Project 1: A simulation study examining three survival models

Margaret Gacheru, Jin Ge, Sibei Liu, and Adina Zhang

2/13/2020

Problem Statement

Time-to-event data can be modeled using proportional hazards to investigate the effect of exposure or treatment on time-to-event. A hazard rate describes the likelihood of an event (ie death, disease onset) occuring at time t and is written as the following: $h_i(t) = h_0(t)exp(x_i)$. A proportional hazards model can then used to describe a hazard rate between two treatment or exposure groups: $\frac{h(t|x_1)}{h(t|x_2)} = exp(\beta^T(x_1 - x_2))$. This model is useful as it no longer becomes dependent on time t. However, when fitting a survival model, assumptions must still be made about the baseline hazard function,$h_0(t)$, which is a function that describes the risk of death when all other covariates are equal to zero.

The exponential and Weibull proportional hazards models are parametric functions, whereby the exponential assumes a constant baseline hazard ($h_0(t) = \lambda$) while the Weibull assumes a baseline hazard as a function of time-to-event ($h_0(t) = \lambda\gamma t^{\gamma-1}$). On the other hand, the Cox proportional hazards model is a semi-parametric model, designed to be a more flexible model by not having to specify the baseline hazard function. Depending on the data, choosing the wrong assumptions can lead to inaccurate estimates of the treatment effect . In order to explore the issues related to misspecifying the baseline hazard function, we designed a simulation study to assess three survival models on three different baseline hazard function scenarios. Each scenario generates data based on different baseline hazard functions: exponential, Weibull, and an unspecified Gompertz hazard function. Then an exponential, Weibull, and Cox proportional hazards model are evaluated with each scenario.

Methods

Data Generation

Generating survival data involves 5 components: actual event time, censoring time, status indicator, observed time, and covariates, if needed. In order to generate event time, we can utilize the relationship between the survival function, $S(t)$, and hazard function $h(t)$ to obtain a direct relationship between the survival function, $S(t)$, and the baseline hazard function, $h_0(t)$. Survival function is the probability of surviving beyond time t

$$S(t) = P(T > t) = 1 - F(t)$$

Hazard function is the instantaneous rate at which the event occurs at time t and can be defined as

$$h(t) = \frac{f(t)}{S(t)} = \frac{f(t)}{1 - F(t)} = -\frac{\partial}{\partial y}log[1 - F(t)] = -\frac{\partial}{\partial y}log[S(t)]$$

Isolating $S(t)$, we find

$$S(t) = e^{-H(t)}, where\ H(t) = \int_0^t h(t)dt$$

Additionally, we can find the connection between the cumulative hazard, $H(t)$, and the baseline hazard. Given that $h(t) = h_0(t)e^{x^T\beta}$,

$$H(t) = \int_0^t h_0(t)e^{x^T\beta}dt = e^{x^T\beta}H_0(t), where\ H_0(t) = \int_0^t h_0(t)dt$$

Putting it all together, we obtain

$$S(t) = e^{-H(t)} \qquad\qquad = e^{x^T\beta}H_0(t)$$

Finally, utilize the inverse transformation method to obtain T, event time

$$T = H_0^{-1}\bigg(\frac{-log(u)}{e^{x^T\beta}}\bigg), where\ U \sim U(0,1)$$

With a specific baseline hazard function, it is straightfoward to find generate time to event

1. Under exponential distribution:

$$\text{Baseline hazard function: } h_0(t) = \lambda, where\ \lambda > 0$$
$$\text{Cumulative hazard function: } H_0(t) = \lambda t$$
$$\text{Inverse cumulative hazard function: } H_0^{-1}(t) = \frac{t}{\lambda}$$
$$\text{Survival time: } T = -\frac{log(u)}{\lambda e^{x^T\beta}}$$

2. Under weibull distribution:

$$\text{Baseline hazard function: } h_0(t) = \lambda\gamma t^{\gamma-1}, where\ \lambda, \gamma > 0$$
$$\text{Cumulative hazard function: } H_0(t) = \lambda t^{\gamma}$$
$$\text{Inverse cumulative hazard function: } H_0^{-1}(t) = \bigg(\frac{t}{\lambda}\bigg)^{\frac{1}{\gamma}}$$
$$\text{Survival time: } T = \bigg(-\frac{log(u)}{\lambda e^{x^T\beta}}\bigg)^{\frac{1}{\gamma}}$$

3. Under gompertz distribution:

$$\text{Baseline hazard function: } h_0(t) = e^{\alpha t}, where\ \lambda > 0, -\infty < \alpha < \infty$$
$$\text{Cumulative hazard function: } H_0(t) = \frac{\lambda}{\alpha}\bigg(e^{\alpha t} - 1\bigg)$$
$$\text{Inverse cumulative hazard function: } H_0^{-1}(t) = \frac{1}{\alpha}log\bigg(\frac{\alpha}{\lambda}t + 1\bigg)$$
$$\text{Survival time: } T = \frac{1}{\alpha}log\bigg(1 - \frac{\alpha log(u)}{\lambda e^{x^T\beta}}\bigg)$$

Under the 3 scenarios, we use the following steps to generate the survival dataset

1. Randomly generate X_i, treatment assignment variable, from a bernoulli distribution with $p = 0.5$

2. Generate T_i, time to event, using X_i from step 1 and pre-specified β

$$T = H_0^{-1}\left(\frac{-log(u)}{e^{x^T\beta}}\right)$$

3. Randomly generate C_i, censoring time, from an exponential distribution

4. Determine the observe time, Y_i by comparing event and censoring time

$$Y_i = min(T_i, C_i)$$

5. Create the status indicator variable, where 1 represents if event is observed and 0 if event is censored

$$Status = \begin{cases} 1, & T_i \leq C_i \\ 0, & T_i > C_i \end{cases}$$

Scenario Simiulation

We conducted simulation studies to assess the performance of the proposed framework under three scenarios with different covariate forms, censoring time distributions, and baseline hazards. We simulated 1000 data sets with sample size n=400. The parameters in each model were held constant with $\beta = 4$, $\lambda = 0.1$ and $\alpha = 4$. Each time, baseline data was generated to follow the scenario baseline models: exponential, Weibull, and an unspecified Gompertz hazard function. Generated data were fitted to exponential, Weibull, and Cox proportional hazard models. After simulating through each model, a set of β were generated and used to calculate the mean and 95% confidence interval to compare with each other. In addition, we simulated different sample sizes from 100 to 500 by 50 as each step. The mean squared errors (MSE) and bias (first-ordered) were calculated to demonstrate the performance of survival models as the sample size increased and show how poorly misspecified models perform. We also change the β to show how MSE and bias would change in order to explore how the beta is associated with the model fitting. All the data generation and simulation were performed in R version 3.6.1

Results

Association with Beta and MES(or Bias) among three scenarios

Association with sample size and MSE(or Bias) among three scenarios

Discussion

From the results above, we generally conclude that using the correctly specified proportional hazard model will have the best performance. These patterns hold as sample size and treatment effects are varied in additional simulations. In our initial thinking, Cox model should perform the best under any scenario, with lower MSEs and Bias. This is because Cox model was explicitly designed to estimate the proportional hazard ratios without having to estimate the baseline hazard function, or in other words, limiting errors from incorrect assumptions. Indeed, the Cox model performed quite well in Scenarios 2 and 3 where the baseline hazard functions were more complicated. But this kind of flexibility has weakness in the case of a simple baseline hazard function as we saw in Scenario 1. We also noticed that Cox models needed a large sample size (greater than 200) to perform well. In addition, we expected the models to perform worse on censored data (higher MSEs and increased bias) with the beta approximations increasing in variability. Unlike uncensored data, censored data is not clean with loss to follow-up and dropouts making approximation of the true effect more difficult. However, this was not strongly observed in our simulations and might require further inspection.

Further study can be carried on the following three aspects. Our study fixed the values of parameters. To extend the breadth of our conclusions, varied inputs of alpha, lambda and gamma should be implemented. Second, our study only includes one categorical covariable which was the treatment status. But in real life, other categorical and continuous variables like gender and age may also have influence on the survival time. Third, different censoring times can be generated under multiple distributions, not only limited to exponential distribution. These distributions may also reflect strong versus weak censoring, in which data may experience many or few loss-to-followup.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
library(tidyverse)
library(survival)
source("./sim.R")

# Create a function that will create a dataset from a specific baseline hazard function

set.seed(123123)

# N = sample size
# lambda = scale parameter in exponential, weibull, "cox" (> 0)
# gamma = shape parameter in weibull (> 0)
# alpha = shape parameter in "cox" (-inf to inf)
# beta = fixed treatment effect parameter

generate_data = function(N, beta, distribution, lambda, gamma, alpha) {

  #Inverse survival function from baseline hazard functions to obtain event times
  exponential = function(u, x, lambda, beta) {
    time = -log(u) / (exp(x * beta) * lambda) }

  weibull = function(u, x, lambda, gamma, beta) {
    time = ( -log(u) / (exp(x * beta) * lambda) ) ^ (1 / gamma) }

  cox = function(u, x, lambda, alpha, beta) {
    time = (1/alpha) * log( 1 - (alpha * log(u) / (lambda * exp(beta * x))) ) }

  #treatment assignment
  x = 1 * (runif(N) < 0.5)

  #generate event times
  u = runif(N)

  if (distribution == "exponential") {
    time = exponential(u, x, lambda, beta) }
  else if (distribution == "weibull") {
    time = weibull(u, x, lambda, gamma, beta) }
  else {
    time = cox(u, x, lambda, alpha, beta) }

  #censoring time
  C = rexp(N, lambda)
```

```
  # follow-up times and event indicators
  observed_time = pmin(time, C)
  status = 1 * (time <= C)

  # data set
  survival_data = data.frame(id = 1:N,
                             time = time,
                             observed_time = observed_time,
                             status = status,
                             x = x)
}


# Simulation (Take the exponential baseline data for example)
# Write a function for simulations
simulate = function(sim, N, beta, censor = FALSE, dist = "exponential") {
  # Set up coefficient vectors
  exp_beta = rep(NA, sim)
  weibull_beta = rep(NA, sim)
  cox_beta = rep(NA, sim)

  # Censored Data
  if (censor == TRUE) {

    for (i in 1:sim) {
      # Generate data
      data = generate_data(N, beta, distribution = dist,
                           lambda = 0.1, gamma = 4, alpha = 4)
      # Fit three survival distributions
      fit.exponential = survreg(Surv(data$observed_time, data$status) ~ data$x,
                                dist = "exponential")
      fit.weibull = survreg(Surv(data$observed_time, data$status) ~ data$x,
                            dist = "weibull")
      fit.cox = coxph(Surv(data$observed_time, data$status) ~ data$x)

      # Save beta coefficients
      exp_beta[i] = -fit.exponential$coefficients[-1]
      weibull_beta[i] = -fit.weibull$coefficients[-1] / fit.weibull$scale
      cox_beta[i] = fit.cox$coefficients[1]
    }

  # Uncensored Data
  } else {

    for (i in 1:sim) {
      # Generate data
      data = generate_data(N, beta, distribution = dist,
                           lambda = 0.1, gamma = 4, alpha = 4)
      # Fit three survival distributions
      fit.exponential = survreg(Surv(data$time) ~ data$x, dist = "exponential")
      fit.weibull = survreg(Surv(data$time) ~ data$x, dist = "weibull")
      fit.cox = coxph(Surv(data$time) ~ data$x)
```

```
      # Save beta coefficients
      exp_beta[i] = -fit.exponential$coefficients[-1]
      weibull_beta[i] = -fit.weibull$coefficients[-1] / fit.weibull$scale
      cox_beta[i] = fit.cox$coefficients[1]
    }

  }

  # Store beta coefficients
  coef = tibble(exp = exp_beta,
                weibull = weibull_beta,
                cox = cox_beta)

  # Calculate bias and MSE for each model
  exp_bias = (sum(exp_beta - beta)) / sim
  weibull_bias = (sum(weibull_beta - beta)) / sim
  cox_bias = (sum(cox_beta - beta)) / sim
  exp_MSE = (sum((beta - exp_beta)^2)) / sim
  weibull_MSE = (sum((beta - weibull_beta)^2)) / sim
  cox_MSE = (sum((beta - cox_beta)^2)) / sim

  # Store performance measures in a dataset
  perf_data = tibble(model = c("exp", "weibull", "cox"),
                     bias = c(exp_bias, weibull_bias, cox_bias),
                     mse = c(exp_MSE, weibull_MSE, cox_MSE))

  results = list(coef, perf_data)
  names(results) = c("coefficients", "performance")
  return(results)
}



# Simulation for different sample sizes
set.seed(12345)
sim_results =
  tibble(sample_size = c(200, 250, 300, 350, 400, 450, 500, 550, 600, 650, 700)) %>%
  mutate(
    output_uncensored = map(.x = sample_size, ~simulate(sim = 1000, N = .x, beta = 4,
                                                        dist = "weibull")$performance),
    output_censored = map(.x = sample_size,
                          ~simulate(sim = 1000, N = .x, beta = 4,
                                    censor = TRUE)$performance),
    output_uncensored = map(.x = output_uncensored, ~mutate(.x, censor = FALSE)),
    output_censored = map(.x = output_censored, ~mutate(.x, censor = TRUE)))

# Join censored and uncensored data
temp1 = sim_results %>% unnest(output_uncensored) %>% select(-output_censored)
temp2 = sim_results %>% unnest(output_censored) %>% select(-output_uncensored)
samp_size_sim = rbind(temp1, temp2)

# Plot MSE simulation results
samp_size_sim %>%
```

```r
  mutate(censor = factor(censor, levels = c(FALSE, TRUE),
                         labels = c("Uncensored", "Censored"))) %>%
  filter(censor == "Censored") %>%
  ggplot(aes(x = sample_size, y = mse, color = model)) +
  geom_point() +
  geom_line(size = 1, se = FALSE) +
  facet_grid(.~censor, scales = "free_y") +
  labs(title = "Sample size vs MSE by Survival Model",
       x = "Sample Size",
       y = "MSE") +
  theme_bw() +
  theme(strip.text.x = element_text(size = 12)) +
  theme(text = element_text(size = 10),
        legend.text=element_text(size = 10))

# Plot bias simulation results
samp_size_sim %>%
  mutate(censor = factor(censor, labels = c("Uncensored", "Censored"))) %>%
  filter(censor == "Censored") %>%
  ggplot(aes(x = sample_size, y = bias, color = model)) +
  geom_point() +
  geom_line(size = 1, se = FALSE) +
  facet_grid(.~censor, scales = "free_y") +
  labs(title = "Sample size vs Bias by Survival Model",
       x = "Sample Size",
       y = "Bias") +
  theme_bw() +
  theme(strip.text.x = element_text(size = 12)) +
  theme(text = element_text(size = 10),
        legend.text=element_text(size = 10))

# Simulation for different effect sizes
set.seed(12345)
sim_results_beta =
  tibble(beta = c(0.1, 0.5, 1, 3, 4, 5)) %>%
  mutate(
    output_uncensored = map(.x = beta, ~simulate(sim = 1000, N = 500, beta = .x,
                                                 dist = "Weibull")$performance))
    output_censored = map(.x = beta, ~simulate(sim = 1000, N = 500, beta = .x,
                                               censor = TRUE)$performance),
    output_uncensored = map(.x = output_uncensored, ~mutate(.x, censor = FALSE)),
    output_censored = map(.x = output_censored, ~mutate(.x, censor = TRUE)))

# Join censored and uncensored data
temp1 = sim_results_beta %>% unnest(output_uncensored) %>% select(-output_censored)
temp2 = sim_results_beta %>% unnest(output_censored) %>% select(-output_uncensored)
beta_sim = rbind(temp1, temp2)

# Plot MSE simulation results
temp1 %>%
  #mutate(censor = factor(censor, labels = c("Uncensored", "Censored"))) %>%
  ggplot(aes(x = beta, y = mse, color = model)) +
  geom_point() +
```

```
  geom_line(size = 1) +
  #facet_grid(.~censor) +
  labs(title = "Beta vs MSE by Survival Model",
       x = "Beta",
       y = "MSE") +
  theme_bw() +
  theme(strip.text.x = element_text(size = 12)) +
  theme(text = element_text(size = 10),
        legend.text=element_text(size = 10))

# Plot bias simulation results
temp1 %>%
  #filter(beta != 6) %>%
  #mutate(censor = factor(censor, labels = c("Uncensored", "Censored"))) %>%
  ggplot(aes(x = beta, y = bias, color = model)) +
  geom_point() +
  geom_line(size = 1) +
  #facet_grid(.~censor) +
  labs(title = "Beta vs Bias by Survival Model",
       x = "Beta",
       y = "Bias") +
  theme_bw() +
  theme(strip.text.x = element_text(size = 12)) +
  theme(text = element_text(size = 10),
        legend.text=element_text(size = 10))
```