DE Analysis

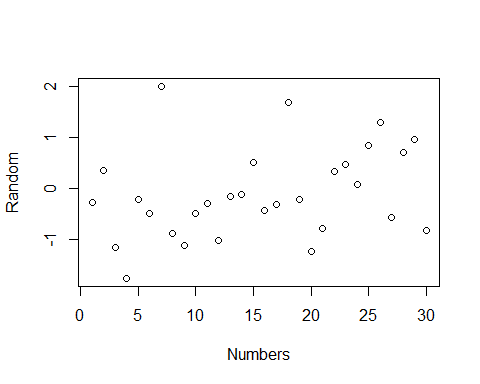
Noel John

2023-12-31

cite<-function(key,bib){  
mybib<-RefManageR::ReadBib("references.bib", check=FALSE)  
bib<-mybib  
paste0("[@",key,"]")  
}

# Abstract

The satisfaction of the German course was the lowest as the gray line shows the highest density (frequency of responses) of very dissatisfied and dissatisfied ratings. The students in our fictitious data set were most satisfied with the Chinese course as the blue line is the lowest for very dissatisfied and “dissatisfied” ratings while the difference between the courses shrinks for “satisfied” and very satisfied. The Japanese language course is in-between the German and the Chinese course.



# Introduction

## Nonspecific filtering

Let us load the dataset from the which we work on. In Chapter 1 you can find a comprehensive description of the acute lymphoblastic leukemia data that we use here.

data("ALL")

First, we construct a list of samples from tumors of B-cells.

bcell = grep("^B", as.character(ALL$BT))

The BCR/ABL translocation – formally, t(9;22)(q34;q11), which is often called the Philadelphia chromosome, producing a fusion gene consisting of the BCR and the ABL1 genes – is relatively prominent in acute lymphocytic leukemias and of therapeutic relevance. Here, we focus on the subset of ALL samples that harbor this translocation and contrast it with the group of samples for which none of the common cytogenetic aberrations (group NEG) was detected.

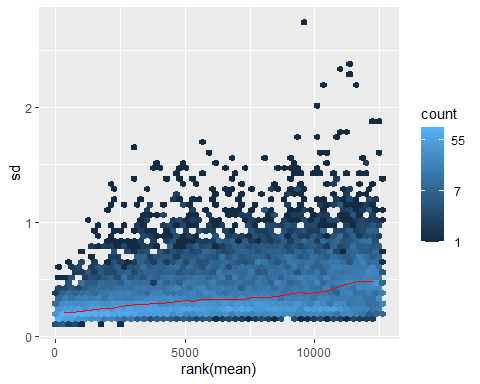
moltyp = which(as.character(ALL$mol.biol) %in% c("NEG", "BCR/ABL"))

Let us now construct a new data object ALL\_bcrneg that contains only those samples that fulfill these two conditions.

ALL\_bcrneg = ALL[, intersect(bcell, moltyp)]  
ALL\_bcrneg$mol.biol = factor(ALL\_bcrneg$mol.biol)

The second line of the above code chunk cleans up the factor variable ALL\_bcrneg$mol.biol by removing the empty levels. Now, if we are going to filter on the basis of variability, we might first want to make sure that the variability is not dominated by its dependence on the mean expression level. If it were, then selecting on the basis of variability would be confounded with selection on the basis of absolute level. There are good reasons, in essence due to the existence of probe-sequence specific background and gain factor effects, not to use the absolute level for gene selection. To check for an association we plot rowwise means versus rowwise standard deviations and plot these together with a smoothed estimate of their regression.

library(vsn)  
meanSdPlot(ALL\_bcrneg)

 From the plot, presuming that we decide that the relationship is not very strong, we proceed. Our next step is to set aside those probe sets with low variability.

In the code below, we set aside the 80% lowest variability probe sets. We choose such a high fraction because we want to limit the length of the subsequent computations. The best choice for this fraction depends on the array design and the biological samples, but in practice it will usually be much lower.

sds = esApply(ALL, 1, sd)  
sel = (sds > quantile(sds, 0.8))  
ALLset1 = ALL\_bcrneg[sel,]

A potential drawback of this approach is found in situations where we are interested in an experimental factor in which one group of samples has few members. In this case, a gene which is differentially expressed between that group and the other(s) may not have a large overall standard deviation. How would you address this situation? At this point you may want to try to look at some heatmaps of the data to see if there are any obvious patterns. Consult the manual page of the function by typing: ?heatmap.

## Differential expression

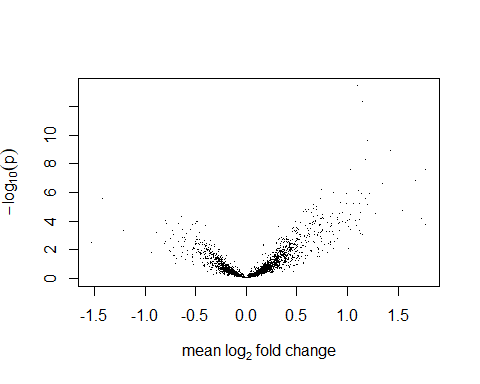
In Bioconductor, the genefilter package allows you to easily select genes using a variety of filters. Additionally, for some tests and comparisons we have developed fast versions. These include rowttests, which perform a t-test for every row in a gene expression matrix; rowFtests, which does F-tests; and rowQ, which calculates a quantile for each row. First, and perhaps easiest is to use a t-test (Dudoit et al., 2002).

library(genefilter)  
tt = rowttests(ALLset1, "mol.biol")  
names(tt)

## [1] "statistic" "dm" "p.value"

Many practitioners have learned that small p-values do not always correspond to genes for which there have been large changes. Let us look at the so-called volcano plot.

plot(tt$dm, -log10(tt$p.value), pch=".",  
 xlab = expression(mean~log[2]~fold~change),  
 ylab = expression(-log[10](p)))



## Multiple testing

One of the subject areas that has received a great deal of attention is that of multiple testing. We provide a brief introduction to the functionality in the multtest package (Pollard et al., 2005). Many of the algorithms in the multtest package depend on random permutations of the samples. The number of permutations is controlled by the parameter B. In the following, we call mt.maxT to perform a permutation test, using the Welch statistic.

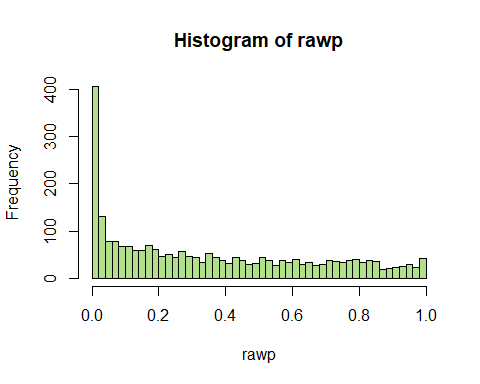
library("multtest")  
cl = as.numeric(ALLset1$mol.biol=="BCR/ABL")  
resT = mt.maxT(exprs(ALLset1), classlabel=cl, B=1000)

## b=10 b=20 b=30 b=40 b=50 b=60 b=70 b=80 b=90 b=100   
## b=110 b=120 b=130 b=140 b=150 b=160 b=170 b=180 b=190 b=200   
## b=210 b=220 b=230 b=240 b=250 b=260 b=270 b=280 b=290 b=300   
## b=310 b=320 b=330 b=340 b=350 b=360 b=370 b=380 b=390 b=400   
## b=410 b=420 b=430 b=440 b=450 b=460 b=470 b=480 b=490 b=500   
## b=510 b=520 b=530 b=540 b=550 b=560 b=570 b=580 b=590 b=600   
## b=610 b=620 b=630 b=640 b=650 b=660 b=670 b=680 b=690 b=700   
## b=710 b=720 b=730 b=740 b=750 b=760 b=770 b=780 b=790 b=800   
## b=810 b=820 b=830 b=840 b=850 b=860 b=870 b=880 b=890 b=900   
## b=910 b=920 b=930 b=940 b=950 b=960 b=970 b=980 b=990 b=1000

ord = order(resT$index) ## the original gene order  
rawp = resT$rawp[ord] ## permutation p-values

The figure below shows the histogram of unadjusted permutation p-values as given by the vector rawp. The high proportion of small p-values suggests that indeed a substantial fraction of the genes is differentially expressed between the two groups.

hist(rawp, breaks=50, col="#B2DF8A")



In order to control the familywise error rate (FWER), that is, the probability of at least one false positive in the set of significant genes, mt.maxT used the permutation-based maxT procedure ofWestfall and Young (1993). We obtain 34 genes with an adjusted p-value below 0.05:

sum(resT$adjp<0.05)

## [1] 34

A comparison of this number to the height of the leftmost bar in the histogram suggests that we are missing a large number of differentially expressed genes. The FWER is a very stringent criterion, and in some microarray studies, few or no genes may be significant in this sense, even if many more are truly differentially expressed. A more sensitive criterion is provided by the false discovery rate (FDR), that is, the expected proportion of false positives among the genes that are called significant.We can use the procedure of Benjamini and Hochberg (1995) as implemented in multtest to control the FDR:

res = mt.rawp2adjp(rawp, proc = "BH")  
sum(res$adjp[,"BH"]<0.05)

## [1] 209

## Moderated test statistics and the limma package

We use the p-values for ranking genes, and do not advocate interpreting them as true probabilities. Nevertheless, the results of a multiple testing adjustment can be informative for choosing selection cut-offs. Note that a more formal treatment would need to take into account the multiple t-tests as well as the implicit testing of the nonspecific filtering. A t-test analysis can also be conducted with functions of the limma package (Smyth, 2004). First, we have to define the design matrix. One possibility is to use an intercept term that represents the mean log2 intensity of a gene across all samples (the first column in the below matrix, consisting of 1s), and to encode the difference between the two classes in the second column.

#library("limma")  
design = cbind(mean = 1, diff = cl)

Next a linear model is fitted for every gene by the function lmFit, and an empirical Bayes moderation of the standard errors can be performed with the function eBayes (Smyth, 2004). This employs information from all genes to arrive at more stable estimates of each individual gene’s variance.

library(limma)  
fit = lmFit(exprs(ALLset1), design)  
fit = eBayes(fit)

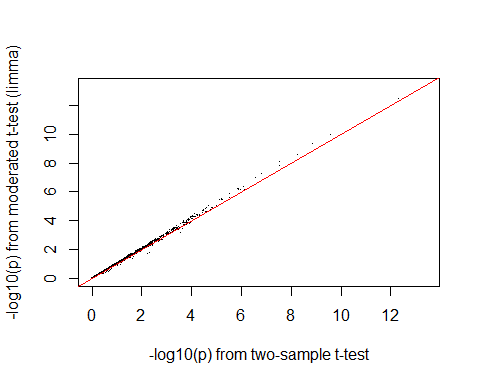
We can list the ten most differentially expressed genes using the function topTable. The three probe sets with the lowest p-value all map to the ABL1 gene which is part of the fusion gene product caused by the t(9;22)(q34;q11) translocation and which is known to be over-expressed and acting as a strong oncogene in acute lymphoblastic leukemia.

library("hgu95av2.db")  
ALLset1Syms = unlist(mget(featureNames(ALLset1),env = hgu95av2SYMBOL))  
topTable(fit, coef = "diff", adjust.method = "fdr",sort.by = "p", genelist = ALLset1Syms)

## ID logFC AveExpr t P.Value adj.P.Val B  
## 1636\_g\_at ABL1 1.100012 9.196420 9.033955 4.879256e-14 1.232012e-10 21.293157  
## 39730\_at ABL1 1.152527 9.000049 8.587774 3.877138e-13 4.894886e-10 19.341283  
## 1635\_at ABL1 1.202675 7.897095 7.338622 1.228926e-10 1.034346e-07 13.905543  
## 1674\_at YES1 1.427212 5.001771 7.050134 4.554138e-10 2.874800e-07 12.667763  
## 40504\_at PON2 1.181029 4.244478 6.664739 2.571219e-09 1.298466e-06 11.032194  
## 40202\_at KLF9 1.779378 8.621443 6.391779 8.622708e-09 3.628723e-06 9.889235  
## 37015\_at ALDH1A1 1.032702 4.330511 6.242224 1.662442e-08 5.996667e-06 9.269402  
## 32434\_at MARCKS 1.678550 4.466311 5.971866 5.375808e-08 1.696739e-05 8.161945  
## 37027\_at AHNAK 1.348702 8.444161 5.805421 1.097066e-07 3.077880e-05 7.489382  
## 37403\_at ANXA1 1.117721 5.086540 5.483354 4.265368e-07 1.077006e-04 6.210589

When you compare the resulting p-value with those from the parametric t-test, you will see that they are almost identical:

plot(-log10(tt$p.value), -log10(fit$p.value[, "diff"]),  
 xlab = "-log10(p) from two-sample t-test",  
 ylab = "-log10(p) from moderated t-test (limma)",  
 pch=".")  
abline(c(0, 1), col = "red")



The result is shown in the above figure. Because of the large number of samples, the empirical Bayes moderation is not so relevant here: in these dataset the gene-specific variance can be well estimated from the data of each gene.

## Small sample sizes

However, the empirical Bayes moderation may be quite useful in cases with fewer replicates. Let us draw a subsample with three arrays from each group from our data:

subs = c(35, 65, 75, 1, 69, 71)  
ALLset2 = ALL\_bcrneg[, subs]  
table(ALLset2$mol.biol)

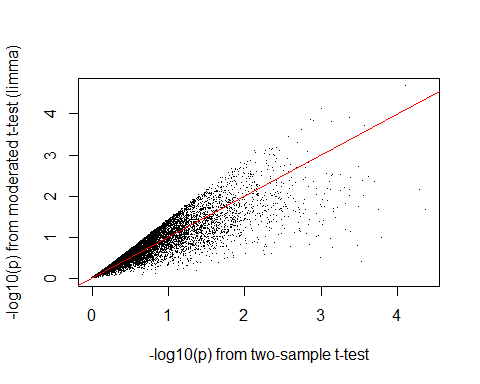
##   
## BCR/ABL NEG   
## 3 3

We repeat the testing procedure in the same way as before.

tt2 = rowttests(ALLset2, "mol.biol")  
fit2 = eBayes(lmFit(exprs(ALLset2), design=design[subs, ]))

Below is a plot of the results above.

plot(-log10(tt2$p.value), -log10(fit2$p.value[, "diff"]),  
 xlab = "-log10(p) from two-sample t-test",  
 ylab = "-log10(p) from moderated t-test (limma)",  
 pch=".")  
abline(c(0, 1), col = "red")

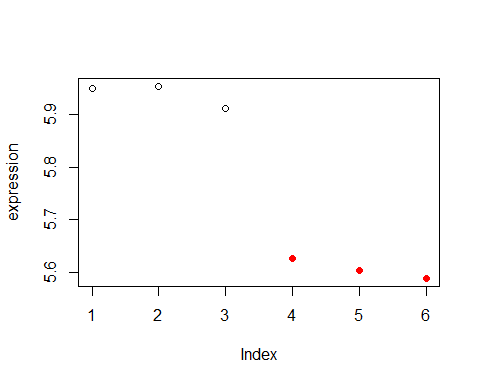


Let us have a look at a gene that has a small p-value in the normal t-test but a large one in the moderated test.

g = which(tt2$p.value < 1e-4 &fit2$p.value[, "diff"] > 0.02)

We plot its expression values and use different symbols and colors to encode the sample classes.

sel = (ALLset2$mol.bio == "BCR/ABL")+1  
col = c("black", "red")[sel]  
pch = c(1,16)[sel]  
plot(exprs(ALLset2)[g,], pch=pch, col=col,ylab="expression")



## Gene Annotation

After having obtained a list of reporters from a microarray experiment and mapping them to their target genes, one will want to use the annotation of the genes and gene products to better interpret the experimental results. Often, it is beneficial to use gene annotation in the course of the primary analysis, in order to narrow down the set of data to be considered and ameliorate multiple testing problems, or in order to explore specific biological hypotheses.

From the ALL dataset, we select those samples that were obtained from tumors harboring either the BCR/ABL or the ALL1/AF4 translocation.

data(ALL)  
types = c("ALL1/AF4", "BCR/ABL")  
bcell = grep("^B", as.character(ALL$BT))  
ALL\_af4bcr = ALL[, intersect(bcell,which(ALL$mol.biol %in% types))]  
ALL\_af4bcr$mol.biol = factor(ALL\_af4bcr$mol.biol)

We want to apply a nonspecific filtering step in order to remove probe sets that are likely to be noninformative.We use the function nsFilter from the genefilter package for that purpose. The default measure used by nsFilter for the variance filtering step is the IQR. This is a reasonable choice as long as the sizes of the sample groups are approximately similar. This is not the case for our BCR/ABL ALL1/AF4 subset, where the ALL1/AF4 positive group is much smaller:

table(ALL\_af4bcr$mol.biol)

##   
## ALL1/AF4 BCR/ABL   
## 10 37

For calculation of the IQR, 50% of the most extreme values are discarded as outliers, thus the measure of variance will be dominated mainly by the much larger BCR/ABL positive group. We could address this problem by using a non-robust measure of variance like the standard deviation, however this would make the filtering more susceptible to outliers. Instead, we will look at the range between more extreme quantiles, here.

qrange <- function(x){  
 diff(quantile(x, c(0.1, 0.9)))  
}  
  
library("genefilter")  
filt\_af4bcr = nsFilter(ALL\_af4bcr, require.entrez=TRUE,require.GOBP=TRUE, var.func=qrange, var.cutoff=0.5)  
ALLfilt\_af4bcr = filt\_af4bcr$eset

Now, let us load the packages with the necessary tools and annotation data

#library("Biobase")  
#library("annotate")  
#library("hgu95av2.db")

Our first step is to use the function rowttests to carry out a two-group comparison and to select the top 100 genes.

rt = rowttests(ALLfilt\_af4bcr, "mol.biol")  
names(rt)

## [1] "statistic" "dm" "p.value"

## Working with GO

The Gene Ontology (GO) is a structured vocabulary of terms describing gene products according to molecular function, biological process, and cellular component (The Gene Ontology Consortium, 2000). The molecular function of a gene product describes what it can do at the biochemical level but without reference to where or when this activity might occur. The biological process of a gene product describes a biological objective to which the gene product contributes. The cellular component ontology describes locations, at the levels of subcellular structures and macromolecular complexes. Examples of cellular components include nuclear inner membrane and the ubiquitin ligase complex.

# Supervised Machine Learning

Machine learning (ML) is typically divided into two separate areas, supervised ML and unsupervised ML. The first of these is referred to as classification in the statistics literature, and the second is referred to as clustering. Both types of machine learning are concerned with the analysis of datasets containing multivariate observations.

One issue that typically arises in ML applications to high-throughput biological data is feature selection. For example, in the case of microarray data one typically has tens of thousands of features that were collected on all samples, but many will correspond to genes that are not expressed. Other features will be important for predicting one phenotype, but largely irrelevant for predicting other phenotypes. Thus, feature selection is an important issue. Fundamental to the task of ML is selecting a measure of similarity among (or distance between) multivariate data points.

# Supervised machine learning check list

-Filter out features (genes) that show little variation across samples, or that are known not to be of interest. If appropriate, transform the data of each feature so that they are all on the same scale. -Select a distance, or similarity, measure. What does it mean for two samples to be close? Make sure that the selected distance embodies your notion of similarity. -Feature selection: Select features to be used for ML. If you are using cross-validation, be sure that feature selection according to your criteria, which may be data-dependent, is performed at each iteration. -Select the algorithm: Which of the many ML algorithms do you want to use? -Assess the performance of your analysis. With supervised ML, performance is often assessed using cross-validation, but this itself can be performed in various ways.

The ALL dataset contains over 100 samples, for a variety of different subtypes of the disease. In the code below we load the data, and then subset to the particular phenotypes in which we are interested. The specific information we need is to select those with B-cell ALL, and then within that subset, those that are NEG and those that are labeled as BCR/ABL. The last line in the code below is used to drop unused levels of the factor encoding mol.biol.

data(ALL)  
bcell = grep("^B", as.character(ALL$BT))  
moltyp = which(as.character(ALL$mol.biol)%in% c("NEG", "BCR/ABL"))  
ALL\_bcrneg = ALL[, intersect(bcell, moltyp)]  
ALL\_bcrneg$mol.biol = factor(ALL\_bcrneg$mol.biol)

## Nonspecific filtering of features

Nonspecific filtering removes those genes that we believe are not sufficiently informative for any phenotype, so that there is little point in considering them further. For the purpose of this teaching exercise, we used a very stringent filter so that the dataset is small and the examples will run quickly; in practice you would probably use a less stringent filter. We use the function nsFilter from the genefilter package to filter for a number of different criteria. For instance, by default it removes the control probes on Affymetrix arrays, which can be identified by their AFFX prefix. We also exclude genes without Entrez Gene identifiers, and as suggested above, we select the top 25% of genes on the basis of variability across samples.

ALLfilt\_bcrneg = nsFilter(ALL\_bcrneg, var.cutoff=0.75)$eset  
head(exprs(ALLfilt\_bcrneg))

## 01005 01010 03002 04007 04008 04010 04016 06002 08001  
## 41654\_at 8.828648 8.094511 8.168164 8.254469 7.361298 9.503099 8.603330 6.773515 9.295929  
## 35430\_at 5.855489 3.641874 5.017166 5.311914 4.778362 4.325899 6.393523 6.633259 5.880185  
## 38924\_s\_at 8.424110 7.729307 8.644447 8.945631 8.216529 7.195125 9.337314 8.823942 8.881610  
## 36023\_at 7.405947 6.301844 7.136245 7.527066 6.148952 6.330935 7.471733 6.944525 7.638783  
## 266\_s\_at 9.045327 7.578499 8.765199 8.972763 8.254930 9.395069 7.792077 8.246472 8.771152  
## 37569\_at 8.079626 7.767153 8.117210 8.220167 7.643456 8.014061 8.302689 7.488749 8.899068  
## 08011 08012 08024 09008 09017 11005 12006 12007 12012  
## 41654\_at 8.386748 9.326383 7.927259 9.138667 8.006808 7.967930 7.727360 8.616821 8.074623  
## 35430\_at 6.044584 6.629995 5.842344 6.420538 5.536865 6.114543 6.342375 5.674307 5.259431  
## 38924\_s\_at 9.097399 9.082794 8.353832 7.224571 7.755654 8.501485 8.247791 8.668081 8.232745  
## 36023\_at 7.111556 7.837012 7.490714 7.173974 7.250574 6.413746 7.416139 6.963664 7.347160  
## 266\_s\_at 9.292926 8.701688 5.520708 9.219472 8.722878 9.231635 9.070559 8.473812 6.288550  
## 37569\_at 7.812161 8.042430 7.738814 8.707214 8.538109 8.298557 7.381348 7.528288 7.249217  
## 12019 12026 14016 15001 15005 16009 20002 22009 22010  
## 41654\_at 9.302439 7.332120 7.914735 7.273544 7.620857 7.749546 8.507203 7.556782 7.777151  
## 35430\_at 4.868694 5.864238 4.924253 5.722033 6.677922 4.958387 4.724708 5.287356 6.180261  
## 38924\_s\_at 8.753591 8.138303 7.697848 8.908445 7.957503 7.531842 8.598408 7.655558 7.476499  
## 36023\_at 7.168034 7.070591 7.124375 6.739134 7.105273 6.183825 6.208769 6.422143 7.142425  
## 266\_s\_at 8.065411 7.279904 5.141673 8.315370 8.487607 6.135236 5.835822 4.775661 6.990659  
## 37569\_at 7.662015 6.878414 7.243849 7.483123 9.217584 6.376393 7.260814 7.311884 8.335432  
## 22011 22013 24001 24008 24010 24011 24017 24018 24022  
## 41654\_at 8.777299 8.731828 8.479426 8.148210 7.932522 8.335359 9.517224 8.563869 8.336168  
## 35430\_at 7.081967 5.613332 6.475653 5.129776 6.026798 5.542433 5.189920 4.876395 6.070339  
## 38924\_s\_at 9.053423 8.255573 8.862210 8.587378 8.680288 8.240342 8.063822 7.510568 7.590677  
## 36023\_at 7.775586 7.171030 8.193865 6.307482 7.185639 7.079452 6.497058 6.418974 7.335754  
## 266\_s\_at 8.056750 6.605943 7.604823 7.926746 8.696195 8.364659 7.754378 6.930111 7.232982  
## 37569\_at 8.680554 7.155823 8.639928 7.404970 7.649956 7.107873 6.634875 7.901837 8.756573  
## 25003 25006 26001 26003 27003 27004 28001 28005 28006  
## 41654\_at 8.234872 8.630322 8.541641 8.466201 7.782210 9.221615 7.841782 9.087692 8.373875  
## 35430\_at 5.917114 4.722375 6.520764 6.235439 4.330426 5.205263 4.657934 5.366176 5.028125  
## 38924\_s\_at 8.038698 7.219289 8.829956 7.891159 7.446301 8.886631 7.458876 8.387229 8.253687  
## 36023\_at 6.873535 6.075753 6.668481 7.276040 7.127008 6.859286 6.405451 7.005145 6.919874  
## 266\_s\_at 7.824575 5.372289 6.974280 8.974530 8.481343 9.476918 8.866697 9.198704 5.619280  
## 37569\_at 7.504818 7.256328 6.816550 8.721719 8.627308 9.023705 8.119095 9.175909 8.566743  
## 28007 28019 28021 28023 28024 28031 28035 28036 28037  
## 41654\_at 9.467728 8.837256 8.997232 9.121741 9.871054 7.857103 9.218828 8.569648 9.106992  
## 35430\_at 4.393043 4.104417 4.957406 5.640488 3.372191 5.145946 3.784037 5.897394 4.516038  
## 38924\_s\_at 8.027459 6.692071 7.709409 8.475243 8.983860 7.633971 7.620777 7.570270 6.994572  
## 36023\_at 7.367557 6.010747 6.650392 7.293290 7.076400 7.025873 6.456335 7.222923 5.987907  
## 266\_s\_at 6.639194 8.815818 9.722175 9.209258 6.900522 8.762969 7.199150 8.512645 8.076551  
## 37569\_at 8.849892 7.703401 9.575725 8.776570 8.719523 8.212823 7.860119 8.890932 8.305911  
## 28042 28043 28044 28047 30001 31011 33005 36002 37013  
## 41654\_at 8.665500 9.123309 8.988580 9.671188 8.990145 9.418143 9.464176 7.885616 8.413325  
## 35430\_at 5.779521 6.340348 5.736402 6.155322 4.573978 6.117581 6.308283 5.608931 5.855131  
## 38924\_s\_at 7.930592 8.002395 7.391385 8.921583 7.596922 8.061912 7.726571 8.142278 8.173580  
## 36023\_at 6.973413 7.747994 7.353532 7.304685 6.560346 7.493523 6.492410 7.677477 7.696518  
## 266\_s\_at 9.109580 9.249810 7.554509 9.582006 8.615381 9.240385 8.476273 9.543679 5.844497  
## 37569\_at 8.187730 8.869171 8.307514 8.877778 8.788335 9.358917 7.951865 8.071890 8.089461  
## 43001 43004 43007 43012 48001 49006 57001 62001 62002  
## 41654\_at 7.732309 8.110521 8.992291 9.090736 8.280921 7.613821 7.351801 8.594811 9.070206  
## 35430\_at 4.829416 4.768723 7.044922 5.276924 4.877153 5.569176 6.557429 6.200918 6.070897  
## 38924\_s\_at 8.236623 7.594109 8.702661 7.780634 7.793355 9.081649 8.197571 8.285420 7.499041  
## 36023\_at 7.572554 7.047133 6.682941 7.048038 7.665009 7.664315 7.531252 7.804634 7.976873  
## 266\_s\_at 7.436804 8.814873 7.309128 9.408554 8.794465 9.024997 9.056086 9.424484 8.344995  
## 37569\_at 7.962245 8.150832 8.154709 8.604458 8.227035 7.376492 8.827597 8.797025 8.700340  
## 62003 64001 64002 65005 68001 68003 84004  
## 41654\_at 8.457478 8.423355 9.109379 6.956484 8.731574 7.849030 8.960964  
## 35430\_at 5.928910 5.934606 6.563591 5.046739 3.900298 6.571250 6.372936  
## 38924\_s\_at 8.015164 8.476753 8.160178 8.994477 7.414112 7.664263 8.940619  
## 36023\_at 6.951412 7.305877 6.367466 7.048391 6.584269 6.584391 8.762137  
## 266\_s\_at 7.894858 7.475200 6.015826 6.737809 5.597120 8.954388 6.668957  
## 37569\_at 8.207598 7.669785 6.823858 8.069104 5.780545 8.220527 8.156519

## Feature selection and standardization

Feature selection is an important component of machine learning. Typically the identification and selection of features used for supervised ML relies on knowledge of the system being studied, and on univariate assessments of predictive capability. Among the more commonly used methods are the selection of features that are predictive using t-statistic and ROC curves (at least for two-sample problems). In order to correctly assess error rates it is essential to accounted for the effects of feature selection. If cross-validation is used then feature selection must be incorporated within the cross-validation process and not performed ahead of time using all of the data. A second important aspect is standardization. For gene expression data the recorded expression level is not directly interpretable, and so users must be careful to ensure that the statistics used are comparable. This standardization ensures that all genes have equal weighting in the ML applications. In most cases this is most easily achieved by standardizing the expression data, within genes, across samples. In some cases (such as with a t-test) there is no real need to standardize because the statistic itself is standardized. In the code segments below we standardize all gene expression values. It is important that nonspecific filtering has already been performed. We first write a helper function to compute the rowwise IQRs for us.

rowIQRs = function(eSet) {  
 numSamp = ncol(eSet)  
 lowQ = rowQ(eSet, floor(0.25 \* numSamp))  
 upQ = rowQ(eSet, ceiling(0.75 \* numSamp))  
 upQ - lowQ  
}

Next we subtract the row medians and divide by the row IQRs. Again, we write a helper function, standardize, that does most of the work.

standardize = function(x) (x - rowMedians(x)) / rowIQRs(x)  
exprs(ALLfilt\_bcrneg) = standardize(exprs(ALLfilt\_bcrneg))

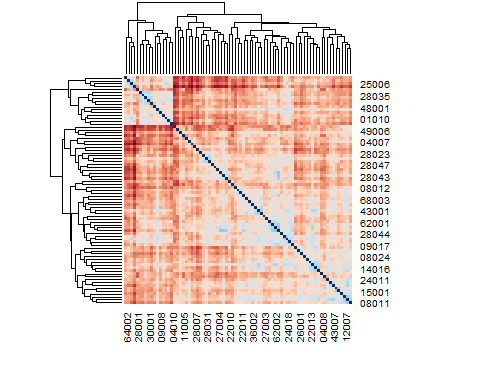
# Selecting a distance

The dist function computes the distance between rows of an input matrix. We want the distances between samples, thus we transpose the matrix using the function t. The return value is an instance of the dist class. Because this class is not supported by some R functions that we want to use, we also convert it to a matrix.

eucD = dist(t(exprs(ALLfilt\_bcrneg)))  
eucM = as.matrix(eucD)

We next visualize the distances using a heatmap. In the code below we generate a range of colors to use in the heatmap. The RColorBrewer package provides a number of different palettes to use and we have selected one that uses red and blue. Because we want red to correspond to high values, and blue to low, we must reverse the palette.

library("RColorBrewer")  
hmcol = colorRampPalette(brewer.pal(10, "RdBu"))(256)  
hmcol = rev(hmcol)  
heatmap(eucM, sym=TRUE, col=hmcol, distfun=as.dist)



## Actual Machine Learning

We start by looking at the k nearest neighbors (KNN) and diagonal linear discriminant analysis (DLDA) methods, because they are conceptually simple and serve well to demonstrate most of the general principles. It is easiest to understand most supervised ML methods in the setting where one has both a training set on which to build the model, and a test set on which to test the model. We begin by artificially dividing our data into a test and training set. Such a dichotomy is not actually that useful and in practice one tends to rely on cross-validation, or other similar schemes.

library(MLInterfaces)  
Negs = which(ALLfilt\_bcrneg$mol.biol == "NEG")  
Bcr = which(ALLfilt\_bcrneg$mol.biol == "BCR/ABL")  
S1 = sample(Negs, 20, replace=FALSE)  
S2 = sample(Bcr, 20, replace = FALSE)  
TrainInd = c(S1, S2)  
TestInd = setdiff(1:79, TrainInd)

In the ALL dataset, some features are not likely to be predictive of the phenotypes of interest, and so we now want to explore what happens if we instead select genes that are able to discriminate between those with BCR/ABL and those samples labeled NEG. We use the t-test to select genes; those with small p-values for comparing BCR/ABL to NEG are used. Although it is tempting to use all the data to do this selection, that is not really a good idea as it tends to give misleadingly low values for the error rates. You can, and probably should, use attenuated t-tests, and you can select the ones to use by the observed p-value. But, these approaches would complicate the exposition further, so we simply select those 50 genes with the most extreme t-statistics. In the code below, we compute the t-tests on the training set, then sort them from largest to smallest, and then obtain the names of the 50 that have the largest observed test statistics.

Traintt = rowttests(ALLfilt\_bcrneg[, TrainInd], "mol.biol")  
ordTT = order(abs(Traintt$statistic), decreasing=TRUE)  
fNtt = featureNames(ALLfilt\_bcrneg)[ordTT[1:50]]

Repeat this exercise on the whole dataset. How many of the genes selected on the training set were also selected when you used the whole dataset? Now we can see how well the different machine learning algorithms work when the features have been selected to help discriminate between the two groups.

## KNN

BNf = ALLfilt\_bcrneg[fNtt,]  
knnf = MLearn( mol.biol ~ ., data=BNf, knnI(k=1,l=0),TrainInd)

## [1] "mol.biol"

c\_mat <- confuMat(knnf)  
c\_mat

## predicted  
## given BCR/ABL NEG  
## BCR/ABL 14 3  
## NEG 1 21

## Random Forest

library("randomForest")  
set.seed(123)  
rf1 = MLearn( mol.biol~., data=ALLfilt\_bcrneg,randomForestI, TrainInd, ntree=1000, mtry=55,importance=TRUE)

## [1] "mol.biol"

trainY = ALLfilt\_bcrneg$mol.biol[TrainInd]  
confuMat(rf1, "train")

## predicted  
## given BCR/ABL NEG  
## BCR/ABL 20 0  
## NEG 0 20

confuMat(rf1, "test")

## predicted  
## given BCR/ABL NEG  
## BCR/ABL 12 5  
## NEG 1 21

Next we use a much smaller value of mtry so that we can compare the results.

rf2 = MLearn( mol.biol~., data=ALLfilt\_bcrneg,randomForestI, TrainInd, ntree=1000, mtry=10,importance=TRUE)

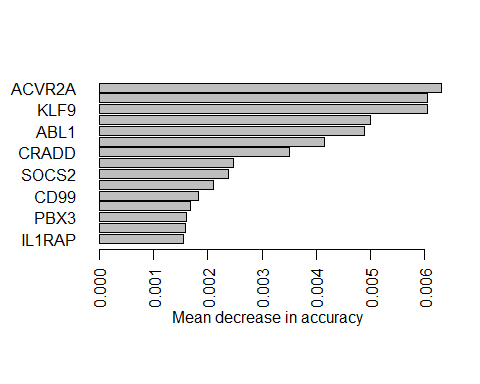
## [1] "mol.biol"

We can use the prediction function to assess the ability of these two forests to predict the class for the test set. For each model we show the confusion matrix for both the training and test sets. Naturally the error rates are much smaller (zero in both cases) for the training set.

## Feature selection

One of the nice things about the randomforest technology is that it provides an indication of which variables were most important in the classification process. The specific definitions of these measures are provided in the manual page for the importance function, which can be used to extract the measures from an instance of the randomForest class. These features can be compared to those selected by t-test or selected by some other means. In the next two code chunks we plot the variable importance statistics for the two random forests. The output is shown in Figure.

opar = par(no.readonly=TRUE, mar=c(7,5,4,2))  
par(las=2)  
impV1 = getVarImp(rf1)  
plot(impV1, n=15, plat="hgu95av2", toktype="SYMBOL")



par(opar)

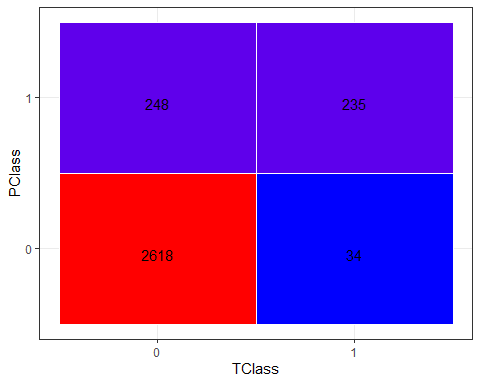
#library(caret)  
#imp<-varImp(rf1)  
#ggplot(imp)

# Confusion Table

## TClass PClass Y  
## 1 1 1 235  
## 2 1 0 34  
## 3 0 1 248  
## 4 0 0 2618

The following is the confusion table:

# Confusion Matrix



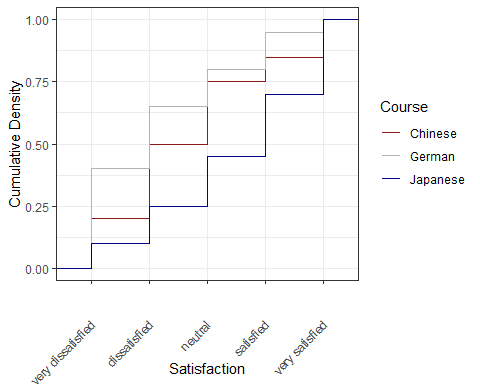
Confusion Matrix Plot

If a diagnosis/test turns out positive ,the likelihood that the tested patient has disease X is PPV Whereas If a diagnosis/test turns out negative ,the likelihood that the tested patient don’t have disease X is NPV with, Risk ratio of risk\_ratio 95% CI of between lci\_risk\_ratio and uci\_risk\_ratio.

# Survey Table

The ldat data set has only two columns: a column labeled *Course* which has three levels (*German*, *Japanese*, and *Chinese*) and a column labeled *Satisfaction* which contains values from 1 to 5 which represent values ranging from *very dissatisfied* to *very satisfied*.

A plot of cummulative satisfaction for all groups in the survey data is shown below.

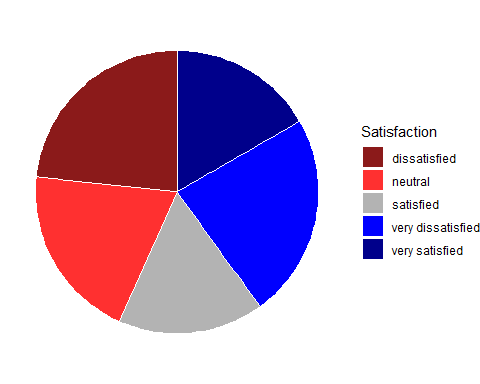


The satisfaction of the German course was the lowest as the gray line shows the highest density (frequency of responses) of *very dissatisfied* and *dissatisfied* ratings. The students in our fictitious data set were most satisfied with the Chinese course as the blue line is the lowest for *very dissatisfied* and “dissatisfied” ratings while the difference between the courses shrinks for “satisfied” and *very satisfied*. The Japanese language course is in-between the German and the Chinese course.

## Pie charts

Most commonly, the data for visualization comes from tables of absolute frequencies associated with a categorical or nominal variable. The default way to visualize such frequency tables are pie charts and bar plots. In a first step, we modify the data to get counts and percentages.

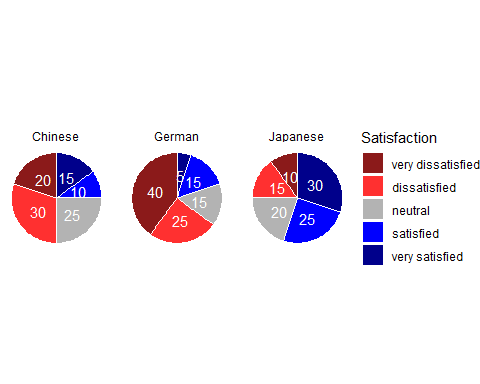
A pie chart of percentage distribution by groups.



If the slices of the pie chart are not labelled, it is difficult to see which slices are smaller or bigger compared to other slices. This problem can easily be avoided when using a bar plot instead. This issue can be avoided by adding labels to pie charts. The labeling of pie charts is, however, somewhat tedious as the positioning is tricky. Below is an example for adding labels without specification.

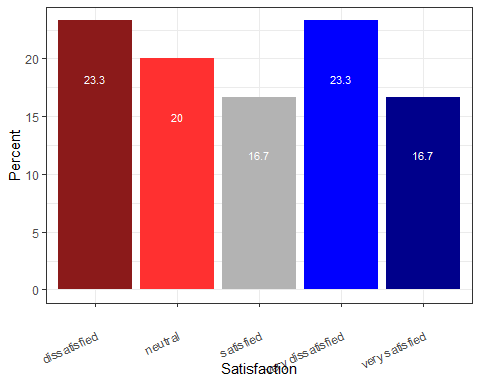
## Pie plots Separated into Groups

Now that we have created the data, we can plot separate pie charts for each course.



## Bar plots

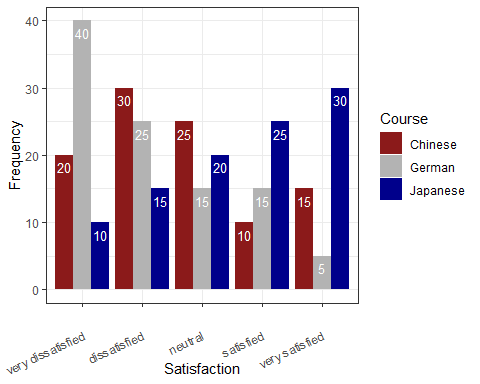
Like pie charts, bar plot display frequency information across categorical variable levels.



Compared with the pie chart, it is much easier to grasp the relative size and order of the percentage values which shows that pie charts are unfit to show relationships between elements in a graph and, as a general rule of thumb, should be avoided.

## Grouped bar plots

Bar plots can be grouped which adds another layer of information that is particularly useful when dealing with frequency counts across multiple categorical variables. But before we can create grouped bar plots, we need to create an appropriate data set.

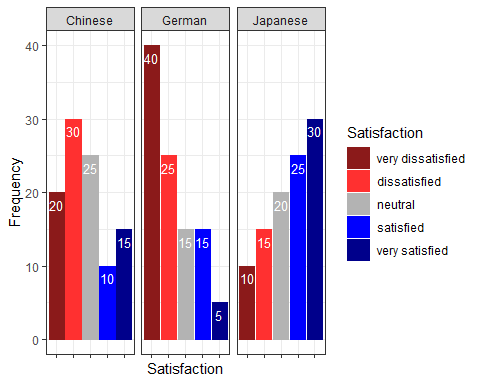


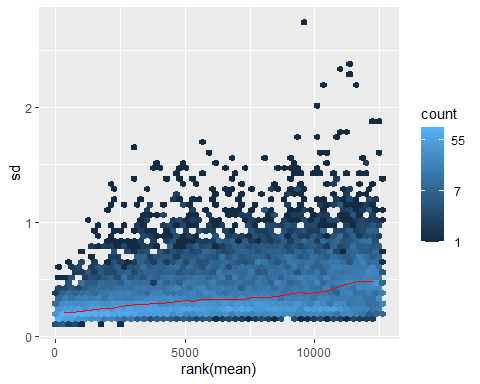
Bar plots are particularly useful when visualizing data obtained through Likert items. As this is a very common issue that empirical researchers face. There are two basic ways to display Likert items using bar plots: grouped bar plots and more elaborate scaled bar plots.

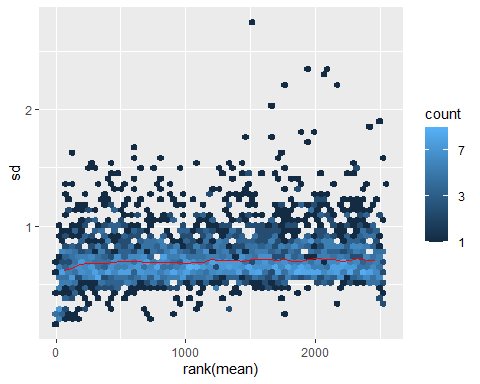
Although we have seen above how to create grouped bar plots, we will repeat it here with the language course example used above when we used cumulative density line graphs to visualize how to display Likert data.

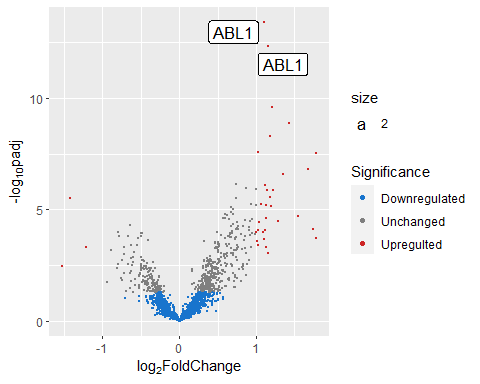
In a first step, we recreate the data set which we have used above. The data set consists of a Likert-scaled variable (Satisfaction) which represents rating of students from three courses about how satisfied they were with their language-learning course. The response to the Likert item is numeric so that “strongly disagree/very dissatisfied” would get the lowest and “strongly agree/very satisfied” the highest numeric value.

Again, we can also plot separate bar graphs for each class by specifying “facets”.









# References