CD138+ SPECTRA for tracking changes over time

Code to clean, normalize, and batch correction in the longitudinal CoMMpass RNA samples. Calculation of the CD138+ spectra in the longitudinal samples. Generation of Figure 7 - survival and spectra changes over time.

0. Setup

Define data directory

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

Load packages

```
# Install and load required R packages
library(dplyr)
library(data.table)
library(ggplot2)
library(MASS)
library(survivalAnalysis)
library(sva)
library(gridExtra)
library(RColorBrewer)
```

Load data

```
# expression estimates in longitudinal samples (not processed)
exp_time = read.csv(file = paste0(data_dir,"/followup-expression.csv")) %>%
  data.table()

# baseline spectra gene load
gene.load = read.csv(file = paste0(data_dir,"/spectra-gene-loadings.csv")) %>%
  data.table()

# baseline spectra values (not standardized)
pca.score = read.csv(file = paste0(data_dir,"/baseline-spectra.csv")) %>%
  data.table()

# read in clinical data on all samples
clinical = read.csv(file = paste0(data_dir,"/clinical.csv")) %>%
  data.table()

# baseline CD138+ spectra and clinical risk models
load(file = "rdata/mod.clinical-risk.rdata")

# baseline CD138+ spectra and disease course models
load(file = "rdata/mod.disease-course.rdata")
```

1. Calculate CD138+ spectra in longitudinal samples

Find gene level expression from aggregated transcripts

```r
 # Select genes in baseline CD138+ spectra
eps = exp_time[gene_name %in% gene.load$GENE_NAME] %>%
  dplyr::select("gene_name",contains("MMRF"))

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

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

```r
# For each sample, find proportion of genes with <100 reads
remove.samples =
  gene[,-"gene_name"][,lapply(.SD,function(x)sum(x<100)/length(gene.load$GENE_NAME))] %>%
  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.
             Sample(s) removed: ",remove.samples))
```

```
## [1] "0 sample(s) had <100 reads in > 10% of high-quality genes with and was removed. \n
```

1.3. SUBSET TO SPECTRA GENES & REMOVE "BAD" SAMPLES

```r
qc.counts = exp_time[gene_name %in% gene.load$GENE_NAME] %>% # 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,remove.samples) # cleanup variables
```

1.4. 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')]

# Find length of gene in kilo-bases
qc.melt[,kb_length:=length/1000]

 # 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
```

1.5. 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']

# New variable to adjust
final.dt[,adjlogcpkmed:=logcpkmed]

# 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]
```

1.6. 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
```

1.7. 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 = clinical %>% dplyr::select(all_of(vr))
dt = merge(cin_vr,norm.dt,by="SEQ_ID")
```

1.8. RUN COMBAT TO ADJUST EXPRESSION DATA

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

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

```
## Found38batches

## Note: one batch has only one sample, setting mean.only=TRUE

## 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
```

1.9. DERIVE SPECTRA

```
dim_time = cbind(cbat.dt[,"SEQ_ID"],
                 scale(data.matrix(cbat.dt[,-"SEQ_ID"]),
                       center=gene.load$CENTER_MEAN,scale = FALSE) %*%
                   data.matrix(gene.load[,-c("GENE_NAME","CENTER_MEAN")]))
# Note, spectra values in baseline samples are slightly different,
# batch correction included a different sample set
```

1.10. STANDARDIZE TO BASELINE CD138+ SPECTRA

```
# find standard deviation of original MM spectra
pca.sd = apply(pca.score[,-"SEQ_ID"],2,sd)
tme.sd = cbind(dim_time[,"SEQ_ID"],
               data.matrix(dim_time[,-"SEQ_ID"]) %*%
                 diag(1/pca.sd)) # divide by SD
colnames(tme.sd) = c("SEQ_ID",paste(colnames(dim_time[,-"SEQ_ID"]),"_SD",sep = ""))
```

2. PLOT SPECTRA OVER TIME

PSL changes over times

```
theme_set(theme_classic() + theme(axis.title=element_text(size=9),
                                  axis.text=element_text(size=8),
                                  legend.title=element_text(size=9),
                                  legend.text=element_text(size=8)))
cbPalette <- c("#999999", "#E69F00", "#56B4E9", "#009E73",
               "#F0E442", "#0072B2", "#D55E00", "#CC79A7")
```

Setup data

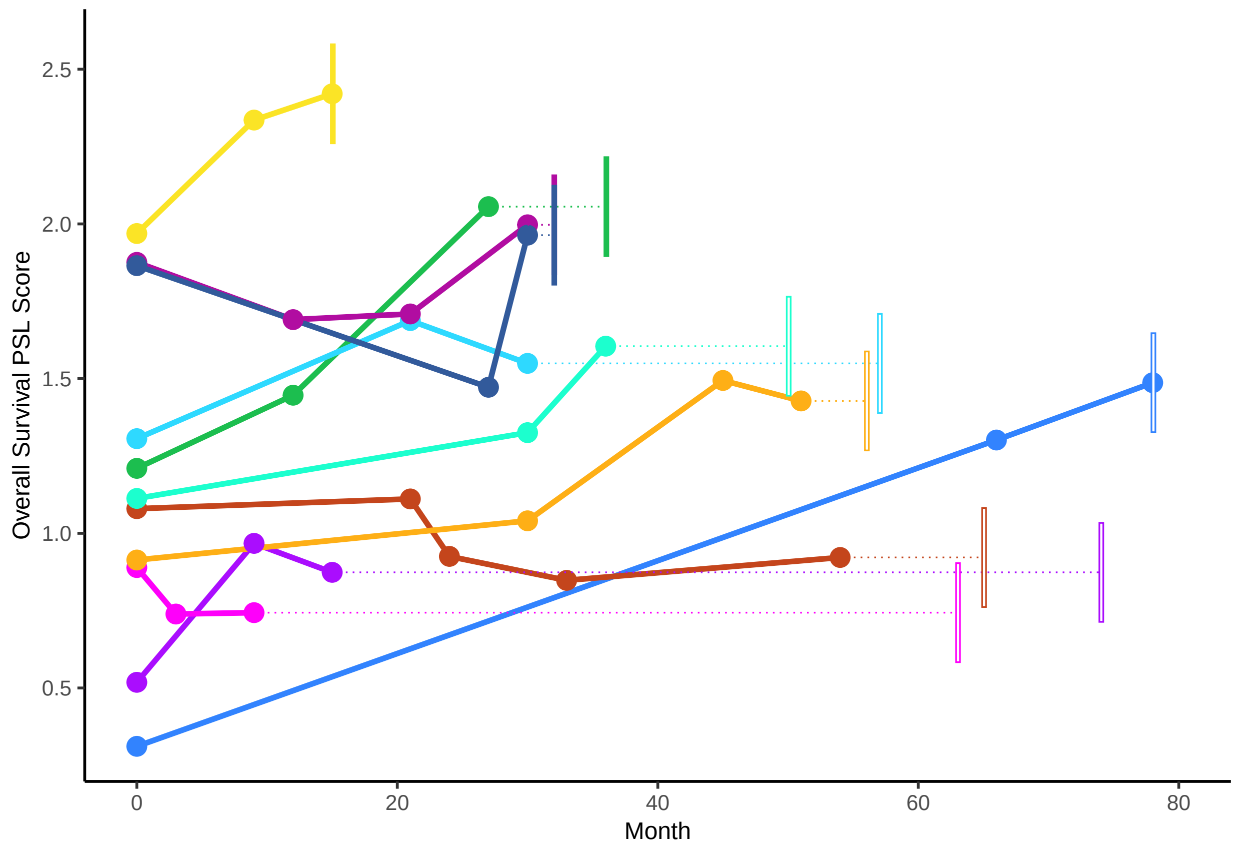
```
# merge with clinical data
dt = merge(
  clinical[,c("SEQ_ID","PUBLIC_ID","COLLECTION_REASON","VJ_INTERVAL",
              "D_TRI_CF_ABNORMALITYPR11","D_TRI_CF_ABNORMALITYPR8",
              "D_TRI_CF_ABNORMALITYPR13","D_TRI_CF_ABNORMALITYPR3",
              "D_TRI_CF_ABNORMALITYPR6","D_TRI_CF_1PAMPLIFICATI2",
              "D_TRI_CF_T1416ABNORMAL","D_PT_iss","ttcos","censos")],
  tme.sd,by="SEQ_ID")
```

```r
# find samples with multiple timepoints
count.dt = as.data.frame(table(dt$PUBLIC_ID))
ids = droplevels(count.dt[count.dt$Freq>2,]$Var1)

# select samples with multiple timepoints
pdt = dt[PUBLIC_ID%in%ids]

# change timepoint to months
#pdt$VJ_INTERVAL
pdt$Month = c(0,9,15,0,66,78,0,12,27,0,21,24,33,54,0,3,9,0,
              30,45,51,0,21,30,0,30,36,0,12,21,30,0,9,15,0,27,30)

#pdt[,c('VJ_INTERVAL',"Month")]
```

Compute poly-spectra liability (PSL) scores at each time point for overall survival

```r
# 2.2.2. Overall Survival
pdt$os_psl = rowSums(data.matrix(pdt %>%
                                   dplyr::select(starts_with("PC"))) %*%
                       diag(os$coxph$coef))

rng = c(min(os$coxph$linear.predictors),max(os$coxph$linear.predictors))

melt.os = melt(pdt[,c("PUBLIC_ID","Month","os_psl","ttcos","censos")],
               id.vars = c("PUBLIC_ID","Month","ttcos","censos"))
melt.os$id = as.integer(gsub("MMRF_","",melt.os$PUBLIC_ID))
melt.os$id_os = melt.os$id * melt.os$censos

melt.os$id = as.factor(melt.os$id)
melt.os$id_os = as.factor(melt.os$id_os)

melt.os$tdy = melt.os$Month/12*365.25
melt.os$smt = round(melt.os$ttcos*0.0328767)
melt.os$censos = as.factor(melt.os$censos)

p1 = ggplot(data=melt.os,aes(x=Month,y=value,color=id)) +
        geom_line(size=1,linetype=1) +
        geom_point(size=3) +
        xlim(c(0,80)) +
        ylab("Overall Survival PSL Score") +
        xlab("Month") + #ggtitle("Overall Survival") +
        scale_color_manual(values = c("#AA0DFE","#3283FE","#1CBE4F",
                                      "#C4451C","#FE00FA","#FEAF16",
                                      "#2ED9FF","#1CFFCE","#B10DA1",
                                      "#FBE426","#325A9B")) +
        theme(legend.position = "right")


# add survival point
melt.os.last = melt.os %>% group_by(PUBLIC_ID) %>% top_n(1, Month)
melt.os.last[melt.os.last$smt==77,"smt"] = 78
melt.os.last = as.data.table(melt.os.last)
```

```
p2 = p1 + geom_segment(data = melt.os.last,size=0.3,
                  aes(color=id, x = Month, xend = smt,
                      y = value, yend = value),linetype = 3) +
  geom_rect(data=melt.os.last, mapping=aes(xmin=smt - 0.1,
                                            xmax=smt + 0.2,
                                            ymin=value - 0.16,
                                            ymax= value + 0.16,
                                            fill = id_os),size=0.3) +
  scale_fill_manual(values = c("white","#1CBE4F","#B10DA1","#FBE426","#325A9B")) +
  theme(legend.position = "none")
p2
```



```
ggsave(p2,filename = "plots/survival-time.png",width=7.5,height=4)
```