PH C240B: Assignment 2

Rachael Phillips

10/20/2017

Problem 1

Consider the following data generating system for (T, C), a random person's survival time and censoring time in days. T follows an exponential distribution with $\lambda = 0.01$ and C follows a weibull distribution with shape and scale parameters, 4 and 100. Simulate 1000 random draws of $n = 1000$ from this distribution and for each draw compute the Kaplan-Meier estimator for survival curve at times 50, 60 and 70, where we consider the study ending at time 100. Form 95% simultaneous confidence bands from each draw and check simultaneous coverage of the true survival. Use the influence curve for the KM estimator to obtain the CI's for each draw. Report the coverage percentage. Note, use foreach for this as shown in lab3sol.Rnw on bcourses to save some time.

```r
S0 = function(t) 1-pexp(t,.01)

sim.cov = function(n) {
  T = rexp(n, .01)
  C = rweibull(n, 4, scale = 100)

  # record the observed right censored data
  Ttilde = pmin(T,C)
  Delta = T <= C & T < 100

  # do a KM fit
  km = survfit(Surv(Ttilde, Delta, type = "right") ~ 1, type = "kaplan-meier",
             conf.int = .95)

  # make step functions
  km_step <- stepfun(km$time, c(1, km$surv))
  km_lower <- stepfun(km$time, c(1, km$lower))
  km_upper <- stepfun(km$time, c(1, km$upper))

  # order the times and corresponding deltas
  T_ord = Ttilde[order(Ttilde)]
  D_ord = Delta[order(Ttilde)]

  # Get all known death times in order and compute the hazard at such times
  # lambda = km$n.event[D_ord]/km$n.risk[D_ord]
  T_death = T_ord[D_ord]
  nrisk = vapply(T_death, FUN = function(x) sum(Ttilde >= x), FUN.VALUE = 1)
  lambda = 1/nrisk

  # compute the prob of surviving up to or past each death time
  Pbar = vapply(T_death, FUN = function(time) mean(Ttilde >= time), FUN.VALUE = 1)

  IC_time = function(Ttilde, Delta, time, n) {
    sum((-(T_death <= time)*km_step(time)/(1 - lambda))*
        lambda^2*(1 - lambda)^2/((1 - lambda)^2/n + lambda^2*Pbar)*
```

```r
          ((Ttilde == T_death & Delta == 1)/lambda - (Ttilde > T_death)/(1 - lambda)))
  }

  # compute the IC matrix with each col an IC
  IC = vapply(c(50,60,70),FUN = function(t) {
    unlist(lapply(1:n, FUN = function(i) {
      IC_time(Ttilde[i], Delta[i], t, n)
    }))
  }, FUN.VALUE = rep(1,n))


  # compute the correlation and draw the random three-d normals
  Sigma = cor(IC)
  z = rmvnorm(1e6, c(0,0,0), Sigma)

  # compute the max abs val of each of the 3-d normals then choose the
  # 95th quantile of that vector, which is the simultaneous number of SE's
  z_abs = apply(z, 1, FUN = function(row) max(abs(row)))
  SE_num = quantile(z_abs, .95)

  # Note how the CI is wider when demanding simultaneous coverage
  Cov50 = km_lower(50) <= S0(50) & km_upper(50) >= S0(50)
  Cov60 = km_lower(60) <= S0(60) & km_upper(60) >= S0(60)
  Cov70 = km_lower(70) <= S0(70) & km_upper(70) >= S0(70)

  indy_cov = all(Cov50, Cov60, Cov70)

  # simultaneous for 50, 60, 70
  Sim50 = km_step(50) - SE_num*sd(IC[,1])*sqrt(n-1)/n <= S0(50) &
    S0(50) <= km_step(50) + SE_num*sd(IC[,1])*sqrt(n-1)/n
  Sim60 = km_step(60) - SE_num*sd(IC[,2])*sqrt(n-1)/n <= S0(60) &
    S0(60) <= km_step(60) + SE_num*sd(IC[,2])*sqrt(n-1)/n
  Sim70 = km_step(70) - SE_num*sd(IC[,3])*sqrt(n-1)/n <= S0(70) &
    S0(70) <= km_step(70) + SE_num*sd(IC[,3])*sqrt(n-1)/n

  sim_cov = all(Sim50, Sim60, Sim70)

  return(c(indy_cov = indy_cov, sim_cov = sim_cov))
}

sim.cov(1000)
```

```
## indy_cov  sim_cov
##     TRUE     TRUE
```

```r
registerDoParallel(cores = detectCores())
getDoParWorkers()
```

```
## [1] 4
```

```r
B = 1000
n = 1000
ALL = foreach(i=1:B,.packages=c("mvtnorm"), .errorhandling = "remove") %dopar% {sim.cov(n)}
res = do.call(rbind, ALL)
colMeans(res)
```

```
## indy_cov  sim_cov
##    0.905    0.949
```

We calculated 'indy_cov' based on the combination of individually based confidence intervals (i.e. a confidence for *one* time point) and we see that, even though each confidence interval probably covers it's correspond truth 95% of the time, these individual confidence intervals only cover the truth simultaneously (i.e. across our set of time points) roughly 91% of the time. Alternatively, we calculated 'sim_cov' based on a 95% simultaneous confidence interval. A simultaneous confidence interval is constructed according to a *set* of points and simultaneous confidence intervals stem from a multivariate normal distribution with mean equal to 0 and variance equal to the correlation matrix of the covariance matrix for the set. They are also wider than individually based confidence intervals because they need to cover a set of points opposed to a singular point. For a simutaneous confidence interval, we expect all intervals to contain their corresponding truth with 95% confidence and, indeed, we see that this is the case as the 'sim_cov' is almost exactly 95%!

Problem 2

Let (W, A, Y) be the observed data with distribution, $P_0 \in M$ nonparametric. Define the following parameter mapping for $P \in M : \Psi(P) = E_P\left[(1,1,W)\beta\right]$ where $\beta = \Psi^1(P) = \text{argmin}\gamma E_P\left[(Y-(1,A,W)\gamma)^2\right]$.

(a) The empirical distribution, $\mathbf{P}_n$, is the NPMLE for the true distribution. We use $\mathbf{P}_n$ as a plug-in estimator, $\Psi(\mathbf{P}_n)$, for the true parameter, $\Psi(P_0)$. This is called the NPMLE for $\Psi(P_0)$. Derive this estimator's influence curve. You may derive the efficient influence curve, $D^*_\Psi(P)$ first and then your answer is $D^*_\Psi(\mathbf{P}_n)$ This is a valid approach but not the only approach.

Because $\Psi(P)$ is a function of $\Psi^1(P)$ we first derive the influence curve for $\Psi^1(P) = \beta = \text{argmin}\gamma E_P\left[(Y - (1,A,W)\gamma)^2\right]$. First we need to compute the pathwise derivative (also let's assume that W is d-dimensional):

$$D_\Psi(P) = \frac{d\Psi}{d\beta} D_\beta(P)(O) \ = E_P(1,1,W)D_\beta(P)(O)$$

First, we compute $D_\beta(P)(O)$:

Because $\Psi^1(P) = \beta = \text{argmin}\gamma E_P\left[(Y-(1,A,W)\gamma)^2\right]$, β satisfies

$$\frac{d}{d\beta}\text{argmin}\gamma E_P\left[(Y-(1,A,W)\beta)^2\right] = 0_{d\times 1}\ .$$

Thus,

$$2\Big[E_P\left[(Y-(1,A,W)\beta)(1,A,W)^\top\right]\Big] = 0_{d\times 1}$$

and

$$E_P\left[(Y-(1,A,W)\beta)(1,A,W)^\top\right] = 0_{d\times 1} \qquad (1)\ .$$

Also, because $\Psi^1(P_\epsilon) = \beta_\epsilon = \text{argmin}\gamma E_{P_\epsilon}\left[(Y-(1,A,W)\gamma)^2\right]$, β_ϵ satisfies

$$\frac{d}{d\beta_\epsilon}\text{argmin}\gamma E_{P_\epsilon}\left[(Y-(1,A,W)\beta_\epsilon)^2 = 0_{d\times 1}\right]$$

we have that

$$2\Big[E_{P_\epsilon}\left[(Y-(1,A,W)\beta_\epsilon)(1,A,W)^\top\right]\Big] = 0_{d\times 1}$$

which implies

$$E_{P_\epsilon}\left[(Y-(1,A,W)\beta_\epsilon)(1,A,W)^\top\right] = 0_{d\times 1} \qquad (2)\ .$$

From (1) and (2),

$$E_P\left[(Y-(1,A,W)\beta)(1,A,W)^\top\right] - E_{P_\epsilon}\left[(Y-(1,A,W)\beta_\epsilon)(1,A,W)^\top\right] = 0_{d\times 1}$$

which implies

$$E_P\left[(Y-(1,A,W)\beta)(1,A,W)^\top\right] - E_{P_\epsilon}\left[(Y-(1,A,W)\beta)(1,A,W)^\top\right]$$

$$+\ E_{P_\epsilon}\left[(Y-(1,A,W)\beta)(1,A,W)^\top\right] - E_{P_\epsilon}\left[(Y-(1,A,W)\beta_\epsilon)(1,A,W)^\top\right] = 0_{d\times 1}\ .$$

Therefore,

$$E_{P_\epsilon}\left[(Y-(1,A,W)\beta_\epsilon)(1,A,W)^T\right] - E_{P_\epsilon}\left[(Y-(1,A,W)\beta)(1,A,W)^T\right] \qquad (3)$$

$$= E_P\left[(Y-(1,A,W)\beta)(1,A,W)^\top\right] - E_{P_\epsilon}\left[(Y-(1,A,W)\beta)(1,A,W)^\top\right]\ . \qquad (4)$$

For

$$lim_{\epsilon\to 0}\frac{(4)}{\epsilon} = E_{P_\epsilon}(Y-(1,A,W)\frac{\beta_\epsilon-\beta}{\epsilon})(1,A,W)^\top$$

$$= lim_{\epsilon\to 0}\ E_{P_\epsilon}(1,A,W)^\top(1,A,W)\frac{\beta_\epsilon-\beta}{\epsilon}) = E_P(1,A,W)^\top(1,A,W)\frac{d\Psi^1(P_\epsilon)}{d\epsilon}|_{\epsilon=0} \qquad (5)\ .$$

Also,

$$lim_{\epsilon\to 0}\ \frac{(3)}{\epsilon}\ = lim_{\epsilon\to 0}\int(Y-(1,A,W))\beta(1,A,W)^\top\frac{P_\epsilon(O)-P(O)}{\epsilon}d\nu(O)$$

and because

$$P_\epsilon(O) = (1+\epsilon h)P(O))\ = \int(Y-(1,A,W))\beta(1,A,W)^\top S(O)P(O)d\nu(O)\ = E_P[(Y-(1,A,W))\beta(1,A,W)^\top S(O)] \qquad (6)\ .$$

Therefore, from (5) = (6) it follows that,

$$E_P(1,A,W)^\top(1,A,W)\frac{d\Psi^1(P_\epsilon)}{d\epsilon}|_{\epsilon=0}\ = E_P[(Y-(1,A,W))\beta(1,A,W)^\top S(O)]$$

which implies

$$\frac{d\Psi^{'}(P\epsilon)}{d\epsilon}|_{\epsilon=0}\ =\ \left[E_P(1,A,W)^\top(1,A,W)\right]^{-1}E_P\left[(Y-(1,A,W))\beta(1,A,W)^\top S(O)\right]$$

$$= <\left[E_P(1,A,W)^\top(1,A,W)\right]^{-1}E_P\left[(Y-(1,A,W))\beta(1,A,W)^\top\right], S(O) >.$$

Thus,

$$D_\beta(P)(O) = [E_P(1,A,W)^\top(1,A,W)]^{-1}E_P[(Y-(1,A,W))\beta(1,A,W)^\top]\ .$$

(P.S. Thank you Jonathan for your help in solving the influence curve for β!)

Now we compute $D_\Psi(P)$:

$$D_\Psi(P) = E_P(1,1,W)[E_P(1,A,W)^\top(1,A,W)]^{-1}E_P[(Y-(1,A,W))\beta(1,A,W)^\top]$$

and it follows that

$$D^*_\Psi(P) = E_P(1,1,W)[E_P(1,A,W)^\top(1,A,W)]^{-1}E_P[(Y-(1,A,W))\beta(1,A,W)^\top] - \Psi(P)\ .$$

Finally,

$$D^*_\Psi(\mathbf{P}_n) = E_P(1,1,W)[E_P(1,A,W)^\top(1,A,W)]^{-1}E_P[(Y-(1,A,W))\beta(1,A,W)^\top] - \Psi(\mathbf{P}_n)\ .$$

(b) Consider the following data generating process: W_1, W_2 are standard normals, $Pr(A=1) = expit(1 + 0.4W_1 - 0.4W_2W_1)$, $Y = (W_1+W_2)^2 + \epsilon$, $\epsilon \sim N[0,1]$. What is the true parameter value, $\Psi(P_0)$, for this data generating process?

To determine the true parameter value, $\Psi(P_0)$, we ran a simulation:

```r
expit <- function(x){
  1/(1+exp(-x))
}

# generate the data
n <- 10000
W1 <- rnorm(n, 0, 1)
W2 <- rnorm(n, 0, 1)
A <- rbinom(n, 1, expit(1 + .4 * W1 - .4 * W2 * W1))
epsilon <- rnorm(n, 0 ,1)
Y <- (W1 + W2)^2 + epsilon

df <- data.frame(W1, W2, A, Y)

beta_hat <- lm(Y ~ ., data = df)

mean(beta_hat$fitted.values)
```

```
## [1] 2.019982
```

```r
# pseudo-counterfactual dataframe forcing the treatment to 1
cf_df <- df
cf_df$A <- 1
cf_df <- cf_df[c("W1", "W2", "A")]

pred <- predict(beta_hat, cf_df)

mean(pred)
```

```
## [1] 1.770836
```

(c) Now consider the parameter, $\Psi^2(P) = E_P E_P[Y|A=1,W]$ or the true treatment specific mean for our non-parametric model. What is the true parameter value for the data generating process in part (b)?

The treatment specific mean is $E_{U,X}[Y(A=1)]$. Since A is fixed, Y is not directly affected by A so we can simply solve the mean of our Y values to find the treatment specific mean. Note that Y is indirectly affected by A through $W1$ and $W2$. We have that

$$E_{U,X}[Y(A=1)] = EE[Y|A=1,W]$$

When we condition on both A and W, we are making the assumption that $Y \perp A | W$.

Unfortunately, these results do not indicate a unique value for the influence curve for each individual observation. Based on page 537 of the Targeted Learning book, we would expect the following IC:

$$D^*_\Psi(\mathbf{P}n) = E_P(1,1,W)[E_P(1,A,W)^\top(1,A,W)]^{-1}[(Y-(1,A,W))\beta(1,A,W)^\top] - \Psi(P_n)$$

which is almost the same as our result.

```
n <- 10000
W1 <- rnorm(n, 0, 1)
W2 <- rnorm(n, 0, 1)
A <- 1
epsilon <- rnorm(n, 0 ,1)
Y <- (W1 + W2)^2 + epsilon

df <- data.frame(W1, W2, A, Y)

mean(df$Y)
```

```
## [1] 1.992731
```

(d) Run a simulation where you take 1000 draws of $n = 1000$ from the data generating process in part (b). Using your NPMLE from part (a) and its influence curve to form 95% Wald confidence intervals, check coverage of the true Ψ and Ψ^2 for each draw. Report the coverage percentage for each parameter. Considering that estimating Ψ is often used to estimate Ψ^2, comment on your results.

```
sim <- function(n) {
 W1 <- rnorm(n, 0, 1)
 W2 <- rnorm(n, 0, 1)
 A <- rbinom(n, 1, expit(1 + .4 * W1 - .4 * W2 * W1))
 epsilon <- rnorm(n, 0 ,1)
 Y <- (W1 + W2)^2 + epsilon
 df <- data.frame(W1, W2, A, Y)
 beta_hat <- lm(Y ~ ., data = df)
 IC <- mean(beta_hat$fitted.values)*(Y-beta_hat$fitted.values)
 sigma <- sqrt(var(IC)/(n))
 Psi.n <- mean(beta_hat$fitted.values)
 upperbound <- Psi.n + 1.96*sigma
 lowerbound <- Psi.n - 1.96*sigma

 # true Psi_1 from part (b)
 Psi_1 <- 1.75
 # true Psi_2 from part (c)
 Psi_2 <- 2

 indicator_Psi1 <- as.numeric(upperbound > Psi_1 && lowerbound < Psi_1)

 indicator_Psi2 <- as.numeric(upperbound > Psi_2 && lowerbound < Psi_2)

 return(c(indicator_Psi1, indicator_Psi2))
}
sim(1000)
```

```
## [1] 1 1
```

```r
registerDoParallel(cores = detectCores())
getDoParWorkers()
```

```
## [1] 4
```

```r
B <- 1000
n <- 1000
ALL <- foreach(i=1:B,.packages=c("mvtnorm"), .errorhandling = "remove") %dopar% {sim(n)}
res <- do.call(rbind, ALL)

colMeans(res)
```

```
## [1] 0.959 0.999
```

Considering that estimating Ψ is often used to estimate Ψ^2, we do expect these coverages to be similar.

Problem 3

Bonus question: What is the remainder term for your estimator in part (a)? What remainder term conditions need to be satisfied for the estimator to be asymptotically linear?

We assume the NPMLE is well-behaved in the sense that it estimates the parameters in the second-order remainder at a rate faster than $n^{-1/4}$ (i.e. consistent at rate $< n^{-1/4}$). Then, $R_2(\mathbf{P}_n, P_0) = o_P(n^{-1/2})$ and $\Psi(\mathbf{P}_n) - \Psi(P_0) = (P_n - P_0)D^*(\mathbf{P}_n) + o_P(n^{-1/2})$. We had to make this assumption to derive the efficient influece curve for the NPMLE and the remainder term in part (a) is

$$R_2(\mathbf{P}_n, P_0) = \Psi(\mathbf{P}_n) - \Psi(P_0) - (P_n - P_0)D^*_\Psi(\mathbf{P}_n)$$

where $\Psi(\mathbf{P}_n) = \frac{1}{n}\sum_{i=1}^n (1, 1, W_i)\hat{\beta}$, $\Psi(P_0) = E_P[(1, 1, W_i)\beta]$, P_n is the empirical measure, P_0 is the true data distribution, and $D^*_\Psi(\mathbf{P}_n)$ is defined in part (a). Note that $(P_n - P_0)D^*_\Psi(\mathbf{P}_n)$ is the empirical processes applied to random functions, not a sum of i.i.d random variables because the random variable is defined by the data.

Collaborators & Resources

Tommy Carpenito, Yue You, Kevin Benac

Jonathan's lab3sol & OH