Cox simulations

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.0.5
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.0     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
```

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(magrittr)
```

```
##
## Attaching package: 'magrittr'
```

```
## The following object is masked from 'package:purrr':
##
##     set_names
```

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library(purrr)
# library(coxphw)
library(survival)
```

```
## Warning: package 'survival' was built under R version 4.0.5
```

Weibull simple linear

Function to generate simulated data for Weibull with simple linear predictors. Randomly sets some to be categorical and some to be normal.

$\beta \sim \mathcal{N}_{n_\beta}(0, \sigma)$, $x_i \sim \mathcal{N}_{n_\beta}(0, 1)$, $y_i \sim \text{Weib}(c, \exp(x_i^T \beta / c))$, $c \sim \Gamma(1)$.

```r
rweibull_v <- Vectorize(rweibull, vectorize.args = "scale")

weibull_simple_linear_sim <- function(n_beta, prop_cat, obs, censor_prop, sigma = 1){
  betas <- rnorm(n_beta, sd = sigma)
  print(betas)
  X_norm <- rnorm(obs*floor(n_beta*prop_cat))
  X_norm <- matrix(X_norm, nrow = obs, ncol = floor(n_beta*prop_cat))
  X_cat <- as.numeric(rbernoulli(obs*ceiling(n_beta*(1-prop_cat))))
  X_cat <- matrix(X_cat, nrow = obs, ncol = ceiling(n_beta*(1-prop_cat)))
  X <- cbind(X_norm,X_cat)

  c = rgamma(1,1)

  lin_pred <- X %*% betas

  sim_data <- as.data.frame(X)
  sim_data[["lin_pred"]] <- lin_pred
  sim_data %<>%
    mutate(y = rweibull_v(1,c, scale = exp(-lin_pred/c)))

  dropout_prop <- runif(1,max = 0.5)*censor_prop
  sim_data[["dropout"]] <- rbernoulli(obs,dropout_prop)
  sim_data %<>%
    mutate(y_dropout = ifelse(dropout,runif(1,max = y),y))
  max_time <- quantile(sim_data[["y_dropout"]], probs = 1 - censor_prop)
  sim_data %<>%
    mutate(censor = y_dropout > max_time,
           y_censor = pmin(y_dropout, max_time),
           event = !(dropout | censor))

  return(sim_data)
}
```

Demonstration of function:

```r
weibull_simple_linear_sim(3,0.5,10,0.7)
```

```
## [1] 0.3278641 1.1565396 0.1238887

##            V1 V2 V3   lin_pred            y dropout    y_dropout censor
## 1   0.2754768  0  1  0.2142076 1.277911e-01    TRUE 4.564503e-02  FALSE
## 2   1.8014582  0  0  0.5906334 1.332444e+01   FALSE 1.332444e+01   TRUE
## 3   0.3612098  1  1  1.3988560 1.656992e-03   FALSE 1.656992e-03  FALSE
## 4  -1.5227527  0  0 -0.4992559 2.539290e+02   FALSE 2.539290e+02   TRUE
## 5   0.5357536  0  0  0.1756544 8.031433e-02   FALSE 8.031433e-02   TRUE
## 6  -2.1407308  0  1 -0.5779800 2.994662e-02    TRUE 4.564503e-02  FALSE
## 7   0.2738387  0  1  0.2136705 4.413429e+00   FALSE 4.413429e+00   TRUE
## 8  -1.4257110  0  0 -0.4674394 4.274257e-02    TRUE 4.564503e-02  FALSE
## 9  -0.5276738  1  0  0.9835343 3.024508e-01   FALSE 3.024508e-01   TRUE
## 10  1.4920913  0  1  0.6130918 4.754414e+00   FALSE 4.754414e+00   TRUE
##       y_censor event
## 1  0.045645029 FALSE
## 2  0.045645029 FALSE
```

```
## 3  0.001656992  TRUE
## 4  0.045645029 FALSE
## 5  0.045645029 FALSE
## 6  0.045645029 FALSE
## 7  0.045645029 FALSE
## 8  0.045645029 FALSE
## 9  0.045645029 FALSE
## 10 0.045645029 FALSE
```

Single sample test

Now let us draw a single sample to test out Cox regression:

```
sample <- weibull_simple_linear_sim(10,0.5,1500,0.9)
```

```
##  [1]  0.08526672 -1.55426897 -1.28397268 -1.10421655 -0.40564770 -0.01938524
##  [7]  0.49798656 -0.74223749  1.13998318 -0.22529847
```

```
# Use first 1000 samples as the cohort set
cohort <- head(sample,1000)
head(cohort)
```

```
##            V1          V2         V3         V4          V5 V6 V7 V8 V9 V10
## 1 -0.05551053 -0.99100854 -0.5058857  1.5180596 -1.45784106  0  0  0  0   1
## 2  0.52961721 -0.02750987 -0.6453633 -0.9551900  0.03710833  0  0  1  0   1
## 3 -0.94233733  0.57801724 -0.1107261  1.3248814  0.85199667  1  1  0  1   0
## 4 -0.87403145  1.26117880 -1.3962816  0.1152531 -0.59391969  1  0  1  0   1
## 5  0.97954943  0.66539499 -1.0934693  0.9292984  0.38398423  1  1  0  1   0
## 6  0.73040306 -1.68788474 -1.4092175 -1.6582680 -0.11807060  1  1  0  1   0
##     lin_pred           y dropout   y_dropout censor    y_censor event
## 1  0.8749088 0.319299130   FALSE 0.319299130   TRUE 0.063615680 FALSE
## 2  0.9886930 0.739655556   FALSE 0.739655556   TRUE 0.063615680 FALSE
## 3 -1.0265568 1.045166305    TRUE 0.089002028   TRUE 0.063615680 FALSE
## 4 -1.1152128 0.624938362   FALSE 0.624938362   TRUE 0.063615680 FALSE
## 5  0.8899804 0.599391020    TRUE 0.089002028   TRUE 0.063615680 FALSE
## 6  7.9926693 0.003748073   FALSE 0.003748073  FALSE 0.003748073  TRUE
```

Extracting cases:

```
cases <- cohort %>%
  filter(event)
head(cases)
```

```
##           V1          V2         V3         V4         V5 V6 V7 V8 V9 V10
## 1  0.7304031 -1.68788474 -1.4092175 -1.6582680 -0.1180706  1  1  0  1   0
## 2  2.1556630 -1.31920395 -1.1547011 -0.5112055  0.9102995  1  0  0  1   1
## 3 -0.5924869 -1.11976807 -1.3032191 -0.5716566  0.5306993  0  0  0  1   0
## 4  1.4087126  0.09003655 -1.4167097 -0.8067553 -0.8442811  1  1  0  1   0
## 5  1.5079040  0.22716877  1.0417886 -0.1932439 -0.9724709  1  0  1  1   0
## 6 -0.4839085  0.09906609  0.8249988 -0.6971709 -1.4109646  1  1  0  1   0
```

```
##     lin_pred           y dropout   y_dropout censor    y_censor event
## 1  7.9926693 0.003748073   FALSE 0.003748073  FALSE 0.003748073  TRUE
## 2  4.8073288 0.039244184   FALSE 0.039244184  FALSE 0.039244184  TRUE
## 3  4.9191380 0.024922326   FALSE 0.024922326  FALSE 0.024922326  TRUE
## 4  4.6510895 0.057881689   FALSE 0.057881689  FALSE 0.057881689  TRUE
## 5 -0.5759114 0.038140061   FALSE 0.038140061  FALSE 0.038140061  TRUE
## 6  1.7062541 0.052112116   FALSE 0.052112116  FALSE 0.052112116  TRUE
```

Selecting subcohort:

```
subcohort <- cohort %>%
  slice_sample(n = 109)
head(subcohort)
```

```
##           V1          V2         V3         V4         V5 V6 V7 V8 V9 V10
## 1  0.4756711 -0.53712026 -1.1109155 -0.2041377 -0.6561525  0  0  1  1   0
## 2 -1.9794090  0.05745785 -0.2221177 -0.8000982  1.6795200  0  1  1  1   1
## 3  0.1435004  0.38789220 -0.2861339  0.4988161 -0.7251543  1  1  0  0   0
## 4 -0.2941718  0.92346361 -1.1644179 -1.1511134  0.7499303  1  1  0  0   0
## 5 -1.0315078  0.38817540  2.1125619 -0.2661104  0.7583514  1  0  1  0   1
## 6  2.0928788  1.09760534 -1.1216599  0.4346984  0.1206154  0  0  0  1   0
##      lin_pred          y dropout  y_dropout censor   y_censor event
## 1  3.19109804 0.07610872   FALSE 0.07610872   TRUE 0.06361568 FALSE
## 2  0.89973247 0.78424965    TRUE 0.08900203   TRUE 0.06361568 FALSE
## 3 -0.00130747 0.08059112   FALSE 0.08059112   TRUE 0.06361568 FALSE
## 4  1.48015914 0.08962469   FALSE 0.08962469   TRUE 0.06361568 FALSE
## 5 -4.40445517 9.60918820    TRUE 0.08900203   TRUE 0.06361568 FALSE
## 6  0.52371421 0.55767568   FALSE 0.55767568   TRUE 0.06361568 FALSE
```

The case-subcohort dataset:

```
case_subcohort <- rbind(cases,subcohort)
case_subcohort <- case_subcohort[,c(1:10,16:17)]
head(case_subcohort)
```

```
##           V1          V2         V3         V4         V5 V6 V7 V8 V9 V10
## 1  0.7304031 -1.68788474 -1.4092175 -1.6582680 -0.1180706  1  1  0  1   0
## 2  2.1556630 -1.31920395 -1.1547011 -0.5112055  0.9102995  1  0  0  1   1
## 3 -0.5924869 -1.11976807 -1.3032191 -0.5716566  0.5306993  0  0  0  1   0
## 4  1.4087126  0.09003655 -1.4167097 -0.8067553 -0.8442811  1  1  0  1   0
## 5  1.5079040  0.22716877  1.0417886 -0.1932439 -0.9724709  1  0  1  1   0
## 6 -0.4839085  0.09906609  0.8249988 -0.6971709 -1.4109646  1  1  0  1   0
##      y_censor event
## 1 0.003748073  TRUE
## 2 0.039244184  TRUE
## 3 0.024922326  TRUE
## 4 0.057881689  TRUE
## 5 0.038140061  TRUE
## 6 0.052112116  TRUE
```

Now to fit the model: