

# Supplementary Information

*Huh et al*

*Feb 13, 2017*

## Contents

<b>1 Loading relevant data</b>	<b>2</b>
1.1 Packages used in this analysis . . . . .	2
1.2 Summary of data-sets used in this study . . . . .	2
1.3 Publicly available information used in this analysis . . . . .	3
<b>2 Patterns in FACS data: Figures 1 and S1</b>	<b>6</b>
<b>3 Tcell RNA seq data: Figure 2</b>	<b>11</b>
3.1 Overview of RNA seq data . . . . .	11
3.2 edgeR conversion to logCPM space: Figure 2A . . . . .	13
3.3 edgeR: differential analysis . . . . .	18
3.4 DESeq2 standardisation . . . . .	20
3.5 PCA visualisation . . . . .	21
3.6 Differential Gene expression analysis . . . . .	22
3.7 Figure 2B: Heatmap of differentially expressed genes . . . . .	22
<b>4 Gene Set enrichment analysis: Figure 2</b>	<b>29</b>
4.1 GSEA in HTSanalyzeR . . . . .	29
4.2 Immune c7 . . . . .	34
4.3 Cell types enriched . . . . .	36
4.4 CIBERSORT . . . . .	43
<b>5 Cell activity: exhaustion, cytotoxic signatures: Figures S3, S5</b>	<b>46</b>
5.1 Existing Signatures . . . . .	46
5.2 Analysis of TIGIT staining the CD3+ cells . . . . .	52
5.3 PD1 staining . . . . .	55
<b>6 TCR clonality</b>	<b>59</b>
<b>7 PDL1 amplification: Figure 4</b>	<b>64</b>
7.1 TCGA data . . . . .	64
7.2 Oslo cohort . . . . .	74
<b>8 Frequency of co-amplification with the CCL data set</b>	<b>76</b>
8.1 TCGA data set: Figure S7 . . . . .	76
8.2 TCGA RNA-seq data . . . . .	79
8.3 TCGA neoantigen data . . . . .	80
8.4 Linear model for TCGA dataset . . . . .	81
8.5 Oslo Dataset: Figure 4 . . . . .	85
<b>9 CCL-ERBB2 amplification based on FISH images: Figure 5</b>	<b>88</b>
9.1 DCIS analysis . . . . .	88
9.2 IDC cases: . . . . .	99
9.3 Relationship between Diversity and age (Figure 5F): . . . . .	101
<b>10 R Session Info</b>	<b>114</b>

# 1 Loading relevant data

## 1.1 Packages used in this analysis

```
library(ape)
library(ade4)
library(beeswarm)
library(boot)
library(compiler)
library(DESeq2)
library(dplyr)
library(edgeR)
library(fpc)
library(gdata)
library(gee)
library(geepack)
library(GGally)
library(ggplot2)
library(gplots)
library(GSEABase)
library(heatmap.plus)
library(HTSanalyzeR)
library(knitr)
library(leaps)
library(lme4)
library(matrixStats)
library(parallel)
library(plyr)
library(RColorBrewer)
library(reshape2)
library(rgl)
library(scatterplot3d)
library(spatstat)
library(TDA)
library(vegan)

knit_hooks$set(webgl = hook_webgl)
# set the colour palette
palette(c( "#FF6FCF", "#8000FF", "#0F82F4", "#008040", "#66FFCC", "#FF0000", "#43F708"))
## in the order of :
# DCIS, HER2, IDC/LUM, PA, NP, TN , unkown

# Colors for heatmap: Rd high and Blue low
HMPallete=brewer.pal(11, "RdBu")
```

## 1.2 Summary of data-sets used in this study

The analyses conducted in this study come from a number of different sources:

- FACS and I-FISH data (generated in house):
  - FACS was performed on patient samples, separating cells according to CD markers
  - IF for TIGIT and PD1

- FISH for CCL cluster and ERBB2 was performed on a cohort of DCIS (20 patients) and matched DCIS-IDC (18 samples)
- RNA-seq data:
  - RNA-seq was performed on T-cells isolated from a set of patients in house
  - Public data for IDCs was attained from TCGA
- CN data:
  - CN data for PDL1, CCL, ERBB2 from TCGA
  - CN data for PDL1, CCL, ERBB2 from Oslo

### 1.3 Publicly available information used in this analysis

Publicly available information from the following sources were used in this study:

#### 1.3.1 Existing immune Databases

Upload comprehensive list of immune related genes (from ImmPort: <http://immport.org/immport-open/public/reference/genelists>) and from Innate DB: <http://www.innatedb.com/annotatedGenes.do?type=innatedb>

## Antigen_Processing_and_Presentation		Antimicrobials
##	148	544
## BCRSignalingPathway		Chemokine_Receptors
##	275	53
## Chemokines		Cytokine_Receptors
##	102	308
## Cytokines		Interferon_Receptor
##	456	3
## Interferons		Interleukins
##	17	47
## Interleukins_Receptor		NaturalKiller_Cell_Cytotoxicity
##	43	135
## TCRsignalingPathway		TGFb_Family_Member
##	291	33
## TGFb_Family_Member_Receptor		TNF_Family_Members
##	12	12
## TNF_Family_Members_Receptors		
##	19	

#### 1.3.2 Cytotoxic, exhaustion, activation gene Lists:

Obtained from Tirosh, Science 2016; Pardoll and Singer 2016.

```
GeneListAct=read.csv("online_public_data/Supplementary Table 5.csv", header=T, stringsAsFactors = F)
GeneListAct[GeneListAct==""] = NA
GeneListAct=as.list(GeneListAct, na.rm=T)
GeneListAct=lapply(GeneListAct, na.omit)
GeneListAct=lapply(GeneListAct, toupper)
names(GeneListAct)

## [1] "Activated"    "Inhibitory"    "Cytotoxic"     "Exhausted"     "Naive"
## [6] "Activation"   "Dysfunction"
```

### 1.3.3 GSEA: Collections from MSigDB

Load collections for GSEA. These were downloaded from MSigDB <http://software.broadinstitute.org/gsea/msigdb/>: The c2 canonical pathways and c7 immune collection

```
immunec7=getGmt(con="online_public_data/c7.all.v5.1.symbols.gmt", geneIdType=SymbolIdentifier(),
                  collectionType=BroadCollection(category="c7"))
CanPath=getGmt(con="online_public_data/c2.cp.v5.1.symbols.gmt", geneIdType=SymbolIdentifier(),
                  collectionType=BroadCollection(category="c2"))

# convert to lists:
immunec7entrez=mapIdentifiers(immunec7, EntrezIdentifier('org.Hs.eg.db'))

## Loading required package: DBI
##
immunec7listentrez=geneIds(immunec7entrez)
Canentrez=mapIdentifiers(CanPath, EntrezIdentifier('org.Hs.eg.db'))
Canlistentrez=geneIds(Canentrez)
CanlistSym=geneIds(CanPath)
```

Fetch all immune related genes from canonical pathways:

```
STS=c("IMMUNE", "_IL_[0-9]+", "_IL[0-9]+", "B_CELL", "T_CELL", "LYMPHOCYTE", "BCR", "TCR", "CCR[0-9]",
      "COMPLEMENT", "CXCR[0-9]", "CXCR[0-9]", "TH1", "TH2", "FC_EPSILON", "GRANZYME", "IFN", "IGF",
      "TCR", "MACROPHAGE", "OX40", "PDGF", "T-CELL", "TCELL", "CYTOTOXIC", "T_HELPER", "CTLA4", "ANTIGEN",
      "INFECTION", "MHC", "INTERFERON", "INTERLEUKIN", "BASOPHIL", "EOSINOPHIL", "MAST_CELL", "DENDRITIC")

anew=sapply(STS, function(x) grep(x, names(CanlistSym)))
anew2=unlist(anew)

GeneList=stack(CanlistSym[anew2])
write.csv(GeneList, file="output/Canonical_pathways_immune_related.csv")
```

Compile different sources together to get combined immune gene list:

```
CombinedGeneList=data.frame(Symbol=c(as.character(ImmPort$Symbol),
                                       as.character(GeneList$values),
                                       as.character(InnatDB$Gene.Symbol)),
                           Fn=c(as.character(ImmPort$Category), as.character(GeneList$ind),
                                 rep("Innate_Immune_System", length(InnatDB$Gene.Symbol))))
```

### 1.3.4 Gene Locations

The following file curated from UCSC genome website genome.ucsc.edu identifies the chromosomal position of a number of genes of interest. This will be used in CCL and PDL1 analysis:

```
Genes=read.csv("data/chr17arm.csv", header=T)
```

```
Genes
```

```
##       Locus      Start        End Chr
## 1      CCL2  32582296  32584220  17
## 2      CCL13  32683471  32685629  17
## 3      CCL7  32597235  32599261  17
## 4      CCL8  32646066  32648421  17
## 5      CCL4  34431220  34433014  17
```

```

## 6      CCL3   34415603 34417506 17
## 7      CCL3L1  34522268 34524156 17
## 8      CCL5   34198496 34207377 17
## 9      CCL23  34340097 34345005 17
## 10     CCL14  34310692 34313764 17
## 11     CCL15  34324618 34329084 17
## 12     CCL16  34303535 34308523 17
## 13     CCL11  32612687 32615199 17
## 14     CCL1   32687399 32690252 17
## 15     CCL18  34391643 34398841 17
## 16     ERBB2  37856254 37884915 17
## 17     CCL3L3 34623842 34625716 17
## 18     TIGIT  114012154 114029135 3
## 19     PDCD1  242792033 242801058 2
## 20     GZMB  25100161 25103432 14
## 21     CD274  5450503 5470567 9
## 22     PDCD1LG2 5510545 5571282 9
## 23     STAT3  40465343 40540513 17
## 24     JAK2   4985245 5128183 9

```

### 1.3.5 TCGA data

Copy number, RNA-seq, patient clinical data and GISTIC information was downloaded from firebrowse (firebrowse.org) Here, we load this data:

```

TCGArna=read.delim("online_public_data/RNAseqTCGA.txt")
colnames(TCGArna)=gsub("\\.", "-", colnames(TCGArna))
rownames(TCGArna)=TCGArna$X
TCGArna=TCGArna[, -1]

## do a vst transformation
TCGAlongCPM=cpm(TCGArna, log=T, prior.counts=1)

```

Download level 4 GISTIC data from firebrowse.org

```

GisticData=read.delim("online_public_data/TCGA_gistic_lvl_4.txt", sep="\t", header=T)
colnames(GisticData)=gsub("\\.", "-", substr(colnames(GisticData), 1, 12))

NewGistic=GisticData[match(Genes$Locus, GisticData$`Gene-Symbol`), ]
rownames(NewGistic)=NewGistic$`Gene-Symbol`
NewGistic=NewGistic[, -c(1:3)]

# cap the max and min values to ensure colour scheme is ok
NewGistic[which(NewGistic>2, arr.ind = T)]=2
NewGistic[which(NewGistic<(-2), arr.ind=T)]=(-2)

```

Load clinical data:

```
load("online_public_data/TCGAClinInfo_Appended_Mar16.RData")
```

Additional information on Immune infiltration, estimated neoantigen load was obtained from the following studies:

- ESTIMATE (Yoshihara et al, 2013): Supplementary Data 2

```

ESTscore=read.csv("online_public_data/ESTIMATE_scores.csv", header=T)
ESTscore$Description=gsub("\\.", "-", ESTscore$Description)
ESTscore$Description=substr(ESTscore$Description, 3, 14)
summary(ESTscore)

## Description StromalScore ImmuneScore ESTIMATEScore
## Length:1212 Min.   :-3584.1  Min.   :-2449.16 Min.   :-5343.4
## Class :character 1st Qu.: 448.5  1st Qu.: -14.98 1st Qu.: 627.3
## Mode  :character Median : 1399.8 Median : 794.80 Median : 2266.4
##                           Mean   : 1253.6 Mean   : 975.80 Mean   : 2229.4
##                           3rd Qu.: 2150.0 3rd Qu.: 1792.90 3rd Qu.: 3945.7
##                           Max.   : 4053.4  Max.   : 6062.83 Max.   : 9493.2

ESTscore$TumPurity=cos(0.6049872018+0.0001467884*ESTscore$ESTIMATEScore)

```

- Estimated neoantigen load from Table S4A from Rooney et al 2015

```

NeoData=read.csv("online_public_data/s4A_mutations_Rooney2015.csv", header=T)
NeoData=NeoData[NeoData$Cancer=="BRCA", ]
summary(NeoData)

## PatientID Cancer CYT Total.Mutations
## TCGA-A1-A0SB: 1 BRCA :760 Min.   : 0.06302 Min.   : 1.00
## TCGA-A1-A0SD: 1 BLCA  : 0 1st Qu.: 2.51813 1st Qu.: 24.00
## TCGA-A1-A0SE: 1 CESC  : 0 Median : 5.46181 Median : 37.00
## TCGA-A1-A0SF: 1 CRC   : 0 Mean   : 10.24959 Mean   : 52.35
## TCGA-A1-A0SG: 1 GBM   : 0 3rd Qu.: 11.45561 3rd Qu.: 62.00
## TCGA-A1-A0SH: 1 HNSC  : 0 Max.   :108.87068 Max.   :393.00
## (Other)    :754 (Other) : 0
## Predicted.NeoAgs NeoAgs_Observed.Expected AnyVirusPresent HBV
## Min.   : 0.000 Min.   :0.0000 Mode :logical Mode :logical
## 1st Qu.: 4.000 1st Qu.:0.6478 FALSE:759 FALSE:760
## Median : 7.000 Median :0.9361 TRUE :1 NA's  :0
## Mean   : 9.637 Mean   :1.0185 NA's  :0
## 3rd Qu.:11.000 3rd Qu.:1.3023
## Max.   :64.000 Max.   :5.3781
## NA's   :25    NA's  :32
## HPV      EBV
## Mode :logical Mode :logical
## FALSE:760   FALSE:760
## NA's :0     NA's :0
## 
## 
## 
## 
```

## 2 Patterns in FACS data: Figures 1 and S1

Load FACS data (Supplementary Table 2):

```

facs=read.csv("online_public_data/Supplementary Table 2 FACS data.csv", row.names=1)
tissues=c(rep("NP", 12), rep("PA", 10), rep("DCIS", length(grep("DCIS", rownames(facs)))),
          rep("LUM", 9), rep("HER2", 6), rep("TN", 10))
tissues2=rep("", length(tissues))

```

```

tissues2[grep("stroma", rownames(facs))]="S"
tissues2[grep("org", rownames(facs))]="O"
cells <- colnames(facs)[-1]
#cells <- facs[-1, 1]

```

Was Figure 1 made in R as a barplot?

Overview of the Shannon entropies:

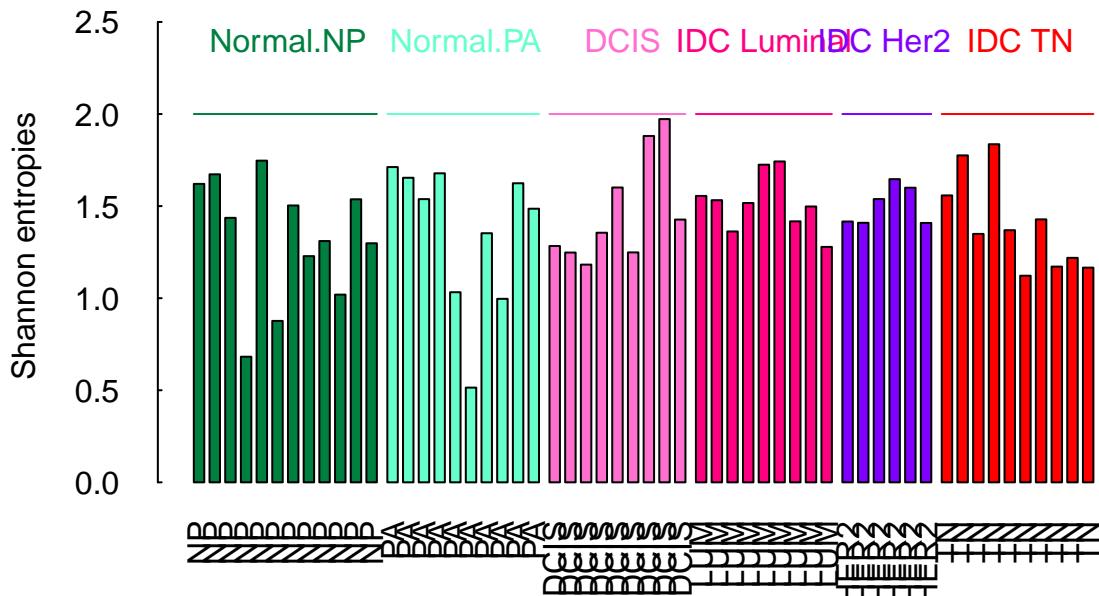
```

colors <- c(rep('#008040', length(which(tissues == "NP"))), rep('#66FFCC', length(which(tissues == "PA")))
           rep('#FF6FCF', length(which(tissues == "DCIS"))), rep('#FF0080', length(which(tissues == "LUM"))
           rep('#8000FF', length(which(tissues == "HER2"))), rep('#FF0000', length(which(tissues == "TN"))))

diversities <- sapply(1:nrow(facs), function(i) diversity(as.numeric(facs[i, ])))

par(mfrow = c(1, 1), tcl = 0.1, las = 2, oma = c(1, 1, 0, 0))
barplot(diversities, space = c(c(0, rep(0.5, length(which(tissues == "NP")) - 1)),
                               c(1, rep(0.5, length(which(tissues == "PA")) - 1)),
                               c(1, rep(0.5, length(which(tissues == "DCIS")) - 1)),
                               c(1, rep(0.5, length(which(tissues == "LUM")) - 1)),
                               c(1, rep(0.5, length(which(tissues == "HER2")) - 1)),
                               c(1, rep(0.5, length(which(tissues == "TN")) - 1))),
                               col = colors, ylab = 'Shannon entropies', names = tissues, ylim = c(0, diversity(rep(1/13, 13)))
text(9, 2.4, 'Normal.NP', col = unique(colors)[1])
lines(c(0, 17.5), c(2, 2), col = unique(colors)[1])
text(26, 2.4, 'Normal.PA', col = unique(colors)[2])
lines(c(18.5, 18.5 + 10 * 1.5 - 0.5), c(2, 2), col = unique(colors)[2])
text(41, 2.4, 'DCIS', col = unique(colors)[3])
lines(c(34, 34 + 9 * 1.5 - 0.5), c(2, 2), col = unique(colors)[3])
text(54.5, 2.4, 'IDC Luminal', col = unique(colors)[4])
lines(c(48, 48 + 9 * 1.5 - 0.5), c(2, 2), col = unique(colors)[4])
text(66, 2.4, 'IDC Her2', col = unique(colors)[5])
lines(c(62, 62 + 6 * 1.5 - 0.5), c(2, 2), col = unique(colors)[5])
text(79, 2.4, 'IDC TN', col = unique(colors)[6])
lines(c(71.5, 71.5 + 10 * 1.5 - 0.5), c(2, 2), col = unique(colors)[6])

```



See if there are any associations between the CD4 and CD8 components: Figure S1B

```
par(mfrow=c(2,2))
# plot CD4 vs CD8 colour coded:
plot(facs$CD8..T.cells, facs$CD4..T.cells, col=factor(tissues), xlab="CD8 proportion", ylab="CD4 proportion")
r1=cor.test(facs$CD8..T.cells, facs$CD4..T.cells)

r2=sapply(unique(tissues), function(x) cor.test(facs$CD8..T.cells[tissues==x],
                                              facs$CD4..T.cells[tissues==x])$p.value)
r3=sapply(unique(tissues), function(x) cor.test(facs$CD8..T.cells[tissues==x],
                                              facs$CD4..T.cells[tissues==x])$estimate)

legend("topright", as.character(c(round(r1$estimate*100)/100, round(r3*100)/100)),
       col=c("black", factor(unique(tissues))), lty=ifelse(c(r1$p.value, r2)<0.05,3, 1), lwd=3)

plot(facs$CD8..T.cells, facs$macrophages, col=factor(tissues), xlab="CD8 proportion",
      ylab="macrophage proportion", pch=19)

s1=cor.test(facs$CD8..T.cells, facs$macrophages)
s2=sapply(unique(tissues), function(x) cor.test(facs$CD8..T.cells[tissues==x],
                                              facs$macrophages[tissues==x])$p.value)
s3=sapply(unique(tissues), function(x) cor.test(facs$CD8..T.cells[tissues==x],
                                              facs$macrophages[tissues==x])$estimate)

legend("topright", as.character(c(round(s1$estimate*100)/100, round(s3*100)/100)),
       col=c("black", factor(unique(tissues))), lty=ifelse(c(s1$p.value, s2)<0.05,3, 1), lwd=3)

plot((facs$CD8..T.cells+facs$CD4..T.cells), facs$dendritic.cells, col=factor(tissues), xlab="CD8 + CD4")

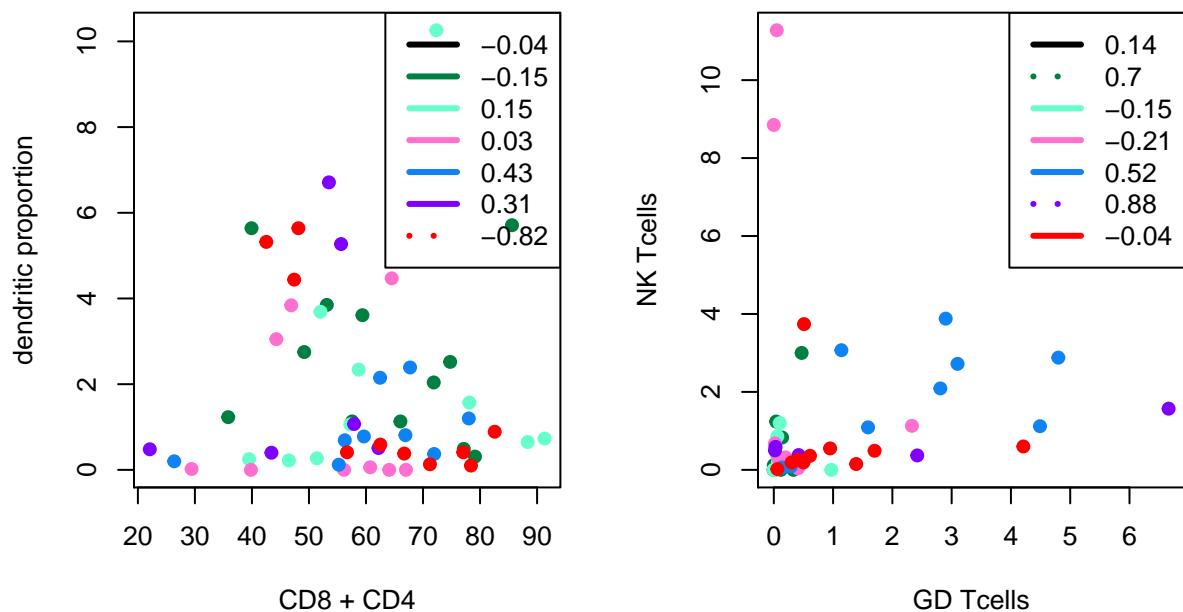
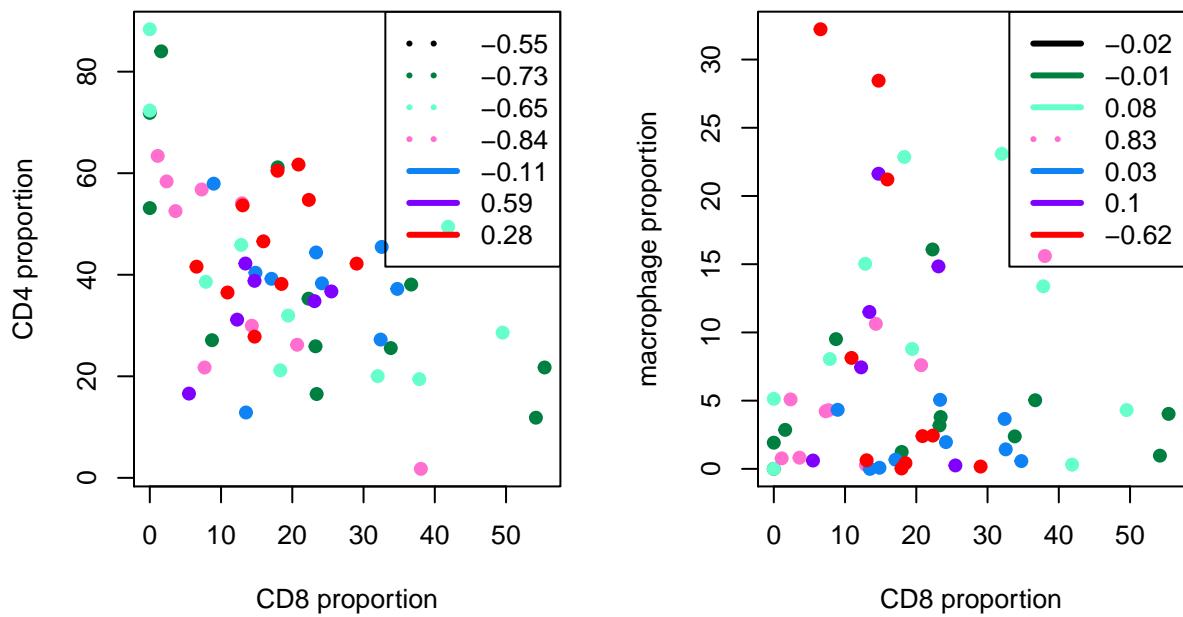
t1=cor.test((facs$CD8..T.cells+facs$CD4..T.cells), facs$dendritic.cells)
t2=sapply(unique(tissues), function(x) cor.test((facs$CD8..T.cells+facs$CD4..T.cells)[tissues==x],
                                              facs$dendritic.cells[tissues==x])$p.value)
t3=sapply(unique(tissues), function(x) cor.test((facs$CD8..T.cells+facs$CD4..T.cells)[tissues==x],
                                              facs$dendritic.cells[tissues==x])$estimate)

legend("topright", as.character(c(round(t1$estimate*100)/100, round(t3*100)/100)),
       col=c("black", factor(unique(tissues))), lty=ifelse(c(t1$p.value, t2)<0.05,3, 1), lwd=3)

plot(facs$gammadelta.Tcells, facs$NK.T.cells, col=factor(tissues), xlab="GD Tcells", ylab="NK Tcells", pch=19)

u1=cor.test(facs$gammadelta.Tcells, facs$NK.T.cells)
u2=sapply(unique(tissues), function(x) cor.test(facs$gammadelta.Tcells[tissues==x],
                                              facs$NK.T.cells[tissues==x])$p.value)
u3=sapply(unique(tissues), function(x) cor.test(facs$gammadelta.Tcells[tissues==x],
                                              facs$NK.T.cells[tissues==x])$estimate)

legend("topright", as.character(c(round(u1$estimate*100)/100, round(u3*100)/100)),
       col=c("black", factor(unique(tissues))), lty=ifelse(c(u1$p.value, u2)<0.05,3, 1), lwd=3)
```



Overall test to look for correlations between the different populations of interest:

```

fac$CD4total=fac$CD8..T.cells+fac$CD4..T.cells
N=ncol(fac)
cor.mat=matrix(NA, N, N)
for (i in 1:N){
  if (i<N){
    for (j in (i+1):N){
      cor.mat[i,j]=cor.test(fac[,i], fac[,j])$p.value
      cor.mat[j,i]=cor.test(fac[,i], fac[,j])$estimate
    }
  }
}
  
```

```

    }
}

cor.mat1=upper.tri(cor.mat)*cor.mat
cor.mat1[lower.tri(cor.mat1)]=NA
cor.mat1new=p.adjust(cor.mat1)
cor.mat1new=matrix(cor.mat1new, N, N)
cor.mat2=lower.tri(cor.mat)*cor.mat

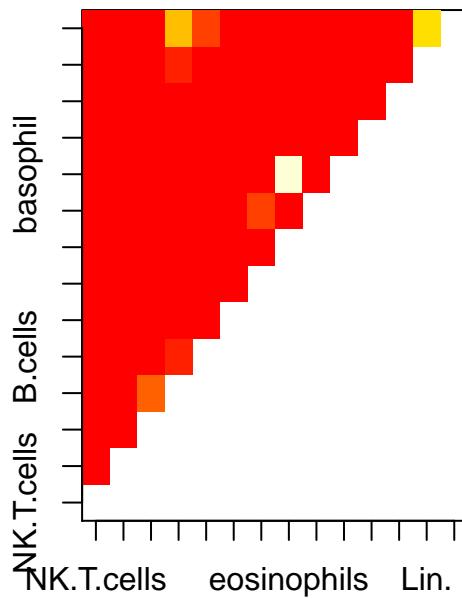
par(mfrow=c(1,2))

image(-log10(cor.mat1new), xaxt="n", yaxt="n", main="overall pvalues")
axis(1, seq(0, 1, length=N), colnames(facs))
axis(2, seq(0, 1, length=N), colnames(facs))

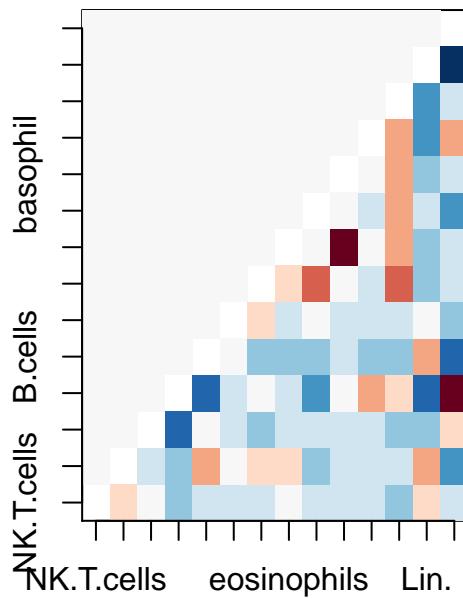
image(cor.mat2, main="overall correlation", xaxt="n", yaxt="n", col=HMPallette[11:1])
axis(1, seq(0, 1, length=N), colnames(facs))
axis(2, seq(0, 1, length=N), colnames(facs))

```

**overall pvalues**



**overall correlation**



Looking at associations in DCIS only

```

pdf('Facs_associations.pdf', useDingbats = FALSE)
par(mfrow=c(2,2))

TestValues=c("DCIS", "LUM", "HER2", "TN", "NP", "PA")

for (k in TestValues){
  facs2=facs[tissues==k, ]
  cor.mat=matrix(NA, N, N)
  for (i in 1:N){
    if (i < N){
      for (j in (i+1):N){
        cor.mat[i,j]=cor.test(facs2[,i], facs2[,j])$p.value
    }
  }
}

```

```

        cor.mat[j,i]=cor.test(facs2[ ,i], facs2[ ,j])$estimate
    }
}
}
cor.mat1=upper.tri(cor.mat)*cor.mat
cor.mat1[lower.tri(cor.mat1)]=NA
cor.mat1new=p.adjust(cor.mat1)
cor.mat1new=matrix(cor.mat1new, N, N)
cor.mat2=lower.tri(cor.mat)*cor.mat

image(-log10(cor.mat1new), xaxt="n", yaxt="n", main=sprintf("%s pvalues", k))
axis(1, seq(0, 1, length=N), colnames(facs), las=2)
axis(2, seq(0, 1, length=N), colnames(facs), las=2)

image(cor.mat2, main=sprintf("%s correlation", k), xaxt="n", yaxt="n", col=HMPallete[11:1])
axis(1, seq(0, 1, length=N), colnames(facs), las=2)
axis(2, seq(0, 1, length=N), colnames(facs), las=2)
}
dev.off()

## pdf
## 2

```

### 3 Tcell RNA seq data: Figure 2

#### 3.1 Overview of RNA seq data

##### 3.1.1 Loading in data

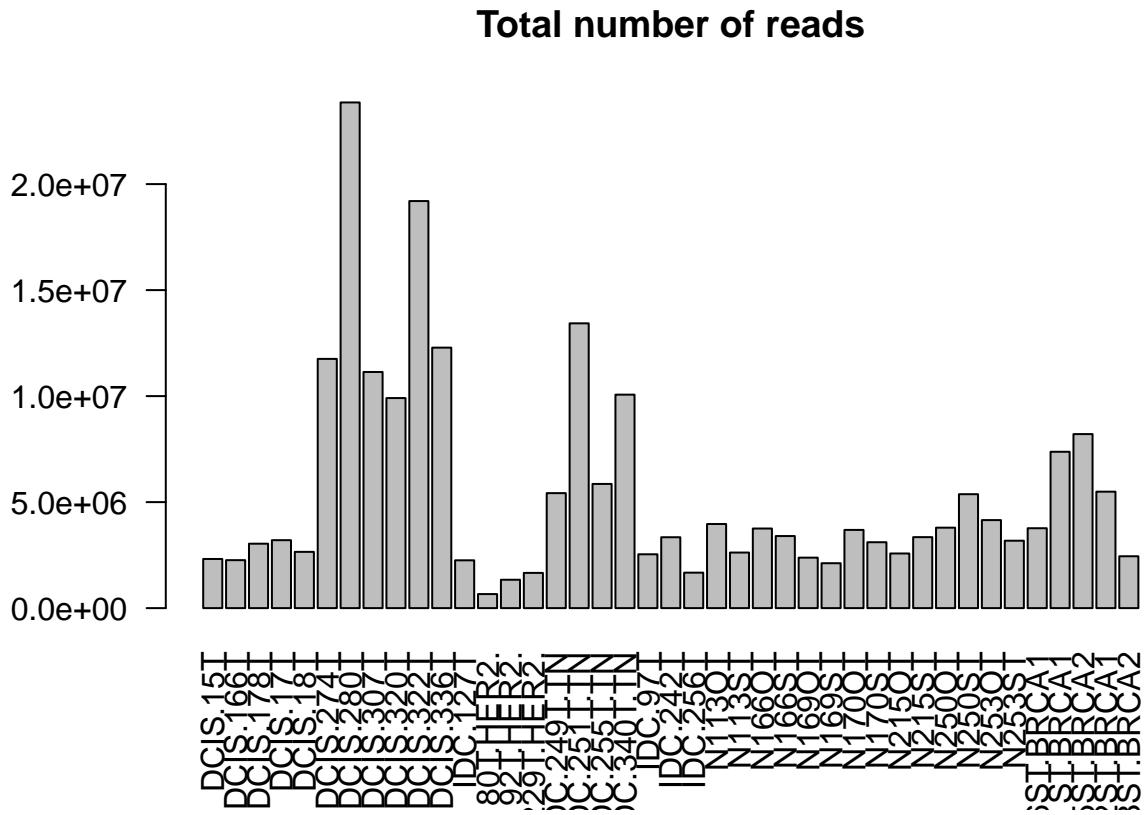
Load the new GE data. Remove genes with less than 45 counts

```

GEnew=read.csv(file ="data/Leukocytes_RNA-Seq_STAR_htseq-count.csv", header=T, row.names = 1)
QCinfo=GEnew[1:5, ]
GEnew=GEnew[-c(1:5), ]

TotRead=colSums(GEnew)
barplot(TotRead, las=2, main="Total number of reads")

```



```
GeneCounts=which(rowSums(GEnew)<45)
GEnewfilt=GEnew[-GeneCounts, ]
```

Load in the sample information.

```
SampleInfo=read.csv("data/sample_data.csv", header=T, row.names = 1)
GEnewfiltlog=log2(GEnewfilt+1)
```

### 3.1.2 Check for epithelial contamination

Check whether there is possible contamination with epithelial cells. For example, in the HER2+ samples, is HER2 expression observed? The markers checked are HER2, Epcam, Vimentin and various keratin. Report the fraction of counts wrt to total reads.

```

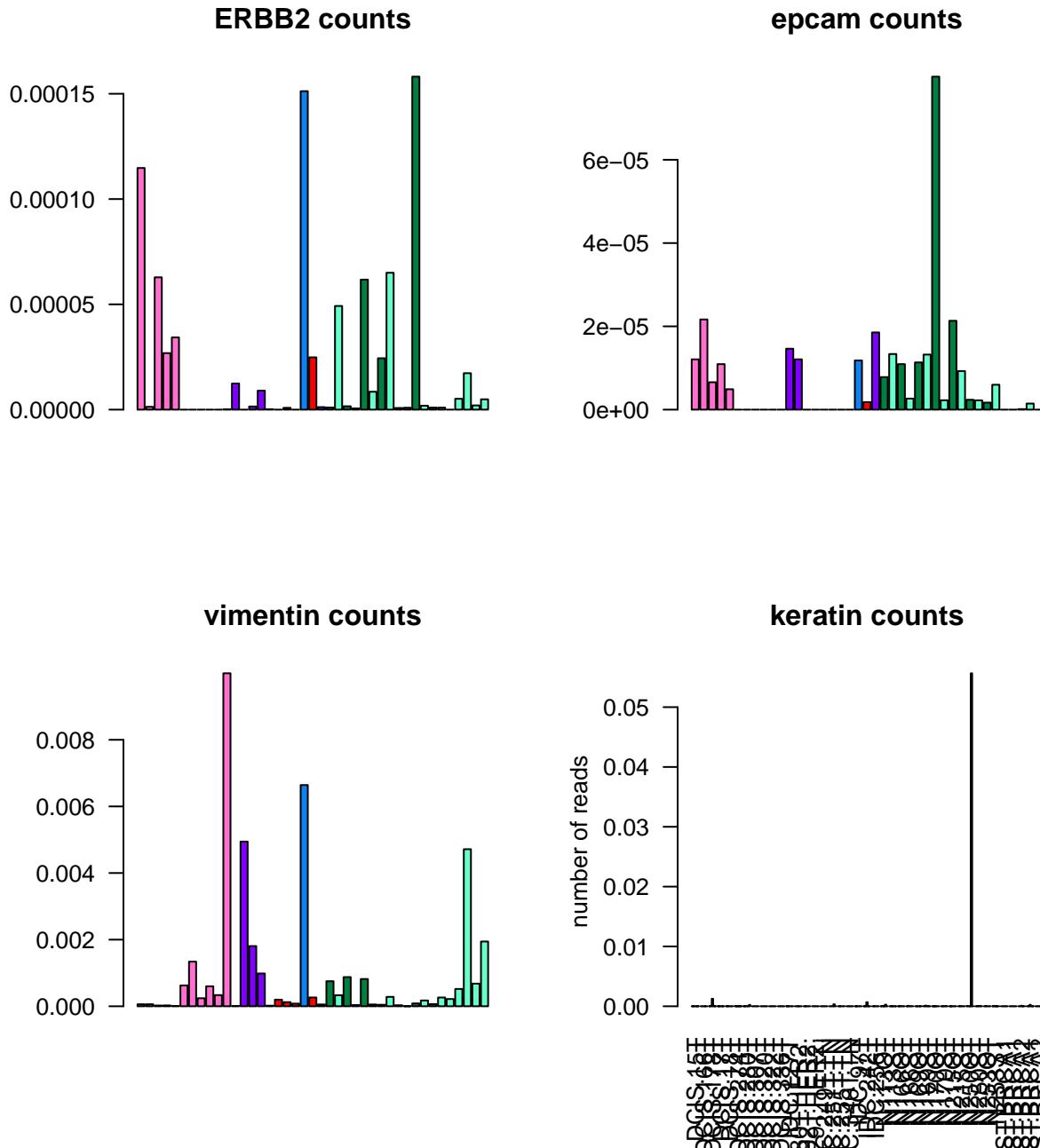
par(mfrow=c(2,2))
barplot(as.numeric(GEnew[grep("ERBB2", rownames(GEnew)) ,]/colSums(GEnew)),
       col=as.numeric(SampleInfo$Subtype), las=2, main="ERBB2 counts")

barplot(as.numeric(GEnew[grep("EPCAM", rownames(GEnew)) ,]/colSums(GEnew)),
       col=as.numeric(SampleInfo$Subtype), las=2, main="epcam counts")

barplot(as.numeric(GEnew["VIM",]/colSums(GEnew)), col=as.numeric(SampleInfo$Subtype), las=2,
       main="vimentin counts")

barplot(as.matrix(GEnew[c("KRT5","KRT7","KRT8", "KRT17", "KRT18") ,]/colSums(GEnew)), col=grey.colors(5),
       main="keratin counts", beside=T, ylab="number of reads")

```



Overall, samples do not look contaminated (Epithelial fractions have the most KRT) when assessing total counts. One DCIS sample has high vimentin counts.

Firstly, take into account batch effects: (i) This can be done by subtracting an estimated “batch” value from log-transformed data, or (ii) can be specified in the design model

ie.  $\sim \text{IDCvsDCIS} + \text{Batch}$

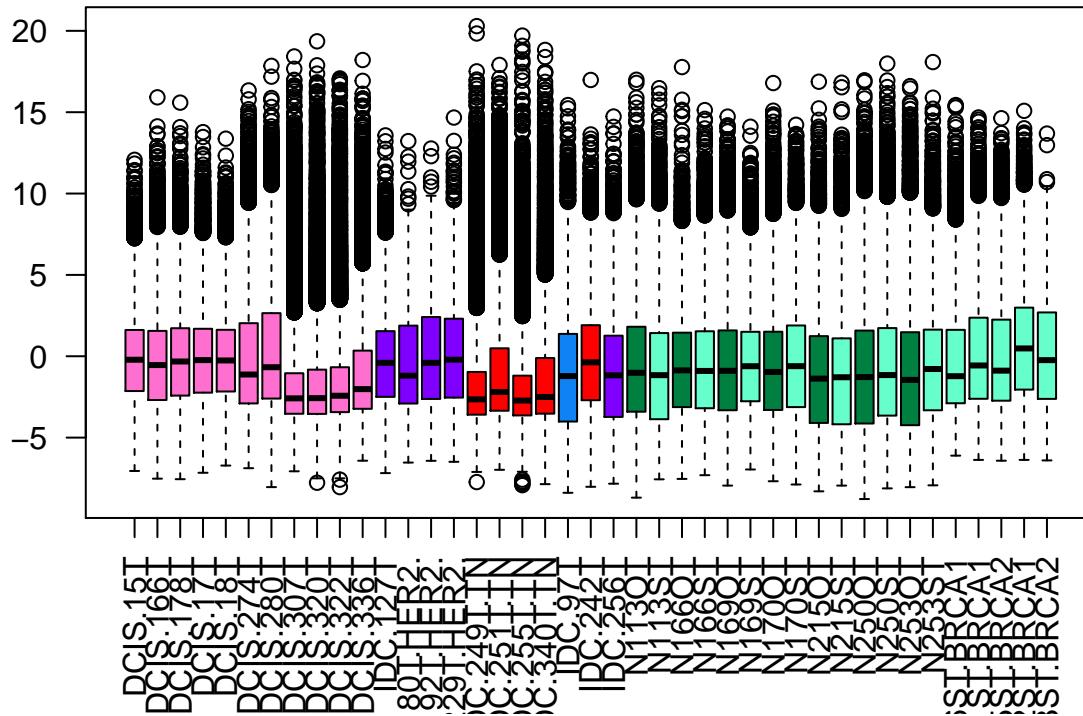
### 3.2 edgeR conversion to logCPM space: Figure 2A

edgeR allows direct correction of Batch effects (on log transformed data) - this is useful for visualisation and clustering.

We use edgeR to normalise the counts wrt to total counts. This will be then further corrected to correct for batch effects within the data set:

```
y=DGEList(counts=Gnewfilt)
y2=calcNormFactors(y)
logCPM=cpm(y2, log=T, prior.counts=1)
mod=model.matrix(~Subtype, data=SampleInfo)
logCPM=removeBatchEffect(logCPM, batch=SampleInfo$Batch, design=mod)

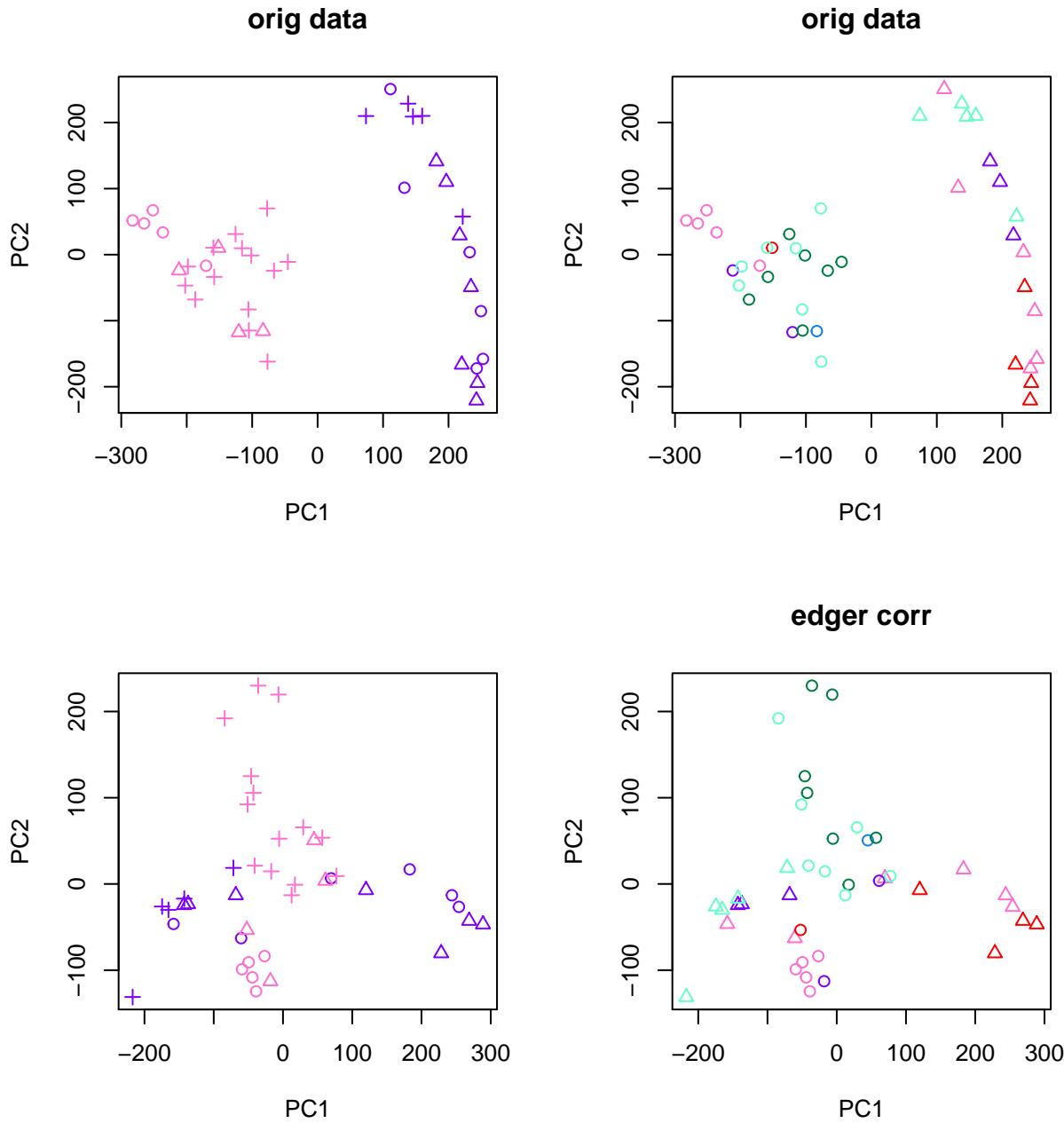
# boxplot of the batch corrected expression profiles
boxplot(logCPM, col=SampleInfo$Subtype, las=2)
```



Perform a PCA of the data before and after correction

```
# PCA of the result
par(mfrow=c(2,2))
pca1=prcomp(t(log2(Gnewfilt+1)))
plot(pca1$x[,1:2], col=SampleInfo$Batch, pch=as.numeric(SampleInfo$DCISIDC), main="orig data")
plot(pca1$x[,1:2], col=SampleInfo$Subtype, pch=as.numeric(SampleInfo$Batch), main="orig data")

pca2=prcomp(t(logCPM), scale=F)
plot(pca2$x[,1:2], col=SampleInfo$Batch, pch=as.numeric(SampleInfo$DCISIDC))
plot(pca2$x[,1:2], col=SampleInfo$Subtype, pch=as.numeric(SampleInfo$Batch), main="edger corr")
```



Create a 3D plot of the corrected data (FIGURE 2A):

```

SampC=SampleInfo$Subtype
SampC=as.character(SampC)
SampC[SampC=="Str"]="Org"
SampC=factor(SampC, levels=c("DCIS", "HER2", "LUM", "Org", "Str", "TN"))
plot3d(pca2$x[ ,1], pca2$x[ ,2], pca2$x[ ,3], radius=15, type="s", col=as.numeric(SampC),
       xlab = "PC1", ylab = "PC2", zlab="PC3")
rgl.postscript("output/All_samples_pca_3D_edger.pdf", fmt = "pdf")

```

You must enable Javascript to view this page properly.

Have a quick look at the relative levels of CD8 and CD4 across the different samples:

```

ColSide=SampC
levels(ColSide)=palette()[1:6]

```

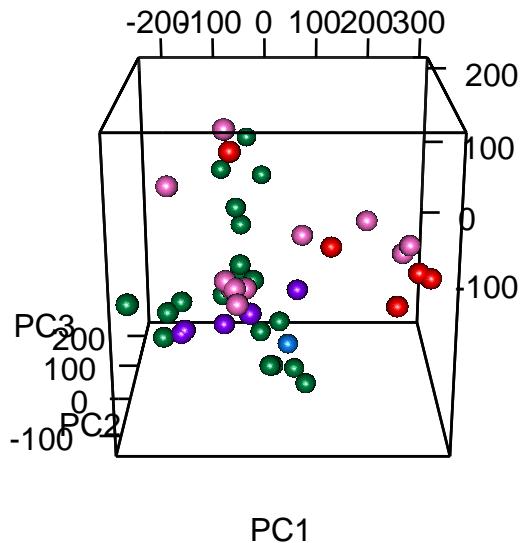


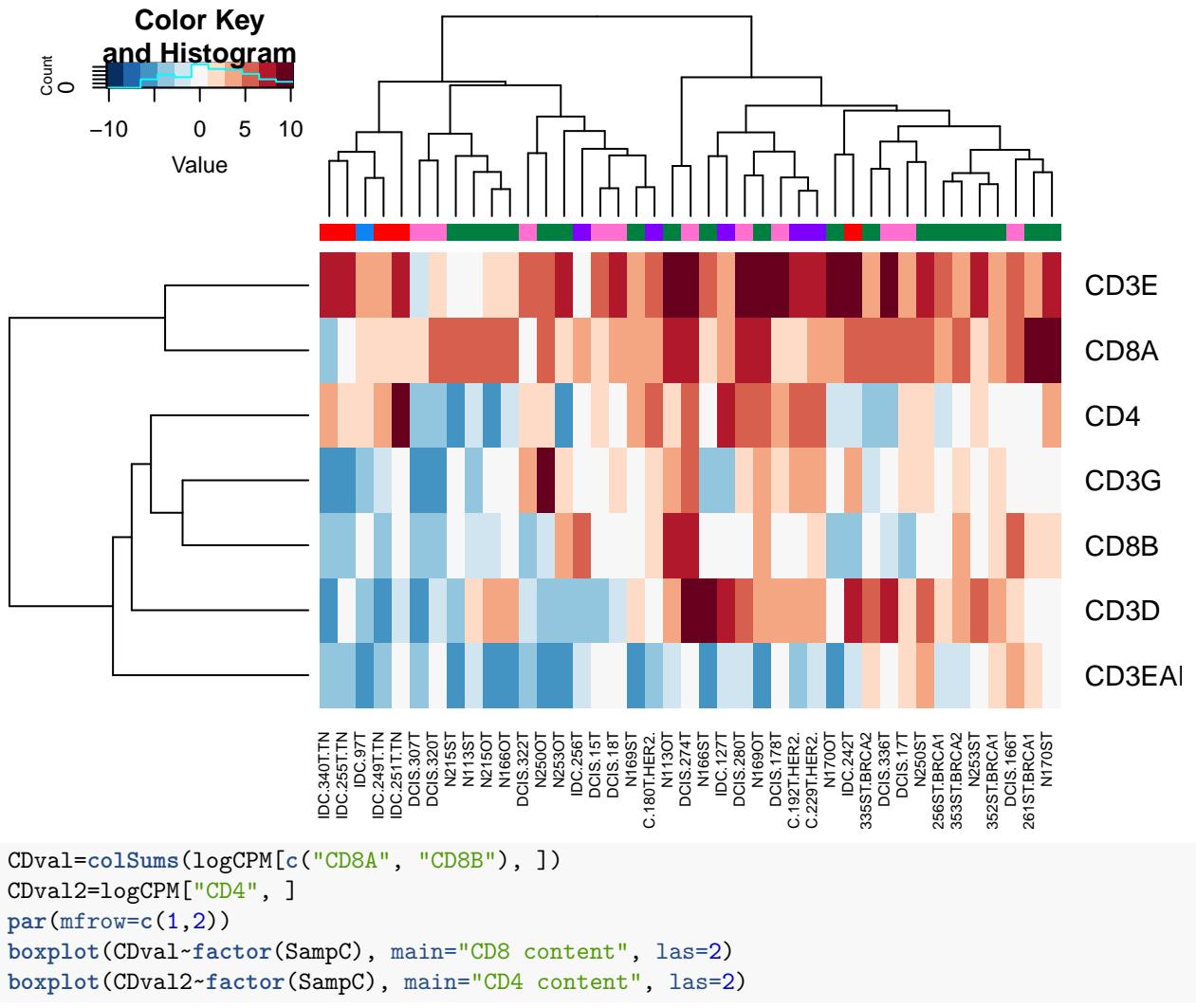
Figure 1: PCA of batch adjusted data

```

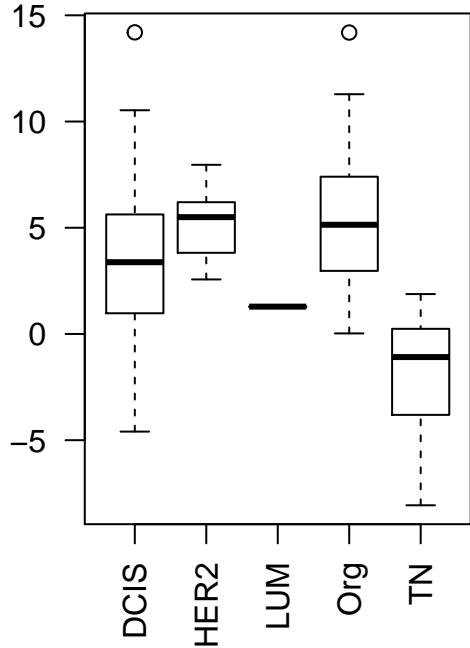
ColSide=as.character(ColSide)

# Look for CD4 and CD8 markers:
cdmark=sapply(c("CD4$", "CD8[A-Z]", "CD3[A-Z]"), function(x) grep(x, rownames(logCPM)))
heatmap.2(logCPM[unlist(cdmark), ], ColSideColors = ColSide, trace="none", col=HMPallete[11:1])

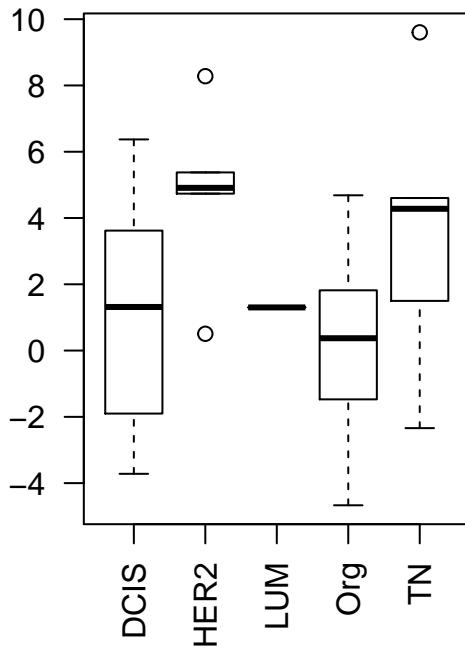
```



**CD8 content**



**CD4 content**



### 3.3 edgeR: differential analysis

#### 3.3.1 DCIS, IDC and normals

Using edgeR with a linear term for the batch effect, run a DGEA comparing DCIS vs IDC vs Normal. We use a BH-corrected p value of 0.05 to assign significance.

```

mod=model.matrix(~DCISIDC+Batch, data=SampleInfo)
y3=estimateGLMCommonDisp(y2, mod)
y4=estimateGLMTagwiseDisp(y3, mod)
fit=glmFit(y4, mod)

t2=factor(SampleInfo$DCISIDC, levels=c("IDC", "N", "DCIS"))
mod2=model.matrix(~t2+Batch, data=SampleInfo)
fit2=glmFit(y4, mod2)

# Find differences in IDC and DCIS
lrtA=glmLRT(fit, coef=2)
tabAall=topTags(lrtA, adjust.method = "BH", p.value=1, n=23000, sort.by = "none")
tabA=topTags(lrtA, adjust.method = "BH", p.value=0.05, n=10000)
#tabA=tabA$table[abs(tabA$table$logFC)>5, ]

# differences in DCIS and Normal
lrtB=glmLRT(fit, coef=3)
tabBall=topTags(lrtB, adjust.method = "BH", p.value=1, n=23000, sort.by = "none")
tabB=topTags(lrtB, adjust.method = "BH", p.value=0.05, n=10000)
#tabB=tabB$table[abs(tabB$table$logFC)>5, ]

# diff IDC and Normal
lrtC=glmLRT(fit2, coef=2)

```

```

tabCall=topTags(lrtC, adjust.method = "BH", p.value=1, n=23000, sort.by = "none")
tabC=topTags(lrtC, adjust.method = "BH", p.value=0.05, n=10000)
#tabC=tabC$table[abs(tabC$table$logFC)>5, ]

```

There are 2159 differentially expressed genes in the DCISvsIDC, DCIS vs normal and IDC vs Normal comparisons respectively. 138 genes in each set intersect with the Immport database.

### 3.3.2 Further information on IDC subtype

We can further break down information - firstly, IDC samples can be broken down into predominantly HER2 and Basal groups

```

# Evaluate HER2 vs Basal
SampleInfo$Subtype=relevel((SampleInfo$Subtype), "HER2")
mod=model.matrix(~factor(Subtype)+Batch, data=SampleInfo)
y3=estimateGLMCommonDisp(y2, mod)
y4=estimateGLMTagwiseDisp(y3, mod)
fitD=glmFit(y4, mod)
lrtCS=glmLRT(fitD, coef=6)

tabCS=topTags(lrtCS, adjust.method = "BH", p.value=0.05, n=10000)
tabCSall=topTags(lrtCS, adjust.method = "BH", p.value=1, n=30000, sort.by="none")

```

There are 635 differentially expressed genes in the DCISvsHER2, DCIS vs IDC and HER2vBasal comparisons respectively. 35 genes in each set intersect with the Immport database.

### 3.3.3 Normal samples

What are the differences between stroma and organoid and parous and nulliparous samples?

```

SampleInfo$Subtype=relevel((SampleInfo$Subtype), "Str")
mod=model.matrix(~factor(Subtype)+Batch, data=SampleInfo)
y3=estimateGLMCommonDisp(y2, mod)
y4=estimateGLMTagwiseDisp(y3, mod)
fitOS=glmFit(y4, mod)
lrtOS=glmLRT(fitOS, coef=5)
tabOS=topTags(lrtOS, adjust.method = "BH", p.value=0.05, n=10000)
tabOSall=topTags(lrtOS, adjust.method = "BH", p.value=1, n=30000, sort.by="none")

```

There are 1742 differentially expressed genes in stroma vs organoid comparison of which 91 genes intersect with the Immport database.

```

SampleInfo$Parity[grep("BRCA", SampleInfo$Sample)]=NA
mod=model.matrix(~factor(Parity), data=SampleInfo)
y3=estimateGLMCommonDisp(y2[ ,!is.na(SampleInfo$Parity)], mod)
y4=estimateGLMTagwiseDisp(y3, mod)
fitPAR=glmFit(y4, mod)
lrtPAR=glmLRT(fitPAR, coef=2)
tabPAR=topTags(lrtPAR, adjust.method = "BH", p.value=0.05, n=10000)
tabPARall=topTags(lrtPAR, adjust.method = "BH", p.value=1, n=30000, sort.by="none")

AllDGEA=cbind(tabAall$table, tabBall$table, tabCall$table,
              tabC$all$table, tabOSall$table, tabPARall$table)
colnames(AllDGEA)=paste(rep(c("DCIS-IDC", "DCIS-Norm", "IDC-Norm", "HER2-Basal", "Str-Org", "Null-Parou"))

```

```

x1=sapply(rownames(tabPARall), function(x) grep(x, ImmPort$Symbol))
x2=sapply(x1, length)
x3=sapply(x1, function(x) paste(ImmPort$Category[x], collapse = ", "))
AllDGEA=cbind(AllDGEA, ImmPortCategory=x3)
write.csv(AllDGEA, file="output/Supp_Table_3_Combined_DGEA_analyses.csv")

```

There are 2377 differentially expressed genes in null vs parous comparison of which 131 genes intersect with the Immport database.

### 3.4 DESeq2 standardisation

“Standardisation” in DESeq2: Adjust counts for normal counts in DESeq2, and perform a variance Stabilising Transform to adjust the data

```

SampleInfo$Subtype=factor(SampleInfo$Subtype)
SampleInfo$Batch=factor(SampleInfo$Batch)
dds <-DESeqDataSetFromMatrix(countData = GEnewfilt, colData = SampleInfo, design=~Subtype+Batch)
dds=estimateSizeFactors(dds)
dds=estimateDispersions(dds)

ddsCounts=counts(dds, normalized=T)
colnames(ddsCounts)=colnames(GEnewfilt)
ddsCount2=log10(ddsCounts+1)

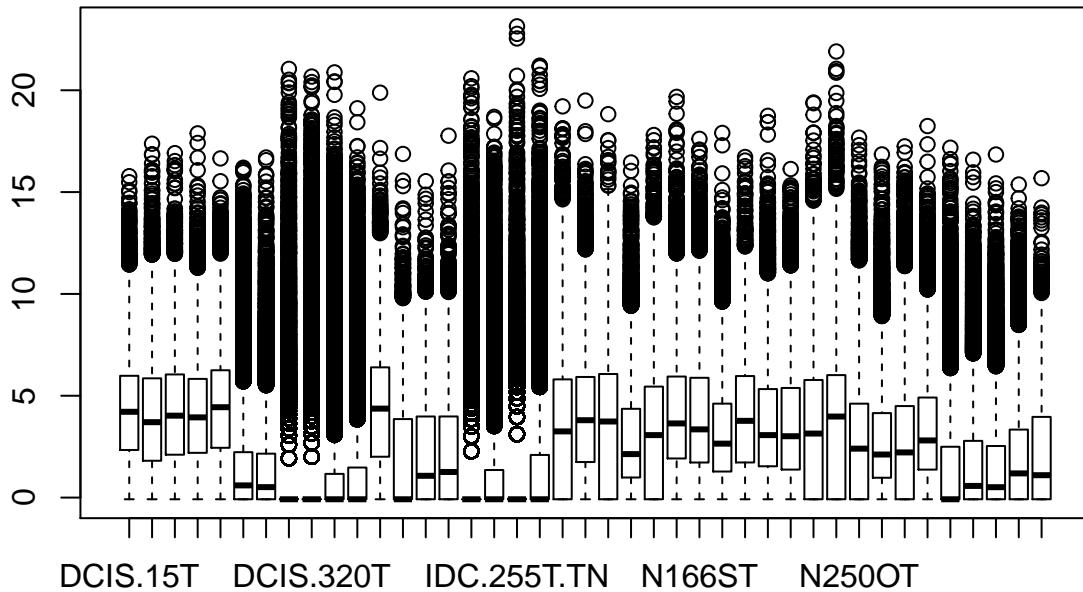
# can also use a variance Stabilizing Transform for correction
vstNew=varianceStabilizingTransformation(dds)
vstCounts=assay(vstNew)
colnames(vstCounts)=colnames(GEnewfilt)

# perform correction using limma?
mod=model.matrix(~factor(Subtype), data=SampleInfo)
newvstCounts=removeBatchEffect(vstCounts, batch=SampleInfo$Batch, design=mod)

boxplot(vstCounts, main="Vst transformed data")

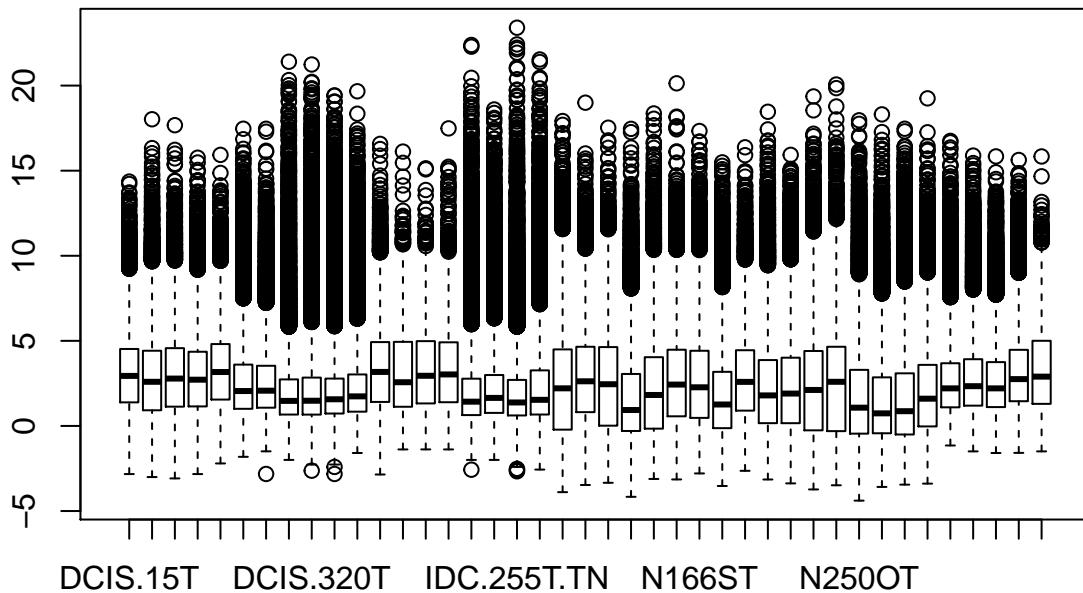
```

## Vst transformed data



```
boxplot(newvstCounts, main="Vst transformed batch corrected")
```

## Vst transformed batch corrected

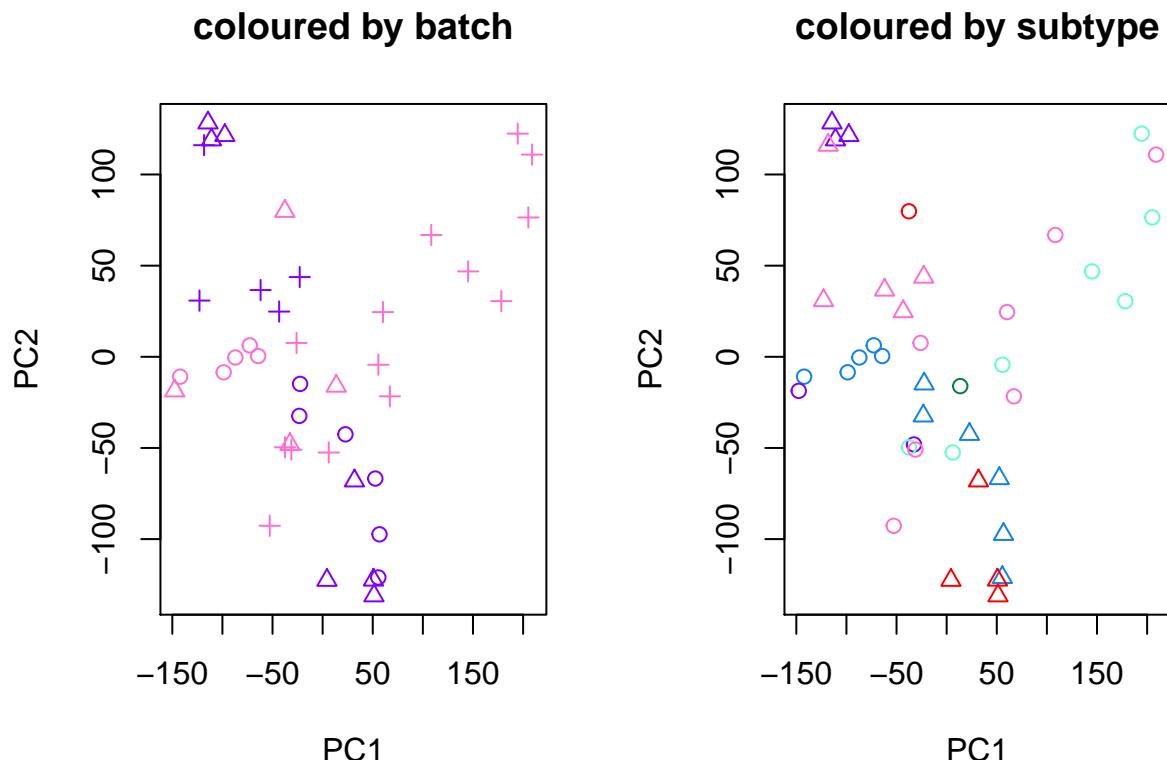


### 3.5 PCA visualisation

Perform a PCA using the vst transformed batch corrected data

```
pca3=prcomp(t(newvstCounts), scale=F)
# colour by batch
par(mfrow=c(1,2))
plot(pca3$x[,1:2], col=SampleInfo$Batch, pch=as.numeric(SampleInfo$DCISIDC), main="coloured by batch")
```

```
plot(pca3$x[, 1:2], col=SampleInfo$Subtype, pch=as.numeric(SampleInfo$Batch), main="coloured by subtype")
```



```
# colour 3d
plot3d(pca3$x[, 1], pca3$x[, 2], pca3$x[, 3], radius=15, type="s",
        col=as.numeric(SampC), xlab = "PC1", ylab = "PC2", zlab="PC3")
rgl.postscript("output/All_samples_pca_vst_transform_DESEQ.pdf", fmt = "pdf")
```

### 3.6 Differential Gene expression analysis

Run DeSeq2 adding in batch information into the model:

```
dds=DESeq(dds, betaPrior=F)
DEresC1=results(dds, contrast=c("Subtype", "HER2", "TN"))
DEresC2=results(dds, contrast=c("Subtype", "Str", "Org"))

# Run to find DCIS vs IDC
dds1 <- DESeqDataSetFromMatrix(countData = GEnewfilt, colData = SampleInfo, design=~DCISIDC+factor(Batch))
dds1=DESeq(dds1, betaPrior=F)
DEresA1=results(dds1, contrast=c("DCISIDC", "DCIS", "IDC"))
DEresA2=results(dds1, contrast=c("DCISIDC", "DCIS", "N"))
DEresA3=results(dds1, contrast=c("DCISIDC", "IDC", "N"))
```

### 3.7 Figure 2B: Heatmap of differentially expressed genes

How do the gene sets intersect with the EdgeR analysis? Have a look at HER2 vs Basal for example:

```
# HER2 vs Basal
summary(DEresC1)
```

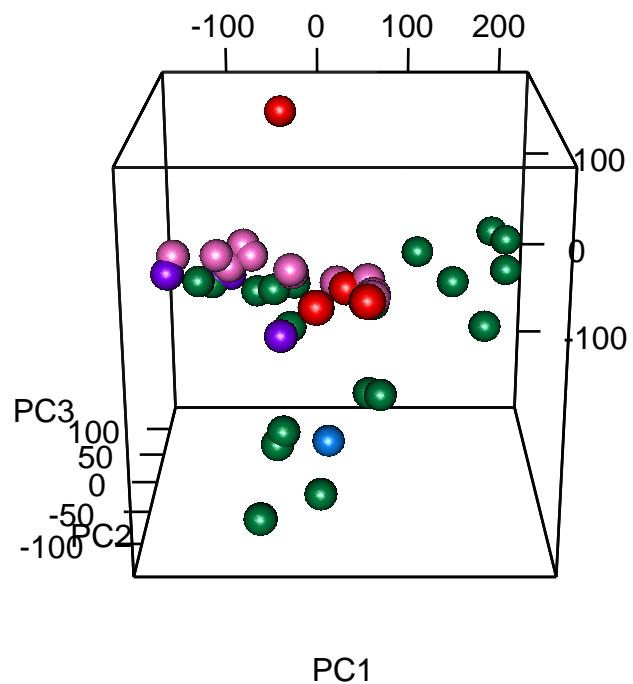


Figure 2: PCA of batch adjusted data

```

## 
## out of 21859 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)      : 1480, 6.8%
## LFC < 0 (down)    : 924, 4.2%
## outliers [1]       : 3411, 16%
## low counts [2]     : 2242, 10%
## (mean count < 3)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
GenesDiff=rownames(DEresC1)[which(DEresC1$padj<0.05)]
# DCIS vs IDC
summary(DEresA1)

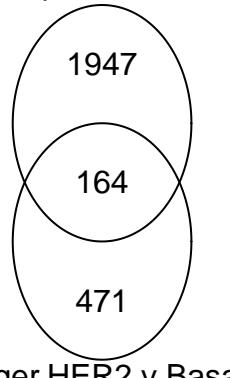
```

```

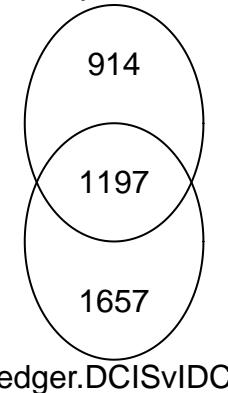
## 
## out of 21859 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)      : 1011, 4.6%
## LFC < 0 (down)    : 1990, 9.1%
## outliers [1]       : 4829, 22%
## low counts [2]     : 2168, 9.9%
## (mean count < 3)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
GenesDiff=rownames(DEresA1)[which(DEresA1$padj<0.05)]
par(mfrow=c(1,2))
venn(list(edger.HER2.v.Basal=rownames(tabCS), DESeq.HER2.v.Basal=GenesDiff))
venn(list(edger.DCISvIDC=rownames(tabA), DESeq.DCISvIDC=GenesDiff))

```

DESeq.HER2.v.Basal



DESeq.DCISvIDC



Approximately half the genes found to be differentially expressed by both DESeq and edger. We can use the intersection of these to construct a heatmap

```

# List of genes differentially expressed in both edger and deseq2
GeneListInt=unique(c(intersect(rownames(tabCS), rownames(DEresC1)[which(DEresC1$padj<0.05)]),
intersect(rownames(tabA), rownames(DEresA1)[which(DEresA1$padj<0.05 & abs(DEresA1$log2FoldChg)>1)]),
intersect(rownames(tabB), rownames(DEresA2)[which(DEresA2$padj<0.05 & abs(DEresA2$log2FoldChg)>1)]),
intersect(rownames(tabC), rownames(DEresA3)[which(DEresA3$padj<0.05 & abs(DEresA3$log2FoldChg)>1)])

```

```

RowSideColA=factor(CombinedGeneList$Fn[match(GeneListInt, CombinedGeneList$Symbol)])
RowSideColA=as.character(RowSideColA)

```

```

# all genes
# intersection of immune only
x1=which(GeneListInt%in%CombinedGeneList$Symbol)

# fix up category names
RowSideColA=factor(RowSideColA[x1])
RowSideColA2=as.character(RowSideColA)
RowSideColA2[grep("TCR", RowSideColA)]="TCR"
RowSideColA2[grep("IL[0-9]", RowSideColA)]="INTERLEUKIN"
RowSideColA2[grep("Interleukin", RowSideColA)]="INTERLEUKIN"
RowSideColA2[grep("B_CELL", RowSideColA)]="BCR"
RowSideColA2[grep("B_LYMPHOCYTE", RowSideColA)]="BCR"
RowSideColA2[grep("IFN", RowSideColA)]="Interferons"
RowSideColA2[grep("INFECTION", RowSideColA)]="Antimicrobials"
RowSideColA2[grep("TGF", RowSideColA)]="TGF"
RowSideColA2[grep("TNF", RowSideColA)]="TNF"
RowSideColA2[grep("PDGF", RowSideColA)]="PDGF"
RowSideColA2[grep("COMPLEMENT", RowSideColA)]="PDGF"
RowSideColA2[grep("BCR", RowSideColA)]="BCR"
RowSideColA2[grep("CCR", RowSideColA)]="Cytokines"
RowSideColA2[grep("Cytokine", RowSideColA)]="Cytokines"
RowSideColA2[grep("INSULIN", RowSideColA)]="Cytokines"
RowSideColA2[grep("FC", RowSideColA)]="Antimicrobials"
RowSideColA3=RowSideColA2
RowSideColA2=factor(RowSideColA2)
RowSideColA2[RowSideColA2=="NA"]="NA

l1=which(duplicated(CombinedGeneList$Symbol))

lList=unique(CombinedGeneList$Symbol[l1])
GRef=rep(NA, length(lList))

for(i in 1:length(lList)){
  l11=which(CombinedGeneList$Symbol==lList[i])
  GRef[i]=paste(na.omit(unique(CombinedGeneList$Fn[l11])), collapse=", ")
}

NewList=data.frame(Symbol=c(as.character(lList),as.character(CombinedGeneList$Symbol[-l1])), Fn=c(as.cha

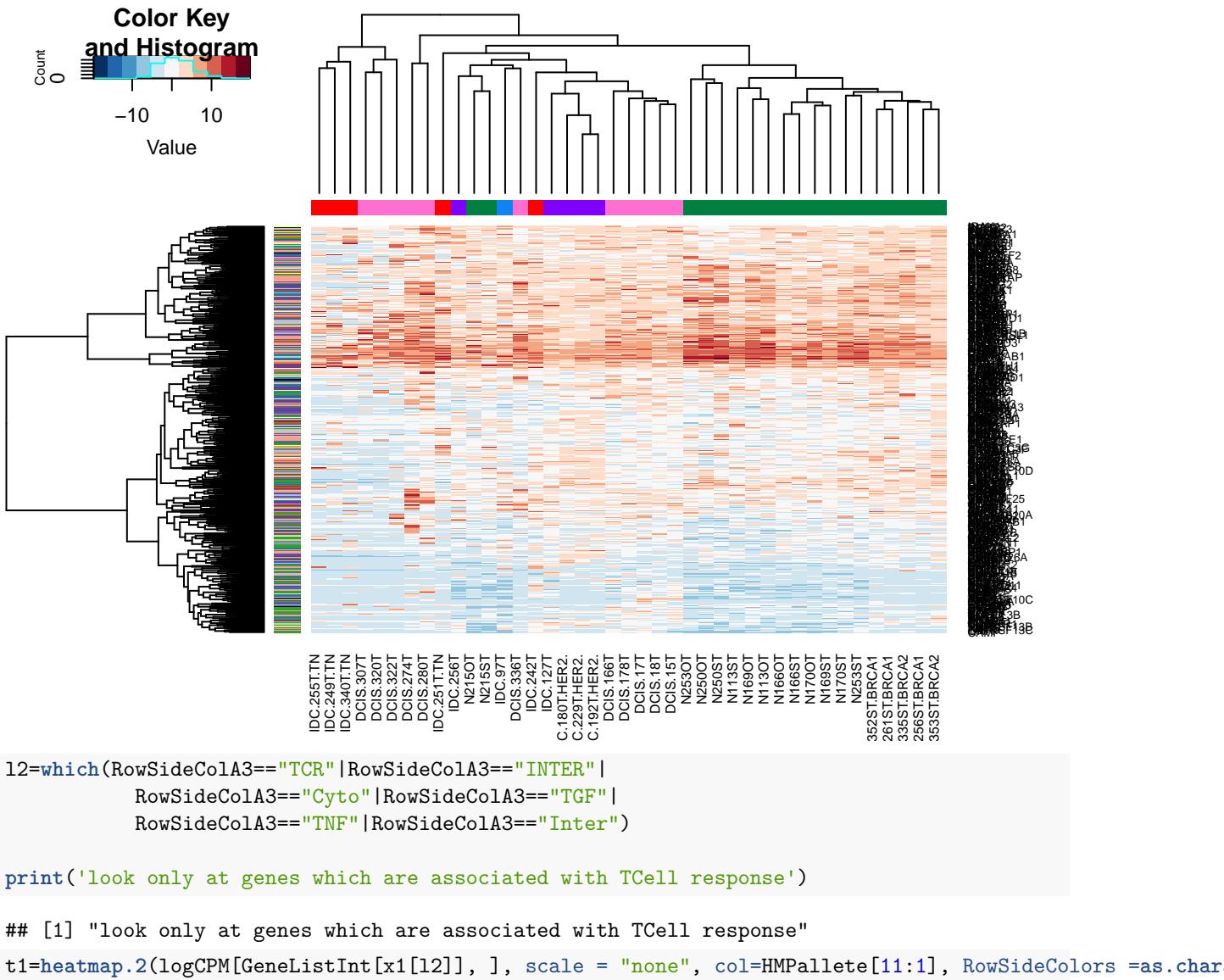
l2=match(rownames(Gnewfilt), NewList$Symbol)
temp>NewList[12,]

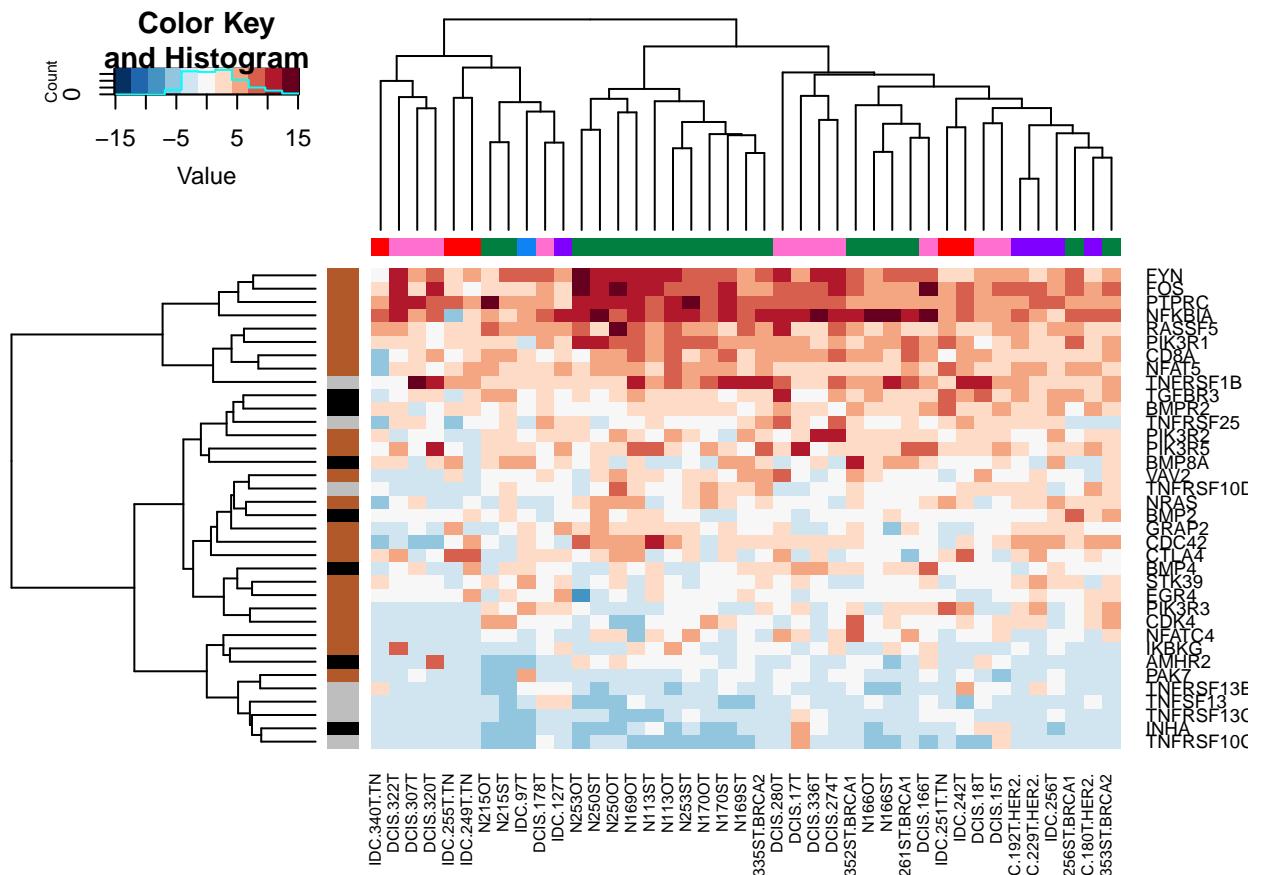
write.csv(temp, file="output/immune_annot.csv")

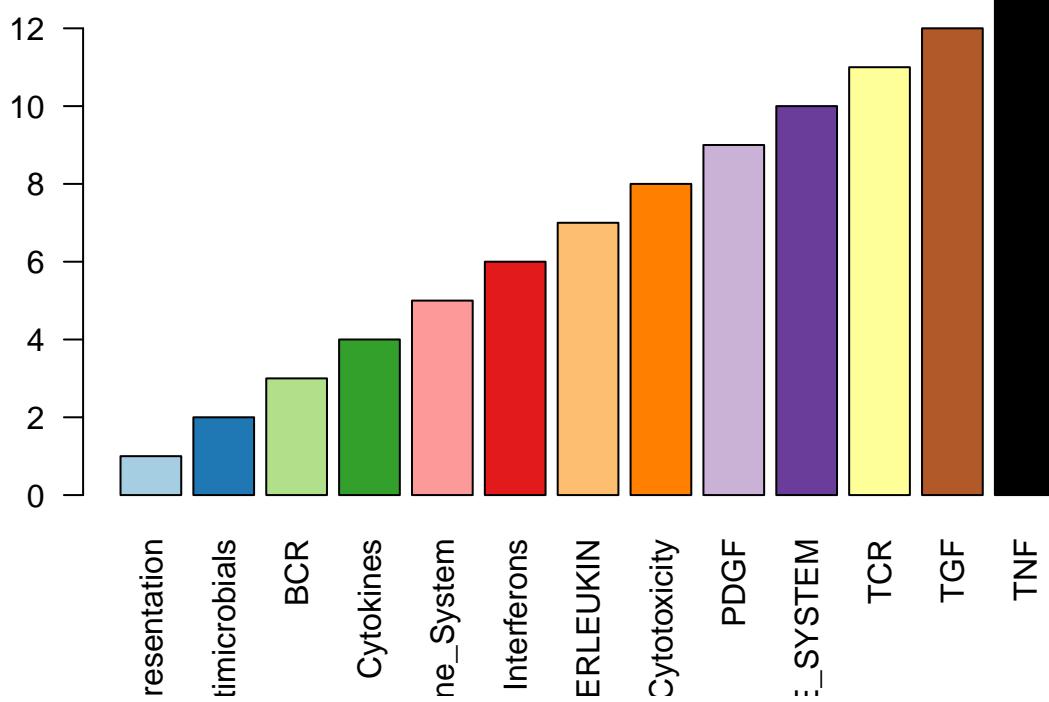
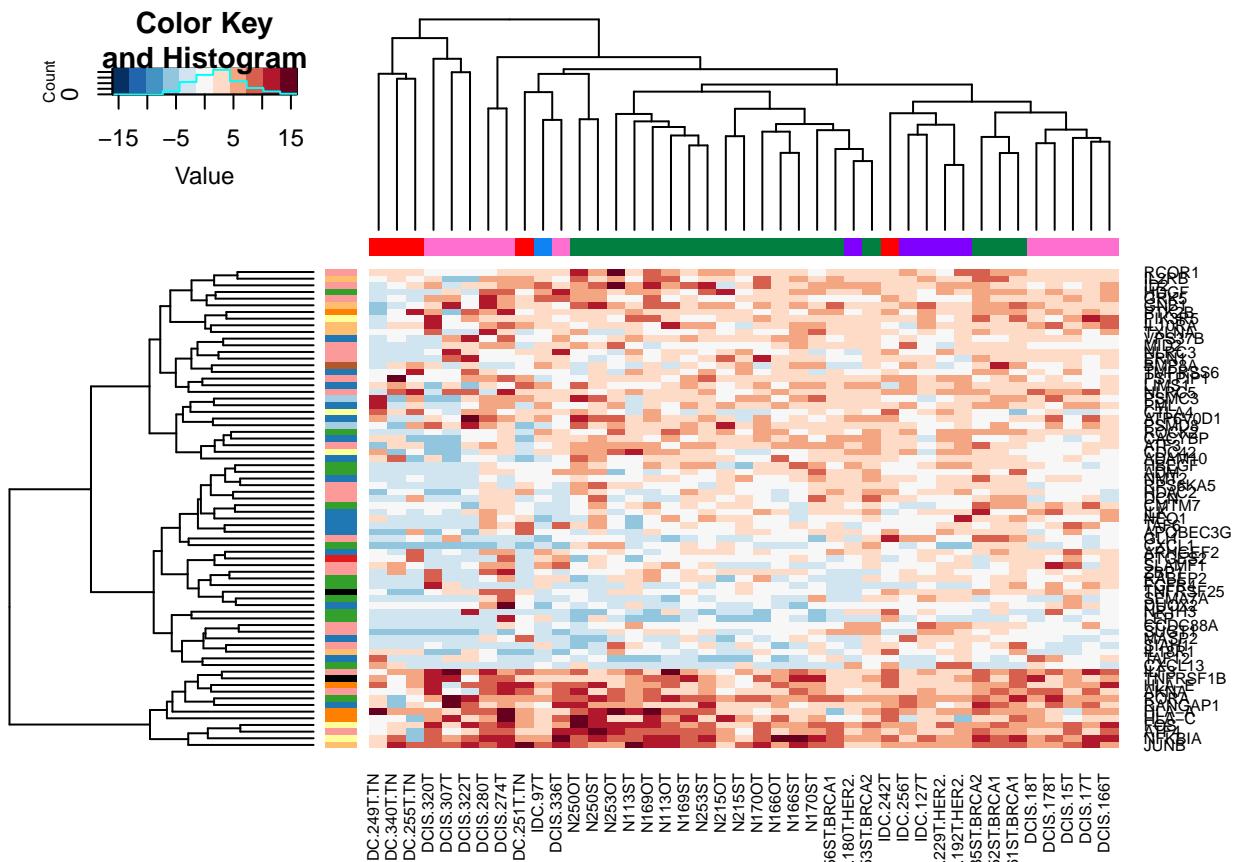
tol14rainbow=c(brewer.pal(12, "Paired"), "black", "grey")
levels(RowSideColA2)=tol14rainbow

heatmap.2(logCPM[GeneListInt[x1], ], scale = "none", col=HMPallete[11:1], ColSideColors = ColSide, trace

```







## 4 Gene Set enrichment analysis: Figure 2

### 4.1 GSEA in HTSanalyzeR

Gene set enrichment analysis is performed using log fold-changes from differential gene expression analysis (from edger analysis).

The set-up is shown below for DCIS vs IDC, this is repeated for all other comparisons: Here it is run with 10 permutations for speed, but run with 100 permutations in this analysis.

```
# define the gene sets
GSEAList=list(Immune=immunec7listentrez, Canonical=Canlistentrez)
# DCIS VS IDC
# Supply phenotypic data (PhData) and hits (differentially expressed genes)
Phdata=tabAall$table$logFC
hits=rownames(tabA)
names(Phdata)=rownames(tabAall)

gscaDvsI=new("GSCA", listOfGeneSetCollections=GSEAList, geneList=Phdata, hits=hits)
gscaDvsI=preprocess(gscaDvsI, species="Hs", initialIDs="Symbol", keepMultipleMappings=T,
                     duplicateRemoverMethod='average')
## Change number of Permutations to 100 for real simulation
gscaDvsI<-analyze(gscaDvsI,para=list(nPermutations=10, minGeneSetSize=10,pValueCutoff = 0.05,pAdjustMet
```

Plot the output for this:

```
load("output/GSEA/example_gsea_DCIS_vs_IDC.RData")
HTSanalyzeR::summarize(gsca_DvI)

##
## -No of genes in Gene set collections:
##           input above min size
## Immune      4872          4872
## Canonical   1330          1286
##
##
## -No of genes in Gene List:
##           input valid duplicate removed converted to entrez
## Gene List 22196 22196          22196          22180
##
##
## -No of hits:
##           input preprocessed
## Hits     2840          2835
##
##
## -Parameters for analysis:
##           minGeneSetSize pValueCutoff pAdjustMethod
## HyperGeo Test 10          0.05        BH
##
##           minGeneSetSize pValueCutoff pAdjustMethod nPermutations exponent
## GSEA 10          0.05        BH          100          1
##
##
## -Significant gene sets (adjusted p-value < 0.05 ):
```

```

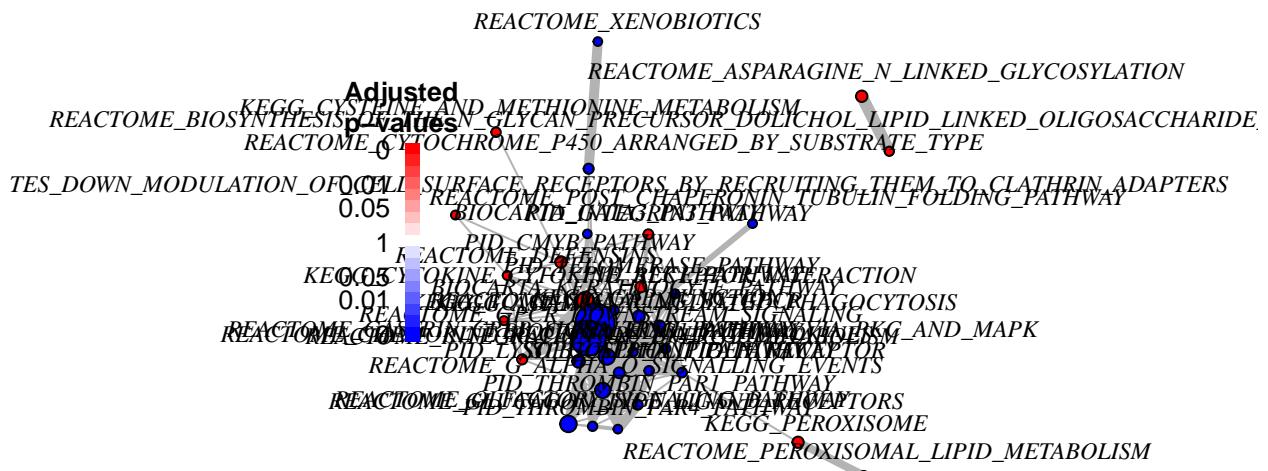
##             Immune Canonical
## HyperGeo      358      20
## GSEA         353      38
## Both          39       0

topTerms_DvsI=getTopGeneSets(gscaD_DvI, "GSEA.results", c("Immune", "Canonical"), allSig=TRUE)

viewEnrichMap(gscaD_DvI, resultName="GSEA.results", "Canonical", ntop=35, allSig=F, gsNameType="id",
              displayEdgeLabel=F)

```

## Enrichment Map of GSEA on "Canonical"

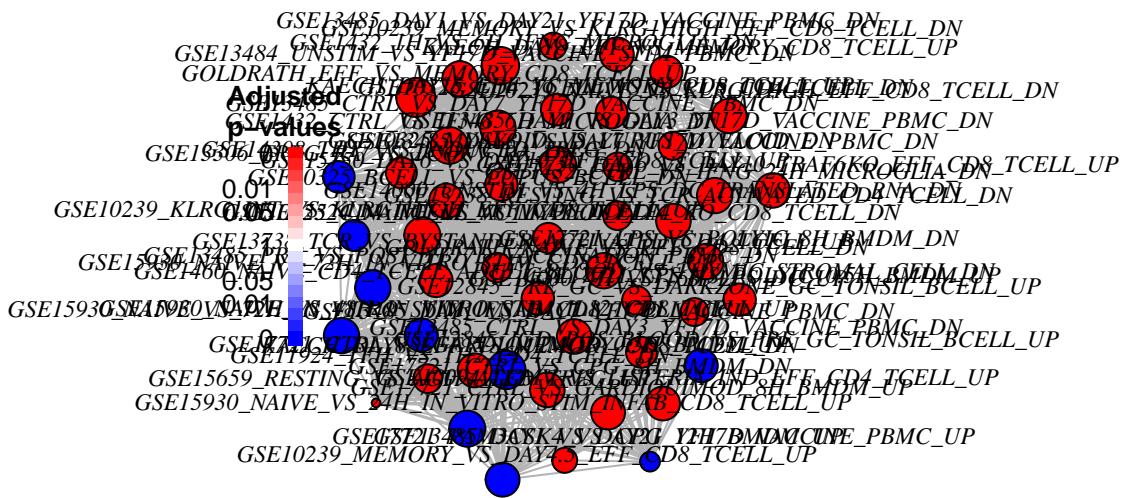


```

## IGRAPH UNW- 35 197 --
## + attr: name (v/c), geneSetSize (v/n), size (v/n), color (v/c),
## | label (v/c), label.dist (v/n), label.cex (v/n), label.font
## | (v/n), label.color (v/c), weight (e/n), color (e/c), width (e/n)
## + edges (vertex names):
## [1] Canonical.KEGG_CYSTEINE_AND METHIONINE_METABOLISM --Canonical.PID_CMYB_PATHWAY
## [2] Canonical.KEGG_CYTOKINE_CYTOKINE_RECECTOR_INTERACTION--Canonical.KEGG_GAP_JUNCTION
## [3] Canonical.KEGG_CYTOKINE_CYTOKINE_RECECTOR_INTERACTION--Canonical.KEGG_ASTHMA
## [4] Canonical.KEGG_CYTOKINE_CYTOKINE_RECECTOR_INTERACTION--Canonical.BIOCARTA_KERATINOCYTE_PATHWAY
## + ... omitted several edges
viewEnrichMap(gscaD_DvI, resultName="GSEA.results", "Immune", ntop=50, allSig=F, gsNameType="id",
              displayEdgeLabel=F)

```

## Enrichment Map of GSEA on "Immune"



```

## IGRAPH UNW- 50 1125 --
## + attr: name (v/c), geneSetSize (v/n), size (v/n), color (v/c),
## | label (v/c), label.dist (v/n), label.cex (v/n), label.font
## | (v/n), label.color (v/c), weight (e/n), color (e/c), width (e/n)
## + edges (vertex names):
## [1] Immune.KAECH_DAY8_EFF_VS_MEMORY_CD8_TCELL_UP--Immune.KAECH_DAY15_EFF_VS_MEMORY_CD8_TCELL_UP
## [2] Immune.KAECH_DAY8_EFF_VS_MEMORY_CD8_TCELL_UP--Immune.GOLDRATH_NAIVE_VS_EFF_CD8_TCELL_DN
## [3] Immune.KAECH_DAY8_EFF_VS_MEMORY_CD8_TCELL_UP--Immune.GOLDRATH_EFF_VS_MEMORY_CD8_TCELL_UP
## [4] Immune.KAECH_DAY8_EFF_VS_MEMORY_CD8_TCELL_UP--Immune.GSE10239_NAIVE_VS_KLRG1HIGH_EFF_CD8_TCELL_D
## + ... omitted several edges

```

Note that some immune terms: Toll-like receptor, NK cells, appear to be enriched.

For all samples, the following function will convert the GSEA objects into plots:

```
# write a function to make the plots, and save the output files from the analysis  
# Note that all the functions to print to file have been commented out
```

```

SaveGSEAData=function(gsea, FnName="expt1"){
  dir.create(FnName)
  # file lists:
  A1=gsea@result$GSEA.results$Canonical
  D1=gsea@result$GSEA.results$Immune
  write.csv(A1, file=sprintf("%s/GSEA_Canonical_%s.csv",FnName, FnName))
  write.csv(D1, file=sprintf("%s/GSEA_Immune_%s.csv",FnName, FnName))

  # get the number of sig gene sets in each:
  toptabSum=getTopGeneSets(gsea, "GSEA.results", c("Immune","Canonical"), allSig=T)
  nGenes=as.numeric(summary(toptabSum)[ ,1])
  GCatNames=c("IFN", "IL[0-9]", "IL_[0-9]", "TNF", "CYTOKINE", "NKCELL", "ANTIGEN", "CHEMOKINE", "TCR",
             "BCR", "CYTOTOXIC", "TGF", "IMMUNE", "TH1", "STAT[0-9]", "INFLAMMATORY", "COMPLEMENT",
             "COAGULATION", "ALLOGRAFT", "INTERFERON", "DISEASE", "IGG", "BCELL", "TCELL", "LYMPHOCYTE",
             "CTL", "TH2", "GRANZYME", "NEUTROPHIL", "T_CELL", "LEUKOCYTE", "B_CELL", "TREG", "INFLUEN",
             "FC_", "IGF", "CLASSIC", "COMP_", "THROMBIN", "DEFENSIN", "ASTHMA", "LUPUS", "PDGF")
  Metab=c("PENTOSE", "METAB", "GLYCO", "GLUC", "BIOSYNTHESIS")
  DNAnames=c("DNA", "G1", "G2", "MITO", "MEIO", "REPLIC", "REPAIR", "CELL CYCLE", "TELOMER", "CENTROSOM

```

```

RNAnames=c("RNA", "TRANSCRIPT", "POL", "PROMOTER", "SPLIC", "EXON", "INTRON")

Cancer=c("CANCER", "CARCIN", "_RB_", "FGFR", "ERBB", "AURORA", "TP53", "ALK")
Translat=c("TRANSLATION", "RIBOSOME", "43S", "PROTEOL", "PEPTIDE", "EIF", "GOLGI", "PROTEIN")
Apop=c("APOPTOSIS", "DEATH")
Signal=c("SECRET", "SIGNAL", "SYNAP", "RECEPTOR", "INTEGRIN", "TRANSDUCTION", "RAB", "NEUROTRANSMIT")
Elect=c("ELECTRON", "OXID", "ATP", "CYTOCHROM", "PEROXI")
ECM=c("MATRISOME", "MATRIX", "EXTRACELLULAR", "KERATIN", "MUSCLE")

TermIdx=list( DNA=DNAnames, RNA=RNAnames, Cancer=Cancer, Protein=Translat,
             Metab=Metab, Secretion=Signal, Apoptosis=Apop, Electron=Elect, ECM=ECM, Imm=GCatNames)
ColID=brewer.pal(12, "Paired")

NamesIdx=c("Immune", "Canonical")
for (i in 1:2){
  Plot1=viewEnrichMap(gsea, resultName="GSEA.results", NamesIdx[i], ntop=nGenes[i], allSig=F, gsNameTyp
                      displayEdgeLabel=F)
  l=layout.fruchterman.reingold(Plot1)
  vertex_attr(Plot1)$label.cex=rep(0.5, nGenes[i])
  vertex_attr(Plot1)$label.dist=rep(0.25, nGenes[i])
  temp1=unlist(regmatches(vertex_attr(Plot1)$label, regexpr("_", vertex_attr(Plot1)$label), invert = TR
  temp1=temp1[seq(2, length(temp1), by=2)]
  vertex_attr(Plot1)$label=temp1
  a1=unique(unlist(sapply(GCatNames, function(x) grep(x, vertex_attr(Plot1)$label))))
  vertex_attr(Plot1)$label.color=rep("darkgrey", nGenes[i])
  if (length(a1)>0){
    vertex_attr(Plot1)$label.color[a1]="purple"
  }
# pdf(file = sprintf("%s/GSEA_shortenedNames_%s_%s.pdf", FnName, FnName, NamesIdx[i]))
# plot(Plot1, layout=l)
# dev.off()
  idxNew=lapply(TermIdx, function(x) unlist(sapply(x, function(y) grep(y, vertex_attr(Plot1)$label))))
  idxNew2=sapply(idxNew, unique)
  idxSum=sapply(idxNew2, length)
  idxSum=c(idxSum, nGenes[i])
  write(idxSum, file=sprintf("%s/GSEA_enrichedno_%s_%s.csv", FnName, FnName, NamesIdx[i]))
  vertex_attr(Plot1)$color=rep("black", length(vertex_attr(Plot1)$label))
  for (j in 1:length(idxNew2)){
    vertex_attr(Plot1)$color[unlist(idxNew2[j])]=ColID[j]
    vertex_attr(Plot1)$label.color[unlist(idxNew2[j])]=ColID[j]
  }
#pdf(file = sprintf("%s/GSEA_coloured_%s_%s.pdf", FnName, FnName, NamesIdx[i]))
# plot(Plot1, layout=l)
#dev.off()
  tNames=vertex_attr(Plot1)$label[unlist(idxNew2[10])]
  vertex_attr(Plot1)$label=rep("", nGenes[i])
  vertex_attr(Plot1)$label[unlist(idxNew2[10])]=tNames
  vertex_attr(Plot1)$label.color=rep("black", length(vertex_attr(Plot1)$label))
#pdf(file = sprintf("%s/GSEA_Nameless_%s_%s.pdf", FnName, FnName, NamesIdx[i]))
  plot(Plot1, layout=l)
#dev.off()
}
}

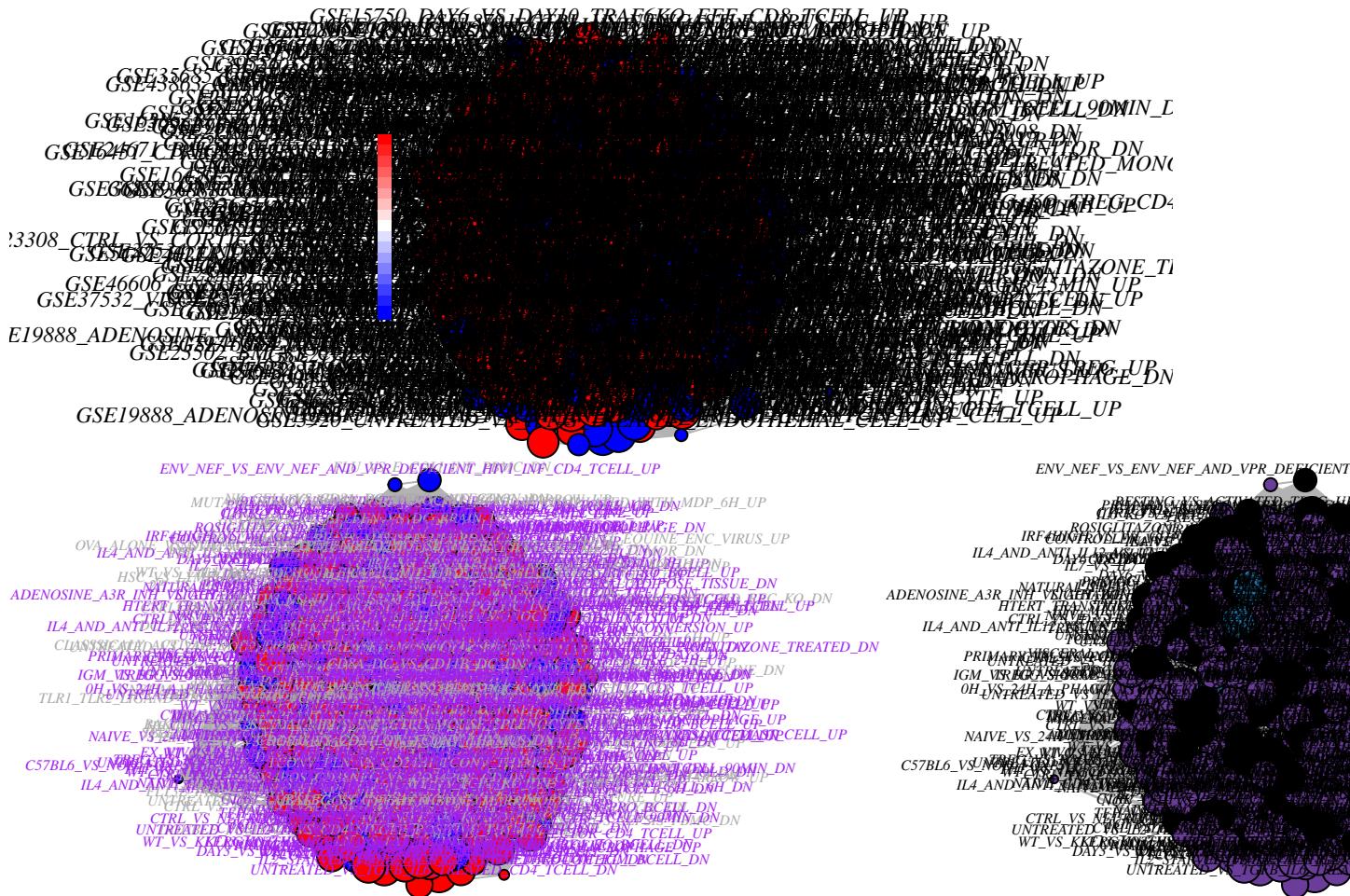
```

Using previously run GSEA analysis, we can visualise the output:

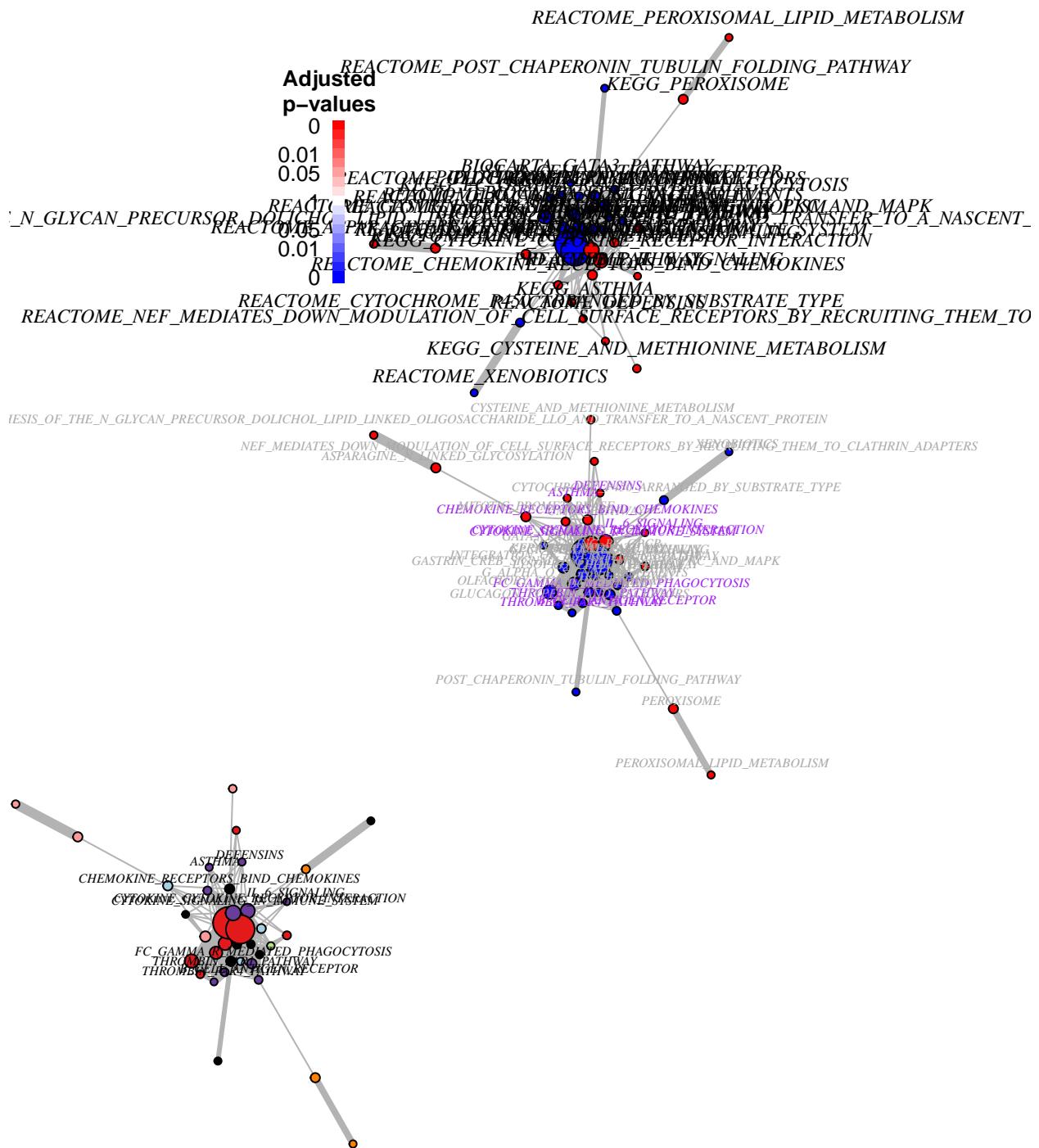
```
SaveGSEAData(gscaD_DvI, "DCISvsIDCtest")
```

## Warning in dir.create(FnName): 'DCISvsIDCtest' already exists

## Enrichment Map of GSEA on "Immune"



## Enrichment Map of GSEA on "Canonical"



### 4.2 Immune c7

Note that the immune c7 analysis provides many inter-connected terms. These are representative of studies which compare condition A to B, and the top 100 differentially expressed genes between the two scenarios are

reported as a gene set. This is directional, ie, genes which are more highly expressed in A belong to the “A vs B UP” set, whereas genes more highly expressed in B belong to the “A vs B DOWN” set.

A and B contains information on cell type (eg. B Cells, CD4 cells), activation state and experimental treatment. A separate .Rmd file is provided to create plots based on outputs from GSEA (`summary-immunec7.Rmd`). Examples of plots from DCIS and IDC comparisons are shown below:

#### 4.2.1 Load data of interest

```
GSEArresults=read.csv("output/GSEA/GSEA_immunec7_DCISvsNormal.csv")
PValCutOff=0.05
CompA="DCIS"
CompB="Normal"
```

Note that the script `ImmuneC7Output.R` can be used to derive more information from the gene sets.

```
# Load the Gene Set summary information and index:
GSInfo=read.csv("data/c7_immune_annotated.csv")
NID=GSEArresults$X[which(GSEArresults$Adjusted.Pvalue<PValCutOff)]
NIDmatch=match(NID, GSInfo$GeneSet)
temp=GSInfo[NIDmatch, ]
```

The immune c7 data set was further curated: For example below, the Gene set was separated into GSE number which was used to look up information such as organism, and source of T-cells. Also determined was the celltype (eg. CD4 T cell) and the treatment applied (eg IL3/HIVP17 and LISTERIA). Stimulation refers to whether they are activated, naive or exhausted:

```
GSInfo[8,1:15 ]
```

```
##   X                   GeneSet      GSE      GrpA
## 8 8 GSE10094_LCMV_VS_LISTERIA_IND_EFF_CD4_TCELL_UP GSE10094 CD4_TCELL
##   TreatA StimA     GrpB   TreatB   StimB CellType Organ TimeCourse
## 8   LCMV <NA> CD4_TCELL LISTERIA EFFECTOR CD4_TCELL <NA> <NA>
##   Treatment   Organism           Source
## 8 LCMVvLISTERIA Mus musculus SMARTA TCR transgenic T cells
```

We can have a quick look at the directionality of the different comparison. Here, we compare average DCIS vs Normal expression in samples in gene sets which are of the format A vs B up and down.

```
l1=which(GSEArresults$Adjusted.Pvalue<0.05)
pos1=which(GSEArresults$Observed.score[11]>0)
neg1=which(GSEArresults$Observed.score[11]<0)
up1=grep("UP", temp$GeneSet[11])
down1=grep("DN", temp$GeneSet[11])

List=c(intersect(pos1, up1)[1], intersect(pos1, down1)[1],
       intersect(neg1, up1)[1], intersect(neg1, down1)[1])

GSNames=cbind(GeneSet=as.character(temp$GeneSet[l1[List]]), Enrichment=GSEArresults$Observed.score[11[Li]
```

We will look at the above four gene sets as examples to assess the directionality in DCIS compared to Normal:

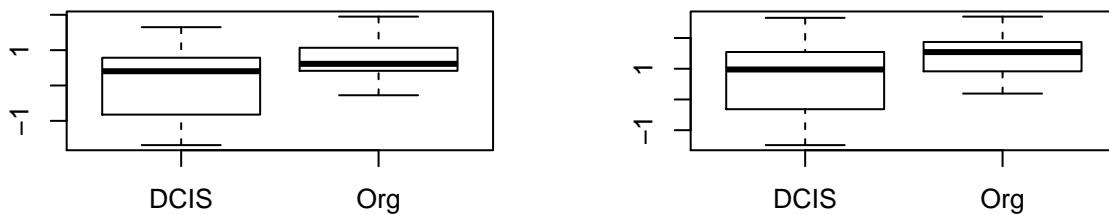
```
subidx=which(SampC%in%c("DCIS", "Org"))
par(mfrow=c(2,2))
for (i in List){
  ImmSet=temp[i, "GeneSet"]
  x2=which(names(immunec7entrez)==ImmSet)
```

```

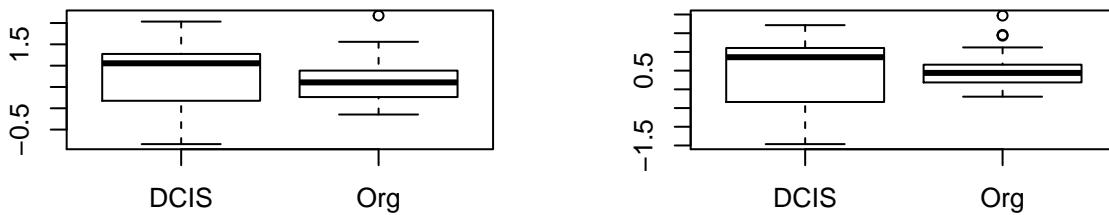
GeneListNew=geneIds(immunec7[[x2]])
m2=colMeans(logCPM[na.omit(match(GeneListNew, rownames(logCPM))), subidx])
boxplot(m2~factor(SampC[subidx]), main=sprintf("%s%g", ImmSet, round(GSEArresults$Observed.score[l1[i]]*}
}

```

## H\_DAY8\_EFF\_VS\_DAY15\_EFF\_CD8\_TCEICH\_NAIVE\_VS\_DAY15\_EFF\_CD8\_TCELL



## 2845\_IGD\_POS\_VS\_NEG\_BLOOD\_BCELISE13229\_IMM\_VS\_MATURE\_NKCELL\_DM



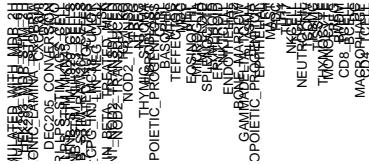
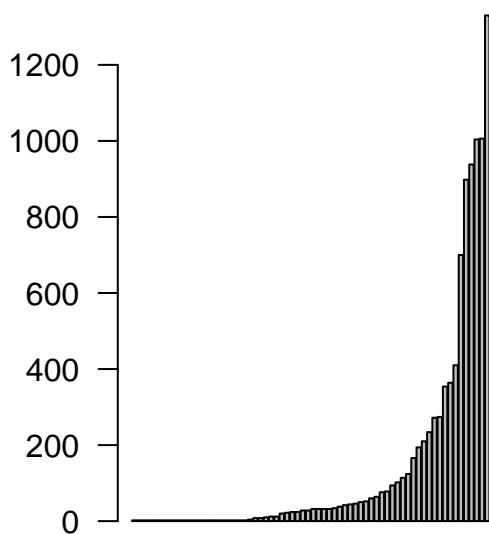
A positive enrichment score means group B (Normal) has higher mean expression compared to the reference (A, DCIS). Also, the gene set directionality is flipped depending on whether a group is “up” or “down”. in C vs D up, group B would be more similar to C.

### 4.3 Cell types enriched

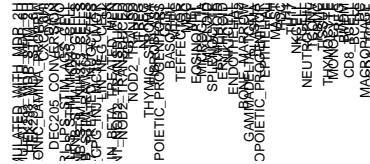
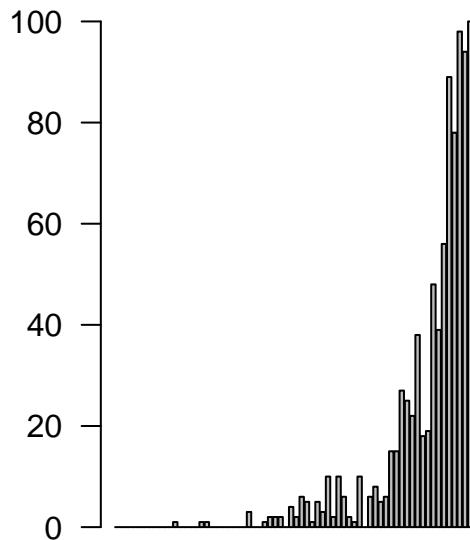
Compare the types of cells (leukocytes) studied in each gene set:

- Overall refers to the overall distribution of gene sets featuring a particular cell type of interest.
- Enriched refers to cell types in gene sets with P-value less than that specified (ie. significant results from GSEA)

**Overall**



**Enriched**



#### 4.3.1 Activation, memory, exhaustion: Fig 2D

Below a function was created to develop a network of the “interaction” between different cell types.

```
ActivationNetwork=function(A, B, GSEAscore, CompA, SearchTerms=NULL){
  library(igraph)
  nx=which(GSEAscore<0)
  df=data.frame(StimA=A, StimB=B)
  if (length(nx)>0){
    A2=df$StimA[nx]
    B2=df$StimB[nx]
    df$StimA[nx]=B2
    df$StimB[nx]=A2
  }
  if (!is.null(SearchTerms)){
    a1=sapply(SearchTerms, function(x) grep(x, df[ ,1]))
    b1=sapply(SearchTerms, function(x) grep(x, df[ ,2]))
    xint=intersect(unlist(a1), unlist(b1))
    df=df[xint, ]
  }

  # look for na values
  naidx=which(is.na(df$StimA) | is.na(df$StimB))
  if (length(naidx)>0){
    df=df[-naidx, ]
  }
}
```

```

allList=factor(c(as.character(df$StimA), as.character(df$StimB)))
df$StimA=factor(df$StimA, levels=levels(allList))
df$StimB=factor(df$StimB,levels=levels(allList))

gr0=graph.edgelist(data.matrix(df), directed=T)
gr0=igraph::simplify(gr0, remove.loops = T, remove.multiple = F)
deg=igraph::degree(gr0, mode="all")
E(gr0)$weight=1
V(gr0)$size=(deg+1)
V(gr0)$color="white"
gr0=igraph::simplify(gr0, remove.multiple = T, edge.attr.comb="sum")
V(gr0)$label=levels(df$StimA)
E(gr0)$width=E(gr0)$weight*2/3

## bug here
edge.start <- ends(gr0, es=E(gr0), names=F)
Ecol=rep("pink", nrow(edge.start))
tabOut=table(df)
tabOut2=tabOut-t(tabOut)
Ecol2=sapply(1:nrow(edge.start), function(x) ifelse(tabOut2[edge.start[ x,1],edge.start[ x,2]]>0, "pink",
Ewidth2=sapply(1:nrow(edge.start), function(x) tabOut2[edge.start[ x,1],edge.start[ x,2]])
##
E(gr0)$edge.color=Ecol2
E(gr0)$width=Ewidth2
gr02=delete.edges(gr0, E(gr0)[Ecol2=="green"])
#browser()
V(gr02)$size=rowSums(abs(tabOut2))+1

plot(gr02, layout=layout_in_circle(gr02),vertex.label.color="black",
      edge.color=E(gr02)$edge.color,vertex.label.font=2,
      vertex.label.color="gray", vertex.label.cex=.7, edge.curved=0.1, mark.border=NA,
      edge.arrow.mode=2, edge.arrow.size=1,
      main=sprintf ("%s", CompA))

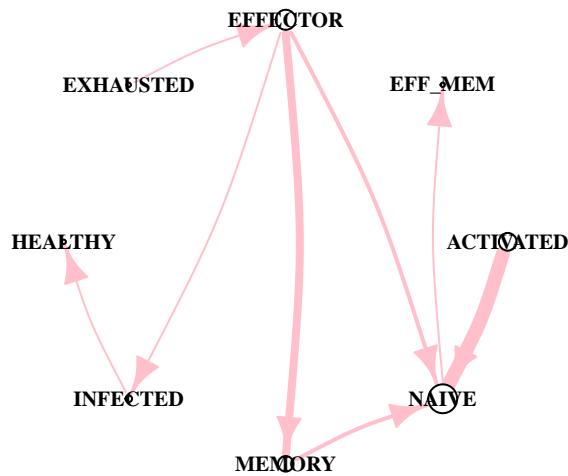
#plot(gr02, layout=layout_in_circle(gr02),vertex.label.color="black",
#      edge.color="gray",vertex.label.font=2,
#      vertex.label.color="gray", vertex.label.cex=.7, edge.curved=0.1, mark.border=NA,
#      edge.arrow.mode=3, edge.arrow.size=1,
#      main=sprintf ("%s", CompA))

if (length(naidx)>0){
ret=list(graph=gr02, data=cbind(df, GSEAscore[-naidx]))
} else {
ret=list(graph=gr02, data=df)
}
ret
}

a1=ActivationNetwork(temp$StimA,temp$StimB, GSEArtsults$Observed.score[which(GSEArtsults$Adjusted.Pvalue

```

## DCIS v Normal



The node size is reflective of the number of times a stimulation type appears, and the width of lines is reflective of the number of gene sets supporting that ‘interaction’. For example, in the above plot a thick blue line points between activated and naive - suggesting that this interaction is downregulated in the DCIS vs IDC comparison (ie. could be higher in IDC compared to DCIS)

```
# Look at the enriched gene sets
summary(a1$data$StimA)
```

```
## ACTIVATED    EFF_MEM    EFFECTOR EXHAUSTED    HEALTHY    INFECTED    MEMORY
##          12          0          9          1          3          8          5
##      NAIVE
##          10
```

```
summary(a1$data$StimB)
```

```
## ACTIVATED    EFF_MEM    EFFECTOR EXHAUSTED    HEALTHY    INFECTED    MEMORY
##          5          1          3          0          4          8          7
##      NAIVE
##          20
```

*# Also check the type of cells studied*

```
summary(factor(rownames(a1$data), "CellType"))
```

```
##      BMDM CD4_TCELL CD8_TCELL      DC      MAC      NKCELL      TCELL
##          2          2          7          2          2          1          1
##      TCONV THYMOCYTE      NA's
##          1          1         29
```

### 4.3.2 Comparing across different studies: Figure 2C

Load all the results and look at the distribution of cells across different samples:

```
GSEAreults[[1]]=read.csv("output/GSEA/GSEA_immuneC7_DCISvsIDC.csv")
GSEAreults[[2]]=read.csv("output/GSEA/GSEA_immuneC7_DCISvsNormal.csv")
GSEAreults[[3]]=read.csv("output/GSEA/GSEA_immuneC7_IDCvsNormal.csv")
GSEAreults[[4]]=read.csv("output/GSEA/GSEA_immuneC7_HER2vsBasal.csv")
GSEAreults[[5]]=read.csv("output/GSEA/GSEA_immuneC7_Stroma_organoid.csv")
```

```

GSEAreults[[6]]=read.csv("output/GSEA/GSEA_immuneC7_Parity_comparison.csv")
GSEAreults[[7]]=read.csv("output/GSEA/gsca_DCIS_vs_HER2_immuneC7.csv")
GSEAreults[[8]]=read.csv("output/GSEA/gsca_DCIS_vs_Basal_immuneC7.csv")

Ctypes=c("TCELL", "TH", "TREG", "NKTCELL", "TCONV")

Cnames=NA
Ccomp=NA

for(i in 1:6){
  temp1=GSInfo[match(GSEAreults[[i]]$X[which(GSEAreults[[i]]$Adjusted.Pvalue<0.05) ], GSInfo$GeneSet)
  Tnull1=unlist(sapply(Ctypes, function(x) grep(x, temp1$GrpA)))
  Tnull2=unlist(sapply(Ctypes, function(x) grep(x, temp1$GrpB)))
  TnullAll=intersect(Tnull1, Tnull2)
  Tnull=c(as.character(temp1$GrpA[TnullAll]), as.character(temp1$GrpB[TnullAll]))
  Cnames=c(Cnames, Tnull)
  Ccomp=c(Ccomp, rep(i, length(Tnull)))
}

Tnull1=unlist(sapply(Ctypes, function(x) grep(x, GSInfo$GrpA)))
Tnull2=unlist(sapply(Ctypes, function(x) grep(x, GSInfo$GrpB)))
TnullAll=intersect(Tnull1, Tnull2)
Tnull=c(as.character(GSInfo$GrpA[TnullAll]), as.character(GSInfo$GrpB[TnullAll]))

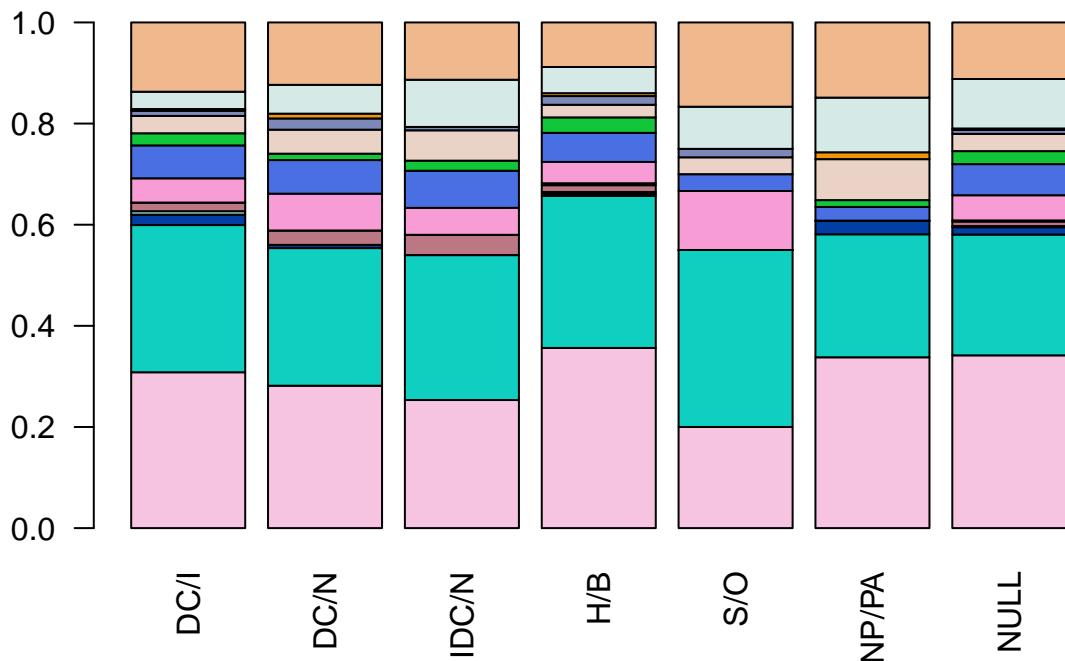
Cnames=c(Cnames, Tnull)
Ccomp=c(Ccomp, rep(7, length(Tnull)))

X1=table(Ccomp, Cnames)
X1=X1/rowSums(X1)
rownames(X1)=c("DC/I", "DC/N", "IDC/N", "H/B", "S/O", "NP/PA", "NULL")

colVals=c("#023FA5", "#7D87B9", "#BEC1D4", "#D6BCC0", "#BB7784", "#FFFFFF", "#4A6FE3", "#8595E1", "#B5BBE3", "#")
colVals=sample(colVals, 24)

barplot(t(X1), col=colVals, las=2)

```



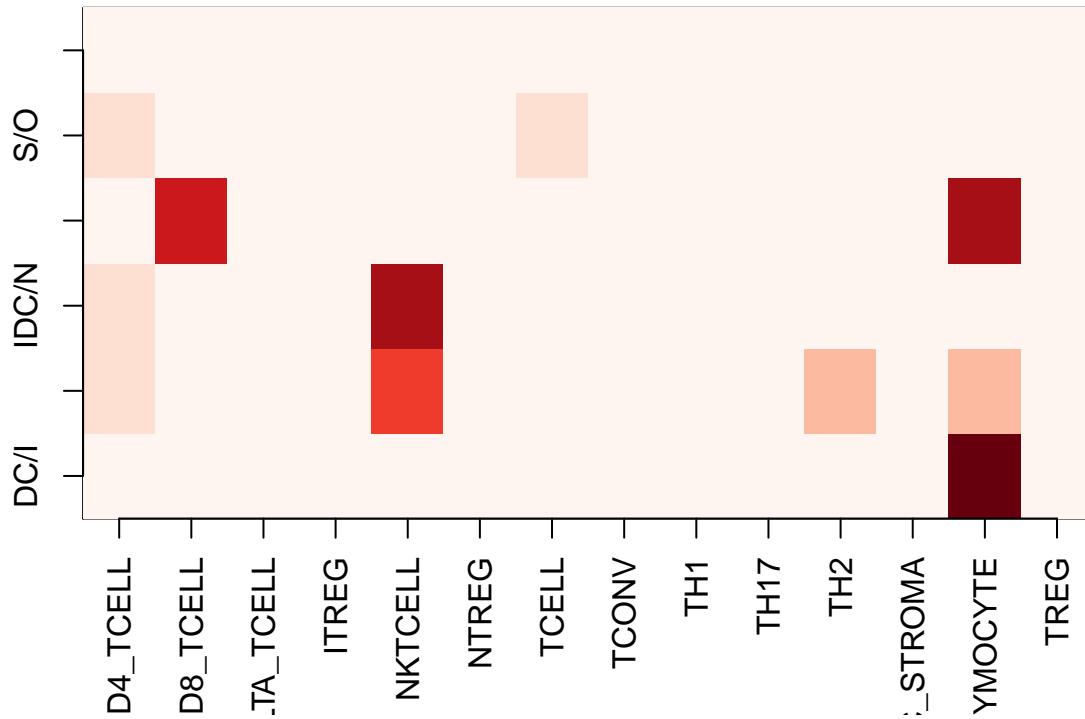
Check for significance across the different cell types

```
L1=table(Ccomp, Cnames)
rownames(L1)=c("DC/I", "DC/N", "IDC/N", "H/B", "S/O", "NP/PA", "NULL")

SigMat=matrix(NA, ncol=ncol(L1), nrow=(nrow(L1)-1))

for (i in 1:6){
  B=c(sum(L1[i,]), sum(L1[7,]))
  for (j in 1:ncol(L1)){
    A=c(L1[i,j], L1[7, j])
    SigMat[i,j]=prop.test(A, B)$p.value
  }
}

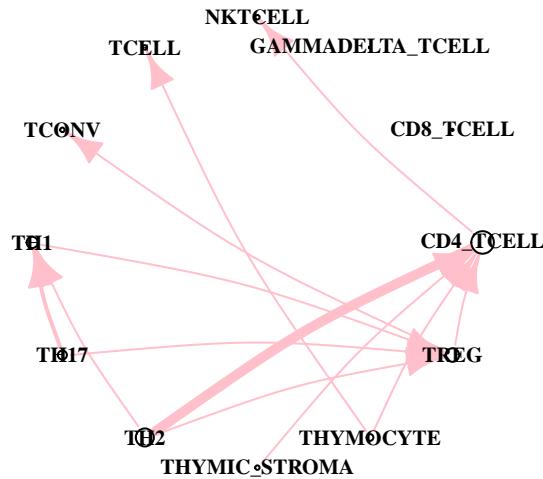
SigMat2=sapply(1:nrow(SigMat), function(x) p.adjust(SigMat[x, ]))
image(-log10(SigMat2), xaxt="n", yaxt="n", col=brewer.pal(9, "Reds"))
axis(1, at=seq(0, 1, length=ncol(L1)), labels=colnames(L1), cex=0.5, las=2)
axis(2, at=seq(0, 1, length=6), labels=rownames(L1)[1:6], cex=0.5)
```



We can also check the types of T cells enriched in these analyses: eg. for DCIS vs IDC

```
SearchTerms=c("TCELL", "TH1", "TH2", "TREG", "TCONV", "THY")
a2=ActivationNetwork(temp$GrpA,temp$GrpB, GSEAre results[[2]]$Observed.score[which(GSEAre results$Adjusted.Pv)
## Warning in Ops.factor(left, right): '<' not meaningful for factors
```

## DCIS



Run the above analysis for all different test sets:

```

pdf("output/GSEA_network_maps.pdf")

CompA=c("DCISvsIDC", "DCISvsNormal", "IDCvsNormal", "HER2vsBasal", "Stroma_organoid.csv", "Parity_compa
#par(mfrow=c(2,2))
par(mfrow=c(1,2))

for (i in 1:length(CompA)){
  PCoff=which(GSEAre results[[i]]$Adjusted.Pvalue<PValCutOff)
  NID=GSEAre results[[i]]$X[PCoff]
  NIDmatch=match(NID, GSInfo$GeneSet)
  temp=GSInfo[NIDmatch, ]

  ActivationNetwork(temp$StimA,temp$StimB, GSEAre results[[i]]$Observed.score[which(GSEAre results[[i]]$Adjusted.
  ActivationNetwork(temp$GrpA,temp$GrpB, GSEAre results[[i]]$Observed.score[which(GSEAre results[[i]]$Adjusted.
}

# Generate a scale matrix to show relative sizes:
m1=factor(c("A",rep("B",3), rep("C",5), rep("D",10), rep("E", 15)))
set.seed(10)
m2=factor(c(sample(c("B","C","D","E"), 1, replace=T), sample(c("C","D","E"), 3, replace=T),
  sample(c("E","D"), 5, replace=T), sample(c("B","C","E"), 10, replace=T),
  sample(c("D"), 15, replace=T)))

levels(m1)=c(1, 4, 11, 24, 20)
levels(m2)=c( 4, 11, 24, 20)
l1=cbind(A=as.factor(m1), B=factor(m2))

out1=ActivationNetwork(m1, m2, rep(1, length(m1)), "scale 1")

dev.off()

## pdf
## 2

```

## 4.4 CIBERSORT

Use CIBERSORT to look at the estimated composition of each sample. This is a deconvolution method based on support-vector regression developed by Neumann et al, Nature Methods 2015 cibersort.stanford.edu.

```

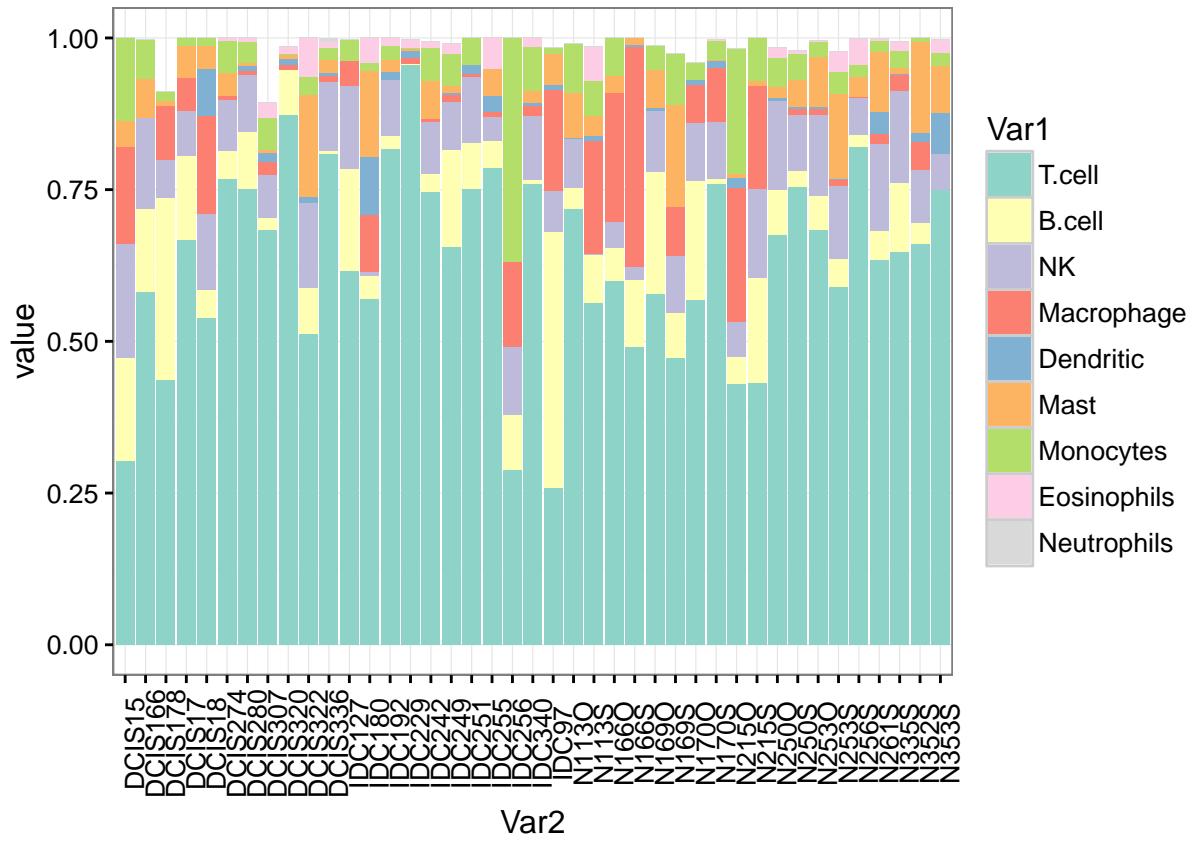
Cres=read.csv("output/GEdata_CIBERSORT_100.csv", header=T, row.names=1)
CresDat=t(Cres[, -c(23:25)])

# do a plot of the different lineages, first grouping by cell type
CellList=c("T.cell", "B.cell", "NK", "Macrophage", "Dendritic", "Mast")

CresComb=t(sapply(CellList, function(x) colSums(CresDat[grep(x, rownames(CresDat))], ))))
CresComb=rbind(CresComb, CresDat[c("Monocytes", "Eosinophils", "Neutrophils") ,])

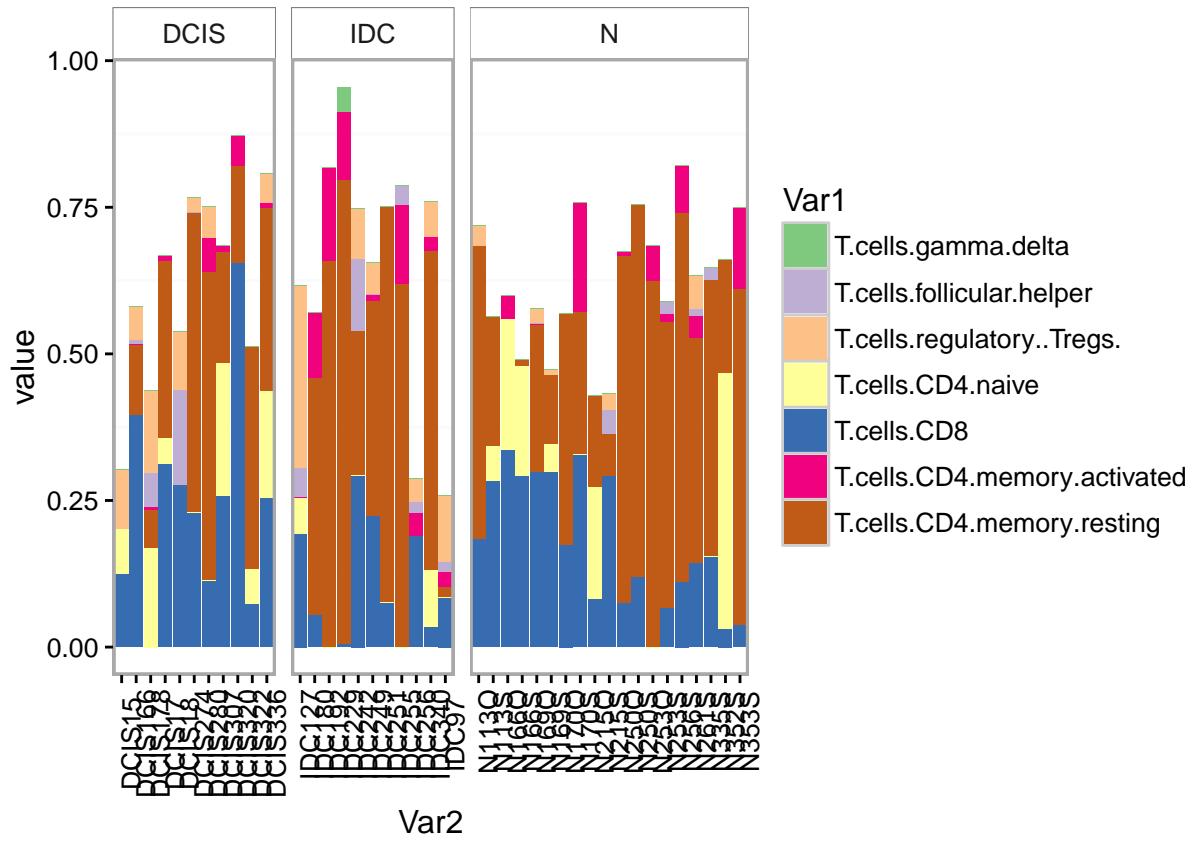
Cmelt=melt(CresComb)
ggplot(Cmelt, aes(x=Var2, y=value, fill=Var1))+geom_bar(stat="identity")+theme_bw() + theme(axis.text.x =

```



Look at the specific T-cell compositions:

```
CresT=CresDat[grep("T.cell", rownames(CresDat)), ]
CresT2=melt(CresT)
CresT2$type="N"
CresT2$type[grep("DCIS", CresT2$Var2)]="DCIS"
CresT2$type[grep("IDC", CresT2$Var2)]="IDC"
CresT2$Var1 <- factor(CresT2$Var1, levels =
                      c("T.cells.gamma.delta","T.cells.follicular.helper","T.cells.regulatory..Tregs..Treg1","T.cells.regulatory..Treg2","T.cells.regulatory..Treg3","T.cells.regulatory..Treg4","T.cells.regulatory..Treg5","T.cells.regulatory..Treg6","T.cells.regulatory..Treg7","T.cells.regulatory..Treg8","T.cells.regulatory..Treg9","T.cells.regulatory..Treg10","T.cells.regulatory..Treg11","T.cells.regulatory..Treg12","T.cells.regulatory..Treg13","T.cells.regulatory..Treg14","T.cells.regulatory..Treg15","T.cells.regulatory..Treg16","T.cells.regulatory..Treg17","T.cells.regulatory..Treg18","T.cells.regulatory..Treg19","T.cells.regulatory..Treg20","T.cells.regulatory..Treg21","T.cells.regulatory..Treg22","T.cells.regulatory..Treg23","T.cells.regulatory..Treg24","T.cells.regulatory..Treg25","T.cells.regulatory..Treg26","T.cells.regulatory..Treg27","T.cells.regulatory..Treg28","T.cells.regulatory..Treg29","T.cells.regulatory..Treg30","T.cells.regulatory..Treg31","T.cells.regulatory..Treg32","T.cells.regulatory..Treg33","T.cells.regulatory..Treg34","T.cells.regulatory..Treg35"))
ggplot(CresT2, aes(x=Var2 , y=value, fill=Var1))+geom_bar(stat="identity")+theme_bw() + theme(axis.text.x = element_text(angle = 90, hjust = 0))
```



Rescale values to take into account T-cell specific content?

```

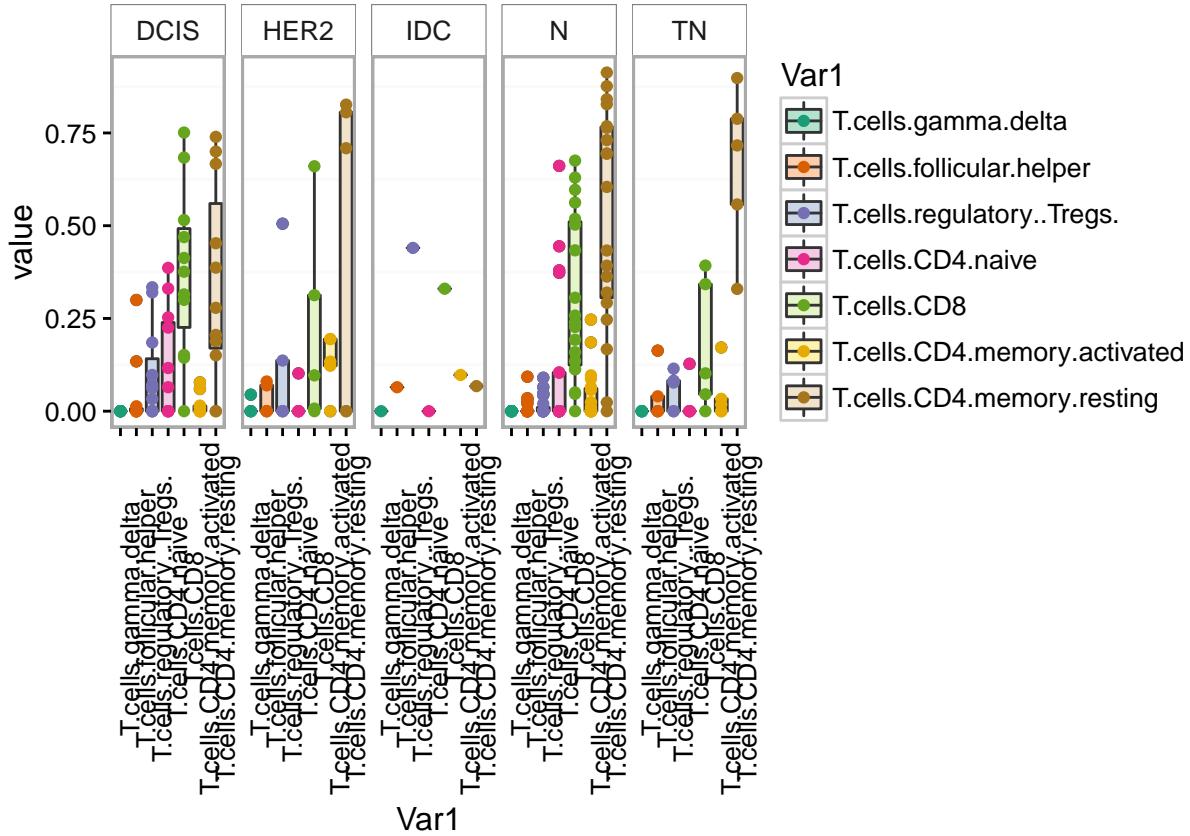
Cnames=gsub("\\.", "", rownames(SampleInfo))
Cnames=gsub("T[A-Z0-9]*$", "", Cnames)
l1=Cnames[SampleInfo$Subtype=="HER2"]
l2=Cnames[SampleInfo$Subtype=="TN"]
#l3=Cnames[SampleInfo$Subtype=="Org"]

CresTstd=t(t(CresT)/colSums(CresT))
CresT2std=melt(CresTstd)
CresT2std$type="N"
CresT2std$type[grep("DCIS", CresT2std$Var2)]="DCIS"
CresT2std$type[grep("IDC", CresT2std$Var2)]="IDC"

CresT2std$Var1 <- factor(CresT2std$Var1, levels =
                           c("T.cells.gamma.delta", "T.cells.follicular.helper", "T.cells.regulatory..Tregs.", "T.cells.CD4.naive", "T.cells.CD8", "T.cells.CD4.memory.activated", "T.cells.CD4.memory.resting"))

#ggplot(CresT2std, aes(x=Var2, y=value, fill=Var1))+geom_bar(stat="identity")+theme(axis.text.x = element_text(angle=90))

CresT2std$type[which(CresT2std$Var2%in%l2)]="TN"
CresT2std$type[which(CresT2std$Var2%in%l1)]="HER2"
#CresT2std$type[which(CresT2std$Var2%in%l3)]="Org"
ggplot(CresT2std, aes(x=Var1, y=value, fill=Var1))+geom_boxplot(stat="boxplot")+
  geom_point(aes(colour=Var1))
  
```



## 5 Cell activity: exhaustion, cytotoxic signatures: Figures S3, S5

Can the gene expression patterns tell us any information on the “activity” of the T-cells

### 5.1 Existing Signatures

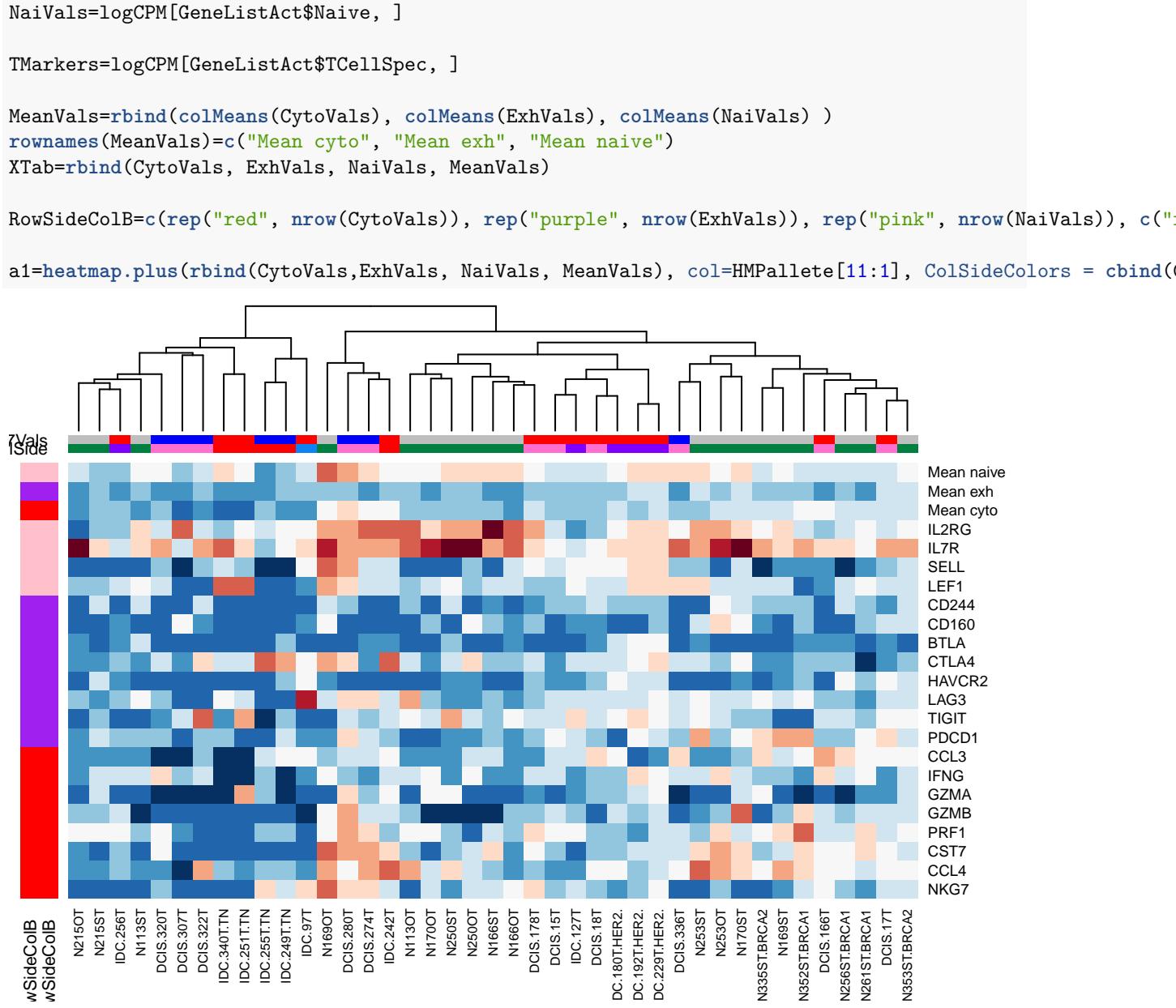
Existing signatures have been shown in Tirosh 2016, and discussed in Pardell 2015. Here, we compare the expression of different samples

Firstly, we colour according to Ki67 expression:

```
Ki67Vals=rep("blue", length(SampC))
Ki67Vals[which(ddsCount2["MKI67", ]>0.5)]="red"
Ki67Vals[grep("^N", colnames(logCPM))]="grey"
#colnames(logCPM)
#SampC
# centre the gene expression at 0
logCPMNew=sapply(1:ncol(logCPM), function(x) (logCPM[,x]-median(logCPM[,x])))
```

#### 5.1.1 Tirsoh 2016

```
# Find the cytotoxic genes:
CytoVals=logCPM[GeneListAct$Cytotoxic, ]
ExhVals=logCPM[GeneListAct$Exhausted, ]
```



Comparison of exhaustion with cytotoxic:

Follow the methods described by Tirosh: compute the average expression of the exhaustion or cytotoxic gene exp and subtract from the naive cells:

```

#MeanV2=t(t(MeanVals)-MeanVals[3, ])
par(mfrow=c(2,2))
plot(MeanVals[1, ], MeanVals[2, ], xlab="cytotoxic", ylab="exhausted", main="Cytotoxic vs Exhaustion signal")
a1=lm(MeanVals[2,which(Ki67Vals=="red") ]~MeanVals[1, which(Ki67Vals=="red")])
cValR=cor(MeanVals[2,which(Ki67Vals=="red") ], MeanVals[1, which(Ki67Vals=="red")])
abline(a1$coefficients, col="red")
a1=lm(MeanVals[2,which(Ki67Vals=="blue") ]~MeanVals[1, which(Ki67Vals=="blue")])
cValB=cor(MeanVals[2,which(Ki67Vals=="blue") ], MeanVals[1, which(Ki67Vals=="blue")])
abline(a1$coefficients, col="blue")
legend("topright", as.character(c(round(cValR*100)/100, round(cValB*100)/100)), col=c("red", "blue"), lty=1)

```

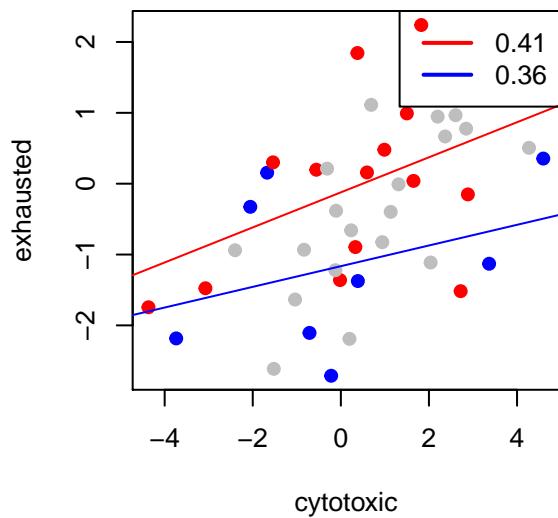
```

plot(MeanVals[1, ], MeanVals[3, ], xlab="cytotoxic", ylab="naive", main="Cytotoxic vs Naive signatures", l
a1=lm(MeanVals[3,which(Ki67Vals=="red") ]~MeanVals[1, which(Ki67Vals=="red"))])
cValR=cor(MeanVals[3,which(Ki67Vals=="red") ], MeanVals[1, which(Ki67Vals=="red"))])
abline(a1$coefficients, col="red")
a1=lm(MeanVals[3,which(Ki67Vals=="blue") ]~MeanVals[1, which(Ki67Vals=="blue"))])
cValB=cor(MeanVals[3,which(Ki67Vals=="blue") ], MeanVals[1, which(Ki67Vals=="blue"))])
abline(a1$coefficients, col="blue")
legend("topright", as.character(c(round(cValR*100)/100, round(cValB*100)/100)), col=c("red", "blue"), l

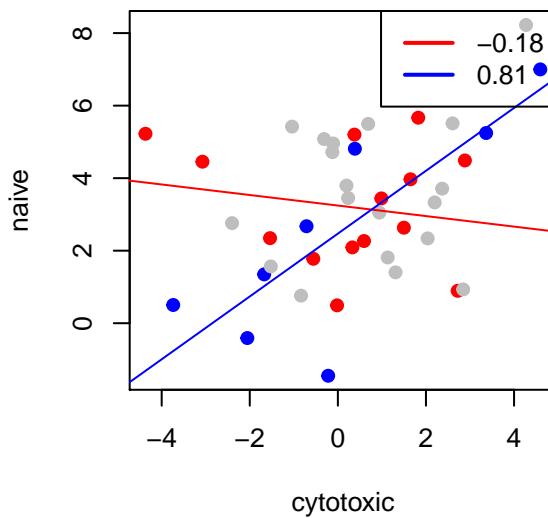
plot(MeanVals[2, ], MeanVals[3, ], xlab="exhausted", ylab="naive", main="Exhausted vs Naive signatures"
a1=lm(MeanVals[3,which(Ki67Vals=="red") ]~MeanVals[2, which(Ki67Vals=="red"))])
cValR=cor(MeanVals[3,which(Ki67Vals=="red") ], MeanVals[2, which(Ki67Vals=="red"))])
abline(a1$coefficients, col="red")
a1=lm(MeanVals[3,which(Ki67Vals=="blue") ]~MeanVals[2, which(Ki67Vals=="blue"))])
cValB=cor(MeanVals[3,which(Ki67Vals=="blue") ], MeanVals[2, which(Ki67Vals=="blue"))])
abline(a1$coefficients, col="blue")
legend("topright", as.character(c(round(cValR*100)/100, round(cValB*100)/100)), col=c("red", "blue"), l

```

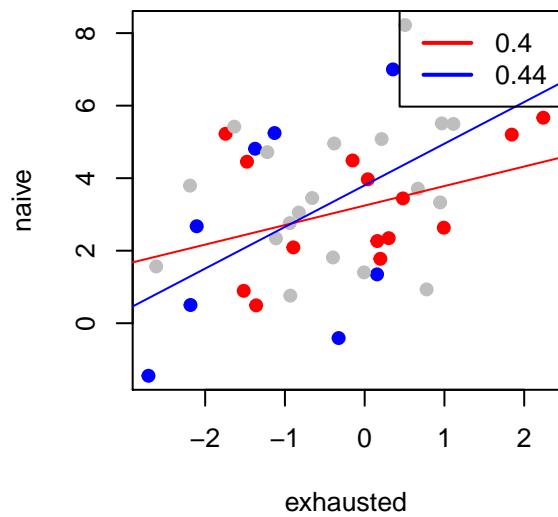
### Cytotoxic vs Exhaustion signatures



### Cytotoxic vs Naive signatures



### Exhausted vs Naive signatures



#### 5.1.2 Check-point genes

Pardell has discussed genes involved in T-cell check points: These have directionality - they can either activate a T-cell or inhibit its activity

```
glCD=rownames(logCPM)[grep("^\u00d7CD3[A-Z]\$", rownames(logCPM))]
```

```
dirnew=c(rep("green", length(GeneListAct$Activated)),
rep("blue", length(GeneListAct$Inhibit)))
```

```
CCVals=logCPM[c(GeneListAct$Activated, GeneListAct$Inhibit), ]
```

```
MeanValsCC=rbind(colMeans(logCPM[GeneListAct$Activated,]),
```

```

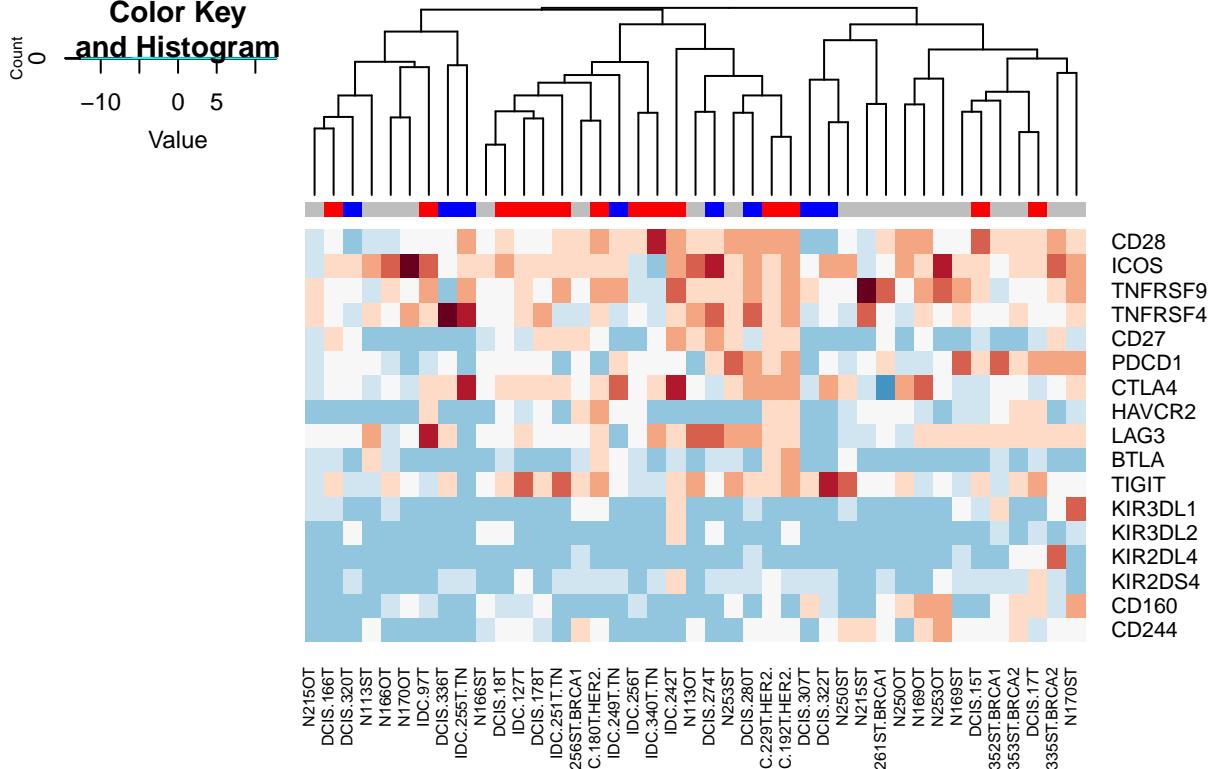
    colMeans(logCPM[GeneListAct$Inhibit,]),
    colMeans(logCPM[gLCD, ]))

rownames(MeanValsCC)=c("checkUp", "checkDown", "CD")

a1=heatmap.2(CCVals, col=HMPallete[11:1], ColSideColors = Ki67Vals, scale="none", Rowv=NA, trace="none")

```

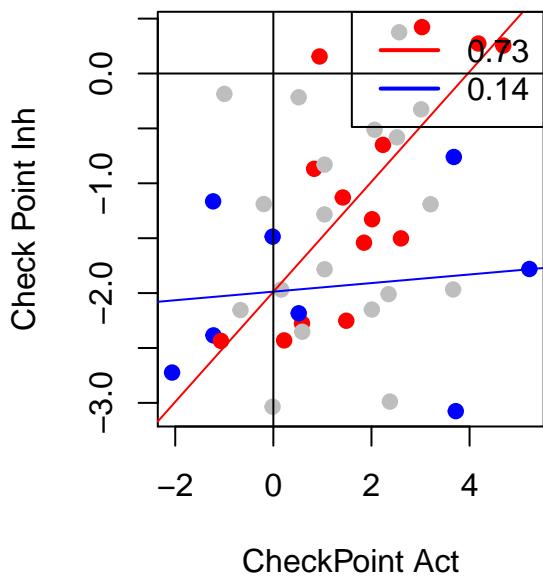
## Warning in heatmap.2(CCVals, col = HMPallete[11:1], ColSideColors =  
## Ki67Vals, : Discrepancy: Rowv is FALSE, while dendrogram is `both'.  
## Omitting row dendrogram.



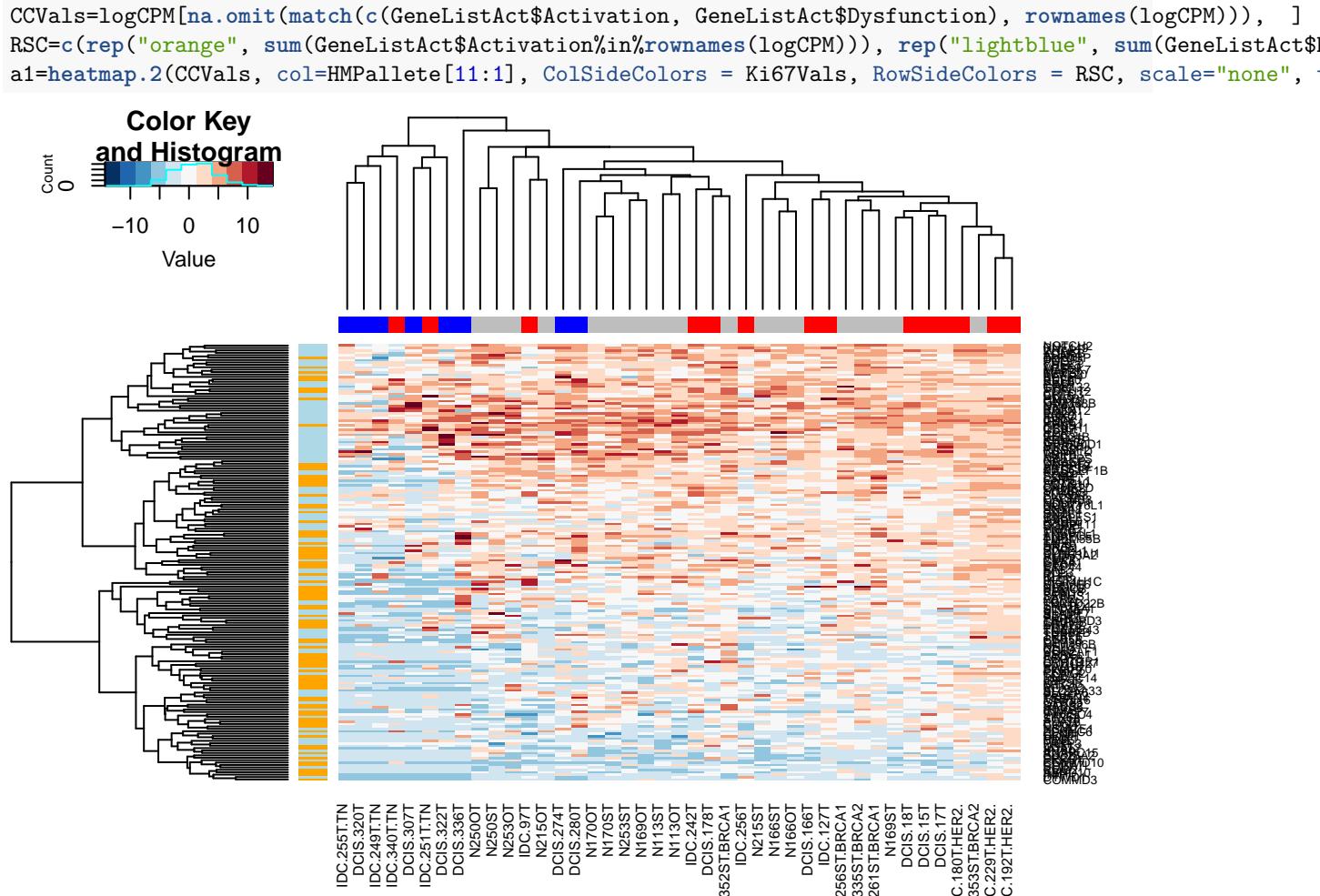
```

par(mfrow=c(1,2))
plot(MeanValsCC[1, ], MeanValsCC[2, ], pch=19, col=Ki67Vals, xlab="CheckPoint Act", ylab="Check Point Inhibit")
a1=lm(MeanValsCC[2,which(Ki67Vals=="red")]~MeanValsCC[1, which(Ki67Vals=="red")])
cValR=cor(MeanValsCC[2,which(Ki67Vals=="red")], MeanValsCC[1, which(Ki67Vals=="red")])
abline(a1$coefficients, col="red")
a1=lm(MeanValsCC[2,which(Ki67Vals=="blue")]~MeanValsCC[1, which(Ki67Vals=="blue")])
cValB=cor(MeanValsCC[2,which(Ki67Vals=="blue")], MeanValsCC[1, which(Ki67Vals=="blue")])
abline(a1$coefficients, col="blue")
legend("topright", as.character(c(round(cValR*100)/100, round(cValB*100)/100)), col=c("red", "blue"), lwd=2)
abline(h=0, v=0)

```



### 5.1.3 Singer Signature



## 5.2 Analysis of TIGIT staining the CD3+ cells

```
tigit <- read.csv('data/TIGIT.csv', stringsAsFactors = FALSE, header = FALSE)[-1, -1]
tigit.indices <- NULL
tigit.cells <- NULL
for (i in 1:nrow(tigit)) {
  for (j in 1:ncol(tigit)) {
    if (is.na(pmatch('M-', tigit[i, j])) == FALSE) {
      tigit.indices <- rbind(tigit.indices, c(i, j))
    }
    if (is.na(pmatch('Her2', tigit[i, j])) == FALSE | is.na(pmatch('HER2', tigit[i, j])) == FALSE | 
      is.na(pmatch('TIGIT', tigit[i, j])) == FALSE) {
      tigit.cells <- rbind(tigit.cells, c(tigit[i, j], i, j))
    }
  }
}
tigit.cells <- as.data.frame(data.matrix(tigit.cells), stringsAsFactors = FALSE)

tigit.sample.names <- tigit[tigit.indices]
unique.rows <- c(unique(tigit.indices[, 1]), nrow(tigit) + 1)
tigit.samples <- NULL
for (i in 1:(nrow(tigit.indices))) {
  temp <- tigit[(tigit.indices[i, 1] + 1):(unique.rows[which(unique.rows == tigit.indices[i, 1]) + 1])
  temp.end <- which(temp[, 1] == '')[1]
  if (is.na(temp.end) == FALSE) {
    temp <- temp[1:(temp.end - 1), ]
  }
  temp.col <- temp[1, ]
  temp <- as.data.frame(data.matrix(temp[-1, ]))
  colnames(temp) <- temp.col
  tigit.samples[[i]] <- list()
  tigit.samples[[i]]$table <- temp
  tigit.samples[[i]]$cell <- tigit.cells[tail(which(as.numeric(tigit.cells$V2) <= tigit.indices[i, 1]), 1)]
}
Types.tigit <- c('TIGIT+ T cells', 'TIGIT- T cells', 'TIGIT+ tumor cells', 'TIGIT- tumor cells')
Types.tigit.indices <- c(1, 2, 7, 5)
Cols <- c('black', 'red', 'forestgreen', 'gold2')
```

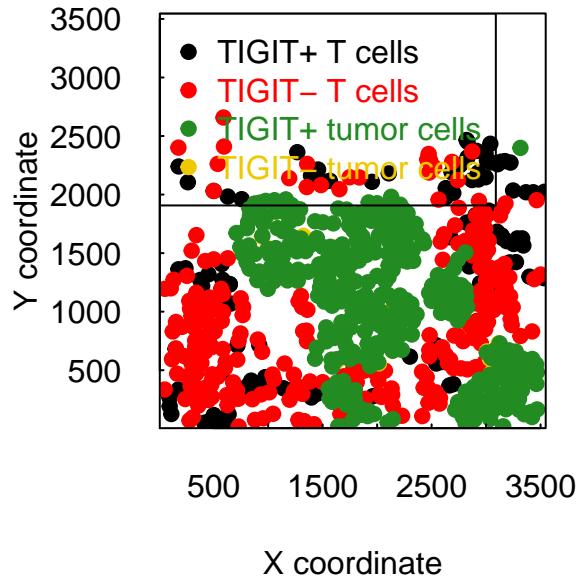
Here, the spatial information from IF is captured, highlighting the different cell populations:

- black: TIGIT + CD3+ TCells
- red: TIGIT- CD3+ Tcells
- green: TIGIT+ tumour cells
- yellow: TIGIT - tumour cells

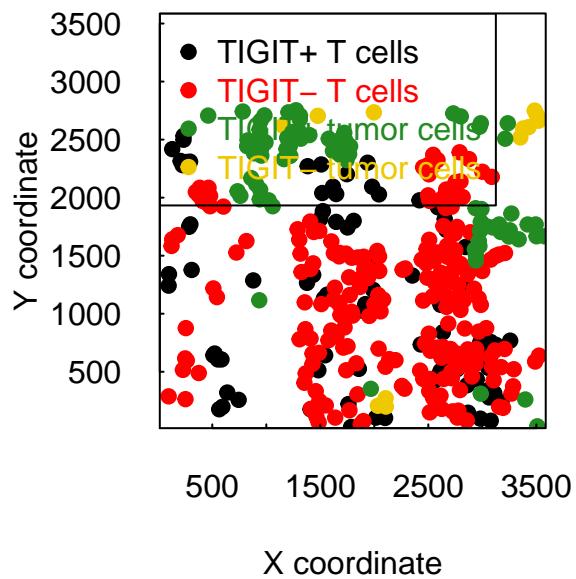
As an example:

```
## plot TIGIT CD3 positive cells
#pdf('TIGIT CD3.pdf', useDingbats = FALSE)
par(mfrow = c(1, 2), las = 1, tcl = .1, xaxs = 'i', yaxs = 'i')
for (i in 1:2) {
  # change 1 to length(tigit.samples)
  plot(tigit.samples[[i]]$table$X[tigit.samples[[i]]$table>Type %in% Types.tigit.indices], tigit.samples[[i]]$table$Y[tigit.samples[[i]]$table>Type %in% Types.tigit.indices], legend('topleft', Types.tigit, col = Cols, text.col = Cols, pch = 19))
}
```

**M-S11-32228\_1: Her2+ IDC**



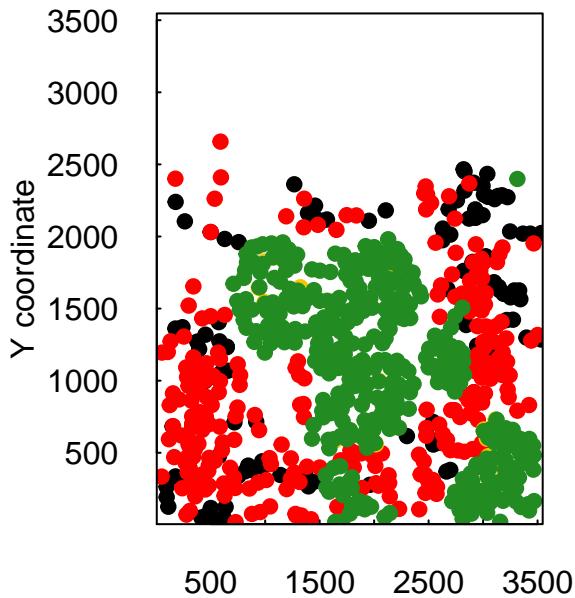
**M-S11-32228\_1: Her2+ IDC**



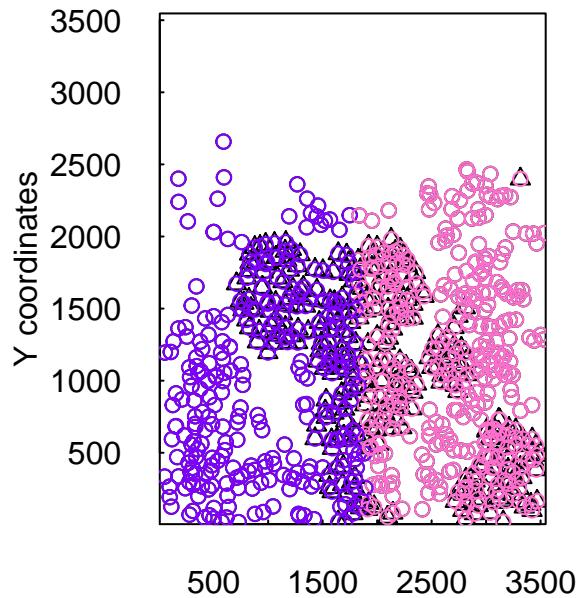
Here, we now analyse the spatial distributions between the different samples: Use k means to using a spatial uniform Poisson process and MC simulation to generate spatial distributions under the null hypothesis (that cells are random) and compared them to our images using the cross K function.

```
set.seed(1234)
distance.tigit <- NULL
par(mfrow = c(1, 2), las = 1, tcl = .1, xaxs = 'i', yaxs = 'i')
for (i in 1:length(tigit.samples)){
  neg.T <- as.data.frame(cbind(tigit.samples[[i]]$table$X[tigit.samples[[i]]$table$type == 2], tigit...
  pos.T <- as.data.frame(cbind(tigit.samples[[i]]$table$X[tigit.samples[[i]]$table$type == 1], tigit...
  neg.C <- as.data.frame(cbind(tigit.samples[[i]]$table$X[tigit.samples[[i]]$table$type == 5], tigit...
  pos.C <- as.data.frame(cbind(tigit.samples[[i]]$table$X[tigit.samples[[i]]$table$type == 7], tigit...
  T <- as.data.frame(cbind(tigit.samples[[i]]$table$X[tigit.samples[[i]]$table$type == 2 | tigit.sample...
  C <- as.data.frame(cbind(tigit.samples[[i]]$table$X[tigit.samples[[i]]$table$type == 5 | tigit.sample...
  colnames(neg.T) <- colnames(pos.T) <- colnames(neg.C) <- colnames(pos.C) <- colnames(T) <- colnames...
  TC <- rbind(T, C)
  km.TC <- kmeans(TC, centers = 2)
  if(i %in% c(1,2)){
    plot(tigit.samples[[i]]$table$X[tigit.samples[[i]]$table$type %in% Types.tigit.indices], tigit.sample...
    plot(T$x, T$y, pch = 1, main = paste0(tigit.sample.names[i], ': ', tigit.samples[[i]]$cell), xlim =...
    points(C$x, C$y, pch = 2)
    points(TC$x, TC$y, col = km.TC$cluster)
  }
  distance.tigit <- rbind(distance.tigit, c(max(length(which(km.TC$cluster[1:nrow(T)] == 1))) / nrow(T...
}
}
```

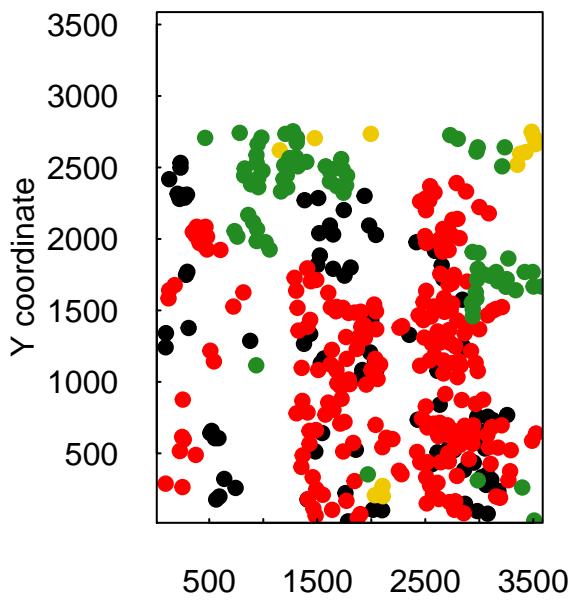
**M-S11-32228 : Her2+ IDC**



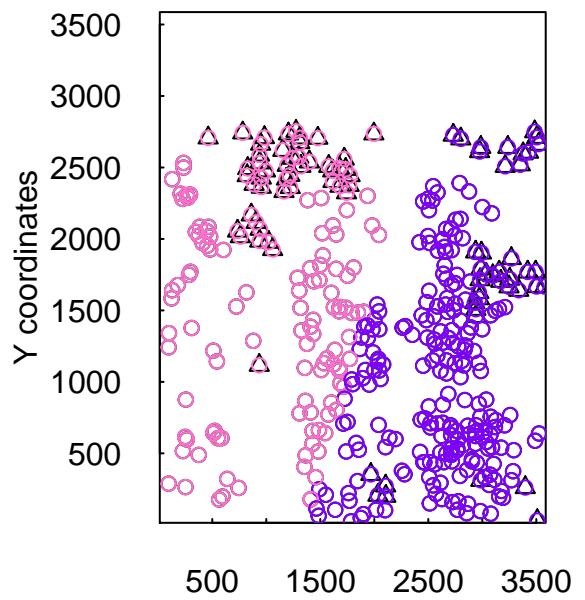
**M-S11-32228 : Her2+ IDC**



**M-S11-32228\_1 : Her2+ IDC**



**M-S11-32228\_1 : Her2+ IDC**



Now, following our summary of the data, we can compare the differences between the distributions:

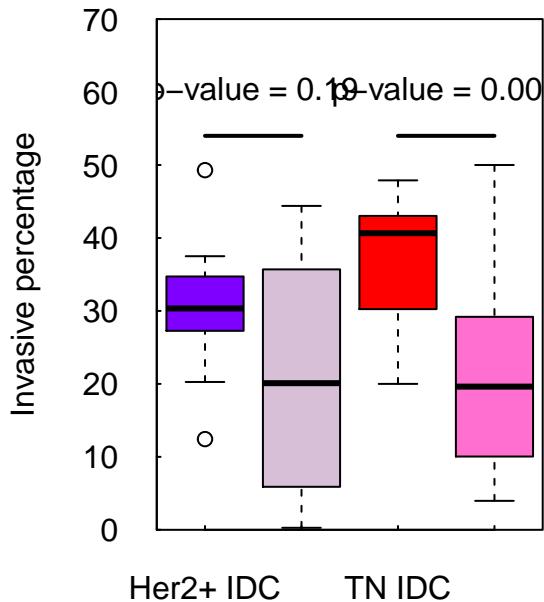
```
## [1] "t-test group 1 vs 2"  
##  
## Welch Two Sample t-test
```

```

## 
## data: distance.tigit[distance.tigit[, 2] < 3, 1] by distance.tigit[distance.tigit[, 2] < 3, 2]
## t = -1.4277, df = 9.0024, p-value = 0.1871
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## -0.24227399 0.05478719
## sample estimates:
## mean in group 1 mean in group 2
## 0.6962007 0.7899441
## [1] "t-test group 3 vs 4"

##
## Welch Two Sample t-test
##
## data: distance.tigit[distance.tigit[, 2] > 2, 1] by distance.tigit[distance.tigit[, 2] > 2, 2]
## t = -3.177, df = 15.486, p-value = 0.00605
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## -0.25480224 -0.05051996
## sample estimates:
## mean in group 3 mean in group 4
## 0.6365692 0.7892303

```



### 5.3 PD1 staining

Do the same analysis for PD1 staining:

```

pd1 <- read.csv('data/PD1.csv', stringsAsFactors = FALSE, header = FALSE)[-1, -1]
pd1.indices <- NULL
pd1.cells <- NULL
for (i in 1:nrow(pd1)) {
  for (j in 1:ncol(pd1)) {
    if (is.na(pmatch('M-', pd1[i, j])) == FALSE) {
      pd1.indices <- rbind(pd1.indices, c(i, j))
    }
  }
}

```

```

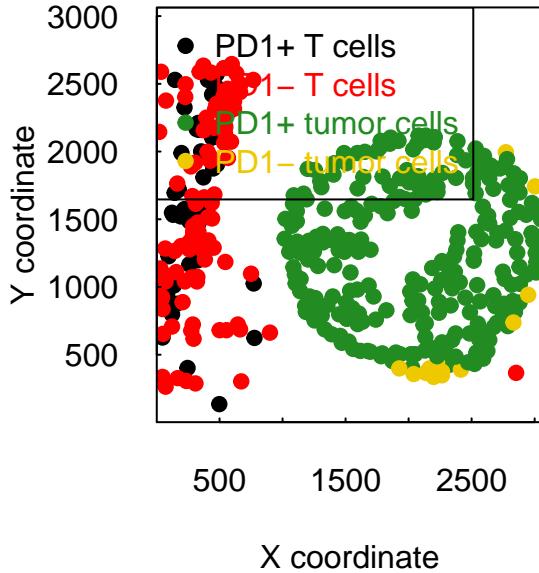
    }
    if (is.na(pmatch('Her2', pd1[i, j])) == FALSE | is.na(pmatch('HER2', pd1[i, j])) == FALSE | is.na(pmatch('HER2+', pd1[i, j])))
      pd1.cells <- rbind(pd1.cells, c(pd1[i, j], i, j))
    }
  }
pd1.cells <- as.data.frame(data.matrix(pd1.cells), stringsAsFactors = FALSE)
pd1.cells$V1[1] <- "Her2+ DCIS"
pd1.cells$V1[5] <- 'Her2+ DCIS'

pd1.sample.names <- pd1[pd1.indices]
unique.rows <- c(unique(pd1.indices[, 1]), nrow(pd1) + 1)
pd1.samples <- NULL
for (i in 1:(nrow(pd1.indices))) {
  temp <- pd1[(pd1.indices[i, 1] + 1):(unique.rows[which(unique.rows == pd1.indices[i, 1]) + 1] - 1),
  temp.end <- which(temp[, 1] == '')[1]
  if (is.na(temp.end) == FALSE) {
    temp <- temp[1:(temp.end - 1), ]
  }
  temp.col <- temp[1, ]
  temp <- as.data.frame(data.matrix(temp[-1, ]))
  colnames(temp) <- temp.col
  pd1.samples[[i]] <- list()
  pd1.samples[[i]]$table <- temp
  pd1.samples[[i]]$cell <- pd1.cells[tail(which(as.numeric(pd1.cells$V2) <= pd1.indices[i, 1]), n = 1)]
}
Types.pd1 <- c('PD1+ T cells', 'PD1- T cells', 'PD1+ tumor cells', 'PD1- tumor cells')
Types.pd1.indices <- c(1, 2, 3, 4)
Cols <- c('black', 'red', 'forestgreen', 'gold2')

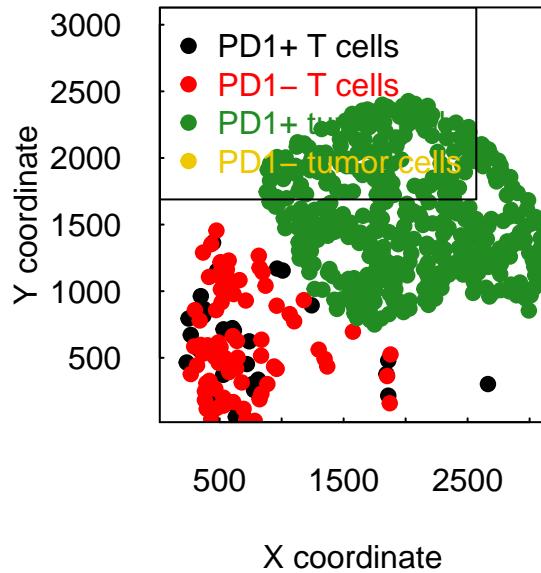
## plot pd1 CD3 positive cells
#pdf('pd1_CD3.pdf', useDingbats = FALSE)
par(mfrow = c(1, 2), las = 1, tcl = .1, xaxs = 'i', yaxs = 'i')
for (i in 1:2) {
  # change 1 to length(pd1.samples)
  plot(pd1.samples[[i]]$table$X[pd1.samples[[i]]$table>Type %in% Types.pd1.indices], pd1.samples[[i]]$table$Y[pd1.samples[[i]]$table>Type %in% Types.pd1.indices], type = 'p',
  legend('topleft', Types.pd1, col = Cols, text.col = Cols, pch = 19)
}

```

M-S10-28388\_1: Her2+ DCIS



M-S10-28388\_1: Her2+ DCIS

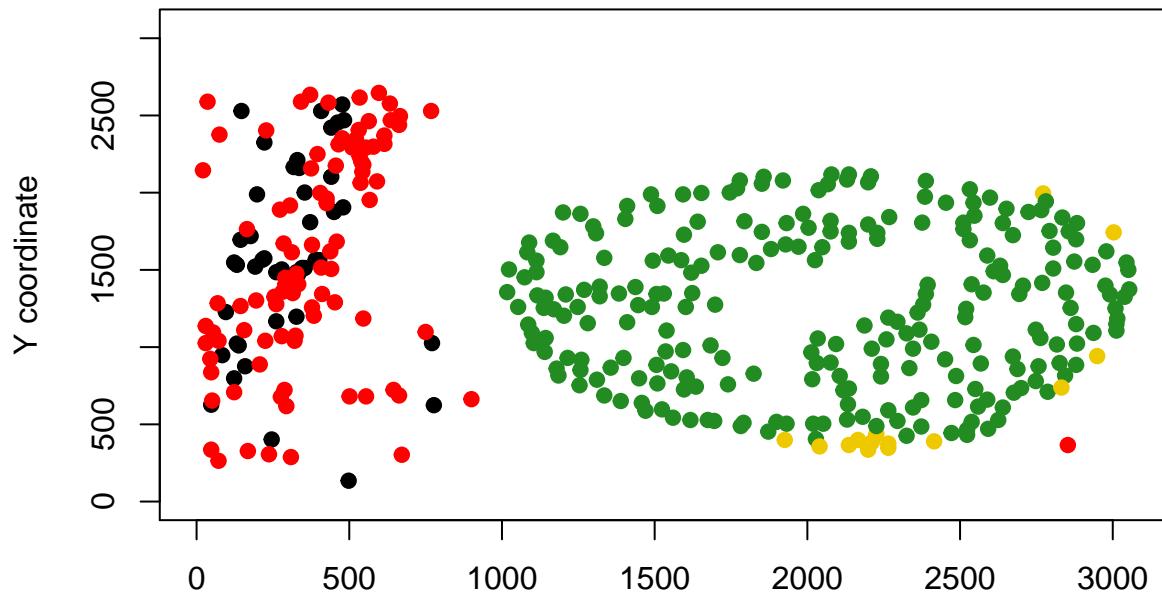


Here, we now analyse the spatial distributions between the different samples: Use k means to using a spatial uniform Poisson process and MC simulation to generate spatial distributions under the null hypothesis (that cells are random) and compared them to our images using the cross K function.

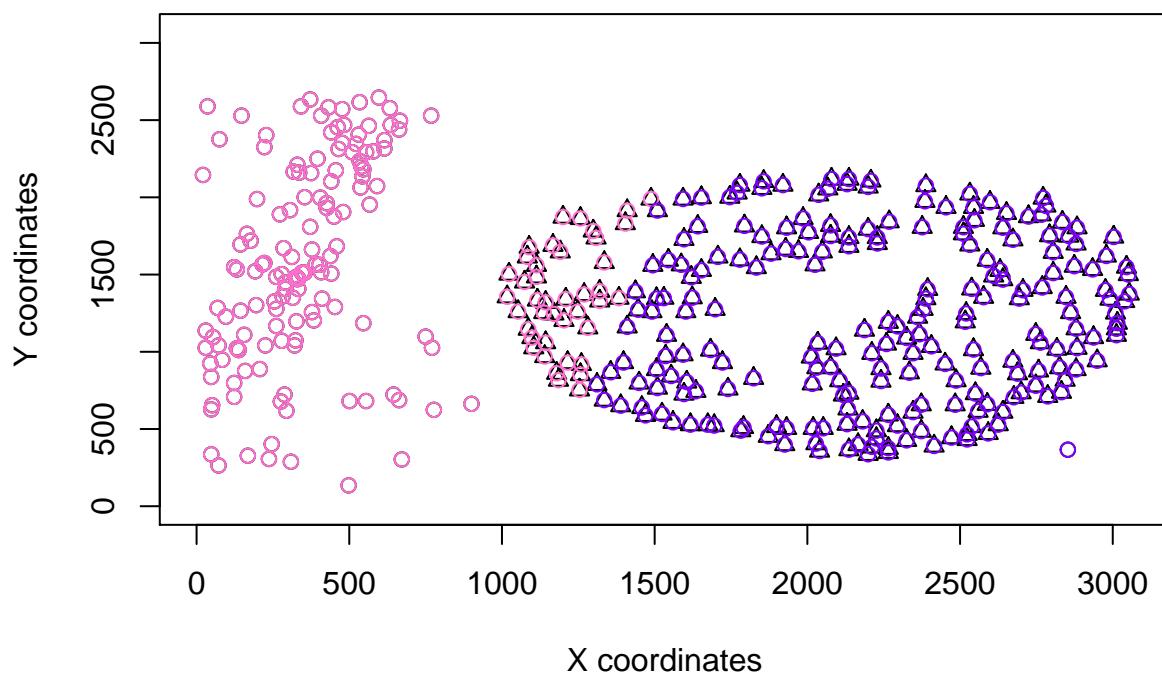
```
set.seed(1234)
distance.pd1 <- NULL
for (i in 1:length(pd1.samples)) {
  neg.T <- as.data.frame(cbind(pd1.samples[[i]]$table$X[pd1.samples[[i]]$table$type == 2], pd1.samples[[i]]$table$Y[pd1.samples[[i]]$table$type == 2]))
  pos.T <- as.data.frame(cbind(pd1.samples[[i]]$table$X[pd1.samples[[i]]$table$type == 1], pd1.samples[[i]]$table$Y[pd1.samples[[i]]$table$type == 1]))
  neg.C <- as.data.frame(cbind(pd1.samples[[i]]$table$X[pd1.samples[[i]]$table$type == 4], pd1.samples[[i]]$table$Y[pd1.samples[[i]]$table$type == 4]))
  pos.C <- as.data.frame(cbind(pd1.samples[[i]]$table$X[pd1.samples[[i]]$table$type == 3], pd1.samples[[i]]$table$Y[pd1.samples[[i]]$table$type == 3]))
  T <- as.data.frame(cbind(pd1.samples[[i]]$table$X[pd1.samples[[i]]$table$type == 2 | pd1.samples[[i]]$table$type == 1], pd1.samples[[i]]$table$Y[pd1.samples[[i]]$table$type == 2 | pd1.samples[[i]]$table$type == 1]))
  C <- as.data.frame(cbind(pd1.samples[[i]]$table$X[pd1.samples[[i]]$table$type == 4 | pd1.samples[[i]]$table$type == 3], pd1.samples[[i]]$table$Y[pd1.samples[[i]]$table$type == 4 | pd1.samples[[i]]$table$type == 3]))
  colnames(neg.T) <- colnames(pos.T) <- colnames(neg.C) <- colnames(pos.C) <- colnames(T) <- colnames(C)
  TC <- rbind(T, C)

  if (i==1){
    plot(pd1.samples[[i]]$table$X[pd1.samples[[i]]$table$type %in% Types.pd1.indices], pd1.samples[[i]]$table$Y[pd1.samples[[i]]$table$type %in% Types.pd1.indices])
  }
  km.TC <- kmeans(TC, centers = 2)
  if (i==1){
    plot(T$x, T$y, pch = 1, main = paste0(pd1.sample.names[i], ': ', pd1.samples[[i]]$cell), xlim = c(min(TC$x), max(TC$x)), ylim = c(min(TC$y), max(TC$y)))
    points(C$x, C$y, pch = 2)
    points(TC$x, TC$y, col = km.TC$cluster)
  }
  distance.pd1 <- rbind(distance.pd1, c(max(length(which(km.TC$cluster[1:nrow(T)] == 1)) / nrow(T), 1))
}
```

### M-S10-28388\_: Her2+ DCIS



### M-S10-28388\_: Her2+ DCIS



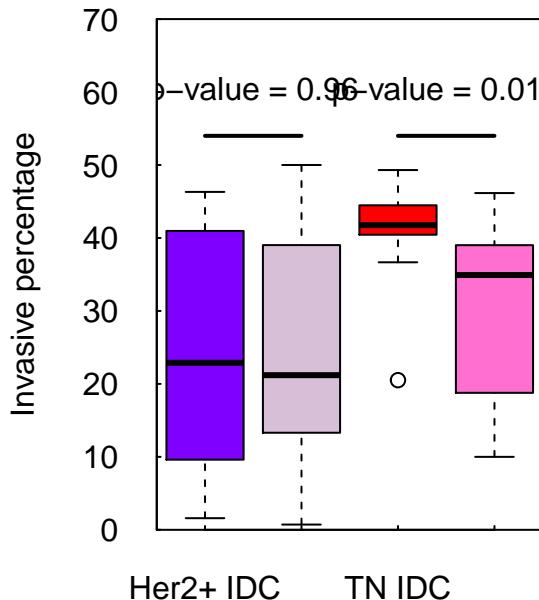
Now, following our summary of the data, we can compare the differences between the distributions:

```
## [1] "t-test group 1 vs 2"  
##  
## Welch Two Sample t-test
```

```

## 
## data: distance.pd1[distance.pd1[, 2] < 3, 1] by distance.pd1[distance.pd1[, 2] < 3, 2]
## t = 0.050595, df = 25.583, p-value = 0.96
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## -0.1243100 0.1305789
## sample estimates:
## mean in group 1 mean in group 2
## 0.7674002 0.7642658
## [1] "t-test group 3 vs 4"
##
## Welch Two Sample t-test
##
## data: distance.pd1[distance.pd1[, 2] > 2, 1] by distance.pd1[distance.pd1[, 2] > 2, 2]
## t = 2.7765, df = 20.24, p-value = 0.01156
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## 0.02738358 0.19232719
## sample estimates:
## mean in group 3 mean in group 4
## 0.6996395 0.5897841

```



## 6 TCR clonality

TCR clonality was determined using mixcr (Bolotin, Nat Methods 2015, <https://mixcr.readthedocs.io>). RNA-seq reads were aligned to a known database of TCR sequences, and are loaded here:

```

fileList <- list.files('output/mixcr/', '.clones.TCR.txt')

combinedDF <- data.frame(stringsAsFactors = FALSE)
droppedSamples <- character()

```

```

for (i in 1:length(fileList))
{
  if (length(readLines(paste('output/mixcr/', fileList[i], sep = ''))) > 0)
  {
    currentDF <- read.table(paste('output/mixcr/', fileList[i], sep = ''), sep = '\t', header = 1, stri
    currentDF <- cbind(Sample = rep(strsplit(fileList[i], '.clones.TCR.txt')[[1]], nrow(currentDF)), cu
    if (nrow(combinedDF) == 0)
    {
      combinedDF <- currentDF
    }
    else
    {
      combinedDF <- rbind(combinedDF, currentDF)
    }
    rm(currentDF)
  }
  else
  {
    droppedSamples <- c(droppedSamples, strsplit(fileList[i], '.clones.TCR.txt')[[1]])
  }
}
combinedDF$Sample <- as.character(combinedDF$Sample)

countsTable <- ddsCounts[,order(colnames(ddsCounts))]
SampleInfo=SampleInfo[order(rownames(SampleInfo)), ]
SampleInfo$SNames <- gsub("\\.", "", rownames(SampleInfo))
SampleInfo$SNames<-gsub("T[A-Z0-9]*$", "", SampleInfo$SNames)
CNamesCT=gsub("\\.", "", colnames(countsTable))
CNamesCT=gsub("T[A-Z0-9]*$", "", CNamesCT)

Filter out samples which have less than 2 counts, and compute Shannon index for each sample
filteredSamples <- unique(combinedDF$Sample)[!unique(combinedDF$Sample) %in% unique(combinedDF[combinedD
droppedSamples <- c(droppedSamples, filteredSamples)
combinedDF <- combinedDF[combinedDF$Clone.count > 1,]

cloneCount <- combinedDF %>% group_by(Sample) %>% count()

## Warning: failed to assign NativeSymbolInfo for env since env is already
## defined in the 'lazyeval' namespace

sampleNames <- unique(combinedDF$Sample)
sampleShannonIndex <- lapply(sampleNames, function (x) { diversity(as.numeric(combinedDF[combinedDF$Sampl

shannonIndexTable <- data.frame('Sample' = sampleNames, 'ShannonIndex' = as.numeric(sampleShannonIndex)
  'Color' = SampleInfo[ SampleInfo$SNames %in% sampleNames,'finalCol'])
shannonIndexTable$Sample <- as.character(shannonIndexTable$Sample)
shannonIndexTable$SampleType <- as.character(shannonIndexTable$SampleType)
shannonIndexTable$Color <- as.character(shannonIndexTable$Color)

dcisIDCWilcoxP <- wilcox.test(shannonIndexTable[shannonIndexTable$SampleType == 'DCIS','ShannonIndex']

## Warning in wilcox.test.default(shannonIndexTable[shannonIndexTable
## $SampleType == : cannot compute exact p-value with ties

```

```

dcisNormalWilcoxP <- wilcox.test(shannonIndexTable[shannonIndexTable$SampleType == 'DCIS', 'ShannonIndex']

## Warning in wilcox.test.default(shannonIndexTable[shannonIndexTable
## $SampleType == : cannot compute exact p-value with ties
idcNormalWilcoxP <- wilcox.test(shannonIndexTable[shannonIndexTable$SampleType == 'IDC', 'ShannonIndex']

## Warning in wilcox.test.default(shannonIndexTable[shannonIndexTable
## $SampleType == : cannot compute exact p-value with ties
pValToText <- function (pVal)
{
  if (pVal > 0.05) { return('ns') }
  else { paste('p=', round(pVal, 2), sep = '') }
}

dcisIDCWilcoxPText <- pValToText(dcisNormalWilcoxP)
dcisNormalWilcoxPText <- pValToText(dcisNormalWilcoxP)
idcNormalWilcoxPText <- pValToText(idcNormalWilcoxP)

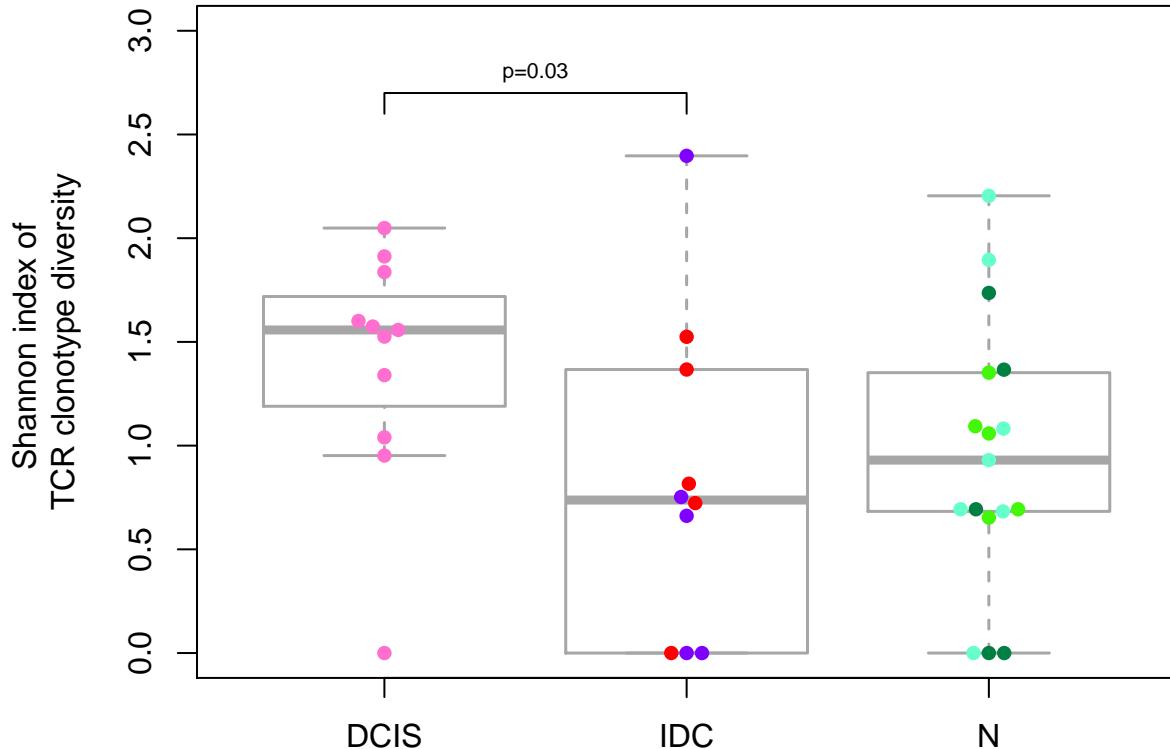
```

Plot the distribution of diversity across different patient samples:

```

boxplot(ShannonIndex ~ SampleType, data = shannonIndexTable, outline = FALSE, border = c('#a6a6a6'), yl
beeswarm(data = shannonIndexTable, ShannonIndex ~ SampleType, add = TRUE, xlab = '', ylab = '', pch = 1
lines(x=c(1,1,2,2), y=c(2.6,2.7,2.7,2.6))
text(1.5, 2.8, dcisIDCWilcoxPText, cex = 0.7)

```

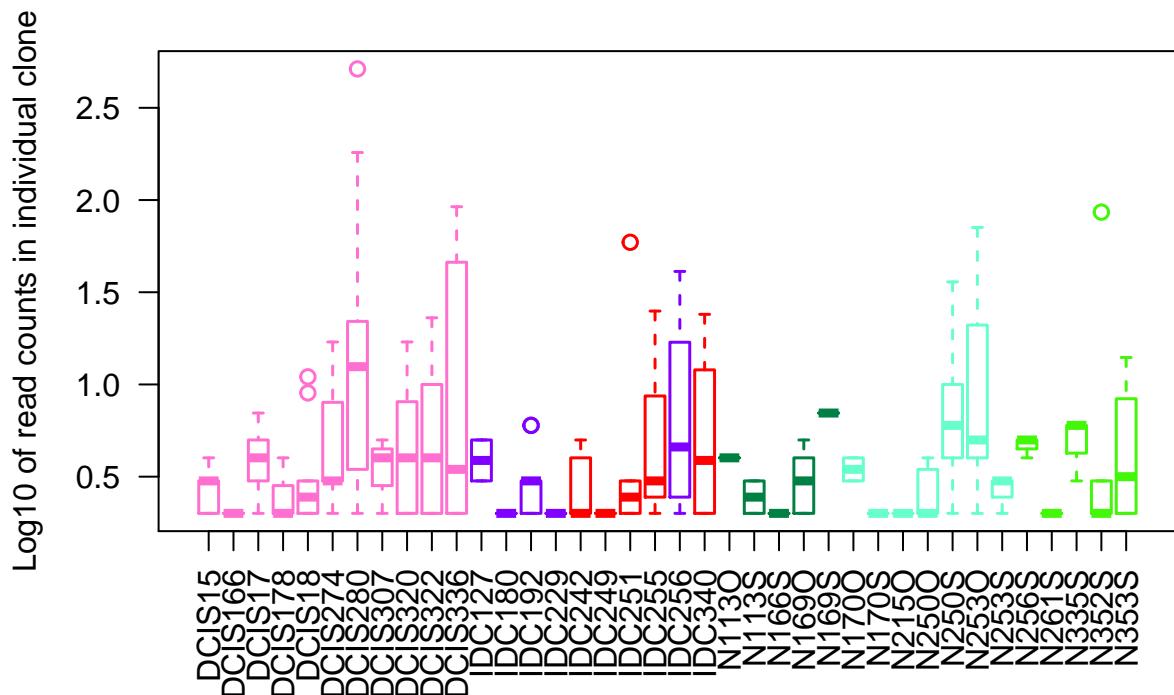


Overview of read counts across different samples:

```

boxplot(log10(combinedDF$Clone.count) ~ combinedDF$Sample, las=2, par(mar=c(8,4,2,2)), ylab='Log10 of r

```



```

individualGenePlot <- function(geneName)
{
  currentGeneTable <- data.frame('Gene'=(countsTable[geneName,match(shannonIndexTable$Sample, CNamesCT)]))

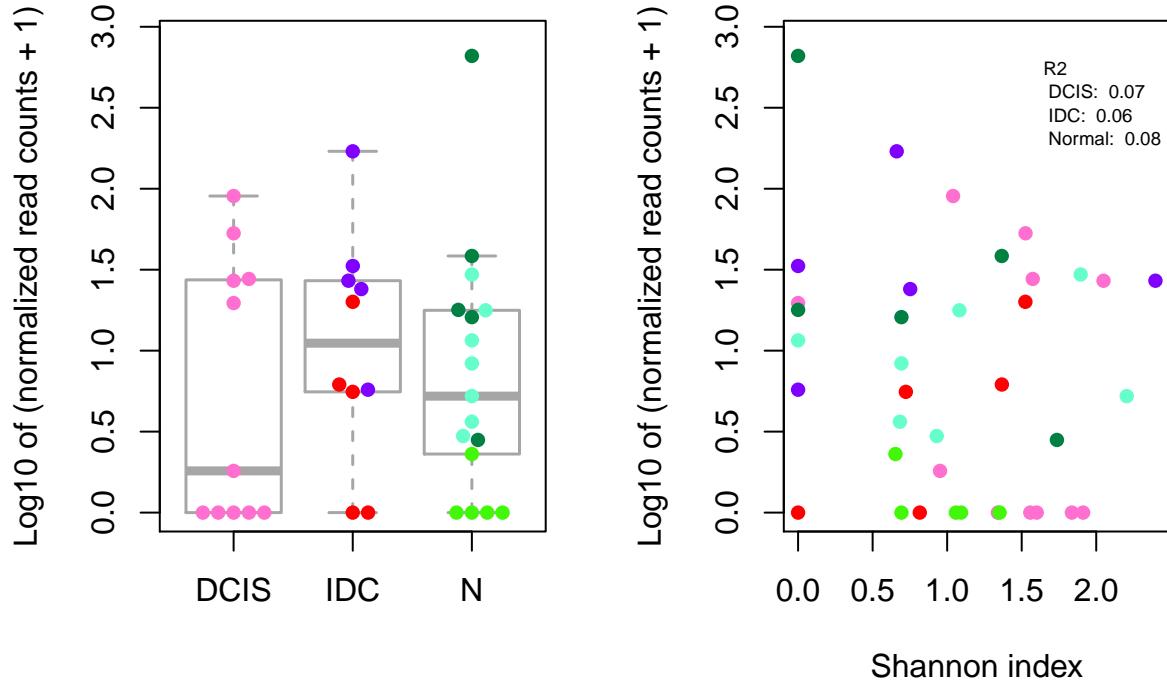
  par(mfrow = c(1,2))
  boxplot(log10(Gene + 1) ~ SampleType, data = currentGeneTable, outline = FALSE, border = c('#a6a6a6'))
  beeswarm(data = currentGeneTable, log10(Gene + 1) ~ SampleType, add = TRUE, xlab = '', ylab = '', pch = 19)

  plot(currentGeneTable$ShannonIndex, log10(currentGeneTable$Gene + 1), xlab = 'Shannon index', ylab =
#text(currentGeneTable$ShannonIndex, log10(currentGeneTable$Gene + 1), labels = rownames(currentGeneTable))
dcisRSq <- round(summary(lm(ShannonIndex ~ log10(Gene + 1), data = currentGeneTable[currentGeneTable$Sample == 'DCIS']), idcRSq <- round(summary(lm(ShannonIndex ~ log10(Gene + 1), data = currentGeneTable[currentGeneTable$Sample == 'IDC']), normalRSq <- round(summary(lm(ShannonIndex ~ log10(Gene + 1), data = currentGeneTable[currentGeneTable$Sample == 'Normal']), legend('topright', paste('R2\n', 'DCIS: ', dcisRSq, '\n', 'IDC: ', idcRSq, '\n', 'Normal: ', normalRSq, title(geneName, outer=TRUE, line=-1)
}

individualGenePlot('MKI67')

```

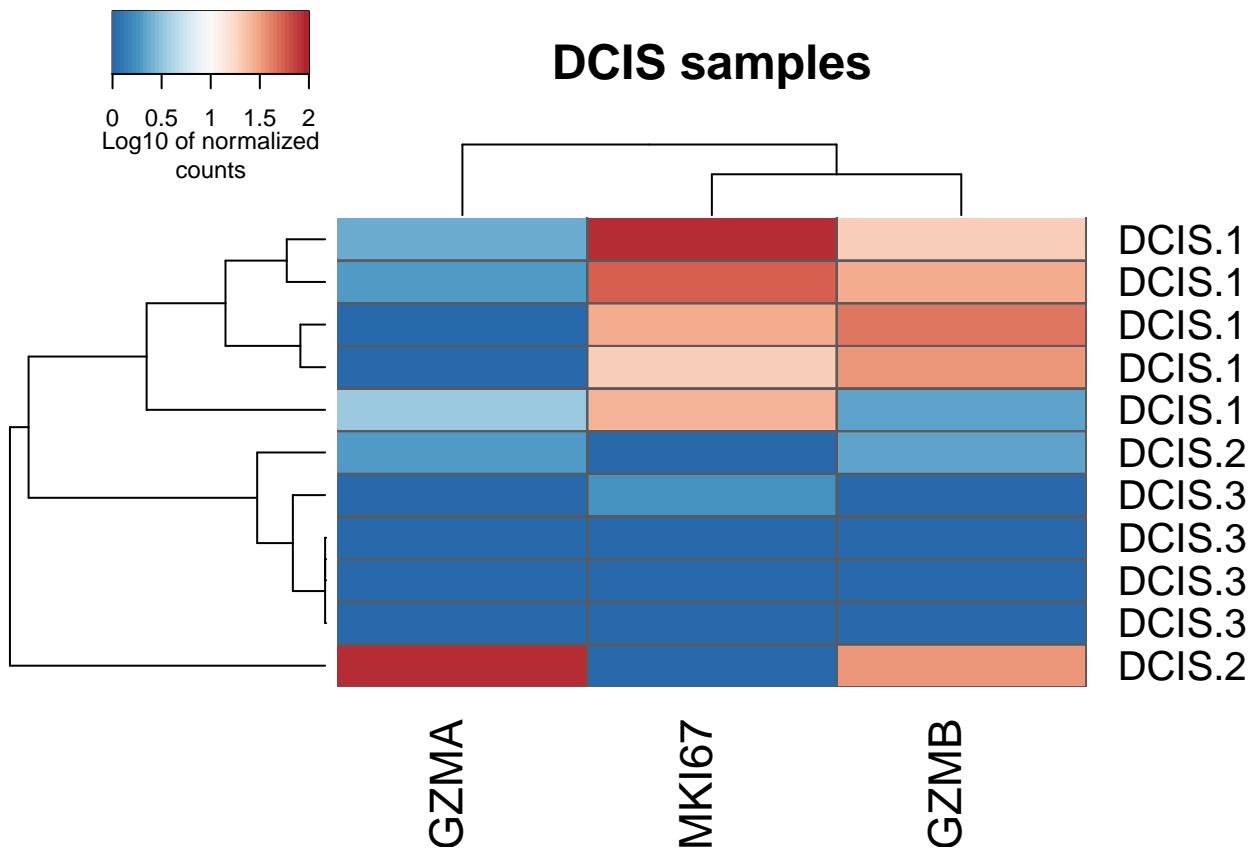
## MKI67



```

cytotoxicGeneTable <- as.data.frame(t(countsTable[c('MKI67', 'GZMA', 'GZMB'),
                                              match(shannonIndexTable$Sample, CNamesCT)]))
cytotoxicGeneTable <- log10(cytotoxicGeneTable + 1)
dcisMatCyto <- as.matrix(cytotoxicGeneTable[
  match(shannonIndexTable$Sample[shannonIndexTable$SampleType == 'DCIS'], CNamesCT),])
heatColors <- colorRampPalette(c('#2869ab', '#4594c2', '#94c6de', '#d2e8f2', '#fafafab', '#fbdac8', '#f3f3f3'))
heatmap.2(dcisMatCyto, trace = 'none', density.info='none', breaks = seq(from = 0, to = 2, length.out =

```



## 7 PDL1 amplification: Figure 4

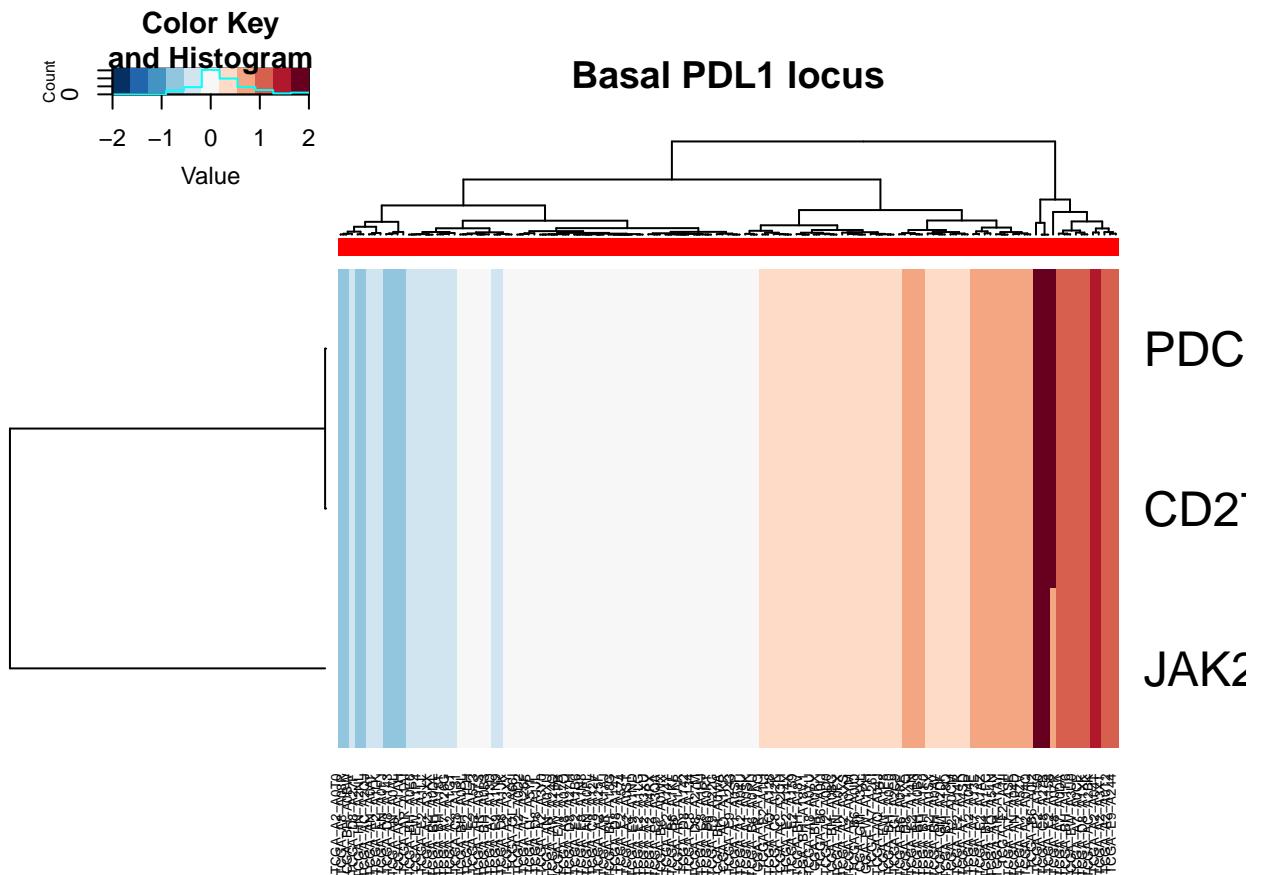
### 7.1 TCGA data

Load the TCGA GISTIC data and assess the frequency of PDL1 amplification in the different subtypes.

```
BasHER2=as.character(TCGAClin$PATIENT_ID[which(TCGAClin$PAM50=="Basal")])
n1=match(BasHER2, colnames(NewGistic))

ColIdNew=as.character(TCGAClin$PAM50[match(colnames(NewGistic), TCGAClin$PATIENT_ID)])
ColIdNew=gsub("Basal", "red", ColIdNew)
ColIdNew=gsub("Her2", "pink", ColIdNew)
ColIdNew=gsub("LumA", "blue", ColIdNew)
ColIdNew=gsub("LumB", "black", ColIdNew)
ColIdNew=gsub("Normal", "green", ColIdNew)

par(mfrow=c(1,2))
heatmap.2(data.matrix(NewGistic[c(21,22,24),n1]), col=HMPallette[11:1], scale="none",
          ColSideColors = ColIdNew[n1], trace="none", main="Basal PDL1 locus")
```

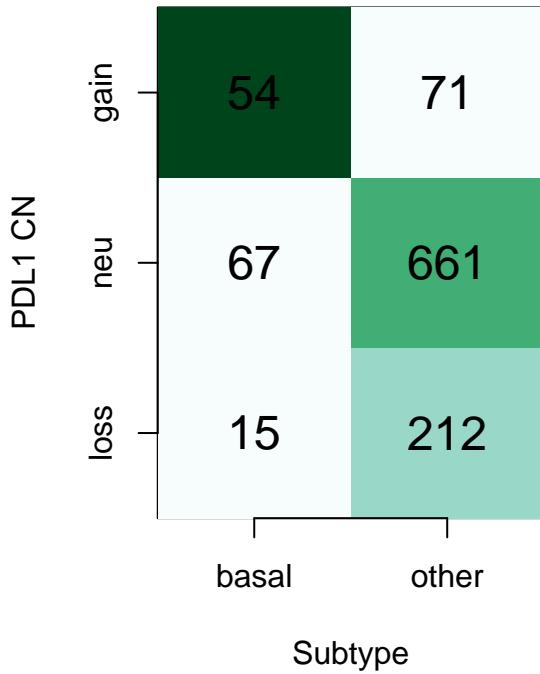


```

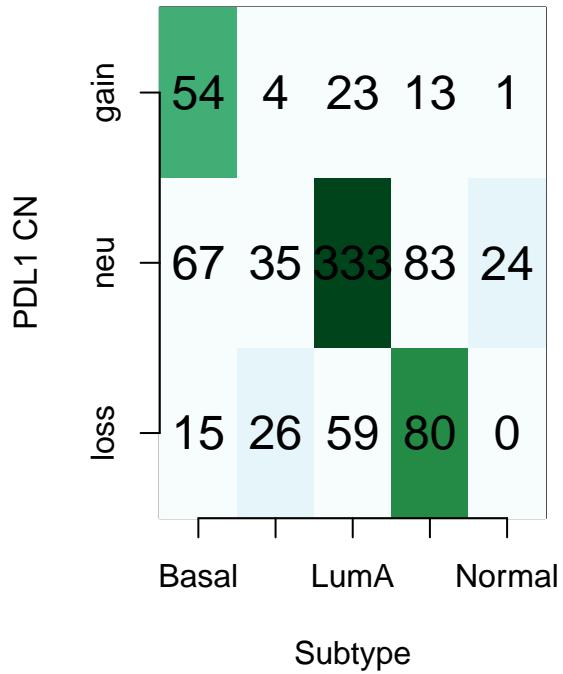
PDScores=colMeans(NewGistic[,c(21:22)])
PDscoresAmp=cut(PDScores, c(-2, -0.3, 0.3, 5), c("loss", "neu", "gain"))
anew=rep("other", length(PDscoresAmp))
anew[n1]="basal"
ContTable(table(anew, PDscoresAmp), title="PD1 amplification", chisqtest = T, xlabL = "Subtype", ylabL =
allSub=TCGAClin$PAM50[match(colnames(NewGistic), TCGAClin$PATIENT_ID)]
ContTable(table(allSub, PDscoresAmp), title="PD1 amplification", chisqtest = T, xlabL = "Subtype", ylabL =

```

**PD1 amplification Chisq = 0**



**PD1 amplification Chisq = 0**



Make a linear model to explain exhaustion or activation status in these cases

```

ActivationScore=colMedians(TCGAlogCPM[GeneListAct$Activated, ], na.rm=TRUE)
CytoScore2=colMeans(TCGAlogCPM[c("IFNG", "GZMB", "GZMA"),], na.rm=TRUE)
CytoScore=colMedians(TCGAlogCPM[GeneListAct$Cytotoxic,], na.rm=TRUE)
ExhScore=colMedians(TCGAlogCPM[GeneListAct$Exhausted, ], na.rm=TRUE)
InhScore=colMedians(TCGAlogCPM[match(GeneListAct$Inhibitory, rownames(TCGAlogCPM)), ], na.rm=TRUE)

MatchData=data.frame(Pat=colnames(NewGistic),CN=PDScores,
                     ActScore=ActivationScore[match(colnames(NewGistic), colnames(TCGAlogCPM))],
                     CytoScore=CytoScore[match(colnames(NewGistic), colnames(TCGAlogCPM))],
                     CytoScore2=CytoScore2[match(colnames(NewGistic), colnames(TCGAlogCPM))],
                     ExhScore=ExhScore[match(colnames(NewGistic), colnames(TCGAlogCPM))],
                     InhScore=InhScore[match(colnames(NewGistic), colnames(TCGAlogCPM))],
                     Neo=NeoData$Predicted.NeoAgs[match(colnames(NewGistic), NeoData$PatientID)],
                     CNA=TCGAClin$Copy.Number.Alterations[match(colnames(NewGistic), TCGAClin$PATIENT_ID)],
                     PDL1exp=TCGAlogCPM["CD274",match(colnames(NewGistic), colnames(TCGAlogCPM))],
                     Subtype=TCGAClin$PAM50[match(colnames(NewGistic), TCGAClin$PATIENT_ID)],
                     EST=ESTscore$ImmuneScore[match(colnames(NewGistic), ESTscore$Description)],
                     Purity=ESTscore$TumPurity[match(colnames(NewGistic), ESTscore$Description)],
                     Age=as.numeric(as.character(TCGAClin$Age[match(colnames(NewGistic), TCGAClin$PATIENT_ID)]))

# model probability of co-amplification
mod1B=glm(CN~log(Neo+1)+CNA+Subtype+EST+log(Age), data=MatchData)
X=summary(mod1B)
X

##
## Call:
## glm(formula = CN ~ log(Neo + 1) + CNA + Subtype + EST + log(Age),
## 
```

```

##      data = MatchData)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -0.95474 -0.20698  0.03683  0.12238  2.10415
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.997e-01 2.467e-01  2.431  0.0153 *
## log(Neo + 1) -7.941e-03 2.295e-02 -0.346  0.7295
## CNA        -1.730e-01 8.778e-02 -1.971  0.0492 *
## SubtypeHer2 -3.754e-01 6.174e-02 -6.081 2.09e-09 ***
## SubtypeLumA -2.963e-01 4.664e-02 -6.353 4.08e-10 ***
## SubtypeLumB -4.174e-01 4.665e-02 -8.948 < 2e-16 ***
## SubtypeNormal -2.646e-01 1.036e-01 -2.554  0.0109 *
## EST         1.432e-05 1.108e-05  1.293  0.1966
## log(Age)    -8.123e-02 6.098e-02 -1.332  0.1833
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1274455)
##
## Null deviance: 92.392 on 631 degrees of freedom
## Residual deviance: 79.399 on 623 degrees of freedom
## (448 observations deleted due to missingness)
## AIC: 502.51
##
## Number of Fisher Scoring iterations: 2
# check only in the Basal cases:
mod1B=glm(CN~log(Neo+1)+CNA+EST+log(Age), data=MatchData[MatchData$Subtype=="Basal", ])
X
```

```

##
## Call:
## glm(formula = CN ~ log(Neo + 1) + CNA + EST + log(Age), data = MatchData[MatchData$Subtype ==
## "Basal", ])
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -1.18818 -0.27432 -0.09784  0.19276  1.56854
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.264e-01 9.663e-01  0.545  0.5871
## log(Neo + 1) -4.008e-02 8.079e-02 -0.496  0.6209
## CNA        7.633e-01 2.970e-01  2.570  0.0116 *
## EST         5.477e-05 3.645e-05  1.503  0.1361
## log(Age)   -1.626e-01 2.319e-01 -0.701  0.4847
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.2755576)
##
```

```

##      Null deviance: 29.730  on 104  degrees of freedom
## Residual deviance: 27.556  on 100  degrees of freedom
## (294 observations deleted due to missingness)
## AIC: 169.51
##
## Number of Fisher Scoring iterations: 2
# model activation scores:
## activation score
mod1A=glm(ActScore~CN+CNA+log(Neo+1)+Subtype+EST+log(Age), data=MatchData)
A1=summary(mod1A)
A1

##
## Call:
## glm(formula = ActScore ~ CN + CNA + log(Neo + 1) + Subtype +
##       EST + log(Age), data = MatchData)
##
## Deviance Residuals:
##      Min        1Q     Median        3Q       Max
## -3.10691 -0.37548  0.09469  0.48926  2.31507
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 9.031e-01 5.288e-01  1.708  0.0882 .
## CN         -1.526e-01 8.549e-02 -1.785  0.0747 .
## CNA        3.843e-01 1.879e-01  2.046  0.0412 *
## log(Neo + 1) 4.883e-02 4.898e-02  0.997  0.3192
## SubtypeHer2  3.568e-02 1.356e-01  0.263  0.7925
## SubtypeLumA -2.275e-01 1.027e-01 -2.215  0.0271 *
## SubtypeLumB -1.546e-01 1.057e-01 -1.462  0.1443
## SubtypeNormal -3.531e-01 2.222e-01 -1.589  0.1125
## EST         8.316e-04 2.367e-05 35.129 <2e-16 ***
## log(Age)    -2.472e-01 1.303e-01 -1.898  0.0582 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.580243)
##
##      Null deviance: 1198.26  on 631  degrees of freedom
## Residual deviance: 360.91  on 622  degrees of freedom
## (448 observations deleted due to missingness)
## AIC: 1461.5
##
## Number of Fisher Scoring iterations: 2
## cytotoxic score
mod1C=glm(CytoScore~CN+CNA+log(Neo+1)+Subtype+EST+log(Age), data=MatchData)
A2=summary(mod1C)
A2

##
## Call:
## glm(formula = CytoScore ~ CN + CNA + log(Neo + 1) + Subtype +
##       EST + log(Age), data = MatchData)
##

```

```

## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.75471 -0.39459  0.09727  0.48325  1.97869
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.982e+00 4.950e-01 4.003 7.02e-05 ***
## CN          -6.153e-02 8.002e-02 -0.769 0.442250
## CNA         6.549e-01 1.759e-01 3.724 0.000214 ***
## log(Neo + 1) -1.931e-02 4.585e-02 -0.421 0.673756
## SubtypeHer2 -9.803e-02 1.269e-01 -0.772 0.440190
## SubtypeLumA -2.095e-01 9.614e-02 -2.179 0.029700 *
## SubtypeLumB -2.842e-01 9.897e-02 -2.872 0.004220 **
## SubtypeNormal -2.871e-01 2.080e-01 -1.380 0.167973
## EST          9.527e-04 2.216e-05 42.989 < 2e-16 ***
## log(Age)     -2.969e-01 1.220e-01 -2.434 0.015204 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.508448)
##
## Null deviance: 1403.04 on 631 degrees of freedom
## Residual deviance: 316.25 on 622 degrees of freedom
## (448 observations deleted due to missingness)
## AIC: 1378
##
## Number of Fisher Scoring iterations: 2
## exhaustion score
mod1E=glm(ExhScore~CN+CNA+log(Neo+1)+Subtype+EST+log(Age), data=MatchData)
A3=summary(mod1E)
A3

##
## Call:
## glm(formula = ExhScore ~ CN + CNA + log(Neo + 1) + Subtype +
##       EST + log(Age), data = MatchData)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.49428 -0.46837  0.03612  0.48548  2.09539
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.381e+00 5.119e-01 2.699 0.007144 **
## CN          -1.657e-01 8.274e-02 -2.002 0.045673 *
## CNA         5.169e-01 1.819e-01 2.843 0.004623 **
## log(Neo + 1) 2.227e-02 4.741e-02 0.470 0.638766
## SubtypeHer2 -1.897e-01 1.312e-01 -1.445 0.148909
## SubtypeLumA -5.587e-01 9.940e-02 -5.620 2.88e-08 ***
## SubtypeLumB -3.920e-01 1.023e-01 -3.831 0.000141 ***
## SubtypeNormal -4.177e-01 2.150e-01 -1.943 0.052502 .
## EST          9.050e-04 2.291e-05 39.497 < 2e-16 ***
## log(Age)     -4.673e-01 1.261e-01 -3.705 0.000230 ***
## ---

```

```

## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.5435655)
##
## Null deviance: 1395.5  on 631  degrees of freedom
## Residual deviance: 338.1  on 622  degrees of freedom
## (448 observations deleted due to missingness)
## AIC: 1420.2
##
## Number of Fisher Scoring iterations: 2
## inhibition score
mod1I=glm(IinhScore~CN+CNA+log(Neo+1)+Subtype+EST+log(Age), data=MatchData)
A4=summary(mod1I)
A4

##
## Call:
## glm(formula = IinhScore ~ CN + CNA + log(Neo + 1) + Subtype +
##       EST + log(Age), data = MatchData)
##
## Deviance Residuals:
##      Min        1Q     Median        3Q       Max
## -2.54739  -0.51755   0.04802   0.53015   2.29021
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 1.304e+00 5.388e-01 2.421 0.015753 *  
## CN          -1.837e-01 8.709e-02 -2.110 0.035285 *  
## CNA         3.570e-01 1.914e-01 1.865 0.062625 .  
## log(Neo + 1) 2.561e-02 4.990e-02 0.513 0.607946    
## SubtypeHer2 -1.313e-01 1.381e-01 -0.951 0.342157    
## SubtypeLumA -4.859e-01 1.046e-01 -4.644 4.17e-06 *** 
## SubtypeLumB -3.739e-01 1.077e-01 -3.471 0.000555 *** 
## SubtypeNormal -3.706e-01 2.263e-01 -1.637 0.102052    
## EST          9.141e-04 2.412e-05 37.902 < 2e-16 *** 
## log(Age)     -5.149e-01 1.327e-01 -3.879 0.000116 *** 
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.6022173)
##
## Null deviance: 1438.40  on 631  degrees of freedom
## Residual deviance: 374.58  on 622  degrees of freedom
## (448 observations deleted due to missingness)
## AIC: 1484.9
##
## Number of Fisher Scoring iterations: 2

Plot PDL1 CN, gene expression and see if there is a relationship with CD8 expression

library(fANCOVA)

## fANCOVA 0.5-1 loaded
l1=which(MatchData$Subtype=="Basal")

```

```

require(geoR)

## Loading required package: geoR

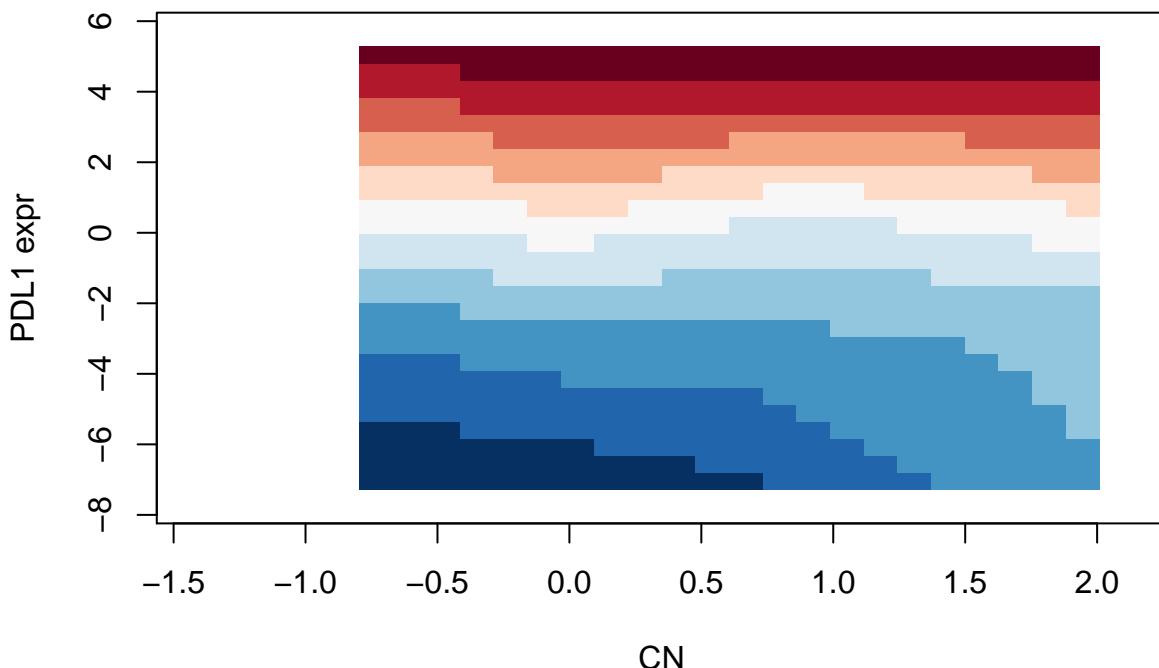
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'geor'

elevation.loess = loess(CytoScore2~CN*PDL1exp, data = MatchData[11, ], loess=0.6, degree=1)
elevation.fit = expand.grid(list(CN = seq(-1.5,2.2, length=30), PDL1exp = seq(-8, 6, length=30)))
z = predict(elevation.loess, newdata = elevation.fit)
elevation.fit$z = as.numeric(z)

image(seq(-1.5,2.2, length=30), seq(-8, 6, length=30), z,
      xlab = "CN", ylab = "PDL1 expr",
      main = "Surface elevation data", col=HMPallete[11:1])

```

**Surface elevation data**

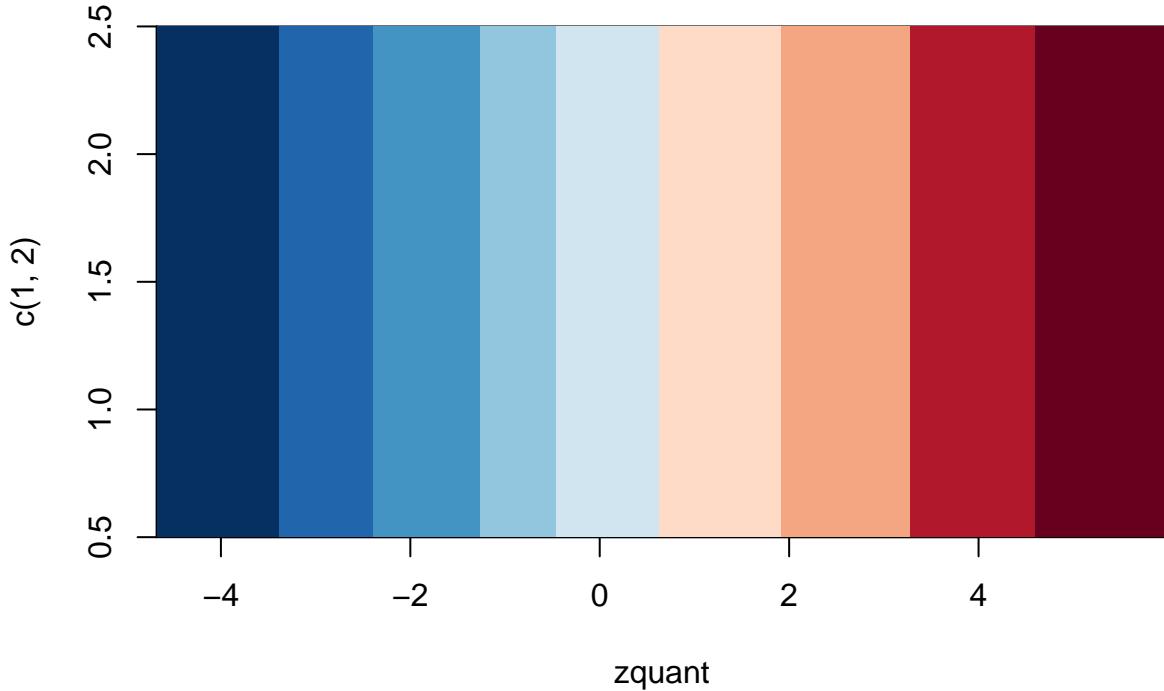


```

# plot the colour bar
zquant=quantile(z, probs=seq(0, 1, length=10), na.rm=TRUE)

image( zquant,c(1,2), cbind(zquant, zquant), col=HMPallete[11:1])

```



```

A=seq(-1.5,2.2, length=30)
B=seq(-8, 6, length=30)
spacing=matrix(NA, length(A)-1, length(B)-1)

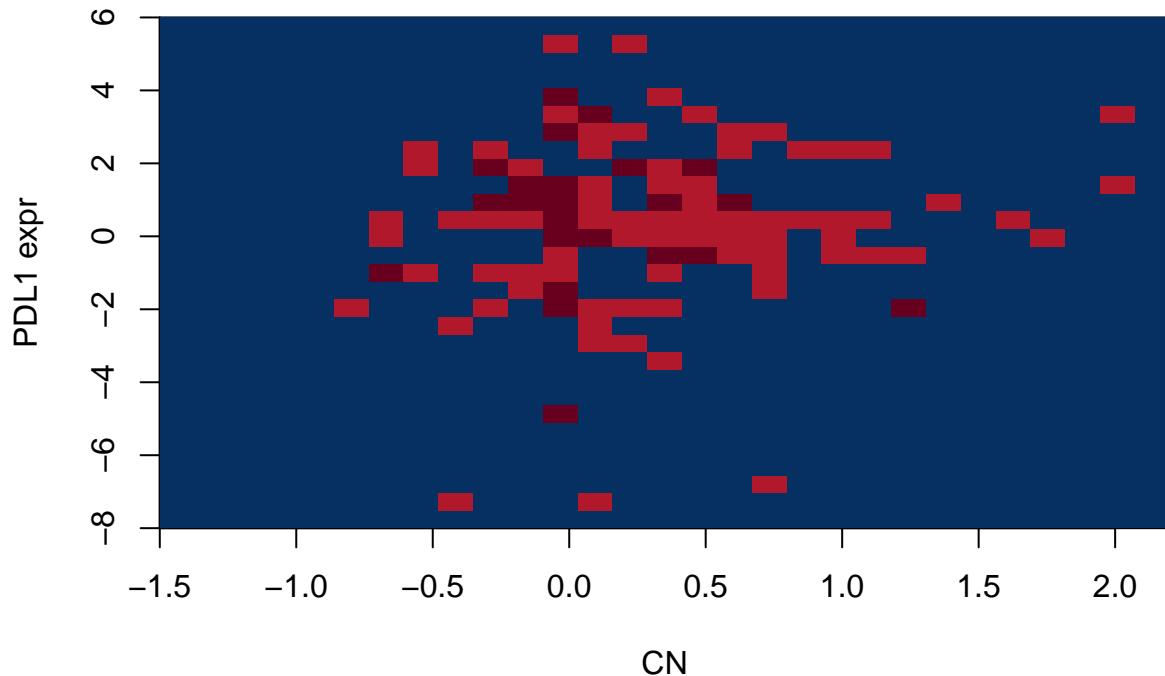
for(i in 1:29){
  for(j in 1:29){
    spacing[i,j]=length(which(MatchData$CN[11]>=A[i] & MatchData$CN[11]<A[i+1] &
      MatchData$PDL1exp[11]>=B[j]&MatchData$PDL1exp[11]<B[j+1]))
  }
}

spacing[spacing>0]=spacing[spacing>0]+20

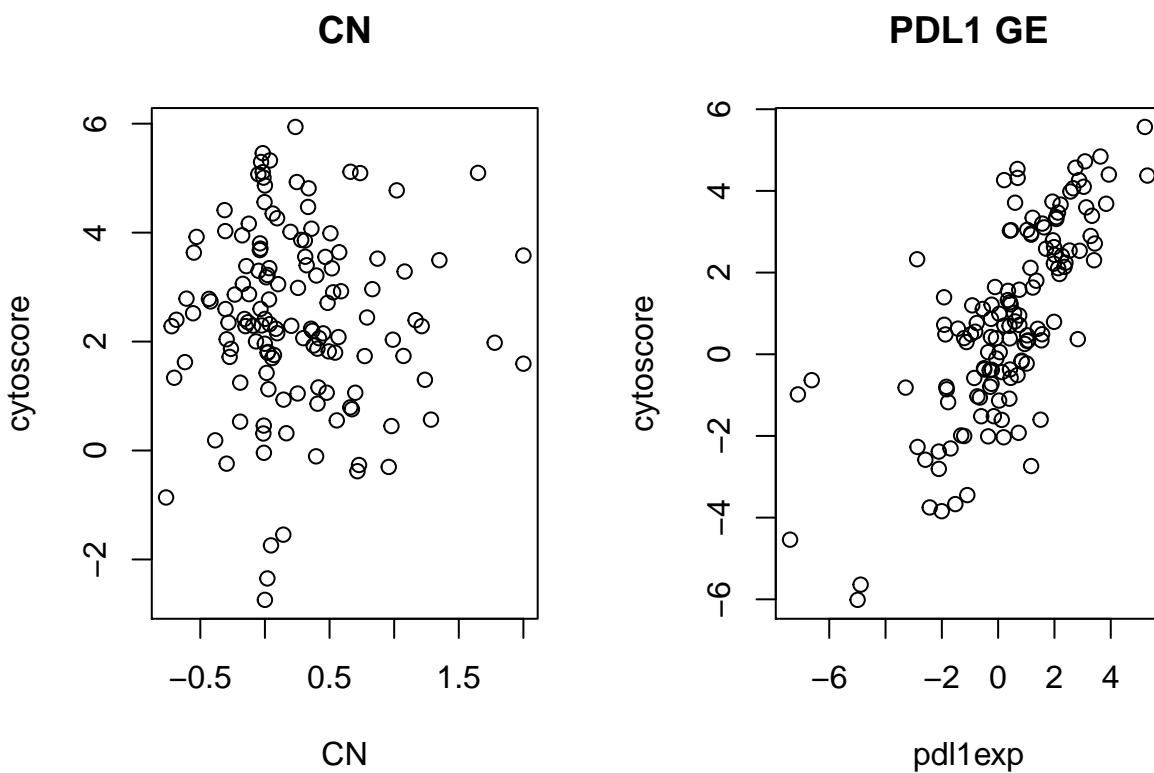
image(seq(-1.5,2.2, length=30), seq(-8, 6, length=30), spacing,
  xlab = "CN", ylab = "PDL1 expr",
  main = "Surface elevation data", col=HMPallete[11:1])

```

### Surface elevation data



```
par(mfrow=c(1,2))
plot(MatchData$CN[11], MatchData$CytoScore[11], main="CN", ylab="cytoscore", xlab="CN")
plot(MatchData$PDL1exp[11], MatchData$CytoScore2[11], main="PDL1 GE", ylab="cytoscore", xlab="pdl1exp")
```



```
a1=lm(CytoScore2~CN*PDL1exp+Purity, data=MatchData)
```

## 7.2 Oslo cohort

Look at the PD1 locus from data from Oslo:

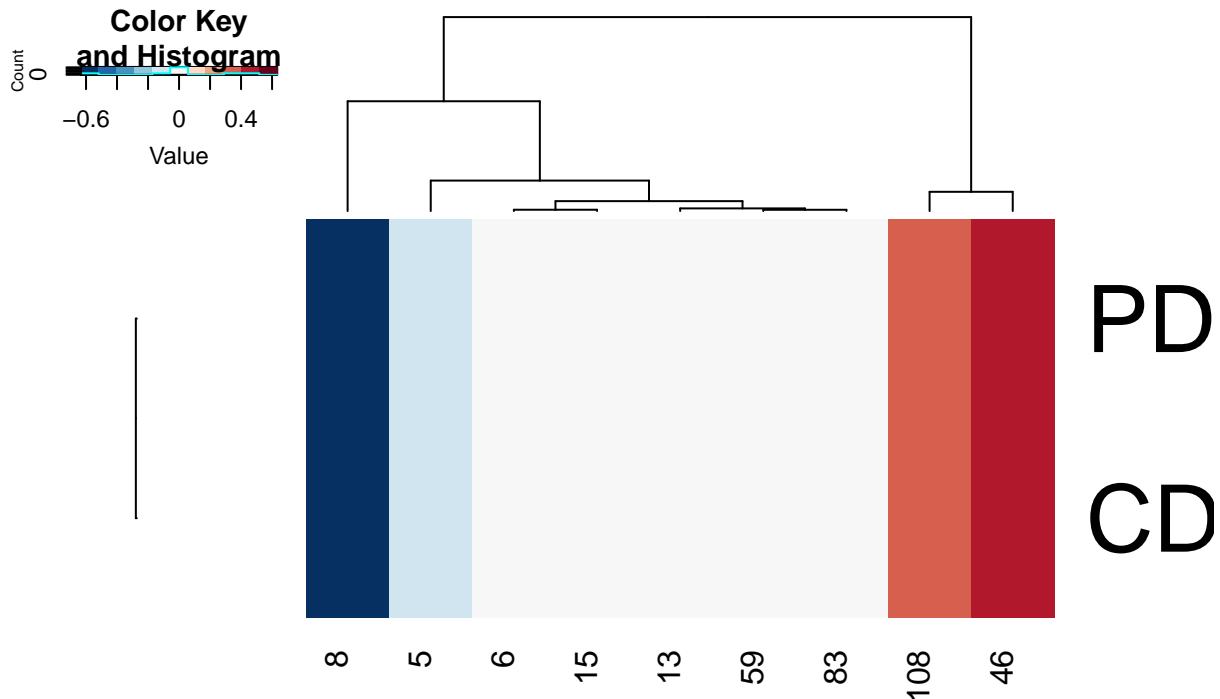
```
OsloGist2=read.csv("data/Oslo_SNP_data/pdls_adj_logr_polyak.csv")
```

```
nidx=which(OsloGist2$tissue=="DCIS")
par(mfrow=c(1,2))
```

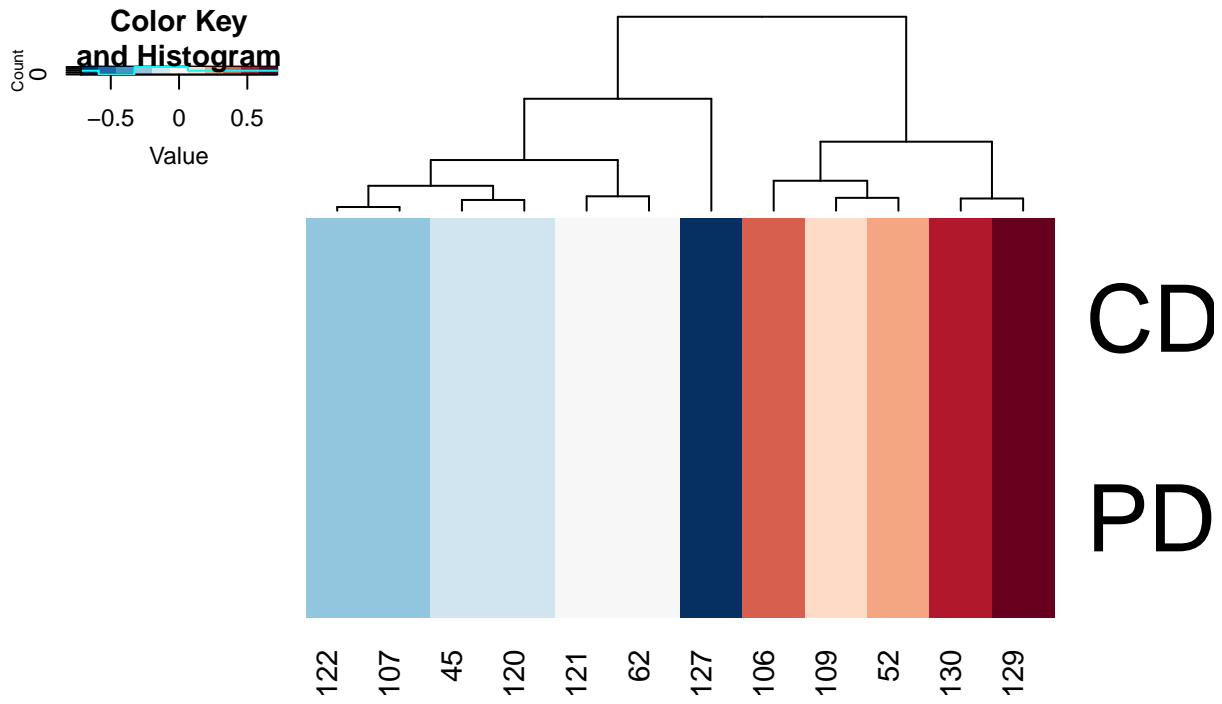
```
OsIdNew=OsloGist2$subtype
OsIdNew=gsub("Basal", "red", OsIdNew)
OsIdNew=gsub("Her2", "pink", OsIdNew)
OsIdNew=gsub("LumA", "blue", OsIdNew)
OsIdNew=gsub("LumB", "blue", OsIdNew)
OsIdNew=gsub("Normal", "green", OsIdNew)
```

```
# check the frequency in the different samples
```

```
heatmap.2(t(data.matrix(OsloGist2[which(OsloGist2$tissue=="DCIS" & OsIdNew=="red")],c(4:5))), trace="none")
```

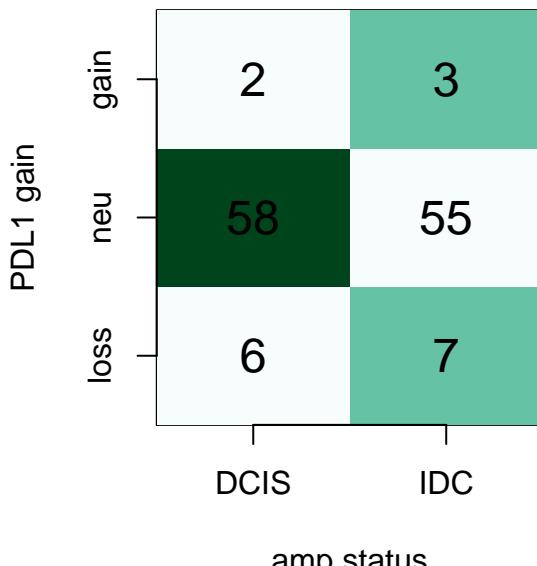


```
heatmap.2(t(data.matrix(OsloGist2[which(OsloGist2$tissue=="IDC" & OsIdNew=="red") ,c(4:5)])), trace="none")
```

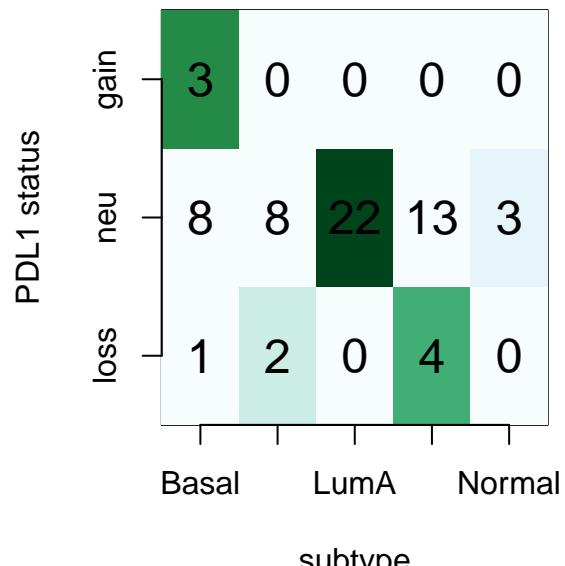


```
# table of frequencies in DCIS vs IDC
OsloPD1amp=cut(OsloGist2[,4], c(-5, -0.3, 0.3, 5), c("loss", "neu", "gain"))
par(mfrow=c(1,2))
ContTable(table(OsloGist2$tissue, OsloPD1amp), title="PD1 locus", chisqtest = T, xlabL = "amp status", ylabL = "PDL1 gain")
# table in IDC cases only:
ContTable(table(OsloGist2$subtype[-nidx], OsloPD1amp[-nidx]), title="PDL1 IDC", chisqtest = T, xlabL = "amp status", ylabL = "PDL1 status")
```

**PD1 locus Chisq = 0.84**

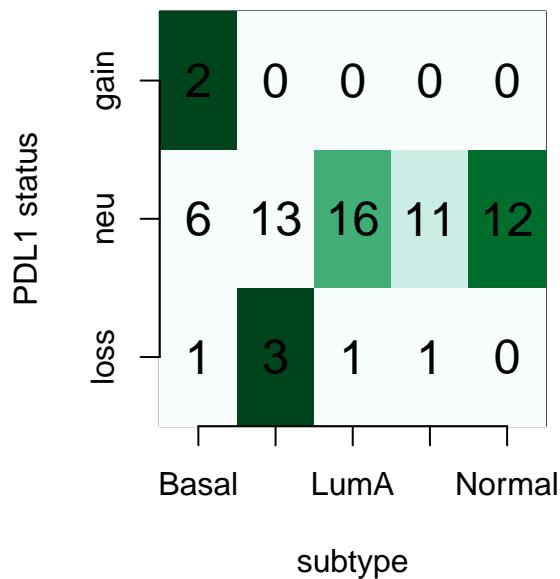


**PDL1 IDC Chisq = 0.01**



```
#table DCIS only
ContTable(table(OsloGist2$subtype[nidx], OsloPD1amp[nidx]), title="PDL1 IDC", chisqtest = T, xlabL = "subtype", ylabL = "PDL1 status")
```

## PDL1 IDC Chisq = 0.04

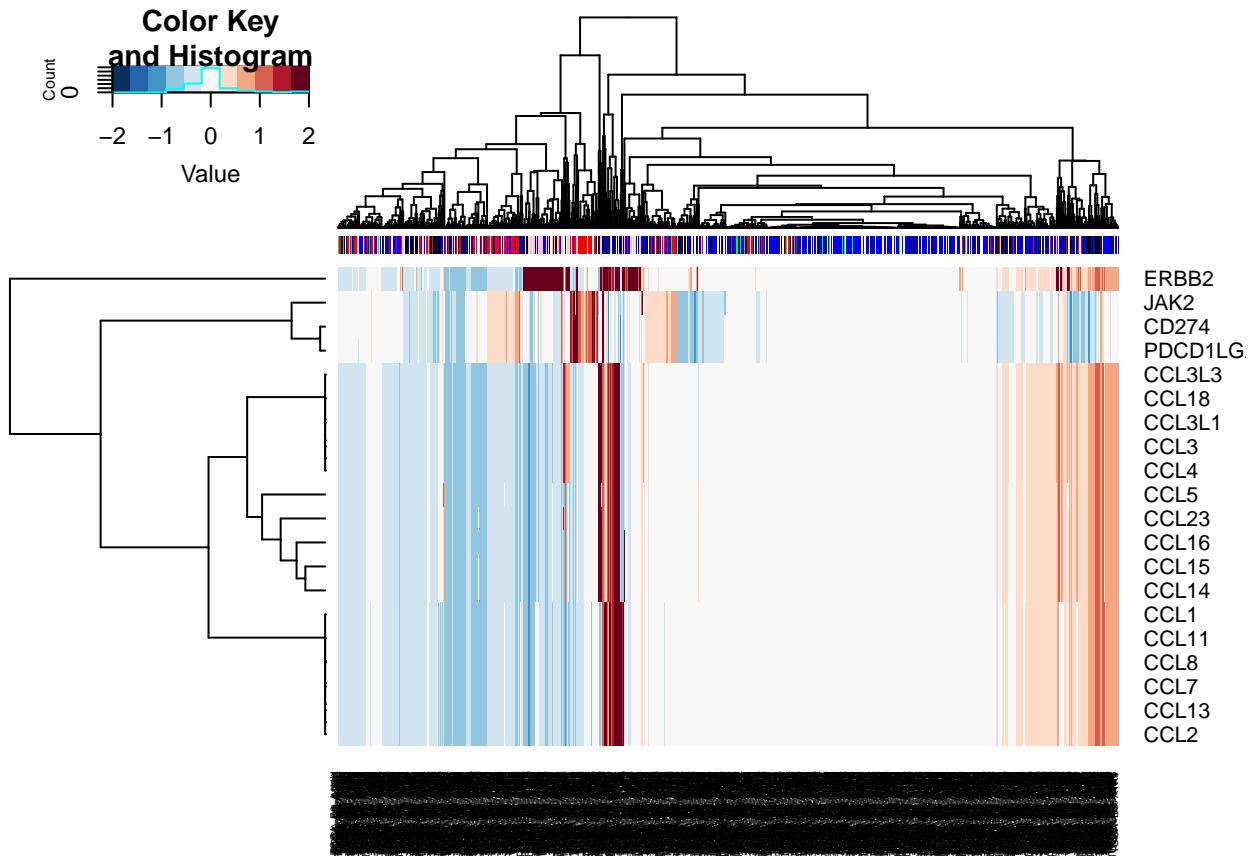


## 8 Frequency of co-amplification with the CCL data set

### 8.1 TCGA data set: Figure S7

To do this, load Gistic data from the TCGA and location of genes of interest. Add information on Patient subtype by loading in TCGA clinical information. (Also available at firebrowse.org)

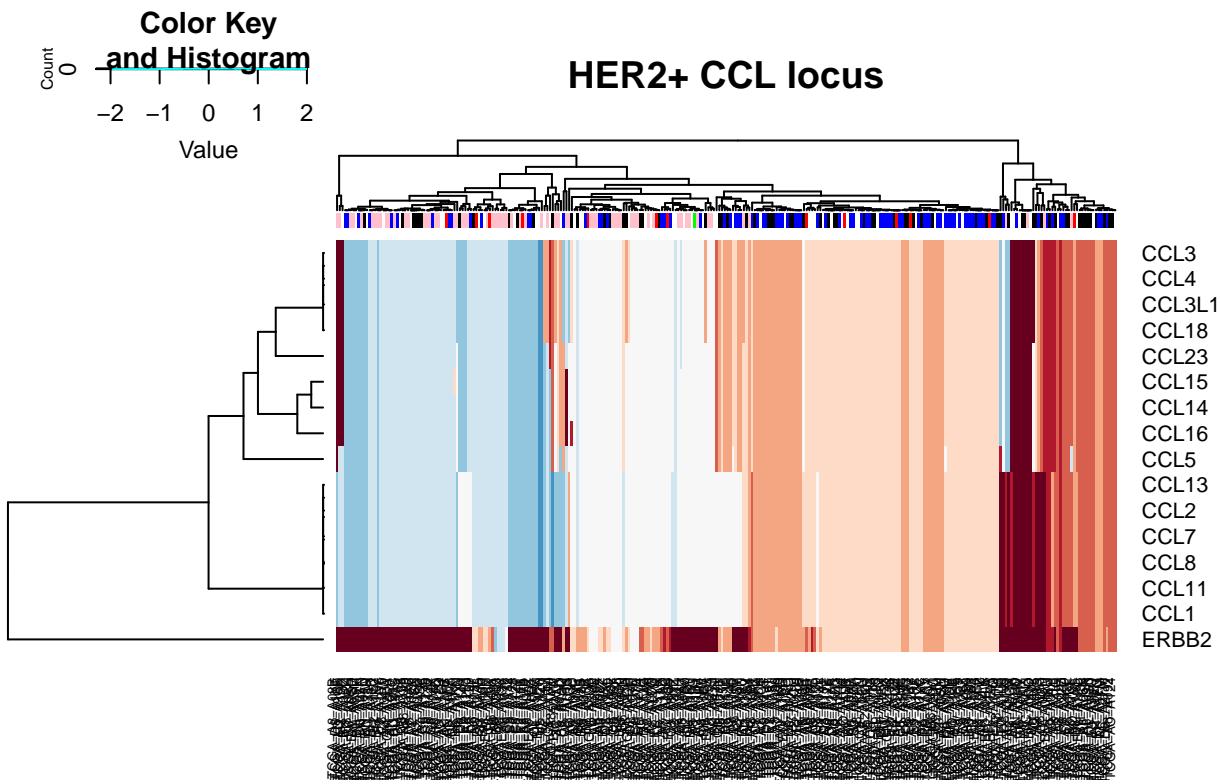
```
heatmap.2(data.matrix(NewGistic[ -c(18:20,23),]), col=HMPallte[11:1], scale="none", ColSideColors = Co
```



### 8.1.1 CCL coamplification: Figure 5

More closely examine HER2 positive cases, or any with amplification: We will need to specify a cut-off for ERBB2 amplification - here it is set at 0.3, which is often used for Gistic Scores

```
library(gplots)
BasHER2=as.character(TCGAClin$PATIENT_ID[which(TCGAClin$PAM50=="Her2")])
ERBamp=colnames(NewGistic)[which(NewGistic["ERBB2", ]>0.3)]
PatID=unique(c(BasHER2, ERBamp))
n1=match(PatID, colnames(NewGistic))
a2=heatmap.2(data.matrix(NewGistic[c(1:16) ,n1]), col=HMPallete[11:1], scale="none", ColSideColors = Co)
```



```

colA=rep("black", length(n1))
colA[a2$colInd[1:148]]="pink"
colA[a2$colInd[149:243]]="red"

grpA=colnames(NewGistic[c(1:16),n1[a2$colInd[1:148]]])
grpB=colnames(NewGistic[c(1:16),n1[a2$colInd[149:243]]])
grpC=colnames(NewGistic[c(1:16),n1[a2$colInd[244:286]]])

```

Calculate the chisq values for frequency of co-amplification in HER2 samples

```

PDScores=colMeans(NewGistic[c(1:15),])
PDscoresAmp=cut(PDScores, c(-2, -0.3, 0.3, 5), c("loss", "neu", "gain"))
anew=rep("other", length(PDscoresAmp))
anew[n1]="HER2"
ContTable(table(anew, PDscoresAmp), title="CCL amplification", chisqtest = T, xlabL = "Subtype", ylabL = "Genes", xlabR = "Chisq", ylabR = "P-value", xtext = "Chisq", ytext = "P-value", xtextR = "P-value", ytextR = "Chisq", xtextY = 0.5, ytextY = 0.5, xtextRY = 0.5, ytextRY = 0.5)

```

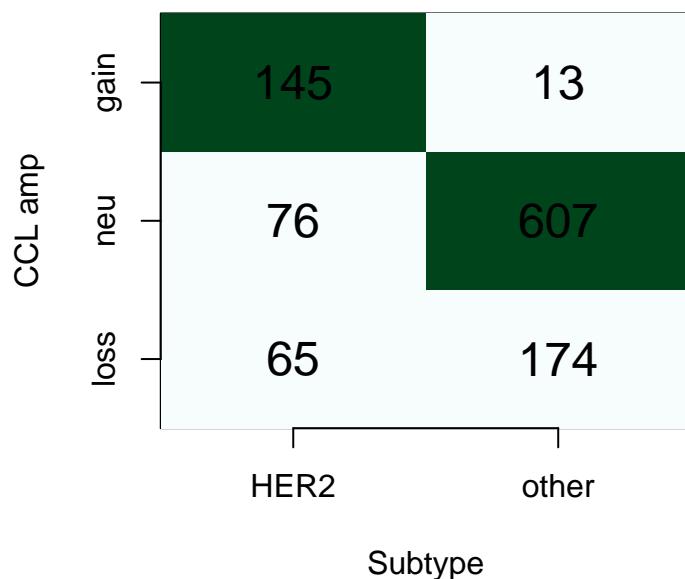
```

## Warning in if (chisqtest == T) {: the condition has length > 1 and only the
## first element will be used

## Warning in if (chisqtest == T) {: the condition has length > 1 and only the
## first element will be used

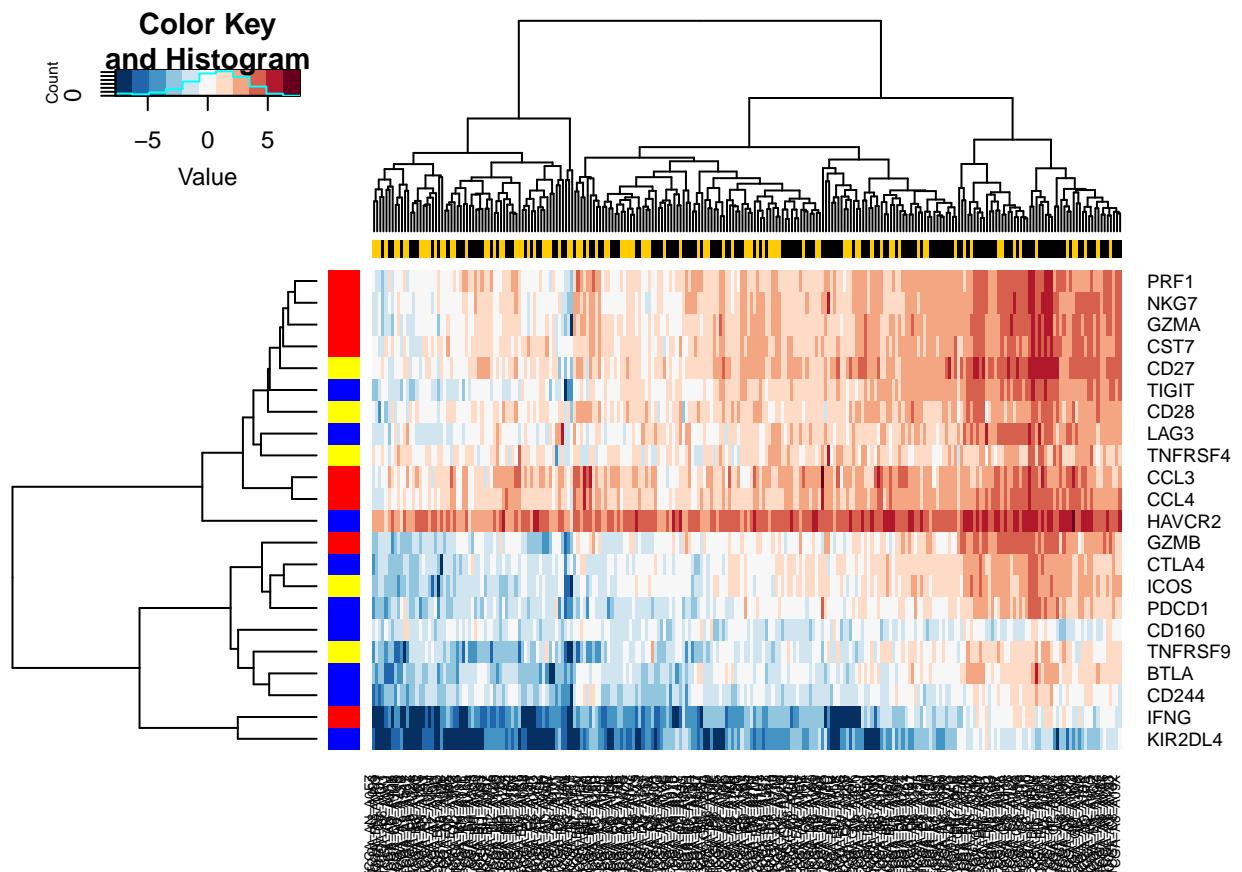
```

## CCL amplification Chisq = 0



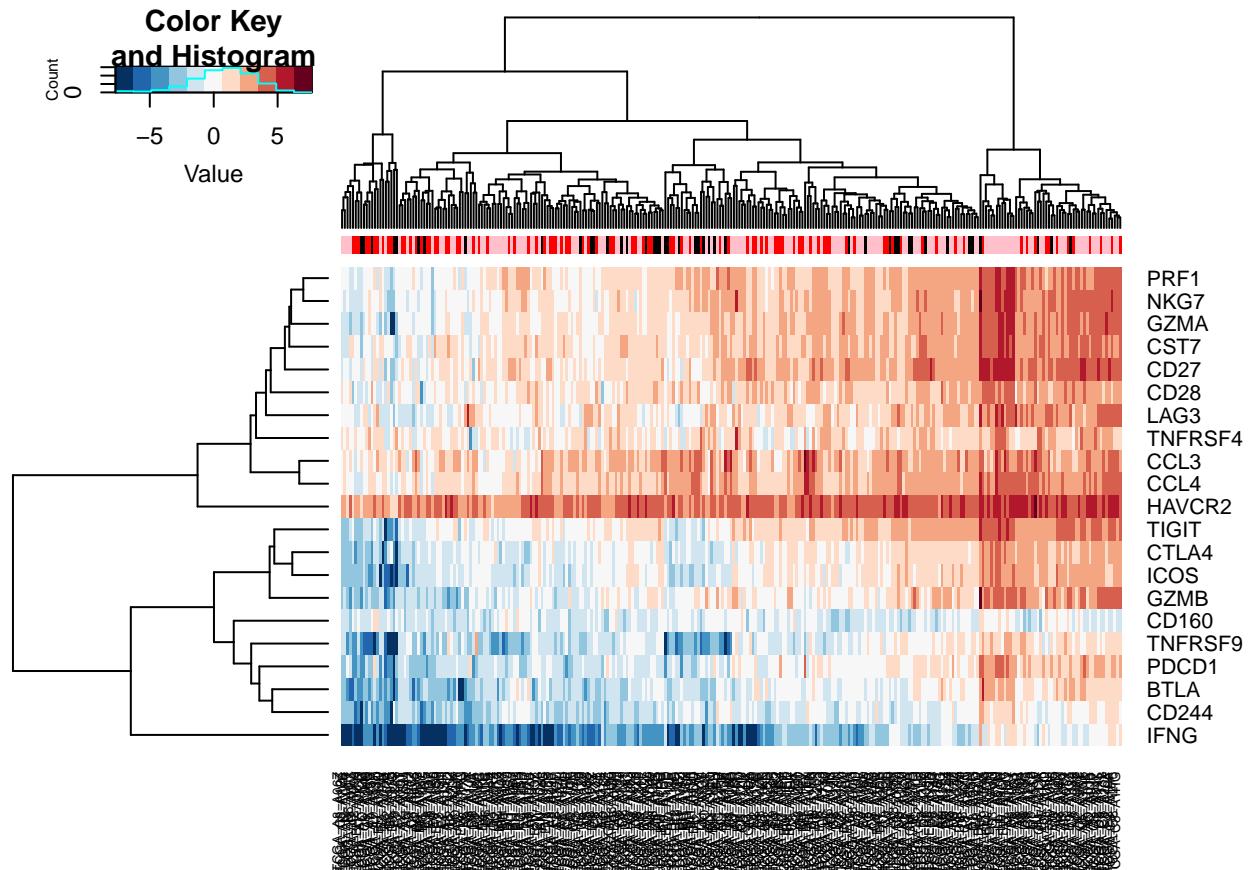
## 8.2 TCGA RNA-seq data

RNAseq data:



```
# Plot heatmap of activation related genes with HER2/Lum/amp status:
Pid1=match(c(grpA, grpB, grpC), colnames(TCGAlogCPM))
RNA2=TCGAlogCPM[match(c(GeneListAct$Activated, GeneListAct$Exhausted, GeneListAct$Cytotoxic), rownames(TCGAlogCPM))]

heatmap.2(RNA2, trace="none", col=HMPallete[11:1], ColSideColors = c(rep("pink", length(grpA)), rep("red", length(grpB)), rep("blue", length(grpC))), scale="none")
```



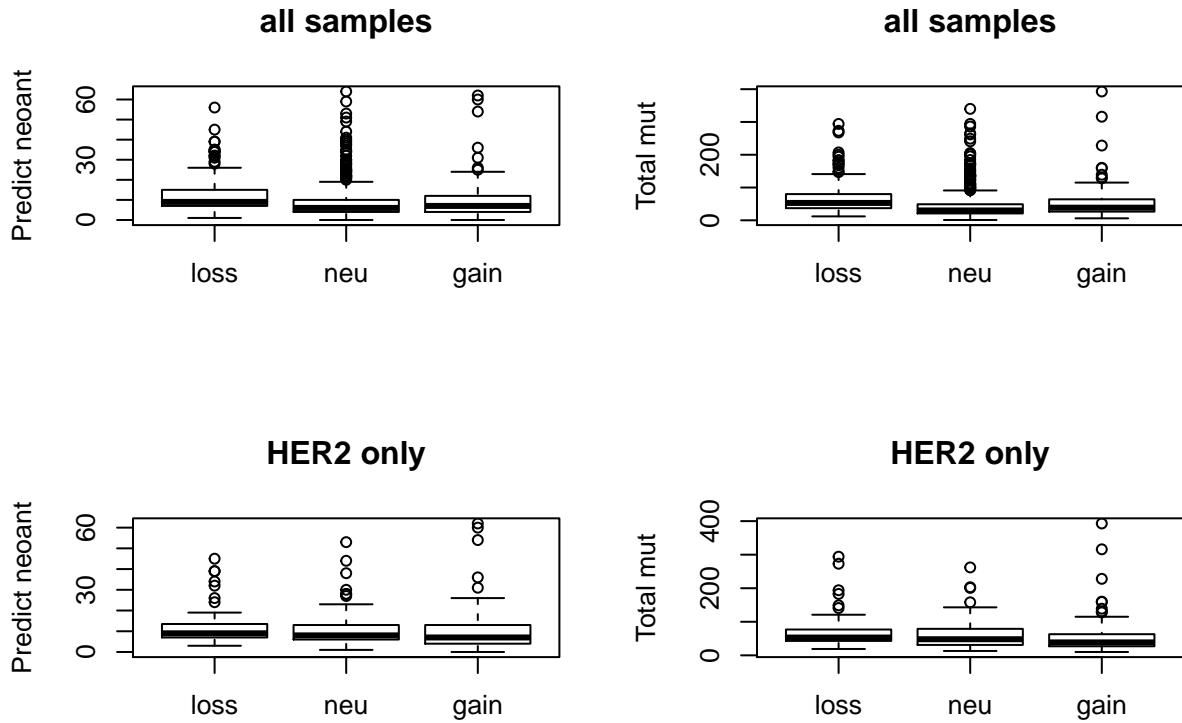
### 8.3 TCGA neoantigen data

This was obtained from supplementary information Table S4A from Rooney et al 2015.

```
mData=match(colnames(NewGistic), NeoData$PatientID)

## construct a boxplot of antigen load for example
par(mfrow=c(2,2))
boxplot(NeoData$Predicted.NeoAgs[mData]~PDscoresAmp, ylab="Predict neoant", main="all samples")
boxplot(NeoData$Total.Mutations[mData]~PDscoresAmp, ylab="Total mut", main="all samples")

## refine search to HER2 cases only
boxplot(NeoData$Predicted.NeoAgs[mData[anew=="HER2"]]~PDscoresAmp[anew=="HER2"], ylab="Predict neoant",
boxplot(NeoData$Total.Mutations[mData[anew=="HER2"]]~PDscoresAmp[anew=="HER2"], ylab="Total mut", main=
```



## 8.4 Linear model for TCGA dataset

Make a linear model to explain exhaustion or activation status:

```
MatchData=data.frame(Pat=colnames(NewGistic),CN=PDScores,
ActScore=ActivationScore[match(colnames(NewGistic), colnames(TCGAlogCPM))],
CytoScore=CytoScore[match(colnames(NewGistic), colnames(TCGAlogCPM))],
ExhScore=ExhScore[match(colnames(NewGistic), colnames(TCGAlogCPM))],
InhScore=InhScore[match(colnames(NewGistic), colnames(TCGAlogCPM))],
ERBB2=t(NewGistic[["ERBB2", ]]),
Neo=NeoData$Predicted.NeoAgs[match(colnames(NewGistic), NeoData$PatientID)],
CNA=TCGAClin$Copy.Number.Alterations[match(colnames(NewGistic), TCGAClin$PATIENT_ID)],
Subtype=TCGAClin$PAM50[match(colnames(NewGistic), TCGAClin$PATIENT_ID)],
EST=ESTscore$ImmuneScore[match(colnames(NewGistic), ESTscore$Description)],
Age=as.numeric(as.character(TCGAClin$Age[match(colnames(NewGistic), TCGAClin$PATIENT_ID)]))

MatchData$Subtype = factor(MatchData$Subtype,levels(MatchData$Subtype)[c(2, 1, 3:5)])

# Model summary:
mod1B=glm(CN~ERBB2+log(Neo+1)+Subtype+log(Age) , data=MatchData)
X=summary(mod1B)
X

##
## Call:
## glm(formula = CN ~ ERBB2 + log(Neo + 1) + Subtype + log(Age),
##      data = MatchData)
##
## Deviance Residuals:
##       Min        1Q     Median        3Q       Max
## -10.000000 -0.000000  0.000000  0.000000  0.000000
```

```

## -1.29241 -0.11548 -0.00722  0.13264  2.10180
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.83805  0.26990 -3.105 0.001988 **
## ERBB2        0.22520  0.02407  9.356 < 2e-16 ***
## log(Neo + 1) -0.05906  0.02321 -2.544 0.011191 *
## SubtypeBasal  0.27087  0.07503  3.610 0.000331 ***
## SubtypeLumA   0.42937  0.06703  6.406 2.95e-10 ***
## SubtypeLumB   0.46717  0.06738  6.933 1.03e-11 ***
## SubtypeNormal 0.33500  0.11849  2.827 0.004844 **
## log(Age)      0.12757  0.06504  1.961 0.050282 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1453391)
##
## Null deviance: 112.090 on 631 degrees of freedom
## Residual deviance: 90.692 on 624 degrees of freedom
## (448 observations deleted due to missingness)
## AIC: 584.56
##
## Number of Fisher Scoring iterations: 2
# Narrow this down to just samples which have the HER2 amplicon
l1=match(c(grpA, grpB, grpC), MatchData$Pat)

# model rna expression expression
## activation signature
mod1A=glm(ActScore~CN+log(Neo+1)+Subtype+EST+log(Age), data=MatchData[l1, ])
summary(mod1A)

##
## Call:
## glm(formula = ActScore ~ CN + log(Neo + 1) + Subtype + EST +
##       log(Age), data = MatchData[l1, ])
##
## Deviance Residuals:
##      Min        1Q     Median        3Q       Max
## -2.5426  -0.3207   0.0439   0.4428   2.2136
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.484e+00  8.703e-01  1.705  0.0899 .
## CN         -4.647e-02  9.441e-02 -0.492  0.6231
## log(Neo + 1) 2.748e-02  8.158e-02  0.337  0.7367
## SubtypeBasal -1.701e-02  2.251e-01 -0.076  0.9398
## SubtypeLumA  -1.730e-01  1.409e-01 -1.228  0.2212
## SubtypeLumB  -9.626e-02  1.477e-01 -0.652  0.5154
## SubtypeNormal 5.528e-01  7.084e-01  0.780  0.4363
## EST          7.998e-04  4.273e-05 18.719 <2e-16 ***
## log(Age)     -3.229e-01  2.179e-01 -1.482  0.1402
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

```

## 
## (Dispersion parameter for gaussian family taken to be 0.4854993)
##
##     Null deviance: 275.64  on 188  degrees of freedom
## Residual deviance:  87.39  on 180  degrees of freedom
##   (97 observations deleted due to missingness)
## AIC: 410.57
##
## Number of Fisher Scoring iterations: 2
## cytotoxic signature
mod1C=glm(CytoScore~CN+log(Neo+1)+Subtype+EST+log(Age), data=MatchData[11, ])
summary(mod1C)

##
## Call:
## glm(formula = CytoScore ~ CN + log(Neo + 1) + Subtype + EST +
##       log(Age), data = MatchData[11, ])
##
## Deviance Residuals:
##      Min        1Q        Median         3Q        Max
## -1.85318  -0.37259   0.01367   0.45942   1.57827
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 1.807e+00 8.224e-01  2.198  0.0292 *  
## CN          1.925e-01 8.921e-02  2.158  0.0322 *  
## log(Neo + 1) -7.207e-02 7.709e-02 -0.935  0.3511  
## SubtypeBasal 4.704e-02 2.127e-01  0.221  0.8252  
## SubtypeLumA -2.458e-01 1.332e-01 -1.846  0.0665 .  
## SubtypeLumB -9.619e-02 1.396e-01 -0.689  0.4915  
## SubtypeNormal 4.943e-01 6.694e-01  0.738  0.4612  
## EST          1.030e-03 4.037e-05 25.516 <2e-16 *** 
## log(Age)     -2.089e-01 2.059e-01 -1.014  0.3117  
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.4335005)
##
##     Null deviance: 383.65  on 188  degrees of freedom
## Residual deviance:  78.03  on 180  degrees of freedom
##   (97 observations deleted due to missingness)
## AIC: 389.16
##
## Number of Fisher Scoring iterations: 2
## exhaustion signature
mod1E=glm(ExhScore~CN+log(Neo+1)+Subtype+EST+log(Age), data=MatchData[11, ])
summary(mod1E)

##
## Call:
## glm(formula = ExhScore ~ CN + log(Neo + 1) + Subtype + EST +
##       log(Age), data = MatchData[11, ])
##
## Deviance Residuals:
```

```

##      Min       1Q    Median       3Q      Max
## -1.77558 -0.43640 -0.00196  0.55993  1.60795
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.850e-01 9.190e-01  0.637  0.5252
## CN          7.089e-02 9.969e-02  0.711  0.4779
## log(Neo + 1) 1.938e-02 8.615e-02  0.225  0.8223
## SubtypeBasal 1.146e-01 2.377e-01  0.482  0.6304
## SubtypeLumA -3.321e-01 1.488e-01 -2.232  0.0269 *
## SubtypeLumB -1.066e-01 1.559e-01 -0.683  0.4953
## SubtypeNormal 2.799e-01 7.481e-01  0.374  0.7087
## EST         9.519e-04 4.512e-05 21.098 <2e-16 ***
## log(Age)     -2.801e-01 2.301e-01 -1.217  0.2251
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.5413373)
##
## Null deviance: 365.196 on 188 degrees of freedom
## Residual deviance: 97.441 on 180 degrees of freedom
## (97 observations deleted due to missingness)
## AIC: 431.15
##
## Number of Fisher Scoring iterations: 2
## inhibition signature
mod1I=glm(InhScore~CN+log(Neo+1)+Subtype+EST+log(Age), data=MatchData[11, ])
summary(mod1I)

##
## Call:
## glm(formula = InhScore ~ CN + log(Neo + 1) + Subtype + EST +
##       log(Age), data = MatchData[11, ])
##
## Deviance Residuals:
##      Min       1Q    Median       3Q      Max
## -1.7979 -0.4949  0.0000  0.5995  1.7067
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.4847239 0.9449912  0.513  0.6086
## CN          0.0776753 0.1025126  0.758  0.4496
## log(Neo + 1) 0.0182187 0.0885853  0.206  0.8373
## SubtypeBasal -0.0324532 0.2443827 -0.133  0.8945
## SubtypeLumA -0.3702346 0.1530340 -2.419  0.0165 *
## SubtypeLumB -0.1535108 0.1603608 -0.957  0.3397
## SubtypeNormal 0.0543063 0.7692516  0.071  0.9438
## EST         0.0009503 0.0000464 20.483 <2e-16 ***
## log(Age)     -0.3132580 0.2366327 -1.324  0.1872
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.5724283)
##

```

```

##      Null deviance: 371.68  on 188  degrees of freedom
## Residual deviance: 103.04  on 180  degrees of freedom
## (97 observations deleted due to missingness)
## AIC: 441.7
##
## Number of Fisher Scoring iterations: 2

```

## 8.5 Oslo Dataset: Figure 4

The Oslo dataset contains CN information from both DCIS and IDC samples. Please contact the corresponding author if you wish to access this information.

```

OsloGist=read.csv("data/Oslo_SNP_data/ccls_her2_adj_cn_polyak.csv")
OsloGistLogR=read.csv("data/Oslo_SNP_data/ccls_her2_adj_logr_polyak.csv")

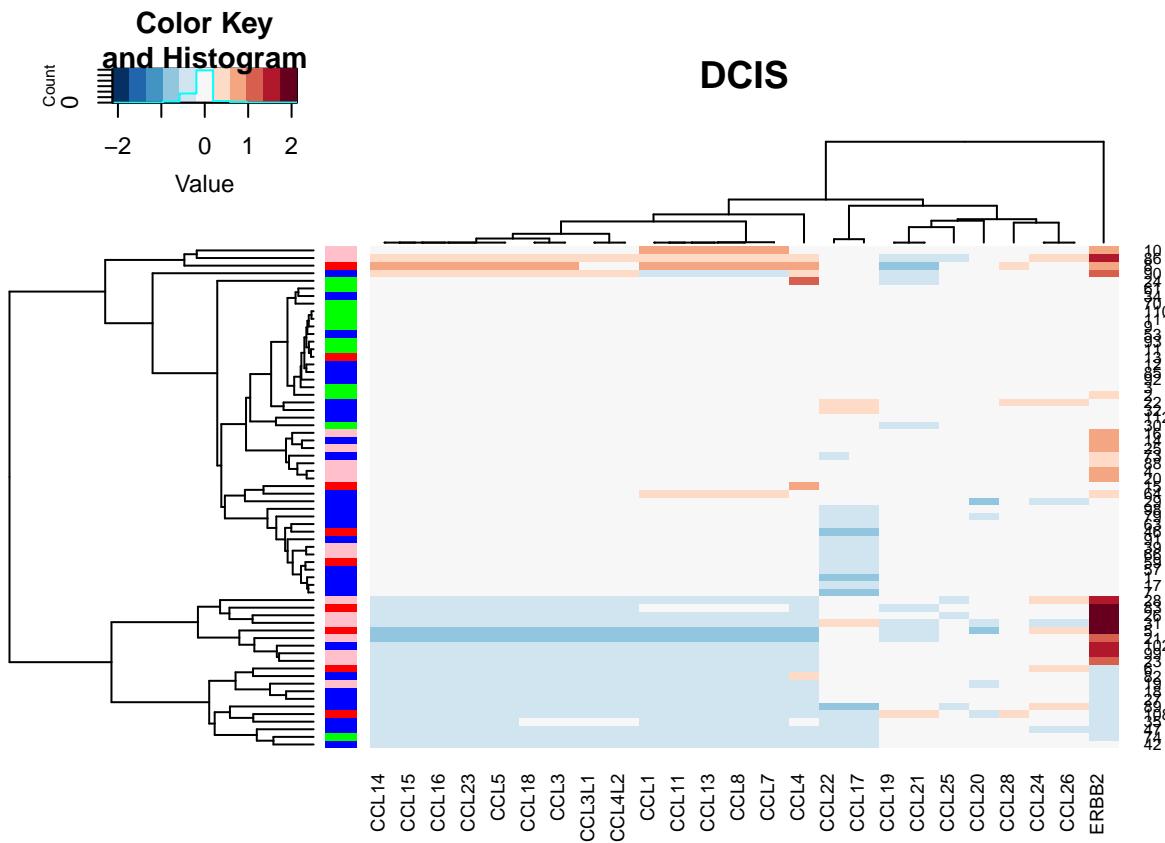
# heatmap of LogRatios
HER2scale=brewer.pal(5, "Greens")
N2=cut(OsloGistLogR$adj_cn_HER2, c(-5, -0.5, 0.5, 2.5, 10, 100), labels=HER2scale)

nidx=which(OsloGist$tissue=="DCIS")
par(mfrow=c(1,2))

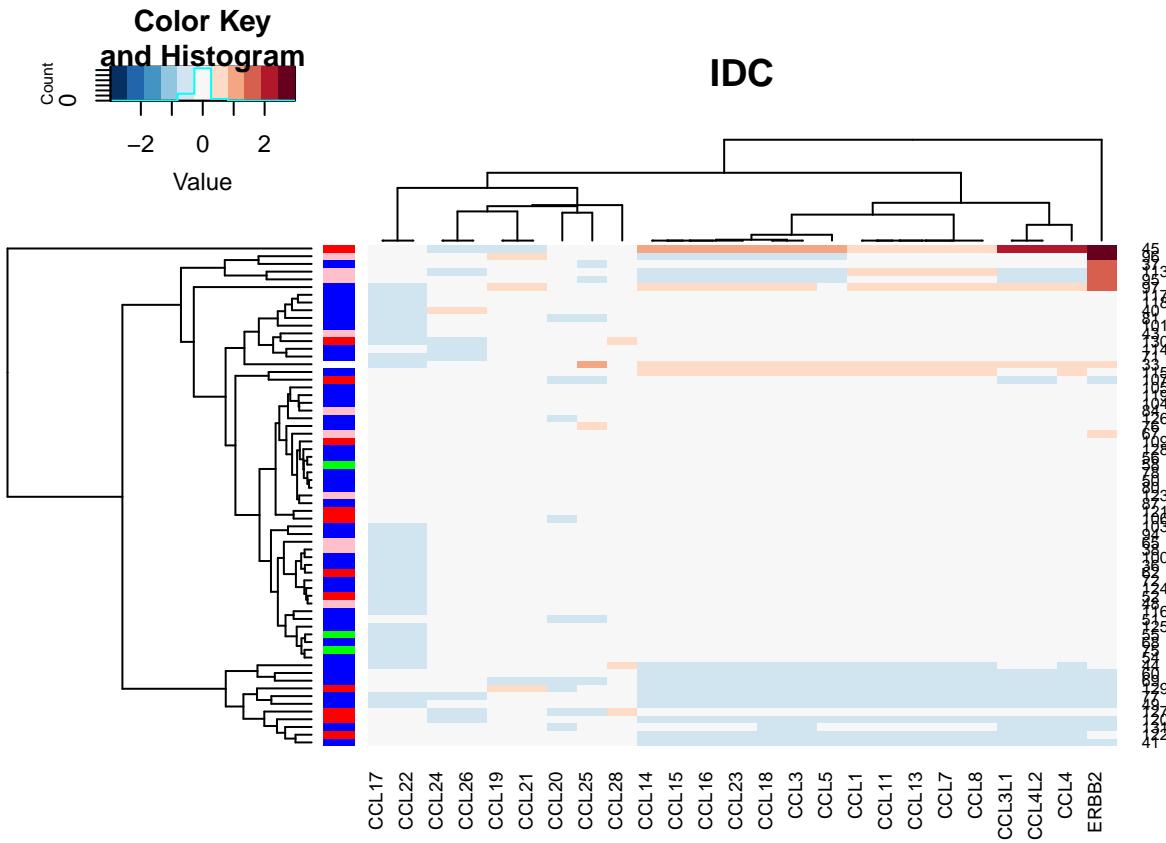
OsIdNew=OsloGistLogR$subtype
OsIdNew=gsub("Basal", "red", OsIdNew)
OsIdNew=gsub("Her2", "pink", OsIdNew)
OsIdNew=gsub("LumA", "blue", OsIdNew)
OsIdNew=gsub("LumB", "blue", OsIdNew)
OsIdNew=gsub("Normal", "green", OsIdNew)

heatmap.2(data.matrix(OsloGistLogR[nidx ,-c(1:7)]), col=HMPallate[11:1],
           scale="none",trace="none", main="DCIS",
           RowSideColors =OsIdNew[nidx])

```



```
heatmap.2(data.matrix(OsloGistLogR[-nidx ,-c(1:7)]), col=HMPallete[11:1] ,
           scale="none", trace="none", RowSideColors =OsIdNew[-nidx],
           main="IDC")
```



%Calculate the percentages of samples with or without co-amplification? Take a conservative value of 0.1 as amplification or loss?? Also, reanalyse with her2 amplification as value of 3 or above

```
# find in the DCIS case
a1=match(Genes$Locus[1:15] , colnames(OsloGistLogR))

OsGismeanl=rowMeans(OsloGistLogR[,na.omit(a1)])
OsGismeanl2=cut(OsGismeanl, c(-5, -0.3, 0.3, 5), c("loss", "neu", "gain"))
HER2CNnew=cut(OsloGistLogR$adj_cn_HER2, c(-10, 2.5, 1000), c("not-amp", "amp"))

# DCIS cases:
table(OsloGistLogR$her2_amplified[nidx], OsGismeanl2[nidx])

##          loss neu gain
##   amp      8   3   0
##   non-amp  9  45   1

# IDC case:
table(OsloGistLogR$her2_amplified[-nidx], OsGismeanl2[-nidx])

##          loss neu gain
##   amp      1   3   2
##   non-amp  7  50   2

par(mfrow=c(1,2))
# cut off of 4 for amplification
ContTable(table(OsloGistLogR$her2_amplified[nidx],OsGismeanl2[nidx]), title = "DCIS", xlabL = "HER2 amp")
```

```

## Warning in if (chisqtest == T) {: the condition has length > 1 and only the
## first element will be used

## Warning in chisq.test(tab): Chi-squared approximation may be incorrect

## Warning in if (chisqtest == T) {: the condition has length > 1 and only the
## first element will be used

ContTable(table(OsloGistLog$her2_amplified[-nidx], OsGismeanl2[-nidx]), title = "IDC", xlabL = "HER2 an

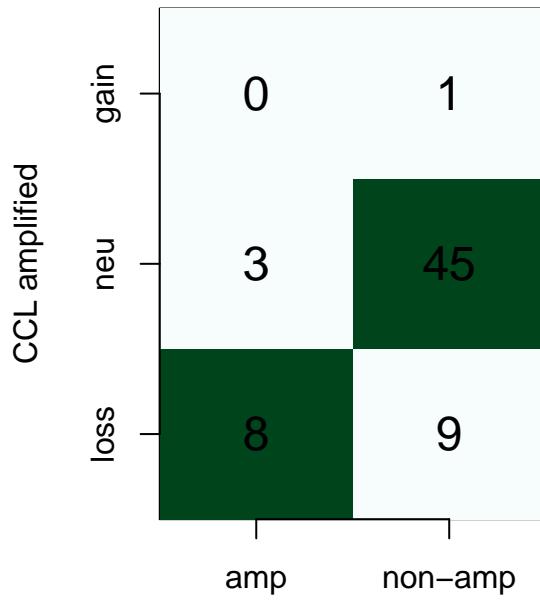
```

## Warning in if (chisqtest == T) {: the condition has length > 1 and only the  
## first element will be used

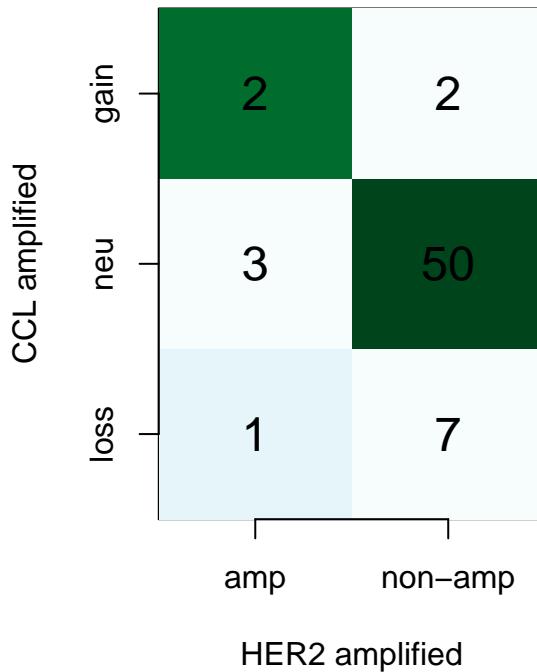
## Warning in chisq.test(tab): Chi-squared approximation may be incorrect

## Warning in if (chisqtest == T) {: the condition has length > 1 and only the  
## first element will be used

**DCIS Chisq = 0**



**IDC Chisq = 0.01**



## 9 CCL-ERBB2 amplification based on FISH images: Figure 5

### 9.1 DCIS analysis

First read in pathology information and summary of our image data

```
all.mat<-read.csv("data/DCIS_CCLdata_new.csv", stringsAsFactors = F)
```

#### 9.1.1 Diversity

Now compute the diversity for each individual image

```

attach(all.mat)
diversity.mat <- NULL
for (i in 1:20) {
  indices <- unique(all.mat$index[all.mat$sample_ID == i])
  for (ind in indices) {
    temp.mat <- all.mat[all.mat$index == ind & all.mat$sample_ID == i, ]
    cep17.abs <- table(temp.mat$CEP17)
    her2.abs <- table(temp.mat$HER2)
    ccluster.abs <- table(temp.mat$CCluster)
    her2.ratio <- table(temp.mat$HER2 / temp.mat$CEP17)
    ccluster.ratio <- table(temp.mat$CCluster / temp.mat$CEP17)
    df = as.data.frame(cbind(temp.mat$CCluster, temp.mat$CEP17))
    colnames(df) = c('CCluster', 'CEP17')
    ccluster.uniq <- ddply(df, .(df$CCluster, df$CEP17), nrow)$V1
    df = as.data.frame(cbind(temp.mat$HER2, temp.mat$CEP17))
    colnames(df) = c('HER2', 'CEP17')
    her2.uniq <- ddply(df, .(df$HER2, df$CEP17), nrow)$V1
    her2.ccluster.bin <- c(length(which(temp.mat$her2.bin == TRUE & temp.mat$cluster.bin == TRUE)), 1)
    df = as.data.frame(cbind(temp.mat$HER2, temp.mat$CCluster))
    colnames(df) = c("HER2", "CCluster")
    her2.ccluster.uniq = ddply(df, .(df$HER2, df$CCluster), nrow)$V1
    her2.ccluster.ratio <- temp.mat$HER2 / temp.mat$CEP17
    diversity.mat <- rbind(diversity.mat, c(i, ind, mean(temp.mat$age), mean(temp.mat$scale.log.age),
    })
  }
  diversity.mat <- as.data.frame(diversity.mat)
  colnames(diversity.mat) <- c('Sample_ID', 'Index', 'age', 'scale.log.age', 'ER', 'PR', 'p53', 'Ki67', 'CCEP17')
}

diversity.mat.ave <- NULL
diversity.mat.max <- NULL
diversity.mat.min <- NULL
for (i in 1:20) {
  temp.div <- diversity.mat[diversity.mat$Sample_ID == i, 3:ncol(diversity.mat)]
  diversity.mat.ave <- rbind(diversity.mat.ave, c(i, apply(temp.div, 2, mean)))
  diversity.mat.max <- rbind(diversity.mat.max, c(i, apply(temp.div, 2, max)))
  diversity.mat.min <- rbind(diversity.mat.min, c(i, apply(temp.div, 2, min)))
}
diversity.mat.ave <- as.data.frame(diversity.mat.ave)
diversity.mat.max <- as.data.frame(diversity.mat.max)
diversity.mat.min <- as.data.frame(diversity.mat.min)

detach(all.mat)

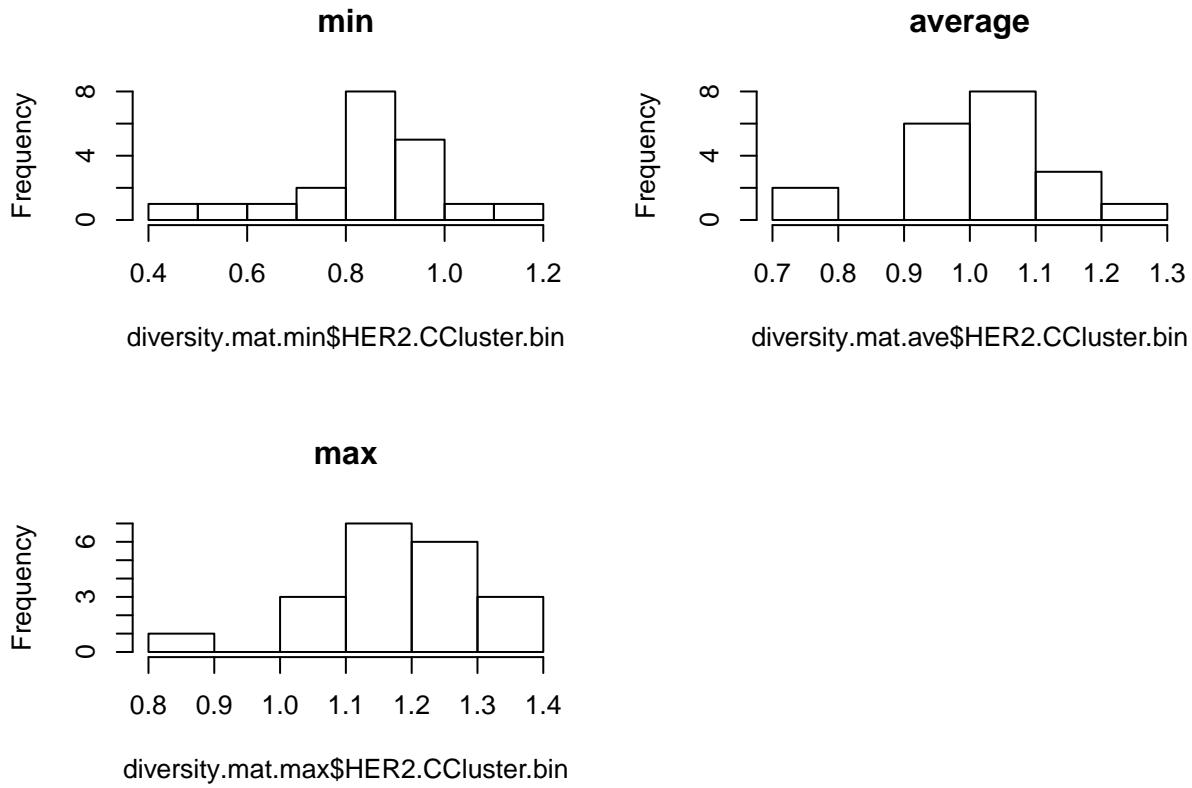
```

Look at the distribution of diversity:

```

par(mfrow=c(2,2))
hist(diversity.mat.min$HER2.CCluster.bin, main="min")
hist(diversity.mat.ave$HER2.CCluster.bin, main="average")
hist(diversity.mat.max$HER2.CCluster.bin, main="max")

```



```
attach(all.mat)
prop.mat.sample <- matrix(NA, nrow = 20, ncol = 2)
for (i in 1:20) {
  prop.mat.sample[i, ] <- c(mean(cluster.bin[her2.bin == TRUE & sample_ID == i]), mean(cluster.bin[her2.
```

- 1}

```
prop.mat.slides <- NULL
for (i in 1:20) {
  for (j in unique(index[sample_ID == i])) {
    prop.mat.slides <- rbind(prop.mat.slides, c(i, j, mean(cluster.bin[her2.bin == TRUE & sample_ID == i])))
  }
}
colnames(prop.mat.slides) <- c('Sample_ID', 'index', 'Her2.DCIS', 'NoHer2.DCIS')
prop.mat.slides[is.nan(prop.mat.slides) == TRUE] <- NA
prop.mat.slides <- as.data.frame(prop.mat.slides)

detach(all.mat)
```

Plot the distribution of HER2 cells DCIS

```
prop.mat.slides$Her2.Sample_ID <- 20 * prop.mat.slides$Sample_ID - 19
prop.mat.slides$NoHer2.Sample_ID <- 20 * prop.mat.slides$Sample_ID - 16
par(mfrow = c(1, 1), las = 1, cex.axis = 1.5, tcl = 0.1, xaxs = 'i', yaxs = 'i')

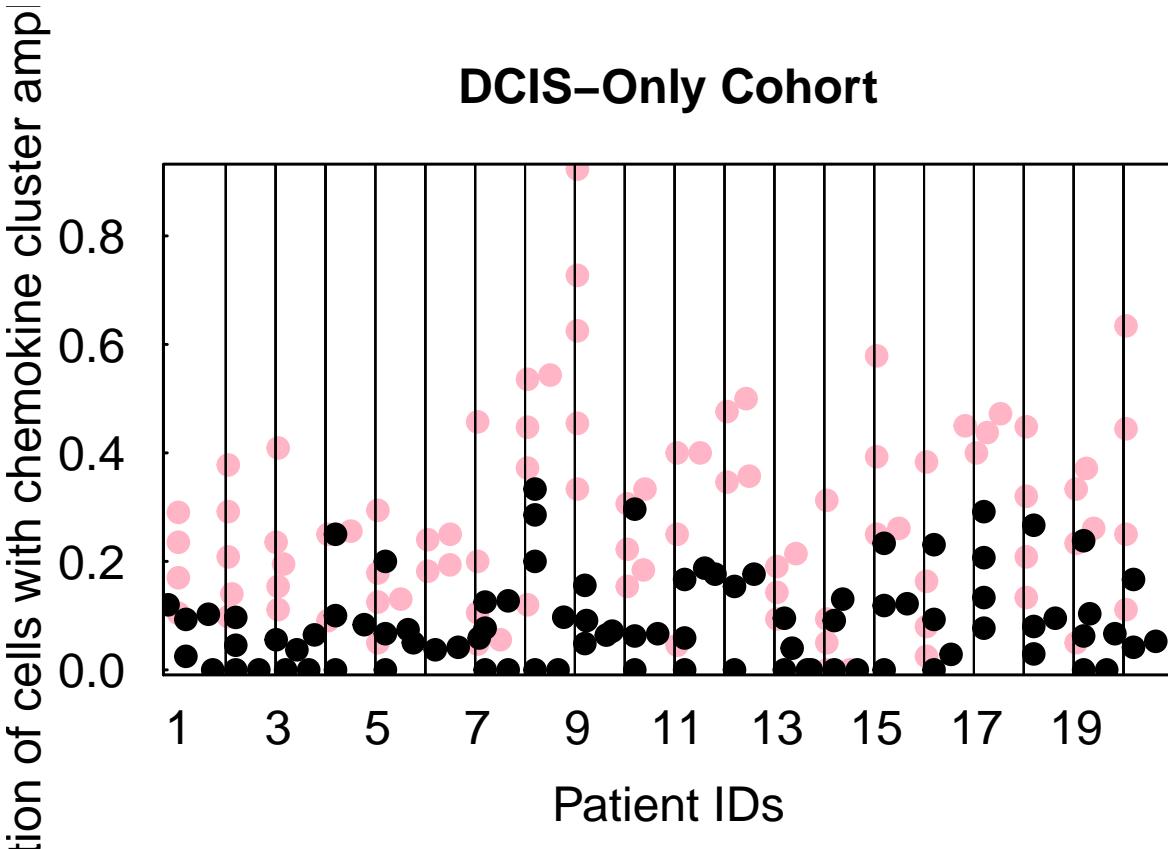
beeswarm(prop.mat.slides$Her2.DCIS ~ prop.mat.slides$Her2.Sample_ID,
          col = 'pink1', xlab = 'Patient IDs', ylab = 'Proportion of cells with chemokine cluster amplifi
```

```
beeswarm(prop.mat.slides$NoHer2.DCIS ~ prop.mat.slides$NoHer2.Sample_ID, col = 'black', lwd = 1.5, pch =
          cex = 1.5, at = unique(prop.mat.slides$NoHer2.Sample_ID))
for (i in (20 * prop.mat.slides$Sample_ID)) {
```

```

    lines(c(i, i), c(-0.5, 2))
}

```



```
#legend('topleft', c('Cells with HER2 amplification', 'Cells without HER2 amplification'), text.col = c('pink', 'black'))
```

### 9.1.2 Spatial statistics

Plot an example of the tissue: eg. patient1, region 1

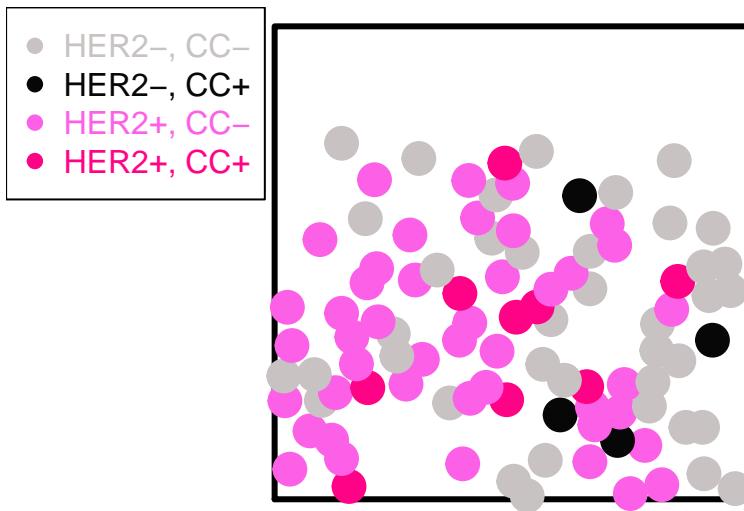
```

i<-1
ind<-1

temp.mat <- all.mat[all.mat$index == ind & all.mat$sample_ID == i, ]
temp.mat$m <- 'HER2-, CC-'
temp.mat$m[temp.mat$her2.bin == TRUE & temp.mat$cluster.bin == TRUE] <- 'HER2+, CC+'
temp.mat$m[temp.mat$her2.bin == FALSE & temp.mat$cluster.bin == TRUE] <- 'HER2-, CC+'
temp.mat$m[temp.mat$her2.bin == TRUE & temp.mat$cluster.bin == FALSE] <- 'HER2+, CC-'
temp.sp <- as.ppp(as.data.frame(cbind(temp.mat$x, temp.mat$y)), c(0, 1, 0, 1))
temp.sp$m <- as.character(temp.mat$m)
marks.uniq <- c('HER2-, CC-', 'HER2-, CC+', 'HER2+, CC-', 'HER2+, CC+')
marks(temp.sp) <- sapply(1:length(temp.mat$m), function(mm) which(marks.uniq == temp.mat$m[mm]))
colours.uniq <- c('#C8C2C2', '#070707', '#FC60E6', '#FF0387')
pch.uniq <- c(19, 19, 19, 19)
plot(temp.sp, border = 'black', cols = function(x) colours.uniq[x], lwd = 3, cex = 2, pch = function(x) marks.uniq[x])
legend('topleft', c('HER2-, CC-', 'HER2-, CC+', 'HER2+, CC-', 'HER2+, CC+'), text.col = colours.uniq)

```

## DCIS-Only Cohort: Patient 1 Region 1



Spatial modelling:

An example for patient 1, region 1:

```
set.seed(1206)

genotypes <- c('Her2-ChemokineCluster-', 'Her2+ChemokineCluster+', 'Her2-ChemokineCluster+', 'Her2+ChemokineCluster-')
phenotypes <- c('HER2.Neg.Chemokine.Neg', 'HER2.Pos.Chemokine.Pos', 'HER2.Neg.Chemokine.Pos', 'HER2.Pos.Chemokine.Neg')

pairs1 <- c(2, 2, 2, 2, 4, 4, 4, 4, 3, 3, 3, 3, 1, 1, 1, 1)
pairs2 <- c(2, 4, 3, 1, 4, 3, 1, NA, 3, 1, NA, NA, 1, NA, NA, NA)

eqfun <- function(m1, m2) {
  m1 == m2
}

i<-1
ind<-1
par(mfrow=c(2,2))

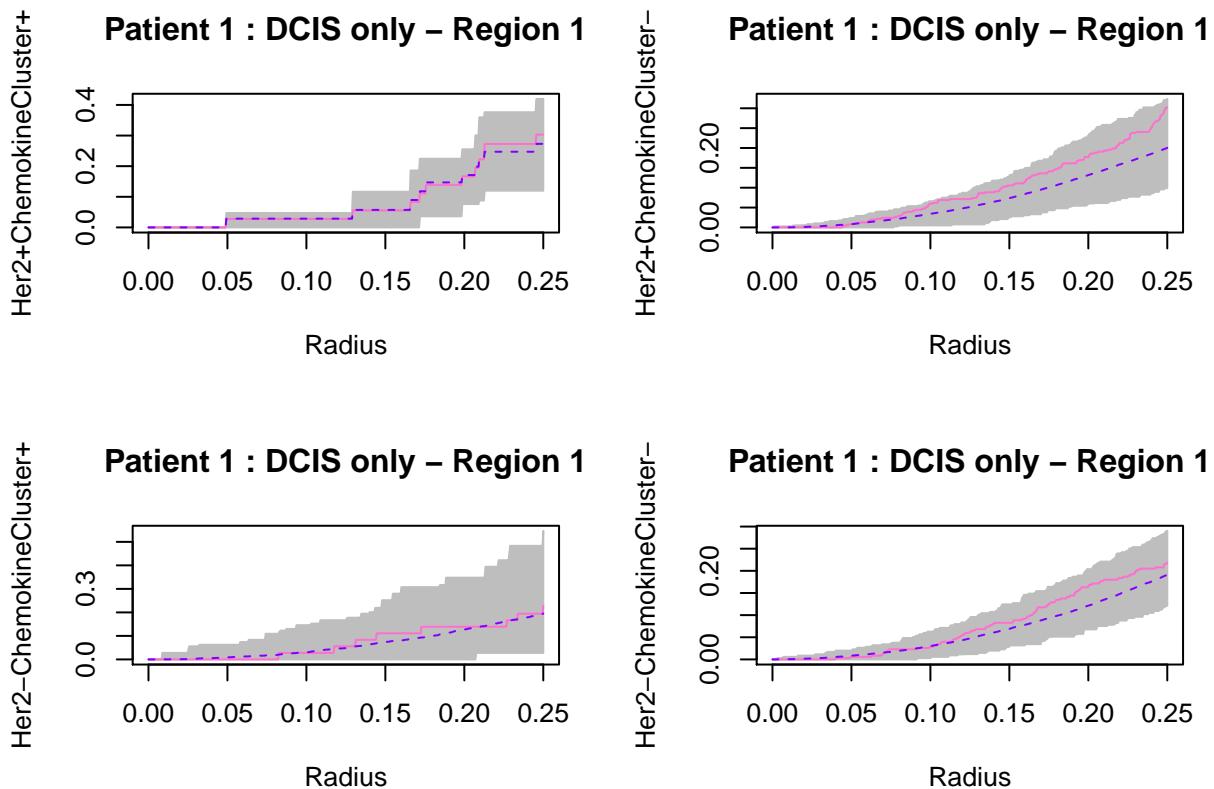
temp.mat <- all.mat[all.mat$index == ind & all.mat$sample_ID == i, ]
temp.mat$m <- 'HER2.Neg.Chemokine.Neg'
temp.mat$m[temp.mat$her2.bin == TRUE & temp.mat$cluster.bin == TRUE] <- 'HER2.Pos.Chemokine.Pos'
temp.mat$m[temp.mat$her2.bin == FALSE & temp.mat$cluster.bin == TRUE] <- 'HER2.Neg.Chemokine.Pos'
temp.mat$m[temp.mat$her2.bin == TRUE & temp.mat$cluster.bin == FALSE] <- 'HER2.Pos.Chemokine.Neg'
temp.sp <- as.ppp(as.data.frame(cbind(temp.mat$x, temp.mat$y)), c(0, 1, 0, 1))
marks(temp.sp) <- as.factor(temp.mat$m)
temp.sp.marks <- as.character(unique(marks(temp.sp)))
E <- list()
w <- 1
for (u in 1:length(pairs1)) {
  if (is.na(pairs2[u]) == TRUE) {
    E[[w]] <- NA
  } else if (!(phenotypes[pairs1[u]] %in% temp.sp.marks) | !(phenotypes[pairs2[u]] %in% temp.sp.marks)) {
    E[[w]] <- as.factor(phenotypes[pairs1[u]])
  } else {
    E[[w]] <- as.factor(phenotypes[pairs2[u]])
  }
  w <- w + 1
}
```

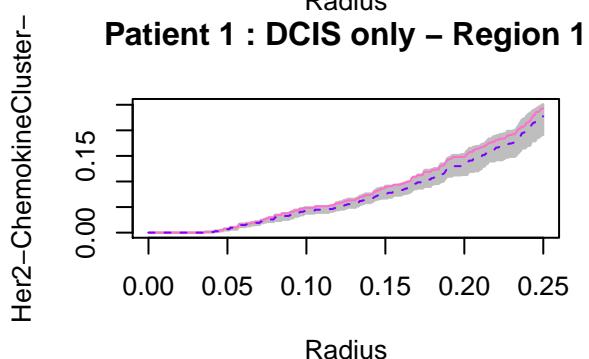
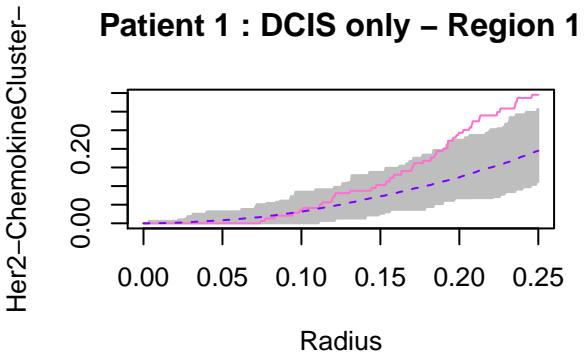
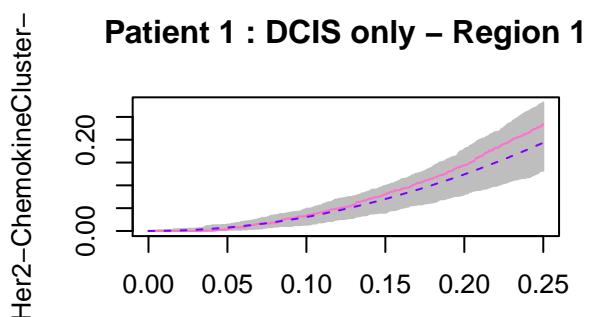
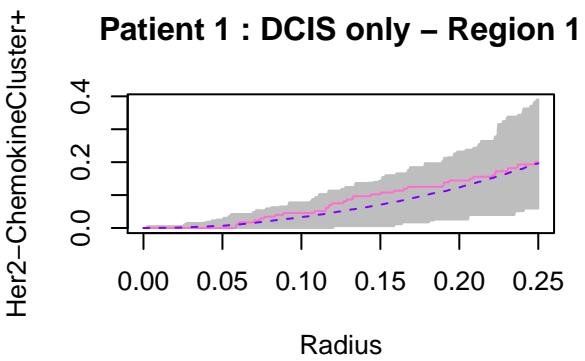
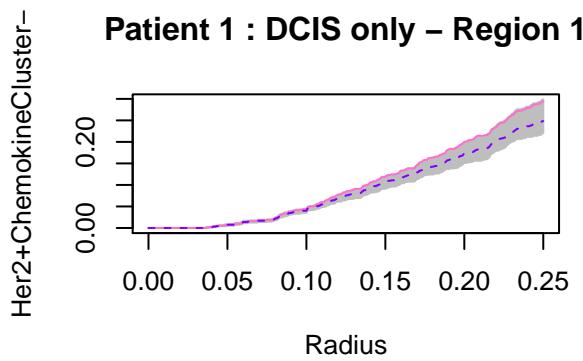
```

    E[[w]] <- NA
} else {
  E[[w]] <- envelope(temp.sp, Kcross, nsim = nrow(temp.mat), i = phenotypes[pairs1[u]], j = phenotypes[pairs1[v]])
}
w <- w + 1
}
w <- w - 1
# plotting function below:

for (ww in 1:w) {
  main = paste0('Patient ', i, ' : DCIS only - Region ', which(indices == ind))
  if (mode(E[[ww]]) == "logical") {
    plot(rnorm(100), rnorm(100), type = 'n', main = '', ylim = c(0, 1), axes = FALSE, xlab = '', ylab = '')
  } else {
    if (is.nan(E[[ww]]$obs[1]) == TRUE) {
      plot(rnorm(100), rnorm(100), type = 'n', main = '', ylim = c(0, 1), axes = FALSE, xlab = '', ylab = '')
    } else {
      plot(E[[ww]], main = main, xlab = 'Radius', ylab = paste0(genotypes[pairs1[ww]], ' vs.\n', genotypes[pairs1[ww]]))
    }
  }
}
}

```





Run for the rest of the cohort, and save to pdf:

```
set.seed(1206)
```

```
genotypes <- c('Her2-ChemokineCluster-', 'Her2+ChemokineCluster+', 'Her2-ChemokineCluster+', 'Her2+ChemokineCluster+')
phenotypes <- c('HER2.Neg.Chemokine.Neg', 'HER2.Pos.Chemokine.Pos', 'HER2.Neg.Chemokine.Pos', 'HER2.Pos.Chemokine.Neg')
```

```

pairs1 <- c(2, 2, 2, 2, 4, 4, 4, 4, 3, 3, 3, 3, 1, 1, 1, 1)
pairs2 <- c(2, 4, 3, 1, 4, 3, 1, NA, 3, 1, NA, NA, 1, NA, NA)

eqfun <- function(m1, m2) {
  m1 == m2
}

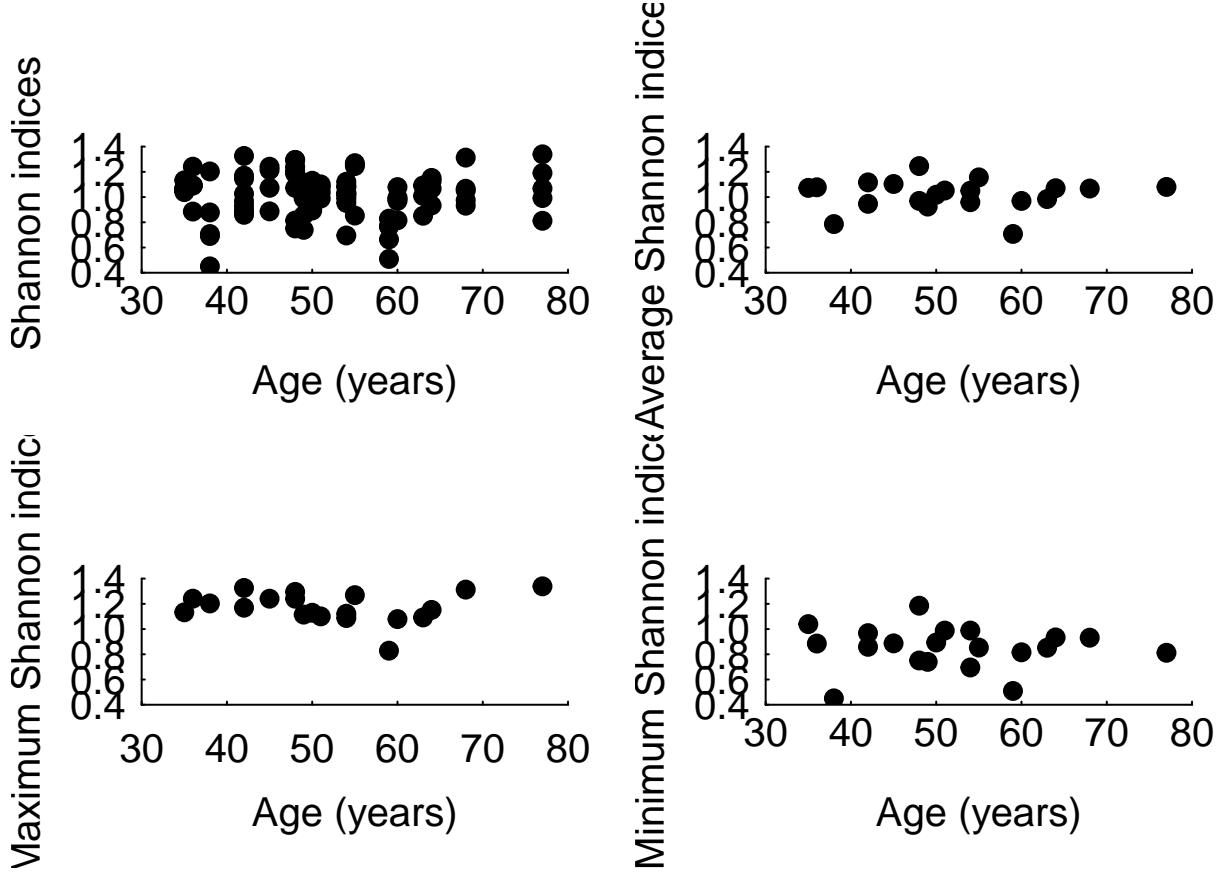
pdf('output/Spatial_Modeling_DCIS_Only.pdf', height = 16, width = 19.5)
for (i in 1:20) {
  indices <- unique(all.mat$index[all.mat$sample_ID == i])
  for (ind in indices) {
    par(mfrow = c(4, 4), mai = c(1, 1.2, 1, 1))
    temp.mat <- all.mat[all.mat$index == ind & all.mat$sample_ID == i, ]
    temp.mat$m <- 'HER2.Neg.Chemokine.Neg'
    temp.mat$m[temp.mat$her2.bin == TRUE & temp.mat$cluster.bin == TRUE] <- 'HER2.Pos.Chemokine.Pos'
    temp.mat$m[temp.mat$her2.bin == FALSE & temp.mat$cluster.bin == TRUE] <- 'HER2.Neg.Chemokine.Pos'
    temp.mat$m[temp.mat$her2.bin == TRUE & temp.mat$cluster.bin == FALSE] <- 'HER2.Pos.Chemokine.Neg'
    temp.sp <- as.ppp(as.data.frame(cbind(temp.mat$x, temp.mat$y)), c(0, 1, 0, 1))
    marks(temp.sp) <- as.factor(temp.mat$m)
    temp.sp.marks <- as.character(unique(marks(temp.sp)))
    E <- list()
    w <- 1
    for (u in 1:length(pairs1)) {
      if (is.na(pairs2[u]) == TRUE) {
        E[[w]] <- NA
      } else if (!(phenotypes[pairs1[u]] %in% temp.sp.marks) | !(phenotypes[pairs2[u]] %in% temp.sp.marks)) {
        E[[w]] <- NA
      } else {
        E[[w]] <- envelope(temp.sp, Kcross, nsim = nrow(temp.mat), i = phenotypes[pairs1[u]], j = phenotypes[pairs2[u]])
      }
      w <- w + 1
    }
    w <- w - 1
    # plotting function below:

    for (ww in 1:w) {
      main = paste0('Patient ', i, ' : DCIS only - Region ', which(indices == ind))
      if (mode(E[[ww]]) == "logical") {
        plot(rnorm(100), rnorm(100), type = 'n', main = '', ylim = c(0, 1), axes = FALSE, xlab = '', ylab = '')
      } else {
        if (is.nan(E[[ww]]$obs[1]) == TRUE) {
          plot(rnorm(100), rnorm(100), type = 'n', main = '', ylim = c(0, 1), axes = FALSE, xlab = '', ylab = '')
        } else {
          plot(E[[ww]], main = main, xlab = 'Radius', ylab = paste0(genotypes[pairs1[ww]], ' vs.\n', genotypes[pairs2[ww]]))
        }
      }
    }
  }
}
dev.off()

```

### 9.1.3 Associations between diversity and age: Figure 5

Visualise the data: Association with age



Compute the correlation coefficient and p-value:

```
cor.test(diversity.mat.ave$age, diversity.mat.ave$HER2.CCluster.bin, use="complete")

##
## Pearson's product-moment correlation
##
## data: diversity.mat.ave$age and diversity.mat.ave$HER2.CCluster.bin
## t = 0.071272, df = 18, p-value = 0.944
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
## -0.4289122 0.4559285
## sample estimates:
##      cor
## 0.01679657
```

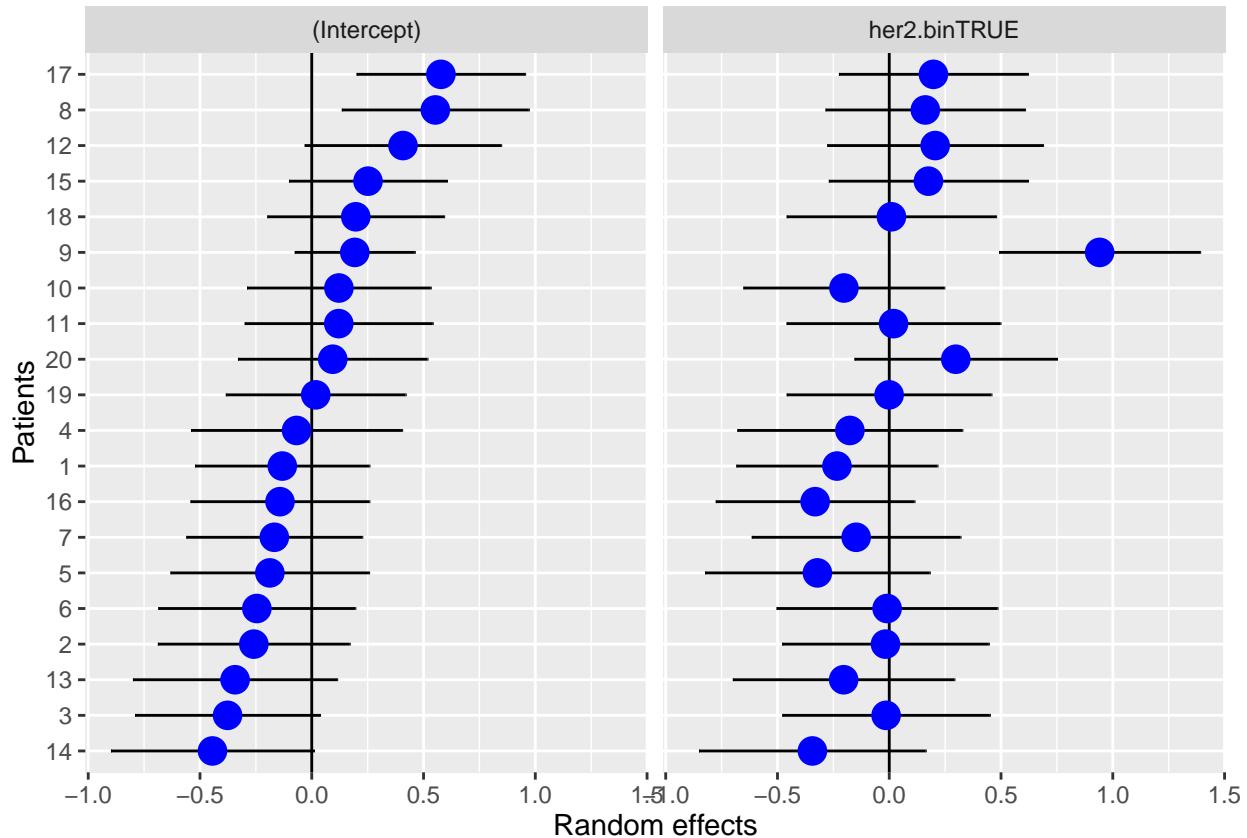
### 9.1.4 Mixed effects modelling

Mixed effects modelling: (note that the leukocyte data file has been updated to remove samples with poor quality images - S11 25135, 24603 and 24809 are removed)

```
# Mixed effect modeling table
dcis.only.mixed <- glmer(cluster.bin ~ 1 + her2.bin + (1 + her2.bin | sample_ID), data = all.mat, family = binomial)
```

```
ggCaterpillar(ranef(dcis.only.mixed, condVar = TRUE), QQ = FALSE)
```

```
## $sample_ID
```



```
summary(dcis.only.mixed)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
## Approximation) [glmerMod]
## Family: binomial ( logit )
## Formula: cluster.bin ~ 1 + her2.bin + (1 + her2.bin | sample_ID)
## Data: all.mat
##
##      AIC      BIC  logLik deviance df.resid
##  4412.2  4444.8 -2201.1   4402.2     4979
##
## Scaled residuals:
##    Min     1Q  Median     3Q    Max
## -1.0459 -0.4942 -0.3387 -0.2702  4.0595
##
## Random effects:
## Groups   Name        Variance Std.Dev. Corr
## sample_ID (Intercept) 0.1287  0.3587
##          her2.binTRUE 0.1398  0.3740  0.11
## Number of obs: 4984, groups: sample_ID, 20
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
```

```

## (Intercept) -2.3579      0.1141   -20.66   <2e-16 ***
## her2.binTRUE  1.3140      0.1255    10.47   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr)
## her2.bnTRUE -0.406

```

### 9.1.5 Correlations between shannon indices and CD3 or CD45 counts

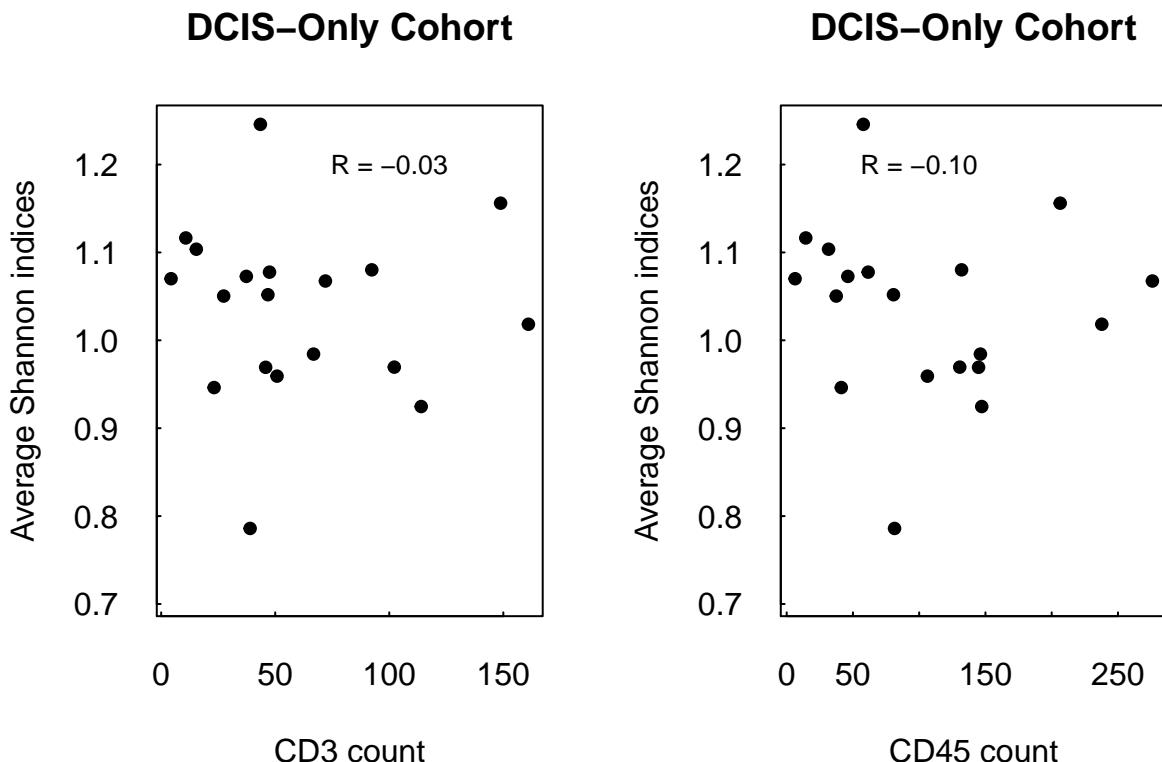
```

leukocytes <- read.csv('data/leukocytes_dcis_only.csv', stringsAsFactors = FALSE)
leukocytes <- leukocytes[-which(is.na(leukocytes$image.) == TRUE), ]
leukocytes$cd45 <- as.numeric(leukocytes$cd45)

leukocytes.ave.cd45 <- sapply(1:20, function(i) mean(leukocytes[leukocytes$Pathology.No == i, 3], na.rm = TRUE))
leukocytes.ave.cd3 <- sapply(1:20, function(i) mean(leukocytes[leukocytes$Pathology.No == i, 4], na.rm = TRUE))
leukocytes.ave.cd3p <- sapply(1:20, function(i) mean(leukocytes[leukocytes$Pathology.No == i, 4]) / leukocytes.ave.cd45[i])

par(mfrow = c(1, 2), tcl = 0.1, las = 1)
plot(leukocytes.ave.cd3, diversity.mat.ave$HER2.CCluster.bin, ylab = 'Average Shannon indices', main = text(100, 1.2, 'R = -0.03', cex = .8))
plot(leukocytes.ave.cd45, diversity.mat.ave$HER2.CCluster.bin, ylab = 'Average Shannon indices', main = text(100, 1.2, 'R = -0.10', cex = .8))

```



Compute the correlation coefficients and p values

```
cor.test(leukocytes.ave.cd3, diversity.mat.ave$HER2.CCluster.bin)
```

```

## 
## Pearson's product-moment correlation
##
## data: leukocytes.ave.cd3 and diversity.mat.ave$HER2.CCluster.bin
## t = -0.13629, df = 17, p-value = 0.8932
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
## -0.4800423 0.4275892
## sample estimates:
##          cor
## -0.03303627

cor.test(leukocytes.ave.cd45, diversity.mat.ave$HER2.CCluster.bin)

## 
## Pearson's product-moment correlation
##
## data: leukocytes.ave.cd45 and diversity.mat.ave$HER2.CCluster.bin
## t = -0.42962, df = 17, p-value = 0.6729
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
## -0.5327677 0.3678895
## sample estimates:
##          cor
## -0.1036375

```

## 9.2 IDC cases:

Load the data for the IDC cases:

```

all.mat<-read.csv("data/IDC_HER2_CCL_data.csv", stringsAsFactors = F)
all.mat$scale.log.age <- scale(log(all.mat$age))

```

### 9.2.1 Compute diversity

```

diversity.mat <- NULL
for (i in 1:18) {
  for (t in 0:1) {
    indices <- unique(all.mat$index[all.mat$sample_ID == i & all.mat>Type == t])
    for (ind in indices) {
      temp.mat <- all.mat[all.mat$index == ind & all.mat$sample_ID == i & all.mat>Type == t, ]
      cep17.abs <- table(temp.mat$CEP17)
      her2.abs <- table(temp.mat$HER2)
      ccluster.abs <- table(temp.mat$CCluster)
      her2.ratio <- table(temp.mat$HER2 / temp.mat$CEP17)
      ccluster.ratio <- table(temp.mat$CCluster / temp.mat$CEP17)
      df = as.data.frame(cbind(temp.mat$CCluster, temp.mat$CEP17))
      colnames(df) = c('CCluster', 'CEP17')
      ccluster.uniq <- ddply(df, .(df$CCluster, df$CEP17), nrow)$V1
      df = as.data.frame(cbind(temp.mat$HER2, temp.mat$CEP17))
      colnames(df) = c('HER2', 'CEP17')
      her2.uniq <- ddply(df, .(df$HER2, df$CEP17), nrow)$V1

```

```

her2.ccluster.bin <- c(length(which(temp.mat$her2.bin == TRUE & temp.mat$cluster.bin == TRUE))

df = as.data.frame(cbind(temp.mat$HER2, temp.mat$CCluster))
colnames(df) = c("HER2", "CCluster")
her2.ccluster.uniq = ddply(df, .(df$HER2, df$CCluster), nrow)$V1

her2.ccluster.ratio <- temp.mat$HER2 / temp.mat$CEP17
if (length(which(her2.ccluster.bin == 0)) > 0) {
    her2.ccluster.bin <- her2.ccluster.bin + 1
}
her2.ccluster.bin <- her2.ccluster.bin / sum(her2.ccluster.bin)
diversity.mat <- rbind(diversity.mat, c(i, t, ind, mean(temp.mat$age), mean(temp.mat$scale.log.age), mean(temp.mat$p53), mean(temp.mat$ER), mean(temp.mat$PR), mean(temp.mat$Ki67)))
}

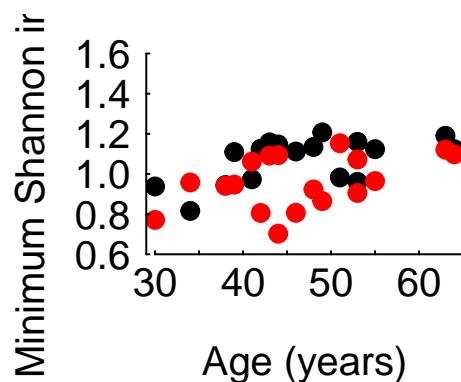
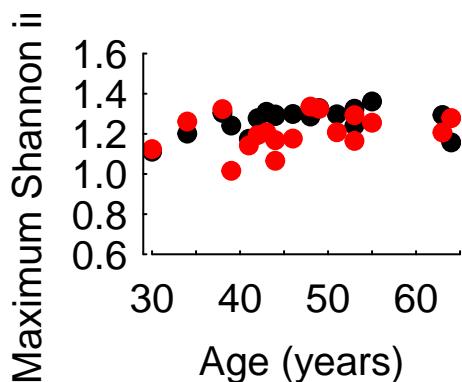
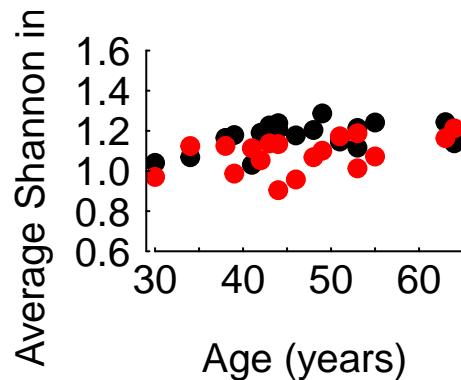
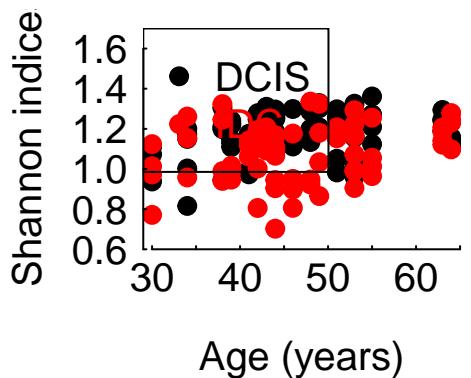
diversity.mat <- as.data.frame(diversity.mat)
colnames(diversity.mat) <- c('Sample_ID', 'Type', 'Index', 'age', 'scale.log.age', 'ER', 'PR', 'p53', 'Ki67')

#pathology$Ki.67 <- sapply(1:nrow(pathology), function(i) as.numeric(sub("%", "", pathology$Ki.67[i])))
diversity.mat.ave <- NULL
diversity.mat.max <- NULL
diversity.mat.min <- NULL
for (i in 1:18) {
    for (t in 0:1) {
        temp.div <- diversity.mat[diversity.mat$Sample_ID == i & diversity.mat>Type == t, 4:ncol(diversity.mat)]
        diversity.mat.ave <- rbind(diversity.mat.ave, c(i, t, apply(temp.div, 2, mean)))
        diversity.mat.max <- rbind(diversity.mat.max, c(i, t, apply(temp.div, 2, max)))
        diversity.mat.min <- rbind(diversity.mat.min, c(i, t, apply(temp.div, 2, min)))
    }
}

diversity.mat.ave <- as.data.frame(diversity.mat.ave)
diversity.mat.max <- as.data.frame(diversity.mat.max)
diversity.mat.min <- as.data.frame(diversity.mat.min)
colnames(diversity.mat.ave) <- c('Sample_ID', 'Type', 'age', 'scale.log.age', 'ER', 'PR', 'p53', 'Ki67')
colnames(diversity.mat.max) <- c('Sample_ID', 'Type', 'age', 'scale.log.age', 'ER', 'PR', 'p53', 'Ki67')
colnames(diversity.mat.min) <- c('Sample_ID', 'Type', 'age', 'scale.log.age', 'ER', 'PR', 'p53', 'Ki67')
diversity.mat.ave.dcis.idc <- diversity.mat.ave
diversity.mat.min.dcis.idc <- diversity.mat.min
diversity.mat.max.dcis.idc <- diversity.mat.max

```

### 9.3 Relationship between Diversity and age (Figure 5F):



Compute the correlation coefficient and p value:

```
# DCIS case:
cor.test(diversity.mat.ave$age[diversity.mat.ave>Type == 0], diversity.mat.ave$HER2.CCluster.bin[diversity.mat.ave>Type == 0], use="complete")

##
## Pearson's product-moment correlation
##
## data: diversity.mat.ave$age[diversity.mat.ave>Type == 0] and diversity.mat.ave$HER2.CCluster.bin[diversity.mat.ave>Type == 0]
## t = 2.0333, df = 16, p-value = 0.05896
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
## -0.01742104 0.75935855
## sample estimates:
##       cor
## 0.4531345

# IDC case:
cor.test(diversity.mat.ave$age[diversity.mat.ave>Type == 1], diversity.mat.ave$HER2.CCluster.bin[diversity.mat.ave>Type == 1], use="complete")

##
## Pearson's product-moment correlation
##
## data: diversity.mat.ave$age[diversity.mat.ave>Type == 1] and diversity.mat.ave$HER2.CCluster.bin[diversity.mat.ave>Type == 1]
## t = 2.0168, df = 16, p-value = 0.06082
```

```

## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
## -0.02110128  0.75779549
## sample estimates:
##       cor
## 0.450204

```

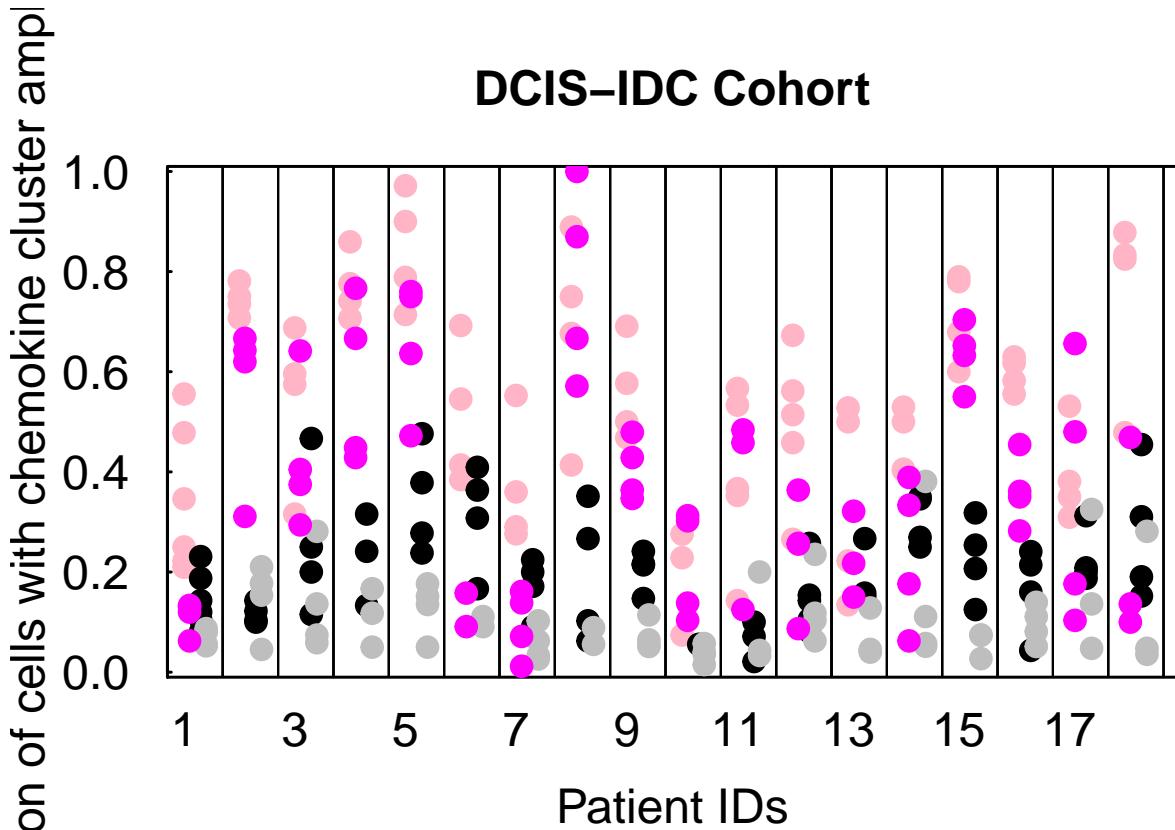
### 9.3.1 Summary of amplification diversity the data sets

```

attach(all.mat)
prop.mat.slides <- NULL
for (i in 1:18) {
    for (j in unique(index[sample_ID == i])) {
        prop.mat.slides <- rbind(prop.mat.slides, c(i, j, mean(cluster.bin[her2.bin == TRUE & Type == 0
    })
}
colnames(prop.mat.slides) <- c('Sample_ID', 'index', 'Her2.DCIS', 'NoHer2.DCIS', 'Her2.IDC', 'NoHer2.IDC')
prop.mat.slides[is.nan(prop.mat.slides) == TRUE] <- NA
prop.mat.slides <- as.data.frame(prop.mat.slides)

prop.mat.slides$Her2.DCIS.Sample_ID <- 20 * prop.mat.slides$Sample_ID - 19
prop.mat.slides$Her2.IDC.Sample_ID <- 20 * prop.mat.slides$Sample_ID - 17
prop.mat.slides$NoHer2.DCIS.Sample_ID <- 20 * prop.mat.slides$Sample_ID - 13
prop.mat.slides$NoHer2.IDC.Sample_ID <- 20 * prop.mat.slides$Sample_ID - 11
par(mfrow = c(1, 1), las = 1, cex.axis = 1.5, tcl = 0.1, xaxs = 'i', yaxs = 'i')
beeswarm(prop.mat.slides$Her2.DCIS ~ prop.mat.slides$Her2.DCIS.Sample_ID, col = 'pink1', xlab = 'Patient ID')
beeswarm(prop.mat.slides$NoHer2.DCIS ~ prop.mat.slides$NoHer2.DCIS.Sample_ID, col = 'black', lwd = 1.5,
beeswarm(prop.mat.slides$NoHer2.IDC ~ prop.mat.slides$NoHer2.IDC.Sample_ID, col = 'grey', lwd = 1.5, pch = 15)
beeswarm(prop.mat.slides$Her2.IDC ~ prop.mat.slides$Her2.IDC.Sample_ID, col = 'magenta', lwd = 1.5, pch = 15)
for (i in (20 * prop.mat.slides$Sample_ID)) {
    lines(c(i-5, i-5), c(-0.5, 2))
}

```



```
detach(all.mat)
```

### 9.3.2 Spatial statistics:

As an example for DCIS-IDC patient 1, region 1, we can firstly plot an example of the tissue, highlighting differences in HER2/CCL status

```
i=1
ind<-1

par(mfrow=c(1,2))

for (t in 0:1) {
  temp.mat <- all.mat[all.mat$index == ind & all.mat$sample_ID == i & all.mat>Type==t, ]
  temp.mat$m <- 'HER2-, CC-'
  temp.mat$m[temp.mat$her2.bin == TRUE & temp.mat$cluster.bin == TRUE] <- 'HER2+, CC+'
  temp.mat$m[temp.mat$her2.bin == FALSE & temp.mat$cluster.bin == TRUE] <- 'HER2-, CC+'
  temp.mat$m[temp.mat$her2.bin == TRUE & temp.mat$cluster.bin == FALSE] <- 'HER2+, CC-'
  temp.sp <- as.ppp(as.data.frame(cbind(temp.mat$x, temp.mat$y)), c(0, 1, 0, 1))
  temp.sp$m <- as.character(temp.mat$m)

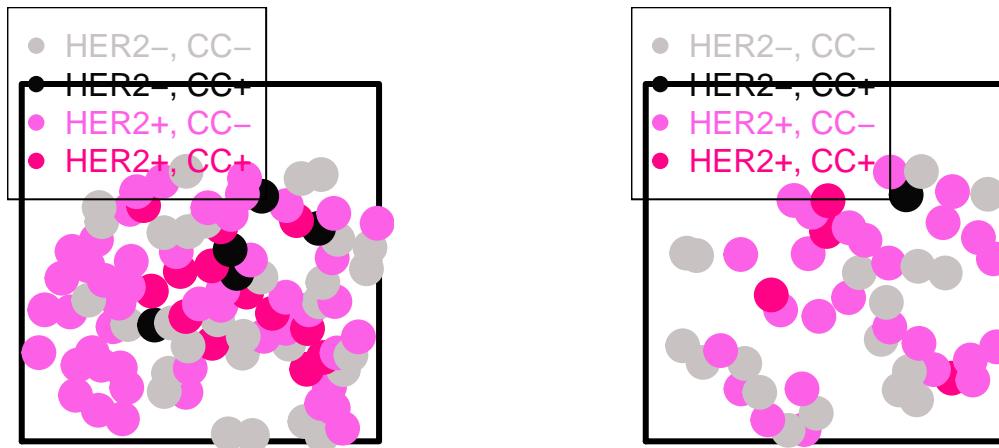
  marks.uniq <- c('HER2-, CC-', 'HER2-, CC+', 'HER2+, CC-', 'HER2+, CC+')
  marks(temp.sp) <- sapply(1:length(temp.mat$m), function(mm) which(marks.uniq == temp.mat$m[mm]))
  if (t == 0) {
    colours.uniq <- c('#C8C2C2', '#070707', '#FC60E6', '#FF0387')
    pch.uniq <- c(19, 19, 19, 19)
```

```

    plot(temp.sp, border = 'black', cols = function(x) colours.uniq[x], lwd = 3, cex = 2, pch = function(x) ifelse(x == 1, 19, 1))
    legend('topleft', c('HER2-, CC-', 'HER2-, CC+', 'HER2+, CC-', 'HER2+, CC+'), text.col = colours)
} else {
  colours.uniq <- c('#C8C2C2', '#070707', '#FC60E6', '#FF0387')
  pch.uniq <- c(19, 19, 19, 19)
  plot(temp.sp, border = 'black', cols = function(x) colours.uniq[x], lwd = 3, cex = 2, pch = function(x) ifelse(x == 1, 19, 1))
  legend('topleft', c('HER2-, CC-', 'HER2-, CC+', 'HER2+, CC-', 'HER2+, CC+'), text.col = colours)
}
}

```

## CIS-IDC Cohort: Patient 1 DCIS Reg CIS-IDC Cohort: Patient 1 IDC Reg



There are four different genotypes, and 4 different phenotypes: 16 combinations to try in terms of spatial clustering

```

genotypes <- c('Her2-ChemokineCluster-', 'Her2+ChemokineCluster+', 'Her2-ChemokineCluster+', 'Her2+ChemokineCluster-')
phenotypes <- c('HER2.Neg.Chemokine.Neg', 'HER2.Pos.Chemokine.Pos', 'HER2.Neg.Chemokine.Pos', 'HER2.Pos.Chemokine.Neg')

pairs1 <- c(2, 2, 2, 2, 4, 4, 4, 4, 3, 3, 3, 3, 1, 1, 1, 1)
pairs2 <- c(2, 4, 3, 1, 4, 3, 1, NA, 3, 1, NA, NA, 1, NA, NA, NA)

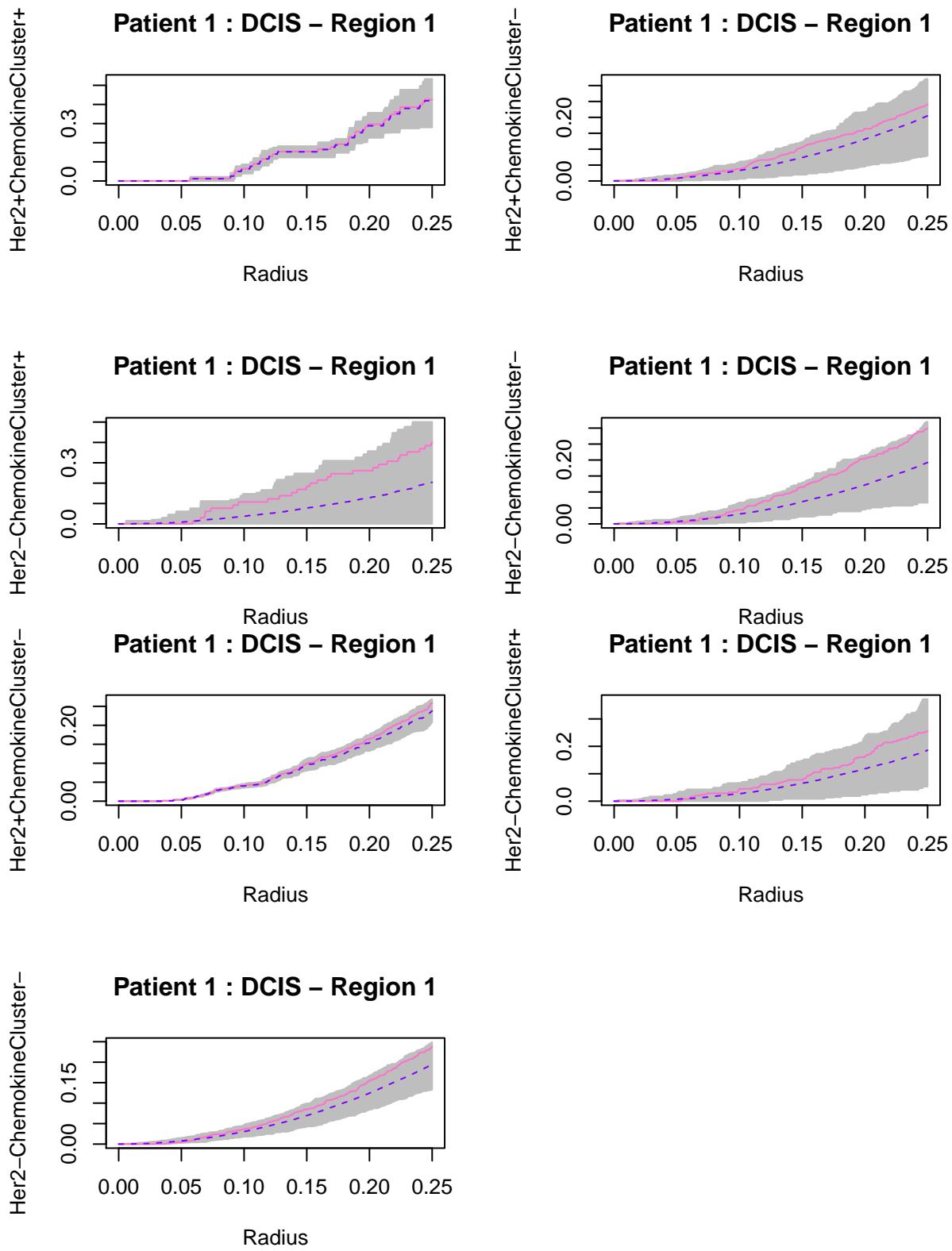
eqfun <- function(m1, m2) {
  m1 == m2
}

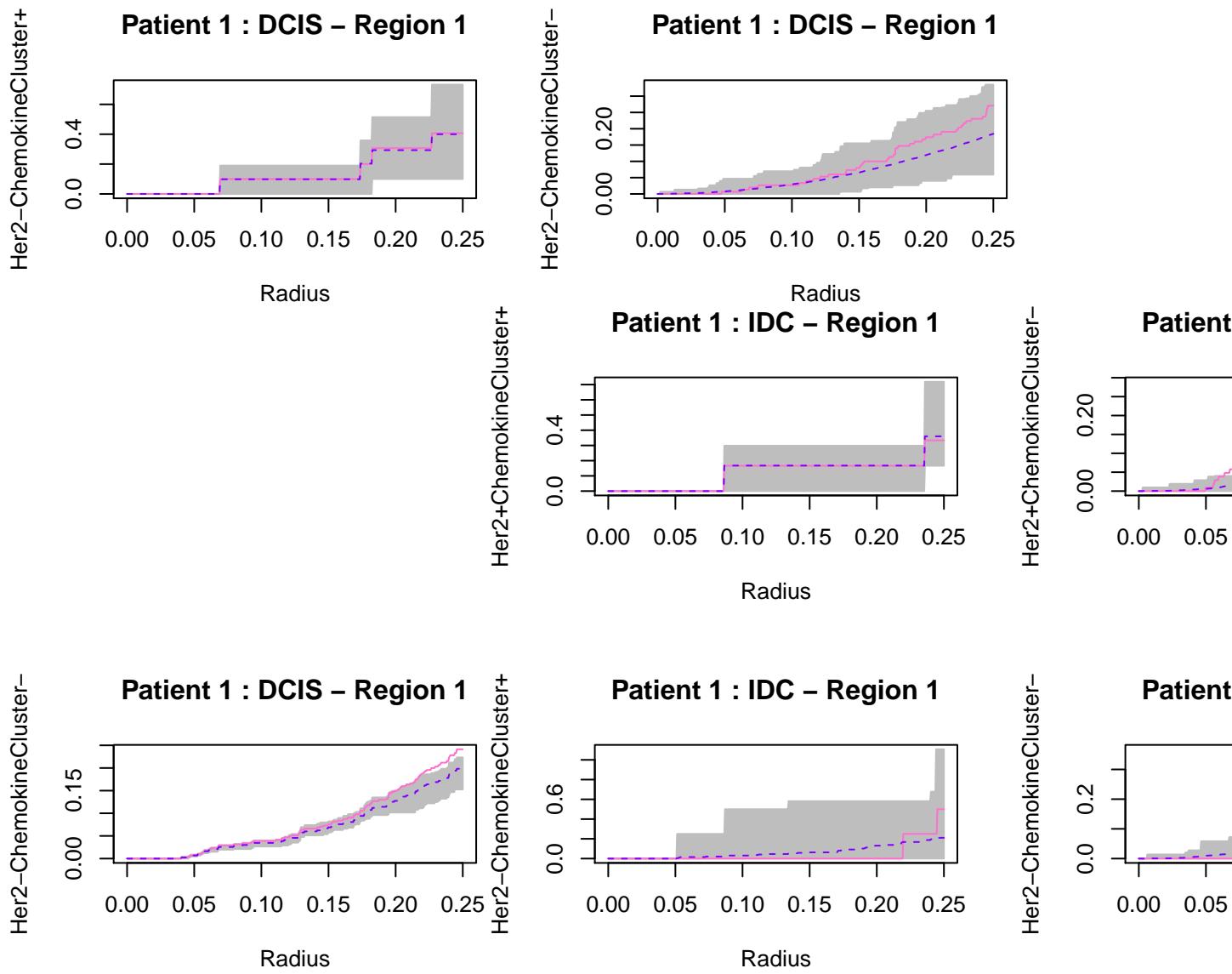
set.seed(1206)
i <- 1
ind <- 1
par(mfrow = c(2, 2))

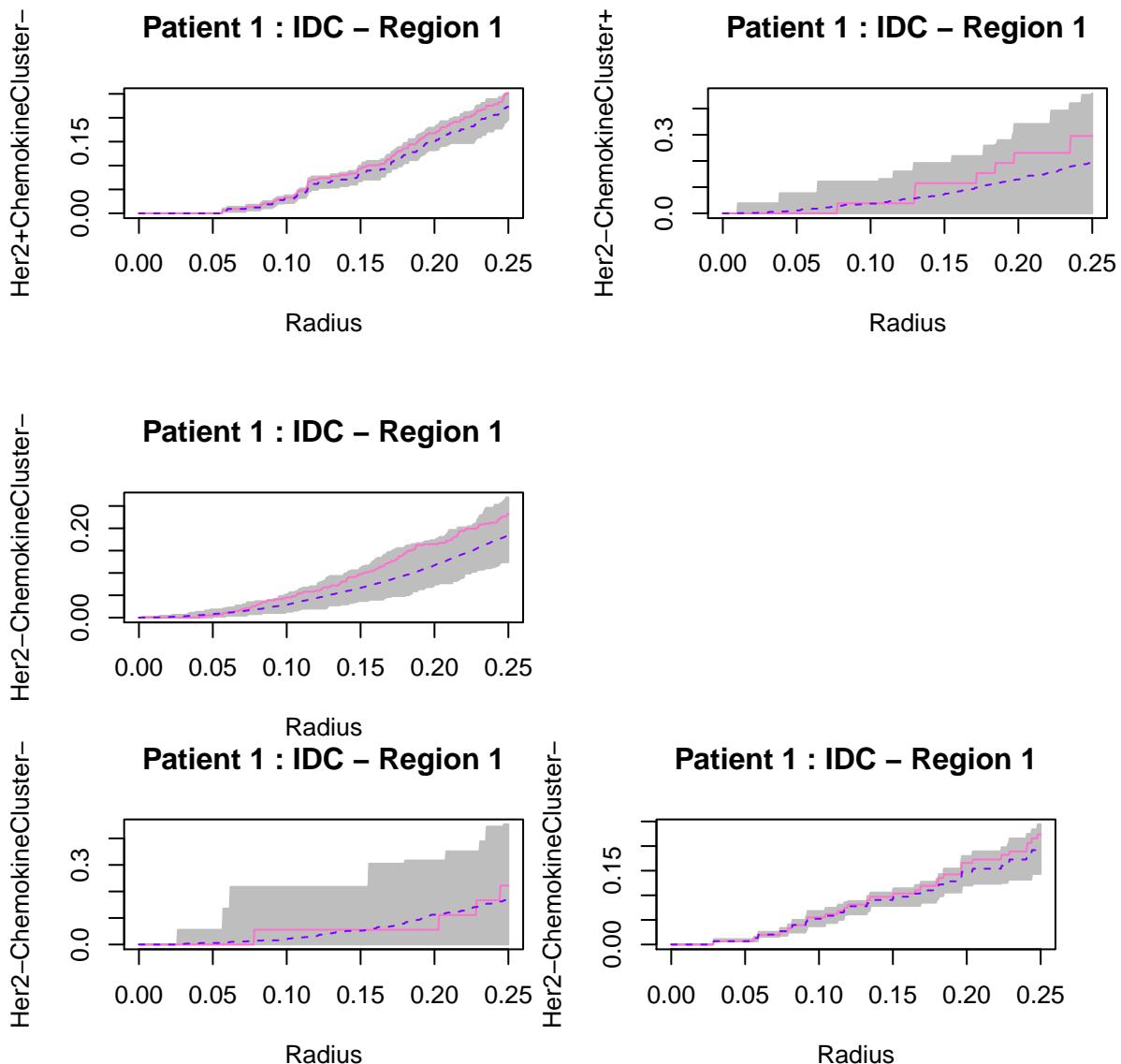
for (t in 0:1) {
  temp.mat <- all.mat[all.mat$index == ind & all.mat$sample_ID == i & all.mat>Type == t, ]
  temp.mat$m <- 'HER2.Neg.Chemokine.Neg'
  temp.mat$m[temp.mat$her2.bin == TRUE & temp.mat$cluster.bin == TRUE] <- 'HER2.Pos.Chemokine.Pos'
  temp.mat$m[temp.mat$her2.bin == FALSE & temp.mat$cluster.bin == TRUE] <- 'HER2.Neg.Chemokine.Pos'
  temp.mat$m[temp.mat$her2.bin == TRUE & temp.mat$cluster.bin == FALSE] <- 'HER2.Pos.Chemokine.Neg'
  temp.sp <- as.ppp(as.data.frame(cbind(temp.mat$x, temp.mat$y)), c(0, 1, 0, 1))
  marks(temp.sp) <- as.factor(temp.mat$m)
  temp.sp.marks <- as.character(unique(marks(temp.sp)))
}

```









```

# Run this when compile for the final time
# as examples, plot Patient 1

pdf('output/Spatial_Modeling_DCIS(IDC.pdf', height = 16, width = 19.5)
for (i in 1:18) {
  for (t in 0:1) {
    indices <- unique(all.mat$index[all.mat$sample_ID == i & all.mat>Type == t])
    for (ind in indices) {
      par(mfrow = c(4, 4), mai = c(1, 1.2, 1, 1))
      temp.mat <- all.mat[all.mat$index == ind & all.mat$sample_ID == i & all.mat$type == t, ]
      temp.mat$m <- 'HER2.Neg.Chemokine.Neg'
      temp.mat$m[temp.mat$her2.bin == TRUE & temp.mat$cluster.bin == TRUE] <- 'HER2.Pos.Chemokine.Pos'
      temp.mat$m[temp.mat$her2.bin == FALSE & temp.mat$cluster.bin == TRUE] <- 'HER2.Neg.Chemokine.Pos'
      temp.mat$m[temp.mat$her2.bin == TRUE & temp.mat$cluster.bin == FALSE] <- 'HER2.Pos.Chemokine.Neg'
      temp.sp <- as.ppp(as.data.frame(cbind(temp.mat$x, temp.mat$y)), c(0, 1, 0, 1))
      marks(temp.sp) <- as.factor(temp.mat$m)
      temp.sp.marks <- as.character(unique(marks(temp.sp)))
      E <- list()
    }
  }
}

```

```

w <- 1
for (u in 1:length(pairs1)) {
  if (is.na(pairs2[u]) == TRUE) {
    E[[w]] <- NA
  } else if (!(phenotypes[pairs1[u]] %in% temp.sp.marks) | !(phenotypes[pairs2[u]] %in% temp.sp.marks)) {
    E[[w]] <- NA
  } else {
    E[[w]] <- envelope(temp.sp, Kcross, nsim = nrow(temp.mat), i = phenotypes[pairs1[u]], j = phenotypes[pairs2[u]])
  }
  w <- w + 1
}
w <- w - 1
if (i==1){
# ww <- 1
if (t == 0) {
  for (ww in 1:w) {
    main = paste0('Patient ', i, ' : DCIS - Region ', which(indices == ind))
    if (mode(E[[ww]]) == "logical") {
      plot(rnorm(100), rnorm(100), type = 'n', main = '', ylim = c(0, 1), axes = FALSE, xlab = '')
    } else {
      if (is.nan(E[[ww]]$obs[1]) == TRUE) {
        plot(rnorm(100), rnorm(100), type = 'n', main = '', ylim = c(0, 1), axes = FALSE, xlab = '')
      } else {
        plot(E[[ww]], main = main, xlab = 'Radius', ylab = paste0(genotypes[pairs1[ww]], ' vs.\n'))
      }
    }
  }
} else {
  for (ww in 1:w) {
    main = paste0('Patient ', i, ' : IDC - Region ', which(indices == ind))
    if (mode(E[[ww]]) == "logical") {
      plot(rnorm(100), rnorm(100), type = 'n', main = '', ylim = c(0, 1), axes = FALSE, xlab = '')
    } else {
      if (is.nan(E[[ww]]$obs[1]) == TRUE) {
        plot(rnorm(100), rnorm(100), type = 'n', main = '', ylim = c(0, 1), axes = FALSE, xlab = '')
      } else {
        plot(E[[ww]], main = main, xlab = 'Radius', ylab = paste0(genotypes[pairs1[ww]], ' vs.\n'))
      }
    }
  }
}
}
dev.off()

```

Average Shannon indices and CD3 or CD45 counts:

```

leukocytes <- read.csv('data/leukocytes_dcis_idc.csv', stringsAsFactors = FALSE)
leukocytes <- leukocytes[-which(is.na(leukocytes$image.) == TRUE), ]
leukocytes <- leukocytes[-which(leukocytes$cd45 == 'IDC'), ]

leukocytes$cd45 <- as.numeric(leukocytes$cd45)
leukocytes.ave.cd45 <- sapply(1:18, function(i) mean(leukocytes[leukocytes$Pa == i, 3], na.rm = TRUE))
leukocytes.ave.cd3 <- sapply(1:18, function(i) mean(leukocytes[leukocytes$Pa == i, 4], na.rm = TRUE))

```

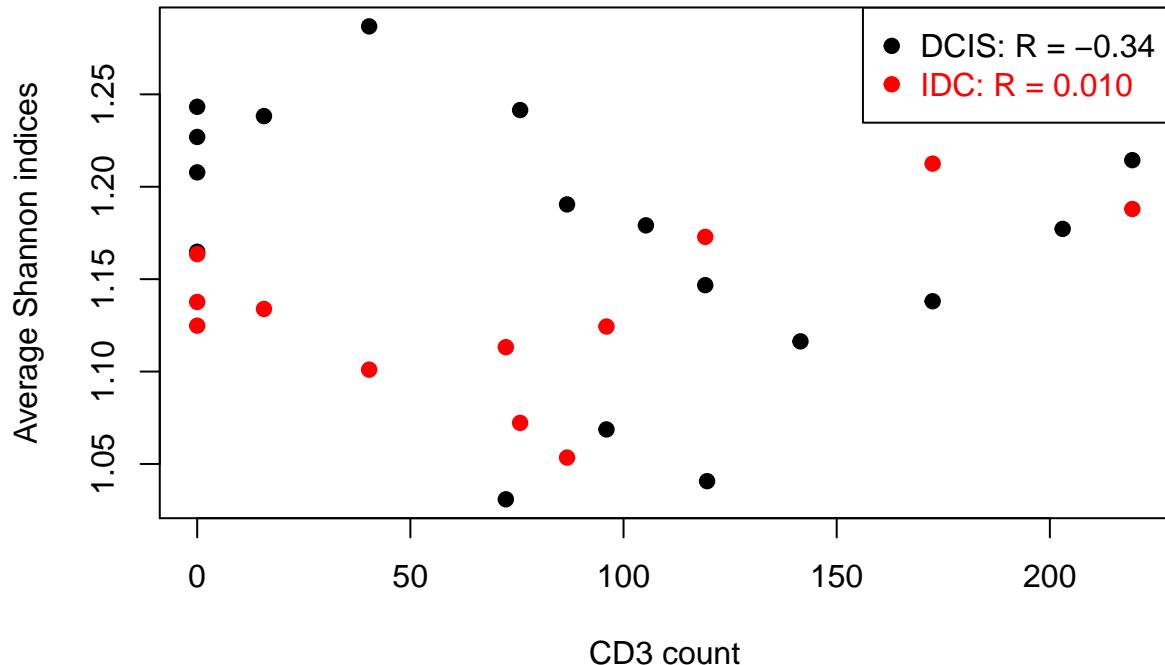
```

leukocytes.ave.cd3p <- sapply(1:18, function(i) mean(leukocytes[leukocytes$Pa == i, 4] / leukocytes[leukocytes$Pa == i, 1], na.rm = TRUE))

plot(leukocytes.ave.cd3, diversity.mat.ave$HER2.CCluster.bin[diversity.mat.ave$type == 0], ylab = 'Average Shannon indices', xlab = 'CD3 count')
points(leukocytes.ave.cd3, diversity.mat.ave$HER2.CCluster.bin[diversity.mat.ave$type == 1], col = 'red')
legend('topright', c(expression(paste('DCIS: ', R, ' = -0.34')), expression(paste('IDC: ', R, ' = 0.010'))))

```

### DCIS-IDC Cohort

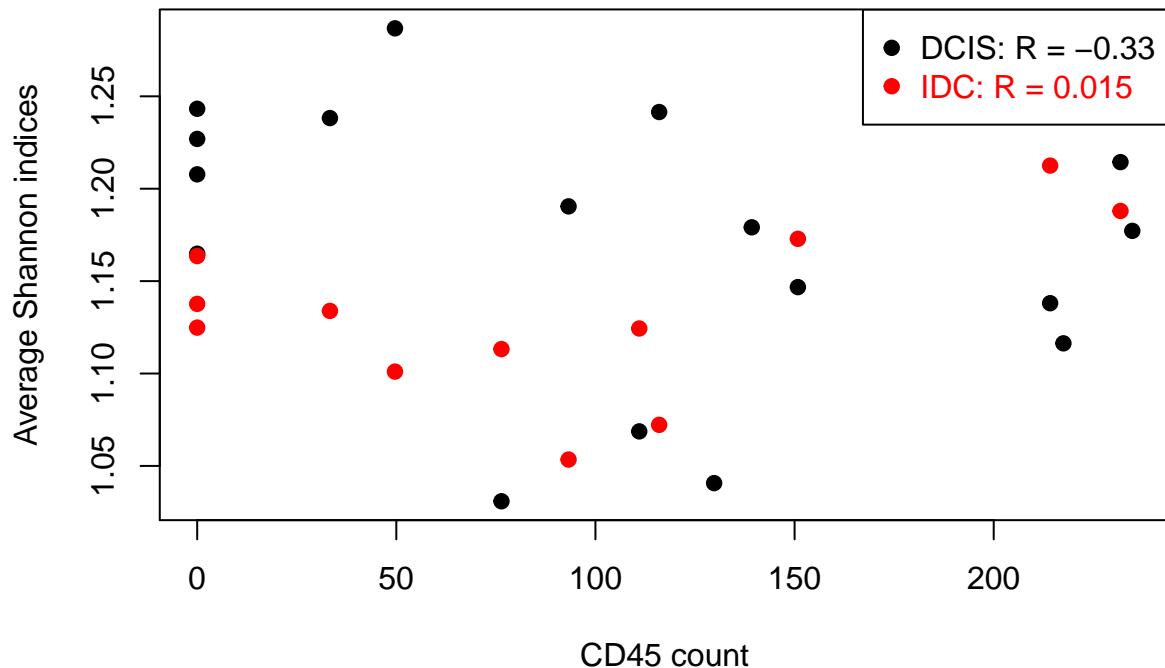


```

plot(leukocytes.ave.cd45, diversity.mat.ave$HER2.CCluster.bin[diversity.mat.ave$type == 0], ylab = 'Average Shannon indices', xlab = 'CD45 count')
points(leukocytes.ave.cd45, diversity.mat.ave$HER2.CCluster.bin[diversity.mat.ave$type == 1], col = 'red')
legend('topright', c(expression(paste('DCIS: ', R, ' = -0.33')), expression(paste('IDC: ', R, ' = 0.010'))))

```

## DCIS-IDC Cohort



```
#CD3
```

```
## DCIS sections:
```

```
cor.test(leukocytes.ave.cd3, diversity.mat.ave$HER2.CCluster.bin[diversity.mat.ave$type == 0])
```

```
##
```

```
## Pearson's product-moment correlation
```

```
##
```

```
## data: leukocytes.ave.cd3 and diversity.mat.ave$HER2.CCluster.bin[diversity.mat.ave$type == leukocyte]
```

```
## t = -1.3611, df = 15, p-value = 0.1936
```

```
## alternative hypothesis: true correlation is not equal to 0
```

```
## 95 percent confidence interval:
```

```
## -0.7005631 0.1773442
```

```
## sample estimates:
```

```
## cor
```

```
## -0.3315628
```

```
## IDC:
```

```
cor.test(leukocytes.ave.cd3, diversity.mat.ave$HER2.CCluster.bin[diversity.mat.ave$type == 1])
```

```
##
```

```
## Pearson's product-moment correlation
```

```
##
```

```
## data: leukocytes.ave.cd3 and diversity.mat.ave$HER2.CCluster.bin[diversity.mat.ave$type == leukocyte]
```

```
## t = -0.0060727, df = 15, p-value = 0.9952
```

```
## alternative hypothesis: true correlation is not equal to 0
```

```
## 95 percent confidence interval:
```

```
## -0.4818496 0.4794381
```

```
## sample estimates:
```

```
## cor
```

```
## -0.001567962
```

```

# CD45
## DCIS sections:
cor.test(leukocytes.ave.cd45, diversity.mat.ave$HER2.CCluster.bin[diversity.mat.ave>Type == 0])

##
## Pearson's product-moment correlation
##
## data: leukocytes.ave.cd45 and diversity.mat.ave$HER2.CCluster.bin[diversity.mat.ave>Type == leukocy
## t = -1.2869, df = 15, p-value = 0.2176
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
## -0.6912162 0.1948405
## sample estimates:
## cor
## -0.3153351
## IDC:
cor.test(leukocytes.ave.cd45, diversity.mat.ave$HER2.CCluster.bin[diversity.mat.ave>Type == 1])

##
## Pearson's product-moment correlation
##
## data: leukocytes.ave.cd45 and diversity.mat.ave$HER2.CCluster.bin[diversity.mat.ave>Type == leukocy
## t = -0.11182, df = 15, p-value = 0.9124
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
## -0.5025342 0.4581395
## sample estimates:
## cor
## -0.02886036

```

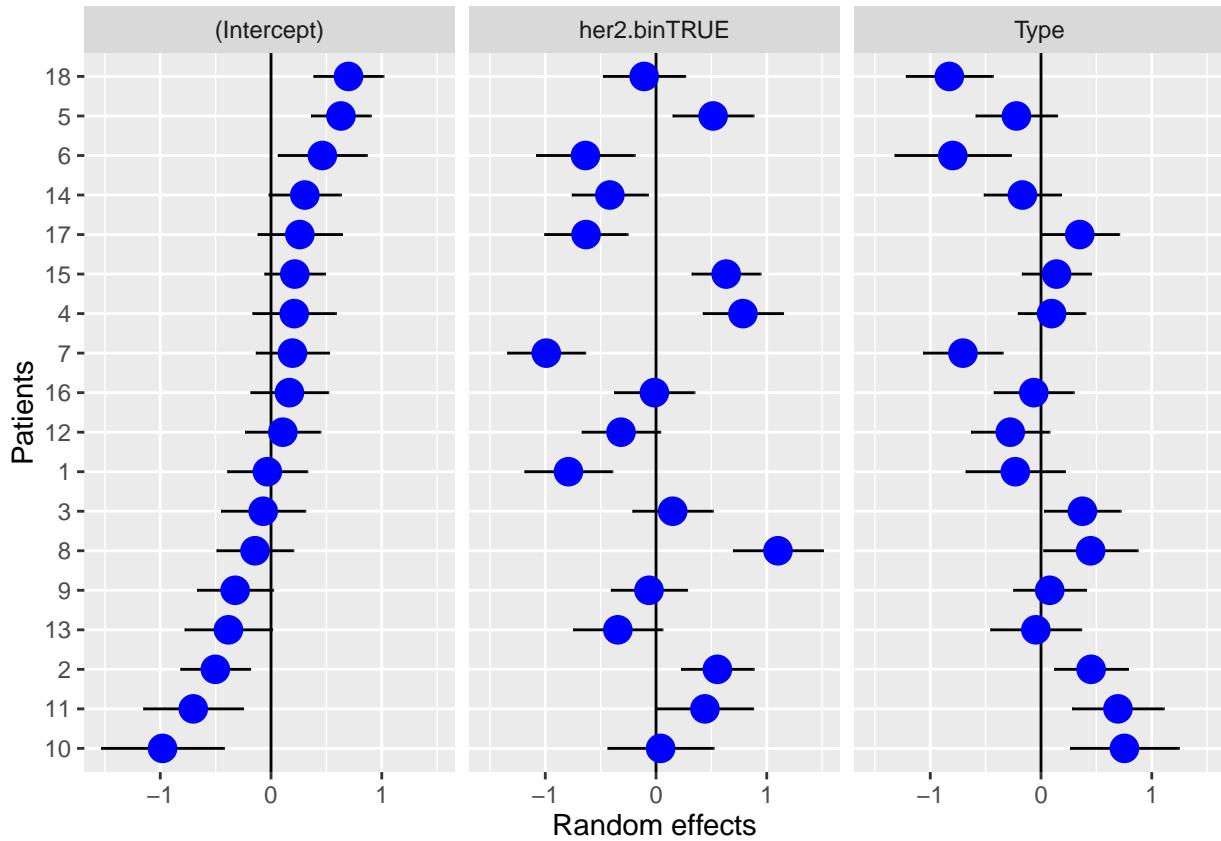
### 9.3.3 Generalised linear model

```

model.final <- glmer(cluster.bin ~ 1 + her2.bin + Type + scale(log(age)) + ER + ER * her2.bin + (1 + he
ggCaterpillar(ranef(model.final, condVar = TRUE), QQ = FALSE)

## $sample_ID

```



```

age.patients = sapply(1:18, function(i) mean(all.mat$scale.log.age[all.mat$sample_ID == i]))
summary(model.final)

## Generalized linear mixed model fit by maximum likelihood (Laplace
## Approximation) [glmerMod]
## Family: binomial ( logit )
## Formula: cluster.bin ~ 1 + her2.bin + Type + scale(log(age)) + ER + ER *
##          her2.bin + (1 + her2.bin + Type | sample_ID)
## Data: all.mat
##
##      AIC      BIC  logLik deviance df.resid
## 10910.8 10997.7 -5443.4 10886.8     10353
##
## Scaled residuals:
##      Min      1Q  Median      3Q      Max
## -2.2110 -0.6571 -0.3567  0.7795  4.9703
##
## Random effects:
##   Groups      Name        Variance Std.Dev. Corr
##   sample_ID (Intercept) 0.2269    0.4764
##             her2.binTRUE 0.3654    0.6045   -0.26
##             Type         0.2518    0.5018   -0.72  0.51
##   Number of obs: 10365, groups: sample_ID, 18
##
## Fixed effects:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.4230    0.2024 -7.030 2.06e-12 ***

```

```

## her2.binTRUE      1.3197    0.2699    4.890 1.01e-06 ***
## Type             -0.8664    0.1299   -6.669 2.58e-11 ***
## scale(log(age))  0.2480    0.1098    2.258  0.0239 *
## ER              -0.1295    0.2200   -0.588  0.5562
## her2.binTRUE:ER  0.5284    0.3086    1.712  0.0869 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) hr2.TRUE Type   sc((()) ER
## her2.bnTRUE -0.163
## Type        -0.389  0.239
## scal(lg(g)) -0.305  0.002  -0.021
## ER          -0.791  0.059  -0.016  0.351
## hr2.TRUE:ER  0.048 -0.823   0.001  0.031 -0.064

```

## 10 R Session Info

```

sessionInfo()

## R version 3.3.1 RC (2016-06-17 r70798)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: OS X 10.11.6 (El Capitan)
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] parallel stats4 compiler stats     graphics grDevices utils
## [8] datasets methods base
##
## other attached packages:
## [1] org.Hs.eg.db_3.2.3      RSQLite_1.0.0
## [3] DBI_0.4-1               fANCOVA_0.5-1
## [5] vegan_2.4-2             lattice_0.20-33
## [7] permute_0.9-4           TDA_1.5
## [9] spatstat_1.48-0          rpart_4.1-10
## [11] nlme_3.1-128            scatterplot3d_0.3-37
## [13] rgl_0.95.1441            reshape2_1.4.1
## [15] RColorBrewer_1.1-2       plyr_1.8.4
## [17] matrixStats_0.51.0       lme4_1.1-12
## [19] Matrix_1.2-6             leaps_3.0
## [21] knitr_1.14                HTSanalyzeR_2.24.0
## [23] igraph_1.0.1              heatmap.plus_1.3
## [25] GSEABase_1.32.0           graph_1.48.0
## [27] annotate_1.48.0            XML_3.98-1.4
## [29] AnnotationDbi_1.32.3      gplots_3.0.1
## [31] ggplot2_2.1.0              GGally_1.1.0
## [33] geepack_1.2-1              gee_4.13-19
## [35] gdata_2.17.0               fpc_2.1-10
## [37] edgeR_3.12.1              limma_3.28.21
## [39] dplyr_0.4.3                DESeq2_1.12.4

```

```

## [41] SummarizedExperiment_1.2.3 Biobase_2.30.0
## [43] GenomicRanges_1.24.2      GenomeInfoDb_1.8.7
## [45] IRanges_2.6.1            S4Vectors_0.10.3
## [47] BiocGenerics_0.18.0     boot_1.3-18
## [49] beeswarm_0.2.3          ade4_1.7-4
## [51] ape_3.5
##
## loaded via a namespace (and not attached):
## [1] backports_1.0.2           Hmisc_3.17-4        lazyeval_0.2.0
## [4] splines_3.3.1             BiocParallel_1.4.3 digest_0.6.9
## [7] BiocInstaller_1.22.3     htmltools_0.3.5   magrittr_1.5
## [10] tensor_1.5                cluster_2.0.4    prada_1.46.0
## [13] colorspace_1.2-6         rrcov_1.3-11     rglwidget_0.1.1434
## [16] RCurl_1.95-4.8           jsonlite_1.0     genefilter_1.52.1
## [19] survival_2.39-4         polyclip_1.5-6  gtable_0.2.0
## [22] zlibbioc_1.16.0         XVector_0.12.1   RankProd_2.42.0
## [25] kernlab_0.9-25          cellHTS2_2.34.1 prabclus_2.2-6
## [28] DEoptimR_1.0-4          abind_1.4-3     scales_0.4.0
## [31] futile.options_1.0.0     vsn_3.38.0      mvtnorm_1.0-5
## [34] Rcpp_0.12.5              xtable_1.8-2     foreign_0.8-66
## [37] mclust_5.2.2             preprocessCore_1.32.0 Formula_1.2-1
## [40] htmlwidgets_0.6           acepack_1.3-3.3 modeltools_0.2-21
## [43] reshape_0.8.5             flexmix_2.3-13  nnet_7.3-12
## [46] deldir_0.1-12            locfit_1.5-9.1 labeling_0.3
## [49] munsell_0.4.3            tools_3.3.1     evaluate_0.9
## [52] stringr_1.0.0             yaml_2.1.13    goftest_1.0-4
## [55] robustbase_0.92-6        caTools_1.17.1  RBGL_1.46.0
## [58] mime_0.4                  formatR_1.4     BioNet_1.32.0
## [61] biomaRt_2.26.1            affyio_1.40.0   geneplotter_1.48.0
## [64] pcaPP_1.9-60              stringi_1.1.1   futile.logger_1.4.1
## [67] trimcluster_0.1-2        nloptr_1.0.4    markdown_0.7.7
## [70] data.table_1.9.6          bitops_1.0-6    httpuv_1.3.3
## [73] R6_2.1.2                 latticeExtra_0.6-28 affy_1.48.0
## [76] KernSmooth_2.23-15       gridExtra_2.2.1 lambda.r_1.1.7
## [79] MASS_7.3-45                gtools_3.5.0    assertthat_0.1
## [82] chron_2.3-47              Category_2.36.0 rprojroot_1.1
## [85] diptest_0.75-7            mgcv_1.8-12    grid_3.3.1
## [88] class_7.3-14              minqa_1.2.4    rmarkdown_1.3
## [91] shiny_0.13.2

```