Executable Analysis Document Supporting:

Cell-to-cell expression variability followed by signal reinforcement progressively segregates early mouse lineages

Yusuke Ohnishi, Wolfgang Huber, Akiko Tsumura, Minjung Kang, Panagiotis Xenopoulos, Kazuki Kurimoto, Andrzej K. Oleś, Marcos J. Araúzo-Bravo, Mitinori Saitou, Anna-Katerina Hadjantonakis and Takashi Hiiragi

Nature Cell Biology 16(1), 27-37 (2014)

doi:10.1038/ncb2881

Andrzej K. Oleś, Wolfgang Huber

October 19, 2026

Contents

1 Data import and preparations . . . 5
1.1 Grouping of samples . . . 6
2 How many genes are expressed? . . . 7
2.1 By variability . . . 7
2.2 By Affymetrix present/absent calls . . . 8
3 Cluster stability analysis . . . 10
3.1 E3.25 and E3.5 WT samples . . . 10
3.2 E3.5/E4.5 WT and E4.5 FGF4-KO samples . . . 11
3.3 E3.25 WT and E3.5 FGF4-KO samples . . . 13
4 Lineage Markers . . . 15
4.1 Clustering of E3.5 WT samples . . . 15
4.2 Differentially expressed genes in E3.5 samples . . . 16
4.2.1 Heatmaps . . . 18
4.3 Differentially expressed genes in E4.5 samples . . . 18
5 Differentially expressed genes from E3.25 to E3.5 . . . 24

6 Principal Component Analysis . . . 26
- 6.1 On the WT samples . . . 26
- 6.2 WT and FGF4-KO samples . . . 29
- 6.3 Heatmap of all WT and FGF4-KO samples . . . 31

7 Further analyses of FGF4-KO . . . 33
- 7.1 FGF4's expression pattern in E3.25 samples . . . 33
- 7.2 Are the E3.25 WT samples with low FGF4 expression more similar to the FGF4-KO samples than those with high FGF4? . . . 34
- 7.3 Variability of the FGF4-KO samples compared to WT samples . . . 35
- 7.4 Do the FGF4-KO samples correspond to a particularly early sub-stage within E3.25 (as indicated by the number of cells)? . . . 38
- 7.5 Heatmap of E3.25 WT and E3.25 FGF-KO samples . . . 38
- 7.6 Differentially expressed genes between FGF4-KO and WT (PE, EPI) at E3.5 . . . 38
  - 7.6.1 The probes for FGF4 . . . 43
  - 7.6.2 Behaviour of the control probes . . . 43
  - 7.6.3 Gene set enrichment analysis . . . 44

8 Jensen-Shannon Divergence analysis . . . 49

9 Classification of temporal profiles . . . 52
- 9.1 Comparison of microarray data with qPCR results . . . 52
- 9.2 Rule-based classification . . . 54
  - 9.2.1 Table export . . . 56
  - 9.2.2 Comparison with manual classification . . . 56

10 qPCR data analysis . . . 59
- 10.1 Heatmaps for all data in `xq` . . . 59
- 10.2 Heatmaps for the seven selected genes and selected samples . . . 59
- 10.3 Distribution of the data and discretisation . . . 60
- 10.4 Temporal order - hierarchy . . . 63
  - 10.4.1 How significant is this? . . . 65

A Influence of cell position on gene expression . . . 67

B Correlation between Fgf ligands and Fgf receptors . . . 68

C Session info . . . 68

D The data import script `readdata.R` . . . 69

List of Figures

1 Histogram of standard deviation of each probe set's signal 7
2 The barplot shows, for each of the `ncol(mas5c)` arrays, the number of genes targeted by probe sets with "A" (light grey), "M" (light blue) and "P" (blue) calls. The `ncol(mas5c)+1`-th bar at the very right corresponds to detection in at least `fractionOfArrays*100`% of arrays. 9
3 Cluster stability analysis with E3.25 and E3.5 WT samples. 11
4 Cluster stability analysis with E3.5/E4.5 WT and E4.5 FGF4-KO samples. . . 12
5 Cluster stability analysis with E3.25 WT and E3.5 FGF4-KO samples. 13
6 Heatmap of the E3.25 WT and E3.5 FGF4-KO samples. 14
7 The influence of the parameter `ngenes` on the clustering result. 15
8 Determination of the cutoff for independent filtering on E3.5 WT samples. . 17
9 Heatmap of all WT arrays. 19
10 Heatmap of all WT arrays with duplicate features collapsed. 20
11 Heatmap of only the WT arrays from E3.5. 21
12 Heatmap of only the WT arrays from E3.5 with duplicate features collapsed. 22
13 Determination of the cutoff for independent filtering on E4.5 WT samples. . 23
14 Heatmap of differentially expressed genes from E3.25 to E3.5 (EPI), and from E3.25 to E3.5 (PE). 25
15 Projection of sample expression profiles on the differential expression signature from Section 4. 27
16 PCA plot, using WT samples. 28
17 Sorted loadings (coefficients) of the first two PCA vectors. 28
18 PCA plot for WT and FGF4-KO samples. 30
19 Same as Figure 18, with labels indicating the array (sample) number. 30
20 Same as Figure 18, with labels indicating `Total.number.of.cells`. 31
21 Heatmap of all arrays. 32
22 FGF4 expression. 33
23 FGF4 expression (microarray signal) in WT and KO samples. 34
24 MDS plot of the E3.25 wild type and FGF4-KO samples. 35
25 Relationship between FGF4 expression and similarity of the transcription profile to the KO. 36
26 Variability of different groups of samples. 37
27 Distance of the E3.25 WT samples to the mean profile of FGF4-KO. 39
28 Distance of the E3.25 WT samples to the mean profile of FGF4-KO. 40
29 Heatmap of all E3.25 WT and E3.25 FGF-KO samples. 41
30 Differentially expressed genes between FGF4-KO and WT (EPI, PE) at E3.5. 42
31 Differentially expressed genes between FGF4-KO and WT (EPI, PE) at E3.5. 43
32 The probes for FGF4. 44
33 Behaviour of some control probe sets. 45
34 Differentially expressed genes between FGF4-KO and WT (EPI, PE) at E3.5. 47
35 Jensen-Shannon divergences. 51
36 Temporal change of the lineage marker expression 53
37 Boxplots of the microarray expression data for exemplary genes. 54
38 Heatmap of all classes 57
39 Heatmap of the qPCR data set. 60
40 Heatmap of the qPCR data for the 7 selected genes. 60
41 Heatmap for the 7 selected genes and the E3.25/E3.5/E4.5 PE samples. . . . 61
42 Visualisation of the qPCR data for the seven genes. 62
43 Heatmap for the discretized values. 63

44 Heatmaps, sorted 64
45 Distribution of bootstrap-resampled optimal costs ("disorder penalties"). . . . 66
46 Multi-dimensional scaling plot. 67
47 Correlation between Fgf ligands and Fgf receptors. 68

1 Data import and preparations

We first load the required R package and set the random seed.

```
> library("Hiiragi2013")
> set.seed(2013)
```

The array data consist of a set of CEL files (the output from the Affymetrix scanner / image analysis software), whose annotation is provided in an Excel table. The CEL files are deposited at Array Express under the accession code E-MTAB-1681. The import of these data and metadata is performed by the script `readdata.R`, whose code is shown on page 69 and following. This script also performs data preprocessing ("normalisation") using the RMA method [1] and arranges the metadata to support the analyses presented in the following. Let us load the result of `readdata.R`.

```
> data("x")
> x

ExpressionSet (storageMode: lockedEnvironment)
assayData: 45101 features, 101 samples
  element names: exprs
protocolData
  sampleNames: 1 E3.25 2 E3.25 ... 101 E4.5 (FGF4-KO) (101 total)
  varLabels: ScanDate
  varMetadata: labelDescription
phenoData
  sampleNames: 1 E3.25 2 E3.25 ... 101 E4.5 (FGF4-KO) (101 total)
  varLabels: File.name Embryonic.day ... sampleColour (8 total)
  varMetadata: labelDescription
featureData
  featureNames: 1415670_at 1415671_at ... AFFX-TrpnX-M_at (45101 total)
  fvarLabels: symbol genename ensembl
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation: mouse4302
```

`x` is an *ExpressionSet* object containing the normalized data for the 101 arrays. These include 66 wild type (WT) samples

```
> with(subset(pData(x), genotype=="WT"),
+      addmargins(table(Embryonic.day, Total.number.of.cells), 2))

             Total.number.of.cells
Embryonic.day 32 33 34 41 49 50 62 75 91 207 <NA> Sum
        E3.25 11  6  5  4  4  6  0  0  0   0    0  36
        E3.5   0  0  0  0  0  0  6  8  8   0    0  22
        E4.5   0  0  0  0  0  0  0  0  0   8    0   8
```

and 35 FGF4-KO mutants

```
> with(subset(pData(x), genotype=="FGF4-KO"), table(Embryonic.day))

Embryonic.day
E3.25  E3.5  E4.5
```

```
    17     8    10
```

1.1 Grouping of samples

The preprocessed data object defines the grouping of the samples and an associated colour map, which will be used in the plots throughout this report.

```
> groups = split(seq_len(ncol(x)), pData(x)$sampleGroup)
> sapply(groups, length)

          E3.25 E3.25 (FGF4-KO)      E3.5 (EPI)  E3.5 (FGF4-KO)       E3.5 (PE)
             36              17              11               8              11
     E4.5 (EPI)  E4.5 (FGF4-KO)       E4.5 (PE)
              4              10               4
```

Each sample has assigned a colour which will be used in the subsequent plots.

```
> sampleColourMap = setNames(unique(pData(x)$sampleColour), unique(pData(x)$sampleGroup))
> sampleColourMap

          E3.25       E3.5 (PE)      E3.5 (EPI)       E4.5 (PE)      E4.5 (EPI)
      "#CAB2D6"       "#B2DF8A"       "#A6CEE3"       "#33A02C"       "#1F78B4"
E3.25 (FGF4-KO)  E3.5 (FGF4-KO)  E4.5 (FGF4-KO)
      "#FDBF6F"       "#FF7F00"       "#E31A1C"
```

For some analyses, we need to specifically address the four probes mapping to FGF4.

```
> FGF4probes = (fData(x)$symbol == "Fgf4")
> stopifnot(sum(FGF4probes)==4)
```

2 How many genes are expressed?

In this section, we aim to determine how many distinct mRNAs were detected by the arrays over the background level in the 66 WT samples.

```
> selectedSamples = with(pData(x), genotype=="WT")
> xe = x[, selectedSamples]
> stopifnot(ncol(xe)==66)
```

Because of the presence of background signal (stray light, cross-hybridisation), the answer to the question whether a transcript is present in a sample, based on Affymetrix GeneChip data, is not straightforward. In addition, the problem is complicated by the fact that the background signal is probe-sequence dependent. We perform two approaches that are used in the literature.

2.1 By variability

The first approach is based on the notion that the existence of *variability* of signal across samples is a more specific indicator of a transcript's presence, than the absolute signal intensity [2]. Let us plot the histogram of the standard deviation of each probe set's signal, across the 66 samples (Figure 1).

```
> library(genefilter)
> sdxe = rowSds(exprs(xe))
> thresh = 0.5
> hist(sdxe, 100, col = "skyblue")
> abline(v = thresh)
```

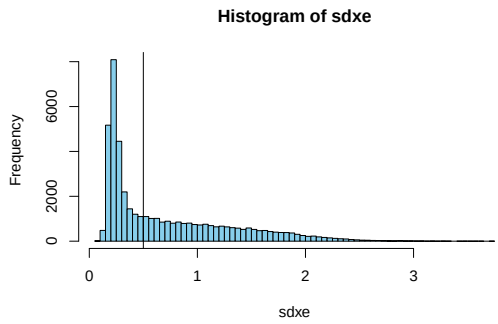


Figure 1: Histogram of standard deviation of each probe set's signal across the 66 samples.

Based on the (visually) chosen threshold[1] `thresh`, we find the following numbers of probe sets and unique target gene identifiers.

```
> table(sdxe>=thresh)

FALSE  TRUE
24140 20961
```

[1]One could also come up with a more automated, "statistical" rule, but the downstream result would be very similar.

```
> length(unique(fData(xe)$ensembl[ sdxe>=thresh ]))

[1] 11130
```

2.2 By Affymetrix present/absent calls

The second approach uses the Wilcoxon signed rank-based gene expression presence/absence detection algorithm first implemented in the Affymetrix Microarray Suite version 5.

```
> mas5c = mas5calls(a[, selectedSamples])
```

where `a` is an *AffyBatch* object containing the unprocessed raw intensity values. See `system.file("scripts", "readdata.R", package = "Hiiragi2013")` for details on how to create it by downloading raw data from ArrayExpress. Some bookkeeping and shuffling around is needed to compute the number of genes, as defined by unique Ensembl identifiers, for the classes *present* (P), *marginal* (M) and *absent* (A). If multiple probe sets map to one gene (which happens frequently on this array type), then P trumps M trumps A.

```
> myUnique = function(x) setdiff(unique(x), "")
> allEnsemblIDs = myUnique(fData(xe)$ensembl)
> callsPerGenePerArray = matrix(0, nrow = length(allEnsemblIDs), ncol = ncol(mas5c)+1,
+                               dimnames = list(allEnsemblIDs, NULL))
> for(j in seq_len(ncol(mas5c))) {
+   for(k in 1:2) {
+     ids = myUnique(fData(xe)$ensembl[ exprs(mas5c)[, j]==c("M","P")[k] ])
+     callsPerGenePerArray[ids, j] = k
+   }
+ }
> fractionOfArrays = 0.1
> for(k in 1:2) {
+   ids = myUnique(fData(xe)$ensembl[ apply(exprs(mas5c)==c("M","P")[k], 1,
+                                           function(v) (mean(v)>fractionOfArrays)) ])
+   callsPerGenePerArray[ids, ncol(mas5c)+1] = k
+ }
> numCalls = apply(callsPerGenePerArray, 2, table)
> numCalls = numCalls[rev(seq_len(nrow(numCalls))), ]
```

```
> numCalls[, 67]
```

```
> gplots::barplot2(numCalls, names.arg = paste(seq_len(ncol(callsPerGenePerArray))),
+          col = c(grDevices::palette.colors(8, "Paired")[2:1], "#e8e8e8"), ylab = "number of genes")
```

See Figure 2.

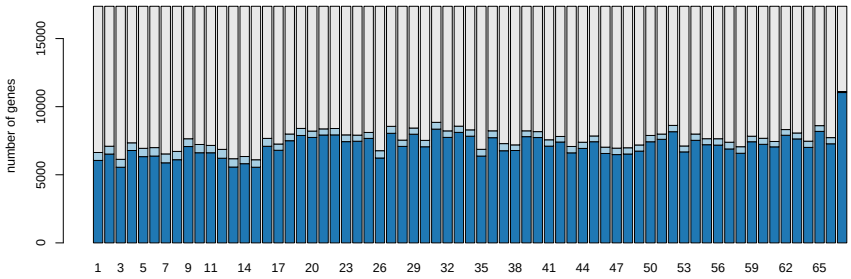


Figure 2: The barplot shows, for each of the `ncol(mas5c)` arrays, the number of genes targeted by probe sets with "A" (light grey), "M" (light blue) and "P" (blue) calls. The `ncol(mas5c)+1`-th bar at the very right corresponds to detection in at least `fractionOfArrays*100`% of arrays.

3 Cluster stability analysis

3.1 E3.25 and E3.5 WT samples

In this section, we investigate the hypothesis that the data for E3.5 fall 'naturally' into two clusters (associated with PE and EPI), while the data for E3.25 do not. For this, we use the framework of the *clue* package [3]. Briefly, the below function `clusterResampling` performs the following steps:

1. Draw a random subset of the full data (the full data are either all E3.25 or all E3.5 samples) by selecting 67% of the samples.
2. Select the top `ngenes` (see below) features by overall variance (in the subset).
3. Apply k-means clustering, and predict the cluster memberships of the samples that were not in the subset with the `cl_predict` method, through their proximity to the cluster centres.
4. Repeat steps 1-3 for `B` $= 250$ times.
5. Apply consensus clustering (`cl_consensus`).
6. For each of the `B` $= 250$ clusterings, measure the agreement with the consensus (`cl_agreement`); here, good agreement is indicated by a value of 1, and less agreement by smaller values. If the agreement is generally high, then the clustering into k classes can be considered stable and reproducible; inversely, if it is low, then no stable partition of the samples into k clusters is evident.

As a measure of between-cluster distance for the consensus clustering, the *Euclidean* dissimilarity of the memberships is used, i. e., the square root of the minimal sum of the squared differences of $\mathbf{u}$ and all column permutations of $\mathbf{v}$, where $\mathbf{u}$ and $\mathbf{v}$ are the cluster membership matrices. As agreement measure for step 6, the quantity $1 - d/m$ is used, where d is the Euclidean dissimilarity, and m is an upper bound for the maximal Euclidean dissimilarity [4].

```
> library(clue)
> clusterResampling = function(x, ngenes, k = 2, B = 250, prob = 0.67) {
+   mat = exprs(x)
+   ce = cl_ensemble(list = lapply(seq_len(B), function(b) {
+     selSamps = sample(ncol(mat), size = round(prob*ncol(mat)), replace = FALSE)
+     submat = mat[, selSamps, drop = FALSE]
+     selFeats = order(rowVars(submat), decreasing = TRUE)[seq_len(ngenes)]
+     submat = submat[selFeats,, drop = FALSE]
+     pamres = cluster::pam(t(submat), k = k, metric = "euclidean")
+     pred = cl_predict(pamres, t(mat[selFeats, ]), "memberships")
+     as.cl_partition(pred)
+   }))
+   cons = cl_consensus(ce)
+   ag = sapply(ce, cl_agreement, y = cons)
+   return(list(agreements = ag, consensus = cons))
+ }
```

```
> ce = list(
+   "E3.25" = clusterResampling(x[, unlist(groups[c("E3.25")])], ngenes = 20),
+   "E3.5"  = clusterResampling(x[, unlist(groups[c("E3.5 (EPI)", "E3.5 (PE)")])],
+                               ngenes = 20))
```

The results are shown in Figure 3. They confirm the hypothesis stated at the beginning of this section.

```
> par(mfrow = c(1,2))
> colours  = c(sampleColourMap["E3.25"], grDevices::palette.colors(9,"Set 1")[9])
> boxplot(lapply(ce, `[[`, "agreements"), ylab = "agreement probabilities", col = colours)
> mems = lapply(ce, function(x) sort(cl_membership(x$consensus)[, 1]))
> mgrp = lapply(seq(along = mems), function(i) rep(i, times = length(mems[[i]])))
> myjitter = function(x) x+seq(-.4, +.4, length.out = length(x))
> plot(unlist(lapply(mgrp, myjitter)), unlist(mems),
+      col = colours[unlist(mgrp)], ylab = "membership probabilities",
+      xlab = "consensus clustering", xaxt = "n", pch = 16)
> text(x = 1:2, y = par("usr")[3], labels = c("E3.25","E3.5"), adj = c(0.5, 1.4), xpd = NA)
```

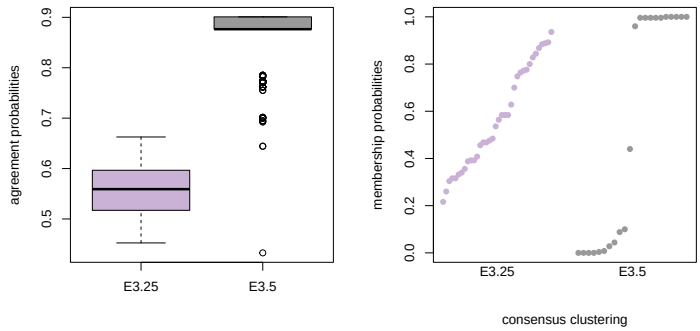


Figure 3: Cluster stability analysis with E3.25 and E3.5 WT samples. Left: boxplot of the cluster agreements with the consensus, for the `B`=250 clusterings; 1 indicates perfect agreement, and the value decreases with worse agreement. The statistical significance of the difference is confirmed by a Wilcoxon test in the main text. Right: membership probabilities of the consensus clustering; colours are as in the left panel. For E3.25, the probabilities are diffuse, indicating that the individual (resampled) clusterings disagree a lot, whereas for E3.5, the distribution is bimodal, with only one ambiguous sample.

We can compute a p-value for the statistical significance of the two distributions shown in the boxplot of Figure 3.

```
> wilcox.test(ce$E3.25$agreements, ce$E3.5$agreements)

        Wilcoxon rank sum test with continuity correction

data:  ce$E3.25$agreements and ce$E3.5$agreements
W = 266, p-value < 2.2e-16
alternative hypothesis: true location shift is not equal to 0
```

3.2 E3.5/E4.5 WT and E4.5 FGF4-KO samples

From the PCA plot in Figure 18, one might derive the impression that the FGF4 KO E4.5 cells cluster together with the EPI E3.5 cells. Here, we perform a cluster stability analysis analogous to that in Section 3.1, using the FGF4-KO E4.5 cells together with

1. the WT EPI E3.5 and WT PE E3.5 cells,
2. the WT EPI E4.5 and WT PE E4.5 cells.

As we will see in the following, in each of the above cases, three distinct clusters exist, and the above mentioned impression is not substantiated; in addition, we can also clearly distinguish the WT and KO samples at E4.5.

```
> sampleSets = list(
+   `E3.5` = unlist(groups[c("E3.5 (EPI)", "E3.5 (PE)", "E4.5 (FGF4-KO)")]),
+   `E4.5` = unlist(groups[c("E4.5 (EPI)", "E4.5 (PE)", "E4.5 (FGF4-KO)")]))
> k = 3
> csa = lapply(sampleSets, function(samps) list(
+   colours = x$sampleColour[samps],
+   r = clusterResampling(x[!FGF4probes, samps], ngenes = 20, k = k)))
```

```
> par(mfrow = c(2,3))
> for(i in seq(along = csa))
+   for(j in seq_len(k))
+     plot(cl_membership(csa[[i]]$r$consensus)[, j],
+          ylab = paste0("P(cluster ", j, ")"), xlab = "samples",
+          main = names(sampleSets)[i], col = csa[[i]]$colours, pch = 16, cex = 1.5)
```

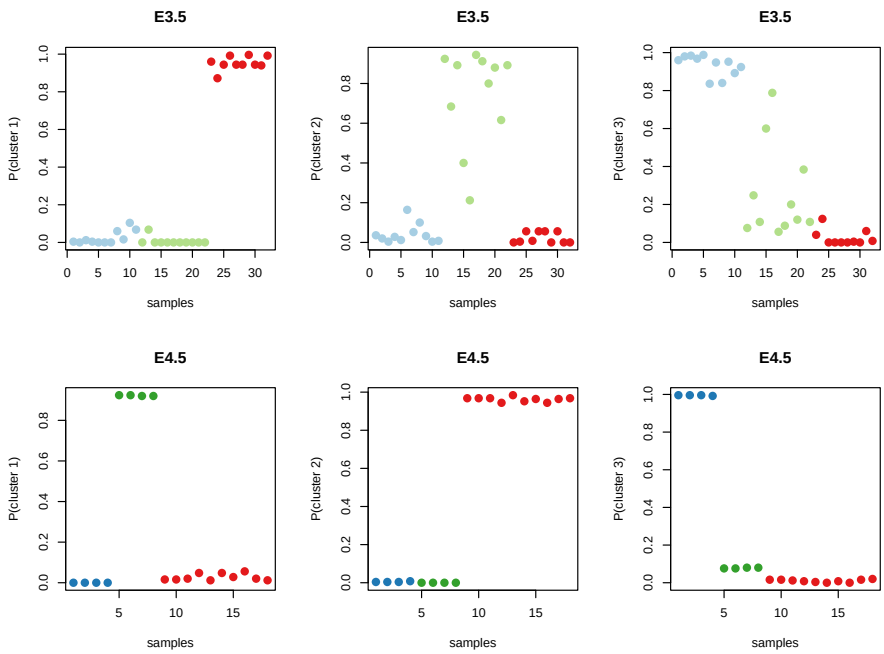


Figure 4: Cluster stability analysis with E3.5/E4.5 WT and E4.5 FGF4-KO samples. Top row: Results of the cluster stability analysis with the E3.5 (EPI), E3.5 (PE), E4.5 (FGF4-KO) samples. Shown on the y-axis is the membership probability P in the three clusters. The actual group membership of the samples (known to us but not used by the clustering algorithm) is indicated by the colours. Bottom row: Similarly for E4.5 (EPI), E4.5 (PE), E4.5 (FGF4-KO). These analyses indicate that the FGF4-KO very clearly separate from the WT samples, which between themselves tend to form the clusters already described in Section 3.1.

The results of the above code are shown in Figure 4 and indicate that indeed FGF4-KO E4.5 cells are distinct from either of the WT groups.

3.3 E3.25 WT and E3.5 FGF4-KO samples

Although in the 2-dimensional PCA projection in Figure 18 dots representing FGF4-KO E3.5 cells appear to overlap with ones representing WT E3.25 samples, cluster stability analysis analogous to that from the previous section indicates that they form a distinct population (Figure 5).

```
> sampleSets = unlist(groups[c("E3.25", "E3.5 (FGF4-KO)")])
> selSamples = x[!FGF4probes, sampleSets]
> resampledSampleSet = clusterResampling(selSamples, ngenes = 20)
```

```
> par(mfrow = c(1,2))
>   for(j in seq_len(2))
+     plot(cl_membership(resampledSampleSet$consensus)[, j],
+          ylab = paste0("P(cluster ", j, ")"), xlab = "Samples",
+          col = x$sampleColour[sampleSets], pch = 16, cex = 1.5)
```

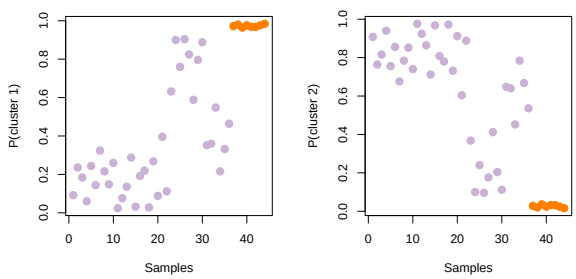


Figure 5: Cluster stability analysis with E3.25 WT and E3.5 FGF4-KO samples. Results of the cluster stability analysis with the E3.25 WT and E3.5 FGF4-KO samples. The y-axis shows the membership probability in the two clusters. The actual group membership of the samples (known but not used by the clustering algorithm) is indicated by the colours.

The cluster stability analysis of the two clusters reveals that the FGF4-KO samples at E3.5 form a single, tight cluster, and are consistently together throughout all of the resamplings. The E3.25 WT cells, on the other hand, are much more diffuse and in the course of the resampling, each cell does not necassaryly cluster together with the same cluster all the time. Therefore, the difference is prevalently one of variability—the E3.25 WT are quite variable and cover a large "expression space", whereas the FGF4-KO are stuck on one particular, narrowly defined expresion profile. This is also evident from the heatmap showed in Figure 6.

```
> ngenes = 100
> selFeats = order(rowVars(exprs(selSamples)), decreasing = TRUE)[seq_len(ngenes)]
> myHeatmap(selSamples[selFeats, ], collapseDuplicateFeatures = TRUE, haveColDend = TRUE)
```

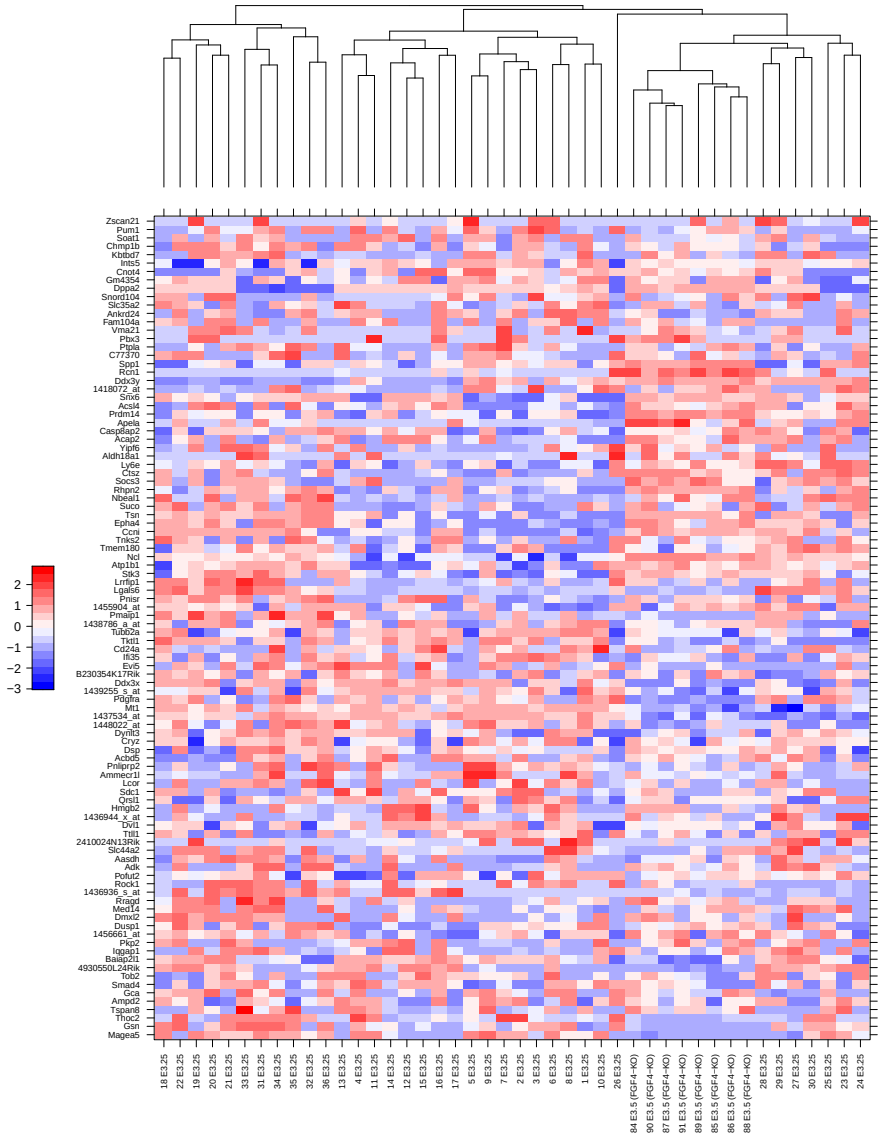


Figure 6: Heatmap of the E3.25 WT and E3.5 FGF4-KO samples. Data of the 100 genes with the highest variance. Even though a few individual cells from E3.25 WT do indeed seem to have similar expression profiles as ones from E3.5 FGF4-KO, which all cluster together, the populations are not the same.

4 Lineage Markers

In this section, the following steps are performed:

1. Section 4.1: Cluster the arrays from E3.5 into two clusters (using k-means clustering with $k = 2$ on the overall expression profiles).
2. Section 4.2: Determine the genes that are differentially expressed between these two clusters, according to a t-test with a nominal cutoff of false discovery rate (FDR) of 10%. These genes are reported in the table `differentially-expressed-features-3.5.csv`, which can be imported into Excel etc.
3. Section 4.2.1: Present a heatmap of these genes.
4. Section 4.3: Produce an analogous table `differentially-expressed-features-4.5.csv` for the E4.5 samples.

4.1 Clustering of E3.5 WT samples

The function `pamCluster` selects the `ngenes` most variable genes in the data matrix `x` and seeks for two clusters using the *partitioning around medoids* method. To assess the influence of the parameter `ngenes`, we call the algorithm for multiple choices, from 10 to 10000.

```
> ngenes = c(10, 25, 50, 100, 250, 500, 1000, 2500, 5000, 10000)
```

```
> xForClustering = x[, x$Embryonic.day=="E3.5" & x$genotype=="WT"]
> clusters = sapply(ngenes, pamCluster, x = xForClustering)
```

The result is displayed in Figure 7.

```
> image(x = seq_len(nrow(clusters)), y = seq_len(ncol(clusters)), z = clusters,
+       col = c("#f0f0f0", "#000000"), ylab = "ngenes", xlab = "samples", yaxt = "n")
> text(x = 0, y = seq_len(ncol(clusters)), paste(ngenes), xpd = NA, adj = c(1, 0.5))
```

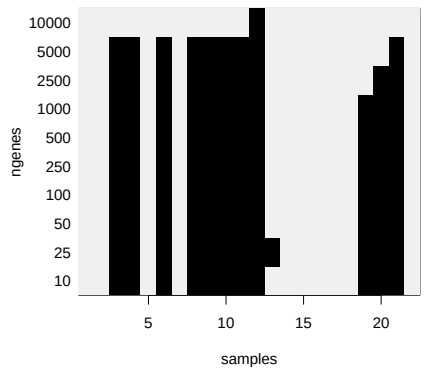


Figure 7: The influence of the parameter `ngenes` on the clustering result. Each of the 22 samples corresponds to a column of the matrix, the rows of the matrix correspond to different choices for `ngenes`. Cluster membership is indicated by the colour code (light gray *vs* black). For most samples, cluster membership is consistent throughout the range of `ngenes` from 10 to 1000. `ngenes` = 1000 is used for subsequent analyses.

From Figure 7 we can conclude that we can proceed with the choice of

```
> i = which(ngenes==1000); stopifnot(length(i)==1)
> ngenes = ngenes[i]
> clusters = factor(clusters[, i])
```

Now we can check how the microarray-data driven clustering compares with the annotation of the cells that was provided by Yusuke in the Excel table:

```
> table(clusters, pData(x)[names(clusters), "lineage"])

clusters EPI PE
       1   0 11
       2  11  0
```

As we can see, the clustering perfectly agrees with Yusuke's labeling. *Note*: the `lineage` annotation was used here only to assess the clustering output. It is not used as input for any of the data analyses shown in this document.

```
> cbind(unlist(groups[c("E3.5 (EPI)", "E3.5 (PE)")]), ce[[2]]$consensus$.Data[, 1])

             [,1]  [,2]
E3.5 (EPI)1    39 1.000
E3.5 (EPI)2    40 1.000
E3.5 (EPI)3    42 1.000
E3.5 (EPI)4    44 0.996
E3.5 (EPI)5    45 0.996
E3.5 (EPI)6    46 0.996
E3.5 (EPI)7    47 0.996
E3.5 (EPI)8    48 0.996
E3.5 (EPI)9    55 0.960
E3.5 (EPI)10   56 1.000
E3.5 (EPI)11   57 1.000
E3.5 (PE)1     37 0.000
E3.5 (PE)2     38 0.044
E3.5 (PE)3     41 0.000
E3.5 (PE)4     43 0.100
E3.5 (PE)5     49 0.440
E3.5 (PE)6     50 0.000
E3.5 (PE)7     51 0.000
E3.5 (PE)8     52 0.028
E3.5 (PE)9     53 0.008
E3.5 (PE)10    54 0.088
E3.5 (PE)11    58 0.004
```

4.2 Differentially expressed genes in E3.5 samples

```
> deCluster = rowttests(xForClustering, fac = clusters)
```

The code below, which produces Figure 8, is used to set the parameters for the independent filtering step [2].

```
> varianceRank = rank(-rowVars(exprs(xForClustering)))
> plot(varianceRank, deCluster$p.value, pch = ".", log = "y",
+      main = "Parameters for the independent filtering",
+      xlab = "variance rank", ylab = "p-value")
> nFilt = 20000
> smallpValue = 1e-4
> abline(v = nFilt, col = "blue")
> abline(h = smallpValue, col = "orange")
```

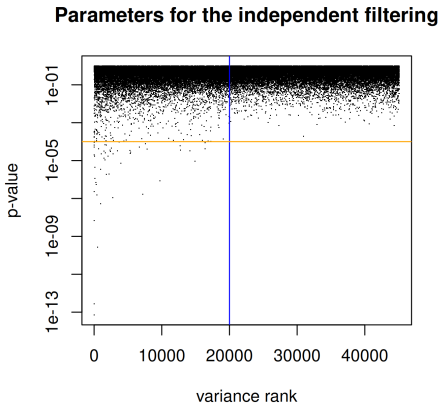


Figure 8: Determination of the cutoff for independent filtering on E3.5 WT samples. Each dot corresponds to one of the 45101 features on the array, the x-axis shows the rank of the features overall variance across the 22 arrays in `xForClustering`, the y-axis the p-value from the t-test on a logarithmic scale. The horizontal orange line corresponds to a p-value of 1e-04. The vertical blue line indicates the cutoff implied by taking only the 20000 features with the highest overall variance. We can see that if we apply this cutoff, we in fact do not miss any of the features with p-value smaller than 1e-04.

Let's perform false discovery rate (FDR) p-value adjustment using the Benjamini-Hochberg method [5].

```
> passfilter = which(varianceRank<=nFilt)
> adjp = rep(NA_real_, nrow(x))
> adjp[passfilter] = p.adjust(deCluster$p.value[passfilter], method = "BH")
> ord = order(adjp)
> numFeaturesReport = 200
> differentially = ord[seq_len(numFeaturesReport)]
> length(unique(fData(x)$symbol[differentially]))

[1] 163
```

The FDR of the selected set of `numFeaturesReport` $=$ 200 features is 22.8%, and these correspond to 163 unique genes. In the following code chunk, we write out the results table into a CSV file.

```
> deFeat35 = cbind(deCluster[differentially,], `FDR-adjusted p-value` = adjp[differentially],
+                  fData(x)[differentially,])
> write.csv(deFeat35, file = "differentially-expressed-features-3.5.csv")
```

4.2.1 Heatmaps

To visualise the data, we use the helper function `myHeatmap`. Heatmaps produced by the function calls below are shown in Figures 9–12.

```
> myHeatmap(x[differentially, x$genotype=="WT"])
```

```
> myHeatmap(x[differentially, x$genotype=="WT"], collapseDuplicateFeatures = TRUE)
```

```
> myHeatmap(xForClustering[differentially, ])
```

```
> myHeatmap(xForClustering[differentially, ], collapseDuplicateFeatures = TRUE)
```

4.3 Differentially expressed genes in E4.5 samples

The following code is analogous to Section 4.2.

```
> x45 = x[, x$Embryonic.day=="E4.5" & x$genotype=="WT"]
> de45 = rowttests(x45, fac = "lineage")
```

```
> varianceRank = rank(-rowVars(exprs(x45)))
> plot(varianceRank, de45$p.value, pch = ".", log = "y",
+      main = "Parameters for the independent filtering",
+      xlab = "variance rank", ylab = "p-value")
> abline(v = nFilt, col = "blue")
> abline(h = smallpValue, col = "orange")
```

See Figure 13.

```
> passfilter = which(varianceRank<=nFilt)
> adjp = rep(NA_real_, nrow(x))
> adjp[passfilter] = p.adjust(de45$p.value[passfilter], method = "BH")
> ord = order(adjp)
> differentially = ord[seq_len(numFeaturesReport)]
> length(unique(fData(x)$symbol[differentially]))

[1] 175
```

The FDR of the selected set of `numFeaturesReport` = 200 features is 1.04%, and these correspond to 175 unique genes. The following code chunk writes out the results table into a CSV file.

```
> deFeat45 = cbind(de45[differentially, ], `FDR-adjusted p-value` = adjp[differentially],
+                  fData(x)[differentially, ])
> write.csv(deFeat45, file = "differentially-expressed-features-4.5.csv")
```

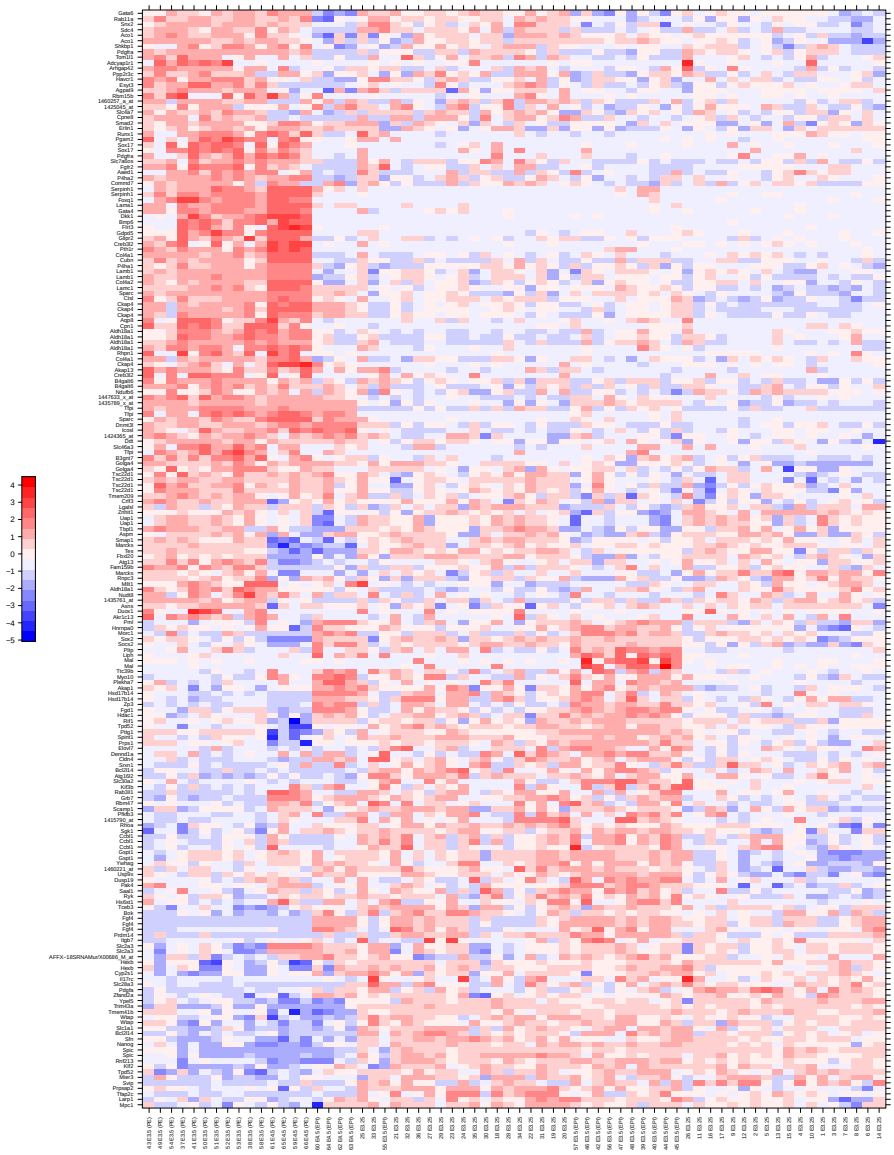


Figure 9: Heatmap of all WT arrays. Data from 200 features with evidence of differential expression between the two clusters are shown. For some genes, multiple features are shown.

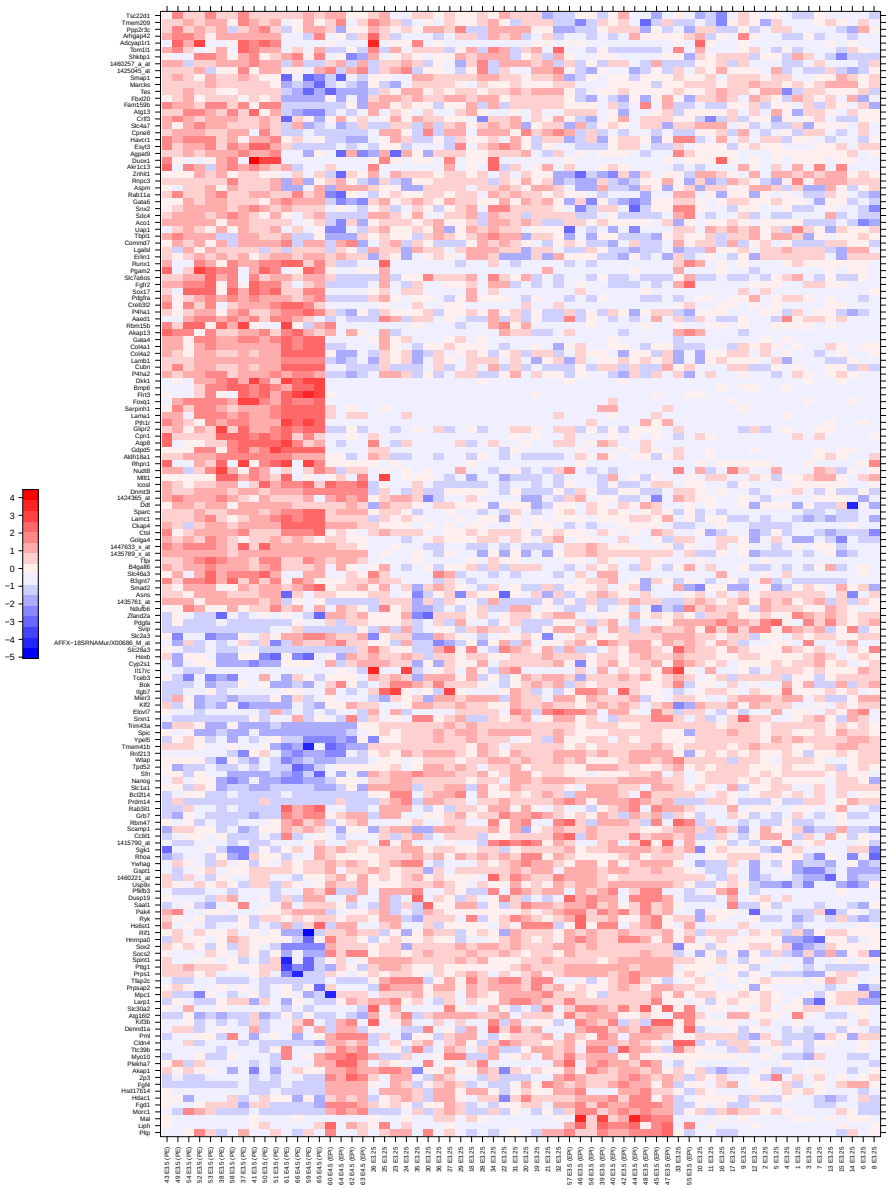


Figure 10: Heatmap of all WT arrays with duplicate features collapsed. Data from 200 features with evidence of differential expression between the two clusters are shown. Duplicate features per gene were averaged.

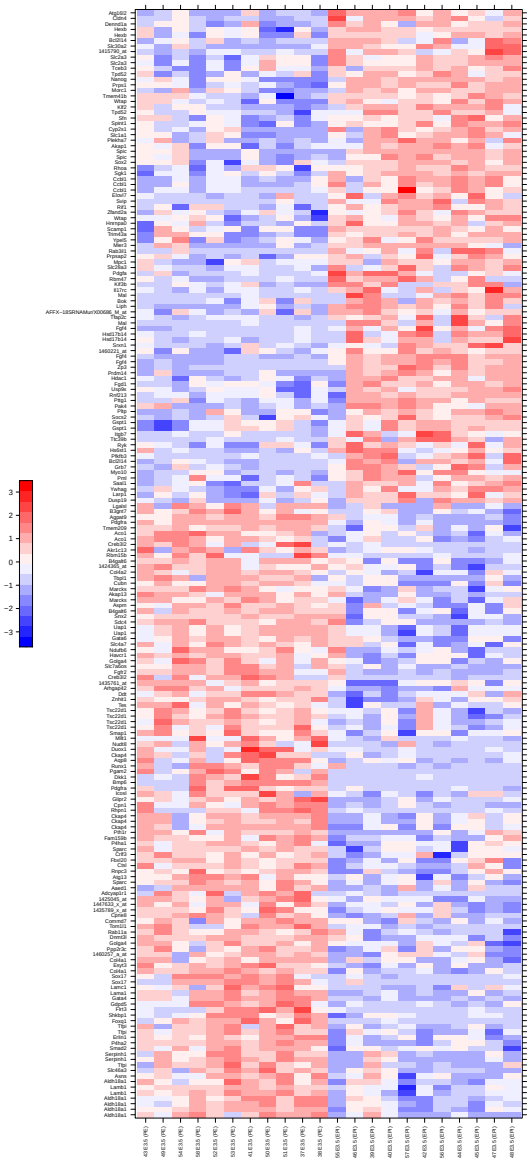


Figure 11: Heatmap of only the WT arrays from E3.5. Data from 200 features with evidence of differential expression between the two clusters are shown. For some genes, multiple features are shown.

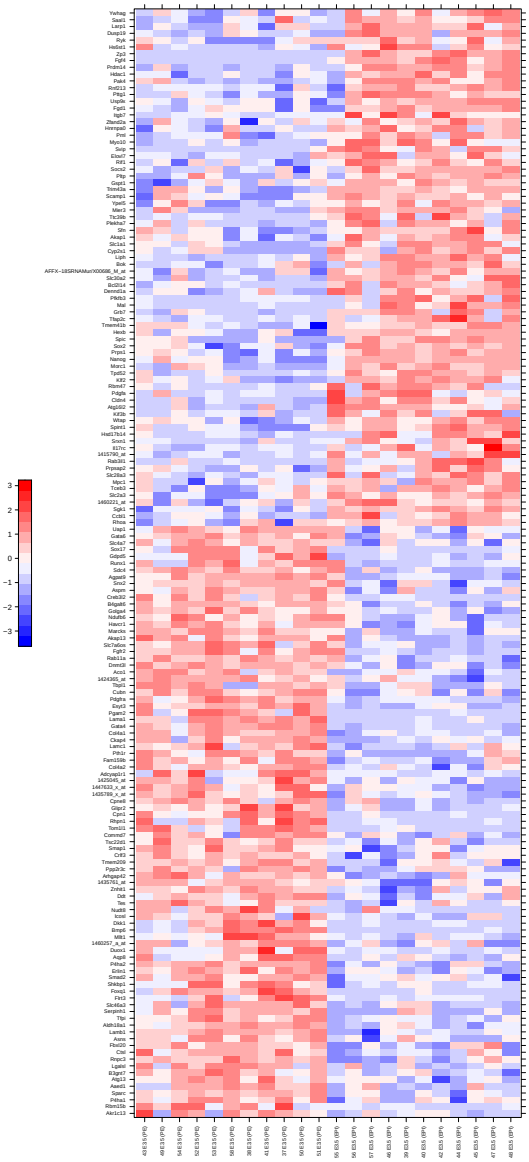


Figure 12: Heatmap of only the WT arrays from E3.5 with duplicate features collapsed. Data from 200 features with evidence of differential expression between the two clusters are shown. Duplicate features per gene were averaged.

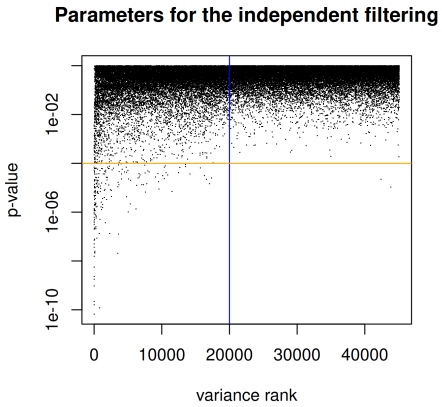


Figure 13: Determination of the cutoff for independent filtering on E4.5 WT samples. Analogous to Figure 8.

5 Differentially expressed genes from E3.25 to E3.5

To understand the transitions shown in the MDS plots in molecular terms, a look at the genes that are differentially expressed along the transitions from E3.25 to E3.5 (EPI) and from E3.25 to E3.5 (PE) might be instructive. To this end, we determine the differentially expressed genes for each of the two comparisons, build the union set, and display their data in the heatmap shown in Figure 14. It is interesting that some genes are specifying for either EPI or PE, whereas other genes show the same trend from E3.25 to E3.5 for both lineages (i. e. they only reflect time).

```
> samples = unlist(groups[c("E3.25", "E3.5 (EPI)", "E3.5 (PE)")])
> deE325toE35 = union(
+   getDifferentialExpressedGenes(x, groups, "E3.25", "E3.5 (EPI)"),
+   getDifferentialExpressedGenes(x, groups, "E3.25", "E3.5 (PE)"))
```

```
> myHeatmap(x[deE325toE35, samples], collapseDuplicateFeatures = TRUE)
```

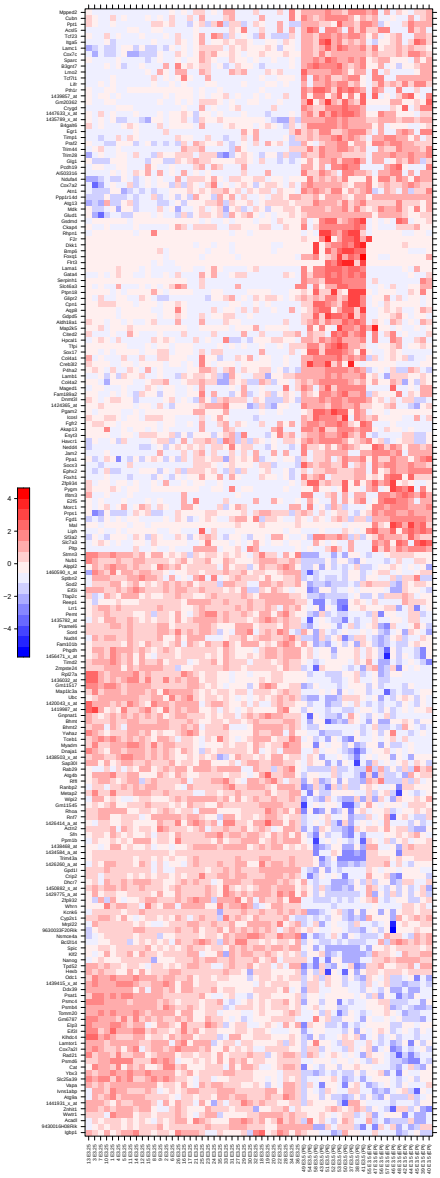


Figure 14: Heatmap of differentially expressed genes from E3.25 to E3.5 (EPI), and from E3.25 to E3.5 (PE). A standalone PDF file with this figure is also available, under the name `Hiiragi2013-figdifferentiallyE325toE35.pdf`.

6 Principal Component Analysis

As a first attempt at getting an overview over the data, see Figure 15.

```
> projection = deCluster$dm[differentially] %*% exprs(x)[differentially, ]
```

```
> plotProjection(projection, label = sampleNames(x),
+                col = x$sampleColour, colourMap = sampleColourMap)
```

6.1 On the WT samples

Select the WT samples.

```
> safeSelect = function(grpnames){
+   stopifnot(all(grpnames %in% names(groups)))
+   unlist(groups[grpnames])
+ }
> g = safeSelect(c("E3.25",
+                  "E3.5 (EPI)", "E3.5 (PE)",
+                  "E4.5 (EPI)", "E4.5 (PE)"))
```

Note: we do not use the data from all the 45101 features on the microarrays, since most of these are dominated by noise. Rather, we use the top 100 features according to overall variance across the WT samples, excluding, however, the FGF4 probes (to avoid any possible concerns about "trivial" effects of the KOs).

```
> nfeatures = 100
> varianceOrder = order(rowVars(exprs(x[, g])), decreasing = TRUE)
> varianceOrder = setdiff(varianceOrder, which(FGF4probes))
> selectedFeatures = varianceOrder[seq_len(nfeatures)]
> xwt = x[selectedFeatures, g]
```

Before embarking on the PCA computation, construct a new data matrix with equal group sizes.

```
> tab = table(xwt$sampleGroup)
> sp = split(seq_len(ncol(xwt)), xwt$sampleGroup)
> siz = max(listLen(sp))
> sp = lapply(sp, sample, size = siz, replace = (siz>length(x)))
> xwte = xwt[, unlist(sp)]
```

Now we are ready to do it.

```
> thepca = prcomp(t(exprs(xwte)), center = TRUE)
> pcatrsf = function(x) scale(t(exprs(x)), center = TRUE, scale = FALSE) %*% thepca$rotation
> stopifnot(all( abs(pcatrsf(xwte) - thepca$x) < 1e-6 ))
```

```
> myPCAplot = function(x, labels, ...) {
+   xy = pcatrsf(x)[, 1:2]
+   plot(xy, pch = 16, col = x$sampleColour, cex = 1, xlab = "PC1", ylab = "PC2", ...)
+   if(!missing(labels))
+     text(xy, labels, cex = 0.35, adj = c(0.5, 0.5))
```

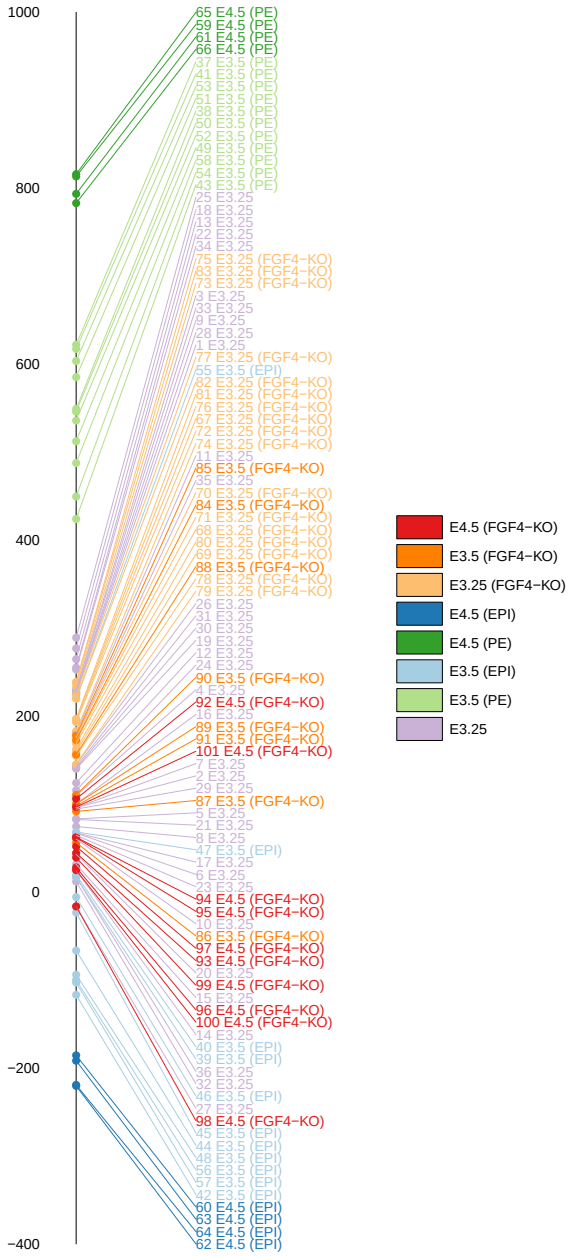


Figure 15: Projection of sample expression profiles on the differential expression signature from Section 4. Expression profiles and signature are each vectors of length 200, and shown is the scalar product between each sample's expression profile for these 200 features and the differential expression signature.

```
+ }
```

```
> myPCAplot(xwt)
```

See Figure 16. We also provide an overview over the distributions of loadings of the PCA components (Figure 17) and the 20 most important genes (10 with highest positive coefficients and 10 with the highest negative coefficients) in the R output below.

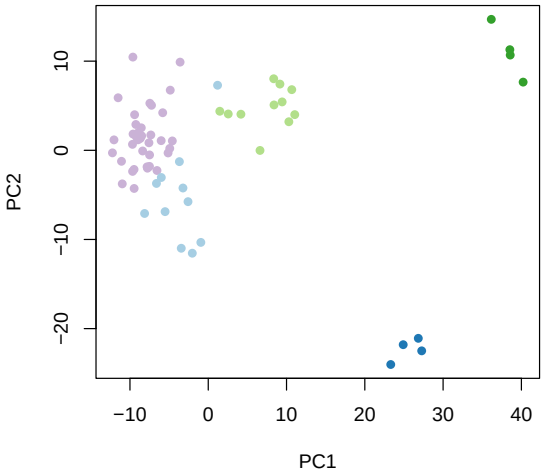


Figure 16: PCA plot, using WT samples. The colour code is as in Figure 15.

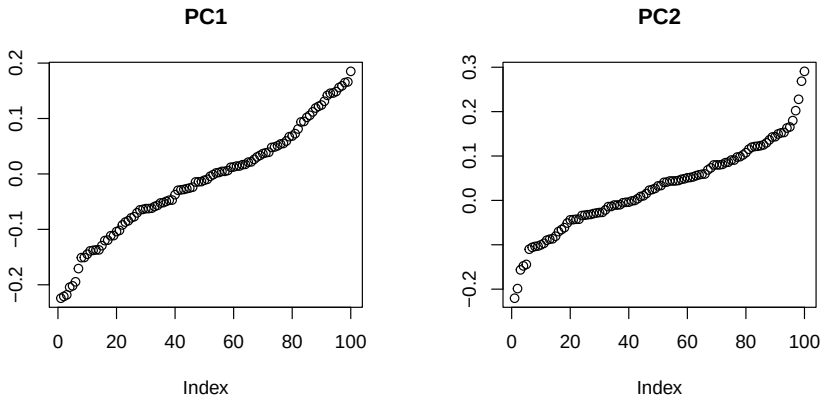


Figure 17: Sorted loadings (coefficients) of the first two PCA vectors. The most important genes are shown in the text.

```
> par(mfrow = c(1,2))
> for(v in c("PC1", "PC2")) {
+   loading = thepca$rotation[, v]
+   plot(sort(loading), main = v, ylab = "")
+   sel = order(loading)[c(1:10, (-9:0)+length(loading))]
+   print(data.frame(
+    symbol = fData(x)$symbol[selectedFeatures[sel]],
```

```
+    probe = names(loading)[sel],
+    loading = loading[sel], stringsAsFactors = FALSE
+   ))
+ }
                    symbol        probe    loading
1429483_at       Calcoco2   1429483_at -0.2242381
1434584_a_at 1434584_a_at 1434584_a_at -0.2210081
1460605_at          Crxos   1460605_at -0.2181986
1456270_s_at      Pramel6 1456270_s_at -0.2044059
1456598_at     1456598_at   1456598_at -0.2019409
1450624_at           Bhmt   1450624_at -0.1945991
1449134_s_at         Spic 1449134_s_at -0.1707423
1448573_a_at     Ceacam10 1448573_a_at -0.1509967
1437534_at     1437534_at   1437534_at -0.1506999
1433509_s_at        Reep1 1433509_s_at -0.1447711
1454737_at          Dusp9   1454737_at  0.1311659
1426255_at           Nefl   1426255_at  0.1417741
1439148_a_at         Pfkl 1439148_a_at  0.1455867
1437308_s_at          F2r 1437308_s_at  0.1460359
1450843_a_at     Serpinh1 1450843_a_at  0.1486257
1420498_a_at         Dab2 1420498_a_at  0.1558477
1448595_a_at         Bex1 1448595_a_at  0.1588866
1426722_at        Slc38a2   1426722_at  0.1648103
1418153_at          Lama1   1418153_at  0.1660538
1419737_a_at         Ldha 1419737_a_at  0.1850638
                    symbol        probe    loading
1419418_a_at        Morc1 1419418_a_at -0.2204497
1423754_at         Ifitm3   1423754_at -0.1987325
1436944_x_at 1436944_x_at 1436944_x_at -0.1565725
1448595_a_at         Bex1 1448595_a_at -0.1476457
1423747_a_at         Pdk1 1423747_a_at -0.1442752
1449254_at           Spp1   1449254_at -0.1100474
1422557_s_at          Mt1 1422557_s_at -0.1058178
1448830_at          Dusp1   1448830_at -0.1037331
1429388_at          Nanog   1429388_at -0.1028595
1426255_at           Nefl   1426255_at -0.1001978
1437325_x_at     Aldh18a1 1437325_x_at  0.1496692
1418153_at          Lama1   1418153_at  0.1521609
1450843_a_at     Serpinh1 1450843_a_at  0.1530877
1439256_x_at   Gpr137b-ps 1439256_x_at  0.1630355
1436838_x_at        Cotl1 1436838_x_at  0.1654778
1439255_s_at 1439255_s_at 1439255_s_at  0.1798579
1429177_x_at        Sox17 1429177_x_at  0.2022259
1421917_at         Pdgfra   1421917_at  0.2277869
1426990_at           Cubn   1426990_at  0.2685773
1452270_s_at         Cubn 1452270_s_at  0.2907478
```

6.2 WT and FGF4-KO samples

In the following we perform PCA analysis using both WT and FGF4-KO samples.

```
> myPCAplot(x[selectedFeatures, ])
```

See Figure 18.

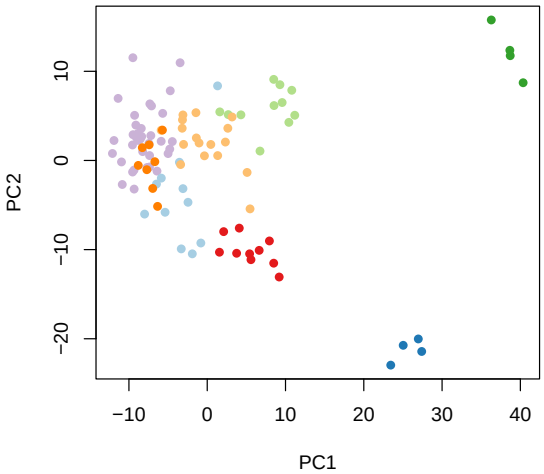


Figure 18: PCA plot for WT and FGF4-KO samples. The colour code is as in Figure 15.

```
> myPCAplot(x[selectedFeatures, ], labels = paste(seq_len(ncol(x))))
```

See Figure 19.

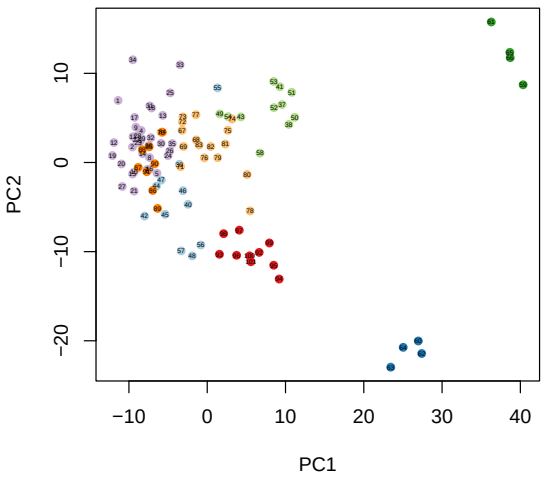


Figure 19: Same as Figure 18, with labels indicating the array (sample) number. This may be useful to detect outlier arrays.

```
> myPCAplot(x[selectedFeatures, ], labels = paste(x$Total.number.of.cells))
```

See Figure 20.

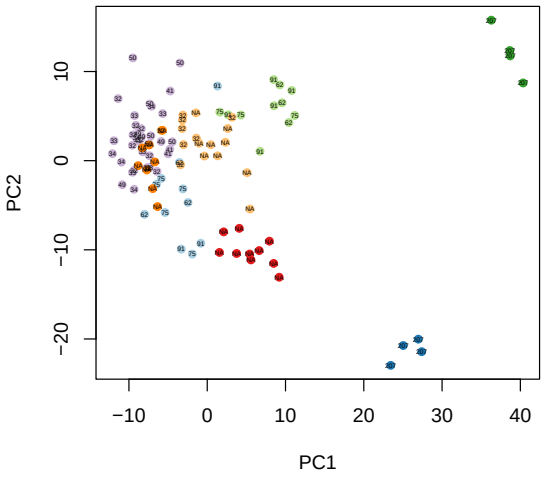


Figure 20: Same as Figure 18, with labels indicating `Total.number.of.cells`. This may be useful to detect "batch effects".

6.3 Heatmap of all WT and FGF4-KO samples

```
> mat = exprs(x[selectedFeatures, ])
> rownames(mat) = fData(x)[selectedFeatures, "symbol"]
```

```
> gplots::heatmap.2(mat, trace = "none", dendrogram = "none", scale = "row",
+           col = gplots::bluered(100), keysize = 0.9,
+           ColSideColors = x$sampleColour, margins = c(7,5))
```

See Figure 21.

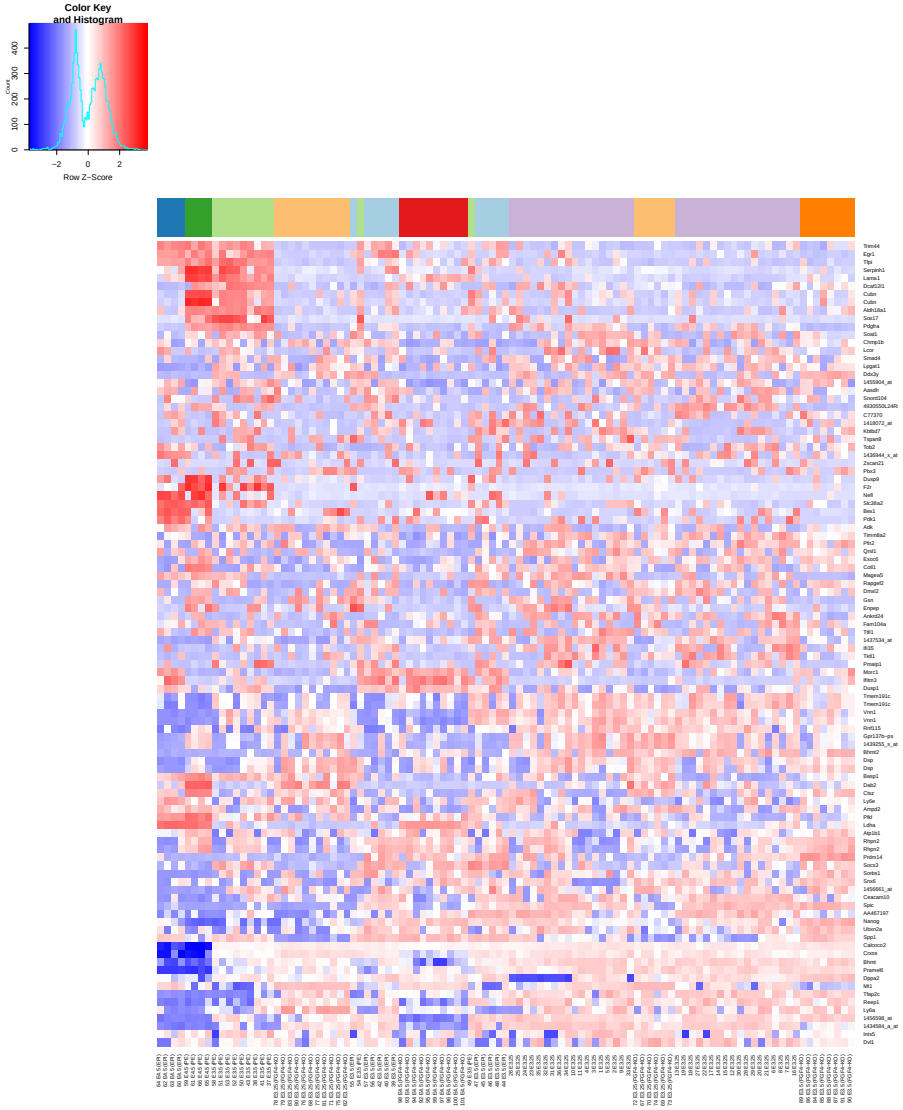


Figure 21: Heatmap of all arrays. Data from the 100 with the highest variance across the WT samples, excluding the FGF4 probes. The colour code of the bar at the top is as in Figure 15.

7 Further analyses of FGF4-KO

7.1 FGF4's expression pattern in E3.25 samples

In Figure 22, produced by the code below, we visualise the expression pattern of FGF4 in the E3.25 samples. The striking result is that there is a lot of natural variation in FGF4 expression even in the WT samples, and some of the lowest levels in the WT samples approach the background signal level seen for the KOs.

```
> x325  = x[, with(pData(x), Embryonic.day=="E3.25")]
> rv325 = rowVars(exprs(x325))
> featureColours = grDevices::palette.colors(sum(FGF4probes), "Dark 2")
> py = t(exprs(x325)[FGF4probes, ])
> matplot(py, type = "p", pch = 15, col = featureColours,
+         xlab = "arrays", ylab = expression(log[2] ~ signal),
+         ylim = range(py) + c(-0.7, 0))
> legend("topright", legend = rownames(fData(x325))[FGF4probes], fill = featureColours)
> points(seq_len(nrow(py)), rep(par("usr")[3]+0.2, nrow(py)),
+        pch = 16, col = x325$sampleColour)
```

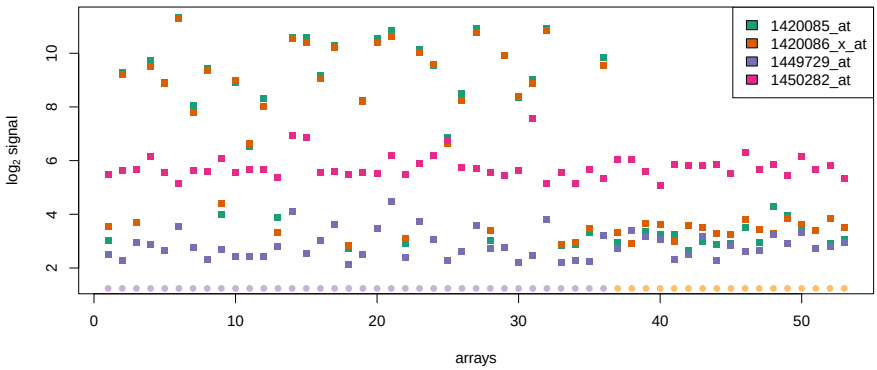


Figure 22: FGF4 expression. The plot shows data from the 4 features on the microarray annotated to FGF4. From these data, we conclude that `1420085_at` and `1420086_x_at` essentially report the same, and are likely to be good reporters for the FGF4 isoform that we are interested in, whereas the other two features measure something else. The circle symbols at the bottom of the plot indicate the samples' genotypes.

For presentation, we also produce another visualisation, this time only showing one value per array, which we obtain by averaging over the two "good" features (Figure 23).

```
> fgf4Expression = colMeans(exprs(x325)[c("1420085_at", "1420086_x_at"), ])
> fgf4Genotype = factor(x325$genotype,
+                       levels = sort(unique(x325$genotype),decreasing = TRUE))
```

```
> plot(x = jitter(as.integer(fgf4Genotype)),
+      y = fgf4Expression,
+      col = x325$sampleColour, xlim = c(0.5, 2.5), pch = 16,
+      ylab = expression(FGF4~expression~(log[2]~units)),
+      xlab = "genotype", xaxt = "n")
```

```
> cm = sampleColourMap[sampleColourMap %in% x325$sampleColour]
> legend("topright", legend = names(cm), fill = cm)
```

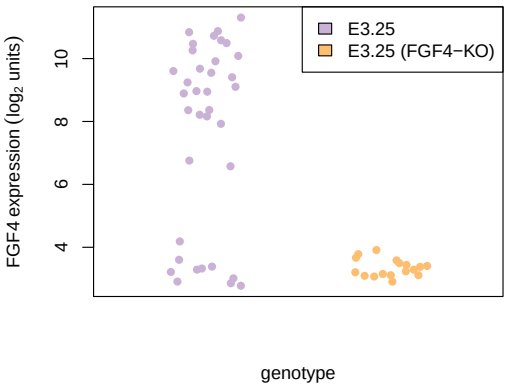


Figure 23: FGF4 expression (microarray signal) in WT and KO samples.

7.2 Are the E3.25 WT samples with low FGF4 expression more similar to the FGF4-KO samples than those with high FGF4?

This question is addressed by the code below, whose result is shown in Figure 24.

```
> zero2one = function(x) (x-min(x))/diff(range(x))
> rgb2col = function(x) {x = x/255; rgb(x[, 1], x[, 2], x[, 3])}
> colours = x325$sampleColour
> wt325 = x325$genotype=="WT"
> colourBar = function(x) rgb2col(colorRamp(c("yellow", "blue"))(zero2one(x)))
> colours[wt325] = colourBar(fgf4Expression)[wt325]
> selMDS = order(rv325, decreasing = TRUE)[seq_len(100)]
```

```
> MDSplot(x325[selMDS, ], col = colours)
```

```
> atColour = seq(min(fgf4Expression), max(fgf4Expression), length = 20)
> image(z = rbind(seq(along = atColour)), col = colourBar(atColour),
+       xaxt = "n", y = atColour, ylab = "")
```

Figure 24 indicates that

1. there is a relationship between FGF4 expression and overall global expression patterns in the WT samples;
2. WT samples with low FGF4 are more similar to the FGF4 KOs than than WT samples with high FGF4.

To more formally explore statement 2, we compute the Euclidean distance between each WT sample and the mean of the KOs, plot this against the FGF4 expression level (Figure 25) and test for correlation:

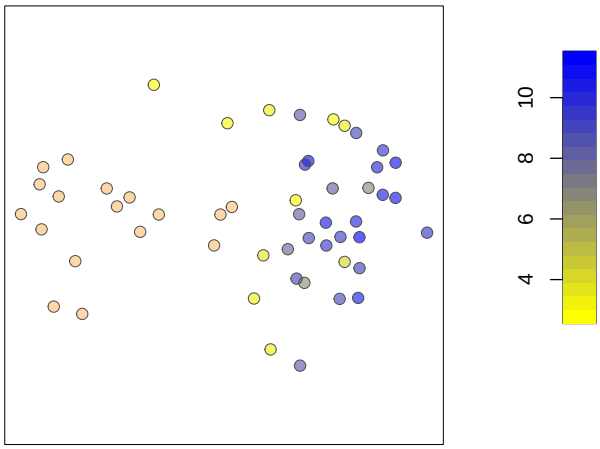


Figure 24: MDS plot of the E3.25 wild type (yellow–blue) and FGF4-KO (orange) samples. The yellow–blue colour scale represents the FGF4 expression level, as indicated in the colour bar.

```
> KOmean = rowMeans(exprs(x325)[selMDS, x325$genotype=="FGF4-KO"])
> dists = colSums((exprs(x325)[selMDS, wt325] - KOmean)^2)^0.5
> ct = cor.test(fgf4Expression[wt325], dists, method = "spearman")
> ct

        Spearman's rank correlation rho

data:  fgf4Expression[wt325] and dists
S = 4190, p-value = 0.005093
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.4607465
```

```
> plot(fgf4Expression[wt325], dists, pch = 16, main = "E3.25 WT samples",
+      xlab = "FGF4 expression", ylab = "Distance to FGF4-KO", col = colours)
```

There is a significant correlation between FGF4 expression in WT and similarity of the overall expression profile with that of the FGF4 KOs.

7.3 Variability of the FGF4-KO samples compared to WT samples

To address this question, let us take some precaution against possible batch effects. Therefore, we split the samples first by the `sampleGroup` classification defined above, but then also into groups according to the value of `Total.number.of.cells`, assuming that the samples within each such group have been processed together. See below.

```
> varGroups = split(seq_len(ncol(x)), f = list(x$sampleGroup,x$Total.number.of.cells),
+                   sep = ":", drop = TRUE)
```

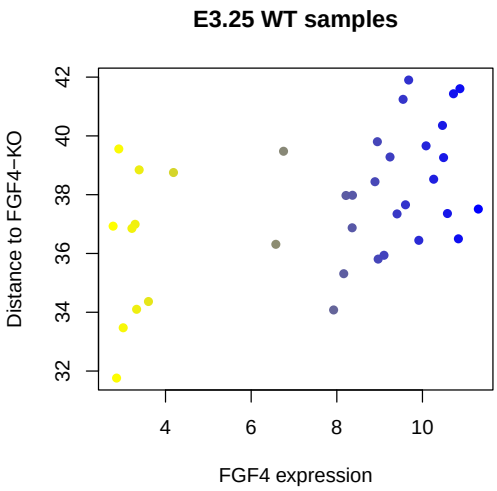


Figure 25: Relationship between FGF4 expression and similarity of the transcription profile to the KO. FGF4 expression of each sample is shown along the x-axis, the y-axis corresponds to the distance to the mean FGF4-KO expression profile computed over the 100 most variable features as selected in `selMDS`.

We can see how many samples are in each of these groups, and what the value of `ScanDate` is.

```
> data.frame(
+   `number arrays` = listLen(varGroups),
+   `ScanDates` = sapply(varGroups, function(v)
+     paste(as.character(unique(x$ScanDate[v])), collapse = ", ")),
+   stringsAsFactors = FALSE)

                    number.arrays              ScanDates
E3.25:32                       11             2011-03-16
E3.25 (FGF4-KO):32              8             2012-03-16
E3.25:33                        6             2011-03-15
E3.25:34                        5             2010-07-02
E3.25:41                        4             2010-07-02
E3.25:49                        4             2010-07-02
E3.25:50                        6             2010-07-02
E3.5 (EPI):62                   3 2010-06-30, 2010-07-01
E3.5 (PE):62                    3 2010-06-30, 2010-07-01
E3.5 (EPI):75                   5             2010-07-01
E3.5 (PE):75                    3             2010-07-01
E3.5 (EPI):91                   3             2010-09-16
E3.5 (PE):91                    5             2010-09-16
E4.5 (EPI):207                  4             2010-07-01
E4.5 (PE):207                   4 2010-07-01, 2010-07-02
E3.25 (FGF4-KO):NA              9             2012-08-16
E3.5 (FGF4-KO):NA               8             2013-03-05
E4.5 (FGF4-KO):NA              10             2013-03-05
```

Again select the top 100 genes according to `varianceOrder`.

```
> sel   = varianceOrder[seq_len(nfeatures)]
> myfun = function(x) median(apply(exprs(x), 1, mad))
> sds = lapply(varGroups, function(j) myfun(x[sel, j]))
> names(sds) = sprintf("%s (n=%d)", names(sds), listLen(varGroups))
> varGroupX = factor(sapply(strsplit(names(varGroups), split = ":"),`[`, 1))
```

```
> op = par(mai = c(2,0.7,0.1,0.1))
> plot(jitter(as.integer(varGroupX)), sds, xaxt = "n", xlab = "", ylab = "")
> axis(1, las = 2, tick = FALSE, at = unique(varGroupX), labels = unique(varGroupX))
> par(op)
```

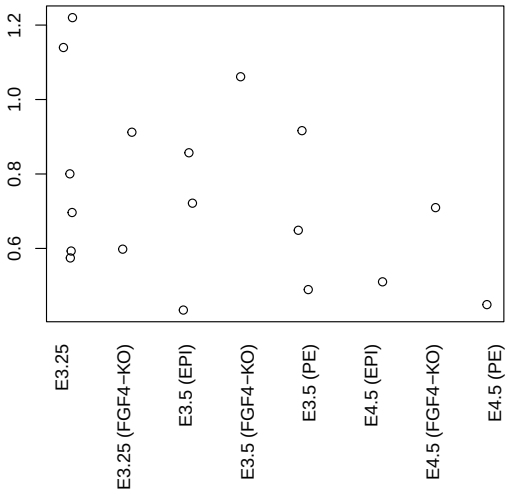


Figure 26: Variability of different groups of samples. The groups were defined by `sampleGroup` and `ScanDate` (see definition of `varGroups` above). Variability was measured by the *median* (across the 100 top variable genes) of the *median absolute deviation* across samples.

See Figure 26. Due to the small numbers of samples, it is difficult to decide whether or not batch effects play an important role for estimating variability. This view is corroborated by the diagnostic plots in the *quality assessment report* from the *arrayQualityMetrics* package.

So let us set aside the batch effect worries, and just compute and compare distributions of standard deviations[2].

```
> gps = split(seq_len(ncol(x)), f = x$sampleGroup)[c("E3.25", "E3.25 (FGF4-KO)")]
> sds = sapply(gps, function(j) apply(exprs(x)[sel, j], 1, mad))
> summary(sds)

     E3.25        E3.25 (FGF4-KO)
 Min.   :0.1210   Min.   :0.0894
 1st Qu.:0.7199   1st Qu.:0.4290
 Median :1.4571   Median :1.1348
 Mean   :1.6861   Mean   :1.5002
 3rd Qu.:2.5774   3rd Qu.:2.5293
```

[2]Since these standard deviations are computed on the logarithmic scale, they correspond to coefficient of variation on the not-log-transformed scale.

```
 Max.   :4.4275   Max.   :4.1889

> apply(sds, 2, function(x) c(`mean` = mean(x), `sd` = sd(x)))

        E3.25 E3.25 (FGF4-KO)
mean 1.686081        1.500179
sd   1.141133        1.177443
```

7.4 Do the FGF4-KO samples correspond to a particularly early substage within E3.25 (as indicated by the number of cells)?

See Figures 27 and 28, which are produced by the code below.

```
> for(n in c(100, 1000)) {
+   sel = order(rv325, decreasing = TRUE)[seq_len(n)]
+   KOmean = rowMeans(exprs(x325)[sel, x325$genotype=="FGF4-KO"])
+   dists = colSums((exprs(x325)[sel, wt325] - KOmean)^2)^0.5
+
+   pdf(file = sprintf("Hiiragi2013-figNumberOfCells-%d.pdf", n), width = 5, height = 10)
+   par(mfrow = c(2,1))
+   plot(x325$Total.number.of.cells[wt325], dists, pch = 16, main = "",
+        xlab = "Total number of cells", ylab = "Distance to FGF4-KO")
+   MDSplot(x325[sel, ], pointlabel = ifelse(x325$genotype=="WT",
+                                            paste(x325$Total.number.of.cells), "KO"), cex = 1)
+   dev.off()
+ }
```

7.5 Heatmap of E3.25 WT and E3.25 FGF-KO samples

For data visualisation, we produce a heatmap (Figure 29) that shows the data from the following groups

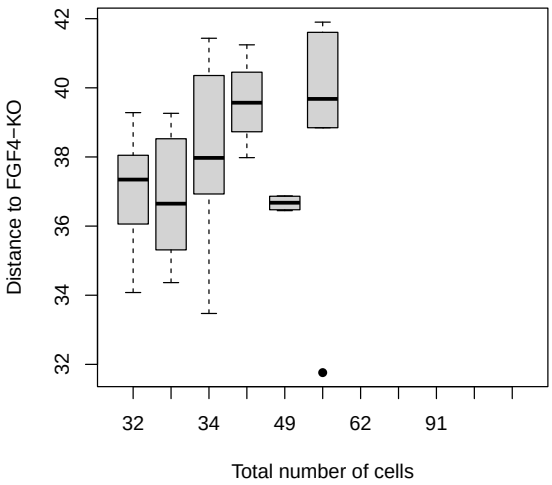
```
> selectedGroups = c("E3.25", "E3.25 (FGF4-KO)")
> xKO = x[, safeSelect(selectedGroups)]
> selectedFeatures = order(rowVars(exprs(xKO)), decreasing = TRUE)[seq_len(100)]
```

```
> myHeatmap(xKO[selectedFeatures, ], collapseDuplicateFeatures = TRUE, haveColDend = TRUE)
```

7.6 Differentially expressed genes between FGF4-KO and WT (PE, EPI) at E3.5

Let us compute the differentially expressed genes between WT and FGF4-KO,

```
> x35 = x[, safeSelect(c("E3.5 (FGF4-KO)", "E3.5 (EPI)", "E3.5 (PE)"))]
> f1 = f2 = x35$sampleGroup
> f1[f1=="E3.5 (PE)" ] = NA
> f2[f2=="E3.5 (EPI)"] = NA
> x35$EPI = factor(f1, levels = c("E3.5 (FGF4-KO)", "E3.5 (EPI)"))
> x35$PE  = factor(f2, levels = c("E3.5 (FGF4-KO)", "E3.5 (PE)"))
```



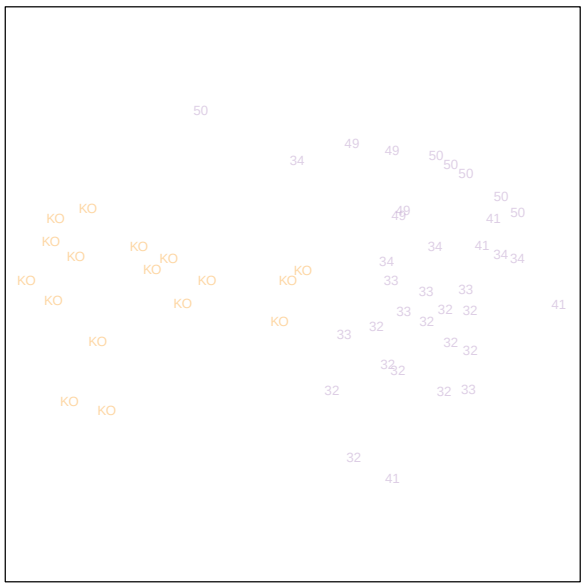


Figure 27: Distance of the E3.25 WT samples to the mean profile of FGF4-KO. The 100 features with highest overall variance were used.

```
> de = list(`EPI` = rowttests(x35, "EPI"),
+           `PE`  = rowttests(x35, "PE"))
> for(i in seq(along = de))
+   de[[i]]$p.adj = p.adjust(de[[i]]$p.value, method = "BH")
```

```
> par(mfcol = c(3,2))
> rkv = rank(-rowVars(exprs(x35)))
> fdrthresh = 0.05
> fcthresh =  1
```

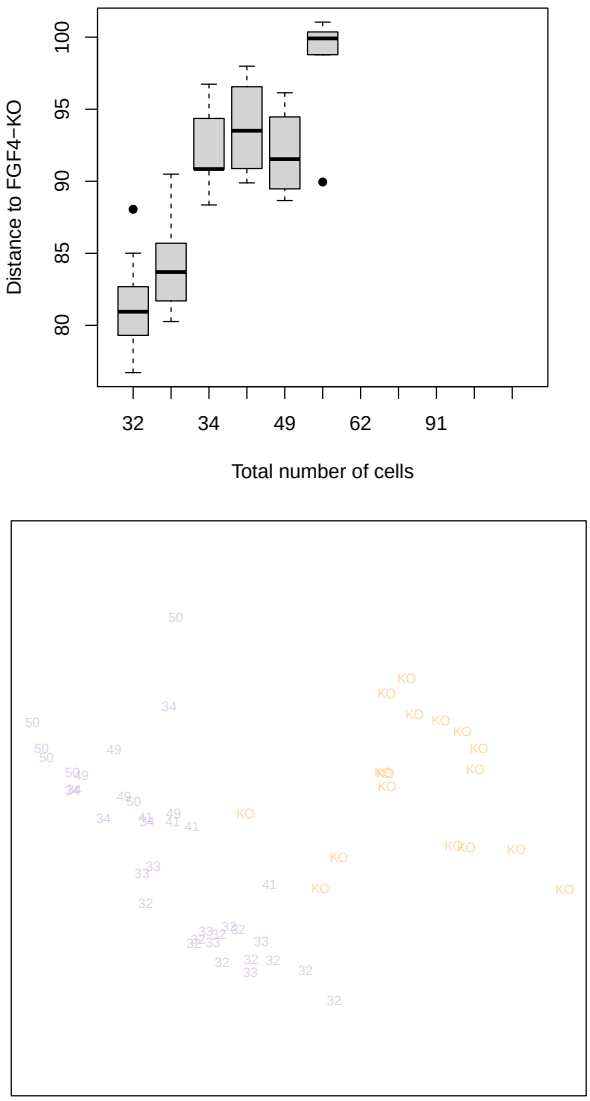


Figure 28: Distance of the E3.25 WT samples to the mean profile of FGF4-KO. The 1000 features with highest overall variance were used.

```
> for(i in seq(along = de)) {
+   hist(de[[i]]$p.value, breaks = 100, col = "lightblue", main = names(de)[i], xlab = "p")
+   plot(rkv, -log10(de[[i]]$p.value), pch = 16, cex = .25, main = "",
+        xlab = "rank of overall variance", ylab = expression(-log[10]~p))
+   plot(de[[i]]$dm, -log10(de[[i]]$p.value), pch = 16, cex = .25, main = "",
+       xlab = "average log fold change", ylab = expression(-log[10]~p))
+   abline(v = c(-1,1)*fcthresh[i], col = "red")
+ }
```

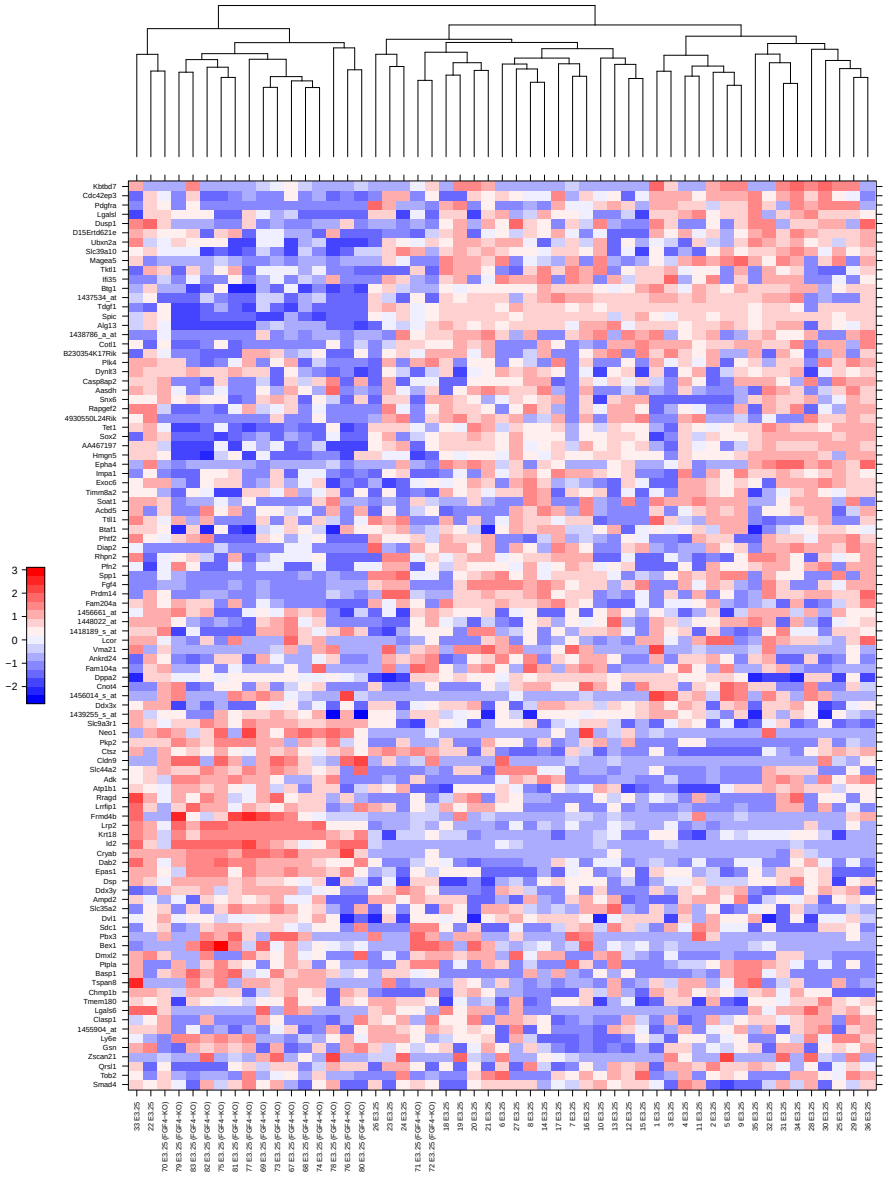


Figure 29: Heatmap of all E3.25 WT and E3.25 FGF-KO samples. The 100 features with highest overall variance were used. One of them shows Fgf4.

The plots are shown in Figure 30. In contrast to Figure 8, this plot indicates no obvious choice of threshold from overall-variance (i. e., independent) filtering. In fact, there seem to be many probes with apparently significant changes but very low average fold changes. These could be caused by "batch effects", and we will apply the fold change threshold `fcthresh` to remove these.

```
> isSig = ((pmin(de$PE$p.adj, de$EPI$p.adj) < fdrthresh) &
+          (pmax(abs(de$PE$dm), abs(de$EPI$dm)) > fcthresh))
> table(isSig)
```

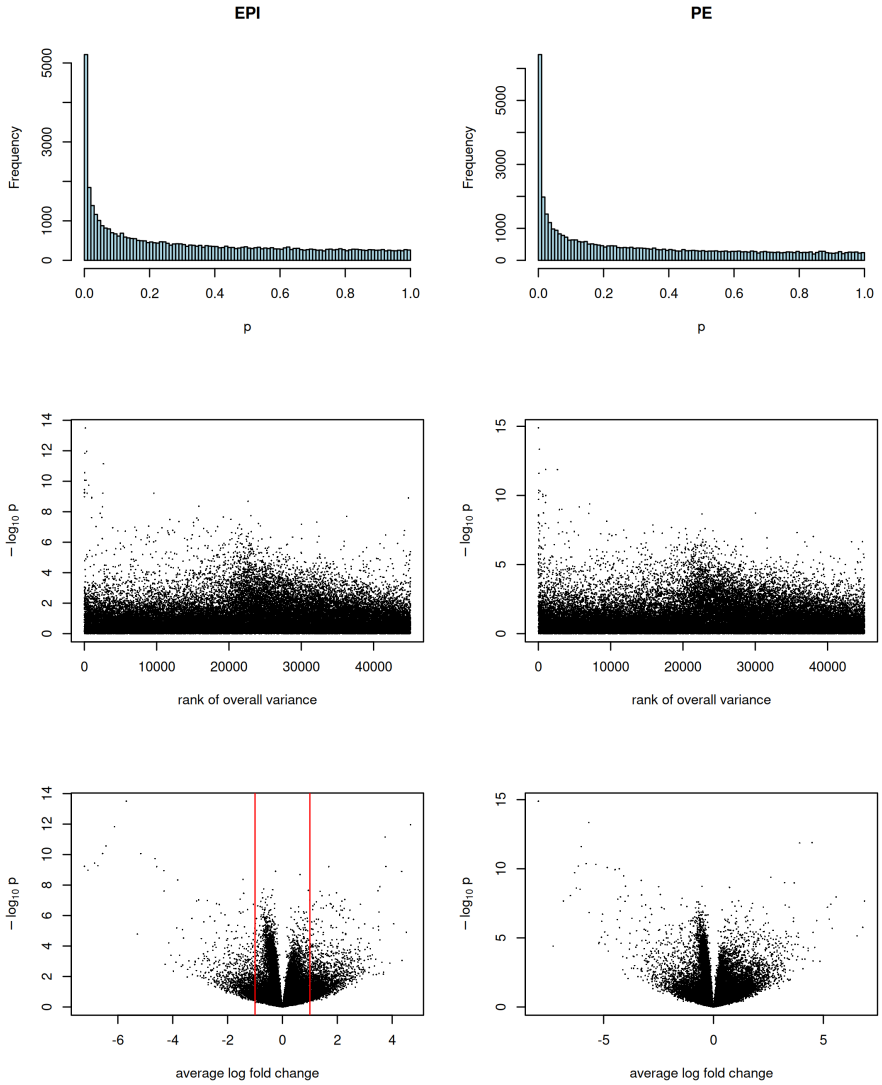


Figure 30: Differentially expressed genes between FGF4-KO and WT (EPI, PE) at E3.5. The upper panels show the histograms of p-values, middle panels the scatter plots of rank of overall variance (`rkv`) versus $-\log_{10} p$, lower panels log_2 fold-change versus $-\log_{10} p$ ("volcano plots"). The red lines indicate possibly useful fold change thresholds.

```
isSig
FALSE  TRUE
43686  1415

> plot(de$PE$dm, de$EPI$dm, pch = 16, asp = 1,
+      xlab = expression(log[2]~FGF4-KO / PE),
+      ylab = expression(log[2]~FGF4-KO / EPI),
+      cex  = ifelse(isSig, 0.6, 0.1), col = ifelse(isSig, "red", "grey"))
```

See Figure 31.

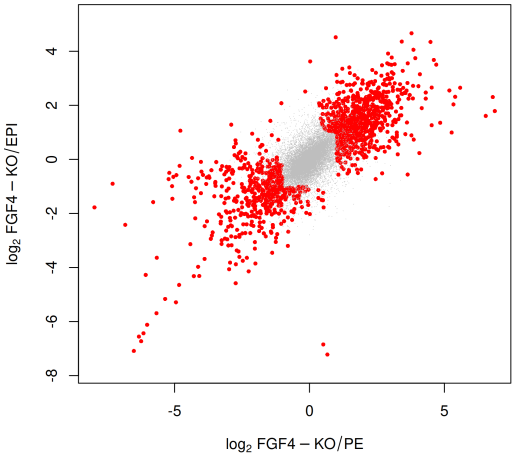


Figure 31: Differentially expressed genes between FGF4-KO and WT (EPI, PE) at E3.5. Shown is a scatterplot of the average fold changes for both comparisons. Large values along the x-axis indicate a relatively higher level of expression in E3.5 (FGF4-KO) compared to E3.5 (PE), small (i. e. negative) values indicate higher expression in E3.5 (PE); analogously for the y-axis.

```
> df = cbind(fData(x35)[isSig, ], `log2FC KO/PE` = de$PE$dm[isSig],
+            `log2FC KO/EPI` = de$EPI$dm[isSig])
> write.csv(df, file = "differentially_expressed_E3.5_vs_FGF4-KO.csv")
> ctrls = grep("^AFFX", rownames(df), value = TRUE)
```

7.6.1 The probes for FGF4

Let us look at the data for the four probes annotated to FGF4.

```
> par(mfrow = c(2,2))
> for(p in which(FGF4probes))
+   plot(exprs(x35)[p, ], type = "p", pch = 16, col = x35$sampleColour,
+        main = rownames(fData(x325))[p], ylab = expression(log[2]~expression))
```

```
> stopifnot(sum(FGF4probes)==4)
```

See Figure 32.

7.6.2 Behaviour of the control probes

A notable problem (probably associated with the "batch effects" discussed above) is that `length(ctrls)`=31 control probes appear as significant in this analysis. Let us plot the first 8 of them (see Figure 33).

```
> par(mfcol = c(4, 2))
> for(p in ctrls[1:8])
+   plot(exprs(x35)[p, ], type = "p", pch = 16, col = x35$sampleColour,
+        main = p, ylab = expression(log[2]~expression))
```

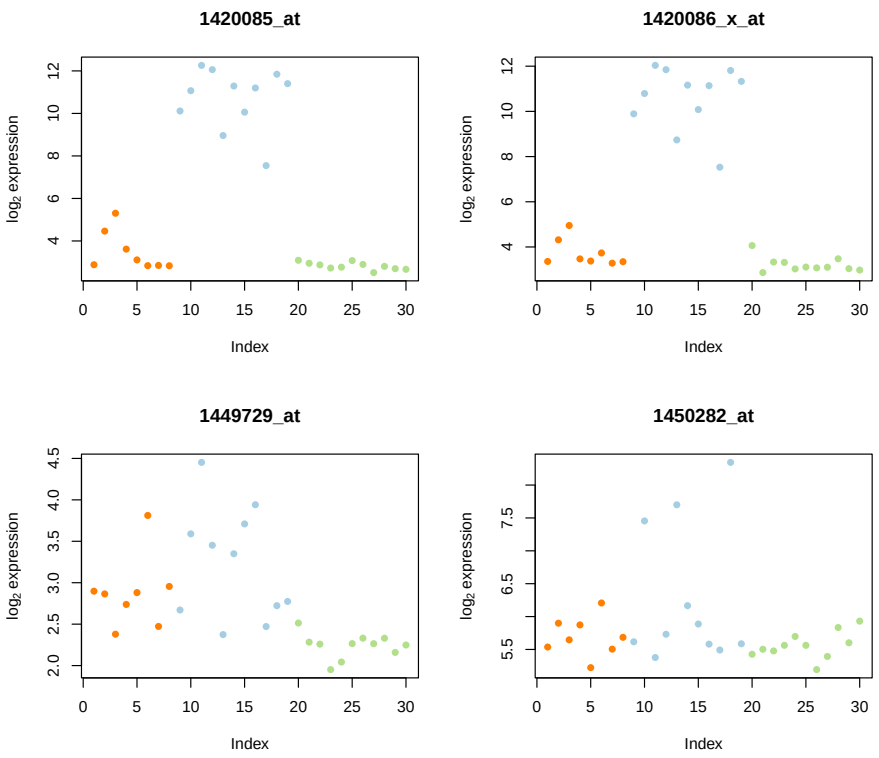


Figure 32: The probes for FGF4. As expected, the signal from these probes in the KO samples is consistent with background.

7.6.3 Gene set enrichment analysis

Kolmogorov-Smirnov tests against KEGG pathways.

```
> library(mouse4302.db)
> keggpw = as.list(mouse4302PATH2PROBE)
> dat  = de$PE$dm+de$EPI$dm
> fts = lapply(keggpw, function(ps) {
+   inpw = rownames(fData(x35)) %in% ps
+   ft   = ks.test( dat[inpw], dat[!inpw] )
+   list(p.value = ft$p.value, statistic = ft$statistic, n = sum(inpw))
+ })
```

First, let us select some prominent signalling pathways.

```
> pws = c("04010", "04070", "04150", "04310", "04330", "04340", "04350", "04370", "04630")
```

In addition, let us add those pathways with the strongest enrichment signal:

```
> pws = unique( c(pws, names(fts)[ sapply(fts, function(x)
+     with(x, (n<=100) && (p.value<0.01))) ] ) )
```

Make an online query at the KEGG database.

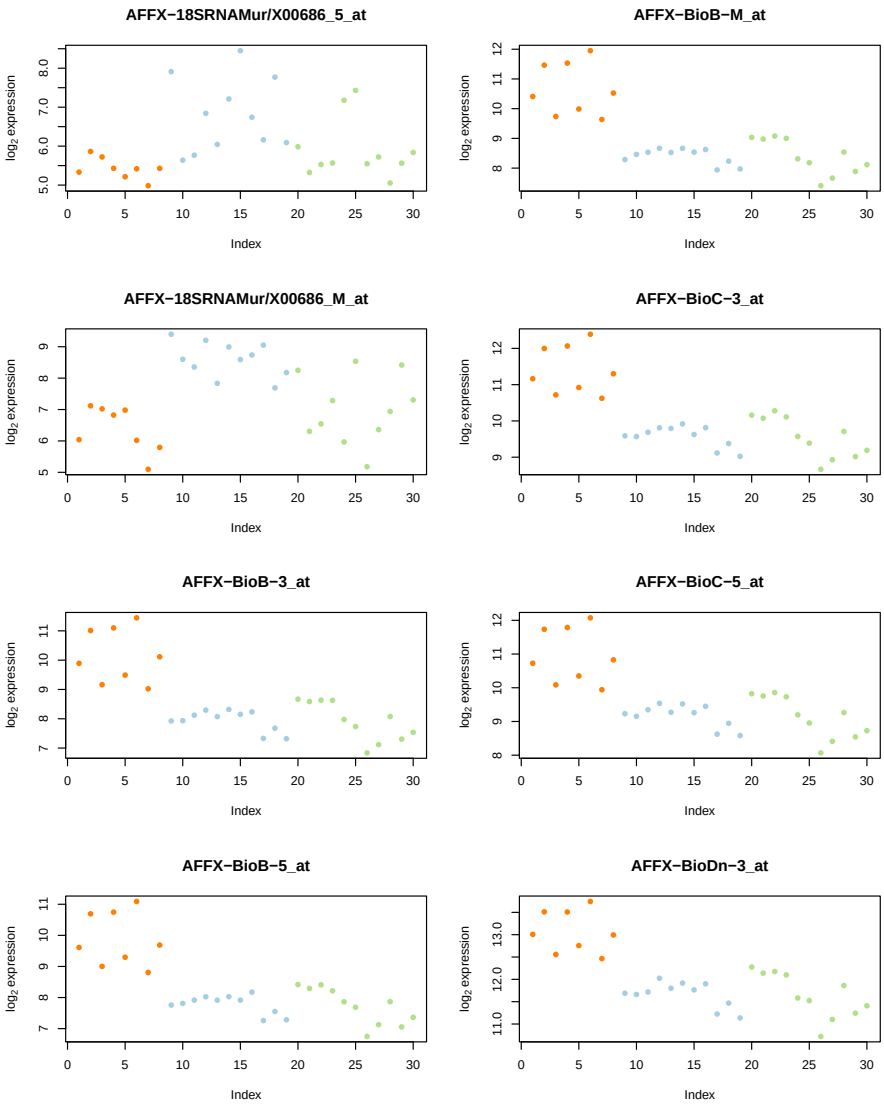


Figure 33: Behaviour of some control probe sets.

```
> library(KEGGREST)
> query  = paste0("mmu", pws)
> query  = split(query, seq(along = query) %/% 10)
> pwInfo = unlist(lapply(query, keggGet), recursive = FALSE)
```

```
> df = data.frame(
+   `id`   = pws,
+   `name` = sub(" - Mus musculus (mouse)", "", sapply(pwInfo, `[[`, "NAME"), fixed = TRUE),
+   `n`         =                    sapply(pws, function(x) fts[[x]]$n),
+   `p`         = as.character(signif(sapply(pws, function(x) fts[[x]]$p.value),  2)),
+   `statistic` = as.character(signif(sapply(pws, function(x) fts[[x]]$statistic), 2)),
+   stringsAsFactors = FALSE, check.names = FALSE)
```

```
> library(xtable)
> print(xtable(df,
+   caption = paste("Gene set enrichment analysis of selected KEGG pathways, for the",
+                   "differentially expressed genes between E3.5 FGF-4 KO and WT samples.",
+                   "n: number of microarray features annotated to genes in the pathway."),
+   label = "tab_KEGG", align = c("l", "l","p{4cm}","r","r","r")),
+   type = "latex", file = "tab_KEGG.tex")
```

Table 1 shows

1. first, the data for the manually selected signalling pathways,
2. then, the pathways with the most prominent changes.

We visualize their data in Figure 34, see code below.

```
> library(geneplotter)
> par(mfrow = c(7,4)); stopifnot(length(pws)<=42)
> for(i in seq(along = pws)) {
+   inpw = factor(ifelse(rownames(fData(x35)) %in% keggpw[[pws[i]]],"in pathway","outside"))
+   ord  = order(inpw)
+   enr  = paste0("p=", df$p[i], if(as.numeric(df$p[i])<0.05)
+                                  paste(" D=", df$statistic[i]) else "")
+   multiecdf( (de$PE$dm+de$EPI$dm) ~ inpw, xlim = c(-1,1)*3,
+             main = paste(df$name[i], enr, sep = "\n"),
+             xlab = "mean difference between FGF4-KO and wildtype samples" )
+ }
```

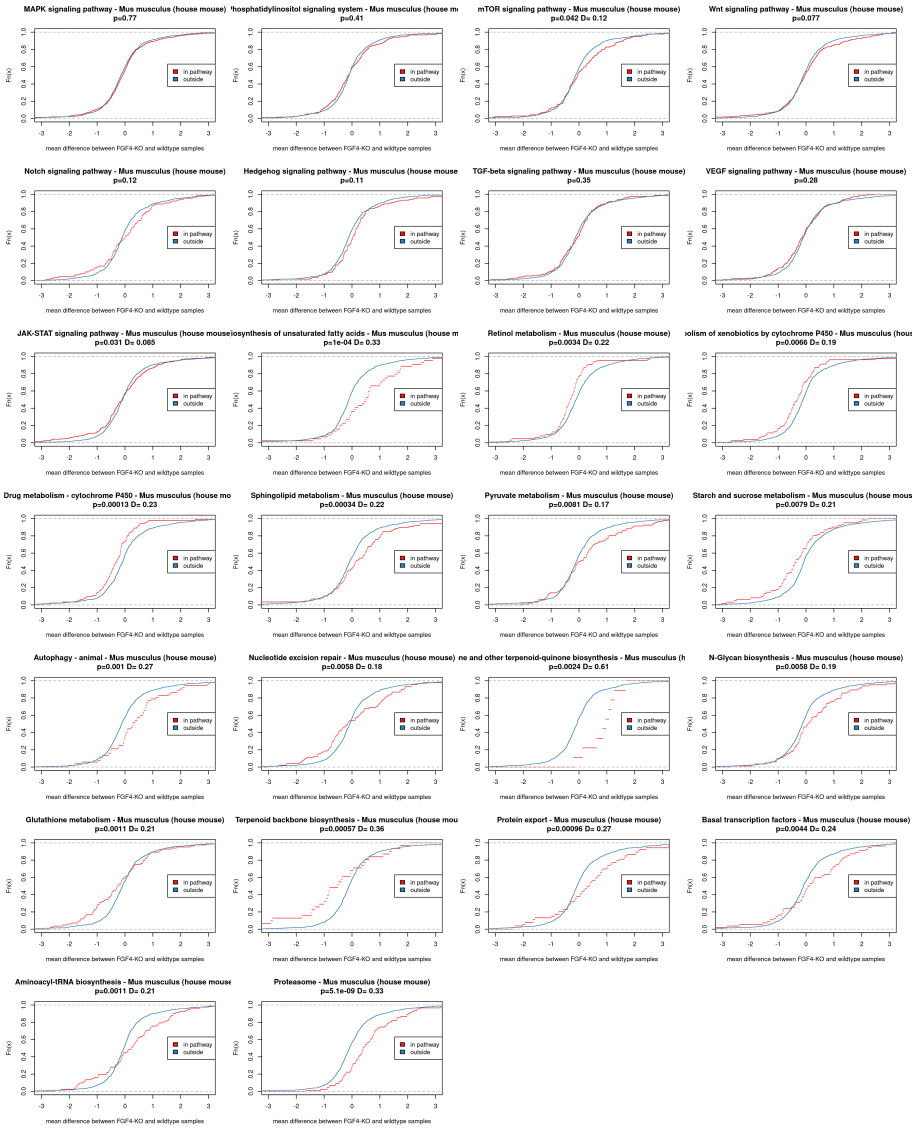


Figure 34: Differentially expressed genes between FGF4-KO and WT (EPI, PE) at E3.5.

	id	name	n	p	statistic
01	04010	MAPK signaling pathway - Mus musculus (house mouse)	628	0.77	0.027
02	04070	Phosphatidylinositol signaling system - Mus musculus (house mouse)	186	0.41	0.065
03	04150	mTOR signaling pathway - Mus musculus (house mouse)	139	0.042	0.12
04	04310	Wnt signaling pathway - Mus musculus (house mouse)	377	0.077	0.066
05	04330	Notch signaling pathway - Mus musculus (house mouse)	89	0.12	0.13
06	04340	Hedgehog signaling pathway - Mus musculus (house mouse)	117	0.11	0.11
07	04350	TGF-beta signaling pathway - Mus musculus (house mouse)	169	0.35	0.072
08	04370	VEGF signaling pathway - Mus musculus (house mouse)	176	0.28	0.075
09	04630	JAK-STAT signaling pathway - Mus musculus (house mouse)	288	0.031	0.085
11	01040	Biosynthesis of unsaturated fatty acids - Mus musculus (house mouse)	44	1e-04	0.33
12	00830	Retinol metabolism - Mus musculus (house mouse)	64	0.0034	0.22
13	00980	Metabolism of xenobiotics by cytochrome P450 - Mus musculus (house mouse)	79	0.0066	0.19
14	00982	Drug metabolism - cytochrome P450 - Mus musculus (house mouse)	91	0.00013	0.23
15	00600	Sphingolipid metabolism - Mus musculus (house mouse)	87	0.00034	0.22
16	00620	Pyruvate metabolism - Mus musculus (house mouse)	93	0.0081	0.17
17	00500	Starch and sucrose metabolism - Mus musculus (house mouse)	61	0.0079	0.21
18	04140	Autophagy - animal - Mus musculus (house mouse)	52	0.001	0.27
19	03420	Nucleotide excision repair - Mus musculus (house mouse)	91	0.0058	0.18
110	00130	Ubiquinone and other terpenoid-quinone biosynthesis - Mus musculus (house mouse)	9	0.0024	0.61
21	00510	N-Glycan biosynthesis - Mus musculus (house mouse)	84	0.0058	0.19
22	00480	Glutathione metabolism - Mus musculus (house	84	0.0011	0.21

8 Jensen-Shannon Divergence analysis

In the following, we will explore whether and how we can detect changes in the degree of "lack of correlation" ("heterogeneity") between cells. [6] considered a measure of Jensen-Shannon Divergence (Figures 2C, D in their paper) with a similar idea in mind. From their Supplement:

> JSD was calculated to assess within-group similarity of gene expression within each cell line according to Lin (1991). Expression values of genes were transformed so that they sum up to 1 in each cell. Each cell i is thus represented as a vector of probabilities P_i. Cells from the same line were grouped together and for each group, the JSD was calculated from the probability vectors $(P_1, P_2, \ldots, P_n)$ of cells in each group.
>
> $$JSD(P_1, \ldots, P_n) = H\left(\frac{1}{n}\sum_{i=1}^{n} P_i\right) - \frac{1}{n}\sum_{i=1}^{n} H(P_i) \quad \boxed{1}$$
>
> where $H(P)$ is the Shannon entropy given by
>
> $$H(P) = -\sum_{i=1}^{k} P(x_i)\log_2 P(x_i). \quad \boxed{2}$$
>
> Confidence intervals (CIs) were estimated by bootstrapping (sampling with replacement). The 95% CIs were shown as error bars.

Let's define R functions for this.

```
> H = function(p) -sum(p*log2(p))
> JSD = function(m, normalize = TRUE) {
+   stopifnot(is.matrix(m), all(dim(m)>1))
+   if(normalize)
+     m = m/rowSums(m)
+   H(colMeans(m)) - mean(apply(m, 2, H))
+ }
```

For comparison, we consider below also the ordinary within-group standard deviation:

```
> SDV = function(m) sqrt(mean(rowVars(m)))
```

This measure of divergence computes for each gene (feature on the array) the variance across arrays, then computes the average of these and takes the square root.

Since the majority of the 45101 features on the array are dominated "noise" and/or potential technical drifts, we want to use only the most highly variable (i. e. most informative) features,

```
> numberFeatures = c(50, 200, 1000, 4000)
```

as selected by overall variance. In the following, we will use the function `computeDivergences` to compute a measure of divergence, such as `JSD`, for each of the groups defined by `strata` and for the different choices of `numberFeatures`.

```
> computeDivergences = function(y, indices, fun) {
+   y = y[indices, ]
+   exprVals = y[, -1]
```

```
+   strata = y[, 1]
+   numStrata = max(strata)
+   stopifnot(setequal(strata, seq_len(numStrata)))
+
+   orderedFeatures = order(rowVars(t(exprVals)), decreasing = TRUE)
+   sapply(numberFeatures, function(n) {
+     selFeat = orderedFeatures[seq_len(n)]
+     sapply(seq_len(numStrata), function(s)
+              fun(exprVals[strata==s, selFeat, drop = FALSE]))
+   })
+ }
```

```
> x$sampleGroup = factor(x$sampleGroup)
> x$strata = as.numeric(x$sampleGroup)
```

Now we are ready to go, we use the bootstrap to assess variability in our divergene estimates:

```
> library(boot)
> bootJSD = boot(data = cbind(x$strata, t(exprs(x))),
+                statistic = function(...) computeDivergences(..., fun = JSD),
+                R = 100, strata = x$strata)
> dim(bootJSD$t) = c(dim(bootJSD$t)[1], dim(bootJSD$t0))
```

```
> bootSDV = boot(data = cbind(x$strata, t(exprs(x))),
+                statistic = function(...) computeDivergences(..., fun = SDV),
+                R = 100, strata = x$strata)
> dim(bootSDV$t) = c(dim(bootSDV$t)[1], dim(bootSDV$t0))
```

The resulting values (mean and distribution indicated by boxplot) are shown in Figure 35.

```
> par(mfrow = c(length(numberFeatures),2))
> colores = sampleColourMap[levels(x$sampleGroup)]
> for(i in seq(along = numberFeatures)) {
+   for(what in c("JSD", "SDV")){
+     obj = get(paste("boot", what, sep = ""))
+     boxplot(obj$t[, , i], main = sprintf("numberFeatures=%d", numberFeatures[i]),
+             col = colores, border = colores, ylab = what, xaxtb = "n")
+     px = seq_len(ncol(obj$t))
+     text(x = px, y = par("usr")[3], labels = levels(x$sampleGroup),
+          xpd = NA, srt = 45, adj = c(1,0.5), col = colores)
+     points(px, obj$t0[, i], pch = 16)
+   }
+ }
```

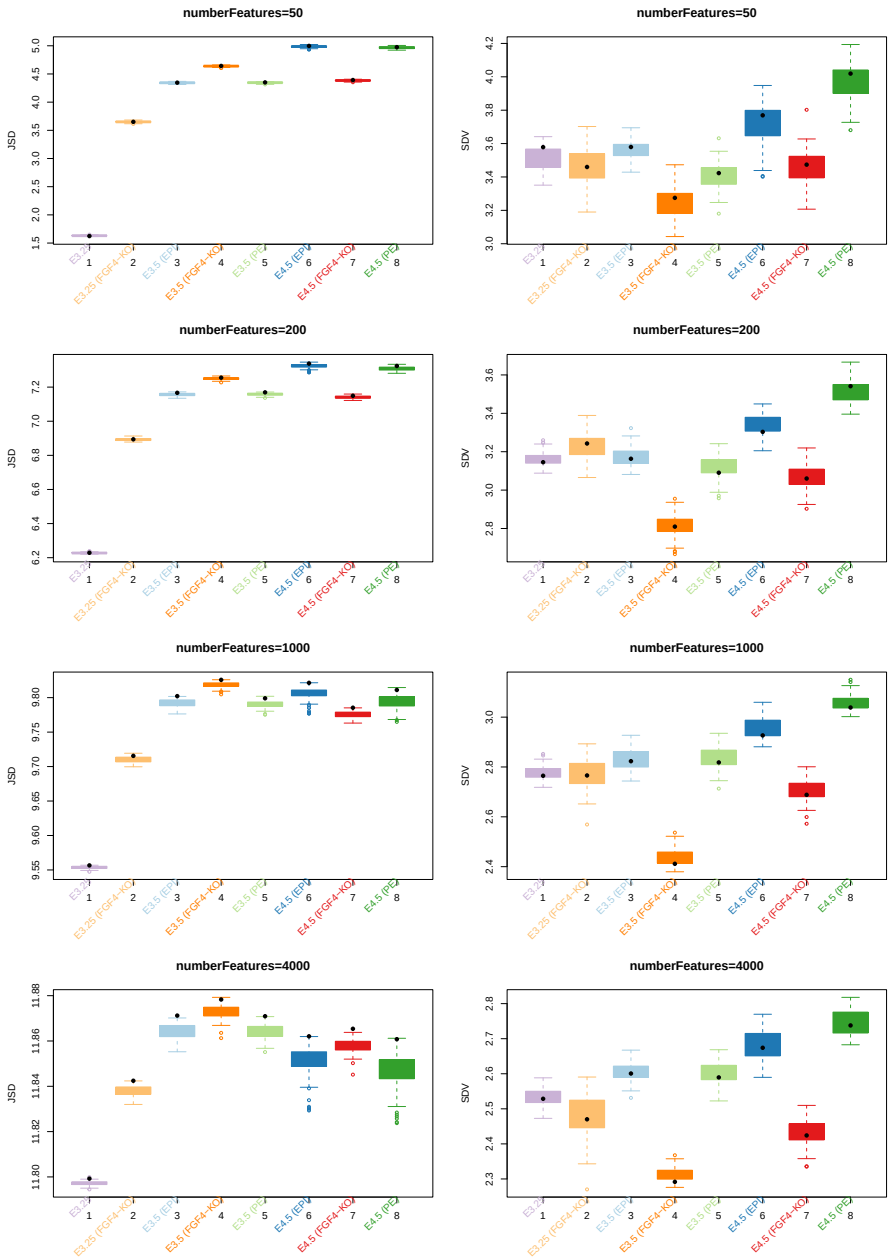


Figure 35: Jensen-Shannon divergences. JSD is shown on the left; on the right, for comparison we performed the same analysis with *within-group standard deviation* as another measure of group spread.

9 Classification of temporal profiles

9.1 Comparison of microarray data with qPCR results

First, let us plot the microarray data for all probesets annotated to the genes shown in Figure 36. We write the output into an extra PDF file, `exemplaryGenes.pdf`.

```
> xs = x[, pData(x)$sampleGroup %in%
+    c("E3.25", "E3.5 (PE)", "E4.5 (PE)", "E3.5 (EPI)", "E4.5 (EPI)")]
> xs$sampleGroup = factor(xs$sampleGroup)
```

```
> myBoxplot = function(ps) {
+   fac = factor(pData(xs)$sampleGroup)
+   boxplot(exprs(xs)[ps, ] ~ fac, col = sampleColourMap[levels(fac)], lim = c(2,14),
+           main = sprintf("%s (%s)", fData(x)[ps, "symbol"], ps), show.names = FALSE)
+   text(seq(along = levels(fac)), par("usr")[3] - diff(par("usr")[3:4])*0.02,
+        levels(fac), xpd = NA, srt = 90, adj = c(1,0.5), cex = 0.8)
+ }
```

```
> exemplaryGenes = read.table(header = TRUE, stringsAsFactors = FALSE, text = "
+ symbol   thclass  probeset
+ Gata6      C-1 1425464_at
+ Tom1l1     C-1 1439337_at
+ Sox2       C-1 1416967_at
+ Pdgfra    PE-2 1438946_at
+ Sox17     PE-2 1429177_x_at
+ P4ha2     PE-2 1417149_at
+ Gata4     PE-3 1418863_at
+ Aldh18a1  PE-3 1415836_at
+ Col4a1    PE-3 1452035_at
+ Col4a2    PE-3 1424051_at
+ Cubn      PE-3 1426990_at
+ Lamb1     PE-3 1424113_at
+ Dab2      PE-4 1423805_at
+ Lrp2      PE-4 1427133_s_at
+ Amn       PE-4 1417920_at
+ Fgf4     EPI-2 1420085_at
+ Nanog    EPI-2 1429388_at
+ Tdgf1    EPI-2 1450989_at
+ Cldn4    EPI-4 1418283_at
+ Enox1    EPI-4 1436799_at")
> pdf(file = "exemplaryGenes.pdf", width = 8, height = 11)
> par(mfrow = c(5,3))
> for(i in seq_len(nrow(exemplaryGenes))) {
+   wh = featureNames(x)[fData(x)[, "symbol"]==exemplaryGenes[i, "symbol"]]
+   stopifnot(length(wh)>=1)
+   sapply(wh, myBoxplot)
+ }
> dev.off()
```

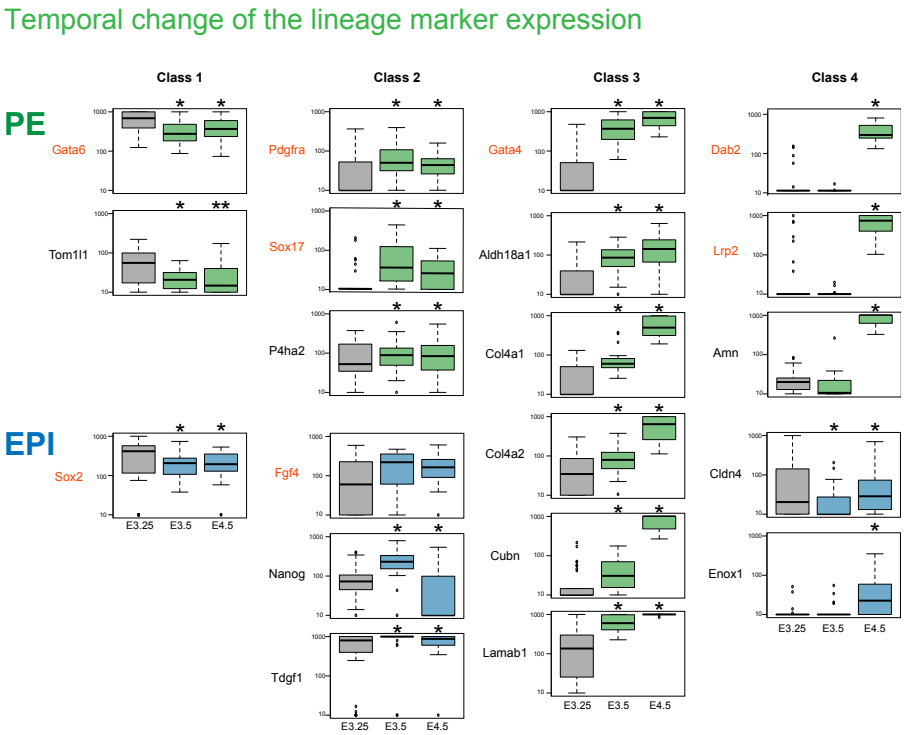


Figure 36: Temporal change of the lineage marker expression Example figure provided by Takashi in Email of 9 Oct 2012. Red genes are known lineage markers.

Based on these plots, we assigned one "trustworthy" probeset to each gene, indicated in the above table.

```
> layout(rbind(c(1, 4,   7, 13),
+              c(2, 5,   8, 14),
+             c(21, 6,   9, 15),
+              c(3, 16, 10, 19),
+             c(22, 17, 11, 20),
+             c(23, 18, 12, 24)))
> xs = x
> xs$sampleGroup = factor(xs$sampleGroup)
> xs$sampleGroup = relevel(xs$sampleGroup, "E4.5 (PE)")
> xs$sampleGroup = relevel(xs$sampleGroup, "E4.5 (EPI)")
> xs$sampleGroup = relevel(xs$sampleGroup, "E3.5 (PE)")
> xs$sampleGroup = relevel(xs$sampleGroup, "E3.5 (EPI)")
> xs$sampleGroup = relevel(xs$sampleGroup, "E3.25")
> for(i in seq_len(nrow(exemplaryGenes)))
+   myBoxplot(exemplaryGenes[i, "probeset"])
```

The output is shown in Figure 37, which we can compare to Figure 36.

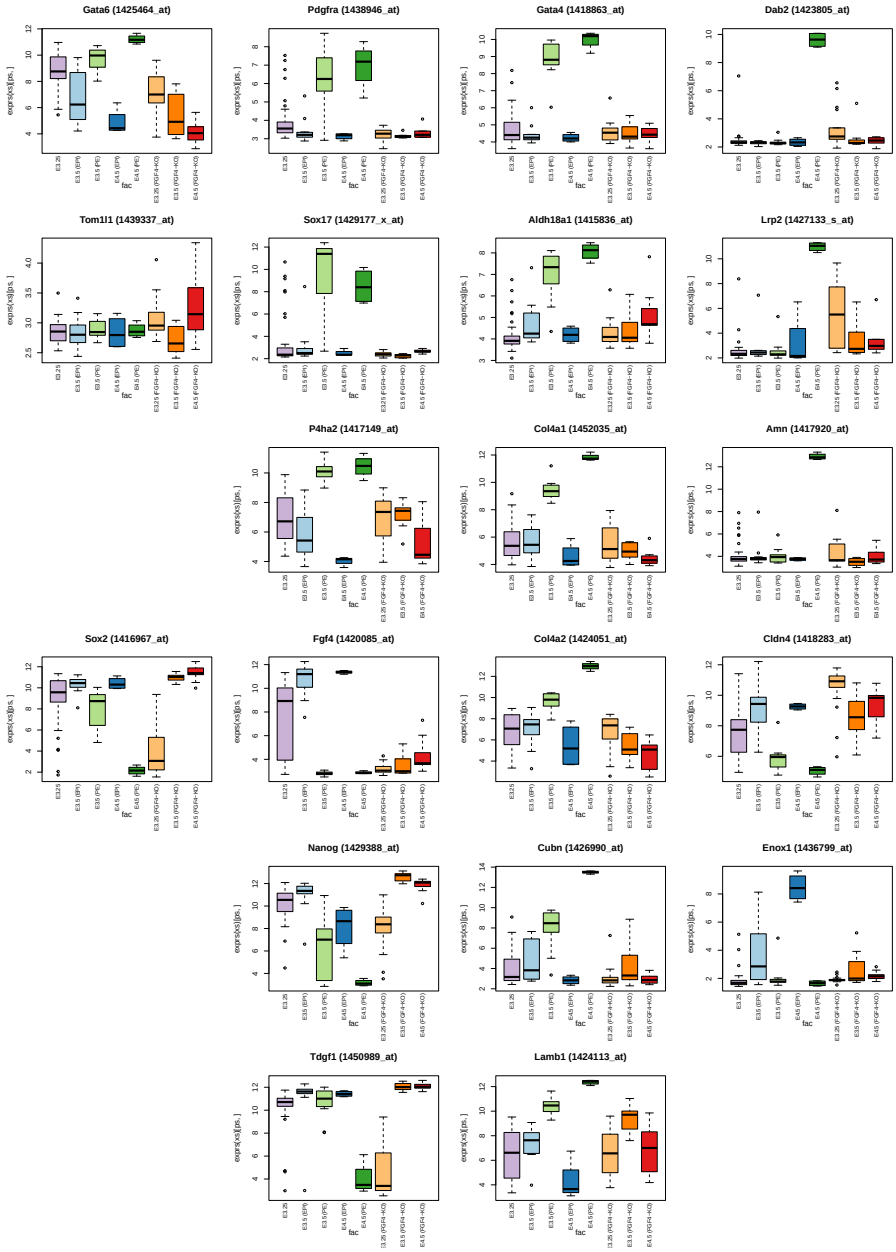


Figure 37: Boxplots of the microarray expression data for exemplary genes. Compare this to Figure 36. The agreement is satisfactory.

9.2 Rule-based classification

For EPI and PE separately, we define 4 classes as indicated in Figure 36:

- class 1: highest in E3.25
- class 2: highest in E3.5 and off in the other lineage (at E3.5, E4.5)
- class 3: E3.25 $<$ E3.5 $<$ E4.5 and off in the other lineage (at E4.5)

- class 4: off in E3.25 and E3.5, on in E4.5, and off in the other lineage (at E4.5)

To implement these rules, let us define the helper function `greater`, which determines if a gene has an average log fold change larger than `thresh` between each of the conditions in `cond1` versus each of the conditions in `cond2`.

```
> greater = function(cond1, cond2, thresh = 2) {
+   stopifnot(all(cond1 %in% names(groups)), all(cond2 %in% names(groups)))
+   rm1 = lapply(groups[cond1], function(j) rowMeans(exprs(x)[, j]))
+   rm2 = lapply(groups[cond2], function(j) rowMeans(exprs(x)[, j]))
+   res = rep(TRUE, nrow(x))
+   for(v1 in rm1) for(v2 in rm2)
+       res = res & ((v1-v2) > thresh)
+   return(res)
+ }
```

Next, we compute `xsc`, a version of the data matrix `x` that has been scaled to the range $[0, 1]$, such that *for each gene*, the lowest value across arrays corresponds to 0, and the highest value to 1. (In the code below, we actually use 0.025% and 97.5% quantiles instead of lowest and highest values, that is just to ensure some statistical robustness.)

```
> xquantiles = apply(exprs(x), 1, quantile, probs = c(0.025, 0.975))
> minLevel = xquantiles[1, ]
> maxLevel = xquantiles[2, ]
> trsf2Zero2One = function(x) {
+   x = (x-minLevel)/(maxLevel-minLevel)
+   x[x<0] = 0
+   x[x>1] = 1
+   return(x)
+ }
> xsc = x
> exprs(xsc) = trsf2Zero2One(exprs(x))
```

We will use `xsc` for the heatmap (Figure 38). The function `isOff` determines whether a gene is off by checking whether more than 90% of the arrays in the condition(s) specified by `cond` have values below `minLevel` + `thresh`.

```
> isOff = function(cond, thresh = 2) {
+   stopifnot(all(cond %in% names(groups)))
+   samps = unlist(groups[cond])
+   rowSums(exprs(x)[, samps] > minLevel+thresh) <= ceiling(length(samps) * 0.1)
+ }
```

Now we can do the classification.

```
>   `C-1` =  greater("E3.25", c("E3.5 (EPI)", "E4.5 (EPI)", "E3.5 (PE)", "E4.5 (PE)"))
> `EPI-2` =  greater("E3.5 (EPI)", c("E3.25")) &
+            greater("E3.5 (EPI)", "E4.5 (EPI)", thresh = 0.5) &
+            isOff("E3.5 (PE)")  & isOff("E4.5 (PE)" )
>  `PE-2` =  greater("E3.5 (PE)",  c("E3.25")) &
+            greater("E3.5 (PE)",  "E4.5 (PE)", thresh = 0.5)  &
+            isOff("E3.5 (EPI)") & isOff("E4.5 (EPI)")
> `EPI-4` = (greater("E4.5 (EPI)", c("E3.25", "E3.5 (EPI)")) &
```

```
+            isOff(c("E3.25", "E3.5 (EPI)")) &
+            isOff("E4.5 (PE)" ))
>  `PE-4` = (greater("E4.5 (PE)", c("E3.25", "E3.5 (PE)")) &
+            isOff(c("E3.25", "E3.5 (PE)")) &
+            isOff("E4.5 (EPI)"))
> `EPI-3` = (greater("E3.5 (EPI)", "E3.25", thresh = 0.5) &
+            greater("E4.5 (EPI)", "E3.5 (EPI)", thresh = 0.5) &
+            greater("E4.5 (EPI)", "E3.25", thresh = 3)   &
+            isOff("E4.5 (PE)") & !`EPI-4`)
>  `PE-3` = (greater("E3.5 (PE)",  "E3.25", thresh = 0.5) &
+            greater("E4.5 (PE)",  "E3.5 (PE)",  thresh = 0.5) &
+            greater("E4.5 (PE)",  "E3.25", thresh = 3) &
+            isOff("E4.5 (EPI)") & !`PE-4`)
> thclasses = cbind(`C-1`, `EPI-2`, `EPI-3`, `EPI-4`, `PE-2`,  `PE-3`,  `PE-4`)
```

We want each feature to be in exactly one group:

```
> multiclass = thclasses[rowSums(thclasses)>1,, drop = FALSE]
> stopifnot(nrow(multiclass)==0)
```

The group sizes:

```
> colSums(thclasses)

  C-1 EPI-2 EPI-3 EPI-4  PE-2  PE-3  PE-4
   18    10    51   186     9    68   201
```

```
> agr = groups[c("E3.25", "E3.5 (EPI)", "E4.5 (EPI)", "E3.5 (PE)", "E4.5 (PE)")]
> fgr = apply(thclasses, 2, which)
> xsub = xsc[unlist(fgr), unlist(agr)]
> mat = exprs(xsub)
> rownames(mat) = fData(xsub)[, "symbol"]
> myHeatmap2(mat, keeprownames = TRUE,
+            rowGroups = factor(rep(seq(along = fgr), listLen(fgr))),
+            colGroups = factor(rep(seq(along = agr), listLen(agr))))
```

The heatmap is shown in Figure 38.

9.2.1 Table export

```
> out = do.call(rbind, lapply(seq(along = fgr), function(i)
+     cbind(`class` = names(fgr)[i], fData(xsc)[fgr[[i]], ])))
> write.csv(out, file = "featureclassification.csv")
```

9.2.2 Comparison with manual classification

Figure 2b in the paper shows qPCR data for these seven genes:

```
> fig2bGenes = c("Aldh18a1", "Col4a1", "Cubn", "Foxq1", "Gata4", "Lamb1", "Serpinh1")
```

The genes classified as PE-3 in the above microarray data rule-based classification are

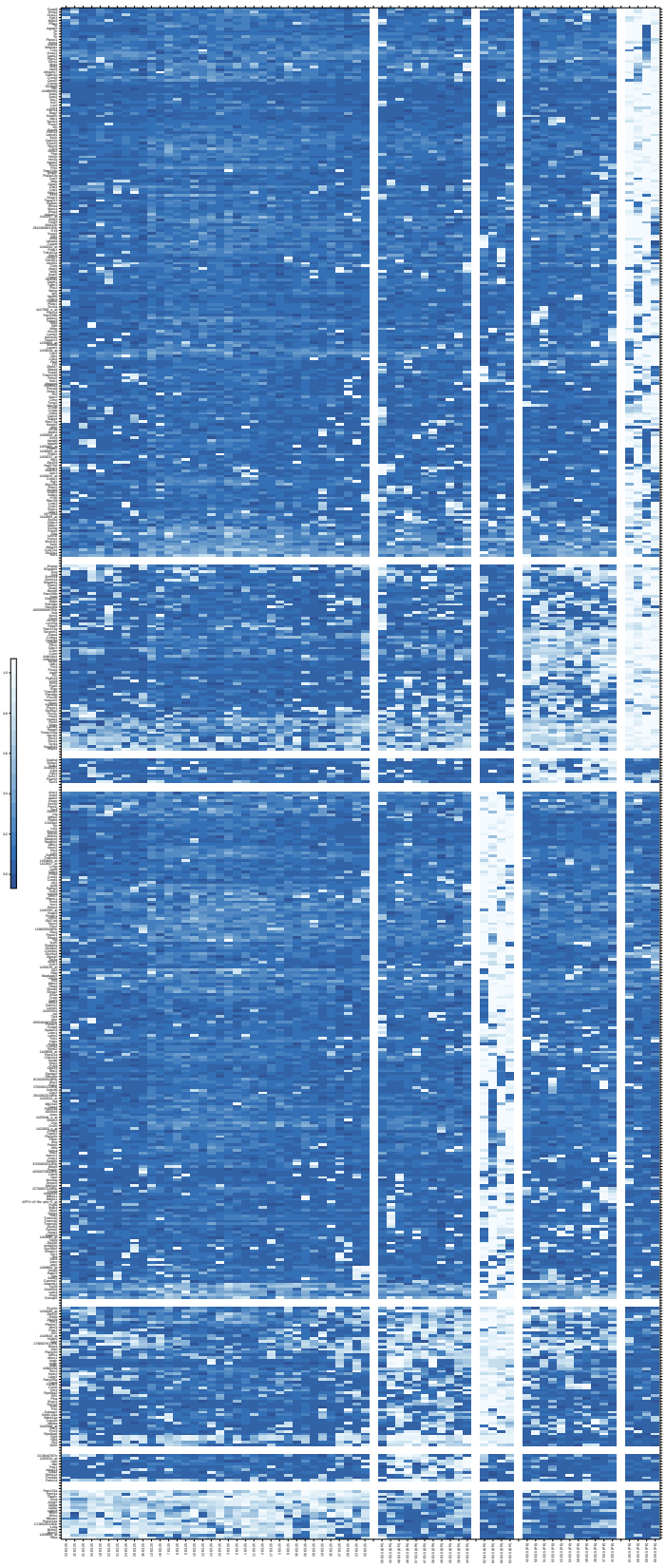


Figure 38: Heatmap of all classes of the [0,1]-scaled data matrix `xsc`. From left to right, the 5 blocks correspond to the conditions E3.25, E3.5 (EPI), E4.5 (EPI), E3.5 (PE), E4.5 (PE). From bottom to top, the 7 blocks correspond to the feature (gene) classes C-1, EPI-2, EPI-3, EPI-4, PE-2, PE-3, PE-4. Within classes, genes are sorted by dendrogram clustering.

```
> rbGenes = sort(unique(fData(x)$symbol[thclasses[, "PE-3"]]))
> rbGenes

 [1] "4930506M07Rik" "Ada"           "Aldh18a1"      "Apom"
 [5] "Aqp8"          "Asph"          "B3galnt1"      "B4galnt1"
 [9] "Bend5"         "Bmp6"          "Clcn5"         "Col4a1"
[13] "Creb3l2"       "Cubn"          "Dkk1"          "Dnajc22"
[17] "Elovl1"        "Enpep"         "F2r"           "Fam198b"
[21] "Fam213a"       "Fam46a"        "Flrt3"         "Foxq1"
[25] "Gata4"         "Gdpd5"         "Glipr2"        "Gstm2"
[29] "Gyg"           "Herpud1"       "Hpcal1"        "Hs3st1"
[33] "Ikbkb"         "Lamb1"         "Lrpap1"        "Lrrc16a"
[37] "Mogs"          "Neo1"          "Pcdh19"        "Pcyox1"
[41] "Pdcd6ip"       "Pde4dip"       "Pdzd3"         "Pga5"
[45] "Pip4k2a"       "Plod2"         "Plod3"         "Prrc2c"
[49] "Prss35"        "Pth1r"         "Rcn1"          "Reep5"
[53] "Rhpn2"         "Serpine2"      "Serpinh1"      "Slc26a2"
[57] "Smim14"        "Soat1"         "Srgn"          "Synj1"
[61] "Tcf19"         "Tmem150a"      "Tmem98"
```

Difference:

```
> setdiff(fig2bGenes, rbGenes)

character(0)
```

10 qPCR data analysis

10.1 Heatmaps for all data in `xq`

We first load the data

```
> data("xq")
```

```
> xq$sampleColours = sampleColourMap[sub("[[:digit:]]+ ", "", sampleNames(xq))]
> stopifnot(!any(is.na(xq$sampleColours)))
```

Next, we define a function `myHeatmap3` to create heatmaps

```
> myHeatmap3 = function(x, log = TRUE,
+   col = colorRampPalette(grDevices::hcl.colors(9, "Blues"))(ncol), ncol = 100, ...) {
+   mat = exprs(x)
+   if(log){
+     mat[mat<1] = 1
+     mat = log10(mat)
+   }
+   rownames(mat) = fData(x)[, "symbol"]
+   gplots::heatmap.2(mat, trace = "none", dendrogram = "none", scale = "none", col = col,
+             keysize = 0.9, ColSideColors = x$sampleColour, margins = c(5,7), ...)
+ }
```

and use it to visualize data in `xq`.

```
> myHeatmap3(xq)
```

See Figure 39.

10.2 Heatmaps for the seven selected genes and selected samples

```
> selectedGenes = c("Col4a1", "Lamab1", "Cubn", "Gata4", "Serpinh1", "Foxq1", "Aldh18a1")
> myHeatmap3(xq[selectedGenes, ])
```

See Figure 40.

```
> selectedSamples = (xq$Cell.type %in% c("ICM", "PE"))
> table(xq$sampleGroup[selectedSamples])

    E3.25 E3.5 (PE) E4.5 (PE)
       33        22        31

> sxq = xq[selectedGenes, selectedSamples]
> groups = c("E3.25", "E3.5 (PE)", "E4.5 (PE)")
> sxq$fsampleGroup = factor(sxq$sampleGroup, levels = groups)
> stopifnot(!any(is.na(sxq$fsampleGroup)))
```

```
> myHeatmap3(sxq)
```

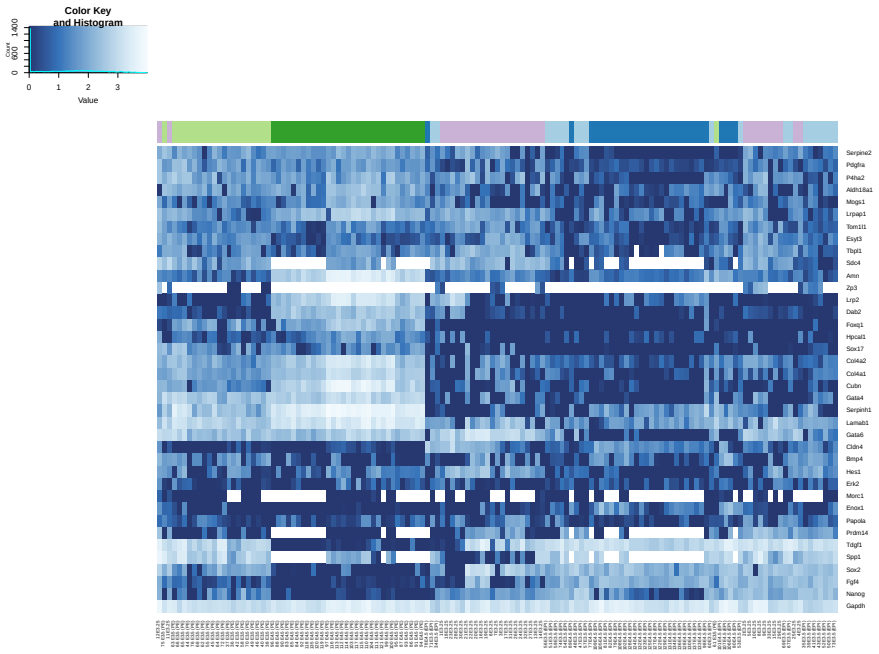


Figure 39: Heatmap of the qPCR data set. The data are shown on the logarithmic (base 10) scale. Before transformation, values < 1 were set to 1. The colour code of the bar at the top is as in Figure 15.

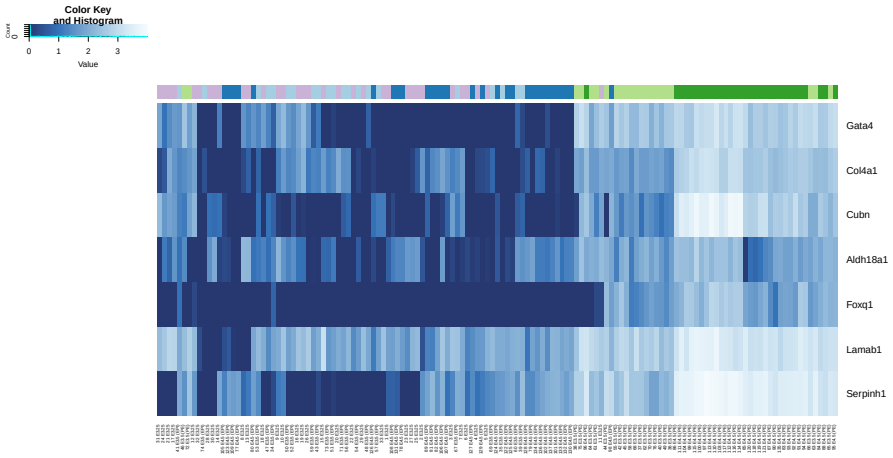


Figure 40: Heatmap of the qPCR data for the 7 selected genes.

See Figure 41.

10.3 Distribution of the data and discretisation

The qPCR expression levels in `sxq` are compared, separately for each gene, against the midpoint of the means of successive groups (E3.25, E3.5 (PE), E4.5 (PE)). This is illustrated in Figure 42.

```
> groupmedians = t(apply(exprs(sxq), 1, function(v) tapply(v, sxq$sampleGroup, median)))
> stopifnot(identical(colnames(groupmedians), groups), length(groups)==3)
```

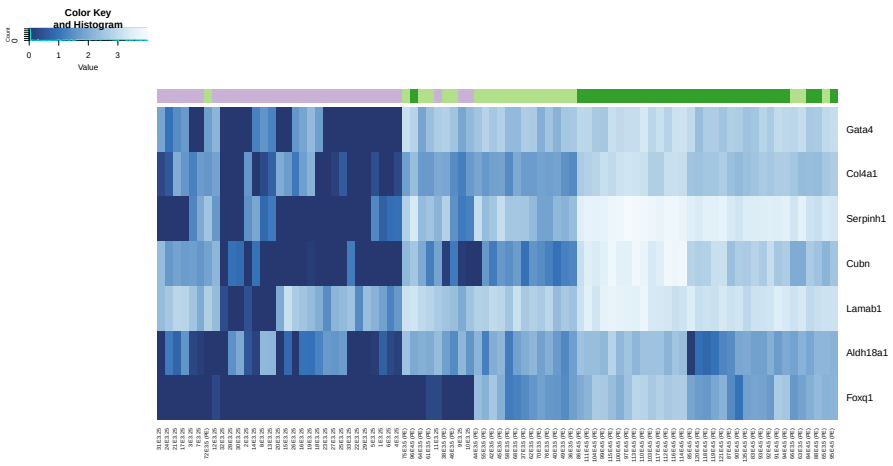


Figure 41: Heatmap for the 7 selected genes and the E3.25/E3.5/E4.5 PE samples. Rows and columns are ordered by clustering dendrogram.

```
> discrthreshs = (groupmedians[, 2:3] + groupmedians[, 1:2]) / 2
> stst = sapply(split(seq(along = sxq$sampleGroup), sxq$sampleGroup),
+     function(v) {i1 = head(v,1); i2 = tail(v,1); stopifnot(identical(v, i1:i2)); c(i1,i2)})
```

```
> groupmedians

                E3.25 E3.5 (PE) E4.5 (PE)
Col4a1   3.927113e+00  59.74835  496.3785
Lamab1   1.357524e+02 600.90534 1798.2736
Cubn     3.998510e-01  30.70686 1193.0876
Gata4    1.000891e+00 368.04084  696.1946
Serpinh1 5.538977e-03 327.21530 3899.7357
Foxq1    2.356829e-01  42.22368  150.9035
Aldh18a1 6.025723e+00  87.25491  142.3794

> discrthreshs

         E3.5 (PE)  E4.5 (PE)
Col4a1    31.83773  278.06342
Lamab1   368.32889 1199.58945
Cubn      15.55335  611.89724
Gata4    184.52087  532.11771
Serpinh1 163.61042 2113.47551
Foxq1     21.22968   96.56359
Aldh18a1  46.64032  114.81716
```

```
> op = par(mfrow = c(nrow(sxq), 1), mai = c(0.1, 0.7, 0.01, 0.01))
> for(j in seq_len(nrow(sxq))) {
+   plot(exprs(sxq)[j, ], type = "n", xaxt = "n", ylab = fData(sxq)$symbol[j])
+   segments(x0 = stst[1, ], x1 = stst[2, ], y0 = groupmedians[j, ], col = "#808080", lty=3)
+   segments(x0 = stst[1,1:2], x1 = stst[2,2:3], y0 = discrthreshs[j, ], col = "#404040")
+   points(exprs(sxq)[j, ], pch = 16, col = sxq$sampleColour)
+ }
> par(op)
```

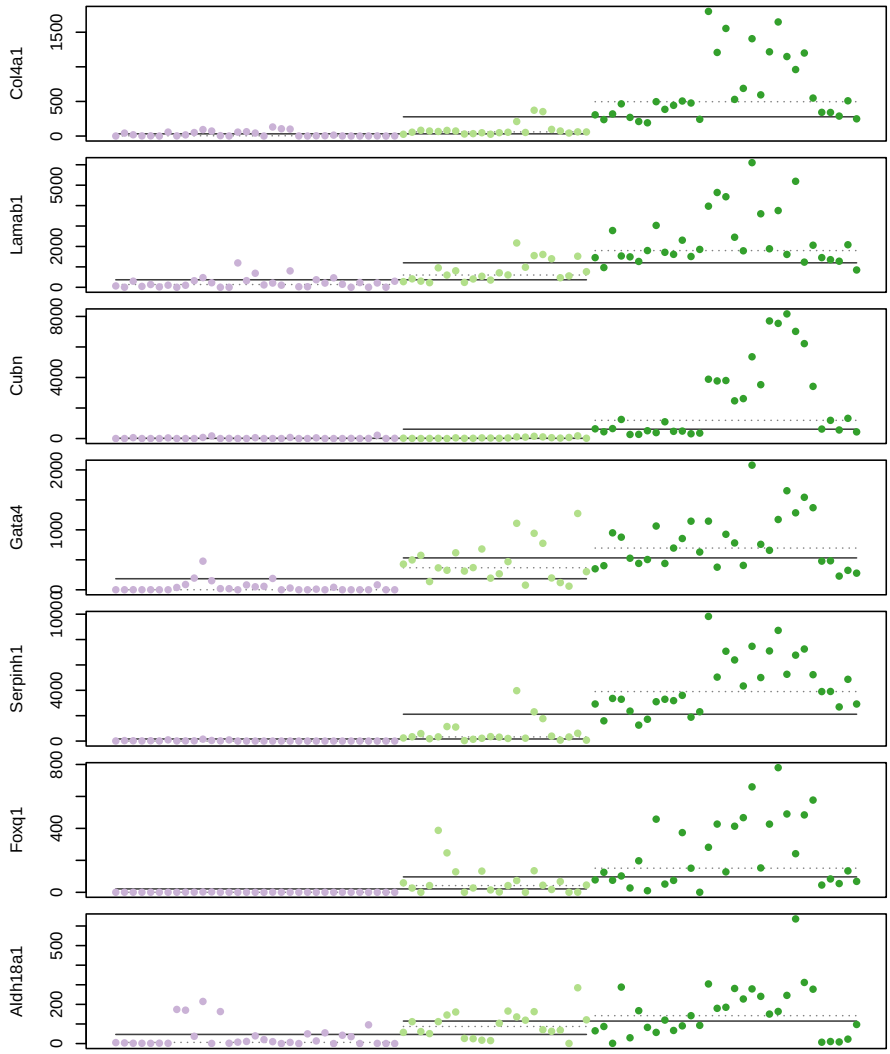


Figure 42: Visualisation of the qPCR data for the seven genes. Shown are also the within-group means (horizontal dotted light grey lines) and the discretisation thresholds (horizontal solid dark grey lines).

In this way, the data are assigned to three discrete levels:

1. less than the midpoint of the means of E3.25 and E3.5 (PE),
2. in between 1. and 3.,
3. higher than the midpoint of the means of E3.5 (PE) and E4.5 (PE),

The result of this is shown in Figure 43, where these three levels are represented by white, light blue, dark blue.

```
> discretize = function(x) {
+   exprs(x) = t(sapply(seq_len(nrow(exprs(x))), function(r) {
+       as.integer(cut(exprs(x)[r, ], breaks = c(-Inf, discrthreshs[r, ],+Inf)))
+   }))
+   return(x)
```

```
+ }
```

```
> sxqd = discretize(sxq)
```

```
> myHeatmap3(sxqd, log = FALSE, Colv = FALSE, Rowv = FALSE, key = FALSE, ncol = 3)
```

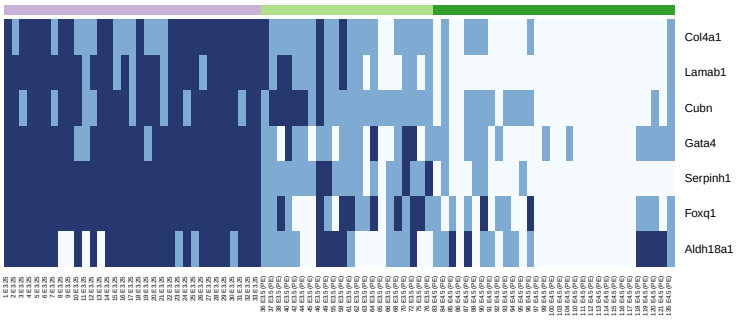


Figure 43: Heatmap for the discretized values. Rows and colums are in the original order.

10.4 Temporal order - hierarchy

Now we look for a (putatively: temporal) ordering of genes, so that the number of cases (samples $\times$ genes) in which a lower level follows a higher one is minimised. More specifically, the function `costfun1` below adds a penalty of 1 for every instance that in a sample, white follows *after* blue in an E3.5 sample (for the E3.25→E3.5 transition), and `costfun2` adds a penalty of 1 when white/light blue follow *after* dark blue in an E4.5 sample (for E3.5→E4.5) .

```
> stopifnot(all(exprs(sxqd)%in%(1:3)))
> costfun1 = function(x, sg) {
+   k = (sg=="E3.5 (PE)")
+   mean( (x[-1,k]==1) & (x[-nrow(x),k]>1 ) )
+ }
> costfun2 = function(x, sg) {
+   k = (sg=="E4.5 (PE)")
+   mean( (x[-1,k]<3 ) & (x[-nrow(x),k]==3) )
+ }
> library(gtools)
> perms = permutations(nrow(sxq), nrow(sxq))
> bruteForceOptimisation = function(fun, samps) {
+   if (missing(samps)) samps = rep(TRUE, ncol(sxqd))
+   apply(perms, 1, function(i) fun(exprs(sxqd)[i, samps], sxq$sampleGroup[samps]))
+ }
> costs = list(
+  `E3.25 -> E3.5` = bruteForceOptimisation(costfun1),
+   `E3.5 -> E4.5` = bruteForceOptimisation(costfun2))
> whopt  = lapply(costs, function(v) which(v==min(v)))
```

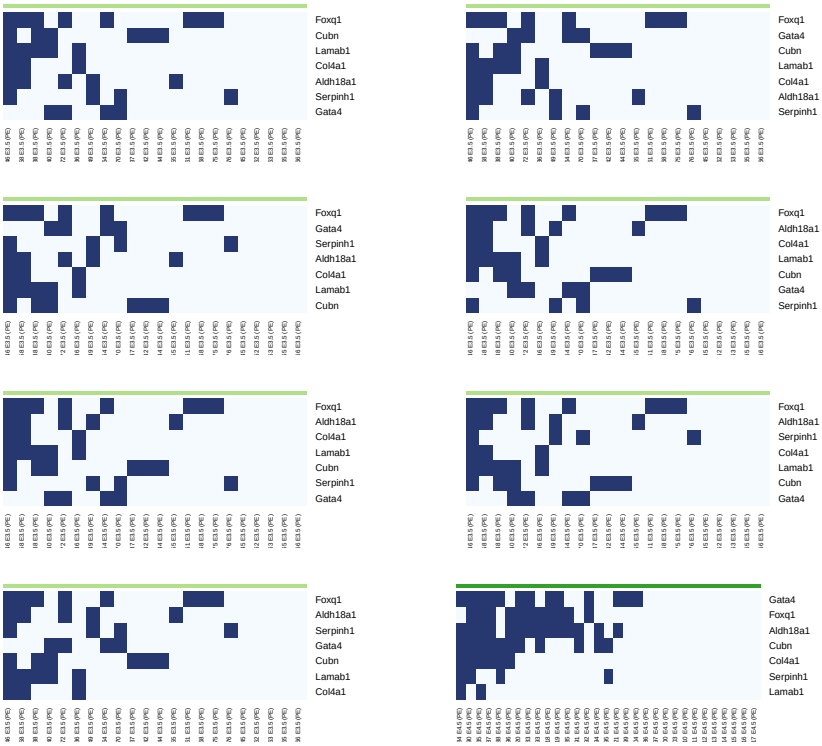


Figure 44: Heatmaps, ordered to show potential hierarchy of successive activation. This was done separately for the E3.25 – E3.5 (PE) and the E3.5 (PE) – E4.5 (PE) comparisons (as indicated by the horizontal colour bar). Note: if multiple equally optimally solutions were found, all are displayed.

```
> outf = file("fighmd.tex", open="w")
> for(i in seq(along = costs)) {
+   sxqdsub = switch(i,
+     {
+       rv = sxqd[, sxqd$sampleGroup %in% groups[2] ]
+       exprs(rv)[ exprs(rv)>2 ] = 2
+       rv
+     }, {
+       rv = sxqd[, sxqd$sampleGroup %in% groups[3] ]
+       exprs(rv)[ exprs(rv)<2 ] = 2
+       exprs(rv) = exprs(rv)-1
+       rv
+     })
+   columnOrder = order(sxqdsub$sampleGroup, colSums(exprs(sxqdsub)))
+   for(w in whopt[[i]]) {
+     fn = sprintf("fighmd-%d-%d.pdf", i, w)
+     pdf(fn, width = 7, height = 3)
+     myHeatmap3(sxqdsub[perms[w, ], columnOrder], log = FALSE, Colv = FALSE, Rowv = FALSE,
+                key = FALSE, ncol = 2, breaks = seq(0.5, 2.5, by = 1))
+     dev.off()
+     cat("\\includegraphics[width=0.49\\textwidth]{", fn, "}\n", file = outf, sep = "")
```

```
+   }
+ }
> close(outf)
```

```
> for(i in seq(along = costs)) {
+   k = unique(costs[[i]][whopt[[i]]])
+   stopifnot(length(k)==1)
+   cat(sprintf("%14s: cost %g\n", names(costs)[[i]], k))
+ }

 E3.25 -> E3.5: cost 0.106061
  E3.5 -> E4.5: cost 0.0591398
```

See Figure 44.

10.4.1 How significant is this?

Repeat the above with bootstrapping.

```
> bopt = lapply(list(costfun1, costfun2), function(fun) {
+   boot(data = seq_len(ncol(sxqd)),
+        statistic = function(dummy, idx) min(bruteForceOptimisation(fun, idx)),
+        R = 250)
+ })
```

```
> names(bopt) = names(costs)
> lapply(bopt, `[[`, "t0")

$`E3.25 -> E3.5`
[1] 0.1060606

$`E3.5 -> E4.5`
[1] 0.05913978

> multiecdf(lapply(bopt, `[[`, "t"))
> t.test(x = bopt[[1]]$t, y = bopt[[2]]$t)

        Welch Two Sample t-test

data:  bopt[[1]]$t and bopt[[2]]$t
t = 21.15, df = 472.55, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.02833504 0.03413928
sample estimates:
 mean of x  mean of y
0.08496477 0.05372761
```

See Figure 45.

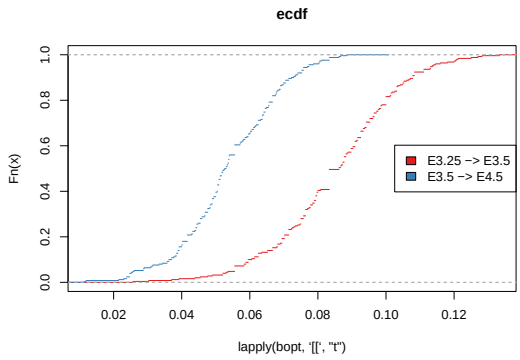


Figure 45: Distribution of bootstrap-resampled optimal costs ("disorder penalties"). Shown are the empirical distribution function of the bootstrap-sampled cost functions for the two situations. The values for E3.5 -> E4.5 are significantly smaller (see also t-test result in the main text).

A Influence of cell position on gene expression

To examine how cell position within the embryo influences its gene expression, cells lying on the surface of the ICM facing the blastocyst cavity were fluorescently labelled and expression profiled versus those located deeper within the ICM. We then used multi-dimensional scaling to compare the labelled and non-labelled cells at E3.5 and E4.5, based on the expression of 10 highly variable genes, as identified from the E3.5 and 4.5 microarray data, and quantified by single-cell qPCR measurements. The result of this analysis is show in Figure 46.

```
> data("xql")
> labeledSampleColourMap = c(grDevices::hcl.colors(5, "RdGy")[c(1,2)], grDevices::hcl.colors(5, "BrBG")[c(4
> names(labeledSampleColourMap) = c("E4.5_high", "E3.5_high", "E3.5_low",  "E4.5_low")
> labeledGroups = with(pData(xql), list(
+   `E3.5_high` = which(Label=="High" & Embryonic.day=="E3.5"),
+   `E3.5_low`  = which(Label=="Low"  & Embryonic.day=="E3.5"),
+   `E4.5_high`        = which(Label=="High" & Embryonic.day=="E4.5"),
+   `E4.5_low`  = which(Label=="Low"  & Embryonic.day=="E4.5")))
> labeledSampleColours = rep(NA_character_, ncol(xql))
> for(i in seq(along = labeledGroups))
+  labeledSampleColours[labeledGroups[[i]]] = labeledSampleColourMap[names(labeledGroups)[i]]
> xql$sampleColours = labeledSampleColours
> md = MASS::isoMDS(dist(t(log(exprs(xql),2))))$points
> plot(md, col = xql$sampleColours, pch = 16, asp = 1, xlab = "",ylab = "")
```

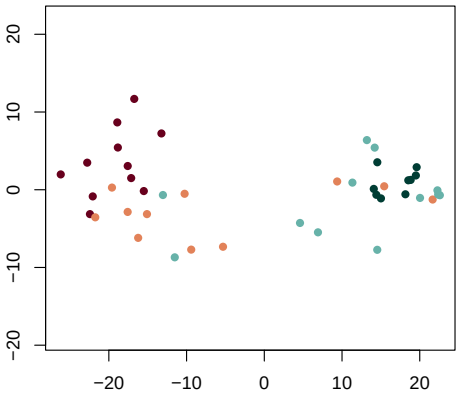


Figure 46: Multi-dimensional scaling plot. Light red and light green dots represent E3.5 labeled and unlabeled cells, while red and green ones represent E4.5 labeled and unlabeled cells, respectively.

B Correlation between Fgf ligands and Fgf receptors

Several Fgf ligands (Fgf3, 4 and 13) and all Fgf receptors (Fgfr1-4) were found to be differentially expressed within the ICM, thus possibly contributing to the EPI versus PrE lineage segregation. We have found that a statistically significant correlation (positive or negative) in gene expression levels is discernible at single cell level for Fgf4 against Fgfr2, Fgf4 against Fgfr3, Fgf3 with Fgfr3, and Fgf3 with Fgfr4 at E3.5 and E4.5 (Figure 47).

```
> xs = x[, (x$genotype %in% "WT") & (x$Embryonic.day %in% c("E3.25","E3.5","E4.5"))]
> Fgf3 = c("1441914_x_at")
> Fgf4 = c("1420086_x_at")
> Fgfr2 = c("1433489_s_at")
> Fgfr3 = c("1421841_at")
> Fgfr4 = c("1418596_at")
> correlationPlot = function(ID){
+   xsl = xs[ID, ]
+   mat = exprs(xsl)
+   rownames(mat) = fData(xsl)[, "symbol"]
+   plot(t(mat), pch = 16, asp = 1, cex = 1.25, cex.lab = 1.2, col = xs$sampleColour,
+        xlim = c(2,12), ylim = c(3,11))
+ }
> par(mfrow = c(1,4))
> correlationPlot(c(Fgf4,Fgfr2))
> correlationPlot(c(Fgf4,Fgfr3))
> correlationPlot(c(Fgf3,Fgfr3))
> correlationPlot(c(Fgf3,Fgfr4))
```

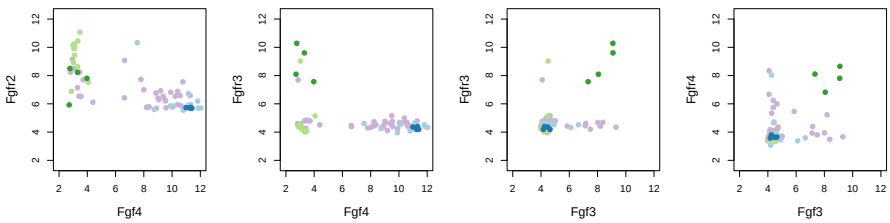


Figure 47: Correlation between Fgf ligands and Fgf receptors. Scatter plots with each dot representing the mRNA expression levels of specific Fgf ligand and receptor pairs in one blastomere. The colour code is as in Figure 15, with pink representing E3.25 cells, light blue and light green E3.5 EPI and PrE cells, and blue and green E4.5 EPI and PrE cells, respectively.

C Session info

Below, the output is shown of `sessionInfo` on the system on which this document was compiled.

```
> toLatex(sessionInfo())
```

- R version 4.6.1 (2026-06-24), `x86_64-pc-linux-gnu`

- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=en_US.UTF-8`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Time zone: `Etc/UTC`
- TZcode source: `system (glibc)`
- Running under: `Ubuntu 26.04.1 LTS`
- Matrix products: default
- BLAS: `/usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3`
- LAPACK: `/usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.32.so` ; LAPACK version3.12.0
- Base packages: base, datasets, graphics, grDevices, methods, stats, stats4, utils
- Other packages: annotate 1.91.0, AnnotationDbi 1.75.2, Biobase 2.73.2, BiocGenerics 0.59.12, boot 1.3-32, clue 0.3-68, genefilter 1.95.0, geneplotter 1.91.0, generics 0.1.4, gtools 3.9.5, Hiiragi2013 1.49.5, IRanges 2.47.5, KEGGREST 1.53.6, lattice 0.23-1, mouse4302.db 3.13.0, org.Mm.eg.db 3.23.0, S4Vectors 0.51.10, XML 3.99-0.25, xtable 1.8-8
- Loaded via a namespace (and not attached): BiocBaseUtils 1.15.1, BiocManager 1.30.27, BiocStyle 2.41.0, Biostrings 2.81.9, bit 4.6.0, bit64 4.8.6, bitops 1.1-0, blob 1.3.0, buildtools 1.0.0, cachem 1.1.0, caTools 1.18.4, cli 3.6.6, cluster 2.1.8.3, compiler 4.6.1, crayon 1.5.3, curl 8.1.0, DBI 1.3.0, deldir 2.0-4, digest 0.6.39, evaluate 1.0.5, fastmap 1.2.0, gplots 3.3.0, grid 4.6.1, htmltools 0.5.9, httr 1.4.9, interp 1.1-6, jpeg 0.1-11, KernSmooth 2.23-27, knitr 1.52, latticeExtra 0.6-31, maketools 1.3.2, MASS 7.3-66, Matrix 1.7-6, MatrixGenerics 1.25.0, matrixStats 1.5.0, memoise 2.0.1, pkgconfig 2.0.3, png 0.1-9, R6 2.6.1, RColorBrewer 1.1-3, Rcpp 1.1.2, rlang 1.3.0, rmarkdown 2.32, RSQLite 3.53.3, Seqinfo 1.3.2, splines 4.6.1, survival 3.8-12, sys 3.4.3, tools 4.6.1, vctrs 0.7.3, xfun 0.62, XVector 0.53.0, yaml 2.3.12

D The data import script `readdata.R`

Listing 1: The script `readdata.R`

```
library("affy")
library("ArrayExpress")
library("arrayQualityMetrics")
library("mouse4302.db")

CELdir    = tempdir()
CELfiles  = getAE("E-MTAB-1681", path = CELdir, type = "raw")$rawFiles

## ------------------------------------------------------------------------------
## Read array metadata table and fill empty cells in the columns Embryonic.day
## and Total.number.of.cells by the values implied ## by the non-empty cells above
## ------------------------------------------------------------------------------

fillColumn = function(x, empty){
```

```
  wh  = which(!empty(x))
  len = length(wh)
  wh  = c(wh, length(x)+1)
  for(i in seq_len(len))
    x[ wh[i]:(wh[i+1]-1) ] = x[wh[i]]
  return(x)
}

readCSVtable = function(name) {
  x = read.csv(name, stringsAsFactors = FALSE, colClasses = "character")
  x$Embryonic.day = factor(fillColumn(x$Embryonic.day, empty = function(x) x==""))

  wh = which(colnames(x)=="Total.number.of.cells")
  if(length(wh)==1) {
    x[[wh]] = fillColumn(x$Total.number.of.cells, empty = function(x) x=="" & !is.na(x))
    x[[wh]] = as.integer(x[[wh]])
  } else {
    x$"Total.number.of.cells" = rep(NA, nrow(x))
  }
  x$Total.number.of.cells = addNA(as.factor(x$Total.number.of.cells))

  wh = which(colnames(x) %in% c("X.EPI...PE.", "Type"))
  if(length(wh)==1) {
    colnames(x)[wh] = "lineage"
  } else {
    x$lineage = rep(NA, nrow(x))
  }

  row.names(x) = paste(x$File.name, "CEL", sep = ".")
  return(x)
}

## ------------------------------- Script starts here ---------------------------------

pdata = readCSVtable(system.file("scripts", "annotation.csv", package = "Hiiragi2013"))

pdata$genotype = as.factor(ifelse(grepl("_KO$", pdata$File.name), "FGF4-KO", "WT"))

## ------------------------------------------------------------------------------------
## Read the CEL files
## ------------------------------------------------------------------------------------

fileNames = row.names(pdata)
fileExists = (fileNames %in% CELfiles)
stopifnot(all(fileExists))

a = ReadAffy(filenames = fileNames, celfile.path = CELdir, phenoData = pdata,
             verbose = TRUE)

pData(a)$ScanDate = factor(as.Date(sub( "10/16/09", "2010-09-16",
    sapply(strsplit( protocolData(a)$ScanDate, split = "[T ]" ), '[', 1) )))

# We no longer include the data directly in the package because it's too heavy.
# In particular, GitHub has a file size limit of 100 MB, which this file exceeds.

# save(a, file="a.rda", compress="xz")

## ------------------------------------------------------------------------------------
## Normalize with RMA
## ------------------------------------------------------------------------------------

x = rma(a)

## Create columns
## fData(x)$symbol:   gene symbols where available, Affy feature ID otherwise
```

```
## fData(x)$genename: a more verbose gene description

annotateGene = function(db, what, missing) {
  tab = toTable(db[ featureNames(x) ])
  mt = match( featureNames(x), tab$probe_id)
  ifelse(is.na(mt), missing, tab[[what]][mt])
}
fData(x)$symbol   = annotateGene(mouse4302SYMBOL,   "symbol", missing = featureNames(x))
fData(x)$genename = annotateGene(mouse4302GENENAME, "gene_name", missing = "")
fData(x)$ensembl  = annotateGene(mouse4302ENSEMBL,  "ensembl_id", missing = "")

## ------------------------------------------------------------------------------------
## Grouping of samples
## ------------------------------------------------------------------------------------

## We define a grouping of the samples and an associated colour map, which will
## be used in the plots throughout this report.

groups = with(pData(x), list(
  `E3.25`           = which(genotype=="WT" & Embryonic.day=="E3.25"),
  `E3.5 (EPI)`      = which(genotype=="WT" & Embryonic.day=="E3.5" & lineage=="EPI"),
  `E4.5 (EPI)`      = which(genotype=="WT" & Embryonic.day=="E4.5" & lineage=="EPI"),
  `E3.5 (PE)`       = which(genotype=="WT" & Embryonic.day=="E3.5" & lineage=="PE"),
  `E4.5 (PE)`       = which(genotype=="WT" & Embryonic.day=="E4.5" & lineage=="PE"),
  `E3.25 (FGF4-KO)` = which(genotype=="FGF4-KO" & Embryonic.day=="E3.25"),
  `E3.5 (FGF4-KO)`  = which(genotype=="FGF4-KO" & Embryonic.day=="E3.5"),
  `E4.5 (FGF4-KO)`  = which(genotype=="FGF4-KO" & Embryonic.day=="E4.5")))

sampleColourMap = character(length(groups))
names(sampleColourMap) = names(groups)
sampleColourMap[c("E3.5 (EPI)", "E4.5 (EPI)")]         = palette.colors(10, "Paired")[1:2]
sampleColourMap[c("E3.5 (PE)", "E4.5 (PE)")]           = palette.colors(10, "Paired")[3:4]
sampleColourMap[c("E3.25 (FGF4-KO)")]                  = palette.colors(10, "Paired")[c(7)]
sampleColourMap[c("E3.5 (FGF4-KO)", "E4.5 (FGF4-KO)")] = palette.colors(10, "Paired")[c(8,6)]
sampleColourMap[c("E3.25")]                            = palette.colors(12, "Paired")[c(9)]
stopifnot(!any(sampleColourMap==""))

## The following assertions aim to make sure that each sample was assigned to
## exactly one group.
stopifnot(!any(duplicated(unlist(groups))),
            setequal(unlist(groups), seq_len(ncol(x))),
            setequal(names(sampleColourMap), names(groups)))

## Next, assign a colour and a name to each sample, which will be used in the
## subsequent plots. For sample names, use the group name and the array numeric
## index.

sampleNames = sampleGroup = rep(NA_character_, ncol(x))
for(i in seq(along = groups)) {
  idx = groups[[i]]
  sampleGroup[idx] = names(groups)[i]
  sampleNames[idx] = paste(idx, names(groups)[i])
}
pData(x)$sampleGroup  = sampleGroup
pData(x)$sampleColour = sampleColourMap[sampleGroup]
sampleNames(x) = sampleNames

save(x, file="x.rda", compress="xz")
```

References

[1] R. A. Irizarry, B. Hobbs, F. Collin, Y. D. Beazer-Barclay, K. J. Antonellis, U. Scherf, and T. P. Speed. Exploration, normalization, and summaries of high density oligonucleotide array probe level data. *Biostatistics*, 4(2):249–264, 2003.

[2] Richard Bourgon, Robert Gentleman, and Wolfgang Huber. Independent filtering increases detection power for high-throughput experiments. *PNAS*, 107(21):9546–9551, 2010.

[3] Kurt Hornik. A CLUE for CLUster Ensembles. *Journal of Statistical Software*, 14(12), 2005.

[4] E. Dimitriadou, A. Weingessel, and K. Hornik. A combination scheme for fuzzy clustering. *International Journal of Pattern Recognition and Artificial Intelligence*, 16:901–912, 2002.

[5] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society Series B*, 57:289–300, 1995.

[6] Y. Buganim, D. A. Faddah, A. W. Cheng, E. Itskovich, S. Markoulaki, K. Ganz, S. L. Klemm, A. van Oudenaarden, and R. Jaenisch. Single-Cell Expression Analyses during Cellular Reprogramming Reveal an Early Stochastic and a Late Hierarchic Phase. *Cell*, 150:1209–1222, 2012.