TMA4275 Lifetime Analysis
Obligatory Project 2, Spring 2022

Jim Totland

Problem 1

In this problem, we will study the data used in [2]. The article uses a regression approach to tire reliability analysis, and considers in particular the failure mode known as tread and belt separation (TBS).

a)

We repeat their analysis, by creating a Cox regression model in `R`, as is shown in the code below. We will refer to the resulting model as *model 1*.

```
# Load dataset
tire <- read.csv("path/to/dataset/tire.txt", sep="")

# Fit the model
cox.reg1 <- coxph(Surv(Survival, Censor) ~ ., data = tire)
```

Table 1 presents a summary of the resulting regression coefficients. We observe that Age has p-value of 0.14, which indicates that the age of the tire does not have a significant effect on TBS. Similarly, the end of belt 2 to buttress (EB2B) and the percent of carbon black (Carbon) have p-values 0.10 and 0.11, respectively, which indicates that they do not have a significant effect on TBS.

Explanatory variable	$\hat{\beta}_i$	SE($\hat{\beta}_i$)	z-value	Pr($>\|z\|$)
Age	2.12	1.45	1.46	0.14
Wedge	-10.3686	4.6836	-2.2138	0.0268
Inter	-11.3045	4.6923	-2.4092	0.0160
EB2B	-14.0517	8.5293	-1.6475	0.0995
Peel	-36.2065	13.8056	-2.6226	0.0087
Carbon	-54.8466	33.9221	-1.6168	0.1059
WxP	22.3442	8.9732	2.4901	0.0128

Table 1: A summary of the coefficients belonging to the Cox regression model.

We note that the variables Wedge, Inter, Peel and WxP have a p-value below 0.05, and therefore construct a model only including these covariates, which we refer to as *model 2*. The construction is shown in the following chunk of code.

```
# Fit model with significant covariates
cox.reg2 <- coxph(Surv(Survival, Censor) ~ Wedge + Inter + Peel + WxP, data = tire)
```

Table 2 presents a summary of the resulting parameter estimates.

We observe that the p-values of the coefficients all decrease. Figure 1 displays the estimated relative risk functions, $r(\hat{\beta}, x_i), i = 1, \dots 34$, for *model 1* and *model 2*. We have plotted the estimated risk functions on log-scale, to better be able to compare the models.

Explanatory variable	$\hat{\beta}_i$	SE($\hat{\beta}_i$)	z-value	Pr($>|z|$)
Wedge	-10.0722	4.1963	-2.4003	0.0164
Inter	-7.3310	2.7308	-2.6845	0.0073
Peel	-28.6773	10.5630	-2.7149	0.0066
WxP	19.1910	7.0885	2.7073	0.0068

Table 2: A summary of the coefficients belonging to the Cox regression model with only significant covariates (p-value < 0.05 in Table 1).

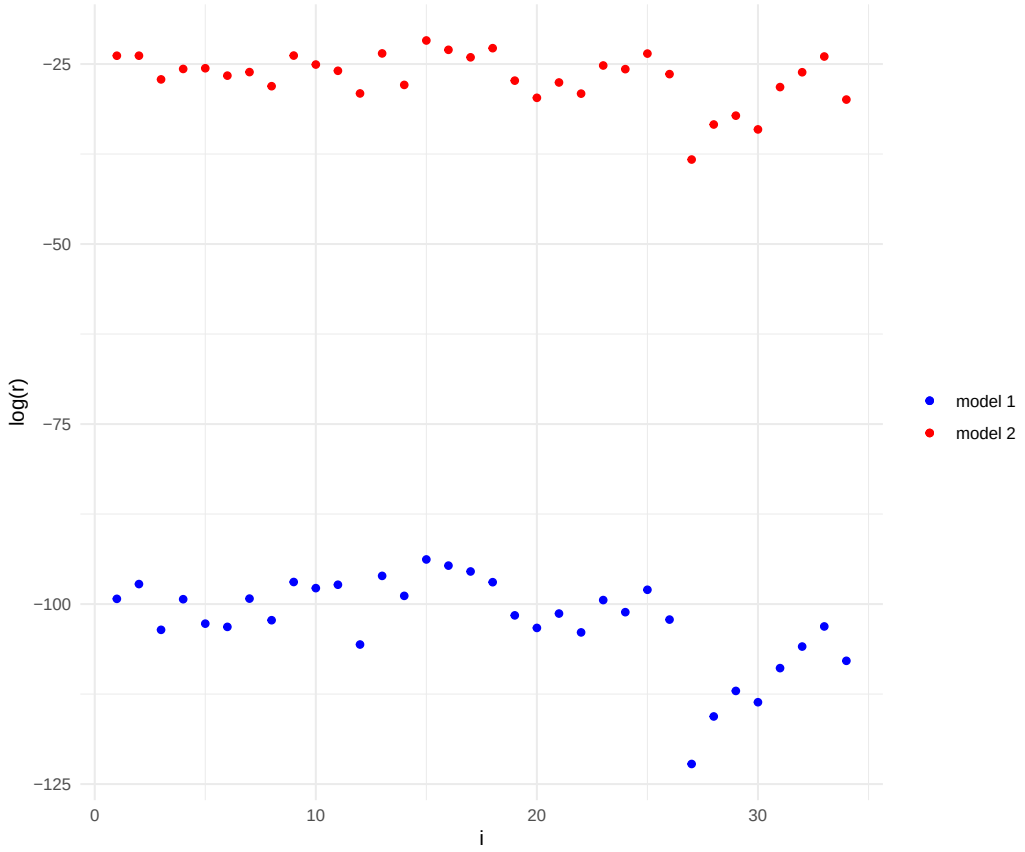


Figure 1: The estimated relative risk functions, $r(\hat{\beta}, x_i), i = 1, \ldots 34$, for *model 1* (blue) and *model 2* (red) plotted on log-scale.

The figure clearly illustrates that *model 2* has higher relative risk functions than *model 1* for all $i = 1, \ldots, 34$. It almost seems as if the estimated relative risk functions have been shifted upwards by a constant value.

b)

We want to estimate the integrated baseline hazard rate, $A_0(t)$, and use the Breslow estimator to achieve this. The estimator is given by

$$\widehat{A}_0(t) = \sum_{T_j \leq t} \frac{1}{\sum_{l \in \mathcal{R}_j} r(\widehat{\boldsymbol{\beta}}, \boldsymbol{x}_l)},$$

and we refer to [1] for a further explanation of the notation and the derivation. Code which computes this quantity and plots it for both model 1 and model 2 is given below.

```
# The Breslow estimator
breslow <- function(df, covariates, beta.vec){
  x.mat <- df[, covariates] # Matrix for covariates
  r.vec <- exp(as.matrix(x.mat) %*% beta.vec) # Vector of rel. risk function estimates
  max.r <- max(r.vec)
  r.vec.norm <- r.vec/max.r # Normalize the rel. risk functions to avoid numerical problems
  T.vec <- sort(df[df$Censor == 1, ]$Survival) # Event times
  A.vec <- rep(NA, length(T.vec)) # Estimated integrated baseline hazard

  for(i in 1:length(T.vec)){
    A.val <- NA
    if(i == 1){
      A.val <- 0
    } else{
      A.val <- A.vec[i - 1]
    }
    t <- T.vec[i]
    idx <- tire$Survival <= t
    denom <- sum(r.vec.norm[idx])
    A.val <- A.val + 1/denom
    A.vec[i] <- A.val
  }

  A.vec <- A.vec/max.r # Fix scale
  return(data.frame(t = T.vec, a = A.vec))
}

# Calculate Breslow estimator for model 1 and model 2 and plot them
b1 <- breslow(tire, covariates1, beta.hat1)
b2 <- breslow(tire, covariates2, beta.hat2)
ggplot(b1) + geom_step(aes(x = t, y = a, color = "model 1")) +
  geom_step(data = b2, aes(x = t, y = a, color = "model 2")) +
  scale_color_manual(name = " ", values = c("model 1" = "blue", "model 2" = "red")) +
  scale_y_continuous(trans = "log") + xlab("Study time, t") +
  ylab(expression(log(hat(A)[0](t)))) + theme_minimal()
```

Figure 2 displays the estimated integrated baseline hazards for model 1 and model 2.

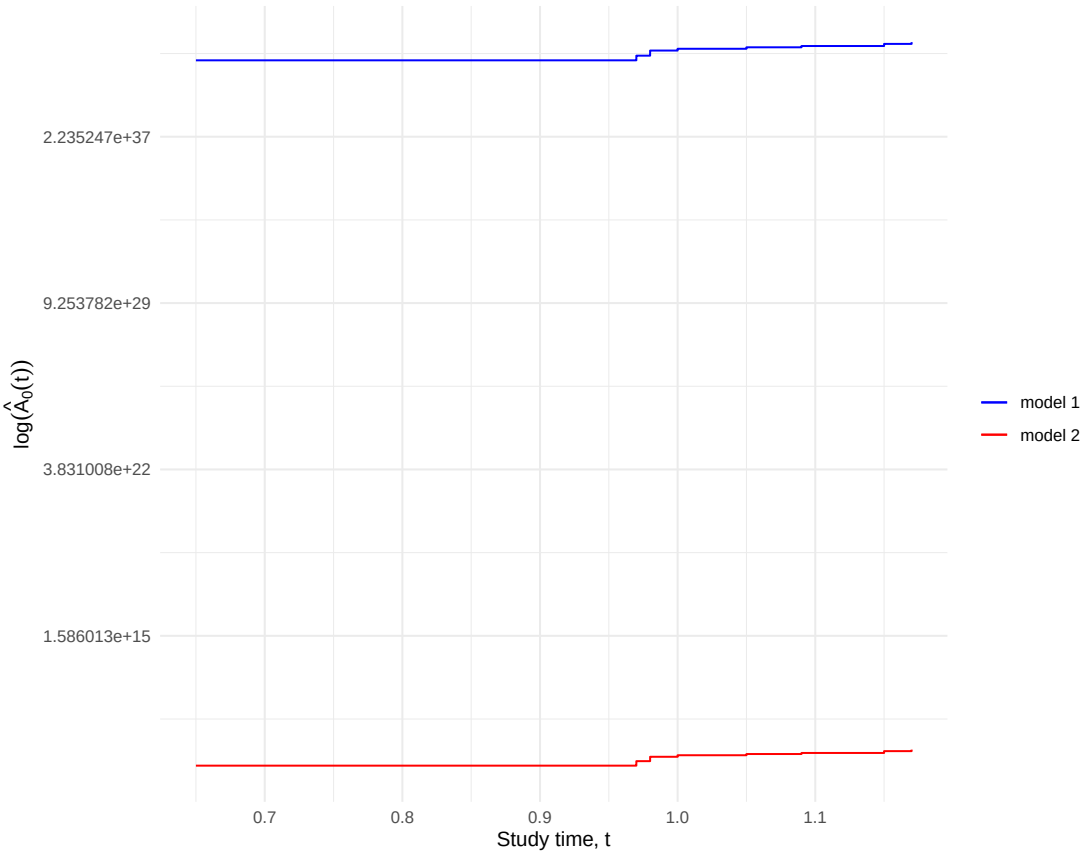


Figure 2: The Breslow estimator of the integrated baseline hazard rate for model 1 (blue) and model 2 (red). Note that the plot is on log-scale.

c)

Starting with the model with all potential covariates included, we will perform a step-wise elimination procedure, where in each step, the variable with the highest associated p-value above 0.05 will be removed. The elimination stops when all p-values are below 0.05. The procedure is performed below.

Step 1

A summary of the initial model is given in Table 1, from which we observe that Age has the highest p-value, so we remove it.

Step 2

A summary of the resulting model is given in Table 3, from which we observe that E2B2 has the highest p-value greater than 0.05. Thus, we remove this one.

Explanatory variable	$\hat{\beta}_i$	SE($\hat{\beta}_i$)	z-value	Pr($>\|z\|$)
Wedge	-10.8974	4.3314	-2.5159	0.0119
Inter	-9.5965	4.1337	-2.3215	0.0203
EB2B	-7.4374	5.7221	-1.2998	0.1937
Peel	-33.3937	12.9722	-2.5743	0.0100
Carbon	-53.8581	33.2377	-1.6204	0.1051
WxP	21.6246	8.4251	2.5667	0.0103

Table 3: A summary of the coefficients belonging to the Cox regression model in step 2 of the elimination procedure.

Step 3

A summary of the current model is given in Table 4, from which we observe that Carbon has the highest p-value greater than 0.05.

Explanatory variable	$\hat{\beta}_i$	SE($\hat{\beta}_i$)	z-value	Pr($>\|z\|$)
Wedge	-11.7396	4.3008	-2.7296	0.0063
Inter	-9.6917	3.4858	-2.7803	0.0054
Peel	-36.6313	11.5427	-3.1736	0.0015
Carbon	-57.6327	31.8021	-1.8122	0.0700
WxP	24.3448	7.8833	3.0882	0.0020

Table 4: A summary of the coefficients belonging to the Cox regression model in step 3 of the elimination procedure.

Step 4

By removing Carbon, we end up with *model 2* defined earlier. A summary of this model is presented in Table 2, from which we see that all coefficients have a p-value below 0.05, and we stop the elimination process. The estimated integrated baseline hazard rate of this model is displayed in Figure 2.

Problem 2

In this problem we will, based on the dataset in Problem 1, simulate new datasets by simulating new survival times and censoring times.

a)

To simulate the survival times, we let the integrated baseline hazard rate be equal to the integrated hazard rate of a Weibull distribution. We use the parametrization from [1] given by

$$f(t; b, k) = bt^{k-1} \exp\{-bt^k/k\},$$

such that

$$\alpha(t; b, k) = bt^{k-1}, \quad \text{and} \quad A_0(t) = \int_0^t bs^{k-1} \mathrm{d}s = \frac{b}{k} t^k.$$

We set $b = 4 \cdot 10^10$ and $k = 15$, such that $A_0(t)$ is reasonably similar to the Breslow estimator. The hazard rate for individual i then becomes

$$\begin{aligned} \alpha(t|\boldsymbol{x}_i) &= \alpha_0(t) r(\boldsymbol{\beta}, \boldsymbol{x}_i) \\ &= bt^{k-1} \cdot \exp\{\boldsymbol{\beta}^{\mathrm{T}} \boldsymbol{x}_i\} \\ &= \tilde{b}_i t^{k-1} \end{aligned}$$

To simulate the survival times, we use the `R` function **`rweibull`**, which takes the same shape parameter, k and a scale parameter, $\lambda = (k/b)^{1/k}$.

The censoring times are simulated from an exponential distribution, where we choose $\lambda = 1/1.2$, because 24 of the survival times (assuming the Weibull distribution) have an expected value of more than 1.2, meaning that approximately 24 survival times will be censored (compared to 23 in the original dataset). The function which simulates the new dataset is given below.

```
simulate.df <- function(df, model){
  n <- nrow(df)

  # Simulate survival times
  b <- 4e10
  k <- 15
  b.vec <- b * exp(as.matrix(df[,covariates]) %*% model$coefficients)
  lambda.vec <- (k/b.vec)^(1/k)
  sim.T <- rweibull(n, k, lambda.vec)

  # Simulate censoring times
  lambda.exp <- 1/1.2
  sim.C <- rexp(n, rate = lambda.exp)

  # Create simulated dataset
  censor <- (sim.C < sim.T)
  sim.df <- data.frame(df)
  sim.df$Survival <- sim.T
  sim.df$Survival <- sim.C[censor]
  sim.df$Censor <-  as.numeric(censor == 0)

  return(sim.df)
}
```

b)

Next, we create a function which automatically runs the backward elimination process described in 1**c)** for a given dataset. The code is given below. Note that we have included some additional elements, such as removing covariates with `NaN` values and stopping the elimination if there is only one covariate left.

```r
# Function which performs the elimination procedure
elim <- function(df){
  y <- "Surv(Survival, Censor)"
  covariates <- c("Age", "Wedge","Inter", "EB2B", "Peel", "Carbon", "WxP")
  p.above <- TRUE
  while(p.above){
    formula <- as.formula(paste(y, paste(covariates, collapse=" + "),
                                sep=" ~ ")) # Create formula
    model <- coxph(formula, data = df)
    s <- summary(model)
    p.vals <- s$coefficients[,5]
    if(sum(is.na(p.vals)) > 0){ # Remove covariates with NaN-values
      remove <- which(is.na(p.vals))
      covariates <- covariates[-remove]
    } else if(sum(p.vals > 0.05) == 0 || length(covariates) == 1){ # Stop elim
      p.above = FALSE
    } else{
      remove <- which(p.vals == max(p.vals))
      covariates <- covariates[-remove]
    }
  }
  return(model)
}
```

Now, we simulate a dataset with `simulate.df` and then use it as input to `elim`. We create a plot where we compare the estimated relative risk functions of the model which `elim` produces with the "true" relative risk functions which the dataset is based on. The result is displayed in Figure 3 and the code is given below.

```r
sim <-  simulate.df(tire, cox.reg2)
sim.df <- sim$sim.df
mod.elim <- elim(sim.df)

# Compare estimated relative risk function with the true rel. risk functions
r.est <- exp(as.matrix(tire[, names(mod.elim$coefficients)]) %*%
            mod.elim$coefficients)
r.true <- exp(as.matrix(tire[, names(cox.reg2$coefficients)]) %*% cox.reg$coefficients)

log.r.df <- data.frame(log.r.est = log(r.est), log.r.true = log(r.true), x = 1:34)
ggplot(log.r.df, aes(x = x)) + geom_point(aes(y = log.r.est, color = "Estimated r")) +
  geom_point(aes(y = log.r.true, color = "True r")) + xlab("i") + ylab("log(r)") +
  scale_color_manual(name = " ", values = c("Estimated r" = "blue", "True r" = "red")) +
  theme_minimal()
```

Finally, we consider the first three cases in the simulated dataset and compare the estimated integrated hazard rate to the "true" integrated hazard rate. Since the covariates are fixed, the estimated integrated hazard rate for case i is given by

$$\widehat{A}(t|\boldsymbol{x}_i) = r(\widehat{\boldsymbol{\beta}}, \boldsymbol{x}_i)\widehat{A}_0(t),$$

and the true integrated hazard rate is

$$A(t|\boldsymbol{x}_i) = \frac{\tilde{b}_i}{k}t^k.$$

.

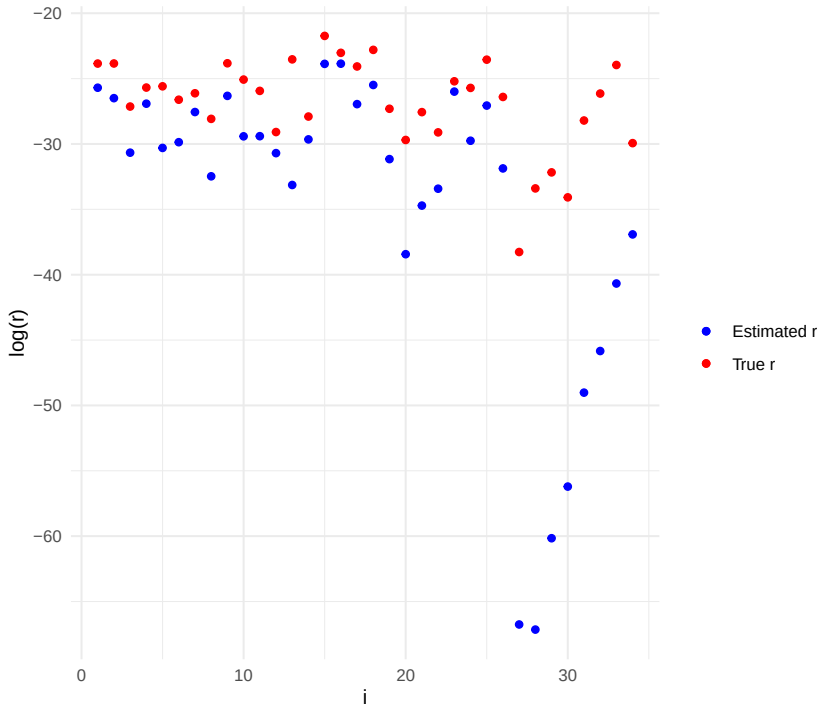


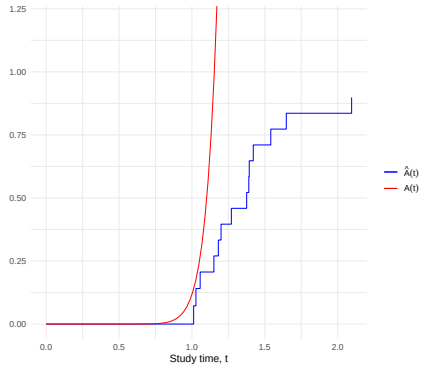
Figure 3: The estimated relative risk functions of the model produced by `elim` in blue and the "true" relative risk functions in red. Note that the plot is on log-scale.

These functions are plotted in Figure 4 for case 1, 2 and 3. The code is given below. The display reveals a poor fit, which gives us less confidence in the estimated results in Problem 1.

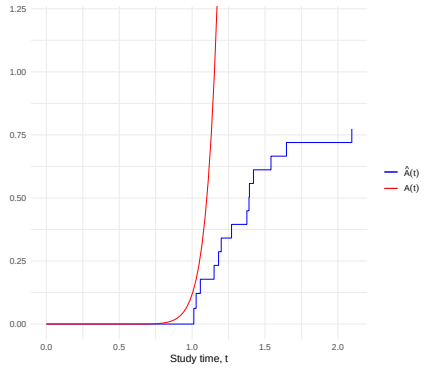
```
# Compare estimated integrated hazard rate with true integrated hazard rate
A0.hat <- breslow(sim.df, names(mod.elim$coefficients), mod.elim$coefficients)
r.vec <- exp(as.matrix(sim.df[, names(mod.elim$coefficients)]) %*% mod.elim$coefficients)
A.hat.df <- data.frame(a.hat1 = c(0, A0.hat$a * r.vec[1]), a.hat2 = c(0, A0.hat$a * r.vec[2]),
                       a.hat3 = c(0, A0.hat$a * r.vec[3]), t = c(0, A0.hat$t))

A.true <- function(x, idx, b.vec, k){
  return(b.vec[idx]/k * x^k)
}

# Plot for one of the cases

ggplot(A.hat.df, aes(x = t)) + geom_step(aes(y = a.hat2, color = "A.hat")) +
 geom_function(fun = A.true, args = list(idx = 2, b.vec = sim$b.vec, k = sim$k),
 aes(color ="A")) +
  scale_color_manual(name = " ", values = c("A.hat" = "blue", "A" = "red"),
                     labels = expression(hat(A)(t), A(t))) +
  coord_cartesian(ylim = c(0, 1.2)) + xlab("Study time, t") + ylab(" ") + theme_minimal()
```

c)

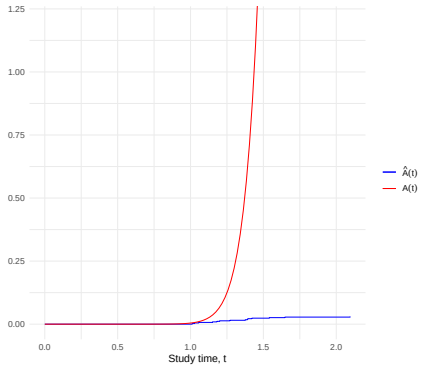
Now, we will repeat the process of simulating a dataset with `simulate.df` and then fitting a model to it with `elim`. The function which performs this is given below.



(a) Case 1



(b) Case 2



(c) Case 3

Figure 4: The estimated integrated hazard rate and the true integrated hazard rate plotted jointly for three cases.

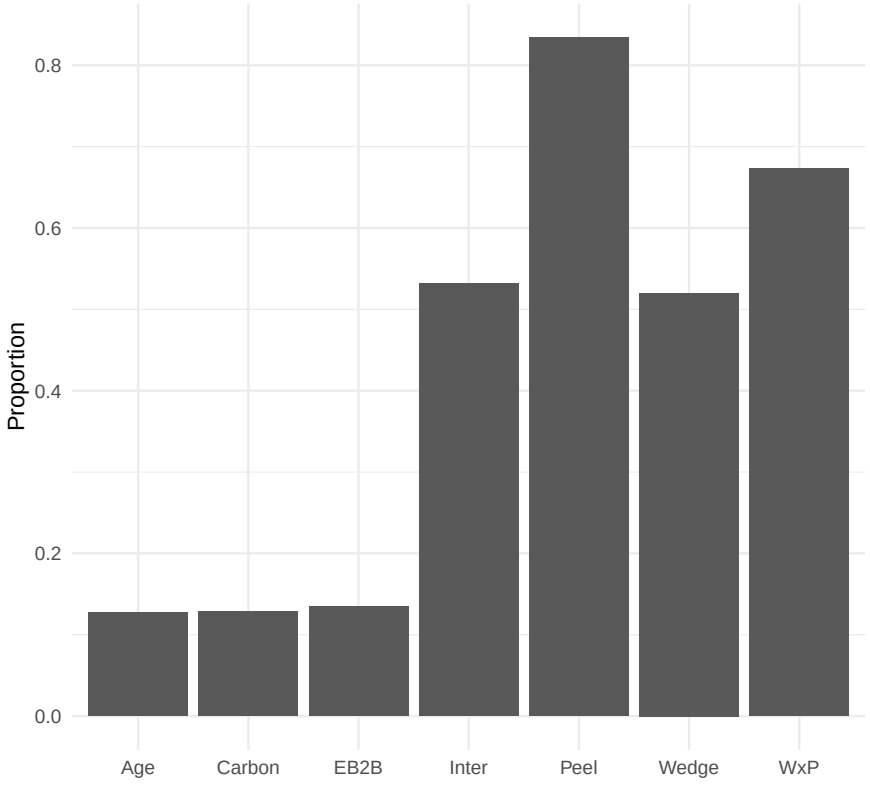


Figure 5: For each of the seven covariates, the proportion of the fitted models which contain the given covariate.

```
repeat.sim <- function(num.iter, plot=FALSE){
  all.covariates <- c("Age", "Wedge","Inter", "EB2B", "Peel", "Carbon", "WxP")
  counts <- rep(0, 7)
  p <- ggplot(environment = environment())
  for(i in 1:num.iter){
    sim.df <- simulate.df(tire, cox.reg2)$sim.df
    mod <- elim(sim.df)
    mod.covariates <- names(mod$coefficients)
    included <- all.covariates %in% mod.covariates
    counts[included] <- counts[included] + 1
    if(plot){
      b <- breslow(sim.df, mod.covariates, mod$coefficients)
      p <- p +  geom_step(data = data.frame(x = b$t, y = b$a),
                          aes(x = x, y = y), color = "blue", alpha = 0.5)
    }
  }
  return(list(p = p, counts = counts))
}
```

We use `num.iter = 1000`, i.e. simulate 1000 datasets and create 1000 models. Figure 5 displays the proportion of fitted models which contain each of the seven covariates.
The numbers seem reasonable, since the four covariates used to simulate the data have significantly higher proportions. The code used to generate the display is given below.

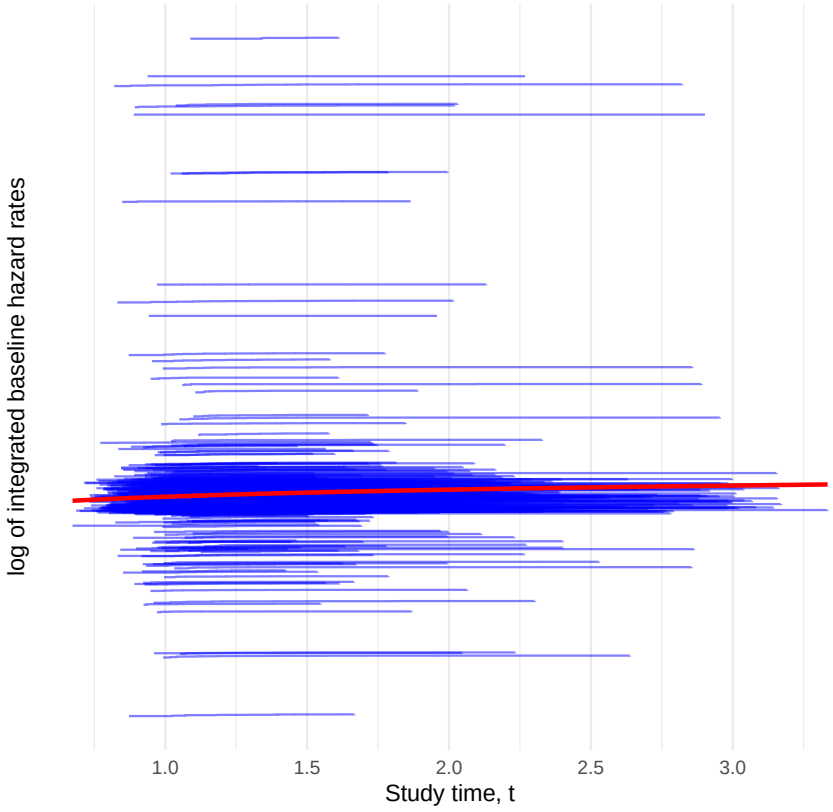


Figure 6: The log of the estimated integrated baseline hazard rates of the fitted models (blue) and the integrated baseline hazard rate which is used to generate the simulated datasets (red)

```
# Run simulations
num.iter <- 1000
c <- repeat.sim(num.iter, plot = TRUE)

# Plot proportions
ggplot(data= data.frame(prop = c$counts/num.iter,
                        x = c("Age", "Wedge","Inter", "EB2B", "Peel", "Carbon", "WxP")),
       aes(x=x, y=prop)) + geom_bar(stat="identity") + xlab(" ") + ylab("Proportion") +
  theme_minimal()
```

We also plot the estimated integrated baseline hazard rate for each of the fitted models, as well as the integrated baseline hazard rate used to generate the datasets. This is shown in Figure 6, and shows that there is high variability among the models' estimated integrated baseline hazard rates. Note that the plot is on log scale. The code to generate it is given below.

```
# Plot baseline hazard rates
A0 <- function(x, b, k){
  return(b/k * x^k)
}
c$p + geom_function(fun = A0, args = list(b = 4e10, k = 15), color = "red", size = 1) +
  scale_y_continuous(trans = "log") + ylab("log of integrated baseline hazard rates") +
  xlab("Study time, t") + theme_minimal()
```

End Note: There are some edge cases where `elim` fails, which I have not managed to resolve, and I get

many warnings when I run the code in this subproblem, due to the fact that the cox regression does not converge. Maybe my choice of k and b is not optimal or maybe there are errors in the code that I have not spotted.

References

[1] Odd Aalen, Ornulf Borgan, and Hakon Gjessing. *Survival and Event History Analysis.* Springer, New York, NY, 2008.

[2] V.V Krivtsov, D.E Tananko, and T.P Davis. Regression approach to tire reliability analysis. *Reliability Engineering System Safety*, 78(3):267–273, 2002.