plasma: Partial LeAst Squares for Multi-omics Analysis

Kevin R. Coombes

Contents

Background

Recent years have seen the development of numerous algorithms and computational packages for the analysis of multi-omics data sets. At this point, one can find several review articles summarizing progress in the field (Subramanian *et al.*, 2020; Graw *et al.*, 2021; Heo *et al.*, 2021; Picard *et al.*, 2021; Reel *et al.*, 2021; Vlachavas *et al.*, 2021; Adossa *et al.*, 2021). As with other applications of machine learning, the kinds of problems addressed by these algorithms are divided into two categories: unsupervised (e.g., clustering or class discovery) or supervised (including class comparison and class prediction) (Simon and Dobbin, 2003). Advances in the area of unsupervised learning have been broader and deeper than advances on supervised learning.

One of the most effective unsupervised methods is Multi-Omic Factor Analysis (MOFA) (Argelaguet *et al.*, 2018, 2020). A key property of MOFA is that it does not require all omics assays to have been performed on all samples under study. In particular, it can effectively discover class structure across omics data sets even when data for many patients have only been acquired on a subset of the omics technologies. As of this writing, we do not know of any supervised multi-omics method that can effectively learn to predict outcomes when samples have only been assayed on a subset of the omics data sets.

MOFA starts with a standard method – Latent Factor Analysis – that is known to work well on a single omics data set. It then fits a coherent model that identifies latent factors that are common to, and able to explain the data well in, all the omics data sets under study. Our investigation (unpublished) of the factors found by MOFA suggests that, at least in come cases, it is approximately equivalent to a two-step process:

1. Use principal components analysis to identify initial latent factors in each individual omics data set.
2. For each pair of omics data sets, use overlapping samples to train and extend models of each factor to the union of assayed samples.

That re-interpretation of MOFA suggests that an analogous procedure might work for supervised analyses as well. In this article, we describe a two-step algorithm, which we call "*plasma*", to find models that can predict time-to-event outcomes on samples from multi-omics data sets even in the presence of incomplete data. We use partial least squares (PLS) for both steps, using Cox regression (Bertrand and Maumy-Bertrand, 2021) to learn the single omics models and linear regression (Mishra and Liland, 2022) to learn how to extend models from one omics data set to another. To illustrate the method, we use a subset of the esophageal cancer (ESCA) data set from The Cancer Genome Atlas (TCGA).

Methods

Our computational method is implemented and the data are available in the `plasma` package.

```
suppressWarnings( library(plasma) )
packageVersion("plasma")
```

```
## [1] '0.9.5'
```

Data

The results included here are in whole or part based upon data generated by the TCGA Research Network. We downloaded the entire esophageal cancer Level 3 data set (Cancer Genome Atlas Research Network, 2017) from the FireBrowse web site (http://firebrowse.org/) (Deng *et al.*, 2017) in August 2018. We filtered the data sets so that only the most variable, and presumably the most informative, features were retained. Here, we load this sample data set.

```
loadESCAdata()
sapply(assemble, dim)
```

```
##      ClinicalBin ClinicalCont MAF Meth450 miRSeq mRNASeq RPPA
## [1,]          53            6 566    1454    926    2520  192
## [2,]         185          185 184     185    166     157  126
```

1. From TCGA, we obtained 162 columns of clinical, demographic, and laboratory data on 185 patients. We removed any columns that always contained the same value. We also removed any columns whose values were missing in more than 25% of the patients. We converted categorical variables into sets of binary variables using one-hot-encoding. We then separated the clinical data into three parts:
   1. Outcome (overall survival)
   2. Binary covariates (53 columns)
   3. Continuous covariates (6 columns)
2. Exome sequencing data for 184 patients with esophageal cancer was obtained as mutation allele format (MAF) files. We removed any gene that was mutated in fewer than 3% of the samples. The resulting data set contained 566 mutated genes.
3. Methylation data for 185 ESCA patients was obtained as beta values computed by the TCGA from Illumina Methylation 450K microarrays. We removed any CpG site for which the standard deviation of the beta values was less than 0.3. The resulting data set contained 1,454 highly variable CpG's.
4. Already normalized sequencing data on 2,566 microRNAs (miRs) was obtained for 185 patients. We removed any miR for which the standard deviation of normalized expression was less than 0.05, which left 926 miRs in the final data set.

5. Already normalized sequencing data on 20,531 mRNAs was obtained in 184 patients. We removed any mRNA whose mean normalized expression was less than 6 or whose standard deviation was less than 1.2. The final data set included 2,520 mRNAs.
6. Normalized expression data from reverse phase protein arrays (RPPA) was obtained from antibodies targeting 192 proteins in 126 patients. All data were retained for further analysis.

Finally, in order to be able to illustrate the ability of the plasma algorithm to work in the presence of missing data, we randomly selected 10% of the patients to remove from the miRSeq data set (leaving 166 patients) and 15% of the patients to remove from the mRNASeq data set (leaving 157 patients). We provide a summary of the outcome data below.

Imputation

We recommend imputing small amounts of missing data in the input data sets. The underlying issue is that the PLS models we use for individual omics data sets will not be able to make predictions on a sample if even one data point is missing. As a result, if a sample is missing at least one data point in every omics data set, then it will be impossible to use that sample at all.

For a range of available methods and R packages, consult the CRAN Task View on Missing Data. We also recommend the R-miss-tastic web site on missing data. Their simulations suggest that, for purposes of producing predictive models from omics data, the imputation method is not particularly important. Because of the latter finding, we have only implemented two simple imputation methods in the `plasma` package:

1. `meanModeImputer` will replace any missing data by the mean value of the observed data if there are more than five distinct values; otherwise, it will replace missing data by the mode. This approach works relatively well for both continuous data and for binary or small categorical data.
2. `samplingImputer` replaces missing values by sampling randomly from the observed data distribution.

```r
set.seed(54321)
imputed <- lapply(assemble, samplingImputer)
```

Computational Approach

The `plasma` algorithm is based on Partial Least Squares (PLS), which has been shown to be an effective method for finding components that can predict clinically interesting outcomes (Bastien *et al.*, 2015). The workflow of the plasma algorithm is illustrated in **Figure 1** in the case of three omics data sets. First, for each of the omics data sets, we apply the PLS Cox regression algorithm (`plsRcox` Version 1.7.6 (Bertrand and Maumy-Bertrand, 2021)) to the time-to-event outcome data to learn three separate predictive models (indicated in red, green, and blue, respectively). Each of these models may be incomplete, since they are not defined for patients who have not been assayed (shown in white) using that particular omics technology. Second, for each pair of omics data sets, we apply the PLS linear regression algorithm (`pls` Version 2.8.0 (Mishra and Liland, 2022)) to learn how to predict the coefficients of the Cox regression components from one data set using features from the other data set. This step extends (shown in pastel red, green, and blue, resp.) each of the original models, in different ways, from the intersection of samples assayed on both data sets to their union. Third, we average all of the different extended models (ignoring missing data) to get a single coherent model of component coefficients across all omics data sets. Assuming that this process has been applied to learn the model from a training data set, we can evaluate the final Cox regression model on both the training set and a test set of patient samples.

All computations were performed in R version 4.2.1 (2022-06-23 ucrt) of the R Statistical Software Environment (R Core Team, 2022). Cox proportional hazards models for survival analysis were fit using version 3.3.1 of the `survival` R package. We used additional exploratory graphical tools from version 1.3.1 of the `beanplot` R package (Kampstra, 2008) and version 1.5.1 of the `Polychrome` R package (Coombes *et al.*, 2019). Gene enrichment (pathway or annotation) analysis was performed using ToppGene (https://toppgene.cchmc.org/) (Chen *et al.*, 2009).

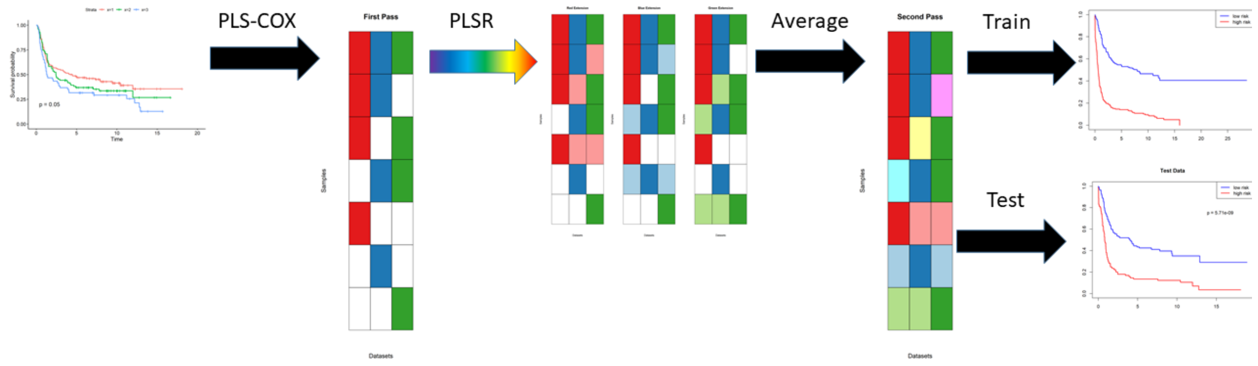


Figure 1: Workflow schematic for plasma algorithm with three omics data sets. See main text for an explanation.

Terminology

Because of the layered nature of the plasma algorithm, we intend to use the following terminology to help clarify the later discussions.

1. The input data contains a list of *omics data sets*.
2. Each omics data set contains measurements of multiple *features*.
3. The first step in the algorithm uses PLS Cox regression to find a set of *components*. Each component is a linear combination of features. The components are used as predictors in a Cox proportional hazards model, which predicts the log hazard ratio as a linear combination of components.
4. The second step in the algorithm creates a secondary layer of components. We do not give these components a separate name. They are not an item of particular focus; we view them as a way to extend the first level components to more samples by "re-interpreting" them in other omics data sets.

Preparing the Data

To be consistent with the `MOFA2` R package (Argelaguet *et al.*, 2020), all of the data sets are arranged so that patient samples are columns and assay features are rows. Our first task is to pad each data set with appropriate `NA`'s to ensure that each set includes the same patient samples in the same order, where that order matches the outcome data frame.

```
MO <- prepareMultiOmics(imputed, Outcome)
summary(MO)

## Datasets:
##      ClinicalBin ClinicalCont MAF Meth450 miRSeq mRNASeq RPPA
## [1,]          53            6 566    1454    926    2520  192
## [2,]         185          185 185     185    185     185  185
## Outcomes:

##    patient_id  vital_status days_to_death    days_to_last_followup      Days
##  a3i8    :  1  alive:108    Min.   :   9.0  Min.   :   4.0        Min.   :   4.0
##  a3ql    :  1  dead : 77    1st Qu.: 180.0  1st Qu.: 336.5        1st Qu.: 232.0
##  a3y9    :  1               Median : 351.0  Median : 402.5        Median : 400.0
##  a3ya    :  1               Mean   : 495.2  Mean   : 570.1        Mean   : 538.9
##  a3yb    :  1               3rd Qu.: 650.0  3rd Qu.: 696.8        3rd Qu.: 681.0
##  a3yc    :  1               Max.   :2532.0  Max.   :3714.0        Max.   :3714.0
##  (Other):179                NA's   :108     NA's   :77
```

We see that the number of patients in each data set is now equal to the number of patients with clinical outcome data.

Split Into Training and Test

As indicated above, we want to separate the data set into training and test samples. We will use 60% for training and 40% for testing.

```
set.seed(54321)
splitVec <- rbinom(nrow(Outcome), 1, 0.6)
```

Figure 2 presents a graphical overview of the number of samples (`N`) and the number of features (`D`) in each omics component of the training and test sets.

```
trainD <- MO[, splitVec == 1]
testD <- MO[, splitVec == 0]
opar <- par(mai = c(1.02, 1.32, 0.82, 0.22), mfrow = c(1,2))
plot(trainD, main = "Train")
plot(testD, main = "Test")

par(opar)
rm(opar)
```

Results

Individual PLS Cox Regression Models

The first step of the `plasma` algorithm is to fit PLS Cox models on each omics data set using the function `fitCoxModels`. The returned object of class `MultiplePLSCoxModels` contains a list of `SingleModel` objects, one for each assay, and within each there are three regression models:

- The `plsRcoxmodel` contains the coefficients of the components learned by PLS Cox regression. The number of components is determined automatically as a function of the logarithm of the number of features in the omics data set. The output of this model is a continuous prediction of "risk" for the time-to-event outcome of interest.
- Two separate models are constructed using the prediction of risk on the training data.
  - The `riskModel` is a `coxph` model using continuous predicted risk as a single predictor.
  - The `splitModel` is a `coxph` model using a binary split of the risk (at the median) as the predictor.

```
set.seed(23258)
suppressWarnings( firstPass <- fitCoxModels(trainD, timevar = "Days",
                             eventvar = "vital_status", eventvalue = "dead") )

if (!interactive()) {
  plot(firstPass, legloc = "bottomleft") # margins too small inside RStudio window
}
```

On the training set, each of the seven contributing omics data sets is able to find a PLS model that can successfully separate high risk from low risk patients (**Figure 3**).

Extend Model Components Across Omics Data Sets

The second step of the algorithm is to extend the individual omics-based models across other omics data sets. This step is performed using the `plasma` function, which takes in the previously created objects of class `multiOmics` and `MultiplePLSCoxModels`. The function operates iteratively, so in our case there are seven different sets of predictions of the PLS components. These different predictions are averaged and saved internally as a data frame called `meanPredictions`. The structure of models created and stored in the

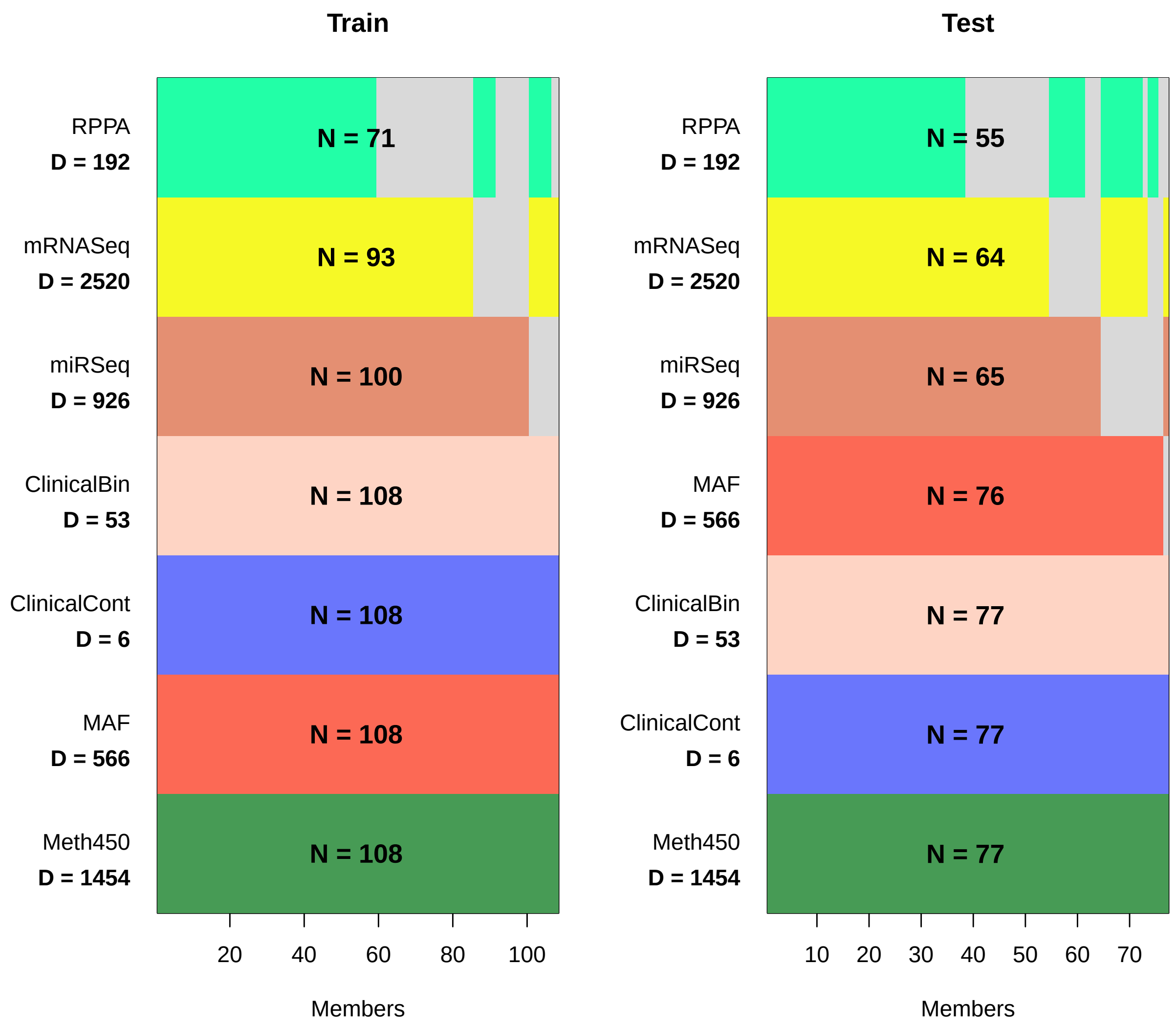


Figure 2: Overview of training and test data.

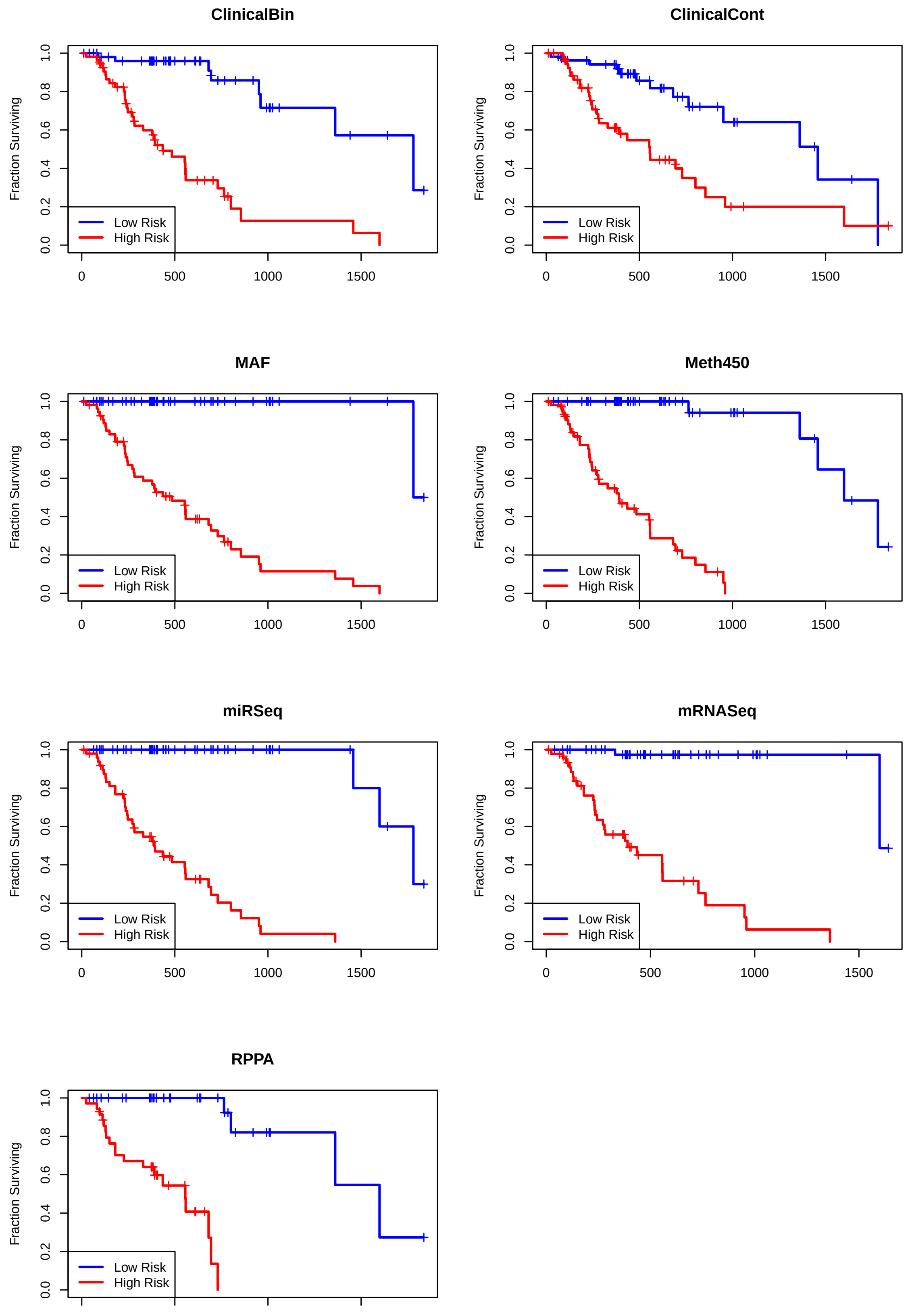


Figure 3: Kaplan-Meier plots of overall survival on the training set from separate PLS Cox omics models

`plasma` object is the same as for the separate, individual, omics models. **Figure 4A** shows the Kaplan-Meier plot using the predicted risk, split at the median value, on the training data set.

Independent Test Set

Now we want to see how well the final composite model generalizes to our test set. **Figure 4B** uses the predicted risk, split at the median of the training data, to construct a Kaplan-Meier plot on the test data. The model yields a statistically significant ($p = 0.0063$) separation of outcomes between the high and low risk patients.

```
pl <- plasma(MO, firstPass)
testpred <- predict(pl, testD)
```

```
opar <- par(mfrow=c(2,1))
plot(pl, legloc = "topright", main = "Training Data", xlab = "Time (Days)")
mtext("A", side = 2, at = 1.2, line = 3, cex = 1.5, las = 2)
plot(testpred, main="Testing Data", xlab = "Time (Days)")
mtext("B", side = 2, at = 1.2, line = 3, cex = 1.5, las = 2)
```

```
par(opar)
rm(opar)
```

Single Omics

We want to compare the test results of the joint omics model to the predictions we make on the test data from the separate single-omics models. The Kaplan-Meier plots in **Figure 5** show that most of the individual models have poor performance on the test data, especially when compared to the results of the combined model learned jointly from all the omics data sets.

```
mypreds <- predict(firstPass, testD, type = "survfit")
if (!interactive()) {
  opar <- par(mfrow = c(4, 2)) #, mai = c(0.2, 0.2, 0.2, 0.2))
  sapply(names(mypreds), function(N) {
    fit <- mypreds[[N]]
    col = c("blue", "red")
    plot(fit, col = col, lwd = 2, main= N, xlab = "Time (Days)",
         ylab = "Fraction Surviving")
    legend("topright", paste(c("low", "high"), "risk"), col = col, lwd = 2)
  })
  par(opar)
}
```

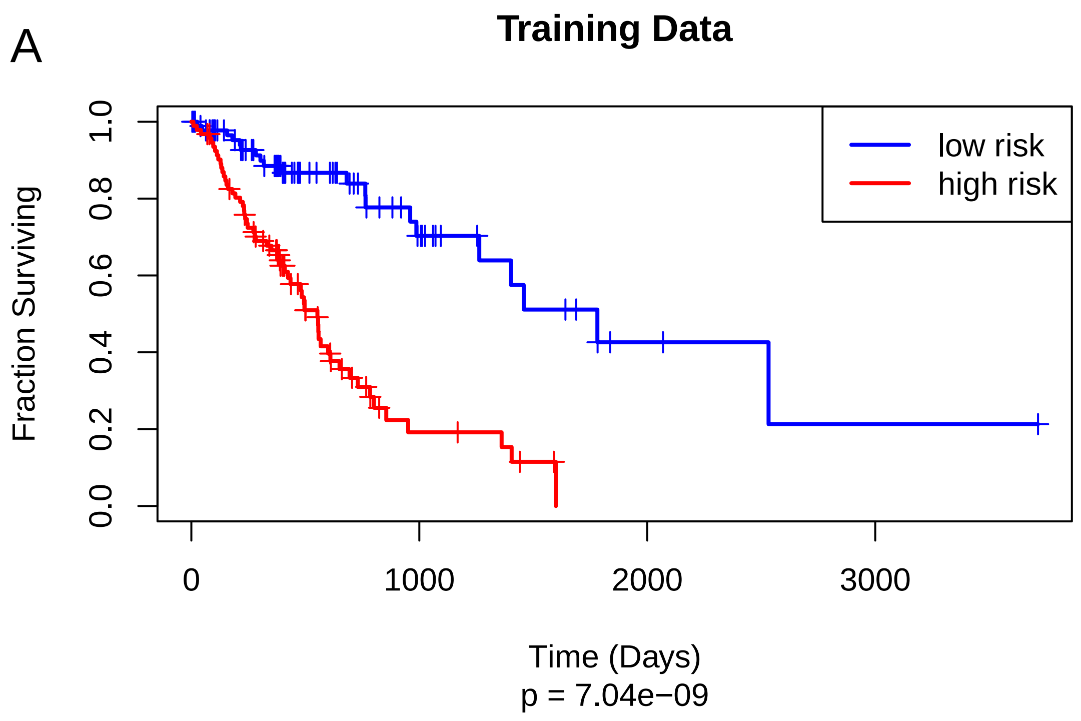
MOFA

Next, we want to compare the plasma method to an alternative approach using MOFA. Here the idea is to first run the unsupervised MOFA algorithm to find latent factors, and then use the latent factors as potential predictors in a Cox proportional hazards model. We use the training data from our plasma analysis as inputs to `MOFA2`.

```
hasMOFA <- suppressMessages( require("MOFA2", quietly = TRUE) )
```

```
suppressMessages( mofaObject <- create_mofa(trainD@data) )
```

We set the options to the MOFA algorithm by indicating which data sets are binary and by choosing to find 10 latent factors.



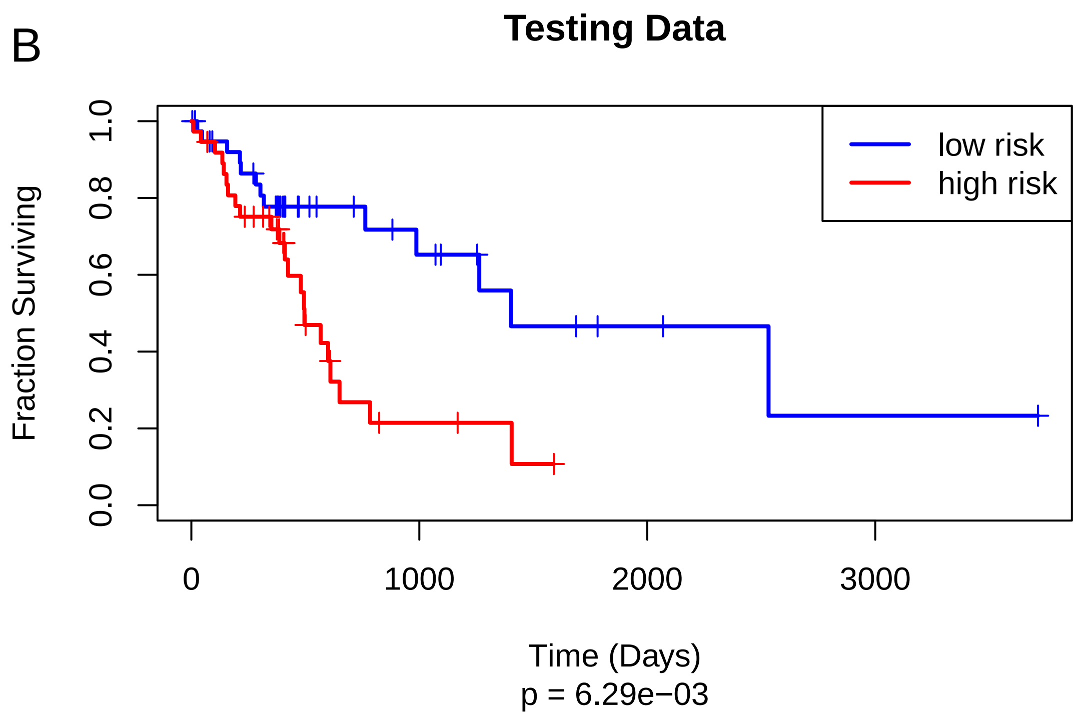


Figure 4: Kaplan-Meier plot of overall survival on (A) the training set and (B) the test set using the unified `plasma` Cox model.

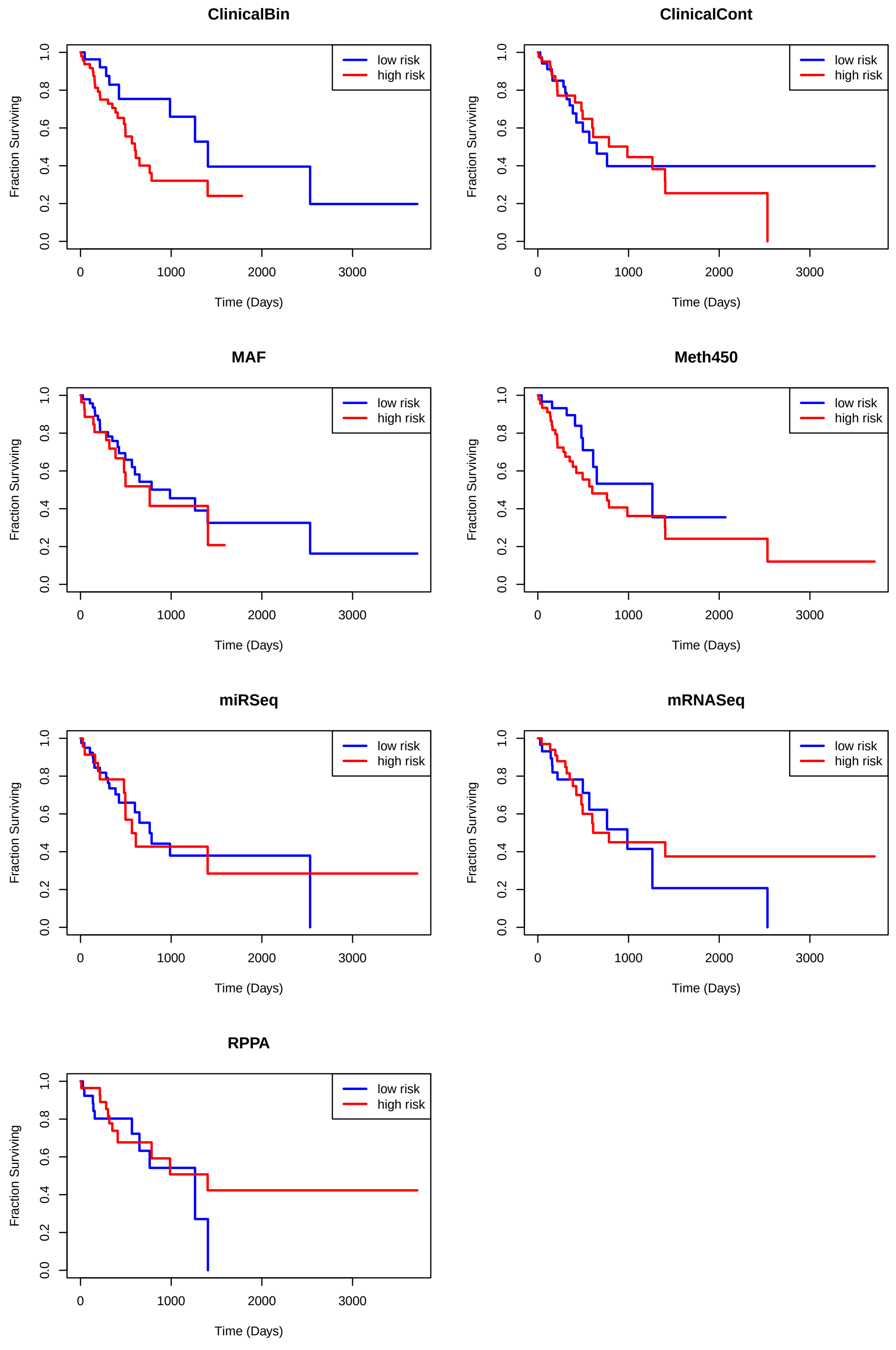


Figure 5: Kaplan-Meier plots of overall survival on the test set from separate PLS Cox omics models

```
dataoptions <- get_default_data_options(mofaObject)
modeloptions <- MOFA2::get_default_model_options(mofaObject)
modeloptions$num_factors <- NF <- 10
modeloptions[["likelihoods"]][["ClinicalBin"]] <- "bernoulli"
modeloptions[["likelihoods"]][["MAF"]] <- "bernoulli"
trainingoptions <- get_default_training_options(mofaObject)
suppressMessages( suppressWarnings(
  mofaObject <- prepare_mofa(mofaObject, data_options = dataoptions,
                             model_options = modeloptions,
                             training_options = trainingoptions)
  )
)
```

Now we can run the algorithm.

```
suppressMessages( suppressWarnings(
  mofaESCA <- run_mofa(mofaObject, use_basilisk = TRUE)
  )
)
```

We want to see whether any of the latent factors are associated with overall survival.

```
LF <- get_factors(mofaESCA, groups = "all", factors = "all") [[1]]
LFwithOutcome <- data.frame(LF, trainD@outcome[, c("Days", "vital_status")])
osmodel <- coxph(Surv(Days, vital_status == "dead") ~ ., data = LFwithOutcome)
AIC <- step(osmodel, trace = 0)
AIC

## Call:
## coxph(formula = Surv(Days, vital_status == "dead") ~ Factor2 +
##     Factor3 + Factor4, data = LFwithOutcome)
##
##              coef exp(coef) se(coef)      z      p
## Factor2 -0.13573   0.87308  0.06024 -2.253 0.0242
## Factor3 -0.16811   0.84526  0.10180 -1.651 0.0987
## Factor4  0.15736   1.17041  0.09634  1.633 0.1024
##
## Likelihood ratio test=7.63  on 3 df, p=0.0544
## n= 108, number of events= 42
```

Factors 2, 3, and 4 are associated with survival, although the model itself is, at best, only borderline significant.

```
mofarisk <- survival:::predict.coxph(AIC)
traindata <- data.frame(trainD@outcome, mofarisk)
traindata$Cut <- mofarisk > 0
mofamodel <- coxph(Surv(Days, vital_status == "dead") ~ Cut, data = traindata)
Strain <- summary(mofamodel)
```

The hard part is to reverse-engineer the MOFA model so we can use its decompositions to make predictions on the test data.

```
weightmats <- get_weights(mofaESCA)
Woo <- do.call(rbind, weightmats)
testMats <- testD@data
testShift <- lapply(testMats, function(X) {
  if (length(unique(as.vector(X))) < 10) {
    X
  } else {
    t(scale(t(X), scale = FALSE))
  }
})
Yoo <- do.call(rbind, testShift)
Voo <- t(qr.solve(Woo, Yoo))
punk <- survival:::predict.coxph(AIC, as.data.frame(Voo))
testdata <- data.frame(testD@outcome, punk)
testdata$Cut <- punk > 0
testmodel <- coxph(Surv(Days, vital_status == "dead") ~ Cut, data = testdata)
Stest <- summary(testmodel)

col2 <- c("blue", "red")
opar <- par(mfrow=c(2,1))
plot(survfit(Surv(Days, vital_status == "dead") ~ Cut, data = traindata),
     col = col2, lwd = 3, main = "MOFA model, training data", xlab = "Time (Days)",
     sub = paste("Score test p-value = ", round(Strain$sctest[3], 3)))
legend("topright", c("Low Risk", "High Risk"), col = col2, lwd = 2)
mtext("A", side = 2, at = 1.2, line = 3, cex = 1.5, las = 2)
plot(survfit(Surv(Days, vital_status == "dead") ~ Cut, data = testdata),
     xlab = "Time (Days)",      col = col2, lwd = 3, main = "MOFA model, test data",
     sub = round(Stest$sctest[3], 3))
legend("topright", c("Low Risk", "High Risk"), col = col2, lwd = 2)
mtext("B", side = 2, at = 1.2, line = 3, cex = 1.5, las = 2)

par(opar)
rm(opar)
```

Interpretation

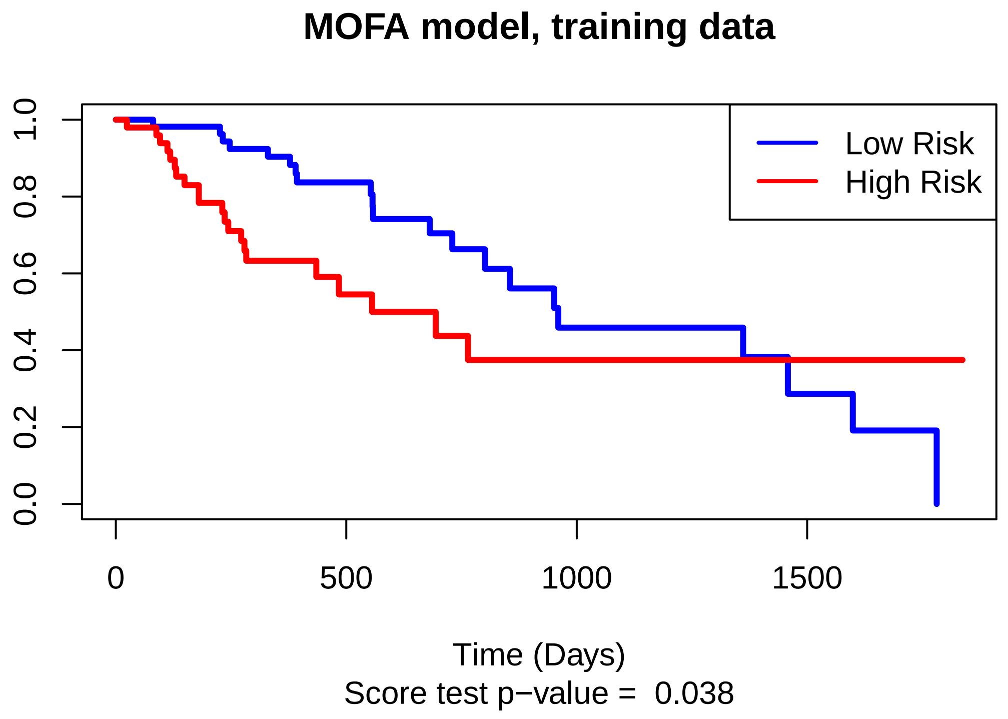
The `plasma` package includes several tools to assist with the interpretation of components and of the features contributing to the final model. These are illustrated in the "interpretation" vignette

Conclusions

We have described a method analogous to that of `MOFA` that allows us to combine different omics data sets without the need for massive prior imputation of missing values. A major difference is that while the `MOFA` model learns "factors" that are composites of the variables in an unsupervised fashion, the `plasma` model learns "components" that are composites of the variables in a supervised fashion, using the outcomes "event" and "time-to-event" as response variables.

Although the factors from `MOFA` are defined such that the first factor, Factor 1, accounts for the greatest variance in the model, the factors may or may not be significantly associated with the outcome, and a post-hoc survival analysis would need to be done to assess this. It may be the case that some factors, although they are significantly associated with outcome, account for very small variance in the final `MOFA` model, which hinders interpretability. This was the case with the TCGA-ESCA data set, in which, when 10 factors were learned from the `MOFA` model, only three factoes were significantly associated with survival, while not generalizing to the test data set. Bu contrast, the components for `plasma` are created in a way that maximizes the covariance in the predictors and the response, and therefore these components will be automatically associated to some degree with the outcome. This observation could be advantageous in that dissecting the weights associated with the components would yield a list of variables from different omics data sets that contribute the most to defining the outcome, and any additional analyses could be refined by looking at these high-weighted variables most closely.

A



B

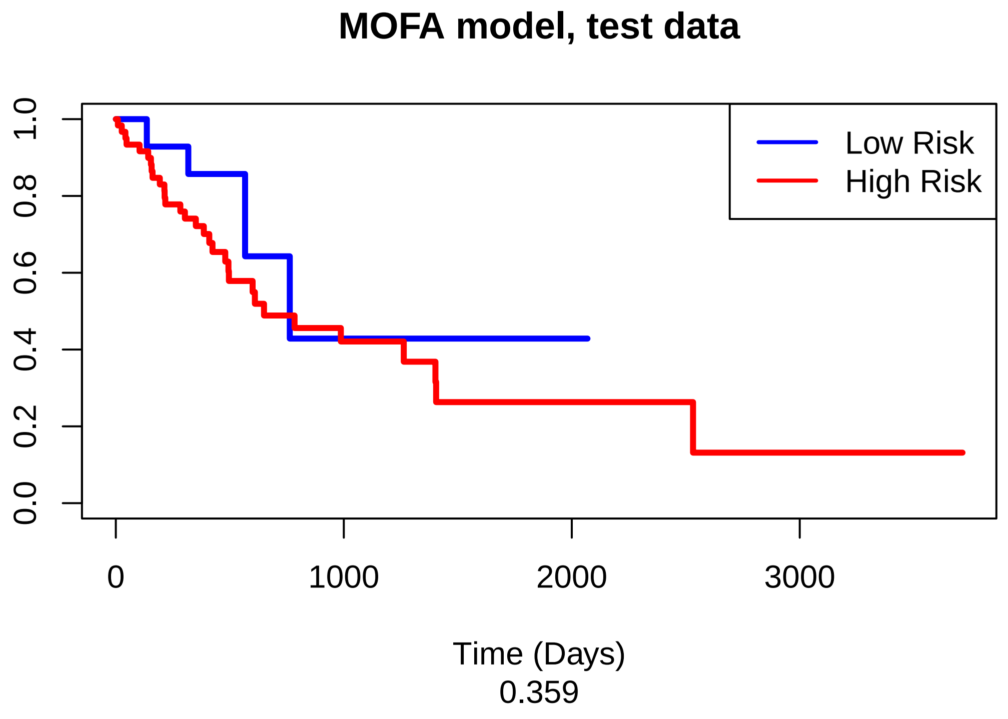


Figure 6: Kaplan-Meier plot of overall survival on (A) the training set and (B) the test set using the `MOFA` Cox model.

We were surprised to find that most of the models from the individual data sets did not generalize well to the test data. Nevertheless, the joint model that combined the components from those data sets had good performance in predicting the risk of patient in the test data set. It is unclear why this happens. We speculate that the phenomenon may be related to the more genral idea of "bagging" weak predictors to create a more effective composite model (Boulesteix and Hothorn, 2010; Abdia *et al.*, 2017; Hillebrand *et al.*, 2021; Poudel *et al.*, 2022). MOre research will be required to determine how general this phenomenon is.

References

Abdia,Y. *et al.* (2017) Propensity scores based methods for estimating average treatment effect and average treatment effect among treated: A comparative study. *Biom J*, **59**, 967–985.

Adossa,N. *et al.* (2021) Computational strategies for single-cell multi-omics integration. *Comput Struct Biotechnol J*, **19**, 2588–2596.

Argelaguet,R. *et al.* (2020) MOFA+: A statistical framework for comprehensive integration of multi-modal single-cell data. *Genome Biol*, **21**, 111.

Argelaguet,R. *et al.* (2018) Multi-omics factor analysis-a framework for unsupervised integration of multi-omics data sets. *Mol Syst Biol*, **14**, e8124.

Bastien,P. *et al.* (2015) Deviance residuals-based sparse PLS and sparse kernel PLS regression for censored data. *Bioinformatics*, **31**, 397–404.

Bertrand,F. and Maumy-Bertrand,M. (2021) Fitting and cross-validating cox models to censored big data with missing values using extensions of partial least squares regression models. *Front Big Data*, **4**, 684794.

Boulesteix,A.L. and Hothorn,T. (2010) Testing the additional predictive value of high-dimensional molecular data. *BMC Bioinformatics*, **11**, 78.

Cancer Genome Atlas Research Network (2017) Integrated genomic characterization of oesophageal carcinoma. *Nature*, **541**, 169–175.

Chen,J. *et al.* (2009) ToppGene suite for gene list enrichment analysis and candidate gene prioritization. *Nucleic Acids Res*, **37**, W305–11.

Coombes,K.R. *et al.* (2019) Polychrome: Creating and assessing qualitative palettes with many colors. *Journal of Statistical Software*, **90**, 1–23.

Deng,M. *et al.* (2017) FirebrowseR: An r client to the broad institute's firehose pipeline. *Database (Oxford)*, **2017**.

Graw,S. *et al.* (2021) Multi-omics data integration considerations and study design for biological systems and disease. *Mol Omics*, **17**, 170–185.

Heo,Y.J. *et al.* (2021) Integrative multi-omics approaches in cancer research: From biological networks to clinical subtypes. *Mol Cells*, **44**, 433–443.

Hillebrand,E. *et al.* (2021) Bagging weak predictors. *Int. J. Forecasting*, **37**, 237–254.

Kampstra,P. (2008) Beanplot: A boxplot alternative for visual comparison of distributions. *Journal of Statistical Software*, **28**, 1–9.

Mishra,P. and Liland,K.H. (2022) Swiss knife partial least squares (SKPLS): One tool for modelling single block, multiblock, multiway, multiway multiblock including multi-responses and meta information under the ROSA framework. *Anal Chim Acta*, **1206**, 339786.

Picard,M. *et al.* (2021) Integration strategies of multi-omics data for machine learning analysis. *Comput Struct Biotechnol J*, **19**, 3735–3746.

Poudel,G.R. *et al.* (2022) Machine learning for prediction of cognitive health in adults using sociodemographic, neighbourhood environmental, and lifestyle factors. *Int J Environ Res Public Health*, **19**.

R Core Team (2022) R: A language and environment for statistical computing R Foundation for Statistical Computing, Vienna, Austria.

Reel,P.S. *et al.* (2021) Using machine learning approaches for multi-omics data analysis: A review. *Biotechnol Adv*, **49**, 107739.

Simon,R.M. and Dobbin,K. (2003) Experimental design of DNA microarray experiments. *Biotechniques*, **Suppl**, 16–21.

Subramanian,I. *et al.* (2020) Multi-omics data integration, interpretation, and its application. *Bioinform Biol Insights*, **14**, 1177932219899051.

Vlachavas,E.I. *et al.* (2021) A detailed catalogue of multi-omics methodologies for identification of putative biomarkers and causal molecular networks in translational cancer research. *Int J Mol Sci*, **22**.