brain.anno = allDepMapData$annotation[allDepMapData$annotation$Subtype.Disease == "Glioblastoma", ]

rownames(brain.anno) = brain.anno$DepMap\_ID

brain.anno = brain.anno[, -1]

cell.lines = dput(rownames(brain.anno))

exp.clean = allDepMapData$expression[, -which(colnames(allDepMapData$expression) %!in% cell.lines)]

copy.clean = allDepMapData$copynumber[, -which(colnames(allDepMapData$copynumber) %!in% cell.lines)]

relevant.mutations = subset(allDepMapData$mutation, names(allDepMapData$mutation) %in% cell.lines)

ceres.clean = allDepMapData$kd.ceres[, -which(colnames(allDepMapData$kd.ceres) %!in% cell.lines)]

prob.clean = allDepMapData$kd.prob[, -which(colnames(allDepMapData$kd.prob) %!in% cell.lines)]

sum(is.na(ceres.clean))

sum(is.na(prob.clean))

sum(is.na(copy.clean))

sum(is.na(exp.clean))

sum(is.na(relevant.mutations))

copy.clean = copy.clean[-which(apply(copy.clean, 1, function(x) {sum(is.na(x))}) > 0), ]

sum(is.na(copy.clean))

sum(exp.clean < 0)

sum(prob.clean < 0)

sum(prob.clean > 1)

sum(ceres.clean == 0)

sum(prob.clean == 0)

sum(copy.clean == 0)

sum(exp.clean == 0)

dim(copy.clean) == dim(exp.clean)

gene.data.co = c(rownames(copy.clean))

gene.data.ex = c(rownames(exp.clean))

exp.clean = exp.clean[-which(rownames(exp.clean) %!in% gene.data.co),]

copy.clean = copy.clean[-which(rownames(copy.clean) %!in% gene.data.ex),]

genes.clean = rownames(exp.clean)

ceres.clean = ceres.clean[-which(rownames(ceres.clean) %!in% genes.clean),]

prob.clean = prob.clean[-which(rownames(prob.clean) %!in% genes.clean),]

genes.clean = rownames(ceres.clean)

exp.clean = exp.clean[-which(rownames(exp.clean) %!in% genes.clean),]

copy.clean = copy.clean[-which(rownames(copy.clean) %!in% genes.clean),]

rm(gene.data.co)

rm(gene.data.ex)

brain.anno$CCLE\_Name = factor(brain.anno$CCLE\_Name)

brain.anno$Aliases = factor(brain.anno$Aliases)

brain.anno$Primary.Disease = factor(brain.anno$Primary.Disease)

brain.anno$Subtype.Disease = factor(brain.anno$Subtype.Disease)

brain.anno$Subtype.Gender = factor(brain.anno$Gender)

brain.anno$Subtype.Source = factor(brain.anno$Source)

ceres.clean = ceres.clean[order(rownames(ceres.clean)), order(colnames(ceres.clean))]

prob.clean = prob.clean[order(rownames(prob.clean)), order(colnames(prob.clean))]

exp.clean = exp.clean[order(rownames(exp.clean)), order(colnames(exp.clean))]

copy.clean = copy.clean[order(rownames(copy.clean)), order(colnames(copy.clean))]

relevant.mutations.combi = do.call(rbind, lapply(which(names(allDepMapData$mutation) %in% cell.lines), function(a) allDepMapData$mutation[[a]]))

common.genes = as.matrix(table(c(relevant.mutations.combi$Hugo\_Symbol)))

summary(common.genes)

rownames(common.genes) = common.genes$Var1

common.genes$Var1 = NULL

barplot(common.genes, beside = T, names.arg = rownames(common.genes), las = 2)

common.genes.c = subset(common.genes, common.genes$Freq >11)

common.genes.c = common.genes.c[c(6, 11, 10, 7, 4, 9, 3, 2, 1, 8),]

barplot\_commongenes <- barplot(common.genes.c, beside = T, names.arg = rownames(common.genes.c), ylab = "Frequency", main = "Most common gene mutations", las = 2)

list.cells = subset(mutations.all, mutations.all$Hugo\_Symbol %in% rownames(common.genes.c))

unique(list.cells$DepMap\_ID)

list.mtnd5 = unique(subset(mutations.all, mutations.all$Hugo\_Symbol == "MT-ND5"))

cells.mtnd5 = c(list.mtnd5$DepMap\_ID)

list.non\_mtnd5 = unique(subset(mutations.all, mutations.all$Hugo\_Symbol != "MT-ND5"))

cells.non\_mtnd5 = c(list.non\_mtnd5$DepMap\_ID)

list.muc16 = unique(subset(mutations.all, mutations.all$Hugo\_Symbol == "MUC16"))

cells.muc16 = c(list.muc16$DepMap\_ID)

list.non\_muc16 = unique(subset(mutations.all, mutations.all$Hugo\_Symbol != "MUC16"))

cells.non\_muc16 = c(list.non\_muc16$DepMap\_ID)

list.tp53 = unique(subset(mutations.all, mutations.all$Hugo\_Symbol == "TP53"))

cells.tp53 = c(list.tp53$DepMap\_ID)

list.non\_tp53 = unique(subset(mutations.all, mutations.all$Hugo\_Symbol != "TP53"))

cells.non\_tp53 = c(list.non\_tp53$DepMap\_ID)

list.ttn = unique(subset(mutations.all, mutations.all$Hugo\_Symbol == "TTN"))

cells.ttn = c(list.ttn$DepMap\_ID)

exp.muc16 = exp.clean[,which(colnames(exp.clean) %in% cells.muc16)]

copy.muc16 = copy.clean[,which(colnames(copy.clean) %in% cells.muc16)]

ceres.muc16 = ceres.clean[,which(colnames(ceres.clean) %in% cells.muc16)]

prob.muc16 = prob.clean[,which(colnames(prob.clean) %in% cells.muc16)]

exp.non\_muc16 = exp.clean[,which(colnames(exp.clean) %in% cells.non\_muc16)]

copy.non\_muc16 = copy.clean[,which(colnames(copy.clean) %in% cells.non\_muc16)]

ceres.non\_muc16 = ceres.clean[,which(colnames(ceres.clean) %in% cells.non\_muc16)]

prob.non\_muc16 = prob.clean[,which(colnames(prob.clean) %in% cells.non\_muc16)]

exp.tp53 = exp.clean[,which(colnames(exp.clean) %in% cells.tp53)]

copy.tp53 = copy.clean[,which(colnames(copy.clean) %in% cells.tp53)]

ceres.tp53 = ceres.clean[,which(colnames(ceres.clean) %in% cells.tp53)]

prob.tp53 = prob.clean[,which(colnames(prob.clean) %in% cells.tp53)]

exp.non\_tp53 = exp.clean[,which(colnames(exp.clean) %in% cells.non\_tp53)]

copy.non\_tp53 = copy.clean[,which(colnames(copy.clean) %in% cells.non\_tp53)]

ceres.non\_tp53 = ceres.clean[,which(colnames(ceres.clean) %in% cells.non\_tp53)]

prob.non\_tp53 = prob.clean[,which(colnames(prob.clean) %in% cells.non\_tp53)]

exp.ttn = exp.clean[,which(colnames(exp.clean) %in% cells.ttn)]

copy.ttn = copy.clean[,which(colnames(copy.clean) %in% cells.ttn)]

ceres.ttn = ceres.clean[,which(colnames(ceres.clean) %in% cells.ttn)]

prob.ttn = prob.clean[,which(colnames(prob.clean) %in% cells.ttn)]

exp.non\_ttn = exp.clean[,which(colnames(exp.clean) %in% cells.non\_ttn)]

copy.non\_ttn = copy.clean[,which(colnames(copy.clean) %in% cells.non\_ttn)]

ceres.non\_ttn = ceres.clean[,which(colnames(ceres.clean) %in% cells.non\_ttn)]

prob.non\_ttn = prob.clean[,which(colnames(prob.clean) %in% cells.non\_ttn)]

non\_tp53.exp.mean = as.matrix(c(rowMeans(exp.non\_tp53)))

non\_tp53.copy.mean = as.matrix(c(rowMeans(copy.non\_tp53)))

non\_tp53.ceres.mean = as.matrix(c(rowMeans(ceres.non\_tp53)))

non\_tp53.prob.mean = as.matrix(c(rowMeans(prob.non\_tp53)))

ttn.exp.mean = as.matrix(c(rowMeans(exp.ttn)))

ttn.copy.mean = as.matrix(c(rowMeans(copy.ttn)))

ttn.ceres.mean = as.matrix(c(rowMeans(ceres.ttn)))

ttn.prob.mean = as.matrix(c(rowMeans(prob.ttn)))

non\_ttn.exp.mean = as.matrix(c(rowMeans(exp.non\_ttn)))

non\_ttn.copy.mean = as.matrix(c(rowMeans(copy.non\_ttn)))

non\_ttn.ceres.mean = as.matrix(c(rowMeans(ceres.non\_ttn)))

non\_ttn.prob.mean = as.matrix(c(rowMeans(prob.non\_ttn)))

boxplot\_expression <- boxplot(exp.clean, ylab ="Expression level", main = "Distribution of expression", par(las =2))

boxplot\_CN <- boxplot(copy.clean, ylab = "Copy number", main = "Distribution of copy number", par(las=2))

boxplot\_mtnd5\_exp <- boxplot(mtnd5.exp.mean, ylab = "Expression level", main = "Mean expression of all genes containing MT-ND5 as DM")

boxplot\_mtnd5\_CN <- boxplot(mtnd5.copy.mean, ylab = "Copy number", main = "Mean copy number of all genes containing MT-ND5 as DM")

setHook("grid.newpage", function() pushViewport(viewport(x=1,y=1,width=0.9, height=0.9, name="vp", just=c("right","top"))), action="prepend")

pheatmap(exp.clean, show\_rownames = F)

setHook("grid.newpage", NULL, "replace")

grid.text("celllines", y=-0.07, gp=gpar(fontsize=16))

grid.text("genes", x=-0.07, rot=90, gp=gpar(fontsize=16))

setHook("grid.newpage", function() pushViewport(viewport(x=1,y=1,width=0.9, height=0.9, name="vp", just=c("right","top"))), action="prepend")

pheatmap(copy.clean, show\_rownames = F)

setHook("grid.newpage", NULL, "replace")

grid.text("celllines", y=-0.07, gp=gpar(fontsize=16))

grid.text("genes", x=-0.07, rot=90, gp=gpar(fontsize=16))

setHook("grid.newpage", function() pushViewport(viewport(x=1,y=1,width=0.9, height=0.9, name="vp", just=c("right","top"))), action="prepend")

pheatmap(ceres.clean, show\_rownames = F)

setHook("grid.newpage", NULL, "replace")

grid.text("celllines", y=-0.07, gp=gpar(fontsize=16))

grid.text("genes", x=-0.07, rot=90, gp=gpar(fontsize=16))

setHook("grid.newpage", function() pushViewport(viewport(x=1,y=1,width=0.9, height=0.9, name="vp", just=c("right","top"))), action="prepend")

pheatmap(prob.clean, show\_rownames = F)

setHook("grid.newpage", NULL, "replace")

grid.text("celllines", y=-0.07, gp=gpar(fontsize=16))

grid.text("genes", x=-0.07, rot=90, gp=gpar(fontsize=16))

sum(prob.clean == 0)

prob.clean.w0 <- prob.clean[!(apply(prob.clean, 1, function(y) any(y == 0))),]

sum(prob.clean.w0 == 0)

dim(copy.clean.w0) == dim(exp.clean.w0)

gene.data.ex = c(rownames(exp.clean.w0))

copy.clean.w0 = copy.clean.w0[-which(rownames(copy.clean.w0) %!in% gene.data.ex),]

genes.clean.w0 = rownames(exp.clean.w0)

ceres.clean.w0 = ceres.clean.w0[-which(rownames(ceres.clean.w0) %!in% genes.clean.w0),]

prob.clean.w0 = prob.clean.w0[-which(rownames(prob.clean.w0) %!in% genes.clean.w0),]

genes.clean.w0 = rownames(prob.clean.w0)

exp.clean.w0 = exp.clean.w0[-which(rownames(exp.clean.w0) %!in% genes.clean.w0),]

copy.clean.w0 = copy.clean.w0[-which(rownames(copy.clean.w0) %!in% genes.clean.w0),]

ceres.clean.w0 = ceres.clean.w0[-which(rownames(ceres.clean.w0) %!in% genes.clean.w0),]

rm(gene.data.ex)

pca\_exp <- prcomp(t(exp.clean.w0), scale = TRUE)

plot(pca\_exp$x[,1], pca\_exp$x[,2])

pca\_exp\_var <- pca\_exp$sdev^2

pca\_exp\_var\_per <- round(pca\_exp\_var/sum(pca\_exp\_var)\*100, 1)

barplot(pca\_exp\_var\_per, main = "Proportion of variance", xlab = "Principal component", ylab = "Percent variation")

pca\_data\_exp <- data.frame(Sample = rownames(pca\_exp$x), X = pca\_exp$x[,1], Y = pca\_exp$x[,2])

pca\_data\_exp

pca\_exp\_plot <- ggplot(data = pca\_data\_exp, aes(x=X, y=Y, label = Sample)), geom\_point(), xlab(paste("PC1 -", pca\_exp\_var\_per[1], "%", sep = "")), ylab(paste("PC2 -", pca\_exp\_var\_per[2], "%", sep="")), theme\_bw(), ggtitle("PCA expression")

pca\_exp\_plot

loading\_scores\_exp <- pca\_exp$rotation[,1]

gene\_scores\_exp <- abs(loading\_scores\_exp)

gene\_score\_exp\_ranked <- sort(gene\_scores\_exp, decreasing = TRUE)

top\_10\_genes\_exp <- names(gene\_score\_exp\_ranked[1:10])

top\_10\_genes\_exp

pca\_exp$rotation[top\_10\_genes\_exp,1]

top\_10\_genes\_copy

pca\_copy$rotation[top\_10\_genes\_copy,1]

pca\_ceres <- prcomp(t(ceres.clean.w0), scale = TRUE)

plot(pca\_ceres$x[,1], pca\_ceres$x[,2])

pca\_ceres\_var <- pca\_ceres$sdev^2

pca\_ceres\_var\_per <- round(pca\_ceres\_var/sum(pca\_ceres\_var)\*100, 1)

barplot(pca\_ceres\_var\_per, main = "Proportion of variance", xlab = "Principal component", ylab = "Percent variation")

pca\_data\_ceres <- data.frame(Sample = rownames(pca\_ceres$x), X = pca\_ceres$x[,1], Y = pca\_ceres$x[,2])

pca\_data\_ceres

pca\_ceres\_plot <- ggplot(data = pca\_data\_ceres, aes(x=X, y=Y, label = Sample)) + geom\_point() + xlab(paste("PC1 -", pca\_ceres\_var\_per[1], "%", sep = "")) + ylab(paste("PC2 -", pca\_ceres\_var\_per[2], "%", sep="")) + theme\_bw() + ggtitle("PCA CERES score")

pca\_ceres\_plot

loading\_scores\_ceres <- pca\_ceres$rotation[,1]

gene\_scores\_ceres <- abs(loading\_scores\_ceres)

gene\_score\_ceres\_ranked <- sort(gene\_scores\_ceres, decreasing = TRUE)

top\_10\_genes\_ceres <- names(gene\_score\_ceres\_ranked[1:10])

top\_10\_genes\_ceres

pca\_ceres$rotation[top\_10\_genes\_ceres,1]

pca\_prob <- prcomp(t(prob.clean.w0), scale = TRUE)

plot(pca\_prob$x[,1], pca\_prob$x[,2])

pca\_prob\_var <- pca\_prob$sdev^2

pca\_prob\_var\_per <- round(pca\_prob\_var/sum(pca\_prob\_var)\*100, 1)

barplot(pca\_prob\_var\_per, main = "Proportion of variance", xlab = "Principal component", ylab = "Percent variation")

pca\_data\_prob <- data.frame(Sample = rownames(pca\_prob$x), X = pca\_prob$x[,1], Y = pca\_prob$x[,2])

pca\_data\_prob

pca\_prob\_plot <- ggplot(data = pca\_data\_prob, aes(x=X, y=Y, label = Sample)) + geom\_point() + xlab(paste("PC1 -", pca\_prob\_var\_per[1], "%", sep = "")) + ylab(paste("PC2 -", pca\_prob\_var\_per[2], "%", sep="")) + theme\_bw() + ggtitle("PCA probability score")

pca\_prob\_plot

loading\_scores\_prob <- pca\_prob$rotation[,1]

gene\_scores\_prob <- abs(loading\_scores\_prob)

gene\_score\_prob\_ranked <- sort(gene\_scores\_prob, decreasing = TRUE)

top\_10\_genes\_prob <- names(gene\_score\_prob\_ranked[1:10])

top\_10\_genes\_prob

pca\_prob$rotation[top\_10\_genes\_prob,1]