Supplementary Code I: *Single-cell transcriptome analysis reveals coordinated ectopic-gene expression patterns in medullary thymic epithelial cells*

Philip Brennecke, Alejandro Reyes, Sheena Pinto, Kristin Rattay,
Michelle Nguyen, Rita Küchler, Wolfgang Huber, Bruno Kyewski and Lars M. Steinmetz

Abstract

This document contains all the code used to analyse the single-cell RNA-seq and the bulk ATAC-seq data from the manuscript titled: *Single-cell transcriptome analysis reveals coordinated ectopic-gene expression patterns in medullary thymic epithelial cells*. The purpose of this document is to provide full transparency of the results presented in the manuscript, as well as to provide a documented and reproducible workflow of the code that was used to generate each number and figure from the manuscript. The detailed legends for each figure can be found in the main text and the Supplementary Figures file.

The data package accompanying this vignette, *Single.mTEC.Transcriptomes*, provides data objects from several sources, including data from [Supp1], among others. Each object is documented in the manual pages of this experiment data package.

1 Mapping statistics

For each single-cell transcriptome, we mapped the sequenced fragments to the Mouse reference genome GRCm38 downloaded from ENSEMBL release 75. For each sample of each sequencing batch, we classified the reads as either mapping uniquely to the reference genome, mapping multiple times to the reference genome, reads which we could not assign to any position of the reference genome or others (e.g reads that only one read pair could be mapped but the read other could not be mapped). The resulting figure can be found in Plot I of this document.

```
data("percentsGG")
ggplot( percentsGG, aes(index, percent, fill=type) ) +
    geom_bar(stat="identity") + facet_grid( batch ~ . ) +
   theme(axis.title = element_text(size = 18),
         axis.text = element_text(size=12),
         legend.text = element_text(size = 12),
         axis.text.x=element_text(angle=90),
```

```
          legend.position="top")
```

2 Identifying variable genes

We summarize single-cell transcriptomes in the form of count matrices in which each row represents a gene and each column represents one cell. The matrix is filled with the number of sequenced fragments whose genomic alignment overlaps with the genomic coordinates of the genes. For this, we only use cells for which more than 40% of the sequenced fragments could be uniquely assigned to the Mouse reference genome. We also discarded 28 cells from the batch 4 (since they were from another cell type that was sequenced together with some of the mTECs).

With the final set of high-quality sequenced transcriptomes, we generated a *DESeq2DataSet* object that also contains the annotation for each cell. We define also the number of cores to use for the calculations.

```
data("mTECdxd")
numCores=1
```

In order to identify genes whose biological variation is above the expected technical noise, we used the method described by [Supp2]. We modified the code from the supplementary material from [Supp2], and implemented the function `testForVar`. This function identifies the genes whose coefficient of variation varies for more than 50% at a FDR of 10% and plots the results (e.g. the one presented in the Plot II).

We run the function `testForvar` with the data from those cells that were not selected for the expression of a TRA-encoding gene (Plot II).

```
deGenesNone = testForVar(
    countTable=counts(dxd)[,colData(dxd)$SurfaceMarker == "None"] )
#save(deGenesNone, file="../data/deGenesNone.RData")
length(deGenesNone)
```

```
## [1] 9689
```

```
nrow(dxd)
```

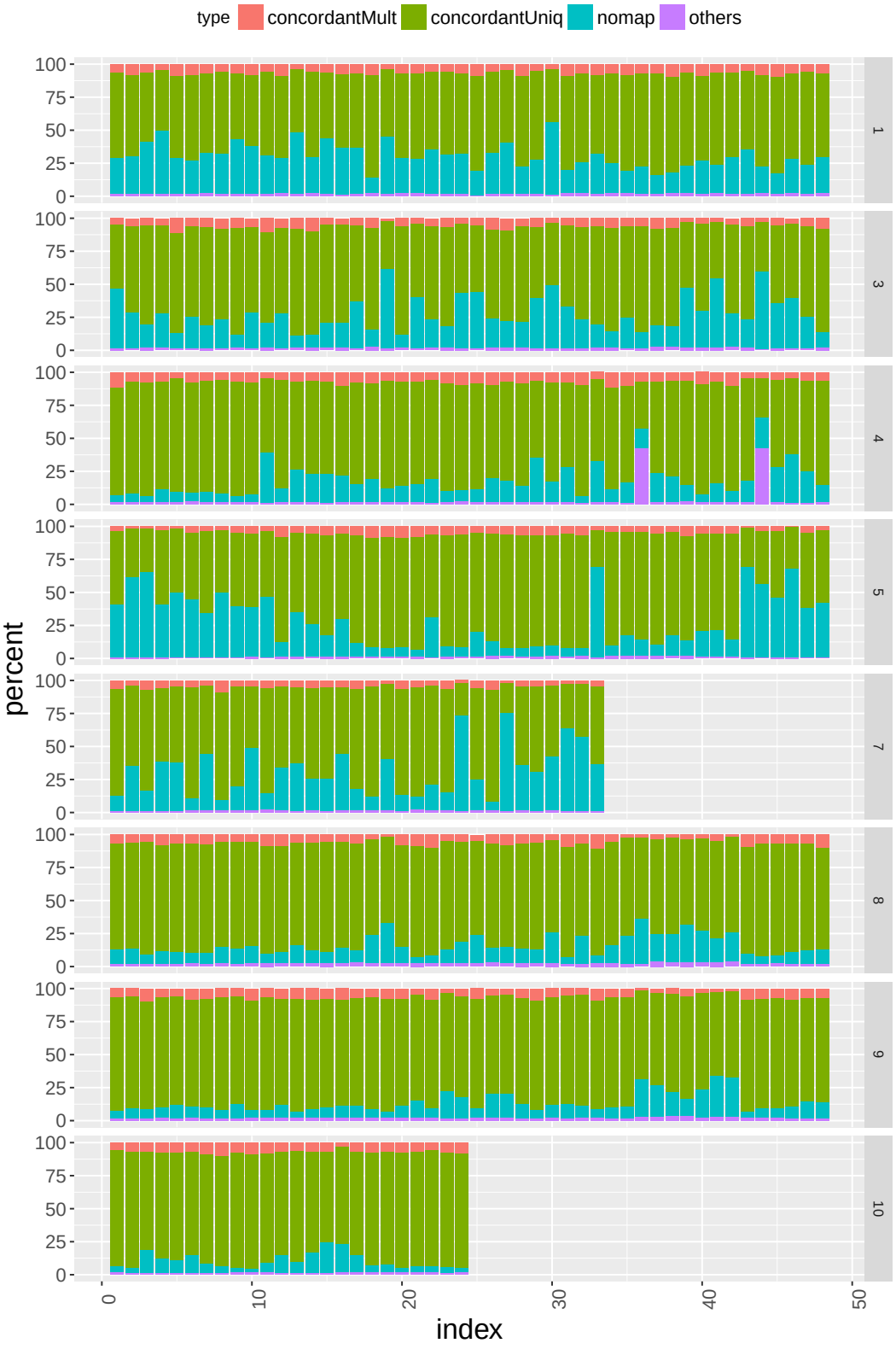
```
## [1] 39271
```

```
ncol(dxd)
```

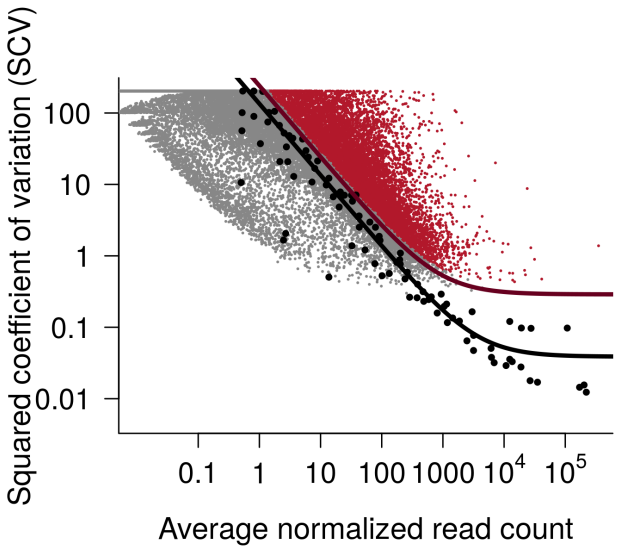
```
## [1] 305
```

```
table( grepl("ERCC", rownames(dxd)) )
```

```
##
## FALSE  TRUE
## 39179    92
```



Plot I: **Mapping statistics stratified by the alignment type.**



Plot II: **Highly variable genes of the unselected cells (n=203).** Corresponds to Figure 1C

The analysis reveals more than 9,000 genes with a coefficient of variation larger than 50% at a False Discovery Rate (FDR) of 10%. This set of genes is used for further analysis.

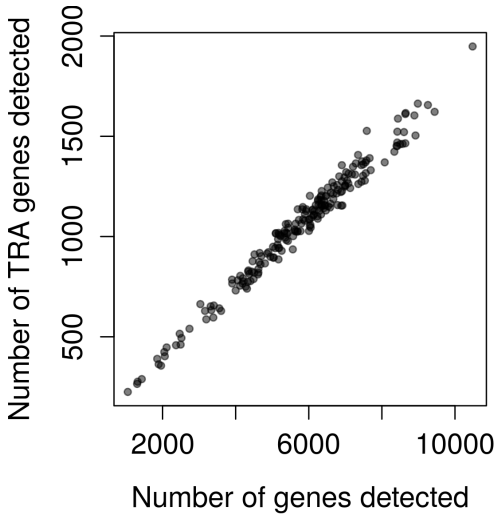
```
cat( sprintf("The number of genes with coefficient of variation larger
than 50 percent at FDR of 0.1 is: %s\n", length(deGenesNone)) )
```

```
## The number of genes with coefficient of variation larger
## than 50 percent at FDR of 0.1 is: 9689
```

3 Number of TRA genes detected per cell compared to the number of genes.

We explore the number of TRA genes (as defined by [Supp3]) and compare this number with the the total number of genes detected per cell. The results can be seen in Plot III.

```
data("tras")
tras <- unique( tras$`gene.ids` )
data("geneNames")
data("biotypes")
proteinCoding <- names( which( biotype == "protein_coding" ) )
isTRA <- rownames( dxd ) %in% tras & rownames(dxd) %in% proteinCoding

isProteinCoding <- rownames(dxd) %in% proteinCoding
sum(isProteinCoding)
```



Plot III: **Number of detected genes vs number of TRA genes.** Corresponds to Figure 1a.

```
## [1] 22740
```

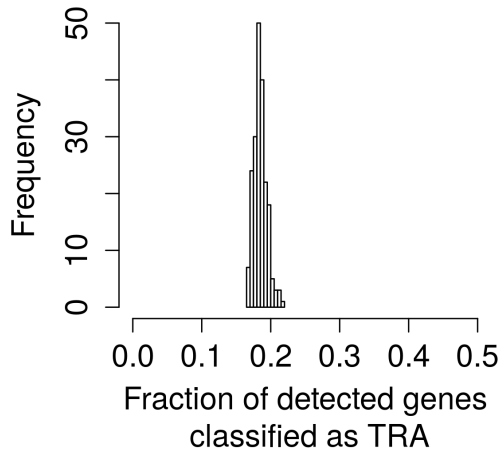
```
par(mar=c(5, 5, 1, 1))
unselectedCells <- colData(dxd)$SurfaceMarker=="None"
sum(unselectedCells)
```

```
## [1] 203
```

```
plot( colSums(counts(dxd)[isProteinCoding,unselectedCells] > 0),
     colSums(counts(dxd)[isTRA,unselectedCells] > 0),
     pch=19, cex=.7, col="#00000080",
#      col=lscol,
     cex.lab=1.3, cex.axis=1.3,
     xlab="Number of genes detected",
     ylab="Number of TRA genes detected")
```

The two numbers seem very correlated and seem to follow a linear relation, indicating that the number of TRA genes that are detected within a cell is proportional to the number of detected genes. The distribution of the fraction of detected genes classified as TRA is depicted in Plot IV.

```
par(mar=c(5, 5, 1, 1))
hist( colSums(counts(dxd)[isTRA,unselectedCells] > 0)/
     colSums(counts(dxd)[isProteinCoding,unselectedCells] > 0),
        col="white", cex.axis=1.4,
        xlab="", cex.lab=1.4,
        pch=19, cex=.5, main="", xlim=c(0, .5))
title(xlab="Fraction of detected genes\n classified as TRA",
      line=4, cex.lab=1.4)
```



Plot IV: **Histogram of the distribution of percentage of TRAs detected per cell.** Corresponds to Figure S1.

```
rng <- range(  colSums(counts(dxd)[isTRA,unselectedCells] > 0)/
      colSums(counts(dxd)[isProteinCoding,unselectedCells] > 0) )
rng <- round(rng*100, 2)

summary(rng)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   16.69   17.98   19.27   19.27   20.56   21.85
```

```
sd(rng)
```

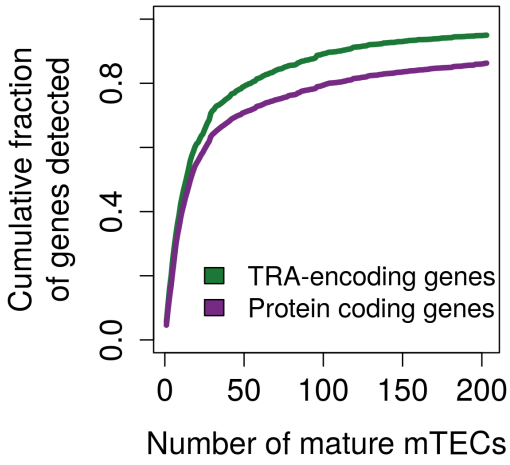
```
## [1] 3.648671
```

```
cat(sprintf("The proportion of TRAs expressed per cell with respect to the
number of protein coding genes ranges from %s to %s percent\n",
            rng[1], rng[2] ) )
```

```
## The proportion of TRAs expressed per cell with respect to the
## number of protein coding genes ranges from 16.69 to 21.85 percent
```

Next, we explore the saturation of genes and TRA genes detection by counting how many genes are detected when progressively adding one cell at the time. To remain our estimates as conservative as possible, we sort the cells based on the number of detected genes (increasingly from left to right in the Plot V).

```
detectedTRAs <- 0+( counts(dxd)[isTRA,unselectedCells] > 0 )
cellOrder <- order( colSums(counts(dxd)[isProteinCoding,unselectedCells] > 0) )
cumFreqTRAs <- sapply( seq_along(cellOrder), function(x){
    sum(!rowSums( detectedTRAs[,cellOrder[seq_len(x)], drop=FALSE] ) == 0)
})
```



Plot V: **Cumulative frequency of detected protein coding genes and TRA genes detected per cell.** Corresponds to Figure 1B.

```
detectedGenes <- 0+( counts(dxd)[isProteinCoding,unselectedCells] > 0 )
cumFreqGenes <- sapply( seq_along(cellOrder), function(x){
    sum(!rowSums( detectedGenes[,cellOrder[seq_len(x)], drop=FALSE] ) == 0)
})

par(mar=c(5, 6, 2, 1))
plot(cumFreqTRAs/nrow(detectedTRAs), type="l", lwd=4,
     col="#1b7837", cex.axis=1.4, cex.lab=1.4,
     ylab="Cumulative fraction\nof genes detected", ylim=c(0, 1),
     xlab="Number of mature mTECs")
lines(cumFreqGenes/nrow(detectedGenes), type="l",
      lwd=4, col="#762a83")
legend(x=10, y=.3,
       legend=c("TRA-encoding genes", "Protein coding genes"),
       fill=c("#1b7837", "#762a83"),
       cex=1.2, bty="n")
```

Pooling our set of 203 mature mTECs, we are able to detect above 85% percent of both TRA genes and the rest of the protein coding genes (Plot V).

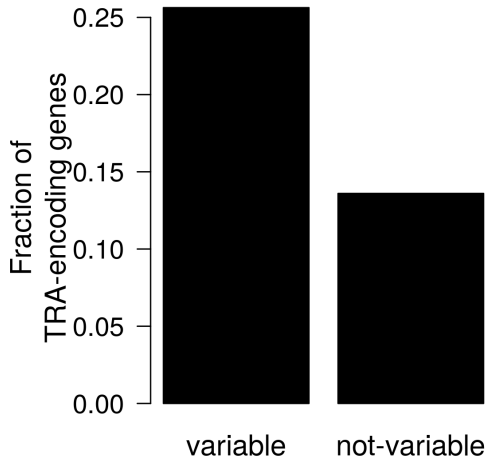
```
cat(sprintf("Total fraction of TRA genes detected: %s",
            round( max(cumFreqTRAs/nrow(detectedTRAs)),2) ) )

## Total fraction of TRA genes detected: 0.95

cat(sprintf("Total number of TRA genes: %s",
            nrow(detectedTRAs) ) )
```

```
## Total number of TRA genes: 3976

cat(sprintf("Total fraction of protein coding genes detected: %s\n",
            round( max(cumFreqGenes/nrow(detectedGenes)), 2) ) )

## Total fraction of protein coding genes detected: 0.86

cat(sprintf("Total number of detected protein coding genes: %s",
            max(cumFreqGenes)))

## Total number of detected protein coding genes: 19619

cat(sprintf("Total number of protein coding genes: %s",
            nrow(detectedGenes) ) )

## Total number of protein coding genes: 22740
```

4 Variable genes are enriched for TRA genes

We create a contigency table comparing the highly variable protein coding genes with respect to all protein coding genes that were not detected as highly variable.

```
background <- rownames(dxd)[!rownames(dxd) %in% deGenesNone]
background <- names( which( rowSums( counts(dxd)[background,] > 0 ) != 0 ) )
background <- intersect(background, proteinCoding)
foreground <- intersect(deGenesNone, proteinCoding )
allTras <- intersect( tras, proteinCoding )

mat <- rbind(
    `variable genes`=table( !foreground %in% allTras ),
    `background`=table( !background %in% allTras ) )

colnames( mat ) <- c("is TRA", "is not TRA")

mat[,1]/rowSums(mat)

## variable genes     background
##      0.2563075      0.1359825

par(mar=c(4, 6, 1, 1))
barplot(
    mat[,1]/rowSums(mat),
    names.arg=c("variable", "not-variable"),
    las=1, col="black",
    cex.axis=1.2, cex.lab=1.3, cex=1.3,
    ylab="Fraction of\nTRA-encoding genes")
```



Plot VI: **Highly variable genes are enriched for TRA-encoding genes.** Corresponds to Fig. S2.

Then, we run the Fisher test on the contigency table, which indicates that the variable genes within the 203 mature mTECs are significantly enriched for TRA genes.

```
fisher.test(mat)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  mat
## p-value < 2.2e-16
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  2.036751 2.354542
## sample estimates:
## odds ratio
##   2.189746
```

5 Both Aire-dependent and Aire-independent TRAs are expressed at low frequencies in single mTECs

This section investigates the number of TRA genes with respect to Aire dependency as defined by [Supp1]. We also use data from the FANTOM consortium (91 manually selected tissues, manually excluding samples that were composed of more than one tissue, e.g. whole body) [Supp4]. For the definition of a gene being detected for a tissues, a threshold of at least five counts was selected, nevertheless this is not a critical parameter for our analysis (One can vary it and the result remains the same).

```
data("fantom")
data("aireDependentSansom")
aireDependent <- aireDependentSansom

countTable <- counts(dxd)[,colData(dxd)$SurfaceMarker == "None"]
meansFANTOM <- sapply( split(seq_len(ncol(dxdFANTOM)),
                  colData( dxdFANTOM )$tissue), function(x){
                  rowMeans(
                    counts(dxdFANTOM, normalized=TRUE)[,x, drop=FALSE] )
})
meansFANTOM <- sapply(
  split( seq_len(
      nrow(meansFANTOM) ),
        sapply( strsplit( rownames( meansFANTOM ), "," ), "[[", 1 )),
    function(x){
      colMeans( meansFANTOM[x,,drop=FALSE] )
    })

meansFANTOM <- t( meansFANTOM )

cat( sprintf("The total number of tissues used from the FANTOM dataset was:%s\n",
             length( unique( colnames(meansFANTOM) ) ) ) )

## The total number of tissues used from the FANTOM dataset was:91

matchedIndexes <- match( geneNames, rownames(meansFANTOM))
stopifnot(
    rownames(meansFANTOM)[matchedIndexes[!is.na(matchedIndexes)]] ==
    geneNames[!is.na(matchedIndexes)] )
rownames(meansFANTOM)[matchedIndexes[!is.na(matchedIndexes)]] <-
    names( geneNames[!is.na(matchedIndexes)] )
meansFANTOM <- meansFANTOM[grep("ENS", rownames(meansFANTOM)),]

numbersOfTissues <- rowSums( meansFANTOM > 5 )
numbersOfTissues <- numbersOfTissues[names(numbersOfTissues) %in% deGenesNone]
aireDependent <- aireDependent[aireDependent %in% deGenesNone]
numbersOfCells <- rowSums( countTable[names(numbersOfTissues),] > 0 )

table( numbersOfTissues < 10 )

##
## FALSE  TRUE
##  7466   912

FifteenPercent <- round( sum(colData(dxd)$SurfaceMarker == "None") * .15 )
```

```
tableResults <- table( lessThan10Cells=numbersOfCells < FifteenPercent,
      isAireDependent=names(numbersOfTissues) %in% aireDependent,
      isLessThan10Tissues=numbersOfTissues < 10)

cat( sprintf("Total number of genes detected in less than 10 tissues: %s\n",
             sum( tableResults[,,"TRUE"]) ) )
```

```
## Total number of genes detected in less than 10 tissues: 912
```

```
cat(sprintf("From which %s are Aire dependent and %s are Aire-independent\n",
            colSums( tableResults[,,"TRUE"] )["TRUE"],
            colSums( tableResults[,,"TRUE"] )["FALSE"] ) )
```

```
## From which 522 are Aire dependent and 390 are Aire-independent
```

```
tableResults["TRUE",,"TRUE"]
```

```
## FALSE  TRUE
##   265   492
```

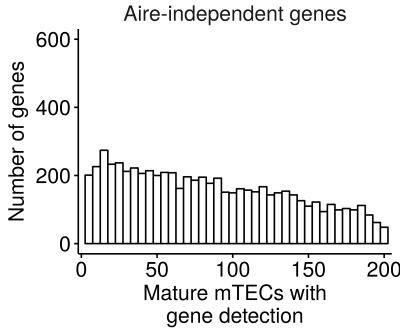
```
percentsLessThan15 <-
    round( tableResults["TRUE",,"TRUE"]/colSums( tableResults[,,"TRUE"] ), 2)


cat(sprintf("And %s and %s were detected in
less than 15 percent of the cells, respectively\n",
            percentsLessThan15[2], percentsLessThan15[1]))
```

```
## And 0.94 and 0.68 were detected in
## less than 15 percent of the cells, respectively
```

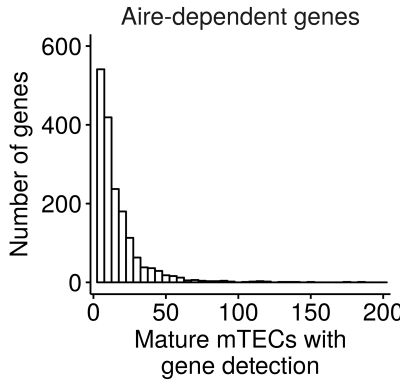
The Plot VII replicates the results from [Supp1], that shows that Aire dependent genes are transcribed at low frequencies in mTECs. Additionally, our analysis shows that transcription at low frequencies is a common property of both Aire-dependent and Aire-independent TRA-encoding genes.

```
df <- data.frame(
  numberOfCells=numbersOfCells,
  numberOfTissues=numbersOfTissues,
  aireDependent=ifelse( names(numbersOfCells) %in% aireDependent,
      "Aire-dependent genes", "Aire-independent genes"),
  TRAs=ifelse( names(numbersOfCells) %in% tras, "TRA", "not TRA" ) )

histogramList <- lapply(c("Aire-independent genes", "Aire-dependent genes"),
                        function(x){
    dfAD <- df[df$aireDependent == x,]
    m <- ggplot( dfAD, aes( x=numberOfCells ) )
    m <- m + geom_histogram(colour = "black", fill = "white", binwidth = 5)
    m <- m + facet_grid( ~ aireDependent )
```

```
    m <- m + xlab("Mature mTECs with\ngene detection") + ylab("Number of genes") +
        scale_x_continuous(limits=c(0, ncol(countTable)+0), expand=c(.01,0) ) +
            ylim(0, 600)
    m <- m + theme(axis.text.x = element_text(size=14, colour="black"),
        panel.border = element_rect(colour = "white", fill=NA),
        panel.background = element_rect(colour="white", fill="white"),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.line=element_line(colour="black"),
        axis.text.y = element_text(size=14, colour="black"),
        axis.title=element_text(size=14, vjust=.2),
        strip.text.x = element_text(size = 14),
        axis.ticks=element_line(colour="black"),
        strip.background = element_rect(colour="white", fill="white"))
    m
})
names(histogramList) <- c("Aire-independent genes", "Aire-dependent genes")


scatterList <- lapply(c("Aire-independent genes", "Aire-dependent genes"),
                      function(x){
    dfAD <- df[df$aireDependent == x,]
        m2 <- ggplot(dfAD, aes(x=numberOfCells,y=numberOfTissues)) +
        geom_point(colour="#15151540", size=1.2) +
        facet_grid( ~ aireDependent ) +
        xlab("Mature mTECs with\n gene detection") +
        ylab("Tissues with gene detection") +
        geom_hline(yintercept=10, size=1.1, colour="darkred") + #linetype="dashed") +
        scale_y_continuous(limits=c(0, max(df$numberOfTissues)+3),
                           expand=c(0,0) ) +
        scale_x_continuous(limits=c(0, ncol(countTable)+0),
                           expand=c(0.01,0) )
        m2 <- m2 + theme(
            strip.background = element_rect(colour="white", fill="white"),
                      strip.text.x =element_text(size = 14),
        panel.border = element_rect(colour = "white", fill=NA),
        axis.line=element_line(colour="black"),
        axis.text = element_text(size=14, colour="black"),
        axis.title=element_text(size=14, vjust=.2),
        legend.position="none",
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank())
   m2})
```



Plot VII: **Histogram and heatscatter of genes stratified by Aire regulation.**



Plot VIII: **Histogram and heatscatter of genes stratified by Aire regulation.**

```
names(scatterList) <- c("Aire-independent genes", "Aire-dependent genes")
```

```
print( histogramList[["Aire-independent genes"]])
```

```
## Warning:  Removed 1 rows containing missing values (geom_bar).
```

```
print( histogramList[["Aire-dependent genes"]])
```

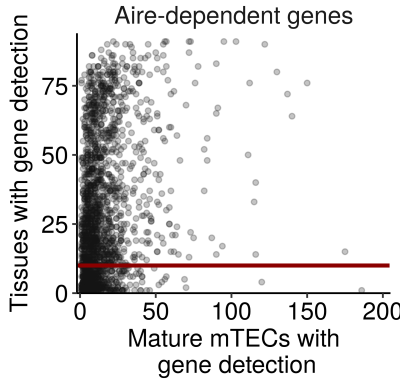
```
## Warning:  Removed 1 rows containing missing values (geom_bar).
```

```
print( scatterList[["Aire-dependent genes"]])
```

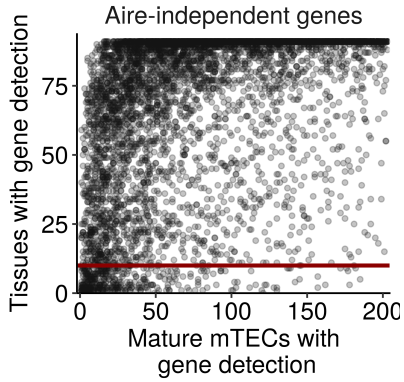
```
print( scatterList[["Aire-independent genes"]])
```

6 Gene-gene correlation analyses

Since cell cycle is a potential confounding factor for gene-gene correlations, we regress out cell cycle variation by using the method *scLVM* described in [Supp5]. Then, in order to explore potential patterns of co-regulation in mTEC cells, we use k-medoids clustering. Importantly, k-medoids minimizes the



Plot IX: **Histogram and heatscatter of genes stratified by Aire regulation.**



Plot X: **Histogram and heatscatter of genes stratified by Aire regulation.**

s

eucledian distance between the data points labeled to be in a cluster and a point designated as the center of that cluster (centroid). We then assess the stability of the clusters using the *R* package "clue", we resampled 1,000 times and estimate the consensus clustering. As we are performing 1,000 permutations to asses the stability of the clusters, the code below does not run during the compilation of this document. Instead, the results were pre-saved in an object of the data package. To continue with the vignette, we load the object that contains the pre-calculated results of the k-medoids clustering.

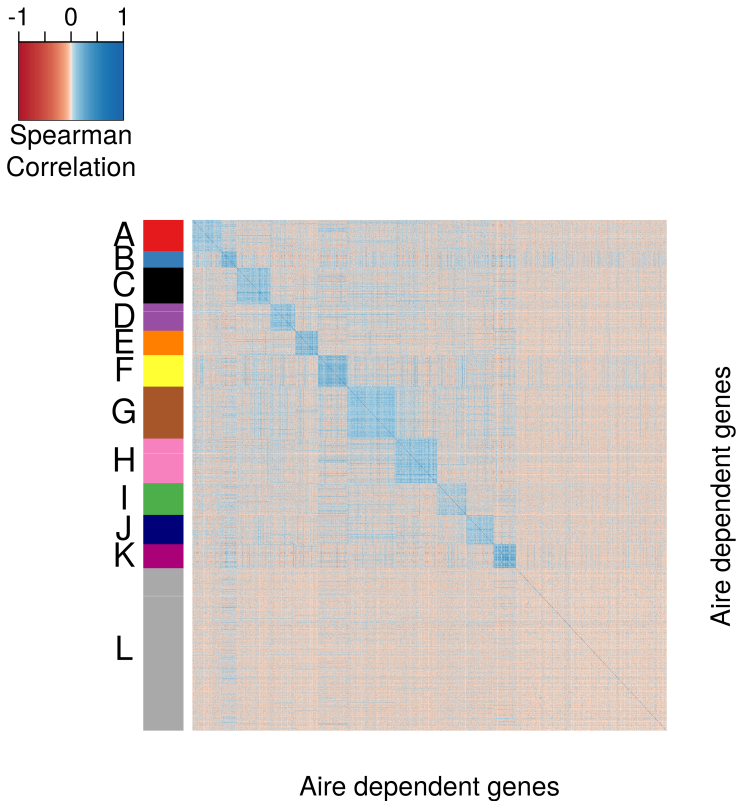
```
library(clue)
library(cluster)
data("scLVM_output")

defineGeneConsensusClustering <-
    function( expressionMatrix=Ycorr,
              selectGenes="", nClusters=12, B=40,
              prob=0.8, nCores=20, ...){
     mat <- expressionMatrix[rownames(expressionMatrix) %in% selectGenes,]
     ce <- cl_ensemble( list=mclapply(  seq_len(B), function(dummy){
        subSamples <- sample(colnames(mat), round( ncol(mat) * prob ) )
```

```
        pamSub <- pam( mat[,subSamples], nClusters )
        pred <- cl_predict( pamSub, mat[,subSamples], "memberships" )
        as.cl_partition(pred)
    }, mc.cores=nCores ))
    cons <- cl_consensus( ce )
    ag <- sapply( ce, cl_agreement, y=cons )
    return(list(consensus=cons, aggrementProbabilities=ag))
}

nomarkerCellsClustering <-
    defineGeneConsensusClustering(
        expressionMatrix=Ycorr[,colData(dxd)$SurfaceMarker == "None"],
        selectGenes=intersect( deGenesNone, aireDependent ),
        nClusters=12, B=1000,
        prob=0.8, nCores=10 )

save( nomarkerCellsClustering,
     file="../data/nomarkerCellsClustering.RData")
```

We visualize the result of the clustering next to the gene-gene Spearman correlation matrix. The resulting heatmap reveals that most of the defined gene clusters, as expected, tend to have higher correlations with the genes grouped in the same the cluster compared to genes from other clusters (Figure XI).

```
library(clue)
library(cluster)
data("nomarkerCellsClustering")
data("scLVM_output")
data("corMatsNoMarker")

#source( file.path(
#    system.file(package="Single.mTec.Transcriptomes"),
#    "extfunction", "heatmap3.R" ) )

nClusters <- 12
colClusters <- brewer.pal(9, "Set1")
colClusters[9] <- colClusters[3]
colClusters[3] <- "black"
colClusters[10] <- "darkgray"
colClusters[11] <- "#000078"
colClusters[12] <- "#AA0078"

nonAssignedCluster <- 10
specialSort <- function(cons=nomarkerCellsClustering[["consensus"]], sendLast=10){
    rt <- sort( cl_class_ids(cons) )
```

```
    wLast <- rt == sendLast
    c( rt[!wLast], rt[wLast])
}
specialOrder <- function(cons=nomarkerCellsClustering[["consensus"]], sendLast=10){
    or <- order(cl_class_ids(cons))
    rt <- sort( cl_class_ids(cons) )
    wLast <- rt == sendLast
    c( or[!wLast], or[wLast])
}

geneGeneCorHeatmap <- function( cons=allCellsClustering[["consensus"]],
                                corMat=corMatSp,
                                expressionMatrix=Ycorr,
                                selectGenes=intersect(deGenes, aireDependent),
                                colorClusters=""){
  mat <- expressionMatrix[rownames(expressionMatrix) %in% selectGenes,]
  or <- rownames(mat)[specialOrder( cl_class_ids(cons), nonAssignedCluster)]
  orderRows <- specialSort( cl_class_ids(cons), nonAssignedCluster)
  corMat <-
      corMat[rownames(corMat) %in% rownames(mat),
             colnames(corMat) %in% rownames(mat)]
  rowCols <- matrix( colClusters[orderRows], nrow=1 )
  br <- seq(-1, 1, length.out=101) ** 3
  cols <-
      colorRampPalette(
          brewer.pal(9, "RdBu"), interpolate="spline", space="Lab")(100)
  heatmap.3( corMat[or,or], symm=TRUE, Colv=FALSE,
          Rowv=FALSE, dendrogram="none", trace="none", breaks=br,
          col=cols, RowSideColors=rowCols,
          ColSideColors=NULL,
          labCol=rep("", nrow(corMat) ),
          labRow=rep("", nrow(corMat) ),
          margins = c(4,4), KeyValueName="Spearman\nCorrelation",
          keysize=1.5, xlab="Aire dependent genes",
          ylab="\t\tAire dependent genes",
          NumColSideColors=1.8, NumRowSideColors=1.2)
}
par(xpd=TRUE)
geneGeneCorHeatmap(cons=nomarkerCellsClustering[["consensus"]],
                   corMat=corMatSpNoMarker,
                   expressionMatrix=Ycorr[,colData(dxd)$SurfaceMarker == "None"],
                   selectGenes=intersect( deGenesNone, aireDependent ),
                   colorClusters=colClusters)
freqs <- rle( specialSort( cl_class_ids(nomarkerCellsClustering[["consensus"]]),
```



Plot XI: **Gene-gene correlation matrix. Rows and columns are ordered based on the k-medoids clustering results.** Corresponds to Figure 2A.

```
                          nonAssignedCluster) )$lengths
freqs <- c(0, freqs)
freqs <- cumsum( freqs ) / max(cumsum(freqs))
freqs <- 1-freqs
freqs <- sapply( seq_len(length(freqs)-1), function(x){
    (freqs[x] + freqs[x+1])/2
})
freqs <- ( freqs-.125 ) / 1.085
text(LETTERS[seq_len(nClusters)], x=.12, y=(freqs), cex=1.5)
length(intersect( deGenesNone, aireDependent ) )
```

```
## [1] 2174
```

When plotting the expression matrix in a heatmap representation, we can see that most of the gene clusters are particularly pronounced in a small fraction of the cells (Plot XII).

```
colTspanRNA <- "#b2df8a"
geneExprHeatmap <- function(cons=allCellsClustering[["consensus"]],
                            corMat=corMatSp,
                            expressionMatrix=Ycorr,
                            selectGenes=intersect(deGenes, aireDependent),
```

```
                            colorClusters="", ylab="\t\tAire dependent genes"){
mat <- expressionMatrix[rownames(expressionMatrix) %in% selectGenes,]
or <- rownames(mat)[specialOrder( cl_class_ids(cons), nonAssignedCluster)]
orderRows <- specialSort( cl_class_ids(cons), nonAssignedCluster)
cols2 <-
    colorRampPalette(
        brewer.pal(9,  name="Blues"),
        interpolate="spline", space="Lab")(100)
br2 <- seq(0.01, 5, length.out=101)
mat <- expressionMatrix[rownames(expressionMatrix) %in% selectGenes,]
rowCols <- matrix( colClusters[orderRows], nrow=1 )
colCols1 <- matrix(
    ifelse( colData(dxd)[colnames(mat),]$SurfaceMarker == "Tspan8",
           "#c51b7d", "white"),
    ncol=1)
colCols2 <- matrix(
    ifelse( colData(dxd)[colnames(mat),]$SurfaceMarker == "Ceacam1",
           "#6a3d9a", "white"),
    ncol=1)
colCols= cbind(colCols1, colCols2)
colnames(colCols) <- c("Tspan8 +", "Ceacam1 +")
if(all(colCols == "white")){
  colCols=NULL
  ranks <- rank(
          counts(dxd,
                 normalized=TRUE)[
                     names( geneNames[geneNames %in% "Tspan8"] ),
                     colnames(mat)],
          ties.method="min")
  cols <- colorRampPalette(c("white", colTspanRNA))(length(unique(ranks)))
  names(cols) <- as.character( sort(unique( ranks ) ))
  colCols <- matrix( cols[as.character(ranks)], ncol=1)
  mat <- mat[,order( ranks )]
  colCols <- colCols[order(ranks),, drop=FALSE]
}
heatmap.3( mat[or,], symm=FALSE, Colv=FALSE,
          Rowv=FALSE, dendrogram="none", trace="none", breaks=br2,
          col=cols2, RowSideColors=rowCols,
          ColSideColors=colCols,
          labCol=rep("", nrow(mat) ),
          labRow=rep("", nrow(mat) ),
          margins=c(4,4),
          keysize=1.45,
          NumColSideColors=1.2,
```

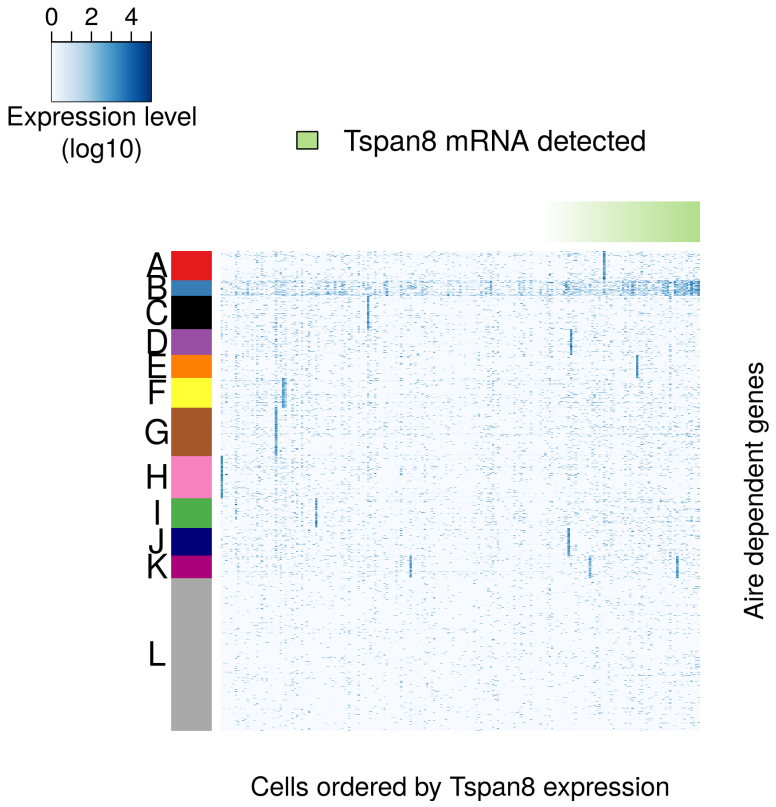
```
          NumRowSideColors=1.2,
          KeyValueName="Expression level\n(log10)",
          xlab="Cells ordered by Tspan8 expression",
          ylab=ylab)
}

genesForClustering <- intersect( deGenesNone, aireDependent )
genesForClustering <- rownames(Ycorr)[rownames(Ycorr) %in% genesForClustering ]

geneClusters <-
    split(genesForClustering,
          cl_class_ids( nomarkerCellsClustering[["consensus"]] ))

geneClusters <- c( geneClusters[-nonAssignedCluster],
                   geneClusters[nonAssignedCluster])

grep(names( geneNames[geneNames %in% "Tspan8"] ),geneClusters)
```

```
## [1] 2
```

```
par(xpd=TRUE)
geneExprHeatmap(cons=nomarkerCellsClustering[["consensus"]],
                corMat=corMatSpNoMarker,
                expressionMatrix=Ycorr[,colData(dxd)$SurfaceMarker == "None"],
                selectGenes=intersect(deGenesNone, aireDependent) )
legend( x=.3, y=1.01,
       legend="Tspan8 mRNA detected",
       fill=colTspanRNA, cex=1.3, bty="n")
freqs <- rle( specialSort(
    cl_class_ids(nomarkerCellsClustering[["consensus"]]) ) )$lengths
freqs <- c(0, freqs)
freqs <- cumsum( freqs ) / max(cumsum(freqs))
freqs <- 1-freqs
freqs <- sapply( seq_len(length(freqs)-1), function(x){
    (freqs[x] + freqs[x+1])/2
})
freqs <- ( freqs-.135 ) / 1.155
text(LETTERS[seq_len(nClusters)], x=.12, y=(freqs), cex=1.5)
```

6.0.1 Correlations in Samson et al. dataset

As an additional check, we processed the *Sansom et al.* single-cell dataset in the same manner as our data. We estimated a gene-gene correlation matrix using the *Samson et al.* dataset. Then, for each gene group defined using our data, we test if we see high correlations within the gene groups compared



Plot XII: **Gene expression heatmap. The rows are ordered based on clustering results and the columns are ordered by the Tspan8 expression levels.** Corresponds to Figure 2B.

to the genes outside the gene group groups on the *Samson et al.* dataset.

```
data("corMatsSansom")
data("deGenesSansom")

geneClustersDESansom <- lapply( geneClusters, function(x){
    intersect( x, rownames(corMatSp) )
})

matToUse <- corMatSp

densities <- lapply( seq_along(geneClusters), function(x){
  inClust <- rownames(matToUse) %in% geneClustersDESansom[[x]]
  as.numeric(matToUse[inClust,inClust][upper.tri(matToUse[inClust,inClust])])
})

backgroundDensity <- sample( unlist(geneClustersDESansom), 500 )
inBack <- rownames(matToUse) %in% backgroundDensity
backgroundDensity <-
    as.numeric(matToUse[inBack,inBack][upper.tri(matToUse[inBack,inBack])])
```

```
densitiesComp <- list()
for(i in seq_along(densities)){
    densities[[i]] <- densities[[i]][!is.na(densities[[i]])]
    densitiesComp[[i]] <- backgroundDensity
}

t.test(unlist(densities[1:11]), backgroundDensity)
```

```
##
##  Welch Two Sample t-test
##
## data:  unlist(densities[1:11]) and backgroundDensity
## t = 12.507, df = 198680, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.004336893 0.005948747
## sample estimates:
##  mean of x  mean of y
## 0.02013782 0.01499500
```

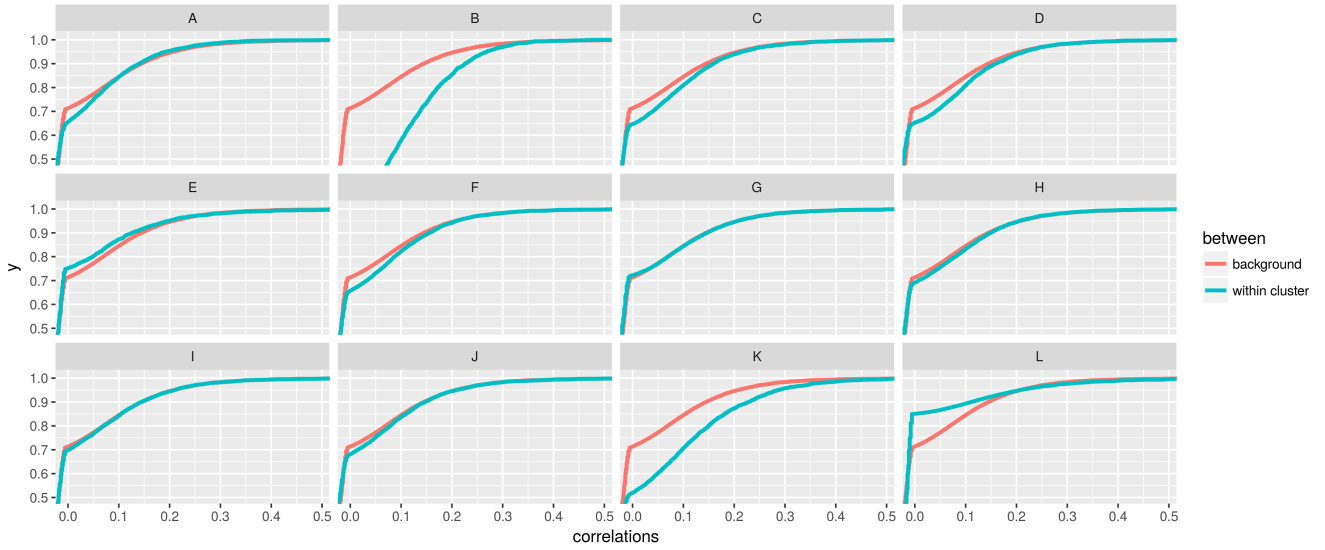
```
pvals <- sapply( seq_along(geneClusters), function(i){
  t.test( densities[[i]], densitiesComp[[i]], alternative="greater")$p.value
})

significant <- which( p.adjust( pvals, method="BH") < 0.05 )
LETTERS[significant]
```

```
## [1] "B" "C" "D" "F" "K"
```

```
significant
```

```
## [1]  2  3  4  6 11
```

```
significant <- 1:12

library(geneplotter)

dfDensities <-
    do.call(rbind,
        lapply(significant, function(i){
  dfDensities <-
      data.frame(
      correlations=c( densities[[i]], densitiesComp[[i]] ),
      between=rep( c("within cluster", "background"),
          time=c(length(densities[[i]]), length(densitiesComp[[i]]) ) ))
  dfDensities <- dfDensities[!is.nan(dfDensities$correlations),]
  dfDensities$cluster <- LETTERS[i]
```



Plot XIII: **Correlation distribution from the Sansom et al. data within our 11 gene clusters.** We estimate the gene-gene correlation coefficients from the *Sansom et al.* data.

```
  dfDensities
}) )


ggplot( dfDensities, aes(correlations, colour=between)) +
    stat_ecdf(lwd=1.2) +
    facet_wrap(~cluster) + coord_cartesian(xlim = c(0, .49), ylim=c(.5, 1.01))
```

7 Data analysis for TRA-selected cells

In order to validate the presence of co-regulated groups of genes based on single mTECs, we perform a second analytical approach inspired on the analysis by [Supp3].

7.1 Tspan8

We select *Tspan8*, that is one of the genes belonging to the Group "B" of the k-medoids clustering.

```
expressionMat <- as.matrix(Ycorr[,colData(dxd)$SurfaceMarker=="None"])
whichCluster <- grep( names( geneNames[geneNames %in% "Tspan8"] ), geneClusters )

cat(sprintf("Tspan8 belonged to cluster %s\n", whichCluster))

## Tspan8 belonged to cluster 2
```

```
maxCorTspan8 <- which.max( sapply(seq_len(nClusters), function(i){
    cor(
      expressionMat[names( geneNames[geneNames %in% "Tspan8"] ),],
      colMeans( expressionMat[geneClusters[[i]],])
    , method="spearman")
}) )

maxCorTspan8
```

```
## [1] 2
```

```
cat(sprintf("Tspan8 displayed the highest correlation with Cluster %s\n",
            maxCorTspan8))
```

```
## Tspan8 displayed the highest correlation with Cluster 2
```

Next, we screen the cells for which the expression of Tspan8 is detected.

```
expressingTspan8 <-
    counts(dxd)[names(geneNames[geneNames %in% "Tspan8"]),
                colData(dxd)$SurfaceMarker == "None"] > 0

sum( expressingTspan8 )
```

```
## [1] 66
```

```
table( expressingTspan8 )[["TRUE"]]/sum(table(expressingTspan8))
```

```
## [1] 0.3251232
```

```
expressingCeacam1 <-
    counts(dxd)[names(geneNames[geneNames %in% "Ceacam1"]),
                colData(dxd)$SurfaceMarker == "None"] > 0

sum( expressingCeacam1 )
```

```
## [1] 31
```

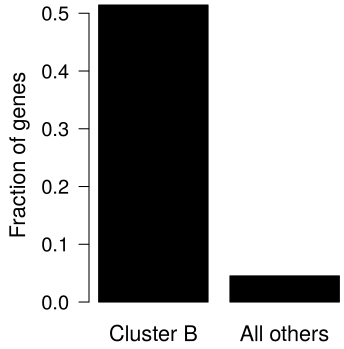
```
table( expressingCeacam1 )[["TRUE"]]/sum(table(expressingCeacam1))
```

```
## [1] 0.1527094
```

Then, for each of the highly variable genes, we test for instances that are up-regulated in the cells detecting expression of *Tspan8* with respect to the cells where the Tspan8 expression is not detected (co-expression). We do this using a Wilcoxon test. As expected, the set of *Tspan8* co-expressed genes is highly enriched for genes from Cluster "B" when compared to the rest of the clusters.

```
dxdUnselected <- dxd[,colData( dxd )$SurfaceMarker == "None"]

rowwilcoxtests <- function(x, fac){
    sp <- split( seq_len(ncol(x)), fac )
```

```
    true <- sp[["TRUE"]]
    false <- sp[["FALSE"]]
    rs <- ( mclapply( seq_len(nrow(x)), function(i){
      wt <- wilcox.test( x[i,true], x[i,false] )
      c(meanA=mean(x[i,true]), meanB=mean(x[i,false]), pval=wt$p.value)
    }, mc.cores=numCores) )
    as.data.frame( do.call(rbind, rs) )
}

deGenesNoneCorrected <- deGenesNone[deGenesNone %in% rownames(Ycorr)]

getCoexpressionGroup <- function(gene){
  cellGroups <- counts(dxdUnselected, normalized=TRUE)[gene,] > 0
  res <- rowwilcoxtests(
    x= as.matrix(Ycorr[deGenesNoneCorrected,
        colData(dxd)$SurfaceMarker=="None"] ),
    fac=cellGroups )
  indFilter <- apply(
      counts(dxdUnselected, normalized=TRUE)[deGenesNoneCorrected,], 1,
      function(x){ mean(x[x != 0]) }) > 150 &
          rowSums(counts(dxdUnselected)[deGenesNoneCorrected,] > 0) > 5
  res$pval[!indFilter] <- NA
  coexpressionGroup <-
      deGenesNoneCorrected[which(p.adjust( res$pval, method="BH" ) < 0.1 &
                           res$meanA - res$meanB > 0)]
  return(coexpressionGroup)
}


gene <- names( geneNames[geneNames %in% "Tspan8"] )
names(gene) <- "Tspan8"
tspan8CoexpressionGroup <- getCoexpressionGroup(gene)

cat( sprintf("Number of differentially expressed genes at a FDR of .05: %s\n",
        length(tspan8CoexpressionGroup)-1) )

## Number of differentially expressed genes at a FDR of .05: 595

coexpressedAndAireDependent <-
    tspan8CoexpressionGroup[tspan8CoexpressionGroup %in% aireDependent]

mat <- rbind(
    table( geneClusters[[whichCluster]] %in% coexpressedAndAireDependent ),
    table( unlist(geneClusters[!seq_len(nClusters) %in% whichCluster] )
      %in% coexpressedAndAireDependent ))[,2:1]
```



Plot XIV: **Enrichment of Tspan8 co-expressed gene set in cluster B.** Corresponds to S3.

```
fisher.test(mat)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  mat
## p-value < 2.2e-16
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  12.90834 38.44891
## sample estimates:
## odds ratio
##   22.23925
```

```
dfPrint <- data.frame(
    `ensembl_gene_name`=tspan8CoexpressionGroup,
    `gene_name`=geneNames[tspan8CoexpressionGroup],
    `aire_dependent`=as.numeric(tspan8CoexpressionGroup %in% aireDependent),
    `cluster_2`=as.numeric( tspan8CoexpressionGroup %in% geneClusters[[whichCluster]]) )


write.table(dfPrint, quote=FALSE, sep="\t", row.names=FALSE, file="figure/tspan8Coexpressi
```

```
par(mar=c(4, 6, 1, 1))
barplot(
    mat[,1]/rowSums(mat),
    names.arg=c("Cluster B", "All others"),
    las=1, col="black",
    cex.axis=1.2, cex.lab=1.3, cex=1.3,
    ylab="Fraction of genes")
```

Next, we used FACS to select mTECs that were expressing Tspan8 in the cell surface. We evaluate the

consistency of the fold changes between the *Tspan8* unselected positive cells and the Tspan8 FACS positive cells, both with respect to the *Tspan8* unselected negative cells. We observe that the fold changes are highly consistent.

```
getFoldChangesForViolin <- function(gene, coexpressedGenes){
   cellGroups <- counts(dxdUnselected, normalized=TRUE)[gene,] > 0
   cellGroups <- split( names(cellGroups), cellGroups)
   outGroup <- rowMeans( Ycorr[deGenesNoneCorrected,
                               cellGroups[["FALSE"]]] )
   inGroupSelected <-
    rowMeans( Ycorr[deGenesNoneCorrected,
                    colData(dxd)$SurfaceMarker==names(gene)] )
   names(outGroup) <- deGenesNoneCorrected
   names(inGroupSelected) <- deGenesNoneCorrected
   geneIndexes <- deGenesNoneCorrected %in% coexpressedGenes
   l2fc <- (inGroupSelected - outGroup)/log10(2)
   toRet <- list()
   toRet[["coexpressed"]] <- l2fc[geneIndexes]
   toRet[["background"]] <- l2fc[!geneIndexes]
   toRet
}

foldChangesTRApos <- list()
foldChangesTRApos[["Tspan8"]] <-
    getFoldChangesForViolin( gene, tspan8CoexpressionGroup)

getFoldChangesForScatter <- function(gene, coexpressedGenes){
   cellGroups <- counts(dxdUnselected, normalized=TRUE)[gene,] > 0
   cellGroups <- split( names(cellGroups), cellGroups)
   outGroup <- rowMeans( Ycorr[deGenesNoneCorrected,
                               cellGroups[["FALSE"]]] )
   inGroup <- rowMeans( Ycorr[deGenesNoneCorrected,
                               cellGroups[["TRUE"]]] )
   inGroupSelected <-
    rowMeans( Ycorr[deGenesNoneCorrected,
                    colData(dxd)$SurfaceMarker==names(gene)] )
   names(outGroup) <- deGenesNoneCorrected
   names(inGroupSelected) <- deGenesNoneCorrected
   names(inGroup) <- deGenesNoneCorrected
   toRet <- list()
   toRet[["preenriched"]] <-
       (inGroupSelected - outGroup)/log10(2)
   toRet[["unselected"]] <-
       (inGroup - outGroup)/log10(2)
   toRet
```

```
}

foldChangesAll <- list()
foldChangesAll[["Tspan8"]] <-
    getFoldChangesForScatter(gene, tspan8CoexpressionGroup)

havePosFoldChange <- table( foldChangesTRApos[["Tspan8"]][["coexpressed"]] > 0 )

cat(sprintf("%s percent (%s out of %s) of the up-regulated
genes based on the unselected mTECs have a consistent fold
change in the Tspan8+ cells\n",
   round( havePosFoldChange["TRUE"] / sum( havePosFoldChange ), 2 ) * 100,
   havePosFoldChange["TRUE"], sum(havePosFoldChange)))
```

```
## 96 percent (572 out of 596) of the up-regulated
## genes based on the unselected mTECs have a consistent fold
## change in the Tspan8+ cells
```

We can observe the validation of the *Tspan8* co-expression group in a heatmap representation of expression Z-scores, defined as,

$$Z_{ij} = \frac{X_{ij} - \text{Median}_j\left(X_{ij}\right)}{\sqrt{\text{Var}_j\left(X_{ij}\right)}} \tag{1}$$

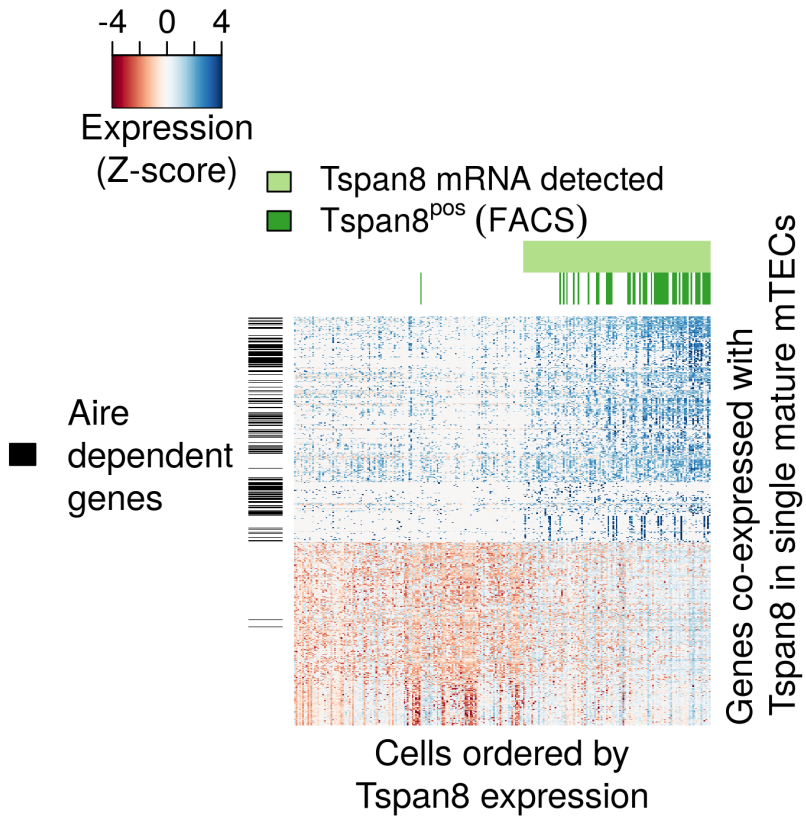
where the genes are indexed by i and the cells are indexed by j. X_{ij} represents the expression levels after regressing the cell cycle variation.

```
library(matrixStats)

colTspanPos <- "#33a02c"

makeHeatmapForGene <- function(gene, coexpressionGroup,
                               colPos, colRNA, enrichMethod, legendLabs){
   cellsToUse <- colData(dxd)$SurfaceMarker %in% c("None", names(gene))
   goodOrder <- order( counts(dxd, normalized=TRUE)[gene,cellsToUse] )
   colPca <-
       ifelse( colData(dxd)[cellsToUse,"SurfaceMarker"] == names(gene),
              colPos, "white")
   nonZero <- counts(dxd, normalized=TRUE)[gene,cellsToUse] > 0
   colPca2 <- rep("white", sum(cellsToUse))
   colPca2[nonZero] <- colRNA
   cols <-
       colorRampPalette( brewer.pal(11,  name="RdBu"),
                        interpolate="spline", space="Lab")(100)
   expr <- as.matrix(Ycorr[coexpressionGroup,cellsToUse] / log10(2))
```

```
   rowCols <- ifelse( rownames(expr) %in% aireDependent, "black", "white" )
   br <- seq( min(expr), 4, length.out=101 )
   rowCols <- matrix(rowCols, nrow=1)
   colCols <- matrix(colPca[goodOrder], ncol=1)
   colCols <- as.matrix(cbind(colPca, colPca2))[goodOrder,]
   colnames(colCols) <- NULL
   heatmapMatrix <- (expr[,goodOrder] - rowMedians(expr[,goodOrder])) /
      rowSds(expr[,goodOrder])
   br <- seq(-4, 4, length.out=101)
   par(xpd=TRUE)
   heatmap.3( heatmapMatrix,
         trace="none", ColSideColors=colCols,
         Colv=FALSE, col=cols, dendrogram="none",
         labCol=rep("", ncol(expr)), labRow=rep("", nrow(expr) ),
         RowSideColor=rowCols,
         margins = c(4,4),
         KeyValueName="Expression\n(Z-score)",
         keysize=1.9,
         NumColSideColors=2, NumRowSideColors=1.2,
         breaks=br,
         xlab=sprintf("Cells ordered by\n%s expression", names(gene)),
         ylab=sprintf("Genes co-expressed with \n%s in single mature mTECs",
             names(gene)) )
   legend( x=.15, y=1.01,
      legend=legendLabs,
      fill=c(colRNA, colPos), cex=1.1,
      bty="n")
   legend( x=-.3, y=.6,
      legend="Aire\ndependent\ngenes",
      fill="black", cex=1.2,
      bty="n")
}

makeHeatmapForGene(gene, tspan8CoexpressionGroup,
                   colTspanPos, colTspanRNA,
                   legendLabs=c("Tspan8 mRNA detected",
                       expression(Tspan8^"pos"~(FACS))))
```

7.2 Ceacam1

Next, we perform the same analysis as with *Tspan8*, but this time we select an Aire-independent TRA.



Plot XV: **Heatmap of the expression of co-expressed genes with Tspan8.** Corresponds to Figure 3B.

```
nonZeros=!counts(dxd)[names( geneNames[geneNames %in% "Ceacam1"] ),
                      colData(dxd)$SurfaceMarker == "None"] == 0

nonZeros <- colData(dxd)$SurfaceMarker == "None" &
    counts(dxd)[names( geneNames[geneNames %in% "Ceacam1"]),] != 0

cor.test(
   as.numeric(Ycorr[names(geneNames[geneNames %in% "Ceacam1"]),nonZeros][1,]),
   colMeans( Ycorr[geneClusters[[whichCluster]],nonZeros]), method="spearman" )

##
##  Spearman's rank correlation rho
##
## data:  as.numeric(Ycorr[names(geneNames[geneNames %in% "Ceacam1"]),  and colMeans(Ycorr
## S = 2944, p-value = 0.02406
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.4064516
```

We do the testing as we did with *Tspan8* previously, but now for *Ceacam1*. We observe a signifcant overlap with the *Tspan8* co-expressed gene set.

```
gene <- names( geneNames[geneNames %in% "Ceacam1"] )
names(gene) <- "Ceacam1"

ceacam1CoexpressionGroup <-  getCoexpressionGroup(gene)

table( ceacam1CoexpressionGroup %in% aireDependent )
```

```
##
## FALSE  TRUE
##    43    23
```

```
cat( sprintf("Number of differentially expressed genes at a FDR of .1: %s\n",
       length(ceacam1CoexpressionGroup) -1) )
```

```
## Number of differentially expressed genes at a FDR of .1: 65
```

```
cat(sprintf("The number of genes overlapping between co-expression
groups is %s\n",
        table( ceacam1CoexpressionGroup %in% tspan8CoexpressionGroup )["TRUE"]))
```

```
## The number of genes overlapping between co-expression
## groups is 39
```

```
percentageOverlap <-
   table( ceacam1CoexpressionGroup %in% tspan8CoexpressionGroup )[["TRUE"]] /
   length( ceacam1CoexpressionGroup )

cat(sprintf("In percentage of Ceacam1 genes %s\n",
           round( percentageOverlap * 100, 2)))
```

```
## In percentage of Ceacam1 genes 59.09
```

```
mat <- rbind(
    table(ceacam1CoexpressionGroup %in% tspan8CoexpressionGroup),
    table( deGenesNone[!deGenesNone %in% ceacam1CoexpressionGroup] %in%
          tspan8CoexpressionGroup ) )[,2:1]

rownames( mat ) <- c("coexpressed", "not coexpressed")
colnames( mat ) <- c("overlaps with Tspan8", "does not overlap")

fisher.test( mat )
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  mat
## p-value < 2.2e-16
```

```
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  13.90046 40.24641
## sample estimates:
## odds ratio
##   23.48032
```

```
mat[1,] /sum(mat[1,])
```

```
## overlaps with Tspan8     does not overlap
##            0.5909091            0.4090909
```

```
table( ceacam1CoexpressionGroup %in% tspan8CoexpressionGroup )
```

```
##
## FALSE  TRUE
##    27    39
```

```
table( ceacam1CoexpressionGroup %in% tspan8CoexpressionGroup )[["FALSE"]] /
   length( ceacam1CoexpressionGroup )
```

```
## [1] 0.4090909
```

```
table( tspan8CoexpressionGroup %in% ceacam1CoexpressionGroup )
```

```
##
## FALSE  TRUE
##   557    39
```

```
table( tspan8CoexpressionGroup %in% ceacam1CoexpressionGroup )[["FALSE"]] /
    length(tspan8CoexpressionGroup)
```

```
## [1] 0.9345638
```

```
length( union( tspan8CoexpressionGroup, ceacam1CoexpressionGroup ) )
```

```
## [1] 623
```

```
ceacam1Aire <-
    ceacam1CoexpressionGroup[ceacam1CoexpressionGroup %in% aireDependent]

mat <- rbind(
    table( geneClusters[[whichCluster]] %in% ceacam1Aire ),
    table( unlist(geneClusters[!seq_len(nClusters) %in% whichCluster] )
      %in% ceacam1Aire ))[,2:1]
fisher.test(mat, alternative="greater")
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  mat
```

```
## p-value = 0.03684
## alternative hypothesis: true odds ratio is greater than 1
## 95 percent confidence interval:
##  1.136181      Inf
## sample estimates:
## odds ratio
##   4.598405
```

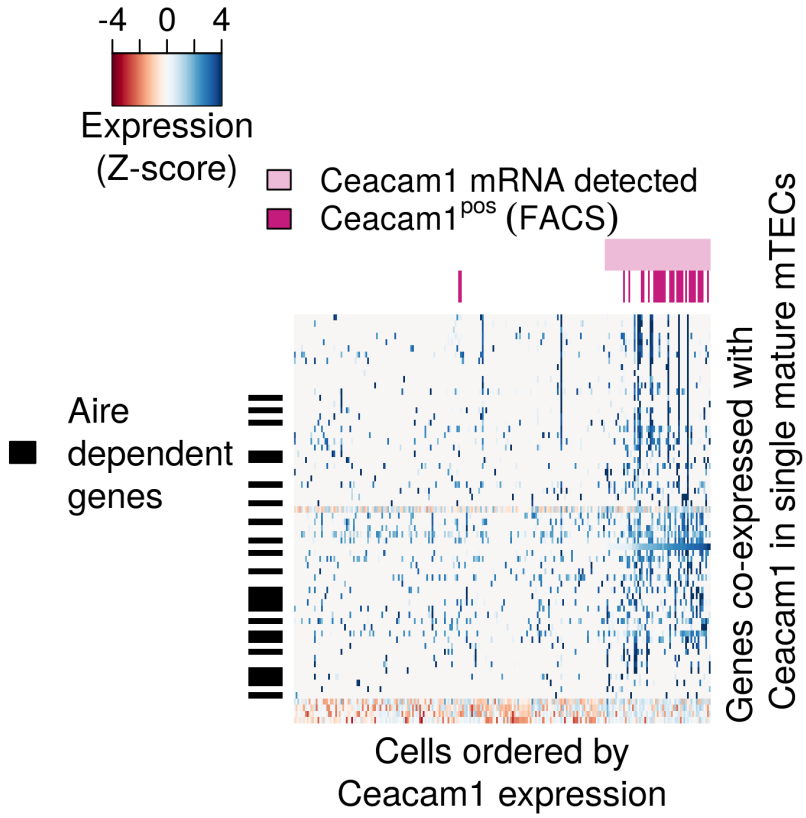
```
dfPrint <- data.frame(
    `ensembl_gene_name`=ceacam1CoexpressionGroup,
    `gene_name`=geneNames[ceacam1CoexpressionGroup],
    `aire_dependent`=as.numeric(ceacam1CoexpressionGroup %in% aireDependent),
    `cluster_2`=as.numeric( ceacam1CoexpressionGroup %in% geneClusters[[whichCluster]]) )


write.table(dfPrint, quote=FALSE, sep="\t", row.names=FALSE, file="figure/ceacam1Coexpressi
```

We also observe a very good agreement of co-expression between the ad hoc *Ceacam1* positive cells and the FACS Ceacam1 positive cells with respect to the unselected Ceacam1 negative cells. Reflecting the validation of our second co-expression group and confirming that the co-expression phenomena is not restricted to Aire-dependent genes.

```
foldChangesTRApos[["Ceacam1"]] <-
    getFoldChangesForViolin( gene, ceacam1CoexpressionGroup)


foldChangesAll[["Ceacam1"]] <-
    getFoldChangesForScatter(gene, ceacam1CoexpressionGroup)


havePosFoldChange <- table( foldChangesTRApos[["Ceacam1"]][["coexpressed"]] > 0 )

cat(sprintf("%s percent (%s out of %s) of the up-regulated
genes based on the unselected mTECs have a consistent fold
change in the Ceacam1+ cells\n",
   round( havePosFoldChange["TRUE"] /
         (sum(havePosFoldChange)), 2 ) * 100,
   havePosFoldChange["TRUE"], (sum(havePosFoldChange))))
```

```
## 92 percent (61 out of 66) of the up-regulated
## genes based on the unselected mTECs have a consistent fold
## change in the Ceacam1+ cells
```

In a similar manner, we plot the heatmap as for the *Tspan8* co-expression group, but using the respective *Ceacam1* FACS data (Plot XVI).



Plot XVI: **Heatmap of expression of genes detected as being co-expressed with Ceacam1.** Corresponds to Figure 3D

```
colCeacamPos <- "#c51b7d"
colCeacamRNA <- "#edbad8"

makeHeatmapForGene(gene, ceacam1CoexpressionGroup,
                   colCeacamPos, colCeacamRNA,
                   legendLabs=c("Ceacam1 mRNA detected",
                       expression(Ceacam1^"pos"~(FACS))))
```

We perform the same validation, now with the TRA Klk5. Plot XVII.

7.3 Klk5

```
gene <- names( geneNames[geneNames %in% "Klk5"] )
names(gene) <- "Klk5"
cellGroups <- counts(dxdUnselected, normalized=TRUE)[gene,] > 0
#dxdUnselected2 <- estimateSizeFactors(dxdUnselected[deGenesNone,])

cat( sprintf("The number of unselected mTECs detecting the expression
```

```
of Klk5 is %s\n", table(cellGroups)["TRUE"] ))



klk5CoexpressionGroup <- getCoexpressionGroup(gene)

wCluster <- grep( names( geneNames[geneNames %in% "Klk5"] ), geneClusters)
wCluster


matKlk <- rbind(
  table( geneClusters[[wCluster]] %in% klk5CoexpressionGroup ),
  table( unlist(geneClusters[-wCluster]) %in% klk5CoexpressionGroup ) )[,2:1]


matKlk




fisher.test( matKlk )
```

```
foldChangesTRApos[["Klk5"]] <-
    getFoldChangesForViolin( gene, klk5CoexpressionGroup)


foldChangesAll[["Klk5"]] <-
    getFoldChangesForScatter(gene, klk5CoexpressionGroup)


havePosFoldChange <- table( foldChangesTRApos[["Klk5"]][["coexpressed"]] > 0 )

table( klk5CoexpressionGroup %in% aireDependent )
```

```
##
## FALSE  TRUE
##    29    40
```

```
cat(sprintf("%s percent (%s out of %s) of the up-regulated
genes based on the unselected mTECs have a consistent fold
change in the Klk5+ cells\n",
   round( havePosFoldChange["TRUE"] /
         (sum(havePosFoldChange)), 2 ) * 100,
   havePosFoldChange["TRUE"], (sum(havePosFoldChange))))
```

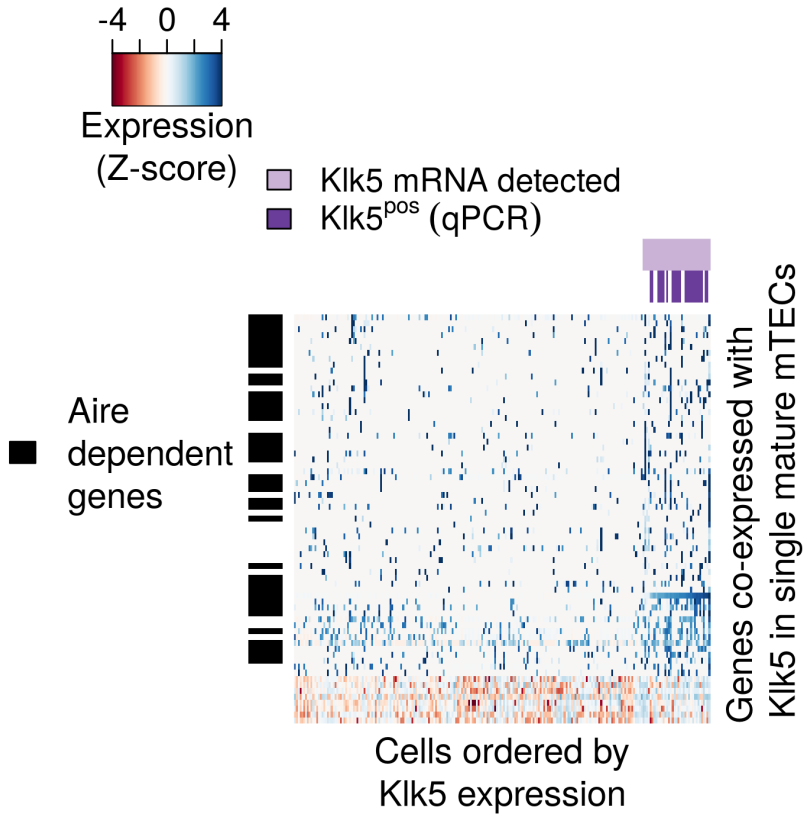
```
## 71 percent (49 out of 69) of the up-regulated
## genes based on the unselected mTECs have a consistent fold
## change in the Klk5+ cells
```

```
colKlkRNA <- "#cab2d6"
colKlkPos <- "#6a3d9a"

makeHeatmapForGene(gene, klk5CoexpressionGroup,
                   colKlkPos, colKlkRNA,
                   legendLabs=c("Klk5 mRNA detected",
                       expression(Klk5^"pos"~(qPCR))))
dfPrint <- data.frame(
    `ensembl_gene_name`=klk5CoexpressionGroup,
    `gene_name`=geneNames[klk5CoexpressionGroup],
    `aire_dependent`=as.numeric(klk5CoexpressionGroup %in% aireDependent),
    `cluster_4`=as.numeric( klk5CoexpressionGroup %in% geneClusters[[whichCluster]]) )

write.table(dfPrint, quote=FALSE, sep="\t", row.names=FALSE, file="figure/klk5Coexpression
```

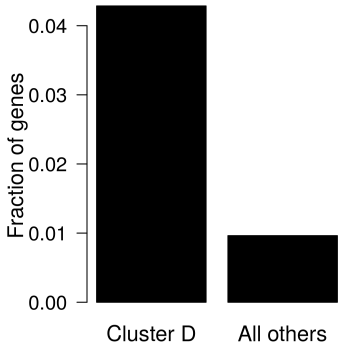
```
par(mar=c(4, 6, 1, 1))
barplot(
    mat[,1]/rowSums(mat),
    names.arg=c("Cluster D", "All others"),
    las=1, col="black",
    cex.axis=1.2, cex.lab=1.3, cex=1.3,
    ylab="Fraction of genes")
```

Distribution of fold changes significantly skewed towards positive values for the three validation experiments, confirming the co-expression phenomena (**??**).

```
dfValidations <-
    data.frame(
    marker= rep( names(foldChangesTRApos),
        sapply(foldChangesTRApos, function(x) length(unlist(x)))),
    gene =
        unlist( lapply(foldChangesTRApos, function(x){
```



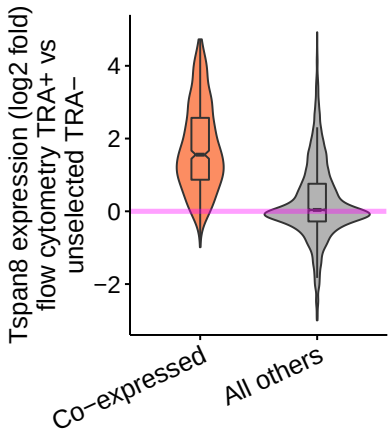
Plot XVII: **Heatmap of expression of genes detected as being co-expressed with Klk5.** Corresponds to 3F



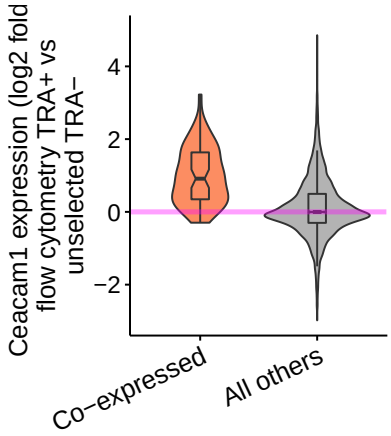
Plot XVIII: **Enrichment of Klk5 co-expressed gene set of genes in cluster D.** Corresponds to S5

```
            rep(names(x), listLen(x))
        }) ),
    foldChange=unlist(foldChangesTRApos)
)
```

```
dfValidations$gene <- relevel(dfValidations$gene, 2)
dfValidations$marker <- relevel( dfValidations$marker, 3)

levels( dfValidations$gene ) <- c("Co-expressed", "All others")

#print( ggplot( dfValidations, aes(x=gene, y=foldChange, fill=gene)) +
#    geom_violin() +
#    geom_boxplot(width=0.15, outlier.shape=NA, notch=.2) +
#    facet_grid(~marker) + scale_y_continuous(limits = c(-3, 5)) +
#    geom_abline(intercept = 0, slope = 0, col="#ff00ff60", lwd=1.3) +
#    theme(legend.position="top", legend.title=element_blank(),
#          strip.text.x = element_text(size = 12, face="bold"),
#          axis.ticks.x = element_blank(), axis.text.x = element_blank(),
#          axis.ticks.y = element_line(colour="black"),
#          legend.text=element_text(size=13),
#          axis.line = element_blank(),
#          legend.key = element_blank(),
#          axis.text.y = element_text(size=12, colour="black"),
#          axis.title=element_text(size=14),
#          panel.background =  element_rect(fill='white', colour='white'),
#          panel.border = element_rect(colour = "black", fill=NA),
#          axis.line = element_line(colour = "black")) +
#ylab("Log fold change (base 2) between
#TRA enriched cells (FACS or qPCR)
#and TRA negative cells (unselected)") + xlab("") +
#    scale_fill_manual( values = c("#fc8d62", "#b3b3b3") ) )

plotViolin <- function(geneName="Tspan8",
           ylab="Tspan8 expression (log2 fold)\nflow cytometry TRA+ vs\nunselected TRA-"){
    dfOp <- dfValidations[dfValidations$marker == geneName,]
    print( ggplot( dfOp, aes(x=gene, y=foldChange, fill=gene)) +
        geom_violin() +
        geom_boxplot(width=0.15, outlier.shape=NA, notch=.2) +
        scale_y_continuous(limits = c(-3, 5)) +
        geom_abline(intercept = 0, slope = 0, col="#ff00ff60", lwd=1.3) +
        theme(legend.position="none",
            legend.title=element_blank(),
            strip.text.x = element_text(size = 12, face="bold"),
            axis.ticks.x = element_line(colour="black"),
            axis.text.x = element_text(size=14, colour="black", angle=25, hjust=1),
            axis.ticks.y = element_line(colour="black"),
            legend.text=element_text(size=13),
            legend.key = element_blank(),
            axis.text.y = element_text(size=13, colour="black"),
```



Plot XIX: **Violin plot depicting distribution of fold changes.** Corresponds to Figure 3A.

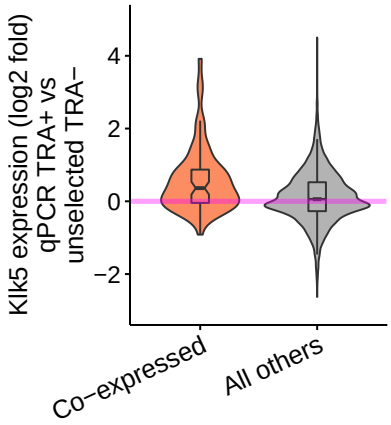


Plot XX: **Violin plot depicting distribution of fold changes.** Corresponds to Figure 3C.

```
            axis.title=element_text(size=14),
            panel.background =  element_blank(),
            panel.grid.major = element_blank(),
            panel.grid.minor = element_blank(),
            panel.border = element_rect(colour = "white", fill=NA),
            axis.line = element_line(colour = "black")) +
  ylab(ylab) + xlab("") +
      scale_fill_manual( values = c("#fc8d62", "#b3b3b3") ) )
}
```

```
plotViolin()
```

```
plotViolin(geneName="Ceacam1",
           ylab="Ceacam1 expression (log2 fold)\nflow cytometry TRA+ vs\nunselected TRA-")
```



Plot XXI: **Violin plot depicting distribution of fold changes.** Corresponds to Figure 3E.

```
plotViolin(geneName="Klk5", ylab="Klk5 expression (log2 fold)\nqPCR TRA+ vs\nunselected TRA
```

The p-values confirming the validation experiments

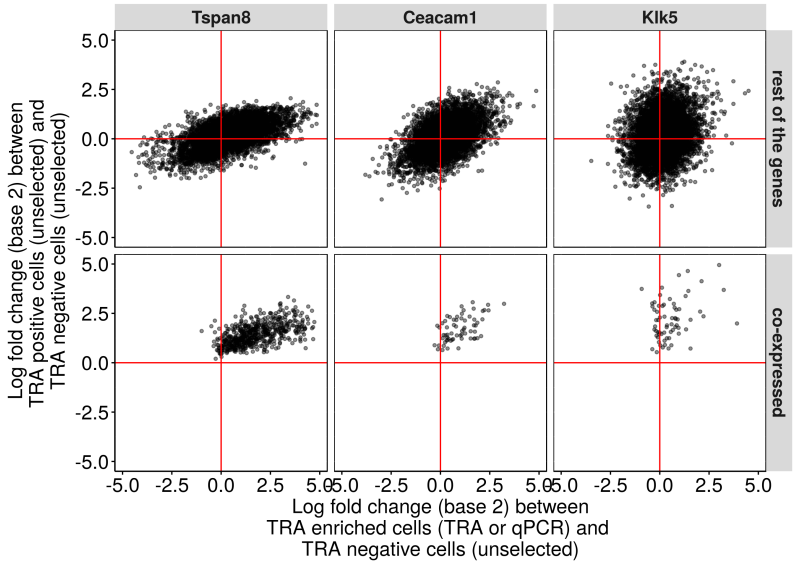
```
sapply(names(foldChangesTRApos), function(x){
    t.test( foldChangesTRApos[[x]][["coexpressed"]],
            foldChangesTRApos[["background"]])
})
```

```
##             Tspan8
## statistic   33.09611
## parameter   595
## p.value     5.056825e-137
## conf.int    Numeric,2
## estimate    1.862926
## null.value  0
## alternative "two.sided"
## method      "One Sample t-test"
## data.name   "foldChangesTRApos[[x]][["coexpressed"]]"
##             Ceacam1
## statistic   7.703735
## parameter   65
## p.value     9.807717e-11
## conf.int    Numeric,2
## estimate    1.090873
## null.value  0
## alternative "two.sided"
## method      "One Sample t-test"
## data.name   "foldChangesTRApos[[x]][["coexpressed"]]"
##             Klk5
```

```
## statistic   4.192224
## parameter   68
## p.value     8.165024e-05
## conf.int    Numeric,2
## estimate    0.7012261
## null.value  0
## alternative "two.sided"
## method      "One Sample t-test"
## data.name   "foldChangesTRApos[[x]][["coexpressed"]]"
```

Scatterplot of fold changes (Plot XXII).

```
coexpressionGroupList <- list(
    tspan8=tspan8CoexpressionGroup,
    ceacam1=ceacam1CoexpressionGroup,
    klk5=klk5CoexpressionGroup )

save(coexpressionGroupList, file="../data/coexpressionGroupList.RData")

foldChangesDf <-
    lapply( foldChangesAll, function(x){ as.data.frame(do.call(cbind, x)) } )
names(foldChangesDf) <- names(foldChangesAll)
for(x in seq_along( foldChangesDf )){
  foldChangesDf[[x]]$marker <- names(foldChangesDf)[x]
}
stopifnot( all( names(coexpressionGroupList) == tolower(names( foldChangesDf))))
for(x in seq_along( foldChangesDf )){
  foldChangesDf[[x]]$marker <- names(foldChangesDf)[x]
  foldChangesDf[[x]]$coexpressed <-
      rownames(foldChangesDf[[x]]) %in% coexpressionGroupList[[x]]
}
foldChangesDf <-do.call(rbind, foldChangesDf)
foldChangesDf$marker <- relevel(factor( foldChangesDf$marker ), 3)
foldChangesDf$coexpressed <- factor( foldChangesDf$coexpressed )
levels( foldChangesDf$coexpressed ) <- c("rest of the genes", "co-expressed")

#png("prueba.png", res=300, width=9, height=9, units="in")
print( ggplot( foldChangesDf,
       aes(x=preenriched, y=unselected)) +
    geom_point(shape=19, size=1, colour="#00000070") +
#    stat_density2d(geom="tile", aes(fill = ..density..), contour = FALSE) +
    facet_grid(coexpressed~marker) +
    theme(legend.position="none", legend.title=element_blank(),
          strip.text = element_text(size = 14, face="bold"),
          axis.ticks.x =  element_line(colour="black"),
```



Plot XXII: **Violin plot depicting distribution of fold changes.** Corresponds to Figure S4.

```
          axis.text.x = element_text(size=16, colour="black"),
          axis.ticks.y = element_line(colour="black"),
          legend.text=element_blank(),
#           legend.key = element_blank(),
          axis.text.y = element_text(size=16, colour="black"),
          axis.title=element_text(size=16),# panel.background = element_blank(),
          panel.background =  element_rect(fill='white', colour='black'),
          panel.border = element_rect(colour = "black", fill=NA),
          axis.line = element_line(colour = "black")) +
    ylim(c(-5, 5)) + xlim(c(-4.9,4.9)) +
xlab("Log fold change (base 2) between
TRA enriched cells (TRA or qPCR) and
TRA negative cells (unselected)") +
ylab("Log fold change (base 2) between
TRA positive cells (unselected) and
TRA negative cells (unselected)") +
    geom_abline(intercept=0, slope=0, col="red") +
    geom_vline(xintercept=0, col="red") +  coord_fixed() )
#dev.off()
```

```
back <- table(isTRA)
lapply(coexpressionGroupList, function(x){
    fore <- table( x %in% rownames(dxd)[isTRA] )
    fisher.test(
        rbind( fore, back - fore)[,2:1])
})
```

```
## $tspan8
##
##  Fisher's Exact Test for Count Data
##
## data:  rbind(fore, back - fore)[, 2:1]
## p-value < 2.2e-16
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  3.873341 5.524287
## sample estimates:
## odds ratio
##   4.631244
##
##
## $ceacam1
##
##  Fisher's Exact Test for Count Data
##
## data:  rbind(fore, back - fore)[, 2:1]
## p-value = 6.932e-05
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  1.826199 5.853750
## sample estimates:
## odds ratio
##   3.339307
##
##
## $klk5
##
##  Fisher's Exact Test for Count Data
##
## data:  rbind(fore, back - fore)[, 2:1]
## p-value = 0.0001297
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  1.726065 5.477802
## sample estimates:
```

```
## odds ratio
##   3.142349
```

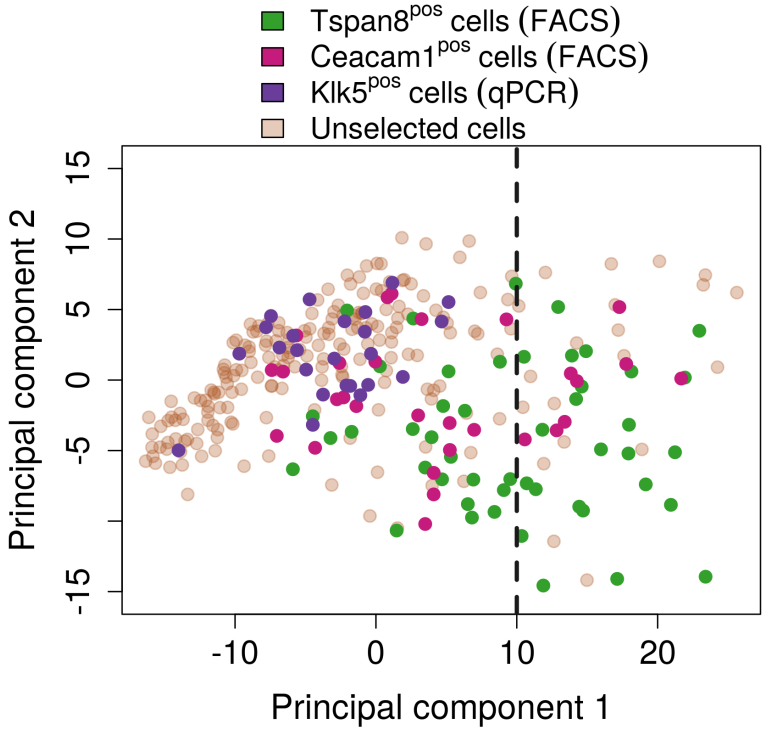
7.4 PCA

In order to explore the hypothesis of sliding windows of co-regulated groups of genes [Supp3], we perform a Principal Component Analysis on the gene expression levels of both the *Ceacam1* and the *Tspan8* co-expression groups.

```
tspan8 <- names( geneNames[geneNames %in% "Tspan8"] )
ceacam1 <- names( geneNames[geneNames %in% "Ceacam1"] )

genesForPca <- union( tspan8CoexpressionGroup, ceacam1CoexpressionGroup )

cellsForPca <- names( which( colSums( counts( dxd,
                        normalized=TRUE )[c(tspan8, ceacam1),] ) >= 0 ) )
length( cellsForPca )
```

```
## [1] 305
```

```
dataForPca <- Ycorr[rownames(Ycorr) %in% genesForPca,cellsForPca]

pr <- prcomp(t(dataForPca))

colUnselected <- "#b1592850"
colors <- c(colUnselected, colCeacamPos, colTspanPos, colKlkPos)
names(colors) <- c("None", "Ceacam1", "Tspan8", "Klk5")
colsForPca <-
    colors[as.character(colData(dxd)[rownames(pr$x),"SurfaceMarker"])]

spCellNames <-
    split( colnames(dataForPca),
          colData(dxd)[cellsForPca,"SurfaceMarker"])

par(mar=c(5, 4.8, 7, 1), xpd=FALSE)
plot( pr$x[spCellNames[["None"]],"PC1"], pr$x[spCellNames[["None"]],"PC2"],
     col=colors["None"],
     pch=19, cex=1.2, ylim=c(-10, 10), asp=1,
     cex.axis=1.45, cex.lab=1.5,
     xlab="Principal component 1",
     ylab="Principal component 2")
points( pr$x[spCellNames[["Tspan8"]],"PC1"],
       pr$x[spCellNames[["Tspan8"]],"PC2"],
       col=colors["Tspan8"], pch=19, cex=1.2)
points( pr$x[spCellNames[["Ceacam1"]],"PC1"],
```



Plot XXIII: **Principal component analysis of cells using the union of the Tspan8 and the Ceacam1 co-expression groups.** Corresponds to Figure 4A.

```
       pr$x[spCellNames[["Ceacam1"]],"PC2"],
       col=colors["Ceacam1"], pch=19, cex=1.2)
points( pr$x[spCellNames[["Klk5"]],"PC1"],
       pr$x[spCellNames[["Klk5"]],"PC2"],
       col=colors["Klk5"], pch=19, cex=1.2)
abline(v=10, lwd=3, col="#1a1a1a", lty="dashed")
legend( x=-10, y=28,
    legend=c(expression(Tspan8^"pos"~cells~(FACS)),
        expression(Ceacam1^"pos"~cells~(FACS)),
        expression(Klk5^"pos"~cells~(qPCR)),
        "Unselected cells"),
    fill=c(colTspanPos, colCeacamPos, colKlkPos, colUnselected),
       xpd=TRUE, cex=1.3,
       horiz=FALSE, bty="n")
```

We observe that the Principal Component 1 is correlated with the transcript levels of *Tspan8* and *Ceacam1*, meaning that most of the variation is explained by the co-expression phenomena.

```
cor( as.vector(as.matrix(dataForPca[tspan8,cellsForPca])),
    pr$x[,"PC1"], method="spearman")
```

```
## [1] 0.6894281
```

```
cor( as.vector(as.matrix(dataForPca[ceacam1,cellsForPca])),
    pr$x[,"PC1"], method="spearman")
```

```
## [1] 0.3984098
```

```
klk5 <- names( geneNames[geneNames %in% "Klk5"] )
```

However, the Tspan8^{pos} cells tend to be more separated from the unselected cells with respect to the Ceacam1^{pos}. For example, considering a theshold of 10 on PC1, this is the fraction of the cells of each group that were above this threshold:

```
more10InPC1 <- sapply(
    split( pr$x[,"PC1"] > 10,
    as.character(colData(dxd)[rownames(pr$x),"SurfaceMarker"] )),
    table)
if(is.na(more10InPC1[["Klk5"]]["TRUE"])){
    more10InPC1[["Klk5"]]["TRUE"] <- 0
}

more10InPC1 <- do.call(cbind, more10InPC1)

round( more10InPC1["TRUE",] / colSums(more10InPC1), 2)
```

```
## Ceacam1    Klk5    None  Tspan8
##     0.3     0.0     0.1     0.5
```

```
plotPCAHeatmap <- function(whichCells=colData(dxd)$SurfaceMarker!="None",
                           showMarkers=FALSE){
  pr <- prcomp(t(dataForPca))
  expr <- asinh( counts(dxd, normalized=TRUE)  )
  expr <- ( expr - rowMedians(expr) )/ rowSds(expr)
  expr <- expr[rownames(dataForPca),whichCells]
  colors["None"] <- "white"
  colors["Tspan8"] <- colTspanRNA
  colors["Ceacam1"] <- colCeacamRNA
  colMiddle <- "#fdbf6f"
  colRows <-
      ifelse( rownames(expr) %in% tspan8CoexpressionGroup,
             colors["Tspan8"], "black" )
  colRows <-
      ifelse( rownames(expr) %in% ceacam1CoexpressionGroup,
             colors["Ceacam1"], colRows )
  colRows <-
      ifelse( rownames(expr) %in%
             intersect( tspan8CoexpressionGroup, ceacam1CoexpressionGroup ),
```

```
            colMiddle, colRows)
colRows <- matrix(colRows, nrow=1)
colors["None"] <- "white"
colors["Tspan8"] <- colTspanPos
colors["Ceacam1"] <- colCeacamPos
br <- seq(-4, 4, length.out=101)
cols <-
    colorRampPalette( brewer.pal(9, "RdBu"),
                      interpolate="spline", space="Lab")(100)
colCols1 <-
    colors[as.character(colData(dxd)[colnames(expr),"SurfaceMarker"])]
colCols2 <- rep("white", ncol(expr))
nonZeros <-
    counts(dxd)[ names( geneNames[geneNames %in% "Tspan8"] ),
                 colnames(expr)] > 0
rankOrder <-
    order( expr[names( geneNames[geneNames %in% "Tspan8"] ),
                 nonZeros] )
colCols2[nonZeros][rankOrder] <-
  colorRampPalette( c("white", colors["Tspan8"]))(sum(nonZeros))
colCols3 <- rep("white", ncol(expr))
nonZeros <-
    counts(dxd)[ names( geneNames[geneNames %in% "Ceacam1"] ),
                 colnames(expr)] > 0
rankOrder <- order( expr[names( geneNames[geneNames %in% "Ceacam1"] ),
                         nonZeros] )
colCols3[nonZeros][rankOrder] <-
  colorRampPalette( c("white", colors["Ceacam1"]))(sum(nonZeros))
if( showMarkers ){
   colCols <- cbind( colCols1, colCols2, colCols3)
   colnames(colCols) <-
    c("Surface Marker", "Tspan8 expression", "Ceacam1 expression")
}else{
   colCols <- cbind( colCols2, colCols3)
   colnames(colCols) <-
    c("Tspan8 expression", "Ceacam1 expression")
}
pr$x <- pr$x[colnames(expr),]
par(xpd=TRUE)
heatmap.3(
        expr[rev(order(colRows[1,])),order( pr$x[,"PC1"] )],
        trace="none", col=cols,
        dendrogram="none",
        labCol=rep("", ncol(expr)),
```

```
          labRow=rep("", nrow(expr) ),
          ColSideColors=colCols[order(pr$x[,"PC1"]),],
          RowSideColors=colRows[,rev(order( colRows[1,]) ),
              drop=FALSE],
          Rowv=FALSE,
          Colv=FALSE,
          NumColSideColors=2.5,
          breaks=br,
          margins = c(4,4),
          KeyValueName="Expression\n(Z-scores)",
          keysize=1.7,
          NumRowSideColors=1.1,
          xlab="Cells ordered by\nprincipal component 1",
          ylab="Co-expressed genes in mature mTECs\n")
  if( showMarkers ){
     legend( x=.45, y=1.1,
          legend=c(expression(Tspan8^"pos"~(FACS)),
              expression(Ceacam1^"pos"~(FACS)),
              expression(Klk5^"pos"~(qPCR))),
          fill=c(colTspanPos, colCeacamPos, colKlkPos), cex=1.2,
          bty="n")
  }
  legend( x=-.1, y=.4,
       title="co-expressed\nwith:",
       legend=c("Tspan8", "Ceacam1", "both"),
       fill=c(colTspanRNA, colCeacamRNA, colMiddle), cex=1.2,
       bty="n")
}
```

Moreover, when focusing only on the cells that were selected for the protein expression of Tspan8 or Ceacam1, the *Tspan8* transcript levels show a strong correlation with both *Ceacam1* and *Tspan8* co-expression groups.

```
expr <- asinh( counts(dxd, normalized=TRUE)  )
expr <- ( expr - rowMedians(expr) )/ rowSds(expr)
expr <-
    expr[rownames(dataForPca),
         !colData(dxd)$SurfaceMarker %in% c( "None", "Klk5")]

factorForSplit <- rep("", nrow(expr) )
factorForSplit[rownames(expr) %in% tspan8CoexpressionGroup] <-
    "Tspan8"
factorForSplit[rownames(expr) %in% ceacam1CoexpressionGroup] <-
    "Ceacam1"
factorForSplit[rownames(expr) %in%
```

```
                intersect( tspan8CoexpressionGroup,
                           ceacam1CoexpressionGroup)] <-
    "Both"

meanExpressionPerGroup <- lapply(
    split( as.data.frame(expr), factorForSplit),
       function(x){
           colMeans(x)
       })


sapply( c(`ceacam1`=ceacam1, tspan8=`tspan8`), function(y){
  sapply( meanExpressionPerGroup, function(x){
      nonZeros <- rep(TRUE, ncol(expr))
#      nonZeros <- expr[y,] != 0
      cor( expr[y,nonZeros], x[nonZeros], method="spearman")
  })
})
```

```
##              ceacam1    tspan8
## Both     0.21303761 0.5012175
## Ceacam1 -0.05508177 0.4732642
## Tspan8  -0.01708151 0.5995941
```

```
sapply( c( `ceacam1`=ceacam1, `tspan8`=tspan8), function(y){
      nonZeros <- rep(TRUE, ncol(expr))
#      nonZeros <- expr[y,] != 0
      cor( colMeans(expr)[nonZeros], expr[y,nonZeros], method="spearman")
})
```

```
##     ceacam1      tspan8
## 0.001527197 0.618327054
```

Now, we calculate the same correlations only in the Ceacam1^{pos} cells, where we observe a positive correlation between the co-expressed genes and the transcript levels of *Tspan8*.

```
expr <- asinh( counts(dxd, normalized=TRUE)  )
expr <- ( expr - rowMedians(expr) )/ rowSds(expr)

expr <- expr[rownames(dataForPca),colData(dxd)$SurfaceMarker=="Ceacam1"]

factorForSplit <- rep("", nrow(expr) )
factorForSplit[rownames(expr) %in% tspan8CoexpressionGroup] <-
    "Tspan8"
factorForSplit[rownames(expr) %in% ceacam1CoexpressionGroup] <-
    "Ceacam1"
factorForSplit[rownames(expr) %in%
```

```
                intersect( ceacam1CoexpressionGroup,
                           tspan8CoexpressionGroup)] <-
    "Both"
meanExpressionPerGroup <- lapply(
    split( as.data.frame(expr), factorForSplit),
       function(x){
           colMeans(x)
       })

sapply( c(`ceacam1`=ceacam1, tspan8=`tspan8`), function(y){
  sapply( meanExpressionPerGroup, function(x){
      nonZeros <- rep(TRUE, ncol(expr))
#       nonZeros <- expr[y,] != 0
      cor( expr[y,nonZeros], x[nonZeros], method="spearman")
  })
})
```

```
##           ceacam1    tspan8
## Both    0.1953499 0.5746977
## Ceacam1 0.1361664 0.5295208
## Tspan8  0.2322839 0.3136228
```

```
sapply( c( `ceacam1`=ceacam1, `tspan8`=tspan8), function(y){
      nonZeros <- rep(TRUE, ncol(expr))
#       nonZeros <- expr[tspan8,] != 0
      cor( colMeans(expr)[nonZeros], expr[y,nonZeros], method="spearman")
})
```

```
##   ceacam1    tspan8
## 0.2258316 0.3450088
```

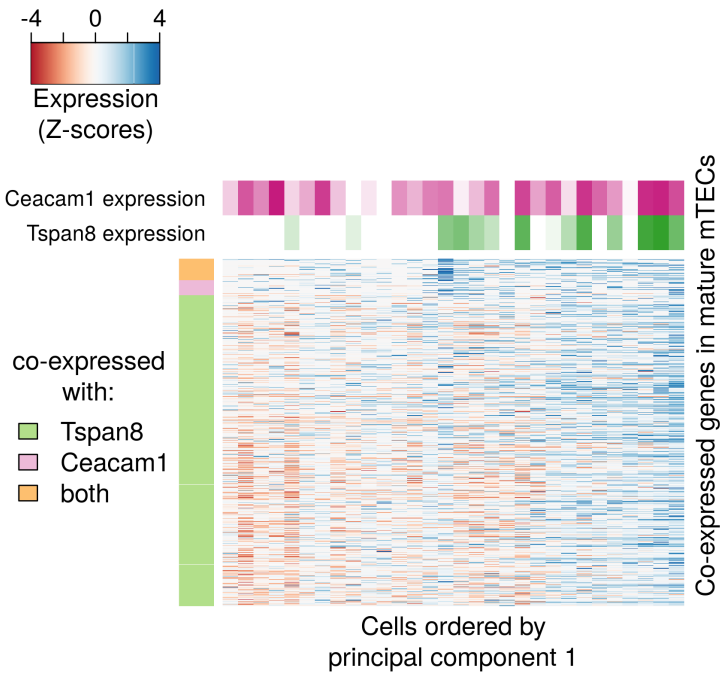
We plot the Ceacam1^{pos} cells (Plot XXIV). Where we see an increase of *Tspan8* levels in the Ceacam1^{pos} cells that is accompanied with an increase in the expression of the co-expressed genes and increasing similarity to the Tspan8^{pos} cells.

```
plotPCAHeatmap(colData(dxd)$SurfaceMarker=="Ceacam1")
```

8 Co-expressed genes tend to be clustered in the genome

As data preparation, we first convert the ensembl gene identifiers to gene names from our 11 groups of co-regulated genes resulting from the k-medoids clustering.

```
geneClusterNames <- sapply( geneClusters, function(x){
  geneNames[names( geneNames ) %in% x]
})
```



Plot XXIV: **Heatmap of co-expressed genes only from the Ceacam1^{pos} cells.** Corresponds to Figure 4B

In order to explore the localization of co-expressed genes in the genome. We load the transcript model annotation and estimate the genomic ranges for each gene.

```
#library(GenomicFeatures)
#transcriptDb <- loadDb( system.file("extdata",
#             "Mus_musculus.GRCm38.75.primary_assembly.sqlite",
#             package="Single.mTec.Transcriptomes") )
#exonsByGene <- exonsBy( transcriptDb, "gene" )
#save(geneRanges, file="../data/geneRanges.RData")
#geneRanges <- unlist( range( exonsByGene ) )
#geneRanges <- keepSeqlevels(geneRanges, paste0( c(1:19, "X")))

data("geneRanges")
dn <-
    geneRanges[names(geneRanges) %in%
                names( which( biotype == "protein_coding" ) ),]
```

The code below was used to generate the null models of the expected genomic distance between genes when selecting a random set of genes (of the same size as the group of genes to test) for 1,000 permutations. For computational running times, the code below was run and its output was saved into the *R* object "permutationResults.RData".

```
dnAire <- dn[names(dn) %in% aireDependentSansom]
dnTested <- dn[names(dn) %in% deGenesNone]
```

```
permutationsForCluster <- function( groupSize, numPerm=1000){
   distancePermuted <- mclapply( seq_len(numPerm), function(x){
       sampleGenes <- sample(seq_len(length(dn)), groupSize )
       median(
         distanceToNearest( dn[sampleGenes] )@elementMetadata$distance,
         na.rm=TRUE)
   }, mc.cores=10)
   unlist( distancePermuted )
}

numberOfPermutations <- 1000

permsAllClusters <- lapply( seq_len(length(geneClusters)), function(i){
  permutationsForCluster( length(geneClusters[[i]]), numberOfPermutations )
})

realAllClusters <- sapply(seq_len(length(geneClusters)), function(i){
   median( distanceToNearest(
       dn[names(dn) %in% geneClusters[[i]],])@elementMetadata$distance,
          na.rm=TRUE )
})

names( permsAllClusters ) <- LETTERS[seq_len(length(permsAllClusters))]
names( realAllClusters ) <- LETTERS[seq_len(length(permsAllClusters))]

permsAllClusters <- permsAllClusters[!names(permsAllClusters) %in% "L"]
realAllClusters <- realAllClusters[!names(realAllClusters) %in% "L"]

save(permsAllClusters, realAllClusters,
     file="../data/permutationResults.RData")
```

Based on this permutations, we can estimate the p-values for each of the 11 groups of genes, testing the hypothesis if the median distance between genes tend to be closer in genomic distance than the expected median distributions (obtained from the null models).

```
names(colClusters) <- LETTERS[1:12]

data("permutationResults")
numberOfPermutations <- 1000

pvals <- sapply( names(permsAllClusters), function(x){
    sum( realAllClusters[x] > permsAllClusters[[x]] )
}) / numberOfPermutations
```

```
p.adjust(pvals, method="BH")
```

```
##      A      B      C      D      E      F      G      H      I      J      K
## 0.2618 0.0660 0.0110 0.0880 0.0605 0.0286 0.0110 0.0110 0.1210 0.0286 0.8950
```

```
sum( p.adjust( pvals, method="BH") < 0.1 )
```

```
## [1] 8
```

```
adjusted <- p.adjust( pvals, method="BH" ) < 0.1
names(permsAllClusters)[adjusted]
```

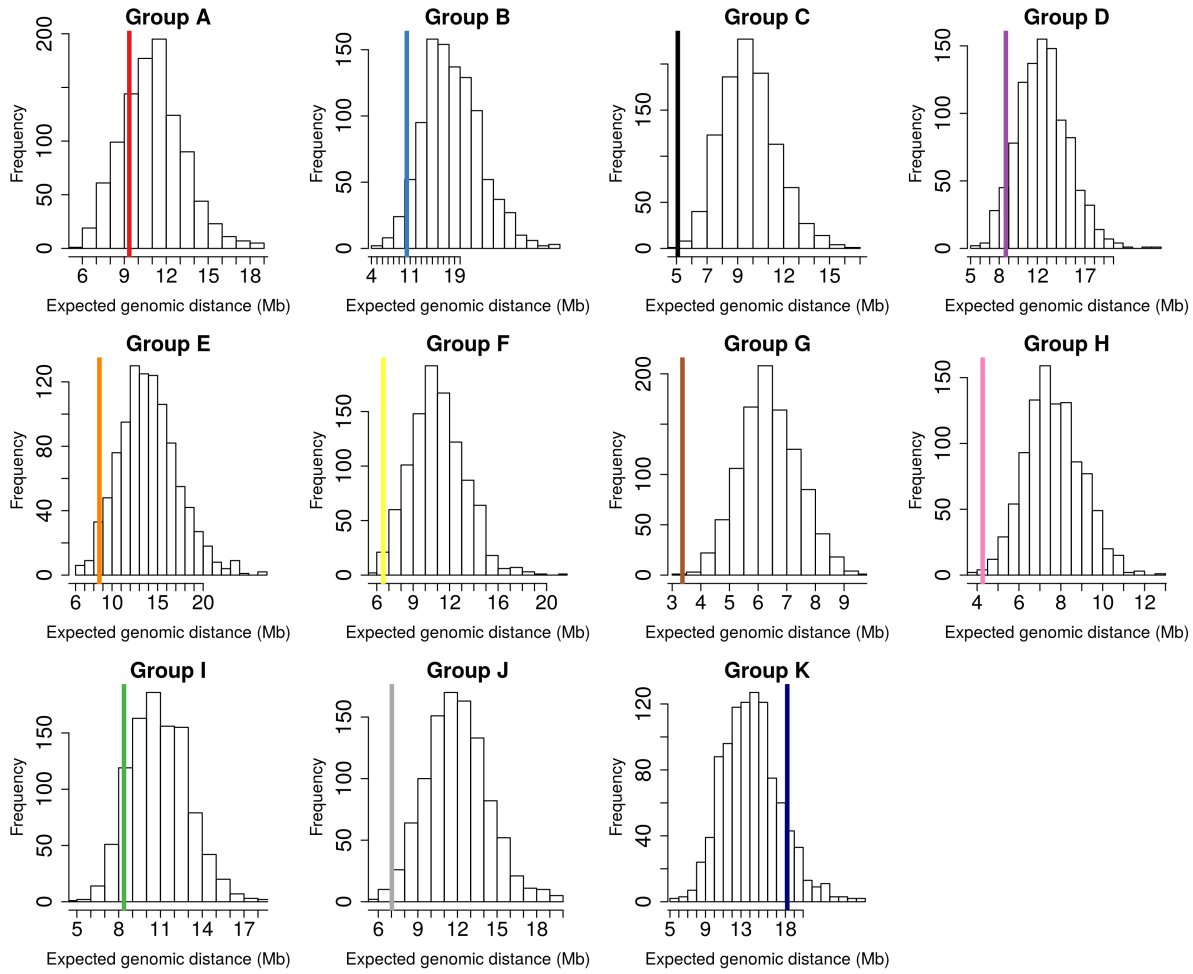
```
## [1] "B" "C" "D" "E" "F" "G" "H" "J"
```

We create a figure based on the expected value distribution and the observed value for each group of co-regulated genes from Plot XI.

```
par(mfrow=c(3, 4), mar=c(5, 5, 2, 2))
for( i in names(permsAllClusters) ){
  coexpressedCoordinates <-
      dn[names(dn) %in% geneClusters[[i]]]
  distancePermuted <- permsAllClusters[[i]]
  xmin <- min(distancePermuted, realAllClusters[i])
  xmax <- max(distancePermuted, realAllClusters[i])
  hist( distancePermuted, 15,
  xlab="Expected genomic distance (Mb)",
       cex.lab=1.4, #main="",
       cex.axis=1.8, xaxt="n",
       xlim=c(xmin, xmax),
       main=paste("Group", i ), cex.main=1.8)
  sq <- seq(0, 20000000, 1000000)
  axis(1, at=sq, label=sq/1000000, cex.axis=1.7)
  md <- realAllClusters[i]
  abline( v=md, col=colClusters[i], lwd=4)
}
```

We also plot karyograms for each of the gene group from Figure XI.

```
names( geneClusters ) <- LETTERS[1:12]

library(ggbio)
for(x in names(permsAllClusters)){
   coexpressedCoordinates <- dn[names(dn) %in% geneClusters[[x]]]
   kr <-
       autoplot(coexpressedCoordinates, layout="karyogram",
              col=colClusters[x], fill=colClusters[x]) +
                  theme(
                      panel.background = element_blank(),
```

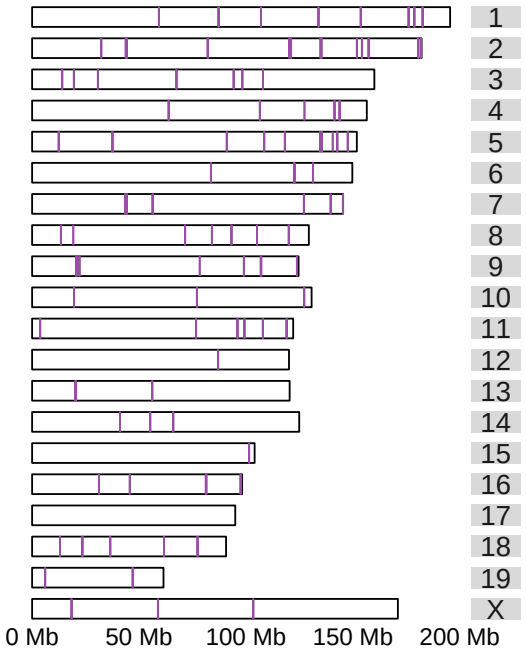


Plot XXV: **expected vs observed distances.** Corresponds to Figure S6

```
                    strip.text.y = element_text(size = 16),
                    axis.text.x = element_text(size=14, colour="black") )
   print(kr)
}
```

We also plot the karyogram for the Group "D", that is used for further examples. The resulting figure is the Plot XXVI.

```
clusterNumber <- LETTERS[wCluster]
 coexpressedCoordinates <-
      geneRanges[names(geneRanges) %in% geneClusters[[clusterNumber]]]
dn2 <- GRanges("7", IRanges(start=18474583, end=18656725))
dn4 <- GRanges("9", IRanges(start=14860210, end=14903949))
library(ggbio)

print( autoplot(coexpressedCoordinates, layout="karyogram",
```

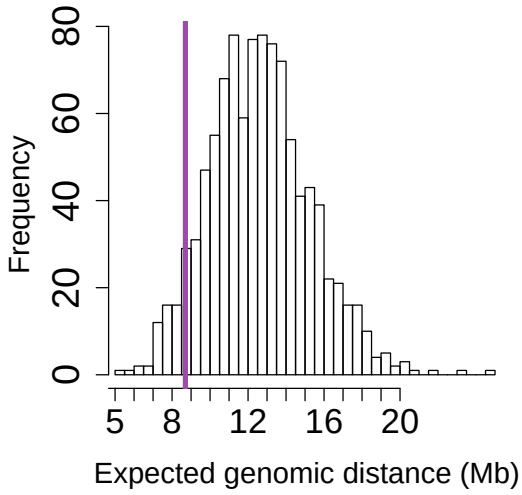


Plot XXVI: **Karyogram of cluster D.** Corresponds to Figure 5a.

```
                col=colClusters[clusterNumber],
                fill=colClusters[clusterNumber]) +
    theme(
        panel.background = element_blank(),
        strip.text.y = element_text(size = 16),
        axis.text.x = element_text(size=14, colour="black")
        ) )
```

For the same Group "D", we plot the expected vs observed genomic distance in Plot XXVII

```
i <- clusterNumber
distancePermuted <- permsAllClusters[[i]]
xmin <- min(distancePermuted, realAllClusters[i])
xmax <- max(distancePermuted, realAllClusters[i])
par(mar=c(4.2, 4.5, 1, 1))
hist( distancePermuted, 30,
xlab="Expected genomic distance (Mb)",
     cex.lab=1.4, #main="",
     cex.axis=1.8, xaxt="n",
     xlim=c(xmin, xmax),
     main="", cex.main=1.8)
sq <- seq(0, 20000000, 1000000)
axis(1, at=sq, label=sq/1000000, cex.axis=1.7)
md <- realAllClusters[i]
```

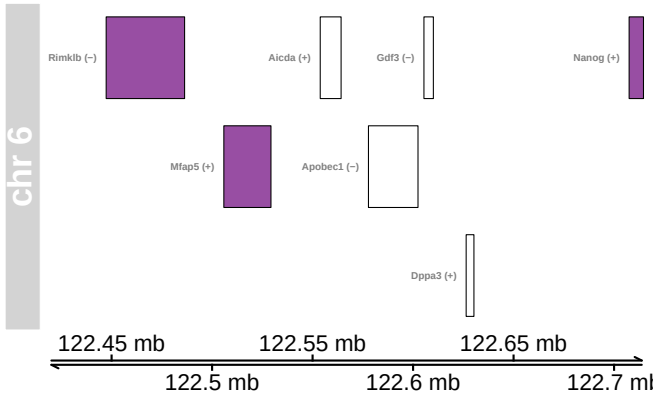


Plot XXVII: **Expected vs observed genomic distance for gene group D.** Corresponds to Figure 5b.

```
abline( v=md, col=colClusters[i], lwd=4)
```

Now, we plot the region of the genomic region of unrelated genes, as an example of neighboring genes different families of genes (Plot XXVIII).

```
library(Gviz)

colVector <- rep(colClusters, sapply(geneClusters, length) )
names(colVector) <- unlist(geneClusters)
range <- dn2
data("geneNames")

length(unlist(geneClusters[1:11])) / length(unlist(geneClusters))

## [1] 0.6818817

sum( unlist(geneClusters[1:11]) %in% tras )/
    sum( unlist(geneClusters) %in% tras )

## [1] 0.7148241

makePlotForRegion <- function(range, left=100000,
                              right=100000, colClusterNumber=NULL){
   start(range) <- start( range ) - left
   end(range) <- end( range ) + right
   geneRanges <- geneRanges[names( geneRanges ) %in% proteinCoding]
   overlap <- findOverlaps( range, geneRanges, ignore.strand=TRUE )
   toPlot <- geneRanges[names(geneRanges[subjectHits(overlap)])]
   toPlot <- keepSeqlevels( toPlot, as.character(1:19))
   ch <- unique( as.character(seqnames(toPlot)) )
```



Plot XXVIII: **Unrelated genes.** Supplementary 8C.

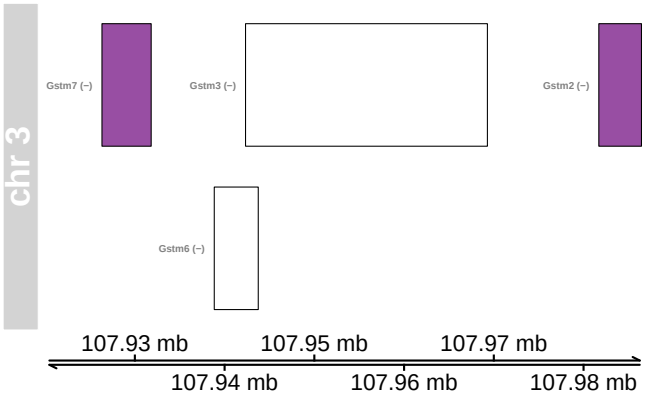
```
    if( !is.null(colClusterNumber)){
        cols <- ifelse( names(toPlot) %in% geneClusters[[colClusterNumber]],
           colClusters[[colClusterNumber]], "white")
    }else{
      cols <- colVector[names( toPlot )]
      cols[is.na(cols)] <- "white"
    }
    labels <- sprintf( "%s (%s)", geneNames[names(toPlot)],
                      as.character(strand(toPlot)))
    plotTracks( list(
      GeneRegionTrack( toPlot, symbol=labels, name=paste("chr", ch)),
      GenomeAxisTrack()),
             showId=TRUE, geneSymbol=TRUE,
             fill=cols, fontsize=20,
             fontcolor="black", detailsBorder.col="black",
             col.line="black", col="black",
             sizes=c(5, 1))
}

dn2 <- GRanges("6", IRanges(start=122447296,end=122714633))
makePlotForRegion(dn2, left=1000, right=1500, colClusterNumber=clusterNumber)
```

Example of the *Gmst* gene family (Plot XXIX).

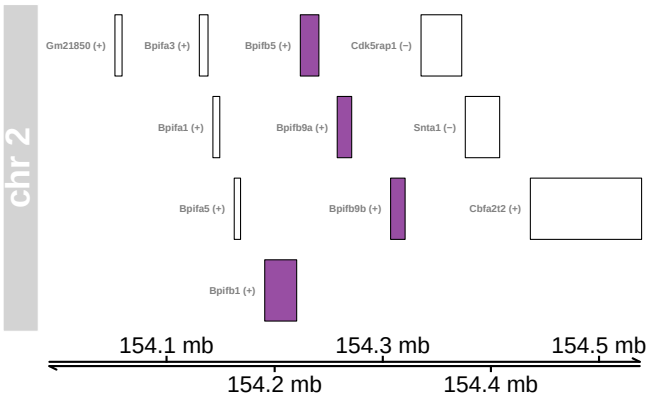
```
dn2 <- GRanges("3", IRanges(start=107923453,end=107999678))
makePlotForRegion(dn2, left=1000, right=1500, colClusterNumber=clusterNumber)
```

The example of the *Bpfib* gene family (Plot XXX).

```
dn2 <- GRanges("2", IRanges(start=154049564,end=154479003))
makePlotForRegion(dn2, left=100, right=100, colClusterNumber=clusterNumber)
```



Plot XXIX: **Other related genes.** Supplementary 8B.



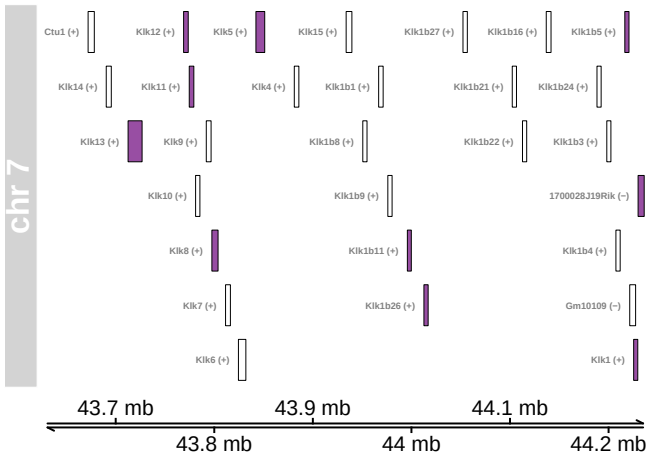
Plot XXX: **Other related genes.** Supplementary 8A.

A high fraction of the genes of the *Klk* loci belong to the same gene group defined in the k-medoids clustering (Plot XXXI).

```
dn5 <- GRanges("7", IRanges(start=43690418,end=44229617))
makePlotForRegion(dn5, left=20000, right=3500, colClusterNumber=clusterNumber)
```

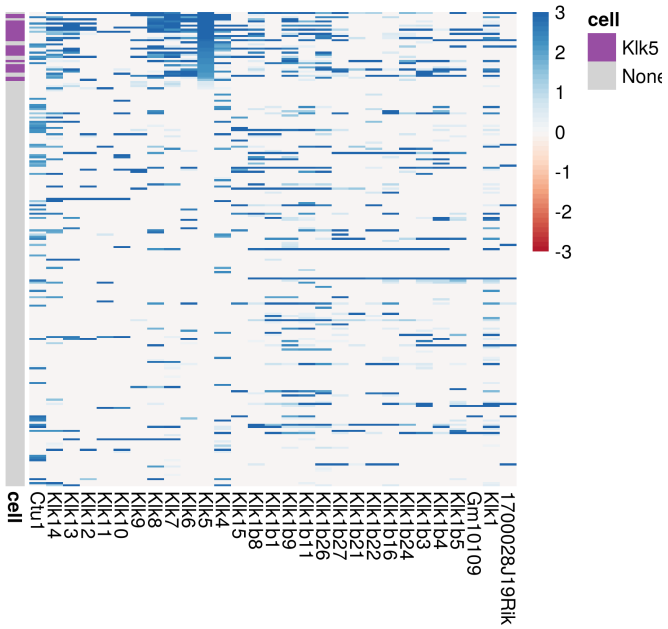
We plot the expression of the Klk loci in a heatmap representation for both the unselected cells and Klk5 cells. The rows are ordered based on Klk5 expression levels. The columns are ordered according to their position on the genome. The expression of Klk5 seems to be a proxy for the expression of the neighboring genes XXXII

```
range <- dn5
start(range) <- start( range ) - 20000
end(range) <- end( range ) + 3500
overlap <- findOverlaps( range, geneRanges, ignore.strand=TRUE )
toPlot <- geneRanges[names(geneRanges[subjectHits(overlap)])]
toPlot <- toPlot[names( toPlot ) %in% proteinCoding,]

coexpressedCoordinates <- geneRanges[names( toPlot ),]
```



Plot XXXI: **Klk gene loci in chromosome 7.** Figure 5C.

```
coexpressedCoordinates <- sort( coexpressedCoordinates, ignore.strand=TRUE )
klkGenes <- names(coexpressedCoordinates)

library(pheatmap)

cellOrder <-
    names( sort(counts(dxd,normalized=TRUE)[gene,
       colData(dxd)$SurfaceMarker %in% c("Klk5", "None")], decreasing=TRUE) )

annotationGenes <-
    data.frame(
       kmeans=1*klkGenes %in% geneClusters[[clusterNumber]],
       wilcox=1*klkGenes %in% klk5CoexpressionGroup,
       aireDependent= 1*(klkGenes %in% aireDependent),
       row.names=klkGenes)


klkMat <- asinh( counts(dxd, normalized=TRUE)[klkGenes,cellOrder])
klkMat <- ( klkMat - rowMedians(klkMat) ) / rowSds(klkMat)
klkMat <- pmin(klkMat, 3)

colsKlk <- colorRampPalette(brewer.pal(9, "Blues"))(50)
colsKlk <- colorRampPalette(brewer.pal(9, "RdBu"))(50)

annotationCells <-
    data.frame(
        droplevels(colData(dxd)[colnames(klkMat),"SurfaceMarker"]))
rownames(annotationCells) <- colnames(klkMat)
colnames(annotationCells) <- "cell"
```

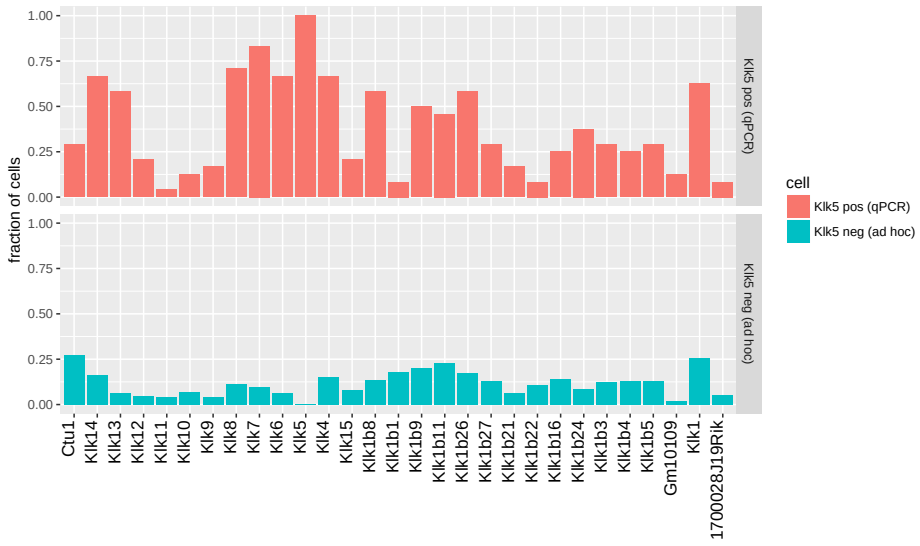


Plot XXXII: **Klk expression.** Figure 5D.

```
ann_colors <-
    list(cell=c(
             Klk5=colClusters[[clusterNumber]],
             None="lightgray") )

pheatmap(( t(klkMat) )*1,
         col=colsKlk,
         cluster_rows=FALSE, cluster_cols=FALSE,
         labels_row="", labels_col=geneNames[klkGenes],
         annotation_row=annotationCells,
         annotation_colors=ann_colors,
         breaks = seq(-3, 3, length.out=51),
         legend=TRUE,
         fontsize = 12)
```

Another representation of the same result, but ploting the fraction of cells that detect the expression of the Klk genes. Stratified both by the qPCR-selected Klk5 positive cells or the Klk5 negative unselected cells (Plot XXXIII).

```
cellGroups <- split( names(cellGroups), cellGroups)
klkMat <- klkMat[,!colnames(klkMat) %in% cellGroups[["TRUE"]]]

cellSplit <-
    split(colnames(klkMat),
      droplevels(colData(dxd)[colnames(klkMat),"SurfaceMarker"]))
```



Plot XXXIII: **Klk expression.** Supplementary S9.

```
klkPercentages <-
 apply( klkMat, 1, function(x){
    c(None=sum( x[cellSplit[["None"]]] > 0 ),
      Klk5=sum( x[cellSplit[["Klk5"]]] > 0 ))
})

klkPercentages["None",] <- klkPercentages["None",] / length(cellSplit[["None"]])
klkPercentages["Klk5",] <- klkPercentages["Klk5",] / length(cellSplit[["Klk5"]])


dfKlkFractions <-
    data.frame(
        fraction=as.vector(klkPercentages),
        cell=rep(rownames(klkPercentages), ncol(klkPercentages)),
        gene=factor( rep(geneNames[colnames(klkPercentages)],
            each=nrow(klkPercentages)),
            levels=geneNames[colnames(klkPercentages)]))

levels(dfKlkFractions$cell) <- c("Klk5 pos (qPCR)", "Klk5 neg (ad hoc)")

print(ggplot( dfKlkFractions, aes( x=gene, y=fraction, fill=cell, stat="bar")) +
    geom_bar(stat="identity") +
    facet_grid(cell~.) +
    ylab("fraction of cells") + xlab("") +
    theme( axis.text.x=element_text(size=12,
                color="black", angle=90, hjust = 1, vjust=0 ) ) )
```

```
dfKlk <-
    as.data.frame( geneRanges[names( geneNames[grepl("Klk", geneNames )] ),] )
dfKlk <- dfKlk[dfKlk$seqnames == 7,]
nms <- sapply( geneClusterNames, function(x){
    sum(grepl("Klk", x))
})
names(nms) <- LETTERS[as.numeric(names(nms))]
nms
```

```
## A B C D E F G H I K L J
## 4 0 0 9 0 1 3 0 2 1 0 4
```

We print the cluster information into a nicely formatted table.

```
clusterInfo <- data.frame(
    `ensemblID`=unlist( geneClusters ),
    `geneNames`=geneNames[unlist( geneClusters )],
    `clusterNumber`=rep(names(geneClusters), sapply(geneClusters, length) ),
    `clusterColor`=rep(colClusters, sapply(geneClusters, length) ) )
rownames( clusterInfo ) <- NULL
write.table( clusterInfo, sep="\t",
             quote=FALSE, row.names=FALSE, col.names=TRUE,
             file="figure/Table_Supp1_CoexpressionGroupsTable.txt" )
```

9 ATAC-seq data

Next, we use ATAC-seq data to measure chromatin accessibility. Because the number of cells needed for the ATAC-seq data protocol were not possible to obtain from Mouse samples, we use Human samples for two co-expression groups defined by a previous study [Supp3].

To reproduce this results, we load the ATAC-seq data for the co-expression groups and load the data objets. We prepared a `DESeqDataSet` object containing the number of ATAC-seq read fragments aligned to the promoter sequences (4Kb window around TSS) of each protein coding gene. Then, based on the microarray data from [Supp3], we define the set of genes that are co-expressed with *CEACAM5* and with *MUC1* (FDR of 10% and fold change of larger than 2). We run DESeq2 to get moderated fold changes between the TRA positive fractions with respect to the TRA negative fractions.

```
data("geneNamesHuman")
data("biotypesHuman")
data(muc1Coexpression)
data(cea1Coexpression)
data(dxdATAC)

dxdCEACAM5 <- dxdATAC[,colData(dxdATAC)$TRA == "CEACAM5"]
colData(dxdCEACAM5) <- droplevels(colData(dxdCEACAM5))
```

```
dxdCEACAM5 <- estimateSizeFactors(dxdCEACAM5)
dxdCEACAM5 <- estimateDispersions(dxdCEACAM5)
dxdCEACAM5 <- nbinomWaldTest(dxdCEACAM5)

CEACAM5Group <-
    as.character(
        cea1Coexpression$SYMBOL[
           cea1Coexpression$`adj.P.Val` < 0.1 &
                                cea1Coexpression$logFC > 2] )
CEACAM5Group <- names( geneNames[geneNames %in% CEACAM5Group] )

dxdMUC1 <- dxdATAC[,colData(dxdATAC)$TRA == "MUC1"]
colData(dxdMUC1) <- droplevels(colData(dxdMUC1))
dxdMUC1 <- estimateSizeFactors(dxdMUC1)
dxdMUC1 <- estimateDispersions(dxdMUC1)
dxdMUC1 <- nbinomWaldTest(dxdMUC1)

MUC1Group <-
    as.character(
        muc1Coexpression$SYMBOL[
           muc1Coexpression$`adj.P.Val` < 0.1 & muc1Coexpression$logFC > 2] )
MUC1Group <- names( geneNames[geneNames %in% MUC1Group] )

length(CEACAM5Group)
```

```
## [1] 288
```

```
length(MUC1Group)
```

```
## [1] 219
```

We observe significant differences in chromatin accessibility between the groups of co-expressed genes compared to the rest of the genes, as assesed by a t-test.

```
t.test(
    results(dxdCEACAM5)$log2FoldChange[rownames(dxdCEACAM5) %in% CEACAM5Group],
    results(dxdCEACAM5)$log2FoldChange, alternative="greater")
```

```
##
##  Welch Two Sample t-test
##
## data:  results(dxdCEACAM5)$log2FoldChange[rownames(dxdCEACAM5) %in%  and results(dxdCEAC
## t = 8.5652, df = 210.73, p-value = 1.142e-15
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  0.3063265       Inf
## sample estimates:
```

```
## mean of x mean of y
## 0.5117886 0.1322546
```

```
t.test(
    results(dxdMUC1)$log2FoldChange[rownames(dxdMUC1) %in% MUC1Group],
    results(dxdMUC1)$log2FoldChange, alternative="greater")
```

```
##
##  Welch Two Sample t-test
##
## data:  results(dxdMUC1)$log2FoldChange[rownames(dxdMUC1) %in% MUC1Group] and results(dx
## t = 8.3749, df = 169.72, p-value = 9.967e-15
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  0.1396944       Inf
## sample estimates:
##  mean of x  mean of y
## 0.18791554 0.01384585
```

We visualize the results in violin plots, showing the differences in chromatin accessibility (Plot XXXIV).

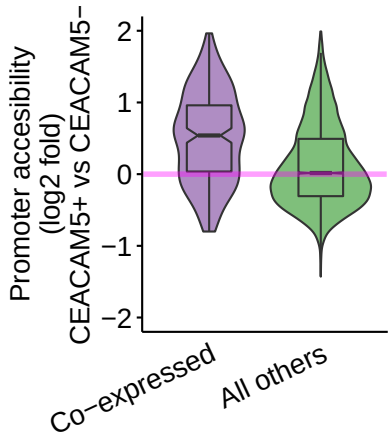
```
dim(results(dxdCEACAM5))
```

```
## [1] 19224     6
```

```
dim(results(dxdMUC1))
```

```
## [1] 19224     6
```

```
df <- data.frame(
    signal = c( results(dxdCEACAM5)$log2FoldChange,
        results(dxdMUC1)$log2FoldChange),
    group = rep(c("CEACAM5", "MUC1"), each=dim(results(dxdMUC1))[1]),
    genes = c(
        ifelse( rownames(dxdCEACAM5) %in% CEACAM5Group,
               "Co-expressed", "All others"),
        ifelse( rownames(dxdMUC1) %in% MUC1Group,
               "Co-expressed", "All others") ))

df$genes <- relevel( df$genes, 2 )

atacViolin <- function(gene, ylab="", ylim=c(-2, 2)){
     dfOp <- df[df$group == gene,]
     p <- ggplot(dfOp, aes(factor(genes), signal, fill=genes))
     p <- p + geom_violin() +
         geom_boxplot(width=.4, notch=.2, outlier.shape=NA) +
#    facet_grid(~group) +
         scale_y_continuous(limits = ylim) +
```

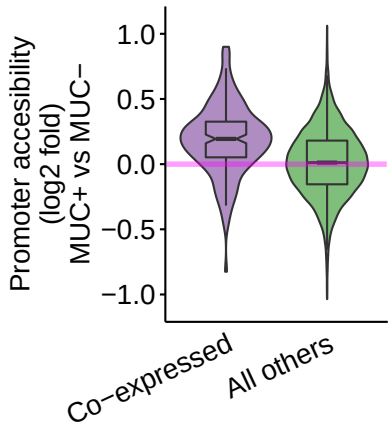


Plot XXXIV: **Chromatin accessibility.** Corresponds to Figure 6a.

```
        geom_abline(intercept = 0, slope = 0, col="#ff00ff60", lwd=1.3) +
        theme(legend.position="none", legend.title=element_blank(),
              strip.text.x = element_text(size = 12, face="bold"),
              axis.ticks.x = element_blank(),
              axis.text.x = element_text(size=14, colour="black", angle=25, hjust=1),
              axis.ticks.y = element_line(colour="black"),
              legend.text=element_text(size=13),
              legend.key = element_blank(),
              axis.text.y = element_text(size=14, colour="black"),
              axis.title=element_text(size=14),
              axis.line=element_line(colour="black"),
              panel.background = element_rect(colour="white", fill="white"),
              panel.border = element_rect(colour = "white", fill=NA),
              panel.grid.major = element_blank(),
              panel.grid.minor = element_blank()) +
        xlab("") +
        ylab(ylab) +
        scale_fill_manual( values = c("#af8dc3", "#7fbf7b") )
    p
}
```

```
#pdf("prueba.pdf", height=3.3, width=2.8)
atacViolin("CEACAM5", "Promoter accesibility\n(log2 fold)\nCEACAM5+ vs CEACAM5-")
#dev.off()
```

```
atacViolin("MUC1",
           "Promoter accesibility\n(log2 fold)\nMUC+ vs MUC-",
           c(-1.1, 1.1))
```



Plot XXXV: **Chromatin accessibility.** Corresponds to Figure 6b.

10 Session Information

This is the information from the *R* session used to generate this document:

```
sessionInfo()

## R version 3.3.2 (2016-10-31)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.1 LTS
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C                 LC_TIME=en_US.UTF-8
##  [4] LC_COLLATE=C               LC_MONETARY=en_US.UTF-8      LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                    LC_ADDRESS=C
## [10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
##  [1] grid      parallel  stats4    stats     graphics  grDevices utils     datasets
##  [9] methods   base
##
## other attached packages:
##  [1] pheatmap_1.0.8                   Gviz_1.18.1
##  [3] ggbio_1.22.3                     matrixStats_0.51.0
##  [5] geneplotter_1.52.0               annotate_1.52.1
##  [7] XML_3.98-1.5                     lattice_0.20-34
##  [9] Single.mTEC.Transcriptomes_1.2.1 gridExtra_2.2.1
## [11] clue_0.3-52                      cluster_2.0.5
## [13] gplots_3.0.1                     ggplot2_2.2.1
## [15] RColorBrewer_1.1-2               gdata_2.17.0
## [17] statmod_1.4.27                   genefilter_1.56.0
```

```
## [19] GenomicFeatures_1.26.2          AnnotationDbi_1.36.0
## [21] DESeq2_1.14.1                   SummarizedExperiment_1.4.0
## [23] Biobase_2.34.0                  GenomicRanges_1.26.2
## [25] GenomeInfoDb_1.10.2             IRanges_2.8.1
## [27] S4Vectors_0.12.1                BiocGenerics_0.20.0
## [29] knitr_1.15.1
##
## loaded via a namespace (and not attached):
##  [1] bitops_1.0-6                 httr_1.2.1
##  [3] tools_3.3.2                  backports_1.0.4
##  [5] R6_2.2.0                     rpart_4.1-10
##  [7] KernSmooth_2.23-15           Hmisc_4.0-2
##  [9] DBI_0.5-1                    lazyeval_0.2.0
## [11] colorspace_1.3-2             nnet_7.3-12
## [13] GGally_1.3.0                 graph_1.52.0
## [15] htmlTable_1.8                rtracklayer_1.34.1
## [17] labeling_0.3                 caTools_1.17.1
## [19] scales_0.4.1                 checkmate_1.8.2
## [21] RBGL_1.50.0                  stringr_1.1.0
## [23] digest_0.6.11                Rsamtools_1.26.1
## [25] foreign_0.8-67               XVector_0.14.0
## [27] base64enc_0.1-3              dichromat_2.0-0
## [29] htmltools_0.3.5              ensembldb_1.6.2
## [31] BSgenome_1.42.0              highr_0.6
## [33] RSQLite_1.1-1                BiocInstaller_1.24.0
## [35] shiny_0.14.2                 BiocParallel_1.8.1
## [37] gtools_3.5.0                 acepack_1.4.1
## [39] VariantAnnotation_1.20.2     RCurl_1.95-4.8
## [41] magrittr_1.5                 Formula_1.2-1
## [43] Matrix_1.2-7.1               Rcpp_0.12.8
## [45] munsell_0.4.3                stringi_1.1.2
## [47] yaml_2.1.14                  zlibbioc_1.20.0
## [49] plyr_1.8.4                   AnnotationHub_2.6.4
## [51] Biostrings_2.42.1            splines_3.3.2
## [53] locfit_1.5-9.1               reshape2_1.4.2
## [55] biomaRt_2.30.0               evaluate_0.10
## [57] biovizBase_1.22.0            latticeExtra_0.6-28
## [59] data.table_1.10.0            httpuv_1.3.3
## [61] gtable_0.2.0                 reshape_0.8.6
## [63] assertthat_0.1               mime_0.5
## [65] xtable_1.8-2                 survival_2.40-1
## [67] tibble_1.2                   OrganismDbi_1.16.0
## [69] GenomicAlignments_1.10.0     memoise_1.0.0
## [71] interactiveDisplayBase_1.12.0 BiocStyle_2.2.1
```

References

[Supp1] S. N. Sansom, N. Shikama-Dorn, S. Zhanybekova, G. Nusspaumer, I. C. Macaulay, M. E. Deadman, A. Heger, C. P. Ponting, and G. A. Hollander. Population and single-cell genomics reveal the Aire dependency, relief from Polycomb silencing, and distribution of self-antigen expression in thymic epithelia. *Genome Res.*, 24(12):1918–1931, Dec 2014.

[Supp2] P. Brennecke, S. Anders, J. K. Kim, A. A. Kołodziejczyk, X. Zhang, V. Proserpio, B. Baying, V. Benes, S. A. Teichmann, J. C. Marioni, and M. G. Heisler. Accounting for technical noise in single-cell RNA-seq experiments. *Nat. Methods*, 10(11):1093–1095, Nov 2013.

[Supp3] S. Pinto, C. Michel, H. Schmidt-Glenewinkel, N. Harder, K. Rohr, S. Wild, B. Brors, and B. Kyewski. Overlapping gene coexpression patterns in human medullary thymic epithelial cells generate self-antigen diversity. *Proc. Natl. Acad. Sci. U.S.A.*, 110(37):E3497–3505, Sep 2013.

[Supp4] A. R. Forrest, H. Kawaji, M. Rehli, J. K. Baillie, M. J. de Hoon, et al. A promoter-level mammalian expression atlas. *Nature*, 507(7493):462–470, Mar 2014.

[Supp5] F. Buettner, K. N. Natarajan, F. P. Casale, V. Proserpio, A. Scialdone, F. J. Theis, S. A. Teichmann, J. C. Marioni, and O. Stegle. Computational analysis of cell-to-cell heterogeneity in single-cell RNA-sequencing data reveals hidden subpopulations of cells. *Nat. Biotechnol.*, Jan 2015.