function:
Loadings_lab <- as.data.frame(unclass(pca_lab$loadings))
vars_lab <- pca_lab$sdev^2
Importance_lab <- vars_lab/sum(vars_lab)

#Specify which covariates are categorical and/or continuous
meta_categorical_lab <- kw_pca_data[,c("chip.id", "Location", "Sex", "lab", "row", "column", "plate", "

# Specify the number of PCs you want shown
# pick enough to view most of the variance
Num_lab <- 10

# Designate what order you want the variables to appear (continuous variables rbinded to categorical va
Order_lab <- c(1:9)

# Apply function on PCA results of data, pulls in the meta data and beta values from above
# With scaled data
after_lab <- heat_scree_plot(Loadings = Loadings_lab,
                             Importance = Importance_lab,
                             Num = Num_lab,
                             Order = Order_lab,
                             Categorical = meta_categorical_lab,
                             Continuous = NULL)
after_lab
```



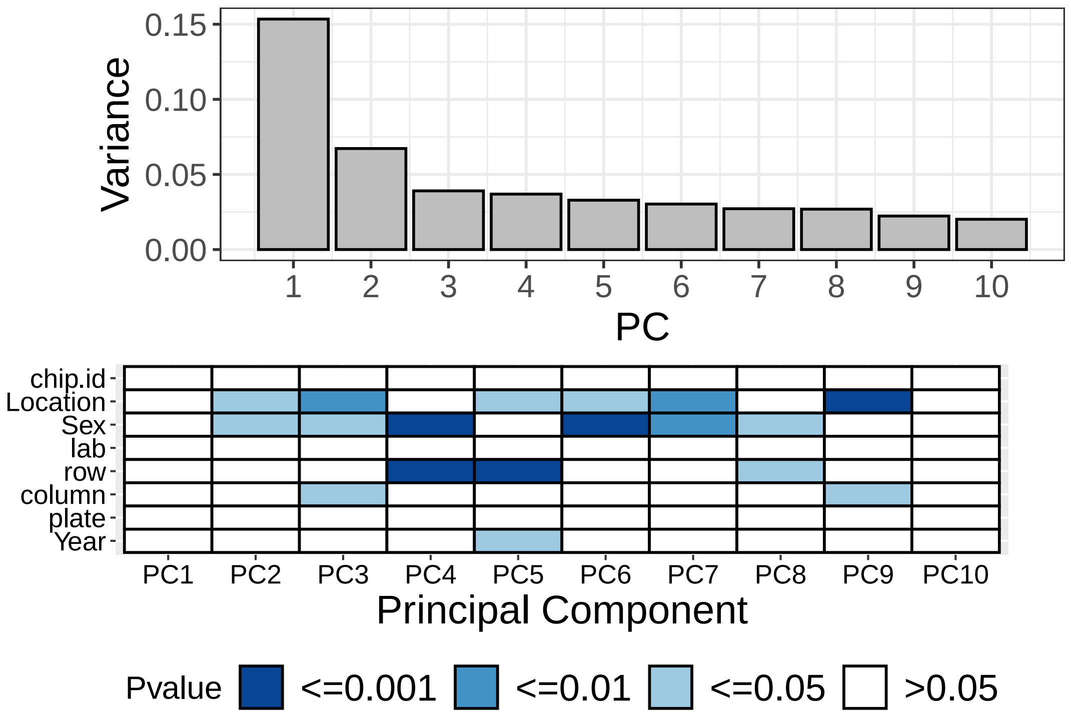
Combat batch correction for chip.id:

```
combat_lab_chip <- ComBat(dat = combat_lab,
                             batch = kw_pca_data$chip.id)
```

```
## Found12batches

## Adjusting for0covariate(s) or covariate level(s)

## Standardizing Data across genes

## Fitting L/S model and finding priors

## Finding parametric adjustments

## Adjusting the Data
```

```
# need a model matrix here?
```

```r
# Now re-run heat scree plot:
# PCA of scaled DNAm data:
uncor_dat_lab_chip <- t(scale(t(combat_lab_chip)))
pca_lab_chip <- princomp(uncor_dat_lab_chip)

# Set variables for heat scree plot function:
Loadings_lab_chip <- as.data.frame(unclass(pca_lab_chip$loadings))
vars_lab_chip <- pca_lab_chip$sdev^2
Importance_lab_chip <- vars_lab_chip/sum(vars_lab_chip)

#Specify which covariates are categorical and/or continuous
meta_categorical_lab_chip <- kw_pca_data[,c("chip.id", "Location", "Sex", "lab", "row", "column", "plat

# Specify the number of PCs you want shown
# pick enough to view most of the variance
Num_lab_chip <- 10

# Designate what order you want the variables to appear (continuous variables rbinded to categorical va
Order_lab_chip <- c(1:9)

# Apply function on PCA results of data, pulls in the meta data and beta values from above
# With scaled data
after_lab_chip <- heat_scree_plot(Loadings = Loadings_lab_chip,
                                  Importance = Importance_lab_chip,
                                  Num = Num_lab_chip,
                                  Order = Order_lab_chip,
                                  Categorical = meta_categorical_lab_chip,
                                  Continuous = NULL)
after_lab_chip
```

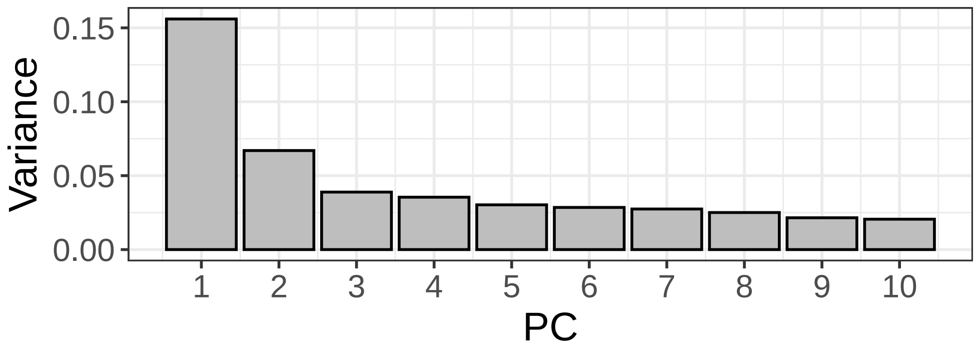


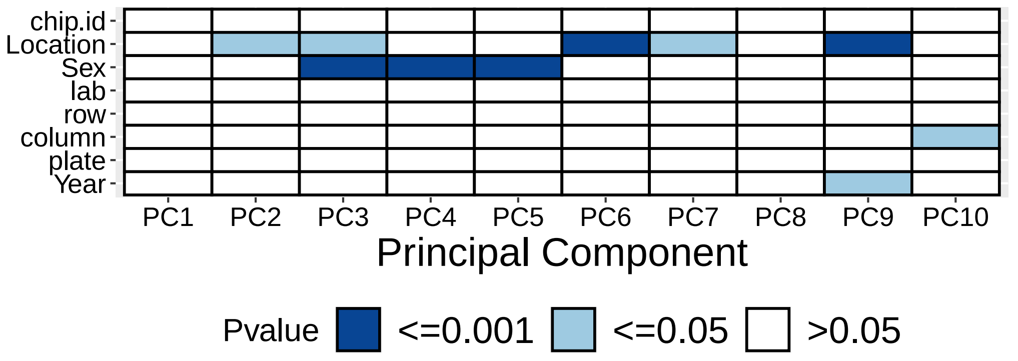
Combat batch correction for row:

```
combat_lab_chip_row <- ComBat(dat = combat_lab_chip,
                              batch = kw_pca_data$row)
```

```
## Found6batches

## Adjusting for0covariate(s) or covariate level(s)

## Standardizing Data across genes

## Fitting L/S model and finding priors

## Finding parametric adjustments

## Adjusting the Data
```

```
# need a model matrix here?

# Now re-run heat scree plot:
# PCA of scaled DNAm data:
uncor_dat_lab_chip_row <- t(scale(t(combat_lab_chip_row)))
pca_lab_chip_row <- princomp(uncor_dat_lab_chip_row)

# Set variables for heat scree plot function:
Loadings_lab_chip_row <- as.data.frame(unclass(pca_lab_chip_row$loadings))
vars_lab_chip_row <- pca_lab_chip_row$sdev^2
Importance_lab_chip_row <- vars_lab_chip_row/sum(vars_lab_chip_row)

#Specify which covariates are categorical and/or continuous
meta_categorical_lab_chip_row <- kw_pca_data[,c("chip.id", "Location", "Sex", "lab", "row", "column", "

# Specify the number of PCs you want shown
# pick enough to view most of the variance
Num_lab_chip_row <- 10

# Designate what order you want the variables to appear (continuous variables rbinded to categorical va
Order_lab_chip_row <- c(1:9)

# Apply function on PCA results of data, pulls in the meta data and beta values from above
# With scaled data
after_lab_chip_row <- heat_scree_plot(Loadings = Loadings_lab_chip_row,
                                      Importance = Importance_lab_chip_row,
                                      Num = Num_lab_chip_row,
                                      Order = Order_lab_chip_row,
                                      Categorical = meta_categorical_lab_chip_row,
                                      Continuous = NULL)
after_lab_chip_row
```





We no longer have any correlations for technical factors, but some minor correlations for biological factors (Location, Sex).

Prep the batch corrected data to export:

```r
# Transpose
n_betas_corrected <- combat_lab_chip_row %>%
  t() %>%
  as.data.frame()

# Add CGid as column names
colnames(n_betas_corrected) <- normalized_betas_minfi$CGid

# Export as .rds
#saveRDS(n_betas_corrected, 'output/tbetas_corrected_combined_KillerWhale_array.rds')
```