BIOS602: Assignment 2

Nam-Anh Tran

Table of contents

Simulations - Study Design

Question 1

We employ the formula of sample size calculation proposed by Schouten [1] for *two means, unequal variances and sample size – independent samples* as follows

$$n_1 \geq (Z_{1-\alpha/2} + Z_{1-\beta})^2 \frac{(\tau+\gamma)\sigma_2^2}{\gamma(\mu_2-\mu_1)^2} + \frac{(\tau^2+\gamma^3)Z_{1-\alpha/2}^2}{2\gamma(\tau+\gamma)^2}, \quad \text{and} \quad n_2 \geq \gamma n_1$$

where $\tau = \sigma_2^2/\sigma_1^2$ and $\gamma = n_2/n_1$. Thus

$$Z_{1-\beta} \leq \sqrt{\left(n - \frac{(\tau^2+\gamma^3)Z_{1-\alpha/2}^2}{2\gamma(\tau+\gamma)^2}\right)\frac{\gamma(\mu_2-\mu_1)^2}{(\tau+\gamma)\sigma_2^2}} - Z_{1-\alpha/2}$$

The power is then calculated using the CDF of the standard normal distribution $\Phi(Z)$. For the sake of simplicity, we assume $\tau = \gamma = 1$, but the general cases $(\tau, \gamma \neq 1)$ can also be implemented with ease based on the above formula. With the total sample size (both arms) of 100, 500 and 1000, we have:

```
pacman::p_load(tidyverse, kableExtra, simstudy, rsimsum,  survival, eha, SurvRegCensCov, MASS,
↪  flexsurv, ggfortify)

options(scipen = 4444)
CalPower = function(n,mu1,mu2, sigma, tau =1, gam = 1, alpha=0.05){

  Z = sqrt((n - (tau^2+
↪  gam^3)*qnorm(1-alpha/2)/2/gam/(tau+gam)^2)*gam*(mu2-mu1)^2/sigma^2/(tau+gam)) -
↪  qnorm(1-alpha/2)

  return(round(pnorm(Z)*100,3))
}
CalPower = Vectorize(CalPower, vectorize.args = "n")

n1 = (c(100, 500, 1000)/2)|> purrr::set_names()

power = CalPower(n= n1, mu1=  0,mu2 = 1,sigma = sqrt(50), tau = 1, gam = 1); power
```

```
    50    250    500
10.450 35.183 60.834
```

Note that to achieve the power of at least 80%, the sample size per arm is at least 790, which is shown below:

```
ss = (seq(760, 825 , 5))|> purrr::set_names()
sim = CalPower(n= ss, mu1=  0,mu2 = 1,sigma = sqrt(50), tau = 1, gam = 1)
```

```
ifelse(sim >=80, paste0(sim,"*"), paste0(sim))|>
  enframe()|>
  rename(`sample size` = name, power = value)|>
  {\(i) kbl(list(i[1:7,], i[8:14,]) ,format = "latex", booktabs = T)}()
```

sample size	power	sample size	power
760	78.697	795	80.476*
765	78.959	800	80.719*
770	79.219	805	80.96*
775	79.476	810	81.198*
780	79.73	815	81.434*
785	79.981	820	81.667*
790	80.23*	825	81.897*

Question 2

```
set.seed(1111)
R = 10000
SimPower = function(R=10000, n1, sigma){
  pow<-
    replicate(10000,{
      y1 = rnorm(n1,0,sigma ); y2 = rnorm(n1,1,sigma)
      Z = abs((mean(y2) - mean(y1))/sigma/sqrt(1/length(y1) + 1/length(y2)))
      Z> qnorm(0.975)
      })|>
    mean()

  return(round(pow*100,3))
}
n2 = c(n1, 790)|> purrr::set_names()
sapply(n2, SimPower, R = R, sigma = sqrt(50))
```

```
   50   250   500   790
10.98 34.72 60.82 80.65
```

This result obtained using simulations does not differ from the previous result considerably.

Suppose that we expect the Monte Carlo SE of power is 0.5%, corresponding number of iterations n_{sim} can then be specify using the formula

$$n_{sim} = \frac{\widehat{\text{Power}} \times (1 - \widehat{\text{Power}})}{\text{Monte Carlo SE(Power)}^2},$$

we now calculate n_{sim} using the powers obtained above associated with 4 sample size per arm (50, 250, 500, 790).

```
set.seed(1111)
power = CalPower(n= n2, mu1=  0,mu2 = 1,sigma = sqrt(50), tau = 1, gam = 1)
nsim = ceiling(power*(100-power)/0.5^2); nsim
```

```
  50  250  500  790
3744 9122 9531 6345
```

Thus, to keep the required Monte Carlo SE of 0.5% (for all 4 cases), we need at least $n_{sim} = 9531$. We choose $n_{sim} = 10000$ and, thus, the corresponding Monte Carlo SE are

```
se = sqrt(power*(100-power)/10000); round(se,3)
```

```
   50   250   500   790
0.306 0.478 0.488 0.398
```

which are lower than 0.5

Question 3

```
sapply(n2, SimPower, R = 10000, sigma = sqrt(100))
```

```
   50   250   500   790
 8.00 19.52 34.29 50.79
```

As we increase the SD, the powers reduce given that we do not change the sample sizes.

Simulations - Confounding

Question 2

Based on the the section of data generation, we have 10 covariates involving:

- *Continuous variables:* W_2, W_4, W_7, W_{10},

- *Binary variables:* $W_1, W_3, W_5, W_6, W_8, W_9$,

where correlated variables are:

- $\text{Cor}(W_1, W_5) = \text{Cor}(W_3, W_8) = 0.2$,
- $\text{Cor}(W_2, W_6) = \text{Cor}(W_4, W_9) = 0.9$.

The first case is the correlation between 2 binary variables while the second is the correlation between a binary and a continuous variables. There is no correlation condition on any two continuous variables.

In lieu of following the procedure described in the paper, which requires two steps of generation, we can simulate the continuous covariates directly based on the multinormal distribution.

For the correlation of two binary variables, we can improve the attenuation caused by dichotimising (which was not mentioned in the paper). To this end, We use the method proposed by Emrich et. al. [2], i.e. we pre-specify the correlation ρ_{prior} of two pre-dichotomised variables such that the correlation ρ_{post} of two corresponding post-dichotomised variables reaches the target value of correlation.

For the correlation of a binary and a continuous variables, we can also improve the attenuation using the probit function input of which is continuous variable and the pre-specified slope. The corresponding binary variable is calculated using the inverse CDF of bernoulli distribution. This approach allows adjusting the correlation based also on the slope.

```
n = 2000
# create the covariance matrix
covmat =  diag(1, 10)
pre_rho = simstudy:::findRhoBin(0.5,0.5, 0.2) # find the pre rho
covmat[1,5]<- covmat[5,1]<- pre_rho # update corr
covmat[3,8]<- covmat[8,3]<- pre_rho # update corr
# pre_rho = simstudy:::findRhoBin(0.5,0.5, 0.9)
covmat[2,6]<- covmat[6,2]<- 0.9 # update corr
covmat[4,9]<- covmat[9,4]<- 0.9 # update corr


set.seed(11)
df<- MASS::mvrnorm(n, mu = rep(0,10), Sigma = covmat)

# a = ifelse(df[,4] < mean(df[,4]),1,0)
# b = ifelse(df[,9] < mean(df[,9]),1,0)
# cor(a,b)
```

```
# dichotomise variables
df[,c(1,3,5,8)]<- ifelse(df[,c(1,3,5,8)] < qnorm(0.5),1,0)
# check the correlation of two binary variables.
df[,c(1,5)]|> cor(); df[,c(3,8)]|> cor()
```

```
          [,1]     [,2]
[1,] 1.000000 0.214771
[2,] 0.214771 1.000000
```

```
           [,1]      [,2]
[1,] 1.0000000 0.1924738
[2,] 0.1924738 1.0000000
```

The above results show that the correlation of (W_1, W_5) and (W_3, W_8) are close to the target value, which is 0.2. We now improve the correlation between binary and continuous variables by adjusting the slope and intercept (actually the slope only).

```
# improve correlation of bin-con variables using the probit function.
beta1 = 2; beta0 = 0.001
p6 = pnorm(beta1*df[,2]+beta0)
p9 = pnorm(beta1*df[,4]+ beta0)
# dichotomise variable
df[,6]<- qbinom(p6, 1, 0.5)
df[,9]<- qbinom(p9, 1, 0.5)
# check the correlation of binary & continuous.
df[,c(2,6)]|> cor();df[,c(4,9)]|> cor()
```

```
           [,1]      [,2]
[1,] 1.0000000 0.7940715
[2,] 0.7940715 1.0000000
```

```
           [,1]      [,2]
[1,] 1.0000000 0.7982196
[2,] 0.7982196 1.0000000
```

The best result of correlation that can be achieved using this method is close to 0.8. Although this goes off the target value of 0.9, it is the best we can achieve.

We now repeat the procedure above to simulate 1000 datasets:

```
R = 1000
dat = replicate(R,{
  df<- MASS::mvrnorm(n, mu = rep(0,10), Sigma = covmat)
  # dichotomise variables
  df[,c(1,3,5,8)]<- ifelse(df[,c(1,3,5,8)] < qnorm(0.5),1,0)

  # improve correlation of bin-con variables using the probit function.
  beta1 = 2; beta0 = 0.1
  p6 = pnorm(beta1*df[,2]+beta0)
  p9 = pnorm(beta1*df[,4]+ beta0)
  # dichotomise variable
  df[,6]<- qbinom(p6, 1, 0.5)
  df[,9]<- qbinom(p9, 1, 0.5)
  return(df)
})
```

We now simulate the exposure and outcome for each dataset

```
bet = c(0.8, -0.25, 0.6, -0.4, -0.8, -0.5, 0.7)
alpha = c(-3.85, 0.3, -0.36, -0.73, -0.2, 0.71, -0.19, 0.26, -0.4)

set.seed(11)

dat2<- apply(dat,3, \(df){

  # simulates the exposure and outcome variables for each
  # simulated dataset. Its input is matrix of covariate W1-W10

  prob =  boot::inv.logit(df[,1:7]%*%bet) # calculate the probability using the inverse logit
↪  function
  A = ifelse(prob > runif(n),1, 0) #  dichotomise exposure
  df1<- cbind(df,A)
  prob =  boot::inv.logit(cbind(1,df1[,c(1:4,8:11)])%*%alpha) # calculate the probability
↪  using the inverse logit function. It is important to note that I used the formula
↪  exp(x)/1[+exp(x)], not 1/[1+exp(x)]. This wouldn't change the result as you defined the <
↪  or > to dichotomise. To check which one is correct, we will calculate the probability of A
↪  and Y, if they are =~ 0.05 ans 0.02, then we are fine.
```

```
  Y = ifelse(prob > runif(n),1, 0) #  dichotomise outcome
  df3<- cbind(df1, Y)
  colnames(df3) <- c(paste0("w",1:10), "A", "Y") # set names of columns

  # glm(df3[,"Y"] ~ df3[,"A"], family = "binomial")$coefficients[2]
  as_tibble(df3)|>
    mutate_at(c(1,3,5,6,8,9,11,12), as_factor)

  }, simplify = F)
```

We now check the average probability of A and Y and correlations of covariates

```
# prob
sapply(dat2, \(i) c(A = mean(i$A==1), Y = mean(i$Y==1)))|>
  apply(1,mean)
```

```
       A        Y
0.508027 0.022484
```

```
# corr bin-bin
sapply(dat2, \(i){
  c15<- cor(i$w1==1,i$w5==1);
  c38<- cor(i$w3==1,i$w8==1)
  c(c15=c15, c38=c38)
}) |>
  apply(1, mean)
```

```
      c15       c38
0.1989620 0.2008593
```

```
# corr bin-cont
sapply(dat2, \(i){
  c15<- cor(i$w2,i$w6==1);
  c38<- cor(i$w4,i$w9==1)
  c(c15=c15, c38=c38)
}) |>
  apply(1, mean)
```

```
      c15       c38
0.7976293 0.7976587
```

(The result shows that we are fine to move on)

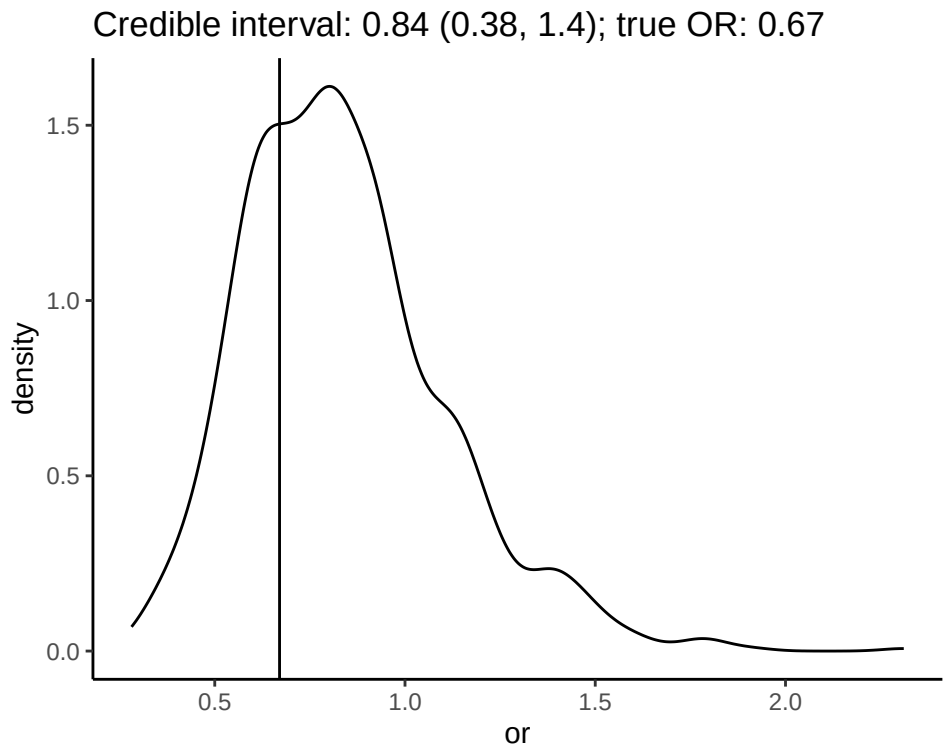
Question 3

```
dats = tibble(dat = dat2)|> # shrink all simulated datasets to tibble (data.frame)
  # fit the logistic model for each dataset
  mutate(fit = map(dat, ~ glm(Y ~ A, data = ., family = "binomial")))|>
  # summary the output of glm
  mutate(summ = map(fit, purrr::compose(~ slice(.,2),broom::tidy)))|>
  unnest(summ)|>
  # calculate OR from logOR
  mutate(or = exp(estimate))|>
  # select main statistics, i.e. or and SD
  dplyr::select(or, std.error)|>
  # add dataset number
  mutate(ds = 1:R, .before = 1)

glimpse(dats)
```

```
Rows: 1,000
Columns: 3
$ ds        <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 1~
$ or        <dbl> 0.8049030, 0.5158957, 0.7363512, 0.5842874, 1.1397306, 0.806~
$ std.error <dbl> 0.3384327, 0.3780765, 0.2843211, 0.3034329, 0.2788686, 0.306~
```

We calculate the mean and credible interval, and plot the density curve of OR as follows.

```
or = mean(dats$or)|> round(2);
crinv = HDInterval::hdi(dats$or)|>
  round(2)|>
  str_flatten(", ")
ci = paste0("Credible interval: ", or, " (", crinv,"); ","true OR: ",round(exp(-0.4),2))

ggplot(data = dats, aes(x = or))+ geom_density()+ geom_vline(xintercept = exp(-0.4))+
↪  ggtitle(ci)+ theme_classic()
```



To assess the uncertainty of simulations, we utilise the R package `rsimsum` [3] to summarise the results and report the result based on the suggestions of Morris et. al. [4].

```
(s1 <- simsum(data = dats, estvarname = "or", se = "std.error", true = exp(-0.4), x = TRUE)|>
↪  summary())
```

```
Values are:

    Point Estimate (Monte Carlo Standard Error)


Non-missing point estimates/standard errors:

 Estimate

     1000


Average point estimate:

 Estimate

   0.8402
```

```
Median point estimate:
 Estimate
   0.8080

Average variance:
 Estimate
   0.0962

Median variance:
 Estimate
   0.0937

Bias in point estimate:
        Estimate
 0.1699 (0.0084)

Relative bias in point estimate:
        Estimate
 0.2535 (0.0126)

Empirical standard error:
        Estimate
 0.2666 (0.0060)

Mean squared error:
        Estimate
 0.0999 (0.0061)

Model-based standard error:
```

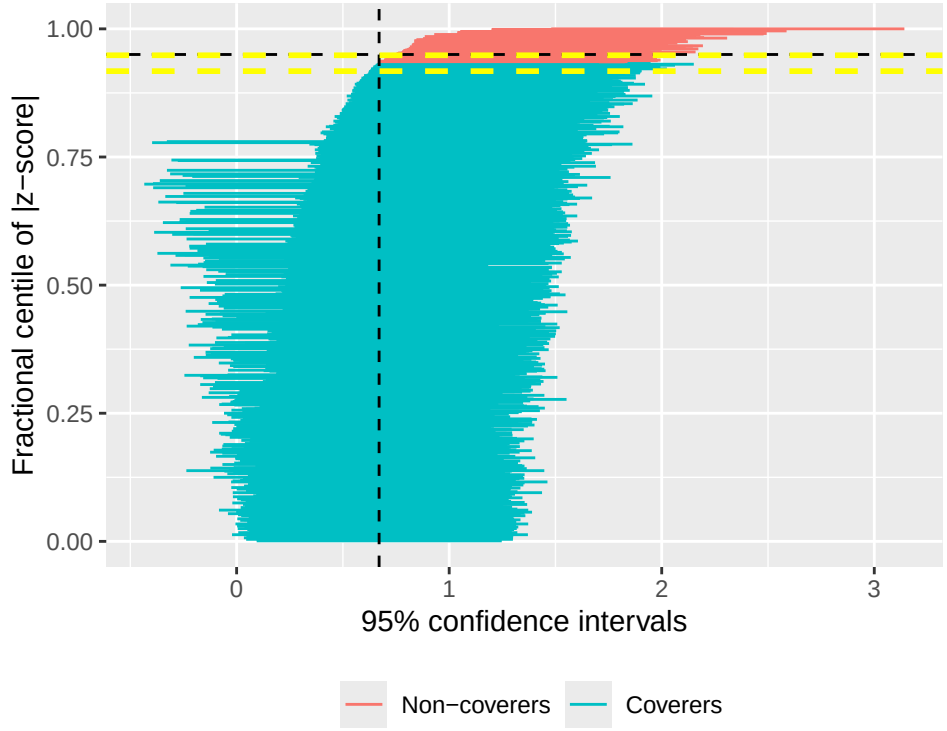
```
        Estimate
 0.3102 (0.0008)


Relative % error in standard error:

         Estimate
 16.3330 (2.6205)


Coverage of nominal 95% confidence interval:

        Estimate
 0.9330 (0.0079)


Bias-eliminated coverage of nominal 95% confidence interval:

        Estimate
 0.9730 (0.0051)


Power of 5% level test:

        Estimate
 0.8150 (0.0123)
```

```
# summarise the output. focus on the Bias and coverage
s1$summ|>
  mutate_if(is.numeric, ~round(.,3))|>
  filter(stat %in% c("bias","cover"))|>
  kbl( format = "latex", booktabs = T)|>
  kable_styling(latex_options = c("striped","hold_position"))
```

stat	est	mcse	lower	upper
bias	0.170	0.008	0.153	0.186
cover	0.933	0.008	0.918	0.948

The bias of 0.17 and the coverage of 0.93 with the Monte Carlo SE of 0.008 indicate a good consistency of the estimate of OR across simulations. We can also create the zip-plot for illustration ad follows:

```
autoplot(s1, type = "zip")
```



question 4

Suppose the model that gives the crude OR is model 1, the model that excludes $W_5 - W_7$ is model 2, and the model that includes all covariates is model 3. We fit all models and obtain the average ORs and credible intervals.

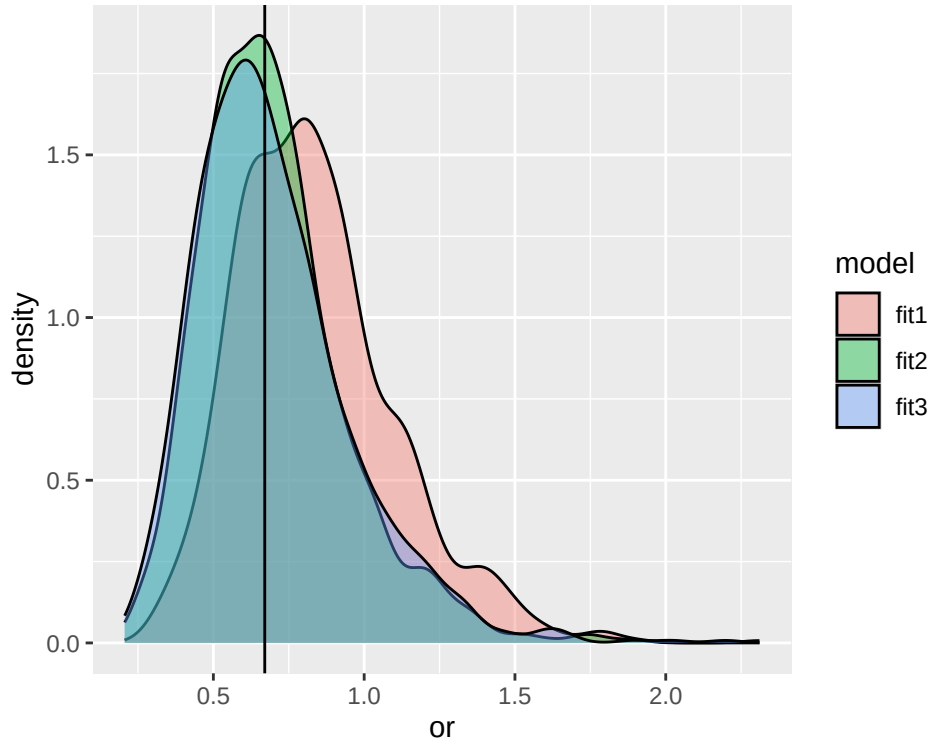
```
dats2 = tibble(dat = dat2)|>
  mutate(fit1 = map(dat, ~ glm(Y ~ A, data = ., family = "binomial")))|>
  mutate(fit2 = map(dat, ~ glm(Y ~ A+ w1+w2+w3+w4+w8+w9+w10, data = ., family = "binomial")
  ↪  ))|>
  mutate(fit3 = map(dat, ~ glm(Y ~ A+ w1+w2+w3+w4+w5+ w6+w7+w8+w9+w10, data = ., family =
  ↪  "binomial") ))

summ_dat<-
mutate(dats2, across(fit1:fit3, ~ map(., purrr::compose(~slice(.,2),broom::tidy) )))|>
  mutate(across(fit1:fit3, ~ map(., \(i) dplyr::select(i,estimate, std.error))))|>
  mutate(across(fit1:fit3, ~ map(., \(i) mutate(i, or = exp(estimate),sd = std.error, .keep =
  ↪  "unused", .before = 1))))|>
```

```
  dplyr::select(-dat)|>
  unnest(names_sep = ".")


or<-
summ_dat|> dplyr::select(contains("or"))|>
  pivot_longer(cols = everything(), names_to = "model", values_to = "or")|>
  mutate(model = str_remove_all(model, "\\.or"))

se <-
  summ_dat|> dplyr::select(contains("sd"))|>
  pivot_longer(cols = everything(), names_to = "model", values_to = "sd")|>
  mutate(model = str_remove_all(model, "\\.sd"))

dat_join<- bind_cols(or, dplyr::select(se, -model))

dat_join|>
  reframe(ave.or = mean(or), lower = HDInterval::hdi(or)[1], upper = HDInterval::hdi(or)[2],
  ↪  .by = model)|>
  mutate(CI.width = upper - lower)|>
  kbl( format = "latex", booktabs = T)|>
  kable_styling(latex_options = c("striped","hold_position"))
```

model	ave.or	lower	upper	CI.width
fit1	0.8402150	0.3822497	1.404421	1.0221713
fit2	0.6976072	0.3050708	1.221964	0.9168929
fit3	0.6982128	0.2813769	1.208706	0.9273289

We plot the density curve of OR of three models as follows:

```
ggplot(dat_join, aes(x = or, fill = model))+ geom_density(alpha = .4)+
  geom_vline(xintercept = exp(-0.4))
```



We then evaluate uncertainty of the models using the `bias` and `coverage` measures that are suggested by Morris et. al. [4].

```
s2 <- simsum(data = dat_join, estvarname = "or", se = "sd", true = exp(-0.4), x = TRUE,
↪  methodvar = "model")
out<- summary(s2)
out$summ|> mutate_if(is.numeric, ~round(.,3))|>
  as_tibble()|>
  filter(stat %in% c("bias","cover"))|>
  kbl( format = "latex", booktabs = T)|>
  kable_styling(latex_options = c("striped", "hold_position") )
```

$W_5 - W_7$ do not impact the estimation significantly. This is indicated by the average OR and its corresponding credible intervals shown above. These values do not go off markedly. This can also be implied by the estimate of `bias` and `cover` in the table above; both `bias` and `cover` are almost the same between two models. Another implication of this is to compare the overlap area between model 2 and model 3 versus model 1 and model 3, the former is large than the later.

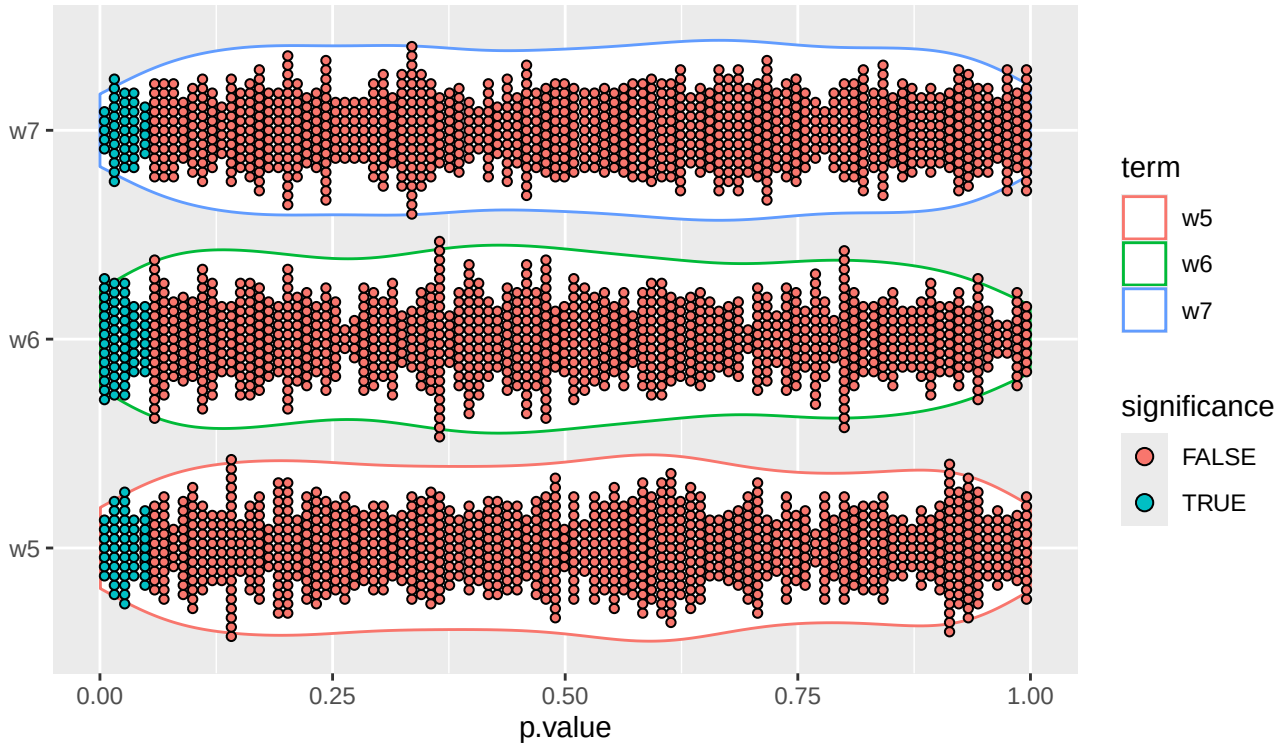
stat	est	mcse	model	lower	upper
bias	0.170	0.008	fit1	0.153	0.186
cover	0.933	0.008	fit1	0.918	0.948
bias	0.027	0.008	fit2	0.012	0.042
cover	0.982	0.004	fit2	0.974	0.990
bias	0.028	0.008	fit3	0.012	0.044
cover	0.982	0.004	fit3	0.974	0.990

This suggests us to explore the estimate of coefficients of these covariates and their p-value to see if they are significant. To this end, we extract information from the fitting object of model 3 and calculate the probability of significance, i.e. the probability that these three corresponding coefficients differ from 0. In other words, this is the probability of rejecting the null hypothesis in the Bayesian framework.

```
fit3<- mutate(dats2, summ = map(fit3, broom::tidy))$summ
sig.dat<-
lapply(fit3, \(i) filter(i, str_detect(term,"5|6|7") ) )|>
  bind_rows()|>
  mutate(significance = p.value <0.05, term = str_remove_all(term,"1"))

summarise(sig.dat, prob.sig = mean(significance),.by = "term")|>
  kbl( format = "latex", booktabs = T)|>
  kable_styling(latex_options = c("striped", "hold_position") )
```

term	prob.sig
w5	0.047
w6	0.056
w7	0.041

```
  ggplot(sig.dat, aes(y = p.value, x = term))+
    geom_violin(aes(color = term))+
    geom_dotplot(binaxis='y', stackdir='center', dotsize=1, binwidth = 1/100, binpositions =
    ↪  "all", aes( fill = significance))+
    coord_flip()+
    labs(x = "")
```



Thus, the probability of significance calculated across 1000 simulated datasets are about 5%.

Survival Data - Analyses

Question 1 & 2

The survivor function under the Weibull distribution has the following form

$$S(t) = \exp\{-(\lambda t)^{\alpha}\}.$$

If we use the scale parameter $\sigma = 1/\alpha$ and $\mu = -\ln \lambda$, we have

$$S_W(t) = \exp\{-e^{-\mu/\sigma} t^{1/\sigma}\}.$$

If we set $\sigma = 1$ we have the survivor exponential function, i.e.

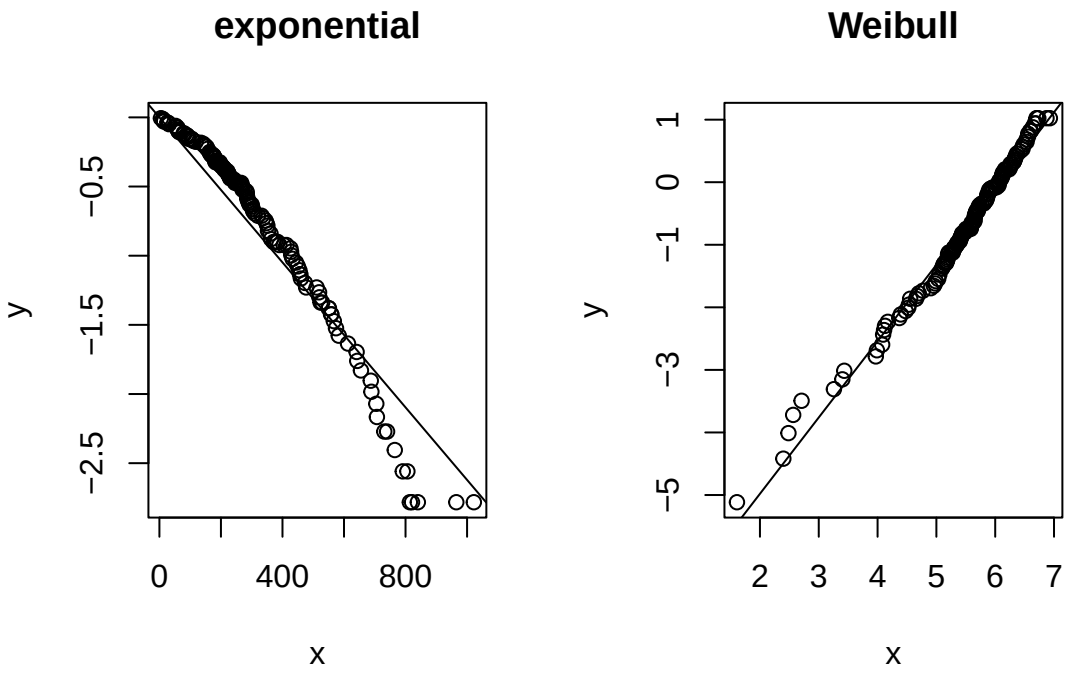
$$S_E(t) = \exp\{-\lambda t\}.$$

If we take $\ln[-\ln(.)]$ and $\ln(.)$ to $S_W(t)$ and $S_E(t)$, respectively, we obtain

$$\ln\{-\ln[S_W(t)]\} = -\mu/\sigma + 1/\sigma \ln(t), \quad \text{and} \quad \ln[S_E(t)] = -\lambda t.$$

These functions suggest a diagnostic tool to evaluate how well a set of survival data follows a Weibull or Exponential distribution.

```
options(scipen = 444)
dat = mutate(survival::cancer|> as_tibble() , ph.karno = ifelse(ph.karno > 80, 1,0)|>
↪  as_factor())|>
  na.omit()|>
  mutate(sex = as_factor(sex))


result.km <- survfit(Surv(time, status) ~ 1, data = dat)
par(mfrow = c(1,2))
## exponential
y = log(result.km$surv)
x = result.km$time
result.lm<- lm(y ~ 0 + x)
plot(y ~ x, main = "exponential")
abline(result.lm)

## Weibull
y<- log(-log(result.km$surv))
x <- log(result.km$time)
result.lm<- lm(y ~  x)
plot(y~ x, main = "Weibull")
abline(result.lm)
```



The plots show the Weibull survivor function fits the data better than the Exponential survivor function.

We now fit all three models.

```
wei.mod <- eha::phreg(Surv(time, status) ~  ph.karno + age+ sex+ ph.ecog + pat.karno +
↪  meal.cal+ wt.loss , data = dat, dist = "weibull", x = TRUE)

# extract information from the output
cap.out = capture.output(wei.mod)[6:17]
cap.out<- cap.out[!str_detect(cap.out, "reference")]
for(i in 2:10) cap.out[i]<- str_replace(cap.out[i],"^ +\\d",cap.out[i-1])
cap.out<- str_replace(cap.out, "Wald p", "Wald.p")
info<- str_split(cap.out, " +")|> purrr::discard(~length(.) <3 )
info<- lapply(info, \(i) i[i!=""] )
summ.wei<-
exec(rbind, !!!info[-1])|>
  `colnames<-`(info[[1]])|>
  as_tibble()|>
  mutate(significance = ifelse(as.numeric(Wald.p) < 0.05, Covariate,""))

#---
```

```
exp.mod <- eha::phreg(Surv(time, status) ~  ph.karno + age+ sex+ ph.ecog + pat.karno +
↪  meal.cal+ wt.loss , data = dat, shape = 1)

cap.out = capture.output(exp.mod)[5:16]
cap.out<- cap.out[!str_detect(cap.out, "reference")]
for(i in 2:10) cap.out[i]<- str_replace(cap.out[i],"^ +\\d",cap.out[i-1])
cap.out<- str_replace(cap.out, "Wald p", "Wald.p")
info<- str_split(cap.out, " +")|> purrr::discard(~length(.) <3 )
info<- lapply(info, \(i) i[i!=""] )
summ.exp<-
exec(rbind, !!!info[-1])|>
  `colnames<-`(info[[1]])|>
  as_tibble()|>
  mutate(significance = ifelse(as.numeric(Wald.p) < 0.05, Covariate,""))

#---

cox.mod<- coxreg(Surv(time, status) ~  ph.karno + age+ sex+ ph.ecog + pat.karno + meal.cal+
↪  wt.loss , data = dat)

cap.out = capture.output(cox.mod)[5:16]
cap.out<- cap.out[!str_detect(cap.out, "reference")]
for(i in 2:10) cap.out[i]<- str_replace(cap.out[i],"^ +\\d",cap.out[i-1])
cap.out<- str_replace(cap.out, "Wald p", "Wald.p")
info<- str_split(cap.out, " +")|> purrr::discard(~length(.) <3 )
info<- lapply(info, \(i) i[i!=""] )
summ.cox<-
exec(rbind, !!!info[-1])|>
  `colnames<-`(info[[1]])|>
  as_tibble()|>
  mutate(significance = ifelse(as.numeric(Wald.p) < 0.05, Covariate,""))|>
  `names<-`(names(summ.exp))

bind_rows(summ.wei, summ.exp, summ.cox)|>
  kbl(format = "latex", booktabs = T)|>
  kable_styling(latex_options = c("striped", "hold_position", "repeat_header"))|>
  pack_rows("Weibull", 1,7)|>
  pack_rows("Exponential", 8,14)|>
  pack_rows("Cox", 15,21)
```

Only two covariates are statistically significant in all three models while the main effect, i.e. `ph.karnol`, is not. This may implies two points:

Covariate	W.mean	Coef	Exp(Coef)	se(Coef)	Wald.p	significance
Weibull						
ph.karno	0.491	0.281	1.324	0.262	0.283	
age	62.083	0.005	1.005	0.011	0.673	
sex	0.421	-0.530	0.588	0.201	0.008	sex
ph.ecog	0.864	0.560	1.751	0.202	0.005	ph.ecog
pat.karno	81.357	-0.011	0.989	0.008	0.179	
meal.cal	949.865	-0.000	1.000	0.000	0.990	
wt.loss	10.019	-0.012	0.988	0.008	0.107	
Exponential						
ph.karno	0.491	0.211	1.235	0.260	0.416	
age	62.083	0.006	1.006	0.011	0.601	
sex	0.421	-0.498	0.607	0.202	0.014	sex
ph.ecog	0.864	0.473	1.605	0.198	0.017	ph.ecog
pat.karno	81.357	-0.008	0.992	0.008	0.304	
meal.cal	949.865	-0.000	1.000	0.000	0.885	
wt.loss	10.019	-0.010	0.990	0.007	0.196	
Cox						
ph.karno	0.491	0.249	1.282	0.264	0.346	
age	62.083	0.006	1.006	0.011	0.578	
sex	0.421	-0.533	0.587	0.202	0.008	sex
ph.ecog	0.864	0.546	1.726	0.201	0.007	ph.ecog
pat.karno	81.357	-0.011	0.989	0.008	0.181	
meal.cal	949.865	0.000	1.000	0.000	0.998	
wt.loss	10.019	-0.013	0.987	0.008	0.085	

1. All three models may be equivalent.
2. Not all covariates are helpful.

However, the diagnostic plots above show the Weibull is preferred for this data.

Thus, we undertake model selection to drop covariates that do not impact the models. Because `ph.karnol` is the main effect, we will keep it while proceeding the step forward/backward.

```
wei.mod2<- MASS::stepAIC(wei.mod, scope = list(lower = Surv(time, status) ~  ph.karno, upper =
↪  wei.mod))

exp.mod2<- MASS::stepAIC(exp.mod, scope = list(lower = Surv(time, status) ~  ph.karno, upper =
↪  exp.mod))
cox.mod2<- MASS::stepAIC(cox.mod, scope = list(lower = Surv(time, status) ~  ph.karno, upper =
↪  cox.mod))
```

```
# extract output from new models

# Weibull
cap.out = capture.output(wei.mod2)[5:13]
cap.out<- cap.out[!str_detect(cap.out, "reference")]
for(i in 2:9) cap.out[i]<- str_replace(cap.out[i],"^ +\\d",cap.out[i-1])
cap.out<- str_replace(cap.out, "Wald p", "Wald.p")
info<- str_split(cap.out, " +")|> purrr::discard(~length(.) <3 )
info<- lapply(info, \(i) i[i!=""] )
summ.wei2<-
exec(rbind, !!!info[-1])|>
  `colnames<-`(info[[1]])|>
  as_tibble()|>
  mutate(significance = ifelse(as.numeric(Wald.p) < 0.05, Covariate,""))

# Exp

cap.out = capture.output(exp.mod2)[5:13]
cap.out<- cap.out[!str_detect(cap.out, "reference")]
for(i in 2:7) cap.out[i]<- str_replace(cap.out[i],"^ +\\d",cap.out[i-1])
cap.out<- str_replace(cap.out, "Wald p", "Wald.p")
info<- str_split(cap.out, " +")|> purrr::discard(~length(.) <3 )
info<- lapply(info, \(i) i[i!=""] )
summ.exp2<-
exec(rbind, !!!info[-1])|>
  `colnames<-`(info[[1]])|>
  as_tibble()|>
  mutate(significance = ifelse(as.numeric(Wald.p) < 0.05, Covariate,""))

# cox

cap.out = capture.output(cox.mod2)[5:13]
cap.out<- cap.out[!str_detect(cap.out, "reference")]
for(i in 2:10) cap.out[i]<- str_replace(cap.out[i],"^ +\\d",cap.out[i-1])
cap.out<- str_replace(cap.out, "Wald p", "Wald.p")
info<- str_split(cap.out, " +")|> purrr::discard(~length(.) <3 )
info<- lapply(info, \(i) i[i!=""] )
summ.cox2<-
exec(rbind, !!!info[-1])|>
  `colnames<-`(info[[1]])|>
  as_tibble()|>
  mutate(significance = ifelse(as.numeric(Wald.p) < 0.05, Covariate,""))|>
  `names<-`(names(summ.exp))

bind_rows(summ.wei2, summ.exp2, summ.cox2)|>
```

```
kbl(format = "latex", booktabs = T)|>
kable_styling(latex_options = c("striped", "hold_position"))|>
pack_rows("Weibull", 1,4)|>
pack_rows("Exponential", 5,7)|>
pack_rows("Cox", 8,11)
```

Covariate	W.mean	Coef	Exp(Coef)	se(Coef)	Wald.p	significance
Weibull						
ph.karno	0.491	0.213	1.237	0.251	0.397	
sex	0.421	-0.544	0.580	0.198	0.006	sex
ph.ecog	0.864	0.667	1.949	0.187	0.000	ph.ecog
wt.loss	10.019	-0.010	0.990	0.007	0.160	
Exponential						
ph.karno	0.491	0.177	1.193	0.252	0.483	
sex	0.421	-0.476	0.622	0.197	0.016	sex
ph.ecog	0.864	0.522	1.685	0.177	0.003	ph.ecog
Cox						
ph.karno	0.491	0.178	1.194	0.253	0.483	
sex	0.421	-0.548	0.578	0.199	0.006	sex
ph.ecog	0.864	0.655	1.926	0.187	0.000	ph.ecog
wt.loss	10.019	-0.011	0.989	0.008	0.126	

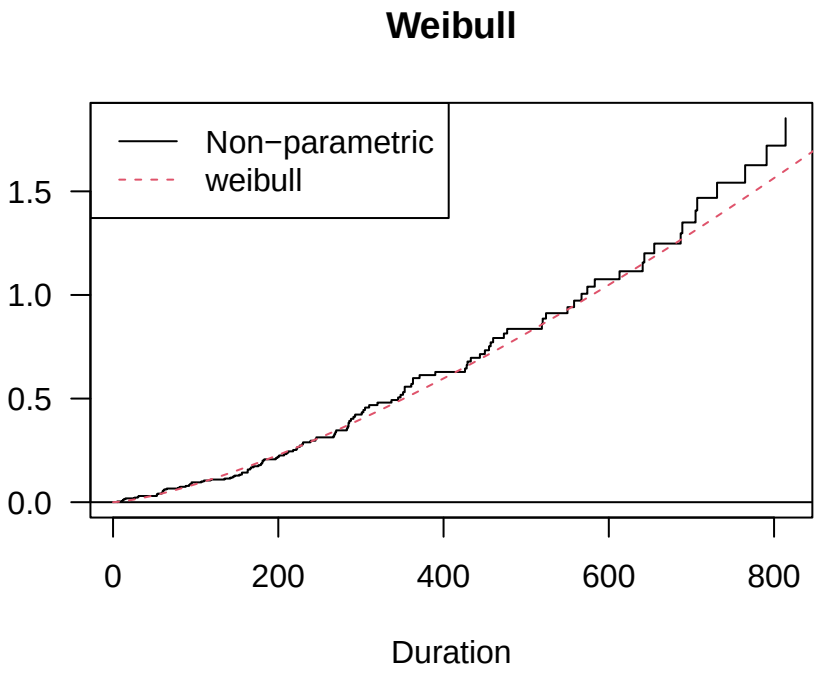
The updated Weibull and Cox models now have 4 covariates, ph.karno1, sex2, ph.ecog, wt.loss while the exponential model has 3 covariates, ph.karno1, sex2, ph.ecog. We will make a comparison now.

The main effects in three model are similar. The effect in Weibull model is the highest, followed by that in Cox and then Exponential (the effects of Cox and exponential are very much the same). All three are higher than 1, implying the effect of `ph.karno`. However, the effect is not statistically significant.

Question 3

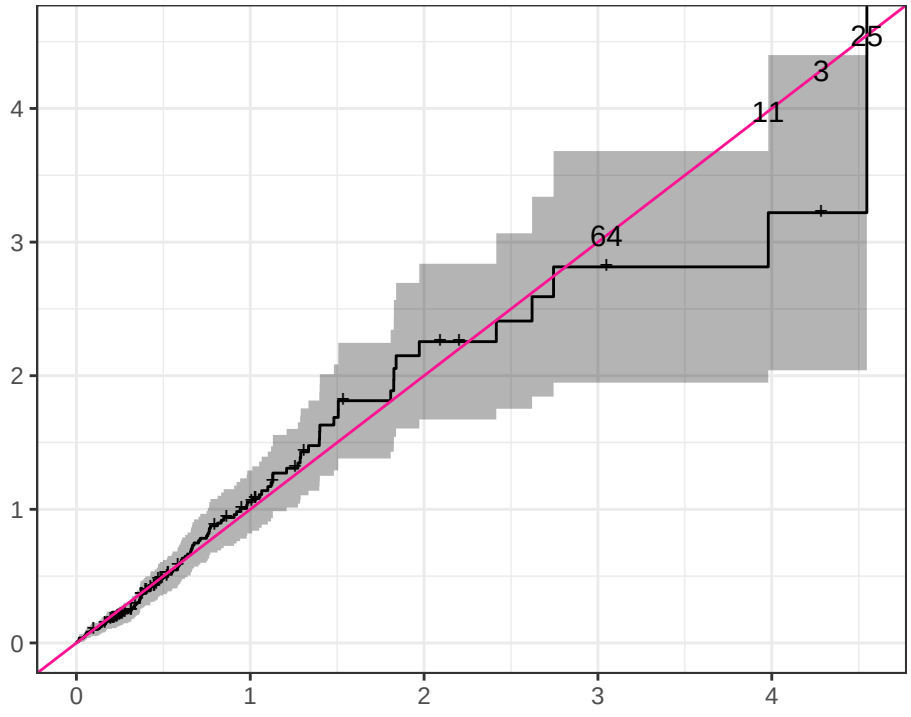
We check the distribution of Weibull and Cox model:

```
check.dist(wei.mod2, cox.mod2)
```



We see that Weibull model fits well to the Cox model. Thus, we only evaluate the goodness of fit of Weibull. We calculate the Cox-Snell residuals and plot it against the cumulative hazard function.

```
wm<- flexsurvreg(Surv(time, status) ~  ph.karno + sex+ ph.ecog  + wt.loss , data = dat, dist =
↪  "weibull")
cs<- coxsnell_flexsurvreg(wm)

surv <- survfit(Surv(cs$est, cs$status) ~ 1)

index = which(cs$est>=3)
val = cs$est[cs$est>=3]

autoplot(surv, fun = "cumhaz")+ geom_abline(intercept = 0, slope = 1, color = "deeppink")+
↪  theme_bw()+ labs(x = " ", y =" ")+ annotate("text", x=val, y=val, label= index)
```



The model fit the data quite well. However, there are some outliers whose indices are shown in the plot.

References

1. Schouten HJ. Sample size formula with a continuous outcome for unequal group sizes and unequal variances. Statistics in Medicine. 1999;18(1):87–91.

2. Emrich LJ, Piedmonte MR. A method for generating high-dimensional multivariate binary variates. The American Statistician. 1991;45(4):302–4.

3. Gasparini A. Rsimsum: Summarise results from monte carlo simulation studies. Journal of Open Source Software. 2018;3(26):739.

4. Morris TP, White IR, Crowther MJ. Using simulation studies to evaluate statistical methods. Statistics in medicine. 2019;38(11):2074–102.