SPECTRA application to CoMMpass

Rosalie Griffin Waller

27-OCT-2020

Code for SPECTRA approach and CD138+ spectra reported in "SPECTRA: AGNOSTIC EXPRESSION VARIABLES FOR FLEXIBLE TRANSCRIPTOME MODELING IN COMPLEX DISEASE" by Waller et al.

Application of the SPECTRA approach to the MMRF CoMMpass study.

This code provides the set of SPECTRA variables for use in various models with disparate outcomes.

See additional Rmd reports for code corresponding to: a) CD138+ spectra and established expression-based risk scores, b) CD138+ spectra and clinical risk, c) CD138+ spectra and disease course, d) CD138+ spectra and demographic risk groups, and e) CD138+ spectra for tracking changes over time.

1. Setup - load required data

Required Data Download publicly available myeloma data and save to "data" directory.

1. **MMRF CoMMpass study gene expression estimates.** Download transcript-based expression estimates processed by Salmon (version 0.7.2) from the CoMMpass Study (release IA14) after registering at the MMRF web portal (https://research.themmrf.org/). Download file: MMRF_CoMMpass_IA14a_E74GTF_Salmon_V7.2_Filtered_Transcript_Counts.txt.gz.
2. **Gene Transfer Format (GTF) file.** Use the same GTF to process as the CoMMpass Study used to generate expression estimates. Download here: ftp://ftp.ensembl.org/pub/release-74/gtf/homo_sapiens/Homo_sapiens.GRCh37.74.gtf.gz.
3. **MMRF CoMMpass study clinical annotation files.** Download CoMMpass IA14 clinical flat files from the MMRF web portal after registering (https://research.themmrf.org/). Required files:

- MMRF_CoMMpass_IA14_PER_PATIENT_VISIT.csv
- MMRF_CoMMpass_IA14_PER_PATIENT.csv
- MMRF_CoMMpass_IA14_STAND_ALONE_SURVIVAL.csv

4. **MMRF CoMMpass study sequence quality control file.** Download CoMMpass sequence quality control file from the MMRF web portal after registering (https://research.themmrf.org/): MMRF_CoMMpass_IA14_Seq_QC_Summary.csv
5. **List of problematic genes in RNAseq pipelines.** Removed protein coding genes that are discordantly quantified at the 4 fold or higher level (Arora et al, Sci Rep, 2020, https://doi.org/10.1038/s41598-020-59516-z). List of genes is included with this Rmd report on github. The list is from Arora et al Supplemental Table 2b.

1.0. Install packages

```r
# Install and load required R packages
#install.packages("BiocManager")
library(rtracklayer) # BiocManager::install("rtracklayer")
library(sva) # BiocManager::install("sva")
library(ggplot2)
library(dplyr)
library(data.table)
```

Define data directory

```r
data_dir = "/path/to/data" # exclude ending "/"
```

Define function

```r
elbow_finder<-function(data) {
  elbow.dt<-data[order(-value)][,idx:=.I-1]
  elbow.dt[,selected:=as.factor('Not used')]

  slope<-(min(elbow.dt$value)-max(elbow.dt$value))/(nrow(elbow.dt)-1)
  perpslope<-(-1/slope)
  intercept<-max(elbow.dt$value)
  elbow.dt[,perpcept:=value - perpslope*idx]

  elbow.dt[,y:=(perpcept*slope - intercept*perpslope)/(slope-perpslope)]
  elbow.dt[,x:=(intercept-perpcept)/(perpslope - slope)]
  elbow.dt[,dist:=sqrt((value-y)^2 + (idx-x)^2)]

  maxidx<-which.max(elbow.dt$dist)
  elbow.dt[idx<maxidx]$selected<-as.factor('Selected')
  elbow.dt[,propvar:=cumsum(value)/sum(value)]
  elbow.dt
}
```

1.1. LOAD GTF WITH RTRACKLAYER PACKAGE & COMPUTE GENE LENGTH AND NUMBER OF TRANSCRIPTS

```r
# Read in GTF
gtf <- readGFF(filepath=paste0(data_dir,"/Homo_sapiens.GRCh37.74.gtf.gz"))
gtf$length <- gtf$end - gtf$start + 1 # Compute length of each GTF feature
tmp <- subset(gtf,type=='exon',) # Select exon features only
# Find the total gene length by summing the exons
sum_exons<-aggregate(length ~ seqid + gene_name + gene_id +
                       gene_biotype + transcript_id, data=tmp, FUN=sum)
# Count the number of transcripts for each gene name
n_name <- sum_exons %>% count(gene_name)
colnames(n_name)[2] <- "name_n_transcripts" # Rename column
# Create table from gene length and transcript number
gtf.dt <- data.table(inner_join(n_name,sum_exons,by='gene_name'))
rm(gtf,tmp,sum_exons,n_name) # Clean-up variable space
```

1.2. LOAD COMMPASS GENE EXPRESSION ESTIMATES & MERGE WITH GTF ANNOTATIONS

```r
# Read in counts and set to data table format
counts <- data.table(read.csv(
  file = paste0(data_dir,
  "/MMRF_CoMMpass_IA14a_E74GTF_Salmon_V7.2_Filtered_Transcript_Counts.txt.gz"),
  header = T,sep = "\t",stringsAsFactors = F)
  )
colnames(counts)[1]<-"transcript_id" # Rename column
# Merge counts with GTF data
counts.gtf <- data.table(inner_join(gtf.dt,counts,by="transcript_id"))
rm(counts,gtf.dt) # Clean variable space
seq_ids = counts.gtf %>%
  dplyr::select(starts_with("MMRF") & ends_with("BM")) %>%
  colnames() # Get list of bone marrow sample ids
seq_nms = unique(gsub("_([1-9])_(BM|PB)","",seq_ids)) # Get list of individuals

print(paste0(length(unique(counts.gtf$gene_name)),
             " genes and ",length(unique(counts.gtf$transcript_id)),
             " transcripts in CoMMpass expression dataset"))
```

```
## [1] "54324 genes and 194622 transcripts in CoMMpass expression dataset"
```

1.3. LOAD COMMPASS CLINICAL & SEQ QC ANNOTATIONS

A. Per-visit

```r
vis = data.table(
  read.csv(file = paste0(data_dir,
                         "/MMRF_CoMMpass_IA14_PER_PATIENT_VISIT.csv")
                          ,stringsAsFactors = F)
  )[PUBLIC_ID%in%seq_nms, # Select samples with RNAseq
                   c("PUBLIC_ID", # Individual ID
                     "SPECTRUM_SEQ", # Spectrum Sample ID and Sequence
                     "VJ_INTERVAL", # Record Interval
                     "AT_TREATMENTRESP", # Treatment Response
                     "D_TRI_CF_WASCYTOGENICS", # Was Cytogenics / FISH performed?
                     "D_TRI_CF_ABNORMALITYPR9", # t(14;20) abnormality present
                     "D_TRI_CF_T1420ABNORMAL", # t(14;20) (Abnormal Cells %)
                     "D_TRI_CF_ABNORMALITYPR12", # 1p deletion abnormality present
                     "D_TRI_CF_1PDELETIONABN", # 1p deletion (Abnormal Cells %)
                     "D_TRI_CF_ABNORMALITYPR13", # 1q amplification abnormality present
                     "D_TRI_CF_1PAMPLIFICATI2", # 1q amplification (Abnormal cells%)
                     "D_TRI_CF_ABNORMALITYPR", # del 13 abnormality present
                     "D_TRI_CF_DEL13ABNORMAL", # del 13 (Abnormal Cells %)
                     "D_TRI_CF_ABNORMALITYPR10", # del 13q abnormality present
                     "D_TRI_CF_13QABNORMALCE", # -13q (Abnormal Cells %)
                     "D_TRI_CF_ABNORMALITYPR2", # del 17 abnormality present
                     "D_TRI_CF_DEL17ABNORMAL", # del 17 (Abnormal Cells %)
                     "D_TRI_CF_ABNORMALITYPR11", # del 17p abnormality present
                     "D_TRI_CF_17PABNORMALCE", # -17p (Abnormal Cells %)
                     "D_TRI_CF_ABNORMALITYPR3", # t(4;14) abnormality present
                     "D_TRI_CF_T414ABNORMALC", # t(4;14) (Abnormal Cells %)
                     "D_TRI_CF_ABNORMALITYPR4", # t(6;14) abnormality present
                     "D_TRI_CF_T614ABNORMALC", # t(6;14) (Abnormal Cells %)
```

```r
                      "D_TRI_CF_ABNORMALITYPR5", # t(8;14) abnormality present
                      "D_TRI_CF_T814ABNORMALC", # t(8;14) (Abnormal Cells %)
                      "D_TRI_CF_ABNORMALITYPR6", # t(11;14) abnormality present
                      "D_TRI_CF_T1114ABNORMAL", # t(11:14) (Abnormal Cells %)
                      "D_TRI_CF_ABNORMALITYPR8", # t(14;16) abnormality present
                      "D_TRI_CF_T1416ABNORMAL" # t(14;16) (Abnormal Cells %)
                      )]
# Create sequence ids that match the counts data
vis$SEQ_ID = paste(vis$SPECTRUM_SEQ,"_BM",sep = "")
# Subset to samples with sequencing data
vis = vis[SEQ_ID%in%seq_ids]
```

B. Sequencing quality control

```r
bat = data.table(
  read.csv(file = paste0(data_dir,"/MMRF_CoMMpass_IA14_Seq_QC_Summary.csv"))
  )[MMRF_Release_Status=="RNA-Yes",c(1:5)] # Read in QC data and select RNAseq samples

# Rename columns
colnames(bat) = c("PUBLIC_ID","SPECTRUM_SEQ","COLLECTION_REASON","SEQ_ID","batch")

# Make sequence ids uniform
bat$SEQ_ID <- gsub("_CD138pos_T([1-9]{1,2})_TSMRU_([L|K0-9]{6})","",bat$SEQ_ID)
```

C. Per-patient

```r
pat = data.table(
  read.csv(file = paste0(data_dir,"/MMRF_CoMMpass_IA14_PER_PATIENT.csv")
           ,stringsAsFactors = F) # Read in per-patient clinical data
  )[PUBLIC_ID%in%seq_nms, # Select samples with RNAseq
                    c("PUBLIC_ID", # Select variables of interest
                      "D_PT_age",
                      "D_PT_gender",
                      "D_PT_race",
                      "D_PT_ethnic",
                      "D_PT_iss")]
```

D. Survival

```r
sur = data.table(
  read.csv(file = paste0(data_dir,
                         "/MMRF_CoMMpass_IA14_STAND_ALONE_SURVIVAL.csv"))
  )[public_id%in%seq_nms, # Select samples with RNAseq
                    c("public_id", # Select variables of interest
                      "ttcos", #Time to OS event (censored)
                      "censos", #Censor flag: overall survival
                      "ttcpfs", #Time to PFS event (censored)
                      "censpfs", #Censor flag: progression-free survival
                      "ttctf1", #Line 1 time to treatment failure (censored)
                      "censtf1" #Line 1 censor flag: time to treatment failure
                      )]
colnames(sur)[1] = "PUBLIC_ID"
```

Merge clinical annotations

```
vis_bat = merge(bat,vis) # MERGE PER-VISIT & SEQ QC
pat_sur = merge(pat,sur) # MERGE PER-PATIENT & SURIVAL
# Note: per-patient annotations are duplicated where multiple samples were sequenced
cin = merge(vis_bat,pat_sur)
rm(bat,pat,pat_sur,sur,vis,vis_bat) # cleanup variables
```

1.4. SUBSET EXPRESSION DATA TO BASELINE SAMPLES

```
baseline = cin[VJ_INTERVAL=="Baseline",]$SEQ_ID # List of baseline samples
base = counts.gtf %>% dplyr::select("gene_name","name_n_transcripts","seqid","gene_id",
                                    "gene_biotype","transcript_id","length",
                                    "gene_name",all_of(baseline))
print(paste0("CoMMpass RNAseq: ",length(cin$SEQ_ID)," samples in ",
             length(unique(cin$PUBLIC_ID))," patients."))
```

```
## [1] "CoMMpass RNAseq: 887 samples in 794 patients."
```

```
print(paste0(length(baseline),
             " individuals with a baseline sample selected to derive MM SPECTRA"))
```

```
## [1] "768 individuals with a baseline sample selected to derive MM SPECTRA"
```

2. Quality control expression estimates

2.1. FILTER TO AUTOSOME & PROTEIN CODING GENES

```
eps = base[seqid %in% c(1:22) & # Filter to autosomal chromosomes
             gene_biotype == 'protein_coding'] %>% # Filter to protein coding genes
  dplyr::select("gene_name",contains("MMRF")) # Select columns: gene_name and all samples
```

2.2. FIND GENES WITH LOW COUNTS IN BASELINE SAMPLES

Find gene level expression from aggregated transcripts

```
# Aggregate transcript counts to gene_name counts
gene = data.table(aggregate(. ~ gene_name, data = eps, FUN = sum))
```

Find genes with >= 95% coverage (>=100 reads)

```
# Find genes with < 5% of samples with < 100 count
keep.genes = cbind(gene[,"gene_name"],
                   rowSums(gene[,-"gene_name"] < 100)/length(baseline))[V2 < 0.05]$gene_name
print(paste0("Keep ",length(keep.genes)," genes where < 5% of samples have < 100 count"))
```

```
## [1] "Keep 8631 genes where < 5% of samples have < 100 count"
```

2.3. REMOVE DQ GENES FROM: https://doi.org/10.1038/s41598-020-59516-z

```
# Compare with discordantly quantified genes
# in GTEx or TCGA from https://doi.org/10.1038/s41598-020-59516-z
dq.genes = read.csv(
  file = paste0(data_dir,
               "/Union_Discordant_genes_TCGA_GTEx_Arora_TS2B.csv")) %>%
  data.table()

print(paste0("Remove ",nrow(dq.genes[Union_DQ_genes%in%keep.genes]),
             " genes of ", nrow(dq.genes[Union_DQ_genes%in%gene$gene_name]),
             " genes on discordantly quanitifed list"))
```

```
## [1] "Remove 1195 genes of 1993 genes on discordantly quanitifed list"
```

```
# Remove genes discordantly quantified
good.gene = gene[gene_name%in%keep.genes &
                   !gene_name%in%dq.genes$Union_DQ_genes]$gene_name
print(paste0(length(good.gene)," high-quality genes selected for analyses"))
```

```
## [1] "7436 high-quality genes selected for analyses"
```

2.4. FIND SAMPLES WITH < 90% COVERAGE ACROSS "GOOD" GENES

```
remove.samples = gene[gene_name %in% good.gene,-"gene_name" # Subset to "good" genes
                      # For each sample, find proportion of genes with <100 reads
                      ][,lapply(.SD,function(x)sum(x<100)/length(good.gene))] %>%
  select_if(. > 0.1) %>% colnames() # List samples with > 10% of genes with < 100 counts
print(paste0(length(remove.samples),
      " sample(s) had <100 reads in > 10% of high-quality genes with and was removed"))
```

```
## [1] "1 sample(s) had <100 reads in > 10% of high-quality genes with and was removed"
```

```
print(paste0("Sample(s) removed: ",remove.samples))
```

```
## [1] "Sample(s) removed: MMRF_1584_1_BM"
```

2.5. SUBSET TO "GOOD" GENES & REMOVE "BAD" SAMPLES

```
qc.counts = base[gene_name %in% good.gene] %>% # Select keep genes
  # Remove extra gene annotation and poor coverage samples
  dplyr::select(-seqid,-gene_id,-gene_biotype,-all_of(remove.samples))
qc.melt <- data.table::melt(qc.counts, # transform format
  id.vars=c("gene_name","name_n_transcripts","transcript_id","length"),
  variable.name="SEQ_ID",
  value.name="count")
rm(qc.counts,eps,gene,keep.genes,remove.samples) # cleanup variables
```

3. Normalize and truncate

3.1. NORMALIZE AND ADJUST BY SIZE FACTOR

```r
# Find total counts per sample per gene
qc.melt[,total_raw_counts:=sum(count),by=c('SEQ_ID','gene_name')]
qc.melt[,kb_length:=length/1000] # Find length of gene in kilo-bases

# Find counts per gene length
final.dt = qc.melt[,list(cpk=sum((count+1/name_n_transcripts)/kb_length)),
                   # For each sample and gene pair
                   by=c('SEQ_ID','gene_name','total_raw_counts')]

# Find size factor = median counts/gene length
final.dt[,size_factor:=median(cpk),by='SEQ_ID']
final.dt[,cpkmed:=cpk/size_factor] # Normalize by size factor
final.dt[,logcpkmed:=log2(cpkmed)] # Log2 transform
```

3.2. TRUNCATE VALUES +/- 5 SD FROM MEAN NORMALIZED GENE COUNT

```r
# Find mean of normalized gene counts per gene
final.dt[,mean:=mean(logcpkmed),by='gene_name']
# Find standard deviation of normalized counts per gene
final.dt[,sd:=sd(logcpkmed),by='gene_name']
final.dt[,adjlogcpkmed:=logcpkmed] # New variable to adjust
# Truncate values > 5 SD
final.dt[(logcpkmed-mean)/sd>=5,adjlogcpkmed:=mean+5*sd]
# Truncate values < 5 SD
final.dt[(logcpkmed-mean)/sd<= -5,adjlogcpkmed:=mean-5*sd]
```

3.3. CONVERT TO SAMPLE X GENE MATRIX FORMAT

```r
norm <- list("melt"=final.dt)
# Sample x gene
norm.dt <- dcast(norm$melt, SEQ_ID ~ gene_name, value.var='adjlogcpkmed')
rm(final.dt,norm) # cleanup variables
```

4. Correct for sequencing batch

4.1. ANNOTATE BATCH & COVARIATES FROM CLINICAL DATA

```r
vr = c("SEQ_ID","batch","D_PT_age","D_PT_gender",
       "ttcos","censos","ttcpfs","censpfs","ttctf1","censtf1")
cin_vr = cin %>% select(all_of(vr))
dt = merge(cin_vr,norm.dt,by="SEQ_ID")
```

4.2. RUN COMBAT TO ADJUST EXPRESSION DATA

```r
# SETUP VARIABLES
# Expression only and gene x sample format
DAT = dt %>% select(-all_of(vr)) %>% t()
colnames(DAT) = dt$SEQ_ID # Annotate samples to DAT
BATCH = as.numeric(dt$batch)
MOD = dt %>% select(all_of(vr[3:10])) %>% data.matrix() # Co-variate model

# RUN COMBAT
cbat = ComBat(dat = DAT, batch = BATCH, mod = MOD)
```

```
## Found42batches

## Adjusting for8covariate(s) or covariate level(s)

## Standardizing Data across genes

## Fitting L/S model and finding priors

## Finding parametric adjustments

## Adjusting the Data
```

```
cbat.dt = data.table(t(cbat)) # Sample x gene data table
cbat.dt$SEQ_ID = colnames(cbat) # Annotate sample ids

rm(vr,cin_vr,dt,DAT,BATCH,MOD,cbat) # Cleanup variables
```

5. Generate SPECTRA (run PCA)

5.1. RUN PCA ON COMBAT DATA TO IDENTIFY SPECTRA

```
# Run PCA on combat data
pca = prcomp(cbat.dt[,-"SEQ_ID"],center=TRUE,scale=FALSE,retx=TRUE)
print(paste0("Myeloma spectra identified from ",nrow(cbat.dt),
             " baseline samples in ",length(good.gene)," genes"))
```

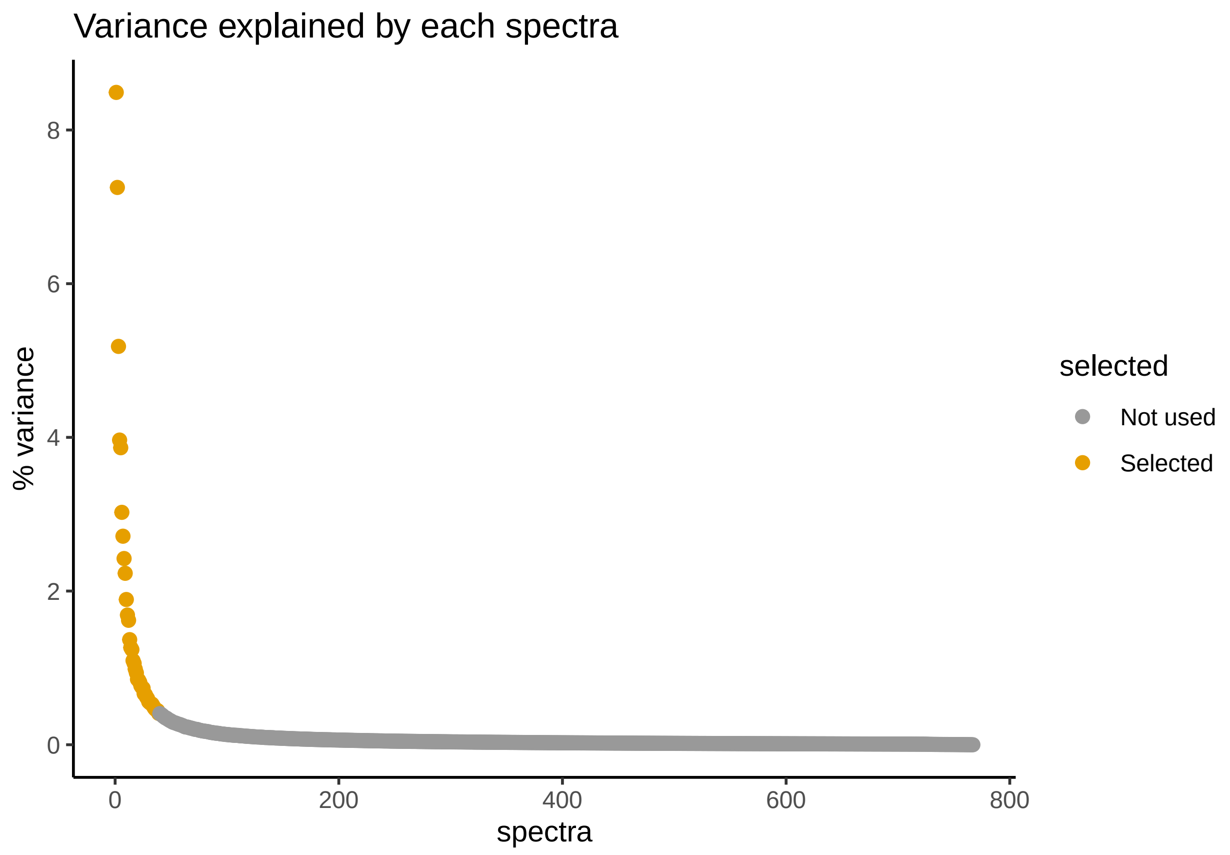
```
## [1] "Myeloma spectra identified from 767 baseline samples in 7436 genes"
```

5.2. FIND K SPECTRA AND PLOT

```
# Compute variance explained by each PC
pcvar <- data.table(pc=colnames(pca$x),value=pca$sdev^2)
# Function in setup, find point of diminishing returns
elbow <- elbow_finder(pcvar)
print(paste0("First ",max(elbow[selected=='Selected']$idx) + 1,
             " spectra selected, describing ",
             round(max(elbow[selected=='Selected']$propvar),
                   digits = 3)*100,"% of the total variance"))
```

```
## [1] "First 39 spectra selected, describing 64.5% of the total variance"
```

```
ggplot(data = elbow,aes(y = value/sum(value)*100,x = idx+1,color = selected)) +
  geom_point(size=2) +
  theme_classic() +
  scale_color_manual(values = c('#999999','#E69F00')) +
  labs(title="Variance explained by each spectra",
       x="spectra", y = "% variance")
```



```r
save(elbow,file = "rdata/elbow.rdata") # save elbow table
```

5.3 TEST FOR ASSOCATION WITH BATCH

```r
# PC value for each baseline sample
pca.score = cbind(cbat.dt$SEQ_ID,pca$x[,elbow[selected=='Selected']$pc]) %>% data.table()
colnames(pca.score)[1] = "SEQ_ID" # Annotate sample names
DAT = merge(cin[,c("SEQ_ID","batch")],pca.score,by="SEQ_ID")
DAT$batch = as.factor(DAT$batch)
pcs = elbow[selected=='Selected']$pc
p = 1
for (i in seq_along(pcs)){
  formula = paste(pcs[i],"~","batch")
  mod = lm(formula,data = DAT)
  p = c(p,pf(summary(mod)$fstatistic[1],summary(mod)$fstatistic[2],
             summary(mod)$fstatistic[3],lower.tail = F))
}

print(paste("F-statistic of PC ~ batch associations, min p =",round(min(p),digits = 3)))
```

```
## [1] "F-statistic of PC ~ batch associations, min p = 0.836"
```

6. Save SPECTRA data

6.1. SAVE SPECTRA VALUES, GENE LOADINGS, AND CENTER ADJUSTMENT VALUES

```r
# PC value for each baseline sample
pca.score = cbind(cbat.dt$SEQ_ID,pca$x[,elbow[selected=='Selected']$pc]) %>% data.table()
colnames(pca.score)[1] = "SEQ_ID" # Annotate sample names
write.csv(pca.score,file = paste0(data_dir,
                                  "/baseline-spectra.csv")
          ,row.names = F)

pca.genes = cbind(
  rownames(pca$rotation), # Gene names
  colMeans(cbat.dt[,-"SEQ_ID"]), # Value to center on for each gene
  pca$rotation[,elbow[selected=='Selected']$pc] %>% data.table() # Save gene loading
)
colnames(pca.genes)[1:2] = c("GENE_NAME","CENTER_MEAN")
write.csv(pca.genes,file = paste0(data_dir,
                                  "/spectra-gene-loadings.csv")
          ,row.names = F)
```

6.2. STANDARDIZE SPCTRA

```r
dt = pca.score %>% select(starts_with("PC")) %>% data.table()
SD = apply(dt,2,sd)
dt_sd = data.matrix(dt) %*% diag(1/SD)
sd.score = cbind(pca.score[,"SEQ_ID"],data.table(dt_sd))
colnames(sd.score) = c("SEQ_ID",
                       paste(colnames(pca.score %>% select(starts_with("PC"))),
                             "_SD",sep = ""))

# check work
unique(round(sd.score$PC1_SD,digits = 8) ==
         round(as.numeric(dt$PC1)/sd(as.numeric(dt$PC1)),digits = 8))
```

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

```r
unique(round(sd.score$PC22_SD,digits = 8) ==
         round(as.numeric(dt$PC22)/sd(as.numeric(dt$PC22)),digits = 8))
```

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

6.3. SAVE AS RDATA FOR ADDITIONAL ANALYSES

```r
exp_cbat = cbat.dt # save normalized and batch corrected expression estimates
write.csv(exp_cbat,file = paste0(data_dir,
                                 "/baseline-expression-norm-combat.csv")
          ,row.names = F)

score_sd = sd.score # standardized spectra scores in baseline samples

# save clinical data in all samples
clinical = cin # clinical data
write.csv(clinical,file = paste0(data_dir,"/clinical.csv"))
```

```r
# merge clinical and standardized spectra scores in baseline samples
clin_spectra_sd = merge(clinical,score_sd,by="SEQ_ID")
write.csv(clin_spectra_sd,file = paste0(data_dir,
                                        "/baseline-clinical-spectra-sd.csv")
          ,row.names = F)

# get list of samples with baseline and follow up RNAseq
followup = clinical %>%
  # Baseline sample passed QC
  subset(PUBLIC_ID%in%clinical[SEQ_ID%in%score_sd$SEQ_ID]$PUBLIC_ID &
           COLLECTION_REASON%like%"Confirm" & # And has follow-up RNAseq
           SEQ_ID%in%colnames(counts.gtf) # And has sequence data available
         )
time_ids = c(clinical[PUBLIC_ID%in%followup$PUBLIC_ID & # Has followup sample
                          COLLECTION_REASON=="Baseline" & # Is baseline sample
                          SEQ_ID%in%score_sd$SEQ_ID # Has clean seq data
                        ]$SEQ_ID, # get seq id
               followup$SEQ_ID
             )
exp_time = counts.gtf %>% select("gene_name","name_n_transcripts",
                                 "seqid","gene_id","gene_biotype",
                                 "transcript_id","length",
                                 all_of(time_ids))
# save expression estimates in longitudinal samples
write.csv(exp_time,file = paste0(data_dir,
                                 "/followup-expression.csv")
          ,row.names = F)
```