Simulation_data_generate

1. Choose different cox distributions to generate data
2. Choose different beta
3. Create functions to simplify codes
4. Visualization 4.1 KM-curve vs. fitted curve 4.2 beta ~ MSE with different parameters to generate data 4.3 …
5. discussion: description, pros and cons …

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

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

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

##
## Attaching package: 'pracma'

## The following object is masked from 'package:purrr':
##
##     cross

## Loading required package: ggpubr

##
## Attaching package: 'survminer'

## The following object is masked from 'package:survival':
##
##     myeloma
```

Generate survival data

```
set.seed(666)

# Generate survival data
genn_dat <- function(n, lambda, beta, gamma, alpha, dist) {
  # Generate key for each observation
  id <- seq(1:n)

  # Predictor X (treatment = 1; control = 0)
  x <- rbinom(n, size = 1, prob = 0.5)

  ## --- Generate Survival Time T ---
  U <- runif(n)
  # Use exponential distribution
  expo <- function(U, lambda, x, beta) {
```

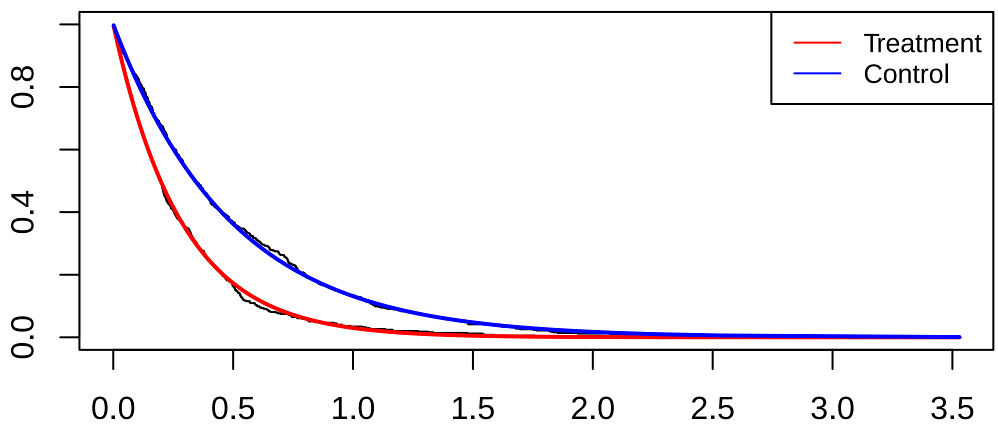
```
    -log(U) / (lambda * exp(x * beta))
  }

  # Use weibull distribution
  weibull <- function(U, lambda, x, gamma, beta) {
    (-log(U) / (lambda * exp(x * beta))) ^ (1 / gamma)
  }

  # --- To be modified (add cox function) ---
  # Use 'Cox' distribution -- gompertz
  gompertz <- function(U, alpha, lambda, x, beta) {
    (1 / alpha) * (1 - alpha * log(U) / (lambda * exp(x * beta)))
  }

  # Select different baseline functions
  if (dist == "expo") {
    surv_t <- expo(U, lambda, x, beta)}
  else if (dist == "weibull") {
    surv_t <- weibull(U, lambda, x, gamma, beta)
  }
  else {
    surv_t <- gompertz(U, alpha, lambda, x, beta)
  }

  return(
    df = data.frame(
      id = id,
      treatment = x,
      time = surv_t
    )
  )
}
```

Visualization(beta = 0.5)

1. data-exponential(lambda = 2)

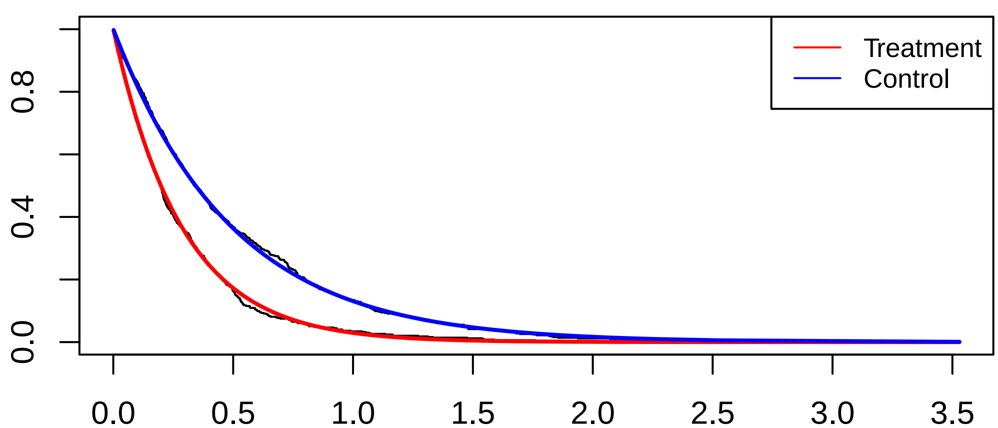
```
## Generate Data
set.seed(666)
dat_1 <- genn_dat(n = 1000, lambda = 2, beta = 0.5, dist = "expo")
s_1 <- with(dat_1, Surv(time))

## Exponential
fit_expo_1 = flexsurvreg(s_1 ~ as.factor(treatment), dist = 'exponential', data = dat_1)
plot(fit_expo_1, col = c('red', 'blue'), main = "Exponential proportional-hazards model")
legend("topright", legend = c('Treatment', 'Control'), col = c('red', 'blue'), lty = 1, cex = 0.8)
```

Exponential proportional−hazards model

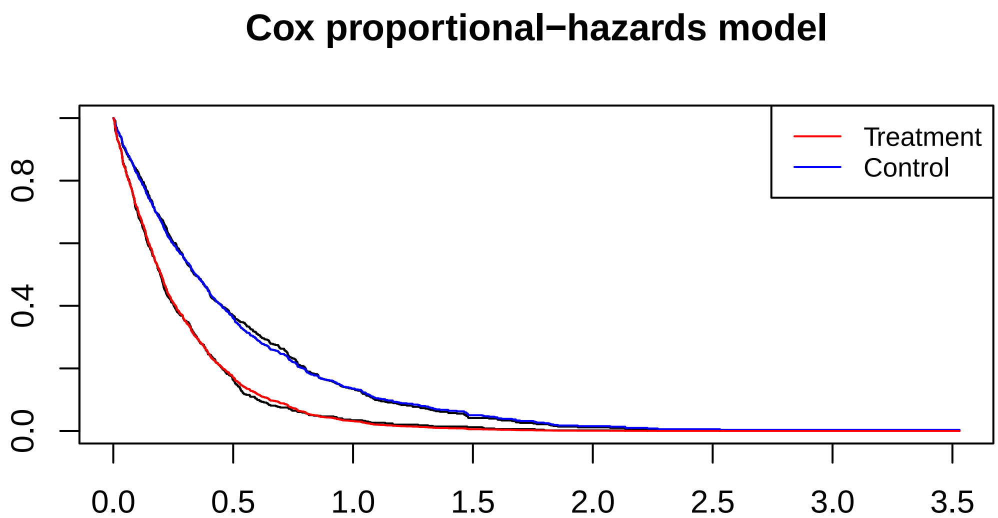



```r
## Weibull
fit_wei_1 = flexsurvreg(s_1 ~ as.factor(treatment), dist = 'weibull', data = dat_1)
plot(fit_wei_1, col = c('red', 'blue'), main = "Weibull proportional-hazards model")
legend("topright", legend = c('Treatment', 'Control'), col = c('red', 'blue'), lty = 1, cex = 0.8)
```

Weibull proportional−hazards model

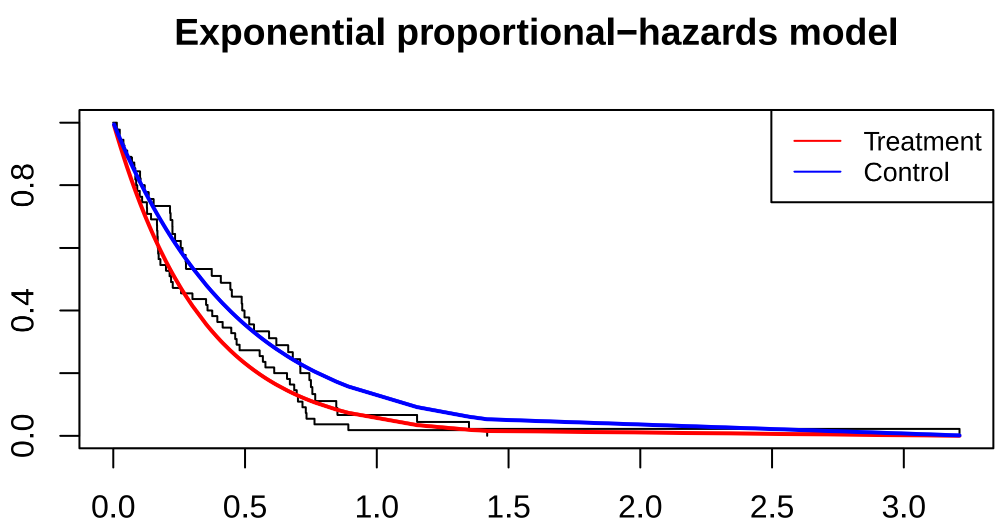



```r
## Cox
fit_km_1 = survfit(s_1 ~ treatment, data = dat_1)
plot(fit_km_1, conf.int = F, main = "Cox proportional-hazards model")
fit_cox_1 = coxph(s_1 ~ as.factor(treatment), data = dat_1)
lines(survfit(fit_cox_1, newdata = data.frame(treatment = 0)), col = "blue", conf.int = F)
lines(survfit(fit_cox_1, newdata = data.frame(treatment = 1)), col = "red", conf.int = F)
legend("topright", legend = c('Treatment', 'Control'), col = c('red', 'blue'), lty = 1, cex = 0.8)
```

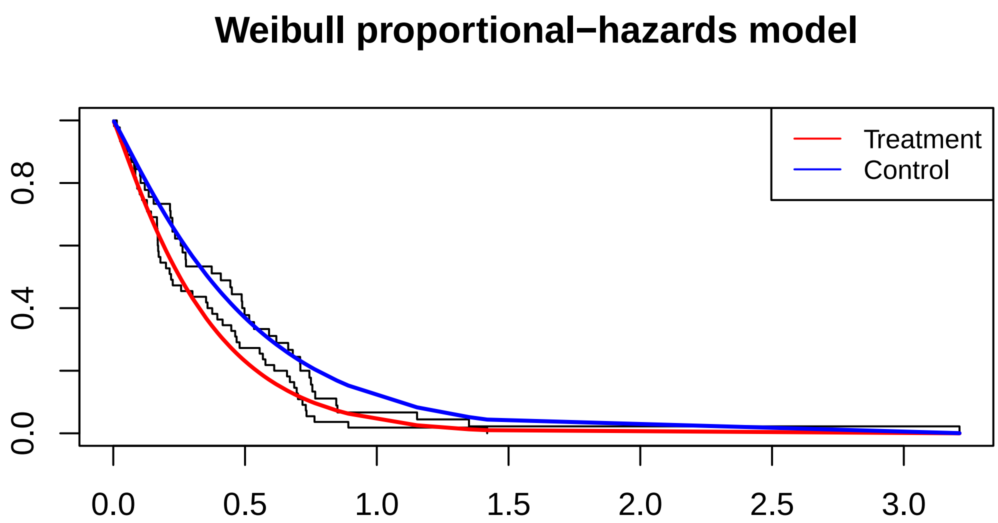


2. data-weibull(lambda = 2, gamma = 1)

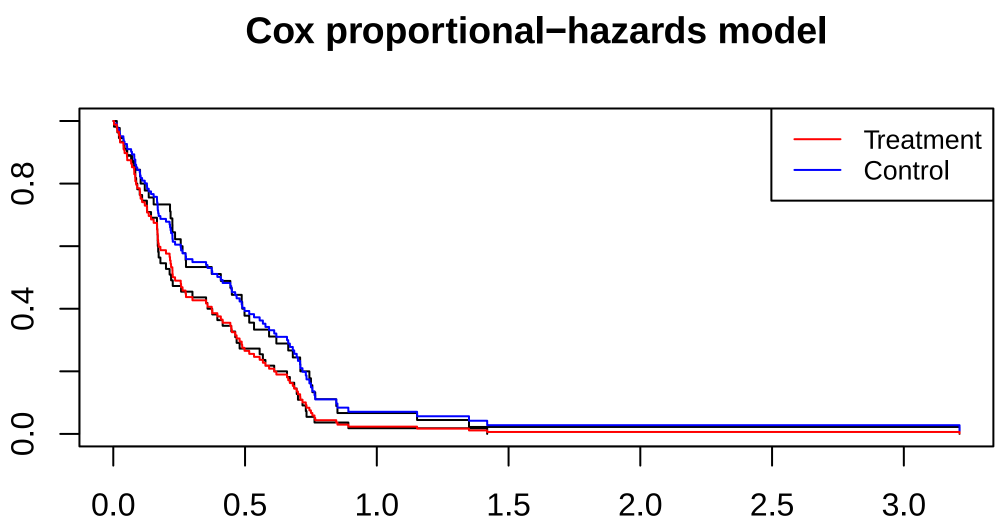
```
## Generate Data
set.seed(666)
dat_2 <- genn_dat(n = 100, lambda = 2, beta = 0.5, gamma = 1, dist = "weibull")
s_2 = with(dat_2, Surv(time))

## Exponential
fit_expo_2 = flexsurvreg(s_2 ~ as.factor(treatment), dist = 'exponential', data = dat_2)
plot(fit_expo_2, col = c('red', 'blue'), main = "Exponential proportional-hazards model")
legend("topright", legend = c('Treatment', 'Control'), col = c('red', 'blue'), lty = 1, cex = 0.8)
```




```
## Weibull
fit_wei_2 = flexsurvreg(s_2 ~ as.factor(treatment), dist = 'weibull', data = dat_2)
plot(fit_wei_2, col = c('red', 'blue'), main = "Weibull proportional-hazards model")
legend("topright", legend = c('Treatment', 'Control'), col = c('red', 'blue'), lty = 1, cex = 0.8)
```

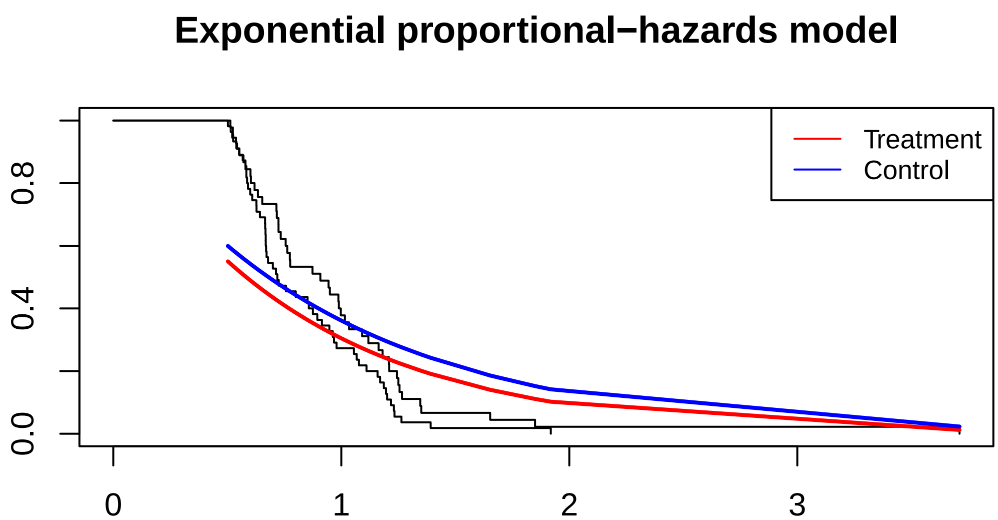



```
## Cox
fit_km_2 = survfit(s_2 ~ treatment, data = dat_2)
plot(fit_km_2, conf.int = F, main = "Cox proportional-hazards model")
fit_cox_2 = coxph(s_2 ~ as.factor(treatment), data = dat_2)
lines(survfit(fit_cox_2, newdata = data.frame(treatment = 0)), col = "blue", conf.int = F)
lines(survfit(fit_cox_2, newdata = data.frame(treatment = 1)), col = "red", conf.int = F)
legend("topright", legend = c('Treatment', 'Control'), col = c('red', 'blue'), lty = 1, cex = 0.8)
```

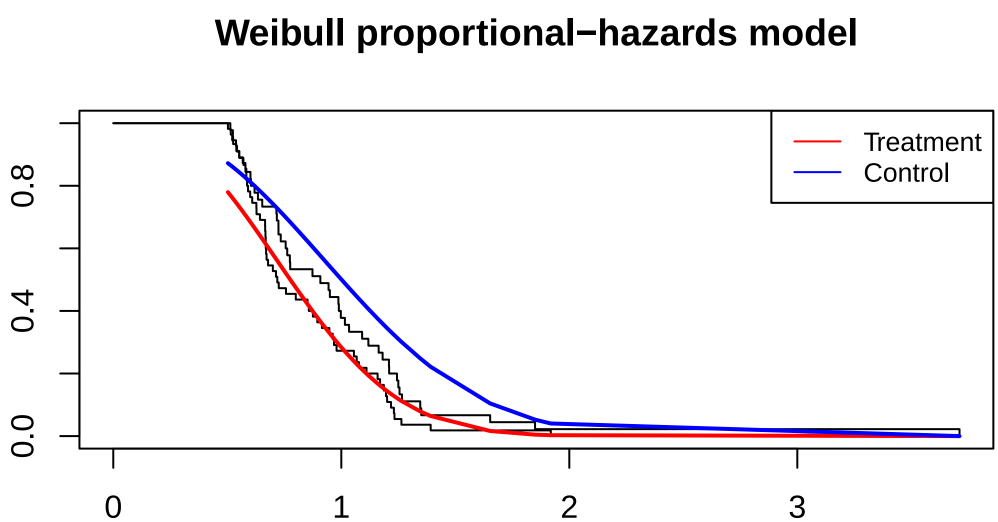


3. data-gompertz(alpha = 2, lambda = 2)

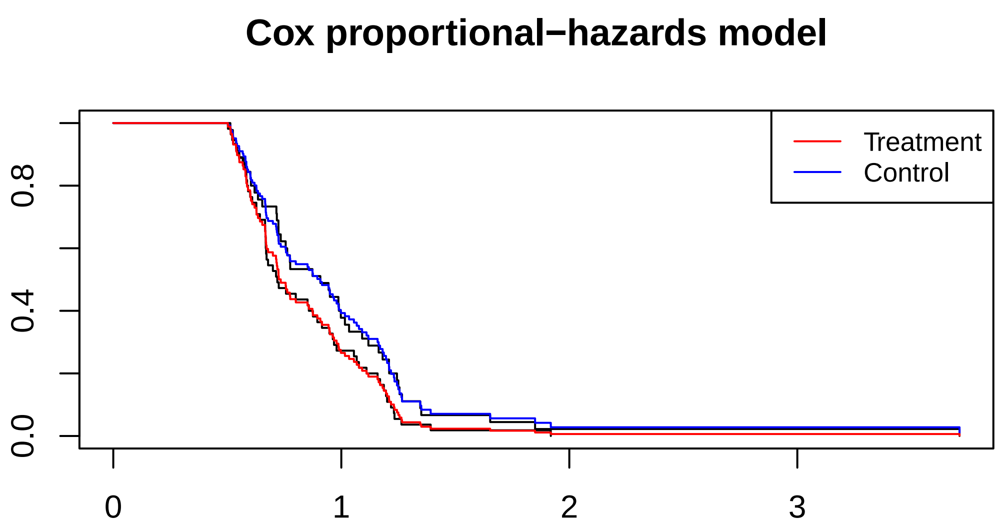
```
## Generate Data
set.seed(666)
dat_3 <- genn_dat(n = 100, lambda = 2, beta = 0.5, alpha = 2, dist = "gompertz")
s_3 = with(dat_3, Surv(time))

## Exponential
fit_expo_3 = flexsurvreg(s_3 ~ as.factor(treatment), dist = 'exponential', data = dat_3)
plot(fit_expo_3, col = c('red', 'blue'), main = "Exponential proportional-hazards model")
legend("topright", legend = c('Treatment', 'Control'), col = c('red', 'blue'), lty = 1, cex = 0.8)
```




```
## Weibull
fit_wei_3 = flexsurvreg(s_3 ~ as.factor(treatment), dist = 'weibull', data = dat_3)
plot(fit_wei_3, col = c('red', 'blue'), main = "Weibull proportional-hazards model")
legend("topright", legend = c('Treatment', 'Control'), col = c('red', 'blue'), lty = 1, cex = 0.8)
```




```
## Cox
fit_km = survfit(s_3 ~ treatment, data = dat_3)
plot(fit_km, conf.int = F, main = "Cox proportional-hazards model")
fit_cox_3 = coxph(s_3 ~ as.factor(treatment), data = dat_3)
lines(survfit(fit_cox_3, newdata = data.frame(treatment = 0)), col = "blue", conf.int = F)
lines(survfit(fit_cox_3, newdata = data.frame(treatment = 1)), col = "red", conf.int = F)
legend("topright", legend = c('Treatment', 'Control'), col = c('red', 'blue'), lty = 1, cex = 0.8)
```



3.MSE vs. beta

```r
# --- MSE vs. beta ---
n = 10^4 # Sample size

sim_result <- function(iteration, n, lambda, beta, gamma, alpha, dist) {
  # Store estimated beta
  result <- tibble(
  expo_beta = rep(NA, iteration),
  weibull_beta = rep(NA, iteration),
  cox_beta = rep(NA, iteration)
  )

  for(i in 1:iteration) {
    # Generate data
    data <- genn_dat(n, lambda, beta, gamma, alpha, dist)

    # Use exponential model
    fit_expo <- survreg(Surv(data$time) ~ data$treatment, dist = "exponential")
    result$expo_beta[i] <- -fit_expo$coefficients[-1]
    # Use weibull model
    fit_weibull <- survreg(Surv(data$time) ~ data$treatment, dist = "weibull")
    result$weibull_beta[i] <- -fit_weibull$coefficients[-1] / fit_weibull$scale
    # Use gompertz model
    fit_cox <- coxph(Surv(data$time) ~ data$treatment)
    result$cox_beta[i] <- fit_cox$coefficients

  }

  # Calculate MSE
  mse <- tibble(
    expo_mse = sum((result$expo_beta - beta)^2) / iteration,
    weibull_mse = sum((result$weibull_beta - beta)^2) / iteration,
    cox_mse = sum((result$cox_beta - beta)^2) / iteration
  )

  return(mse)
}
```