PSC 290 MC Simulations Homework 2

Mijke Rhemtulla

2024-10-24

Instructions

Answer all the questions below. This homework is due on Nov 8 (Friday) by the end of the day. Upload both this Rmd file (completed) and the PDF output. If you are having troubles knitting to pdf, try knitting to html instead and then saving as a pdf (submitting a pdf makes my life easier because I can grade the assignment in canvas without having to download and open it in other software). Be sure your troubles aren't due to missing objects in the Rmd file (e.g., all code should run without producing errors).

You may work on this assignment with others, and you may ask for help (e.g., on Slack) while working. The work that you submit, though, must be your own.

Part 1: Evaluating Bias, Efficiency and Consistency

Question 1

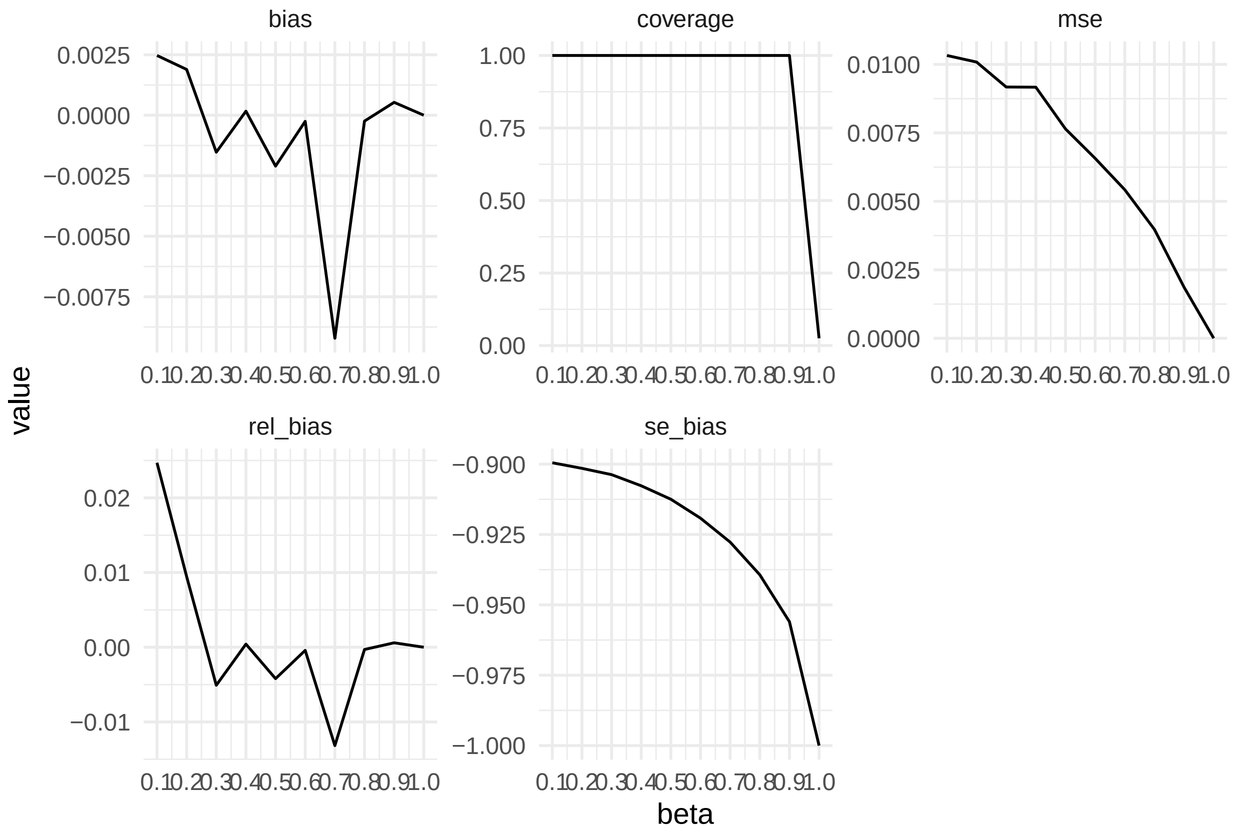
Write a function to generate data from a 1-predictor linear regression model and then run the model. Your function should take the arguments n and *beta*. Within your function, set the variance of the predictor to be 1, and the residual variance to be $var(e) = 1 - \beta^2$ so that the total variance of y is 1. Let the intercept be 0. After generating the data, your function should fit the regression model using the `lm()` function, and return the estimated regression coefficient and its standard error.

```r
q1fun <- function(n, beta) {

  p = 1
  Sigma = matrix(0, p, p)
  diag(Sigma) = 1
  var_e = abs(1 - (beta^2))

  X = MASS::mvrnorm(n = n, rep(0, p) , Sigma)
  y = as.numeric(cbind(1, X) %*% c(0, beta) + rnorm(n, 0, sqrt(var_e)))
  Xy = as.data.frame( cbind(X, y))
  fit <- lm(y ~., data = Xy)
  s <- summary(fit)
  s$coefficients[2,1:2]
  #cis <- confint(fit)[-1,]

}

x <- q1fun(n=100, beta=1.7)
```

Question 2

Use the function you wrote to run a simulation investigating the performance of regression coefficients as a function of β. Choose a fixed level of n and vary β. For each condition compute (a) bias, (b) relative bias, (c) mean-squared error, (d) standard error bias, and (e) confidence interval coverage. Plot your results.

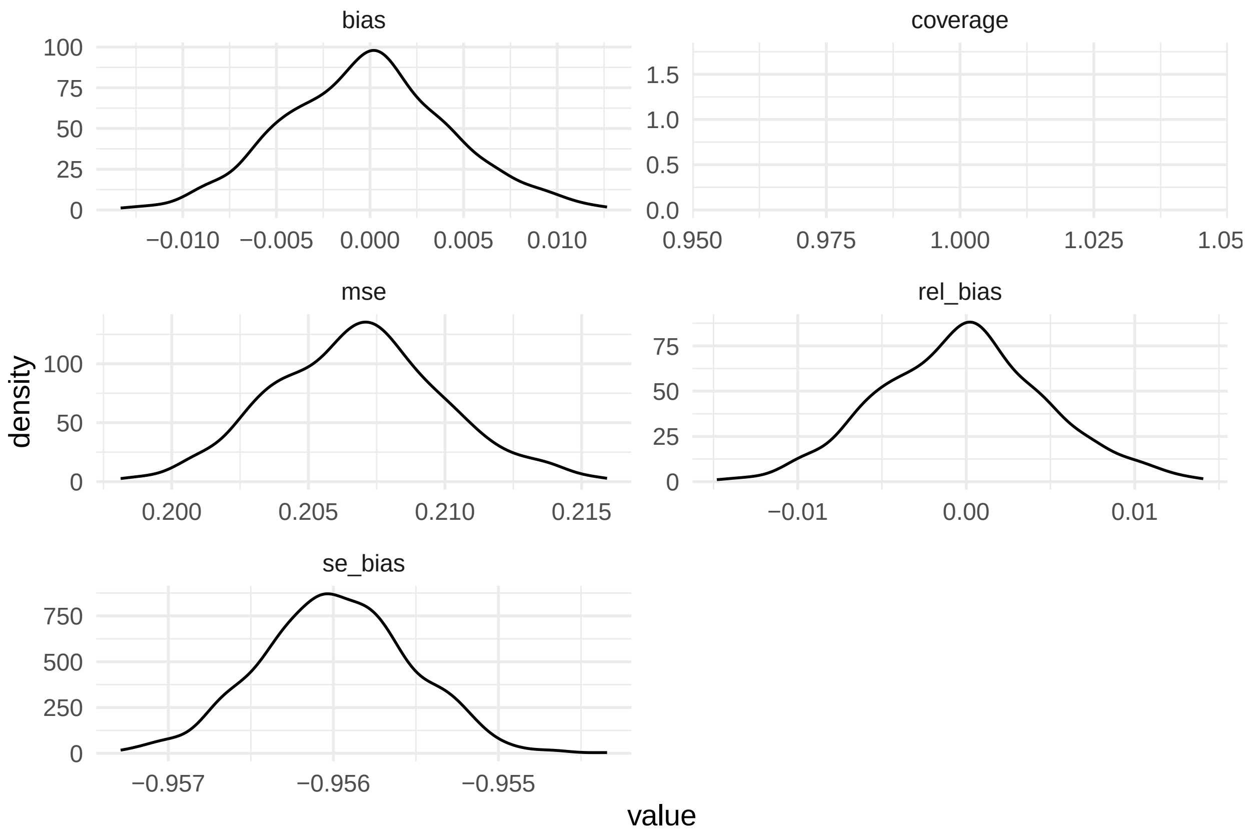
```r
library(ggplot2)

reps <- 1000
n <- 100
betas <- matrix(seq(0.1, 1, by = 0.1), nrow = reps, ncol = 10, byrow = TRUE)

# initiate matrices to save the estimated coefficients, standard errors and coverage
coefs <- serrs <- cover <- matrix(NA, nrow = reps, ncol = ncol(betas))

# for each beta, we will obtain the measures mentioned above

for (b in betas[1,]) {

  for (i in 1:reps) {

    col <- which(betas[1,] == b)
    coefs[i, col] <- q1fun(n, b)[1]
    serrs[i, col] <- q1fun(n, b)[2]
    CIhigh = coefs[i, col] + 1.96 * serrs[i, col]
    CIlow = coefs[i, col] - 1.96 * serrs[i, col]
    cover[i, col] <-  ifelse(CIlow < coefs[i, col] & CIhigh > coefs[i, col], 1, 0)
  }
}


results <- data.frame(
  beta = betas[1,],
  bias = colMeans(coefs - betas),
  rel_bias = colMeans((coefs - betas)/betas),
  mse = colMeans((coefs - betas)^2),
  se_bias = colMeans(serrs - 1),
  coverage = colMeans(cover)
)

results |>
  tidyr::pivot_longer(cols = c(bias:coverage), names_to = "outcome", values_to = "value") |>
  ggplot(aes(x = beta, y = value)) +
  geom_line() +
  scale_x_continuous(breaks = betas[1,] ) +
  facet_wrap(~outcome, scales= "free") +
  theme_minimal()
```



Question 3

Choose one condition from the simulation you ran in the last part (i.e., one value of β). Repeat that condition a large number of times (e.g., 1000), computing the whole set of 5 outcomes each time. Use histograms to plot the sampling variability of these Monte Carlo outcomes across repetitions.

```r
beta <- 0.9
n <- 100

metrics <- data.frame(
  bias = rep(NA, reps),
  rel_bias = rep(NA, reps),
  mse = rep(NA, reps),
  se_bias = rep(NA, reps),
  coverage = rep(NA, reps)
)


for (j in 1:reps) {

  coefs <- serrs <- cover <- c()

  for (i in 1:100) {

    coefs[i] <- q1fun(n, beta)[1]
    serrs[i] <- q1fun(n, beta)[2]
    CIhigh = coefs[i] + 1.96 * serrs[i]
    CIlow = coefs[i] - 1.96 * serrs[i]
    cover[i] <-  ifelse(CIlow < coefs[i] & CIhigh > coefs[i], 1, 0)

  }

    metrics$bias[j] <- mean(coefs - beta)
    metrics$rel_bias[j] <- mean((coefs - beta) / beta)
    metrics$mse[j] <- mean((coefs - betas)^2)
    metrics$se_bias[j] <- mean(serrs - 1)
    metrics$coverage[j] <- mean(cover)

}


metrics |>
  tidyr::pivot_longer(cols = everything(), names_to = "metric", values_to = "value") |>
  ggplot(aes(x = value)) +
  #geom_histogram() +
  geom_density() +
  facet_wrap(~ metric, scales = "free", ncol = 2) +
  theme_minimal()
```



Question 4

Examine the histograms you generated in the previous question: is the degree of Monte Carlo error different for different outcomes? Try to explain the differences you see.

> The distribution of Bias, Relative Bias, Standard Error Bias, and MSE are remarkably similar, indicating that these metrics have consistent sampling variability and relatively large degree of Monte Carlo error . As expected given the results of Question 3, CI coverage achieved 100% across all 1000 replications. Consequently, the coverage values are all concentrated at a single point, resulting in a degenerate distribution that is not shown graphically.

Part 2: Multivariate Data

Question 5

Write a function that generates a sample of size n from a k-variate standard normal distribution with common correlation ρ between all pairs of variables (arguments: `n`, `k`, `rho`; returns: a data frame)

```r
q5fun <- function(n, k, rho) {

  Sigma = matrix(rho, k, k)
  diag(Sigma) = 1

  X = MASS::mvrnorm(n = n, rep(0, k) , Sigma)
  colnames(X) <- paste0("k", 1:k)
  as.data.frame(X)

}

head(q5fun(100, 5, .5))
```

```
##           k1          k2         k3         k4         k5
## 1 -1.8018043 -1.73122248 -0.4936519 -1.0394658  0.5211125
## 2 -0.8367144 -0.61513631  1.2378361  0.4091266  0.2180345
## 3 -0.4959995 -0.62486484  0.3044630  0.1365552 -0.5695242
## 4  0.5207551 -0.14730139  0.6252601  1.2326680  0.2680283
## 5  0.8119665  0.79035596  0.6860357 -0.3646517  1.2460260
## 6 -0.4148344 -0.04150481  1.0982927  0.7197200  0.6382659
```

Question 6

Use the function you just wrote to run a simulation examining the probability of making at least one type-I error (i.e., the family-wise type-I error rate) as a function of n and p. What do you find (describe the results in 1-2 sentences).

```r
reps <- 1000
n <- seq(10, 100, by = 20)
p <- 2:6


type1error <- function(n, p) {
  data <- q5fun(n, p, 0)
  p_values <- c()


  for (i in 1:(p-1)) {
    for (j in (i+1):p) {

      x <- data[, i]
      y <- data[, j]

      pval <- cor.test(x, y)$p.value

      p_values <- c(p_values, pval)
    }
  }
  # return 1 if there was any type 1 error
  ifelse(sum(p_values < 0.05) > 0, 1, 0 )
}

results_df <- expand.grid(n = n, p = p)
results_df$error_rate <- NA

for (i in n) {
  for (j in p) {
      results_df[results_df$n==i & results_df$p==j, "error_rate"] <- mean(replicate(reps, type1error(i,
  }
}

results_df
```

```
##     n p error_rate
## 1  10 2      0.054
## 2  30 2      0.045
## 3  50 2      0.040
## 4  70 2      0.057
## 5  90 2      0.041
## 6  10 3      0.112
## 7  30 3      0.138
## 8  50 3      0.113
## 9  70 3      0.151
## 10 90 3      0.148
## 11 10 4      0.270
## 12 30 4      0.271
## 13 50 4      0.290
## 14 70 4      0.299
## 15 90 4      0.269
## 16 10 5      0.378
## 17 30 5      0.429
## 18 50 5      0.393
## 19 70 5      0.402
## 20 90 5      0.381
## 21 10 6      0.559
## 22 30 6      0.536
## 23 50 6       0.540
## 24 70 6       0.526
## 25 90 6       0.559
```

> The probability of making at least one Type I error increases as the number of variables tested increases. This holds true regardless of sample size, but the effect is more pronounced when the number of predictors is large.

Question 7 (optional question: only if you're having too much fun to quit)

When $\rho \neq 0$, what is the probability that at least 80% of the correlations are found to be significantly different from 0, and how does it change as a function of p and n?

```r
reps <- 1000
n <- seq(10, 100, by = 20)
p <- 3:10


type1error <- function(n, p) {

  data <- q5fun(n, p, 0.3) # now let's set rho = 0.3
  p_values <- c()


  for (i in 1:(p-1)) {
    for (j in (i+1):p) {

      x <- data[, i]
      y <- data[, j]

      pval <- cor.test(x, y)$p.value

      p_values <- c(p_values, pval)
    }
  }
  # return 1 if there was any type 1 error
  ifelse(sum(p_values < 0.05) > 0.8, 1, 0 )
}

results_df <- expand.grid(n = n, p = p)
results_df$sig <- NA

for (i in n) {
  for (j in p) {
      results_df[results_df$n==i & results_df$p==j, "sig"] <- mean(replicate(reps, type1error(i, j)))
  }
}

results_df
```

```
##     n  p   sig
## 1  10  3 0.344
## 2  30  3 0.683
## 3  50  3 0.881
## 4  70  3 0.943
## 5  90  3 0.990
## 6  10  4 0.550
## 7  30  4 0.872
## 8  50  4 0.969
## 9  70  4 0.997
## 10 90  4 1.000
## 11 10  5 0.698
## 12 30  5 0.954
## 13 50  5 0.996
## 14 70  5 0.999
## 15 90  5 1.000
## 16 10  6 0.795
## 17 30  6 0.973
## 18 50  6 1.000
## 19 70  6 1.000
## 20 90  6 1.000
## 21 10  7 0.877
## 22 30  7 0.994
## 23 50  7 1.000
## 24 70  7 1.000
## 25 90  7 1.000
## 26 10  8 0.923
## 27 30  8 0.997
## 28 50  8 1.000
## 29 70  8 1.000
## 30 90  8 1.000
## 31 10  9 0.973
## 32 30  9 1.000
## 33 50  9 1.000
## 34 70  9 1.000
## 35 90  9 1.000
## 36 10 10 0.984
## 37 30 10 1.000
## 38 50 10 1.000
## 39 70 10 1.000
## 40 90 10 1.000
```

The probability of achieving 80% of significant correlations increases both as sample size and the number of predictors variables. Even with small samples, it is very likely that at least 80% of the correlations will be found significant when the number of variables is large (>5).