Phase 1A: RNA-seq data analysis

David Amar, Archana Raja

Phase 1A raw data was preprocessed at the BIC using our pipeline, which was implemented according to the MOP. Here we present QC analyses performed on the output of the pipeline.

1 Input data

Read the data from both sites - FPKMs, counts, and metadata (including the QC scores).

```
# load required libraries
setwd("/Users/David/Desktop/MoTrPAC/data/pass_1a/rnaseq/")
library(data.table);library(DESeq2)
library(preprocessCore);library(ggplot2)
library(corrplot)
# load our helper functions
source(
  "https://raw.githubusercontent.com/david-dd-amar/motrpac/master/tools/preprocessing_helper_functions.
  )
source(
  "https://raw.githubusercontent.com/david-dd-amar/motrpac/master/tools/association_analysis_functions.
  )

# Data paths
site2fpkm_path = list(
  stanford = "./stanford/rsem_genes_fpkm_pass1a_batch1_Stanford.csv",
  sinai = "./sinai/rsem_genes_fpkm_pass1a_batch1_Sinai.csv"
)
site2genecount_path = list(
  stanford = "./stanford/rsem_genes_count_pass1a_batch1_Stanford.csv",
  sinai = "./sinai/rsem_genes_count_pass1a_batch1_Sinai.csv"
)

# load the metadata of the samples
# this is a data frame called rnaseq_meta that contains
# the qc and sample metadata from both sites
load("./rnaseq_meta.RData")
rnaseq_meta$Tissue = tolower(rnaseq_meta$Tissue)
rnaseq_meta$Tissue = gsub(" powder","",rnaseq_meta$Tissue)
rnaseq_meta[rnaseq_meta=="N/A"] = NA
print("Number of samples flagged according to the MOP's thersholds:")
print(sum(rnaseq_meta$IsFlagged))

# Read the gene expression data in
site2fpkm = list()
site2counts = list()
for(site in names(site2fpkm_path)){
  currfpkm = fread(site2fpkm_path[[site]],header = T,
                   stringsAsFactors = F,data.table = F)
  rownames(currfpkm) = currfpkm[,1]
```

```
  currfpkm = currfpkm[,-1]
  site2fpkm[[site]] = currfpkm

  currcounts = fread(site2genecount_path[[site]],
                     header = T,stringsAsFactors = F,data.table = F)
  rownames(currcounts) = currcounts[,1]
  currcounts = currcounts[,-1]
  site2counts[[site]] = currcounts
}

# Metadata from DMAQC: use our merged data frame
system(paste("~/google-cloud-sdk/bin/gsutil",
             "cp gs://bic_data_analysis/pass1a/pheno_dmaqc/merged_dmaqc_data.RData",
             "."))
load("merged_dmaqc_data.RData")
system("rm merged_dmaqc_data.RData")
# restrict the data frame to vial ids in our RNA-seq data
print("The following vial ids are not in the DMAQC data:")
print(setdiff(rownames(rnaseq_meta),merged_dmaqc_data$viallabel))
merged_dmaqc_data = merged_dmaqc_data[
    is.element(set=rownames(rnaseq_meta),merged_dmaqc_data$viallabel),]
rownames(merged_dmaqc_data) = as.character(merged_dmaqc_data$viallabel)
merged_dmaqc_data$tissue = merged_dmaqc_data$sampletypedescription
```

2 Data preprocessing and QC

We tested two simple ways to normalize the count data: (1) FPKM, and (2) factor normalized counts with and without variance stabilizing transformations (to accound for gene dispersion).

2.1 FPKM data

```
#' Takes an FPKM matrix, removes lowly expressed genes and log transform
#' the remaining matrix
#' @return A matrix of log transformed FPKMs
process_fpkm1 <-function(fpkm_matrix, intensity_threshold=0,intensity_pct=0.2){
  lowly_expressed_genes = rowSums(
    fpkm_matrix==intensity_threshold)/ncol(fpkm_matrix) > intensity_pct
  fpkm_matrix = fpkm_matrix[!lowly_expressed_genes,]
  fpkm_matrix = log(fpkm_matrix+1,base = 2)
  return(fpkm_matrix)
}
#' A wrapper for preprocessCore's quantile normalization.
#' Comment: we do not use this by default
run_quantile_normalization<-function(x){
  x = as.matrix(x)
  mode(x) = "numeric"
  newx = preprocessCore::normalize.quantiles.robust(x)
  rownames(newx) = rownames(x)
  colnames(newx) = colnames(x)
  return(newx)
```

```
}
# Process the FPKM matrix from each site separately
site_proc_fpkms = lapply(site2fpkm,process_fpkm1)
# Check the dimension of the reduced data
print("FPKM processing done for each site, matrix dim:")
```

```
## [1] "FPKM processing done for each site, matrix dim:"
```

```
print(sapply(site_proc_fpkms,dim))
```

```
##      stanford sinai
## [1,]    13616 11951
## [2,]      320   320
```

```
# Get the shared genes
shared_genes = intersect(rownames(site_proc_fpkms[[1]]),
                         rownames(site_proc_fpkms[[2]]))
print("Number of shared genes that survive the filter above:")
```

```
## [1] "Number of shared genes that survive the filter above:"
```

```
print(length(shared_genes))
```

```
## [1] 11711
```

```
proc_fpkms = cbind(site_proc_fpkms[[1]][shared_genes,],
                   site_proc_fpkms[[2]][shared_genes,])
# QC: make sure the metadata and the expression matrix have the same sample id:
print("do we have the same samples in the expression and metadata matrices?")
```

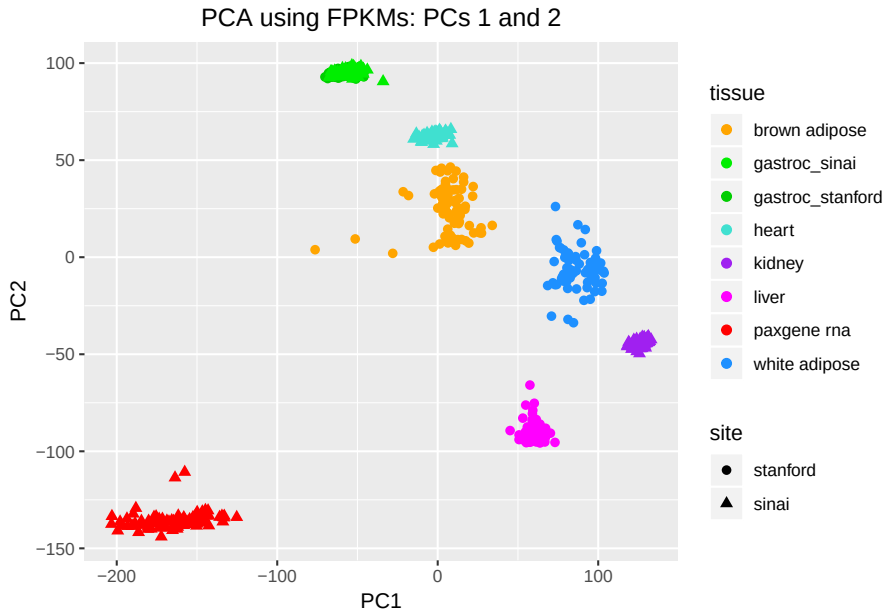
```
## [1] "do we have the same samples in the expression and metadata matrices?"
```

```
print(all(colnames(proc_fpkms) %in% rownames(rnaseq_meta)))
```

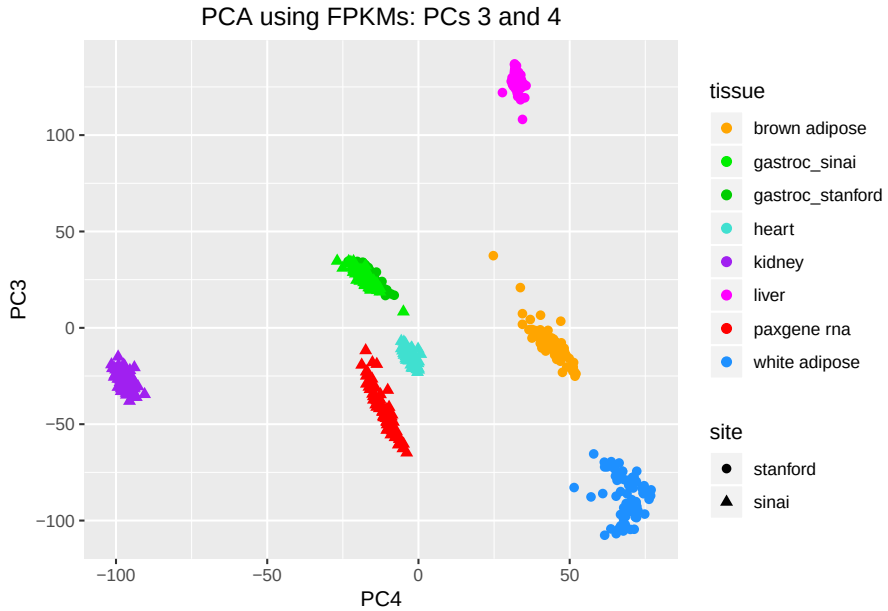
```
## [1] TRUE
```

```
# Run the PCA: try all genes first
fpkm_all = cbind(site2fpkm[[1]],site2fpkm[[2]])
fpkm_all = log(fpkm_all+1,base=2)
fpkm_pca = prcomp(t(fpkm_all))
fpkm_pcax = fpkm_pca$x
tissue_and_site = rnaseq_meta[rownames(fpkm_pcax),"Tissue"]
site_info = rnaseq_meta[rownames(fpkm_pcax),"site"]
tissue_and_site[grepl("gastroc",tissue_and_site) &
                  site_info == "stanford"] = "gastroc_stanford"
tissue_and_site[grepl("gastroc",tissue_and_site) &
                  site_info != "stanford"] = "gastroc_sinai"
df = data.frame(fpkm_pcax[,1:10],
                tissue = tissue_and_site,
                site = site_info)
scatterplot_colors =  scale_colour_manual(values=c(liver='#FF00FF',
         gastroc_stanford='#00cd00',
         gastroc_sinai='#00ee00',
         kidney='#a020f0',
         'brown adipose'='#ffa500',
         'white adipose'='#1e90ff',
         'paxgene rna'='#ff0000',
         heart='#40e0d0'))
```

```
ggplot(df,aes(x=PC1, y=PC2,shape=site,color=tissue)) +
  geom_point(size=2) + ggtitle("PCA using FPKMs: PCs 1 and 2") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scatterplot_colors
```

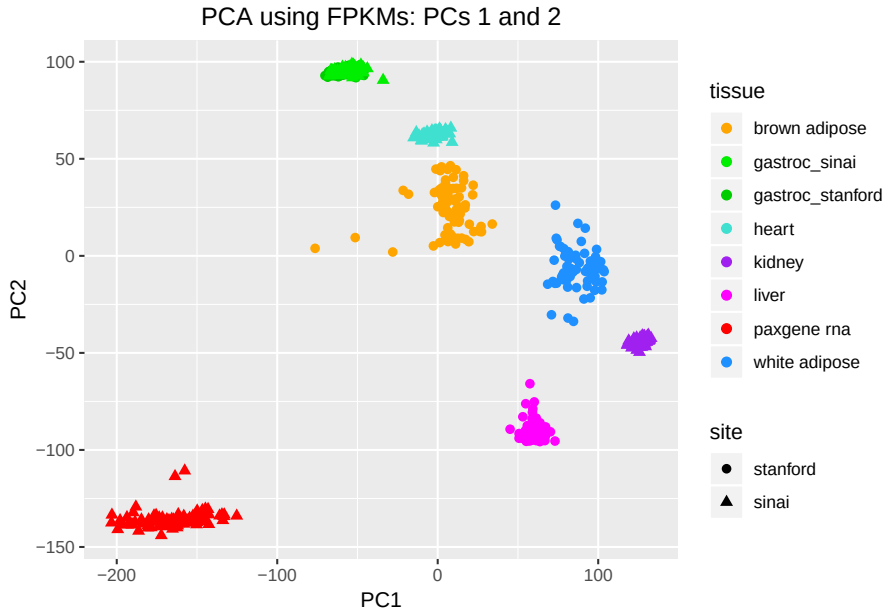



```
ggplot(df,aes(x=PC4, y=PC3,shape=site, color=tissue)) +
  geom_point(size=2) + ggtitle("PCA using FPKMs: PCs 3 and 4") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scatterplot_colors
```

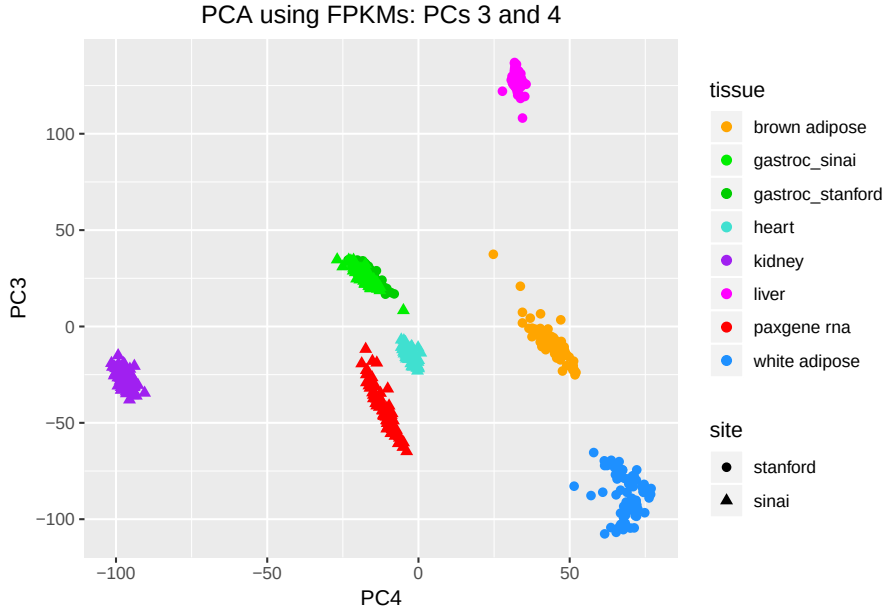



```
# Rerun on the reduced matrix
fpkm_pca = prcomp(t(proc_fpkms),retx = T)
fpkm_pcax = fpkm_pca$x
fpkm_pca_proc = prcomp(t(fpkm_all),retx = T)
fpkm_pca_procx = fpkm_pca_proc$x
```

```
df = data.frame(fpkm_pca_procx[,1:10],
                tissue = tissue_and_site,
                site = site_info)
ggplot(df,aes(x=PC1, y=PC2,shape=site,color=tissue)) +
  geom_point(size=2) + ggtitle("PCA using FPKMs: PCs 1 and 2") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scatterplot_colors
```




```
ggplot(df,aes(x=PC4, y=PC3,shape=site, color=tissue)) +
  geom_point(size=2) + ggtitle("PCA using FPKMs: PCs 3 and 4") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scatterplot_colors
```



2.2 Normalized counts

```
# Pipeline 2: work with count data
# Combine the two count matrices
count_matrix = as.matrix(cbind(site2counts[[1]],site2counts[[2]]))
#' Use DESeq2 to estimate sample factors and gene dispersion
#' @return a DESeqDataSet
process_counts<-function(count_matrix,plotFactors=T){
  mode(count_matrix) = "integer"
  se <- SummarizedExperiment(count_matrix)
  dds <- DESeqDataSet(se, design = ~ 1 )
  #Estimate size factors
  dds <- estimateSizeFactors( dds )
  if(plotFactors){
      # Plot the size factors
    plot(sizeFactors(dds), colSums(counts(dds)),ylab="Library size",
         xlab = "DESeq estimated size factors")
    abline(lm(colSums(counts(dds)) ~ sizeFactors(dds) + 0))
  }
  dds <- estimateDispersions(dds)
  return(dds)
}

# Process the counts and normalize
dds = process_counts(count_matrix)
# Simple normalization and log transform
# The argument normalized equals true, divides each column by its size factor.
logcounts <- log2( counts(dds, normalized=TRUE) + 1 )
pc <- prcomp( t( logcounts ) )
counts_pcax1 = pc$x
# Try variance stabilizing transformation instead
vsd <- varianceStabilizingTransformation(dds)
pc2 <- prcomp( t( assay(vsd) ) )
counts_pcax2 = pc2$x
# PCA plots
df = data.frame(counts_pcax1[,1:10],
                tissue = rnaseq_meta[rownames(counts_pcax1),"Tissue"],
                site = rnaseq_meta[rownames(counts_pcax1),"site"])
ggplot(df,aes(x=PC1, y=PC2,shape=site, color=tissue)) +
  geom_point(size=2) + ggtitle("PCA using normalized counts") +
  theme(plot.title = element_text(hjust = 0.5))
df = data.frame(counts_pcax2[,1:10],
                tissue = rnaseq_meta[rownames(counts_pcax2),"Tissue"],
                site = rnaseq_meta[rownames(counts_pcax2),"site"])
ggplot(df,aes(x=PC1, y=PC2,shape=site, color=tissue)) +
  geom_point(size=2) + ggtitle("PCA using normalized counts (vsd)") +
  theme(plot.title = element_text(hjust = 0.5))
```

2.3 Correlation between PCs and non-RNA variables

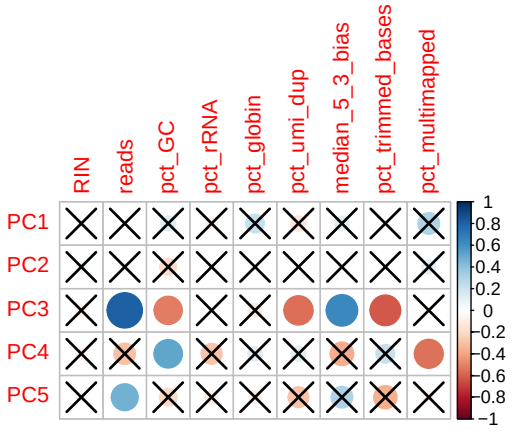
Here we take each tissue and analyze its samples. We quantile normalize the data and run PCA. For the top principal components (5) we compute their association with the RNA-seq qc scores or information collected

about the rats. For the latter we use sex, weight, achieved distances, and shock time during the experiment. Note that all of these scores are highly correlated. Below we perform a simple Spearman correlation-based analysis. We can later recompute the associations after proper adjustments.

```
# RNA-seq meta to correlate with genes/pcs
assay_cols_for_qc_analysis = c("RIN","reads","pct_GC","pct_rRNA","pct_globin",
                      "pct_umi_dup","median_5_3_bias","pct_trimmed_bases",
                      "pct_multimapped")
# Animal information to correlate with genes/pcs
# FUTURE WORK: add site id and batch
animal_data_cols_for_qc_analysis = c(
  "acute.test.weight","animal.registration.sex","acute.test.distance",
  "acute.test.howlongshock_seconds","animal.key.timepoint")
# Labels for separating the data by site and tissue
rnaseq_meta_batchs = unique(rnaseq_meta[,c("GET_site","Tissue")])

qc_scores_results = c()
animal_data_results = c()
p_thr = 0.001
for(i in 1:nrow(rnaseq_meta_batchs)){
  curr_site = rnaseq_meta_batchs[i,1]
  curr_tissue = rnaseq_meta_batchs[i,2]
  curr_samples = as.character(rnaseq_meta$vial_label[rnaseq_meta$GET_site==curr_site &
                                       rnaseq_meta$Tissue==curr_tissue])
  # For rat data, take samples whose label id starts with "9"
  curr_samples = curr_samples[grepl("^9",curr_samples)]
  curr_data = process_fpkm1(fpkm_all[,curr_samples])
  curr_data = run_quantile_normalization(curr_data)
  curr_pca = prcomp(t(curr_data))
  curr_pcax = curr_pca$x[,1:5]
  explained_var = summary(curr_pca)[["importance"]][3,5]
  curr_meta1 = rnaseq_meta[curr_samples,assay_cols_for_qc_analysis]
  corrs = cor(curr_pcax,curr_meta1,method="spearman")
  corrsp = pairwise_eval(curr_pcax,curr_meta1,func=cor.test,f="p.value",method="spearman")
  for(i in 1:nrow(corrsp)){
    for(j in 1:ncol(corrsp)){
      if(corrsp[i,j]>p_thr){next}
      qc_scores_results = rbind(qc_scores_results,
            c(curr_tissue,curr_site,
              rownames(corrsp)[i],colnames(corrsp)[j],corrs[i,j],corrsp[i,j])
            )
    }
  }
  colnames(qc_scores_results) = c("Tissue","CAS_site","PC",
                                  "qc_metric","rho(spearman)","p-value")

  curr_meta2 = merged_dmaqc_data[curr_samples,animal_data_cols_for_qc_analysis]
  # Animal analysis: for correlations consider rats with distance >0
  training_rats = curr_meta2$acute.test.distance > 0
  training_rats_x = curr_pcax[training_rats,]
  training_rats_meta = curr_meta2[training_rats,]
  corrs2 = cor(training_rats_x,training_rats_meta,method="spearman")
  corrsp2 = pairwise_eval(training_rats_x,
                          training_rats_meta,func=cor.test,f="p.value",method="spearman")
```

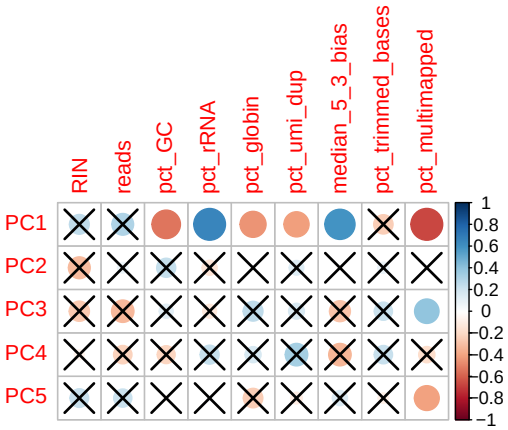
```
  for(i in 1:nrow(corrsp2)){
    for(j in 1:ncol(corrsp2)){
      if(corrsp2[i,j]>p_thr){next}
      animal_data_results = rbind(animal_data_results,
            c(curr_tissue,curr_site,
              rownames(corrsp2)[i],colnames(corrsp2)[j],corrs2[i,j],corrsp2[i,j])
            )
    }
  }
  colnames(animal_data_results) = c("Tissue","CAS_site","PC",
                                    "variable","rho(spearman)","p-value")

  # In case we want a linear test for association
  # corrsp2 = pairwise_eval(as.data.frame(curr_pcax),
  #   curr_meta2,func=linear_association_analysis,f="pval")

  # Add some plots for selected tissues
  if(grepl("white adipose",curr_tissue) ||
     grepl("heart|gastro",curr_tissue,ignore.case = T)){
      main = paste(curr_tissue,curr_site,
                   paste("(",format(explained_var,digits = 2),")",sep=""))
      corrplot(corrs,p.mat = corrsp,
               sig.level = 0.001,title = main,mar = c(3,3,3,3))
      corrplot(corrs2,p.mat = corrsp2,
               sig.level = 0.001,title = main,mar = c(3,3,3,3))
  }
}
```

gastrocnemius Stanford (0.22)



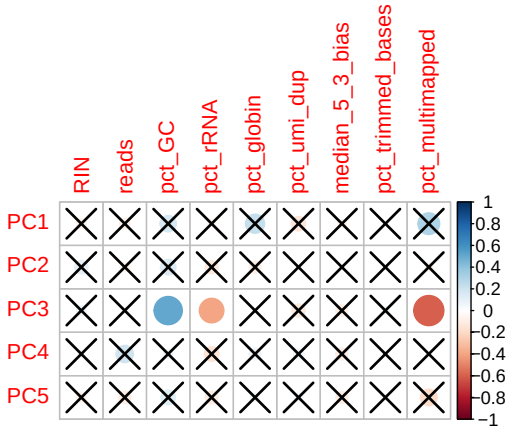
gastrocnemius Stanford (0.22)

white adipose Stanford (0.54)



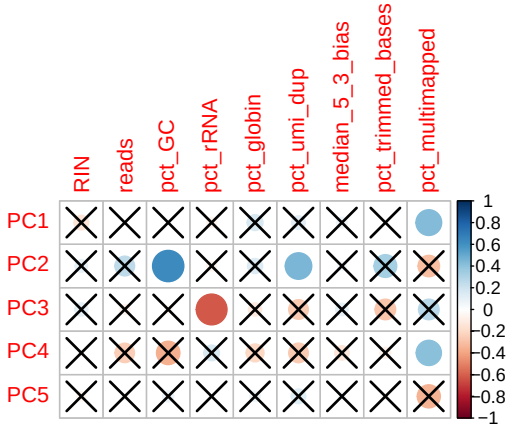
white adipose Stanford (0.54)

gastrocnemius MSSM (0.3)



gastrocnemius MSSM (0.3)

heart MSSM (0.29)



heart MSSM (0.29)


```
write.table(qc_scores_results,sep="\t",quote=F,row.names = F)
```

```
## Tissue   CAS_site    PC  qc_metric   rho(spearman)   p-value
## brown adipose    Stanford    PC1 pct_GC  -0.437487423368339  6.19583319368728e-05
## brown adipose    Stanford    PC2 pct_multimapped 0.77595670033972    7.20618198941024e-17
## brown adipose    Stanford    PC3 pct_multimapped 0.515049195455129   1.40187994593701e-06
## gastrocnemius    Stanford    PC3 reads   0.812997129452825   0
## gastrocnemius    Stanford    PC3 pct_GC  -0.519768727705525  1.07897242366378e-06
## gastrocnemius    Stanford    PC3 pct_umi_dup -0.558846217074065  1.80941117839739e-07
## gastrocnemius    Stanford    PC3 median_5_3_bias 0.644791608804907   1.87824763933658e-10
## gastrocnemius    Stanford    PC3 pct_trimmed_bases   -0.614826100828807  2.12010854416145e-09
## gastrocnemius    Stanford    PC4 pct_GC  0.528849975804027   6.44687715037977e-07
## gastrocnemius    Stanford    PC4 pct_multimapped -0.546843486689456  2.22036352458178e-07
## gastrocnemius    Stanford    PC5 reads   0.471237623136357   1.69066612448149e-05
## white adipose    Stanford    PC1 pct_GC  -0.530813664711835  5.75604637953199e-07
## white adipose    Stanford    PC1 pct_rRNA    0.658711199729247   5.55068001456361e-11
## white adipose    Stanford    PC1 pct_globin  -0.442843952851069  4.90746143477031e-05
## white adipose    Stanford    PC1 pct_umi_dup -0.41408220945838   0.000164111845128755
## white adipose    Stanford    PC1 median_5_3_bias 0.595804046630313   8.69199596017119e-09
## white adipose    Stanford    PC1 pct_multimapped -0.662214643910784  4.04298571435989e-11
## white adipose    Stanford    PC3 pct_multimapped 0.391298305002327   0.000396473256424748
## white adipose    Stanford    PC5 pct_multimapped -0.404477335121029  0.000239870754204283
## liver    Stanford    PC1 pct_multimapped -0.74536464215455   5.01156268752533e-15
## liver    Stanford    PC2 pct_GC  0.471306651619021   1.32906485942042e-05
## gastrocnemius    MSSM    PC3 pct_GC  0.519953600177394   1.06787786514859e-06
## gastrocnemius    MSSM    PC3 pct_rRNA    -0.393778064838239  0.000361275790052747
## gastrocnemius    MSSM    PC3 pct_multimapped -0.593878084440538  9.9759400213109e-09
## paxgene rna  MSSM    PC1 pct_rRNA    0.389753903456624   0.000419952874999523
## paxgene rna  MSSM    PC1 pct_globin  0.679628962860082   7.84327012806937e-12
## paxgene rna  MSSM    PC1 pct_umi_dup 0.391292252051746   0.000438201879690095
## paxgene rna  MSSM    PC1 pct_multimapped 0.644573020610251   1.91359913780488e-10
## paxgene rna  MSSM    PC4 median_5_3_bias -0.492325515926119  4.68754666407703e-06
```

```
## paxgene rna  MSSM    PC5 pct_umi_dup 0.440837643369289   6.37606944116948e-05
## heart    MSSM    PC1 pct_multimapped 0.433054242337751   7.49238070236935e-05
## heart    MSSM    PC2 pct_GC  0.639774982392417   2.87067378242492e-10
## heart    MSSM    PC2 pct_umi_dup 0.459662988922758   2.30004316495151e-05
## heart    MSSM    PC3 pct_rRNA    -0.619932416269632  1.42836995463331e-09
## heart    MSSM    PC4 pct_multimapped 0.416232426796725   0.000150506008683982
## kidney   MSSM    PC1 pct_multimapped -0.705959741532286  5.24998550187039e-13
## kidney   MSSM    PC2 pct_multimapped -0.459450579957392  2.32274296115178e-05
## kidney   MSSM    PC4 pct_GC  -0.524063887533509  8.47295762724391e-07
```

```
write.table(animal_data_results,sep="\t",quote=F,row.names = F)
```

```
## Tissue   CAS_site    PC  variable    rho(spearman)   p-value
## brown adipose    Stanford    PC1 acute.test.weight   0.657506891613539   2.05902343300611e-08
## brown adipose    Stanford    PC1 animal.registration.sex 0.724026597992962   1.34001175101545e-10
## brown adipose    Stanford    PC1 acute.test.distance -0.4879939747422    0.000102009932265962
## brown adipose    Stanford    PC2 animal.key.timepoint    -0.499809755934239  6.48193868559885e-05
## brown adipose    Stanford    PC3 animal.key.timepoint    -0.490720041887603  9.20118710130123e-05
## gastrocnemius    Stanford    PC1 animal.key.timepoint    -0.429706892807439  0.000761947868535974
## gastrocnemius    Stanford    PC2 animal.key.timepoint    0.718834509468949   2.08938436249864e-10
## gastrocnemius    Stanford    PC4 acute.test.weight   -0.673046788663421  7.1294706260672e-09
## gastrocnemius    Stanford    PC4 animal.registration.sex -0.724026597992962  1.34001175101548e-10
## gastrocnemius    Stanford    PC4 acute.test.distance 0.480764206486973   0.000133549114535062
## white adipose    Stanford    PC1 acute.test.weight   0.764839883514509   2.77398220239498e-12
## white adipose    Stanford    PC1 animal.registration.sex 0.866154152079774   1.64937863897965e-18
## white adipose    Stanford    PC1 acute.test.distance -0.589918324655257  1.09876573208871e-06
## white adipose    Stanford    PC3 animal.registration.sex -0.470668784186036  0.000192625355032498
## white adipose    Stanford    PC5 animal.key.timepoint    0.620590887786808   2.03106784068273e-07
## liver    Stanford    PC1 acute.test.weight   -0.751577119893718  1.06101916372173e-11
## liver    Stanford    PC1 animal.registration.sex -0.866154152079774  1.64937863897971e-18
## liver    Stanford    PC1 acute.test.distance 0.567336835721909   3.42154524223648e-06
## liver    Stanford    PC2 animal.key.timepoint    -0.599547577185691  6.59104880395432e-07
## liver    Stanford    PC4 animal.key.timepoint    0.66728462432775    1.0641921488003e-08
## gastrocnemius    MSSM    PC1 animal.key.timepoint    -0.604777275678276  4.9583629720044e-07
## gastrocnemius    MSSM    PC2 animal.key.timepoint    -0.641011615234047  5.94645186478034e-08
## gastrocnemius    MSSM    PC3 acute.test.weight   -0.571560490702011  2.78438743379746e-06
## gastrocnemius    MSSM    PC3 animal.registration.sex -0.616915977521741  2.51019372540051e-07
## gastrocnemius    MSSM    PC3 acute.test.distance 0.549308562540788   7.99210466534504e-06
## gastrocnemius    MSSM    PC3 acute.test.howlongshock_seconds -0.44053093811446   0.00053865767348876
## gastrocnemius    MSSM    PC4 animal.key.timepoint    -0.508027853565445  4.68171948313319e-05
## paxgene rna  MSSM    PC2 animal.registration.sex -0.536583012168326  1.41298167924018e-05
## paxgene rna  MSSM    PC3 acute.test.weight   0.509985571896437   4.32720495649275e-05
## paxgene rna  MSSM    PC3 animal.registration.sex 0.606616879399508   4.48050445617242e-07
## heart    MSSM    PC1 animal.key.timepoint    -0.614863122771119  2.8221460502702e-07
## heart    MSSM    PC3 animal.key.timepoint    -0.55223125749087   6.98847006483592e-06
## heart    MSSM    PC4 acute.test.weight   0.766901612430038   2.23415886205674e-12
## heart    MSSM    PC4 animal.registration.sex 0.798180104473037   6.22240139130505e-14
## heart    MSSM    PC4 acute.test.distance -0.478149183926571  0.000147001307937501
## heart    MSSM    PC5 acute.test.weight   -0.670646566940865  8.43309704397115e-09
## heart    MSSM    PC5 animal.registration.sex -0.651932911137332  2.9678294678719e-08
## heart    MSSM    PC5 acute.test.distance 0.523865931276648   2.4413540587121e-05
## kidney   MSSM    PC1 acute.test.weight   -0.746592044008409  1.71945428766554e-11
## kidney   MSSM    PC1 animal.registration.sex -0.866154152079774  1.64937863897971e-18
## kidney   MSSM    PC1 acute.test.distance 0.54687812980818    8.92706606619304e-06
## kidney   MSSM    PC2 animal.key.timepoint    0.725558407530845   1.17317359390604e-10
```

3 Differential abundance analysis (FPKM)

In this analysis we use simple linear regression to model the log fold changes. Here we perfom a simple analysis excluding the control groups (sampled in two time points only). Assume that we are analyzing a specific gene with a gene expression pattern x in a given tissue. We use the following covatiates in the model: (1) y_{sex} - sex, (2) lt_i - time, linear trend, (3) qt_i - time, quadratic trend, (4) y_d - distance acheived by the rat during the acute test, and (5) Z - technical variables. That is:

$$x_i = \beta_0 + \beta_{sex} * y_{sex} + \beta_l * lt_i + \beta_q * qt_i + \beta_d * y_d + \beta_{\mathbf{z}}^{\mathbf{T}} \mathbf{Z}_{\mathbf{i},*}$$

.

Here is the code for this analysis:

```
# store all betas and their p-values
source("/Users/David/Desktop/repos/motrpac/tools/association_analysis_functions.R")
Z_cols = c("pct_multimapped","pct_umi_dup",
           "median_5_3_bias","pct_rRNA","Lib_batch_ID")
clinical_cols = c("animal.registration.sex","animal.key.timepoint",
                  "animal.key.is_control","acute.test.distance")
tissue2diff_analysis_results = list()
rnaseq_meta_batchs = unique(rnaseq_meta[,c("GET_site","Tissue")])
for(i in 1:nrow(rnaseq_meta_batchs)){
  curr_site = rnaseq_meta_batchs[i,1]
  curr_tissue = rnaseq_meta_batchs[i,2]
  curr_samples = as.character(rnaseq_meta$vial_label[rnaseq_meta$GET_site==curr_site &
                                        rnaseq_meta$Tissue==curr_tissue])
  # For rat data, take samples whose label id starts with "9"
  curr_samples = curr_samples[grepl("^9",curr_samples)]
  curr_data = process_fpkm1(fpkm_all[,curr_samples])
  curr_data = run_quantile_normalization(curr_data)
  curr_data = as.matrix(curr_data)
  curr_meta1 = rnaseq_meta[curr_samples,Z_cols]
  curr_meta2 = merged_dmaqc_data[curr_samples,clinical_cols]
  tp_poly = poly(curr_meta2$animal.key.timepoint,degree = 2)
  colnames(tp_poly) = c("time.linear","time.quad")
  X = cbind(curr_meta2,curr_meta1,tp_poly)
  form = y ~ time.linear + time.quad +
    animal.registration.sex + pct_multimapped + acute.test.distance +
    pct_umi_dup + median_5_3_bias + pct_rRNA

  # to exclude the controls before the analysis:
  non_control_inds = !curr_meta2$animal.key.is_control
  # hist(X$distance[non_control_inds])
  lm_results = t(apply(curr_data[,non_control_inds],1,
                       lm_wrapper_for_diff_abundance_analysis,
                       x=X[non_control_inds,],form=form))

  curr_name = paste(curr_tissue,curr_site,sep=",")
  # Save the results and the analysis input into a single object
  tissue2diff_analysis_results[[curr_name]] = list(
    diff_res = lm_results,
    ge_data = curr_data,
    X = X,
    form=form,
```

```
    non_control_inds = non_control_inds
  )
}

# Save the results to our google bucket
save(tissue2diff_analysis_results,
       file="tissue2diff_analysis_results.RData")
system(paste("~/google-cloud-sdk/bin/gsutil", "cp tissue2diff_analysis_results.RData",
             "gs://bic_data_analysis/pass1a/rnaseq/"))
system("rm tissue2diff_analysis_results.RData")

# Add simple ANOVA
tissue2diff_analysis_results_anova = list()
tissue2diff_analysis_results_kw = list()
for(i in 1:nrow(rnaseq_meta_batchs)){
  curr_site = rnaseq_meta_batchs[i,1]
  curr_tissue = rnaseq_meta_batchs[i,2]
  curr_samples = as.character(rnaseq_meta$vial_label[rnaseq_meta$GET_site==curr_site &
                                          rnaseq_meta$Tissue==curr_tissue])
  # For rat data, take samples whose label id starts with "9"
  curr_samples = curr_samples[grepl("^9",curr_samples)]
  curr_data = process_fpkm1(fpkm_all[,curr_samples])
  curr_data = run_quantile_normalization(curr_data)
  curr_data = as.matrix(curr_data)
  curr_meta2 = merged_dmaqc_data[curr_samples,clinical_cols]
  y = curr_meta2$animal.key.timepoint
  # to exclude the controls before the analysis:
  non_control_inds = !curr_meta2$animal.key.is_control
  y = curr_meta2$animal.key.timepoint[non_control_inds]
  curr_data = curr_data[,non_control_inds]
  curr_name = paste(curr_tissue,curr_site,sep=",")

  curr_res1 = apply(curr_data,1,compare_multigroup_means,y = y)
  curr_res2 = apply(curr_data,1,compare_multigroup_means,
                    y = y,f = kruskal.test)

  tissue2diff_analysis_results_anova[[curr_name]] = curr_res1
  tissue2diff_analysis_results_kw[[curr_name]] = curr_res2
}
save(tissue2diff_analysis_results_anova,tissue2diff_analysis_results_kw,
       file="tissue2diff_analysis_results_simple_univariate.RData")
system(paste("~/google-cloud-sdk/bin/gsutil",
             "cp tissue2diff_analysis_results_simple_univariate.RData",
             "gs://bic_data_analysis/pass1a/rnaseq/"))

# Add simple Kruskal Wallis
```

We next inspect the results of the analysis. At a first step, we inspect the p-value distributions of the time-associated differential analysis.

```
# load precomputed results
system(paste("~/google-cloud-sdk/bin/gsutil",
             "cp gs://bic_data_analysis/pass1a/rnaseq/tissue2diff_analysis_results.RData",
             "."))
```

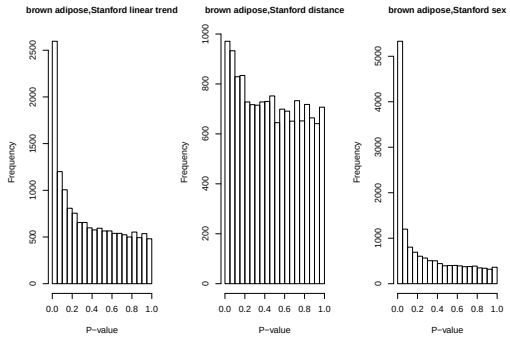
```
load("tissue2diff_analysis_results.RData")
print("number of analyzed genes in each tissue:")
```

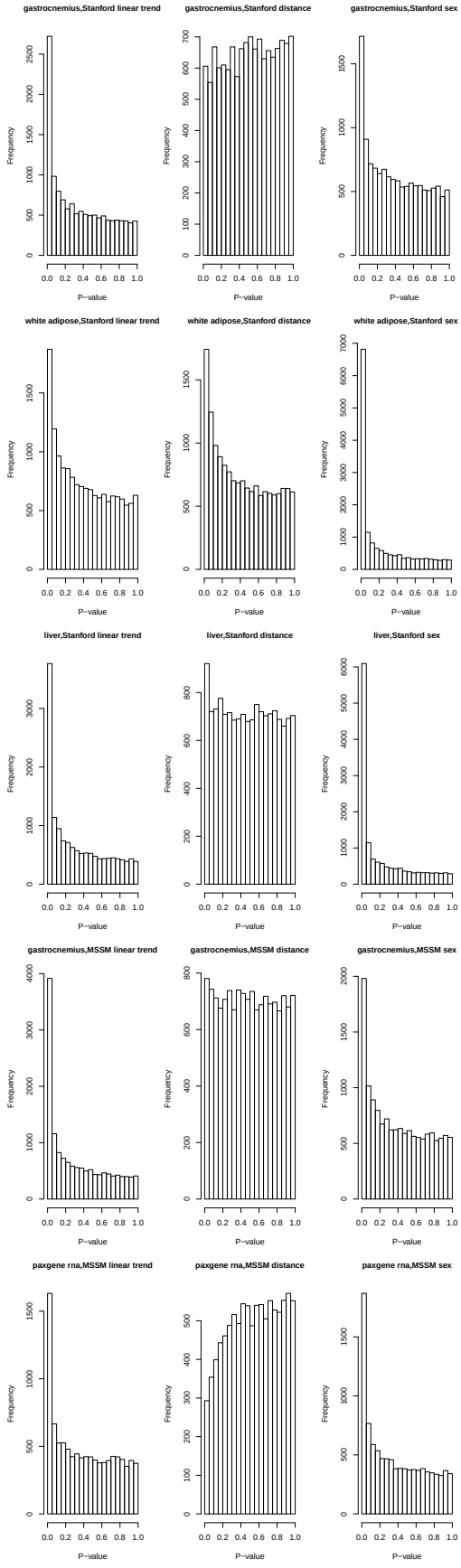
```
## [1] "number of analyzed genes in each tissue:"
```

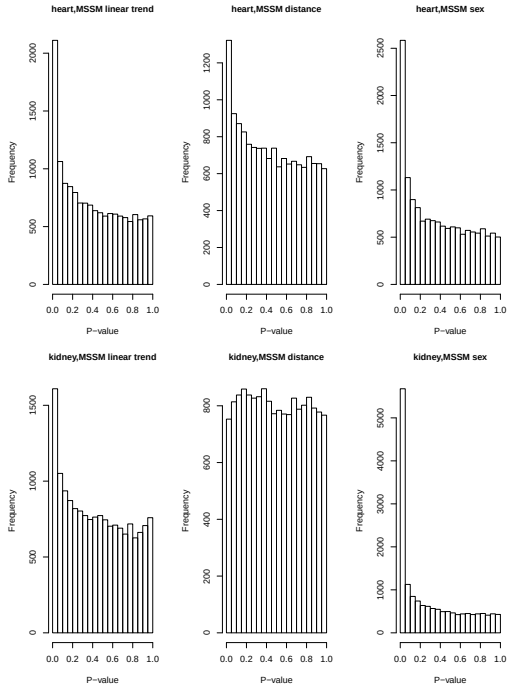
```
print(sapply(tissue2diff_analysis_results,function(x)dim(x[[1]])))
```

```
##      brown adipose,Stanford gastrocnemius,Stanford white adipose,Stanford
## [1,]                  14738                  12927                   15348
## [2,]                     18                     18                      18
##      liver,Stanford gastrocnemius,MSSM paxgene rna,MSSM heart,MSSM
## [1,]          14376              14188             9882      14888
## [2,]             18                 18               18         18
##      kidney,MSSM
## [1,]       16117
## [2,]          18
```

```
par(mfrow=c(1,3))
for(curr_name in names(tissue2diff_analysis_results)){
  lm_results = tissue2diff_analysis_results[[curr_name]][[1]]
  distance_col = grepl("distance",colnames(lm_results)) &
    grepl("pval",colnames(lm_results))
  sex_col = grepl("sex",colnames(lm_results)) &
    grepl("pval",colnames(lm_results))
  time_col = grepl("time.q",colnames(lm_results)) &
    grepl("pval",colnames(lm_results))
  hist(lm_results[,time_col],
       main=paste(curr_name,"linear trend"),xlab="P-value",
       cex.main=1)
  hist(lm_results[,distance_col],
       main=paste(curr_name,"distance"),xlab="P-value",
       cex.main = 1)
  hist(lm_results[,sex_col],
       main=paste(curr_name,"sex"),xlab="P-value",
       cex.main = 1)
}
```




gastrocnemius,Stanford linear trend
gastrocnemius,Stanford distance
gastrocnemius,Stanford sex
white adipose,Stanford linear trend
white adipose,Stanford distance
white adipose,Stanford sex
liver,Stanford linear trend
liver,Stanford distance
liver,Stanford sex
gastrocnemius,MSSM linear trend
gastrocnemius,MSSM distance
gastrocnemius,MSSM sex
paxgene rna,MSSM linear trend
paxgene rna,MSSM distance
paxgene rna,MSSM sex
Frequency
P−value


heart,MSSM linear trend
heart,MSSM distance
heart,MSSM sex
kidney,MSSM linear trend
kidney,MSSM distance
kidney,MSSM sex
Frequency
P−value

4 Site comparison using the Gastrocnemius samples

4.1 Simple comparison of the differential analysis results

As a first, simple analysis we load the differential analysis results and compare Stanford and Sinai.

```
# Load the data, get the shared genes
load(paste(getwd(),"tissue2diff_analysis_results.RData",sep="/"))
res_stanford = tissue2diff_analysis_results[["gastrocnemius,Stanford"]][[1]]
res_mssm = tissue2diff_analysis_results[["gastrocnemius,MSSM"]][[1]]
shared_genes = intersect(rownames(res_stanford),rownames(res_mssm))
print("Loaded the differential analysis results. The number of shared genes in the Stanford and Sinai r
```

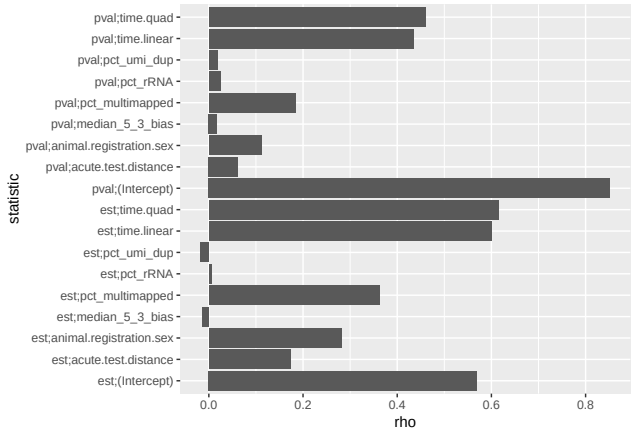
```
## [1] "Loaded the differential analysis results. The number of shared genes in the Stanford and Sinai
```

```
print(length(shared_genes))
```

```
## [1] 12917
```

Compare all statistics of the differential analysis. Here we expect a low correlation between the statistics of technical variables and higher correlation for clinical variables (e.g., time).

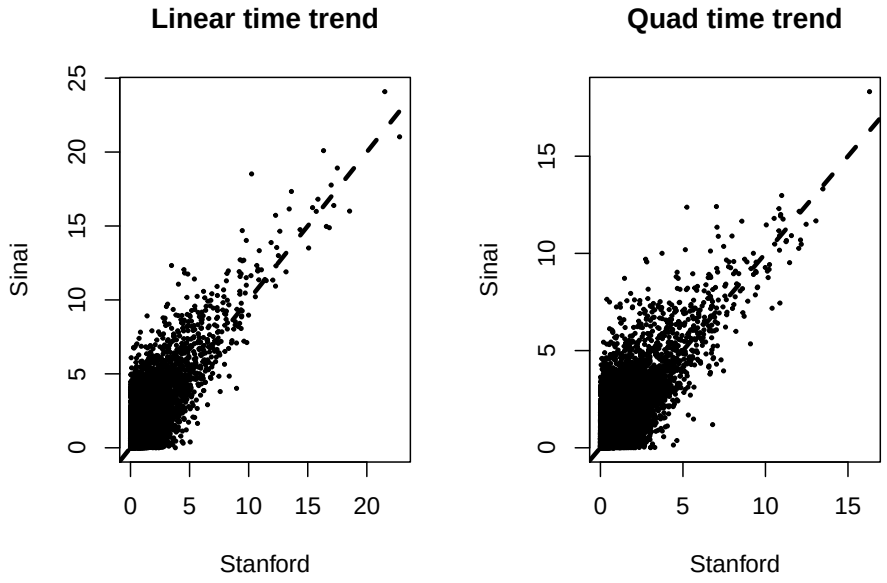
```
res_stanford = res_stanford[shared_genes,]
res_mssm = res_mssm[shared_genes,]
stanford_vs_sinai_diff_comparison_rho = diag(cor(res_mssm,res_stanford,method="spearman"))
p<-ggplot(data=data.frame(
          rho=stanford_vs_sinai_diff_comparison_rho,
          statistic = names(stanford_vs_sinai_diff_comparison_rho)),
          aes(x=statistic,y=rho)) + geom_bar(stat = "identity")
p + coord_flip()
```



Finally, compare the genes by looking at their p-values.

```
res_stanford = res_stanford[shared_genes,]
res_mssm = res_mssm[shared_genes,]
par(mfrow=c(1,2))
rho_linear = cor(-log(res_stanford[,"pval;time.linear"],10),
                 -log(res_mssm[,"pval;time.linear"],10))
plot(x = -log(res_stanford[,"pval;time.linear"],10),
     y = -log(res_mssm[,"pval;time.linear"],10),
     xlab="Stanford",ylab="Sinai",pch=20,cex=0.5,
     main="Linear time trend")
abline(0,1,lty=2,lwd=3)
```

```
plot(x = -log(res_stanford[,"pval;time.quad"],10),
     y = -log(res_mssm[,"pval;time.quad"],10),
     xlab="Stanford",ylab="Sinai",pch=20,cex=0.5,
     main="Quad time trend")
abline(0,1,lty=2,lwd=3)
```



4.2 Load the data

```
# tissue vector - used below for getting the subset of
# Gastrocnemius samples
tissue = rnaseq_meta$Tissue
names(tissue) = rownames(rnaseq_meta)
# get the Gastrocnemius samples from each site
Gastrocnemius_fpkm = lapply(site2fpkm,
    function(x,y)x[,grepl("Gastrocnemius",y[colnames(x)],ignore.case = T)],y=tissue)
print("Gastrocnemius samples, data dim:")
```

```
## [1] "Gastrocnemius samples, data dim:"
```

```
print(sapply(Gastrocnemius_fpkm,dim))
```

```
##      stanford sinai
## [1,]    32883 32883
## [2,]       80    80
```

```
# Process the FPKM data matrix from each site separately
Gastrocnemius_fpkm_processed = lapply(Gastrocnemius_fpkm,process_fpkm1)
print("Filtered FPKM data (separately for each site):")
```

```
## [1] "Filtered FPKM data (separately for each site):"
```

```
print(sapply(Gastrocnemius_fpkm_processed,dim))
```

```
##      stanford sinai
## [1,]    12977 14218
## [2,]       80    80
```

```
shared_genes = intersect(rownames(Gastrocnemius_fpkm_processed[[1]]),
                         rownames(Gastrocnemius_fpkm_processed[[2]]))
print("Number of shared genes that survive the filter above:")
```

```
## [1] "Number of shared genes that survive the filter above:"
```

```
print(length(shared_genes))
```

```
## [1] 12966
```

```
# Merge the datasets, store in a single data frame
Gastrocnemius_fpkm_mat = cbind(Gastrocnemius_fpkm_processed[[1]][shared_genes,],
                        Gastrocnemius_fpkm_processed[[2]][shared_genes,])

# Analysis of the metadata
Gastrocnemius_metadata = rnaseq_meta[colnames(Gastrocnemius_fpkm_mat),]
# We by default keep the vial sample id, which is different even if
# the biospecimen id is the same.
# This vector keeps the BID+PIDs
sample_id = paste(Gastrocnemius_metadata$BID,Gastrocnemius_metadata$PID,sep=";")
names(sample_id) = rownames(Gastrocnemius_metadata)
print("Do we have a copy from each site?")
```

```
## [1] "Do we have a copy from each site?"
```

```
all(table(sample_id)==2) # QC: make sure we have two copies for each id
```

```
## [1] TRUE
```

```
# Reorder the data by the site and sample id
Gastrocnemius_metadata = Gastrocnemius_metadata[order(Gastrocnemius_metadata$site,sample_id),]
sample_id = sample_id[rownames(Gastrocnemius_metadata)]
```

4.3 QC scores: site comparison

Here we compare the two sites by taking all numeric qc scores. We compare the two sites using a paired non-parametric test (Wilcoxon).

```
metadata2site_pval = c()
site_ind = Gastrocnemius_metadata$site==Gastrocnemius_metadata$site[1]
# We go over all numeric columns in the metadata matrix and use
# a paired Wilcoxon test to estimate site differences
for(col in names(Gastrocnemius_metadata)){
  x = Gastrocnemius_metadata[[col]]
  if(! mode(x)=="numeric"){next}
  # data are ordered by site and sample id, which keeps the correct
  # order for the paired test
  x1 = x[site_ind];x2 = x[!site_ind] # define the two vectors
  if(!is.numeric(x1) || !is.numeric(x2)){next}
  sd1 = sd(x1,na.rm = T);sd2=sd(x2,na.rm = T)
  if(is.na(sd1)||is.na(sd2)){next}
  if(sd1==0 || sd2==0){next}
  # Need to try, some numeric columns are constants or have NAs
  metadata2site_pval[col] = wilcox.test(x1,x2,paired=T)$p.value
}
# Take the top 30 significant columns
```

```r
selected_qc_comparisons = sort(metadata2site_pval)[1:30]
# Some of the columns are not informative (e.g., date)
# Take the "pct_" columns and print the p-values
print("Top pct_ qc scores that differ between sites:")
```

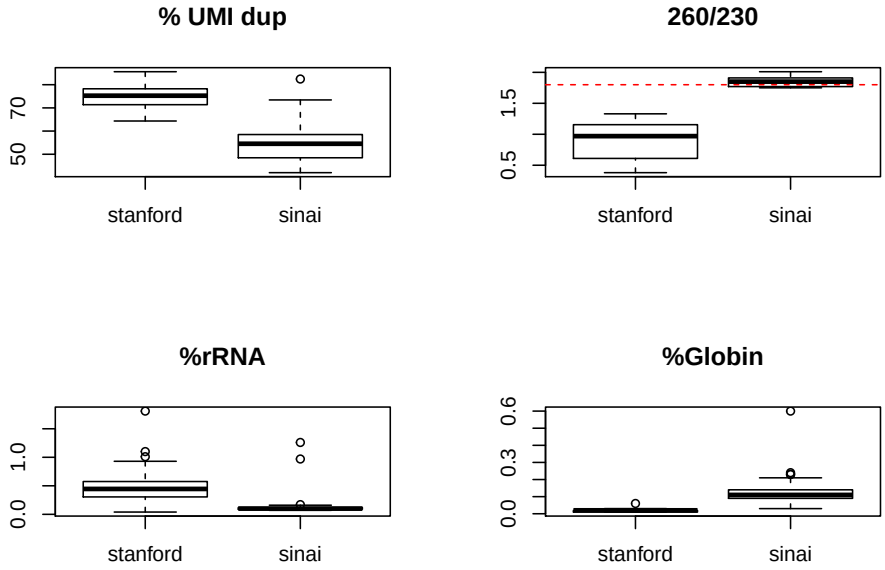
```
## [1] "Top pct_ qc scores that differ between sites:"
```

```r
print(selected_qc_comparisons[grepl("pct_",names(selected_qc_comparisons))])
```

```
##      pct_unmapped_other              pct_globin    pct_adapter_detected
##           4.280729e-15            7.749430e-15            7.985442e-15
##                pct_utr       pct_trimmed_bases         pct_picard_dup
##           7.989667e-15            7.993189e-15            8.628287e-15
##       pct_dup_sequence             pct_umi_dup              pct_coding
##           8.957509e-15            1.003521e-14            1.081158e-14
##               pct_chrX pct_multimapped_toomany                pct_rRNA
##           1.790211e-13            1.014781e-12            2.552097e-12
##    pct_uniquely_mapped         pct_multimapped                pct_mrna
##           3.844703e-12            7.674417e-12            3.366564e-10
##           pct_intronic          pct_intergenic                pct_chrM
##           5.172570e-10            6.216488e-09            4.088722e-08
##            pct_chrAuto   pct_unmapped_tooshort              pct_contig
##           3.172734e-07            4.687337e-07            4.814240e-03
##                 pct_GC
##           6.067464e-02
```

The plot below shows the site differences for selected scores.

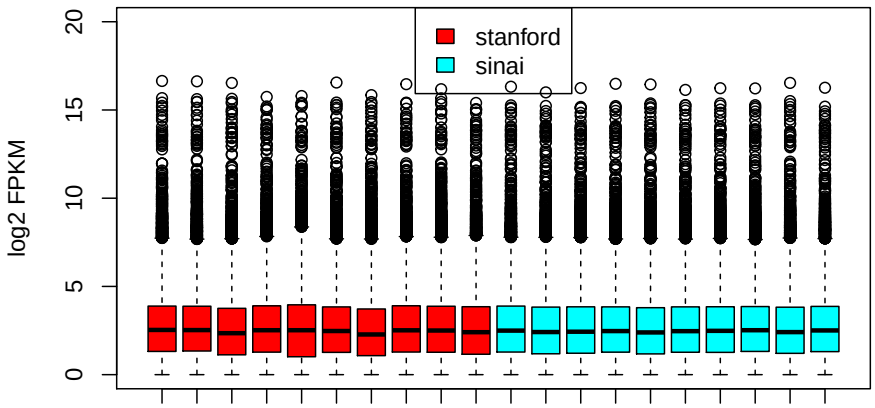
```r
# Comparison 1: selected qc scores
par(mfrow=c(2,2))
boxplot(pct_umi_dup~site,data=Gastrocnemius_metadata,main="% UMI dup")
boxplot(r_260_230~site,data=Gastrocnemius_metadata,main = "260/230")
abline(h = 1.8,lty=2,col="red")
boxplot(pct_rRNA~site,data=Gastrocnemius_metadata,main="%rRNA")
abline(h = 20,lty=2,col="red")
boxplot(pct_globin~site,data=Gastrocnemius_metadata,main="%Globin")
```



4.4 FPKM data comparison

We next compare the sites by looking at the boxplot of the sample data (after removing lowly expressed genes).

```
# Comparison 2: boxplots
#' Helper function to get a color set by a discrete vector
get_cols_vector_from_names<-function(v,pl_func = topo.colors){
  v = as.character(v)
  vals = unique(v)
  cols = pl_func(length(vals))
  names(cols) = vals
  newv = cols[v]
  return(list(newv,cols))
}
currcols = get_cols_vector_from_names(
  rnaseq_meta[colnames(Gastrocnemius_fpkm_mat),"site"],rainbow)
# Select a set of samples for the plot (too many samples otherwise)
inds_for_boxplot = c(1:10,81:90)
x_for_boxplot = Gastrocnemius_fpkm_mat[,inds_for_boxplot]
boxplot(x_for_boxplot,names=rep("",ncol(x_for_boxplot)),
        col=currcols[[1]][inds_for_boxplot],
        ylim = c(0,20),ylab="log2 FPKM") # extend lim to have room for legend
legend(x="top",names(currcols[[2]]),fill=currcols[[2]])
```

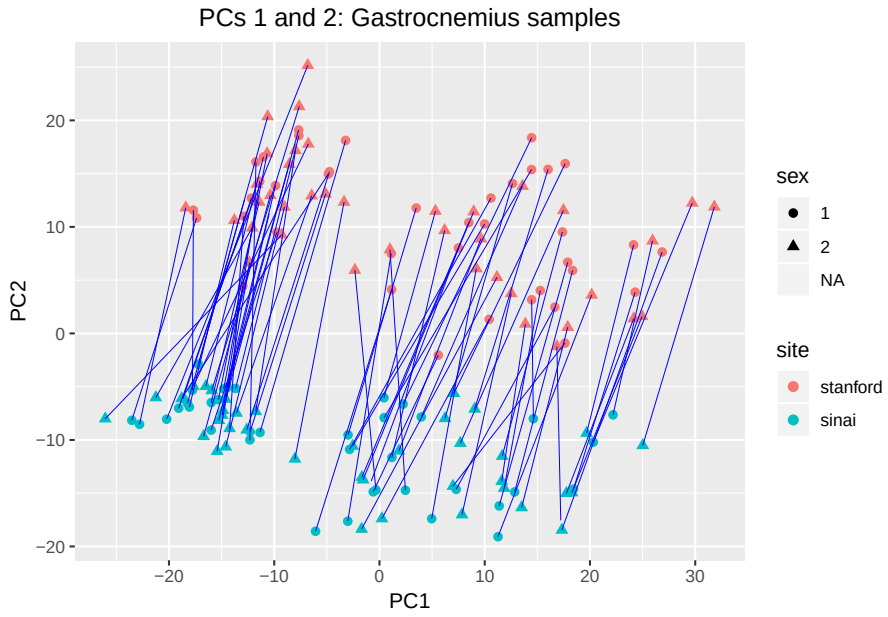


We next plot the PCA of the Gastrocnemius data, coloring the samples by site, shapre corresponds to sex. Two samples had NA for PID (controls?) and where excluded.

```
# Comparison 3: PCA
# Attempt 1: Take the FPKM data and run the PCA
Gastrocnemius_fpkm_pca = prcomp(t(Gastrocnemius_fpkm_mat))
Gastrocnemius_fpkm_pcax = Gastrocnemius_fpkm_pca$x
df = data.frame(Gastrocnemius_fpkm_pcax[,1:10],
                shape=Gastrocnemius_metadata$site, site=Gastrocnemius_metadata$site,
                sample = sample_id[rownames(Gastrocnemius_fpkm_pcax)],
                sex = as.factor(
                  merged_dmaqc_data[rownames(Gastrocnemius_metadata),"animal.registration.sex"]
                ))
df = df[order(df$sample),]
# Add lines between matching samples (i.e., same sample, different site)
ggplot(df,aes(x=PC1, y=PC2, shape=sex, color=site,group=sample)) +
  geom_point(size=2) + geom_path(size=0.02,color="blue") +
```

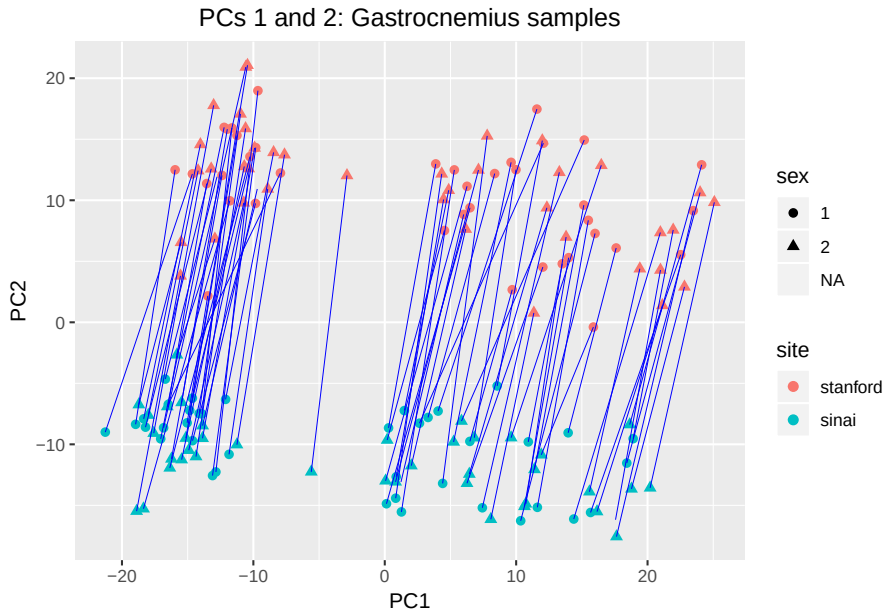
```
  ggtitle("PCs 1 and 2: Gastrocnemius samples") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
## Warning: Removed 4 rows containing missing values (geom_point).
```



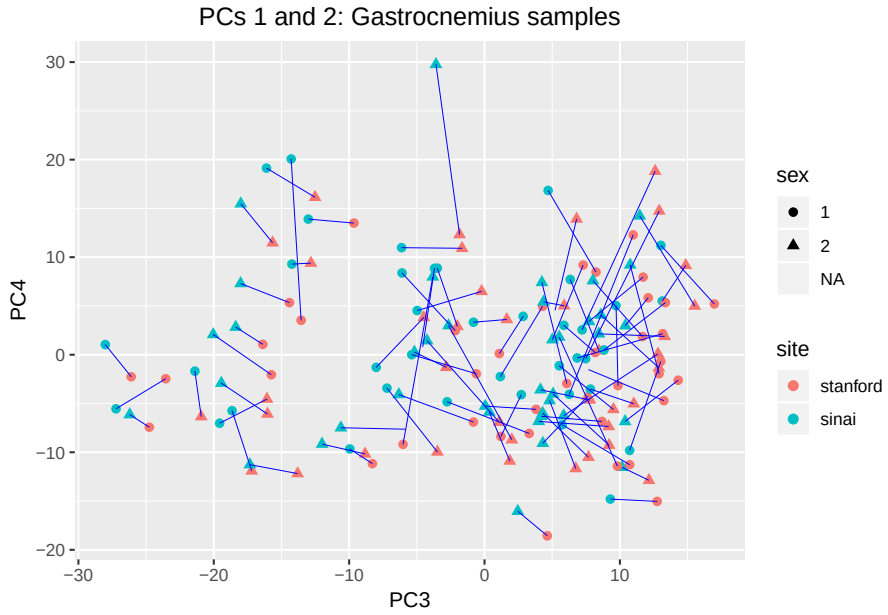

```
# Retry - quantile normalization before PCA (slightly cleaner)
Gastrocnemius_fpkm_mat_q = run_quantile_normalization(Gastrocnemius_fpkm_mat)
Gastrocnemius_fpkm_pca = prcomp(t(Gastrocnemius_fpkm_mat_q))
Gastrocnemius_fpkm_pcax = Gastrocnemius_fpkm_pca$x
df = data.frame(Gastrocnemius_fpkm_pcax[,1:10],
                shape=Gastrocnemius_metadata$site, site=Gastrocnemius_metadata$site,
                sample = sample_id[rownames(Gastrocnemius_fpkm_pcax)],
                sex = as.factor(
                  merged_dmaqc_data[rownames(Gastrocnemius_metadata),"animal.registration.sex"]
                ))
df = df[order(df$sample),]
# Add lines between matching samples (i.e., same sample, different site)
ggplot(df,aes(x=PC1, y=PC2, shape=sex, color=site,group=sample)) +
  geom_point(size=2) + geom_path(size=0.02,color="blue") +
  ggtitle("PCs 1 and 2: Gastrocnemius samples") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
## Warning: Removed 4 rows containing missing values (geom_point).
```




```
ggplot(df,aes(x=PC3, y=PC4, shape=sex, color=site,group=sample)) +
  geom_point(size=2) + geom_path(size=0.02,color="blue") +
  ggtitle("PCs 1 and 2: Gastrocnemius samples") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
## Warning: Removed 4 rows containing missing values (geom_point).
```



4.5 Sample correlation

We next compute the Spearman correlation between the samples (with and without filterling lowly expressed genes). We separate the correlations to those between different samples and those between cross-site replicates.

```
par(mfrow=c(1,2))
# Raw FPKMs
```

```
x1 = site2fpkm[[1]][,colnames(Gastrocnemius_fpkm_processed[[1]])]
x2 = site2fpkm[[2]][,colnames(Gastrocnemius_fpkm_processed[[2]])]
x1 = x1[,order(sample_id[colnames(x1)])]
x2 = x2[,order(sample_id[colnames(x2)])]
print("Do we have the same mapped sample id in the matrices?")
```

```
## [1] "Do we have the same mapped sample id in the matrices?"
```

```
print(all(sample_id[colnames(x1)] == sample_id[colnames(x2)]))
```

```
## [1] TRUE
```

```
corrs = cor(x1,x2,method="spearman")
l = list(
  replicates = diag(corrs),
  between = corrs[lower.tri(corrs,diag = F)]
)
boxplot(l,col=c("gray","green"),
        main="Sample corr (Spearman), raw FPKM",
        cex.main=0.9)

# Processed FPKMs - take the expressed genes
corrs = cor(x1[shared_genes,],x2[shared_genes,],method="spearman")
l = list(
  replicates = diag(corrs),
  between = corrs[lower.tri(corrs,diag = F)]
)
boxplot(l,col=c("gray","green"),
        main="Sample corr (Spearman), filtered FPKM",
        cex.main=0.9)
```