Using the a4 package

Willem Talloen, Tobias Verbeke

February 3, 2011

Contents

1 Preparation of the Data

First we load the package a4 and the example real-life data set ALL.

```
R> library(a4)

Loaded glmnet 1.5.1

R> require(ALL)
R> data(ALL, package = "ALL")
```

For illustrative purposes, simulated data sets can also be very valuable (but not used here).

```
R> require(nlcv)
R> esSim <- simulateData(nEffectRows = 50, betweenClassDifference = 5,
     nNoEffectCols = 5, withinClassSd = 0.2)
```

1.1 ExpressionSet object

The data are assumed to be in an expressionSet object. Such an object structure combines different sources of information into a single structure, allowing easy data manipulation (e.g., subsetting, copying) and data modelling.

The textttfeatureData slot is typically not yet containing all relevant information about the genes. This interesting extra gene information can be added using addGeneInfo.

```
R> ALL <- addGeneInfo(ALL)
```

1.2 Some data manipulation

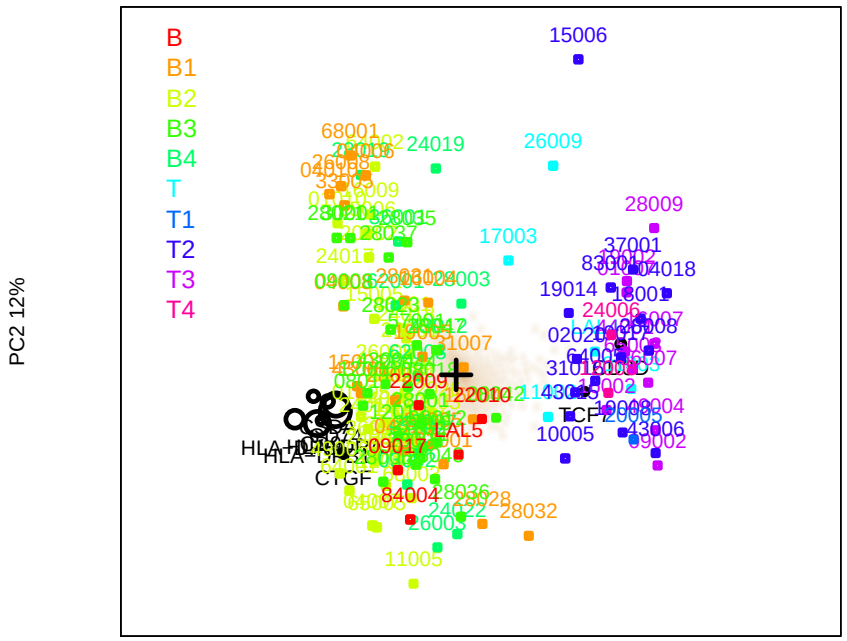
The ALL data consists out of samples obtained from two types of cells with very distinct expression profiles; B-cells and T-cells. To have a more subtle signal, gene expression will also be compared between the BCR/ABL and the NEG group within B-cells only. To this end, we create the expressionSet bcrAblOrNeg containing only B-cells with BCR/ABL or NEG.

```
R> Bcell <- grep("^B", as.character(ALL$BT))
R> subsetType <- "BCR/ABL"
R> bcrAblOrNegIdx <- which(as.character(ALL$mol) %in%
     c("NEG", subsetType))
R> bcrAblOrNeg <- ALL[, intersect(Bcell, bcrAblOrNegIdx)]
R> bcrAblOrNeg$mol.biol <- factor(bcrAblOrNeg$mol.biol)
```

2 Unsupervised data exploration

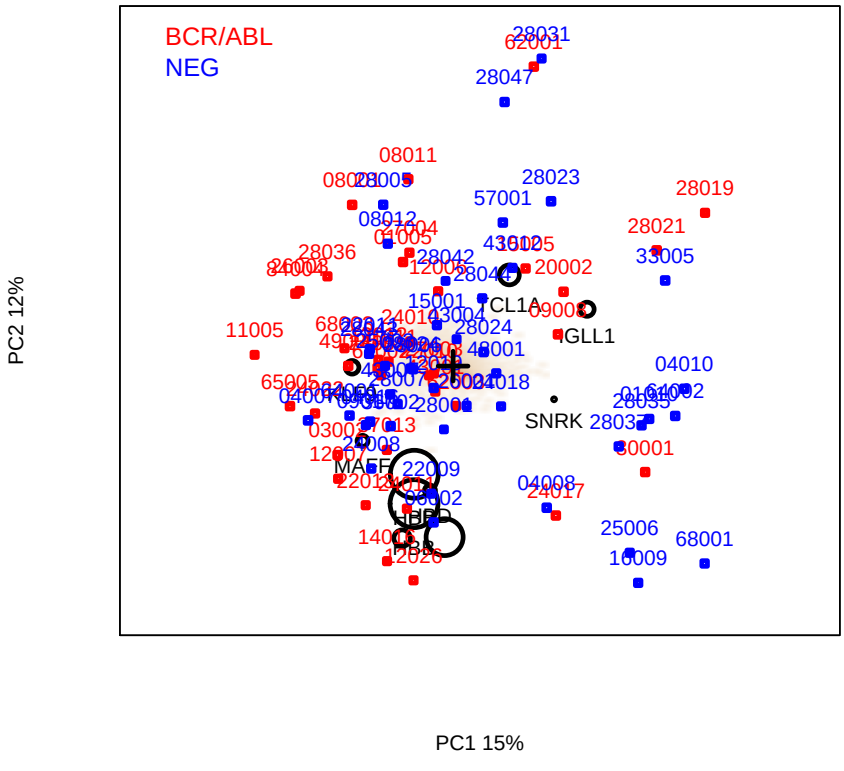
Spectral maps are very powerful techniques to get an unsupervised picture of how the data look like. A spectral map of the `ALL` data set shows that the B- and the T-subtypes cluster together along the x-axis (the first principal component). The plot also indicates which genes contribute in which way to this clustering. For example, the genes located in the same direction as the T-cell samples are higher expressed in these T-cells. Indeed, the two genes at the left (TCF7 and CD3D) are well known to be specifically expressed by T-cells (Wetering 1992, Krissansen 1986).

```
R> spectralMap(object = ALL, groups = "BT")
```



A spectral map of the `bcrAblOrNeg` data subset does not show a clustering of BCR/ABL or NEG cells.

```
R> spectralMap(object = bcrAblOrNeg, groups = "mol.biol",
     probe2gene = TRUE)
```



3 Filtering

The data can be filtered, for instance based on variance and intensity, in order to reduce the high-dimensionality.

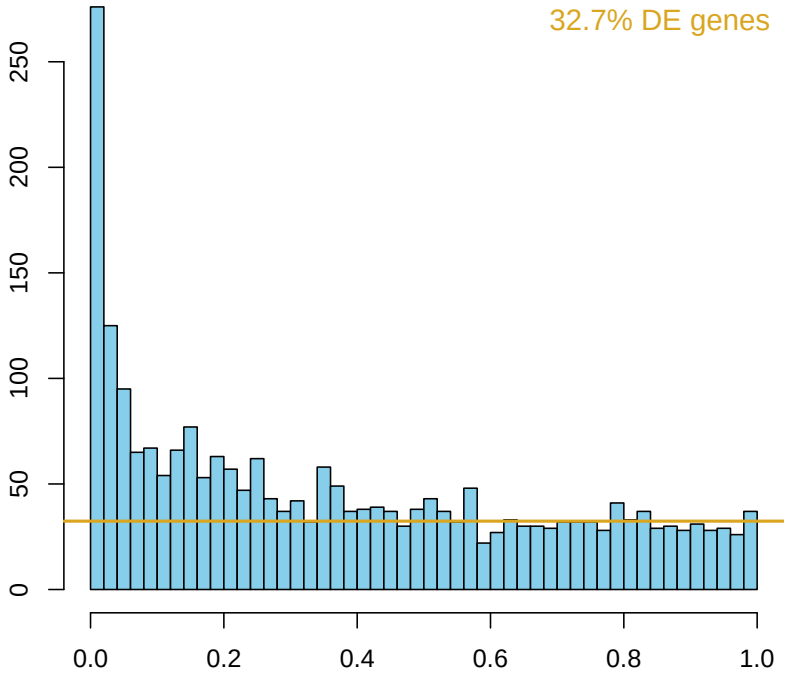
```
R> selBcrAblOrNeg <- filterVarInt(object = bcrAblOrNeg)
R> propSelGenes <- round((dim(selBcrAblOrNeg)[1]/dim(bcrAblOrNeg)[1]) *
     100, 1)
```

This filter selected 18.9 % of the genes (2391 of the in total 12625 genes).

4 Detecting differential expression

4.1 T-test

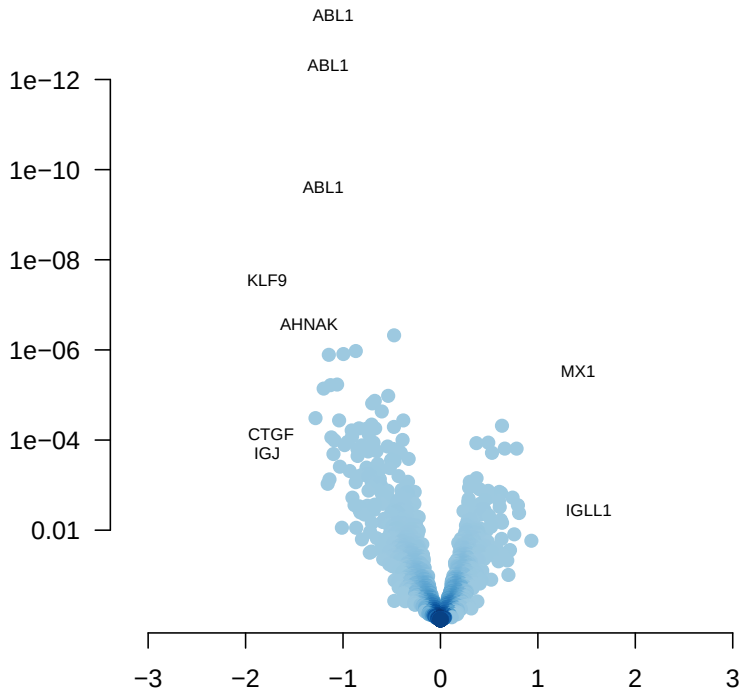
```
R> tTestResult <- tTest(selBcrAblOrNeg, "mol.biol")

R> histPvalue(tTestResult[, "p"], addLegend = TRUE)
R> propDEgenesRes <- propDEgenes(tTestResult[,
     "p"])
```



Using an ordinary t-test, there are 171 genes significant at a FDR of 10%. The proportion of genes that are trully differentially expressed is estimated to be around 32.7.

The toptable and the volcano plot show that three most significant probe sets all target `ABL1`. This makes sense as the main difference between BCR/ABL and NEG cells is a mutation in this particular ABL gene.

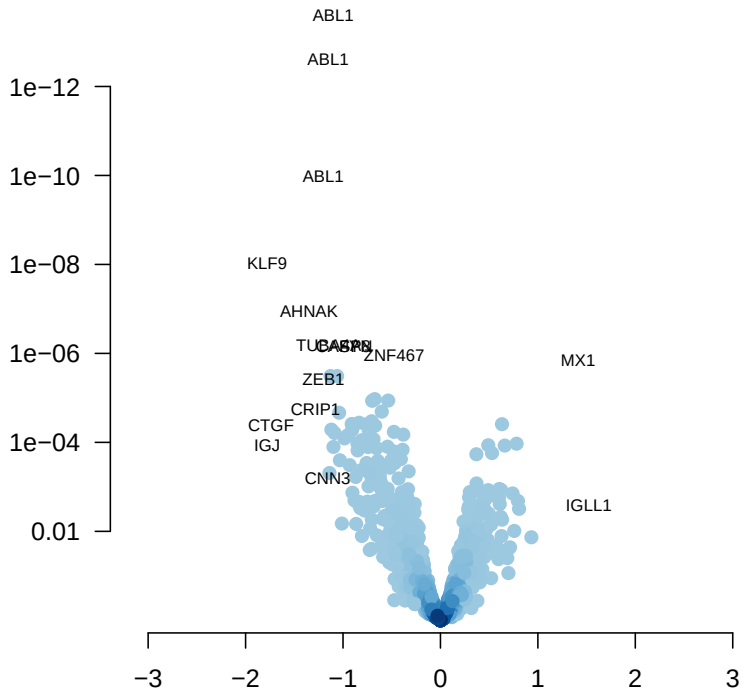
```
R> tabTTest <- topTable(tTestResult, n = 10)
R> xtable(tabTTest, caption = "The top 5 features selected by an ordinary t-test.",
     label = "tablassoClass")

R> volcanoPlot(tTestResult, topPValues = 5, topLogRatios = 5)
```



4.2 Limma for comparing two groups

In this particular data set, the modified t-test using `limmaTwoLevels` provides very similar results. This is because the sample size is relatively large.

```
R> limmaResult <- limmaTwoLevels(selBcrAblOrNeg,
     "mol.biol")
R> volcanoPlot(limmaResult)
```



It is very useful to put lists of genes in annotated tables where the genes get hyperlinks to EntrezGene.

```
R> tabLimma <- topTable(limmaResult, n = 10,
     coef = 2)
```

Gene	logFC	AveExpr	P.Value	adj.P.Val	GENENAME
ABL1	-1.10	9.20	0.00	0.00	c-abl oncogene 1, non-receptor tyrosin
ABL1	-1.15	9.00	0.00	0.00	c-abl oncogene 1, non-receptor tyrosin
ABL1	-1.20	7.90	0.00	0.00	c-abl oncogene 1, non-receptor tyrosin
KLF9	-1.78	8.62	0.00	0.00	Kruppel-like factor 9
AHNAK	-1.35	8.44	0.00	0.00	AHNAK nucleoprotein
TUBA4A	-1.15	9.23	0.00	0.00	tubulin, alpha 4a
FYN	-0.87	7.76	0.00	0.00	FYN oncogene related to SRC, FGR, YES
CASP8	-1.00	8.04	0.00	0.00	caspase 8, apoptosis-related cysteine
ZNF467	-0.48	7.14	0.00	0.00	zinc finger protein 467
MX1	1.41	6.73	0.00	0.00	myxovirus (influenza virus) resistance

4.3 Limma for linear relations with a continuous variable

Testing for (linear) relations of gene expression with a (continuous) variable is typically done using regression. A modified t-test approach improves the results by penalizing small slopes. The modified regressions can be applied using `limmaReg`.

5 Class prediction

There are many classification algorithms with profound conceptual and methodological differences. Given the differences between the methods,there's probably no single classification method that always works best, but that certain methods perform better depending on the characteristics of the data.

On the other hand, these methods are all designed for the same purpose, namely maximizing classification accuracy. They should consequently all pick up (the same) strong biological signal when present, resulting in similar outcomes.

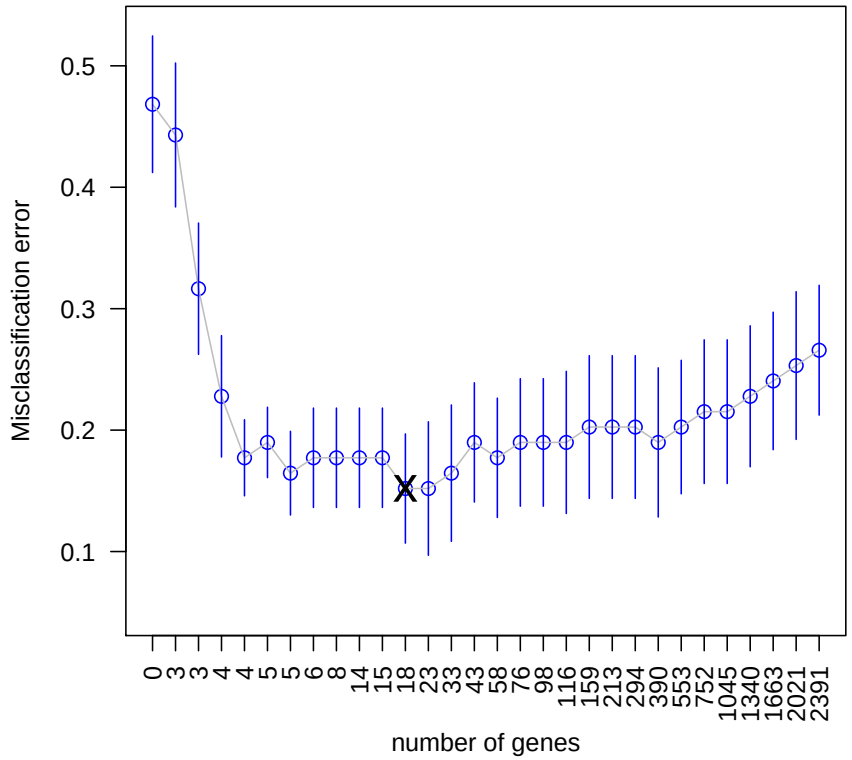
Personally, we like to apply four different approaches; PAM, RandomForest, forward filtering in combination with various classifiers, and LASSO.

All four methods have the property that they search for the smallest set of genes while having the highest classification accuracy. The underlying rationale and algorithm is very different between the four approaches, making their combined use potentially complementary.

5.1 PAM

PAM (Tibshirani 2002) applies univariate and dependent feature selection.

```
R> resultPam <- pamClass(selBcrAblOrNeg, "mol.biol")
R> plot(resultPam)
R> featResultPam <- topTable(resultPam, n = 15)
R> xtable(head(featResultPam$listGenes), caption = "Top 5 features selected by PAM.")
```



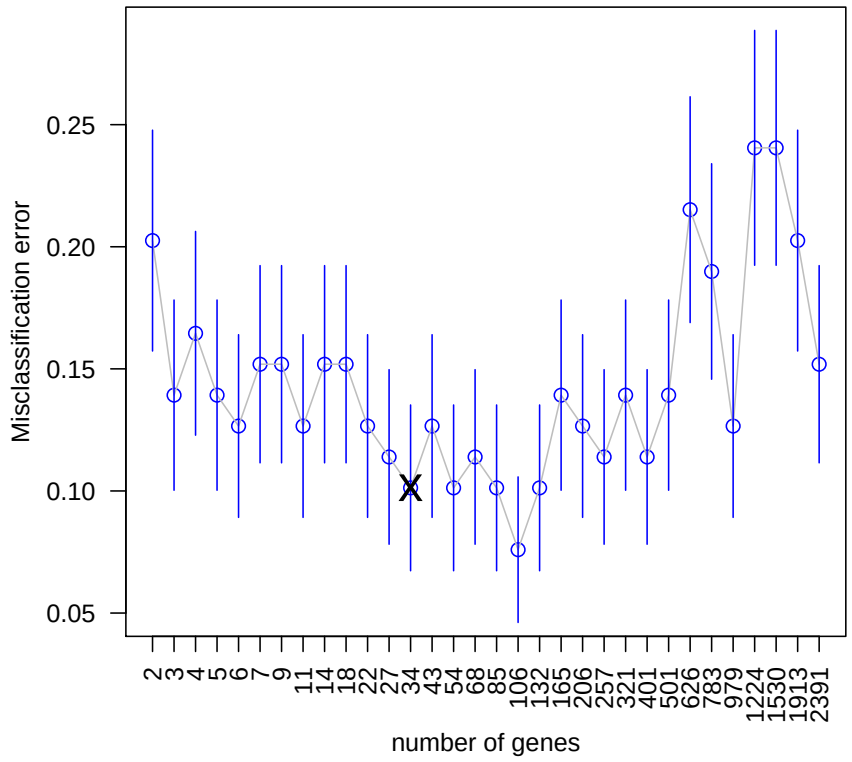
5.2 Random forest

Random forest with variable importance filtering (Breiman 2001, Diaz-Uriarte 2006) applies multivariate and dependent feature selection. Be cautious when interpreting its outcome, as the obtained results are unstable and sometimes overoptimistic.

```
R> resultRF <- rfClass(selBcrAblOrNeg, "mol.biol")
R> plot(resultRF, which = 2)
R> featResultRF <- topTable(resultRF, n = 15)
R> xtable(head(featResultRF$topList), caption = "Features selected by Random Forest variable impor
```

	GeneSymbol
1635_at	ABL1
1636_g_at	ABL1
2039_s_at	FYN
37351_at	UPP1
39730_at	ABL1
39837_s_at	ZNF467

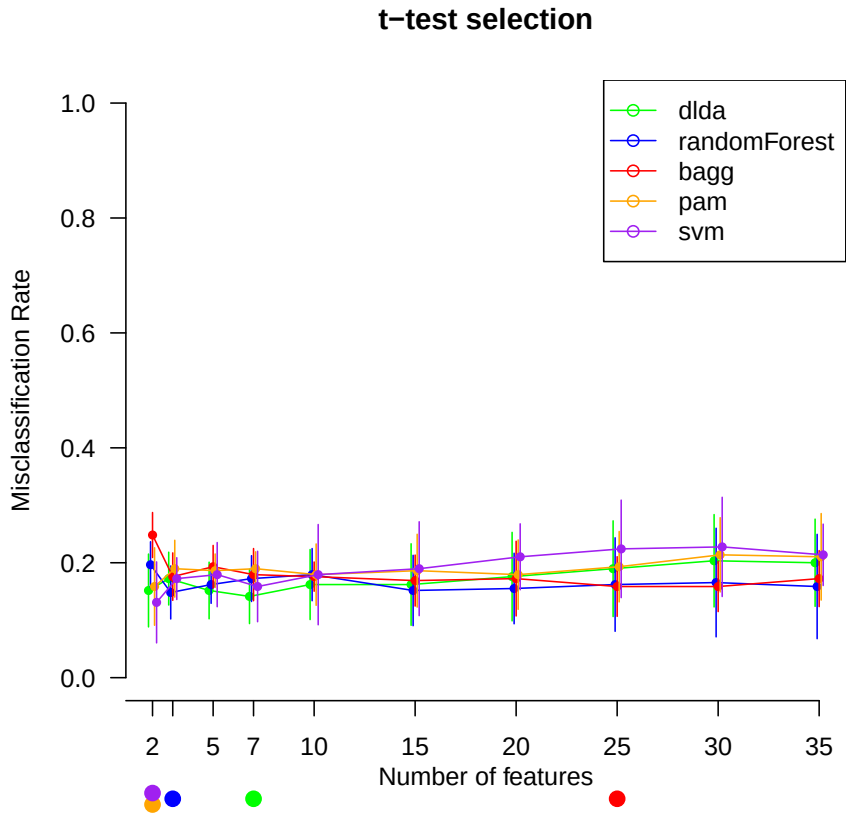
Table 1: Features selected by Random Forest variable importance.



5.3 Forward filtering with various classifiers

Forward filtering in combination with various classifiers (like DLDA, SVM, random forest, etc.) apply an independent feature selection. The selection can be either univariate or multivariate depending on the chosen selection algorithm; we usually choose Limma as a univariate although random forest variable importance could also be used as a multivariate selection criterium.

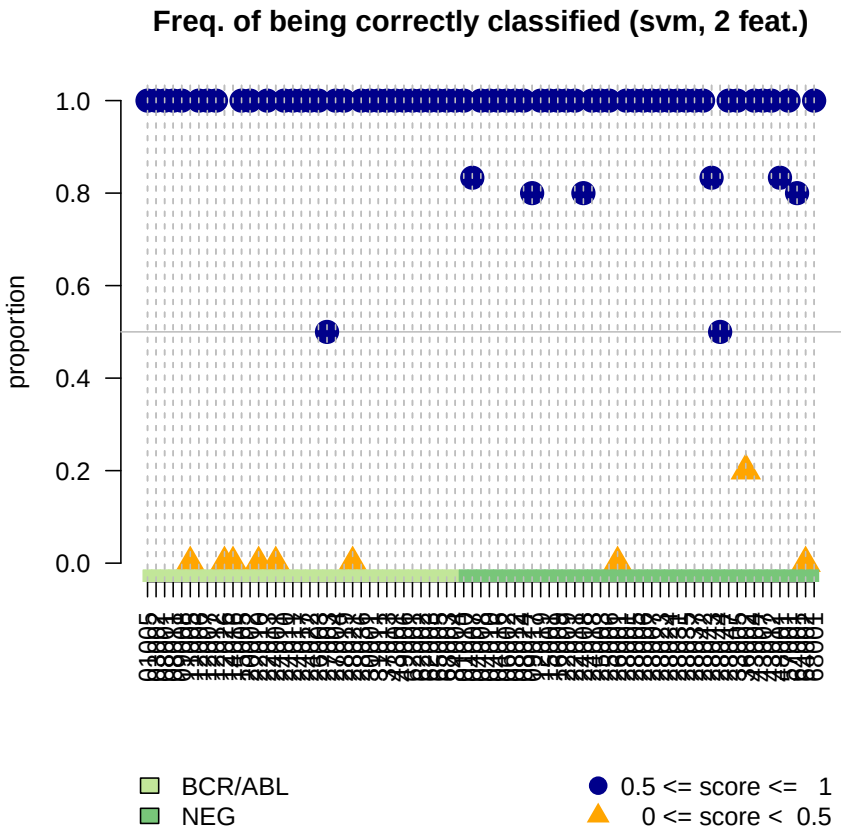
```
R> mcrPlot_TT <- mcrPlot(nlcvTT, plot = TRUE, optimalDots = TRUE, layout = TRUE, main = "t-test se
```



	nFeat_optim	mean_MCR	sd_MCR
dlda	7.00	0.14	0.05
randomForest	3.00	0.15	0.05
bagg	25.00	0.16	0.05
pam	2.00	0.16	0.07
svm	2.00	0.13	0.07

Table 2: Optimal number of genes per classification method together with the respective misclassification error rate (mean and standard deviation across all CV loops).

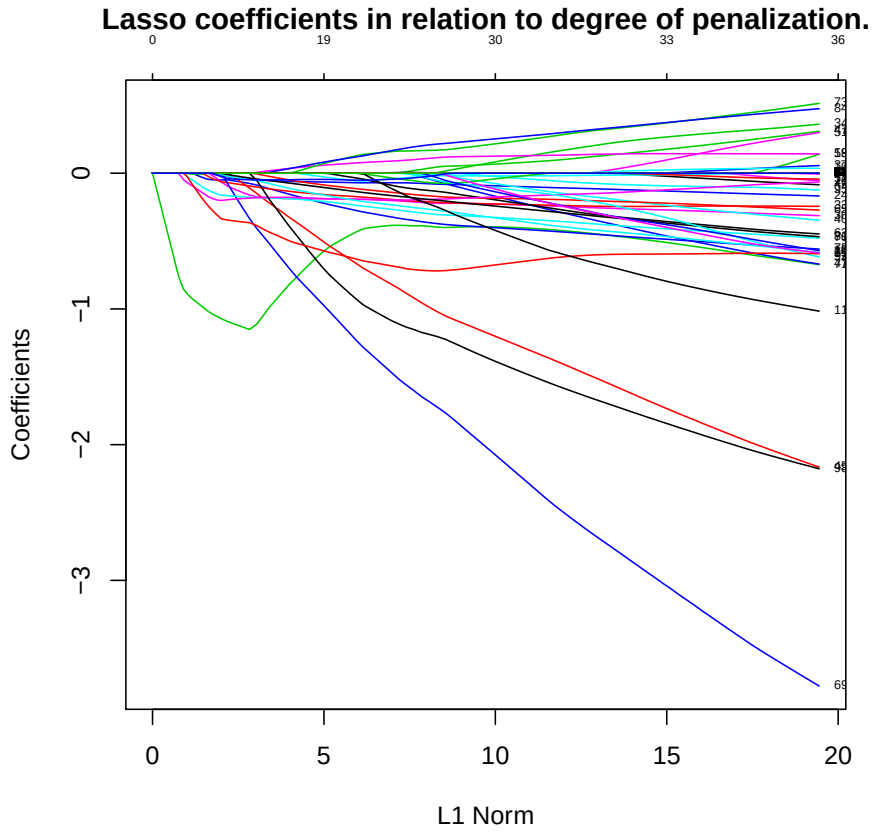
```
R> scoresPlot(nlcvTT, tech = "svm", nfeat = 2)
```



5.4 Penalized regression

LASSO (Tibshirani 2002) or elastic net (Zou 2005) apply multivariate and dependent feature selection.

```
R> resultLasso <- lassoClass(object = bcrAblOrNeg, groups = "mol.biol")
R> plot(resultLasso, label = TRUE, main = "Lasso coefficients in relation to degree of penalizatio
R> featResultLasso <- topTable(resultLasso, n = 15)
```



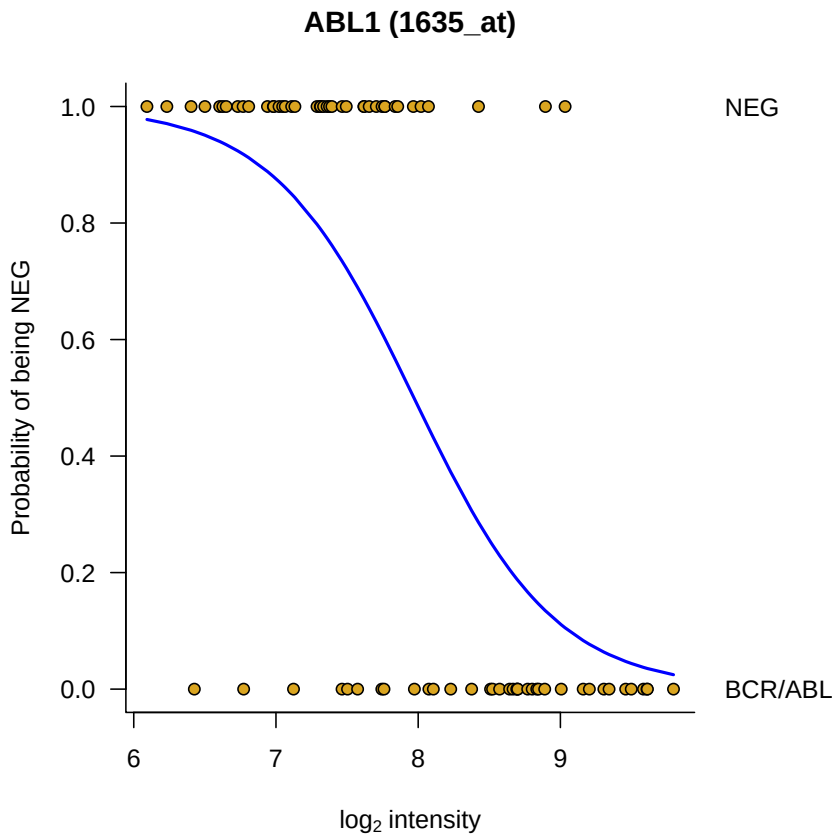
Gene	Coefficient
ITGA7	-3.78
ABL1	-2.18
TCL1B	-2.17
RAB32	-1.02
ABL1	-0.67
CHD3	-0.67
NFATC1	-0.62
ZNF467	-0.59
SERPINE2	-0.59
ANXA1	-0.58
PTPRJ	-0.57
YES1	-0.56
PTDSS1	0.51
ALDH1A1	-0.48
DSTN	0.48

Table 3: Features selected by Lasso, ranked from largest to smallest penalized coefficient.

5.5 Logistic regression

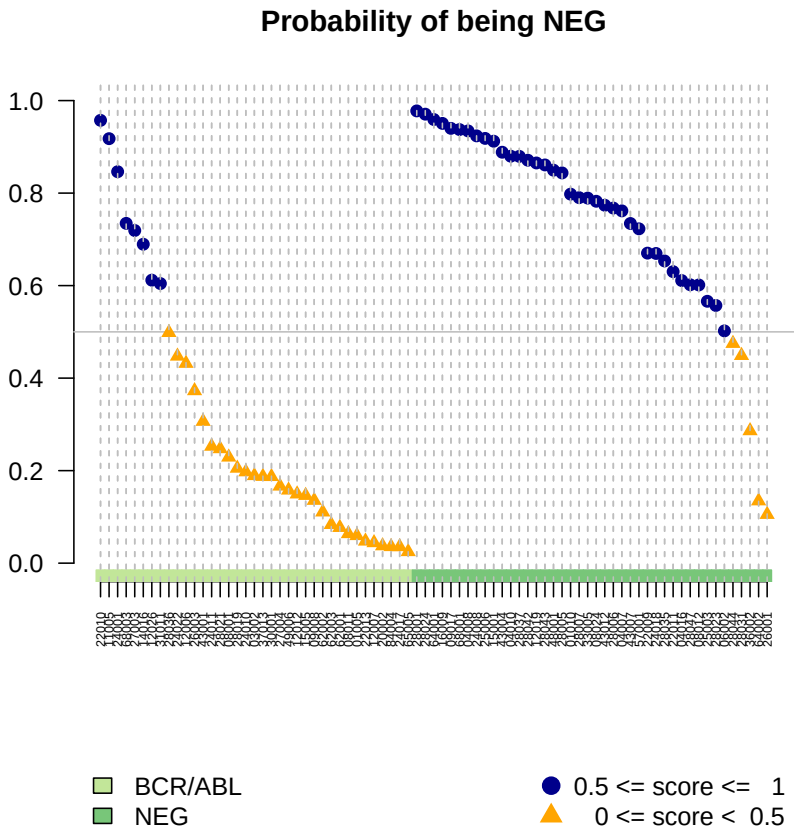
Logistic regression is used for predicting the probability to belong to a certain class in binary classification problems.

```
R> logRegRes <- logReg(geneSymbol = "ABL1", object = bcrAblOrNeg, groups = "mol.biol")
```

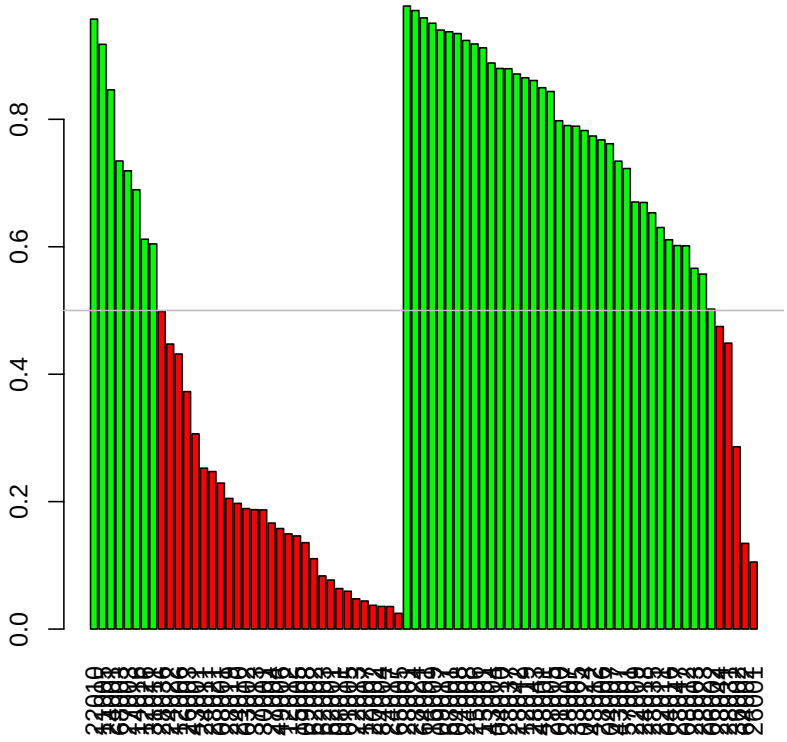


The obtained probabilities can be plotted with `ProbabilitiesPlot`. A horizontal line indicates the 50% threshold, and samples that have a higher probability than 50% are indicated with blue dots. Apparently, using the expression of the gene `ABL1`, quite a lot of samples predicted to with a high probability to be NEG, are indeed known to be NEG.

```
R> probabilitiesPlot(proportions = logRegRes$fit, classVar = logRegRes$y, sampleNames = rownames(l
```



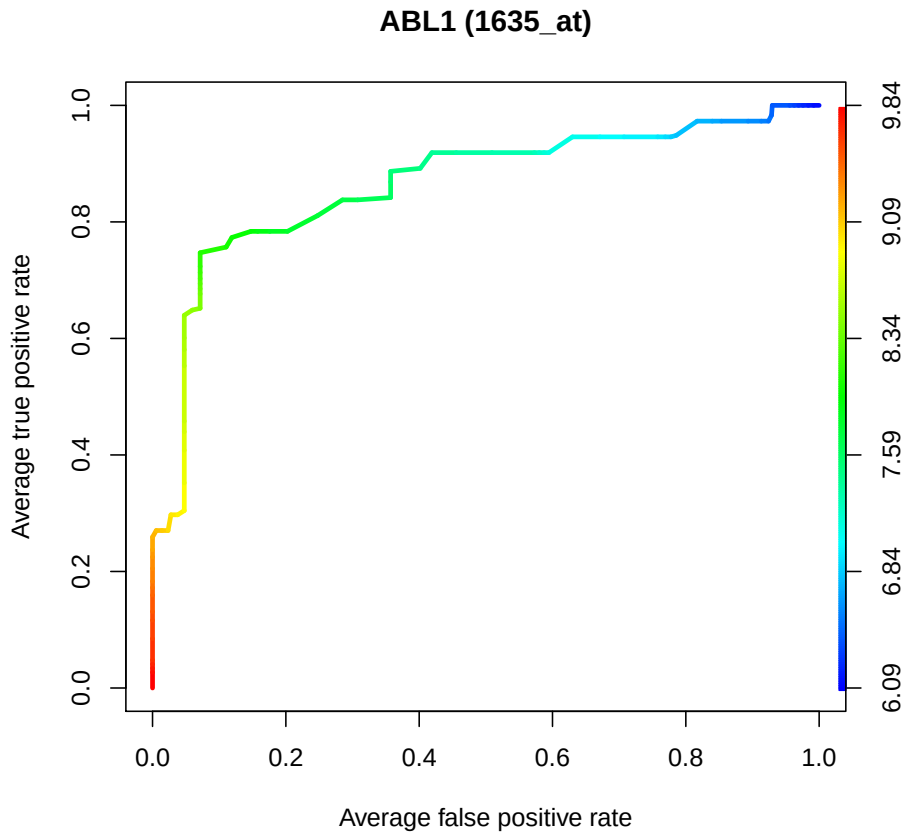

```
R> probabilitiesPlot(proportions = logRegRes$fit, classVar = logRegRes$y, barPlot = TRUE, sampleNa
```



5.6 Receiver operating curve

A ROC curve plots the fraction of true positives (TPR = true positive rate) versus the fraction of false positives (FPR = false positive rate) for a binary classifier when the discrimination threshold is varied. Equivalently, one can also plot sensitivity versus (1 - specificity).

```
R> ROCres <- ROCcurve(geneSymbol = "ABL1", object = bcrAblOrNeg, groups = "mol.biol")
```

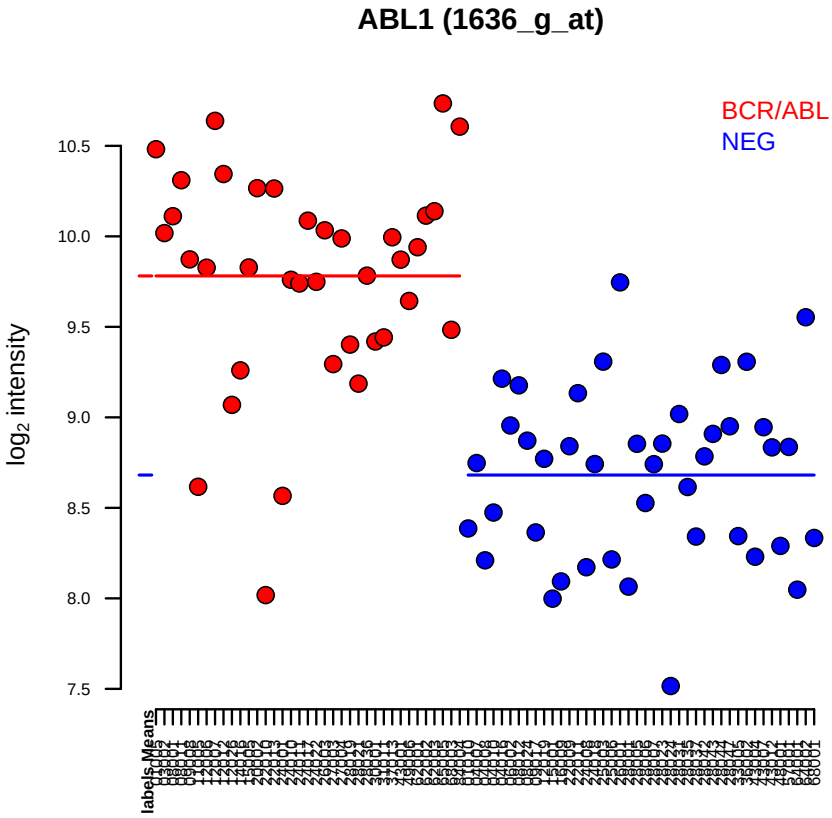


6 Visualization of interesting genes

6.1 Plot the expression levels of one gene

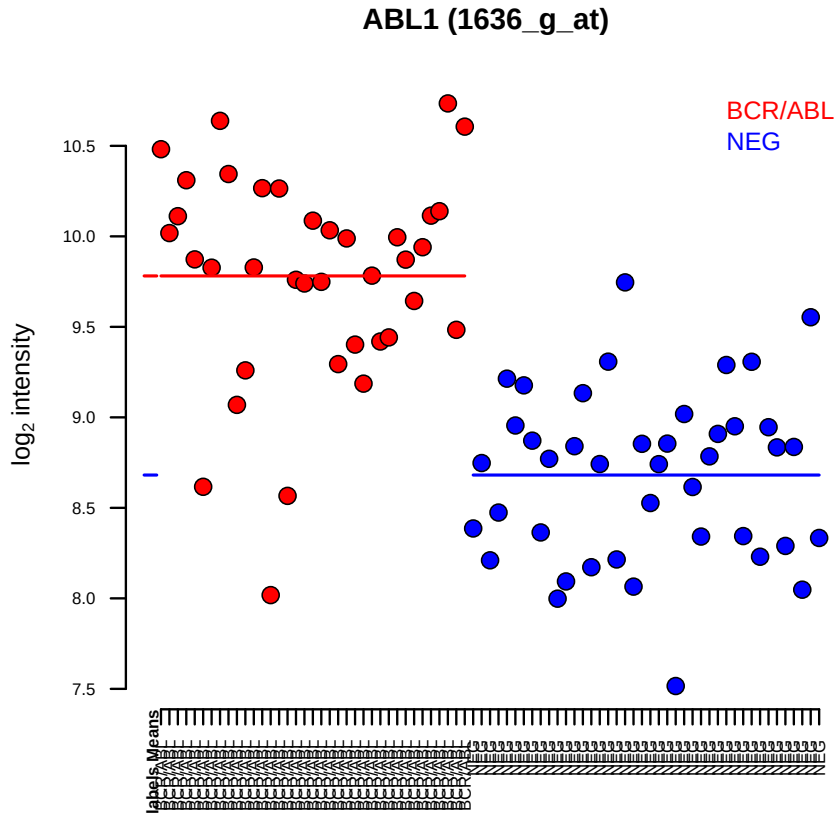
Some potentially interesting genes can be visualized using `plot1gene`. Here the most significant gene is plotted.

```
R> plot1gene(probesetId = rownames(tTestResult)[1], object = selBcrAblOrNeg, groups = "mol.biol",
```



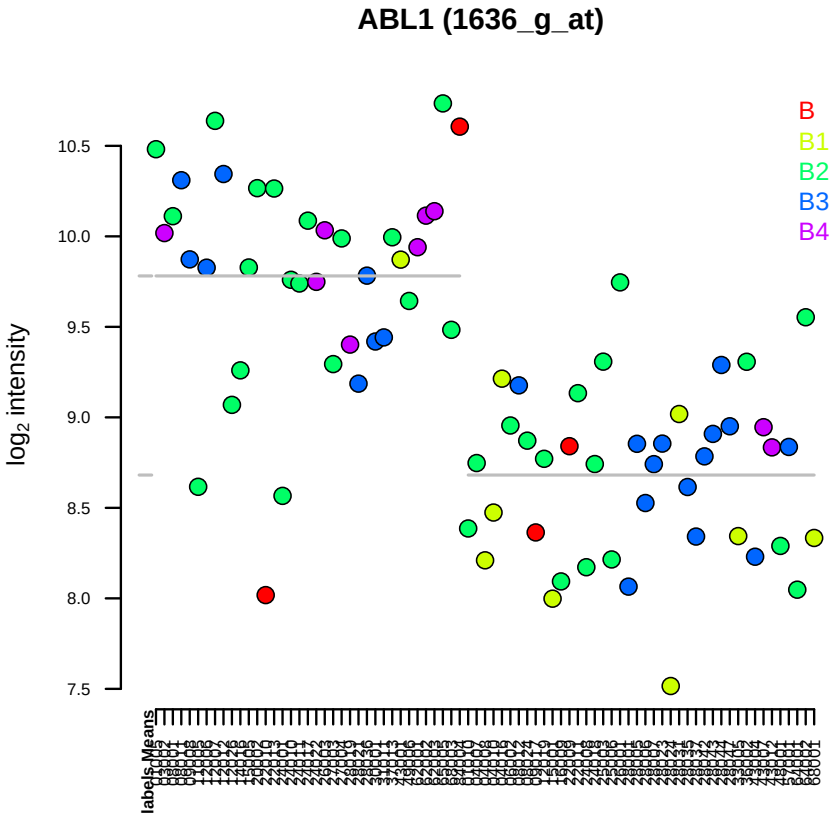
There are some variations possible on the default `plot1gene` function. For example, the labels of x-axis can be changed or omitted.

```
R> plot1gene(probesetId = rownames(tTestResult)[1], object = selBcrAblOrNeg, groups = "mol.biol",
```



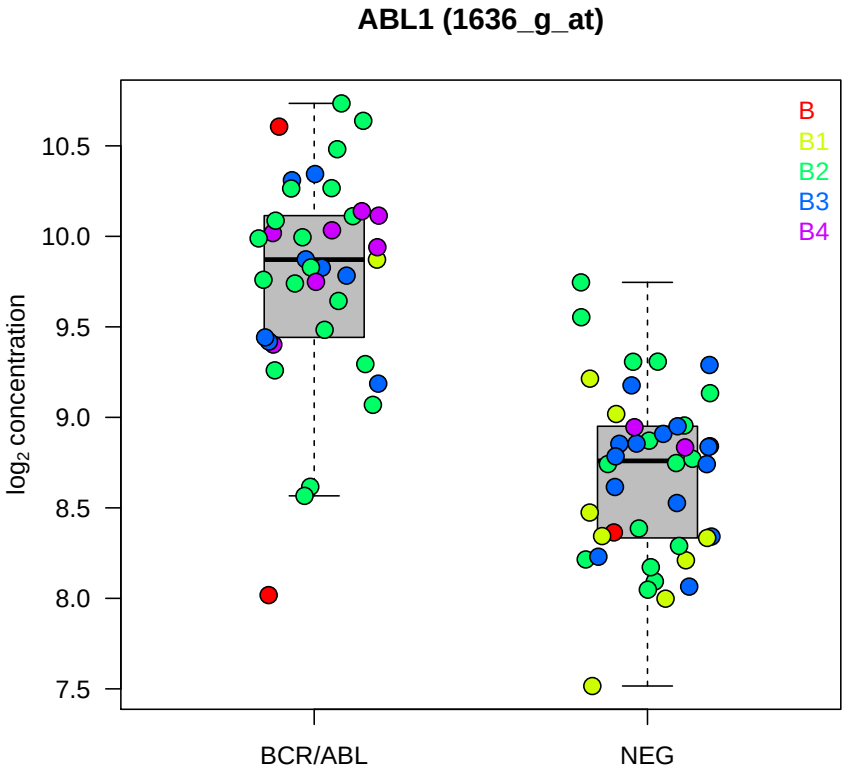
Another option is to color the samples by another categorical variable than used for ordering.

```
R> plot1gene(probesetId = rownames(tTestResult)[1], object = selBcrAblOrNeg, groups = "mol.biol",
```



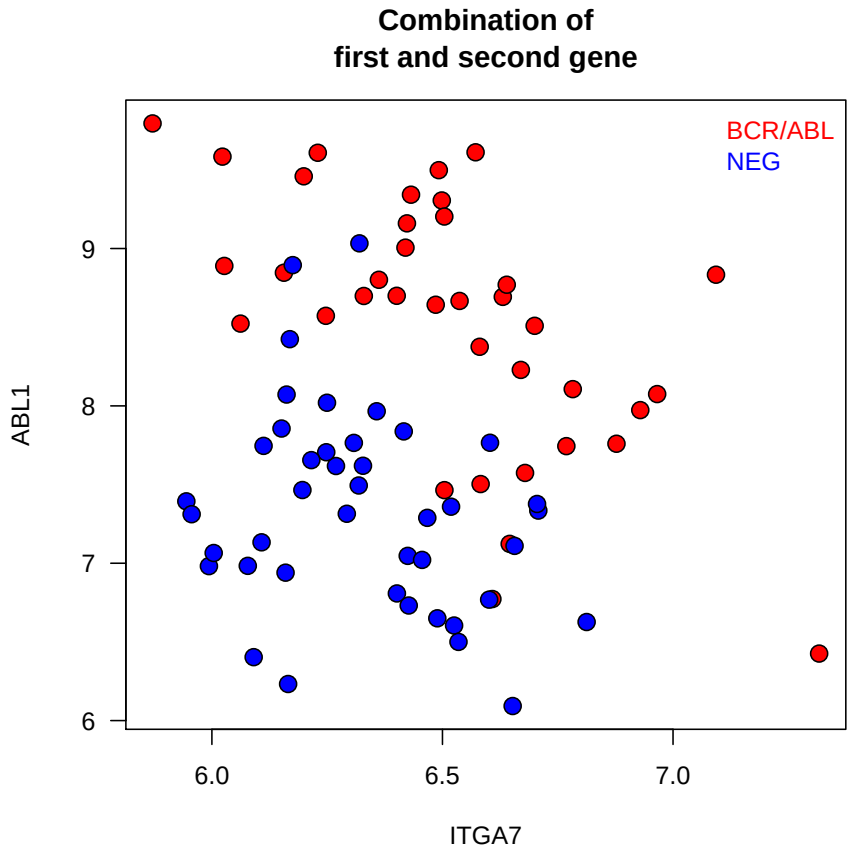
The above graphs plot one sample per tickmark in the x-axis. This is very useful to explore the data as one can directly identify interesting samples. If it is not interesting to know which sample has which expression level, one may want to plot in the x-axis not the samples but the groups of interest. It is possible to pass arguments to the boxplot function to custopmize the graph. For example the `boxwex` argument allows to reduce the width of the boxes in the plot.

```
R> boxPlot(probesetId = rownames(tTestResult)[1], object = selBcrAblOrNeg, boxwex = 0.3, groups =
```



6.2 Plot the expression levels of two genes versus each other

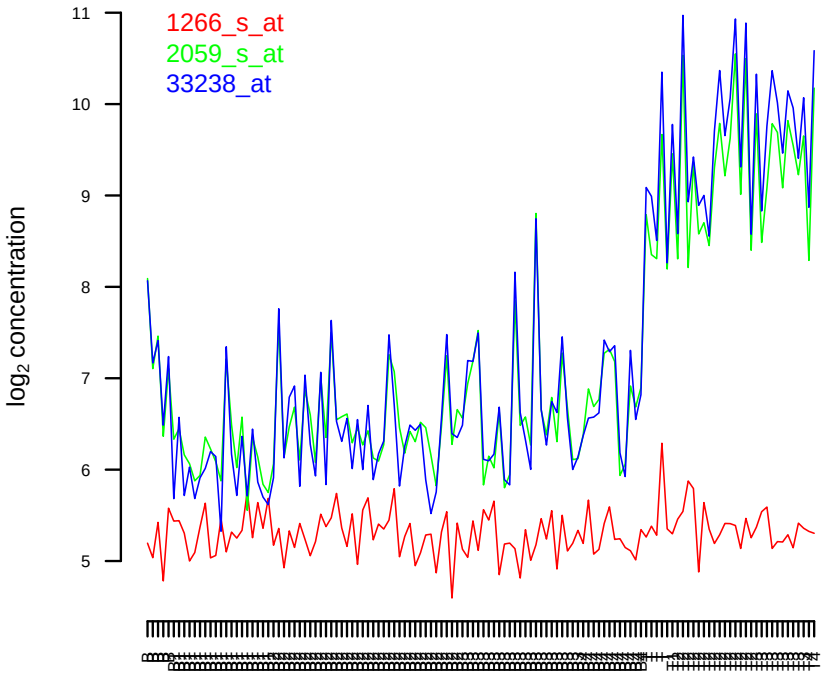
```
R> plotCombination2genes(geneSymbol1 = featResultLasso$topList[1, 1], geneSymbol2 = featResultLass
```



6.3 Plot expression line profiles of multiple genes/probesets across samples

Multiple genes can be plotted simultaneously on a graph using line profiles. Each line reflects one gene and are colored differenly. As an example, here three probesets that measure the gene `LCK`, found to be differentially expressed between B- and T-cells. Apparently, one probeset does not measure the gene appropriately.

```
R> myGeneSymbol <- "LCK"
R> probesetPos <- which(myGeneSymbol == featureData(ALL)$SYMBOL)
R> myProbesetIds <- featureNames(ALL)[probesetPos]
R> profilesPlot(object = ALL, probesetIds = myProbesetIds, orderGroups = "BT", sampleIDs = "BT")
```


1266_s_at
2059_s_at
33238_at
log2 concentration
11
10
9
8
7
6
5

6.4 Smoothscatter plots

It may be of interest to look at correlations between samples. As each dot represents a gene, there are typically many dots. It is therefore wise to color the dots in a density dependent way.

```
R> plotComb2Samples(ALL, "11002", "01003", xlab = "a T-cell", ylab = "another T-cell")
```

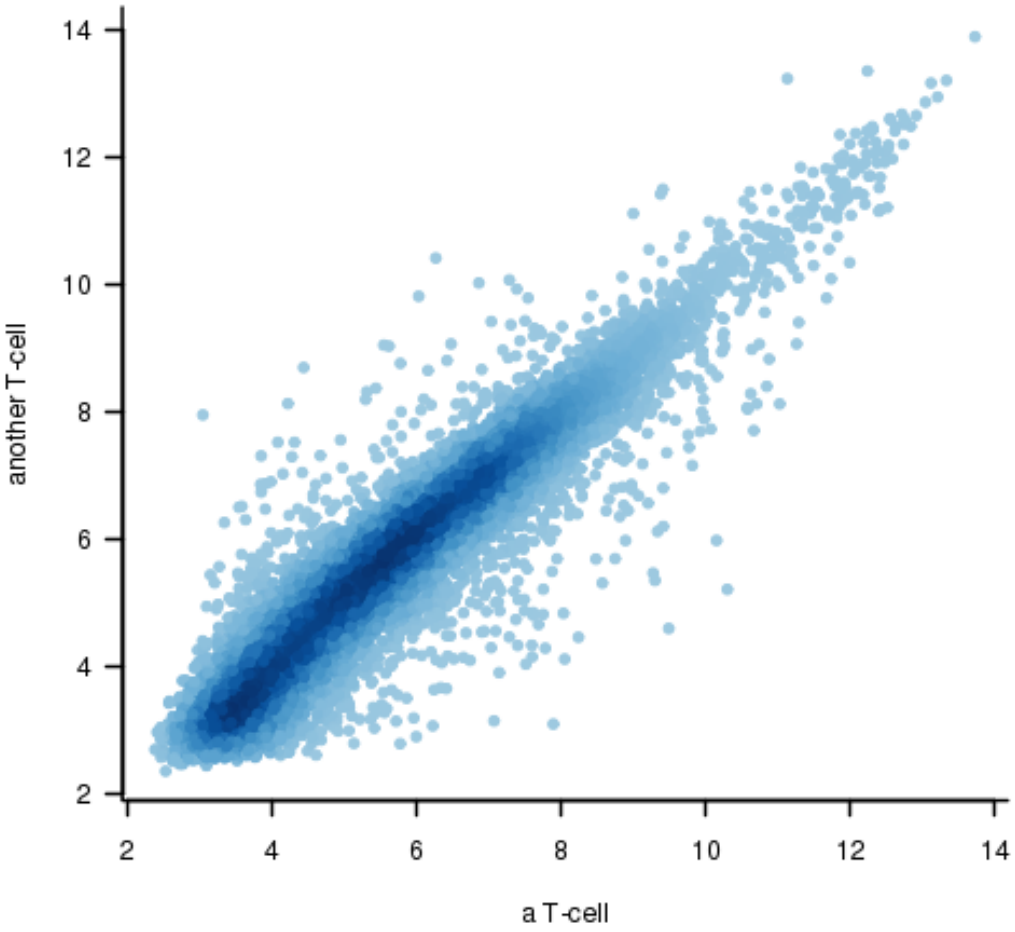


Figure 1: Correlations in gene expression profiles between two T-cell samples (samples 11002 and 01003).

If there are outlying genes, one can label them by their gene symbol by specifying the expression intervals (X- or Y- axis or both) that contain the genes to be highlighted using `trsholdX` and `trsholdY`.

```
R> plotComb2Samples(ALL, "84004", "01003", trsholdX = c(10, 12), trsholdY = c(4, 6), xlab = "a B-c
```

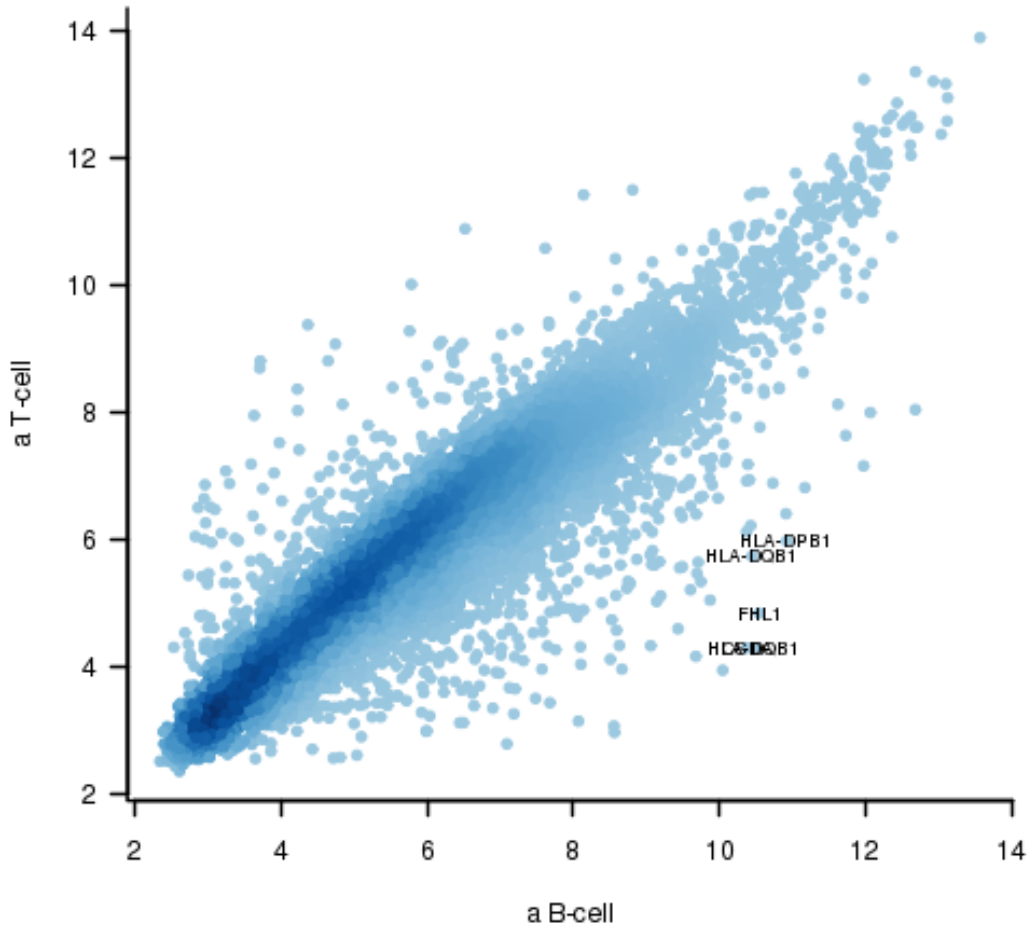


Figure 2: Correlations in gene expression profiles between a B-cell and a T-cell (samples 84004 and 01003). Some potentially interesting genes are indicated by their gene symbol.

One can also show multiple pairwise comparisons in a pairwise scatterplot matrix.

```
R> plotCombMultSamples(exprs(ALL)[, c("84004", "11002", "01003")])
```

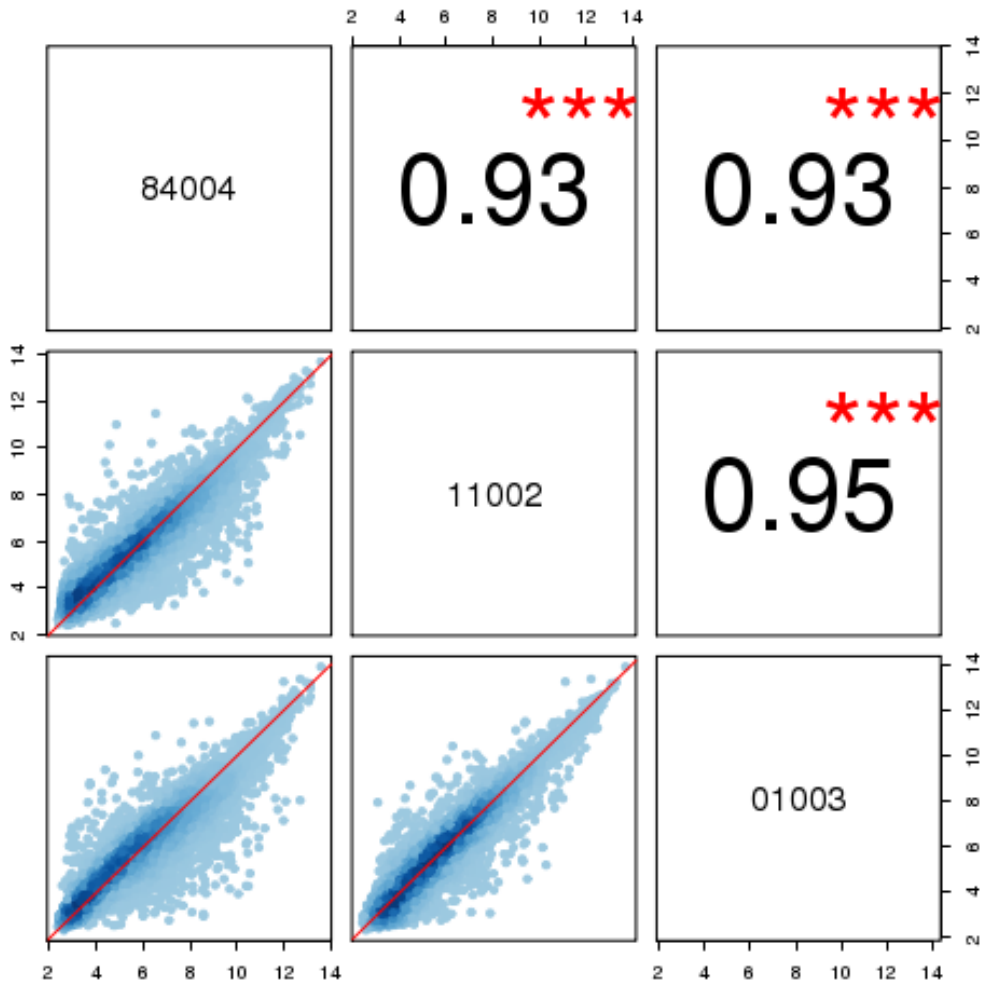


Figure 3: Correlations in gene expression profiles between a B-cell and two T-cell samples (respectively samples 84004, 11002 and 01003).

6.5 Gene lists of log ratios

When analyzing treatments that are primarily interesting relative to a control treatment, it may be of value to look at the log ratios of several treatments (in columns) for a selected list of genes (in rows).

```
R> ALL$BTtype <- as.factor(substr(ALL$BT, 0, 1))
R> ALL2 <- ALL[, ALL$BT != "T1"]
R> ALL2$BTtype <- as.factor(substr(ALL2$BT, 0, 1))
R> tTestResult <- tTest(ALL, "BTtype", probe2gene = FALSE)
R> topGenes <- rownames(tTestResult)[1:20]
R> LogRatioALL <- computeLogRatio(ALL2, reference = list(var = "BT", level = "B"))
R> a <- plotLogRatio(e = LogRatioALL[topGenes, ], openFile = FALSE, tooltipvalues = FALSE, device
```

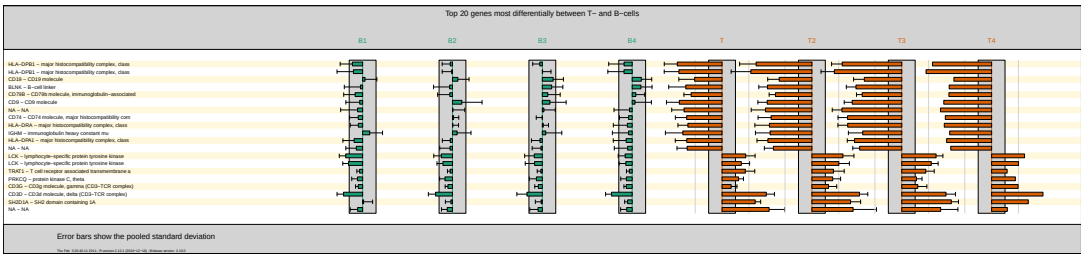


Figure 4: Log ratios of the 20 genes that are most differentially expressed between B-cell and two T-cells.

The following example demonstrates how to display log ratios for four compounds for which gene expression was measured on four timepoints.

```
R> load(system.file("examples", "esetExampleTimeCourse.rda", package = "a4"))
R> logRatioEset <- computeLogRatio(esetExampleTimeCourse, within = "hours", reference = list(var =
R> idx <- order(pData(logRatioEset)$compound, pData(logRatioEset)$hours)
R> logRatioEset <- logRatioEset[, idx]
R> cl <- "TEST"
R> compound <- "COMPOUND"
R> shortvarnames <- unique(interaction(pData(logRatioEset)$compound, pData(logRatioEset)$hours))
R> shortvarnames <- shortvarnames[-grep("DMSO", shortvarnames), drop = TRUE]
R> plotLogRatio(e = logRatioEset, mx = 1, filename = "logRatioOverallTimeCourse.pdf", gene.fontsiz
     level = "DMSO"), device = "pdf")
```

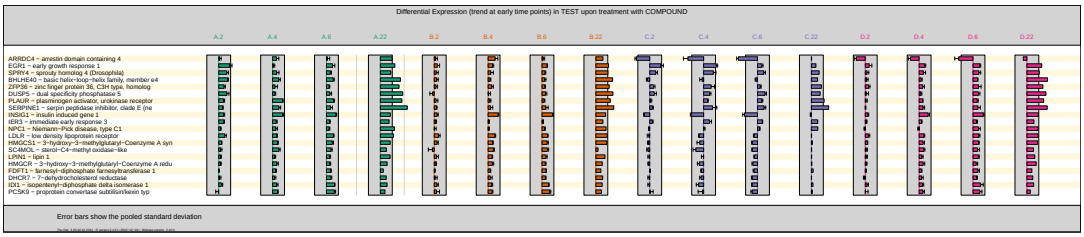


Figure 5: Log ratios for four compounds at four time points (for 20 genes).

7 Pathway analysis

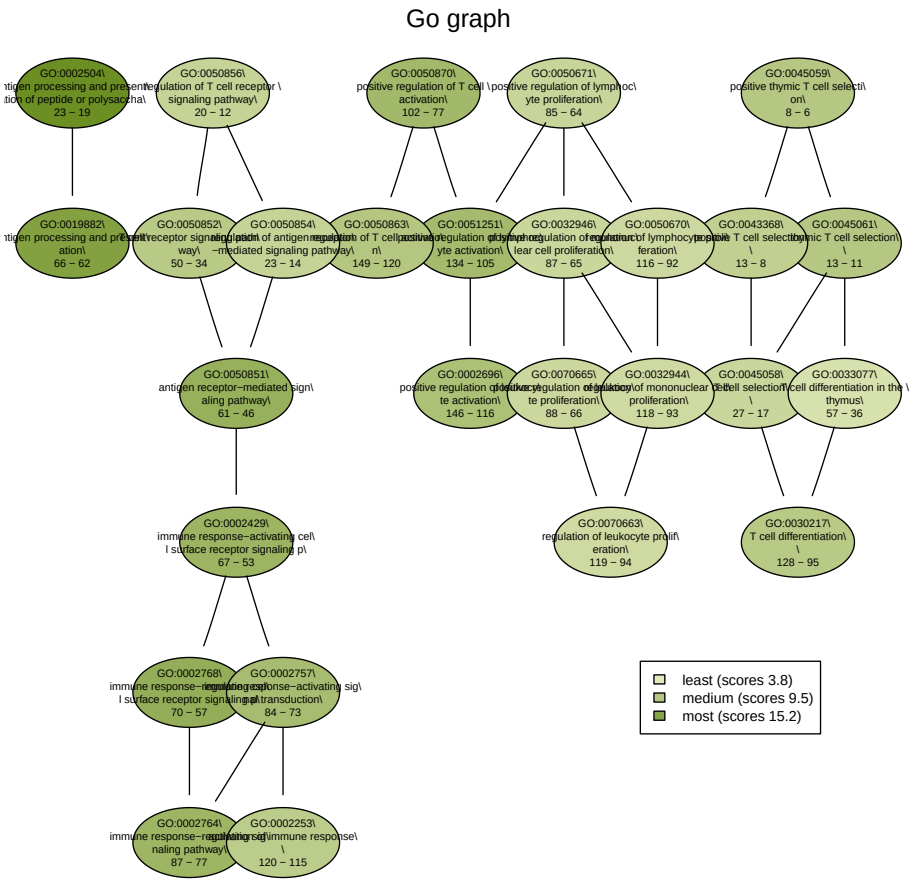
7.1 Minus log p

```
R> require(MLP)
R> labels <- as.factor(ifelse(regexpr("^B", as.character(pData(ALL)$BT)) == 1, "B", "T"))
R> pData(ALL)$BT2 <- labels
R> limmaResult <- limmaTwoLevels(object = ALL, group = "BT2")
R> pValues <- limmaResult@MArrayLM$p.value
R> pValueNames <- fData(ALL)[rownames(pValues), "ENTREZID"]
R> pValues <- pValues[, 2]
R> names(pValues) <- pValueNames
R> pValues <- pValues[!is.na(pValueNames)]

R> geneSet <- getGeneSets(species = "Human", geneSetSource = "GOBP", entrezIdentifiers = names(pVa
R> tail(geneSet, 3)

$`GO:2000027`
 [1] "265"    "324"    "367"    "581"    "596"    "627"    "650"    "652"    "655"    "657"    "83
[56] "22943"  "26585"  "54361"  "54829"  "54880"  "55897"  "59277"  "89780"  "128408" "387628" "65

$`GO:2000036`
[1] "655"

$`GO:2000045`
 [1] "595"    "1026"   "1030"   "1739"   "2765"   "3265"   "3364"   "4683"   "5371"   "5527"   "59

R> mlpOut <- MLP(geneSet = geneSet, geneStatistic = pValues, minGenes = 5, maxGenes = 100, rowPerm

R> pdf(file = "GOgraph.pdf")
R> plot(mlpOut, type = "GOgraph", nRow = 25)
R> dev.off()

X11cairo
       2
```

7.2 Gene set enrichment analysis

8 Software used

- R version 2.12.1 (2010-12-16), x86_64-pc-linux-gnu
- Locale: LC_CTYPE=en_US.utf8, LC_NUMERIC=C, LC_TIME=en_US.utf8, LC_COLLATE=en_US.utf8, LC_MONETARY=en_US.utf8, LC_MESSAGES=en_US.utf8, LC_PAPER=en_US.utf8, LC_NAME=en_US.utf8, LC_ADDRESS=en_US.utf8, LC_TELEPHONE=en_US.utf8, LC_MEASUREMENT=en_US.utf8, LC_IDENTIFICATION=en_US.utf8
- Base packages: base, datasets, graphics, grDevices, grid, methods, splines, stats, tools, utils
- Other packages: a4~0.0-20, a4Base~0.0-26, a4Classif~0.0-1, a4Core~0.0-1, a4Preproc~0.0-2, a4Reporting~0.0-3, affy~1.28.0, ALL~1.4.7, annaffy~1.22.0, annotate~1.28.0, AnnotationDbi~1.12.0, Biobase~2.10.0, bitops~1.0-4.1, Cairo~1.4-5, Category~2.16.0, caTools~1.11, class~7.3-3, cluster~1.13.2, DBI~0.2-5, gdata~2.8.1, genefilter~1.32.0, glmnet~1.5.1, gmodels~2.15.0, GO.db~2.4.5, GOstats~2.16.0, gplots~2.8.0, graph~1.28.0, gtools~2.6.2, hgu95av2.db~2.4.5, ipred~0.8-8, KEGG.db~2.4.5, KernSmooth~2.23-4, lattice~0.19-17, limma~3.6.9, MASS~7.3-9, Matrix~0.999375-46, mlbench~2.1-0, MLInterfaces~1.30.0, MLP~0.99.4, mpm~1.0-16, multtest~2.6.0, nlcv~0.2-0, nnet~7.3-1, org.Hs.eg.db~2.4.6, pamr~1.50, plotrix~3.0-5, randomForest~4.6-2, RColorBrewer~1.0-2, rda~1.0.2, Rgraphviz~1.28.0, rj~0.5.2-1, ROCR~1.0-4, rpart~3.1-48, RSQLite~0.9-4, survival~2.36-2, varSelRF~0.7-3, xtable~1.5-6



- Loaded via a namespace (and not attached): affyio˜1.18.0, GSEABase˜1.12.2, mboost˜2.0-9, preprocessCore˜1.12.0, RBGL˜1.26.0, rJava˜0.8-8, XML˜2.6-0