Protogarrote: R software to fit prediction models with or without interactions using high-troughput proteomic biomarkers

Georg Heinze

Vignette, 27 11 2020

## Introduction

This document introduces the protogarrote functions to analyse proteomics data and explains their use. The following functions are available:

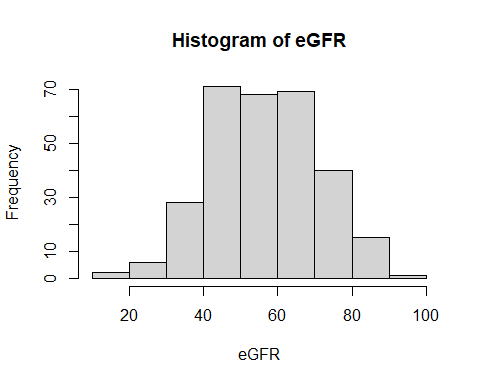
* protogarrote … fits a linear or logistic model with proteomics and clinical predictor variable. May include an interaction of a variable with the proteomics. Uses internal validation to determine the best values of lambda1 and lambda2.
* coefficients.protogarrote … shows the estimated coefficients at the optimal lambda1, lambda2
* predict.protogarrote … predicts outcomes using a protogarrote object and new data
* plot.coefficients.protogarrote … shows the estimated coefficients for the proteomics by means of a plot and a table.

## Data set

To illustrate the use, we first generate a data set synthetically:

source("code.R")  
  
# first a data set  
set.seed(123)  
  
n<-300 ### sample size  
k<-100 ### number of proteomics (filtered)  
  
prop.nonzero<-runif(k,0.1,1) # proportion nonzero for each peptide  
prop.nonzero<-sort(prop.nonzero)  
means.peptide<-4+prop.nonzero\*10 # we create consonant peptides only  
  
peptide <- matrix(rnorm(n\*k), n, k, byrow=TRUE)+matrix(means.peptide, n, k, byrow=TRUE)  
  
#plot(1:k, apply(peptide, 2, mean))  
  
peptide <- peptide \* rbinom(n\*k, size=1, prob=matrix(prop.nonzero, n, k, byrow=TRUE))  
# boxplot(peptide) # only for small k!

# clinical variables  
age <- floor(runif(n, 20, 81))  
sex <- rbinom(n, size=1, prob=0.5)  
  
# outcomes  
  
# eGFR model: define 10 random peptides, each has a delta of 1 and a beta of 1; others not relevant  
predictive <- sample(1:k, size=10)  
  
  
eGFR <- apply(peptide[,predictive],1,sum) + apply((peptide[,predictive]>1),1,sum) + age/10 + sex + rnorm(n, mean=-20, sd=6)  
eGFR[eGFR<15] <- 15  
hist(eGFR)



Next, we generate x and d components of the peptides. Crucial to save the means of the x’s for later application of a model!

x <- peptide  
d <- (peptide != 0)\*1  
xmeans <- apply(x, 2, FUN=function(x) mean(x[x!=0]))  
for(j in 1:k) x[,j][d[,j]==0] <- xmeans[j]

Now we glue everything into a data frame:

dataset <- data.frame(log2eGFR = log2(eGFR), age=age, sex=sex, x=x, d=d)

We fit the ‘oracle’ model: as if we knew (from an oracle) which peptides are truly prognostic. This model serves as a benchmark; it will be hard to outperform it:

# true model on data set  
## true formula  
formel <- as.formula(paste("eGFR ~ age + sex", paste0(" + x.", predictive, collapse=""), paste0(" + d.", predictive, collapse="")))  
  
fit.oracle <- lm(data=dataset, formula=formel)  
summary(fit.oracle)

##   
## Call:  
## lm(formula = formel, data = dataset)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -14.9448 -3.9167 -0.0664 3.9732 16.0546   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -105.72588 15.62675 -6.766 7.88e-11 \*\*\*  
## age 0.10133 0.02045 4.955 1.26e-06 \*\*\*  
## sex 1.08519 0.72240 1.502 0.13418   
## x.24 1.08780 0.61302 1.775 0.07708 .   
## x.81 0.71021 0.39328 1.806 0.07202 .   
## x.77 0.68563 0.40480 1.694 0.09144 .   
## x.86 1.08525 0.37519 2.893 0.00412 \*\*   
## x.49 1.36016 0.54255 2.507 0.01275 \*   
## x.23 1.08055 0.60816 1.777 0.07671 .   
## x.98 0.91548 0.36343 2.519 0.01233 \*   
## x.42 0.84032 0.52876 1.589 0.11315   
## x.54 0.56095 0.50613 1.108 0.26869   
## x.25 0.81597 0.65645 1.243 0.21492   
## d.24 6.49779 0.79875 8.135 1.40e-14 \*\*\*  
## d.81 11.48418 1.00245 11.456 < 2e-16 \*\*\*  
## d.77 13.62734 0.94478 14.424 < 2e-16 \*\*\*  
## d.86 13.07685 1.23918 10.553 < 2e-16 \*\*\*  
## d.49 8.87933 0.72145 12.308 < 2e-16 \*\*\*  
## d.23 7.51458 0.77052 9.753 < 2e-16 \*\*\*  
## d.98 14.36575 1.60174 8.969 < 2e-16 \*\*\*  
## d.42 10.37609 0.72078 14.396 < 2e-16 \*\*\*  
## d.54 9.10655 0.73618 12.370 < 2e-16 \*\*\*  
## d.25 7.99634 0.78712 10.159 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 6.078 on 277 degrees of freedom  
## Multiple R-squared: 0.84, Adjusted R-squared: 0.8272   
## F-statistic: 66.08 on 22 and 277 DF, p-value: < 2.2e-16

Now we add a synthetic binary HF variable, which is generated by some peptides that interact with eGFR:

# we assume that these peptides are important to generate HF (together with eGFR)  
pred.hf <- sample(1:k, size=10)  
pred.hf

## [1] 31 100 71 88 39 5 22 13 45 87

For simplicity, we assume that their effect is as follows:

* for the first 5 predictive x: -0.1 + xlog2eGFR$$0.1 (negative for the second 5)
* for the first 5 predictive d: -0.1 + dlog2eGFR$$0.1 (negative for the second 5)
* for age: just 0.01

x.pred <- apply(x[,pred.hf[1:5]],1,sum) - apply(x[,pred.hf[6:10]],1,sum)  
d.pred <- apply(d[,pred.hf[1:5]],1,sum) - apply(d[,pred.hf[6:10]],1,sum)  
  
linpred.hf <- -x.pred\*0.1 + 0.1\*x.pred\*dataset$log2eGFR - 0.1\*d.pred + 0.1\*d.pred\*dataset$log2eGFR + dataset$age\*0.01  
linpred.hf <- linpred.hf-mean(linpred.hf) # we subtract the mean to get approx 50% HF  
phf <- plogis(linpred.hf)  
dataset$hf <- rbinom(length(phf), 1, phf)  
mean(dataset$hf)

## [1] 0.5066667

Let us fit a simple clinical model with just age, sex and log2eGFR:

fit.clin.hf<-glm(data=dataset, hf~age+sex+log2eGFR, family="binomial")  
summary(fit.clin.hf)

##   
## Call:  
## glm(formula = hf ~ age + sex + log2eGFR, family = "binomial",   
## data = dataset)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.5913 -1.1701 0.8182 1.1224 1.7831   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -6.011283 1.787628 -3.363 0.000772 \*\*\*  
## age -0.008530 0.006782 -1.258 0.208495   
## sex -0.121185 0.237554 -0.510 0.609956   
## log2eGFR 1.131443 0.315717 3.584 0.000339 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 415.83 on 299 degrees of freedom  
## Residual deviance: 401.22 on 296 degrees of freedom  
## AIC: 409.22  
##   
## Number of Fisher Scoring iterations: 4

## Linear model without interactions

Creating the data object:

the data object should consist of a list with

* x: the log2 peptide itensities, with zeros replaced by conditional means
* d: binary indicators if intensity > 0
* clinical: matrix of clinical covariates
* y: outcome variable

x <- as.matrix(dataset[, paste0("x.", 1:k,sep="")])  
d <- as.matrix(dataset[, paste0("d.", 1:k,sep="")])  
age <- dataset$age  
sex <- dataset$sex  
log2eGFR <- dataset$log2eGFR  
  
data.obj<-list(x=x, d=d, clinical=cbind(age, sex), y=log2eGFR)

Fitting the model with 11 values for lambda1 and 11 values for lambda2. The model will first generate the values for lambda1, and for each lambda1 generate a new sequence of lambda2 values. These lambda2 values are then stored in a matrix with one row for each lambda1 value. The rownames of the matrix are the values for lambda 1. The outer option just repeats the cross-validation to get variability out. The cv value determines how many ‘folds’ are generated in the cross-validation.

#fit.simple <- lm(data.obj$y~data.obj$x+data.obj$d+data.obj$clinical)  
fit <- protogarrote(data.obj, family="gaussian", cv=10, outer=2, nlambda=c(11,11))

## Loading required package: glmnet

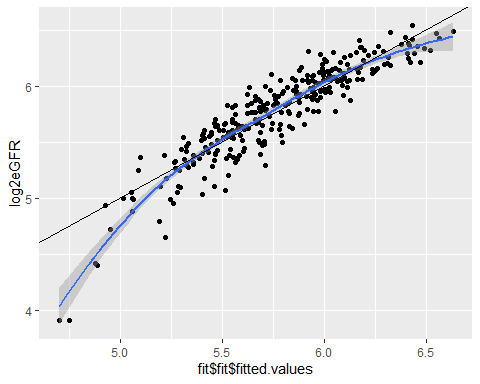
## Loading required package: Matrix

## Loaded glmnet 4.1

Let’s first look at the fitted values, and compare them to the original outcomes:

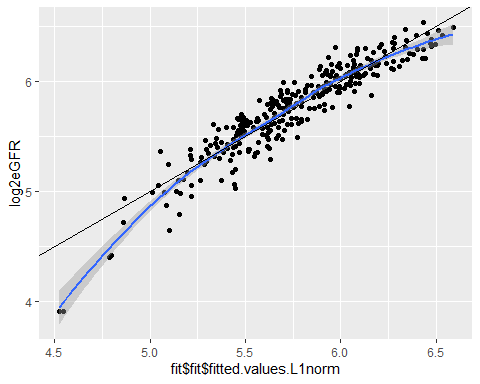
library(ggplot2)  
ggplot(NULL, aes(fit$fit$fitted.values, log2eGFR))+geom\_point()+geom\_smooth()+geom\_abline(intercept=0, slope=1)

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'



ggplot(NULL, aes(fit$fit$fitted.values.L1norm, log2eGFR))+geom\_point()+geom\_smooth()+geom\_abline(intercept=0, slope=1)

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'



The apparent R-square:

cor(fit$fit$fitted.values, log2eGFR)^2

## [,1]  
## [1,] 0.8275913

cor(fit$fit$fitted.values.L1norm, log2eGFR)^2

## [,1]  
## [1,] 0.8734763

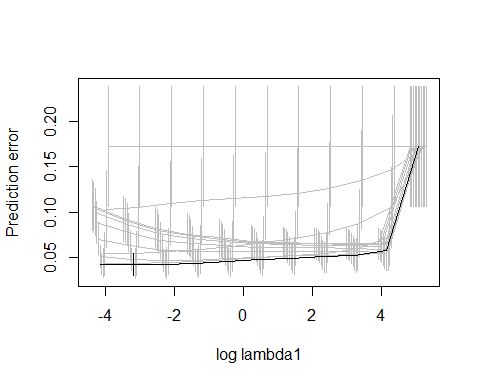
Compared to the cross-validated R-squares:

fit$cv.rsquare

## [,1] [,2] [,3] [,4] [,5] [,6] [,7]  
## [1,] 0 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000  
## [2,] 0 0.2546663 0.4956119 0.6179330 0.6593860 0.6696084 0.6593732  
## [3,] 0 0.2664780 0.5516140 0.6571400 0.6707056 0.6697397 0.6463577  
## [4,] 0 0.2784527 0.6002558 0.6780163 0.6757386 0.6714993 0.6349679  
## [5,] 0 0.2892464 0.6392410 0.6935266 0.6842809 0.6757277 0.6296763  
## [6,] 0 0.3009049 0.6760037 0.7114847 0.6987440 0.6790775 0.6279479  
## [7,] 0 0.3154946 0.7127639 0.7310867 0.7145942 0.6744310 0.5772904  
## [8,] 0 0.3330092 0.7417577 0.7453403 0.7206199 0.6561903 0.4846679  
## [9,] 0 0.3569366 0.7555330 0.7501029 0.7210841 0.6188733 0.4496912  
## [10,] 0 0.4095414 0.7549163 0.7475263 0.7130840 0.5848026 0.4278325  
## [11,] 0 0.3763159 0.7467066 0.7434118 0.7099507 0.5579269 0.3822983  
## [,8] [,9] [,10] [,11]  
## [1,] 0.0000000 0.0000000 0.0000000 0.0000000  
## [2,] 0.6331826 0.5948995 0.5613172 0.5430576  
## [3,] 0.6113021 0.5713134 0.5473461 0.5357687  
## [4,] 0.5963686 0.5583418 0.5366287 0.5261621  
## [5,] 0.5864473 0.5365916 0.5097487 0.4967895  
## [6,] 0.5547977 0.4964825 0.4660328 0.4514115  
## [7,] 0.4739457 0.4104164 0.3729078 0.3545200  
## [8,] 0.4041198 0.3516228 0.3146139 0.3010354  
## [9,] 0.3634109 0.2994321 0.2682997 0.2560778  
## [10,] 0.3175597 0.2601591 0.2305545 0.2176190  
## [11,] 0.2697963 0.2087808 0.1839814 0.1746336

One can ‘plot’ the object, meaning the cross-validated prediction error (expected squared difference between cross-validated fitted values and observed values of the outcome). Here each line correspond to a sequence of lambda1 values, and the different lambda2 values are the ‘inner’ loop. The vertical bars refer to standard errors of the prediction errors. The optimal combination of lambda1 and lambda2 is highlighted in black. One can also see ‘which lambda2’ was chosen:

plot.protogarrote(fit)



The estimated prediction errors can also be assessed as table (optimal value is in row 10, column 5):

fit$cv.pred.err

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] 0.1724006 0.1724006 0.17240061 0.17240061 0.17240061 0.17240061  
## [2,] 0.1724006 0.1471922 0.10497265 0.07227178 0.05766629 0.05426618  
## [3,] 0.1724006 0.1350098 0.08781719 0.06066026 0.05292971 0.05330072  
## [4,] 0.1724006 0.1259947 0.07730541 0.05499430 0.05092159 0.05296108  
## [5,] 0.1724006 0.1207035 0.07110848 0.05202214 0.04951274 0.05217492  
## [6,] 0.1724006 0.1174892 0.06660732 0.04973499 0.04782895 0.05059292  
## [7,] 0.1724006 0.1149030 0.06256689 0.04754528 0.04589855 0.04873410  
## [8,] 0.1724006 0.1123635 0.05907714 0.04570243 0.04411886 0.04736932  
## [9,] 0.1724006 0.1092568 0.05643955 0.04455063 0.04280229 0.04730014  
## [10,] 0.1724006 0.1057032 0.05483239 0.04402291 0.04239257 0.04844385  
## [11,] 0.1724006 0.1032290 0.05430105 0.04399938 0.04261861 0.05102640  
## [,7] [,8] [,9] [,10] [,11]  
## [1,] 0.17240061 0.17240061 0.17240061 0.17240061 0.17240061  
## [2,] 0.05584164 0.05895432 0.06200479 0.06410685 0.06524066  
## [3,] 0.05670050 0.06037967 0.06318270 0.06478803 0.06558822  
## [4,] 0.05727013 0.06105820 0.06368214 0.06511296 0.06582111  
## [5,] 0.05707986 0.06122621 0.06416210 0.06563350 0.06629659  
## [6,] 0.05654362 0.06172571 0.06512710 0.06683605 0.06762473  
## [7,] 0.05648196 0.06364916 0.06796603 0.07052416 0.07178994  
## [8,] 0.05693223 0.06512415 0.07087472 0.07421367 0.07571183  
## [9,] 0.05811193 0.06886215 0.07642399 0.08027414 0.08197076  
## [10,] 0.06301313 0.07774933 0.08675618 0.09104311 0.09290513  
## [11,] 0.07046208 0.08879119 0.10004507 0.10523267 0.10740501

The lambda values can be accessed. The row labels are the values of lambda1. The values in the matrix are the corresponding values of lambda2, which usually are very similar for a given lambda1, but control the number of selected variables.

fit$lambda

## [,1] [,2] [,3] [,4]  
## 157.936587830899 2.545873e-38 1.013530e-38 4.034937e-39 1.606337e-39  
## 62.8756872452973 1.642780e-04 6.540026e-05 2.603631e-05 1.036524e-05  
## 25.031261602291 4.061086e-04 1.616747e-04 6.436388e-05 2.562372e-05  
## 9.96512459510634 9.814235e-04 3.907117e-04 1.555451e-04 6.192363e-05  
## 3.96718750232328 2.255184e-03 8.978049e-04 3.574226e-04 1.422925e-04  
## 1.57936576992915 4.700665e-03 1.871368e-03 7.450051e-04 2.965919e-04  
## 0.62875682930618 8.424555e-03 3.353876e-03 1.335202e-03 5.315535e-04  
## 0.250312598845862 1.262189e-02 5.024866e-03 2.000435e-03 7.963875e-04  
## 0.0996512391127577 1.601307e-02 6.374918e-03 2.537900e-03 1.010356e-03  
## 0.0396718723008543 1.812075e-02 7.213998e-03 2.871944e-03 1.143342e-03  
## 0.0157936566154931 1.976623e-02 7.869080e-03 3.132737e-03 1.247165e-03  
## [,5] [,6] [,7] [,8]  
## 157.936587830899 6.394945e-40 2.545873e-40 1.013530e-40 4.034937e-41  
## 62.8756872452973 4.126477e-06 1.642780e-06 6.540026e-07 2.603631e-07  
## 25.031261602291 1.020099e-05 4.061086e-06 1.616747e-06 6.436387e-07  
## 9.96512459510634 2.465224e-05 9.814234e-06 3.907117e-06 1.555451e-06  
## 3.96718750232328 5.664766e-05 2.255184e-05 8.978048e-06 3.574225e-06  
## 1.57936576992915 1.180753e-04 4.700664e-05 1.871368e-05 7.450051e-06  
## 0.62875682930618 2.116152e-04 8.424554e-05 3.353875e-05 1.335202e-05  
## 0.250312598845862 3.170476e-04 1.262189e-04 5.024865e-05 2.000435e-05  
## 0.0996512391127577 4.022301e-04 1.601307e-04 6.374917e-05 2.537900e-05  
## 0.0396718723008543 4.551725e-04 1.812074e-04 7.213998e-05 2.871944e-05  
## 0.0157936566154931 4.965053e-04 1.976623e-04 7.869079e-05 3.132737e-05  
## [,9] [,10] [,11]  
## 157.936587830899 1.606337e-41 6.394944e-42 2.545873e-42  
## 62.8756872452973 1.036524e-07 4.126477e-08 1.642780e-08  
## 25.031261602291 2.562372e-07 1.020099e-07 4.061086e-08  
## 9.96512459510634 6.192363e-07 2.465224e-07 9.814233e-08  
## 3.96718750232328 1.422925e-06 5.664765e-07 2.255184e-07  
## 1.57936576992915 2.965918e-06 1.180753e-06 4.700664e-07  
## 0.62875682930618 5.315534e-06 2.116152e-06 8.424554e-07  
## 0.250312598845862 7.963875e-06 3.170476e-06 1.262189e-06  
## 0.0996512391127577 1.010356e-05 4.022301e-06 1.601307e-06  
## 0.0396718723008543 1.143342e-05 4.551725e-06 1.812074e-06  
## 0.0157936566154931 1.247165e-05 4.965053e-06 1.976623e-06

The number of selected variables:

fit$df

## [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]  
## [1,] 0 7 11 24 53 73 91 99 102 102 102  
## [2,] 0 7 11 13 29 47 63 68 72 75 75  
## [3,] 0 7 11 13 24 44 61 68 74 75 75  
## [4,] 0 7 11 12 22 44 61 71 74 75 75  
## [5,] 0 7 10 11 22 43 60 73 75 75 75  
## [6,] 0 7 10 11 20 44 63 73 74 76 77  
## [7,] 0 7 10 11 22 44 64 76 81 83 84  
## [8,] 0 8 10 11 23 44 70 82 88 89 91  
## [9,] 0 8 10 11 23 46 70 87 96 100 101  
## [10,] 0 9 10 13 24 49 73 92 97 102 102  
## [11,] 0 9 11 13 24 51 79 90 98 101 102

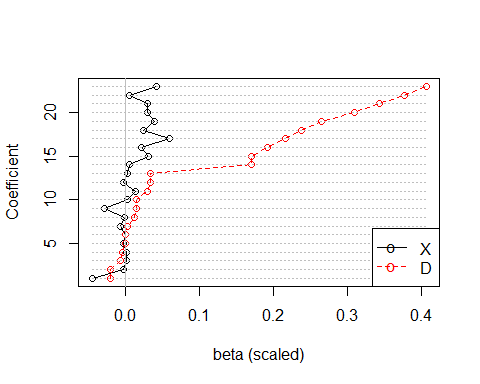
The sum (L1norm) of shrinkage factors:

fit$L1norm

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] 0 211.556685 1052.318241 3623.712988 11316.51047 32450.27340  
## [2,] 0 130.745327 466.063290 893.654013 1379.92546 2130.73561  
## [3,] 0 84.094545 264.236873 442.408868 625.41602 977.78213  
## [4,] 0 45.703353 132.152090 202.539464 274.38257 439.37233  
## [5,] 0 22.893427 63.202794 91.995195 122.28560 201.56715  
## [6,] 0 11.827009 31.797604 44.661184 58.89776 101.62636  
## [7,] 0 6.906411 18.134228 24.745535 33.29351 59.02909  
## [8,] 0 4.861268 12.159505 16.192796 22.44143 41.24298  
## [9,] 0 4.052788 9.478274 12.368712 17.59182 33.32820  
## [10,] 0 3.760853 8.223800 10.695260 15.27547 29.67572  
## [11,] 0 3.490641 7.564164 9.958099 14.02840 27.57097  
## [,7] [,8] [,9] [,10] [,11]  
## [1,] 86579.05682 223347.43206 567285.99951 1431282.4127 3601543.3098  
## [2,] 3243.93846 4388.44124 5373.65897 6144.6096 6608.2581  
## [3,] 1512.08845 1987.79458 2373.59036 2628.8838 2756.4445  
## [4,] 686.45308 882.47964 1041.49454 1128.3220 1166.2611  
## [5,] 316.55662 407.28222 476.66771 508.8218 522.1925  
## [6,] 159.24049 210.83240 243.41912 259.1182 266.8008  
## [7,] 95.24939 135.18365 162.35542 179.9064 192.2113  
## [8,] 71.70871 102.47486 126.46152 140.2605 146.2968  
## [9,] 60.26873 86.62083 106.88132 124.8447 134.8979  
## [10,] 55.54892 82.08905 100.93154 111.2791 116.0635  
## [11,] 53.17432 76.86271 91.93275 101.2106 106.2556

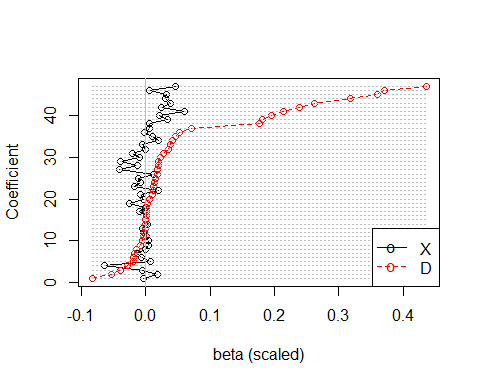
Let’s have a look at the optimal coefficients:

plot.coefficients.protogarrote(fit, order="d", lambda="lambda.min")



## $beta  
## beta.x beta.d  
## x.7 -4.483987e-02 -0.0202409801  
## x.9 -3.067175e-03 -0.0199369758  
## x.92 1.933258e-03 -0.0059246329  
## x.55 1.401191e-03 -0.0037865921  
## x.34 -1.836526e-03 0.0001695895  
## x.10 -9.625657e-06 0.0006157634  
## x.13 -6.268917e-03 0.0027336055  
## x.48 -1.675804e-03 0.0119416207  
## x.51 -2.860189e-02 0.0145105443  
## x.15 3.396043e-03 0.0150499230  
## x.94 1.387697e-02 0.0294626145  
## x.27 -1.949840e-03 0.0340157536  
## x.80 2.563714e-03 0.0345604095  
## x.23 5.503323e-03 0.1698016228  
## x.24 3.155588e-02 0.1700462942  
## x.25 2.148397e-02 0.1914668033  
## x.49 6.012144e-02 0.2158741893  
## x.54 2.450147e-02 0.2376177425  
## x.42 3.889406e-02 0.2639391107  
## x.81 3.018394e-02 0.3095620294  
## x.86 2.989921e-02 0.3423279570  
## x.77 5.701338e-03 0.3760177909  
## x.98 4.266193e-02 0.4058420764  
##   
## $selected  
## [1] "7" "9" "10" "13" "15" "23" "24" "25" "27" "34" "42" "48" "49" "51" "54"  
## [16] "55" "77" "80" "81" "86" "92" "94" "98"

plot.coefficients.protogarrote(fit, order="d", lambda="L1norm")

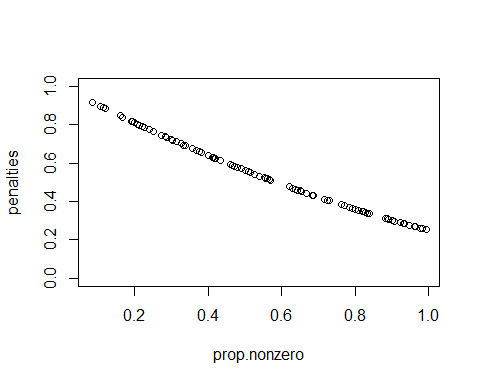


## $beta  
## beta.x beta.d  
## x.100 -4.112987e-03 -0.0837103958  
## x.92 1.755640e-02 -0.0538030667  
## x.9 -5.962589e-03 -0.0387574852  
## x.7 -6.451206e-02 -0.0291211190  
## x.55 7.402337e-03 -0.0200041403  
## x.68 -6.658030e-03 -0.0191486951  
## x.47 -1.324602e-02 -0.0182744465  
## x.56 -5.523473e-04 -0.0149644197  
## x.65 4.334161e-03 -0.0077754180  
## x.53 3.736259e-03 -0.0055286471  
## x.19 1.491186e-03 -0.0037406224  
## x.4 -3.080049e-03 -0.0023006458  
## x.61 -5.331993e-03 -0.0013141653  
## x.74 1.869832e-03 -0.0007454440  
## x.20 7.984213e-05 -0.0002042661  
## x.71 5.334128e-04 0.0001331198  
## x.37 -1.022956e-02 0.0009378212  
## x.93 -9.527100e-05 0.0010691591  
## x.34 -2.618899e-02 0.0024183589  
## x.32 -3.890340e-03 0.0055484892  
## x.84 -8.114235e-03 0.0097457184  
## x.31 1.913117e-02 0.0111216372  
## x.30 -1.820818e-02 0.0125165897  
## x.76 -8.069779e-03 0.0125714339  
## x.16 -1.122811e-02 0.0149748708  
## x.14 1.223445e-02 0.0177612460  
## x.13 -4.170853e-02 0.0181872976  
## x.69 -1.257142e-02 0.0188656152  
## x.51 -3.926312e-02 0.0199192848  
## x.39 -1.016350e-02 0.0224056415  
## x.1 -2.080917e-02 0.0280712153  
## x.10 -5.527493e-04 0.0353599567  
## x.48 -5.272561e-03 0.0375717661  
## x.94 1.916113e-02 0.0406815815  
## x.15 1.032249e-02 0.0457452311  
## x.27 -2.991981e-03 0.0521963240  
## x.80 5.288241e-03 0.0712886907  
## x.23 5.744164e-03 0.1772326286  
## x.24 3.372094e-02 0.1817132315  
## x.25 2.185460e-02 0.1947698720  
## x.49 5.952608e-02 0.2137364614  
## x.54 2.468078e-02 0.2393567327  
## x.42 3.864888e-02 0.2622752946  
## x.81 3.099888e-02 0.3179199247  
## x.86 3.148539e-02 0.3604887345  
## x.77 5.634526e-03 0.3716113240  
## x.98 4.582920e-02 0.4359721906  
##   
## $selected  
## [1] "1" "4" "7" "9" "10" "13" "14" "15" "16" "19" "20" "23"   
## [13] "24" "25" "27" "30" "31" "32" "34" "37" "39" "42" "47" "48"   
## [25] "49" "51" "53" "54" "55" "56" "61" "65" "68" "69" "71" "74"   
## [37] "76" "77" "80" "81" "84" "86" "92" "93" "94" "98" "100"

## Linear model with penalty factors

Now let’s prioritize those variables that have fewer zero values. We define the penalty (1= high penalty, 0=no penalty) for the peptides as follows:

prop.nonzero<-apply(d,2,mean)  
penalties <- (1-(prop.nonzero/2))\*\*2  
plot(prop.nonzero, penalties, ylim=c(0,1))

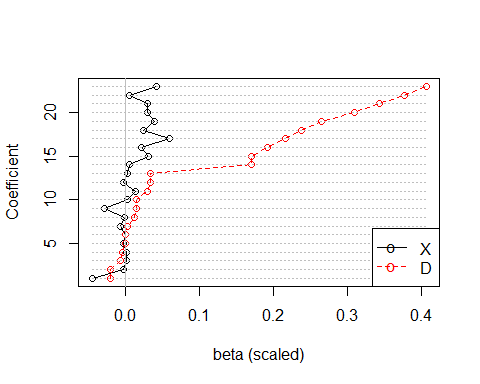


Now refit the linear model. The penalties are defined for all variables, i.e., for the x, the d, and the two clinical variables (which get no penalty):

set.seed(4821)  
fit2 <- protogarrote(data.obj, family="gaussian", cv=10, outer=3, nlambda=c(5,11), penalties=c(penalties, penalties, 0,0))

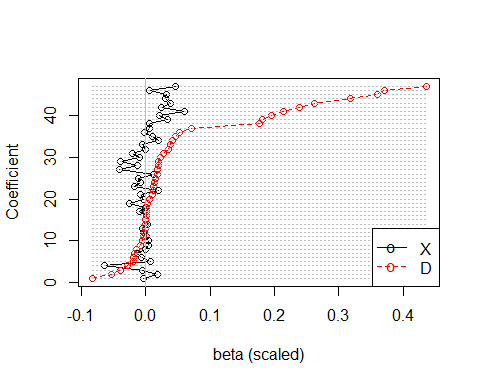
Let’s have a look at the coefficients:

plot.coefficients.protogarrote(fit, order="d")



## $beta  
## beta.x beta.d  
## x.7 -4.483987e-02 -0.0202409801  
## x.9 -3.067175e-03 -0.0199369758  
## x.92 1.933258e-03 -0.0059246329  
## x.55 1.401191e-03 -0.0037865921  
## x.34 -1.836526e-03 0.0001695895  
## x.10 -9.625657e-06 0.0006157634  
## x.13 -6.268917e-03 0.0027336055  
## x.48 -1.675804e-03 0.0119416207  
## x.51 -2.860189e-02 0.0145105443  
## x.15 3.396043e-03 0.0150499230  
## x.94 1.387697e-02 0.0294626145  
## x.27 -1.949840e-03 0.0340157536  
## x.80 2.563714e-03 0.0345604095  
## x.23 5.503323e-03 0.1698016228  
## x.24 3.155588e-02 0.1700462942  
## x.25 2.148397e-02 0.1914668033  
## x.49 6.012144e-02 0.2158741893  
## x.54 2.450147e-02 0.2376177425  
## x.42 3.889406e-02 0.2639391107  
## x.81 3.018394e-02 0.3095620294  
## x.86 2.989921e-02 0.3423279570  
## x.77 5.701338e-03 0.3760177909  
## x.98 4.266193e-02 0.4058420764  
##   
## $selected  
## [1] "7" "9" "10" "13" "15" "23" "24" "25" "27" "34" "42" "48" "49" "51" "54"  
## [16] "55" "77" "80" "81" "86" "92" "94" "98"

plot.coefficients.protogarrote(fit, order="d", lambda="L1norm")



## $beta  
## beta.x beta.d  
## x.100 -4.112987e-03 -0.0837103958  
## x.92 1.755640e-02 -0.0538030667  
## x.9 -5.962589e-03 -0.0387574852  
## x.7 -6.451206e-02 -0.0291211190  
## x.55 7.402337e-03 -0.0200041403  
## x.68 -6.658030e-03 -0.0191486951  
## x.47 -1.324602e-02 -0.0182744465  
## x.56 -5.523473e-04 -0.0149644197  
## x.65 4.334161e-03 -0.0077754180  
## x.53 3.736259e-03 -0.0055286471  
## x.19 1.491186e-03 -0.0037406224  
## x.4 -3.080049e-03 -0.0023006458  
## x.61 -5.331993e-03 -0.0013141653  
## x.74 1.869832e-03 -0.0007454440  
## x.20 7.984213e-05 -0.0002042661  
## x.71 5.334128e-04 0.0001331198  
## x.37 -1.022956e-02 0.0009378212  
## x.93 -9.527100e-05 0.0010691591  
## x.34 -2.618899e-02 0.0024183589  
## x.32 -3.890340e-03 0.0055484892  
## x.84 -8.114235e-03 0.0097457184  
## x.31 1.913117e-02 0.0111216372  
## x.30 -1.820818e-02 0.0125165897  
## x.76 -8.069779e-03 0.0125714339  
## x.16 -1.122811e-02 0.0149748708  
## x.14 1.223445e-02 0.0177612460  
## x.13 -4.170853e-02 0.0181872976  
## x.69 -1.257142e-02 0.0188656152  
## x.51 -3.926312e-02 0.0199192848  
## x.39 -1.016350e-02 0.0224056415  
## x.1 -2.080917e-02 0.0280712153  
## x.10 -5.527493e-04 0.0353599567  
## x.48 -5.272561e-03 0.0375717661  
## x.94 1.916113e-02 0.0406815815  
## x.15 1.032249e-02 0.0457452311  
## x.27 -2.991981e-03 0.0521963240  
## x.80 5.288241e-03 0.0712886907  
## x.23 5.744164e-03 0.1772326286  
## x.24 3.372094e-02 0.1817132315  
## x.25 2.185460e-02 0.1947698720  
## x.49 5.952608e-02 0.2137364614  
## x.54 2.468078e-02 0.2393567327  
## x.42 3.864888e-02 0.2622752946  
## x.81 3.099888e-02 0.3179199247  
## x.86 3.148539e-02 0.3604887345  
## x.77 5.634526e-03 0.3716113240  
## x.98 4.582920e-02 0.4359721906  
##   
## $selected  
## [1] "1" "4" "7" "9" "10" "13" "14" "15" "16" "19" "20" "23"   
## [13] "24" "25" "27" "30" "31" "32" "34" "37" "39" "42" "47" "48"   
## [25] "49" "51" "53" "54" "55" "56" "61" "65" "68" "69" "71" "74"   
## [37] "76" "77" "80" "81" "84" "86" "92" "93" "94" "98" "100"

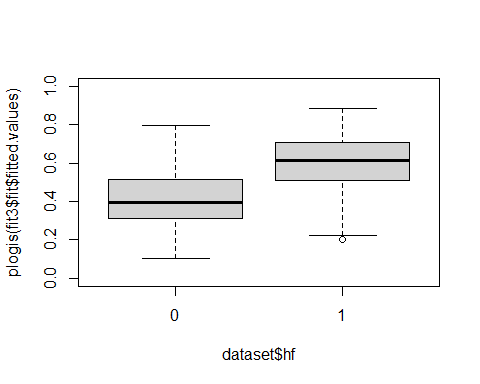
## Logistic model without interactions

The logistic model could be first fit without any interactions, using hf as the outcome variable. Be sure to define the penalties object correctly!

x <- as.matrix(dataset[, paste0("x.", 1:k,sep="")])  
d <- as.matrix(dataset[, paste0("d.", 1:k,sep="")])  
age <- dataset$age  
sex <- dataset$sex  
log2eGFR <- dataset$log2eGFR  
  
data.obj<-list(x=x, d=d, y=dataset$hf, clinical=cbind(age, sex, log2eGFR=dataset$log2eGFR))  
  
fit3 <- protogarrote(data.obj, family="binomial", cv=10, outer=2, nlambda=c(11,11), penalties=c(penalties, penalties, 0,0,0))

Quick check: fitted values against true outcome status:

boxplot(plogis(fit3$fit$fitted.values)~dataset$hf, ylim=c(0,1))



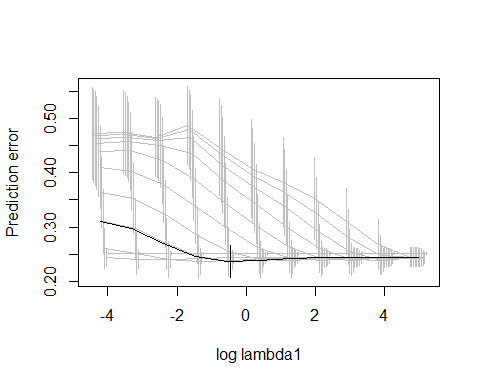
Let us check the r-squared:

fit3$cv.rsquare

## [,1] [,2] [,3] [,4] [,5] [,6] [,7]  
## [1,] 0 0.04882940 0.04879129 0.05290499 0.05713761 0.058091780 0.0592738814  
## [2,] 0 0.04882940 0.04879129 0.05287222 0.05713041 0.058089755 0.0654174140  
## [3,] 0 0.04882940 0.04879129 0.05281526 0.05711789 0.066736744 0.0524381290  
## [4,] 0 0.04882940 0.04879129 0.05267020 0.06664377 0.045776262 0.0565654022  
## [5,] 0 0.04882940 0.04879128 0.06061655 0.04642297 0.055239058 0.1030241718  
## [6,] 0 0.04882940 0.04879128 0.05407436 0.04891893 0.102929058 0.0838247918  
## [7,] 0 0.04882939 0.06806835 0.05395615 0.09010984 0.096210231 0.0732507899  
## [8,] 0 0.06190918 0.04583150 0.08233821 0.10514249 0.079637687 0.0726386098  
## [9,] 0 0.06779746 0.07531951 0.11231752 0.07570560 0.053948772 0.0765654029  
## [10,] 0 0.06471871 0.10539104 0.08953358 0.04969106 0.025016969 0.0086230486  
## [11,] 0 0.07421955 0.12046579 0.06264243 0.02570499 0.003361344 0.0004340653  
## [,8] [,9] [,10] [,11]  
## [1,] 0.0610333210 0.0616776176 0.0619251464 6.202234e-02  
## [2,] 0.0651601990 0.0587119899 0.0799020548 7.934045e-02  
## [3,] 0.0561589486 0.0881752548 0.0788188871 5.741051e-02  
## [4,] 0.0969590938 0.0760091136 0.0517487571 4.222799e-02  
## [5,] 0.0761668954 0.0528786013 0.0469873475 4.541012e-02  
## [6,] 0.0619655397 0.0548217308 0.0605819335 6.529622e-02  
## [7,] 0.0683156388 0.0705492388 0.0677446559 6.675435e-02  
## [8,] 0.0852982546 0.0578889572 0.0034475663 9.472309e-04  
## [9,] 0.0571493803 0.0415252710 0.0341750430 4.019974e-02  
## [10,] 0.0040888774 0.0025238328 0.0017760196 1.376289e-03  
## [11,] 0.0003608694 0.0004436156 0.0002855835 7.952311e-05

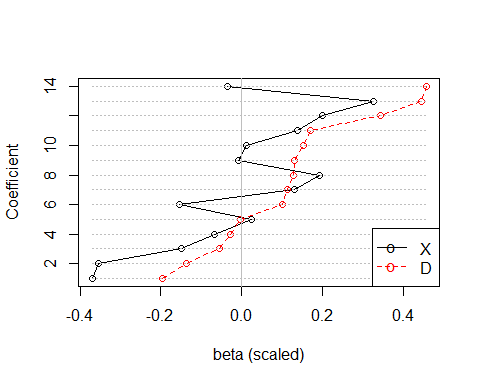
and plot the cross-validation:

plot.protogarrote(fit3)



Let’s check the coefficients (the order option sets the ordering by the D coefficients):

plot.coefficients.protogarrote(fit3, order="d")



## $beta  
## beta.x beta.d  
## x.45 -0.370499937 -0.196748191  
## x.22 -0.355804935 -0.136924780  
## x.13 -0.149733892 -0.053781494  
## x.38 -0.068332792 -0.027067153  
## x.70 0.023454033 -0.002681556  
## x.87 -0.155021646 0.100285318  
## x.71 0.130658368 0.114047427  
## x.39 0.192942085 0.127661601  
## x.97 -0.007557664 0.130942808  
## x.28 0.012531258 0.152480710  
## x.41 0.137375464 0.170174709  
## x.100 0.200181746 0.342833273  
## x.88 0.327186190 0.446728114  
## x.54 -0.035932031 0.457502694  
##   
## $selected  
## [1] "13" "22" "28" "38" "39" "41" "45" "54" "70" "71" "87" "88"   
## [13] "97" "100"

betas<-coefficients.protogarrote(fit3)  
t(t(betas.hot<-betas[betas!=0]))

## [,1]  
## (Intercept) -8.740112653  
## x.13 -0.149733892  
## x.22 -0.355804935  
## x.28 0.012531258  
## x.38 -0.068332792  
## x.39 0.192942085  
## x.41 0.137375464  
## x.45 -0.370499937  
## x.54 -0.035932031  
## x.70 0.023454033  
## x.71 0.130658368  
## x.87 -0.155021646  
## x.88 0.327186190  
## x.97 -0.007557664  
## x.100 0.200181746  
## d.13 -0.053781494  
## d.22 -0.136924780  
## d.28 0.152480710  
## d.38 -0.027067153  
## d.39 0.127661601  
## d.41 0.170174709  
## d.45 -0.196748191  
## d.54 0.457502694  
## d.70 -0.002681556  
## d.71 0.114047427  
## d.87 0.100285318  
## d.88 0.446728114  
## d.97 0.130942808  
## d.100 0.342833273  
## log2eGFR 0.957164325

## Logistic model with interactions

Now we fit the most advanced model: the logistic model for HF with interactions with log2eGFR. The mean and SD of log2eGFR are approximately 5.8 and 0.5, respectively. We also submit these to the protogarrote function as it will be helpful for numerical stability and interpretation of coefficients.

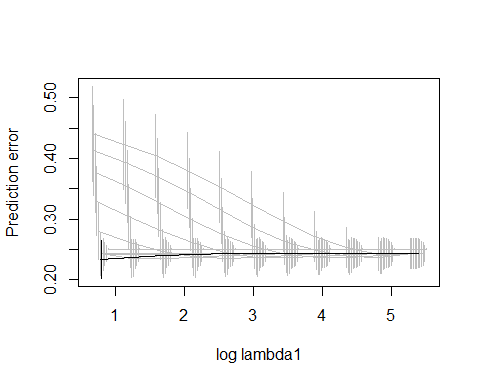
The four blocks of variables are: x, d, xlog2eGFR, and dlog2eGFR. For the interactions, log2eGFR is standardized with the center and scaling parameters submitted to the function. log2eGFR itself is internally added to the clinical variables.

It is crucial that there are always variables with 0 penalty factor, otherwise the program may have an error in the cross-validation, as all betas might be penalized to 0 then. Anyway, it is always good to not penalize the clinical variables. We define the penalty factors for the four blocks of proteomics variables (x, d, xlog2eGFR, and dlog2eGFR) and for the clinical variables. Despite the clinical variables get a penalty factor of 0, this penalty factor only applies to the first, ridge, stage. At the second, garrote, stage, they can as well be shrunken to zero.

data.obj<-list(x=x, d=d, y=dataset$hf, clinical=cbind(age, sex), interaction.x=log2eGFR)  
  
fit4 <- protogarrote(data.obj, center.interaction.x=5.8, scale.interaction.x=0.5, family="binomial",   
 penalties=c(penalties, penalties, penalties,penalties,0,0,0))

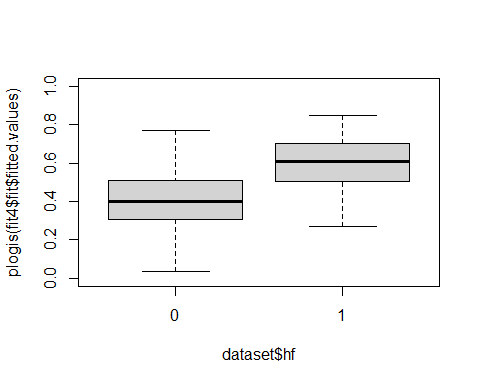
We first plot the object:

plot.protogarrote(fit4)



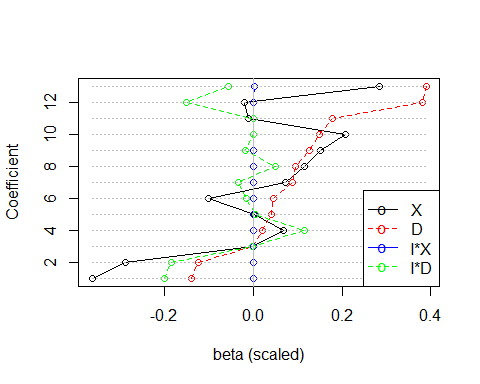
And we plot the apparent predictions vs. the true status:

boxplot(plogis(fit4$fit$fitted.values)~ dataset$hf, ylim=c(0,1))



Now let’s see the coefficients for the proteomics:

(pcp<-plot.coefficients.protogarrote(fit4, order="d"))



## $beta  
## beta.x beta.d beta.ix beta.id  
## x.45 -0.364292802 -0.140004540 4.862110e-04 -2.017213e-01  
## x.22 -0.288516538 -0.124413865 -2.676206e-04 -1.851877e-01  
## x.38 -0.003089366 -0.001075225 -1.265966e-05 -2.224451e-03  
## x.36 0.067144428 0.020632948 -7.019965e-05 1.140720e-01  
## x.28 0.002574359 0.041659218 8.153003e-05 4.218161e-03  
## x.87 -0.101686782 0.046117796 3.671033e-04 -1.508457e-02  
## x.41 0.071392531 0.088448771 -2.575491e-04 -3.277561e-02  
## x.71 0.115667794 0.093923123 -5.887083e-04 4.963273e-02  
## x.39 0.150459173 0.127497207 6.140218e-04 -1.889023e-02  
## x.100 0.207184107 0.148774498 1.166130e-04 -3.228127e-07  
## x.97 -0.012154526 0.178541032 -1.796918e-04 -6.191144e-04  
## x.54 -0.020029857 0.381295295 6.467553e-04 -1.509052e-01  
## x.88 0.285010363 0.390115787 1.555595e-03 -5.536082e-02  
##   
## $selected  
## [1] "22" "28" "36" "38" "39" "41" "45" "54" "71" "87" "88" "97"   
## [13] "100"

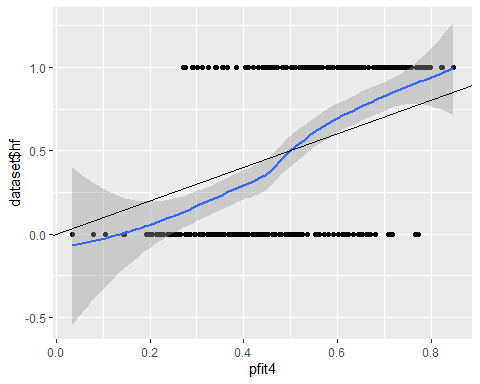
# these were the peptides used to generate the HF data:  
sort(predictive)

## [1] 23 24 25 42 49 54 77 81 86 98

Let us check the apparent calibration:

pfit4<-predict.protogarrote(fit4, newdata=data.obj, type="response")  
ggplot(NULL, aes(pfit4, dataset$hf))+geom\_point()+geom\_smooth()+geom\_abline(intercept=0, slope=1)

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'



etafit4<-predict.protogarrote(fit4, newdata=data.obj, type="link")  
summary(calibfit4<-glm(dataset$hf~etafit4, family="binomial"))

##   
## Call:  
## glm(formula = dataset$hf ~ etafit4, family = "binomial")  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.1936 -0.8578 0.3219 0.8592 2.0327   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.0343 0.1374 -0.250 0.803   
## etafit4 1.9228 0.2403 8.001 1.24e-15 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 415.83 on 299 degrees of freedom  
## Residual deviance: 318.30 on 298 degrees of freedom  
## AIC: 322.3  
##   
## Number of Fisher Scoring iterations: 4

The calibration slope is 1.9228521 which seems to point at underfit -> strong shrinkage, which should guard against overfit when the model is applied to new data.