P8160 Project-1-1

Yijing Tao Renjie Wei Jibei Zheng Haolin Zhong Anyu Zhu

2022-02-17

Contents


```
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  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     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
```

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

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

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

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

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

```
library(caret)
```

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

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

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

```
library(survival)
```

```
##
## Attaching package: 'survival'
```

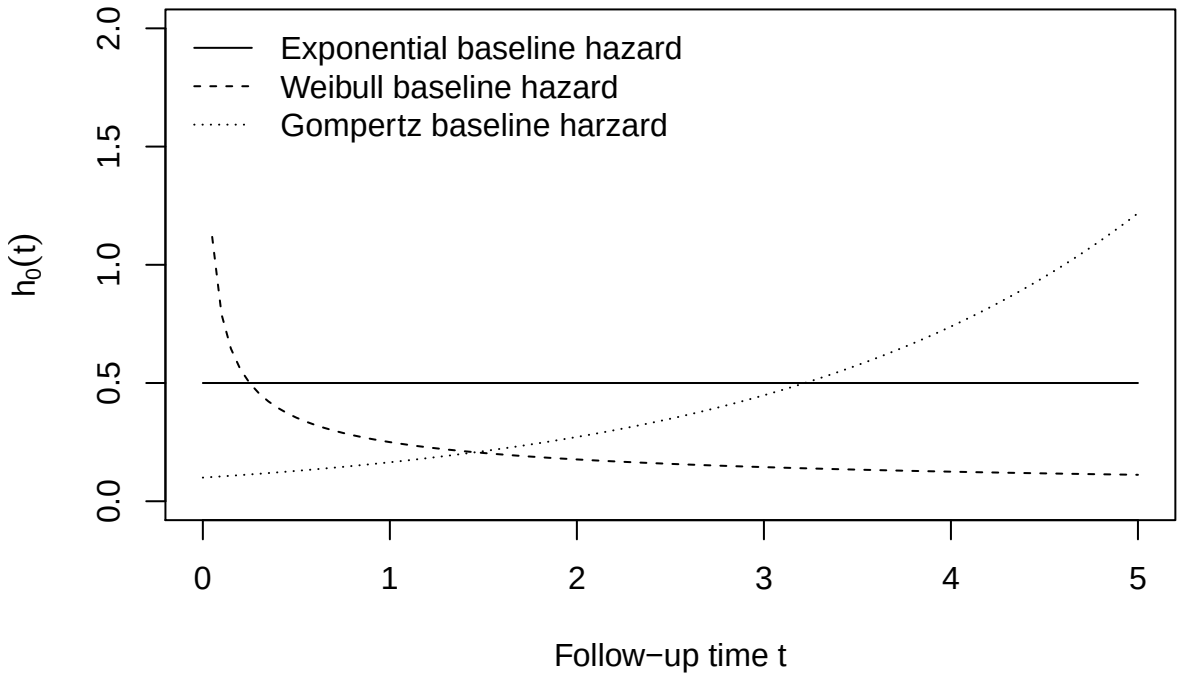
```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
library(fastmap)
```

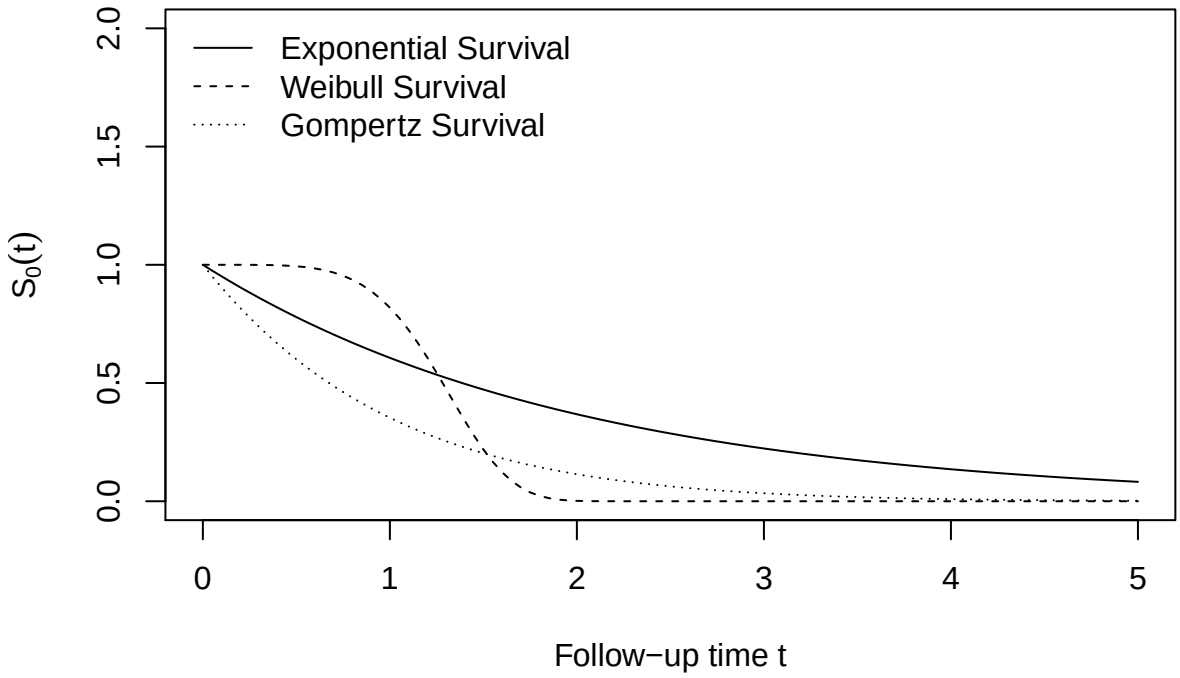
```
## Warning: package 'fastmap' was built under R version 4.0.5
```

Draw curves of hazard functions and survival functions

```
exp_haz <- function(t, lambda = 0.5) lambda * 1 * t^0
weibull_haz <- function(t, lambda = 0.5, gamma = 0.5) lambda * gamma * t^(gamma - 1)
gompertz_haz <- function(t, alpha = 0.5, lambda = 0.1) lambda * exp(alpha* t)
curve(exp_haz, from = 0, to = 5, lty = 1, ylim = c(0, 2), ylab = expression(h[0](t)), xlab = "Follow-up
curve(weibull_haz, from = 0, to = 5, lty = 2, add = TRUE)
curve(gompertz_haz, from = 0, to = 5, lty = 3, add = TRUE )
legend(x = "topleft", lty = 1:3, legend = c("Exponential baseline hazard", "Weibull baseline hazard","G
```




```
exp_surv <- function(t, lambda = 0.5) exp((-lambda)*t)
weibull_surv <- function(t, lambda = 0.2, gamma = 5) exp((-lambda)*t^gamma)
gompertz_surv <- function(t, alpha = 0.08, lambda = 1) exp(lambda/alpha*(1-exp(alpha*t)))
curve(exp_surv, from = 0, to = 5, lty = 1, ylim = c(0, 2), ylab = expression(S[0](t)), xlab = "Follow-up
curve(weibull_surv, from = 0, to = 5, lty = 2, add = TRUE)
curve(gompertz_surv, from = 0, to = 5, lty = 3, add = TRUE )
legend(x = "topleft", lty = 1:3, legend = c("Exponential Survival", "Weibull Survival", "Gompertz Survi
```



Culmulative Distribution of Time t

$$S(t) = \int_t^{\infty} f(t)dt = 1 - F(t)$$

$$h(t) = \frac{f(t)}{S(t)} = \frac{F'(t)}{1 - F(t)}$$

$$F(t) = 1 - \exp\left(-\int_0^t h(s)ds\right) = 1 - \exp\left[-(\int_0^t h_0(s)ds) \cdot \exp(\beta^T X)\right]$$

- exponential proportional-hazards model:

$$F(t) = 1 - \exp\left[-\lambda t \cdot \exp(\beta^T X)\right] \quad \Longrightarrow \quad F^{-1}(U) = -\frac{\ln(U)}{\lambda \exp(\beta^T X)}$$

- Weibull proportional-hazards model:

$$F(t) = 1 - \exp\left[-\lambda t^{\gamma} \cdot \exp(\beta^T X)\right] \quad \Longrightarrow \quad F^{-1}(U) = \left(-\frac{\ln(U)}{\lambda \exp(\beta^T X)}\right)^{\frac{1}{\gamma}}$$

- Geompartz proportional-hazards model:

$$F(t) = 1 - \exp\left[-\lambda t^{\gamma} \cdot \exp(\beta^T X)\right] \quad \Longrightarrow \quad F^{-1}(U) = \frac{1}{\alpha} \ln\left(1 - \frac{\alpha \ln U}{\lambda \exp \beta^T X}\right)$$

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

Data Generation

- generate mixed random t under given sample size `n` and mixing proportion `p`:
  - when `p = 1`, use pure exponential distribution
  - when `p = 0`, use pure weibull distribution
  - This function use lambda = 0.5, gamma = 1.5 as default
  - In later analysis we will always use `b1 = 0.5` as the treatment effect.

```
simdat = function(n, p, lambda = 0.1, gamma = 1.5, eff = list()) {
  # randomly assign group
  x1 = rbinom(n, 1, 0.5)

  u = runif(n)
  useExp = runif(n) < p
  # generating mixed data
  t =  useExp * (-log(u) / (lambda * exp(eff$b1 * x1))) + (1 - useExp) * (-log(u) / (lambda * exp(eff$b

  # two features
  # x2 = rbinom(n, 1, 0.5)
  # t =  useExp * (-log(u) / (lambda * exp(eff$b1 * x1))) + (1 - useExp) * (-log(u) / (lambda * exp(eff

  t[t < 1/365] = 1/365
  t[t == 1 / 365] = t[t == 1 / 365] + rnorm(length(t[t == 1 / 365]), 0, 1e-4)
  t = abs(t)

  e = as.numeric(t < 5)
  t = pmin(t, 5)

  name = paste("n =", n, ", p =", p, ", eff =", eff[[1]])

  return(tibble(name = name, time = t, event = e, x1 = x1))
  # two features
  # return(tibble(name = name, time = t, event = e, x1 = x1, x2 = x2))
}
```

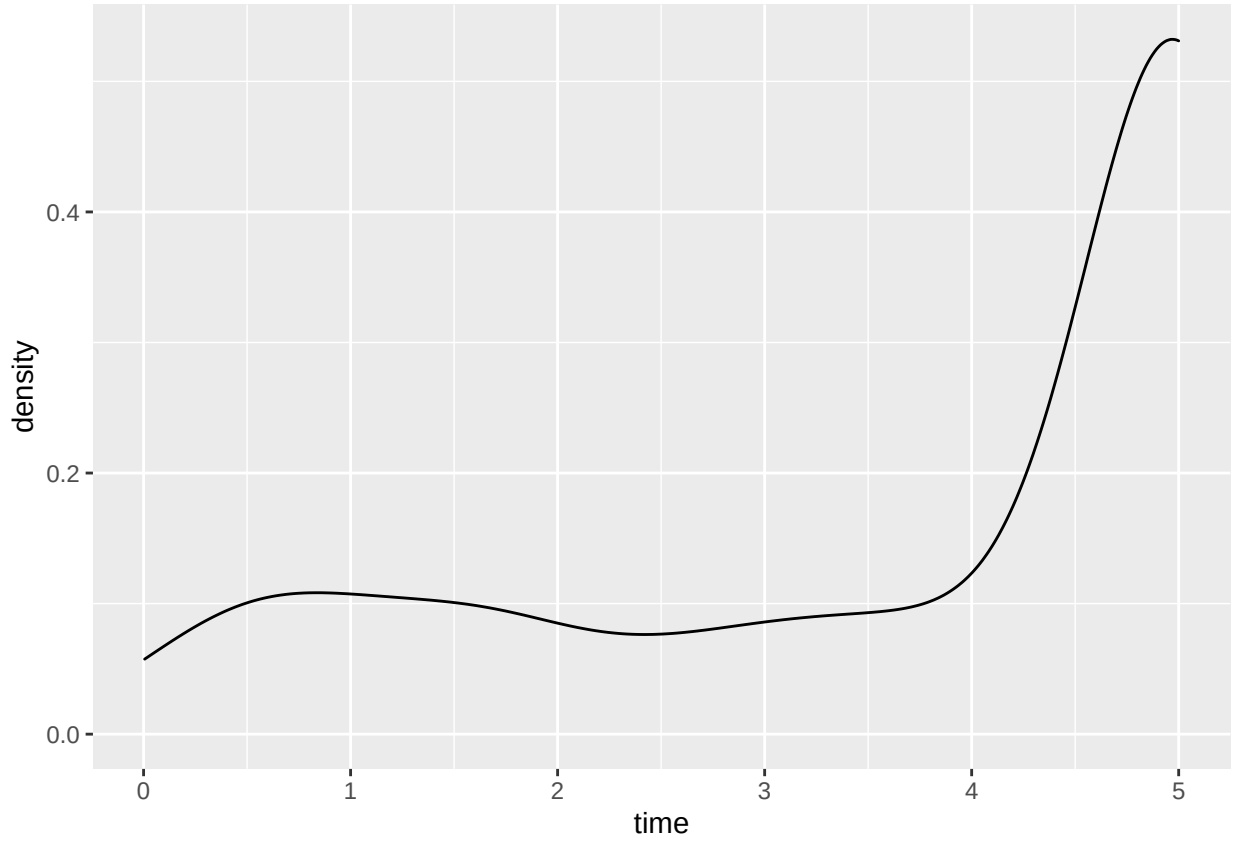
- mixing exponential and gompertz

```
simdat2 = function(n, p, lambda = 0.1, alpha = 0.5, eff = list()) {
  # randomly assign group
  x1 = rbinom(n, 1, 0.5)

  u = runif(n)
```

```
  useExp = runif(n) < p
  # generating mixed data
  t =  useExp * (-log(u) / (lambda * exp(eff$b1 * x1))) + (1 - useExp) * (1/alpha) * log(1-(alpha*log(u)

  # two features
  # x2 = rbinom(n, 1, 0.5)
  # t =  useExp * (-log(u) / (lambda * exp(eff$b1 * x1))) + (1 - useExp) * (-log(u) / (lambda * exp(eff

  t[t < 1/365] = 1/365
  t[t == 1 / 365] = t[t == 1 / 365] + rnorm(length(t[t == 1 / 365]), 0, 1e-4)
  t = abs(t)

  e = as.numeric(t < 5)
  t = pmin(t, 5)

  name = paste("n =", n, ", p =", p, ", eff =", eff[[1]])

  return(tibble(name = name, time = t, event = e, x1 = x1))
  # two features
  # return(tibble(name = name, time = t, event = e, x1 = x1, x2 = x2))
}
```

- test:

```
data = simdat(500, 1, eff = list(b1 = 0.5))

data %>%
  ggplot() +
  geom_density(aes(x = time))
```



fit function

```
fit_exp = function(df) {
  fit.exponential = survreg(Surv(time, event) ~ x1, dist = "exponential", data = df)
  return(as.numeric(-fit.exponential$coefficients[-1]))
}

fit_weibull = function(df) {
  fit.weibull <- survreg(Surv(time, event) ~ x1, dist = "weibull", data = df)
  return(as.numeric(-fit.weibull$coefficients[-1] / fit.weibull$scale))
}

fit_cox = function(df) {
  fit.cox <- coxph(Surv(time, event) ~ x1, data = df)
  return(as.numeric(fit.cox$coefficients))
}
```

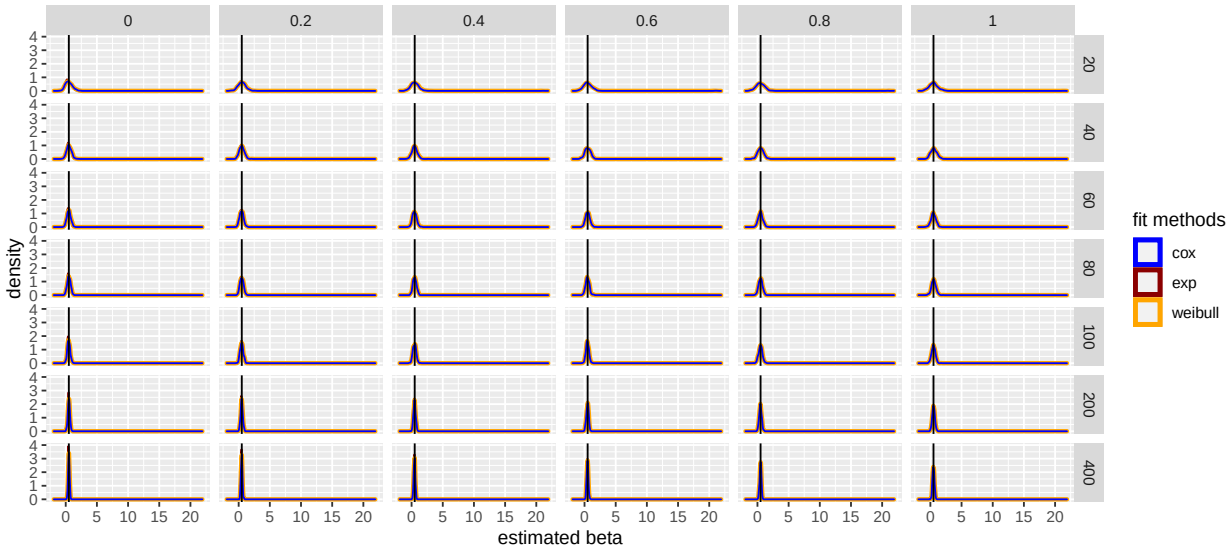
The distribution of estimated beta

- visualization:

```r
param_grid = expand.grid(p = seq(0, 1, by = 0.2), n = c(20, 40, 60, 80, 100, 200, 400), rep = 1:500)

sim_dat = param_grid %>%
  mutate(
    data = map2(n, p, ~simdat(n = .x, p = .y, eff = list(b1 = 0.5)))
  ) %>%
  mutate(
    b_exp = map_dbl(data, fit_exp),
    b_weibull = map_dbl(data, fit_weibull),
    b_cox = map_dbl(data, fit_cox)
  )
```

```
## Warning in fitter(X, Y, istrat, offset, init, control, weights = weights, :
## Loglik converged before variable 1 ; coefficient may be infinite.

## Warning in fitter(X, Y, istrat, offset, init, control, weights = weights, :
## Loglik converged before variable 1 ; coefficient may be infinite.

## Warning in fitter(X, Y, istrat, offset, init, control, weights = weights, :
## Loglik converged before variable 1 ; coefficient may be infinite.

## Warning in fitter(X, Y, istrat, offset, init, control, weights = weights, :
## Loglik converged before variable 1 ; coefficient may be infinite.

## Warning in fitter(X, Y, istrat, offset, init, control, weights = weights, :
## Loglik converged before variable 1 ; coefficient may be infinite.

## Warning in fitter(X, Y, istrat, offset, init, control, weights = weights, :
## Loglik converged before variable 1 ; coefficient may be infinite.

## Warning in fitter(X, Y, istrat, offset, init, control, weights = weights, :
## Loglik converged before variable 1 ; coefficient may be infinite.

## Warning in fitter(X, Y, istrat, offset, init, control, weights = weights, :
## Loglik converged before variable 1 ; coefficient may be infinite.

## Warning in fitter(X, Y, istrat, offset, init, control, weights = weights, :
## Loglik converged before variable 1 ; coefficient may be infinite.

## Warning in fitter(X, Y, istrat, offset, init, control, weights = weights, :
## Loglik converged before variable 1 ; coefficient may be infinite.

## Warning in fitter(X, Y, istrat, offset, init, control, weights = weights, :
## Loglik converged before variable 1 ; coefficient may be infinite.

## Warning in fitter(X, Y, istrat, offset, init, control, weights = weights, :
## Loglik converged before variable 1 ; coefficient may be infinite.

## Warning in fitter(X, Y, istrat, offset, init, control, weights = weights, :
## Loglik converged before variable 1 ; coefficient may be infinite.

## Warning in fitter(X, Y, istrat, offset, init, control, weights = weights, :
## Loglik converged before variable 1 ; coefficient may be infinite.
```

```
sim_dat %>%
  ggplot() +
  geom_density(aes(x = b_exp, color = "darkred"), alpha = 0.6) +
  geom_density(aes(x = b_weibull, color = "orange" ), size = 1.2) +
  geom_density(aes(x = b_cox, color = "blue"), alpha = 0.3) +
  geom_vline(xintercept = 0.5) +
  facet_grid(n~p) +
  xlab("estimated beta") +
  scale_colour_manual(name = 'fit methods',
        values =c('blue'='blue','darkred'='darkred', 'orange' = 'orange'), labels = c('cox','exp', 'weil
```



- metrics:
  - MSE may reflect accuracy: (the trend is indeed very obvious)

```
bias = sim_dat %>%
  select(-data, -rep) %>%
  mutate(
    b_exp = abs(b_exp - 0.5),
    b_weibull = abs(b_weibull - 0.5),
    b_cox = abs(b_cox - 0.5)
  )%>%
  group_by(n, p) %>%
  summarize(
    exp_bias = mean(b_exp),
    weibull_bias = mean(b_weibull),
    cox_bias = mean(b_cox)
  ) %>%
  pivot_longer("exp_bias":"cox_bias", names_to = "fit_method", values_to = "bias") %>%
  mutate(fit_method = as.factor(fit_method))
```

```
## `summarise()` has grouped output by 'n'. You can override using the `.groups`
## argument.
```

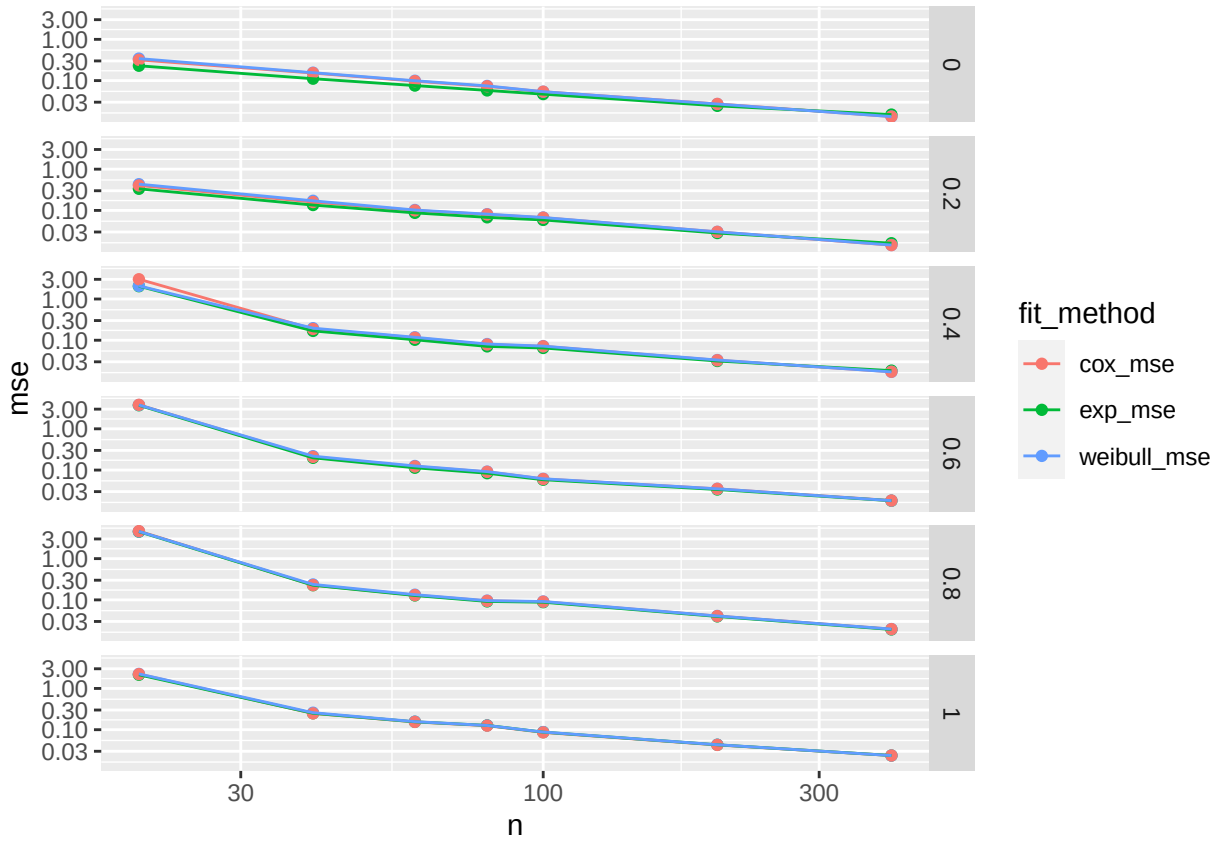
```r
var = sim_dat %>%
  select(-data, -rep) %>%
  group_by(n, p) %>%
  summarize(
    exp_var = var(b_exp),
    weibull_var = var(b_weibull),
    cox_var = var(b_cox)
  ) %>%
  pivot_longer("exp_var":"cox_var", names_to = "fit_method", values_to = "var") %>%
  mutate(fit_method = as.factor(fit_method))
```

```
## `summarise()` has grouped output by 'n'. You can override using the `.groups`
## argument.
```

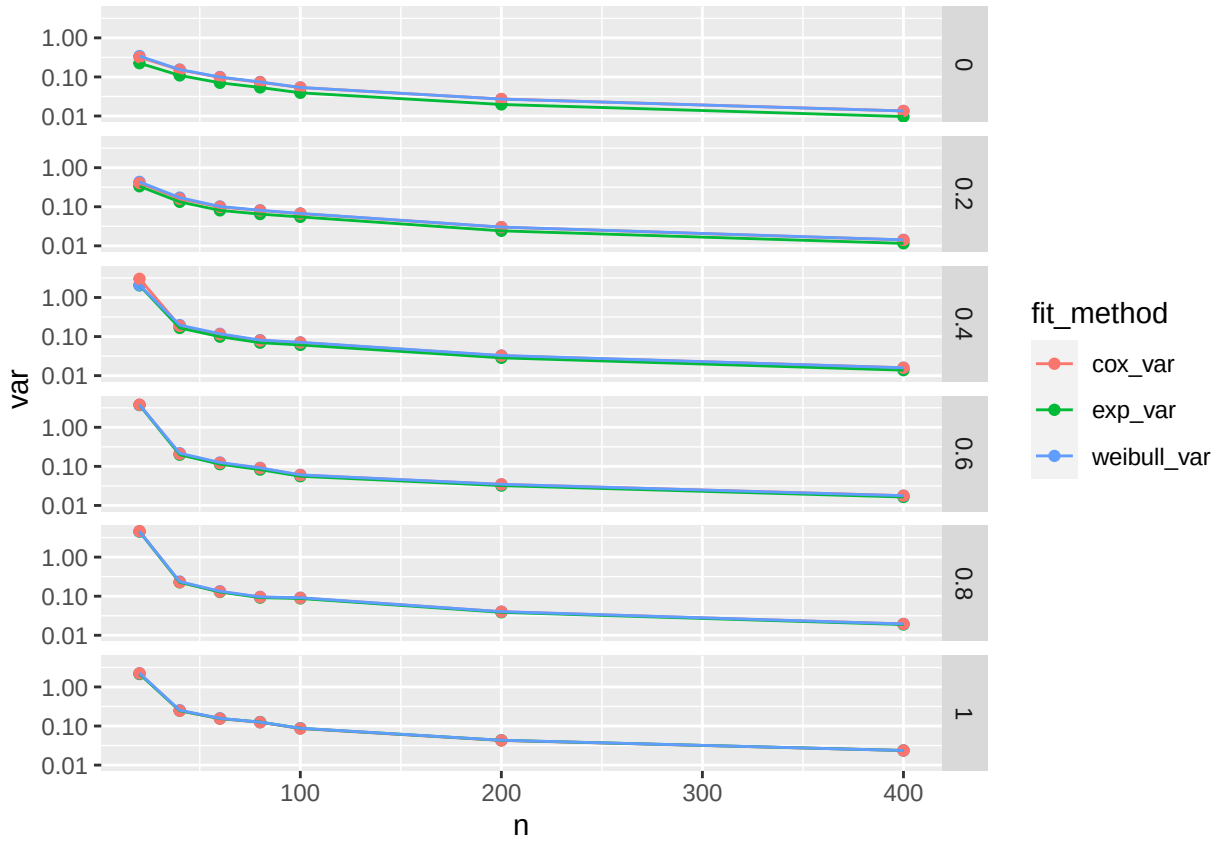
```r
mse = sim_dat %>%
  select(-data, -rep) %>%
  mutate(
    b_exp = (b_exp - 0.5)^2,
    b_weibull = (b_weibull - 0.5)^2,
    b_cox = (b_cox - 0.5)^2
  ) %>%
  group_by(n, p) %>%
  summarize(
    exp_mse = mean(b_exp),
    weibull_mse = mean(b_weibull),
    cox_mse = mean(b_cox)
  ) %>%
  pivot_longer("exp_mse":"cox_mse", names_to = "fit_method", values_to = "mse") %>%
  mutate(fit_method = as.factor(fit_method))
```

```
## `summarise()` has grouped output by 'n'. You can override using the `.groups`
## argument.
```

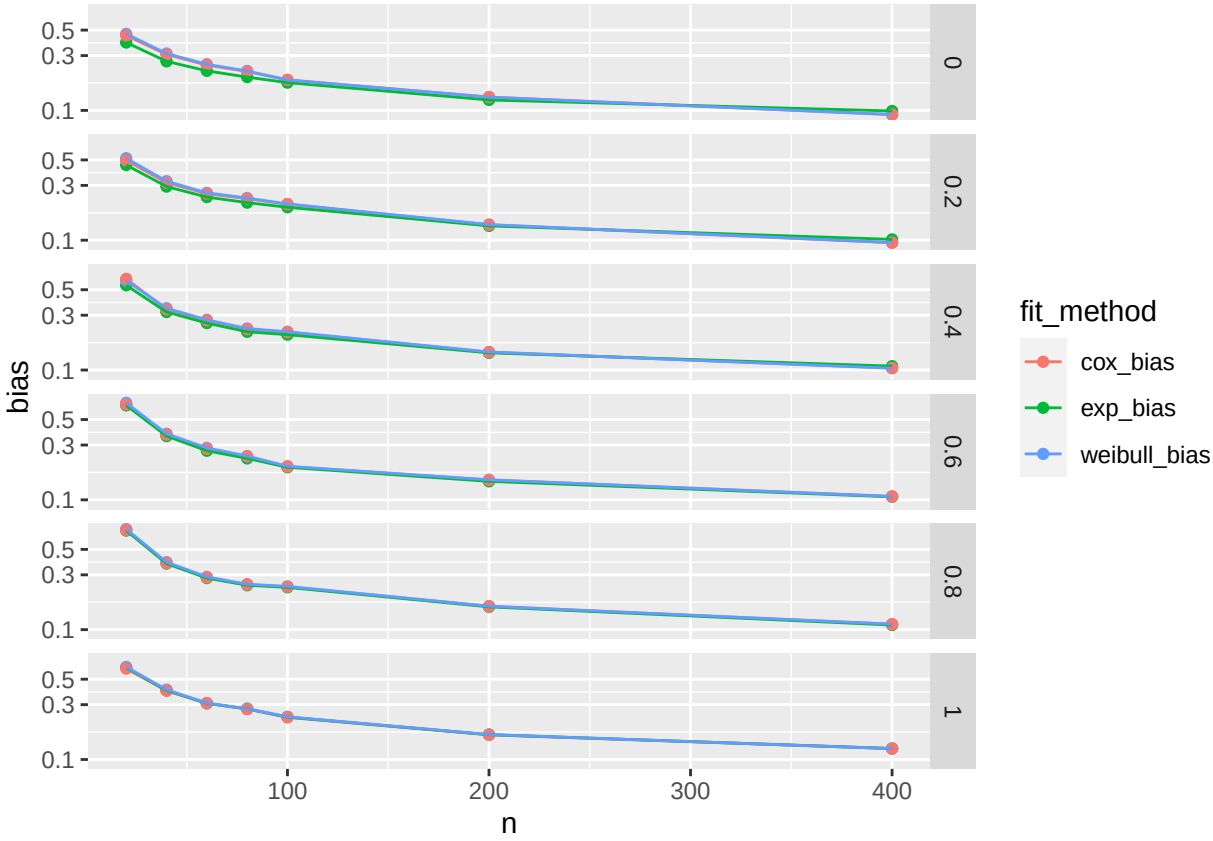
```r
ggplot(mse, aes(x = n, y = mse, color = fit_method)) +
  geom_point() +
  geom_line() +
  scale_y_log10() +
  scale_x_log10() +
  facet_grid("p")
```




```
ggplot(var, aes(x = n, y = var, color = fit_method)) +
  geom_point() +
  geom_line() +
  scale_y_log10() +
  facet_grid("p")
```




```
ggplot(bias, aes(x = n, y = bias, color = fit_method)) +
  geom_point() +
  geom_line() +
  scale_y_log10() +
  facet_grid("p")
```




```
#simdat2(100,0,eff = list(b1 = 0.5))
param_grid2 = expand.grid(p = seq(0, 0.5, by = 0.1), n = c(20, 40, 60, 80, 100, 200, 400), rep = 1:500)
sim_dat2 = param_grid2 %>%
  mutate(
    data = map2(n, p, ~simdat2(n = .x, p = .y, eff = list(b1 = 0.5)))
  ) %>%
  mutate(
    b_exp = map_dbl(data, fit_exp),
    b_weibull = map_dbl(data, fit_weibull),
    b_cox = map_dbl(data, fit_cox)
  )
```

```
## Warning in survreg.fit(X, Y, weights, offset, init = init, controlvals =
## control, : Ran out of iterations and did not converge
```

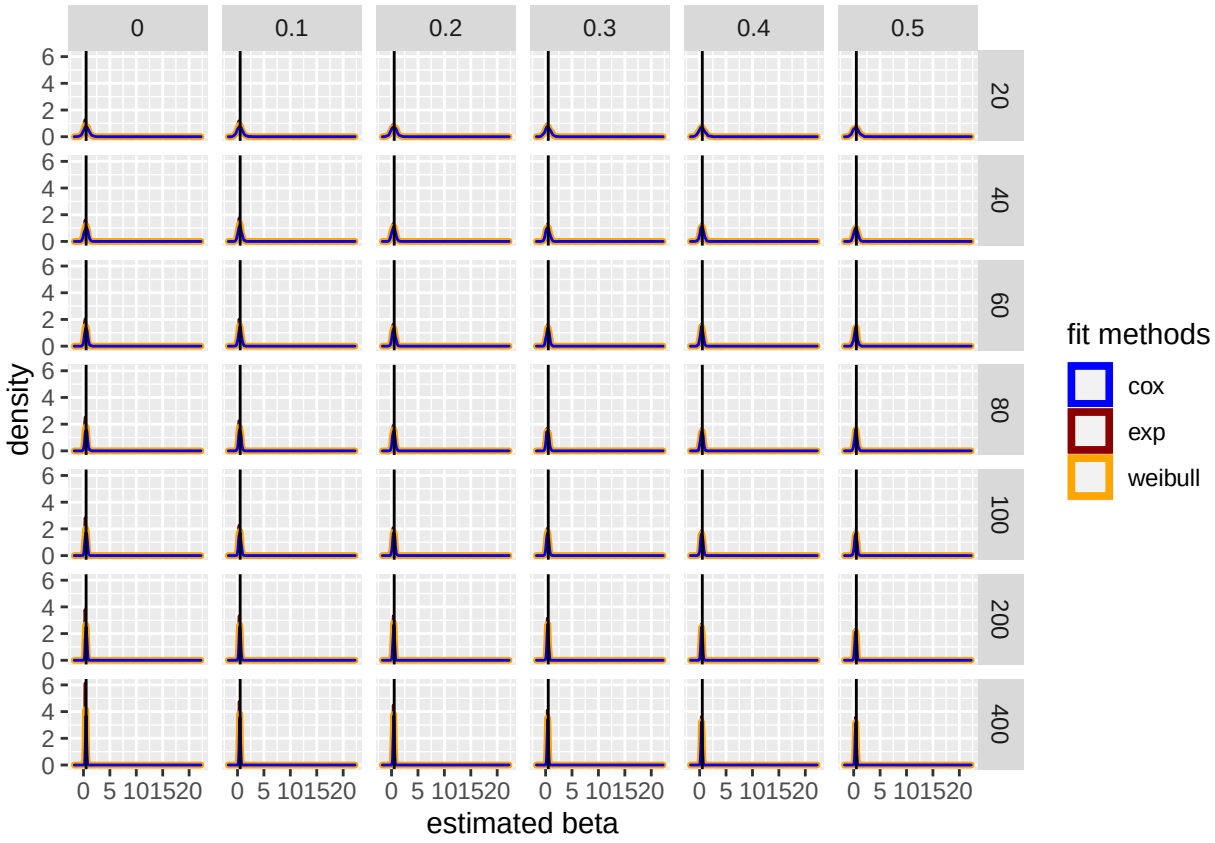
```
## Warning in fitter(X, Y, istrat, offset, init, control, weights = weights, :
## Loglik converged before variable 1 ; coefficient may be infinite.
```

```
## Warning in fitter(X, Y, istrat, offset, init, control, weights = weights, :
## Loglik converged before variable 1 ; coefficient may be infinite.
```

```
## Warning in fitter(X, Y, istrat, offset, init, control, weights = weights, :
## Loglik converged before variable 1 ; coefficient may be infinite.
```

```
sim_dat2 %>%
  ggplot() +
  geom_density(aes(x = b_exp, color = "darkred"), alpha = 0.6) +
  geom_density(aes(x = b_weibull, color = "orange" ), size = 1.2) +
  geom_density(aes(x = b_cox, color = "blue"), alpha = 0.3) +
  geom_vline(xintercept = 0.5) +
  facet_grid(n~p) +
  xlab("estimated beta") +
  scale_colour_manual(name = 'fit methods',
        values =c('blue'='blue','darkred'='darkred', 'orange' = 'orange'), labels = c('cox','exp', 'weil
```

```
## Warning: Removed 1 rows containing non-finite values (stat_density).
```



- analysis of robustness

```
bias2 = sim_dat2 %>%
  select(-data, -rep) %>%
  mutate(
    b_exp = abs(b_exp - 0.5),
    b_weibull = abs(b_weibull - 0.5),
    b_cox = abs(b_cox - 0.5)
  )%>%
  group_by(n, p) %>%
  summarize(
```

```
    exp_bias = mean(b_exp),
    weibull_bias = mean(b_weibull),
    cox_bias = mean(b_cox)
  ) %>%
  pivot_longer("exp_bias":"cox_bias", names_to = "fit_method", values_to = "bias") %>%
  mutate(fit_method = as.factor(fit_method))
```

```
## `summarise()` has grouped output by 'n'. You can override using the `.groups`
## argument.
```

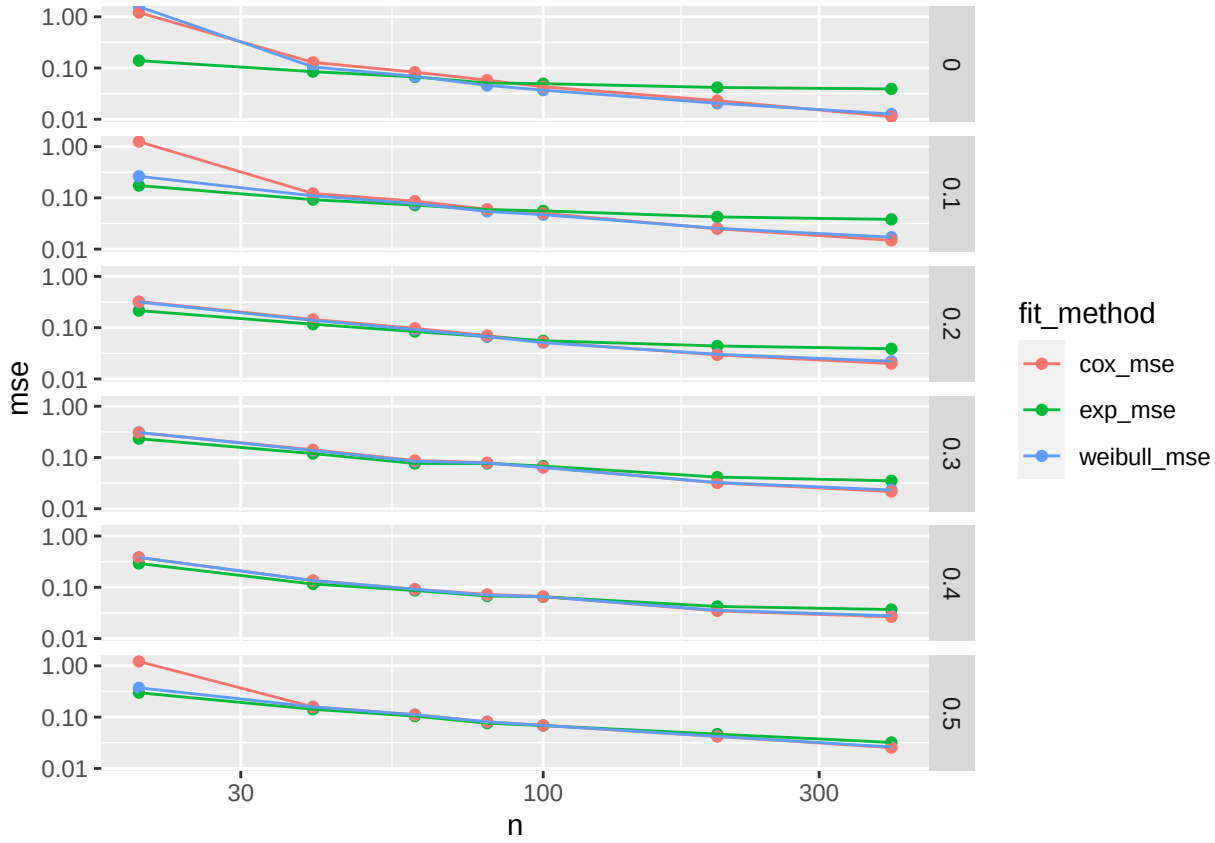
```
var2 = sim_dat2 %>%
  select(-data, -rep) %>%
  group_by(n, p) %>%
  summarize(
    exp_var = var(b_exp),
    weibull_var = var(b_weibull),
    cox_var = var(b_cox)
  ) %>%
  pivot_longer("exp_var":"cox_var", names_to = "fit_method", values_to = "var") %>%
  mutate(fit_method = as.factor(fit_method))
```

```
## `summarise()` has grouped output by 'n'. You can override using the `.groups`
## argument.
```

```
mse2 = sim_dat2 %>%
  select(-data, -rep) %>%
  mutate(
    b_exp = (b_exp - 0.5)^2,
    b_weibull = (b_weibull - 0.5)^2,
    b_cox = (b_cox - 0.5)^2
  ) %>%
  group_by(n, p) %>%
  summarize(
    exp_mse = mean(b_exp),
    weibull_mse = mean(b_weibull),
    cox_mse = mean(b_cox)
  ) %>%
  pivot_longer("exp_mse":"cox_mse", names_to = "fit_method", values_to = "mse") %>%
  mutate(fit_method = as.factor(fit_method))
```

```
## `summarise()` has grouped output by 'n'. You can override using the `.groups`
## argument.
```

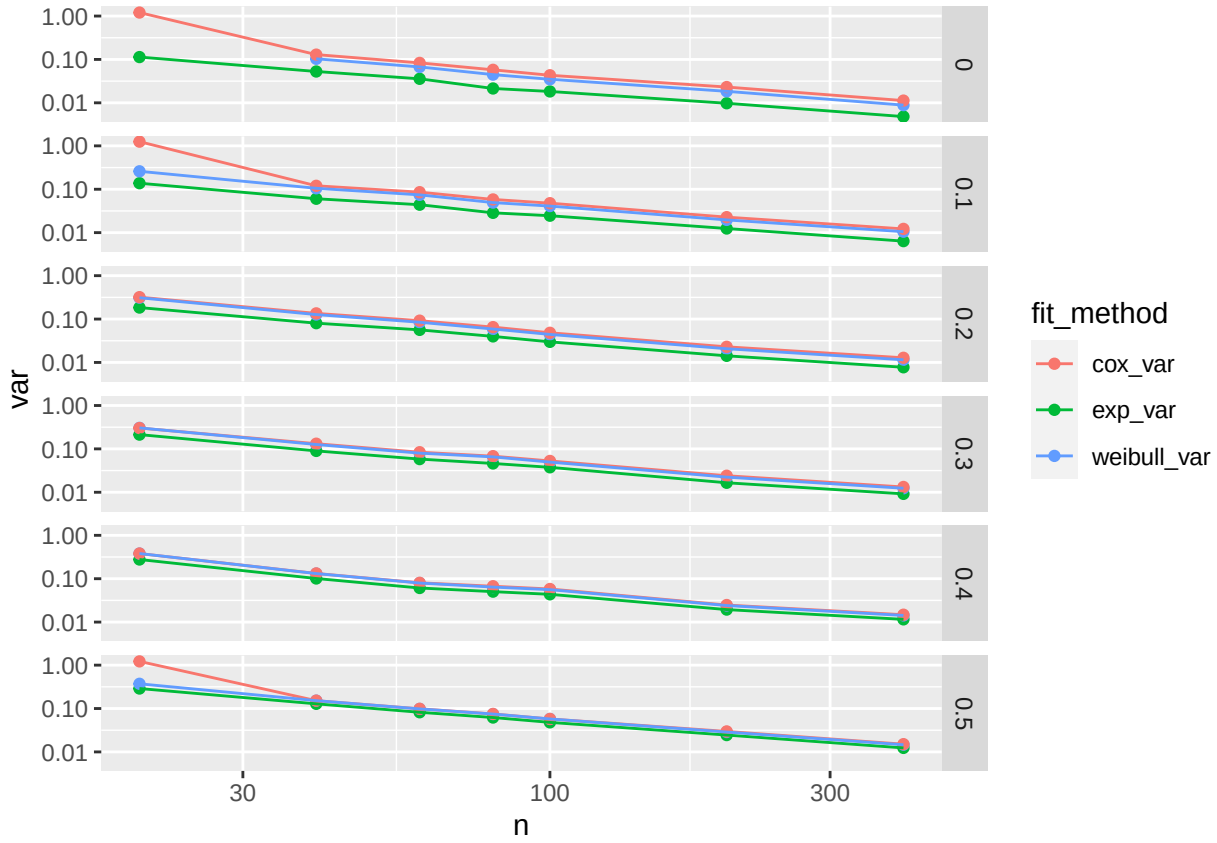
```
ggplot(mse2, aes(x = n, y = mse, color = fit_method)) +
  geom_point() +
  geom_line() +
  scale_y_log10() +
  scale_x_log10() +
  facet_grid("p")
```



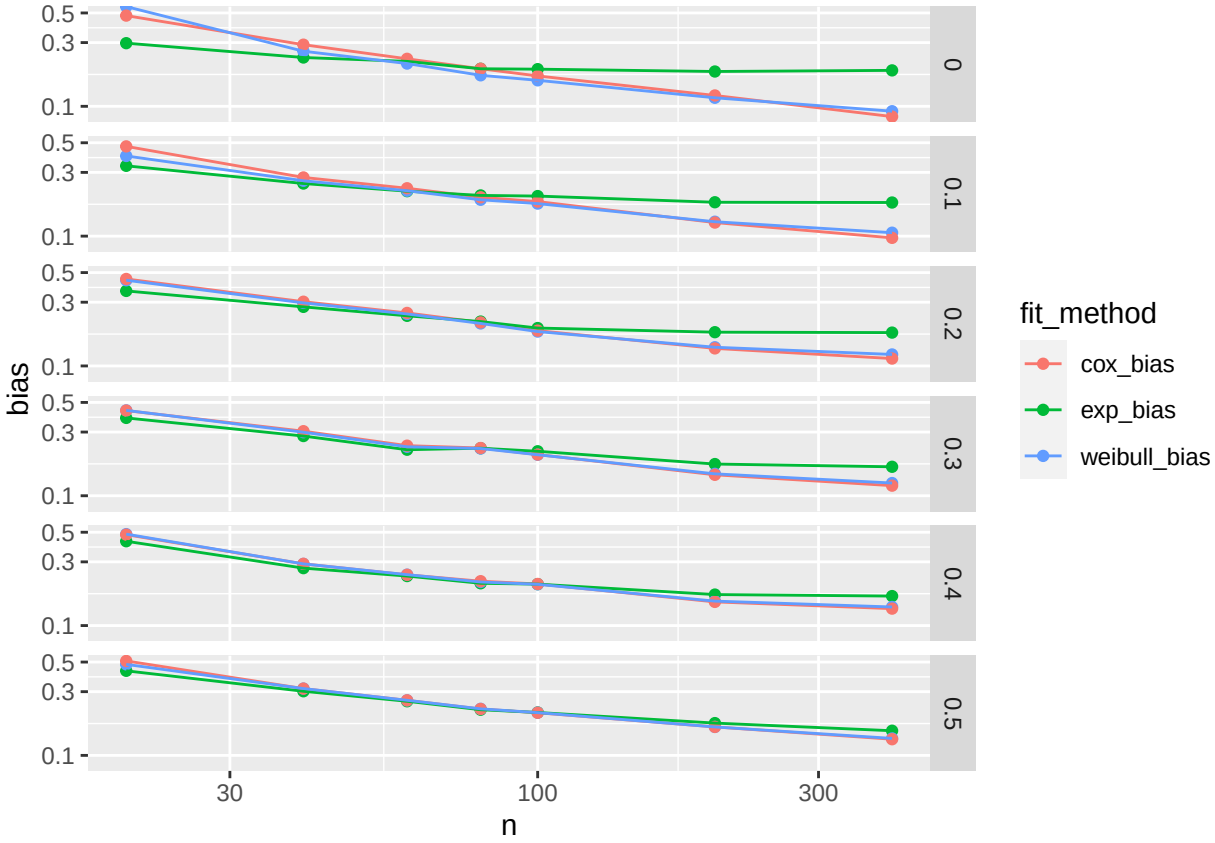

```
ggplot(var2, aes(x = n, y = var, color = fit_method)) +
  geom_point() +
  geom_line() +
  scale_y_log10() +
  scale_x_log10() +
  facet_grid("p")
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

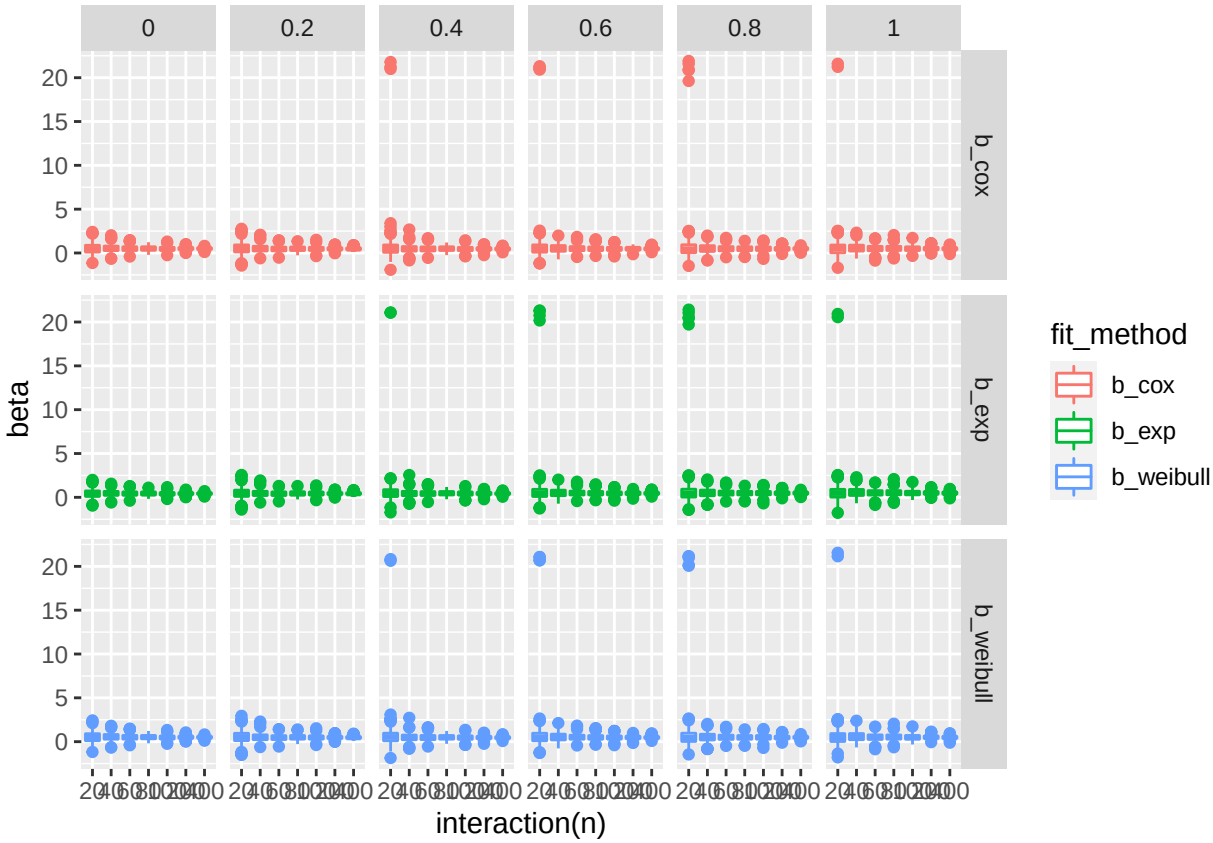
```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```




```
ggplot(bias2, aes(x = n, y = bias, color = fit_method)) +
  geom_point() +
  geom_line() +
  scale_y_log10() +
  scale_x_log10() +
  facet_grid("p")
```




```
sim_dat %>%
  select(-data, -rep) %>%
  pivot_longer("b_exp":"b_cox", names_to = "fit_method", values_to = "beta") %>%
  ggplot(aes(x = interaction(n), y = beta, color = fit_method)) +
  geom_boxplot() +
  facet_grid(fit_method~ p)
```


0
0.2
0.4
0.6
0.8
1
b_cox
b_exp
b_weibull
beta
interaction(n)
fit_method
b_cox
b_exp
b_weibull