Using the a4 package

Willem Talloen, Tobias Verbeke

April 24, 2017

Contents

1 Introduction

The a4 suite of packages is a suite for convenient analysis of Affymetrix microarray experiments which supplements Goehlmann and Talloen (2010). The suite currently consists of several packages which are centered around particular tasks:

- a4Preproc: package for preprocessing of microarray data. Currently the only function in the package adds complementary annotation information to the ExpressionSet objects (in function addGeneInfo). Many of the subsequent analysis functions rely on the presence of such information.
- a4Core: package made to allow for easy interoperability with the nlcv package which is currently being developed on R-Forge at http://r-forge.r-project.org/projects/nlcv.
- a4Base: all basic functionality of the a4 suite
- a4Classif: functionality for classification work that has been split off a.o. in order to reduce a4Base loading time
- a4Reporting: a package which provides reporting functionality and defines xtable-methods that are foreseen for tables with hyperlinks to public gene annotation resources.

This document provides an overview of the typical analysis workflow for such microarray experiments using functionality of all of the mentioned packages.

2 Preparation of the Data

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

```
R> library(a4)
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, withinClas
```

2.1 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)
```

2.2 Some data manipulation

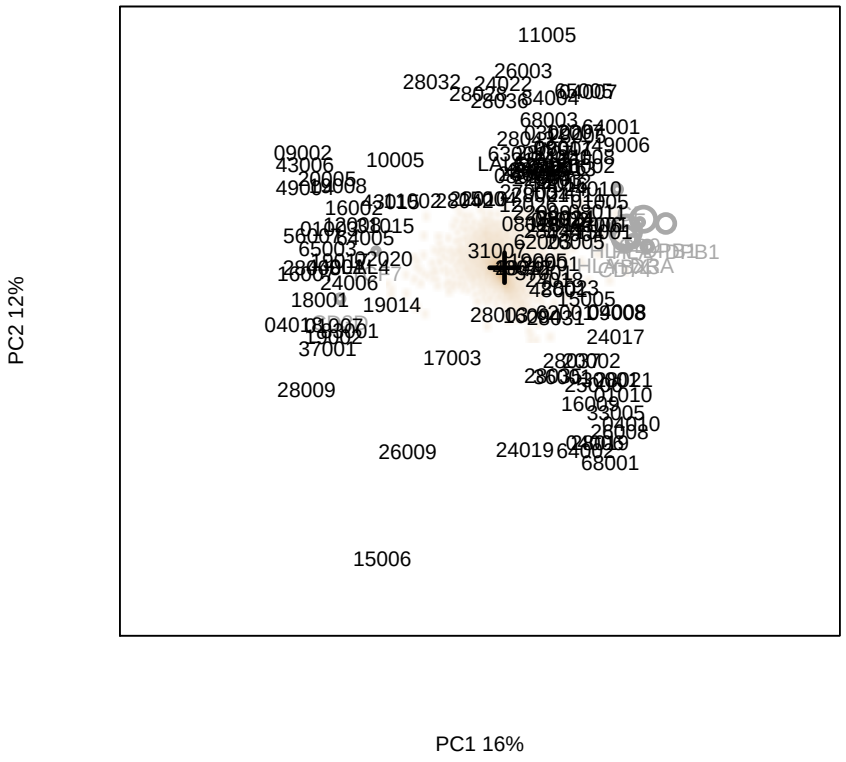
The ALL data consists out of samples obtained from two types of cells with very distinct expression profiles; B-cells and T-cells. To have a more subtle signal, gene expression will also be compared between the BCR/ABL and the NEG group within B-cells only. To this end, we create the expressionSet bcrAblOrNeg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

```
R> Bcell <- grep("^B", as.character(ALL$BT))  # create B-Cell subset for ALL
R> subsetType <- "BCR/ABL"  # other subsetType can be "ALL/AF4"
R> bcrAblOrNegIdx <- which(as.character(ALL$mol) %in% c("NEG", subsetType))
R> bcrAblOrNeg <- ALL[, intersect(Bcell, bcrAblOrNegIdx)]
R> bcrAblOrNeg$mol.biol <- factor(bcrAblOrNeg$mol.biol)
```

3 Unsupervised data exploration

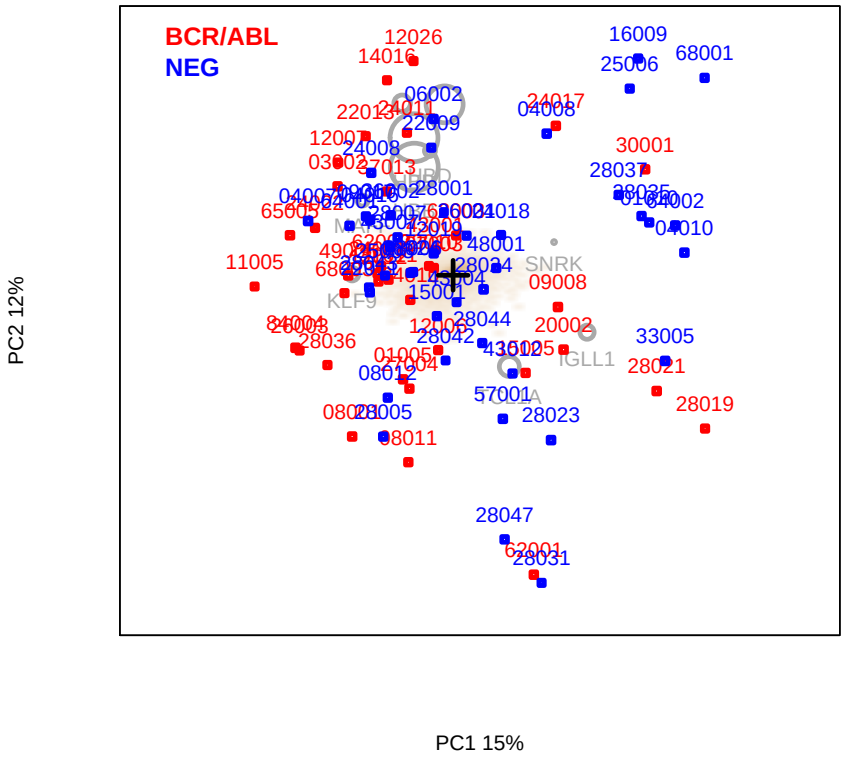
Spectral maps are very powerful techniques to get an unsupervised picture of how the data look like. A spectral map of the `ALL` data set shows that the B- and the T-subtypes cluster together along the x-axis (the first principal component). The plot also indicates which genes contribute in which way 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")
R>
R>   # optional argument settings
R>   #    plot.mpm.args=list(label.tol = 12, zoom = c(1,2), do.smoothScatter = TRUE),
R>   #    probe2gene = TRUE)
```



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)
```



4 Filtering

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

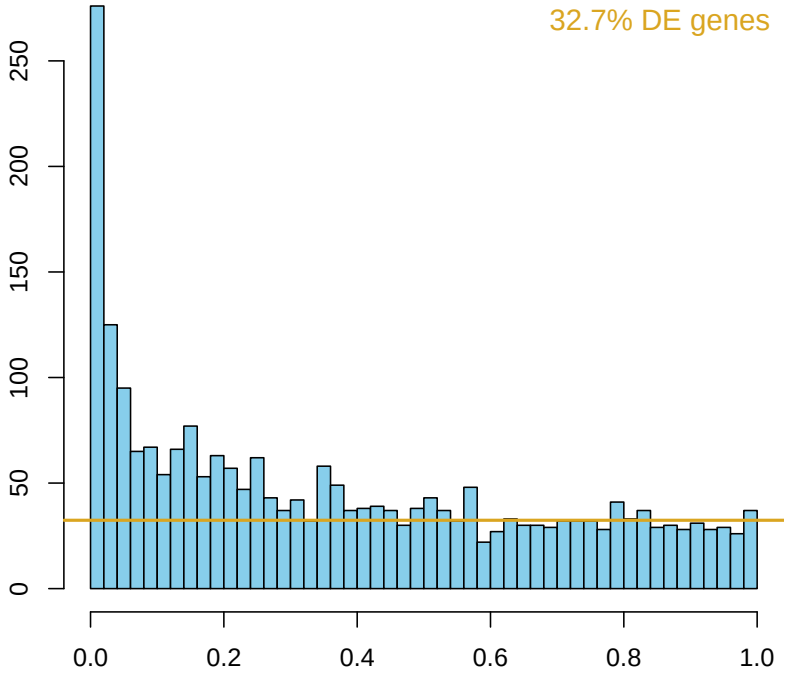
```
R> selBcrAblOrNeg <- filterVarInt(object = bcrAblOrNeg)
R> propSelGenes <- round((dim(selBcrAblOrNeg)[1]/dim(bcrAblOrNeg)[1])*100,1)
```

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

5 Detecting differential expression

5.1 T-test

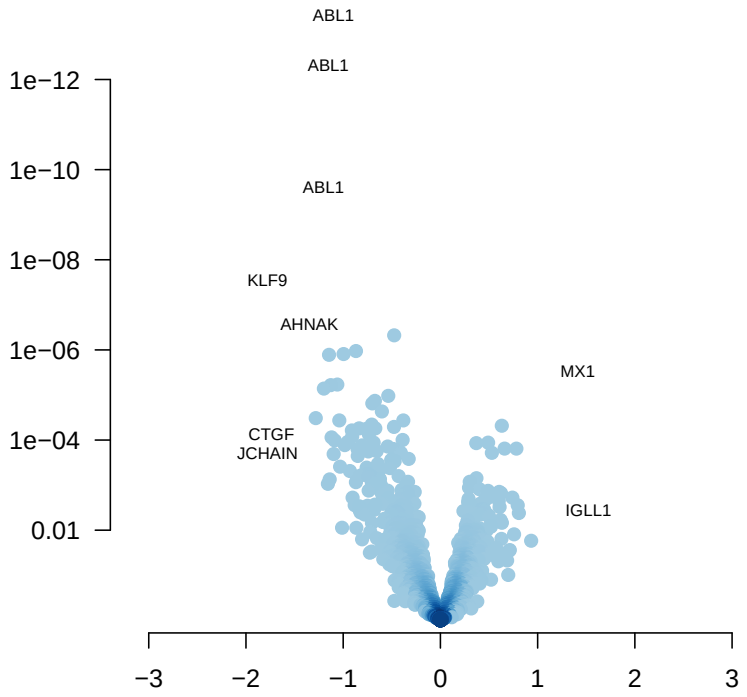
```
R> tTestResult <- tTest(selBcrAblOrNeg, "mol.biol")

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.

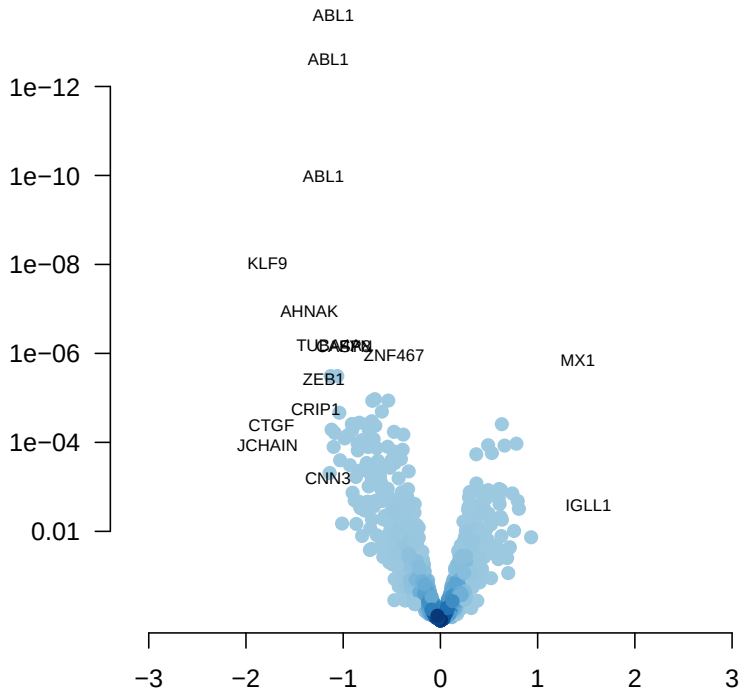
```
R> tabTTest <- topTable(tTestResult, n = 10)
R> xtable(tabTTest,
     caption="The top 5 features selected by an ordinary t-test.",
   label ="tablassoClass")

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



5.2 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)
R> # histPvalue(limmaResult)
R> # propDEgenes(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) # 1st is (Intercept)
```

Gene	logFC	AveExpr	P.Value	adj.P.Val	GENENAME
ABL1	-1.10	9.20	0.00	0.00	ABL proto-oncogene 1, non-receptor tyr
ABL1	-1.15	9.00	0.00	0.00	ABL proto-oncogene 1, non-receptor tyr
ABL1	-1.20	7.90	0.00	0.00	ABL proto-oncogene 1, non-receptor tyr
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 proto-oncogene, Src family tyrosin
CASP8	-1.00	8.04	0.00	0.00	caspase 8
ZNF467	-0.48	7.14	0.00	0.00	zinc finger protein 467
MX1	1.41	6.73	0.00	0.00	MX dynamin like GTPase 1

5.3 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`.

```
R>
```

6 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.

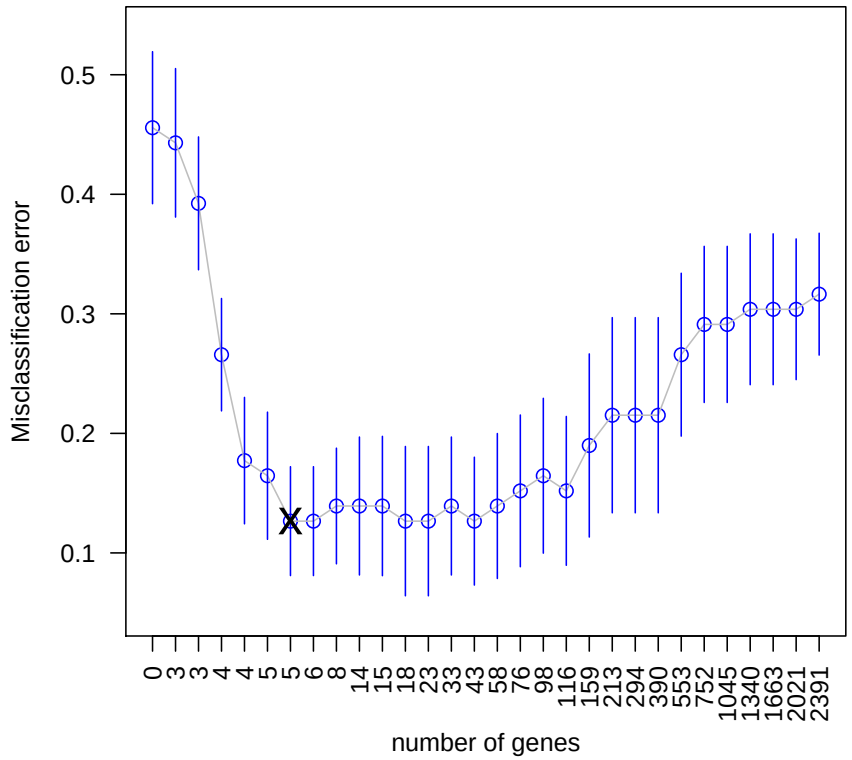
Personally, we like to apply four different approaches; PAM, RandomForest, forward filtering in combination with various classifiers, and LASSO.

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.

6.1 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.")
```



6.2 Random forest

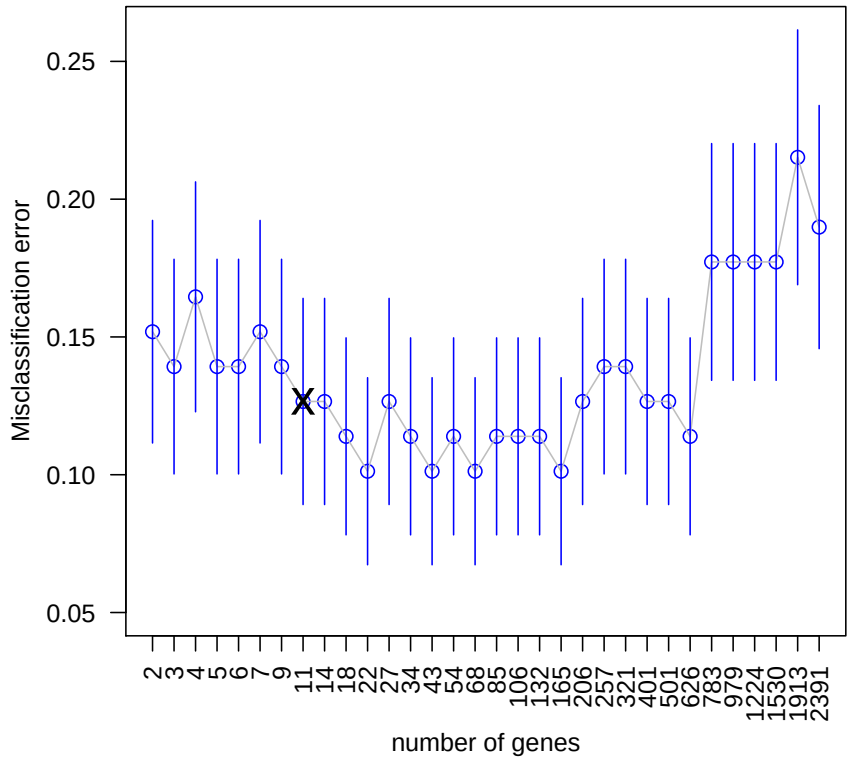
Random forest with variable importance filtering (Breiman 2001, Diaz-Uriarte 2006) applies multivariate and dependent feature selection. Be cautious when interpreting its outcome, as the obtained results are unstable and sometimes overoptimistic.

```
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 importance.")
```

	GeneSymbol
1635_at	ABL1
1636_g_at	ABL1
33440_at	ZEB1
36591_at	TUBA4A
37006_at	JCHAIN
37027_at	AHNAK

Table 1: Features selected by Random Forest variable importance.

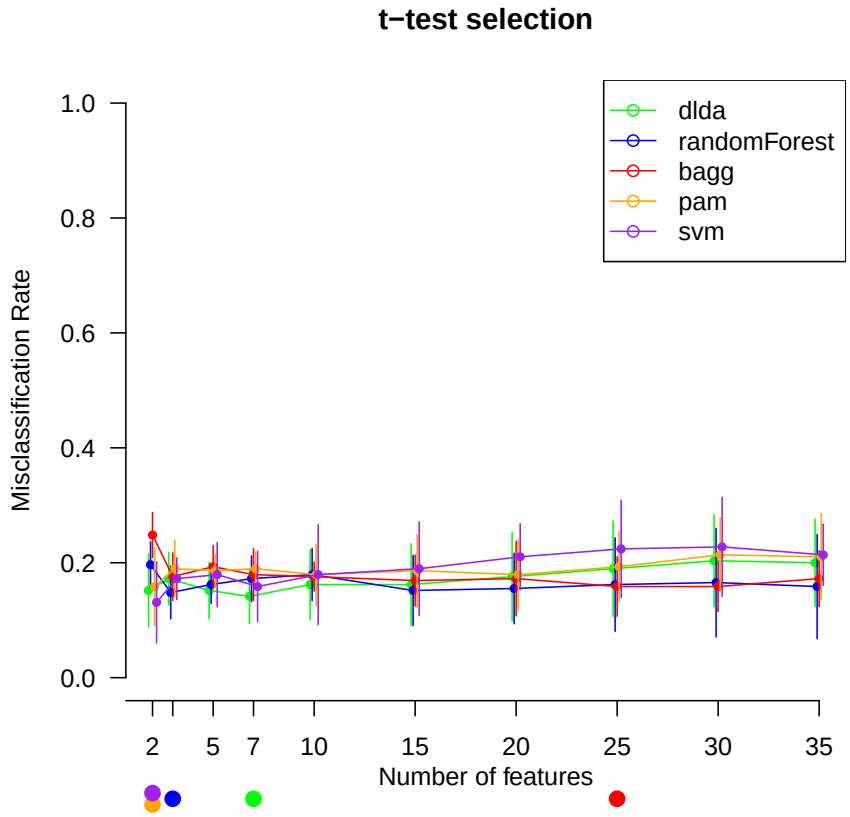
```
R>
```



6.3 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.

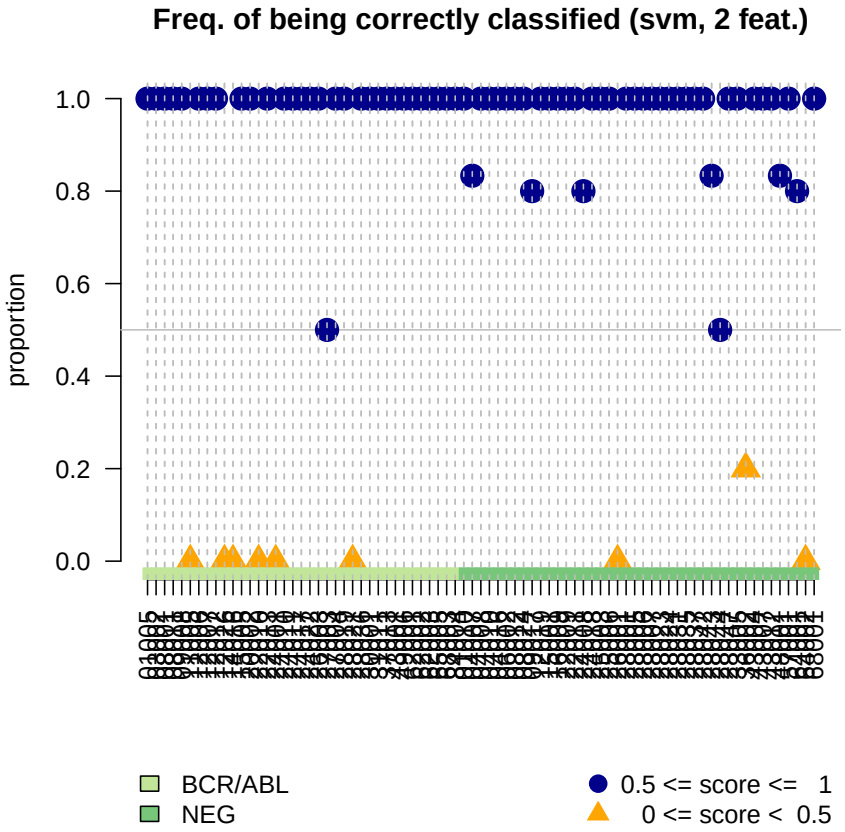
```
R> mcrPlot_TT <- mcrPlot(nlcvTT, plot = TRUE, optimalDots = TRUE,
     layout = TRUE, main = "t-test selection")
```



	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).

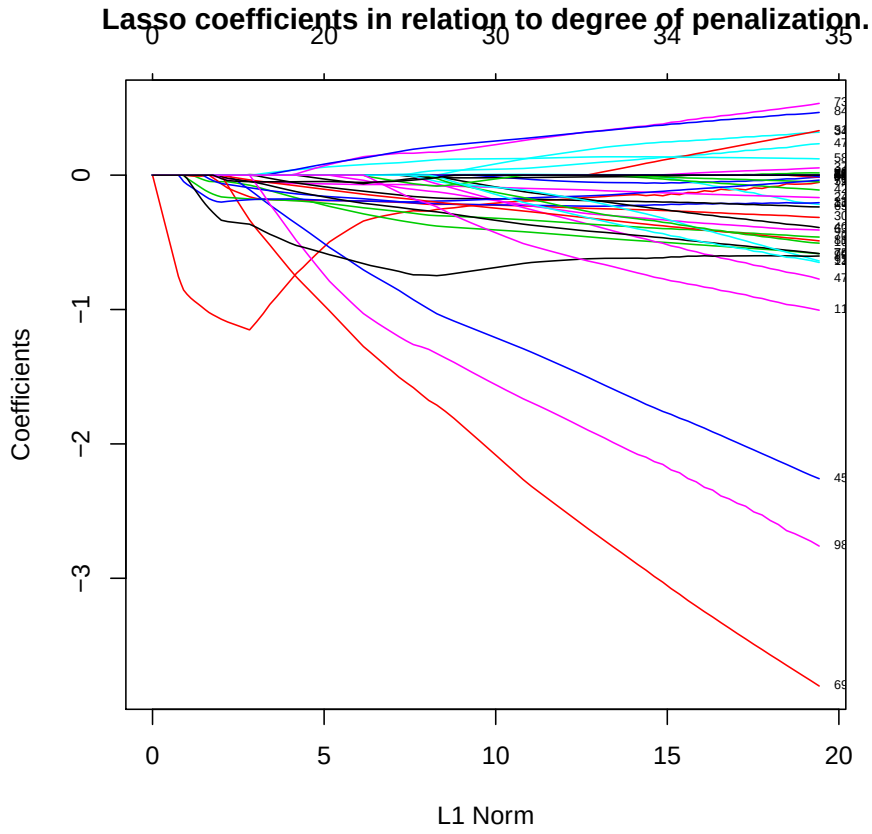
```
R> scoresPlot(nlcvTT, tech = "svm", nfeat = 2)
```



6.4 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 penalization.")
R> featResultLasso <- topTable(resultLasso, n = 15)
```



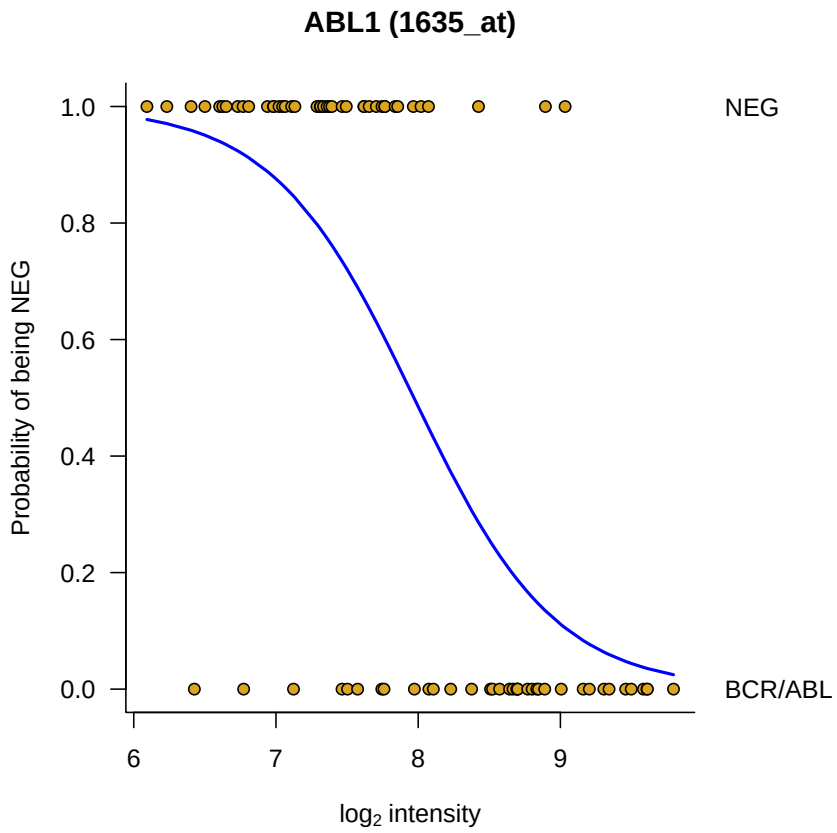
Gene	Coefficient
ITGA7	-3.80
ABL1	-2.76
TCL1B	-2.26
RAB32	-1.01
CHD3	-0.77
SERPINE2	-0.65
NFATC1	-0.64
ZNF467	-0.60
YES1	-0.58
ANXA1	-0.58
PTDSS1	0.53
PTPRJ	-0.51
F13A1	-0.49
DSTN	0.47
ALDH1A1	-0.46

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

6.5 Logistic regression

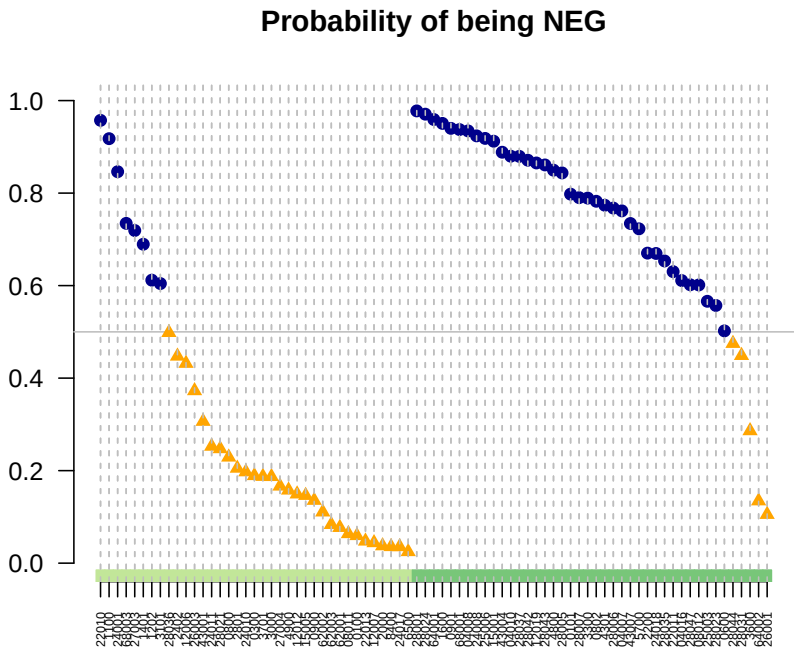
Logistic regression is used for predicting the probability to belong to a certain class in binary classification problems.

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

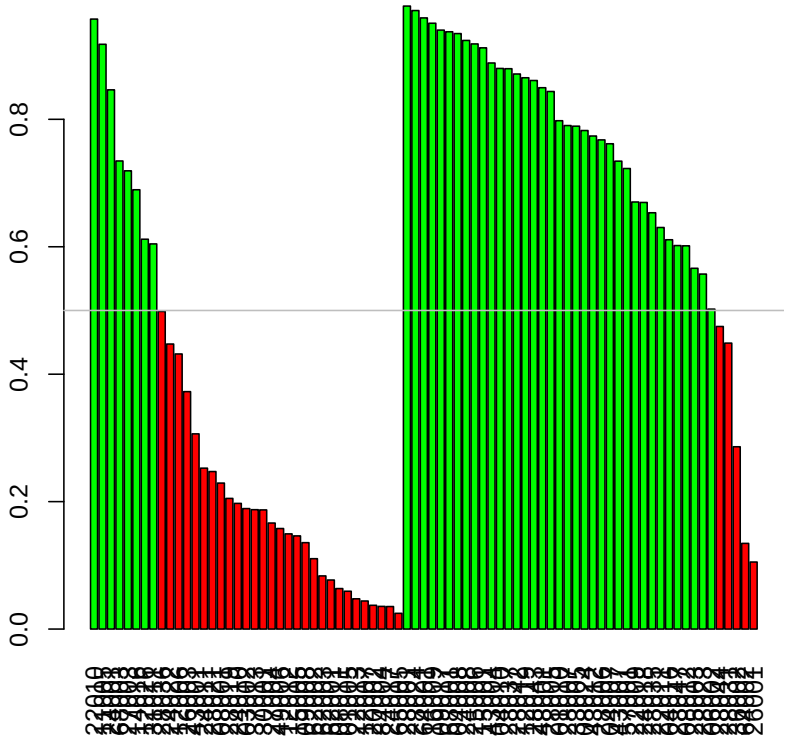


The obtained probabilities can be plotted with `ProbabilitiesPlot`. A horizontal line indicates the 50% threshold, and samples that have a higher probability than 50% are indicated with blue dots. Apparently, using the expression of the gene `ABL1`, quite a lot of samples predicted to with a high probability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

```
R> probabilitiesPlot(proportions = logRegRes$fit, classVar = logRegRes$y,
     sampleNames = rownames(logRegRes), main = "Probability of being NEG")
```



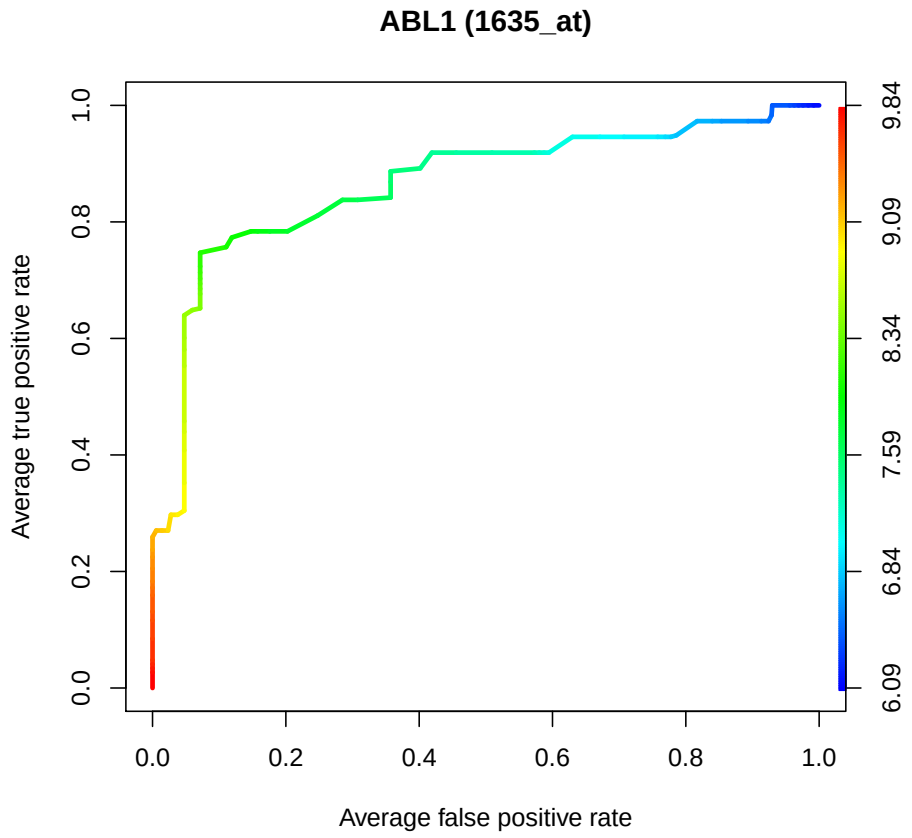

```
R> probabilitiesPlot(proportions = logRegRes$fit, classVar = logRegRes$y, barPlot= TRUE,
     sampleNames = rownames(logRegRes), main = "Probability of being NEG")
```



6.6 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")
```

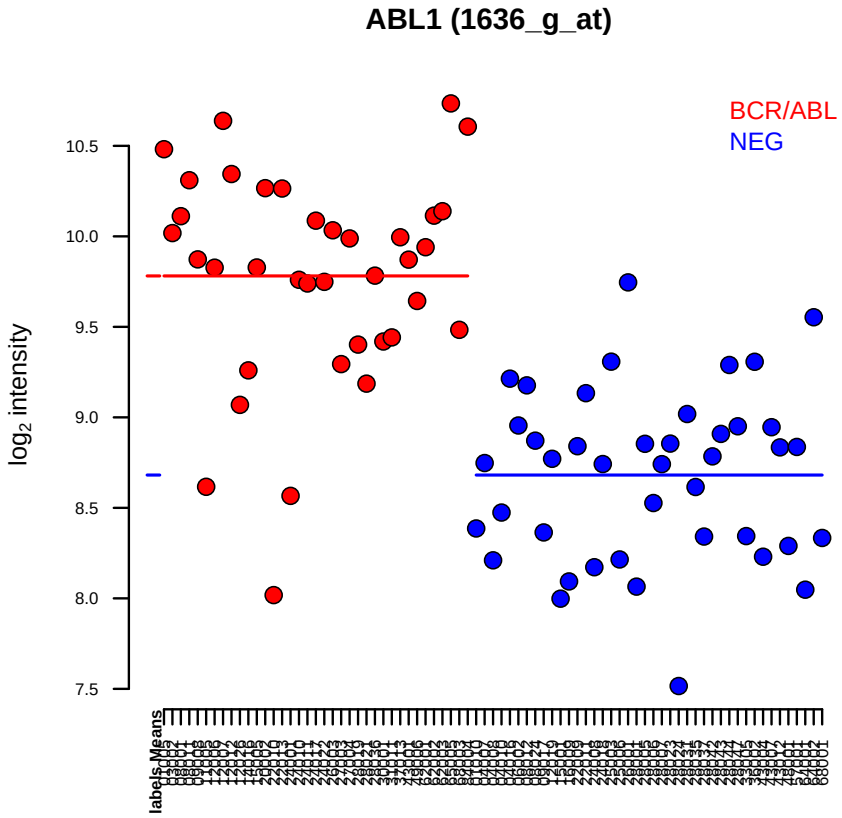


7 Visualization of interesting genes

7.1 Plot the expression levels of one gene

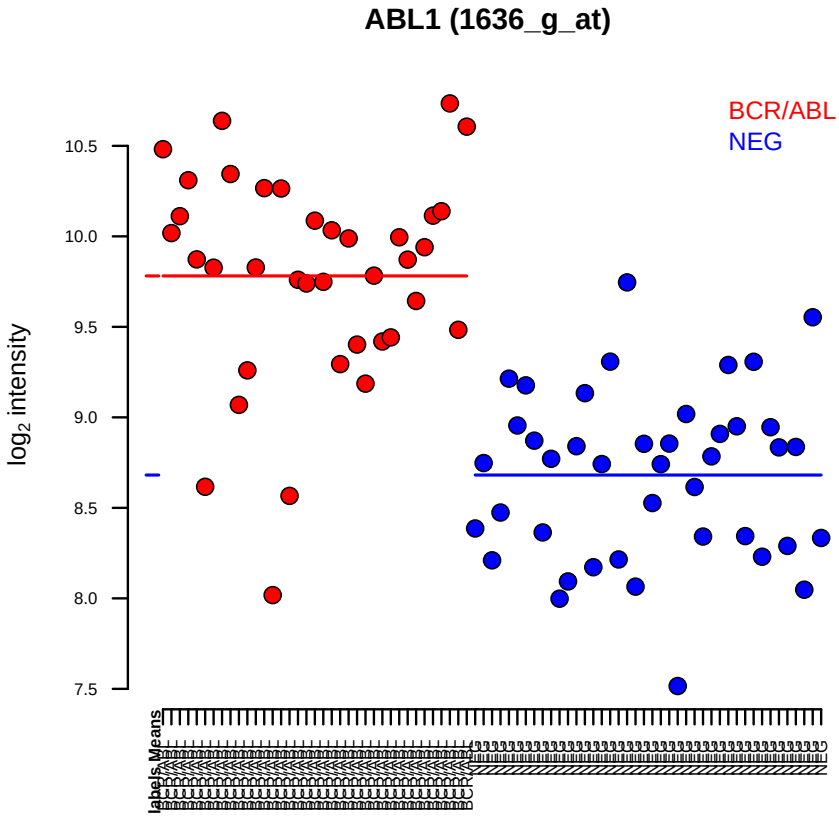
Some potentially interesting genes can be visualized using `plot1gene`. Here the most significant gene is plotted.

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



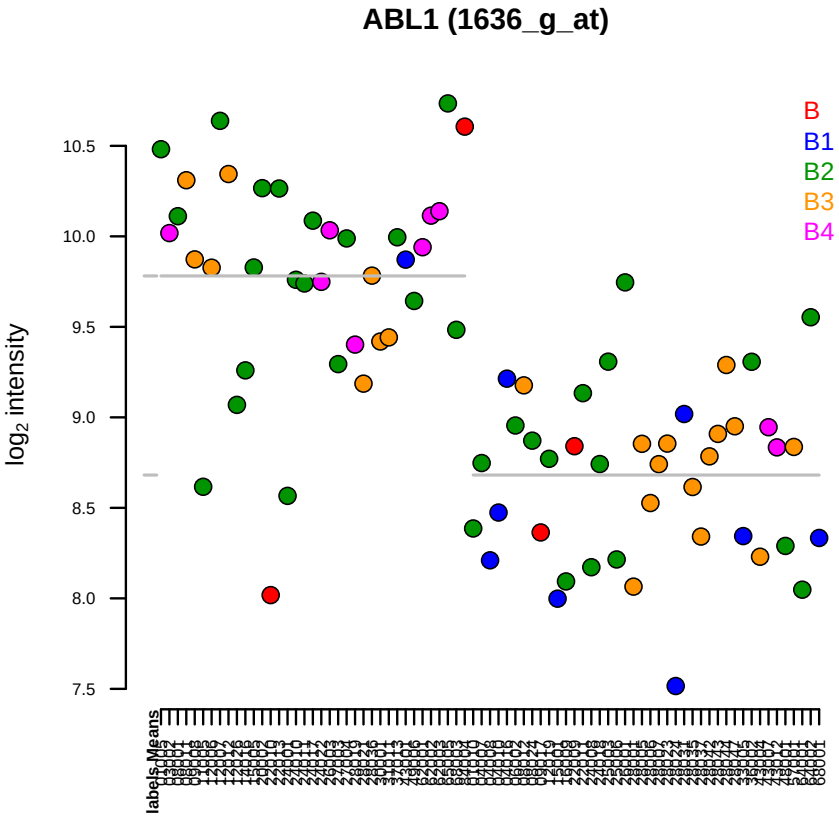
There are some variations possible on the default `plot1gene` function. For example, the labels of x-axis can be changed or omitted.

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



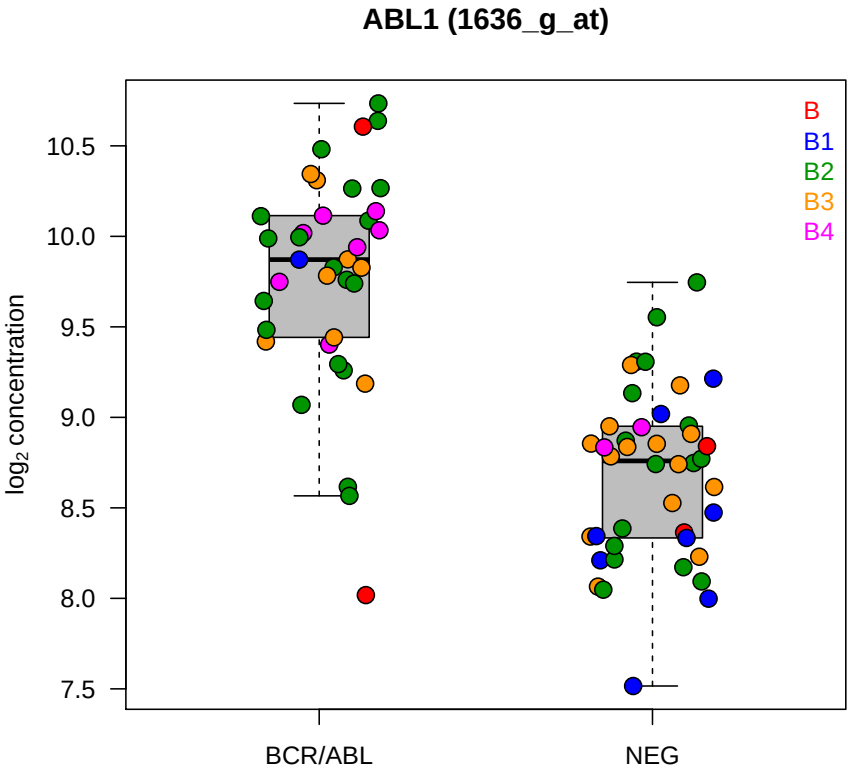
Another option is to color the samples by another categorical variable than used for ordering.

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



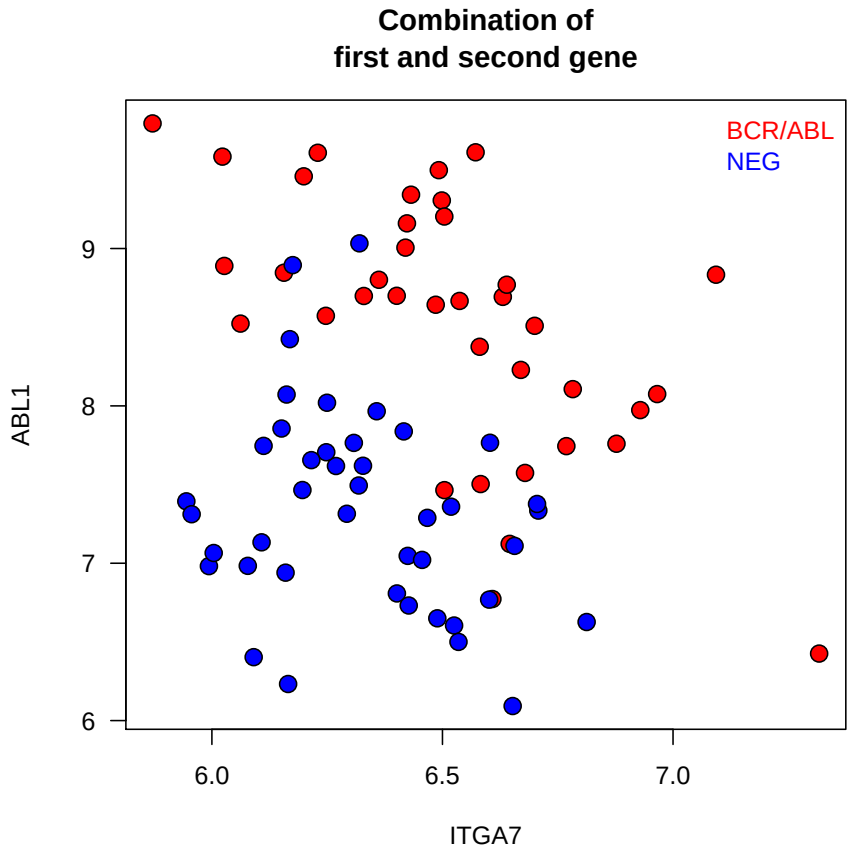
The above graphs plot one sample per tickmark in the x-axis. This is very useful to explore the data as one can directly identify interesting samples. If it is not interesting to know which sample has which expression level, one may want to plot in the x-axis not the samples but the groups of interest. 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 = "mol.biol", colgroups = "BT", legendPos = "topright")
```



7.2 Plot the expression levels of two genes versus each other

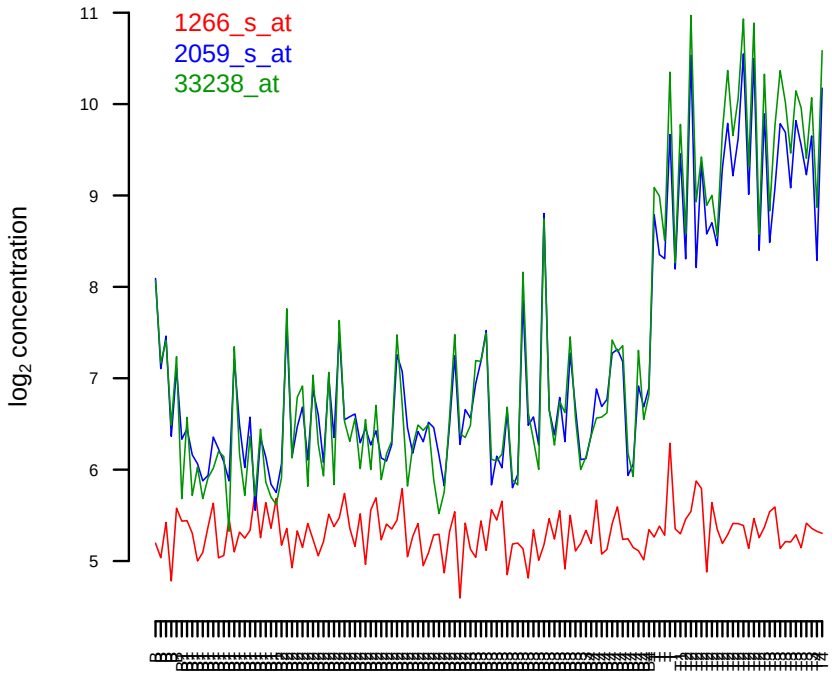
```
R> plotCombination2genes(geneSymbol1 = featResultLasso$topList[1, 1],
     geneSymbol2 = featResultLasso$topList[2, 1],
     object = bcrAblOrNeg, groups = "mol.biol",
     main = "Combination of\nfirst and second gene", addLegend = TRUE,
     legendPos = "topright")
```



7.3 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

7.4 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")
```

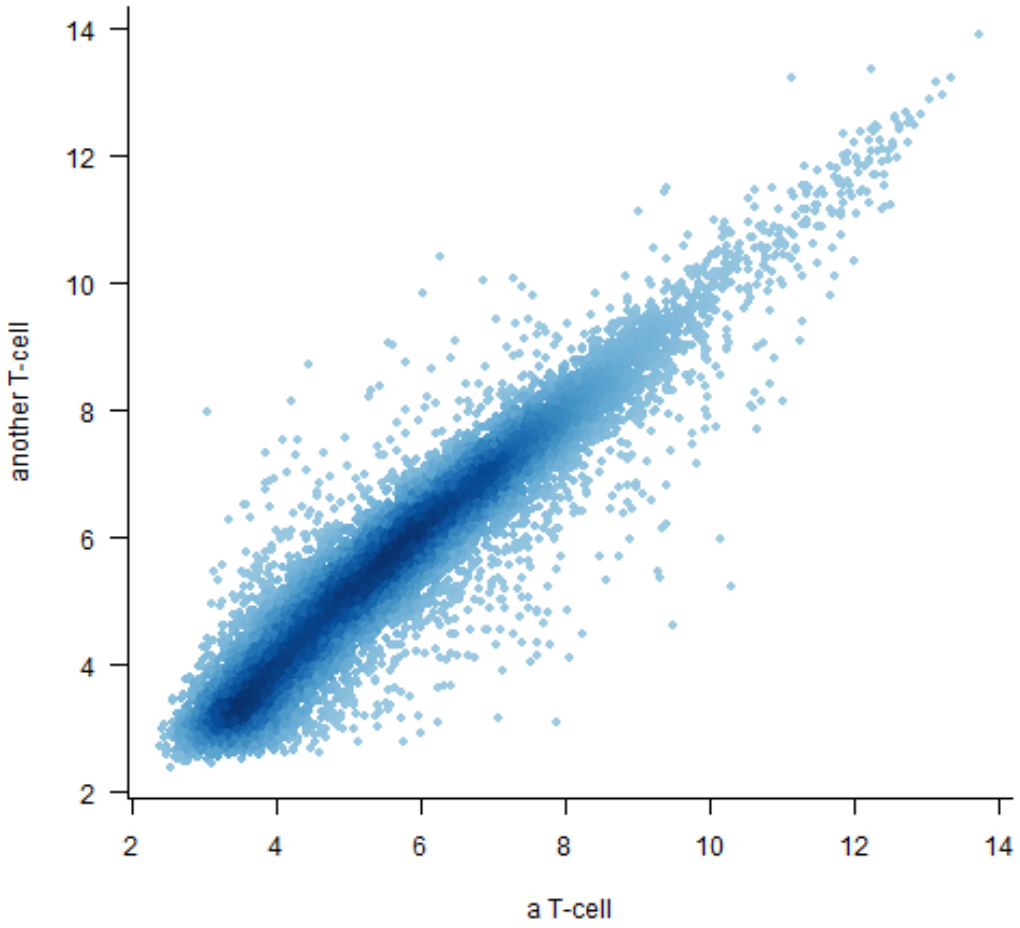


Figure 1: Correlations in gene expression profiles between two T-cell samples (samples 11002 and 01003).

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-cell", ylab = "a T-cell")
```

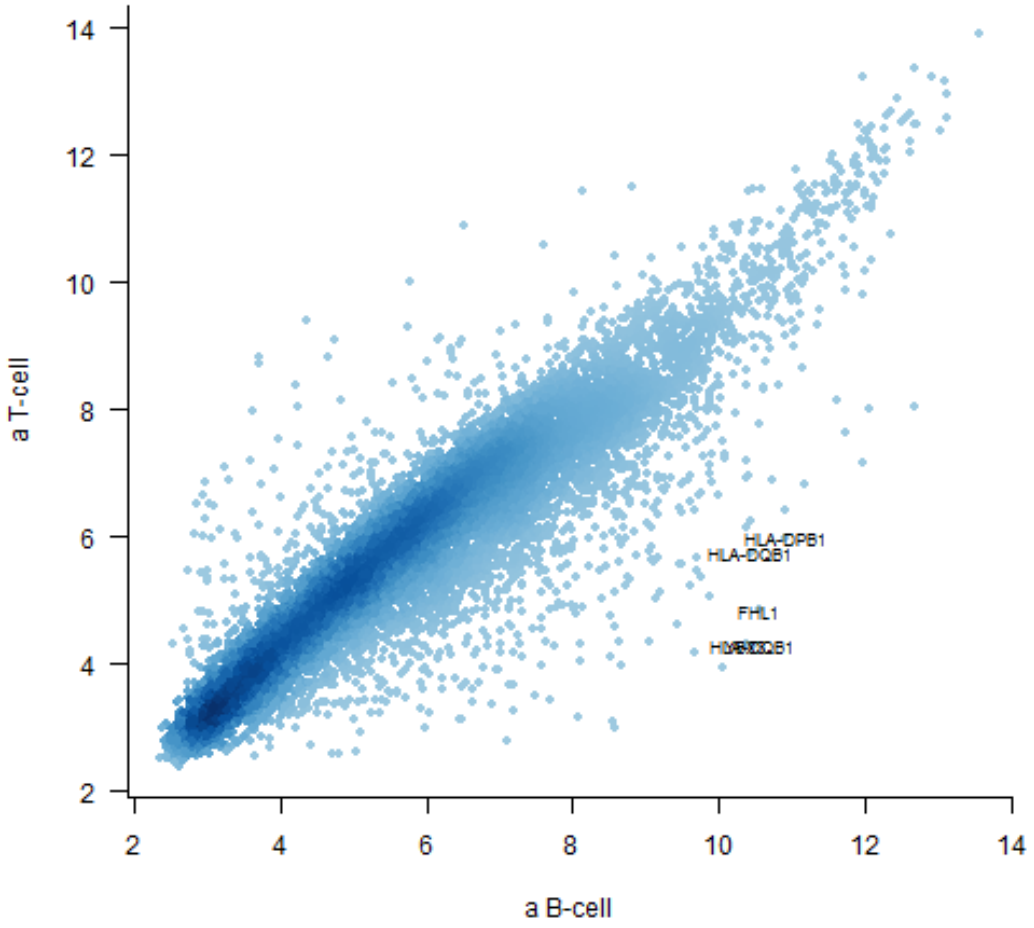


Figure 2: Correlations in gene expression profiles between a B-cell and a T-cell (samples 84004 and 01003). Some potentially interesting genes are indicated by their gene symbol.

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

```
R> plotCombMultSamples(exprs(ALL)[,c("84004", "11002", "01003")])
R> # text.panel= function(x){x, labels = c("a B-cell", "a T-cell", "another T-cell")})
```

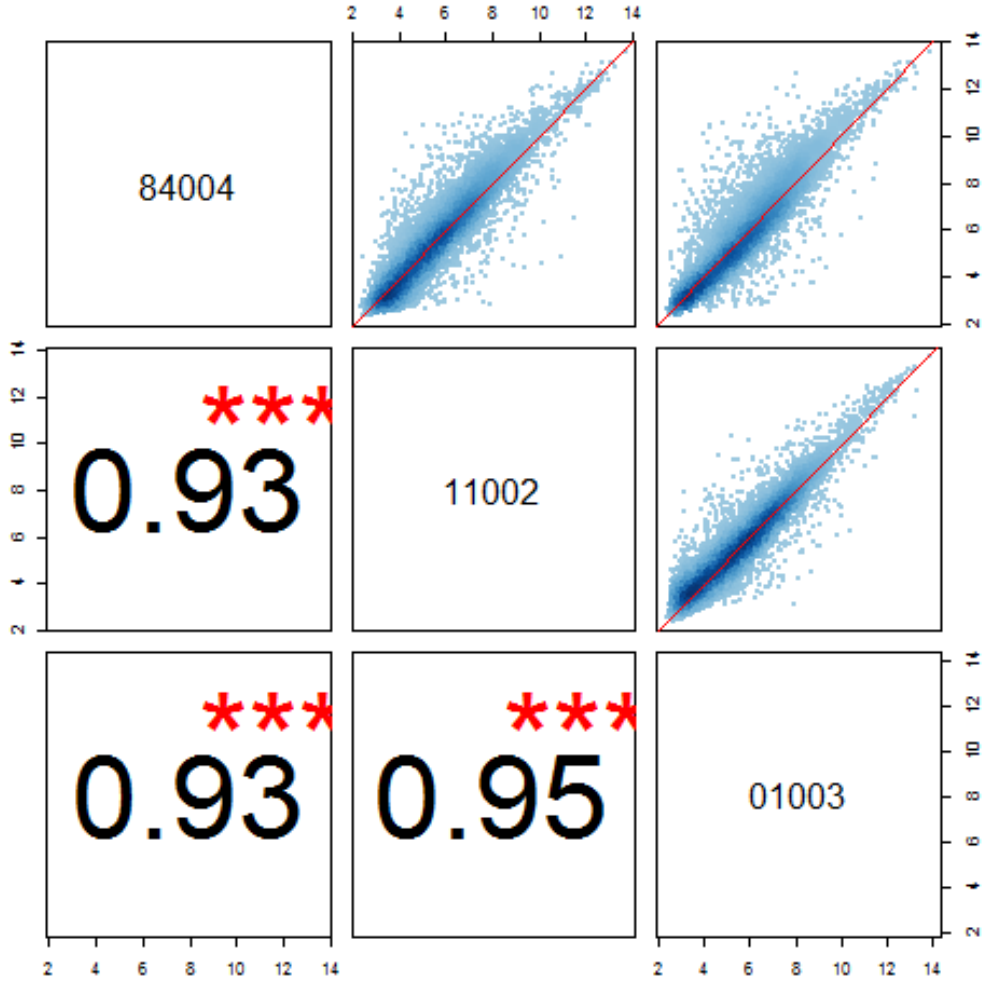


Figure 3: Correlations in gene expression profiles between a B-cell and two T-cell samples (respectively samples 84004, 11002 and 01003).

7.5 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$BT != 'T1']  # omit subtype T1 as it only contains one sample
R> ALL2$BTtype <- as.factor(substr(ALL2$BT,0,1)) # create a vector with only T and B
R> # Test for differential expression between B and T cells
R> tTestResult <- tTest(ALL, "BTtype", probe2gene = FALSE)
R> topGenes <- rownames(tTestResult)[1:20]
R> # plot the log ratios versus subtype B of the top genes
R> LogRatioALL <- computeLogRatio(ALL2, reference = list(var="BT", level="B"))
R> a <- plotLogRatio(e = LogRatioALL[topGenes,], openFile = FALSE, tooltipvalues = FALSE,
     device = "pdf", filename = "GeneLRlist",
     colorsColumnsBy = "BTtype",
     main = 'Top 20 genes most differentially between T- and B-cells',
     orderBy = list(rows = "hclust"), probe2gene = TRUE)
```

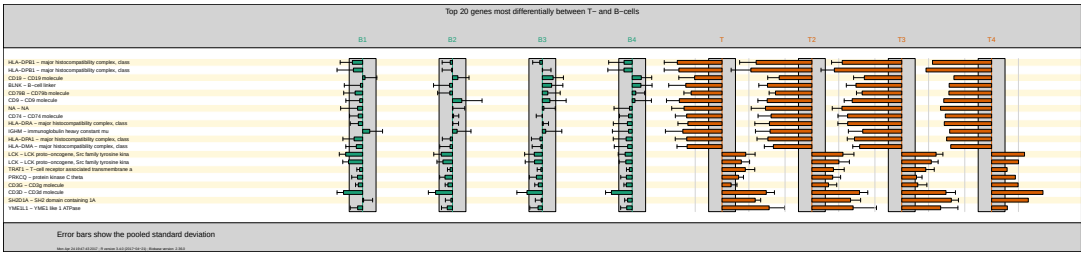


Figure 4: Log ratios of the 20 genes that are most differentially expressed between B-cell and two T-cells.

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("extdata", "esetExampleTimeCourse.rda", package = "a4"))
R>   logRatioEset <- computeLogRatio(esetExampleTimeCourse, within = "hours",
     reference = list(var = "compound", level = "DMSO"))
R>   # re-order
R>   idx <- order(pData(logRatioEset)$compound, pData(logRatioEset)$hours)
R>   logRatioEset <- logRatioEset[,idx]
R>   # plot LogRatioEset across all
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.fontsize = 8,
       orderBy = list(rows = "hclust", cols = NULL), colorsColumnsBy = c('compound'),
       within = "hours", shortvarnames = shortvarnames, exp.width = 1,
       main = paste("Differential Expression (trend at early time points) in",
           cl, "upon treatment with", compound),
       reference = list(var = "compound", level = "DMSO"), device = 'pdf')
```

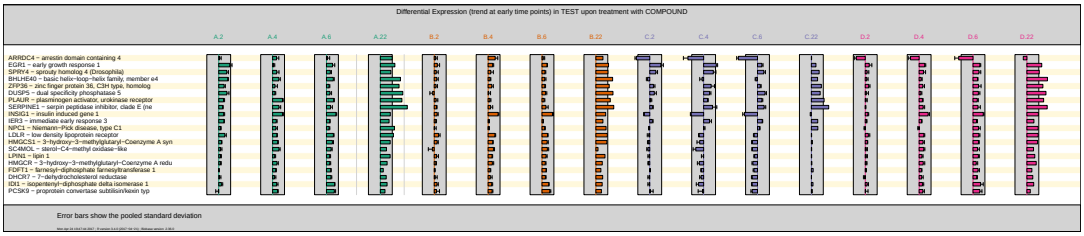


Figure 5: Log ratios for four compounds at four time points (for 20 genes).

8 Pathway analysis

8.1 Minus log p

The MLP method is one method of pathway analysis that is commonly used by the a4 suite user base. Although the method is explained in detail in the MLP package vignette we briefly walk throught the analysis steps using the same example dataset used in the preceding parts of the analysis. In order to detect whether certain gene sets are enriched in genes with low p values, we obtain the vector of p values for the genes and the corresponding relevant gene sets:

```
R> require(MLP)
R> # create groups
R> labels <- as.factor(ifelse(regexpr("^B", as.character(pData(ALL)$BT))==1, "B", "T"))
R> pData(ALL)$BT2 <- labels
R> # generate p-values
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(pValues)
 )
R> tail(geneSet, 3)

$`GO:2001303`
[1] "239" "246"

$`GO:2001304`
[1] "239"

$`GO:2001306`
[1] "239"
```

Next, we run the MLP analysis:

```
R> mlpOut <- MLP(
     geneSet = geneSet,
     geneStatistic = pValues,
     minGenes = 5,
     maxGenes = 100,
     rowPermutations = TRUE,
     nPermutations = 50,
     smoothPValues = TRUE,
     probabilityVector = c(0.5, 0.9, 0.95, 0.99, 0.999, 0.9999, 0.99999),
     df = 9)
```

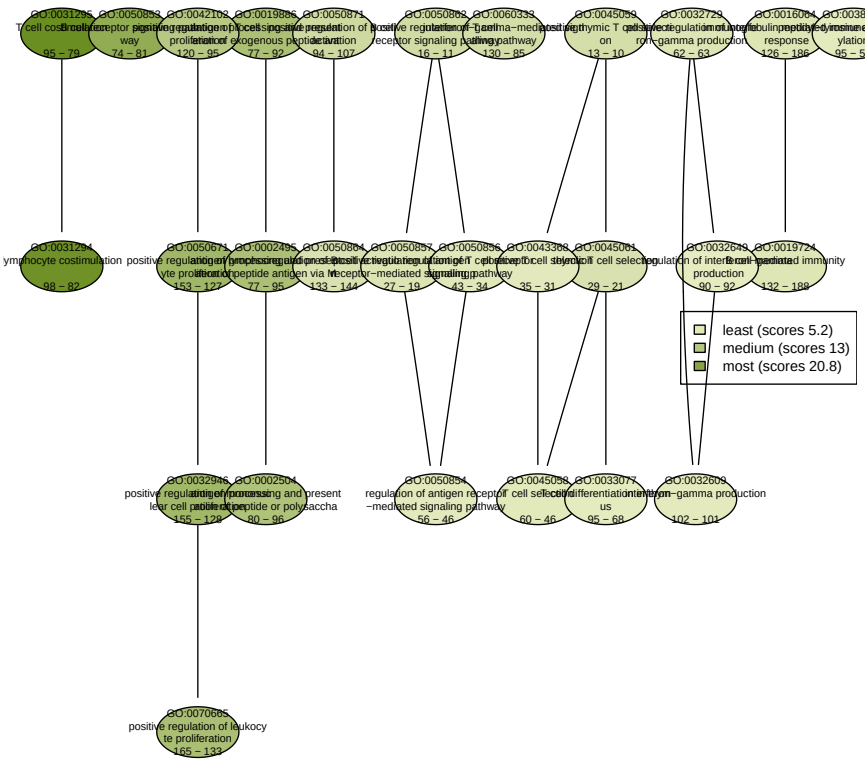
The results can be visualized in many ways, but for Gene Ontology based gene set definitions, the following graph may be useful:

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

9 Software used

- R version 3.4.0 (2017-04-21), `x86_64-w64-mingw32`

Go graph



- Locale: `LC_COLLATE=C`, `LC_CTYPE=English_United States.1252`, `LC_MONETARY=English_United States.1252`, `LC_NUMERIC=C`, `LC_TIME=English_United States.1252`
- Running under: `Windows Server 2012 R2 x64 (build 9600)`
- Matrix products: default
- Base packages: base, datasets, grDevices, graphics, grid, methods, parallel, stats, stats4, utils
- Other packages: ALL 1.17.0, AnnotationDbi 1.38.0, Biobase 2.36.0, BiocGenerics 0.22.0, Category 2.42.0, GO.db 3.4.1, GOstats 2.42.0, IRanges 2.10.0, KEGG.db 3.2.3, KernSmooth 2.23-15, MASS 7.3-47, MLInterfaces 1.56.0, MLP 1.24.0, Matrix 1.2-9, RColorBrewer 1.1-2, ROCR 1.0-7, Rgraphviz 2.20.0, S4Vectors 0.14.0, XML 3.98-1.6, a4 1.24.0, a4Base 1.24.0, a4Classif 1.24.0, a4Core 1.24.0, a4Preproc 1.24.0, a4Reporting 1.24.0, affy 1.54.0, annaffy 1.48.0, annotate 1.54.0, cluster 2.0.6, foreach 1.4.3, gdata 2.17.0, genefilter 1.58.0, glmnet 2.0-5, gmodels 2.16.2, gplots 3.0.1, graph 1.54.0, gtools 3.5.0, hgu95av2.db 3.2.3, ipred 0.9-6, limma 3.32.0, mpm 1.0-22, multtest 2.32.0, nlcv 0.2-0, org.Hs.eg.db 3.4.1, pamr 1.55, plotrix 3.6-4, randomForest 4.6-12, survival 2.41-3, varSelRF 0.7-5, xtable 1.8-2
- Loaded via a namespace (and not attached): AnnotationForge 1.18.0, BiocInstaller 1.26.0, DBI 0.6-1, DEoptimR 1.0-8, GSEABase 1.38.0, R6 2.2.0, RBGL 1.52.0, RCurl 1.95-4.8, RSQLite 1.1-2, Rcpp 0.12.10, affyio 1.46.0, assertthat 0.2.0, base64enc 0.1-3, bitops 1.0-6, caTools 1.17.1, class 7.3-14, codetools 0.2-15, compiler 3.4.0, digest 0.6.12, diptest 0.75-7, dplyr 0.5.0, flexmix 2.3-13, fpc 2.1-10, gbm 2.1.3, ggvis 0.4.3, htmltools 0.3.5, htmlwidgets 0.8, httpuv 1.3.3, hwriter 1.3.2, iterators 1.0.8, jsonlite 1.4, kernlab 0.9-25, lattice 0.20-35, lava 1.5, magrittr 1.5, mclust 5.2.3, memoise 1.1.0, mime 0.5, mlbench 2.1-1, modeltools 0.2-21, mvtnorm 1.0-6, nnet 7.3-12, pls 2.6-0, prabclus 2.2-6,

preprocessCore 1.38.0, prodlim 1.6.1, rda 1.0.2-2, robustbase 0.92-7, rpart 4.1-11, sfsmisc 1.1-0, shiny 1.0.2, splines 3.4.0, threejs 0.2.2, tibble 1.3.0, tools 3.4.0, trimcluster 0.1-2, zlibbioc 1.22.0