TMA4275 Lifetime Analysis
Obligatory Project 2, Spring 2022

Jim Totland

Problem 1

In this problem we will consider the dataset ”pbc”, which is available through the `R` package `survival`. A description of the dataset can be found on page 75 here. We will consider the covariates listed below.

- treatment (`trt`)
- age (`age`)
- sex (`sex`)
- presence of ascites (`ascites`)
- presence of hepatomegaly or enlarged liver (`hepato`)
- blood vessel malformations in the skin (`spiders`)
- presence of edema (`edema`)
- serum bilirunbin in mg/dl (`bili`)
- serum albumin in mg/dl (`albumin`)
- standardised blood clotting time (`protime`)
- alkaline phosphotase in U/liter (`alk.phos`)
- aspartate aminotransferase (`ast`)
- histologic stage of disease (`stage`)

For the continuous variables in the list above, we will also consider their log-transformed and squared values as separate covariates. Now, we will conduct a backward elimination procedure, where we initially fit a Cox regression with all the described covariates. Then the covariate whose corresponding coefficient has the highest p-value is removed, and we fit a new model without it. This procedure is repeated until all p-values are below 0.05. Note that we only use the first 312 cases in the dataset to fit the model. Code which implements the backward elimination and plots the estimated relative risk functions is given below.

```r
# Construct train dataset
train <- pbc[1:312,
             c("age", "sex", "ascites", "hepato", "spiders", "edema",
               "bili", "albumin", "protime", "alk.phos", "ast", "stage",
               "trt", "time")] # pick out first 312 observation
# Add censor column
censor <- pbc$status[1:312]
censor[censor == 1] = 0
censor[censor == 2] = 1
train$censor <-  censor
```

```r
# Add square and log() of continuous variables
train$age2 <- (train$age)^2
train$lage <- log(train$age)
train$bili2 <- (train$bili)^2
train$lbili <- log(train$bili)
train$albumin2 <- (train$albumin)^2
train$lalbumin <- log(train$albumin)
train$protime2 <- (train$protime)^2
train$lprotime <- log(train$protime)
train$alk.phos2 <- (train$alk.phos)^2
train$lalk.phos <- log(train$alk.phos)
train$ast2 <- (train$ast)^2
train$last <- log(train$ast)

# Elimination strategy: Remove highest p-value until all <= 0.05.
# Function for backward elimination of features:
elim <- function(df, model.type = "cox"){
  y <- "Surv(time, censor)"
  covariates <- setdiff(colnames(df), c("time", "censor"))
  removed <- c()
  p.above <- TRUE
  while(p.above){
    formula <- as.formula(paste(y, paste(covariates, collapse=" + "),
                                sep=" ~ ")) # Create formula
    model <- p.vals <- NA
    if(model.type == "cox"){
      model <- coxph(formula, data = df)
      s <- summary(model)
      p.vals <- s$coefficients[,5]
    } else if(model.type == "surv"){
      model <- survreg(formula, data = df)
      s <- summary(model)
      p.vals <- s$table[2:(length(covariates)+1), 4]
    } else{
      stop("Invalid model.")
    }
    if(sum(p.vals > 0.05) == 0 || length(covariates) == 1){ # Stop elim
      p.above = FALSE
    } else{
      remove <- which(p.vals == max(p.vals))
      removed <- c(removed, covariates[remove])
      covariates <- covariates[-remove]
    }
  }
  return(list(model = model, removed = removed))
}

# Run step-wise backward elimination procedure
elim.cox <- elim(train)
elim.cox$removed # Removed covariates
mod.cox <- elim.cox$model

# Calculate exponential relative risk functions (errf)
```

Order	Covariate
1	alk.phos2
2	lalbumin
3	ast
4	albumin2
5	lprotime
6	hepato
7	trt
8	protime2
9	ast2
10	lalk.phos
11	alk.phos
12	spiders
13	bili2
14	bili
15	sex
16	lage
17	age2
18	ascites

Table 1: For the Cox regression model: the covariates which were removed in the step-wise backward elimination procedure and the order in which they were removed.

```r
r.cox <- exp(as.matrix(train[, names(mod.cox$coefficients)]) %*%
               mod.cox$coefficients)
# Plot (log of) errf
r.df <- data.frame(r.cox = r.cox, x = 1:312)
ggplot(r.df, aes(x = x)) + geom_point(aes(y = log(r.cox), color = "Estimated r")) +
  xlab("Case no. i") + ylab("log(r)") +
  scale_color_manual(name = " ", values = c("Estimated r" = "blue")) +
  theme_minimal()
ggsave("figures/errf_cox.pdf", height = 5, width = 8)
```

The covariates which have been removed and the order in which they were removed are given in Table 1. The estimated relative risk functions,

$$r(\hat{\boldsymbol{\beta}}, \boldsymbol{x}_i) = \exp\left\{\hat{\boldsymbol{\beta}}^{\mathrm{T}} \boldsymbol{x}_i\right\}, \quad i = 1, \dots, 312,$$

of the resulting model are displayed in Figure 1.

b)

Here, we will redo the analysis in **a)**, but now using a Weibull regression model with exponential relative risk function. Note that we use the same `elim` function to do the backward step-wise elimination of covariates. To fit the Weibull regression model, we use the `survreg` function in `R`. We also want to compare the estimated relative risk functions of the Weibull model to the ones of the Cox regression. Then, we must take into account that the `survreg` function outputs parameters in the framework of an acceleration time model [2]. This means that it estimates a scale parameter, σ, and an intercept, μ, in addition to the parameters associated with the covariates, $\boldsymbol{\gamma} = (\gamma_1, \dots, \gamma_p)^{\mathrm{T}}$. We can write the resulting hazard function mathematically as

$$\alpha(t \mid \boldsymbol{x}_i) = \frac{1}{\sigma} t^{\frac{1}{\sigma}-1} \exp\left\{-\frac{\mu}{\sigma} - \frac{1}{\sigma}\boldsymbol{\gamma}^{\mathrm{T}} \boldsymbol{x}_i\right\},$$

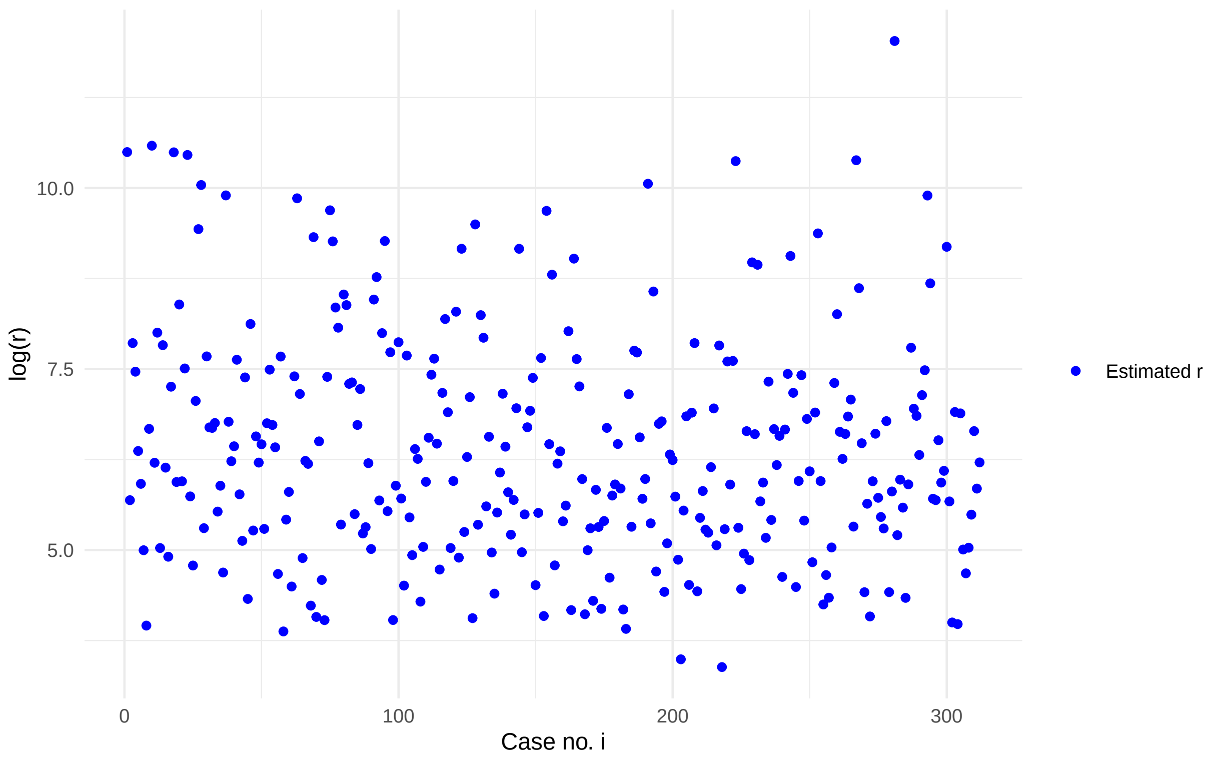

Figure 1: The log-transformed estimated relative risk functions for the 312 cases used to fit the Cox regression model in **a)**.

from which we observe that the corresponding proportional hazard parameters are given by

$$\beta_j = -\gamma_j/\sigma, \quad j = 1, \dots, p. \tag{1}$$

Now, we can compare the estimated relative risk functions of the Weibull regression model and the proportional hazard Cox model. Below is the code used to achieve this.

```
elim.surv <- elim(train, model.type = "surv")
elim.surv$removed # Removed covariates
mod.surv <- elim.surv$model

# Find corresponding coefficients and errf
gamma <- tail(mod.surv$coefficients, -1)
beta <- -gamma/mod.surv$scale
r.surv <- exp(as.matrix(train[, names(beta)]) %*%
               beta)

# Plot errf together
r.df$r.surv <- r.surv
ggplot(r.df, aes(x = x)) +
  geom_point(aes(y = log(r.cox), color = "Estimated r (Cox)"))+
  geom_point(aes(y = log(r.surv), color = "Estimated r (Weibull)")) +
  xlab("Case no. i") + ylab("log(r)") +
  scale_color_manual(name = " ",
                     values = c("Estimated r (Cox)" = "blue",
                                "Estimated r (Weibull)" = "red")) +
```

Order	Covariate
1	`albumin`
2	`ast2`
3	`albumin2`
4	`alk.phos`
5	`lprotime`
6	`hepato`
7	`trt`
8	`protime2`
9	`spiders`
10	`ast`
11	`lalk.phos`
12	`alk.phos2`
13	`bili2`
14	`bili`
15	`sex`
16	`lage`
17	`age2`
18	`ascites`

Table 2: For the Weibull regression model: the covariates which were removed in the step-wise backward elimination procedure and the order in which they were removed.

```r
  theme_minimal()
ggsave("figures/errf_both.pdf", height = 5, width = 8)
```

The covariates which have been removed in the step-wise backward elimination procedure and the order in which they were removed are given in Table 2.

Using Equation (1), we plot the log-transformed estimated relative risk functions of the Weibull regression model, together with those of the Cox regression model in **a)**. The result is displayed in Figure 2. We observe that the estimates are similar. Since the estimated relative risk functions are so similar, the baseline hazard rates should also be similar. Considering that the Weibull model is fully parametric and less flexible than the semi-parametric Cox regression model, one may prefer the Weibull model, since it in that regard is the simplest model and one may argue that the Cox model introduces unnecessary variance. The Weibull model also has the advantage that the baseline hazard rate is a known function of the parameters, which is convenient if we want to study e.g. the survival function and construct confidence intervals for it.

c)

Based on the model estimated in **b)**, we ask the question: Which covariate is the most important for the survival probability? The survival probability is given by

$$S(t \mid \boldsymbol{x}) = \exp\left\{-\int_0^t \alpha(s \mid \boldsymbol{x})\mathrm{d}s\right\} = \exp\left\{-\exp\{\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{x}\}\int_0^t \alpha_0(s)\mathrm{d}s\right\}, \tag{2}$$

where $\alpha_0(t)$ is the baseline hazard rate. Table 3 presents the coefficients corresponding to the different covariates, their p-value and the standard deviation of the covariate values times the corresponding coefficient-estimate. The covariate `lbili` (the log of `bili`) has the lowest p-value, which arguably makes it the most important covariate for the survival probability, since it is most likely to be different from 0.

If we, on the other hand, regard all coefficients as significantly different from zero, it may be more natural to ask which covariate has the greatest impact on (2). The practical importance of a covariate will depend on the magnitude of the coefficient and the variability of the corresponding covariate. Therefore, we consider the absolute value of the estimated coefficient times the standard deviation of the corresponding covariate values as a measure of importance. Table 3 then tells us that (incidentally) `lbili` still is the most important

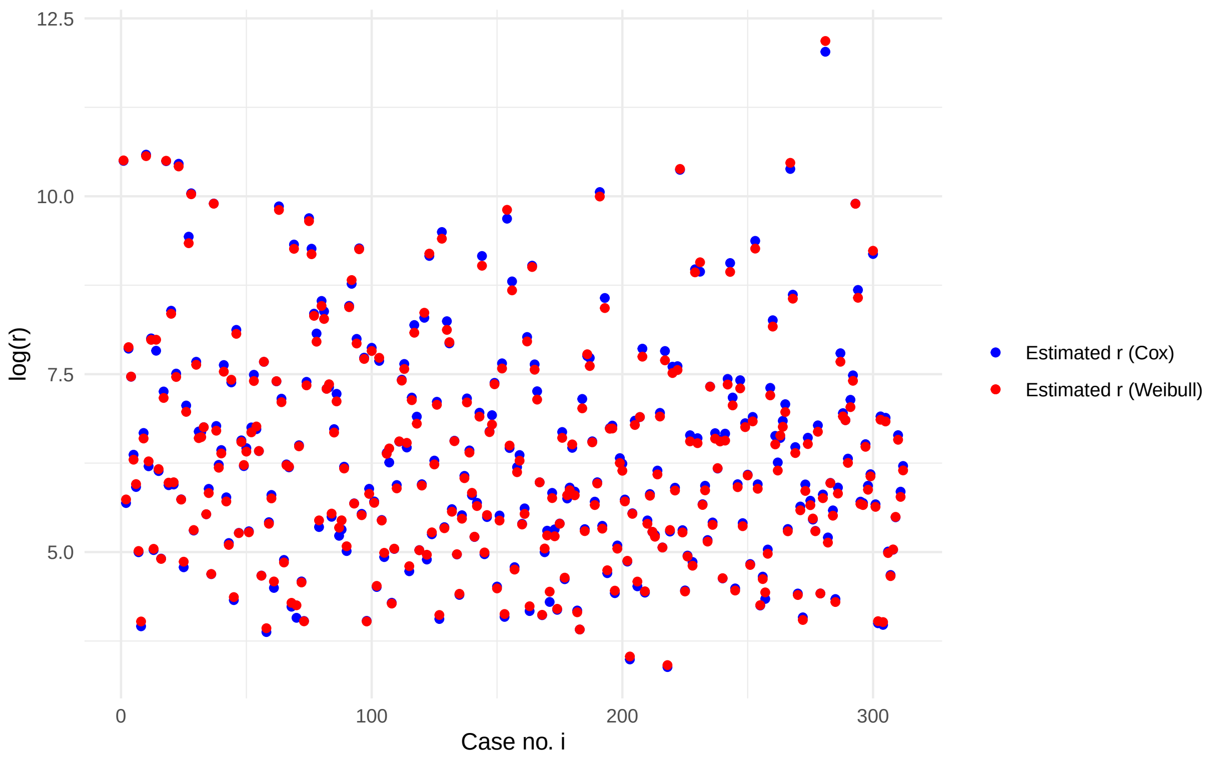

Figure 2: The log-transformed estimated relative risk functions for the 312 cases used to fit the Weibull regression model (red) and the Cox regression model (blue).

covariate. However, there may be correlations between the covariates, which would weaken this argument. The code used in this sub-problem is given below.

```
# Construct table with coefficients and stats
covariates <- names(tail(mod.surv$coefficients,-1))
X <- train[ , covariates]
sd.X <- apply(X, 2, sd)
variation <- sd.X * beta
sum.surv <- summary(mod.surv)
p.vals <- sum.surv$table[2:(length(mod.surv$coefficients)),4]
df <- data.frame("coef" = beta, "p-value" = p.vals,"var" = variation)
xtable(df, digits = 5)
```

d)

Next, we will create a 90% confidence intervals for the parameters of the Weibull regression model. We note that `survreg` estimates $\ln(\sigma)$ and not σ, so we define $\phi = \ln(\sigma)$. We collect the parameters in the vector $\boldsymbol{\theta} = (\phi, \mu, \gamma_1, \dots, \gamma_p)^{\mathrm{T}}$. Then, we utilize the fact that the resulting estimator enjoys the usual properties of a maximum likelihood estimator, i.e

$$\hat{\boldsymbol{\theta}} \sim \mathcal{N}\left(\boldsymbol{\theta}, \mathbf{I}^{-1}(\boldsymbol{\theta})\right),$$

where $\hat{\boldsymbol{\theta}}$ is the estimator of the true parameters, and we can estimate the variance as $\mathbf{I}^{-1}(\boldsymbol{\theta}) \approx \mathbf{I}^{-1}(\hat{\boldsymbol{\theta}})$ [1]. The approximated variance is readily available in `R`, so we can easily calculate the confidence interval, which for the j'th parameter takes the form

Covariate	Coefficient, $\hat{\beta}_j$	p-value	$\hat{\beta}_j \cdot \mathrm{SD}(x_j)$
`age`	0.03661	0.00006	0.38736
`edema`	0.92027	0.00144	0.25262
`protime`	0.27676	0.00335	0.27796
`stage`	0.32664	0.01549	0.28675
`lbili`	0.68609	0.00000	0.70816
`lalbumin`	-2.35592	0.00130	-0.29890
`last`	0.64263	0.00988	0.28802

Table 3: The estimated coefficients of the Weibull regression model, along with their p-value.

$$\left[\hat{\theta}_j - z_{\alpha/2}\sqrt{\left(\mathbf{I}^{-1}(\hat{\boldsymbol{\theta}})\right)_{j,j}}\,,\ \hat{\theta}_j + z_{\alpha/2}\sqrt{\left(\mathbf{I}^{-1}(\hat{\boldsymbol{\theta}})\right)_{j,j}}\,\right].$$

Code which constructs the confidence intervals is given below.

```r
marg.var <- diag(sum.surv$var)
z.alpha <- qnorm(0.05, lower.tail = FALSE)
params <- sum.surv$table[,1]
lower <- params - z.alpha*sqrt(marg.var)
upper <- params + z.alpha*sqrt(marg.var)
conf.ints <- data.frame(upper = upper, lower = lower)
xtable(conf.ints, digits = 5)
```

Table 4 contains the 90% confidence intervals of the parameters in $\hat{\boldsymbol{\theta}}$.

Parameter	Upper bound	Lower bound
μ	14.44966	9.99258
`age`, γ_1	-0.01314	-0.03132
`edema`, γ_2	-0.27045	-0.84725
`protime`, γ_3	-0.07384	-0.26229
`stage`, γ_4	-0.06358	-0.33314
`lbili`, γ_5	-0.30652	-0.52676
`lalbumin`, γ_6	2.16250	0.69885
`last`, γ_7	-0.14146	-0.63903
ϕ	-0.38260	-0.61497

Table 4: 90% confidence intervals for the parameters in the Weibull regression model created in **b)**.

Now, we wish to find 90% confidence intervals for the exponential relative risk functions of the Weibull regression model, i.e. for

$$\exp\left\{-\frac{1}{\sigma}\boldsymbol{\gamma}^{\mathrm{T}}\boldsymbol{x}_i\right\} = \exp\left\{-\exp\left\{-\phi\right\}\boldsymbol{\gamma}^{\mathrm{T}}\boldsymbol{x}_i\right\}, \quad i = 1, \ldots, 312. \tag{3}$$

Our strategy is to take the log of this quantity and then insert the maximum likelihood estimators, which gives the function

$$f(\hat{\phi}, \hat{\boldsymbol{\gamma}}; \boldsymbol{x}_i) := -\exp\left\{-\hat{\phi}\right\}\hat{\boldsymbol{\gamma}}^{\mathrm{T}}\boldsymbol{x}_i.$$

This function is approximated with a first order Taylor expansion around the maximum likelihood estimates $(\hat{\phi}, \hat{\boldsymbol{\gamma}}) = (\tilde{\phi}, \tilde{\boldsymbol{\gamma}})$:

$$
\begin{aligned}
f(\hat{\phi}, \hat{\boldsymbol{\gamma}}; \boldsymbol{x}_i) &\approx f(\tilde{\phi}, \tilde{\boldsymbol{\gamma}}; \boldsymbol{x}_i) + \exp\left\{-\tilde{\phi}\right\} \tilde{\boldsymbol{\gamma}}^{\mathrm{T}} \boldsymbol{x}_i (\hat{\phi} - \tilde{\phi}) - \exp\left\{-\tilde{\phi}\right\} \sum_{j=1}^{p} x_j (\hat{\gamma}_j - \tilde{\gamma}_j) \\
&= \exp\left\{-\tilde{\phi}\right\} \left[\tilde{\boldsymbol{\gamma}}^{\mathrm{T}} \boldsymbol{x}_i (\hat{\phi} - \tilde{\phi} - 1) - \sum_{j=1}^{p} x_j (\hat{\gamma}_j - \tilde{\gamma}_j)\right] =: T(\hat{\phi}, \hat{\boldsymbol{\gamma}}; \boldsymbol{x}_i).
\end{aligned}
$$

Notice that $T(\hat{\phi}, \hat{\boldsymbol{\gamma}}; \boldsymbol{x}_i)$ is a linear combination of normally distributed variables and is thus itself normally distributed. In order to construct confidence intervals, we need to find the variance:

$$
\mathrm{Var}[T(\hat{\phi}, \hat{\boldsymbol{\gamma}}; \boldsymbol{x}_i)] = \exp\left\{-2\tilde{\phi}\right\} \left[(\tilde{\boldsymbol{\gamma}}^{\mathrm{T}} \boldsymbol{x}_i)^2 \mathrm{Var}[\hat{\phi}] + \sum_{j=1}^{p} x_j^2 \mathrm{Var}[\hat{\gamma}_j] - 2\tilde{\boldsymbol{\gamma}}^{\mathrm{T}} \boldsymbol{x}_i \sum_{j=1}^{p} x_j \mathrm{Cov}[\hat{\phi}, \hat{\gamma}_j]\right].
$$

Using this as an approximation to the variance of $f(\hat{\phi}, \hat{\boldsymbol{\gamma}}, \boldsymbol{x}_i)$, we can construct approximate confidence intervals for the exponential relative risk functions. Such 90% confidence intervals for the log of the exponential relative risk functions, i.e. $\boldsymbol{\beta}^{\mathrm{T}} \boldsymbol{x}_i$, $i = 1, \ldots 312$, are displayed in Figure 3. The plot clearly illustrates that the confidence intervals have a considerable width, which demonstrates that there may be a large difference between the estimated relative risk functions and the true relative risk functions, which is important to be aware of. The code used for this second part of the current sub-problem is given below.

```r
# Variance of Taylor expansion
var.T <- function(phi, gamma, x, var.phi, var.g, cov.pg){
  g.x <- t(gamma)%*%x
  val <- exp(-2*phi)*(g.x^2*var.phi + sum(x^2*var.g) -
                      2*g.x*sum(x*cov.pg))
  return(val)
}

# Find variance of errf
var.vec <- rep(NA, 312)
phi <- params[9]
var.phi <- marg.var[9]
var.gamma <- marg.var[2:8]
cov.pg <- mod.surv$var[9, 2:8]
for(i in 1:312){
  x <- train[i, covariates]
  var.vec[i] <- var.T(phi, as.numeric(gamma), as.numeric(x),
                      var.phi, var.gamma, cov.pg)
}

# Add confindence intervals to dataframe
r.df$upper <- log(r.df$r.surv) + z.alpha*sqrt(var.vec)
r.df$lower <- log(r.df$r.surv) - z.alpha*sqrt(var.vec)

# Plot result
ggplot(r.df, aes(x = x)) +
  geom_errorbar(aes(ymin=lower, ymax=upper, colour="Confidence interval"),
  width = 2, size = 0.5) +
  geom_point(aes(y = log(r.surv), color = "Estimated r (Weibull)"), size = 0.2) +
  xlab("Case no. i") + ylab(" ") +
  scale_color_manual(name = " ",
                     values = c("Estimated r (Weibull)" = "blue4",
```

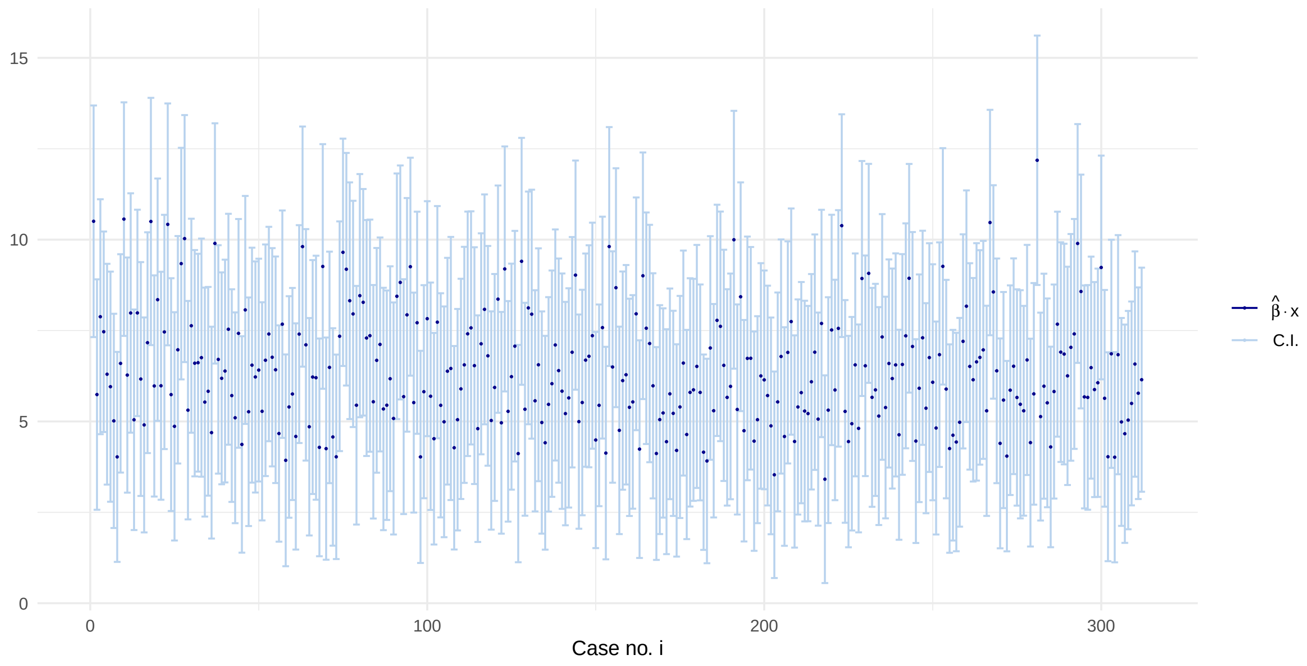

Figure 3: Confidence intervals in light blue for the log of the exponential relative risk functions in (3). The maximum likelihood estimates for the 312 cases are displayed as dark blue points.

```
                              "Confidence interval" = "slategray2" ),
                    labels = expression(hat(beta)%.%x, paste("C.I."))) +
  theme_minimal()
ggsave("figures/errf_confint.pdf", height = 5, width = 10)
```

e)

Again, we consider the Weibull regression model fitted in **b)**. We wish to find an approximate 90% prediction interval for $\Pr(T > t \mid \boldsymbol{x}) =: S(t \mid \boldsymbol{x})$, i.e. the probability of surviving up to or longer than time t given the covariate vector $\boldsymbol{x}$. This quantity is given by

$$\begin{aligned} S(t \mid \boldsymbol{x}) &= \exp\left\{-\int_0^t \alpha(s \mid \boldsymbol{x}) \mathrm{d}s\right\} \\ &= \exp\left\{-\int_0^t \frac{1}{\sigma} s^{\frac{1}{\sigma}-1} \exp\left\{-\frac{1}{\sigma}(\mu + \boldsymbol{\gamma}^{\mathrm{T}}\boldsymbol{x})\right\} \mathrm{d}s\right\} \\ &= \exp\left\{-\frac{1}{\sigma} \exp\left\{-\frac{1}{\sigma}(\mu + \boldsymbol{\gamma}^{\mathrm{T}}\boldsymbol{x})\right\} \int_0^t s^{\frac{1}{\sigma}-1} \mathrm{d}s\right\} \\ &= \exp\left\{-t^{\frac{1}{\sigma}} \exp\left\{-\frac{1}{\sigma}(\mu + \boldsymbol{\gamma}^{\mathrm{T}}\boldsymbol{x})\right\}\right\}. \end{aligned} \tag{4}$$

To construct the confidence intervals, we consider the complementary log-log transformation of this quantity, i.e.

$$
\begin{aligned}
\operatorname{cloglog}(S(t \mid \boldsymbol{x})) &= \frac{1}{\sigma}\left(\ln t - \mu - \boldsymbol{\gamma}^{\mathrm{T}}\boldsymbol{x}\right)\\
&= \exp\{-\phi\}\left(\ln t - \mu - \boldsymbol{\gamma}^{\mathrm{T}}\boldsymbol{x}\right)\\
&=: f(\boldsymbol{\theta}; t, \boldsymbol{x}),
\end{aligned}
$$

where $\phi = \ln\sigma$ and $\boldsymbol{\theta} = (\mu, \boldsymbol{\gamma}, \phi)^{\mathrm{T}}$. Then, we insert the maximum likelihood estimator of the parameters, $\hat{\boldsymbol{\theta}}$, and approximate $f(\hat{\boldsymbol{\theta}}; t, \boldsymbol{x})$ by a first order Taylor expansion around the maximum likelihood estimates which we obtained, $\hat{\boldsymbol{\theta}} = \tilde{\boldsymbol{\theta}}$:

$$
\begin{aligned}
f(\hat{\boldsymbol{\theta}}; t, \boldsymbol{x}) &\approx f(\tilde{\boldsymbol{\theta}}; t, \boldsymbol{x}) + \sum_j \frac{\partial f}{\partial \hat{\theta}_j}\bigg|_{\hat{\boldsymbol{\theta}}=\tilde{\boldsymbol{\theta}}} \cdot (\hat{\theta}_j - \tilde{\theta}_j)\\
&= \exp\{-\tilde{\phi}\}\left[\ln t - \tilde{\boldsymbol{\gamma}}^{\mathrm{T}}\boldsymbol{x} - (\ln t - \tilde{\mu} - \tilde{\boldsymbol{\gamma}}^{\mathrm{T}}\boldsymbol{x})(\hat{\phi} - \tilde{\phi}) - \hat{\mu} - \sum_{j=1}^{p} x_j(\hat{\gamma}_j - \tilde{\gamma}_j)\right]\\
&= \boldsymbol{a}^{\mathrm{T}}\hat{\boldsymbol{\theta}} + b, \qquad (5)
\end{aligned}
$$

where

$$
\boldsymbol{a} = \exp\{-\tilde{\phi}\}\begin{pmatrix} -1 \\ -\boldsymbol{x} \\ \tilde{\boldsymbol{\gamma}}^{\mathrm{T}}\boldsymbol{x} + \tilde{\mu} - \ln t \end{pmatrix} \quad \text{and} \quad b = \exp\{-\tilde{\phi}\}\left[\ln t - \tilde{\boldsymbol{\gamma}}^{\mathrm{T}}\boldsymbol{x} + (\ln t - \tilde{\mu} - \tilde{\boldsymbol{\gamma}}^{\mathrm{T}}\boldsymbol{x})\tilde{\phi} + \sum_{j=1}^{p} x_j\tilde{\gamma}_j\right].
$$

We observe that (5) is a linear combination of normally distributed variables and is thus itself normally distributed. We find its variance:

$$
\begin{aligned}
\operatorname{Var}[\boldsymbol{a}^{\mathrm{T}}\hat{\boldsymbol{\theta}} + b] &= \boldsymbol{a}^{\mathrm{T}}\operatorname{Cov}[\hat{\boldsymbol{\theta}}]\boldsymbol{a}\\
&= \exp\left\{-2\tilde{\phi}\right\}\Bigg[(\ln t - \tilde{\mu} - \tilde{\boldsymbol{\gamma}}^{\mathrm{T}}\boldsymbol{x})^2\operatorname{Var}[\hat{\phi}] + \operatorname{Var}[\hat{\mu}] + \sum_{j=1}^{p} x_j^2\operatorname{Var}[\hat{\gamma}_j]\\
&\quad - 2(\ln t - \tilde{\mu} - \tilde{\boldsymbol{\gamma}}^{\mathrm{T}}\boldsymbol{x})\operatorname{Cov}[\hat{\phi}, \hat{\mu}] + 2\sum_{j=1}^{p} x_j\operatorname{Cov}[\hat{\mu}, \hat{\gamma}_i] - 2(\ln t - \tilde{\mu} - \tilde{\boldsymbol{\gamma}}^{\mathrm{T}}\boldsymbol{x})\sum_{j=1}^{p} x_j\operatorname{Cov}[\hat{\phi}, \hat{\gamma}_j]\Bigg].
\end{aligned}
$$

Now, we can create an approximate prediction interval for $f(\boldsymbol{\theta}; t, \boldsymbol{x})$. For each of the 312 cases in the dataset, we evaluate the prediction interval for surviving up to time $t = 1500$ days. This is displayed in Figure 4. The code used in this sub-problem is given below.

```r
# Initialize parameters
t <- 1500
f.vec <- f.lower <- f.upper <- rep(NA, 312)
mu <- params[1]

# Find pred.int. for cloglog transformation
for(i in 1:312){
  x <- train[i, covariates]
  g.x <- t(as.numeric(gamma))%*%as.numeric(x)
  c <- log(t) - mu - g.x # const. which is reused
  f.val <- exp(-phi)*c
  a <- -exp(-phi)*as.numeric(c(1, x, c))
```

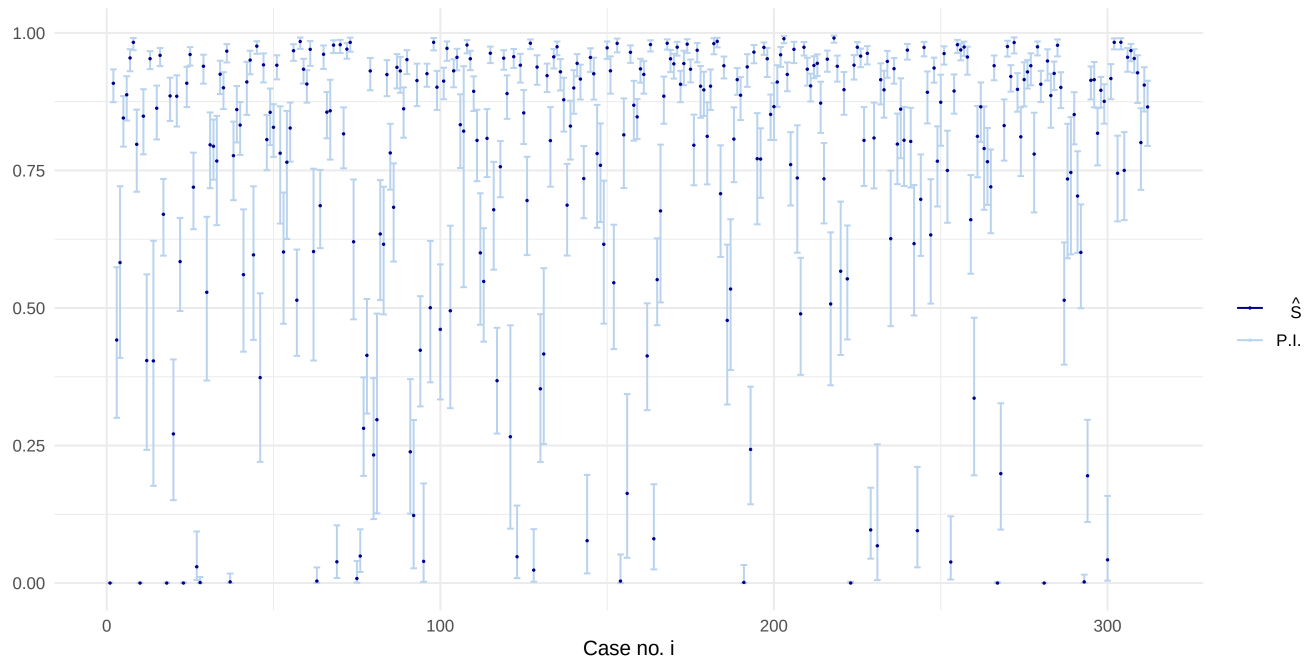

Figure 4: Prediction intervals in light blue for the survival function in (4). The survival functions with inserted maximum likelihood estimates, $\hat{S}$, is displayed for each of the 312 cases as dark blue points.

```
  b <- exp(-phi)*(log(t) - g.x + c*phi + sum(x*gamma))
  sdev <- sqrt(t(a)%*%mod.surv$var%*%a)

  f.lower[i] <- f.val - z.alpha*sdev
  f.upper[i] <- f.val + z.alpha*sdev
  f.vec[i] <- f.val
}

# Inverse of cloglog
inv.cloglog <- function(x){
  return(exp(-exp(x)))
}

# Plot pred.ints.
S.df <- data.frame(S = inv.cloglog(f.vec), lower = inv.cloglog(f.upper),
                   upper = inv.cloglog(f.lower), x = 1:312)

ggplot(S.df, aes(x = x)) +
  geom_errorbar(aes(ymin=lower, ymax=upper, colour="Prediction Interval"),
  width = 2, size = 0.5) +
  geom_point(aes(y = S, color = "cloglog(S)"), size = 0.2) +
  xlab("Case no. i") + ylab(" ") +
  scale_color_manual(name = " ",
                     values = c("cloglog(S)" = "blue4",
                                "Prediction Interval" = "slategray2"),
                     labels = expression(hat(S), paste("P.I."))) +
  theme_minimal()
ggsave("figures/S.pdf", width = 10, height = 5)
```

References

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

[2] Dirk F. Moore. *Applied Survival Analysis Using R.* Springer, Cham, 2016.