Simulation Study

```r
library(survival)
library(glmnet)
```

```
## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-16
```

```r
library(polspline)
library(knitr)
library(EnvStats)
```

```
##
## Attaching package: 'EnvStats'

## The following object is masked from 'package:Matrix':
##
##     print

## The following objects are masked from 'package:stats':
##
##     predict, predict.lm

## The following object is masked from 'package:base':
##
##     print.default
```

```r
library(bda)
library(tictoc)
library(abind)
```

Simulating Survival Time with a Weibull Distribution

This function is based on simulate_data in https://cran.r-project.org/web/packages/rsimsum/vignettes/relhaz.html

```r
#' Simulate survival times with censoring, based on a weibull baseline hazard
#'
#' This function simulates survival times with censoring, according to a weibull
#' baseline hazard that the user parameterizes. The survival/censoring times are
#' simulated for user-given covariates and coefficients.
#' @param x model matrix of x-values for the Cox model of survival times
#' @param fcts_select subset of fcts from a hare object containing the coefficients of interest.
#' @params params parameters shape and scale for the baseline Weibull distribution,
#' by default the exponential distribution with scale = 1
#' @param FUN random generation function for the distribution of censoring times,
#' expected to be uniform, exponential, or weibull.
#' @param ... arguments for FUN, the random generation function
#' @return dataframe appending survival time and censoring indicator to the model matrix x
#' @export
simulate_weibull <- function(x, fcts_select, params = list(shape = 1, scale = 1), FUN, ...) {
  n <- nrow(x)

  # extract unique list of covariates selected
  cov_nums <- sort(fcts_select[,1][fcts_select[,1] != 0])
  cov_names <- colnames(x)[cov_nums]
  x_select <- x[,cov_names]

  # extract the coefficient values from fcts_select
  betas <- fcts_select[,5][fcts_select[,1] != 0]

  # simulate survival times according to Bender et al. (2005)
  u <- runif(n)
  time <- (-log(u) / (params$scale * exp(x_select %*% betas)))^(1 / params$shape)

  # Winsorising tiny values for time (smaller than one day on a yearly-scale, e.g. 1 / 365.242),
  # and adding a tiny amount of white noise not to have too many concurrent values
  time <- ifelse(time < 1 / 365.242, 1 / 365.242, time)
  time[time == 1 / 365.242] <- time[time == 1 / 365.242] +
    rnorm(length(time[time == 1 / 365.242]), mean = 0, sd = 1e-4)
  # ...and make sure that the resulting value is positive
  time <- abs(time)

  # Censoring
  # cid <- sample(c(0, 1), size = n, replace = TRUE, prob = c(.75, .25))
  cid_time <- FUN(n, ...)

  cid <- ifelse(time <= cid_time, 1, 0)

  time <- pmin(time, cid_time)

  # return a dataframe
  data.frame(time, cid, x)
}
```

```r
load("actg175.RData")

x <- model.matrix( ~ trt + age + wtkg + hemo + drugs +
                     karnof + oprior + preanti + race +
                     gender + symptom + offtrt + cd40 +
                     cd80, actg175)[,-1]

nphm_hare <- readRDS("nphm_hare.rds")

# extracting the coefficients for basis functions
# that do not correspond to knots and/or tensor products
fcts <- nphm_hare$fcts
fcts_select <- fcts[fcts[,2] == 0 & is.na(fcts[,3]),]
```

Using arbitrary Weibull parameter values to get similar survival times as original study

```
set.seed(2)

test_sim_mat <- simulate_weibull(x, fcts_select,
                                 params = list(shape = 500,
                                               scale = 1),
                                 FUN = rexp, 1.3)

summary(test_sim_mat$time)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000123 0.2066407 0.4692176 0.5644418 0.8135539 2.0275567
```

```
mean(test_sim_mat$cid)
```

```
## [1] 0.2431043
```

Using Weibull parameter estimates from fit.Weibull

```
set.seed(1)

parm_res <- fit.Weibull(rhare(100000, cov = rep(0, nphm_hare$ncov), nphm_hare), dist="Weibull")
```

```
set.seed(2)

sim_mat <- simulate_weibull(x, fcts_select,
                            params = list(shape = parm_res$pars[2],
                                          scale = parm_res$pars[1]),
                            FUN = rexp, 1.3)

summary(sim_mat$time)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000123 0.1901507 0.4084255 0.5705628 0.7844852 3.0695179
```

```
mean(sim_mat$cid)
```

```
## [1] 0.2435718
```

Coxph Simulation

Using arbitrary Weibull parameter values

```
phm_test_sim_mat <- coxph(Surv(time, cid) ~ ., data = test_sim_mat)

phm_test_sim_mat
```

```
## Call:
## coxph(formula = Surv(time, cid) ~ ., data = test_sim_mat)
##
```

```
##                  coef  exp(coef)   se(coef)       z      p
## trtZDV.ddi -0.0315080  0.9689832  0.1519655  -0.207 0.8357
## trtZDV.ZAL -0.3707338  0.6902277  0.1559796  -2.377 0.0175
## trtddi     -0.0275478  0.9728282  0.1535824  -0.179 0.8576
## age         0.0036990  1.0037058  0.0071670   0.516 0.6058
## wtkg        0.0003467  1.0003468  0.0049080   0.071 0.9437
## hemo1       0.0405321  1.0413648  0.1896735   0.214 0.8308
## drugs1      0.0377496  1.0384712  0.1773127   0.213 0.8314
## karnof     -0.0007312  0.9992691  0.0090552  -0.081 0.9356
## oprior1    -0.0030029  0.9970016  0.3480679  -0.009 0.9931
## preanti     0.1620180  1.1758814  0.0075108  21.571 <2e-16
## race1       0.0269866  1.0273540  0.1287316   0.210 0.8340
## gender1    -0.0460705  0.9549747  0.1598970  -0.288 0.7733
## symptom1   -0.3391385  0.7123838  0.1374381  -2.468 0.0136
## offtrt1    -0.2270705  0.7968646  0.1189267  -1.909 0.0562
## cd40       -0.2027929  0.8164473  0.0093967 -21.581 <2e-16
## cd80       -0.0067717  0.9932511  0.0003382 -20.025 <2e-16
##
## Likelihood ratio test=4453  on 16 df, p=< 2.2e-16
## n= 2139, number of events= 520
```

Using Weibull parameter estimates from fit.Weibull

```
phm_sim_mat <- coxph(Surv(time, cid) ~ ., data = sim_mat)

phm_sim_mat
```

```
## Call:
## coxph(formula = Surv(time, cid) ~ ., data = sim_mat)
##
##                  coef  exp(coef)   se(coef)       z       p
## trtZDV.ddi -0.0125363  0.9875420  0.1502395  -0.083 0.93350
## trtZDV.ZAL -0.3480515  0.7060625  0.1579983  -2.203 0.02760
## trtddi     -0.0259318  0.9744016  0.1538496  -0.169 0.86615
## age         0.0069482  1.0069724  0.0068392   1.016 0.30966
## wtkg       -0.0015353  0.9984659  0.0048394  -0.317 0.75106
## hemo1       0.0087963  1.0088351  0.1808414   0.049 0.96121
## drugs1     -0.0237253  0.9765540  0.1825600  -0.130 0.89660
## karnof      0.0045485  1.0045589  0.0090543   0.502 0.61542
## oprior1     0.0555272  1.0570978  0.2967025   0.187 0.85154
## preanti     0.1623560  1.1762790  0.0074766  21.715 < 2e-16
## race1      -0.0682251  0.9340502  0.1323887  -0.515 0.60632
## gender1     0.0445937  1.0456030  0.1648220   0.271 0.78673
## symptom1   -0.3890929  0.6776713  0.1394388  -2.790 0.00526
## offtrt1    -0.2722028  0.7616998  0.1202460  -2.264 0.02359
## cd40       -0.2031480  0.8161575  0.0093562 -21.713 < 2e-16
## cd80       -0.0068301  0.9931931  0.0003365 -20.295 < 2e-16
##
## Likelihood ratio test=5013  on 16 df, p=< 2.2e-16
## n= 2139, number of events= 521
```

Weibull model (to compare with Coxph)

The Weibull model should be more powerful (with less variance in the coefficient estimates) than the Cox Proportional Hazards model, as the data is from a Weibull distribution. The coefficient estimates themselves should be similar.

```r
summary(survreg(Surv(time, cid) ~ ., data = sim_mat))
```

```
##
## Call:
## survreg(formula = Surv(time, cid) ~ ., data = sim_mat)
##                  Value Std. Error       z      p
## (Intercept)  2.05e-03   2.71e-02    0.08   0.94
## trtZDV.ddi  -6.59e-04   4.28e-03   -0.15   0.88
## trtZDV.ZAL   3.44e-03   4.48e-03    0.77   0.44
## trtddi      -1.09e-04   4.31e-03   -0.03   0.98
## age          8.62e-06   1.94e-04    0.04   0.96
## wtkg        -4.07e-05   1.39e-04   -0.29   0.77
## hemo1       -1.45e-03   5.20e-03   -0.28   0.78
## drugs1       5.92e-04   5.12e-03    0.12   0.91
## karnof       1.91e-05   2.46e-04    0.08   0.94
## oprior1     -2.86e-04   8.64e-03   -0.03   0.97
## preanti     -1.65e-03   3.27e-06 -505.37 <2e-16
## race1        2.13e-04   3.85e-03    0.06   0.96
## gender1      3.01e-04   4.56e-03    0.07   0.95
## symptom1     2.99e-03   4.04e-03    0.74   0.46
## offtrt1      1.79e-03   3.32e-03    0.54   0.59
## cd40         2.06e-03   1.52e-05  136.00 <2e-16
## cd80         6.98e-05   3.37e-06   20.73 <2e-16
## Log(scale)  -3.36e+00   4.23e-02  -79.39 <2e-16
##
## Scale= 0.0347
##
## Weibull distribution
## Loglik(model)= 1033.9   Loglik(intercept only)= -908.8
##  Chisq= 3885.39 on 16 degrees of freedom, p= 0
## Number of Newton-Raphson Iterations: 17
## n= 2139
```

For coefficients with very low p-values ($p < 2e-16$) according to the coxph model, the values from the Weibull model are approximately equal to those from the coxph model divided by 100.

Glmnet Simulation

```r
cv_phmnet <- cv.glmnet(as.matrix(sim_mat[-c(1,2)]),
                       Surv(sim_mat$time, sim_mat$cid),
                       family = "cox", alpha = .95)
```

```
## Warning: from glmnet Fortran code (error code -74); Convergence for 74th
## lambda value not reached after maxit=100000 iterations; solutions for
## larger lambdas returned
```

```
## Warning: from glmnet Fortran code (error code -76); Convergence for 76th
## lambda value not reached after maxit=100000 iterations; solutions for
## larger lambdas returned

## Warning: from glmnet Fortran code (error code -76); Convergence for 76th
## lambda value not reached after maxit=100000 iterations; solutions for
## larger lambdas returned

## Warning: from glmnet Fortran code (error code -76); Convergence for 76th
## lambda value not reached after maxit=100000 iterations; solutions for
## larger lambdas returned

## Warning: from glmnet Fortran code (error code -76); Convergence for 76th
## lambda value not reached after maxit=100000 iterations; solutions for
## larger lambdas returned

## Warning: from glmnet Fortran code (error code -69); Convergence for 69th
## lambda value not reached after maxit=100000 iterations; solutions for
## larger lambdas returned

## Warning: from glmnet Fortran code (error code -72); Convergence for 72th
## lambda value not reached after maxit=100000 iterations; solutions for
## larger lambdas returned

## Warning: from glmnet Fortran code (error code -76); Convergence for 76th
## lambda value not reached after maxit=100000 iterations; solutions for
## larger lambdas returned

## Warning: from glmnet Fortran code (error code -76); Convergence for 76th
## lambda value not reached after maxit=100000 iterations; solutions for
## larger lambdas returned

## Warning: from glmnet Fortran code (error code -73); Convergence for 73th
## lambda value not reached after maxit=100000 iterations; solutions for
## larger lambdas returned

## Warning: from glmnet Fortran code (error code -77); Convergence for 77th
## lambda value not reached after maxit=100000 iterations; solutions for
## larger lambdas returned
```

```r
coef(cv_phmnet, s = cv_phmnet$lambda.1se)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## trtZDV.ddi   0.0051094207
## trtZDV.ZAL -0.0632111356
## trtddi      .
## age         .
## wtkg        .
## hemo1       .
## drugs1     -0.0007287251
## karnof      0.0009972092
## oprior1     0.0013966838
## preanti     0.0473722429
## race1       .
## gender1     .
## symptom1   -0.1163350309
## offtrt1    -0.0559478504
## cd40       -0.0592515887
## cd80       -0.0019513857
```

PH HARE Simulation

```
phm_hare <- hare(sim_mat$time, sim_mat$cid, as.matrix(sim_mat[-c(1,2)]), prophaz = TRUE)

# extracting the coefficients for basis functions
# that do not correspond to time, knots, and/or tensor products
fcts <- phm_hare$fcts
fcts[fcts[,1] != 0 & fcts[,2] == 0 & is.na(fcts[,3]),]
```

```
##    dim1 knot1 dim2 knot2         beta           SE
## 2    10     0   NA    NA  0.188417970 0.0073612133
## 5    15     0   NA    NA -0.234731573 0.0091819926
## 6     2     0   NA    NA -0.330200342 0.1092425677
## 10   16     0   NA    NA -0.008007272 0.0003278145
## 24   14     0   NA    NA -0.356229005 0.0983720909
```

PH HARE Simulation

```
nphm_hare_sim <- hare(sim_mat$time, sim_mat$cid, as.matrix(sim_mat[-c(1,2)]))

# extracting the coefficients for basis functions
# that do not correspond to time, knots, and/or tensor products
fcts <- nphm_hare_sim$fcts
fcts[fcts[,1] != 0 & fcts[,2] == 0 & is.na(fcts[,3]),]
```

```
##   dim1 knot1 dim2 knot2         beta           SE
## 2   10     0   NA    NA  0.299330488 0.0133413983
## 5   15     0   NA    NA -0.262780398 0.0091456414
## 9   16     0   NA    NA -0.009501459 0.0003553189
```

Simulating N Times

```
#' Select and calculate coefficient estimates for Cox PH, penalized PH, HARE PH,
#' and HARE non-PH models based on N simulated data sets.
#'
#' This function calls the simulation function for the distribution of interest N
#' times and calculates the corresponding regression coefficient estimates for
#' each simulated data set.
#' @param N number of simulated data sets to fit the models to.
#' @param p number of covaraites in original dataset
#' @param simulate_fun simulation function for the distribution of interest.
#' @param ... arguments for simulate_fun
#' @return p x 4 x N array containing the coefficient estimates for variables
#' selected among p initial variables by 4 models fitted on N data sets
#' @export
```

```r
simulate_regression <- function(N, p, simulate_fun = simulate_weibull, ...) {

  res <- array(NA, dim = c(p, 4, N))

  for (i in 1:N) {

    sim_mat <- simulate_fun(...)


    # Cox Proportional Hazards model
    phm_sim_mat <- coxph(Surv(time, cid) ~ ., data = sim_mat)

    s_phm_sim_mat <- summary(phm_sim_mat)

    # only storing the coefficient values for which the p-value is <= .05
    res[s_phm_sim_mat$coefficients[,5] <= .05, 1, i] <-
      phm_sim_mat$coefficients[s_phm_sim_mat$coefficients[,5] <= .05]


    # Penalized Proportional Hazards model
    cv_phmnet <- cv.glmnet(as.matrix(sim_mat[-c(1,2)]),
                           Surv(sim_mat$time, sim_mat$cid),
                           family = "cox", alpha = .95)

    selected_coef <- as.numeric(coef(cv_phmnet, s = cv_phmnet$lambda.1se))

    res[selected_coef != 0, 2, i] <- selected_coef[selected_coef != 0]



    # PH HARE model
    phm_hare <- hare(sim_mat$time, sim_mat$cid,
                     as.matrix(sim_mat[-c(1,2)]), prophaz = TRUE)

    # extracting the coefficients for basis functions
    # that do not correspond to time, knots, and/or tensor products
    fcts <- phm_hare$fcts
    fcts_select <- fcts[fcts[,1] != 0 & fcts[,2] == 0 & is.na(fcts[,3]),]

    res[fcts_select[,1], 3, i] <- fcts_select[,5]



    # non-PH HARE model
    nphm_hare <- hare(sim_mat$time, sim_mat$cid, as.matrix(sim_mat[-c(1,2)]))

    # extracting the coefficients for basis functions
    # that do not correspond to time, knots, and/or tensor products
    nphm_fcts <- nphm_hare$fcts
    nphm_fcts_select <- nphm_fcts[nphm_fcts[,1] != 0 &
                                  nphm_fcts[,2] == 0 & is.na(nphm_fcts[,3]),]

    res[nphm_fcts_select[,1], 4, i] <- nphm_fcts_select[,5]
  }

  return(res)

}
```

I group the simulate_regression iterations in groups of a hundred each to make it easier for R to handle. The total is 500 simulations.

```
# set.seed(636)
#
# tic()
# sims <- simulate_regression(N = 100, p = 16, simulate_fun = simulate_weibull, x, fcts_select,
#                                 params = list(shape = parm_res$pars[2],
#                                               scale = parm_res$pars[1]),
#                                 FUN = rexp, rate = 1.3)
#
# save(sims, file = "sims1.RData")
# toc()
#
# tic()
# sims_next <- simulate_regression(N = 100, p = 16, simulate_fun = simulate_weibull, x, fcts_select,
#                                 params = list(shape = parm_res$pars[2],
#                                               scale = parm_res$pars[1]),
#                                 FUN = rexp, rate = 1.3)
#
# sims <- abind(sims, sims_next)
#
# save(sims, file = "sims2.RData")
# toc()
#
# tic()
# sims_next <- simulate_regression(N = 100, p = 16, simulate_fun = simulate_weibull, x, fcts_select,
#                                 params = list(shape = parm_res$pars[2],
#                                               scale = parm_res$pars[1]),
#                                 FUN = rexp, rate = 1.3)
#
# sims <- abind(sims, sims_next)
#
# save(sims, file = "sims3.RData")
# toc()
#
# tic()
# sims_next <- simulate_regression(N = 100, p = 16, simulate_fun = simulate_weibull, x, fcts_select,
#                                 params = list(shape = parm_res$pars[2],
#                                               scale = parm_res$pars[1]),
#                                 FUN = rexp, rate = 1.3)
#
# sims <- abind(sims, sims_next)
#
# save(sims, file = "sims4.RData")
# toc()
#
# tic()
# sims_next <- simulate_regression(N = 100, p = 16, simulate_fun = simulate_weibull, x, fcts_select,
#                                  params = list(shape = parm_res$pars[2],
#                                                scale = parm_res$pars[1]),
#                                  FUN = rexp, rate = 1.3)
#
# sims <- abind(sims, sims_next)
#
# save(sims, file = "sims5.RData")
# toc()
```

If I use set.seed(3), then there would be an error regarding the fitter() function (something about NA/NaN/Inf“”). But with this set.seed, the coxph model still works.

Regarding “Loglik converged before variable. . . ” see https://stat.ethz.ch/pipermail/r-help/2008-September/174201.html.

```
load("sims1.RData")

prop_weibull <- matrix(0, nrow = 16, ncol = 4)

for (i in 1:16) {

  for (j in 1:4) {

    prop_weibull[i,j] <- sum(!is.na(sims[i, j,])) / dim(sims)[3]

  }

}

prop_weibull
```

```
##       [,1] [,2] [,3] [,4]
##  [1,] 0.00 0.59 0.00 0.00
##  [2,] 0.59 0.94 0.87 0.35
##  [3,] 0.00 0.47 0.00 0.03
##  [4,] 0.02 0.50 0.01 0.01
##  [5,] 0.00 0.50 0.00 0.00
##  [6,] 0.00 0.63 0.00 0.00
##  [7,] 0.00 0.38 0.00 0.09
##  [8,] 0.00 0.57 0.00 0.00
##  [9,] 0.00 0.46 0.00 0.01
## [10,] 1.00 1.00 1.00 1.00
## [11,] 0.00 0.41 0.00 0.02
## [12,] 0.00 0.48 0.00 0.00
## [13,] 0.37 0.93 0.40 0.15
## [14,] 0.64 0.93 0.88 0.46
## [15,] 1.00 1.00 1.00 1.00
## [16,] 1.00 1.00 1.00 0.97
```