sprinter: An R package for screening prognostic interactions

Isabell Hoffmann, Murat Sariyar, Harald Binder

November 20, 2014

Contents

1 Introduction

In order to predict patients' mortality risks, it is of special interest to determine predictive biomarkers. Multivariate risk prediction models are constructed to predict such mortality risks. In order to generate sparse models for molecular data, established methods can be used, that are able to extract important main effects (such as elastic net [1], LASSO [2] and CoxBoost [3]).

Some effects arise only if two genes interact, e.g. expression of one gene depends on the expression level of the other gene. Such interactions may play an important role in molecular applications.

Considering all two-way interactions increases the problem of multiple testing enormously. Therefore, it is meaningful to pre-select possibly relevant interactions to reduce the number of interactions. Machine learning approaches such as penalized regression models [4], logic regression [5], multifactor-dimensionality reduction [6] or random forest [7] are able to pre-select relevant interaction terms in high-dimensional datasets.

The package *sprinter* offers a modular framework for pre-selecting important interactions and building prognostic models for time-to-event settings [8] by combining available statistical components. The framework consists of the following three steps (for a detailed explanation, see below):

(1) Fitting a main effects model

(2) Modifying the data and pre-selecting interaction terms

(3) Building a comprehensive considering interactions

In addition to the model building step, our package provide a method to evaluate the relevance of selected interactions within the final model.

In the first step of our strategy, a main effects model is fitted in oder to determine the relevant main effects. These main effects will be used in the second step to generate a modified data set. With this modified data set, weaker interaction effects are no longer masked by the stronger main effects. After modifying the data, possibly relevant interaction terms are pre-selected. In the last step, the final model is fitted by using the main effects detected in step (1) and the interaction terms detected in step (2) as covariates.

The package `sprinter` offers a new approach to interaction detection and has the following achievements:

1. The modular structure of this approach leads to the simultaneous consideration of main effects and interactions in high-dimensional data sets.

2. It is possible to compare the main effects extracted by the main effects model with those selected by the final model in order to assess the relevance of the main effects.

3. It is a flexible approach in which different statistical methods can be combined in order to perform the different steps.

2 Data sets

To demonstrate how the package works, simulated data sets are provided for which the true effects are known. The function `simul.int` creates exponentially distributed survival times with baseline hazard `lambda`. The number of covariates is `p` and the sample size is `n`. All covariates are standard normally distributed. The number of true main effects and the number of true interactions are given by `n.main` and `n.int`. The interactions are generated by multiplications of non-main-effects variables. The absolute effect sizes of the main effects is assigned to `beta.main`, and the absolute effect sizes of the true interactions are assigned to `beta.int`.

We use the following settings for simulating the data set:

```
library(sprinter)
simulation <- simul.int(seed = 12345,n = 200, p = 500,
                         beta.int = 1,
                         beta.main = 0.9,
                         censparam = 1/20,
                         lambda = 1/20)
data <- simulation$data
```

The function simul.int returns two components: the simulated data set (data) and information about the main effects and interactions and their effect sizes (info). The information about the true main effects and interactions is printed as follows:

```
print(simulation$info)

##          ID Effect size
## 1      ID1         0.9
## 2      ID2        -0.9
## 3 ID3:ID4           1
## 4 ID5:ID6          -1
```

Figure 1 shows the Kaplan-Meier curve of the simulated data set. The question we are going to address is the following: Is it possible to find true interactions in such a data set together with true main effects?

3 How to use sprinter

3.1 Using existing functions

sprinter provides a prognostic model after pre-selecting relevant interactions and main effects. Methods for pre-selecting the important variables can be set in the arguments screen.main and screen.inter.

```
sprinter(x,
         time,
         status,
         mandatory= NULL,
         repetitions = 25,
         n.inter.candidates =1000,
         screen.main,
         screen.inter = fit.rf,
         fit.final = screen.main,
         args.screen.main = list(),
         args.screen.inter = list(),
         args.fit.final = args.screen.main,
         orthogonalize = TRUE,
         parallel = FALSE, mc.cores = detectCores(), ...)
```

To pre-select the important main effects, it is possible to perform CoxBoost or univariate Cox regressions; the parameter screen.main accepts a value in the form of a function to indicate the desired approach. Using the function fit.CoxBoost means to select the main effects which are chosen by CoxBoost and using fit.uniCox means to select variables with adjusted p-values [9] smaller than sig. Functions fit.CoxBoost

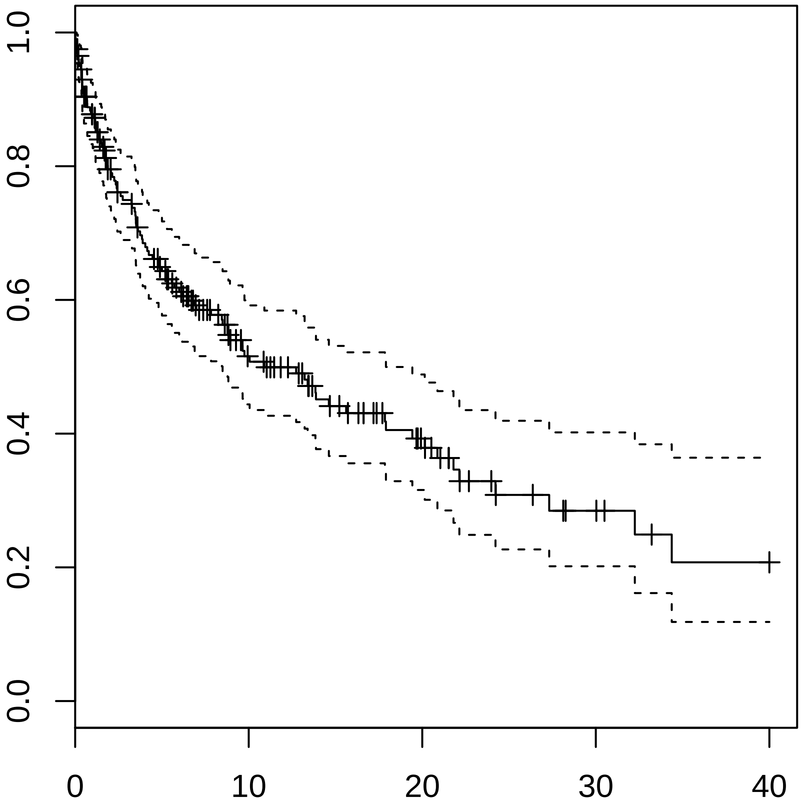


Figure 1: Kaplan-Meier simulated data set.

and `fit.uniCox` are adapted function for the usage in `sprinter` (see more information in Section 3.3). The arguments for the methods used for screening the main effects can be set in `args.screen.main`.

Correspondently, the approach for screening interactions is assigned to `screen.inter`. It is possible to perform this step by using random forest (`screen.inter = fit.rf`) or by using logic regression (`screen.inter = fit.logicReg`). For more information about the structure of the adapted functions `fit.rf` and `fit.logicReg`, see Section 3.3. By using random forest for each variable, a variable importance measurement is calculated that considers the underlying interaction structure and reflects the meaning of a variable for the forest. By default the permutation accuracy importance [10] is used for evaluating the variables in the forest. This measure can be replaced by other importance measures such as minimal depth. The variable importance is used to construct the relevant interactions for the model. The random forest arguments can be set in `args.screen.inter`.

In very large data sets the number of variables must be restricted in order to preselect relevant interactions. To be able to perform random forest or logic regression in such data sets, use `fit.rf.select`, respectively `fit.logicReg.select`. These functions restrict the data set by selecting the variables with the `n.select` smallest p-values evaluated by univariate Cox regressions before performing random forest, respectively the logic regression.

Before pre-selecting the interactions, the data are modified such that weaker interactions can be detected more easily (orthogonalize = TRUE). All variables are orthogonalized to those that are assessed as main effects in the first step.

For better stabilization subsamples are created and random forests are performed on each subsampled data set. The number of subsamples can be set in `repetitions` (default value: 25). To summarize the results of all subsamples, variable inclusion frequencies (VIFs) of the constructed interactions terms are computed and the `n.inter.candidates` most frequent pairs are selected as relevant interaction terms. As the pre-selection of interactions can be computationally expensive it is possible to parallelize this step by `parallel = TRUE`.

Further approaches for screening main effects and interactions can be implemented by the user, see Section 3.3.

As an example we perform `sprinter` to screen main effects by CoxBoost and to detect interactions by random forest:

```
set.seed(518)
testcb <- sprinter( x=data[,1:500],
                    time = data$obs.time,
                    status= data$obs.status,
                    repetitions = 25,
                    mandatory = c("ID1","ID2"),
                    n.inter.candidates = 10000,
                    screen.main = fit.CoxBoost,
                    screen.inter = fit.rf,
                    fit.final = fit.CoxBoost,
                    #args.screen.main = list(seed=123, stepno = 10,
                    #                        K = 10, criterion ='pscore',
                    #                        nu = 0.05),
                    parallel = TRUE)
```

Clinical covariates can be specified in the argument `mandatory` as a vector to be forcefully included in a model. In this example we set the main effects "ID1" and "ID2" to mandatory; these are the colnames of the mandatory covariates in the data set `data`. To build the final model, the user can choose between CoxBoost and univariate Cox-regression, as in the main effects model building step. In contrast to building the main effects model, the final model is constructed using the variables selected in the main effects model plus the `n.inter.candidates` pre-selected interactions of the screening step. In this example we choose CoxBoost for fitting the final model. As an example for using `args.screen.main`, a comment is included on how to set arguments for the CoxBoost approach. For more information about using CoxBoost see [11] .

As a result, `sprinter` prints the candidates for interactions with the largest inclusion frequencies together with the final model.

```
print(testcb)

## Top 20 Interaction Candidates with Inclusion Frequencies:
##      ID2:ID1    ID19:ID1   ID329:ID1     ID4:ID1    ID19:ID2   ID329:ID2
##          24          23          23          22          22          22
##   ID329:ID4    ID54:ID1    ID73:ID1     ID4:ID2  ID329:ID19   ID144:ID1
##          22          21          21          21          21          20
##   ID303:ID1    ID54:ID2    ID73:ID2    ID19:ID4   ID303:ID4   ID54:ID19
##          20          20          20          20          20          20
##   ID73:ID19 ID329:ID303
##          20          20
##
##  Final model:
##
## 35 boosting steps resulting in 11 non-zero coefficients (with 2
## being mandatory)
## partial log-likelihood: -403.6513
```

In this example the final model consists of eleven variables. To obtain more information about the variables selected in the models call `summary()`, as in the following command:

```
summary(testcb)

## Main candidates with coefficients:
##         ID1         ID2         ID4        ID12        ID37
##  0.99687801 -0.70730288  0.01396648  0.02875253 -0.15896531
```

```
##
##  Top 20 Interaction Candidates with Inclusion Frequencies:
##     ID2:ID1    ID19:ID1   ID329:ID1     ID4:ID1    ID19:ID2   ID329:ID2
##          24          23          23          22          22          22
##   ID329:ID4    ID54:ID1    ID73:ID1     ID4:ID2  ID329:ID19   ID144:ID1
##          22          21          21          21          21          20
##   ID303:ID1    ID54:ID2    ID73:ID2    ID19:ID4   ID303:ID4   ID54:ID19
##          20          20          20          20          20          20
##   ID73:ID19 ID329:ID303
##          20          20
##
##  Final Model:
## 35 boosting steps resulting in 11 non-zero coefficients (with 2 being
## mandatory)
## partial log-likelihood: -403.6513
##
## Parameter estimates for mandatory covariates at boosting step 35:
##     Estimate
## ID1   0.9792
## ID2  -0.7772
##
## Optional covariates with non-zero coefficients at boosting step 35:
## parameter estimate > 0:
##  ID1, ID3:ID4, ID4:ID373, ID18:ID321, ID267:ID319
## parameter estimate < 0:
##  ID2, ID37, ID39:ID358, ID214:ID412, ID230:ID319, ID401:ID498
```

The summary of sprinter shows the main candidates with their coefficients, the interaction candidates with the largest inclusion frequencies and gives information about the final model. In this example the true interaction ID3:ID4 is selected in the final model with the correct sign for its parameter estimate. The second interaction ID5:ID6 is not included in this model; instead, seven interactions and one main effect are selected which have no effect on the outcome.

For real data sets the true main effects and interactions are unknown. In order to distinguish between true and false positive interactions, a subsampling step using resample.sprinter, which is a wrapper, is performed. See Section 3.2.

If the user wants to perform predictions with the final model, this package provides the S3 method predict, which can be applied to objects of class sprinter.

3.2 Evaluating interactions

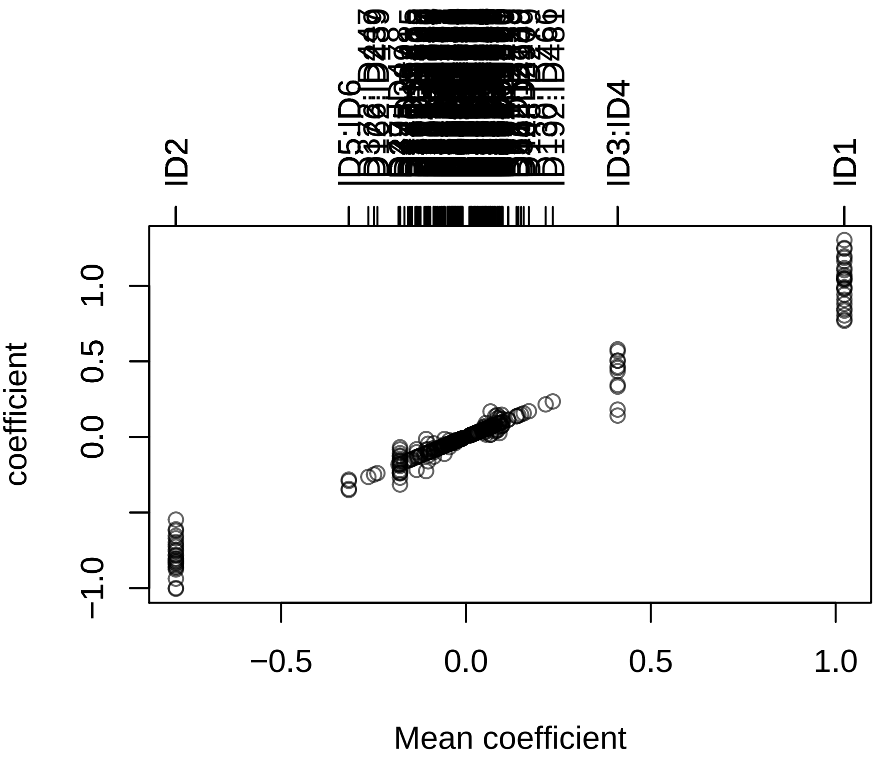
The sprinter package provides the possibility to evaluate the relevance of an interaction by using resampling techniques [12] and the resultant variable inclusion frequency (VIF). The implemented wrapper resample.sprinter subsamples the original data set fold-times and applies the whole procedure to each subsample. The proportion of samples that should be drawn from the original data set can be set in the argument `oob.rel` (default value: 0.632). The results of this wrapper are inclusion frequencies and mean coefficients of each interaction term.

As an example we perform the function `resample.sprinter` on the data set we simulated in Section 2. For the purpose of displaying the results, we set the number of subsamples to 25 with a subsampling frequency of 0.632. Although 20 cores are used for running this operation, it needs about 15 minutes (`system.time`) to run. If the user intends to test the function, it is possible reduce the system time by reducing the values for `fold` and `repetitions`. On each subsample, the function `sprinter` is applied as in the previous example (see Section 3.1).

```
set.seed(123)
resamcb <- resample.sprinter(  x=data[,1:500],
                               time = data$obs.time,
                               status= data$obs.status,
                               fold = 25,
                               oob.rel = 0.632,
                               repetitions = 25,
                               mandatory = c("ID1","ID2"),
                               n.inter.candidates = 1000,
                               screen.main = fit.CoxBoost,
                               screen.inter = fit.rf,
                               fit.final = fit.CoxBoost,
                               parallel = T,mc.cores = 20)
```

For showing the results the summary() provides a plot of the mean coefficients of each selected variable with their single coefficients and displays the mean coefficients of the interactions with variable inclusion frequencies larger than 1/`fold`.

```
summary(resamcb)
```

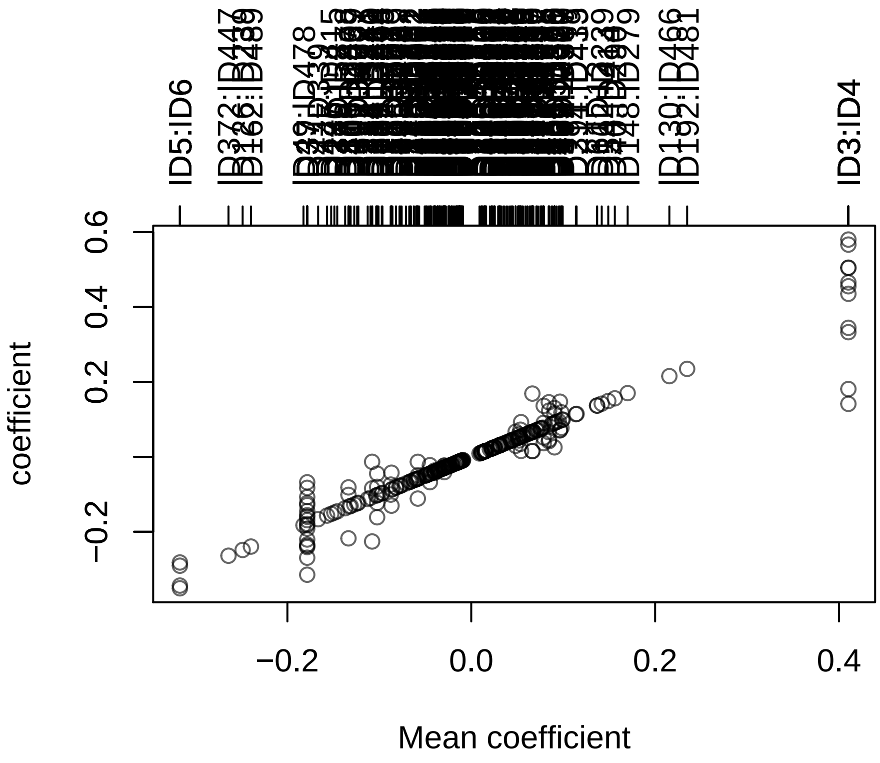


```
##              Variable Inclusion Frequency Mean Coefficient
## ID3:ID4                             0.44       0.41023042
## ID5:ID6                             0.16      -0.31689659
## ID1:ID330                           0.12      -0.08676463
## ID208:ID390                         0.12       0.09053558
## ID4:ID358                           0.12      -0.02938755
## ID1:ID300                           0.12      -0.05811525
## ID214:ID412                         0.12      -0.10781557
## ID1:ID78                            0.08       0.05196649
## ID71:ID361                          0.08      -0.08750961
## ID101:ID440                         0.08       0.05351890
## ID290:ID370                         0.08      -0.03989412
## ID2:ID10                            0.08       0.09813999
## ID394:ID447                         0.08      -0.04496056
```

The plot generated by `summary()` shows that the clinical covariates ID1 and ID2 are the strongest main effects (on average 1.0234 and -0.7845). There are many false positive main effects and interactions in the final model which are chosen only once. To distinguish between the true and the false positive interactions, the user can consult the variable inclusion frequencies in the provided table. This table shows the variable inclusion frequencies and the mean coefficients of the variables selected more often than once. The true interactions have the highest variable inclusion frequencies of 44 percent and 16 percent. All other interactions have inclusion frequencies of 12 percent and lower.

For plotting the coefficients of the optional variables without the mandatory ones, the user can set the argument `optional = TRUE`.

```
plot(resamcb, optional.only = T)
```



3.3 Implementing new functions for pre-selecting covariates.

In the current version of the package some methods are available to pre-select main effects or to build the final model. It is easy to add further methods by implementing new functions. By writing a new function for pre-selecting main effects ,the user should at least enclose the following arguments for committing the data: `time`, `status`, `x` and `unpen.index`.

As an example the code of the function `fit.uniCox` is printed as follows:

```
fit.uniCox

## function (time, status, x, unpen.index = NULL, method = "bonferroni",
##     sig = 0.05, ...)
## {
##     pvalue <- beta <- rep(NA, ncol(x))
##     for (i in 1:ncol(x)) {
##         res <- coxph(Surv(time, status) ~ x[, i])
##         pvalue[i] <- summary(res)$coefficients[, 5]
##         beta[i] <- summary(res)$coefficients[, 1]
##     }
##     pvalueadjust <- p.adjust(p = pvalue, method = method)
##     indmain <- unique(c(which(pvalueadjust < sig), unpen.index))
##     datamulti <- as.data.frame(x[, indmain])
##     cox <- coxph(Surv(time, status) ~ ., data = datamulti)
##     res <- list()
##     res$model <- cox
##     res$xnames <- colnames(x)[indmain]
##     res$indmain <- indmain
##     res$beta <- coefficients(cox)
##     return(res)
## }
## <environment: namespace:sprinter>
```

In the first part of the function the main effects are identified using univariate Cox regressions. All variables with univariate adjusted *p*-values smaller than the significance level `sig` are used for fitting the multivariate model. In the second part of the function the multivariate Cox regression is fitted.

The resultant object returns a list comprising the following components:

`model` Cox proportional hazards model

`xnames` Names of the selected covariates

`indmain` Vector containing their indices

`beta` Vector of coefficients

If the user wants to implement a new function for pre-selecting interactions, the function has to enclose the following arguments:

nr: Value for displaying the actual resampling run.
data: Data frame containing the y-outcome and x-variables in the model.
indices: Indices indicating the samples of the subsample.
seed.interselect: Seed for random number generator.

As an example the code of the function `fit.rf` is printed as follows:

```
fit.rf

## function (nr, data, indices, seed.interselect, ...)
## {
##     cat(nr)
##     rsf <- rfsrc(Surv(time, status) ~ ., data = data[indices,
##         ], big.data = T, seed = seed.interselect)
##     return(rsf$importance)
## }
## <environment: namespace:sprinter>
```

As a result, the function must provide a vector containing a variable importance measure for each variable in the data. This measure must be interpretable in such a way that a positive value indicates that the investigated variable is relevant.

References

[1] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, pp. 301–320, Apr. 2005.

[2] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society (Series B)*, vol. 58, pp. 267–288, 1996.

[3] H. Binder and M. Schumacher, "Incorporating pathway information into boosting estimation of high-dimensional risk prediction models," *BMC Bioinformatics*, vol. 10, no. 1, p. 18, 2009.

[4] M. Y. Park and T. Hastie, "Penalized logistic regression for detecting gene interactions," *Biostatistics*, vol. 9, no. 1, pp. 30–50, 2008.

[5] H. Schwender and K. Ickstadt, "Identification of snp interactions using logic regression," *Biostatistics*, vol. 2008, pp. 9–187, 2007.

[6] L. W. Hahn, M. D. Ritchie, and J. H. Moore, "Multifactor dimensionality reduction software for detecting gene-gene and gene-environment interactions," *Bioinformatics*, vol. 19, no. 3, pp. 376–382, 2003.

[7] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[8] M. Sariyar, I. Hoffmann, and H. Binder, "Combining techniques for screening and evaluating interaction terms on high-dimensional time-to-event data.," *BMC bioinformatics*, vol. 15, p. 58, 2014.

[9] S. Wright, "Adjusted p-values for simultaneous inference," *Biometrics*, vol. 48, no. 4, pp. 1005–1013, 1992.

[10] H. Ishwaran and et al, "High-Dimensional Variable Selection for Survival Data," *JASA*, vol. 105, no. 489, pp. 205–217, 2010.

[11] P. Buhlmann and T. Hothorn, "Boosting Algorithms: Regularization, Prediction and Model Fitting," *Statistical Science*, vol. 22, no. 4, pp. 477–505, 2007.

[12] W. Sauerbrei and et al, "Stability investigations of multivariable regression models derived from low- and high-dimensional data.," *J Biopharm Stat*, vol. 21, no. 6, pp. 1206–31, 2011.