doi:10.1038/nature15260

Supplementary Methods

Analysis of single-cell real-time qPCR experiments using dynamic arrays in R statistical language

I. R Script for unsupervised hierarchical clustering of normal mammary cells

For the article: Lawson, D.A. *et al.*, Single-cell analysis reveals a stem cell program in human metastatic breast cancer cells.

```
#Load libraries
library(gplots)
library(reshape2)
library(plyr)

# Load data from each dynamic array (chip), and then extract tissue type information
# The qPCR data was exported from Fluidigm in the table results format
df1 = read.csv("File location", na.strings = c("999",""),as.is=TRUE,skip=11)
df1 = df1[,c(1,2,5,7:13)]
names(df1) =
        c("position","sample","gene","ct","quality","call","ct.threshold","Tm_in_range","Tm.out.
        range","Tm.peak.ratio")
df1$chip = "1"

df1$tissue = NA
df1$tissue[grep("B",df1$sample)] = "B"
df1$tissue[grep("L",df1$sample)] = "L"
df1$tissue[grep("LP",df1$sample)] = "LP"

#repeat above for all dynamic arrays, df2, df3, etc.

# combine all dynamic array runs
dfcomb = rbind(df1, df2, df3…)

# retain only genes that were run on all dynamic arrays
dfcomb = ddply(dfcomb,.(gene),transform,commongene =
        length(unique(chip))==length(unique(dfcomb$chip)))
dfcomb = dfcomb[dfcomb$commongene,]

# remove genes that are 100% failures
dfcomb = ddply(dfcomb,.(gene),transform,removegene = all(is.na(ct)))
dfcomb = dfcomb[!dfcomb$removegene,]

# replace failed ct with max ct observed for that gene (to assign a value approximating # a lower
limit of detection when a cell does not express a gene)
dfcomb = ddply(dfcomb,.(gene),transform,maxct = max(ct,na.rm=TRUE))
```

```
dfcomb$ct[is.na(dfcomb$ct)] = dfcomb$maxct[is.na(dfcomb$ct)] + 1

# Ct normalization: Subtract mean of basal/stem cells on a per-gene, per-array basis to correct for
        batch-to-batch differences
dfcomb = ddply(dfcomb,.(chip,gene),transform,ct.conB = ct –
        mean(ct[tissue=="B"],na.rm=TRUE))
dfcomb$log2exp.conB = -1* dfcomb$ct.conB

#Specify dataset to cluster
dfhcl = dfcomb

# some sample names are repeated across chips, so create a unique id # by pasting together
sample and chip
dfhcl$id = paste(dfhcl$sample,dfhcl$chip)

# convert the data to a table that has samples as rows and genes as columns, and the log2
# expression value as the data in each cell of the table
expres = data.frame(dcast(dfhcl,id~gene,value.var = "log2exp.conB"),row.names=1)

# standardize (mean center and scale by standard deviation) the data across each gene
expres = t(scale(expres))

# get max/min 90% of data for the color lookup table that will be used to color the individual
# blocks of the heatmap. This prevents outliers from setting the color limits, which would
# otherwise compress most of the meaningful data to a few dim colors.
x = as.vector(expres)
z = (x-mean(x,na.rm=TRUE)) / sd(x,na.rm=TRUE)
rm.index = which(pnorm(abs(z), lower.tail=FALSE) < 0.05)
x[rm.index] = NA
maxvalue = max(x,na.rm=TRUE)
minvalue = min(x,na.rm=TRUE)

#OPTIONAL: set colors for sample tissue types
pheno =
        as.numeric(factor(dfhcl$tissue[match(colnames(expres),dfhcl$id)],levels=c("B","L",
        "LP”)))
phenocolors = c("blue”,”yellow”,”red”)[pheno]

# need to define distance function due to the presence of missing data:
# Pearson correlation that will prevent missing values in the distance matrix.
# pairs of data rows that have no pairs of observations in which both vectors have non-missing
# values will have a distance assigned to them of 1.1 times the maximum distance observed for
# any pairs of rows in the matrix.
mydistfunPearson = function(x) {
  t.dist = 1 - cor(t(x), use="pairwise")
  t.limit = 1.1*max(t.dist,na.rm=TRUE)
  t.dist[is.na(t.dist)] = t.limit
  t.dist = as.dist(t.dist)
  return(t.dist)
}

# use the heatmap.2 function from the gplots package. pdf("Filename.pdf",w=26,h=20)
heatmap.2(expres,distfun=mydistfunPearson,hclustfun=function(x)
        hclust(x,method="average"),breaks=seq(minvalue,maxvalue,length.out=76),col=bluered(
        75),na.color="grey",trace="none",ColSideColors=phenocolors,density.info="none",keysi
        ze=1.0,margins=c(2,7),cexRow=1,lmat=rbind(c(0,4),
        c(0,1),c(3,2),c(0,5)),lhei=c(0.90,0.30,5.05,1.00),lwid=c(1,11))
dev.off()
```

II. R Script for clustering of tumor cells from PDX mice

For the article: Lawson, D.A. *et al.*, Single-cell analysis reveals a stem cell program in human metastatic breast cancer cells.

```
#Load libraries
library(gplots)
library(reshape2)
library(plyr)
library(dendextend)

# Load data from each dynamic array (chip), and then extract tissue type information
# The qPCR data was exported from Fluidigm in the table results format.
df1 = read.csv("File location", na.strings = c("999",""),as.is=TRUE,skip=11)
df1 = df1[,c(1,2,5,7:13)]
names(df1) =
        c("position","sample","gene","ct","quality","call","ct.threshold","Tm_in_range","Tm.out.
        range","Tm.peak.ratio")
df1$chip = "1"

df1$tissue = NA
df1$tissue[grep("T",df1$sample)] = "T"
df1$tissue[grep("LN",df1$sample)] = "LN"
df1$tissue[grep("LU",df1$sample)] = "LU"
df1$tissue[grep("BR",df1$sample)] = "BR"
df1$tissue[grep("BM",df1$sample)] = "BM"
df1$tissue[grep("BM",df1$sample)] = "PB"

# repeat above for all dynamic arrays, df2, df3, etc.

# combine all dynamic array runs
dfcomb = rbind(df1, df2, df3…)

# retain only genes that were run on all dynamic arrays
dfcomb = ddply(dfcomb,.(gene),transform,commongene =
        length(unique(chip))==length(unique(dfcomb$chip)))
dfcomb = dfcomb[dfcomb$commongene,]

# remove genes that are 100% failures
dfcomb = ddply(dfcomb,.(gene),transform,removegene = all(is.na(ct)))
dfcomb = dfcomb[!dfcomb$removegene,]

# some sample names are repeated across chips, so create a unique id # by pasting together
sample and chip
dfcomb$id = paste(dfcomb$sample,dfhcl$chip)
```

```
# remove cell samples expressing low levels of all genes (set at 80% failure here)
dfcomb = ddply(dfcomb,.(sample,chip),transform,removesample = sum(is.na(ct))/length(ct)>0.8)
if (sum(dfcomb$removesample)>0) dfcomb = dfcomb[!dfcomb$removesample,]

# OPTIONAL: retain genes with specific names (for focus on specific gene networks).
dfcomb = dfcomb[dfcomb$gene %in% c("GENE1","GENE2","GENE3",…),]

# Ct normalization: Subtract mean of primary tumor cells on a per-gene, per-array basis to
# correct for batch-to-batch differences
dfcomb = ddply(dfcomb,.(chip,gene),transform,ct.conT = ct –
        mean(ct[tissue=="T"],na.rm=TRUE))
dfcomb$log2exp.conT = -1* dfcomb$ct.conT

#Specify dataset to cluster
dfhcl = dfcomb

#OPTIONAL: Remove primary tumor, or other specific tissues from heatmap
dfhcl = dfhcl[!dfhcl$tissue %in% c("T"),]

# convert the data to a data frame that has samples as rows and genes as columns, and the tumor
# normalized log2 expression value as the data in each cell of the table
expres = data.frame(dcast(dfhcl,id~gene,value.var = "log2exp.conT"),row.names=1)

# standardize (mean center and scale by standard deviation) the data across each gene
expres = t(scale(expres))

# get max/min 90% of data for the color lookup table that will be used to color the individual
# blocks of the heatmap. This prevents outliers from setting the color limits, which would
# otherwise compress most of the meaningful data to a few dim colors.
x = as.vector(expres)
z = (x-mean(x,na.rm=TRUE)) / sd(x,na.rm=TRUE)
rm.index = which(pnorm(abs(z), lower.tail=FALSE) < 0.05)
x[rm.index] = NA
maxvalue = max(x,na.rm=TRUE)
minvalue = min(x,na.rm=TRUE)

#OPTIONAL: For supervised clustering based on tissue identity
dfhcltissueorder=NA
dfhcltissueorder[dfhcl$tissue==”BR”] = 1
dfhcltissueorder[dfhcl$tissue==”PB”] = 2
dfhcltissueorder[dfhcl$tissue==”BM”] =3
dfhcltissueorder[dfhcl$tissue==”LN”] = 4
dfhcltissueorder[dfhcl$tissue==”LU”] = 5
dfhcl = dfhcl[order(dfhcl$tissue),]

#OPTIONAL: set colors for sample tissue types
```

```
pheno =
        as.numeric(factor(dfhcl$tissue[match(colnames(expres),dfhcl$id)],levels=c("T","LU",
        "LN","BR","PB","BM")))
phenocolors =
        c("yellow","mediumturquoise",”mediumpurple1”,"seagreen3","deeppink4","blue")[phen
o]

#OPTIONAL: generate groups manually by ID (e.g., the metastatic cells, rank ordered by
# metastatic burden in the tissue they were found in. In the study, there were 14 metastatic
# tissues and therefore 14 groups )
dfhcl$group = "nogroup"
dfhcl$group[dfhcl$id %in% c("LN1 1",”LN2 1”,”LN3 1”,…)] = “1”
dfhcl$group[dfhcl$id %in% c("LN4 1",”LN5 1”,”LN6 1”,…)] = “2”
dfhcl$group[dfhcl$id %in% c("LN7 1",”LN8 1”,”LN9 1”,…)] = “3”

#OPTIONAL: set colors for groups
pheno =
        as.numeric(factor(dfhcl$group[match(colnames(expres),dfhcl$id)],levels=c("1",
        "2","3","nogroup")))
phenocolors = c("grey85","grey50","black","white")[pheno]

# For unsupervised clustering, need to define distance function due to the presence of missing
data:
# Pearson correlation that will prevent missing values in the distance matrix
# pairs of data rows that have no pairs of observations in which both vectors have non-missing
# values will have a distance assigned to them of 1.1 times the maximum distance observed for
# any pairs of rows in the matrix.
mydistfunPearson = function(x) {
  t.dist = 1 - cor(t(x), use="pairwise")
  t.limit = 1.1*max(t.dist,na.rm=TRUE)
  t.dist[is.na(t.dist)] = t.limit
  t.dist = as.dist(t.dist)
  return(t.dist)
}

# use the heatmap.2 function from the gplots package.
pdf("Filename.pdf",w=24,h=12)
heatmap.2(expres,distfun=mydistfunPearson,hclustfun=function(x)
        hclust(x,method="average"),breaks=seq(minvalue,maxvalue,length.out=76),col=bluered(
        75),na.color="grey",trace="none",ColSideColors=phenocolors,density.info="none",keysi
        ze=1.0,margins=c(2,7),cexRow=1,lmat=rbind(c(0,4),
        c(0,1),c(3,2),c(0,5)),lhei=c(0.90,0.30,5.05,1.00),lwid=c(1,11))
dev.off()

# OPTIONAL: For supervised clustering of the samples:
pdf("Filename.pdf",w=24,h=12)
```

```
heatmap.2(expres,distfun=mydistfunPearson,hclustfun=function(x)
        hclust(x,method="average"),Colv=FALSE,breaks=seq(minvalue,maxvalue,length.out=76
        ),col=bluered(75),na.color="grey",trace="none",ColSideColors=phenocolors,density.info
        ="none",keysize=1.0,margins=c(2,7),cexRow=1,lmat=rbind(c(0,4),
        c(0,1),c(3,2),c(0,5)),lhei=c(1.25,0.35,5.70,0.60),lwid=c(1,11))
dev.off()

# OPTIONAL: For supervised clustering of the genes:
# i.e.To fix the gene order on the heatmap based on a previous unsupervised clustering run
# run this to get the order of the genes from the unsupervised clustering run
expres1=expres
thewts = rowMeans(expres1,na.rm=TRUE)
den = as.dendrogram(hclust(mydistfunPearson(expres1),method="average"))
den = reorder(den,wts=thewts)
geneorder = order.dendrogram(den)
geneorder = rev(geneorder)

# Then run this to do supervised clustering of the genes based on that previous run:
pdf("Filename.pdf",w=24,h=12)
heatmap.2(expres[geneorder,],distfun=mydistfunPearson,hclustfun=function(x)
        hclust(x,method="average"),Rowv=FALSE,dendrogram="column",breaks=seq(minvalue
        ,maxvalue,length.out=76),col=bluered(75),na.color="grey",trace="none",ColSideColors=
        phenocolors,density.info="none",keysize=1.0,margins=c(2,7),cexRow=1,lmat=rbind(c(0,
        4), c(0,1),c(3,2),c(0,5)),lhei=c(1.15,0.25,5.75,0.60),lwid=c(1,11))
dev.off()

# OPTIONAL: Re-order the dendrogram to optimize placement of group members next to each
# other:
den = as.dendrogram(hclust(mydistfunPearson(t(expres)),method="average"))
newden = reorder(den,pheno)
pdf("Filename.pdf",w=24,h=12)
heatmap.2(expres,distfun=mydistfunPearson,hclustfun=function(x)
        hclust(x,method="average"),Rowv=FALSE,dendrogram="column",Colv=newden,breaks
        =seq(minvalue,maxvalue,length.out=76),col=bluered(75),na.color="grey",trace="none",
        ColSideColors=phenocolors,density.info="none",keysize=1.0,margins=c(2,7),cexRow=1,
        lmat=rbind(c(0,4), c(0,1),c(3,2),c(0,5)),lhei=c(1.15,0.35,5.65,0.60),lwid=c(1,11))
dev.off()

# OPTIONAL: to rotate branches on a dendogram:
# chose the samples you want rotated. Inside c(…), put a comma-separated list of numbers that
# indicate the re-ordering of the tree by position (numbers). For the following example, the
# cluster containing samples 1-331 will be rotated. The remaining samples, from 332-889 will
# remain unrotated.
den = as.dendrogram(hclust(mydistfunPearson(t(expres)),method="average"))
newden = rotate(den,c(331:1,332:879))
pdf("Filename.pdf",w=24,h=12)
```

```
heatmap.2(expres[geneorder,],distfun=mydistfunPearson,hclustfun=function(x)
        hclust(x,method"average"),Rowv=FALSE,dendrogram="column",Colv=newden,breaks=
        seq(minvalue,maxvalue,length.out=76),col=bluered(75),na.color="grey",trace="none",Co
        lSideColors=phenocolors,density.info="none",keysize=1.0,margins=c(2,7),cexRow=1,lm
        at=rbind(c(0,4), c(0,1),c(3,2),c(0,5)),lhei=c(1.15,0.35,5.65,0.60),lwid=c(1,11))
dev.off()
```

III. R Script for principal component analysis (PCA) of normal mammary cells

For the article: Lawson, D.A. *et al.*, Single-cell analysis reveals a stem cell program in human metastatic breast cancer cells.

```
#Load libraries
library(ggplot2)
library(plyr)
library(reshape2)
library(grid)

# Load data from each dynamic array (chip), and then extract tissue type information
# The qPCR data was exported from Fluidigm in the table results format.
df1 = read.csv("File location", na.strings = c("999",""),as.is=TRUE,skip=11)
df1 = df1[,c(1,2,5,7:13)]
names(df1) =
        c("position","sample","gene","ct","quality","call","ct.threshold","Tm_in_range","Tm.out.
        range","Tm.peak.ratio")

df1$tissue = NA
df1$tissue[grep("L",df1$sample)] = "L"
df1$tissue[grep("LP",df1$sample)] = "LP"
df1$tissue[grep("LU",df1$sample)] = "B"

##################
# PCA
##################

dfpca$log2exp = -1*dfpca$ct

# remove genes that have failed in all samples
dfpca = ddply(dfpca,.(gene),transform,keepgene = !all(is.na(log2exp)))
dfpca = dfpca[dfpca$keepgene,]

# remove cell samples expressing low levels of all genes (set at 80% failure here)
dfpca = ddply(dfpca,.(sample),transform,remove = sum(is.na(log2exp))/length(log2exp)>0.8)
if (sum(dfpca$remove)>0) dfpca = dfpca[-which(dfpca$remove),]

# convert NA to the lowest expression value minus 1, on a per-gene basis, so all missing values
# are tied at the same very low value, with each value different for a different gene
dfpca = ddply(dfpca,.(gene),transform,min = min(log2exp,na.rm=TRUE)-1)
dfpca$log2exp[is.na(dfpca$log2exp)] = dfpca$min[is.na(dfpca$log2exp)]

# make a data.frame with samples as rows and genes as columns and log2 expression as value
df = data.frame(dcast(dfpca,sample~gene,value.var =
        "log2exp"),row.names=1,check.names=FALSE)
```

```
# compute PCA using singular value decomposition (SVD)
pcdat = prcomp(x=df,scale.=TRUE)

# get first two principal components and relevant phenotype data into data frame
pc =
        data.frame(pcdat$x[,1:2],tissue=dfpca$tissue[match(rownames(pcdat$x),dfpca$sample)],
        sample=rownames(pcdat$x))

# get loadings and retain genes with high loadings
loadings = data.frame(pcdat$rotation[,1:2])
loadings = loadings[apply(loadings, 1, function(x) max(abs(x))) > .0124,]

# amplify loadings so they can be visualized on the plot
loadings = loadings * 20
loadings$gene = rownames(loadings)

# OPTIONAL: choose loadings to visualize on the plot
loadings = loadings[loadings$gene %in% c("GENE1", "GENE2", "GENE3",…)]

# OPTIONAL: choose tissue colors and shapes
pc$tissue = factor(pc$tissue,levels=c("B", "LP", "L”))
mycolors = c("blue", "red", "yellow")
myshapes = c(16,15,17)

myTheme = theme(
  text = element_text(size=14,colour="black"),
  panel.background = element_blank(),
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(),
  axis.text=element_text(colour="black",size=16),
  axis.title.x=element_text(colour="black",size=18),
  axis.title.y=element_text(colour="black",size=18,angle=90),
  axis.line = element_line(size=1,colour="black"),
  axis.ticks = element_line(size=1,colour="black"),
  panel.border = element_blank(),
  legend.background = element_blank(),
  legend.key = element_blank(),
  legend.title = element_text(colour="black",size=12),
  legend.text = element_text(colour="black",size=12)
)

# create plot
p = ggplot(data=pc, aes(PC1, PC2)) + geom_point(aes(colour=tissue, shape=tissue),size=3.5)
p = p + scale_colour_manual(values=mycolors) + scale_size_manual(values=c(2,5,5)) +
        scale_shape_manual(values=myshapes)
```

```
p = p + geom_segment(data=loadings, aes(x=0, y=0, xend=PC1, yend=PC2),
          arrow=arrow(length=unit(0.2,"cm")), alpha=1.0)
p = p + geom_text(data=data.frame(PC1=sapply(loadings$PC1,function(x) ifelse(x>0,x+0.1,x-
          0.1)),PC2=sapply(loadings$PC2,function(x) ifelse(x>0,x+0.1,x-
          0.1)),gene=loadings$gene), aes(x=PC1, y=PC2, label=gene), alpha=1,
          size=5,face="bold")
p = p + geom_text(aes(label=sample))
print(p + myTheme)

pdf("Filename”)
print(p + myTheme)
dev.off()
```

IV. R Script for principal component analysis (PCA) of tumor cells from individual PDX mice

For the article: Lawson, D.A. *et al.*, Single-cell analysis reveals a stem cell program in human metastatic breast cancer cells.

```
#Load libraries
library(ggplot2)
library(plyr)
library(reshape2)
library(grid)

# Load data from each dynamic array (chip), and then extract tissue type information
# The qPCR data was exported from Fluidigm in the table results format.
df1 = read.csv("File location", na.strings = c("999",""),as.is=TRUE,skip=11)
df1 = df1[,c(1,2,5,7:13)]
names(df1) =
        c("position","sample","gene","ct","quality","call","ct.threshold","Tm_in_range","Tm.out.
        range","Tm.peak.ratio")

df1$tissue = NA
df1$tissue[grep("T",df1$sample)] = "T"
df1$tissue[grep("LN",df1$sample)] = "LN"
df1$tissue[grep("LU",df1$sample)] = "LU"
df1$tissue[grep("BR",df1$sample)] = "BR"
df1$tissue[grep("BM",df1$sample)] = "BM"
df1$tissue[grep("BM",df1$sample)] = "PB"


#################
# PCA
#################

dfpca$log2exp = -1*dfpca$ct

# remove genes that have failed in all samples
dfpca = ddply(dfpca,.(gene),transform,keepgene = !all(is.na(log2exp)))
dfpca = dfpca[dfpca$keepgene,]

# remove cell samples expressing low levels of all genes (set at 80% failure here)
dfpca = ddply(dfpca,.(sample),transform,remove = sum(is.na(log2exp))/length(log2exp)>0.8)
if (sum(dfpca$remove)>0) dfpca = dfpca[-which(dfpca$remove),]

# convert NA to the lowest expression value minus 1, on a per-gene basis, so all missing values
# are tied at the same very low value, with each value different for a different gene
dfpca = ddply(dfpca,.(gene),transform,min = min(log2exp,na.rm=TRUE)-1)
```

```
dfpca$log2exp[is.na(dfpca$log2exp)] = dfpca$min[is.na(dfpca$log2exp)]

# make a data.frame with samples as rows and genes as columns and log2 expression as value
df = data.frame(dcast(dfpca,sample~gene,value.var =
        "log2exp"),row.names=1,check.names=FALSE)

# compute PCA using singular value decomposition (SVD)
pcdat = prcomp(x=df,scale.=TRUE)

# get first two principal components and relevant phenotype data into data frame
pc =
        data.frame(pcdat$x[,1:2],tissue=dfpca$tissue[match(rownames(pcdat$x),dfpca$sample)],
        sample=rownames(pcdat$x))

# get loadings and retain genes with high loadings
loadings = data.frame(pcdat$rotation[,1:2])
loadings = loadings[apply(loadings, 1, function(x) max(abs(x))) > .0124,]

# amplify loadings so they can be visualized on the plot
loadings = loadings * 20
loadings$gene = rownames(loadings)

# OPTIONAL: choose loadings to visualize on the plot
loadings = loadings[loadings$gene %in% c("GENE1", "GENE2", "GENE3",…)]

# OPTIONAL: choose tissue colors and shapes
pc$tissue = factor(pc$tissue,levels=c("LN", "LU", "T"))
mycolors = c("red", "blue", "black")
myshapes = c(17,16,18)

myTheme = theme(
  text = element_text(size=14,colour="black"),
  panel.background = element_blank(),
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(),
  axis.text=element_text(colour="black",size=16),
  axis.title.x=element_text(colour="black",size=18),
  axis.title.y=element_text(colour="black",size=18,angle=90),
  axis.line = element_line(size=1,colour="black"),
  axis.ticks = element_line(size=1,colour="black"),
  panel.border = element_blank(),
  legend.background = element_blank(),
  legend.key = element_blank(),
  legend.title = element_text(colour="black",size=12),
  legend.text = element_text(colour="black",size=12)
)
```

```
# plot generation
p = ggplot(data=pc, aes(PC1, PC2)) + geom_point(aes(colour=tissue, shape=tissue),size=3.5)
p = p + scale_colour_manual(values=mycolors) + scale_size_manual(values=c(2,5,5)) +
        scale_shape_manual(values=myshapes)
p = p + geom_segment(data=loadings, aes(x=0, y=0, xend=PC1, yend=PC2),
        arrow=arrow(length=unit(0.2,"cm")), alpha=1.0)
p = p + geom_text(data=data.frame(PC1=sapply(loadings$PC1,function(x) ifelse(x>0,x+0.1,x-
        0.1)),PC2=sapply(loadings$PC2,function(x) ifelse(x>0,x+0.1,x-
        0.1)),gene=loadings$gene), aes(x=PC1, y=PC2, label=gene), alpha=1,
        size=5,face="bold")
p = p + geom_text(aes(label=sample))
print(p + myTheme)

pdf("Filename”)
print(p + myTheme)
dev.off()
```

V. R Script for generation of box plots to display gene expression in normal mammary cells

For the article: Lawson, D.A. *et al.*, Single-cell analysis reveals a stem cell program in human metastatic breast cancer cells.

```
#Load libraries
library(ggplot2)
library(plyr)
library(reshape2)

# Load data from each dynamic array (chip), and then extract tissue type information
# The qPCR data was exported from Fluidigm in the table results format
df1 = read.csv("File location", na.strings = c("999",""),as.is=TRUE,skip=11)
df1 = df1[,c(1,2,5,7:13)]
names(df1) =
        c("position","sample","gene","ct","quality","call","ct.threshold","Tm_in_range","Tm.out.
        range","Tm.peak.ratio")
df1$chip = "1"

df1$tissue = NA
df1$tissue[grep("L",df1$sample)] = "L"
df1$tissue[grep("LP",df1$sample)] = "LP"
df1$tissue[grep("LU",df1$sample)] = "B"

# repeat above for all dynamic arrays, df2, df3, etc.

# combine all dynamic array runs
dfcomb = rbind(df1, df2, df3…)

# remove genes that are 100% failures
dfcomb = ddply(dfcomb,.(gene),transform,removegene = all(is.na(ct)))
dfcomb = dfcomb[!dfcomb$removegene,]

# some sample names replicated across chips, but samples are not the same, so create a unique id
# by pasting together sample and chip
dfcomb$id = paste(dfcomb$sample,dfcomb$chip)

# remove cell samples expressing low levels of all genes (set at 80% failure here)
dfcomb = ddply(dfcomb,.(id),transform,removesample = sum(is.na(ct))/length(ct)>0.8)
if (sum(dfcomb$removesample)>0) dfcomb = dfcomb[!dfcomb$removesample,]

# Ct normalization: Subtract mean of basal/stem cells on a per-gene, per-array basis to correct for
# batch-to-batch differences
dfcomb = ddply(dfcomb,.(chip,gene),transform,ct.conB = ct –
        mean(ct[tissue=="B"],na.rm=TRUE))
dfcomb$log2exp.conB = -1* dfcomb$ct.conB
```

```
# choose order of tissue types on plots
dfcomb$tissue = factor(dfcomb$tissue,levels=c("B","LP","L"))

#####################
# MAKE PLOTS
#####################
myTheme = theme(
  text = element_text(size=14,colour="black"),
  panel.background = element_blank(),
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(),
  axis.text=element_text(colour="black",size=18),
  axis.title.x=element_text(colour="black",size=18),
  axis.title.y=element_text(colour="black",size=18,angle=90),
  axis.line = element_line(size=1,colour="black"),
  axis.ticks = element_line(size=1,colour="black"),
  panel.border = element_blank(),
  legend.background = element_blank(),
  legend.key = element_blank(),
  legend.title = element_text(colour="black",size=12),
  legend.text = element_text(colour="black",size=12),
  strip.text = element_text(colour="black",size=18),
  strip.background = element_rect(fill=NA)
)

# choose which data frame to plot (dfcomb)
dfbox = dfcomb

# display the data so that the failed points are plotted on a separate layer and vertically jittered
dfbox.stats = dfbox

# make a separate data frame just for the missing data in which NAs are converted to the lowest
# expression value minus 2, on a per-gene basis, so all missing values are tied at the same very
# low value, with each value different for a different gene
dfbox$failed = is.na(dfbox$log2exp.conB)
dfbox = ddply(dfbox,.(gene),transform,min = min(log2exp.conB,na.rm=TRUE)-2)
dfbox$log2exp.conB[dfbox$failed] = dfbox$min[dfbox$failed]
dfbox.failed = dfbox[dfbox$failed,]
dfbox$log2exp.conB[dfbox$failed] = NA

# plot B, LP, and L
pdf("Filename.pdf",w=30,h=30)
ggplot(dfbox,aes(tissue,log2exp.conB)) +
        geom_boxplot(data=dfbox.stats,outlier.size=NA,fill=NA) +
        geom_point(position=position_jitter(w=0.2,h=0),colour="black") +
        geom_point(data=dfbox.failed,position=position_jitter(w=0.2,h=0.5),colour="red") +
        facet_wrap(~gene,scales="free") + ylab("log2 relative expression, normalized by
        basal/stem cells on a per-gene basis") + myTheme
dev.off()
```

VI. R Script for generation of box plots to display gene expression in tumor cell populations

For the article: Lawson, D.A. *et al.*, Single-cell analysis reveals a stem cell program in human metastatic breast cancer cells.

```
# In this study, there were several types of box plots generated:
    1. Expression in metastatic tissues: Lung, lymph node, brain, peripheral blood, and bone
       marrow
    2. Expression by metastatic burden: Low-burden, intermediate-burden, high-burden, and
       primary tumor cells
    3. Expression in lung mets from different PDX models: HCI-001, HCI-002, HCI-010

#Load libraries
library(ggplot2)
library(plyr)
library(reshape2)

# Load data from each dynamic array (chip), and then extract tissue type information
# The qPCR data was exported from Fluidigm in the table results format
df1 = read.csv("File location", na.strings = c("999",""),as.is=TRUE,skip=11)
df1 = df1[,c(1,2,5,7:13)]
names(df1) =
        c("position","sample","gene","ct","quality","call","ct.threshold","Tm_in_range","Tm.out.range","Tm.peak.ratio")
df1$chip = "1"

df1$tissue = NA
df1$tissue[grep("T",df1$sample)] = "T"
df1$tissue[grep("LN",df1$sample)] = "LN"
df1$tissue[grep("LU",df1$sample)] = "LU"
df1$tissue[grep("BR",df1$sample)] = "BR"
df1$tissue[grep("BM",df1$sample)] = "BM"
df1$tissue[grep("BM",df1$sample)] = "PB"

# repeat above for all dynamic arrays, df2, df3, etc.

# combine all dynamic array runs
dfcomb = rbind(df1, df2, df3…)

# remove genes that are 100% failures
dfcomb = ddply(dfcomb,.(gene),transform,removegene = all(is.na(ct)))
dfcomb = dfcomb[!dfcomb$removegene,]

# some sample names replicated across chips, but samples are not the same, so create a unique id
# by pasting together sample and chip
dfcomb$id = paste(dfcomb$sample,dfcomb$chip)
```

```
# remove cell samples expressing low levels of all genes (set at 80% failure here)
dfcomb = ddply(dfcomb,.(id),transform,removesample = sum(is.na(ct))/length(ct)>0.8)
if (sum(dfcomb$removesample)>0) dfcomb = dfcomb[!dfcomb$removesample,]

# OPTIONAL: Remove samples that are not of interest (e.g., for PDX model comparison, only
# lung mets were examined and other mets were excluded)
dfcomb = dfcomb[!dfcomb$tissue %in% c("BR","LN","PB","BM"),]

# Ct normalization: Subtract mean of tumor cells on a per-gene, per-array basis to correct for
# batch-to-batch differences
dfcomb = ddply(dfcomb,.(chip,gene),transform,ct.conT = ct –
          mean(ct[tissue=="T"],na.rm=TRUE))
dfcomb$log2exp.conT = -1* dfcomb$ct.conT

# OPTIONAL: generate groups to manually designate colors by ID (specific cells)
# For grouping based on PDX model: (Note: In the study, this was only done on lung mets)
dfcomb$group = "nogroup"
dfcomb$group[dfcomb$id %in% c("LU1 1",”LU2 1”,”LU3 1”,…)] = “HCI-001”
dfcomb$group[dfcomb$id %in% c("LU4 1",”LU5 1”,”LU6 1”,…)] = “HCI-002”
dfcomb$group[dfcomb$id %in% c("LU7 1",”LU8 1”,”LU9 1”,…)] = “HCI-010”
dfcomb = dfcomb[dfcomb$group!="nogroup",]

# choose order of groups on plots
dfcomb$group = factor(dfcomb$group,levels=c("001","002","010"))

# OPTIONAL: generate groups to manually designate colors by ID (specific cells)
# For grouping based on low, intermediate, and high burden, and tumor:
dfcomb$group = "nogroup"
dfcomb$group[dfcomb$id %in% c("LN1 1",”LN2 1”,”LN3 1”,…)] = “low”
dfcomb$group[dfcomb$id %in% c("LN4 1",”LN5 1”,”LN6 1”,…)] = “intermediate”
dfcomb$group[dfcomb$id %in% c("LN7 1",”LN8 1”,”LN9 1”,…)] = “high”
dfcomb$group[dfcomb$tissue=="T"] = "T"
dfcomb = dfcomb[dfcomb$group!="nogroup",]

# OPTIONAL: choose order of groups on plots
dfcomb$group = factor(dfcomb$group,levels=c("T","early","mid","late"))


#####################
# MAKE PLOTS
#####################
myTheme = theme(
  text = element_text(size=14,colour="black"),
  panel.background = element_blank(),
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(),
  axis.text=element_text(colour="black",size=18),
  axis.title.x=element_text(colour="black",size=18),
  axis.title.y=element_text(colour="black",size=18,angle=90),
  axis.line = element_line(size=1,colour="black"),
  axis.ticks = element_line(size=1,colour="black"),
  panel.border = element_blank(),
  legend.background = element_blank(),
  legend.key = element_blank(),
  legend.title = element_text(colour="black",size=12),
  legend.text = element_text(colour="black",size=12),
  strip.text = element_text(colour="black",size=18),
  strip.background = element_rect(fill=NA)
)

# choose which data frame to plot (dfcomb)
dfbox = dfcomb

# display the data so that the failed points are plotted on a separate layer and vertically jittered
dfbox.stats = dfbox

# make a separate data frame just for the missing data in which NAs are converted to the lowest
# expression value minus 2, on a per-gene basis, so all missing values are tied at the same very
# low value, with each value different for a different gene
dfbox$failed = is.na(dfbox$log2exp.conT)
dfbox = ddply(dfbox,.(gene),transform,min = min(log2exp.conT,na.rm=TRUE)-2)
dfbox$log2exp.conT[dfbox$failed] = dfbox$min[dfbox$failed]
dfbox.failed = dfbox[dfbox$failed,]
dfbox$log2exp.conT[dfbox$failed] = NA

# OPTIONAL: plot lung mets by PDX model: HCI-001, HCI-002, HCI-010
pdf("Filename.pdf",w=30,h=30)
ggplot(dfbox,aes(group,log2exp.conT)) +
        geom_boxplot(data=dfbox.stats,outlier.size=NA,fill=NA) +
        geom_point(position=position_jitter(w=0.2,h=0),colour='black') +
        geom_point(data=dfbox.failed,position=position_jitter(w=0.2,h=0.5),colour='red') +
        facet_wrap(~gene,scales="free") + ylab("log2 relative expression, chips normalized by
        tumor cells on a per-gene basis") + myTheme
dev.off()


# OPTIONAL: plot mets by burden (low, intermediate, high, T)
pdf("Filename.pdf",w=30,h=30)
ggplot(dfbox,aes(group,log2exp.conT)) +
        geom_boxplot(data=dfbox.stats,outlier.size=NA,fill=NA) +
        geom_point(position=position_jitter(w=0.2,h=0),colour="black") +
        geom_point(data=dfbox.failed,position=position_jitter(w=0.2,h=0.5),colour="red") +
        facet_wrap(~gene,scales="free") + ylab("log2 relative expression, chips normalized by
        tumor cells on a per-gene basis") + myTheme
dev.off()

# OPTIONAL: plot mets by tissue (T, Lung, lymph node, brain, peripheral blood, bone marrow)
pdf("Filename.pdf",w=30,h=30)
ggplot(dfbox,aes(tissue,log2exp.conT)) +
        geom_boxplot(data=dfbox.stats,outlier.size=NA,fill=NA) +
        geom_point(position=position_jitter(w=0.2,h=0),colour="black") +
        geom_point(data=dfbox.failed,position=position_jitter(w=0.2,h=0.5),colour="red") +
        facet_wrap(~gene,scales="free") + ylab("log2 relative expression, normalized by tumor
        cells on a per-gene basis") + myTheme
dev.off()
```

VII. R Script for identification of differentially expressed genes in normal mammary cells

For the article: Lawson, D.A. *et al.*, Single-cell analysis reveals a stem cell program in human metastatic breast cancer cells.

```
#Load libraries
library(ggplot2)
library(plyr)
library(limma)
library(reshape2)

# Load data from each dynamic array (chip), and then extract tissue type information
# The qPCR data was exported from Fluidigm in the table results format
df1 = read.csv("File location", na.strings = c("999",""),as.is=TRUE,skip=11)
df1 = df1[,c(1,2,5,7:13)]
names(df1) =
        c("position","sample","gene","ct","quality","call","ct.threshold","Tm_in_range","Tm.out.
        range","Tm.peak.ratio")

df1$tissue = NA
df1$tissue[grep("L",df1$sample)] = "L"
df1$tissue[grep("LP",df1$sample)] = "LP"
df1$tissue[grep("LU",df1$sample)] = "B"

# repeat above for all dynamic arrays, df2, df3, etc.

# combine all dynamic array runs
dfcomb = rbind(df1, df2, df3…)

# remove genes that are 100% failures
dfcomb = ddply(dfcomb,.(gene),transform,removegene = all(is.na(ct)))
dfcomb = dfcomb[!dfcomb$removegene,]

# some sample names repeated across chips, so create a unique id # by pasting together sample
and chip
dfcomb$id = paste(dfcomb$sample,dfcomb$chip)

# Ct normalization: Subtract mean of basal/stem cells on a per-gene, per-array basis to correct for
# batch-to-batch differences
dfcomb = ddply(dfcomb,.(chip,gene),transform,ct.conB = ct –
        mean(ct[tissue=="B"],na.rm=TRUE))
dfcomb$log2exp.conB = -1*dfcomb$ct.conB
```

```
#####################################
#####################################
# Three-group comparisons (B vs. L vs. LP)
#####################################
#####################################

# define helper computation function first

######################################
# START OF FUNCTION DEFINTION
######################################

compute.results = function(dfavg,ptypes,filename,sample.column.name,expressionchoice) {

  dfavg$expression.choice = dfavg[,expressionchoice]
  dfavg$sample = dfavg[,sample.column.name]
  dfavg$log2exp = dfavg$expression.choice

  dedata=dfavg

  # retain phenotypes of interest
  dedata = dedata[dedata[,ptypes["phenotype"]] %in%
          c(ptypes["level1"],ptypes["level2"],ptypes["level3"]),]

  # remove genes that failed in all samples
  dedata = ddply(dedata,.(gene),transform,remove = all(is.na(expression.choice)))
  if (sum(dedata$remove)>0) dedata = dedata[-which(dedata$remove),]

  # record how many NA there are per condition for each gene
  genedata = data.frame(unique(dedata$gene), 0, 0, 0, 0, 0,0,0)
  names(genedata) =
          c("gene",ptypes[2],ptypes[3],ptypes[4],"failed.total",paste("failed",ptypes[2],sep="."),pas
          te("failed",ptypes[3],sep="."),paste("failed",ptypes[4],sep="."))
  expres = data.frame(dcast(dedata,sample~gene,value.var =
          "log2exp"),row.names=1,check.names=FALSE)
  pheno = dedata[,ptypes[1]][match(rownames(expres),dedata$sample)]
  pheno  = factor(pheno,levels=c(ptypes[2],ptypes[3],ptypes[4]))
  for (i in 1:dim(genedata)[1]) {
          genedata[,ptypes[2]][i] = sum(pheno==ptypes[2]) –
                  sum((as.numeric(is.na(t(expres)[match(genedata$gene[i],names(expres)),])) *
                  as.numeric(pheno)) == 1)
          genedata[,ptypes[3]][i] = sum(pheno==ptypes[3]) –
                  sum((as.numeric(is.na(t(expres)[match(genedata$gene[i],names(expres)),])) *
                  as.numeric(pheno)) == 2)
          genedata[,ptypes[4]][i] = sum(pheno==ptypes[4]) –
                  sum((as.numeric(is.na(t(expres)[match(genedata$gene[i],names(expres)),])) *
                  as.numeric(pheno)) == 3)
          genedata$failed.total[i] = sum(is.na(t(expres)[match(genedata$gene[i],names(expres)),]))
          genedata[,paste("failed",ptypes[2],sep=".")][i] =
                  sum(is.na(t(expres)[match(genedata$gene[i],names(expres)),]) &
                  pheno==ptypes[2])
          genedata[,paste("failed",ptypes[3],sep=".")][i] =
                  sum(is.na(t(expres)[match(genedata$gene[i],names(expres)),]) &
                  pheno==ptypes[3])
          genedata[,paste("failed",ptypes[4],sep=".")][i] =
                  sum(is.na(t(expres)[match(genedata$gene[i],names(expres)),]) &
                  pheno==ptypes[4])

  }

# convert NA to the lowest expression value minus 0.01, on a per-gene basis, so all missing
# values are tied at the same very low value, with each value different for a different gene
dedata$wasNA = FALSE
dedata$wasNA[is.na(dedata$log2exp)] = TRUE
dedata = ddply(dedata,.(gene),transform,min = min(log2exp,na.rm=TRUE)-0.01)
dedata$log2exp[is.na(dedata$log2exp)] = dedata$min[is.na(dedata$log2exp)]

################
# ANOVA
################
an = ddply(dedata,.(gene),summarize,p.an=anova(lm(log2exp~factor(tissue)))[,5][1])
an$FDR.an = p.adjust(an$p.an,method="BH")

################################
# Kruskal-Wallis rank sum test
################################
kw = ddply(dedata,.(gene),summarize,p.kw=kruskal.test(log2exp,factor(tissue))$p.value)
kw$FDR.kw = p.adjust(kw$p.kw,method="BH")

################
# Merge results
################
res = merge(genedata,an)
res = merge(res,kw)

############################################
# Pick a recommended p and list minimum p
############################################
res$p.recommended.type = "p.an"
res$p.recommended.type[(res[,paste("failed",ptypes[2],sep=".")]/(res[,paste("failed",ptypes[2],
          sep=".")] + res[,ptypes[2]])) > 0.5 |
        (res[,paste("failed",ptypes[3],sep=".")]/(res[,paste("failed",ptypes[3],sep=".")] +
        res[,ptypes[3]])) > 0.5 |
        (res[,paste("failed",ptypes[4],sep=".")]/(res[,paste("failed",ptypes[4],sep=".")] +
        res[,ptypes[4]])) > 0.5] = "kw"
res$p.recommended = res$p.an
res$p.recommended[res$p.recommended.type=="kw"] =
        res$p.kw[res$p.recommended.type=="kw"]
# compute FDR from list of recommended p values
res$FDR.recommended = p.adjust(res$p.recommended,method="BH")
# sort stats by p value
res = res[order(res$p.recommended),]
# list the minimum across the tests for each gene in separate column
res$p.minimum = apply(res[,c("p.an","p.kw")],1,function(x) min(x,na.rm=TRUE))
res$FDR.minimum = p.adjust(res$p.minimum,method="BH")

#################
# Write data to a file
#################
write.csv(res,file=filename,quote=FALSE,row.names=FALSE,na="")

# return the data frame of stats
return(res)
}

#####################################
# END OF FUNCTION DEFINTION
#####################################

######################################################
# Use the function defined above to compute statistics: three-group
######################################################

# compare B, LP, L
ptypes = c(phenotype="tissue",level1="B",level2="LP",level3="L")
resBvsLPvsL = compute.results(dfcomb,ptypes,"Filename.csv","id","log2exp.conB")

#################################
#################################
# Two-group comparisons (e.g., B vs. L)
#################################
#################################

# define computation function first
```

```
#######################################
# START OF FUNCTION DEFINTION
#######################################

compute.results = function(dfavg,ptypes,filename,sample.column.name,expressionchoice) {

  dfavg$expression.choice = dfavg[,expressionchoice]
  dfavg$sample = dfavg[,sample.column.name]
  dfavg$log2exp = dfavg$expression.choice

  dedata=dfavg

  # retain phenotypes of interest
  dedata = dedata[dedata[,ptypes["phenotype"]] %in% c(ptypes["level1"],ptypes["level2"]),]

  # remove genes that failed in all samples
  dedata = ddply(dedata,.(gene),transform,remove = all(is.na(expression.choice)))
  if (sum(dedata$remove)>0) dedata = dedata[-which(dedata$remove),]

  # record how many NA there are per condition for each gene
  genedata = data.frame(unique(dedata$gene), 0, 0, 0, 0, 0)
  names(genedata) =
          c("gene",ptypes[2],ptypes[3],"failed.total",paste("failed",ptypes[2],sep="."),paste("failed"
          ,ptypes[3],sep="."))
  expres = data.frame(dcast(dedata,sample~gene,value.var =
          "log2exp"),row.names=1,check.names=FALSE)
  pheno = dedata[,ptypes[1]][match(rownames(expres),dedata$sample)]
  pheno  = factor(pheno,levels=c(ptypes[2],ptypes[3]))
  for (i in 1:dim(genedata)[1]) {
          genedata[,ptypes[2]][i] = sum(pheno==ptypes[2]) –
                    sum((as.numeric(is.na(t(expres)[match(genedata$gene[i],names(expres)),])) *
                    as.numeric(pheno)) == 1)
          genedata[,ptypes[3]][i] = sum(pheno==ptypes[3]) –
                    sum((as.numeric(is.na(t(expres)[match(genedata$gene[i],names(expres)),])) *
                    as.numeric(pheno)) == 2)
          genedata$failed.total[i] = sum(is.na(t(expres)[match(genedata$gene[i],names(expres)),]))
          genedata[,paste("failed",ptypes[2],sep=".")][i] =
                    sum(is.na(t(expres)[match(genedata$gene[i],names(expres)),]) &
                    pheno==ptypes[2])
          genedata[,paste("failed",ptypes[3],sep=".")][i] =
                    sum(is.na(t(expres)[match(genedata$gene[i],names(expres)),]) &
                    pheno==ptypes[3])
  }

# convert NA to the lowest expression value minus 0.01, on a per-gene basis, so all missing
# values are tied at the same very low value, with each value different for a different gene
```

```
dedata$wasNA = FALSE
dedata$wasNA[is.na(dedata$log2exp)] = TRUE
dedata = ddply(dedata,.(gene),transform,min = min(log2exp,na.rm=TRUE)-0.01)
dedata$log2exp[is.na(dedata$log2exp)] = dedata$min[is.na(dedata$log2exp)]

################
# limma
################
expres = data.frame(dcast(dedata,sample~gene,value.var =
          "log2exp"),row.names=1,check.names=FALSE)
pheno = dedata[,ptypes[1]][match(rownames(expres),dedata$sample)]
pheno  = factor(pheno,levels=c(ptypes[2],ptypes[3]))
design = model.matrix(~pheno)
fit=lmFit(t(expres),design)
fit=eBayes(fit)
limma = topTable(fit, coef=2, adjust.method="BH", sort.by="p", number=1000)
limma = limma[,-which(names(limma)=="B")]
names(limma) = paste(names(limma),"limma",sep=".")
names(limma)[which(names(limma)=="ID.limma")] = "gene"
names(limma)[which(names(limma)=="adj.P.Val.limma")] = "FDR.limma"
names(limma)[which(names(limma)=="P.Value.limma")] = "p.limma"

####################
# Mann-Whitney
####################
wilcox = ddply(dedata,.(gene),.fun = function(x,colname) summarize(x,logFC.wilcox =
          mean(x[,"log2exp"][x[,colname]==ptypes["level2"]],na.rm=TRUE)-
          mean(x[,"log2exp"][x[,colname]==ptypes["level1"]],na.rm=TRUE),logFC.NAexcluded
          = mean(x[,"log2exp"][x[,colname]==ptypes["level2"] &
          x[,"wasNA"]!=TRUE],na.rm=TRUE)-
          mean(x[,"log2exp"][x[,colname]==ptypes["level1"] &
          x[,"wasNA"]!=TRUE],na.rm=TRUE),p.wilcox =
          wilcox.test(x[,"log2exp"][x[,colname]==ptypes["level2"]],x[,"log2exp"][x[,colname]==p
          types["level1"]])$p.value),colname=ptypes[1])
wilcox$FDR.wilcox = p.adjust(wilcox$p.wilcox,method="BH")

####################
# Welch t
####################
welch = ddply(dedata,.(gene),.fun = function(x,colname) summarize(x,p.welch =
          t.test(x[,"log2exp"][x[,colname]==ptypes["level2"]],x[,"log2exp"][x[,colname]==ptypes[
          "level1"]])$p.value),colname=ptypes[1])
welch$FDR.welch = p.adjust(welch$p.welch,method="BH")
```

```
################
# Merge results
################
res = merge(genedata,limma)
res = merge(res,wilcox)
res = merge(res,welch)

############################################
# Pick a recommended p and list minimum p
############################################
# recommended, and sort list based on this p value
res$p.recommended.type = "t.limma"
res$p.recommended.type[(res[,paste("failed",ptypes[2],sep=".")]/(res[,paste("failed",ptypes[2],se
        p=".")] + res[,ptypes[2]])) > 0.5 |
        (res[,paste("failed",ptypes[3],sep=".")]/(res[,paste("failed",ptypes[3],sep=".")] +
        res[,ptypes[3]])) > 0.5] = "wilcox"
res$p.recommended = res$p.limma
res$p.recommended[res$p.recommended.type=="wilcox"] =
res$p.wilcox[res$p.recommended.type=="wilcox"]
res$FDR.recommended = p.adjust(res$p.recommended,method="BH")
res = res[order(res$p.recommended),]
# minimum
res$p.minimum = apply(res[,c("p.limma","p.wilcox","p.welch")],1,function(x)
        min(x,na.rm=TRUE))
res$FDR.minimum = p.adjust(res$p.minimum,method="BH")

#############
# Write data
#############
write.csv(res,file=filename,quote=FALSE,row.names=FALSE,na="")

return(res)
}

########################################
# END OF FUNCTION DEFINTION
########################################

##########################################################
# Use the function defined above to compute statistics: two-group
##########################################################

# grouped comparison (B vs. (L+LP))
dfcomb$group = "L+LP"
dfcomb$group[dfcomb$tissue=="B"] = "B"
ptypes = c(phenotype="group",level1="L+LP",level2="B")
```

```
resBvsLLP = compute.results(dfcomb,ptypes,"Filename.csv","id","log2exp.conB")

# pairwise comparison (L vs. B)
dfcomb$group = "B"
dfcomb$group[dfcomb$tissue=="L"] = "L"
ptypes = c(phenotype="group",level1="B",level2="L")
resLvsB = compute.results(dfcomb,ptypes,"Filename.csv","id","log2exp.conB")

# pairwise comparison (LP vs. L)
dfcomb$group = "L"
dfcomb$group[dfcomb$tissue=="LP"] = "LP"
ptypes = c(phenotype="group",level1="L",level2="LP")
resLPvsL = compute.results(dfcomb,ptypes,"Filename.csv","id","log2exp.conB")
```

VIII. R Script for identification of differentially expressed genes in cancer cell populations

For the article: Lawson, D.A. *et al.*, Single-cell analysis reveals a stem cell program in human metastatic breast cancer cells.

```
# In this study, there were several types of differential expression analyses:
    4.  Lung vs. lymph node vs. brain vs. peripheral blood vs. bone marrow mets– 5 groups
    5.  Low-burden mets vs primary tumor cells – 2 groups
    6.  High-burden mets and primary tumor cells – 2 groups
    7.  High-burden mets vs low-buden mets – 2 groups
    8.  Lung mets from HCI-001 vs. HCI-002 vs. HCI-010 – 3 groups

#Load libraries
library(ggplot2)
library(plyr)
library(limma)
library(reshape2)
library(PMCMR)

# Load data from each dynamic array (chip), and then extract tissue type information
# The qPCR data was exported from Fluidigm in the table results format
df1 = read.csv("File location", na.strings = c("999",""),as.is=TRUE,skip=11)
df1 = df1[,c(1,2,5,7:13)]
names(df1) =
        c("position","sample","gene","ct","quality","call","ct.threshold","Tm_in_range","Tm.out.
        range","Tm.peak.ratio")
df1$chip = “1”

df1$tissue = NA
df1$tissue[grep("T",df1$sample)] = "T"
df1$tissue[grep("LN",df1$sample)] = "LN"
df1$tissue[grep("LU",df1$sample)] = "LU"
df1$tissue[grep("BR",df1$sample)] = "BR"
df1$tissue[grep("BM",df1$sample)] = "BM"
df1$tissue[grep("BM",df1$sample)] = "PB"

# combine all dynamic array runs
dfcomb = rbind(df1, df2, df3…)

# OPTIONAL: retain only genes that were run on all chips
dfcomb = ddply(dfcomb,.(gene),transform,commongene =
        length(unique(chip))==length(unique(dfcomb$chip)))
dfcomb = dfcomb[dfcomb$commongene,]

# remove genes that are 100% failures
dfcomb = ddply(dfcomb,.(gene),transform,removegene = all(is.na(ct)))
dfcomb = dfcomb[!dfcomb$removegene,]

# some sample names repeated across chips, so create a unique id # by pasting together sample
and chip
dfcomb$id = paste(dfcomb$sample,dfcomb$chip)

# Ct normalization: Subtract mean of basal/stem cells on a per-gene, per-array basis to correct for
# batch-to-batch differences
dfcomb = ddply(dfcomb,.(chip,gene),transform,ct.conT = ct –
            mean(ct[tissue=="T"],na.rm=TRUE))
dfcomb$log2exp.conT = -1*dfcomb$ct.conT

# OPTIONAL: generate groups to manually designate colors by ID (specific cells)
# For grouping based on low, intermediate, and high burden, and tumor:
dfcomb$group = "nogroup"
dfcomb$group[dfcomb$id %in% c("LN1 1",”LN2 1”,”LN3 1”,…)] = “low”
dfcomb$group[dfcomb$id %in% c("LN4 1",”LN5 1”,”LN6 1”,…)] = “intermediate”
dfcomb$group[dfcomb$id %in% c("LN7 1",”LN8 1”,”LN9 1”,…)] = “high”
dfcomb$group[dfcomb$tissue=="T"] = "T"
dfcomb = dfcomb[dfcomb$group!="nogroup",]

#############################################################
#############################################################
# Two-group comparisons (e.g., low-burden mets vs. primary tumor cells)
#############################################################
#############################################################

# define helper computation function first

######################################
# START OF FUNCTION DEFINTION
######################################

compute.results = function(dfavg,ptypes,filename,sample.column.name,expressionchoice) {

  dfavg$expression.choice = dfavg[,expressionchoice]
  dfavg$sample = dfavg[,sample.column.name]
  dfavg$log2exp = dfavg$expression.choice

  dedata=dfavg

  # retain phenotypes of interest
  dedata = dedata[dedata[,ptypes["phenotype"]] %in% c(ptypes["level1"],ptypes["level2"]),]

  # remove genes that do not have at least three non-NA values for each level of the phenotype
```

```
 dedata = ddply(dedata,.(gene),.fun = function(x,colname) transform(x,remove =
          sum(!is.na(x[,"log2exp"][x[,colname]==ptypes["level2"]]))<3 |
          sum(!is.na(x[,"log2exp"][x[,colname]==ptypes["level1"]]))<3),colname=ptypes[1])
 if (sum(dedata$remove)>0) dedata = dedata[-which(dedata$remove),]

 # record how many NA there are per condition for each gene
 genedata = data.frame(unique(dedata$gene), 0, 0, 0, 0, 0)
 names(genedata) =
          c("gene",ptypes[2],ptypes[3],"failed.total",paste("failed",ptypes[2],sep="."),paste("failed"
          ,ptypes[3],sep="."))
 expres = data.frame(dcast(dedata,sample~gene,value.var =
          "log2exp"),row.names=1,check.names=FALSE)
 pheno = dedata[,ptypes[1]][match(rownames(expres),dedata$sample)]
 pheno  = factor(pheno,levels=c(ptypes[2],ptypes[3]))
 for (i in 1:dim(genedata)[1]) {
          genedata[,ptypes[2]][i] = sum(pheno==ptypes[2]) –
                    sum((as.numeric(is.na(t(expres)[match(genedata$gene[i],names(expres)),])) *
                    as.numeric(pheno)) == 1)
          genedata[,ptypes[3]][i] = sum(pheno==ptypes[3]) –
                    sum((as.numeric(is.na(t(expres)[match(genedata$gene[i],names(expres)),])) *
                    as.numeric(pheno)) == 2)
          genedata$failed.total[i] = sum(is.na(t(expres)[match(genedata$gene[i],names(expres)),]))
          genedata[,paste("failed",ptypes[2],sep=".")][i] =
                    sum(is.na(t(expres)[match(genedata$gene[i],names(expres)),]) &
                    pheno==ptypes[2])
          genedata[,paste("failed",ptypes[3],sep=".")][i] =
                    sum(is.na(t(expres)[match(genedata$gene[i],names(expres)),]) &
                    pheno==ptypes[3])
 }

# convert NA to the lowest expression value minus 0.01, on a per-gene basis, so all missing
# values are tied at the same very low value, with each value different for a different gene
dedata$wasNA = FALSE
dedata$wasNA[is.na(dedata$log2exp)] = TRUE
dedata = ddply(dedata,.(gene),transform,min = min(log2exp,na.rm=TRUE)-0.01)
dedata$log2exp[is.na(dedata$log2exp)] = dedata$min[is.na(dedata$log2exp)]

###############
# limma
###############
expres = data.frame(dcast(dedata,sample~gene,value.var =
          "log2exp"),row.names=1,check.names=FALSE)
pheno = dedata[,ptypes[1]][match(rownames(expres),dedata$sample)]
pheno  = factor(pheno,levels=c(ptypes[2],ptypes[3]))
design = model.matrix(~pheno)
fit=lmFit(t(expres),design)
```

```
fit=eBayes(fit)
limma = topTable(fit, coef=2, adjust.method="BH", sort.by="p", number=1000)
limma = limma[,-which(names(limma)=="B")]
names(limma) = paste(names(limma),"limma",sep=".")
names(limma)[which(names(limma)=="ID.limma")] = "gene"
names(limma)[which(names(limma)=="adj.P.Val.limma")] = "FDR.limma"
names(limma)[which(names(limma)=="P.Value.limma")] = "p.limma"

###################
# Mann-Whitney
###################
wilcox = ddply(dedata,.(gene),.fun = function(x,colname) summarize(x,logFC.wilcox =
        mean(x[,"log2exp"][x[,colname]==ptypes["level2"]],na.rm=TRUE)-
        mean(x[,"log2exp"][x[,colname]==ptypes["level1"]],na.rm=TRUE),logFC.NAexcluded
        = mean(x[,"log2exp"][x[,colname]==ptypes["level2"] &
        x[,"wasNA"]!=TRUE],na.rm=TRUE)-
        mean(x[,"log2exp"][x[,colname]==ptypes["level1"] &
        x[,"wasNA"]!=TRUE],na.rm=TRUE),p.wilcox =
        wilcox.test(x[,"log2exp"][x[,colname]==ptypes["level2"]],x[,"log2exp"][x[,colname]==p
        types["level1"]])$p.value),colname=ptypes[1])
wilcox$FDR.wilcox = p.adjust(wilcox$p.wilcox,method="BH")

###################
# Welch t
###################
welch = ddply(dedata,.(gene),.fun = function(x,colname) summarize(x,p.welch =
        t.test(x[,"log2exp"][x[,colname]==ptypes["level2"]],x[,"log2exp"][x[,colname]==ptypes[
        "level1"]])$p.value),colname=ptypes[1])
welch$FDR.welch = p.adjust(welch$p.welch,method="BH")

###############
# Merge results
###############
res = merge(genedata,limma)
res = merge(res,wilcox)
res = merge(res,welch)

#########################################
# Pick a recommended p and list minimum p
#########################################
# recommended, and sort list based on this p value
res$p.recommended.type = "t.limma"
res$p.recommended.type[(res[,paste("failed",ptypes[2],sep=".")]/(res[,paste("failed",ptypes[2],se
        p=".")] + res[,ptypes[2]])) > 0.5 |
        (res[,paste("failed",ptypes[3],sep=".")]/(res[,paste("failed",ptypes[3],sep=".")] +
        res[,ptypes[3]])) > 0.5] = "wilcox"
```

```
res$p.recommended = res$p.limma
res$p.recommended[res$p.recommended.type=="wilcox"] =
res$p.wilcox[res$p.recommended.type=="wilcox"]
res$FDR.recommended = p.adjust(res$p.recommended,method="BH")
res = res[order(res$p.recommended),]
# minimum
res$p.minimum = apply(res[,c("p.limma","p.wilcox","p.welch")],1,function(x)
        min(x,na.rm=TRUE))
res$FDR.minimum = p.adjust(res$p.minimum,method="BH")

############
# Write data
############
write.csv(res,file=filename,quote=FALSE,row.names=FALSE,na="")

return(res)
}

#####################################
# END OF FUNCTION DEFINTION
#####################################

######################################################
# Use the function defined above to compute statistics: two-group
######################################################

# OPTIONAL: now, use the above function for low-burden mets vs. primary tumor cells
ptypes = c(phenotype="group",level1="T",level2="low")
resLowvsT = compute.results(dfcomb,ptypes,"Filename.csv","id","log2exp.conT")

# OPTIONAL: now, use the above function for high-burden mets vs. primary tumor cells
ptypes = c(phenotype="group",level1="T",level2="high")
resHighvsT = compute.results(dfcomb,ptypes,"Filename.csv","id","log2exp.conT")

# OPTIONAL: now, use the above function for high-burden mets vs. low-burden mets
ptypes = c(phenotype="group",level1="low",level2="high")
resHighvsLow =
        compute.results(dfcomb,ptypes,"Filename.csv","id","log2exp.conT")

####################################################
####################################################
# Three-group comparisons (HCI-001 vs. HCI-002 vs. HCI-010)
####################################################
####################################################

# define helper computation function first
```

```
#####################################
# START OF FUNCTION DEFINTION
#####################################

compute.results = function(dfavg,ptypes,filename,sample.column.name,expressionchoice) {

  dfavg$expression.choice = dfavg[,expressionchoice]
  dfavg$sample = dfavg[,sample.column.name]
  dfavg$log2exp = dfavg$expression.choice
  dfavg$grouping = dfavg[,ptypes[1]]

  dedata=dfavg

  # retain phenotypes of interest
  dedata = dedata[dedata[,ptypes["phenotype"]] %in%
          c(ptypes["level1"],ptypes["level2"],ptypes["level3"]),]

  # get rid of genes that don't have at least 3 non-NA values for each level of the phenotype
  dedata = ddply(dedata,.(gene),.fun = function(x,colname) transform(x,remove =
          sum(!is.na(x[,"log2exp"][x[,colname]==ptypes["level2"]]))<3 |
          sum(!is.na(x[,"log2exp"][x[,colname]==ptypes["level1"]]))<3 |
          sum(!is.na(x[,"log2exp"][x[,colname]==ptypes["level3"]]))<3),colname=ptypes[1])
  if (sum(dedata$remove)>0) dedata = dedata[-which(dedata$remove),]

  # record how many NA there are per condition for each gene
  genedata = data.frame(unique(dedata$gene), 0, 0, 0, 0, 0,0,0)
  names(genedata) =
          c("gene",ptypes[2],ptypes[3],ptypes[4],"failed.total",paste("failed",ptypes[2],sep="."),pas
          te("failed",ptypes[3],sep="."),paste("failed",ptypes[4],sep="."))
   expres = data.frame(dcast(dedata,sample~gene,value.var =
          "log2exp"),row.names=1,check.names=FALSE)
  pheno = dedata[,ptypes[1]][match(rownames(expres),dedata$sample)]
  pheno  = factor(pheno,levels=c(ptypes[2],ptypes[3],ptypes[4]))
  for (i in 1:dim(genedata)[1]) {
          genedata[,ptypes[2]][i] = sum(pheno==ptypes[2]) -
                  sum((as.numeric(is.na(t(expres)[match(genedata$gene[i],names(expres)),])) *
                  as.numeric(pheno)) == 1)
          genedata[,ptypes[3]][i] = sum(pheno==ptypes[3]) -
                  sum((as.numeric(is.na(t(expres)[match(genedata$gene[i],names(expres)),])) *
                  as.numeric(pheno)) == 2)
          genedata[,ptypes[4]][i] = sum(pheno==ptypes[4]) -
                  sum((as.numeric(is.na(t(expres)[match(genedata$gene[i],names(expres)),])) *
                  as.numeric(pheno)) == 3)
          genedata$failed.total[i] = sum(is.na(t(expres)[match(genedata$gene[i],names(expres)),]))
```

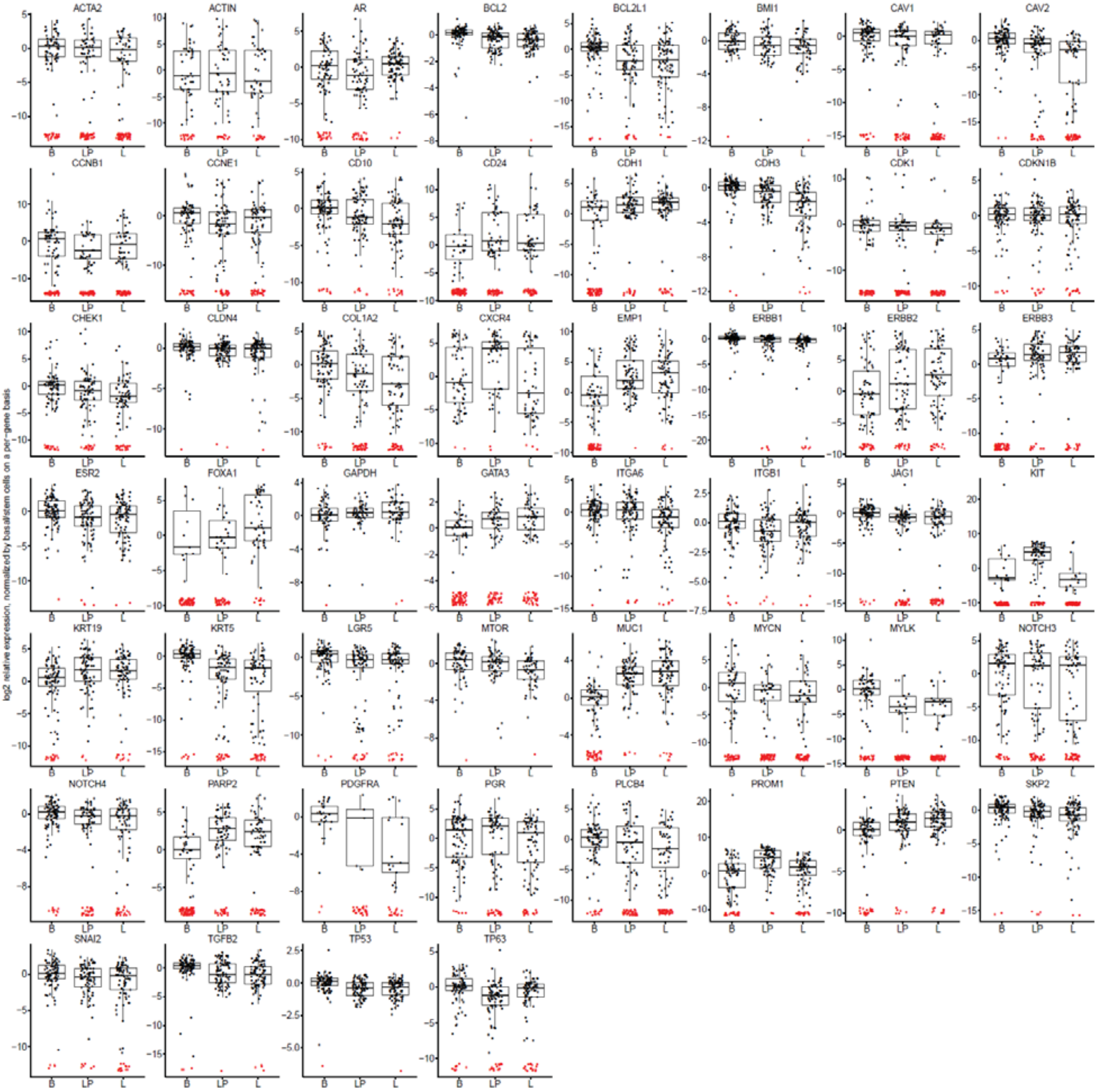
```
    genedata[,paste("failed",ptypes[2],sep=".")][i] =
            sum(is.na(t(expres)[match(genedata$gene[i],names(expres)),]) &
            pheno==ptypes[2])
    genedata[,paste("failed",ptypes[3],sep=".")][i] =
            sum(is.na(t(expres)[match(genedata$gene[i],names(expres)),]) &
            pheno==ptypes[3])
    genedata[,paste("failed",ptypes[4],sep=".")][i] =
            sum(is.na(t(expres)[match(genedata$gene[i],names(expres)),]) &
            pheno==ptypes[4])
}


# convert NA to the lowest expression value minus 0.01, on a per-gene basis, so all missing
# values are tied at the same very low value, with each value different for a different gene
dedata$wasNA = FALSE
dedata$wasNA[is.na(dedata$log2exp)] = TRUE
dedata = ddply(dedata,.(gene),transform,min = min(log2exp,na.rm=TRUE)-0.01)
dedata$log2exp[is.na(dedata$log2exp)] = dedata$min[is.na(dedata$log2exp)]

################
# With anova
################
an = ddply(dedata,.(gene),summarize,p.an=anova(lm(log2exp~factor(grouping)))[,5][1])
an$FDR.an = p.adjust(an$p.an,method="BH")

################################
# Kruskal-Wallis rank sum test
################################
kw = ddply(dedata,.(gene),summarize,p.kw=kruskal.test(log2exp,factor(grouping))$p.value)
kw$FDR.kw = p.adjust(kw$p.kw,method="BH")

################
# Merge results
################
res = merge(genedata,an)
res = merge(res,kw)

############################################
# Pick a recommended p and list minimum p
############################################
# recommended, and sort list based on this p value
res$p.recommended.type = "p.an"
res$p.recommended.type[(res[,paste("failed",ptypes[2],sep=".")]/(res[,paste("failed",ptypes[2],s
      ep=".")] + res[,ptypes[2]])) > 0.5 |
      (res[,paste("failed",ptypes[3],sep=".")]/(res[,paste("failed",ptypes[3],sep=".")] +
      res[,ptypes[3]])) > 0.5 |
          (res[,paste("failed",ptypes[4],sep=".")]/(res[,paste("failed",ptypes[4],sep=".")] +
          res[,ptypes[4]])) > 0.5] = "kw"
  res$p.recommended = res$p.an
  res$p.recommended[res$p.recommended.type=="kw"] =
          res$p.kw[res$p.recommended.type=="kw"]
  res$FDR.recommended = p.adjust(res$p.recommended,method="BH")
  res = res[order(res$p.recommended),]
  # minimum
  res$p.minimum = apply(res[,c("p.an","p.kw")],1,function(x) min(x,na.rm=TRUE))
  res$FDR.minimum = p.adjust(res$p.minimum,method="BH")

  ############
  # Write data
  ############
  write.csv(res,file=filename,quote=FALSE,row.names=FALSE,na="")

  return(res)
}

#####################################
# END OF FUNCTION DEFINTION
#####################################

######################################################
# Use the function defined above to compute statistics: three-group
######################################################

# OPTIONAL: generate groups to manually designate colors by ID (specific cells)
# For grouping based on PDX model: (Note: In the study, this was only done on lung mets)
dfcomb$group = "nogroup"
dfcomb$group[dfcomb$id %in% c("LU1 1",”LU2 1”,”LU3 1”,…)] = “HCI-001”
dfcomb$group[dfcomb$id %in% c("LU4 1",”LU5 1”,”LU6 1”,…)] = “HCI-002”
dfcomb$group[dfcomb$id %in% c("LU7 1",”LU8 1”,”LU9 1”,…)] = “HCI-010”
dfcomb = dfcomb[dfcomb$group!="nogroup",]

# HCI-001 vs. HCI-002 vs. HCI-010
ptypes = c(phenotype="group",level1="HCI-001",level2="HCI-002",level3="HCI-010")
res1vs2vs10 = compute.results(dfcomb,ptypes, “Filename.csv","id","log2exp.conT")

######################################
######################################
# Five-group comparisons (5 metastatic tissues)
# with post-hoc pairwise comparisons
######################################
######################################
```

```
# define helper computation function first

#######################################
# START OF FUNCTION DEFINTION
#######################################

compute.results = function(dfavg,ptypes,filename,sample.column.name,expressionchoice) {

  dfavg$expression.choice = dfavg[,expressionchoice]
  dfavg$sample = dfavg[,sample.column.name]
  dfavg$log2exp = dfavg$expression.choice

  dedata=dfavg

  # retain phenotypes of interest
  dedata = dedata[dedata[,ptypes["phenotype"]] %in%
          c(ptypes["level1"],ptypes["level2"],ptypes["level3"],ptypes["level4"],ptypes["level5"]),]

  # remove genes that failed in all samples
  dedata = ddply(dedata,.(gene),transform,remove = all(is.na(expression.choice)))
  if (sum(dedata$remove)>0) dedata = dedata[-which(dedata$remove),]

  # remove genes that only have one tissue type
  dedata = ddply(dedata,.(gene),transform,remove=length(unique(tissue))==1)
  if (sum(dedata$remove)>0) dedata = dedata[-which(dedata$remove),]

  # record how many NA there are per condition for each gene
  genedata = data.frame(unique(dedata$gene), 0, 0, 0, 0, 0, 0, 0, 0,0, 0, 0)
  names(genedata) =
          c("gene",ptypes[2],ptypes[3],ptypes[4],ptypes[5],ptypes[6],"failed.total",paste("failed",pt
          ypes[2],sep="."),paste("failed",ptypes[3],sep="."),paste("failed",ptypes[4],sep="."),paste(
          "failed",ptypes[5],sep="."),paste("failed",ptypes[6],sep="."))
  expres = data.frame(dcast(dedata,sample~gene,value.var =
          "log2exp"),row.names=1,check.names=FALSE)
  pheno = dedata[,ptypes[1]][match(rownames(expres),dedata$sample)]
  pheno  = factor(pheno,levels=c(ptypes[2],ptypes[3],ptypes[4],ptypes[5],ptypes[6]))
  for (i in 1:dim(genedata)[1]) {
            genedata[,ptypes[2]][i] = sum(pheno==ptypes[2]) -
                    sum((as.numeric(is.na(t(expres)[match(genedata$gene[i],names(expres)),])) *
                    as.numeric(pheno)) == 1)
            genedata[,ptypes[3]][i] = sum(pheno==ptypes[3]) -
                    sum((as.numeric(is.na(t(expres)[match(genedata$gene[i],names(expres)),])) *
                    as.numeric(pheno)) == 2)
            genedata[,ptypes[4]][i] = sum(pheno==ptypes[4]) -
                    sum((as.numeric(is.na(t(expres)[match(genedata$gene[i],names(expres)),])) *
                    as.numeric(pheno)) == 3)
```

```
    genedata[,ptypes[5]][i] = sum(pheno==ptypes[5]) -
        sum((as.numeric(is.na(t(expres)[match(genedata$gene[i],names(expres)),])) *
        as.numeric(pheno)) == 4)
    genedata[,ptypes[6]][i] = sum(pheno==ptypes[6]) -
        sum((as.numeric(is.na(t(expres)[match(genedata$gene[i],names(expres)),])) *
        as.numeric(pheno)) == 5)
    genedata$failed.total[i] =
        sum(is.na(t(expres)[match(genedata$gene[i],names(expres)),]))
    genedata[,paste("failed",ptypes[2],sep=".")][i] =
        sum(is.na(t(expres)[match(genedata$gene[i],names(expres)),]) &
        pheno==ptypes[2])
    genedata[,paste("failed",ptypes[3],sep=".")][i] =
        sum(is.na(t(expres)[match(genedata$gene[i],names(expres)),]) &
        pheno==ptypes[3])
    genedata[,paste("failed",ptypes[4],sep=".")][i] =
        sum(is.na(t(expres)[match(genedata$gene[i],names(expres)),]) &
        pheno==ptypes[4])
    genedata[,paste("failed",ptypes[5],sep=".")][i] =
        sum(is.na(t(expres)[match(genedata$gene[i],names(expres)),]) &
        pheno==ptypes[5])
    genedata[,paste("failed",ptypes[6],sep=".")][i] =
        sum(is.na(t(expres)[match(genedata$gene[i],names(expres)),]) &
        pheno==ptypes[6])
}

# convert NA to the lowest expression value minus 0.01, on a per-gene basis, so all missing
# values are tied at the same very low value, with each value different for a different gene
dedata$wasNA = FALSE
dedata$wasNA[is.na(dedata$log2exp)] = TRUE
dedata = ddply(dedata,.(gene),transform,min = min(log2exp,na.rm=TRUE)-0.01)
dedata$log2exp[is.na(dedata$log2exp)] = dedata$min[is.na(dedata$log2exp)]

################
# With anova
################
an = ddply(dedata,.(gene),summarize,p.an=anova(lm(log2exp~factor(tissue)))[,5][1])
an$FDR.an = p.adjust(an$p.an,method="BH")

################################
# Kruskal-Wallis rank sum test
################################
kw = ddply(dedata,.(gene),summarize,p.kw=kruskal.test(log2exp,factor(tissue))$p.value)
kw$FDR.kw = p.adjust(kw$p.kw,method="BH")
```

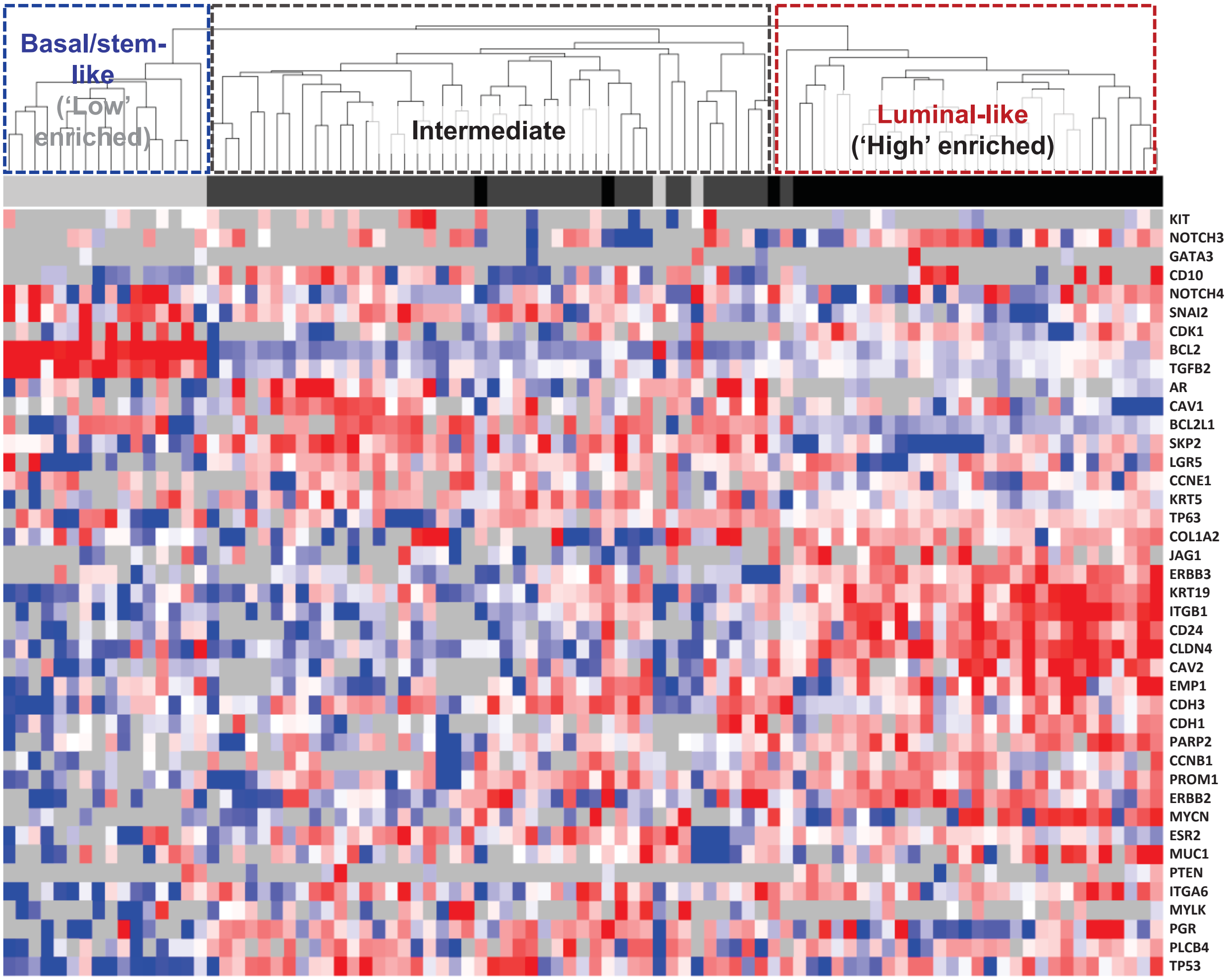
```
################
# Merge results
################
res = merge(genedata,an)
res = merge(res,kw)

############################################
# Pick a recommended p and list minimum p
############################################
# recommended, and sort list based on this p value
res$p.recommended.type = "p.an"
res$p.recommended.type[(res[,paste("failed",ptypes[2],sep=".")]/(res[,paste("failed",ptypes[2],s
        ep=".")] + res[,ptypes[2]])) > 0.5 |
        (res[,paste("failed",ptypes[3],sep=".")]/(res[,paste("failed",ptypes[3],sep=".")] +
        res[,ptypes[3]])) > 0.5 |
        (res[,paste("failed",ptypes[4],sep=".")]/(res[,paste("failed",ptypes[4],sep=".")] +
        res[,ptypes[4]])) > 0.5 |
        (res[,paste("failed",ptypes[5],sep=".")]/(res[,paste("failed",ptypes[5],sep=".")] +
        res[,ptypes[5]])) > 0.5 |
        (res[,paste("failed",ptypes[6],sep=".")]/(res[,paste("failed",ptypes[6],sep=".")] +
        res[,ptypes[6]])) > 0.5] = "kw"
res$p.recommended = res$p.an
res$p.recommended[res$p.recommended.type=="kw"] =
        res$p.kw[res$p.recommended.type=="kw"]
res$FDR.recommended = p.adjust(res$p.recommended,method="BH")
res = res[order(res$p.recommended),]
# minimum
res$p.minimum = apply(res[,c("p.an","p.kw")],1,function(x) min(x,na.rm=TRUE))
res$FDR.minimum = p.adjust(res$p.minimum,method="BH")

############################################
# Post-hoc multiple pair-wise comparisons
############################################

# subset data to retain genes that have FDR < 0.05 for the comparison across all groups
ress = res[res$FDR.recommended < 0.05,]

# only run the post-hoc code if any FDR.recommended is < 0.05
if (dim(ress)[1]!=0) {

# for each gene, if anova was recommended above, then do all post-hoc pairwise comparisons
# using TukeyHSD; if kruskal-wallis was recommended, use posthoc.kruskal.nemenyi.test
# with Chisquare method given ties in failures

# first set up data frame to populate with the post-hoc p value results from tukey
```

```
  res.posthoc = data.frame(gene="gene",rec.group.test="rec.p",pvalues =
        "pvalues",stringsAsFactors=FALSE)

# second set up a data frame to populate with the post-hoc logFC results
        res.posthoc.logFC = data.frame(gene="gene",rec.group.test="rec.p",logFC =
        "logFC",stringsAsFactors=FALSE)

# now, loop through for each gene
for (i in 1:dim(ress)[1]) {

        # if anova was recommended, run this to populate p value data frame
        if (ress$p.recommended.type[i]=="p.an") {
            cur.post.hoc =
                TukeyHSD(aov(log2exp~tissue,data=dedata[dedata$gene==ress$gene[i],]))
            cur.post.hoc = cur.post.hoc$tissue
            cur.post.hoc = paste(rownames(cur.post.hoc),cur.post.hoc[,"p adj"])
            cur.post.hoc = data.frame(gene=as.character(ress$gene[i]),
                rec.group.test=as.character(ress$p.recommended.type[i]),paste(cur.post.hoc,colla
                pse=" "),stringsAsFactors=FALSE)
            res.posthoc[i,] = cur.post.hoc

            }   # end of "if anova"

        # if kruskal-wallis was recommended, run this to populate p value data frame
        if (ress$p.recommended.type[i]=="kw") {
            cur.post.hoc =
                posthoc.kruskal.nemenyi.test(x=dedata[dedata$gene==ress$gene[i],"log2exp"],
                g=dedata[dedata$gene==ress$gene[i],"tissue"], method="Chisquare")
            cur.post.hoc = melt(cur.post.hoc$p.value)
            cur.post.hoc = cur.post.hoc[!is.na(cur.post.hoc$value),]
            cur.post.hoc$name = paste(cur.post.hoc$Var1,cur.post.hoc$Var2)
            cur.post.hoc = paste(cur.post.hoc$name,cur.post.hoc$value)
            cur.post.hoc = data.frame(gene=as.character(ress$gene[i]),
                rec.group.test=as.character(ress$p.recommended.type[i]),paste(cur.post.hoc,colla
                pse=" "),stringsAsFactors=FALSE)
            res.posthoc[i,] = cur.post.hoc

            }   # end of "if krukal-wallis"

        # populate logFC data frame
            cur.post.hoc =
                TukeyHSD(aov(log2exp~tissue,data=dedata[dedata$gene==ress$gene[i],]))
            cur.post.hoc = cur.post.hoc$tissue
            cur.post.hoc = paste(rownames(cur.post.hoc),cur.post.hoc[,"diff"])
            cur.post.hoc.logFC = data.frame(gene=as.character(ress$gene[i]),
               rec.group.test=as.character(ress$p.recommended.type[i]),paste(cur.post.hoc,colla
               pse=" "),stringsAsFactors=FALSE)
            res.posthoc.logFC[i,] = cur.post.hoc.logFC

            }   # end of for loop

            }   # end of if any FDR.recommend < 0.05


  #############
  # Write data
  #############
  write.csv(res,file=filename,quote=FALSE,row.names=FALSE,na="")

  if (dim(ress)[1]!=0) {
  posthoc.filename = sub(".csv","",filename,fixed=TRUE)
  posthoc.filename = paste(posthoc.filename,"-posthoc-pairwise.csv",sep="")
  posthoc.filename.logFC = sub("pairwise","pairwise-logFC",posthoc.filename,fixed=TRUE)
  write.csv(res.posthoc,file=posthoc.filename,quote=FALSE,row.names=FALSE,na="")
   write.csv(res.posthoc.logFC,file=posthoc.filename.logFC,quote=FALSE,row.names=FALSE,n
        a="")

  }

  return(res)
}

###########################################################
# Use the function defined above to compute statistics: five-group
###########################################################

# Now, use the above function for LU vs. LN vs. BM vs. PB vs. BR
ptypes =
        c(phenotype="tissue",level1="LU",level2="LN",level3="BM",level4="PB",level5="BR"
        )
restissues = compute.results(dfcomb,ptypes, “Filename.csv","id","log2exp.conT")
```

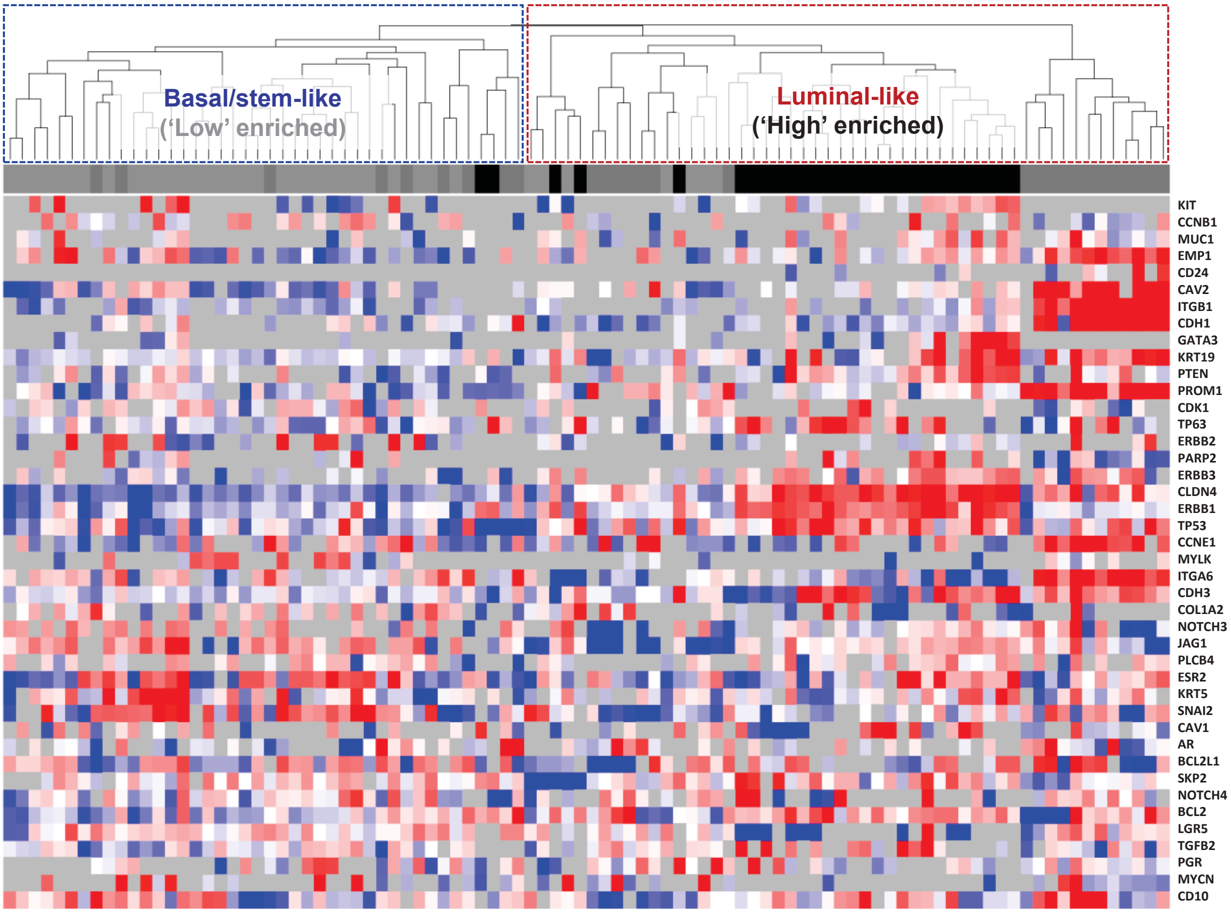


Supplementary Data 1: Gene expression in individual basal/stem, luminal, and luminal progenitor cells. Box plots show expression levels for each gene from the 49-gene differentiation signature in individual cells. P-values and fold change for each gene are shown in **Supplementary Table 2**. Black dots, single cells; red dots, single cells with no expression. B, basal; LP, luminal progenitor; L, luminal.

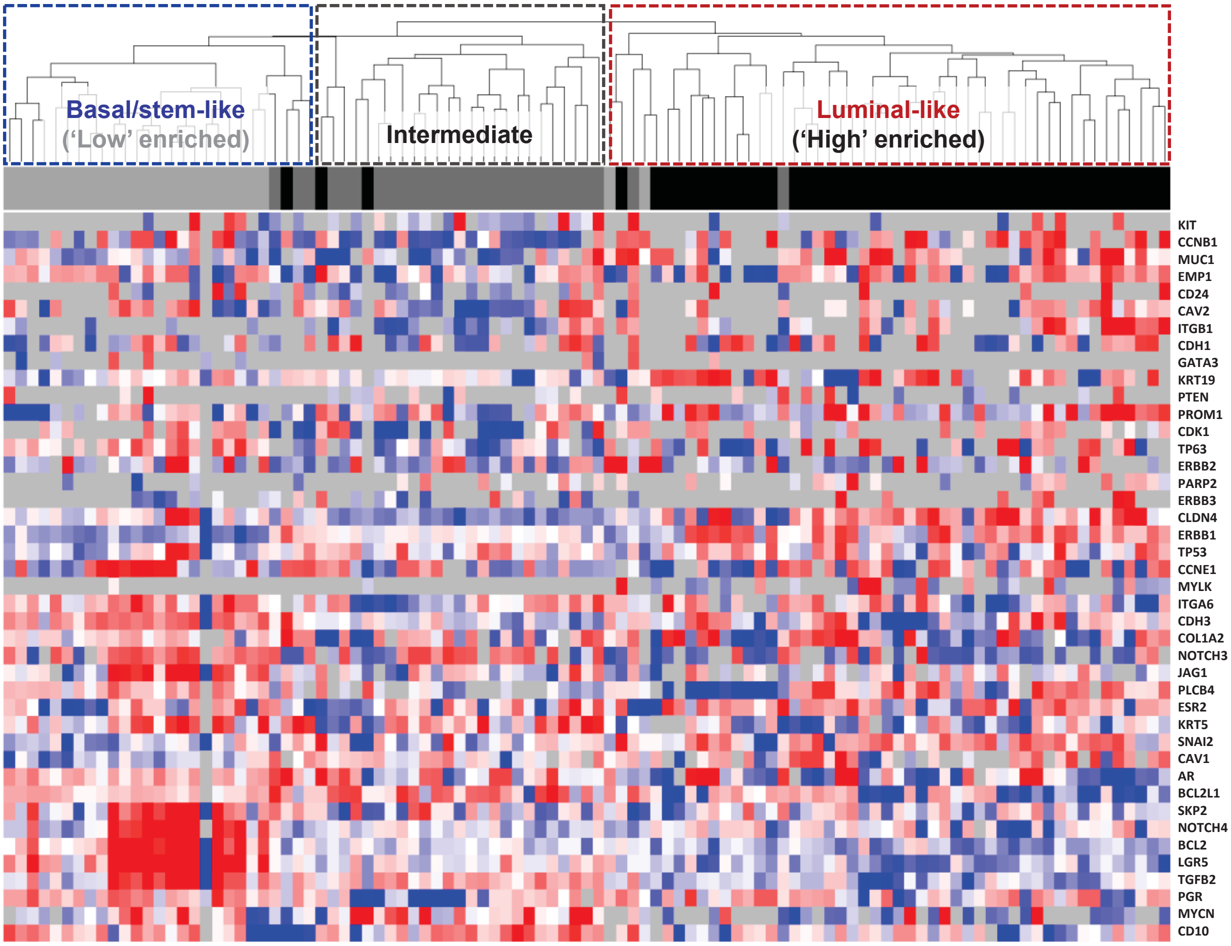
HCI-001


Basal/stem-like
('Low' enriched)
Intermediate
Luminal-like
('High' enriched)
KIT
NOTCH3
GATA3
CD10
NOTCH4
SNAI2
CDK1
BCL2
TGFB2
AR
CAV1
BCL2L1
SKP2
LGR5
CCNE1
KRT5
TP63
COL1A2
JAG1
ERBB3
KRT19
ITGB1
CD24
CLDN4
CAV2
EMP1
CDH3
CDH1
PARP2
CCNB1
PROM1
ERBB2
MYCN
ESR2
MUC1
PTEN
ITGA6
MYLK
PGR
PLCB4
TP53

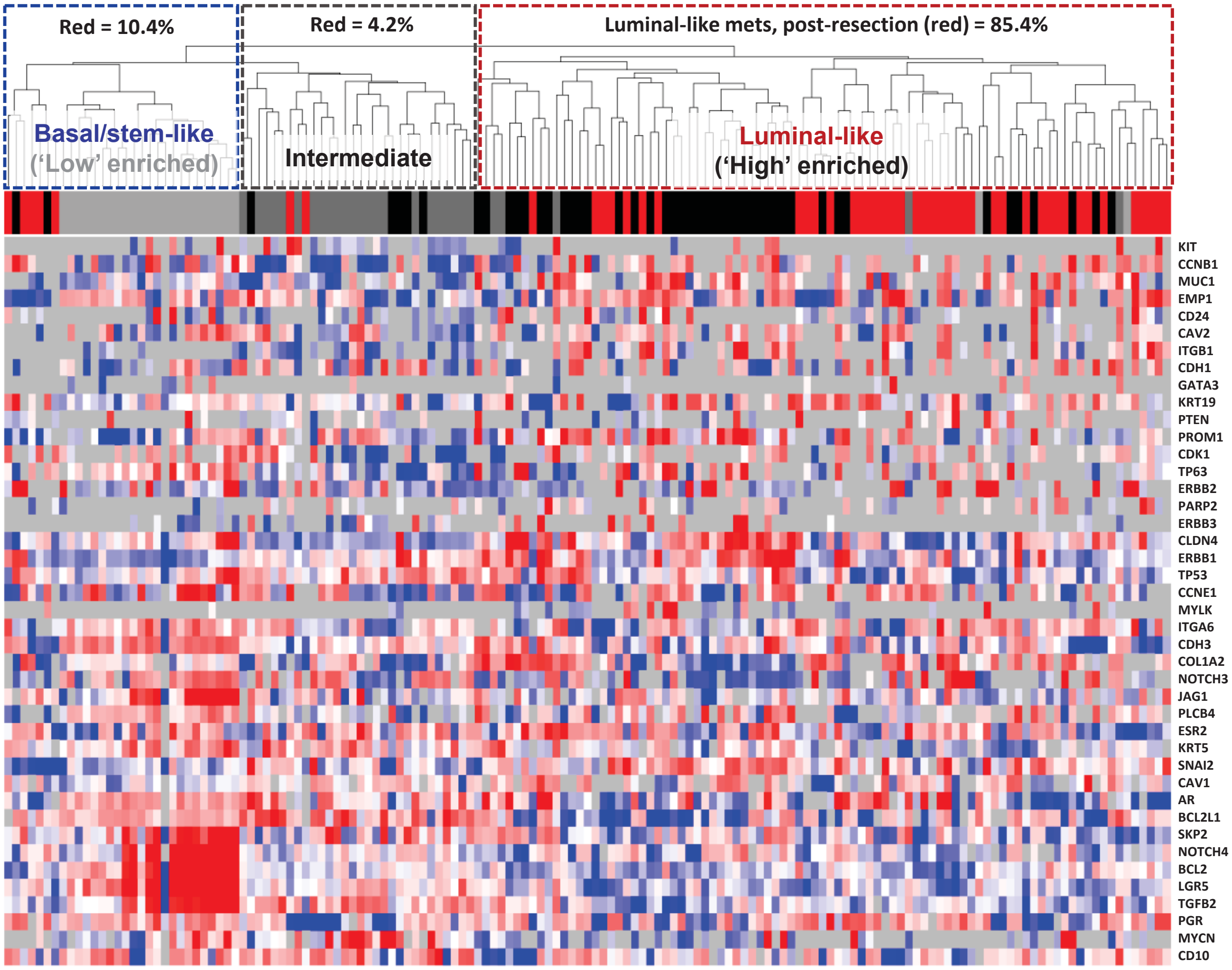
HCI-002


Basal/stem-like
('Low' enriched)
Luminal-like
('High' enriched)
KIT
CCNB1
MUC1
EMP1
CD24
CAV2
ITGB1
CDH1
GATA3
KRT19
PTEN
PROM1
CDK1
TP63
ERBB2
PARP2
ERBB3
CLDN4
ERBB1
TP53
CCNE1
MYLK
ITGA6
CDH3
COL1A2
NOTCH3
JAG1
PLCB4
ESR2
KRT5
SNAI2
CAV1
AR
BCL2L1
SKP2
NOTCH4
BCL2
LGR5
TGFB2
PGR
MYCN
CD10

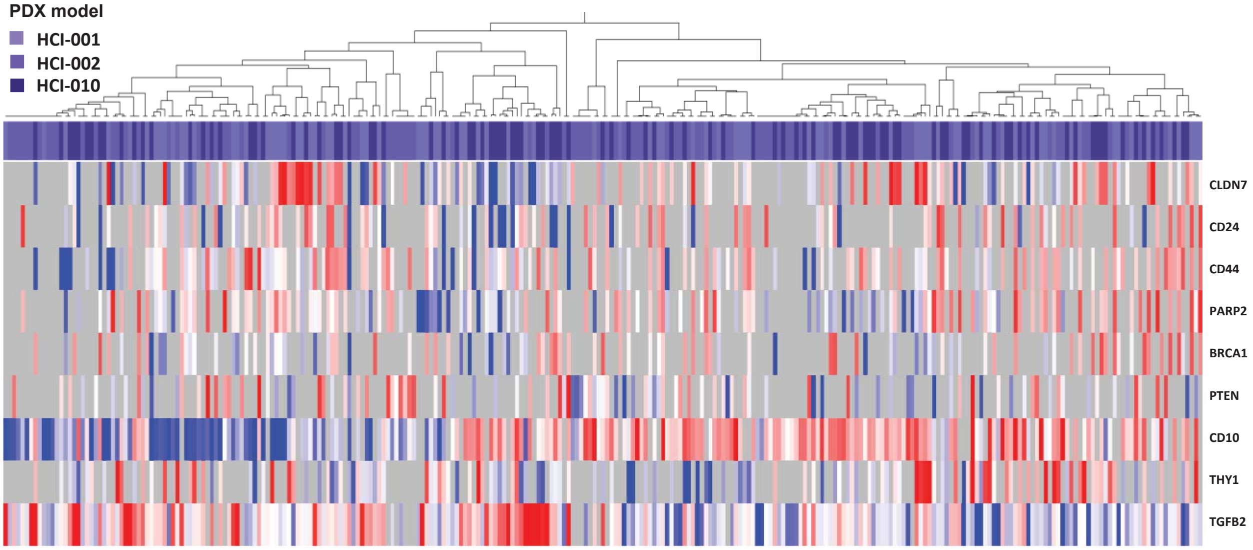
HCI-010


Basal/stem-like
('Low' enriched)
Intermediate
Luminal-like
('High' enriched)
KIT
CCNB1
MUC1
EMP1
CD24
CAV2
ITGB1
CDH1
GATA3
KRT19
PTEN
PROM1
CDK1
TP63
ERBB2
PARP2
ERBB3
CLDN4
ERBB1
TP53
CCNE1
MYLK
ITGA6
CDH3
COL1A2
NOTCH3
JAG1
PLCB4
ESR2
KRT5
SNAI2
CAV1
AR
BCL2L1
SKP2
NOTCH4
BCL2
LGR5
TGFB2
PGR
MYCN
CD10

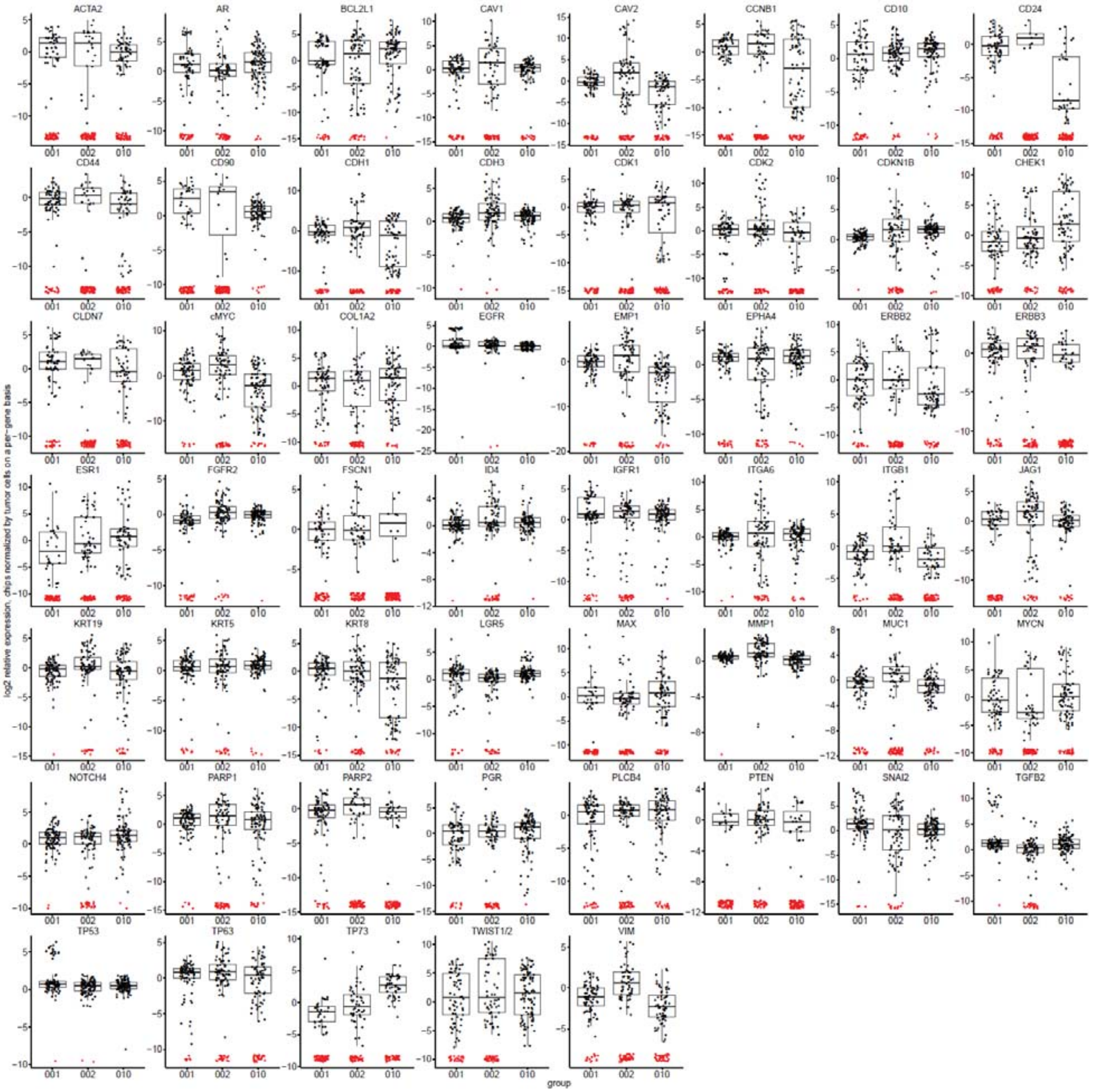
HCI-010 + #453 (Resected animal)



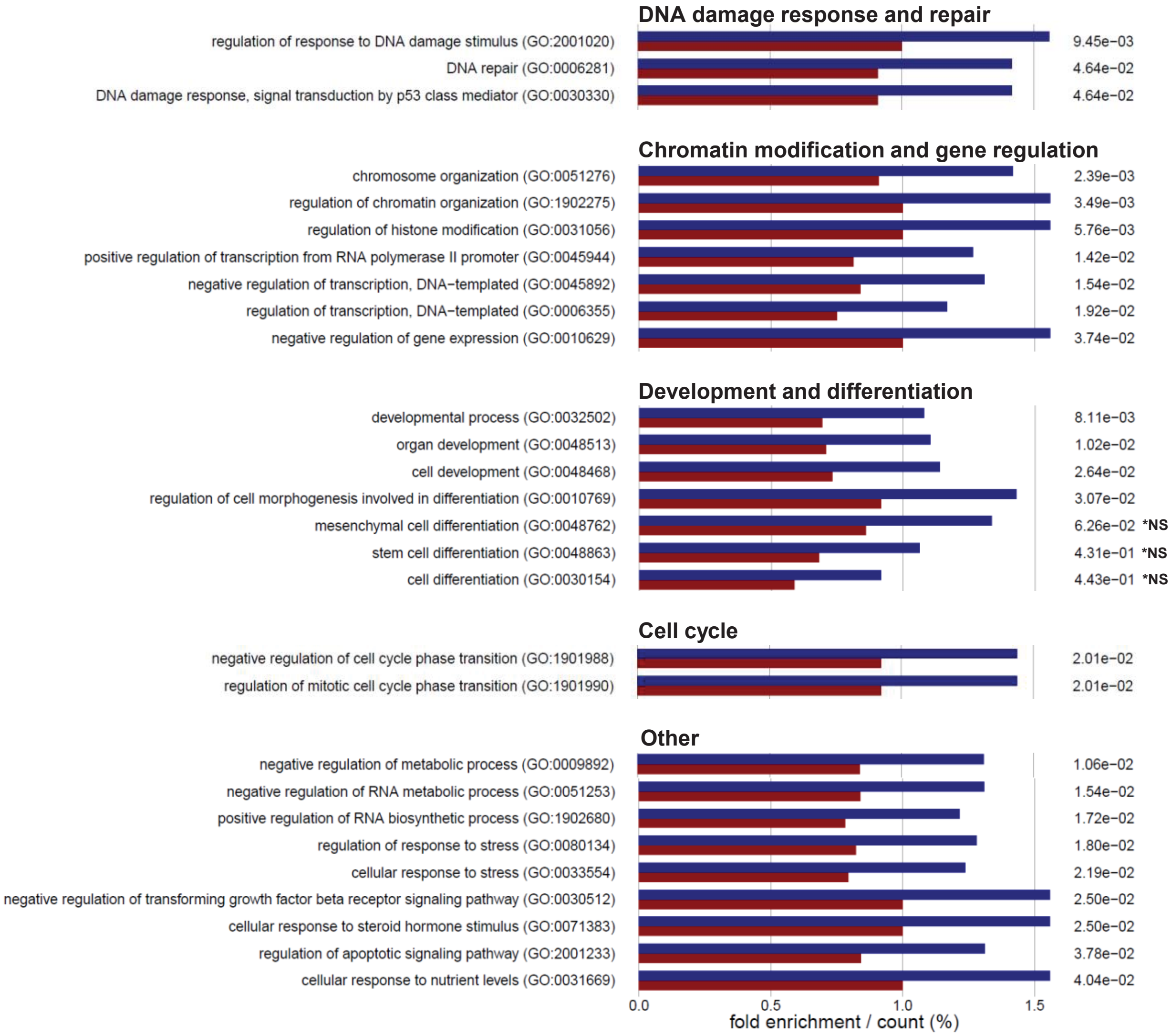
Clustering by PDX model



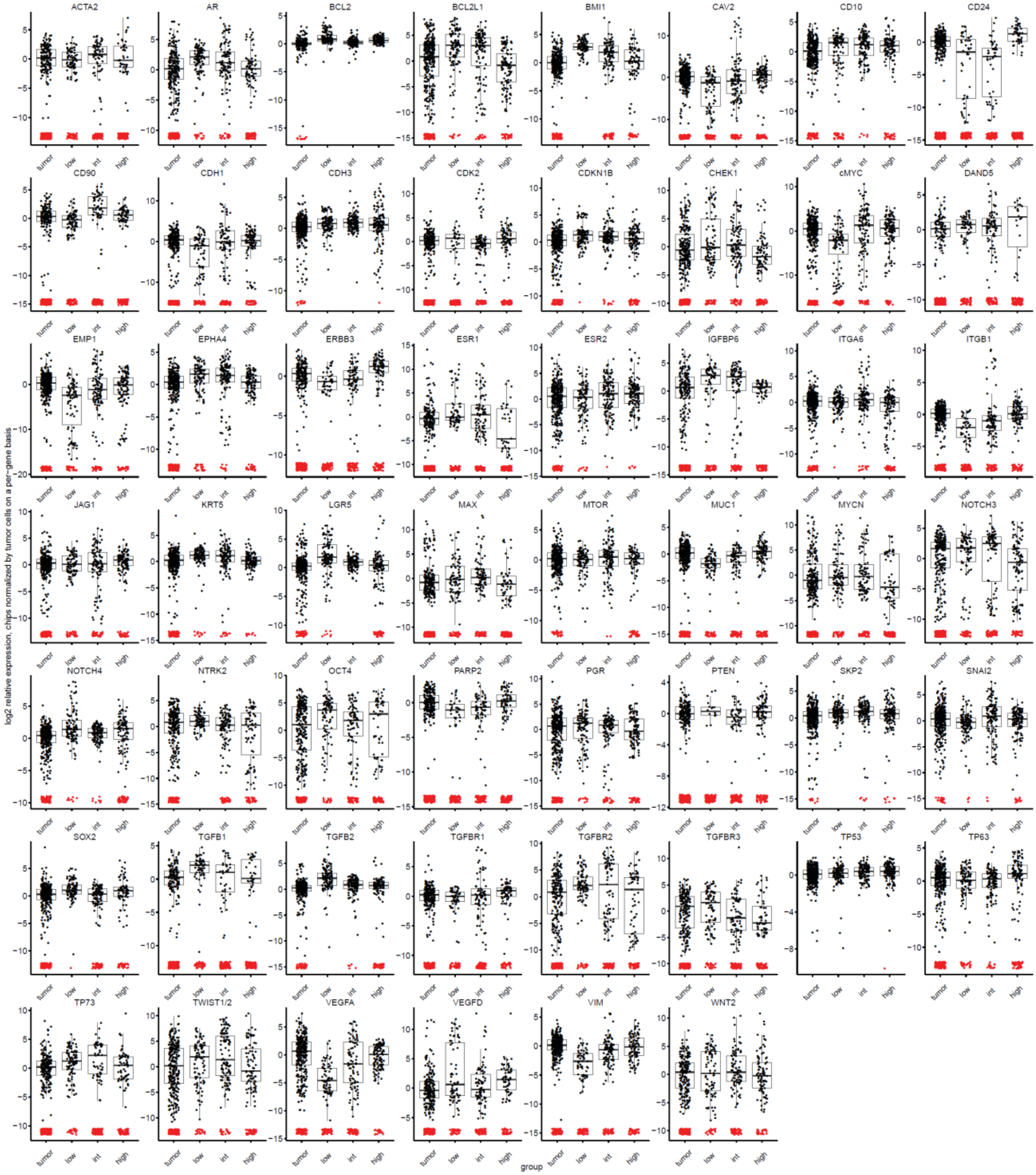
Supplementary Data 2: Full heatmaps showing unsupervised hierarchical clustering analyses shown in Extended Data Fig. 4



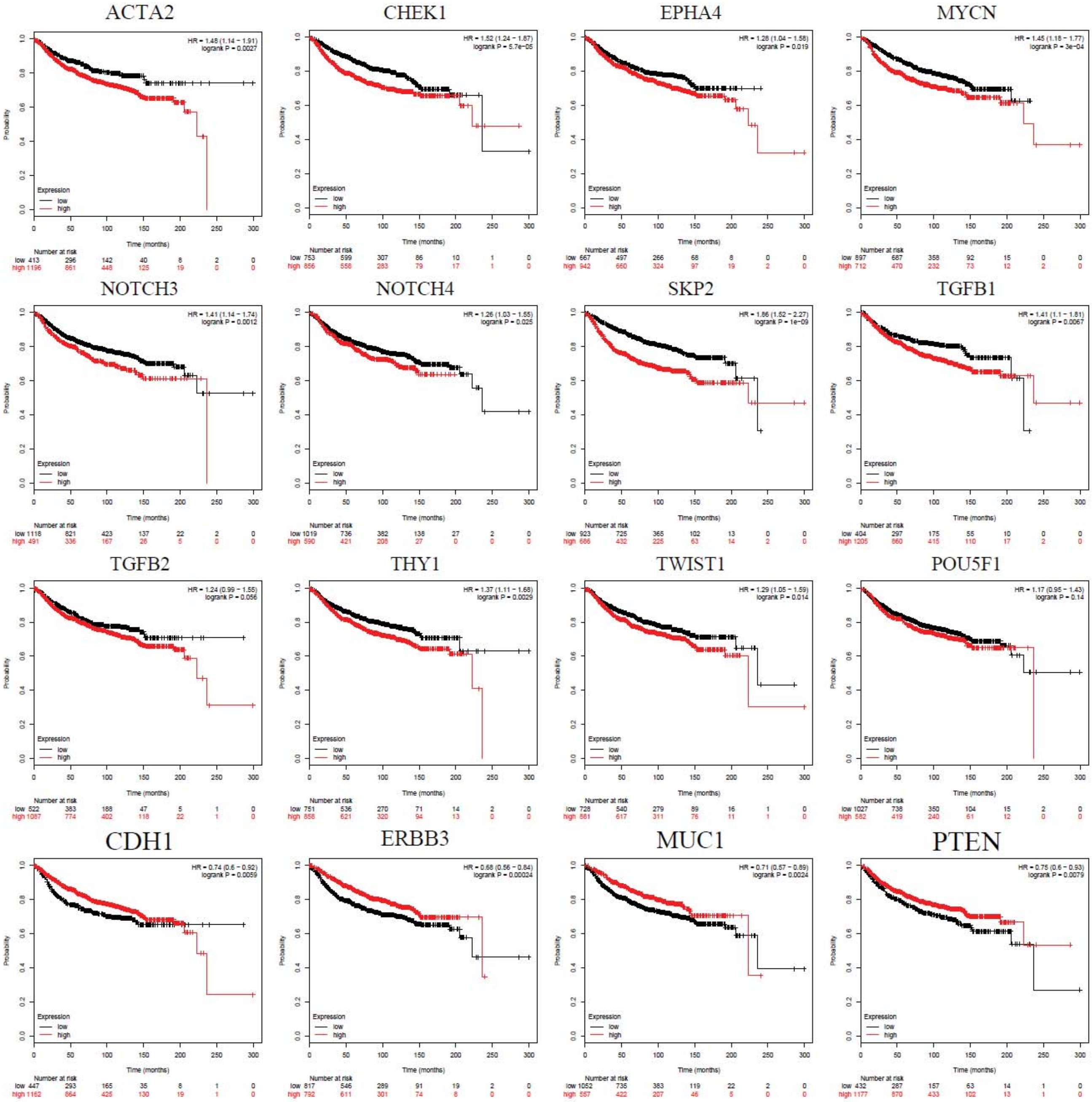
Supplementary Data 3: Gene expression in individual lung metastatic cells from each of the three PDX models (HCI-001, HCI-002, and HCI-010). Box plots show expression levels for each of the 53 genes differentially expressed between the models by Anova. Black dots, single cells; red dots, single cells with no expression.



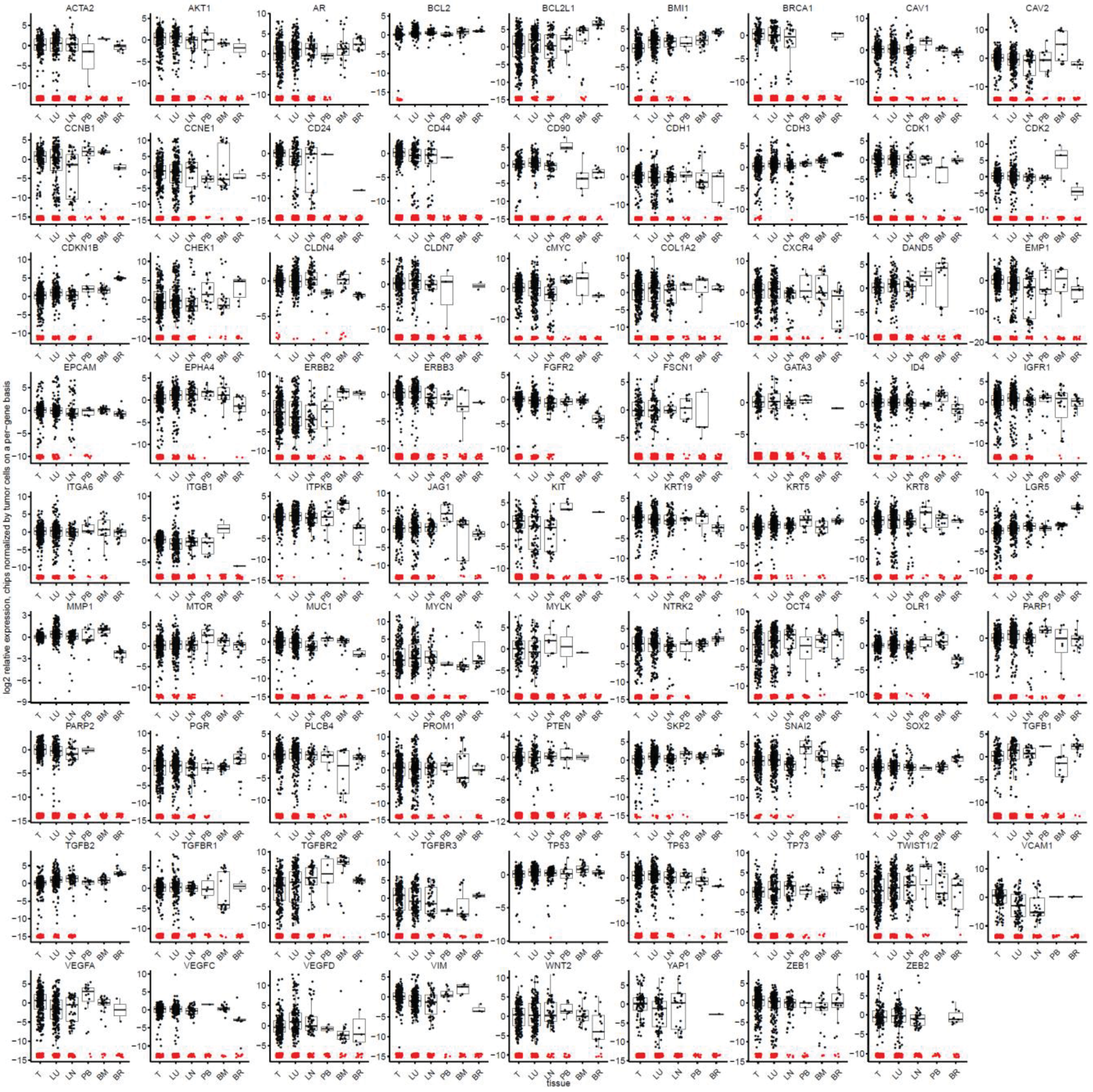
Supplementary Data 4: Gene ontology (GO) enrichment analysis reveals several biological processes are differentially regulated in low-burden metastatic cells. GO enrichment analysis was performed to identify pathways that were more represented in the set of significantly differentially expressed genes than would be expected by chance alone. Pathways involving DNA damage response and repair, chromatic modification and gene regulation, development and differentiation, and cell cycle were identified. p-values are shown at the right of each bar. *NS, not significant.



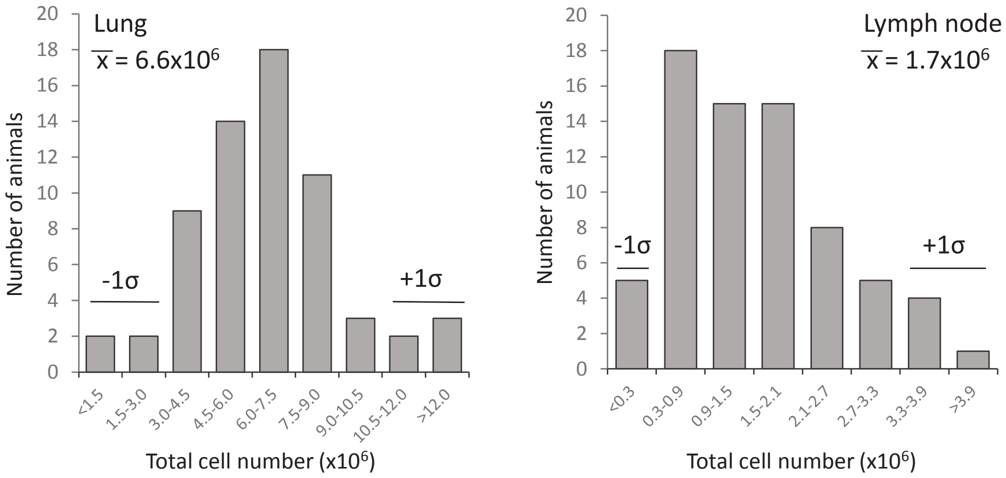
Supplementary Data 5: Gene expression in individual cells from tissues with low, intermediate, or high metastatic burden. Box plots show expression levels for each differentially expressed gene ($p<0.05$) in individual cells. P-values and fold change for each gene are shown in **Extended Data Table 2**. Black dots, single cells; red dots, single cells with no expression.



Supplementary Data 6: Kaplan-Meier analysis shows the prognostic relevance of 16 genes characteristic of low-burden metastatic cells. The prognostic value of each of the 55 genes characteristic of low-burden metastatic cells **(Extended Data Table 2)** was determined by Kaplan-Meier analysis using Km-plotter online software.[29] Km plots are shown as probability of distant metastasis-free survival (DMFS). 16 of 55 genes were found to be prognostic: High *ACTA2, CHEK1, EPHA4, MYCN, NOTCH3, NOTCH4, SKP2, TGFB1, TGFB2, THY1* and *TWIST1* expression was associated with increased risk of distant metastasis, and increased *CDH1, ERBB3, MUC1 and PTEN* expression was associated with reduced risk of metastasis.



Supplementary Data 7: Gene expression in individual primary tumor and metastatic cells from the lung, lymph node, peripheral blood, bone marrow, and brain. Box plots show all genes (80) that were significantly differentially expressed ($p<0.05$, ANOVA) between at least one pair of tissues. The p-value and fold change for each gene and tissue pair and are shown in **Supplementary Table 3**. Each black dot represents expression in a single cell, and red dots represent non-expressing cells. LU, lung; LN, lymph node; BM, bone marrow; PB, peripheral blood (CTC); BR, brain.



Supplementary Data 8: Histogram plots show the distribution of cells yielded following tissue dissociation of lungs and lymph nodes from PDX animals. For consistency, any tissue digests which yielded an abnormal number of cells (>1 standard deviation from the mean) was excluded from the study. This allowed for comparison of the number of metastatic cells in different animals with reasonable consistency.

NCBI Gene Symbol	Description	GenBank Accession	Rationale/Function	Forward	Reverse
ACTA2	actin, alpha 2, smooth muscle	NM_001613	Basal/Myoep/Stem	GTGTTGCCCCTGAAGAGCAT	GCTGGGACATTGAAAGTCTCA
AKT1	v-akt murine thymoma viral oncogene homolog 1	NM_001014431	TNBRCA	AGCGACGTGGCTATTGTGAAG	GCCATCATTCTTGAGGAGGAAGT
AR	androgen receptor	NM_000044	TNBRCA	GACGACCAGATGGCTGTCATT	GGGCGAAGTAGAGCATCCT
ASPSCR1	alveolar soft part sarcoma chromosome region, candidate 1	NM_001251888	Luminal	GAGCGTGCTCGACCTTTCTC	AGCGATGCGAACCATGTTCTC
BCL2	B-cell CLL/lymphoma 2	NM_000657	Anti-apoptosis	GGTGGGGTCATGTGTGTGG	CGGTTCAGGTACTCAGTCATCC
BCL2L1	BCL2-like 1	NM_001191	Anti-apoptosis	GAGCTGGTGGTTGACTTTCTC	TCCATCTCCGATTCAGTCCCT
BMI1	B lymphoma Mo-MLV insertion region	NM_005180	Stem cell	CCACCTGATGTGTGTGCTTTG	TTCAGTAGTGGTCTGGTCTTGT
BRCA1	breast cancer 1, early onset	NM_007297	TNBRCA	GAAACCGTGCCAAAAGACTTC	CCAAGGTTAGAGAGTTGGACAC
BRCA2	breast cancer 2, early onset	NM_000059	TNBRCA	TGCCTGAAAACCAGATGACTATC	AGGCCAGCAAACTTCCGTTTA
CASP3	caspase 3, apoptosis-related cysteine peptidase	NM_004346	TNBRCA	AGAGGGGATCGTTGTAGAAGTC	ACAGTCCAGTTCTGTACCACG
CAV1	caveolin 1	NM_001172895	TNBRCA; EMT	CATCCCGATGGCACTCATCTG	TGCACTGAATCTCAATCAGGAAG
CAV2	caveolin 2	NM_001206748	TNBRCA	AAGACCTGCCTAATGGTTCTGC	TCGTACACAATGGAGCAATGAT
CCNB1	cyclin B1	NM_031966	Cell cycle	AATAAGGCGAAGATCAACATGGC	TTTGTTACCAATGTCCCCAAGAG
CCND1	cyclin D1	NM_053056	Cell cycle	GCTGCGAAGTGGAAACCATC	CCTCCTTCTGCACACATTTGAA
CCNE1	cyclin E1	NM_001238	Cell cycle	GCCAGCCTTGGGACAATAATG	CTTGCACGTTGAGTTTGGGT
MME	metallo-endopeptidase/CD10	NM_000902	Cell surface marker	GATCGCACTCTATGCAACCTAC	TGTTTTGGATCAGTCGAGCAG
CD24	Heat stable antigen CD24	NM_013230	Cell surface marker	CTCCTACCCACGCAGATTTATTC	AGAGTGAGACCACGAAGAGAC
CD44	CD44 molecule (Indian blood group)	NM_001001392	Cell surface marker	CTGCCGCTTTGCAGGTGTA	CATTGTGGGCAAGGTGCTATT
THY1	Thymocyte differentiation antigen 1, CD90	NM_006288	Cell surface marker	TCACCCATCCAGTACGAGTTC	GGAGCGGTATGTGTGCTCAG
CDH1	cadherin 1, type 1	NM_004360	Epithelial	ATTTTTCCCTCGACACCCGAT	TCCCAGGCGTAGACCAAGA
CDH3	cadherin 3, type 1, P-cadherin (placental)	NM_001793	TNBRCA; EMT	ATCATCGTGACCGACCAGAAT	GACTCCCTCTAAGACACTCCC
CDK1	cyclin-dependent kinase 1	NM_001786	Cell cycle	GGATGTGCTTATGCAGGATTCC	CATGTACTGACCAGGAGGGATAG
CDK2	cyclin-dependent kinase 2	NM_001798	Cell cycle	CCAGGAGTTACTTCTATGCCTGA	TTCATCCAGGGGAGGTACAAC
CDKN1B	cyclin-dependent kinase inhibitor 1B (p27, Kip1)	NM_004064	Cell cycle	TAATTGGGGCTCCGGCTAACT	TGCAGGTCGCTTCCTTATTCC
CDKN2A	cyclin-dependent kinase inhibitor 2A (p16)	NM_058197	Cell cycle	GTAACTATTCGGTGCGTTGGG	ATGGAGCCTTCGGCTGACT
CHEK1	checkpoint kinase 1	NM_001274	Cell cycle	ATATGAAGCGTGCCGTAGACT	TGCCTATGTCTGGCTCTATTCTG
CLDN1	claudin 1	NM_021101	Claudin; biomarker; TNBRCA	TCTGGCTATTTTAGTTGCCACAG	AGAGAGCCTGACCAAATTCGT
CLDN3	claudin 3	NM_001306	Claudin; biomarker; TNBRCA	AACACCATTATCCGGGACTTCT	GCGGAGTAGACGACCTTGG
CLDN4	claudin 4	NM_001305	Claudin; biomarker; TNBRCA	GGGGCAAGTGTACCAACTG	GACACCGGCACTATCACCA
CLDN7	claudin 7	NM_001185023	Claudin; biomarker; TNBRCA	GCTGCAAAATGTACGACTCG	GGAGACCACCATTAGGGCTC
MYC	v-myc myelocytomatosis viral oncogene homolog (avian) (c-MYC)	NM_002467	Cell cycle; Proliferation; TNBRCA	GGCTCCCAAAAGGTCA	CTGCGTAGTTGTGCTGATGT
COL1A2	collagen, type I, alpha 2	NM_000089	Basal/Myoep/Stem	GAGCGGTAACAAGGGTGAGC	CTTCCCCATTAGGGCCTCTC
CXCL12	chemokine (C-X-C motif) ligand 12	NM_001178134	Angio/lymphangiogenesis	ATTCTCAACACTCCAAACTGTGC	ACTTTAGCTTCGGGTCAATGC
CXCR4	chemokine (C-X-C motif) receptor 4	NM_003467	Angio/lymphangiogenesis	GCGCAATGGATTGGTCATCCT	TGCAGCCTGTACTTGTCCG
DAND5	DAN domain family, member 5	NM_152654	CTC	AAGTGATCCAGGGGATGTGTA	GATGATTTCGGAGGCGTATGG
DLL1	delta-like 1 (Drosophila)	NM_005618	Notch signalling	GACGAACACTACTACGGAGAGG	AGCCAGGGTTGCACACTTT
EMP1	epithelial membrane protein 1	NM_001423	Luminal progenitor	GTGCTGGCTGTGCATTCTTG	CCGTGGTGATACTGCGTTCC
EPCAM	epithelial cell adhesion molecule	NM_002354	Epithelial; Cell surface marker; CTC	CTTGTCTGTTCTTCTGACCCC	TGATCCTGACTGCGATGAGAG
EPHA4	EPH receptor A4	NM_004438	Basal/Myoep/Stem	CTTGGGTGAAGCTCTCATCAG	GCAAGGAGACGTTTAACCTGT
EGFR	epidermal growth factor receptor (ERBB1, HER1)	NM_201284	Growth factor receptor; TNBRCA	TTGCCGCAAAGTGTGTAACG	GTCACCCCTAAATGCCACCG
ERBB2	v-erb-b2 erythroblastic leukemia viral oncogene homolog 2	NM_001005862	Growth factor receptor; biomarker	TGTGACTGCCTGTCCCTACAA	CCAGACCATAGCACACTCGG
ERBB3	v-erb-b2 erythroblastic leukemia viral oncogene homolog 3	NM_001982	Growth factor receptor	GACCCAGGTCTACGATGGGAA	GTGAGCTGAGTCAAGCGGAG
ESR1	estrogen receptor 1	NM_001122741	Hormone receptor; biomarker	GGGAAGTATGGCTATGGAATCTG	TGGCTGGACACATATAGTCGTT
ESR2	estrogen receptor 2	NM_001214902	Hormone receptor	TCCATCGCCAGTTATCACATCT	CTGGACAGTAACAGGGCTG
FGFR2	fibroblast growth factor receptor 2	NM_001144915	TNBRCA	AGCACCATACTGGACCAACAC	GGCAGCGAAACTTGACAGTG
FOXA1	forkhead box A1	NM_004496	Luminal	CTGCACCTGAAAGGGGACC	GCCTTGAAGTCCAGCTTATGC
FSCN1	fascin homolog 1, actin-bundling protein (Strongylocentrotus purpuratus)	NM_003088	TNBRCA; EMT	CCAGGGTATGGACCTGTCTG	GTGTGGGTACGGAAGGCAC
GATA3	GATA binding protein 3	NM_001002295	Luminal	GCGGGCTCTATCA	GCCTTCGCTTGGGCTTAAT
ID4	inhibitor of DNA binding 4	NM_001546	TNBRCA	CACGTTATCGACTACATCCTGG	TGTCGCCCTGCTTGTTCAC
IGFBP6	insulin-like growth factor binding protein 6	NM_002178	Basal/Myoep/Stem	TGTGAACCGCAGAGACCAAC	GCCCATCTCAGTGTCTTGGA
IGF1R	insulin-like growth factor 1 receptor	NM_000875	TNBRCA	ATGCTGACCTCTGTTACCTCT	GGCTTATTCCCCACAATGTAGTT
ITGA6	integrin, alpha 6	NM_001079818	Cell surface marker	GGCGGTGTTATGTCCTGAGTC	AATCGCCCATCACAAAAGCTC
ITGB1	integrin, beta 1	NM_033668	Cell surface marker	GTAACCAACCGTAGCAAAGGA	TCCCCTGATCTTAATCGCAAAAC
ITPKB	inositol-trisphosphate 3-kinase B	NM_002221	Luminal progenitor	TCTCCTCATCCTACGAAGACTCA	GCTCACTCTAGGTTTCTGCTGG
JAG1	jagged 1	NM_000214	Basal/Myoep/Stem	GAGCTATTTGCCGACAAGGC	GGAGTTTGCAAGACCCATGC
KIT	v-kit Hardy-Zuckerman 4 feline sarcoma viral oncogene homolog	NM_000222	Luminal progenitor	CGTTCTGCTCCTACTGCTTCG	CCCACGCGGACTATTAAGTCT
KRT14	keratin 14	NM_000526	Basal/Myoep/Stem	CCCAGCTCCGCTGCGAGATG	GGGAGGAGGAGAGGTGGGCG
KRT17	keratin 17	NM_000422	Basal/Myoep/Stem	GGTGGGTGGTGAGATCAATGT	CGCGGTTCAGTTCCTCTGTC
KRT18	keratin 18	NM_199187	Luminal	TCGCAAATACTGTGGACAATGC	GCAGTCGTGTGATATTGGTGT
KRT19	keratin 19	NM_002276	Luminal	AACGGCGAGCTAGAGGTGA	GGATGGTCGTGTAGTAGTGGC
KRT5	keratin 5	NM_000424	Basal/Myoep/Stem	CCAAGGTTGATGCACTGATGG	TGTCAGAGACATGCGTCTGC
KRT8	keratin 8	NM_001256293	Luminal	TCCTCAGGCAGCTATATGAAGAG	GGTTGGCAATATCCTCGTACTGT
LGR5	leucine-rich repeat containing G protein-coupled receptor 5	NM_003667	Basal/Myoep/Stem	CACCTCCTACCTAGACCTCAGT	CGCAAGACGTAACTCCTCCAG
MAX	myc-associated factor X	NM_002382	Myc co-regulator	CAATCTGCGGCTGACAAACG	GCTTGAGGTCGTCAATATCTTGC
MKI67	antigen identified by monoclonal antibody Ki-67	NM_002417	Proliferation	AGAAGAAGTGGTGCTTCGGAA	AGTTTGCGTGGCCTGTACTAA
MLL4	myeloid/lymphoid or mixed-lineage leukemia 4	NM_014727	Luminal progenitor	TTGTCCTTGGGACTCGAATCA	CCTGTCCAGATCCAACTCTTCT
MMP1	matrix metallopeptidase 1	NM_002421	Basal/myope/stem; matrix metalloproteinase	AAAATTACACGCCAGATTTGCC	GGTGTGACATTACTCCAGAGTTG
MMP10	matrix metallopeptidase 10 (stromelysin 2)	NM_002425	Matrix metalloproteinase	TGCTCTGCCTATCCTCTGAGT	TCACATCCTTTTCGAGGTTGTAG
MTOR	mechanistic target of rapamyci	NM_004958	TNBRCA	GCAGATTTGCCAACTATCTTCGG	CAGCGGTAAAAGTGTCCCCTG
MUC1	mucin 1	NM_001204294	Luminal; Cell surface marker	TGCCGCCGAAAGAACTACG	TGGGGTACTCGCTCATAGGAT
MYCN	v-myc myelocytomatosis viral related oncogene	NM_005378	Myc co-regulator	TGATCCTCAAACGATGCCTTC	GGACGCCTCGCTCTTTATCT
MYLK	myosin light chain kinase	NM_053028	Basal/Myoep/Stem	CCCGAGGTTGTCTGGTTCAAA	GCAGGTGTACTTGGCATCGT
NOTCH3	notch 3	NM_000435	Luminal	CGTGGCTACACTGGACCTC	AGATACAGGTGAACTGGCCTAT
NOTCH4	notch 4	NM_004557	Basal/Myoep/Stem	CCTGGCTCCTTCAACTGCC	GCAAGTAGGTCCAGACAGGT
NTRK2	neurotrophic tyrosine kinase, receptor, type 2	NM_006180	CTC	ACCCGAAACAAACTGACGAGT	AGCATGTAAATGGATTGCCCA
POU5F1	POU domain, class 5, transcription factor 1, OCT4	NM_002701	Pluripotency	CTTGAATCCCGAATGGAAAGGG	GTGTATATCCCAGGGTGATCCTC
OLR1	oxidized low density lipoprotein (lectin-like) receptor 1	NM_001172632	CTC	TTGCCTGGGATTAGTAGTGACC	GCTTGCTCTTGTGTTAGGAGGT
PARP1	poly (ADP-ribose) polymerase 1	NM_001618	TNBRCA	TGGAAAAGTCCCACACTGGTA	AAGCTCAGAGAACCCATCCAC
PARP2	poly (ADP-ribose) polymerase 2	NM_001042618	TNBRCA	GCCTTGCTGTTAAAGGGCAAA	TCCTTCACAATACACATGAGCC
PDGFRA	platelet-derived growth factor receptor, alpha polypeptide	NM_006206	TNBRCA	TGGCAGTACCCCATGTCTGAA	CCAAGACCGTCACAAAAGGC
PDGFRB	platelet-derived growth factor receptor, beta polypeptide	NM_002609	TNBRCA	AGACACGGGAGAATACTTTTGC	AGTTCCTCGGCATCATTAGGG
PGR	progesterone receptor	NM_000926	Hormone receptor; biomarker	TTATGGTGTCCTTACCTGTGGG	GCGGATTTTATCAACGATGCAG
PIM1	pim-1 oncogene	NM_001243186	TNBRCA	GGCTCGGTCTACTCAGGCA	GGAAATCCGGTCCTTCTCCAC
PLCB4	phospholipase C, beta 4	NM_001172646	Luminal	TATTCGGTCGGGAGCCATAC	GACACAAACTATCCGCCCTTC
PROM1	prominin 1	NM_001145848	Luminal; Cell surface marker	AGTCGGAAACTGGCAGATAGC	GGTAGTGTTGTACTGGGCCAAT
PTEN	phosphatase and tensin homolog	NM_000314	TNBRCA	AGGGACGAACTGGTGTAATGA	CTGGTCCTTACTTCCCCATAGAA
RAB11B	RAB11B, member RAS oncogene family	NM_004218	Luminal progenitor	TCACCCGCAACGAGTTCAAC	CTGCACCACGGTAGTACGC
RB1	retinoblastoma 1	NM_000321	Cell cycle; TNBRCA	TTGGATCACAGCGATACAAACTT	AGCGCACGCCAATAAAGACAT
ROR1	receptor tyrosine kinase-like orphan receptor 1	NM_001083592	Luminal progenitor	TCTCGGCTGCGGATTAGAAAC	TCCAGTGGAAGAAACCACCTC
SKP2	S-phase kinase-associated protein 2	NM_032637	Cell cycle	ATGCCCCAATCTTGTCCATCT	CACCGACTGAGTGATAGGTGT
SNAI2	snail homolog 2 (Drosophila)	NM_003068	EMT	CGAACTGGACACACATACAGTG	CTGAGGATCTCTGGTTGTGGT
SOX2	SRY (sex determining region Y)-box 2	NM_003106	Pluripotency	TACAGCATGTCCTACTCGCAG	GAGGAAGAGGTAACCACAGGG
SPARC	secreted protein, acidic, cysteine-rich (osteonectin)	NM_003118	Basal/Myoep/Stem; EMT	TGAGGTATCTGTGGGAGCTAATC	CCTTGCCGTGTTTGCAGTG
SPON1	spondin 1 (F-spondin)	NM_006108	Basal/Myoep/Stem	CCCAAGTCAGAGGGCTACTG	GGTTCCCGGCTTGTAGAAGT
TGFB1	transforming growth factor, beta 1	NM_000660	Angio/lymphangiogenesis	CAATTCCTGGCGATACCTCAG	GCACAACTCCGGTGACATCAA
TGFB2	transforming growth factor, beta 2	NM_001135599	Angio/lymphangiogenesis	CAGCACACTCGATATGGACCA	CCTCGGGCTCAGGATAGTCT
TGFB3	transforming growth factor, beta 3	NM_003239	Angio/lymphangiogenesis	CACCCAGGAAAACACCGAGTC	GCGGAAAACCTTGGAGGTAAT
TGFBR1	transforming growth factor, beta receptor 1	NM_001130916	Angio/lymphangiogenesis	GCTGTATTGCAGACTTAGGACTC	TTTTTGTTCCCACTCTGTGGTT
TGFBR2	transforming growth factor, beta receptor II	NM_003242	Angio/lymphangiogenesis	GTAGCTCTGATGAGTGCAATGAC	CAGATATGGCAACTCCCAGTG
TGFBR3	transforming growth factor, beta receptor III	NM_001195684	Angio/lymphangiogenesis	GTGTTCCCTCCAAAGTGCAAC	AGCTCGATGATGTGTACTTCCT
THBS1	thrombospondin 1	NM_003246	TNBRCA	TGCTATCACAACCACAACGGAGTTCAGT	GCAGGACACCTTTTTGCAGATG
TNFRSF14	tumor necrosis factor receptor superfamily, member 14	NM_003820	Luminal progenitor	ATACAAGAGCGAAGGTCTCACG	CTGAGTCTCCCATAACAGCGG
TNK1	tyrosine kinase, non-receptor, 1	NM_001251902	Luminal progenitor	CACTCGGCCAGAGCACTTC	TTTTCAGAGCTTCGGACAGTC
TP53	tumor protein p53	NM_001126118	TNBRCA; Anti-apoptosis	TAACAGTTCCTGCATGGGCGGC	AGGACAGGCACAAACACGCACC
TP63	tumor protein p63	NM_001114978	Basal/Myoep/Stem	GTCATTTGATTCGAGTAGAGGGG	CTGGGGTGGCTCATAAGGT
TP73	tumor protein p73	NM_001204187	TNBRCA	CGGGCCATGCCTGTTTACA	TGTCCTTCGTTGAAGTCCCTC
TWIST1	TWIST homolog of drosophila	NM_000474	EMT	GTCCGCAGTCTTACGAGGAG	GCTTGAGGGTCTGAATCTTGCT
VCAM1	vascular cell adhesion molecule 1	NM_001199834	CTC	GGGAAGATGGTCGTGATCCTT	TCTGGGGTGGTCTCGATTTTA
VEGFA	vascular endothelial growth factor A	NM_001171627	Angio/lymphangiogenesis	CCGCATAATCTGC	ACGAGGGCCTGGAGTGTGT
VEGFC	vascular endothelial growth factor C	NM_005429	Angio/lymphangiogenesis	AGGCTGGCAACATAACAGAGA	TCCCCACATCTATACACACCTC
FIGF	C-fos induced growth factor (vascular endothelial growth factor D)	NM_004469	Angio/lymphangiogenesis	AGGGCTGCACTGAGTTCTTTG	TCCCATCGGTCCACTAGGTTT
VIM	vimentin	NM_003380	EMT	AGTCCACTGAGTACCGGAGAC	CATTTCACGCATCTGGCGTTC
WNT2	wingless-type MMTV integration site family member 2	NM_003391	CTC	GCCTTTGTTTATGCCATCTCCT	CTTGGCGCTTCCCATCTTCTT
YAP1	Yes-associated protein 1	NM_006106	CTC	TAGCCCTGCGTAGCCAGTTA	TCATGCTTAGTCCACTGTCTGT
ZEB1	zinc finger E-box binding homeobox 1	NM_001174094	EMT	TTACACCTTTGCATACAGAACCC	TTTACGATTACACCCAGACTGC
ZEB2	zinc finger E-box binding homeobox 2	NM_001171653	EMT	AACAACGAGATTCTACAAGCCTC	TCGCGTTCCTCCAGTTTTCTT

Basal/Stem

Gene	Fold change (B/(L+LP))	*P* value
CAV2	89.7	9.3x10^{-14}
MYLK	74.1	6.2x10^{-16}
PDGFRA	55.2	3.2x10^{-8}
KRT5	29.3	2.9x10^{-11}
CAV1	21.3	9.1x10^{-8}
JAG1	18.3	1.6x10^{-8}
COL1A2	13.2	3.2x10^{-8}
PLCB4	10.7	2.0x10^{-6}
MTOR	7.4	0.002
NOTCH3	6.4	0.001
BCL2L1	6.4	2.5x10^{-4}
ACTA2	6.2	0.001
LGR5	5.5	5.2x10^{-6}
MYCN	5.2	0.004
CCNB1	4.9	0.023
NOTCH4	4.2	5.7x10^{-5}
CDK1	4.2	0.023
PGR	4.2	0.009
CCNE1	3.9	0.047
EGFR	3.5	0.002
TP63	3.0	0.002
SNAI2	2.9	0.003
CDH3	2.8	2.6x10^{-6}
ITGA6	2.6	0.009
CHEK1	2.5	0.045
ESR2	2.2	0.006
MME	2.1	0.039
TGFB2	2.0	0.039
SKP2	2.0	0.023
BMI1	1.6	0.054
CLDN4	1.5	0.055
ITGB1	1.4	0.038
BCL2	1.4	2.5x10^{-4}
TP53	1.3	1.8x10^{-4}

Luminal

Gene	Fold change (L/B)	*p-value
FOXA1	98.7	8.4x10^{-12}
CDH1	47.5	5.4x10^{-8}
EMP1	45.9	1.0x10^{-13}
ERBB3	44.3	6.0x10^{-9}
MUC1	13.0	2.4x10^{-16}
CD24	9.2	2.8x10^{-5}
ERBB2	8.3	4.7x10^{-4}
PTEN	4.5	9.9x10^{-5}
AR	3.7	0.001
KRT19	3.6	0.018
GATA3	2.4	4.7x10^{-4}

Luminal progenitor

Gene	Fold change (LP/L)	*p-value
KIT	108.6	1.7x10^{-10}
PROM1	36.9	2.2x10^{-10}
CXCR4	11.7	0.002
PARP2	2.3	0.057

		BR vs. BM		LN vs. BM		LU vs. BM		PB vs. BM		LN vs. BR		LU vs. BR		PB vs. BR		LU vs. LN		PB vs. LN		PB vs LU	
	P value (ANOVA)	FC	*P* value	FC	*P* value	FC	*P* value	FC	*P* value	FC	*P* value	FC	*P* value	FC	*P* value	FC	*P* value	FC	*P* value	FC	*P* value
ACTA2	2.02E-04			46.82	1.72E-02																
AKT1	6.42E-08					11.43	4.37E-02					72.80	3.72E-03			7.88	2.12E-03				
AR	2.46E-06			-19.64	4.38E-02			-71.69	7.99E-03	-42.74	7.11E-04	-14.44	1.15E-02	-155.99	1.80E-04						
BCL2	0.01													-3.12	1.74E-03						
BCL2L1	1.61E-06	500.95	9.94E-06							-530.81	7.47E-08	-191.44	2.21E-06	-69.64	1.98E-02						
BMI1	5.95E-11	4.35	1.93E-02					-657.66	6.34E-04	-18.10	1.73E-06	-37.70	6.81E-08	-2858.47	2.10E-10			-157.96	2.05E-02	-75.81	1.33E-02
BRCA1	2.35E-04																				
CAV2	8.97E-03									31.38	2.16E-02	49.95	8.43E-03								
CCNB1	6.69E-04															9.46	3.30E-02				
CCNE1	2.40E-06	-916.67	1.23E-03									204.94	2.07E-04			8.15	3.25E-02				
CD24	2.55E-05																			-9.36	4.84E-01
CD44	1.63E-08			26.13	1.54E-02	33.15	4.92E-03					33.15	3.41E-02								
THY1	1.41E-03																				
CDH1	6.60E-03	-965.96	9.89E-03							61.91	5.03E-02	50.90	3.90E-02								
CDH3	1.74E-08			-3.77	1.13E-03					-11.97	1.15E-08	-5.02	1.95E-04	-7.78	8.74E-04	2.38	6.70E-05				
CDK1	4.23E-05					47.24	1.11E-02	229.85	3.83E-02												
CDK2	2.41E-09				6.22E-01	36.00	1.63E-03					73.14	3.36E-03	207.26	5.05E-02	8.61	3.88E-03				
CDKN1B	1.15E-12	12.13	4.14E-03	-7.81	7.14E-04			-19.06	4.61E-04			-33.37	1.08E-08	-231.15	9.93E-11	2.84	2.83E-03			-6.93	7.73E-03
CHEK1	6.60E-03													20.80	5.36E-02			11.51	5.10E-02		
CLDN4	4.62E-11			5.69	2.27E-07	4.44	1.51E-06			4.04	9.73E-04	3.15	5.81E-03		8.08E-01			-6.72	2.60E-06	-5.24	1.81E-05
CLDN7	2.18E-07					30.72	6.91E-03														
MYC	1.15E-12					537.98	8.17E-07	363.25	3.51E-03			473.22	2.40E-05	319.52	5.09E-03	15.09	5.46E-04				
COL1A2	4.72E-04																				
CXCR4	1.67E-03															4.06	4.95E-02				
DAND5	1.92E-04	-73.91	5.72E-03							26.38	3.37E-02	19.07	2.83E-02								
EMP1	2.18E-07					146.80	2.69E-03					372.51	3.36E-04			11.26	3.47E-02				
EPCAM	1.00E-04							-37.10	3.42E-04									-26.52	3.98E-02	-20.15	2.28E-03
EPHA4	0.03	-10.06	5.10E-02																		
ERBB2	0.02																				
ERBB3	1.13E-04									29.24	3.36E-02	22.00	1.65E-02								
FGFR2	9.44E-04	-12.65	1.03E-03									6.84	2.14E-03	7.27	4.74E-02						
FSCN1	2.47E-05													29.15	3.73E-02						
GATA3	4.56E-06																				
ID4	2.06E-03															3.06	1.62E-03				
IGF1R	3.75E-04									-12.15	4.97E-02					7.31	6.33E-05				
ITGA6	9.66E-03															3.14	1.89E-02				
ITGB1	6.42E-08			5.41	6.54E-03	7.23	5.25E-04	7.26	3.24E-02	8.24	7.25E-03	11.01	1.12E-03	11.06	2.28E-02						
ITPKB	1.07E-08	-35.77	9.64E-10	-3.00	4.25E-02			-5.71	1.37E-02	11.92	1.05E-06	14.42	1.25E-08	6.27	1.90E-02						
JAG1	1.52E-04							67.25	2.32E-03					326.16	4.06E-05			26.52	7.41E-03	48.45	2.70E-04
KIT	6.36E-05			14.02	5.38E-02																
KRT17	0.01																				
KRT19	4.29E-04			13.52	4.37E-04	8.65	2.00E-03			7.93	4.14E-02										
KRT5	4.98E-04	39.69	1.05E-04	7.75	6.54E-03	6.28	8.11E-03	12.78	2.16E-02			-6.32	4.25E-02								
LGR5	3.10E-10	23.61	3.35E-03	-10.03	3.27E-03							-100.37	1.39E-09	-78.02	9.59E-05						
MLL4	9.85E-03																				
MMP1	2.22E-15	-9.16	0.00E+00	-2.29	4.28E-04					4.00	6.11E-08	6.52	0.00E+00	5.04	1.75E-06	1.63	1.62E-04				
MTOR	1.76E-04			-8.24	2.99E-03											3.52	2.74E-03	13.75	1.55E-03		
MUC1	6.26E-05	-96.76	1.72E-03									31.91	5.87E-03								
MYCN	3.83E-04	84.62	1.83E-02							-38.75	5.00E-02	-28.44	3.95E-02	-351.84	1.61E-03						
MYLK	7.64E-06																				
NTRK2	1.03E-06							-258.84	1.35E-02	-25.69	2.09E-04	-51.46	1.89E-04	-932.24	8.29E-06						
POU5F1	2.65E-04							-358.74	9.73E-03					-1262.54	1.02E-03			-93.89	1.77E-02	-50.95	3.38E-02
OLR1	2.71E-08	14.53	1.60E-07	-3.37	1.10E-03	-5.31	1.21E-03	-45.63	8.95E-05	4.32	2.23E-02	2.74	2.31E-03								
PARP1	2.45E-06			285.78	1.02E-02	426.00	4.89E-06	123.87	9.23E-03												
PARP2	5.91E-08					46.42	5.30E-03					46.42	3.59E-02								
PGR	1.33E-05			-14.36	6.93E-03			-88.23	5.20E-04	-30.26	2.15E-03	-10.40	5.29E-02	-185.90	1.53E-04					-17.87	1.04E-02
PLCB4	8.74E-06	89.71	8.94E-04			17.86	1.58E-03			-25.05	8.52E-03			-48.96	1.85E-02	4.99	5.21E-03				
PROM1	8.25E-04	-184.91	3.70E-02									96.41	2.57E-03								
PTEN	1.29E-04																				
SKP2	4.42E-03									-7.29	5.44E-03			-10.53	1.53E-02						
SNAI2	3.76E-05			-8.23	6.86E-03									18.66	1.79E-02	3.02	1.36E-02	31.94	4.03E-05	10.59	7.10E-03
SOX2	3.10E-10	4.69	6.53E-03					-408.80	1.82E-03	-27.66	2.96E-06	-22.88	4.38E-06	-1918.84	1.36E-10			-69.37	1.20E-02	-83.85	7.53E-04
TGFB1	1.07E-06	190.66	2.71E-02							-195.96	3.96E-03	-175.69	2.37E-03	-2106.13	7.08E-04						
TGFB2	2.36E-06			-5.61	3.30E-02					-34.22	5.75E-06	-7.06	2.43E-02	-12.78	4.14E-02	4.84	2.86E-05				
TGFBR1	1.00E-04	-62.23	1.43E-02							31.97	5.27E-02	47.52	2.28E-03								
TGFBR2	6.26E-09			-232.76	2.85E-06	-273.06	1.28E-07	-1611.58	9.09E-06					-121.94	1.28E-01						
TGFBR3	4.87E-03																	-14.46	3.97E-02	-13.75	2.91E-02
TP53	2.65E-03															1.78	2.65E-03				
TP63	4.18E-05									20.32	5.97E-03	29.98	9.82E-05	41.38	2.62E-02						
TP73	3.15E-08											-30.83	2.55E-04	-24.05	4.48E-02	-5.34	4.30E-04				
TWIST1	0.03			-18.28	1.51E-02																
VCAM1	0.01																				
VEGFA	2.31E-07	-182.32	3.98E-04							24.39	2.78E-02	84.39	1.66E-04	709.51	7.93E-06			29.09	1.05E-02		
VEGFC	2.47E-05	-24.35	4.11E-02					-257.57	1.03E-03									-71.90	2.89E-02	-73.20	3.00E-03
FIGF	1.80E-04																			-10.87	1.58E-02
VIM	1.84E-06					9.03	9.62E-03	15.16	3.97E-02	8.76	4.55E-02	18.09	1.53E-03	30.38	7.70E-03						
WNT2	8.77E-04							-45.20	1.46E-02									-21.04	3.84E-02		
YAP1	2.30E-03																				
ZEB1	4.40E-04													-55.59	3.24E-02			-138.13	7.07E-04	-54.25	7.71E-04
ZEB2	0.01																				