Exploratory data analysis of genomic data from breast cancer studies

Urminder Singh

December 8, 2018

Introduction

Our data consists of gene expression data from various samples collected to study breast cancer (Breast Invasive Carcinoma). These samples were collected from a number of different individuals. Then, the samples' RNA were extracted sequenced to get RNA-seq reads from the sample. Using the RNA-seq, the gene expression was estimated. However, we used the data from (Wang et. al. 2018) where the expression data was mapped and normalized again to remove batch effects, which made the dataset comparable across the samples. Apart from the gene expression data for each sample, we also have associated metadata for each sample. This metadata provides information about the individual, tissue, disease etc.associated with each sample.

Our dataset has expression estimates of around 20,000 genes across 1,092 samples. Out of these 1,092 samples, 982 had tumor and 110 were normal tissues without tumor. We also added the gene metadata to our dataset which describes the gene features and functions. This also includes mutation information of the genes. This information can provide crucial information about how different mutations in the genome can lead to cancer.

Overall, our dataset is GxS matrix where each row corresponds to a gene and each column corresponds to a RNA-seq sample. Additionally, we have metadata for all the rows and all the columns.

Data

We used data from TCGA. TCGA is a comprehensive repository of human cancer molecular and clinical data, TCGA database has collected clinical and molecular phenotypes of thousands of tumor patients across different tumor types. The TCGA dataset, contains:

- Clinical information about participants
- Metadata about the samples (e.g. the weight of a sample portion, etc.)
- Histopathology slide images from sample portions
- Molecular information derived from the samples (e.g. mRNA/miRNA expression, protein expression, copy number, etc.)

First, we collected data clinical data from TCGA for breast cancer (BRCA) studies. This data contains Then we downloaded the mutation information for the BRCA studies. This data Finally, I collected the gene expression data from (Wang et. al. 2018). This data contains the estimated expression values of genes in the BRCA study. This data was normalized and batch corrected so it is comparable across different TCGA samples.

Data downloading and processing

This section describes how the data was downloaded and pre-processed.

Downloading clinical data from TCGA

We used the TCGABiolinks package to download TCGA data. The function *GDCquery_clinic* is used to download the data. We wrote additional functions to clean the data and arrange it into a dataframe.

```
colsToKeep <- c("clinical.submitter_id", "clinical.classification_of_tumor",
    "clinical.primary_diagnosis", "clinical.tumor_stage", "clinical.age_at_diagnosis",
    "clinical.vital_status", "clinical.days_to_death", "clinical.tissue_or_organ_of_origin",
    "clinical.days_to_birth", "clinical.site_of_resection_or_biopsy",
    "clinical.days_to_last_follow_up", "clinical.cigarettes_per_day",
    "clinical.weight", "clinical.alcohol_history", "clinical.bmi",
    "clinical.years_smoked", "clinical.height", "clinical.gender",
    "clinical.year_of_birth", "clinical.race", "clinical.ethnicity",
    "clinical.year_of_death", "clinical.bcr_patient_barcode",
    "clinical.disease", "submitter_id", "sample_type", "tissue_type",
    "portions.submitter_id", "portions.analytes.analyte_type",
    "portions.analytes.submitter_id", "portions.analytes.analyte_type_id",
    "portions.analytes.aliquots.analyte_type", "portions.analytes.aliquots.submitter_id")
# Function takes a df and expands it by unlisting elements at
# a column
expand <- function(df, colName) {
    res <- data.frame()
    # for each row
    for (i in 1:dim(df)[1]) {
        thisRow <- df[i, !(colnames(df) %in% c(colName))]
        tempdf <- as.data.frame(df[i, c(colName)])
        # if list is empty skip that row
        if (dim(tempdf)[1] < 1) {
            next
        }
        # change colnames so they are unique
        colnames(tempdf) <- paste(paste(colName, ".", sep = ""),
            colnames(tempdf), sep = "")
        # print(paste(i,colnames(tempdf)))
        newRow <- cbind(thisRow, tempdf, row.names = NULL)
        res <- bind_rows(res, newRow)

    }
    # print(res)
    return(res)
}

getjoinedBiospcClinc <- function(projName) {
    print(paste("Downloading", projName))
    clinicalBRCA <- GDCquery_clinic(project = projName, type = "clinical")
    biospecimenBRCA <- GDCquery_clinic(project = projName, type = "Biospecimen")

    # rename all cols from clinical table with suffix clinical
    colnames(clinicalBRCA) <- paste0("clinical.", colnames(clinicalBRCA))

    # expand biospecimen data in the order portions,
    # portions.analytes, portions.analytes.aliquots
    toUnpack <- c("portions", "portions.analytes", "portions.analytes.aliquots")
    for (s in toUnpack) {
```

```
        biospecimenBRCA <- expand(biospecimenBRCA, s)
    }
    # add patient barcode to biospecimen data
    biospecimenBRCA <- biospecimenBRCA %>% mutate(clinical.bcr_patient_barcode = substr(submitter_id,
        1, nchar(as.character(submitter_id)) - 4))
    # join clinical and biospecimen
    finalJoined <- join(clinicalBRCA, biospecimenBRCA, by = "clinical.bcr_patient_barcode")
    return(finalJoined)
}
########################################################################
projName <- "BRCA"
## Download only BRCA metadata
brcaDF <- getjoinedBiospcClinc(paste("TCGA", projName, sep = "-"))
```

```
## [1] "Downloading TCGA-BRCA"
```

```
brcaDF <- brcaDF[, colsToKeep]
# remove cols with all NA values
naCols <- colnames(brcaDF)[sapply(brcaDF, function(x) all(is.na(x)))]
brcaDF <- brcaDF[, !(colnames(brcaDF) %in% naCols)]
```

The clinical data for BRCA studies is stored in a dataframe *brcaDF*.

Downloading mutation data from TCGA

Next, we downloaded mutation data using TCGABiolinks. After getting the mutation data, we joined the clinical table with the mutation table to have clinical information for each mutation in the mutation table.

```
brcaMAF <- GDCquery_Maf(projName, pipelines = "varscan2")
# join mutation information with the clinical data
colnames(brcaDF)[which(colnames(brcaDF) == "portions.analytes.aliquots.submitter_id")] = "Tumor_Sample_
brcaMAF_MD <- join(brcaMAF, brcaDF)
```

We now have a data frame of dimensions 93612, 145. This data frame contains clinical information and coressponding mutation information for each BRCA patient. We will use this data to explore various...

Downloading gene expression data

Next, we downloaded the gene expression data for the BRCA studies.

```
# download exp data from github for BRCA samples. reading
# files can take several minutes
brcaexp_nontumor <- read_delim("https://raw.githubusercontent.com/mskcc/RNAseqDB/master/data/normalized/
    "\t", escape_double = FALSE, trim_ws = TRUE)
brcaexp_tumor <- read_delim("https://raw.githubusercontent.com/mskcc/RNAseqDB/master/data/normalized/br
    "\t", escape_double = FALSE, trim_ws = TRUE)

# formatting. add row names remove extra columns
rn <- brcaexp_nontumor$Hugo_Symbol
brcaexp_nontumor <- brcaexp_nontumor[, 3:dim(brcaexp_nontumor)[2]]
rownames(brcaexp_nontumor) <- rn

rn <- brcaexp_tumor$Hugo_Symbol
```

```
brcaexp_tumor <- brcaexp_tumor[, 3:dim(brcaexp_tumor)[2]]
rownames(brcaexp_tumor) <- rn
```

We now have two expression datasets for tumor and normal or non-tumor tissues from BRCA studies. These datasets contains the gene expression patterns over the tumor samples and normal samples. Each row corresponds to a gene and each column corresponds to a sample. The data dimentions for tumor expression data is 19738, 982 and for normal expression data is 19738, 110

Finally, our data consist of three data frames

1. *brcaMAF_MD* contains all the mutation information
2. *brcaexp_nontumor* contains the gene expression values over normal samples from TCGA
3. *brcaexp_tumor* contains the gene expression values over tumor samples from TCGA

Analysis

First, we looked at the mutation data to find which genes are highly mutated in BRCA and what types of different mutations are present in the data. We used the package *maftools* to plot a summary of the BRCA mutation dataset.

```
# function to take a list of maf files and summarize them
plotmafSummaryList <- function(mafList) {
    l1 <- mafList
    # for each item in list do calculations
    lnames <- names(l1)
    plotList <- list()
    k <- 1
    plist <- list()
    for (i in lnames) {
        print((i))
        print(dim(l1[[i]]))
        if (dim(l1[[i]])[1] < 1) {
            next
        }
        # plot summary and save to pdf
        maf <- read.maf(l1[[i]], isTCGA = T)
        plotmafSummary(maf = maf, rmOutlier = TRUE, addStat = "median",
            dashboard = T, titvRaw = FALSE, showBarcodes = F,
            top = 10)
        mtext(paste("Mutation summary by", i), outer = T, cex = NA,
            line = -1, side = 3)
    }

}
# split maf file into categories by a given variable in the
# clinical metadata file
splitMafby <- function(clinicalData, by, mafData) {
    # get tumor samps fo different values of by
    myList <- list()
    uniqVals <- clinicalData %>% select(by) %>% unique
    for (i in 1:nrow(uniqVals)) {
        # i<-1
        s <- as.character(uniqVals[i, 1])
        print(s)
```

```
        tList <- clinicalData %>% filter(clinicalData[, by] ==
            s) %>% select(Tumor_Sample_Barcode) %>% unique
        thisData <- mafData %>% filter(Tumor_Sample_Barcode %in%
            tList$Tumor_Sample_Barcode)
        myList[[s]] <- thisData
    }

    return(myList)
}

# function to plot top mutated genes with ggplot
plotGeneVarFreq <- function(mafList) {
    # for each item in list
    lnames <- names(mafList)
    plotList <- list()
    k <- 1
    plist <- list()
    getPalette = colorRampPalette(brewer.pal(9, "Paired"))
    # assign colors manually to be consistent with missing data
    variants <- c("Missense_Mutation", "Silent", "3'UTR", "Nonsense_Mutation",
        "5'Flank", "Intron", "Splice_Region", "RNA", "5'UTR",
        "Splice_Site", "In_Frame_Del", "Frame_Shift_Ins", "Frame_Shift_Del",
        "In_Frame_Ins", "3'Flank", "Nonstop_Mutation", "Translation_Start_Site",
        "IGR")
    colourCount = length(unique(variants))
    palette <- getPalette(colourCount)
    names(palette) <- variants

    for (i in lnames) {
        print((i))
        print(dim(mafList[[i]]))
        if (dim(mafList[[i]])[1] < 1) {
            next
        }

        # find top mutated genes
        thisMaf <- mafList[[i]]
        topGenes <- thisMaf %>% filter(Variant_Classification !=
            "Silent") %>% select(Hugo_Symbol) %>% group_by(Hugo_Symbol) %>%
            count %>% arrange(desc(freq)) %>% top_n(n = 10)
        maftest <- thisMaf %>% filter(Hugo_Symbol %in% topGenes$Hugo_Symbol &
            Variant_Classification != "Silent") %>% select(Hugo_Symbol,
            Variant_Classification)


        p <- ggplot(data = maftest, aes(x = Hugo_Symbol, fill = Variant_Classification)) +
            geom_bar(stat = "count") + scale_x_discrete(limits = rev(topGenes$Hugo_Symbol)) +
            coord_flip() + theme(legend.position = "right") +
            theme(axis.text.x = element_text(size = 11, face = "bold"),
                axis.text.y = element_text(size = 11, face = "bold"),
                panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
                panel.border = element_blank(), panel.background = element_blank(),
                axis.title = element_text(size = 12, face = "bold")) +
```

```
            ylab("") + xlab("") + scale_fill_manual(values = palette) +
            ggtitle(i)

        plist[[k]] <- p
        k = k + 1

    }
    # arrange plots on a grid
    do.call("grid_arrange_shared_legend", c(plist))

}

# plot on a grid with single legend function reference
# https://stackoverflow.com/questions/13649473/add-a-common-legend-for-combined-ggplots
grid_arrange_shared_legend <- function(...) {
    plots <- list(...)
    g <- ggplotGrob(plots[[1]] + theme(legend.position = "bottom"))$grobs
    legend <- g[[which(sapply(g, function(x) x$name) == "guide-box")]]
    lheight <- sum(legend$height)
    grid.arrange(do.call(arrangeGrob, lapply(plots, function(x) x +
        theme(legend.position = "none"))), legend, ncol = 1,
        heights = unit.c(unit(1, "npc") - lheight, lheight))
}
```

```
# plot summary for full dataset
tempList <- list()
tempList[["BRCA"]] <- brcaMAF
plotmafSummaryList(tempList)
```

Figure 1 shows the summary plot generated by maftools. From the plot we can see that majority of mutations are of type missense mutation. We also see the top 10 genes with highest number of mutations reported. Out of these genes PIK3CA, TP53, CDH1 and PTEN are known to be associated with cancer.

Next we looked at how the gene mutations differ for different categories e.g gender, race cancer type etc. First, we looked at which genes were mutated for samples from different race.

```
# split the maf data by clinical.race and plot summary for
# each category
l1 <- splitMafby(brcaDF, "clinical.race", brcaMAF)
# plot only white black and asian as others are not reported
l1 <- l1[c("white", "black or african american", "asian")]
plotGeneVarFreq(l1)
```

We found that

Then, we looked at how genes are mutated for top two cancer types i.e. ductal carcinoma and lobular carcinoma.

```
# split the maf data by clinical.race and plot summary for
# each category
filterList <- brcaDF %>% select(clinical.primary_diagnosis) %>%
    count %>% arrange(desc(freq)) %>% top_n(n = 2)
l2 <- splitMafby(brcaDF %>% filter(clinical.primary_diagnosis %in%
    filterList$clinical.primary_diagnosis), "clinical.primary_diagnosis",
    brcaMAF)
plotGeneVarFreq(l2)
```

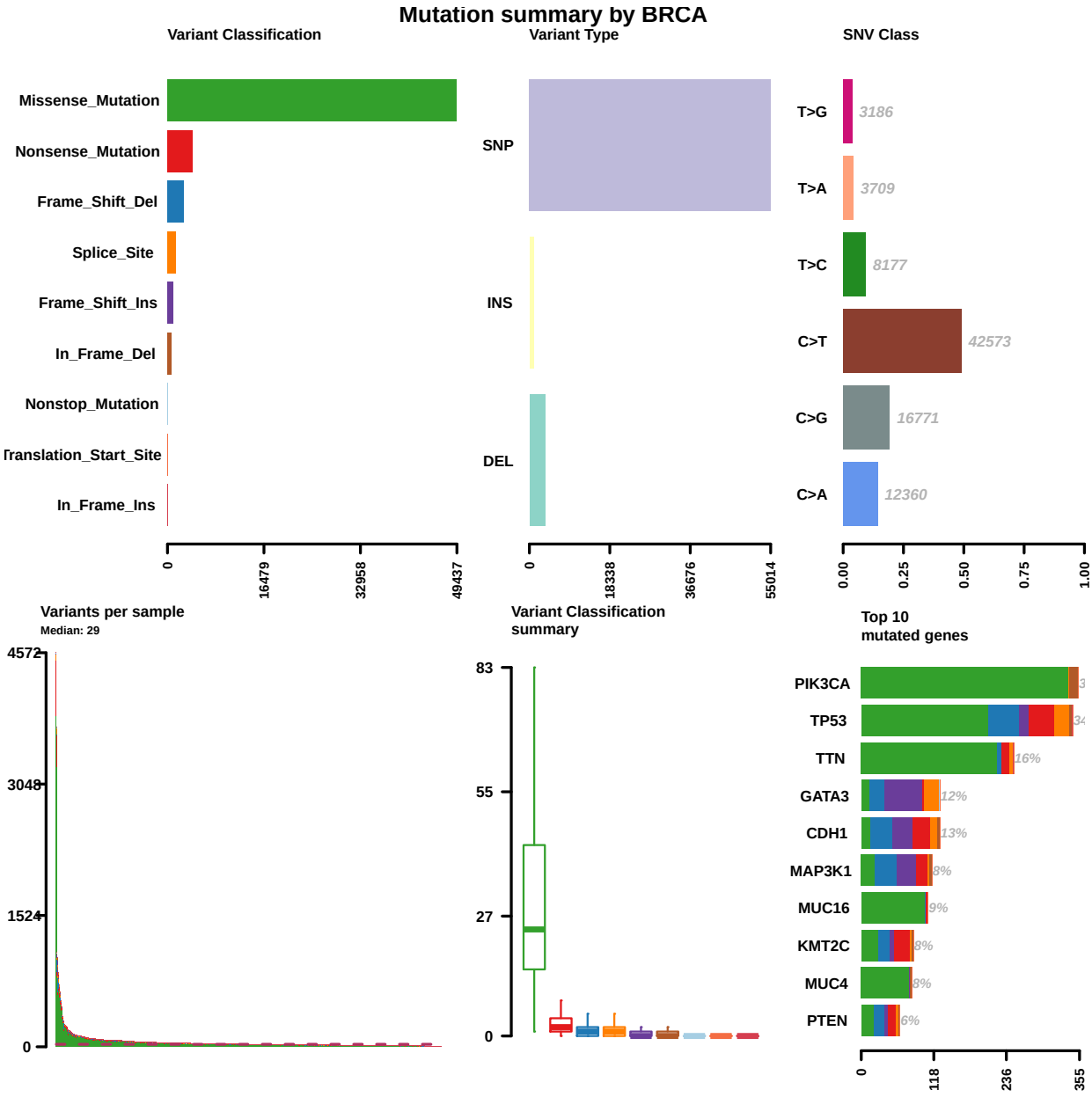

Figure 1: BRCA mutations summary

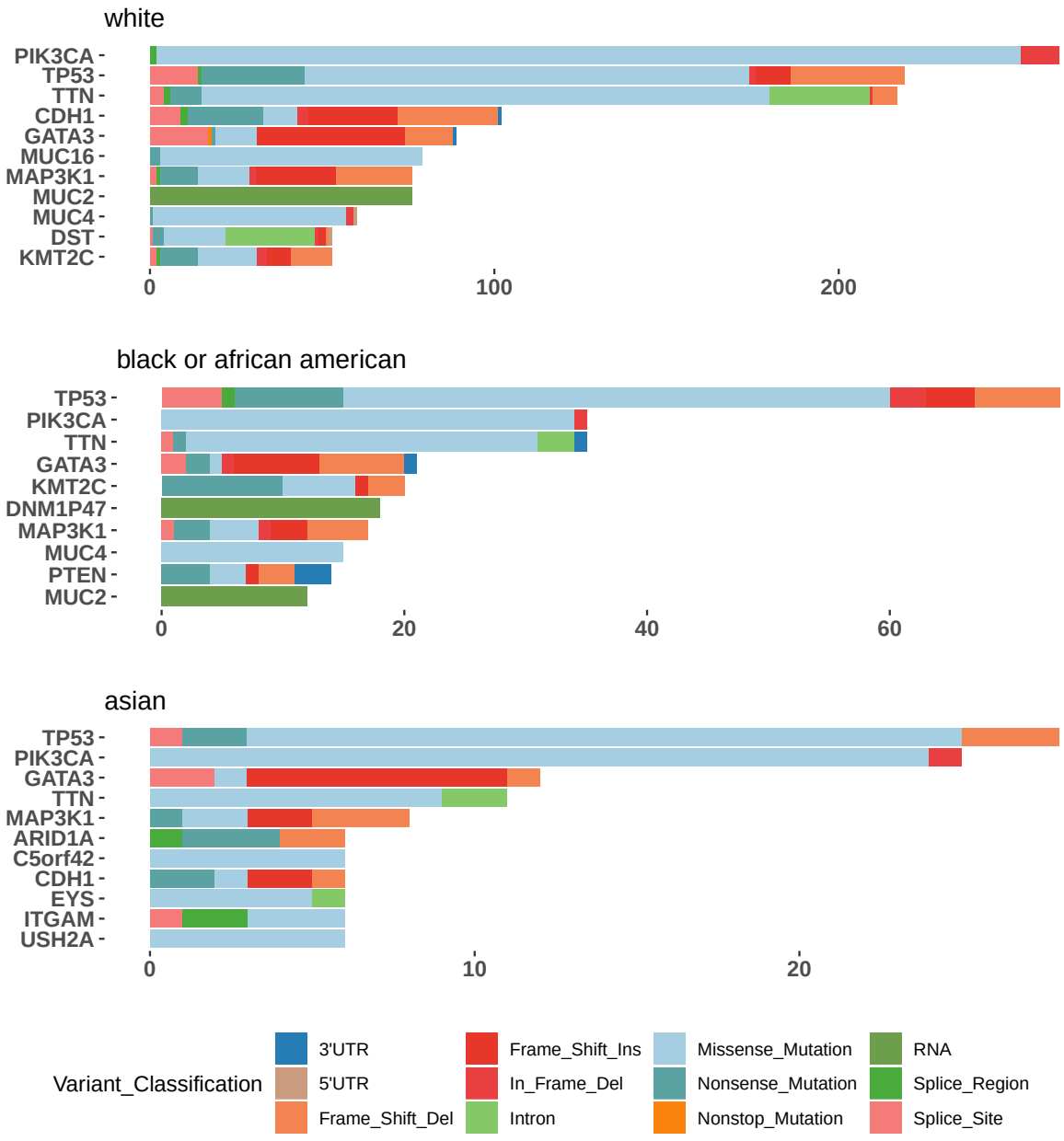

Figure 2: plotting by race

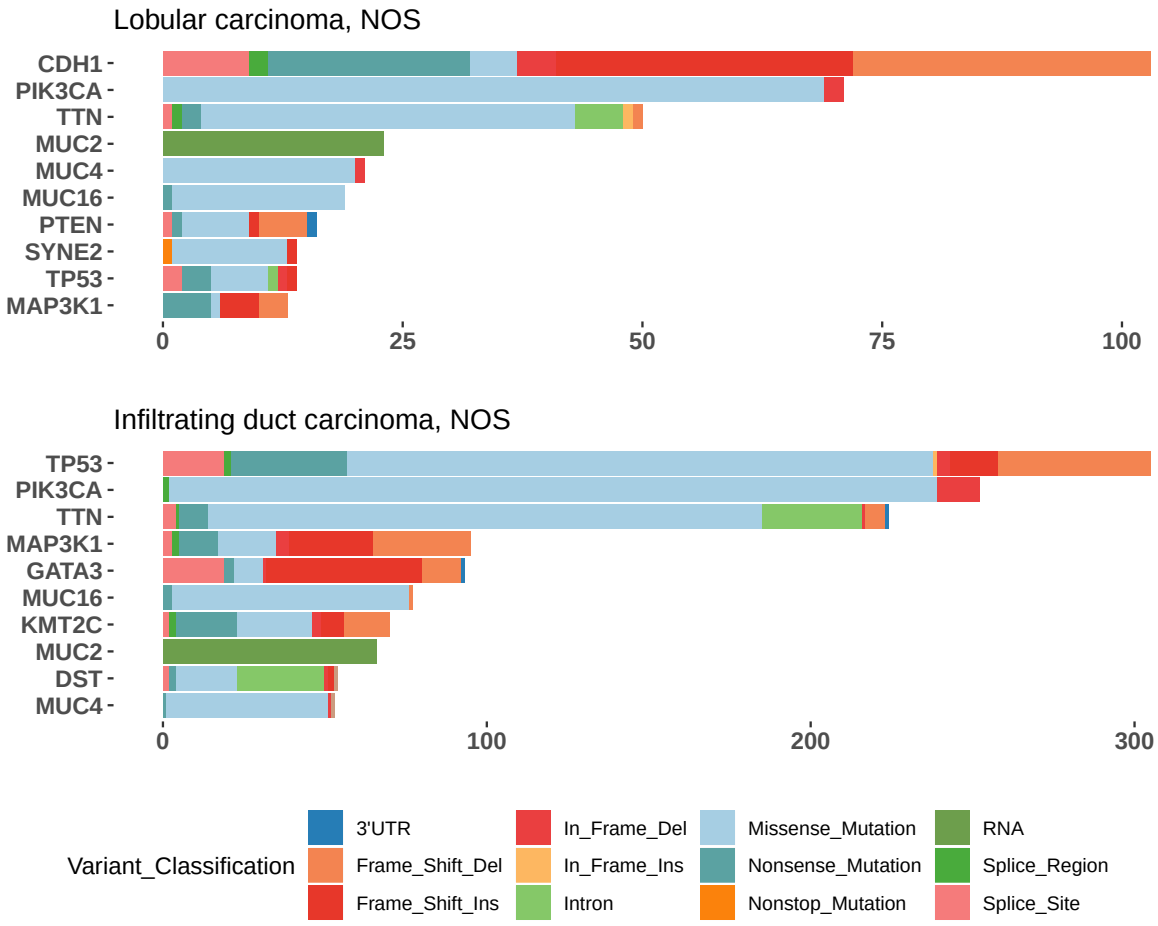

Figure 3: plotting by cancer

From the analysis above we noticed that BRCA1 and BRCA2 are not in the set of top mutated genes. These genes are known to be associated with cancer and a mutation in these genes can increase the risk of breast and ovarian cancer in women. We then compared how many mutations were associated with BRCA1 and BRCA2.

```
brcaMAF_MD_geneGroupBRCA <- brcaMAF_MD %>% select(Hugo_Symbol) %>%
    mutate(geneName = ifelse(Hugo_Symbol == "BRCA1", "BRCA1",
        ifelse(Hugo_Symbol == "BRCA2", "BRCA2", "Others"))) %>%
    select(geneName) %>% group_by(geneName) %>% count %>% mutate(logFreq = log(freq))

p1 <- ggplot(data = brcaMAF_MD_geneGroupBRCA, aes(x = geneName,
    y = logFreq, fill = geneName)) + geom_bar(stat = "identity") +
    theme(legend.position = "none") + theme(axis.text.x = element_text(size = 15,
    face = "bold"), axis.text.y = element_text(size = 10, face = "bold"),
    panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
    panel.border = element_blank(), panel.background = element_blank(),
    axis.title = element_text(size = 12, face = "bold")) + scale_fill_brewer(palette = "Paired")



p2 <- ggplot(data = brcaMAF_MD %>% filter(Hugo_Symbol %in% c("BRCA1",
    "BRCA2")), aes(x = Hugo_Symbol, fill = Variant_Classification)) +
    geom_bar() + theme(axis.text.x = element_text(size = 15,
    face = "bold"), axis.text.y = element_text(size = 10, face = "bold"),
    legend.text = element_text(size = 10, face = "bold"), panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(), panel.border = element_blank(),
    panel.background = element_blank(), axis.title = element_text(size = 12,
        face = "bold")) + scale_fill_brewer(palette = "Dark2")

grid.arrange(p1, p2, nrow = 1)
```

From figure we see that ...

After looking at various mutated genes, we looked at the expression pattern of these genes for tumor and normal samples.

```
# plot box plots of top mutated genes
topGenes <- c("PIK3CA", "TP53", "TTN", "GATA3", "CDH1", "MAP3K1",
    "MUC16", "KMT2C", "MUC4", "PTEN")
nt <- as.data.frame(t(brcaexp_nontumor[topGenes, ]))
colnames(nt) <- topGenes
# tumordata
tdata <- as.data.frame(t(brcaexp_tumor[topGenes, ]))
colnames(tdata) <- topGenes
nt <- nt %>% mutate(source = "Non-Tumor")
tdata <- tdata %>% mutate(source = "Tumor")

topCombined <- bind_rows(nt, tdata)
ggplot(melt(topCombined), aes(x = factor(variable), y = value,
    fill = factor(source))) + geom_boxplot(outlier.colour = NA) +
    scale_y_log10() + theme(legend.position = "top", legend.text = element_text(size = 15,
    face = "bold")) + theme(axis.text.x = element_text(angle = 45,
    size = 15, face = "bold"), axis.text.y = element_text(size = 10,
    face = "bold"), panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
    panel.border = element_blank(), panel.background = element_blank(),
    axis.title = element_text(size = 12, face = "bold")) + scale_fill_discrete(name = "Gene") +
```

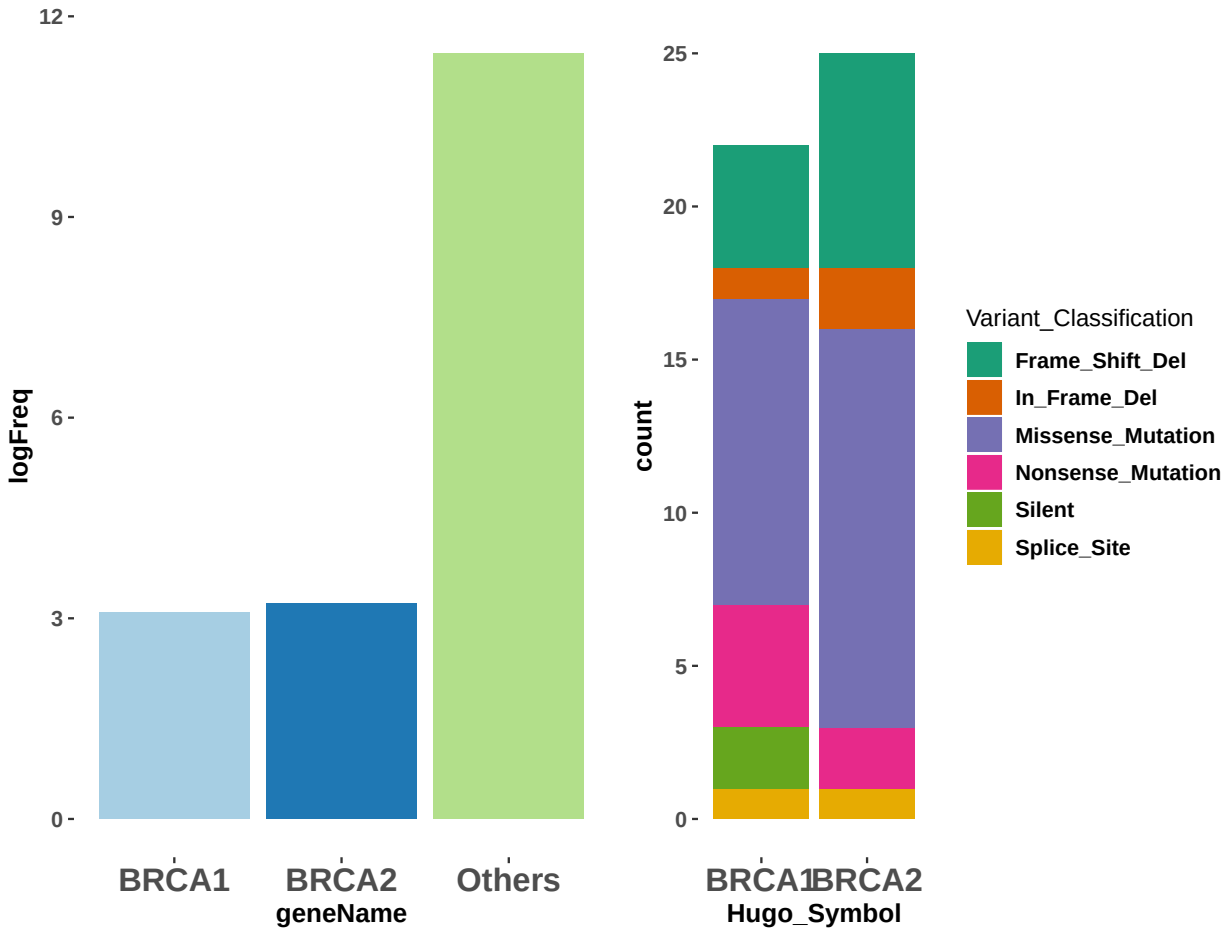

Figure 4: plotting exampleBRCA

```
    ylab("log(expression)") + xlab("") + scale_fill_brewer(palette = "Set1")
```

From fig we found that the expression of

After that we compiled a list of genes are known to be differentially expressed in cancer.

```
# ref for data
# https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5470989/
upGenes <- c("E2F1", "EZH2", "FOXM1", "MYBL2", "PLK1")
dwnGenes <- c("SCARA5", "MYOM1", "NKAPL", "PEG3", "USP2")


ntUP <- as.data.frame(t(brcaexp_nontumor[upGenes, ]))
colnames(ntUP) <- upGenes
ntUP <- ntUP %>% mutate(source = "Non-Tumor")
tdataUP <- as.data.frame(t(brcaexp_tumor[upGenes, ]))
colnames(tdataUP) <- upGenes
tdataUP <- tdataUP %>% mutate(source = "Tumor")

diffExpgenes <- bind_rows(ntUP, tdataUP)
dodge <- position_dodge(width = 0.5)

p1 <- ggplot(melt(diffExpgenes), aes(x = factor(variable), y = value,
    fill = factor(source))) + geom_violin(position = dodge) +
    geom_boxplot(width = 0.1, outlier.colour = NA, position = dodge) +
    scale_y_log10() + theme(legend.position = "top") + theme(axis.text.x = element_text(angle = 45,
    size = 15, face = "bold"), axis.text.y = element_text(size = 10,
    face = "bold"), panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
    panel.border = element_blank(), panel.background = element_blank(),
    axis.title = element_text(size = 12, face = "bold")) + scale_fill_discrete(name = "Gene") +
    ylab("log(expression)") + xlab("") + scale_fill_brewer(palette = "Set1")

# for down regulated in cancer
ntDWN <- as.data.frame(t(brcaexp_nontumor[dwnGenes, ]))
colnames(ntDWN) <- dwnGenes
ntDWN <- ntDWN %>% mutate(source = "Non-Tumor")
tdataDWN <- as.data.frame(t(brcaexp_tumor[dwnGenes, ]))
colnames(tdataDWN) <- dwnGenes
tdataDWN <- tdataDWN %>% mutate(source = "Tumor")
diffExpgenesDWN <- bind_rows(ntDWN, tdataDWN)


p2 <- ggplot(melt(diffExpgenesDWN), aes(x = factor(variable),
    y = value, fill = factor(source))) + geom_violin(position = dodge) +
    geom_boxplot(width = 0.1, outlier.colour = NA, position = dodge) +
    scale_y_log10() + theme(legend.position = "top") + theme(axis.text.x = element_text(angle = 45,
    size = 15, face = "bold"), axis.text.y = element_text(size = 10,
    face = "bold"), panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
    panel.border = element_blank(), panel.background = element_blank(),
    axis.title = element_text(size = 12, face = "bold")) + scale_fill_discrete(name = "Gene") +
    ylab("log(expression)") + xlab("") + scale_fill_brewer(palette = "Set1")

grid.arrange(p1, p2, nrow = 1)
```

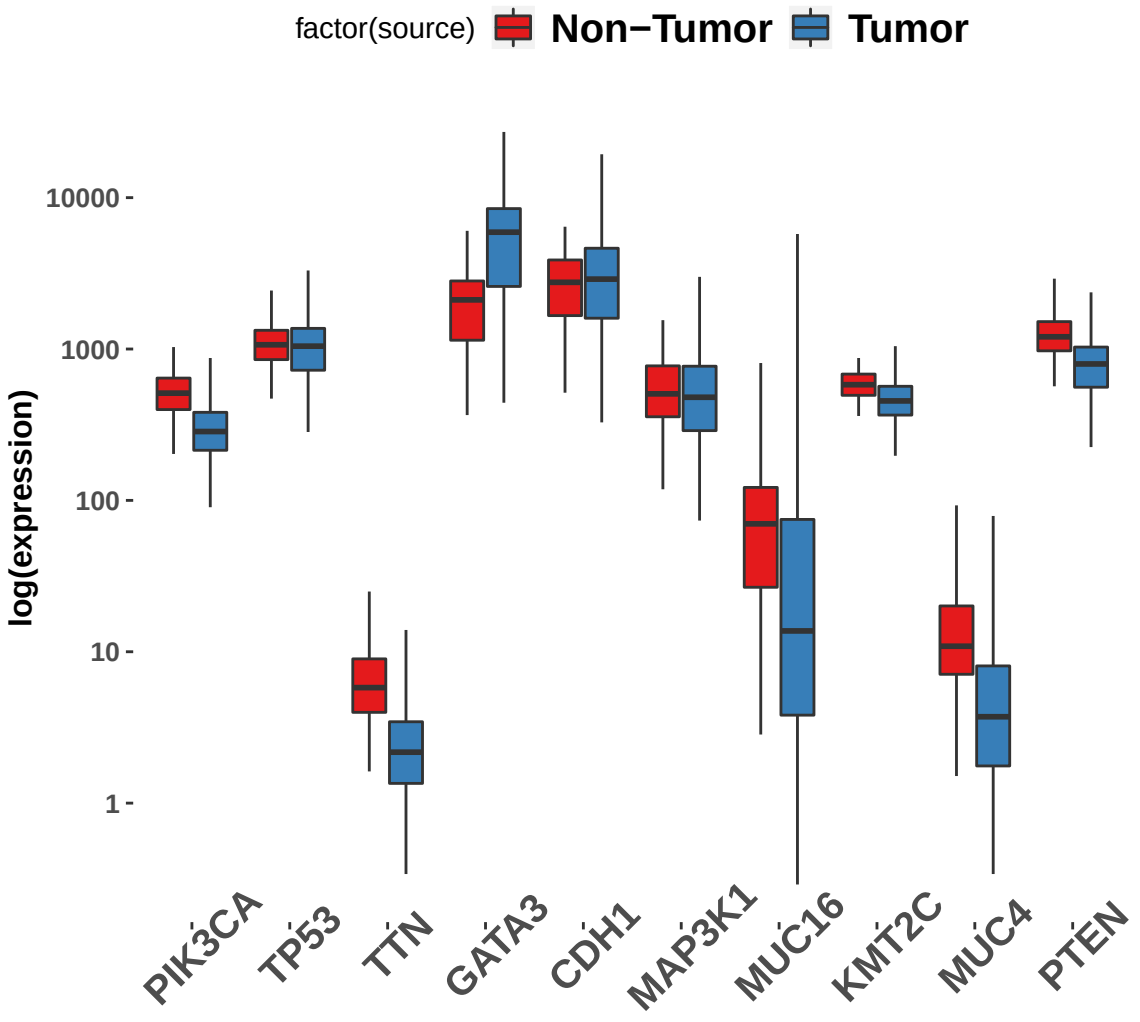

Figure 5: plotting express

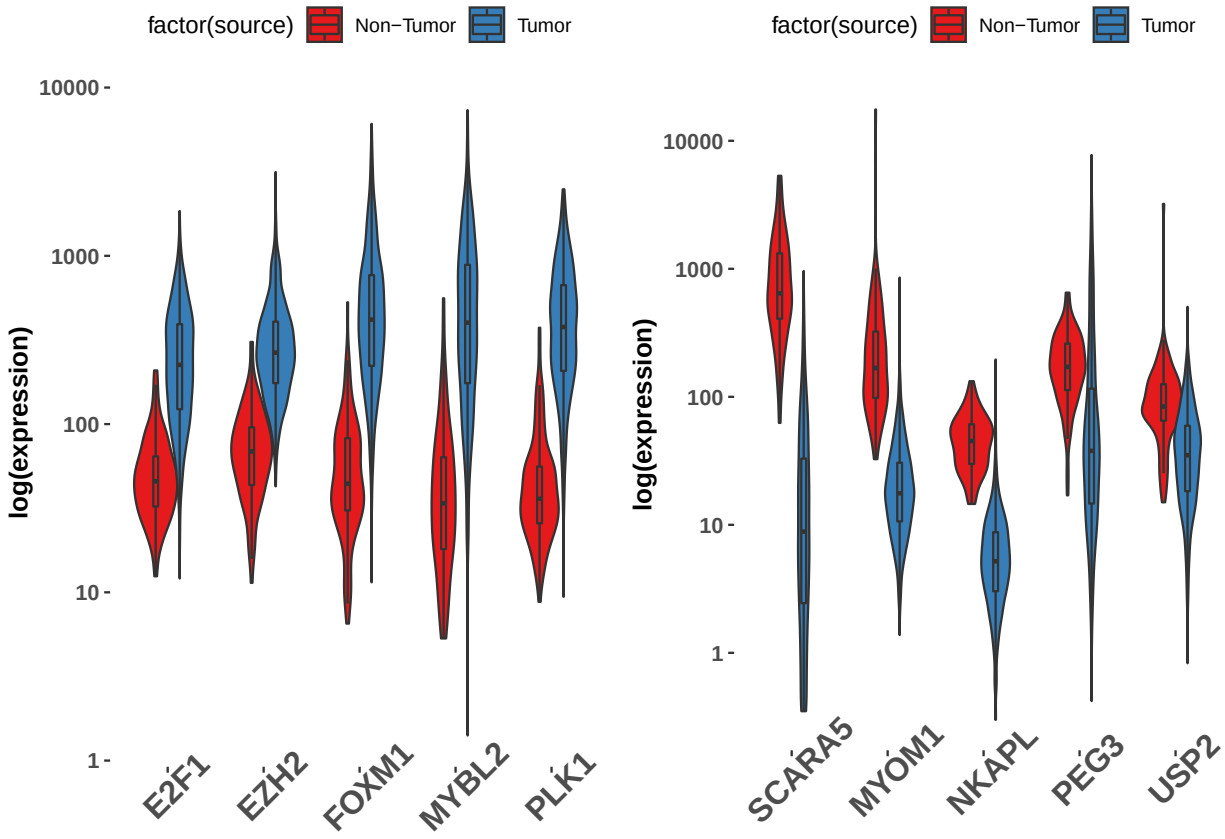

Figure 6: plotting express

Comparing expression of mutated BRCA genes with other cancer types

Finally, we wanted to see expression of genes in different cancer types.

```
# download more expression data read tumor data for
# brca,liver,stomach,colon,Lung
brca <- brcaexp_tumor

coad <- read_delim("https://raw.githubusercontent.com/mskcc/RNAseqDB/master/data/normalized/coad-rsem-f
    "\t", escape_double = FALSE, trim_ws = TRUE)
coadRownames <- coad$Hugo_Symbol
coad <- coad[, 3:dim(coad)[2]]
rownames(coad) <- coadRownames

lihc <- read_delim("https://raw.githubusercontent.com/mskcc/RNAseqDB/master/data/normalized/lihc-rsem-f
    "\t", escape_double = FALSE, trim_ws = TRUE)
lihcRownames <- lihc$Hugo_Symbol
lihc <- lihc[, 3:dim(lihc)[2]]
rownames(lihc) <- lihcRownames

luad <- read_delim("https://raw.githubusercontent.com/mskcc/RNAseqDB/master/data/normalized/luad-rsem-f
    "\t", escape_double = FALSE, trim_ws = TRUE)
luadRownames <- luad$Hugo_Symbol
luad <- luad[, 3:dim(luad)[2]]
rownames(luad) <- luadRownames

stad <- read_delim("https://raw.githubusercontent.com/mskcc/RNAseqDB/master/data/normalized/stad-rsem-f
    "\t", escape_double = FALSE, trim_ws = TRUE)
stadRownames <- stad$Hugo_Symbol
stad <- stad[, 3:dim(stad)[2]]
rownames(stad) <- stadRownames

topGenes <- c("PIK3CA", "TP53", "GATA3", "CDH1", "MAP3K1", "PTEN")
expVals <- c()
temp <- t(brca[topGenes, ])
colnames(temp) <- topGenes
meanbrca <- colMeans(temp)

temp <- t(coad[topGenes, ])
colnames(temp) <- topGenes
meancoad <- colMeans(temp)

temp <- t(lihc[topGenes, ])
colnames(temp) <- topGenes
meanlihc <- colMeans(temp)

temp <- t(luad[topGenes, ])
colnames(temp) <- topGenes
meanluad <- colMeans(temp)

temp <- t(stad[topGenes, ])
colnames(temp) <- topGenes
meanstad <- colMeans(temp)
```

```
organs <- c("breast", "colon", "liver", "lung", "stomach")
type <- c("other", "digestion", "digestion", "respiratory", "digestion")
colour <- c("#41ab5d", "orange", "orange", "steelblue", "orange")


plots <- list()
for (i in 1:6) {
    vals <- c(meanbrca[i], meancoad[i], meanlihc[i], meanluad[i],
        meanstad[i])
    gganatogramData <- data.frame(organ = organs, type = type,
        colour = colour, value = as.numeric(vals), stringsAsFactors = F)

    p <- gganatogram(data = gganatogramData, fillOutline = "#a6bddb",
        organism = "human", sex = "female", fill = "value") +
        theme_void() + scale_fill_gradient(low = "yellow", high = "red",
        name = paste(topGenes[i], "(fpkm)")) + theme(legend.text = element_text(size = 15,
        face = "bold"), legend.position = c(0.75, 0.2))

    plots[[i]] <- p
}

n <- length(plots)
nCol <- floor(sqrt(n))
do.call("grid.arrange", c(plots, ncol = 3))
```

The plots in fig shows that

Conclusion

System information

```
sessionInfo()
```

```
## R version 3.5.1 (2018-07-02)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 17134)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] grid      parallel  stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
```

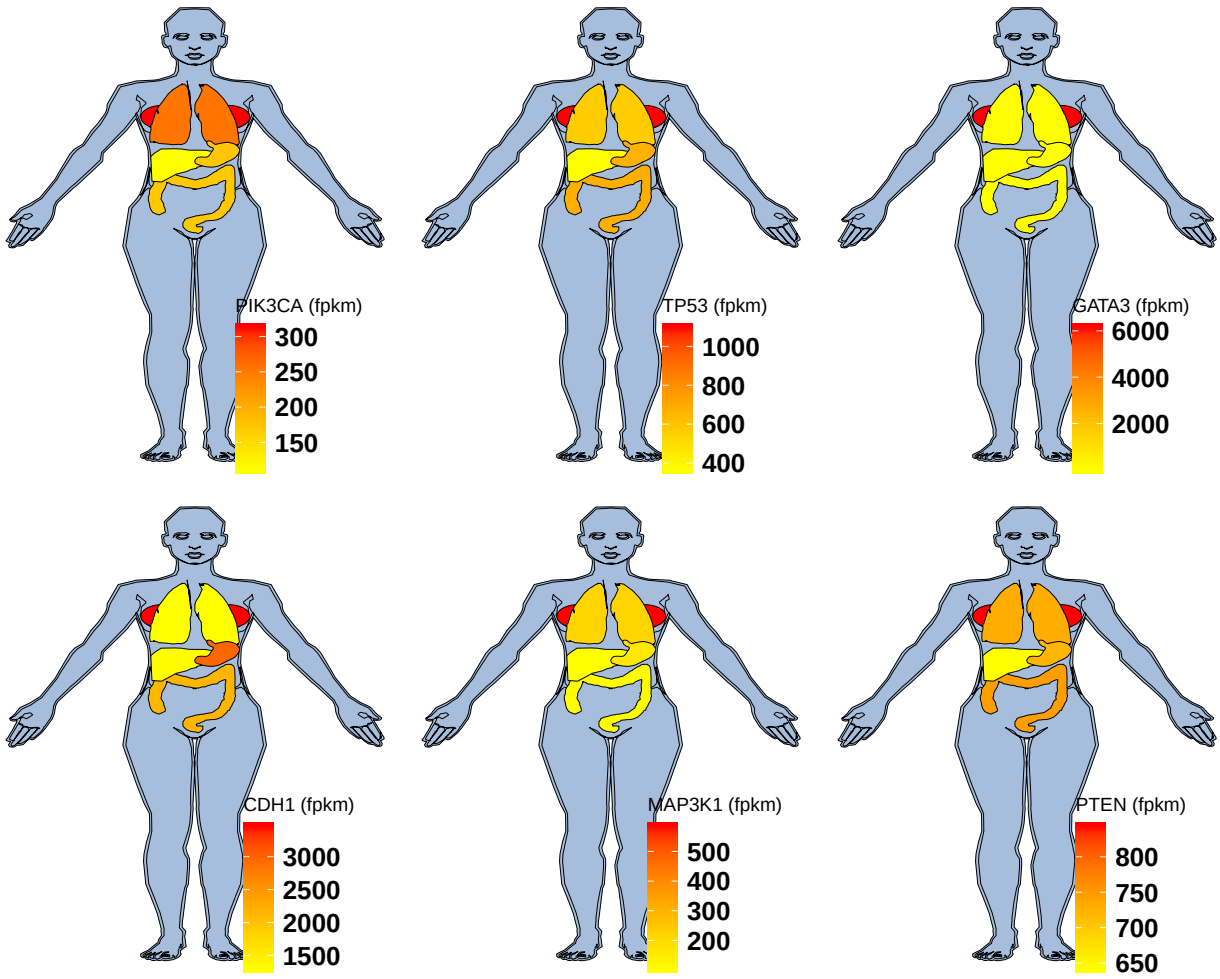

Figure 7: plotting express tissue

```
##  [1] bindrcpp_0.2.2      RColorBrewer_1.1-2  gridExtra_2.3
##  [4] viridis_0.5.1       viridisLite_0.3.0   gganatogram_1.1.1
##  [7] ggpolypath_0.1.0    ggplot2_3.1.0       readr_1.2.1
## [10] maftools_1.8.0      Biobase_2.42.0      BiocGenerics_0.28.0
## [13] plyr_1.8.4          data.table_1.11.8   DT_0.5
## [16] dplyr_0.7.8         TCGAbiolinks_2.10.0
##
## loaded via a namespace (and not attached):
##   [1] backports_1.1.2             circlize_0.4.5
##   [3] aroma.light_3.12.0          NMF_0.21.0
##   [5] selectr_0.4-1               ConsensusClusterPlus_1.46.0
##   [7] lazyeval_0.2.1              splines_3.5.1
##   [9] BiocParallel_1.16.2         GenomeInfoDb_1.18.1
##  [11] gridBase_0.4-7              sva_3.30.0
##  [13] digest_0.6.18               foreach_1.4.4
##  [15] htmltools_0.3.6             magrittr_1.5
##  [17] memoise_1.1.0               BSgenome_1.50.0
##  [19] cluster_2.0.7-1             doParallel_1.0.14
##  [21] limma_3.38.2                ComplexHeatmap_1.20.0
##  [23] Biostrings_2.50.1           annotate_1.60.0
##  [25] wordcloud_2.6               matrixStats_0.54.0
##  [27] R.utils_2.7.0               prettyunits_1.0.2
##  [29] colorspace_1.3-2            blob_1.1.1
##  [31] rvest_0.3.2                 ggrepel_0.8.0
##  [33] crayon_1.3.4                RCurl_1.95-4.11
##  [35] jsonlite_1.5                genefilter_1.64.0
##  [37] bindr_0.1.1                 survival_2.42-3
##  [39] zoo_1.8-4                   iterators_1.0.10
##  [41] glue_1.3.0                  survminer_0.4.3
##  [43] registry_0.5                gtable_0.2.0
##  [45] zlibbioc_1.28.0             XVector_0.22.0
##  [47] GetoptLong_0.1.7            DelayedArray_0.8.0
##  [49] shape_1.4.4                 scales_1.0.0
##  [51] DESeq_1.34.0                rngtools_1.3.1
##  [53] DBI_1.0.0                   edgeR_3.24.0
##  [55] bibtex_0.4.2                ggthemes_4.0.1
##  [57] Rcpp_1.0.0                  xtable_1.8-3
##  [59] progress_1.2.0              cmprsk_2.2-7
##  [61] mclust_5.4.2                bit_1.1-14
##  [63] matlab_1.0.2                km.ci_0.5-2
##  [65] stats4_3.5.1                htmlwidgets_1.3
##  [67] httr_1.3.1                  pkgconfig_2.0.2
##  [69] XML_3.98-1.16               R.methodsS3_1.7.1
##  [71] locfit_1.5-9.1              labeling_0.3
##  [73] tidyselect_0.2.5            rlang_0.3.0.1
##  [75] reshape2_1.4.3              AnnotationDbi_1.44.0
##  [77] munsell_0.5.0               tools_3.5.1
##  [79] downloader_0.4              RSQLite_2.1.1
##  [81] broom_0.5.0                 evaluate_0.12
##  [83] stringr_1.3.1               yaml_2.2.0
##  [85] knitr_1.20                  bit64_0.9-7
##  [87] survMisc_0.5.5              purrr_0.2.5
##  [89] EDASeq_2.16.0               nlme_3.1-137
##  [91] formatR_1.5                 R.oo_1.22.0
```

```
##  [93] xml2_1.2.0                 biomaRt_2.38.0
##  [95] compiler_3.5.1             curl_3.2
##  [97] tibble_1.4.2               geneplotter_1.60.0
##  [99] stringi_1.2.4              highr_0.7
## [101] GenomicFeatures_1.34.1     lattice_0.20-35
## [103] Matrix_1.2-14              KMsurv_0.1-5
## [105] pillar_1.3.0               GlobalOptions_0.1.0
## [107] cowplot_0.9.3              bitops_1.0-6
## [109] rtracklayer_1.42.1         GenomicRanges_1.34.0
## [111] R6_2.3.0                   latticeExtra_0.6-28
## [113] hwriter_1.3.2              ShortRead_1.40.0
## [115] IRanges_2.16.0             codetools_0.2-15
## [117] assertthat_0.2.0           SummarizedExperiment_1.12.0
## [119] pkgmaker_0.27              rprojroot_1.3-2
## [121] rjson_0.2.20               withr_2.1.2
## [123] GenomicAlignments_1.18.0   Rsamtools_1.34.0
## [125] S4Vectors_0.20.1           GenomeInfoDbData_1.2.0
## [127] mgcv_1.8-24                hms_0.4.2
## [129] tidyr_0.8.2                rmarkdown_1.10
## [131] ggpubr_0.2
```