Simulation Study

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

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-16
```

```
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
```

```
library(bda)
library(tictoc)
library(abind)
library(fitdistrplus)
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:EnvStats':
##
##     boxcox
```

```
## Loading required package: npsurv
```

```
## Loading required package: lsei
```

Function for Simulating Survival Time from a Weibull Distribution

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

```
#' 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 times are
```

```r
#' simulated for user-given covariates and coefficients, and the censoring times
#' are simulated for a user-given distribution.
#' @param x model matrix of covariate values
#' @param fcts_select subset of fcts from a hare object containing the coefficients of interest.
#' @param 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_time <- FUN(n, ...)

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

  time <- pmin(time, cid_time)

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

Function for Simulating Survival Time from a Log-normal Distribution

```r
#' Simulate survival times with censoring, based on a log-normal distribution
#'
```

```r
#' This function simulates survival times with censoring, according to a log-normal distribution
#' that the user parameterizes. The survival times are
#' simulated for user-given covariates and coefficients, and the censoring times
#' are simulated for a user-given distribution.
#' @param x model matrix of covariate values
#' @param fcts_select subset of fcts from a hare object containing the coefficients of interest.
#' @param params_not_off parameters for the baseline log-normal distribution of survival times,
#' for the not off-treatment group (Offtrt = 0)
#' @param params_off parameters for the baseline log-normal distribution of survival times,
#' for the off-treatment group (Offtrt = 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_lnorm <- function(x, fcts_select, params_not_off, params_off, 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]

  # Take out the offtrt covariate and offtrt coefficient from x_select and betas, because
  # the simulation will be stratified on offtrt
  x_select <- x_select[,-7]
  betas <- betas[-7]

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

  # simulate survival times based on the survsim package
  time <- c()
  # going row by row in the original ACTG-175 data set, simulating the survival time depending on
  # whether Offtrt = 0 or 1.
  for (i in 1:n) {

    if (x[i, 14] == 0) { # not off-treatment group

      time[i] <- rlnorm(1, params_not_off$meanlog + x_select[i,] %*% betas, params_not_off$sdlog)

    } else if (x[i, 14] == 1) { # off-treatment group

      time[i] <- rlnorm(1, params_off$meanlog + x_select[i,] %*% betas, params_off$sdlog)

    }

  }
```

```
  # 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_time <- FUN(n, ...)

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

  time <- pmin(time, cid_time)

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

Setting up the requisite parameters for simulation

```
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]),]
```

Generating example data set from the Weibull distribution

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
```

```
# censoring rate for the simulated data set
1 - mean(sim_mat$cid)
```

```
## [1] 0.7564282
```

```
# censoring rate for the original study
1 - mean(actg175$cid)
```

```
## [1] 0.7564282
```

```
1 - mean(sim_mat$cid) == 1 - mean(actg175$cid)
```

```
## [1] TRUE
```

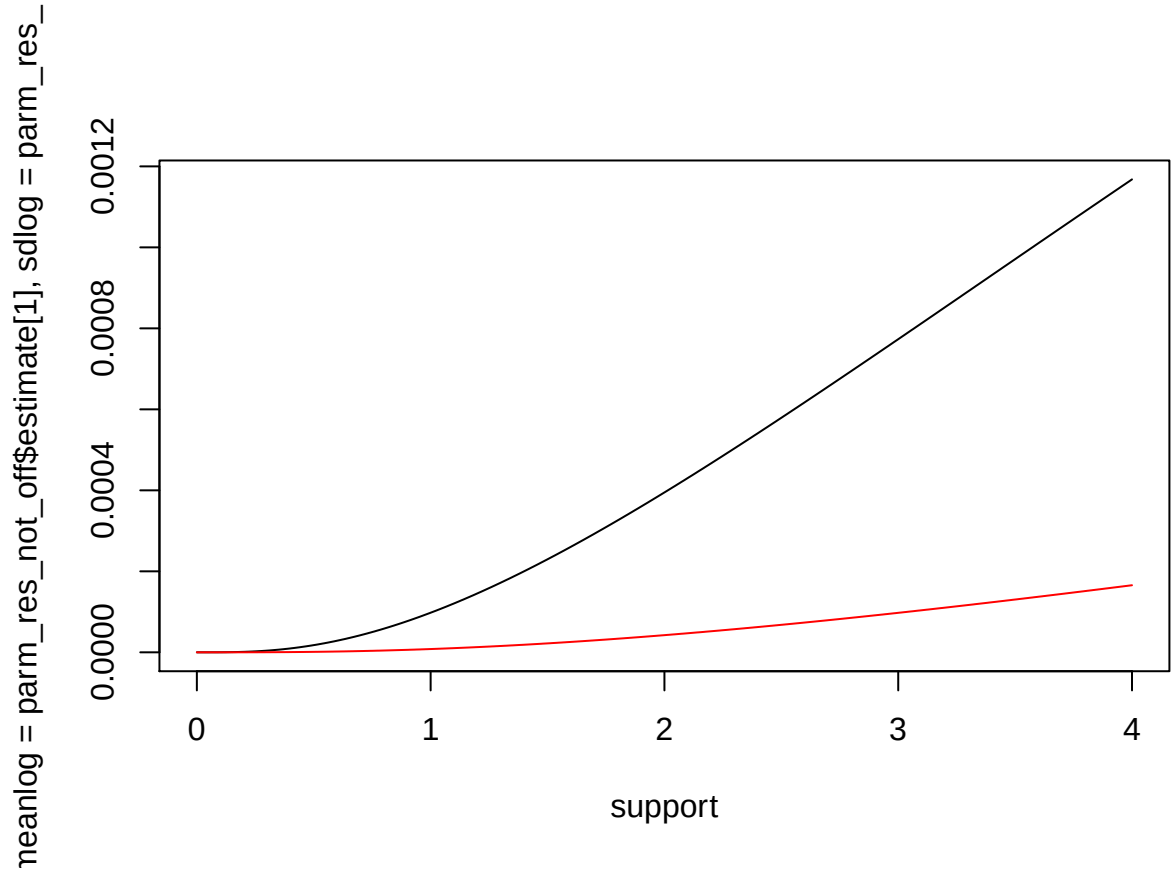
If I use a random seed of 1 in line 111, a random seed of 2 in line 117, and use the rate parameter of 1.3 in line 122, I get the exact same censoring rate as in the original study, which is very strange.

Generating example data set from the Log-normal distribution

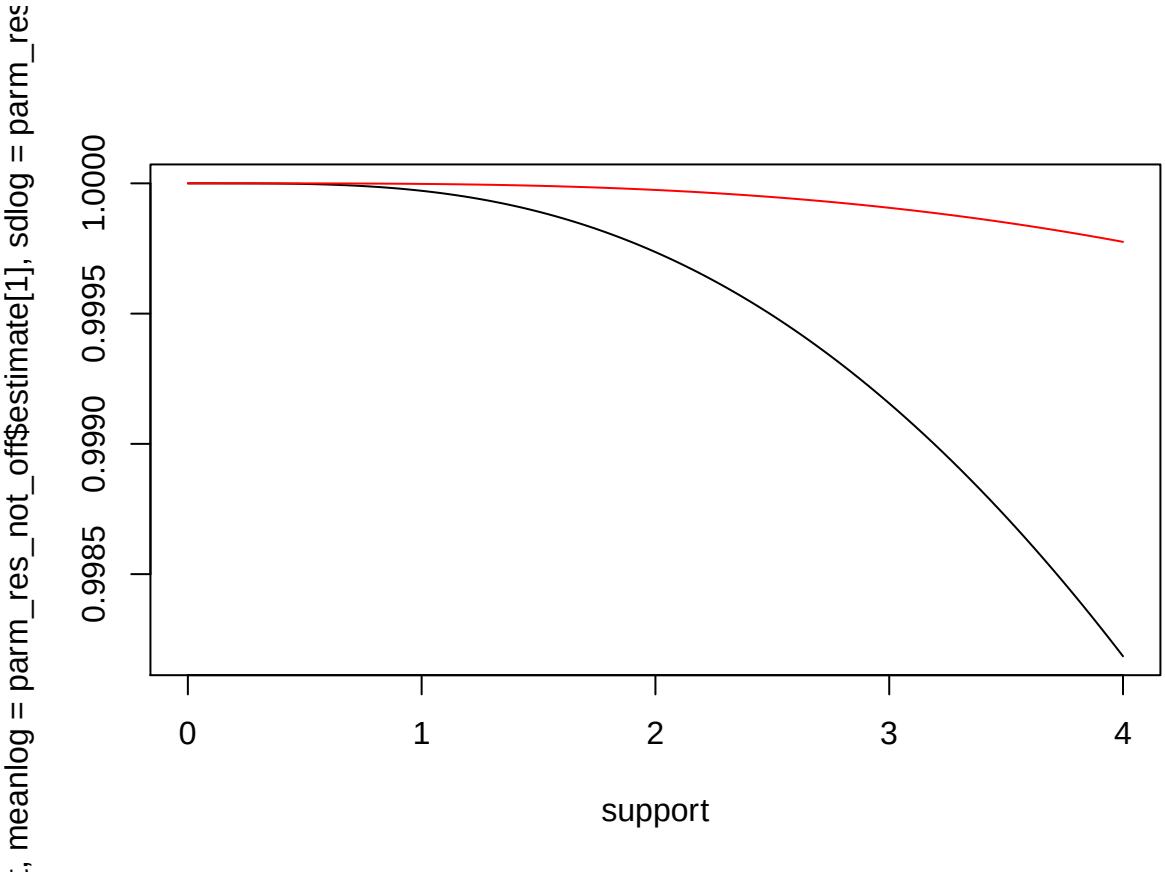
```
set.seed(1)

parm_res_not_off <- fitdist(rhare(100000, cov = rep(0, nphm_hare$ncov), nphm_hare), distr ="lnorm")

set.seed(1)

parm_res_off <- fitdist(rhare(100000, cov = c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0),
                            nphm_hare), distr ="lnorm")
```

```
support <- seq(0, 4, length.out = 1000)
```

```r
plot(support, dlnorm(support, meanlog = parm_res_not_off$estimate[1],
     sdlog = parm_res_not_off$estimate[2]), type = "l")

lines(support, dlnorm(support, meanlog = parm_res_off$estimate[1],
     sdlog = parm_res_off$estimate[2]), type = "l", col = "red")
```




```r
plot(support, 1 - plnorm(support, meanlog = parm_res_not_off$estimate[1],
     sdlog = parm_res_not_off$estimate[2]), type = "l")

lines(support, 1 - plnorm(support, meanlog = parm_res_off$estimate[1],
     sdlog = parm_res_off$estimate[2]), type = "l", col = "red")
```




```
set.seed(2)

sim_mat_lnorm <- simulate_lnorm(x, fcts_select,
                                params_not_off = list(meanlog = parm_res_not_off$estimate[1],
                                                      sdlog = parm_res_not_off$estimate[2]),
                                params_off = list(meanlog = parm_res_off$estimate[1],
                                                  sdlog = parm_res_off$estimate[2]),
                                FUN = rexp, 1.3)

summary(sim_mat_lnorm$time)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.000143 0.002715 0.002826 0.280423 0.313414 4.047476
```

```
# censoring rate for the simulated data set
1 - mean(sim_mat_lnorm$cid)
```

```
## [1] 0.3712015
```

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
```

Log-normal distributed survival times

```
phm_sim_mat_lnorm <- coxph(Surv(time, cid) ~ ., data = sim_mat_lnorm)

summary(phm_sim_mat_lnorm)
```

```
## Call:
## coxph(formula = Surv(time, cid) ~ ., data = sim_mat_lnorm)
##
##   n= 2139, number of events= 1345
##
##                  coef  exp(coef)   se(coef)       z Pr(>|z|)
## trtZDV.ddi  5.753e-02  1.059e+00  7.787e-02   0.739  0.46007
## trtZDV.ZAL  1.152e-01  1.122e+00  7.809e-02   1.476  0.14005
## trtddi      4.431e-02  1.045e+00  7.729e-02   0.573  0.56647
## age        -1.050e-03  9.990e-01  3.266e-03  -0.322  0.74778
## wtkg        8.136e-04  1.001e+00  2.150e-03   0.378  0.70512
## hemo1      -6.261e-02  9.393e-01  1.220e-01  -0.513  0.60770
## drugs1      9.193e-02  1.096e+00  8.162e-02   1.126  0.26004
## karnof     -2.088e-03  9.979e-01  4.872e-03  -0.429  0.66824
## oprior1    -1.692e-01  8.443e-01  2.947e-01  -0.574  0.56590
## preanti    -4.874e-03  9.951e-01  1.731e-04 -28.155  < 2e-16 ***
## race1      -1.062e-02  9.894e-01  6.172e-02  -0.172  0.86335
## gender1    -4.631e-02  9.547e-01  7.824e-02  -0.592  0.55390
## symptom1    2.767e-02  1.028e+00  7.567e-02   0.366  0.71456
## offtrt1     2.510e-03  1.003e+00  5.716e-02   0.044  0.96497
## cd40        5.914e-04  1.001e+00  2.243e-04   2.637  0.00836 **
## cd80        2.904e-05  1.000e+00  5.548e-05   0.523  0.60068
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##            exp(coef) exp(-coef) lower .95 upper .95
## trtZDV.ddi    1.0592     0.9441    0.9093    1.2339
## trtZDV.ZAL    1.1221     0.8912    0.9629    1.3077
## trtddi        1.0453     0.9567    0.8984    1.2163
## age           0.9990     1.0011    0.9926    1.0054
## wtkg          1.0008     0.9992    0.9966    1.0050
## hemo1         0.9393     1.0646    0.7396    1.1929
## drugs1        1.0963     0.9122    0.9342    1.2865
## karnof        0.9979     1.0021    0.9884    1.0075
## oprior1       0.8443     1.1844    0.4738    1.5045
## preanti       0.9951     1.0049    0.9948    0.9955
## race1         0.9894     1.0107    0.8767    1.1167
## gender1       0.9547     1.0474    0.8190    1.1130
## symptom1      1.0281     0.9727    0.8864    1.1924
## offtrt1       1.0025     0.9975    0.8963    1.1214
## cd40          1.0006     0.9994    1.0002    1.0010
## cd80          1.0000     1.0000    0.9999    1.0001
##
## Concordance= 0.766  (se = 0.007 )
```

```
## Rsquare= 0.631   (max possible= 1 )
## Likelihood ratio test= 2134  on 16 df,   p=<2e-16
## Wald test            = 874.3  on 16 df,   p=<2e-16
## Score (logrank) test = 1430  on 16 df,   p=<2e-16
```

```
print(cox.zph(phm_sim_mat_lnorm))
```

```
##                 rho    chisq        p
## trtZDV.ddi -0.01039 1.46e-01 7.03e-01
## trtZDV.ZAL  0.00564 4.33e-02 8.35e-01
## trtddi      0.00116 1.83e-03 9.66e-01
## age        -0.00456 2.67e-02 8.70e-01
## wtkg        0.03756 2.14e+00 1.44e-01
## hemo1      -0.05092 3.48e+00 6.20e-02
## drugs1      0.00935 1.21e-01 7.28e-01
## karnof      0.00197 5.26e-03 9.42e-01
## oprior1    -0.00331 1.50e-02 9.03e-01
## preanti    -0.26535 8.14e+01 1.86e-19
## race1       0.01531 3.22e-01 5.71e-01
## gender1    -0.00193 5.24e-03 9.42e-01
## symptom1    0.00532 3.86e-02 8.44e-01
## offtrt1    -0.00505 3.39e-02 8.54e-01
## cd40        0.01942 4.80e-01 4.88e-01
## cd80        0.05169 3.22e+00 7.26e-02
## GLOBAL           NA 1.04e+02 7.61e-15
```

Preanti instead of offtrt was determined to be the covariate violating the proportional hazards assumption.

Weibull model (to compare with Coxph), (Weibull distributed survival times)

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.

```
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 ($p < 2e\text{-}16$) 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

Weibull distributed survival times

```
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 -71); Convergence for 71th
## 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

## 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 -72); Convergence for 72th
## lambda value not reached after maxit=100000 iterations; solutions for
## larger lambdas returned

## Warning: from glmnet Fortran code (error code -75); Convergence for 75th
## lambda value not reached after maxit=100000 iterations; solutions for
## larger lambdas returned

## Warning: from glmnet Fortran code (error code -75); Convergence for 75th
## 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
```

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

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

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## trtZDV.ddi  0.007038608
## trtZDV.ZAL -0.114983825
## trtddi      .
## age         0.001021004
## wtkg        .
## hemo1       0.004250722
## drugs1     -0.020800623
## karnof      0.002117495
## oprior1     0.022070064
## preanti     0.063986146
## race1      -0.012875863
## gender1     .
## symptom1   -0.167332114
## offtrt1    -0.093540660
## cd40       -0.080090874
## cd80       -0.002662061
```

Log-normal distributed survival times

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

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

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## trtZDV.ddi  .
## trtZDV.ZAL  .
## trtddi      .
## age         .
## wtkg        .
## hemo1       .
## drugs1      .
## karnof      .
## oprior1     .
## preanti    -0.0039227608
```

```
## race1       .
## gender1     .
## symptom1    .
## offtrt1     .
## cd40        0.0003075028
## cd80        .
```

PH HARE Simulation

Weibull distributed survival times

```
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)
```

```
##    dim1 knot1 dim2 knot2          beta           SE
## 1     0     0   NA    NA  1.316615e+02 5.084137e+00
## 2    10     0   NA    NA  1.884180e-01 7.361213e-03
## 3     0    17   NA    NA -3.959986e+01 2.273290e+00
## 4     0     9   NA    NA -2.955539e+01 5.360675e+00
## 5    15     0   NA    NA -2.347316e-01 9.181993e-03
## 6     2     0   NA    NA -3.302003e-01 1.092426e-01
## 7     0    12   NA    NA -1.546355e+01 4.676063e+00
## 8     0     7   NA    NA -5.013321e+01 9.059121e+00
## 9     0     1   NA    NA -5.951434e+02 3.482019e+01
## 10   16     0   NA    NA -8.007272e-03 3.278145e-04
## 11    0    14   NA    NA -1.796385e+01 1.629462e+00
## 12    0     5   NA    NA -8.645924e+01 1.433438e+01
## 13    0    15   NA    NA -1.700083e+01 1.493717e+00
## 14    0     3   NA    NA -2.061715e+02 1.885999e+01
## 15    0     4   NA    NA -1.101233e+02 1.670802e+01
## 16    0    13   NA    NA -2.416604e+01 2.560932e+00
## 17    0     8   NA    NA -4.988366e+01 6.560408e+00
## 18    0     6   NA    NA -7.073175e+01 1.186131e+01
## 19    0     2   NA    NA -3.066214e+02 3.209411e+01
## 20    0    11   NA    NA -3.033331e+01 4.361621e+00
## 21    0    10   NA    NA -4.695981e+01 4.362436e+00
## 22   10     1   NA    NA -1.210437e-01 5.530689e-03
## 23    0    16   NA    NA -8.636981e+00 2.193363e+00
## 24   14     0   NA    NA -3.562290e-01 9.837209e-02
```

```
# cat("\n")
#
# fcts[fcts[,1] != 0 & fcts[,2] == 0 & is.na(fcts[,3]),]
```

Log-normal distributed survival times

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

```r
# extracting the coefficients for basis functions
# that do not correspond to time, knots, and/or tensor products
(fcts <- phm_hare_lnorm$fcts)
```

```
##    dim1 knot1 dim2 knot2          beta           SE
## 1     0     0   NA    NA -5.847979e+00 7.557452e-01
## 2     0     7   NA    NA  3.320887e+02 8.749183e+01
## 3     0     4   NA    NA -1.673973e+03 3.986092e+03
## 4     0     6   NA    NA  3.775339e+04 2.560339e+03
## 5     0     2   NA    NA  2.238683e+03 4.153657e+03
## 6     0     5   NA    NA -3.223535e+04 3.349193e+03
## 7     0     8   NA    NA  4.574372e+01 9.283324e+00
## 8     6     0   NA    NA -5.808158e-01 1.183483e-01
## 9    11     0   NA    NA  2.683663e-01 5.822841e-02
## 10    9     0   NA    NA -1.215469e+00 2.900502e-01
## 11    0     1   NA    NA -1.858771e+04 4.753587e+03
## 12    0     3   NA    NA -1.162188e+04 3.541454e+03
```

```r
# cat("\n")
#
# fcts[fcts[,1] != 0 & fcts[,2] == 0 & is.na(fcts[,3]),]
```

non-PH HARE Simulation

Weibull distributed survival times

```r
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)
```

```
##    dim1 knot1 dim2 knot2          beta           SE
## 1     0     0   NA    NA  1.781219e+02 5.910650e+00
## 2    10     0   NA    NA  2.993305e-01 1.334140e-02
## 3     0     2   NA    NA -7.758294e+01 2.623420e+00
## 4     0     2   10     0 -9.409264e-02 7.053682e-03
## 5    15     0   NA    NA -2.627804e-01 9.145641e-03
## 6    10     0   15     0  1.230676e-04 4.639792e-06
## 7    10     1   NA    NA -1.357698e-02 2.115474e-03
## 8    15     1   NA    NA -9.799035e-02 5.369573e-03
## 9    16     0   NA    NA -9.501459e-03 3.553189e-04
## 10   10     2   NA    NA  1.166182e-01 1.294319e-02
## 11   10     3   NA    NA -1.400758e-02 2.540694e-03
## 12    0     1   NA    NA -2.685507e+01 2.476446e+00
## 13    0     1   16     0  1.666146e-02 9.489371e-04
## 14   10     4   NA    NA -3.106394e-02 7.091992e-03
## 15   15     2   NA    NA -9.092884e-02 4.544857e-03
## 16   15     3   NA    NA -4.786413e-02 3.461272e-03
## 17   10     5   NA    NA  1.315205e-01 1.170438e-02
## 18    0     2   10     2 -2.938791e-02 7.845664e-03
```

```
## 19   10     0   15     1  6.797377e-05 4.185657e-06
## 20   10     6   NA    NA  2.166350e-02 2.953596e-03
## 21    0     2   10     5 -4.724067e-02 5.285055e-03
## 22   10     9   NA    NA -7.683764e-02 1.135178e-02
## 23   10     7   NA    NA -2.148029e+00 2.081067e-01
## 24    0     2   10     7  7.234727e-01 7.029549e-02
## 25   10     8   NA    NA -2.565901e-02 8.476820e-03
```

```
# cat("\n")
#
# fcts[fcts[,1] != 0 & fcts[,2] == 0 & is.na(fcts[,3]),]
```

Log-normal distributed survival times

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

nphm_hare_sim_lnorm$fcts
```

```
##    dim1 knot1 dim2 knot2          beta           SE
## 1     0     0   NA    NA -5.844732e+00 7.557379e-01
## 2     0     7   NA    NA  3.320813e+02 8.749193e+01
## 3     0     4   NA    NA -1.654703e+03 3.986033e+03
## 4     0     6   NA    NA  3.775240e+04 2.560331e+03
## 5     0     2   NA    NA  2.592338e+03 4.147498e+03
## 6     0     5   NA    NA -3.224150e+04 3.349178e+03
## 7     0     8   NA    NA  4.574198e+01 9.283269e+00
## 8     6     0   NA    NA -5.816211e-01 1.183497e-01
## 9    11     0   NA    NA  2.685591e-01 5.822879e-02
## 10    9     0   NA    NA -1.456997e+00 3.192258e-01
## 11    0     2    9     0  1.520931e+04 3.762290e+03
## 12    0     1   NA    NA -1.972341e+04 4.785501e+03
## 13    0     3   NA    NA -1.165758e+04 3.541299e+03
```

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

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

```r
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)
```

```
}
```

Weibull distributed survival times

I tried to do 500 simulate_regression iterations, but it was too much for R to handle. I do 100 simulations instead.

```
# 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()
```

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.

Proportion of times each covariate is selected across the four models fitted on simulated survival times from the Weibull distribution (this will be Table 5a in the manuscript)

```
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
```

Log-normal distributed survival times

```
# set.seed(636)
#
# tic()
# sims <- simulate_regression(N = 100, p = 16, simulate_fun = simulate_lnorm, x, fcts_select,
#                             params_not_off = list(meanlog = parm_res_not_off$estimate[1],
#                                                   sdlog = parm_res_not_off$estimate[2]),
#                              params_off = list(meanlog = parm_res_off$estimate[1],
#                                                sdlog = parm_res_off$estimate[2]),
#                             FUN = rexp, rate = 1.3)
#
# save(sims, file = "sims_lnorm.RData")
# toc()
```

This simulation process took 417.356 seconds (~ 7 minutes). This error, "Convergence problems... stopping addition", occured 72 times.

```
load("sims_lnorm.RData")

num_no_cov <- 0

for (k in 1:100) {

  if (sum(!is.na(sims[,3:4, k])) == 0)
    num_no_cov <- num_no_cov + 1

}

num_no_cov
```

```
## [1] 36
```

There were 36 simulations out of 100 in which Models 3 and 4 did not select any covariates. This must be related to the 72 instances of the "Convergence problems... stopping addition" error.

`Proportion of times each covariate is selected across the four models fitted on simulated survival times from the Log-normal distribution (this will be Table 5c in the manuscript)`

```
load("sims_lnorm.RData")

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

for (i in 1:16) {

  for (j in 1:4) {

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

```
  }

}

prop_lnorm
```

```
##       [,1] [,2] [,3] [,4]
##  [1,] 0.01 0.00 0.00 0.00
##  [2,] 0.02 0.00 0.00 0.00
##  [3,] 0.04 0.02 0.00 0.00
##  [4,] 0.03 0.03 0.01 0.01
##  [5,] 0.08 0.07 0.00 0.00
##  [6,] 0.03 0.00 0.64 0.64
##  [7,] 0.06 0.04 0.00 0.00
##  [8,] 0.02 0.00 0.00 0.00
##  [9,] 0.00 0.00 0.64 0.64
## [10,] 1.00 1.00 0.00 0.00
## [11,] 0.02 0.01 0.64 0.62
## [12,] 0.09 0.01 0.00 0.00
## [13,] 0.03 0.01 0.00 0.00
## [14,] 0.05 0.02 0.01 0.00
## [15,] 0.94 1.00 0.00 0.00
## [16,] 0.10 0.07 0.00 0.00
```

Update to do later: track how many times the tensor product with time (indicating violation of proportional hazards) was selected for a specific covariate (ideally Offtrt)