Supplemental code for 'Rapid changes in DNA methylation associated with the initiation of reproduction in a small songbird'

January 2021

Contents

1 Differential methylation analysis

1.1 Set up r environment

```r
# load required packages
library(lme4)
## for how to install the development version see here:
## https://github.com/variani/lme4qtl
library(lme4qtl)
library(Matrix)
library(emmeans)
library(multcomp)

# load data
START_DMA = load(file = "data_in/START_DMA.RData")
```

1.2 Run differential methylation analysis

```r
# 1. Create objects for saving loop result

DMA_Contrasts_pValues <- NULL
DMA_Contrasts_Estimates <- NULL
DMA_Contrasts_SEs <- NULL
DMA_GLMM_FixedEffectEstimates <- NULL
DMA_GLMM_RandomEffect_Var <- NULL
DMA_GLMM_DispersionStat <- NULL

# 2. Run loop

for (i in seq(1, nrow(RRBS_5097CpGs_28Samples), by = 28)) {

    # model each site
    CpGsite <- RRBS_5097CpGs_28Samples[i:(i + 27), ]

    # run the model and get the contrasts between reproductive
    # stages
    glmm_CpG <- relmatGlmer(cbind(MethCounts, UnMethCounts) ~
        RepStage + TempEnv + SamplBatch + (1 | Id2) + (1 | Id1),
    data = CpGsite, relmat = list(Id2 = M_relatedness), family = "binomial",
    control = glmerControl(optimizer = "bobyqa", boundary.tol = 0.01,
        optCtrl = list(maxfun = 2e+08)))
glmm_CpG_contrast <- emmeans(glmm_CpG, "RepStage")

# get estimates 6 pairwise contrasts
Contrasts_pValues <- summary(contrast(glmm_CpG_contrast,
    method = "pairwise"), adjust = "none")$p.value
Contrasts_Estimates <- summary(contrast(glmm_CpG_contrast,
    method = "pairwise"), adjust = "none")$estimate
Contrasts_SEs <- summary(contrast(glmm_CpG_contrast, method = "pairwise"),
    adjust = "none")$SE
## 6 fixed effetcs (1*intercept, 3*RepStage, 1*TempEnv,
## 1*SamplBatch)
GLMM_FixedEffectEstimates <- as.numeric(fixef(glmm_CpG))
## 2 random effects (1*repeated measurements (i.e. females),
## 1*relatedness)
GLMM_RandomEffect_Var <- as.data.frame(VarCorr(glmm_CpG))$sdcor

# calculate dispersion statistic (following AF Zuur, JM Hilbe
# & EN Ieno. A beginner's guide to GLM and GLMM with R - a
# frequentist and Bayesian perspective for
# ecologists(Highland Statistics Ltd., 2013).)
E1 <- residuals(glmm_CpG)
# number of parameters: fixed effects + 2 random effects
p1 <- length(fixef(glmm_CpG)) + 2
GLMM_DispersionStat <- sum(E1^2)/(nrow(CpGsite) - p1)

# format estimates to add estimates for each loop to
# estimates from previous loops
Contrasts_pValues_f <- data.frame(site = as.character(CpGsite[1,
    1]), RS1.RS2 = Contrasts_pValues[1], RS1.RS3 = Contrasts_pValues[2],
    RS1.RS4 = Contrasts_pValues[3], RS2.RS3 = Contrasts_pValues[4],
    RS2.RS4 = Contrasts_pValues[5], RS3.RS4 = Contrasts_pValues[6])
Contrasts_Estimates_f <- data.frame(site = as.character(CpGsite[1,
    1]), RS1.RS2 = Contrasts_Estimates[1], RS1.RS3 = Contrasts_Estimates[2],
    RS1.RS4 = Contrasts_Estimates[3], RS2.RS3 = Contrasts_Estimates[4],
    RS2.RS4 = Contrasts_Estimates[5], RS3.RS4 = Contrasts_Estimates[6])
Contrasts_SEs_f <- data.frame(site = as.character(CpGsite[1,
    1]), RS1.RS2 = Contrasts_SEs[1], RS1.RS3 = Contrasts_SEs[2],
    RS1.RS4 = Contrasts_SEs[3], RS2.RS3 = Contrasts_SEs[4],
    RS2.RS4 = Contrasts_SEs[5], RS3.RS4 = Contrasts_SEs[6])
GLMM_FixedEffectEstimates_f <- data.frame(site = as.character(CpGsite[1,
    1]), Int = GLMM_FixedEffectEstimates[1], RS2 = GLMM_FixedEffectEstimates[2],
    RS3 = GLMM_FixedEffectEstimates[3], RS4 = GLMM_FixedEffectEstimates[4],
    TE1 = GLMM_FixedEffectEstimates[5], SB2 = GLMM_FixedEffectEstimates[6])
GLMM_RandomEffect_Var_f <- data.frame(site = as.character(CpGsite[1,
    1]), Relatedness = GLMM_RandomEffect_Var[1], RepFemales = GLMM_RandomEffect_Var[2])
GLMM_DispersionStat_f <- data.frame(site = as.character(CpGsite[1,
    1]), DispStat = GLMM_DispersionStat)

# add estimates for each loop to estimates from previous
# loops
DMA_Contrasts_pValues <- rbind(DMA_Contrasts_pValues, Contrasts_pValues_f)
DMA_Contrasts_Estimates <- rbind(DMA_Contrasts_Estimates,
    Contrasts_Estimates_f)
DMA_Contrasts_SEs <- rbind(DMA_Contrasts_SEs, Contrasts_SEs_f)
DMA_GLMM_FixedEffectEstimates <- rbind(DMA_GLMM_FixedEffectEstimates,
    GLMM_FixedEffectEstimates_f)
    DMA_GLMM_RandomEffect_Var <- rbind(DMA_GLMM_RandomEffect_Var,
        GLMM_RandomEffect_Var_f)
    DMA_GLMM_DispersionStat <- rbind(DMA_GLMM_DispersionStat,
        GLMM_DispersionStat_f)
}

# 3. Save loop results
save(DMA_Contrasts_pValues, DMA_Contrasts_Estimates, DMA_Contrasts_SEs,
    DMA_GLMM_FixedEffectEstimates, DMA_GLMM_RandomEffect_Var,
    DMA_GLMM_DispersionStat, file = "out/Results_DMA.RData")
```

2 Co-methylation analysis (based on r package WGCNA)

2.1 Set up R environment

```r
library(WGCNA)
options(stringsAsFactors = FALSE)
allowWGCNAThreads()
## the above settings are important, do not omit.
library(dplyr)
library(ggplot2)
library(cowplot)
library(corrplot)
library(ggcorrplot)

# load data
START_CoMeth = load(file = "data_in/START_CoMeth.RData")
```

2.2 Run co-methylation analysis

```r
# 1. Quality check

# check for quality of CpG sites
gsg <- goodSamplesGenes(datMeth, verbose = 3)
gsg$allOK
## should be TRUE. If FALSE; remove all low quality sites:

# if (!gsg$allOK) { if (sum(!gsg$goodGenes)>0)
# printFlush(paste('Removing genes:',
# paste(names(datMeth)[!gsg$goodGenes], collapse = ', ')));
# if (sum(!gsg$goodSamples)>0) printFlush(paste('Removing
# samples:', paste(rownames(datMeth)[!gsg$goodSamples],
# collapse = ', '))); datMeth2 <- datMeth[gsg$goodSamples,
# gsg$goodGenes] } datMeth <- datMeth2

# check for quality of Samples
sampleTree <- hclust(dist(datMeth), method = "average")
sizeGrWindow(12, 9)
par(cex = 0.6)
par(mar = c(0, 6, 1, 0))
plot(sampleTree, main = "", sub = "", xlab = "", cex.lab = 2,
    cex.axis = 1.5, cex = 1.3)
## If outliers, remove them:

# row.names.remove <- c('SAMPLE_NAME') datMeth <-
# datMeth[!(row.names(datMeth) %in% row.names.remove), ]
# sampleTree2 <- hclust(dist(datMeth), method = 'average')

# Convert traits to a color representation: white means low,
# red means high, grey means missing entry
traitData <- datTraits
traitColors <- numbers2colors(datTraits, signed = FALSE)
plotDendroAndColors(sampleTree2, traitColors, groupLabels = names(datTraits),
    main = "Sample dendrogram and trait heatmap - RRBS After filter")

# 2. Call network

# Choose a set of soft-thresholding powers
powers <- c(c(1:10), seq(from = 12, to = 30, by = 2))

sft <- pickSoftThreshold(datMeth, powerVector = powers, verbose = 5,
    networkType = "signed")
## time consuming; save result to avoid running it again!
save(sft, file = "temp/sft_CoMeth.RData")

# plot result: (1) scale free topology and (2) mean
# connectivity as a function of the soft-thresholding power
sizeGrWindow(9, 5)
par(mar = c(5, 5, 3, 2))
par(mfrow = c(1, 2))
cex1 = 1
# (1)
plot(sft$fitIndices[, 1], -sign(sft$fitIndices[, 3]) * sft$fitIndices[,
    2], xlab = "Soft Threshold (power)", ylab = expression(paste("Scale Free Topology, R"^"2")),
    type = "n", main = "", cex.lab = 1.2)
text(sft$fitIndices[, 1], -sign(sft$fitIndices[, 3]) * sft$fitIndices[,
    2], labels = powers, cex = cex1, col = "red")
## line corresponds to using an R^2 cut-off of h, change if
## needed
abline(h = 0.91, col = "red")
# (2)
plot(sft$fitIndices[, 1], sft$fitIndices[, 5], xlab = "Soft Threshold (power)",
    ylab = "Mean Connectivity", type = "n", main = "", cex.lab = 1.2)
text(sft$fitIndices[, 1], sft$fitIndices[, 5], labels = powers,
    cex = cex1, col = "red")

# Module making, here use soft-thresholding power=14
# maxBlockSize is the number of genes (here CpG sites) in
# your dataset mergecutheight defines threshold for mergeing
# modules
dat <- as.matrix(sapply(datMeth, as.numeric))
net <- blockwiseModules(dat, power = 14, networkType = "signed",
    TOMType = "signed", minModuleSize = 30, maxBlockSize = 5100,
    reassignThreshold = 0, mergeCutHeight = 0.65, numericLabels = TRUE,
    pamRespectsDendro = FALSE, saveTOMs = TRUE, saveTOMFileBase = "methAfterFilterTOM0.15",
    verbose = 3)
## time consuming; save result to avoid running it again!
save(net, file = "temp/net_CoMeth.RData")

# 3. Visualize network

# get module label for each CpG site ('0' corresponding to no
# module) and convert label to a module color ('grey'
# corresponding to no module)
moduleLabels <- net$colors
moduleColors <- labels2colors(net$colors)
## number of sites without module (assigned to grey module):
length(moduleLabels[moduleLabels == 0])
```

```r
# get sites per module
SitesPerModule <- NULL
for (c in seq(1, length(levels(as.factor(moduleColors))), by = 1)) {
    color <- levels(as.factor(moduleColors))[c]
    length <- length(moduleColors[moduleColors == color])
    module <- data.frame(Module = color, NoSites = length)
    SitesPerModule <- rbind(SitesPerModule, module)
}

# calculate module eigensites
MEList <- moduleEigengenes(datMeth, colors = moduleColors, nPC = 10)

# get var explained by PC1-10
VarExplained <- MEList$varExplained
names(VarExplained) <- names(MEList$eigengenes)

# isolate module eigensites and calculate dissimilarity
MEs <- MEList$eigengenes
MEDiss <- 1 - cor(MEs)
METree <- hclust(as.dist(MEDiss), method = "average")
sizeGrWindow(7, 6)
plot(METree, main = "Clustering of module eigengenes", xlab = "",
    sub = "")

sizeGrWindow(6, 6)
par(cex = 1)
plotEigengeneNetworks(orderMEs(MEs), "", plotAdjacency = T, colorLabels = F,
    marDendro = c(0, 4, 2, 4), marHeatmap = c(3, 4, 1, 1.5),
    plotDendrograms = T, xLabelsAngle = 90)

# get correlation between modules
cor(orderMEs(MEs))

# 4. Correlation between modules and reproductive timing

# Define numbers of genes and samples
nGenes <- ncol(datMeth)
nSamples <- nrow(datMeth)

# get correlation
moduleTraitCor <- cor(MEs, datTraits[, c("RepState")], use = "p")
# add other sample traits if of interest, like:
# moduleTraitCor <- cor(MEs, datTraits[,c('RepState',
# 'SamplingDate', 'TempEnv', 'LayingDate', 'FemaleNo')], use
# = 'p');

# get p-values
moduleTraitPvalue <- corPvalueStudent(moduleTraitCor, nSamples)

ModuleRepStateCor <- data.frame(Correlation = moduleTraitCor[,
    1], Pvalue = moduleTraitPvalue[, 1])
ModuleRepStateCor_data <- data.frame(MEs[, c("MEturquoise", "MEgreen")],
    RepState = datTraits[, "RepState"])

# prepare data for plotting
ModuleRepStateCor_plot <- ModuleRepStateCor
ModuleRepStateCor_plot$Pvalue_MultipleTesting <- ModuleRepStateCor_plot$Pvalue *
    (10)
```

```r
Plot_Cor <- as.matrix(ModuleRepStateCor_plot[, 1])
row.names(Plot_Cor) <- c("Magenta", "Black", "Turquoise", "Salmon",
    "Green", "Yellow", "Purple", "Brown", "Cyan", "Grey")

# plot correlation between modules and reproductive state
CorrPlot_ReproductiveState <- ggcorrplot(Plot_Cor, p.mat = as.matrix(ModuleRepStateCor_plot[,
    3]), method = "circle", colors = c("black", "gray70", "gold1"),
    legend.title = "Correlation") + theme(legend.text = element_text(size = 13),
    legend.title = element_text(size = 16), axis.text.x = element_text(size = 14,
        color = "black"), axis.text.y = element_text(size = 14,
        color = "black")) + scale_y_discrete(labels = c("Reproductive state"))

# 5. Save the results

# get correlation of all CpG sites with the reproductive
# timing; trait based site significance (SS) repeat for other
# traits if needed!
RepState <- as.data.frame(datTraits$RepState)
names(RepState) <- "RepState"

SiteTraitSignificance = as.data.frame(cor(datMeth, RepState,
    use = "p"))
SSPvalue = as.data.frame(corPvalueStudent(as.matrix(SiteTraitSignificance),
    nSamples))
names(SiteTraitSignificance) = "SS_RepState"
names(SSPvalue) = "p.SS_RepState"

# Create the starting data frame
SiteInfo0 = data.frame(site = names(datMeth), moduleColor = moduleColors,
    SiteTraitSignificance, SSPvalue)

# get correlation of all CpG sites with each module
# eigensite; site module membership (MM)
SiteModuleMembership = as.data.frame(cor(datMeth, MEs, use = "p"))
MMPvalue = as.data.frame(corPvalueStudent(as.matrix(SiteModuleMembership),
    nSamples))
names(geneModuleMembership) = paste("MM_", modNames, sep = "")
names(MMPvalue) = paste("p.MM_", modNames, sep = "")

# Order modules by their significance for the reproductive
# state
modNames = substring(names(MEs), 3)
modOrder = order(-abs(cor(MEs, RepState, use = "p")))

# Add module membership information in the chosen order
for (mod in 1:ncol(SiteModuleMembership)) {
    oldNames = names(SiteInfo0)
    SiteInfo0 = data.frame(SiteInfo0, SiteModuleMembership[,
        modOrder[mod]], MMPvalue[, modOrder[mod]])
    names(SiteInfo0) = c(oldNames, paste("MM_", modNames[modOrder[mod]],
        sep = ""), paste("p.MM_", modNames[modOrder[mod]], sep = ""))
}

# Order CpG sites in the 'SiteInfo0' first by module color,
# then by trait based SS
SiteOrder = order(SiteInfo0$moduleColor, -abs(SiteInfo0$SS_RepState))
SiteInfo = SiteInfo0[SiteOrder, ]

# save trait specific results
```

```
write.table(SiteInfo, file = "out/Results_CoMeth.txt", quote = F,
    sep = "\t")
```