Phase 1A: RNA-seq data analysis

David Amar, Archana Raja

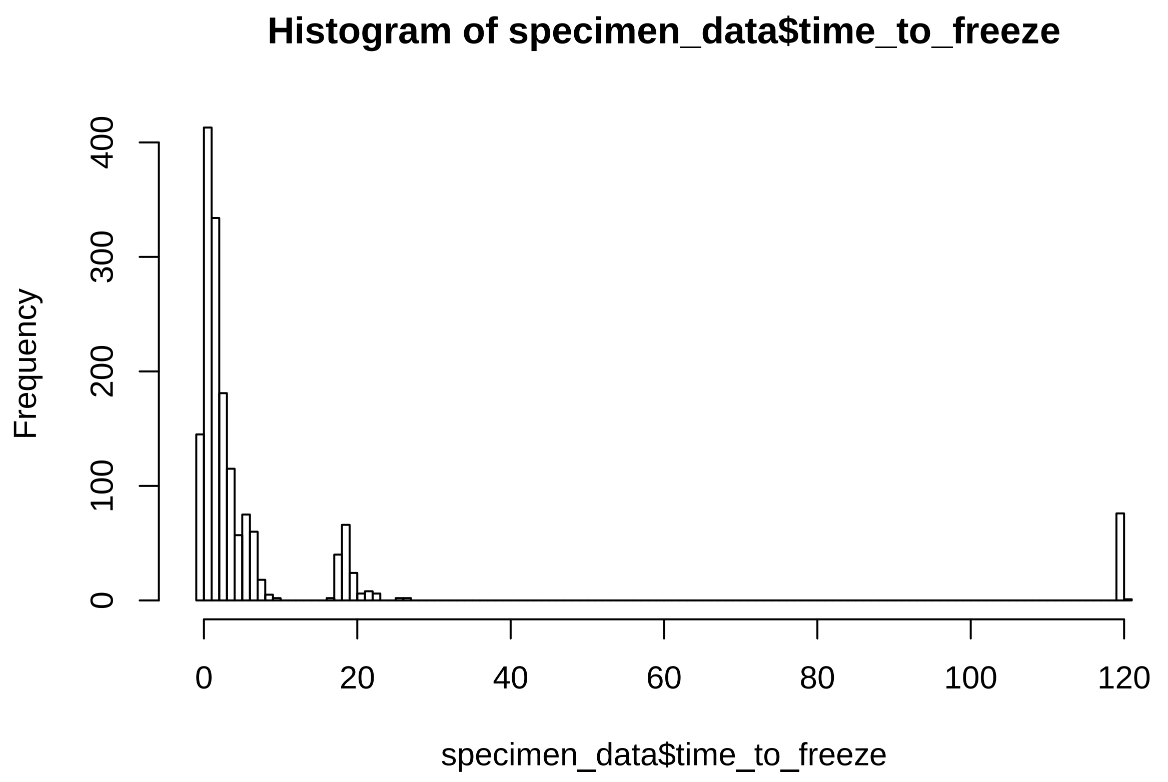
Phase 1A raw data was preprocessed at the BIC using our pipeline, which was implemented according to the MOP. Here we present QC analyses performed on the output of the pipeline.

1 Input data

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

```
setwd("/Users/David/Desktop/MoTrPAC/data/pass_1a/rnaseq/")
library(data.table);library(DESeq2)
library(preprocessCore);library(ggplot2)

# 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))

# Metadata of the animals and biospecimen: analyze the DMAQC data directly
# This is a merged data frame containing the animal key and
# registry data from dmaqc, created using:
dmaqc_metadata_path =
  "/Users/David/Desktop/MoTrPAC/data/pass_1a/dmaqc_pheno/"
dmaqc_files = list.files(dmaqc_metadata_path,full.names = T)
animal_key_file = dmaqc_files[grepl("Animal.Key",dmaqc_files)]
animal_regstr_file = dmaqc_files[grepl("Regis",dmaqc_files)]
animal_acute_test_file = dmaqc_files[grepl("Acute",dmaqc_files)]
animal_acute_test_data = read.csv(animal_acute_test_file,stringsAsFactors = F)
animal_key_data = read.csv(animal_key_file,stringsAsFactors = F)
animal_reg_data = read.csv(animal_regstr_file,stringsAsFactors = F)
rownames(animal_key_data) = animal_key_data$pid
rownames(animal_reg_data) = animal_reg_data$pid
rownames(animal_acute_test_data) = animal_acute_test_data$pid
animal_metadata = cbind(animal_key_data,animal_reg_data[rownames(animal_key_data),],
```

```r
                        animal_acute_test_data[rownames(animal_key_data),])
# Show some fields
print("Animal metadata, time label table:")
print(table(animal_metadata$ANIRandGroup))
print("Animal metadata, sex table:")
print(table(animal_metadata$sex))

# Analysis of biospecimen data
specimen_data_path = dmaqc_files[grepl("Specimen.Processing.csv",dmaqc_files)]
specimen_data = read.csv(specimen_data_path,stringsAsFactors = F)
# Load the labelid to vialid mapping
bic_label_ids = read.csv(dmaqc_files[grepl("BIC",dmaqc_files)])
label2vial = as.character(bic_label_ids$vialLabel)
names(label2vial) = as.character(bic_label_ids$labelID)
specimen_data = specimen_data[is.element(specimen_data$labelid,set=names(label2vial)),]
dim(specimen_data)
rownames(specimen_data) = label2vial[as.character(specimen_data$labelid)]

# Parse the times and compute the difference between the freeze time and
# the collection time
time_to_freeze1 = as.difftime(specimen_data$t_freeze,units = "mins") -
  as.difftime(specimen_data$t_collection,units="mins")
# For some samples we have the edta spin time instead of the collection
# time, use these when there are no other options
time_to_freeze2 = as.difftime(specimen_data$t_freeze,units = "mins") -
  as.difftime(specimen_data$t_edtaspin,units="mins")
time_to_freeze = time_to_freeze1
# Fill in the NAs by taking the time between the edta spin and the freeze
table(is.na(time_to_freeze1),is.na(time_to_freeze2))
time_to_freeze[is.na(time_to_freeze1)] = time_to_freeze2[is.na(time_to_freeze1)]
specimen_data$time_to_freeze = as.numeric(time_to_freeze)
hist(specimen_data$time_to_freeze,breaks=100)
```



```
print("Histogram of freeze times, computed from the DMAQC data")

# 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
}

# At this point we have all count and FPKM data as well as three shources
# of non-transcriptomic data:
# rnaseq_meta (rownames are vial_label, which is labelid) - assay qc scores
# animal_metadata (rownames are pids) - animal/subject level data like sex, weight, etc.
# Note that animal and sample here are the same. The human data will be different as we
# will have repeated samples.
# specimen_data (rownames are labelid, which is vial_label) -
#     bioscpecimen level data like time to freeze

# # Some tests - need to make sure all metadata has the same info
# length(intersect(as.character(bic_label_ids$vialLabel),rnaseq_meta$vial_label))
# setdiff(rnaseq_meta$vial_label,as.character(bic_label_ids$vialLabel))
# # As of June 2019: ignore the biospecimen data and move on with the animal data
# test_bid = intersect(rnaseq_meta$BID,specimen_data$bid)[1]
# rnaseq_meta$vial_label[grepl(test_bid,rnaseq_meta$vial_label)]
# specimen_data$labelid[grepl(test_bid,specimen_data$labelid)]
```

2 PCA plots (all samples)

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
```

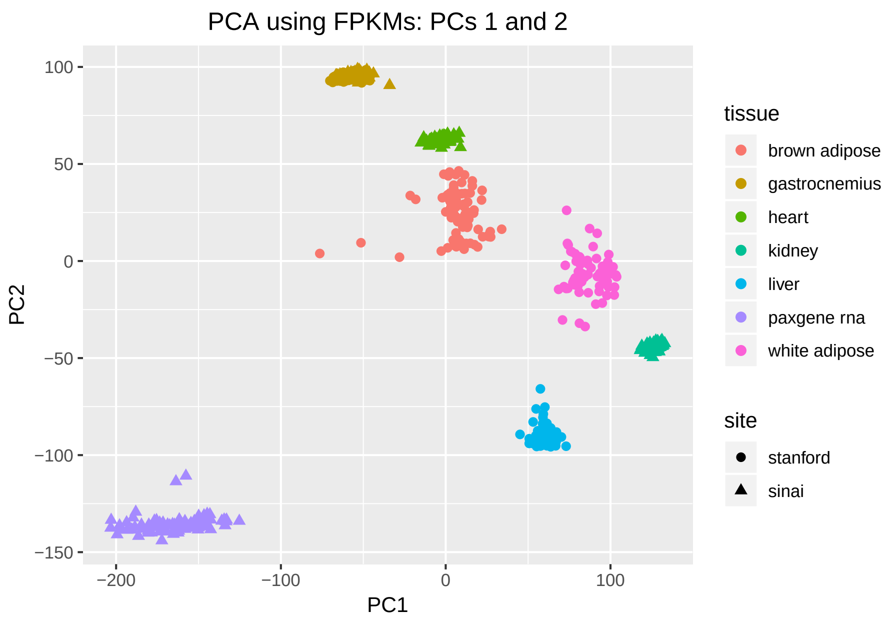
```
proc_fpkms = cbind(site_proc_fpkms[[1]][shared_genes,],
                   site_proc_fpkms[[2]][shared_genes,])
# QC: make sure the metadata and the expression matrix have the same sample id:
print("do we have the same samples in the expression and metadata matrices?")
```

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

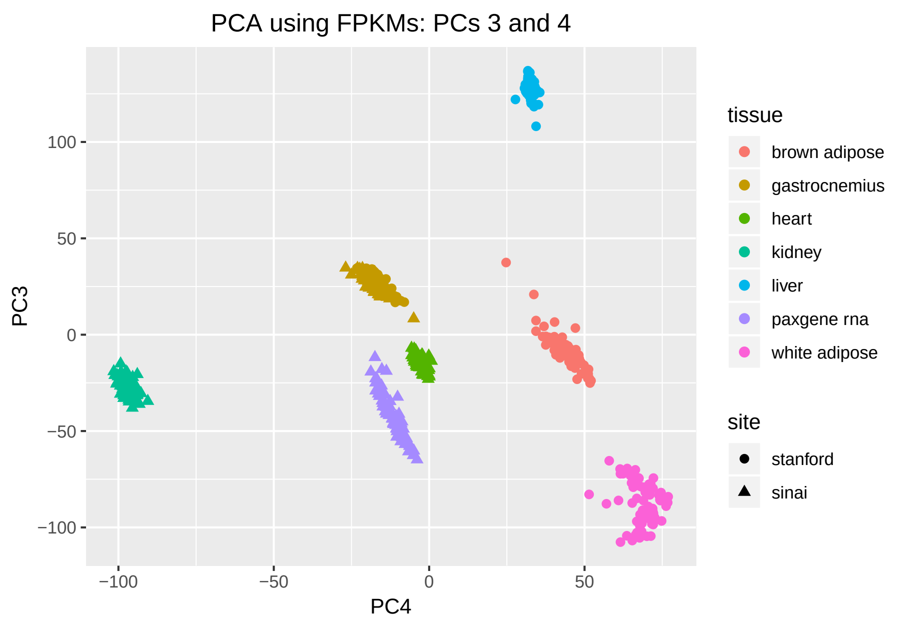
```
print(all(colnames(proc_fpkms) %in% rownames(rnaseq_meta)))
```

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

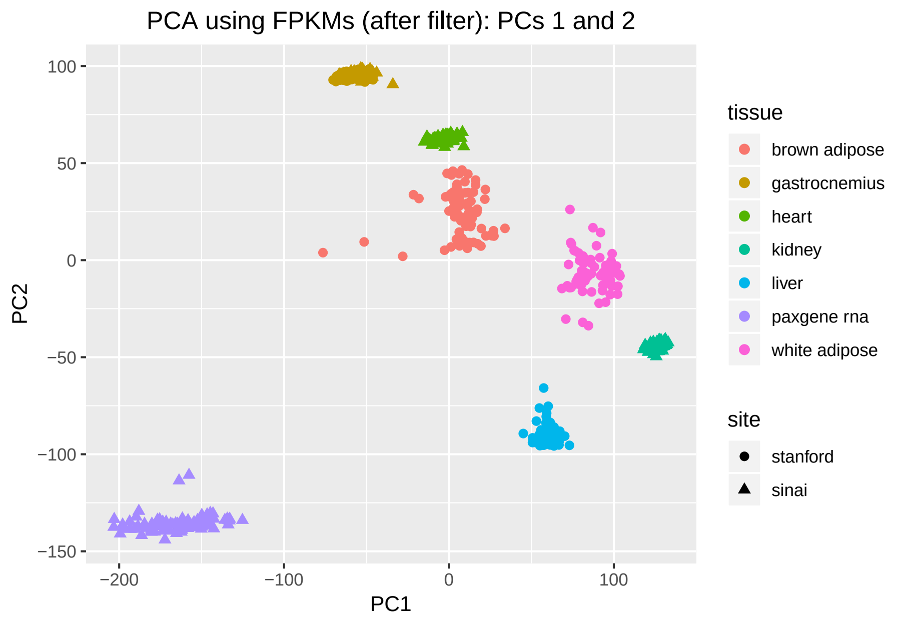
```
# Run the PCA: try all genes first
fpkm_all = cbind(site2fpkm[[1]],site2fpkm[[2]])
fpkm_all = log(fpkm_all+1,base=2)
fpkm_pca = prcomp(t(fpkm_all))
fpkm_pcax = fpkm_pca$x
df = data.frame(fpkm_pcax[,1:10],
                tissue = rnaseq_meta[rownames(fpkm_pcax),"Tissue"],
                site = rnaseq_meta[rownames(fpkm_pcax),"site"])
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))
```



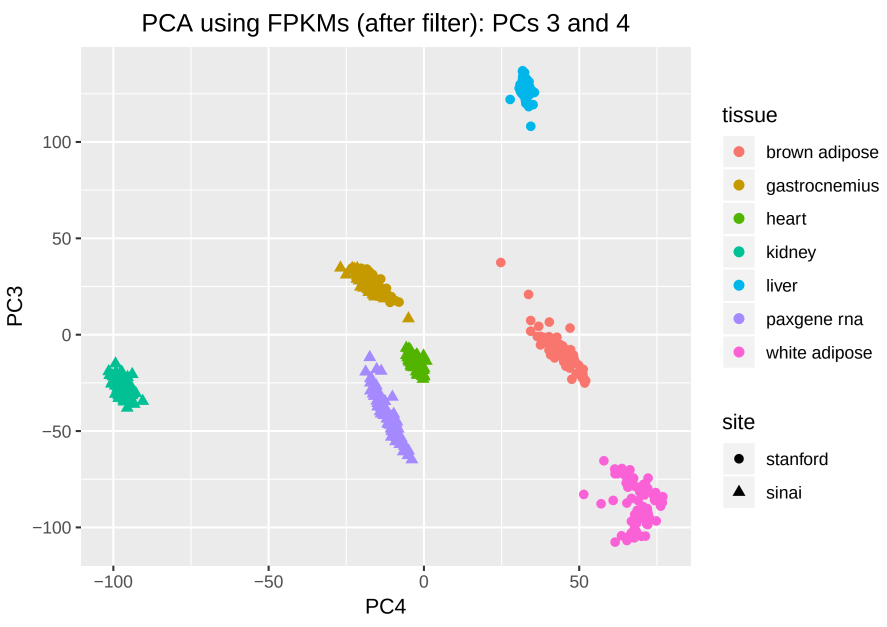
```
ggplot(df,aes(x=PC4, y=PC3,shape=site, color=tissue)) +
  geom_point(size=2) + ggtitle("PCA using FPKMs: PCs 3 and 4") +
  theme(plot.title = element_text(hjust = 0.5))
```



```r
# 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 = rnaseq_meta[rownames(fpkm_pcax),"Tissue"],
                site = rnaseq_meta[rownames(fpkm_pcax),"site"])
ggplot(df,aes(x=PC1, y=PC2,shape=site, color=tissue)) +
  geom_point(size=2) + ggtitle("PCA using FPKMs (after filter): PCs 1 and 2") +
  theme(plot.title = element_text(hjust = 0.5))
```



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



2.2 Normalized counts

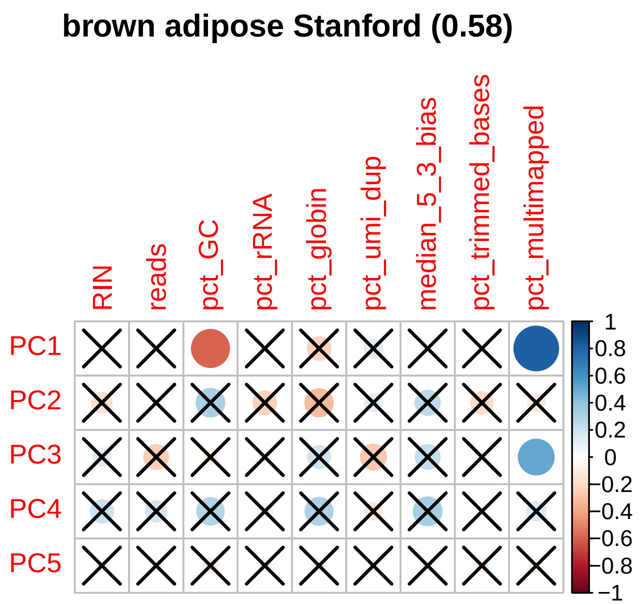
```
# Pipeline 2: work with count data
# Combine the two count matrices
count_matrix = as.matrix(cbind(site2counts[[1]],site2counts[[2]]))
#' Use DESeq2 to estimate sample factors and gene dispersion
#' @return a DESeqDataSet
process_counts<-function(count_matrix,plotFactors=T){
  mode(count_matrix) = "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)
```

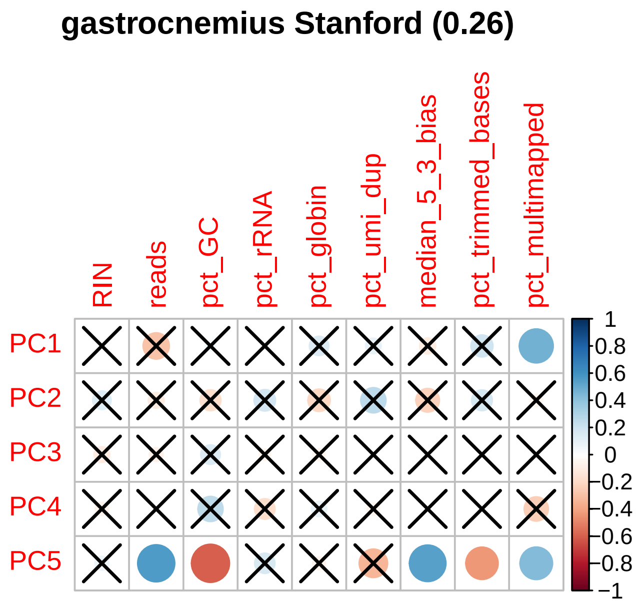
```r
pc2 <- prcomp( t( assay(vsd) ) )
counts_pcax2 = pc2$x
# PCA plots
df = data.frame(counts_pcax1[,1:10],
                tissue = rnaseq_meta[rownames(counts_pcax1),"Tissue"],
                site = rnaseq_meta[rownames(counts_pcax1),"site"])
ggplot(df,aes(x=PC1, y=PC2,shape=site, color=tissue)) +
  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))
```

3 Within tissue analysis

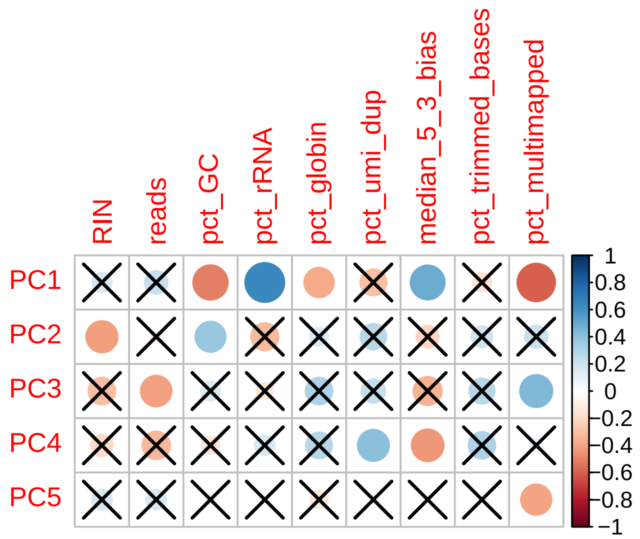
Here we take each tissue and analyze its samples. We quantile normalize the data and run PCA. For the top principal components (5) we compute their association with the metadata. Metadata in this context is all non-transcriptomic data, which includes technical and clinical variables.

```r
library(corrplot)
# load some auxiliary functions for association analysis
source("/Users/David/Desktop/repos/motrpac/tools/association_analysis_functions.R")
# 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
animal_data_cols_for_qc_analysis = c("siteID","weight",
                                     "Batch","sex","distance","howlongshock")

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$Tissue==curr_tissue])
  # For rat data, take samples whose label id starts with "9"
  curr_samples = curr_samples[grepl("^9",curr_samples)]
  curr_data = process_fpkm1(fpkm_all[,curr_samples])
  # curr_data = run_quantile_normalization(curr_data)
  curr_pca = prcomp(t(curr_data))
  curr_pcax = curr_pca$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")
  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))
}
```

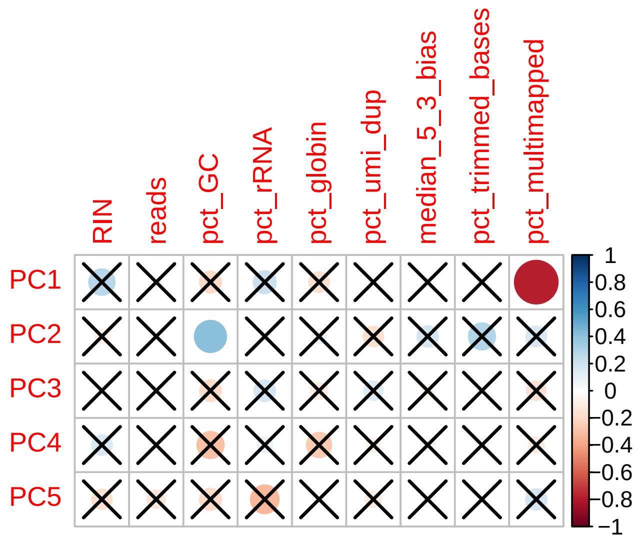

brown adipose Stanford (0.58)
RIN
reads
pct_GC
pct_rRNA
pct_globin
pct_umi_dup
median_5_3_bias
pct_trimmed_bases
pct_multimapped
PC1
PC2
PC3
PC4
PC5
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1

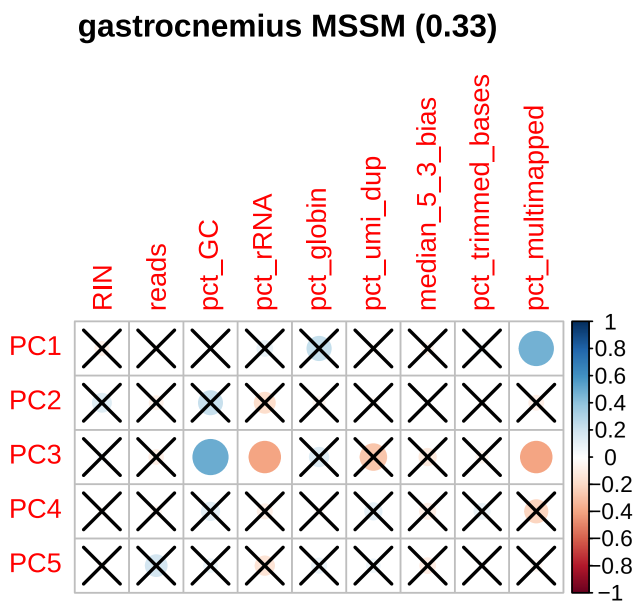

gastrocnemius Stanford (0.26)
RIN
reads
pct_GC
pct_rRNA
pct_globin
pct_umi_dup
median_5_3_bias
pct_trimmed_bases
pct_multimapped
PC1
PC2
PC3
PC4
PC5
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1

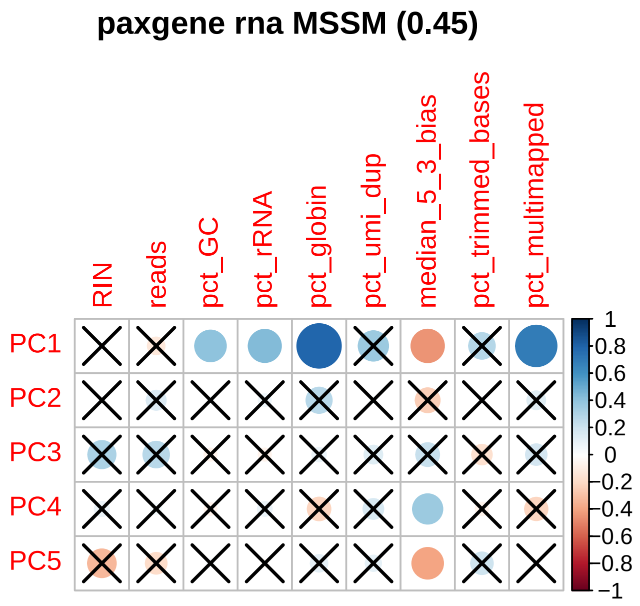
white adipose Stanford (0.58)

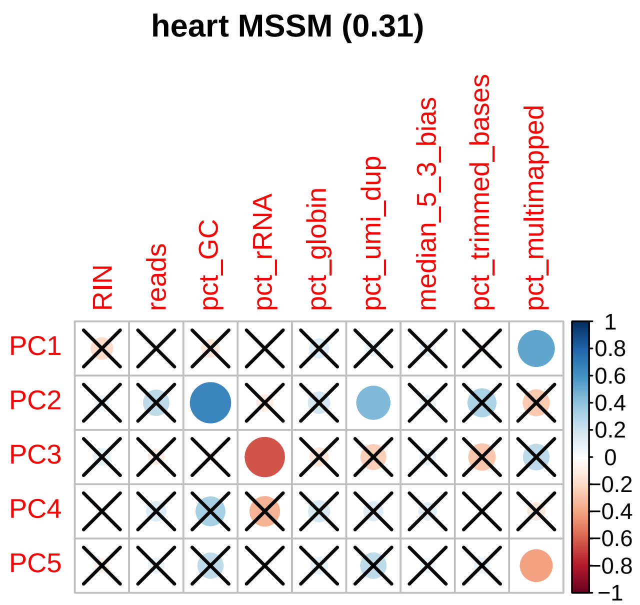


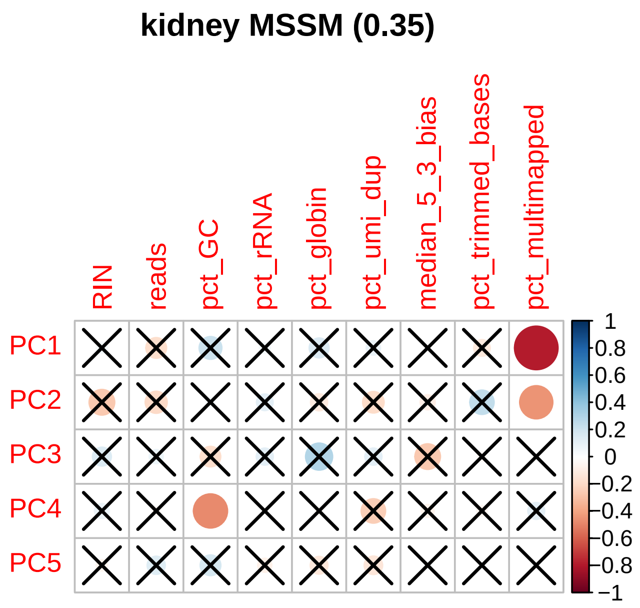
liver Stanford (0.46)




gastrocnemius MSSM (0.33)
RIN
reads
pct_GC
pct_rRNA
pct_globin
pct_umi_dup
median_5_3_bias
pct_trimmed_bases
pct_multimapped
PC1
PC2
PC3
PC4
PC5
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1


paxgene rna MSSM (0.45)
RIN
reads
pct_GC
pct_rRNA
pct_globin
pct_umi_dup
median_5_3_bias
pct_trimmed_bases
pct_multimapped
PC1
PC2
PC3
PC4
PC5
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1





4 Site comparison using the Gastrocnemius samples

4.1 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)]
```

4.2 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).

```r
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
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:")
```

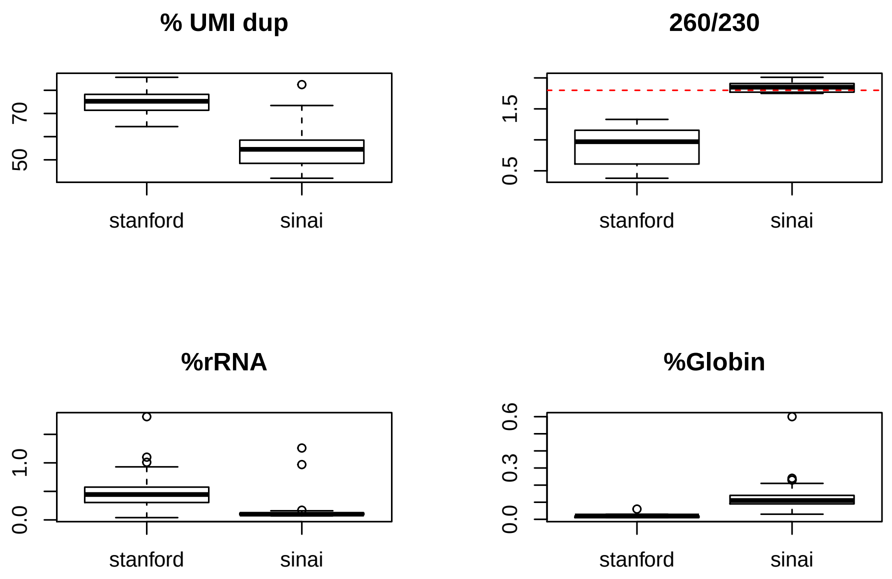
```
## [1] "Top pct_ qc scores that differ between sites:"
```

```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.

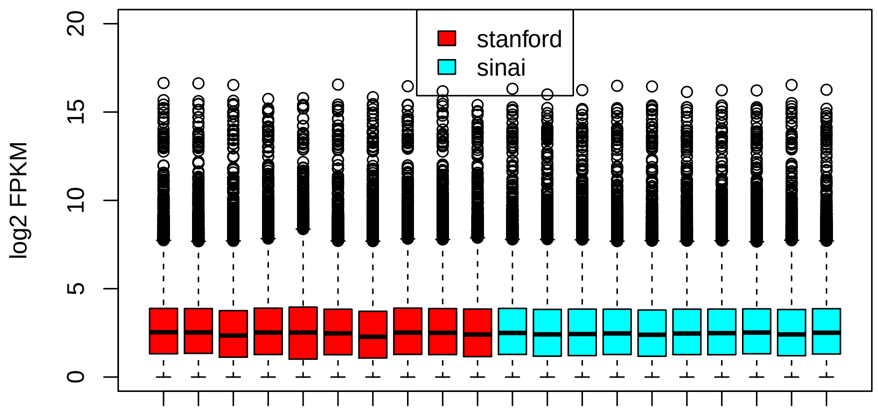
```r
# Comparison 1: selected qc scores
par(mfrow=c(2,2))
boxplot(pct_umi_dup~site,data=Gastrocnemius_metadata,main="% UMI dup")
boxplot(r_260_230~site,data=Gastrocnemius_metadata,main = "260/230")
abline(h = 1.8,lty=2,col="red")
boxplot(pct_rRNA~site,data=Gastrocnemius_metadata,main="%rRNA")
abline(h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



4.3 FPKM data comparison

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

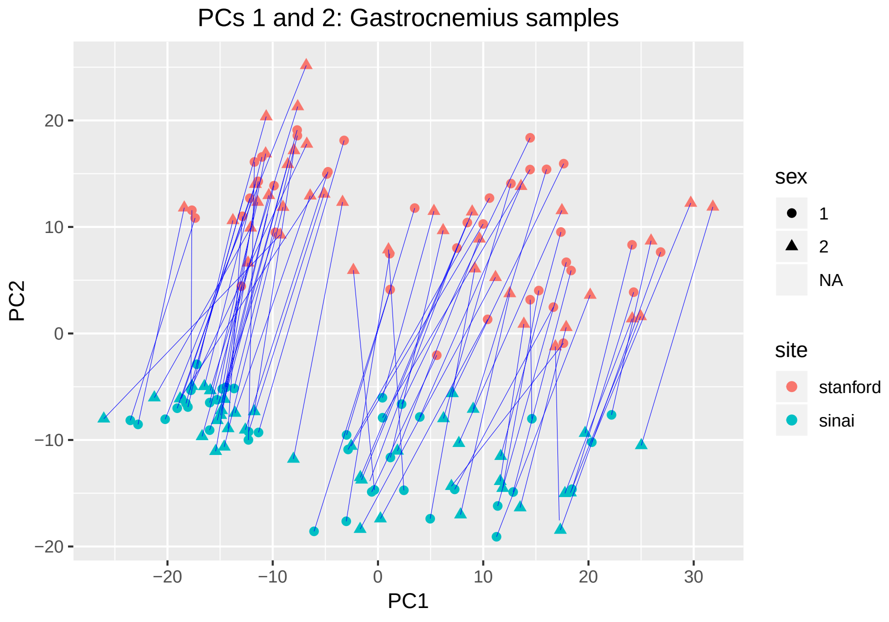
```
# Comparison 2: boxplots
#' Helper function to get a color set by a discrete vector
get_cols_vector_from_names<-function(v,pl_func = topo.colors){
  v = as.character(v)
  vals = unique(v)
  cols = pl_func(length(vals))
  names(cols) = vals
  newv = cols[v]
  return(list(newv,cols))
}
currcols = get_cols_vector_from_names(
  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$site,
                sample = sample_id[rownames(Gastrocnemius_fpkm_pcax)],
                sex = as.factor(animal_metadata[Gastrocnemius_metadata$PID,"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))
```

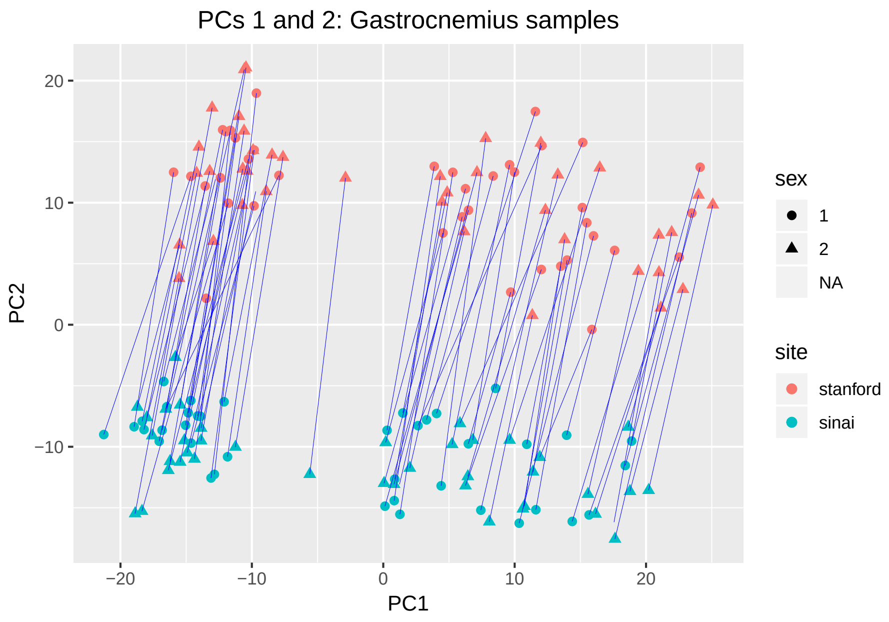
```
## Warning: Removed 4 rows containing missing values (geom_point).
```



```
# Retry - quantile normalization before PCA (slightly cleaner)
Gastrocnemius_fpkm_mat_q = run_quantile_normalization(Gastrocnemius_fpkm_mat)
```

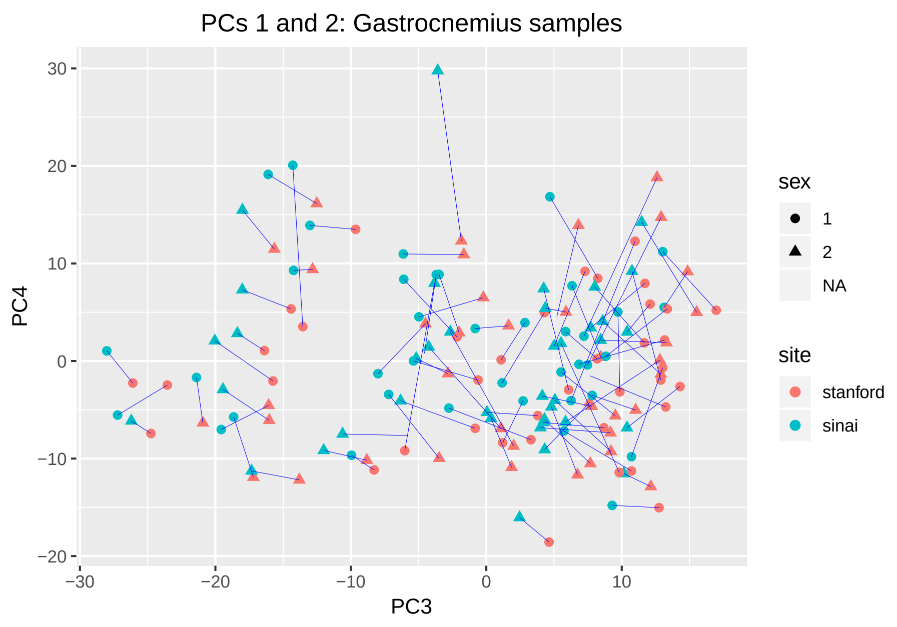
```r
Gastrocnemius_fpkm_pca = prcomp(t(Gastrocnemius_fpkm_mat_q))
Gastrocnemius_fpkm_pcax = Gastrocnemius_fpkm_pca$x
df = data.frame(Gastrocnemius_fpkm_pcax[,1:10],
                shape=Gastrocnemius_metadata$site, site=Gastrocnemius_metadata$site,
                sample = sample_id[rownames(Gastrocnemius_fpkm_pcax)],
                sex = as.factor(animal_metadata[Gastrocnemius_metadata$PID,"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).
```



```r
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).
```



4.4 Look at the 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.

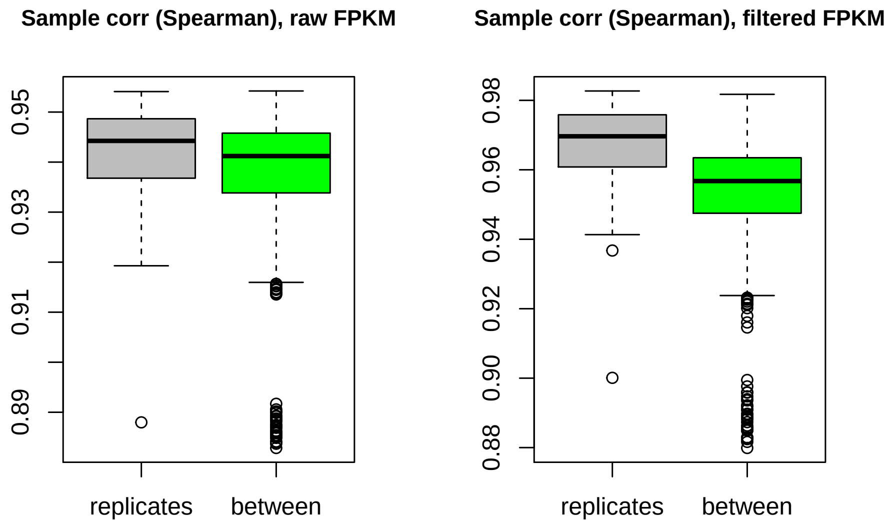
```
par(mfrow=c(1,2))
# Raw FPKMs
x1 = site2fpkm[[1]][,colnames(Gastrocnemius_fpkm_processed[[1]])]
x2 = site2fpkm[[2]][,colnames(Gastrocnemius_fpkm_processed[[2]])]
x1 = x1[,order(sample_id[colnames(x1)])]
x2 = x2[,order(sample_id[colnames(x2)])]
print("Do we have the same mapped sample id in the matrices?")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

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

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

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

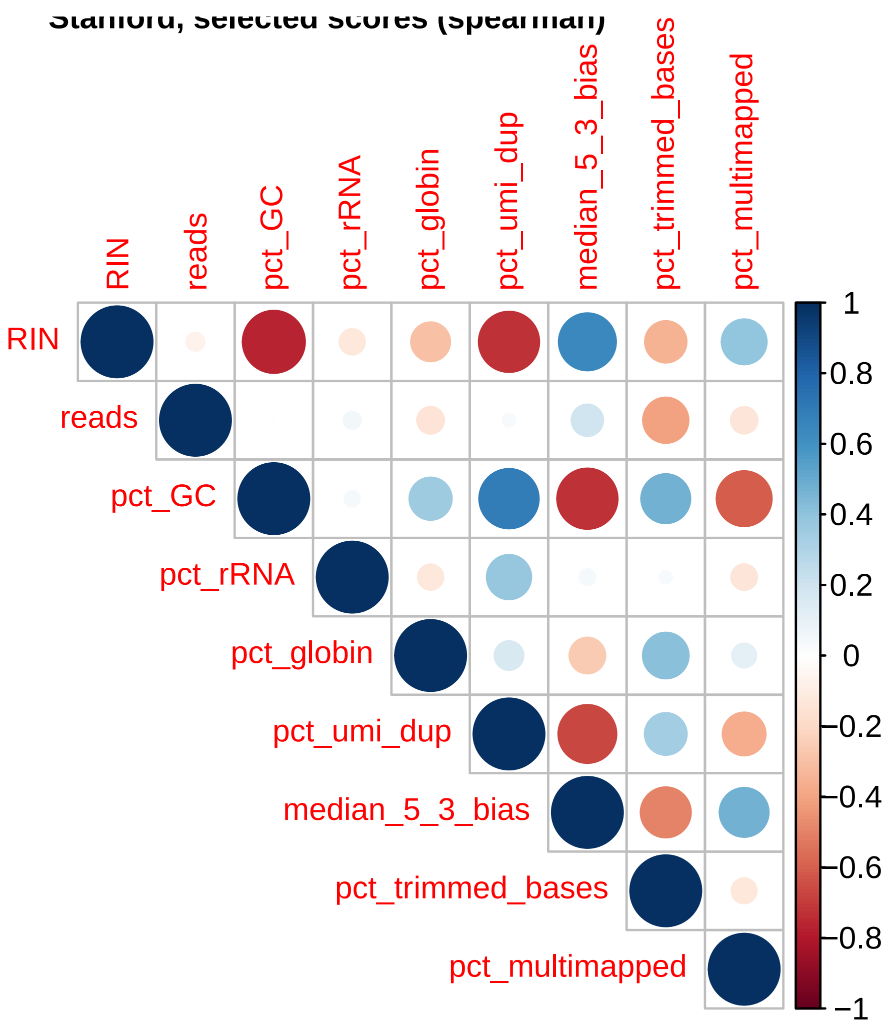
```
corrs = cor(x1,x2,method="spearman")
l = list(
  replicates = diag(corrs),
  between = corrs[lower.tri(corrs,diag = F)]
)
boxplot(l,col=c("gray","green"),
        main="Sample corr (Spearman), raw FPKM",
        cex.main=0.9)

# 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)
```



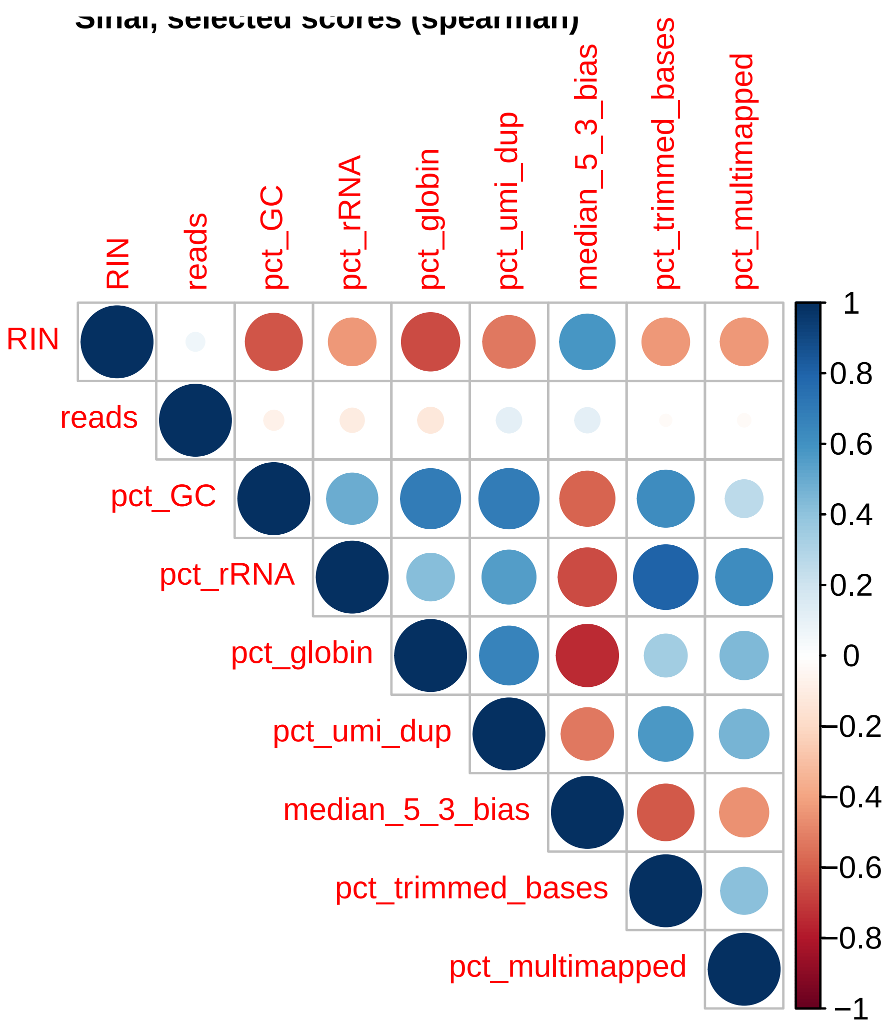
5 In progress

1. Looking at qc scores correlation (assay specific), for example for RNA-seq:

```
library(corrplot)
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")
x = rnaseq_meta[rnaseq_meta$site=="stanford",cols_for_qc_analysis]
corrplot(cor(x,method="spearman"),tl.cex = 0.8,type = "upper", main="Stanford, selected scores (spearma
```



```
x = rnaseq_meta[rnaseq_meta$site=="sinai",cols_for_qc_analysis]
corrplot(cor(x,method="spearman"),tl.cex = 0.8,type = "upper", main="Sinai, selected scores (spearman)"
```



3. Differential abundance analysis using LMMs
4. Site comparison by differential genes