midtern1_2023

Yiran Jia (A59013411)

2023-02-11

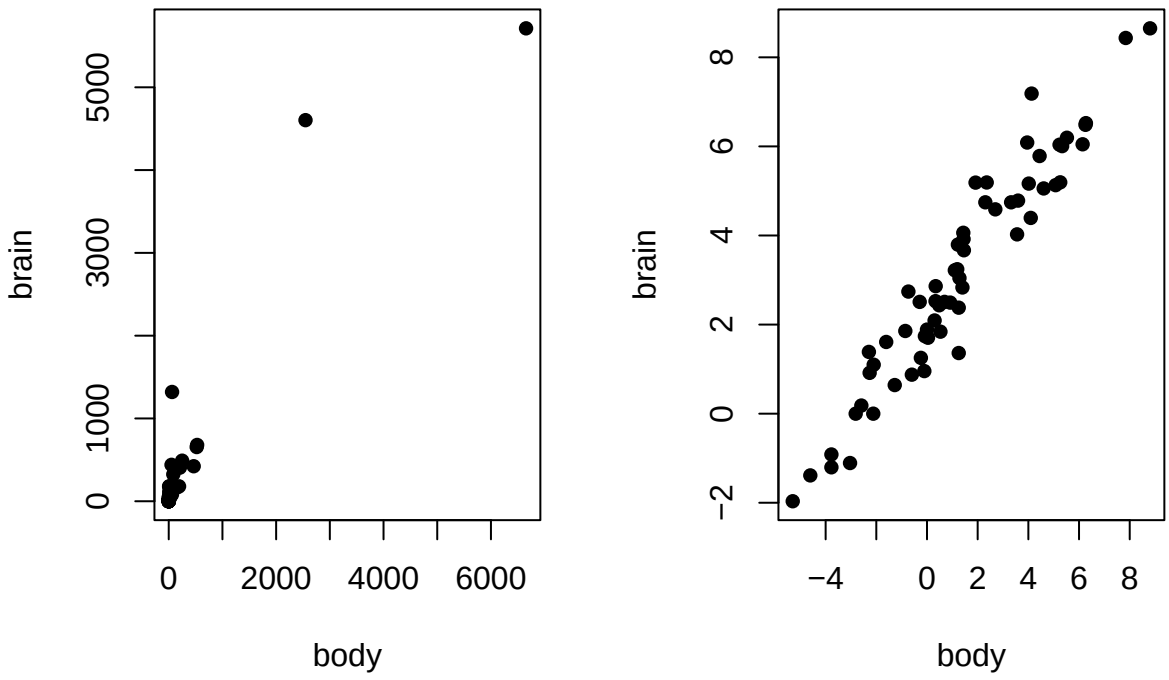
Problem 1

Part A

```
library(knitr)
opts_chunk$set(tidy.opts = list(width.cutoff = 40), tidy = TRUE)

library(MASS)
dat <- mammals
dat.a <- log(mammals)

par(mfrow = c(1, 2))
plot(dat, pch = 16)
plot(dat.a, pch = 16)
```



By comparing the two scatter plots - one without log scale versus one with log scale, we see that the relationship between two variables is better understood on a logarithmic scale. The relationship between body and brain's weight showed on the left plot is not linear, the log-log scale plot on the right helps us visualize this relationship better. Both body and brain's weight have a wide range of values. By doing log scale, the pattern of the data becomme more apparent.

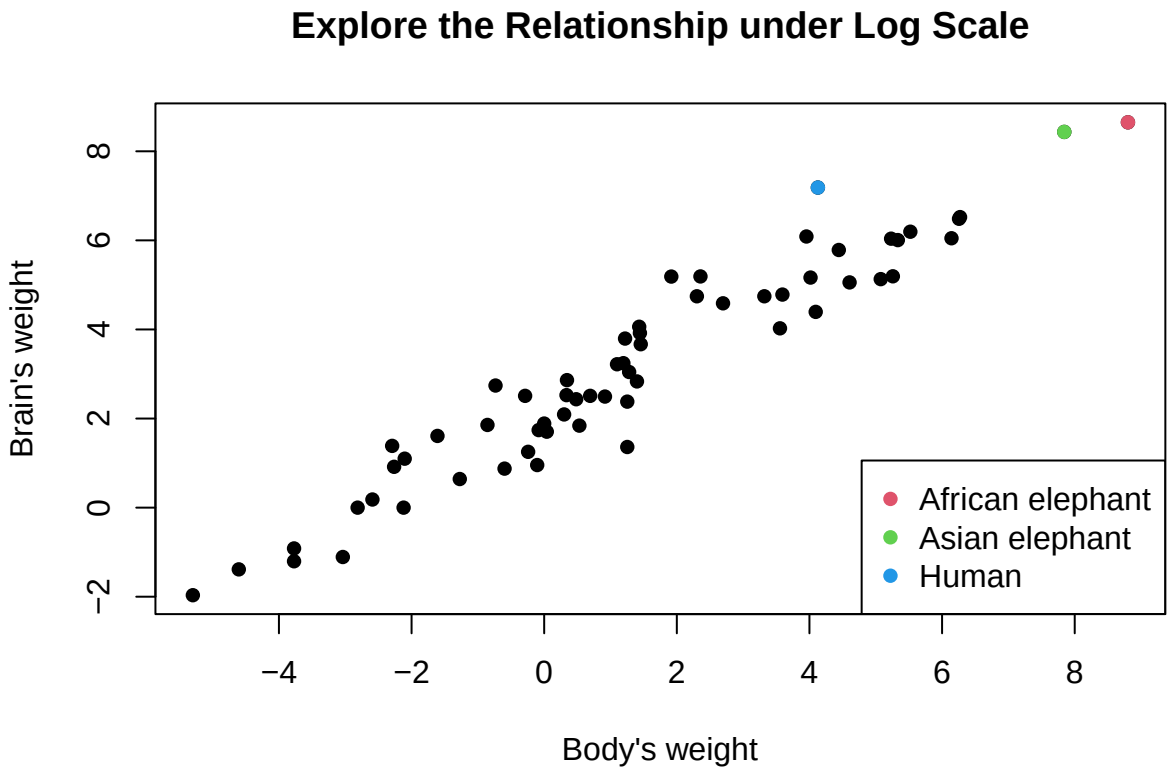
Part B

```
index <- c(which(row.names(dat.a) == "African elephant"),
    which(row.names(dat.a) == "Asian elephant"),
    which(row.names(dat.a) == "Human"))

pick_body <- c(unlist(dat.a[index, ][1, ])[1],
    unlist(dat.a[index, ][2, ])[1], unlist(dat.a[index,
        ][3, ])[1])
pick_brain <- c(unlist(dat.a[index, ][1,
    ])[2], unlist(dat.a[index, ][2, ])[2],
    unlist(dat.a[index, ][3, ])[2])


plot(dat.a, pch = 16, xlab = "Body's weight",
    ylab = "Brain's weight", main = "Explore the Relationship under Log Scale")
points(pick_body, pick_brain, pch = 16, col = 2:4)
legend("bottomright", c("African elephant",
```

```
    "Asian elephant", "Human"), pch = 16,
    col = 2:4)
```

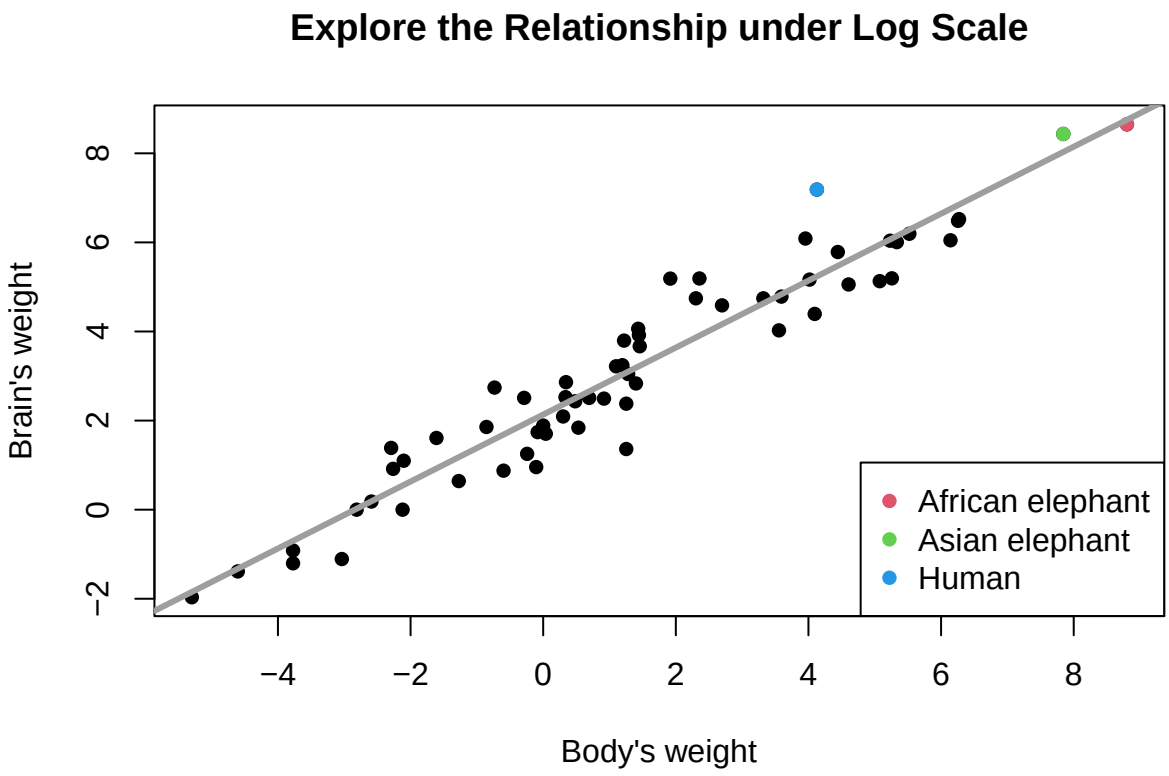


We created a scatter plot with nice label. And we color the African Elephant, Asian Elephant, and Human with colors orange, light green, and dark green. For easier read, we also add legend to ehlp identify which color indicates which species.

Part C

```
fit <- lm(brain ~ body, data = dat.a)

plot(dat.a, pch = 16, xlab = "Body's weight",
    ylab = "Brain's weight", main = "Explore the Relationship under Log Scale")
points(pick_body, pick_brain, pch = 16, col = 2:4)
abline(fit, col = 8, lwd = 3)
legend("bottomright", c("African elephant",
    "Asian elephant", "Human"), pch = 16,
    col = 2:4)
```




```
fit$coefficients[1]  # the intercept
```

```
## (Intercept)
##    2.134789
```

```
fit$coefficients[2]  # the slope
```

```
##      body
## 0.7516859
```

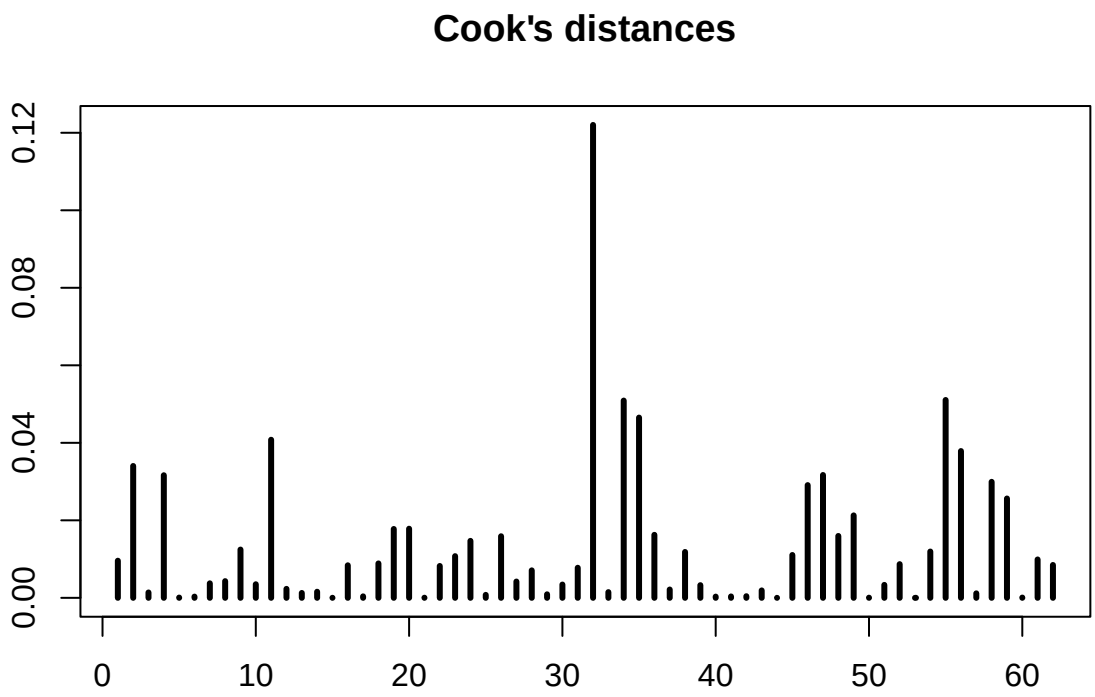
```
confint(fit, level = 0.99)
```

```
##                 0.5 %    99.5 %
## (Intercept) 1.8792861 2.3902913
## body        0.6759648 0.8274071
```

We add the regression line (gray) in the plot. The intercept and slop are listed, along with the confidence interval for each coefficient.

Part D

```
plot(cooks.distance(fit), type = "h", lwd = 3,
    xlab = "", ylab = "", main = "Cook's distances")
```




```
which.max(cooks.distance(fit))
```

```
## Human
##    32
```

From the plot we see that the observation Human has the highest cook's distance, which makes us to think whether or not Human is an influential point. From the previous scatter plot, we did see that Human is not on the main trend.

Part E

By using `?mammals`, we found that the body weight is measured in kg, and then brain weight is measured in g. From the University of Washington Brain Facts and Figures, we found that Sperm Whale has brain weight 7800 g. From AMERICAN CETACEAN S OCIETY FACT SHEET, we take the average body weight of the adult males, which is roughly about 36275 kg.

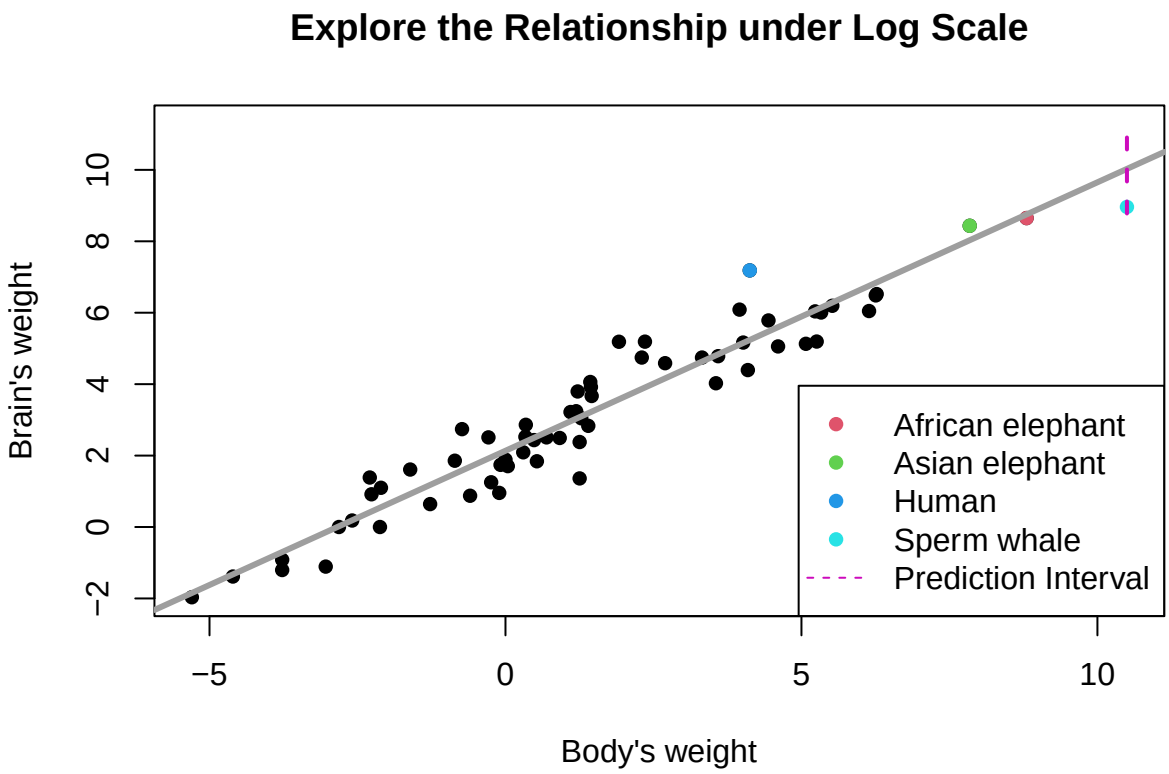
Part F

```r
new <- data.frame(body = log(36275), brain = log(7800))
pred <- predict(fit, new, interval = "prediction",
    level = 0.9)
print(pred)
```

```
##        fit      lwr      upr
## 1 10.02665 8.778891 11.27441
```

```r
low.body <- min(range(dat.a$body)[1], unlist(new)[1])
high.body <- max(range(dat.a$body)[2], unlist(new)[1])
low.brain <- min(range(dat.a$brain)[1], unlist(new)[2],
    pred[2])
high.brain <- max(range(dat.a$brain)[2],
    unlist(new)[2], pred[3])

plot(dat.a, pch = 16, xlab = "Body's weight",
    ylab = "Brain's weight", xlim = c(low.body,
        high.body), ylim = c(low.brain, high.brain),
    main = "Explore the Relationship under Log Scale")
points(pick_body, pick_brain, pch = 16, col = 2:4)
points(unlist(new)[1], unlist(new)[2], pch = 16,
    col = 13)
abline(fit, col = 8, lwd = 3)
coords <- cbind(c(log(36275), log(36275)),
    c(pred[2], pred[3]))
lines(coords, col = 6, lwd = 2, lty = 2)
legend("bottomright", c("African elephant",
    "Asian elephant", "Human", "Sperm whale",
    "Prediction Interval"), pch = c(16, 16,
    16, 16, NA), col = c(2:4, 13, 6), lty = c(NA,
    NA, NA, NA, 2))
```



Above we first generate a scatter plot that includes the new observation Sperm whale, labeled it in light blue. And then we we add the original linear regression (computed without the obsercation Sperm whale) in this plot just for a reminder. Then we use this original linear regression `fit` to predict the brain weight of Sperm whale from the body weight. And generate a 90 prediction interval. We see that the prediction on Sperm whale's weight is reasonable, but not very accurate. The predicted brain weight is 10.03 in log scale, but the true observation is 8.96 in log scale, which is lower than the predicted. We can also observe this direction from the graph since the gray regression line is above the light blue dot, whicih represent Sperm whale. However, it is not so far off.

We finally add this 90 prediction interval (correspoding to the prediction of Sperm whale) to the scatter plot.

Problem 2

Part A

```
N = 1000
sample.size <- c(20, 50, 100)
poss.a <- c(0.8, 0.9, 0.99)

op <- function(n, a, N) {

    coll.confint.length.inte <- c()
    coll.confint.length.s1 <- c()
    coll.confint.length.s2 <- c()
```

```
coll.if.contain.inte <- c()
coll.if.contain.s1 <- c()
coll.if.contain.s2 <- c()

coll.vif.s1 <- c()
coll.vif.s2 <- c()

for (i in 1:N) {
    x1 <- rnorm(n, 0, 1)
    z <- rnorm(n, 0, 1)
    x2 <- a * x1 + sqrt(1 - a^2) * z
    eps <- rnorm(n, 0, 0.5^2)
    y <- -1 + 2 * x1 - 2 * x2 + eps

    fit <- lm(y ~ x1 + x2)

    cf <- confint(fit, level = 0.95)
    confint.length.inte <- cf[1, 2] -
        cf[1, 1]
    coll.confint.length.inte[i] <- confint.length.inte
    confint.length.s1 <- cf[2, 2] - cf[2,
        1]
    coll.confint.length.s1[i] <- confint.length.s1
    confint.length.s2 <- cf[3, 2] - cf[3,
        1]
    coll.confint.length.s2[i] <- confint.length.s2

    if.contain.inte <- (-1 >= cf[1, 1] &&
        -1 <= cf[1, 2])
    coll.if.contain.inte[i] <- if.contain.inte
    if.contain.s1 <- (2 >= cf[2, 1] &&
        2 <= cf[2, 2])
    coll.if.contain.s1[i] <- if.contain.s1
    if.contain.s2 <- (-2 >= cf[3, 1] &&
        -2 <= cf[3, 2])
    coll.if.contain.s2[i] <- if.contain.s2

    library(car)
    # It is not meaningful to
    # calculate the VIF for the
    # intercept coefficient because
    # the intercept coefficient
    # represents the mean response
    # value when all predictor
    # variables are zero, which
    # does not have the issue of
    # multicollinearity involved.
    # Therefore we do not calculate
    # vif for the intercept.
    vif.s1 <- vif(fit)[1]
    coll.vif.s1[i] <- vif.s1
    vif.s2 <- vif(fit)[2]
    coll.vif.s2[i] <- vif.s2
```

```
    }

    avg.length.inte <- mean(coll.confint.length.inte)
    avg.length.s1 <- mean(coll.confint.length.s1)
    avg.length.s2 <- mean(coll.confint.length.s2)

    prop.contain.inte <- sum(coll.if.contain.inte)/N
    prop.contain.s1 <- sum(coll.if.contain.s1)/N
    prop.contain.s2 <- sum(coll.if.contain.s2)/N

    avg.vif.s1 <- mean(coll.vif.s1)
    avg.vif.s2 <- mean(coll.vif.s2)

    return(c(avg.length.inte, avg.length.s1,
        avg.length.s2, prop.contain.inte,
        prop.contain.s1, prop.contain.s2,
        avg.vif.s1, avg.vif.s2))
}

summary <- matrix(NA, length(sample.size) *
    length(poss.a), 8)

for (i in 1:length(sample.size)) {
    for (j in 1:length(poss.a)) {
        summary[j + (i - 1) * 3, ] <- op(sample.size[i],
            poss.a[j], N)
    }
}
```

```
## Loading required package: carData
```

```
rownames(summary) <- c("(20, 0.8)", "(20, 0.9)",
    "(20, 0.99)", "(50, 0.8)", "(50, 0.9)",
    "(50, 0.99)", "(100, 0.8)", "(100, 0.9)",
    "(100, 0.99)")
colnames(summary) <- c("avg.CI.length coe0",
    "avg.CI.length coe1", "avg.CI.length coe2",
    "prop.time.coe0", "prop.time.coe1", "prop.time.coe2",
    "avg.vif.coe1", "avg.vif.coe2")
print(summary)
```

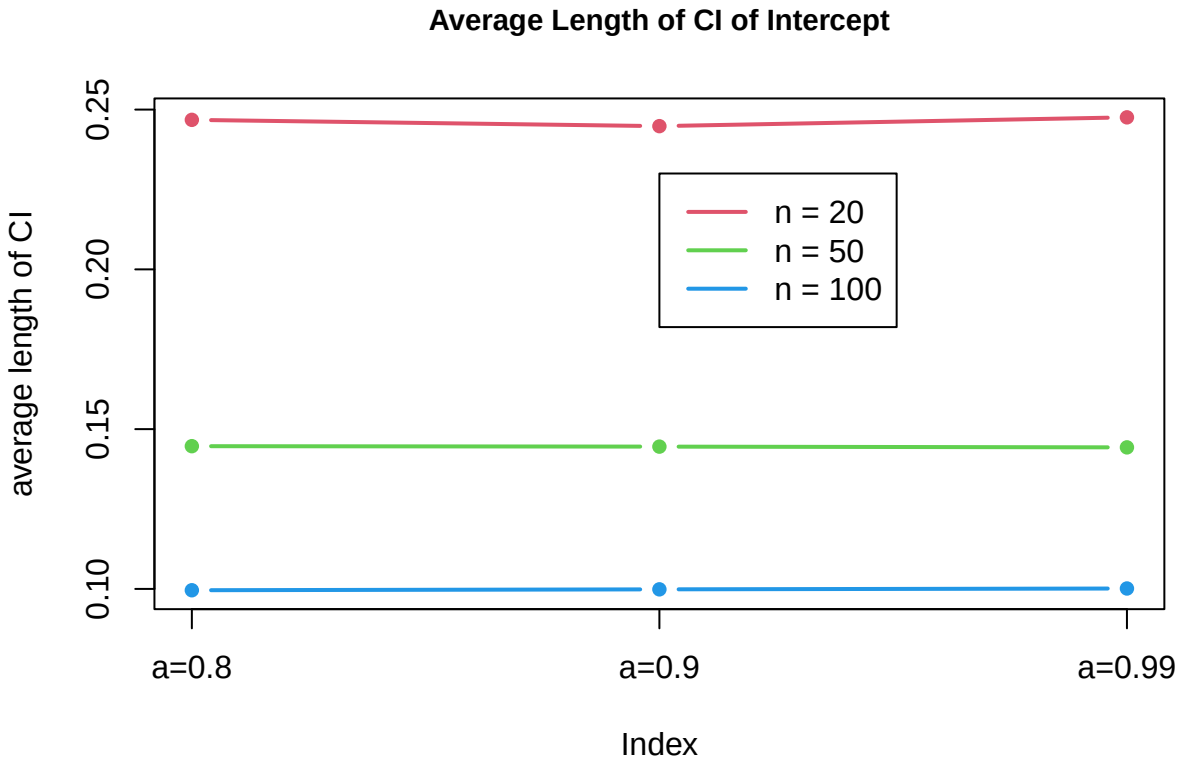
```
##             avg.CI.length coe0 avg.CI.length coe1 avg.CI.length coe2
## (20, 0.8)           0.24678895          0.4287324          0.4305156
## (20, 0.9)           0.24482017          0.5816829          0.5813952
## (20, 0.99)          0.24757079          1.8273080          1.8272320
## (50, 0.8)           0.14466571          0.2451908          0.2443561
## (50, 0.9)           0.14452734          0.3376410          0.3371314
## (50, 0.99)          0.14429094          1.0378876          1.0381830
## (100, 0.8)          0.09958358          0.1668589          0.1670491
## (100, 0.9)          0.09986297          0.2314464          0.2308931
## (100, 0.99)         0.10012962          0.7149077          0.7151765
##             prop.time.coe0 prop.time.coe1 prop.time.coe2 avg.vif.coe1
## (20, 0.8)            0.949          0.954          0.955     3.208649
```

```
## (20, 0.9)            0.952          0.961          0.957     6.089981
## (20, 0.99)           0.953          0.947          0.944    60.179846
## (50, 0.8)            0.942          0.946          0.945     2.867238
## (50, 0.9)            0.959          0.950          0.950     5.594485
## (50, 0.99)           0.952          0.941          0.940    53.256126
## (100, 0.8)           0.955          0.952          0.954     2.838044
## (100, 0.9)           0.961          0.953          0.964     5.398686
## (100, 0.99)          0.949          0.956          0.959    51.821748
##              avg.vif.coe2
## (20, 0.8)        3.208649
## (20, 0.9)        6.089981
## (20, 0.99)      60.179846
## (50, 0.8)        2.867238
## (50, 0.9)        5.594485
## (50, 0.99)      53.256126
## (100, 0.8)       2.838044
## (100, 0.9)       5.398686
## (100, 0.99)     51.821748
```

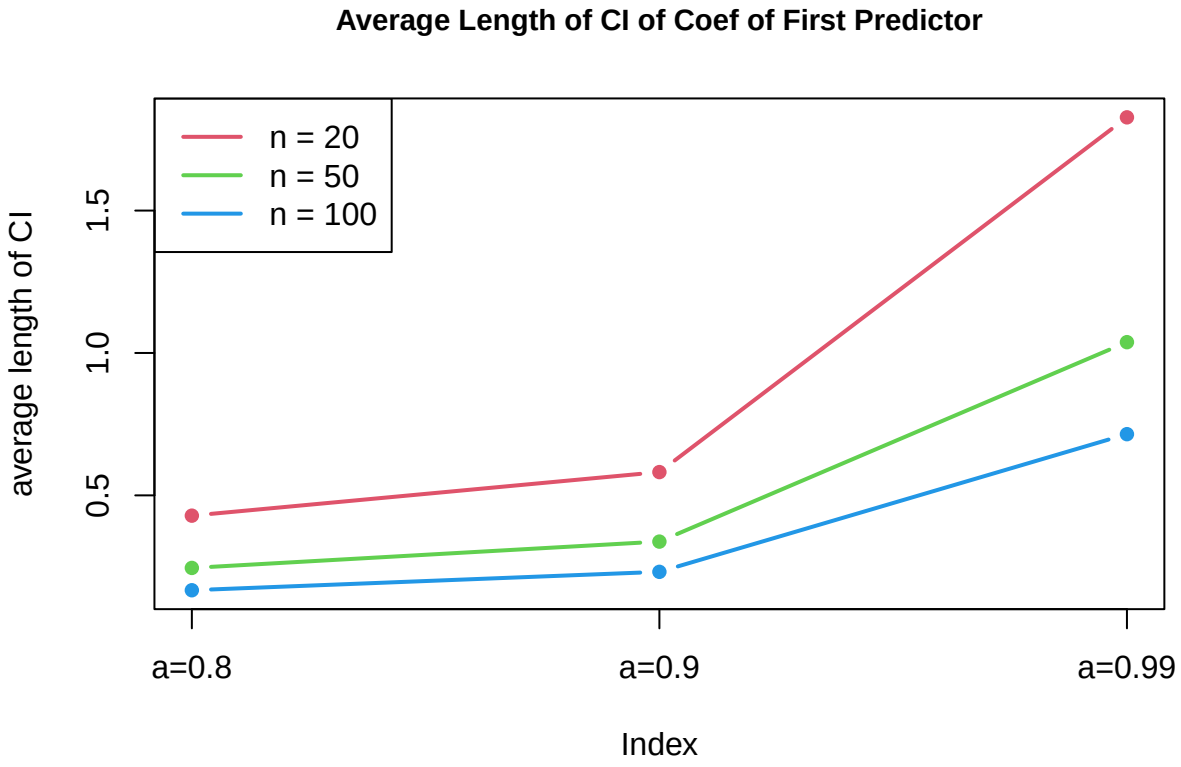
Above we have generated a matrix where each row is one of nine combination of sample size and a values. There are total nine of them. The first column stores the information average length of the CI of the intercept coefficient; the second column stores the information average length of the CI of the coefficient of first predictor x_1; the third column stores the information average length of the CI of the coefficient of second predictor x_2; the fourth column stores the information proportion of times the CI of the intercept coefficient contained the true value, which is -1; the fifth column stores the information proportion of times the CI of the coefficient corresponding to the first predictor contained the true value, which is 2; the sixth column stores the information proportion of times the CI of the coefficient corresponding to the second predictor contained the true value, which is -2; the seventh column (and the eighth) stores the information about the VIF value of the coefficient. Since we only have two predictions x_1 and x_2, their VIF value are the same. Therefore, we are going to just use column 7 in our analysis later.

In the following, we are going to generate some plot to helps us better understand these information.

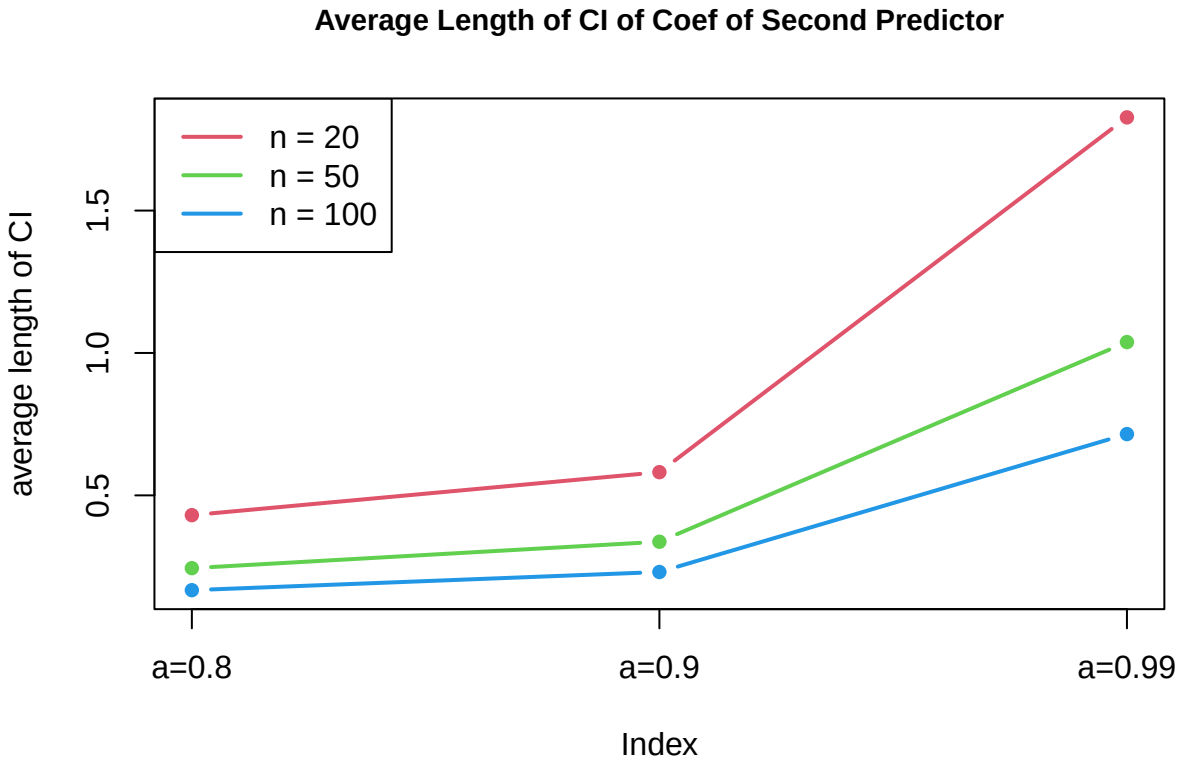
```
par(cex.main = 0.9)
plot(summary[c(1, 2, 3), 1], type = "b",
    pch = 16, col = 2, lwd = 2, ylim = range(summary[,
        1]), ylab = "average length of CI",
    main = "Average Length of CI of Intercept",
    xaxt = "n")
axis(1, at = c(1, 2, 3), labels = c("a=0.8",
    "a=0.9", "a=0.99"))
lines(summary[c(4, 5, 6), 1], type = "b",
    pch = 16, col = 3, lwd = 2)
lines(summary[c(7, 8, 9), 1], type = "b",
    pch = 16, col = 4, lwd = 2)
legend(legend = c("n = 20", "n = 50", "n = 100"),
    col = 2:4, lwd = 2, x = 2, y = 0.23)
```




```
par(cex.main = 0.9)
plot(summary[c(1, 2, 3), 2], type = "b",
    pch = 16, col = 2, lwd = 2, ylim = range(summary[,
        2]), ylab = "average length of CI",
    main = "Average Length of CI of Coef of First Predictor",
    xaxt = "n")
axis(1, at = c(1, 2, 3), labels = c("a=0.8",
    "a=0.9", "a=0.99"))
lines(summary[c(4, 5, 6), 2], type = "b",
    pch = 16, col = 3, lwd = 2)
lines(summary[c(7, 8, 9), 2], type = "b",
    pch = 16, col = 4, lwd = 2)
legend("topleft", legend = c("n = 20", "n = 50",
    "n = 100"), col = 2:4, lwd = 2)
```




```
par(cex.main = 0.9)
plot(summary[c(1, 2, 3), 3], type = "b",
    pch = 16, col = 2, lwd = 2, ylim = range(summary[,
        2]), ylab = "average length of CI",
    main = "Average Length of CI of Coef of Second Predictor",
    xaxt = "n")
axis(1, at = c(1, 2, 3), labels = c("a=0.8",
    "a=0.9", "a=0.99"))
lines(summary[c(4, 5, 6), 3], type = "b",
    pch = 16, col = 3, lwd = 2)
lines(summary[c(7, 8, 9), 3], type = "b",
    pch = 16, col = 4, lwd = 2)
legend("topleft", legend = c("n = 20", "n = 50",
    "n = 100"), col = 2:4, lwd = 2)
```



We first generated three plots about the average length of CI for each of the coefficients. In the first one related to the intercept coefficient, we see very clearly that as the sample size increase, the average length of the corresponding CI become shorter. This is within our expectation, since one way to get a more precise CI is to increase the sample size.

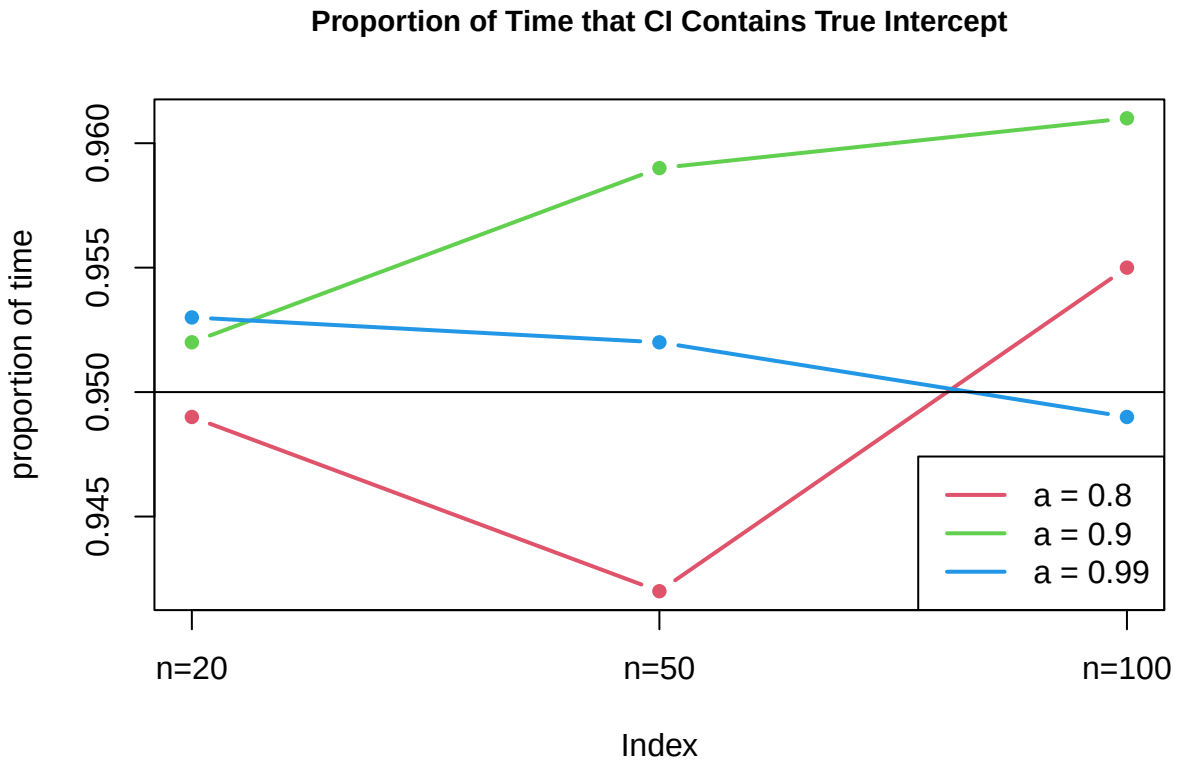
Moreover, we observe that with a fixed sample size, there is not much variation of the average CI length due to different a, which is also the value of $\text{cor}(x_1, x_2)$. This means that the CI of the intercept coefficient is not affected by the multicolinearity among the predictors. This aligns with the comment we made earlier when explaining why `R` does not return VIF for the intercept coefficient.

However, the other two plots show something differently. If we fix a, then as the sample size increasing from 20 to 100, the length of the CI of the coefficient corresponding to the second predictor is decreasing. On the other hand, if we fix n, then we see the larger the a, the wider the length of the CI. Therefore, it will be more difficult to obtain the more precise CI due to the multicolinearity among. The observation made from the third plot is same as the one from the second plot.

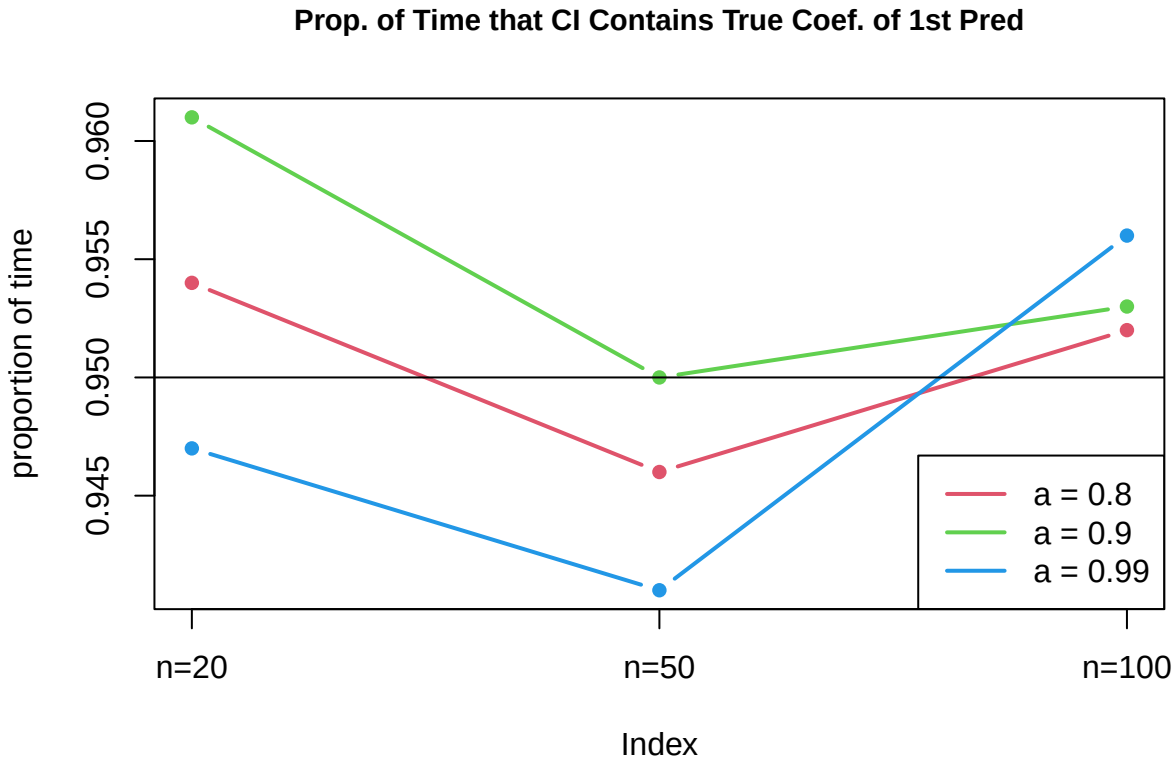
By comparing the first plots with the second and the third plots, we see the challenge brought up by the issue of multicolinearity.

```
par(cex.main = 0.9)
plot(summary[c(1, 4, 7), 4], type = "b",
    pch = 16, col = 2, lwd = 2, ylim = range(summary[,
        4]), ylab = "proportion of time",
    main = "Proportion of Time that CI Contains True Intercept",
    xaxt = "n")
axis(1, at = c(1, 2, 3), labels = c("n=20",
    "n=50", "n=100"))
lines(summary[c(2, 5, 8), 4], type = "b",
```

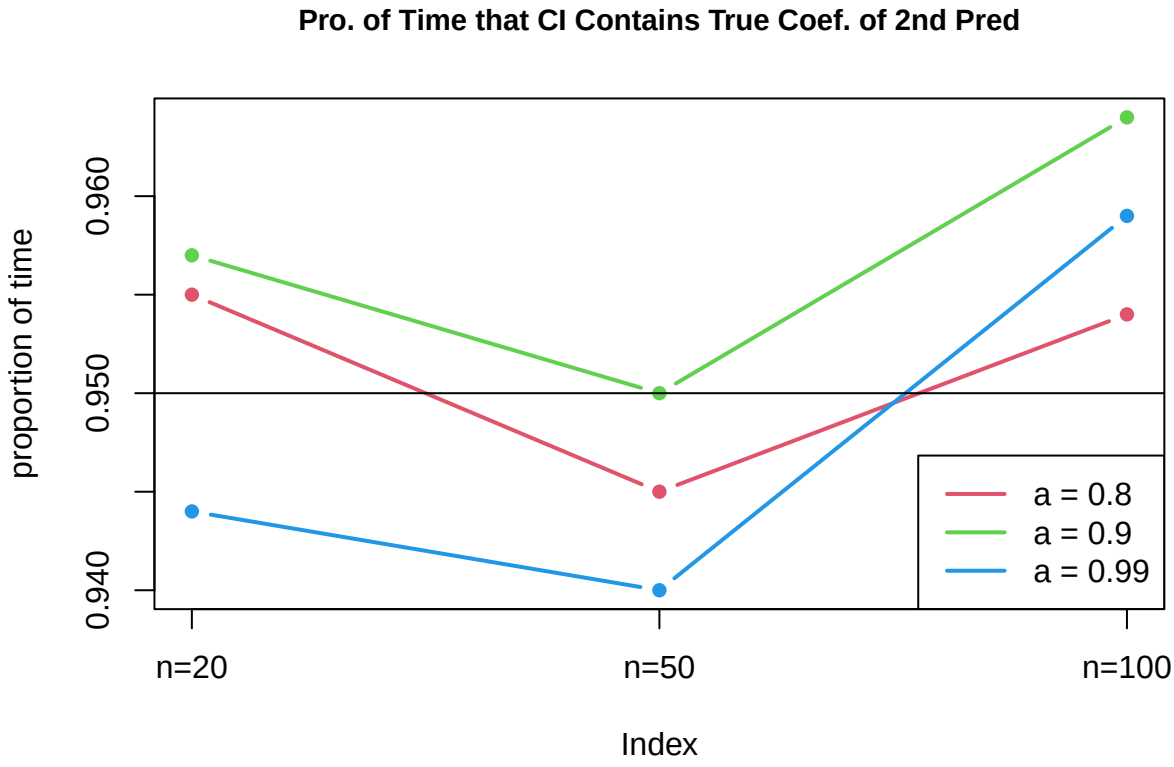
```
    pch = 16, col = 3, lwd = 2)
lines(summary[c(3, 6, 9), 4], type = "b",
    pch = 16, col = 4, lwd = 2)
abline(h = 0.95)
legend("bottomright", legend = c("a = 0.8",
    "a = 0.9", "a = 0.99"), col = 2:4, lwd = 2)
```




```
par(cex.main = 0.9)
plot(summary[c(1, 4, 7), 5], type = "b",
    pch = 16, col = 2, lwd = 2, ylim = range(summary[,
        5]), ylab = "proportion of time",
    main = "Prop. of Time that CI Contains True Coef. of 1st Pred",
    xaxt = "n")
axis(1, at = c(1, 2, 3), labels = c("n=20",
    "n=50", "n=100"))
lines(summary[c(2, 5, 8), 5], type = "b",
    pch = 16, col = 3, lwd = 2)
lines(summary[c(3, 6, 9), 5], type = "b",
    pch = 16, col = 4, lwd = 2)
abline(h = 0.95)
legend("bottomright", legend = c("a = 0.8",
    "a = 0.9", "a = 0.99"), col = 2:4, lwd = 2)
```




```
par(cex.main = 0.9)
plot(summary[c(1, 4, 7), 6], type = "b",
    pch = 16, col = 2, lwd = 2, ylim = range(summary[,
        6]), ylab = "proportion of time",
    main = "Pro. of Time that CI Contains True Coef. of 2nd Pred",
    xaxt = "n")
axis(1, at = c(1, 2, 3), labels = c("n=20",
    "n=50", "n=100"))
lines(summary[c(2, 5, 8), 6], type = "b",
    pch = 16, col = 3, lwd = 2)
lines(summary[c(3, 6, 9), 6], type = "b",
    pch = 16, col = 4, lwd = 2)
abline(h = 0.95)
legend("bottomright", legend = c("a = 0.8",
    "a = 0.9", "a = 0.99"), col = 2:4, lwd = 2)
```



We shall explore and try to answer the question how the level of multicollinearity affects the relationship between sample size and the proportion of times the confidence interval includes the true parameter.

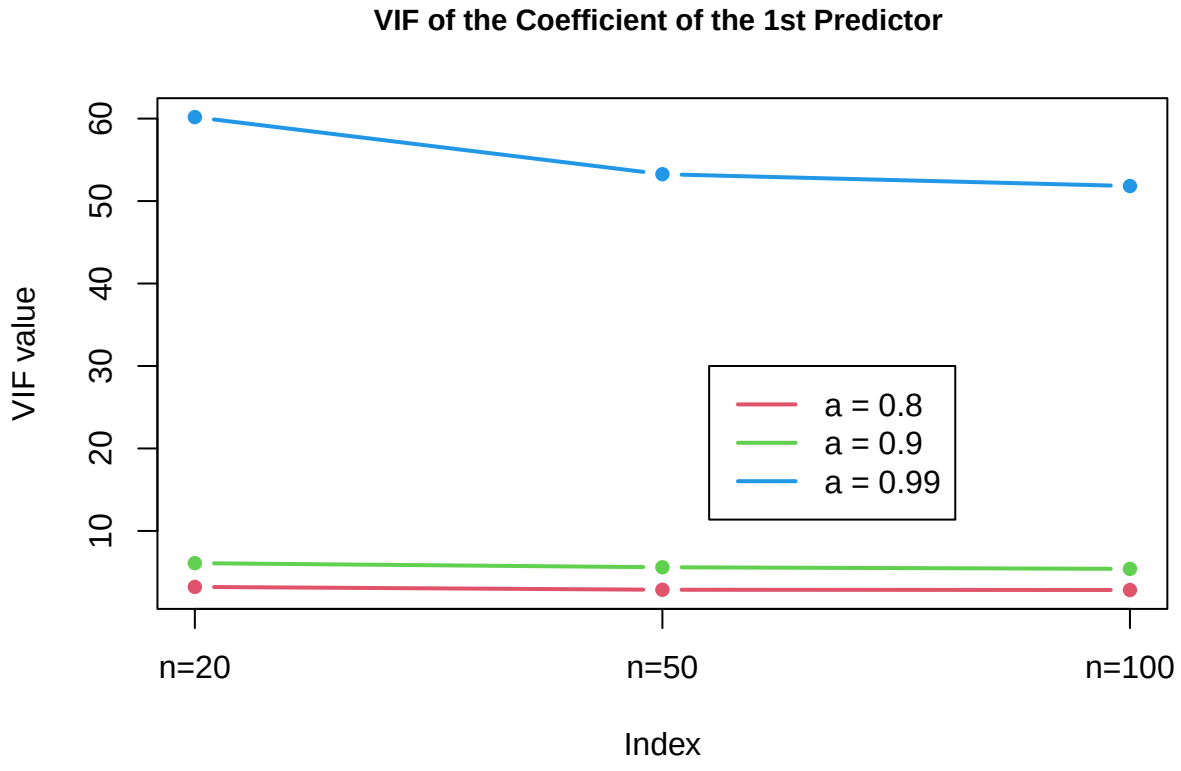
Please note that the following observation is made based on one complete run of the code. Due to the randomness, the pattern of the proportion of times the confidence interval includes the true parameter is not always the same. We wrote down the following observation based on one run of the code; however, after rendering the document, due to the randomness from the simulation, the pattern changes, and the following comment may not be applied any more. We have tried to render multiple times but overall the patter is difficulte to generalize than the average length of CI above or VIF below. While reading the following comment, please keep in mind that it is based on one run, and the description may be not be aligh with the graph generated in this document. We conclude that the proportion of times the confidence interval includes the true parameter does influence by multicolinearity, but it is hard to give a general statement on their relationship. we could also have use `set.seed()`, but it is interesting to be aware of this unstability of the pattern.

General theory tells us the larger the sample size, the smaller the width of the CI. Overall compare to the CI with smaller sample size, there is a higher the probability that the true parameter is included in that shorter interval. However, we see that when there is high multicollinearity among the predictor variables, the effect of increasing sample on the proportion of time the confidence interval includes the true parameter is a little bit ambiguous.

One observation is that from the second and the third plot, with higher level of multicollinearity, the proportion of time the confidence interval includes the true parameter is lower. We have learned that with higher multicollinearity, the estimated coefficient will have a higher variance, therefore higher standard error, which leads to a wider CI. If the CI is wider, then the range of possible parameter values is larger, and the probability of containing the true parameter is lower. (Just a review: the confidence level is fixed as 95, but the width of the confidence interval varies due to variability of data. So a wider CI means that there is more uncertainty about the true parameter, which leads a lower probability of containing it).

We did observe that with the combination $a = 0.8, n = 100$, the proportion of time the CI contains the true parameter is the lowest across all three plots. However we did not find a good theoretical explanation for this. Moreover, in the first plot, the blue line is on the top, which illustrate that the proportion of time the CI contains the true parameter is higher with level of multicollinearity 0.99, we did not find a good theoretical either. Lastly, still from the first plot, we see that the proportion of time the CI contains the true parameter is decreasing as the sample size increase, which does not align with the theory that increasing the sample size will reduce the variability of data therefore get a more precise confidence interval.

```
par(cex.main = 0.9)
plot(summary[c(1, 4, 7), 7], type = "b",
    pch = 16, col = 2, lwd = 2, ylim = range(summary[,
        7]), ylab = "VIF value", main = "VIF of the Coefficient of the 1st Predictor ",
    xaxt = "n")
axis(1, at = c(1, 2, 3), labels = c("n=20",
    "n=50", "n=100"))
lines(summary[c(2, 5, 8), 7], type = "b",
    pch = 16, col = 3, lwd = 2)
lines(summary[c(3, 6, 9), 7], type = "b",
    pch = 16, col = 4, lwd = 2)
legend(legend = c("a = 0.8", "a = 0.9", "a = 0.99"),
    col = 2:4, lwd = 2, x = 2.1, y = 30)
```



We observe that the highest level of multicollinearity has the highest VIF across all three cases with different sample sizes. Visually, the blue line is on the top of the red and green. Moreover, by looking at the blue line, we found that as the sample increase, the VIF value ddecreases a little bit. This aligns with the theory that increasing sample size will decrease the variability. It is less obvious that the the green line and the red

lines are both decreasing as the sample size increasing from 20 to 100, but we observed from the `summary` table that it is indeed what is happening.