Analysis of GC-MS data of TCC patients

Michael Schubert

19th December 2010

1 Analysis

This is a short summary of the results attained in the analysis performed on GC-MS data of urinary samples of TCC (transitional cell carcinoma) patients plus controls. Figure 1 shows a comparison of Total Ion Chromatograms of a healthy control, an unhealthy control, and a TCC patient sample. Only by eye, one can distinguish between the healthy and unhealthy controls, but it is difficult to find peaks that differ between the unhealthy control and the TCC patients. This file, as well as two other comparisons are supplied as images.

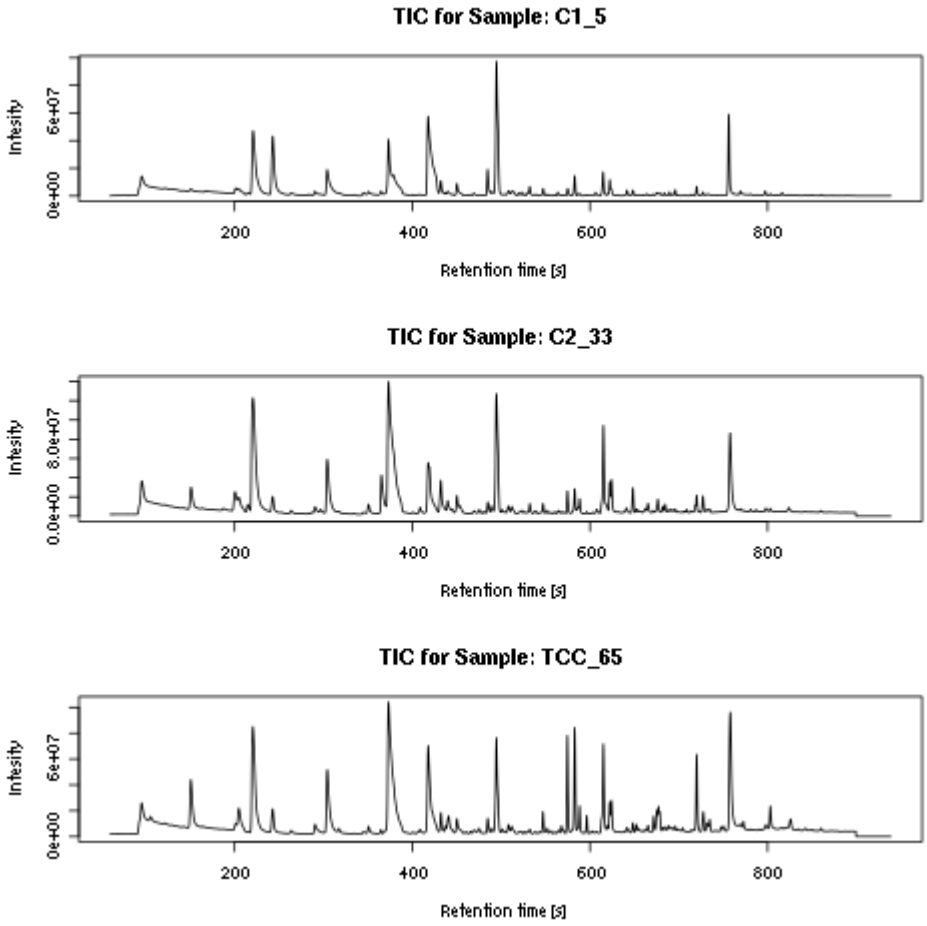

Figure 1: Comparison of different Total Ion Chromatograms: Healthy Control, Unhealthy Control, and TCC patient

The next step was to perform a Principle Component Analysis of the unscaled sample data. The results are shown in figure 2. Most of the samples form a relatively tight cluster, where a tendency of separation between healthy controls and TCC patients including unhealthy controls can be seen, however not really significant. Two samples lie outside the main cluster, namely C2_29 and TCC_54.

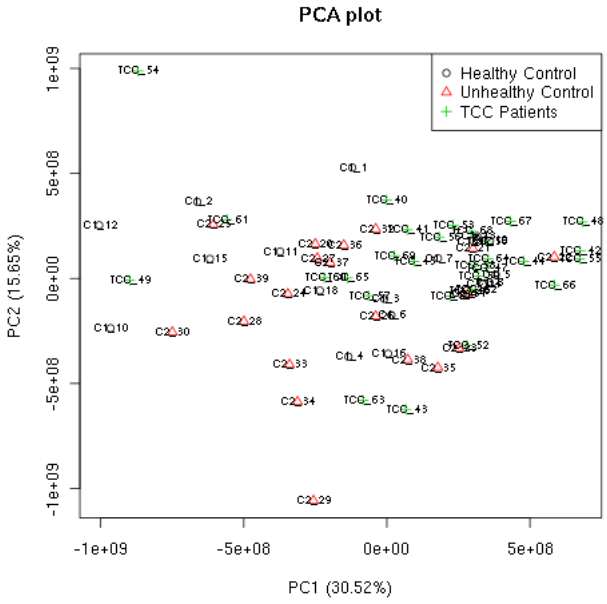

Figure 2: Principle Component Analysis

Also, a 3D score plot was created as seen in figure 3. The results are similar to PCA, with a relatively tight cluster of all samples except two outliers. This time they are comprised of C1_12 and again TCC_54. Combining the result of PCA with the score plot, one can suggest TCC_54 as almost certainly being a true outlier, while the two other samples may or may not be. However, no samples were removed for further analysis.

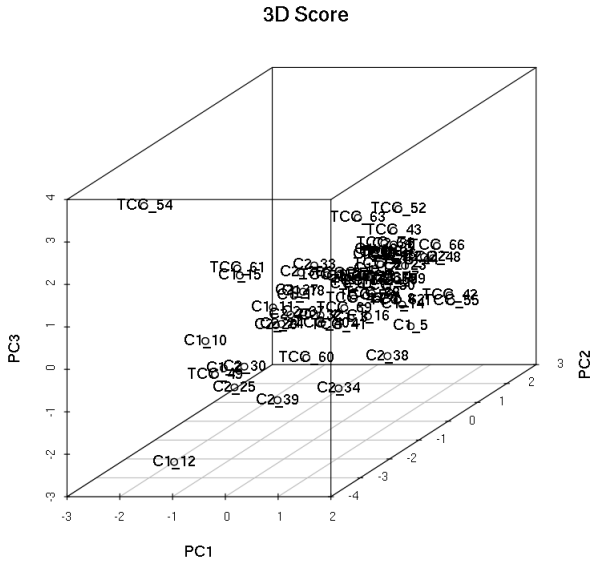

Figure 3: 3D score plot for outlier identification

Finally, leave-one-out cross validation was performed in three runs: healthy control (C1) vs TCC, unhealhty control (C2) vs TCC, and both controls vs TCC. In each run, 20 latent variables, as well as 5 scaling methods (raw data, mean-centered, auto-scaled, range-scaled between 0 and 1 and -1 and 1, and normalisation) were performed and tested for their accuracy. Looking at the results listed in table 1, one can draw several conclusions:

- C1 vs TCC is the most successful classification, which is not surprising looking at the TICs in figure 1
- there is no best scaling method, it is strongly dependent on the data
- internal cross validation seems to produce better models than leave-one-out

Table 1: Most successful approaches including LV, scaling method, specificity, and sensitivity

Protocol	LV	Scaling	% Overall	% Specificity	% Sensitivity
C1 vs TCC	5	Normalise	77.55	100	100
C2 vs TCC	3	Auto-Scale	74.10	94.12	100
C1 and C2 vs TCC	18	Range[-1,1]	77.10	100	100

There are various methods which might increase prediction accuracy. Eliminating outliers before performing the rest of the analysis would be an obvious suggestion. Another possibility would be to use another classification algorithm, e.g. SVMs. Also, a proper alignment of the chromatogram peaks could help.

What is a little unexpected is that the leave-one-out cross validation differs quite significantly from the built-in cross validation of PLS-DA. Obviously, the building of these two classification models must be different to some extend. I would assume that the internal function is somewhat more sophisticated than the realtively crude approach of just leaving one sample out every time and then taking the mean of prediction accuracies. Thus, I would not assign all too much significance to the overall score. However, this could be tested by looking at the variation within a run/scaling method where different samples have been left out. Looking at these, one can find that the values have a distance of 5-10% up and down the mean. This confirms the assumption that the overall score has got some significance, but specificity and sensitivity scores seem to be a better fit to assert the real prediction strength.

To transform these results in order to identify a specific biomarker, one would need to assert the prediction strengths of individual features (or chromatogram peaks). One or a couple that score high in that regard could then be identified as compounds in the biological sample, which could in turn act as biomarkers of the TCC.

2 R script (also supplied as separate file)

```
#!/usr/bin/Rscript
# ============================================================================
# >> R script for analysing the Transitional Cell Carcinoma (TCC) GC-MS data <<
# >> Proteomics Module Assignment, Cranfield University
# >> Michael Schubert, 12/19/2010
```

```
# ================================================================================
# >> 0.) Clear workspace and any open figures, load required libraries <<
#
rm(list=ls())
graphics.off()
require(R.matlab)
require(matlab)
require(plsgenomics)
require(scatterplot3d)
source("biplot3d2.r")
source("DoLOOCV2.r")
# ================================================================================
# >> 1.1) Clear workspace and any open figures <<
#
DATA = readMat("GCMS_P4D.mat")
XTIC = DATA$Xtic # data
RT = DATA$SCANS # retention time
SAM = as.character(DATA$SAM) # labels
CLASSES = as.numeric(factor(substr(SAM, 0,2)))
TYPES = c("Healthy Control", "Unhealthy Control", "TCC Patients")
POSITIVE_CLASS = 3
# ================================================================================
# >> 1.2) Prompt the user for comparative visualisation plots, 3 chromatograms each <<
#
visualise = function() {
while (TRUE) {
cmp = NULL
# read in 3 chromatogram indices to plot and compare
for (i in c(1:3)) {
n = readline(paste("Chromatogram", i, "to compare (0 to exit): "))
if (!(n>0 && n<length(SAM))) {
return("Visualisation aborted")
}
cmp = c(cmp, as.numeric(n))
}
# read to filename to save plot to
fname = readline("Filename to save image to: ")
# save the plot
tiff(filename=fname)
```

```
par(mfcol=c(3,1))
for (i in cmp) {
title = paste("TIC for Sample:", SAM[i])
plot(RT, XTIC[i,], type="l", main=title, ylab="Intesity", xlab="Retention time [s]")
}
dev.off()
}
}
print("TIC comparison, possible indices:")
print(SAM)
visualise()
# ================================================================================
# >> 2a.) Do PCA analysis and do plot with PC weights <<
#
PCA = prcomp(XTIC)
# get the PC weights and construct labels from them
eigen = PCA$sdev^2
pc1lab = paste("PC1 (", round(eigen[1] / sum(eigen) * 100, 2), "%)", sep="")
pc2lab = paste("PC2 (", round(eigen[2] / sum(eigen) * 100, 2), "%)", sep="")
# save figure
tiff(filename="PCA.tif")
plot(PCA$x, col=CLASSES, pch=CLASSES, xlab=pc1lab, ylab=pc2lab, main="PCA plot")
legend(x="topright", legend=TYPES, col=c(1:length(TYPES)), pch=c(1:length(TYPES)))
text(PCA$x, SAM, cex=0.5)
dev.off()
# ================================================================================
# >> 2b.) Draw a 3D score plot to identify any outliers <<
#
biplot3d2(XTIC, SAM, RT, 1, 2, 3, -1, 0)
# ================================================================================
# >> 3.) Construction of Classification Vectors <<
#
# The following vectors should be constructed: C1 vs TCC, C2 vs TCC and C(1+2)
# vs TCC. The way this was proposed in the assignment sheet is highly inefficient
# and leads to code that is hard to maintain. Instead, two filters are created,
# where each row serves as a view (i.e., lists the TIC indices that should be
# used for analysis) for a certain test case. This in combination with the first
# 3 lines of the nested for loops is fully equivalent and more efficient to what
# was asked
```

```
# the first filter has TRUE on all positive samples and FALSE otherwise
PositiveClassFilter = (CLASSES==POSITIVE_CLASS)
# the training filter has 3 rows, each representing a test case; TRUE if the sample
# belongs to the test case, FALSE otherwise
XtrainFilter = rbind(CLASSES!=2, CLASSES!=1, TRUE)
# ================================================================================
# >> 4+5.) Leave one out Cross Validation + Specificity/Sensitivity <<
#
# define different runs and scaling methods, and initialise an empty result table
RUN = c("C1 vs TCC", "C2 vs TCC", "C1+2 vs TCC")
SCALING = c("None", "Mean-Centre", "Auto-Scale", "Range[0,1]", "Range[-1,1]", "Normalise")
RESULT = NULL
# do cross-validation for each run and each scaling method
for (run in 1:length(RUN)) {
for (sc in 1:length(SCALING)) {
# use the filters created in point 3 to extract the right samples for each
# run, as well as their labels
Xtrain = XTIC[XtrainFilter[run,],]
Ctrain = PositiveClassFilter[XtrainFilter[run,]] + 1
SAMtrain = SAM[XtrainFilter[run,]]
# do leave-one-out cross validation using the script provided by Michael
# Cauchy, and take the mean of the overall classification score of the run
cur = DoLOOCV2(Xtrain, Ctrain, sc-1, 1, 20)
overall = round(mean(cur$Overall), 2)
lv = cur$OptLVs[1]
# for the optimal LV, perform PLS-DA again using the same scaling method
# and try to predict class labels for the whole set with built-in cross
# valiation (this is not leave-one-out and might produce slight differences)
Xscaled = pretreat1(Xtrain, sc-1)
plsda = pls.lda(Xscaled$SCALED, Ctrain, Xtest=NULL, lv, nruncv=20)
pred = plsda$predclass
# calculate true/false positives/negatives from Ctrain (=correct labels) and
# pred (=assigned labels)
TP = sum(pred[which(Ctrain==2)]==2) / sum(pred==2)
TN = sum(pred[which(Ctrain==1)]==1) / sum(pred==1)
FP = sum(pred[which(Ctrain==1)]==2) / sum(pred==2)
FN = sum(pred[which(Ctrain==2)]==1) / sum(pred==1)
# calculate specificity and sensitivity from TFPN
spec = round(100 * TN/(TN+FP), 2)
```

```
sens = round(100 * TP/(TP+FN), 2)
# add the result of the current run to our result matrix
RESULT = rbind(RESULT, c(RUN[run], lv, SCALING[sc], overall, spec, sens))
}
}
# use descriptive column names for the result table and print a summary of each
# run to the user; the top runs are easy to find in this table
colnames(RESULT) = c("Protocol", "LV", "Scaling", "% Overall", "% Specificity", "% Sensitivity")
print(as.table(RESULT))
```