iSLS10 Data Analysis Workshop, Part II

Hyungwon Choi

Department of Medicine, NUSmed

March 4, 2022

Contents

1 Data management and inspection

In this section, we introduce basic operations in R language to inspect, clean up, and work with data frames (spreadsheet). We will get familiar with the command line syntax of R, and this process will get us ready for subsequent data analysis steps.

We will first read the lipidomics data (rigorously QC'ed!) and the sample meta data into the workspace (consider it as R's memory).

```
> getwd() ## verify your current working directory
```

```
[1] "/Users/hwchoi912/My Drive (hwchoi912@cssblab.org)/ISLS_Short_Course"

> dir() ## See what files are in the directory

 [1] "Edges_data.txt"
 [2] "figure"
 [3] "GLASSO.txt"
 [4] "Icon\r"
 [5] "iSLS10_Code_Chunk.R"
 [6] "iSLS10_part2.pdf"
 [7] "iSLS10_part2.Rnw"
 [8] "iSLS10_part2.tex"
 [9] "iSLS10_part2.toc"
[10] "iSLS9"
[11] "lipid_data.txt"
[12] "Lipidomic_data"
[13] "MEC_T2Dincidence_lipidomics.pdf"
[14] "Nodes_data.txt"
[15] "Practicum"
[16] "Rplots.pdf"
[17] "sample_data.txt"
[18] "Sample_meta_data"
[19] "Summary_HC.docx"
[20] "SVAS_multivariate_15Feb2022.R"
[21] "Test_network.cys"
[22] "W5_Regularized_regression_Part_b.R"
[23] "W5_regularized_regression.pdf"
[24] "W6_lipidomics_network_and_grpLasso.R"
[25] "W6_sparse_precision_matrix_and_correlation_networks.pdf"
[26] "W7_tree_based_methods_lipidomics.R"
[27] "W7_TreeBased_Methods.pdf"
[28] "W9_unsupervised_analysis.pdf"

> ### Confirms that the two main files are in the directory
> ldata = read.delim("lipid_data.txt", header=T, as.is=T, check.names=F)
> sdata = read.delim("sample_data.txt", header=T, as.is=T, check.names=F)
> colnames(ldata) = gsub("\\(-H2O\\)", "", colnames(ldata))
> ### Remove the neutral loss tag in the lipid names
> ### For special characters, we need to tell R that they are special by prefixing double backslashes
> ### Similar commands can be used to manipulate text headers in an automated manner
>
> ldata[1:5,1:5]
```

```
  ID Cer d16:1/C16:0 Cer d16:1/C18:0 Cer d16:1/C20:0 Cer d16:1/C22:0
1 S1    0.0018321806     0.002185681     0.006609402      0.02718963
2 S2    0.0013672951     0.001605267     0.003260313      0.02328471
3 S3    0.0013615316     0.002492004     0.006519285      0.03365524
4 S4    0.0008583807     0.003512810     0.005867816      0.01619162
5 S5    0.0017791874     0.002861309     0.007761635      0.02653510
```

```
> sdata[1:5,]
```

```
  ID DM Days Age Gender   BMI HbA1c SBP  HDL  LDL  CRP   TG Insulin Glucose
1 S1  0   NA  52      2 20.45    NA 105 1.76 4.16 0.29 1.12    4.65    4.43
2 S2  0   NA  54      1 22.93 5.558 137 1.64 2.74 0.72 0.90    5.87    4.30
3 S3  0   NA  52      1 24.47 5.267 111 1.44 2.91 0.26 2.68    6.78    4.24
4 S4  0   NA  53      1 24.21 6.062 143 1.71 2.36 1.30 1.20    3.96    5.22
5 S5  1 2190  65      2 29.23 4.987 150 1.63 2.95 3.85 1.65   12.35    5.72
```

As the output lines of the last two commands show, most data are organized in spreadsheets. In mathematics and statistics, we call spreadsheet a "data matrix" or "data frame", which can be considered as stacks of column vectors (vertical) and row vectors (horizontal). Inside your computer's physical memory (memory heap), they are recorded as stacks of column vectors. In this case, we have saved both the sample meta data and the lipidomics data in a way that each column vector represents a variable (e.g. a lipid, or a clinical parameter), whereas each row is one study participant (sample).

Now that we have the data read into the workspace, we check whether the sample meta data and the lipidomics data are properly aligned.

```
> ### First match the sample IDs between the two data sets
> all(ldata$ID %in% sdata$ID)
```

```
[1] TRUE
```

```
> all(sdata$ID %in% ldata$ID)
```

```
[1] TRUE
```

```
> all(ldata$ID == sdata$ID) ## All sample IDs are aligned between the two data sets
[1] TRUE
```

Now we are going to explore the data by quick visualization. In this section, we will "attach" the data frame first, so we can call the variable names easily. After we are done with plotting the data, we will detach the data frame to avoid confusion in the rest of the tutorial.

```
> ################################
> ### Plotting sample meta data
> ### while learning R syntax
> ################################
>
> #install.packages("scales")
> library(scales)
> attach(sdata)
> nsamples = nrow(sdata)
> hist(BMI, breaks=50)
> hist(BMI[DM == 1], breaks=50, add=TRUE, col=2)
> plot(SBP ~ Age, pch=19)
> plot(SBP ~ Age, cex=.3, pch=19, col=alpha(2, 0.4))  ### red, opaque dots
> par(mfrow=c(2,2))
> plot(SBP ~ Age, cex=.3, pch=19, col=alpha(2, 0.4))
> plot(HbA1c ~ Age, cex=.3, pch=19, col=alpha(2, 0.4))
> plot(TG ~ Age, cex=.3, pch=19, col=alpha(2, 0.4))
> plot(LDL ~ Age, cex=.3, pch=19, col=alpha(2, 0.4))
> dev.off()
null device
          1
> ## Boxplots
> par(mfrow=c(1,2))
> boxplot(BMI ~ Gender, boxwex=0.3)
> boxplot(HbA1c ~ DM, boxwex=0.3)
> dev.off()
null device
          1
```

So far the plotting exercise involved the continuous variables only. What about the categorical variables?

```
> barplot(table(DM), width=0.2, col=c("red","blue"), horiz=TRUE, names.arg = c("Control","Case"))
```

Alternative, we can also test if there is any association between two categorical variables.

```
> ## Chi-squared test for categorical data
> ## and see if there is any correlation / association
> tab = table(DM, Gender)
> print(tab)

   Gender
DM     1    2
  0  977 1152
  1   78   92

> chisq.test(tab)  ### not significant

        Pearson's Chi-squared test with Yates' continuity correction


data:  tab
X-squared = 1.9136e-29, df = 1, p-value = 1

> ## Discretizing a continuously scaled variable
> bmi.new = rep(NA, nsamples)
> bmi.new[BMI <= 18.5] = 1
> bmi.new[BMI > 18.5 & BMI <= 23] = 2
> bmi.new[BMI > 23 & BMI <= 27.5] = 3
> bmi.new[BMI > 27.5] = 4
> tab = table(DM, bmi.new)
> print(tab)

   bmi.new
DM     1    2    3    4
  0  229 1054  678  168
  1    3   50   76   41

> chisq.test(tab)
```

```
        Pearson's Chi-squared test

data:  tab
X-squared = 79.386, df = 3, p-value < 2.2e-16

> ### significant
> detach(sdata)
```

2 Dimension reduction and sample projection analysis

In this section, we will start working with the lipidomics data. Before going into the unsupervised projection analysis, there are a few “logistical” things to do:

```
> rownames(ldata) = ldata$ID ### First column of the data matrix contains identifiers
> ldata = ldata[,-1]
> ### Since the downstream data analysis will be performed on numerical data only,
> ### we put them into ``rownames'' and exclude them from the analysis.
>
> ### Check zero or negative concentrations before log transformation
> any(ldata <= 0)

[1] TRUE

> sum(ldata <= 0)

[1] 1

> ### Turns out there is a zero or negative value in the data frame somewhere.
>
> ### In case there are zeros, we plug-in a small value for each analyte.
> ### The downstream analysis should not be sensitive to the choice of this value
> ### as long as we choose it close to the limite of detection, e.g.
> ### the smallest positive value across the sample, times 0.9, say.
> ### To accomplish this, we run through a "for" loop.
> for(k in 1:ncol(ldata)) {
+   ### temporary storage
+   xvec = ldata[,k]
+
+   ### take the subset of positive values
+   xpos = xvec[xvec > 0]
+
+   ### get the 90% of the minimum intensity
+   ximpute = 0.9 * min(xpos)
+
+   zid = (xvec <= 0)  ### TRUE/FALSE vector in the original vector
+   if(any(zid)) {  ### Perform this only if there is a zero
+     print(colnames(ldata)[k])
+     xvec[zid] = ximpute  ### Plug in values wherever there is zero
+     ldata[,k] = xvec  ### Store back to the original data frame
+   }
+ }

[1] "SM d18:1/C23:1"

> tmp = log2(ldata)
```

The frequency distributions of ion count or sequence read count data are often skewed, i.e. their distribution is not bell-shaped. Many statistical analysis methods implicitly assume that data are normally distributed. To meet this requirement, we prefer to transform data in logarithmic scale.

Now we try principal component analysis (PCA):

```
> tmp.pca = prcomp(tmp)
> vv = tmp.pca$sdev^2
> vv = vv / sum(vv) * 100
> vv = round(vv, 2)
> print(vv)

  [1] 22.62 14.23  6.14  5.21  3.91  3.70  3.10  2.49  2.25  1.99  1.87  1.79
 [13]  1.55  1.32  1.17  1.16  1.03  0.97  0.89  0.83  0.78  0.72  0.67  0.61
 [25]  0.59  0.58  0.54  0.53  0.52  0.48  0.46  0.42  0.41  0.39  0.38  0.37
 [37]  0.34  0.34  0.33  0.32  0.31  0.30  0.29  0.28  0.27  0.27  0.26  0.25
 [49]  0.24  0.24  0.23  0.23  0.23  0.21  0.21  0.20  0.20  0.20  0.20  0.19
 [61]  0.19  0.18  0.18  0.17  0.17  0.17  0.16  0.16  0.16  0.16  0.15  0.15
 [73]  0.15  0.15  0.14  0.14  0.14  0.14  0.13  0.13  0.13  0.13  0.12  0.12
 [85]  0.12  0.12  0.11  0.11  0.11  0.11  0.11  0.11  0.11  0.10  0.10  0.10
 [97]  0.10  0.10  0.09  0.09  0.09  0.09  0.09  0.09  0.09  0.08  0.08  0.08
```

```
[109] 0.08 0.08 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.06 0.06 0.06
[121] 0.06 0.06 0.05 0.05 0.05 0.04 0.04 0.03 0.00 0.00 0.00 0.00
[133] 0.00
```

This code chunk performs PCA on the input data and reports the percentages of variance explained by individual principal components (PCs). PCs are linear combinations, or weighted averages of lipids – determined in a particular way (orthogonal and associated with largest eigenvalues). When the first few PCs explain a bulk of the total variation in the data, it is a good sign – there should be a subset of variables that explain most of the sample-to-sample differences. On the other hand, if the first PCs only account for a small proportion of the variation, it's a grim sign. We may be dealing with highly noisy data, or lipidomic data without any interesting features. Let us do "projection analysis" of the samples onto the first two PCs.

```
> ccc = rep("gray", nsamples)
> ccc[sdata$DM == 1] = "red"
> Thres = max(abs(tmp.pca$x[,1])) / 2  ### automatically calculate the axis range
> XLAB = paste("PC1 (", vv[1], "%)", sep="")  ## PC1
> YLAB = paste("PC2 (", vv[2], "%)", sep="")  ## PC2
> library(scales)  ### To allow opaqueness in dots
> #pdf("PCAplot.pdf", height=5.5, width=5, useDingbats = FALSE)
> plot(tmp.pca$x[,1], tmp.pca$x[,2],   ### X=PC1, Y=PC2 coordinate
+      col=alpha(ccc,0.2), pch=19,
+      xlim=c(-Thres,Thres), ylim=c(-Thres,Thres),   ### Range set equally for both axes
+      xlab=XLAB, ylab=YLAB,   ### Automatically assembled above
+      main="MEC cohort (N=2,299)", cex=0.8)
> legend("bottomright", c("No event","Incident DM"), pch=19, col=c("gray","red"), cex=0.8)
> abline(v=0, lty=2)
> abline(h=0, lty=2)
> #dev.off()
```

This plot shows that the incident DM cases are enriched in the bottom of the projection plot, i.e. along the PC2 axis. It implies that the lipids with large contributions to the makeup of PC2 may have predictive values for DM incidence. We can visualize the pattern with a heatmap:

```
> ############### Draw the entire data (on a relative scale)
> tmp.ctr = sweep(tmp, 2, apply(tmp, 2, median))
> ### We are normalizing each lipid by its own median value
> ### so that all lipids are on a comparable scale.
>
> #install.packages("gplots")
> library(gplots)
> #pdf("heatmap.pdf", height=20, width=25, useDingbats = FALSE)
> heatmap.2(as.matrix(t(tmp.ctr)), trace="n", col=bluered(20), breaks=seq(-2,2,by=0.2),
+           distfun=function(x) as.dist(1-cor(t(x))),
+           hclustfun=function(x) hclust(x, method="average"),
+           ColSideColors = ccc,
+           cexRow=0.2, cexCol=0.2,
+           mar=c(10,10))
> #dev.off()
```

The heatmap confirms two things: (i) incident DM cases are enriched on the right hand side of the heatmap; (ii) ceramides are higher and hexosyl ceramides are lower at baseline in high-risk individuals.

In metabolomics/lipidomics studies, many papers also show a projection plot called "PLS-DA", a.k.a. partial least squares - discriminant analysis, or "OPLS-DA", a modification for mathematically correct treatment of binary outcomes (Yes, PLS-DA with binary outcomes involves a bit of cheating; but it works still). Unlike PCA, PLS-DA is a supervised analysis in the sense that the axes on which the samples are shown are found in a way that shows the largest contrast between the sample groups. In PCA, we did not use the information of which participants are DM cases and which are not. By contrast, PLS-DA explicitly uses this information. Here is the code snippet:

```
> ############### PLS-DA analysis (for fun)
> ############### Explain why PLS-DA is a "supervised" analysis
> #if (!requireNamespace("BiocManager", quietly = TRUE))
> #  install.packages("BiocManager")
> #
> #BiocManager::install("mixOmics")
>
> library(mixOmics)
> sample.class = ifelse(sdata$DM == 1, "Case", "Control")
> X = as.matrix(tmp)
> Y = sample.class
> tmp.out = plsda(X=X, Y=Y, ncomp=2)
> #pdf("PLSDA_projection.pdf")
> plotIndiv(tmp.out, ind.names = TRUE, ellipse = TRUE, legend = TRUE)
> #dev.off()
```

3 Understanding the association between lipids and other risk factors at baseline

So far we analyzed the lipidomics data on its own, at most in association with the clinical endpoint (DM incidence). That is, we did not probe their correlation with other clinical parameters in the causal pathway. In the sample meta data,

```
> head(sdata)

  ID DM Days Age Gender   BMI HbA1c SBP  HDL  LDL  CRP   TG Insulin Glucose
1 S1  0   NA  52      2 20.45    NA 105 1.76 4.16 0.29 1.12    4.65    4.43
2 S2  0   NA  54      1 22.93 5.558 137 1.64 2.74 0.72 0.90    5.87    4.30
3 S3  0   NA  52      1 24.47 5.267 111 1.44 2.91 0.26 2.68    6.78    4.24
4 S4  0   NA  53      1 24.21 6.062 143 1.71 2.36 1.30 1.20    3.96    5.22
5 S5  1 2190  65      2 29.23 4.987 150 1.63 2.95 3.85 1.65   12.35    5.72
6 S6  0   NA  68      2 22.42 5.609 177 1.26 2.66 0.19 0.95    3.79    4.92
```

Let us look at the correlation structure between the biochem / clinical data and the lipid levels. For continuous scaled measurements, it is very easy to do this:

```
> ### We first re-arrange the columns of the sample data matrix
> ### so that all continuous variables are shifted to the right
> sdata = sdata[,c(1:3,5,4,6:14)]
> cp = colnames(sdata)[5:14]  ### Storing variable names
> ### Now we compute correlations between clinical data and lipids in an one liner
> ### Recall that ``tmp'' object holds lipidomic data in logarithmic scale (base 2)
> cormat = cor(sdata[,5:14], tmp, use="pairwise.complete.obs")
> ### Plot it in a heatmap
> #pdf("cor_heatmap.pdf", height=20, width=6, useDingbats = FALSE)
> heatmap.2(as.matrix(t(cormat)), trace="n", main="At baseline",
+           col=bluered(20), breaks=seq(-1,1,by=0.1),
+           #distfun=function(x) as.dist(1-cor(t(x))),
+           hclustfun=function(x) hclust(x, method="average"),
+           cexRow=0.2, cexCol=1,
+           mar=c(10,10))
> #dev.off()
```

From the heatmap, we derive the following conclusions. First, LDL levels are positively correlated with most sphingolipids in this cohort. In contrast, HDL levels are positively correlated with hexosylceramides but negative correlated with ceramides. Total triglycerides shows strong positive correlations with ceramides, but not with other sphingolipids. Blood pressure, insulin, glucose, and HbA1c levels are age dependent, and specifically associated with cerdmides.

4 Association analysis: logistic regression

We now enter the second last stage of the analysis: regression modeling of lipid levels at baseline as predictors of the clinical endpoint (DM incidence during follow-up). This data set comes from a prospective study – blood samples were drawn at recruitment (baseline) and the subjects were followed over time. This is an important point since the design allows for regression modeling in accordance with the hypothesized causal pathway: lipids modulate the risk of DM incidence.

The risk of DM is already determined by prominent risk factors such as age, BMI, and SBP, etc. Therefore, we start with a base model consisting of all known risk factors, add each lipid to the model, and see if the lipid still has residual statistical association with incident DM risk, after accounting for those risk factors. More complex models will follow later.

```
> ### Causal pathway: Lipid --> DM incidence,
> ### controlling for HbA1c at baseline, age, gender
> ### Start the model with no lipid, "base model"
> ### Then run through a for loop to test contribution of one lipid
> sdata$Gender = factor(sdata$Gender, levels=c(1,2))
> baseModel = glm(DM ~ Age + Gender + BMI + HbA1c + SBP + HDL + LDL + TG,
+                 family=binomial, data=sdata)
> summary(baseModel)

Call:
glm(formula = DM ~ Age + Gender + BMI + HbA1c + SBP + HDL + LDL +
    TG, family = binomial, data = sdata)


Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6194  -0.3555  -0.2049  -0.1200   3.5031
```

```
Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -20.750393   1.867394 -11.112  < 2e-16 ***
Age           0.034744   0.010815   3.213  0.00132 **
Gender2       0.627206   0.216344   2.899  0.00374 **
BMI           0.133470   0.027403   4.871 1.11e-06 ***
HbA1c         2.040225   0.281218   7.255 4.02e-13 ***
SBP           0.017534   0.005336   3.286  0.00102 **
HDL          -0.724902   0.368472  -1.967  0.04915 *
LDL          -0.189623   0.123222  -1.539  0.12383
TG            0.437113   0.140752   3.106  0.00190 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 980.61  on 1908  degrees of freedom
Residual deviance: 735.71  on 1900  degrees of freedom
  (390 observations deleted due to missingness)
AIC: 753.71

Number of Fisher Scoring iterations: 6
```

This base model suggests that most parameter contributes to the increased (log) odds of DM incidence, except the protective effect of HDL (LDL is not statistically significant), adjusting for the effects of one another. Our next operation is to throw one lipid at a time into the model and see if the lipid attains significantly significant association with the odds of DM incidence.

Furthermore, we would later compare the effect size of lipids. To this end, we first standardize the lipid data by subtracting the mean of log2 values and setting the standard deviation to 1 for each lipid. This way, the logistic regression coefficient for a lipid can be compared to another lipid on the standardized scale (e.g. one SD of lipid X versus one SD of lipid Y).

```
> nlipid = ncol(tmp)
> nsample = nrow(tmp)
> lipid.name = colnames(tmp)
> tmp2 = tmp
> for(k in 1:nlipid) {
+   mm = mean(tmp[,k], na.rm=TRUE)
+   ss = sd(tmp[,k], na.rm=TRUE)
+   tmp2[,k] = (tmp[,k] - mm) / ss
+ }
```

Once we have the data normalized, we next make placeholders for key statistics to record for each lipid, and run through a "for" loop as follows:

```
> ### Build model (logistic)
> coef.logit = rep(NA, nlipid)
> pval.logit = rep(NA, nlipid)
> for(k in 1:nlipid) {
+   tmpModel = glm(DM ~ Age + Gender + BMI + HbA1c + SBP + HDL + LDL + TG + tmp2[,k],
+                  family=binomial, data=sdata)
+   ### Last variable k-th column of tmp2 holds the data for lipid k
+   nr = nrow(summary(tmpModel)$coef)
+   coef.logit[k] = summary(tmpModel)$coef[nr,1]
+   pval.logit[k] = summary(tmpModel)$coef[nr,4]
+   if(k %% 50 == 0) print(k)
+ }

[1] 50
[1] 100

> hist(pval.logit, breaks=50)

> # qvalue or Benjamini-Hochberg (BH)
> #if (!requireNamespace("BiocManager", quietly = TRUE))
> #  install.packages("BiocManager")
> #BiocManager::install("qvalue")
> library(qvalue)
> qval.logit = qvalue(pval.logit)$qvalues
> plot(pval.logit, qval.logit, cex=.3, pch=19)
> abline(0,1,lty=2) ### properly modeled, it seems
> tab.logit = data.frame(lipid=lipid.name,
+                        coefficient=round(coef.logit, 2),
+                              pval=pval.logit, qval=qval.logit,
+                              stringsAsFactors=FALSE, check.names=FALSE)
> tab.logit = tab.logit[order(tab.logit$pval), ]
```

At this point, we have a table with statistical associations for lipids, with multiple testing correction (q-value). Since the total number of significant lipids is modest, we can be lenient with the FDR threshold, as the total number of expected false positive is also small. We draw a "volcano" plot with regression coefficients in the X-axis and -log10 p-value in the Y-axis, with significant findings in red (FDR 20%).

```
> # volcano plot with labels for significant ones
> plot(coef.logit, -log10(pval.logit), pch=19, cex=.8, col=alpha("gray", 0.5))
> sid = qval.logit <= 0.2
> points(coef.logit[sid], -log10(pval.logit[sid]), pch=19, cex=.8, col=alpha("red", 0.5))
> tab.logit[tab.logit$qval <= 0.2, ]

                lipid coefficient         pval       qval
106    SM d16:1/C18:0        0.41 0.0006333953 0.08424157
116    SM d18:1/C18:0        0.33 0.0019439750 0.09456536
12    Cer d18:0/C18:0        0.43 0.0021330533 0.09456536
104 MHCer d18:2/C25:0       -0.26 0.0060745050 0.18086810
23    Cer d18:1/C18:0        0.32 0.0079711937 0.18086810
117    SM d18:1/C20:0        0.28 0.0081594632 0.18086810
2     Cer d16:1/C18:0        0.31 0.0100143746 0.19027312
```

This table contains the lipids with residual statistical significance after adjusting for known risk factors. However, this does not imply that these lipids have predictive properties: association is a weaker condition than predictive-ness. Going into prediction analysis probably takes another tutorial session involving receiver operating characteristic (ROC) and its area under the curve (AUC).

5 Association analysis with Cox regression (time-to-event)

The logistic regression analysis classifies each subject into two categories: incident DM case, or control (event free). In prospective cohort studies, they often record the "time-to-event", i.e. the time from baseline to DM diagnosis. For those subjects with no event recorded, time refers to the time till last follow-up. In some cases, study participants are lost during follow-up for various reasons, and these participants contribute to the risk profile estimation up to that point only. This is called censoring. In our example data, however, we were not able to retrieve the time to last follow-up accurately due to technical reasons.

Therefore, we will add an artificially large number as the follow-up time, pretending that all of them were followed through the entire study period. Even if we do this, the association model with time information often gives equivalent or better sensitivity of detecting true associations.

```
> ### Step 2: Cox model with time info (time-to-event)
> #install.packages("survival")
> library(survival)
> mid = is.na(sdata$Days)
> sdata$Days[mid] = 10000    ### Artificial follow-up date for event-free subjects
> baseModelCox = coxph(Surv(Days, DM) ~ Age + Gender + BMI + HbA1c + SBP + HDL + LDL + TG,
+                      data=sdata)
> summary(baseModelCox)

Call:
coxph(formula = Surv(Days, DM) ~ Age + Gender + BMI + HbA1c +
    SBP + HDL + LDL + TG, data = sdata)

  n= 1909, number of events= 136
   (390 observations deleted due to missingness)

             coef exp(coef)  se(coef)      z Pr(>|z|)
Age      0.033930  1.034512  0.009768  3.474 0.000514 ***
Gender2  0.553661  1.739609  0.188266  2.941 0.003273 **
BMI      0.119923  1.127411  0.022433  5.346 8.99e-08 ***
HbA1c    1.845945  6.334082  0.241374  7.648 2.05e-14 ***
SBP      0.016046  1.016176  0.004630  3.465 0.000529 ***
HDL     -0.768789  0.463574  0.334563 -2.298 0.021568 *
LDL     -0.184168  0.831796  0.108583 -1.696 0.089865 .
TG       0.320409  1.377691  0.117168  2.735 0.006245 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

        exp(coef) exp(-coef) lower .95 upper .95
Age        1.0345     0.9666    1.0149    1.0545
Gender2    1.7396     0.5748    1.2028    2.5160
BMI        1.1274     0.8870    1.0789    1.1781
```

```
HbA1c      6.3341     0.1579    3.9466   10.1658
SBP        1.0162     0.9841    1.0070    1.0254
HDL        0.4636     2.1572    0.2406    0.8931
LDL        0.8318     1.2022    0.6723    1.0291
TG         1.3777     0.7259    1.0950    1.7333

Concordance= 0.84  (se = 0.016 )
Likelihood ratio test= 250.3  on 8 df,   p=<2e-16
Wald test            = 232.6  on 8 df,   p=<2e-16
Score (logrank) test = 265.7  on 8 df,   p=<2e-16
```

If you compare the base Cox model to the base logistic model fitted above, you will see that the association patterns (direction of regression coefficients, statistical significance scores) are quite consistent. If the follow-up times were properly recorded and there were many loss-of-follow-up cases, this consistency would have been shaken (possibly).

Now, we proceed to throw one lipid at a time onto this Cox model, by running through a for loop as before:

```
> coef.cox = rep(NA, nlipid)
> pval.cox = rep(NA, nlipid)
> for(k in 1:nlipid) {
+   tmpModel = coxph(Surv(Days, DM) ~ Age + Gender + BMI + HbA1c + SBP + HDL + LDL + TG + tmp2[,k],
+                    data=sdata)
+   nr = nrow(summary(tmpModel)$coef)
+   coef.cox[k] = summary(tmpModel)$coef[nr,1]  ### In Cox, they don't report intercept in summary()
+   pval.cox[k] = summary(tmpModel)$coef[nr,5]  ### Also, need to track the columns
+   if(k %% 50 == 0) print(k)
+ }

[1] 50
[1] 100

> hist(pval.cox, breaks=100)
```

And multiple testing correction by q-value:

```
> # qvalue or BH
> #if (!requireNamespace("BiocManager", quietly = TRUE))
> #  install.packages("BiocManager")
> #BiocManager::install("qvalue")
> library(qvalue)
> qval.cox = qvalue(pval.cox)$qvalues
> plot(pval.cox, qval.cox, cex=.3, pch=19)
> abline(0,1,lty=2) ### properly modeled, it seems
> tab.cox = data.frame(lipid=lipid.name, coefficient=round(coef.cox, 2),
+                   pval=pval.cox, qval=qval.cox,
+                   stringsAsFactors=FALSE, check.names=FALSE)
> tab.cox = tab.cox[order(tab.cox$pval), ]
```

Again, we see that the results for statistical significance (FDR 20%) are almost identical to that of the logistic regression analysis above.

```
> # volcano plot with labels for significant ones
> plot(coef.cox, -log10(pval.cox), pch=19, cex=.8, col=alpha("gray", 0.5))
> sid = (qval.cox <= 0.2)
> points(coef.cox[sid], -log10(pval.cox[sid]), pch=19, cex=.8, col=alpha("red", 0.5))
> tab.cox[tab.cox$qval <= 0.2, ]
```

```
                lipid coefficient         pval       qval
106    SM d16:1/C18:0        0.40 0.0001571633 0.02090271
116    SM d18:1/C18:0        0.31 0.0003898359 0.02592409
12    Cer d18:0/C18:0        0.41 0.0006470032 0.02868381
23    Cer d18:1/C18:0        0.31 0.0027163872 0.09031987
117    SM d18:1/C20:0        0.26 0.0037006416 0.09843707
104 MHCer d18:2/C25:0       -0.22 0.0049314165 0.10931306
49  DHCer d18:1/C18:0        0.25 0.0081742495 0.15531074
2     Cer d16:1/C18:0        0.27 0.0107700975 0.17905287
```

One way to visually verify the impact of a molecule selected by Cox regression on the DM risk is to draw Kaplan-Meier curves in percentile groups. For simplicity, we look at above and below mean value for each lipid. See the code below:

```
> ##### Kaplan-Meier curve
> sdata$`SM d16:1/C18:0` = tmp2$`SM d16:1/C18:0`
> km1 = survfit(Surv(Days, DM) ~ (`SM d16:1/C18:0` > 0), data=sdata)  ## Fit curve
> km1.diff = survdiff(Surv(Days, DM) ~ (`SM d16:1/C18:0` > 0), data=sdata)  ## Test differences
> sdata$`Cer d18:0/C18:0` = tmp2$`Cer d18:0/C18:0`
> km2 = survfit(Surv(Days, DM) ~ (`Cer d18:0/C18:0` > 0), data=sdata)
> km2.diff = survdiff(Surv(Days, DM) ~ (`Cer d18:0/C18:0` > 0), data=sdata)
> plot(km1, lty=1, col=c("red","blue"), mark.time=TRUE,
+      xlab="Days", ylab="Survival", main="SM d16:1/C18:0", xlim=c(0,3000))
> legend("bottomleft", c("Above zero", "Below zero"), lty=1, col=c("red","blue"))
> p.val.1 <- 1 - pchisq(km1.diff$chisq, length(km1.diff$n) - 1)  ## Log-rank test
> text(2000, 0.8, paste("p=", round(p.val.1, 4), sep=""))
> plot(km2, lty=1, col=c("red","blue"), mark.time=TRUE,
+      xlab="Days", ylab="Survival", main="Cer d18:0/C18:0", xlim=c(0,3000))
> legend("bottomleft", c("Above zero", "Below zero"), lty=1, col=c("red","blue"))
> p.val.2 <- 1 - pchisq(km2.diff$chisq, length(km2.diff$n) - 1)
> text(2000, 0.8, paste("p=", round(p.val.2, 4), sep=""))
```

While the sphingolipid species seem to segregate the two groups in the later stage of the follow-up (5 years and beyond), the ceramide species can separate the risk profiles from the early period (slightly) better.

6 Multi-variable prediction modeling I: regression with LASSO and group LASSO penalty

The obvious next step is to build a predictive multi-variable model to understand the predictive equation where the effect of one variable on the odds of DM incidence is adjusted for other variables. Before we explore various forms of regularized regression, we first impute missing values using K-nearest neighbor.

```
> #install.packages("grpreg")
> #install.packages("glmnet")
> #if (!require("BiocManager", quietly = TRUE))
> #  install.packages("BiocManager")
> #BiocManager::install("impute")
> library(impute)
> tmpX = sdata[,5:14]
> tmpX.new = impute.knn(as.matrix(tmpX))$data
```

```
Cluster size 2299 broken into 1395 904
Done cluster 1395
Done cluster 904
```

```
> ### K-nearest neighbor (default K=10)
> ### Find the most correlated subjects (not variables)
> ### and estimate the missing data by the average of the 10.
>
> Y = sdata$DM  ### Event 0=noDM / 1=DM
> Yt = sdata$Days  ### Follow-up duration or time to event
> X = data.frame(tmpX.new, tmp)  ### tmp object holds the lipid data
> ### In LASSO implementations, each variable is usually standardized within
> ### But just in case, we'll do that now before running the methods
>
> for(k in 1:ncol(X)) {
+   mm = mean(X[,k], na.rm=TRUE)
+   ss = sd(X[,k], na.rm=TRUE)
+   X[,k] = (X[,k] - mm) / ss
+ }
```

You may wish to check the imputed values, whether they are reasonable numbers within the range of non-missing values (check Hb1Ac in the table, for instance). Once all data points are confirmed, we proceed to model fitting. See the syntax for glmnet function below, which provides a diverse collection of regularized regression, including ridge regression, LASSO, and elastic net. These are different names for different penalty imposed on the regression coefficients.

The idea of regularized regression model is simple. When we have too many variables, fitting a regression model that reduces the magnitude of coefficients for redundant variables helps us build a predictive model with better prediction error. Regression coefficients will be biased, but only by a little bit. Meanwhile, by limiting the ability of individual predictors to influence on the predicted outcome, we can make the variance of predictions a lot smaller (making predictions more stable). This trade-off, sacrificing a little bit of bias to gain in variance, is called bias-variance trade-off.

Take logistic regression with LASSO penalty as example:

```
> library(glmnet)
> library(grpreg)
> logit.lasso = glmnet(x=as.matrix(X), y=Y, family=binomial, alpha=1)
> plot(logit.lasso)
```

Note here that “alpha=1” tells the glmnet function to fit a regression with LASSO penalty. In LASSO, certain variables are completely removed from the model by penalizing their coefficients to zero, leading to automatic variable selection. In the diagram above, one line is the solution trajectory for the coefficient of one variable across the range of penalty levels, or regularization parameter λ (in L_1 norm). The tick marks on top of the diagram show how many variables were retained at different λ’s. The model on the far left therefore represents a model with all coefficients penalized to zero – the simplest prediction model of all. The one on the far right represents the ordinary least squares solution (if it exists). The optimal solution is somewhere in the middle.

Let us digress a little bit here, though. Coming back to the parameter “alpha”, if alpha is set to 0, the regression is called Ridge Regression, where the coefficients are penalized but not completely obliterated to zero (L_2 norm). If alpha is set to any value between 0 and 1, it corresponds to Elastic Net (ENet) regression. In the end, the parameter alpha plays the role of weighting coefficients between two different ways to penalize coefficients, LASSO and ridge penalties. As usual, a compromise between two extreme forms usually yields an optimal solution: ENet is known to have a nice property to deal better with correlated predictors than LASSO. While LASSO often picks the best winner among correlated predictors, ENet tends to retain them together, with coefficients shared among them in the same direction (positive / negative).

Coming back to the operation side of the model fitting, whichever model you choose to work with, regularized regression analysis requires optimization of λ:

```
> logit.lasso.cv = cv.glmnet(x=as.matrix(X), y=Y, family=binomial, alpha=1)
> plot(logit.lasso.cv)
> bestlambda = logit.lasso.cv$lambda.min
> bestLASSO = predict(logit.lasso, type="coefficients", s=bestlambda)
> #bestLASSO
> #bestlambda2 = logit.lasso.cv$lambda.1se  ### 1st dev rule
> #bestLASSO2 = predict(logit.lasso, type="coefficients", s=bestlambda2)
```

As you see above, we normally use cross-validation (a re-sampling technique within the training data) to find the best λ. The line shows at what λ value the overall objective function (equivalent to the penalized likelihood of logistic regression model in log scale) is minimized, around $\log(\lambda) \approx -5.5$, retaining about 39 predictors.

The same routine can be run to obtain elastic net regression solution path and optimal solution:

```
> logit.enet = glmnet(x=as.matrix(X), y=Y, family=binomial, alpha=0.2)
> plot(logit.enet)
> logit.enet.cv = cv.glmnet(x=as.matrix(X), y=Y, family=binomial, alpha=0.2)
> plot(logit.enet.cv)
> bestlambda = logit.enet.cv$lambda.min
> bestENET = predict(logit.enet, type="coefficients", s=bestlambda)
> #bestENET
```

The two models report similar coefficients, but ENet penalty keeps more predictors than LASSO penalty.

```
> #cbind(bestLASSO, bestENET)
```

Some people may feel uncomfortable with too many lipids of different classes of lipids being selected in the optimal prediction model. They would prefer incorporating prior knowledge in the variable selection. For instance, one may wish to have lipids of the same class, or those with the same fatty acyl chain length to be either selected or excluded together. In this case, group LASSO penalty can be used to regularize them jointly.

```
> ### Define variable groups
> var.class = colnames(X)
> var.class[grep("Cer.d16.1", colnames(X))] = "Cer_d16_1"
> var.class[grep("Cer.d18.0", colnames(X))] = "Cer_d18_0"
> var.class[grep("Cer.d18.1", colnames(X))] = "Cer_d18_1"
> var.class[grep("Cer.d18.2", colnames(X))] = "Cer_d18_2"
> var.class[grep("DHCer.d18.1", colnames(X))] = "DHCer_d18_1"  ### This will override two lines above
> var.class[grep("GM3.d16.1", colnames(X))] = "GM3_d16_1"
> var.class[grep("GM3.d18.1", colnames(X))] = "GM3_d18_1"
> var.class[grep("GM3.d18.2", colnames(X))] = "GM3_d18_2"
> var.class[grep("MHCer.d16.1", colnames(X))] = "MHCer_d16_1"
> var.class[grep("MHCer.d18.0", colnames(X))] = "MHCer_d18_0"
> var.class[grep("MHCer.d18.1", colnames(X))] = "MHCer_d18_1"
> var.class[grep("MHCer.d18.2", colnames(X))] = "MHCer_d18_2"
> var.class[grep("SM.d16.1", colnames(X))] = "SM_d16_1"
> var.class[grep("SM.d18.1", colnames(X))] = "SM_d18_1"
> var.class[grep("SM.d18.2", colnames(X))] = "SM_d18_2"
> var.class[grep("Sphd", colnames(X))] = "Sphd"
> levs = unique(var.class)
> var.class = factor(var.class, levels=levs)
> ### Fit the regression with group LASSO penalty, using ``grpreg'' function (not glmnet)
> grpLASSO.fit = grpreg(X=as.matrix(X), y=Y, group=var.class, family="binomial", alpha=1, penalty="grLas
> plot(grpLASSO.fit)  ### Solution path
> cv.grpLASSO = cv.grpreg(X=as.matrix(X), y=Y, group=var.class, family="binomial", alpha=1, penalty="grL
> plot(cv.grpLASSO)  ### Cross-validation to find the optimal regularization parameter(s)
> bestlam.grp = cv.grpLASSO$lambda.min
> bestGrpLASSO = coef(grpLASSO.fit, lambda=bestlam.grp)
```

Note that the optimization is now with regard to how many groups, not individual variables, will be selected (top of the cross-validation error plot). Now let's see which groups of variables have been included in the final predictive model:

```
> barplot(bestGrpLASSO[-1], las=2, ylim=c(-0.3,0.3), cex.names=0.3, mar=c(10,10))
```

Now let us compare the coefficients from the three models we've tried:

```
> round( cbind(bestLASSO, bestENET, bestGrpLASSO) , 3)

144 x 3 sparse Matrix of class "dgCMatrix"
                        s1     s1 bestGrpLASSO
(Intercept)         -3.422 -3.293       -3.313
Age                  0.326  0.278        0.349
BMI                  0.161  0.153        0.238
HbA1c                0.534  0.441        0.539
```

SBP	0.232	0.223	0.234
HDL	-0.021	-0.026	-0.051
LDL	.	.	.
CRP	.	.	0.005
TG	0.034	0.033	0.130
Insulin	0.112	0.122	0.153
Glucose	0.416	0.374	0.434
Cer.d16.1.C16.0	0.015	0.042	.
Cer.d16.1.C18.0	.	.	.
Cer.d16.1.C20.0	.	.	.
Cer.d16.1.C22.0	.	.	.
Cer.d16.1.C23.0	.	.	.
Cer.d16.1.C24.0	.	.	.
Cer.d16.1.C24.1	.	0.010	.
Cer.d16.1.C25.1	.	.	.
Cer.d16.1.C26.0	.	.	.
Cer.d16.1.C26.1	.	.	.
Cer.d18.0.C16.0	.	.	0.001
Cer.d18.0.C18.0	0.179	0.123	0.015
Cer.d18.0.C20.0	0.027	0.039	-0.002
Cer.d18.0.C22.0	.	0.033	0.000
Cer.d18.0.C22.1	.	0.005	-0.003
Cer.d18.0.C23.0	0.060	0.066	0.008
Cer.d18.0.C24.0	.	.	0.002
Cer.d18.0.C24.1	.	0.024	0.004
Cer.d18.0.C25.0	.	.	-0.007
Cer.d18.0.C26.0	.	.	-0.006
Cer.d18.1.C14.0	.	.	0.000
Cer.d18.1.C16.0	.	.	-0.002
Cer.d18.1.C18.0	.	0.023	0.019
Cer.d18.1.C20.0	.	.	-0.011
Cer.d18.1.C22.0	.	.	-0.010
Cer.d18.1.C23.0	.	.	0.012
Cer.d18.1.C23.1	.	.	-0.013
Cer.d18.1.C24.0	.	.	0.002

```
Cer.d18.1.C24.1    0.160  0.094         0.021
Cer.d18.1.C25.0    .      .            -0.017
Cer.d18.1.C25.1    .      .             0.007
Cer.d18.1.C26.0    .      .             0.004
Cer.d18.1.C26.1    .      .            -0.008
Cer.d18.2.C14.0    .      .             0.031
Cer.d18.2.C16.0   -0.028 -0.037        -0.034
Cer.d18.2.C18.0    .      .             0.028
Cer.d18.2.C20.0    .      .            -0.039
Cer.d18.2.C22.0    .      .             0.008
Cer.d18.2.C23.0    .      .            -0.045
Cer.d18.2.C24.0    .      .             0.043
Cer.d18.2.C24.1    .      .             0.052
Cer.d18.2.C25.0   -0.227 -0.168        -0.093
Cer.d18.2.C25.1    0.024  0.025         0.062
Cer.d18.2.C26.0    .      .             0.004
Cer.d18.2.C26.1    .      .            -0.017
DHCer.d16.1.C16.0  .      .             .
DHCer.d18.1.C14.0  .      .            -0.026
DHCer.d18.1.C16.0  .      .             0.014
DHCer.d18.1.C18.0  0.224  0.174         0.070
DHCer.d18.1.C20.0  .      .             0.027
DHCer.d18.1.C22.0  .     -0.005        -0.018
DHCer.d18.1.C23.0  .      .            -0.014
DHCer.d18.1.C24.0 -0.021 -0.024        -0.020
DHCer.d18.1.C24.1  .      .            -0.030
DHCer.d18.2.C16.0  .      .             0.024
DHCer.d18.2.C24.1 -0.020 -0.031        -0.041
GM3.d16.1.C14.0    0.026  0.032         .
GM3.d16.1.C16.0    .      .             .
GM3.d16.1.C18.0    .      .             .
GM3.d16.1.C20.0    .      .             .
GM3.d16.1.C22.0   -0.056 -0.029         .
GM3.d16.1.C24.0    .      .             .
GM3.d18.1.C14.0    .      .             .
```

```
GM3.d18.1.C16.0    .      .            .
GM3.d18.1.C18.0   -0.115 -0.088        .
GM3.d18.1.C20.0   -0.001 -0.029        .
GM3.d18.1.C22.0    .      .            .
GM3.d18.1.C24.0   -0.025 -0.037        .
GM3.d18.1.C24.1    .      .            .
GM3.d18.2.16.0     .      .            0.000
GM3.d18.2.18.0    -0.040 -0.049       -0.053
GM3.d18.2.24.0     .      .           -0.016
MHCer.d16.1.C16.0  .      .            0.035
MHCer.d16.1.C20.0  0.041  0.046        0.067
MHCer.d16.1.C22.0  .      0.017        0.078
MHCer.d16.1.C23.0  .      .            0.024
MHCer.d16.1.C24.0 -0.060 -0.047       -0.143
MHCer.d16.1.C24.1  .      .           -0.048
MHCer.d18.0.C16.0  .      .            .
MHCer.d18.0.C16.1 -0.067 -0.062        .
MHCer.d18.0.C22.0  .      .            .
MHCer.d18.0.C23.0  .      .            .
MHCer.d18.0.C24.0  .      .            .
MHCer.d18.0.C24.1  .      .            .
MHCer.d18.1.C14.0  0.082  0.056        0.052
MHCer.d18.1.C16.0  0.032  0.043        0.026
MHCer.d18.1.C18.0  .      0.004        0.029
MHCer.d18.1.C20.0  .      .           -0.014
MHCer.d18.1.C22.0  .      .            0.027
MHCer.d18.1.C23.0  .      .            0.032
MHCer.d18.1.C23.1  .      .           -0.005
MHCer.d18.1.C24.0  .      .            0.010
MHCer.d18.1.C24.1  .      .           -0.008
MHCer.d18.1.C25.0  .     -0.024       -0.048
MHCer.d18.1.C25.1 -0.177 -0.139       -0.059
MHCer.d18.1.C26.0 -0.094 -0.078       -0.070
MHCer.d18.1.C26.1  .      .            0.007
MHCer.d18.2.C16.0  .      .            0.027
```

MHCer.d18.2.C20.0	.	.	-0.007
MHCer.d18.2.C22.0	.	.	-0.003
MHCer.d18.2.C23.0	.	-0.028	-0.031
MHCer.d18.2.C24.0	.	.	0.012
MHCer.d18.2.C24.1	.	.	0.000
MHCer.d18.2.C25.0	-0.080	-0.076	-0.042
SM.d16.1.C16.0	.	.	-0.011
SM.d16.1.C18.0	0.211	0.142	0.128
SM.d16.1.C20.0	.	.	-0.045
SM.d16.1.C22.0	.	.	-0.033
SM.d16.1.C23.0	.	.	0.014
SM.d16.1.C24.0	.	.	0.001
SM.d16.1.C24.1	.	.	0.016
SM.d18.0.C16.0	.	.	.
SM.d18.1.C12.0	.	.	0.013
SM.d18.1.C14.0	.	.	-0.020
SM.d18.1.C16.0	.	0.006	0.007
SM.d18.1.C18.0	0.019	0.052	0.034
SM.d18.1.C20.0	.	.	-0.005
SM.d18.1.C22.0	-0.006	-0.014	-0.034
SM.d18.1.C22.1	-0.121	-0.099	-0.040
SM.d18.1.C23.0	0.102	0.084	0.045
SM.d18.1.C23.1	.	.	-0.011
SM.d18.1.C24.0	.	.	-0.002
SM.d18.1.C24.1	.	.	0.015
SM.d18.2.C14.0	.	.	.
SM.d18.2.C16.0	.	.	.
SM.d18.2.C18.0	.	.	.
SM.d18.2.C20.0	.	.	.
SM.d18.2.C22.0	.	.	.
SM.d18.2.C23.0	.	.	.
SM.d18.2.C24.0	-0.002	-0.017	.
SM.d18.2.C24.1	.	.	.
Sphd18.0	.	.	.
Sphd18.1	.	.	.

One can clearly see the benefit and shortfall of the group LASSO regression. In an effort to retain lipids of the same classes in the model, the model assigned very small coefficients to individual members of each lipid class. On the one hand, the model attained the result that the analyst wished, yet it ballooned the size/complexity of the model, and perhaps negated the advantage of sparse prediction models in the bias-variance trade-off spectrum.

7 Multi-variable prediction modeling II: Random Forest

The regression modeling was limited to linear models so far (there are non-liear regression models!). While linear models often perform well, we live in a time where machine learning is taking over our lives, especially in biomedical sciences. So let us also practice one of the most popular machine learning methods, the random forest (RF). RF is perhaps the most flexible model-free classification (and regression method) in the field – it is based on decision trees.

Literally speaking, RF is made of an ensemble of many, many decision trees. For a classification problem, a decision tree classifier partitions the feature / predictor space by splitting one variable into two segments at a time. When the tree is fully grown with splits (branches) and pruned back (for avoiding an overly complex form), each terminal node of the tree has observations assigned to them, and the majority class is the predicted class for future observations similar to them. However, one tree is too crude and rugged – so we grow a ton of trees and average the prediction results. But here is the catch: when we split the feature space, we randomly sample a subset of variables in each tree of the forest. This way, classification trees in the RF ensemble will explore many different predictors and the trees will look different from one another, often resulting in better classification results. If we don't do this sub-sampling, all trees will look about the same, and we're wasting time in growing all those trees. Hence, "random" forest, not just a forest.

```
> ### Random forest
> library(randomForest)
> set.seed(12345)
> nvar = ncol(X)
> X2 = data.frame(DM=Y, X, stringsAsFactors = FALSE, check.names = FALSE)
> ### Create a data frame with both response variable (DM) and predictive features (X)
> X2$DM = factor(X2$DM, levels=c(0,1))  ### Response variable has to be a factor; X2 object holds the tr
> rf.fit = randomForest(DM ~ ., data=X2, mtry = sqrt(nvar), importance = TRUE, ntree = 1000)
> ### RF fitting
> #rf.fit
>
```

```
> yhat.rf = predict(rf.fit, newdata=X2)
> ### Making predictions onto the training data itself (no test data available)
> table(yhat.rf, X2$DM)
```

```
yhat.rf    0    1
      0 2129    0
      1    0  170
```

```
> ### Does fairly well.
```

Since RF uses all lipids and clinical variables as predictive features without variable selection, we need a device to rank the importance of individual predictors. The variable importance score provides this metric:

```
> itab = importance(rf.fit)   ### Variable importance measures
> ord = order(itab[,3], decreasing=TRUE)
> itab = itab[ord, ]
> itab[1:20,]
```

```
                          0          1 MeanDecreaseAccuracy MeanDecreaseGini
Glucose           10.075084 19.4257146            16.702045        11.553981
HbA1c              3.260418 25.0584132            15.917307        14.303094
Cer.d18.0.C18.0   13.459909  3.6585003            13.917201         4.761408
Cer.d18.0.C24.1   13.152651 -1.2013497            13.156429         3.872483
TG                11.004439  1.3651410            11.344900         3.110681
Cer.d18.0.C22.0   10.407156 -0.1632416            10.753026         2.831983
Cer.d18.0.C20.0   10.279687 -1.5143874            10.333312         2.650097
BMI                8.419380  6.7467729            10.281030         5.199771
Cer.d18.2.C25.0   10.301872 -2.6175038            10.056760         1.881201
Cer.d18.1.C24.1    9.913788 -0.6379101             9.891785         2.310736
Cer.d18.0.C23.0    9.604177 -0.7631056             9.712717         2.258878
Cer.d18.0.C24.0    9.823555 -2.6402698             9.655357         2.089232
Cer.d18.1.C26.1    9.869614 -3.5322507             9.524040         2.087059
Cer.d16.1.C18.0    8.746160  3.2113854             9.423606         2.444253
Cer.d16.1.C24.1    9.037959 -1.8452294             9.021708         1.869604
MHCer.d18.2.C23.0  7.832514  2.1512606             8.642110         2.609860
Cer.d16.1.C20.0    8.746980 -2.3803121             8.556734         1.885955
Cer.d16.1.C23.0    8.498128 -2.3040312             8.488961         1.615274
```

```
MHCer.d18.1.C26.1  8.349465  0.4551527              8.318574         2.256383
Cer.d18.1.C25.0    8.652588 -3.3950331              8.317508         1.694079
```

Although the ranking may be discrepant between RF and regression models, close examination will reveal that the predictors with high importance scores tend to be with non-zero coefficients in the regularized models.

A final note: RF does not do variable selection as LASSO. There are a variety of ways one can incorporate explicit feature selection step with RF (or any other classifiers). This involves evaluation of multiple RFs with different sets of input variables, mostly through the evaluation of ROC or other metrics of classification accuracy. This is a topic for another day.

8 Network of statistical correlations via sparse precision matrix estimation

Finally, we cover the very last topic of this course – correlations among the predictors of T2D risk. Many investigators resort to Pearson / Spearman correlation to examine the relationship between predictors / variables. This is a perfectly valid practice, with only one problem: as you calculate correlations among many variables, some of those correlations represent passenger correlations created by transitivity. Further, in a dataset like this with large $N = 2,299$, most correlations will be considered statistically significant correlations, regardless of their "effect sizes."

For this reason, a more reasonable, interpretable correlation structure in high-dimensional data can be delineated by calculating partial correlations: the conditional dependence between two variables after accounting for the impact of all other variables on those two. In a way, partial correlation is considered as correlations of residual effects. In fact, the partial correlation between variable X and variable Y is computed as follows. For two variables X and Y, we fit a regression model on each of them using all other variables as predictors, and record the residuals. The correlation between the residuals for X and the residuals for Y is the partial correlation between X and Y.

We do this using a graphical model estimation method called "GLASSO" by Friedman, Hastie, and Tibshirani (https://doi.org/10.1093/biostatistics/kxm045), re-implemented in HUGE package in R (https://jmlr.org/papers/volume13/zhao12a/zhao12a.pdf).

```
> ### GLASSO for network estimation
> #install.packages("huge")
> library(huge)
> library(gplots)
> glasso.out = huge(as.matrix(X), method="glasso")
```

```
Conducting the graphical lasso (glasso) wtih lossless screening....in progress: 9%
Conducting the graphical lasso (glasso) wtih lossless screening....in progress: 19%
Conducting the graphical lasso (glasso) wtih lossless screening....in progress: 30%
Conducting the graphical lasso (glasso) wtih lossless screening....in progress: 40%
Conducting the graphical lasso (glasso) wtih lossless screening....in progress: 50%
Conducting the graphical lasso (glasso) wtih lossless screening....in progress: 60%
Conducting the graphical lasso (glasso) wtih lossless screening....in progress: 70%
Conducting the graphical lasso (glasso) wtih lossless screening....in progress: 80%
Conducting the graphical lasso (glasso) wtih lossless screening....in progress: 90%
Conducting the graphical lasso (glasso)....done.
```

```
> out.select = huge.select(glasso.out, criterion = "ric") ### Rotation information criterion (Lysen 200
```

```
Conducting rotation information criterion (ric) selection....done
Computing the optimal graph....done
```

```
> #plot(out.select)
>
> glasso.fit = out.select$opt.icov  ### taking out the inverse covariance matrix
> rownames(glasso.fit) = colnames(glasso.fit) = colnames(X)
> glasso.fit = data.frame(Var=colnames(X), glasso.fit, stringsAsFactors=F, check.names=F)
> write.table(glasso.fit, "GLASSO.txt", sep="\t", quote=F, row.names=F)
> heatmap.2(as.matrix(glasso.fit[,-1]), trace="n", col=bluered(20),
+           breaks=seq(-0.1,0.1,by=0.01), cexRow=0.2, cexCol=0.2)
> prec = read.delim("GLASSO.txt", header=T, as.is=T, row.names=1, check.names=F)
> ### prec refers to precision value
```

The routines above generate the precision matrix (inverse covariance matrix) of lipids and clinical variables. Due to its penalized estimation technique like LASSO, we naturally get zero precision values for many pairs of variables, and non-zero values for others. The "surviving" precision values point to partial correlations between variables "worthy of further consideration." If you collect all pairs with non-zero partial correlations, this reveals a network among the variables. As for the strength of correlations, we can directly convert the precision values into partial correlations by the following step:

```
> nvar = nrow(prec)
> pcor = prec
> ### pcor refers to partial correlation
>
```

```
> for(i in 1:nvar) {
+   for(j in 1:nvar) {
+     pcor[i,j] = ifelse(i==j, 1, -1) * prec[i,j] / sqrt(prec[i,i]) / sqrt(prec[j,j])
+   }
+ }
> heatmap.2(as.matrix(pcor), trace="n", col=bluered(20),
+           breaks=seq(-0.1,0.1,by=0.01), cexRow=0.2, cexCol=0.2)
```

It is perhaps no coincidence that the strongest correlations are retained between lipids of the same fatty acyl chain length within each head group. LDL, HDL, and total TG are more correlated with sphingolipids than any other clinical characteristics.

Let us now pivot to data visualization with these results. We create input files for the great Cytoscape, freely available at https://cytoscape.org/. It requires two input files – network edge file and node attribute file. The former is a list of "interactions" between variables, and the latter is a list of characteristic of variables.

```
> #### Create source data for the network diagram
> vars = colnames(X)
> tvars = c(rep("Clinical",10), rep("Lipids",133))  ### Making class labels for variables
> tvars[grep("Cer.", vars)] = "Cer"
> tvars[grep("MHCer.", vars)] = "MHCer"
> tvars[grep("DHCer.", vars)] = "DHCer"
> tvars[grep("GM3.", vars)] = "GM3"
> tvars[grep("SM.", vars)] = "SM"
> tvars[grep("Sphd", vars)] = "Sphd"
> ### Node attribute table
> nodes = data.frame(Variable=vars, Type = tvars, stringsAsFactors = FALSE)
> ### Start building the edge attribute table
> v1 = v2 = v3 = NULL
> for(i in 1:(nvar-1)) {
+   for(j in i:nvar) {
+     if(pcor[i,j] != 0) {
+       v1 = c(v1, vars[i])
+       v2 = c(v2, vars[j])
+       v3 = c(v3, pcor[i,j])  ## partial correlation
+     }
+   }
+ }
> ### Create edge table
> ### For edge properties -- record partial correlation, absolute partial correlation,
> ### direction of partial correlation (positive / negative) for edge coloring
> edges = data.frame(A=v1, B=v2, pcorr=v3, pcorrAbs=abs(v3),
+                    pcorrSign=ifelse(v3 > 0, "pos", "neg"),
+                    stringsAsFactors = FALSE)
> ### Remove the lines with self-self interaction
> edges = edges[v1 != v2, ]
> ### Add another filter: absolute partial correlation > 0.05 (arbitrary cutoff)
> ### to make the graph represent the strongest correlations only
> edges = edges[edges$pcorrAbs >= 0.05, ]
> ### Export the files for network visualization
> write.table(nodes, "Nodes_data.txt", sep="\t", quote=F, row.names=F)
> write.table(edges, "Edges_data.txt", sep="\t", quote=F, row.names=F)
```

Now fire up Cytoscape and have fun visualizing!