Lab 4

Spring 2025

Simulations

Today's lab will primarily be using simulated data. Often times, we want to assess how a statistical method might perform in practice. We can apply the methods to real data and see if the results make, but the actual performance can be hard to assess if we don't know what the "correct answer" is. Simulations allow us to assess performance by generating data where we know the "correct answer." Simulations are also helpful because they allow us to approximate quantities that might be hard to calculate exactly.

Simulations for approximation

Steph Curry and Sabrina Ionescu are both basketball players and recently competed against each other in a 3 point shooting contest (3 pointers are shots made from beyond a certain distance). Curry's career 3 point shooting percentage (the proportion of times he succesfully makes a 3 point shot) is .427 and Ionescu's career 3 point shooting percentage is .377

Suppose the players are playing a game where they each take 10 3 point shots, and compare who makes more. Let's assume that, for each player, every free throw has the same probability of success and is independent of the other free throws. Calculating the proportion of times Steph will win, Sabrina will win, or there will be a tie is not so easy, but we can easily simulate the outcomes. The code below does exactly that. We use 1000 replications of data to approximate what would happen if we did this an infinite amount of times

```
prob.curry <- .427
prob.ionescu <- .377

## Number of games we will simulate
sim.size <- 1000
## number of free throws in each game
num.shots <- 10



winner <- rep("", sim.size)

for(i in 1:sim.size){

  # Number of succesful free throws for durant
  curry <- rbinom(n = 1, size = num.shots, prob = prob.curry)
# Number of succesful free throws for beal
  ionescu <- rbinom(n = 1, size = num.shots, prob = prob.ionescu)

if(curry > ionescu){
  winner[i] <- "curry"
  } else if (ionescu > curry){
    winner[i] <- "ionescu"
  } else{
```

```
    winner[i] <- "tie"
  }
}
table(winner) / sim.size
```

```
## winner
##   curry ionescu     tie
##   0.528   0.323   0.149
```

Question

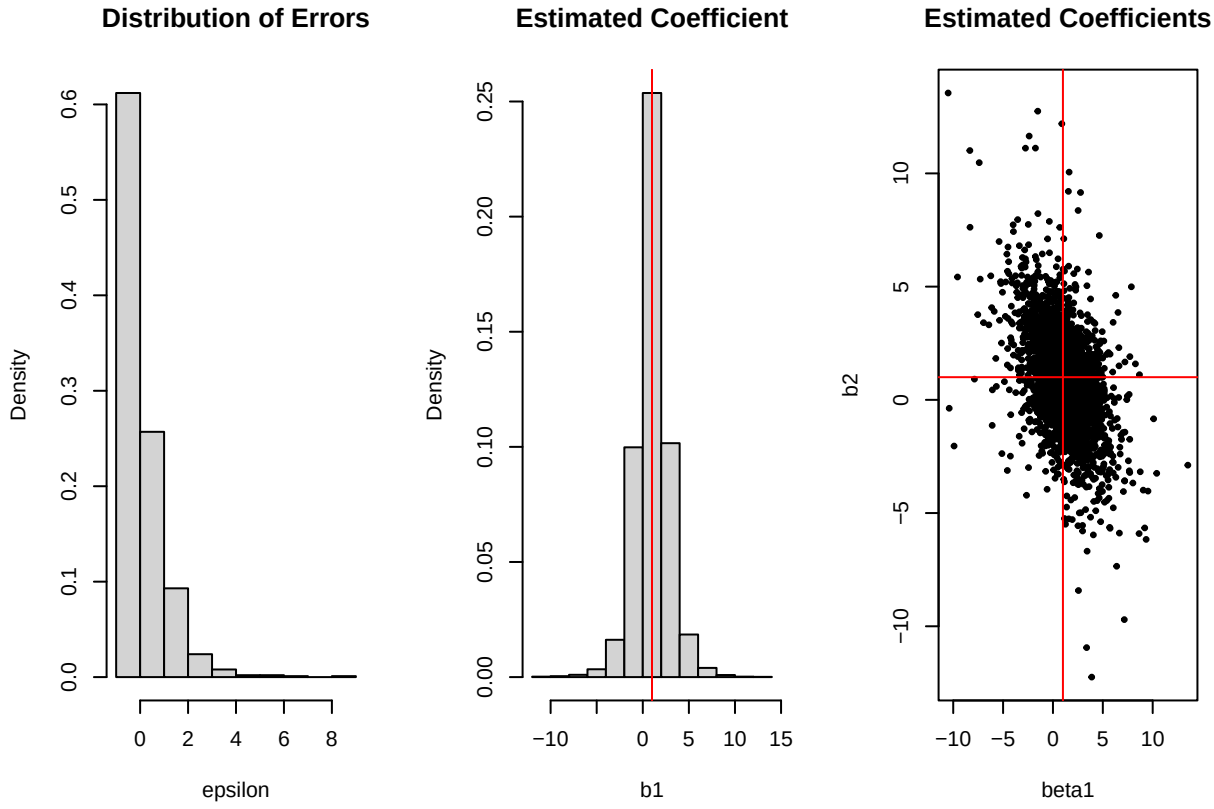
- What do you think would happen if they each shot 20 free throws instead of 10? Would the probability of victory for each player change? Why or why not?
- Test it out. Change `num.ft` to $20, 50, 100$ and see what happens

Sampling Distributions

We will use a small simulation study to examine how the sampling distribution of estimated coefficients depends on In particular, we will simulate many different data sets and record the estimated coefficients each time. We can then look at the distribution of the estimates, and by changing certain features of the data generating process, we can see how it effects the distribution of the resulting estimates. The true value of each coefficient is 1. Play around with each of the parameters below, and see how the resulting sampling distribution of the estimated coefficients changes.

```
#### Change the code below ####

# number of observations in the sample
n <- 8
# number of covariates (must be less than n)
p <- 3
# standard deviation of the X values
x.sd <- 1
# covariance of the X values (choose a positive value less than x.sd)
rho <- .95

# distribution of the errors (choose either "normal" or "gamma" or "T")
# gamma distribution is skewed, T distribution has outliers
errDist <- "gamma"

# Standard deviation of the epsilon terms
err.sd <- 1



#### Don't change the code below ####
# Number of times we will simulate a new data set
sim.size <- 5000
# True coefficients
beta <- rep(1, p)
rec <- matrix(0, sim.size, p)
# Covariance matrix of X
cov.mat <- matrix(rho, p,p); diag(cov.mat) <- x.sd^2

# Helper function to draw errors
```

```r
drawErrors <- function(n, sd = 1, type = "normal"){
  if(type == "normal"){
    err <- rnorm(n)
  } else if (type == "gamma"){
    err <- rgamma(n, 1, 1) - 1
  } else if (type == "T"){
    err <- rt(n, df = 3)
  }
  return(err * sd)
}

for(i in 1:sim.size){
  X <- mvtnorm::rmvnorm(n, sigma = cov.mat)
  Y <- X %*% beta + drawErrors(n, sd = err.sd, type = errDist)

  mod1 <- lm(Y~X - 1)
  rec[i, ] <- mod1$coefficients
}

par(mfrow = c(1,3))
hist(drawErrors(1000, sd = err.sd, type = errDist), main = "Distribution of Errors", xlab = "epsilon", 
hist(rec[, 1], main = "Estimated Coefficient", xlab = "b1", freq = F)
abline( v = 1, col = "red")
if(p > 1){
  plot(rec[ ,1], rec[ ,2], xlab = "beta1", ylab = "b2", main = "Estimated Coefficients", pch = 19, cex =
  abline(v = 1, h = 1, col = "red")
}
```



Questions

- What is the mean of the distribution of $\hat{b}_1$ in each case?
- How do the variances of the two distributions (normal errors vs gamma errors) compare? How do the shapes compare?
- Increase x.sd from 1 to 3 and re-run the simulation. This will cause the X values to be more spread out. What happens to to the distributions of $\hat{b}_1$?
- Change x.sd back to 1 and set n to 50. What happens to to the distributions of $hatb_1$? How do the variances of the two distributions (normal errors vs gamma errors) compare? How do the shapes compare?

Sampling distribution of $\hat{\sigma}_\varepsilon^2$

Let's take a look at the sampling distribution of $\hat{\sigma}_\varepsilon^2$. We also record three different estimators of the variance of ε_i:

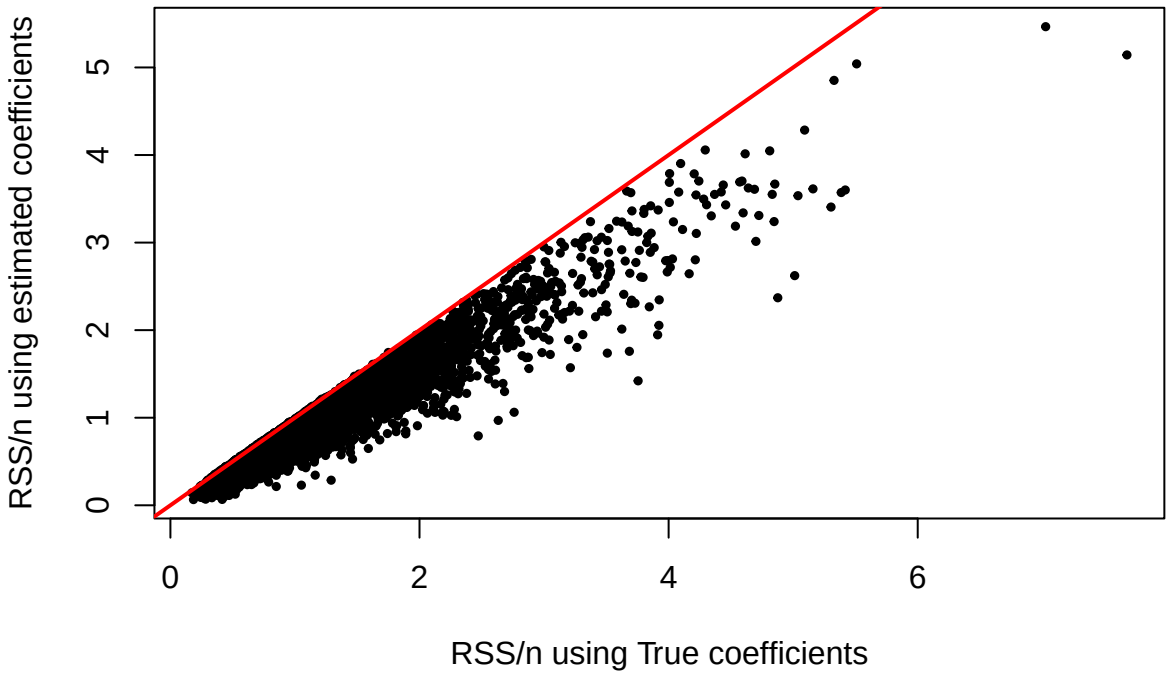
- An estimate which uses the true errors, ε_i. In practice, we can't compute this since we won't know the true errors, but since this is simulated data, we can. This would be the same as using the residuals if we knew the true linear coefficients.
$$\frac{1}{n}\sum_i \varepsilon_i^2$$
- An estimate which uses the residuals, $\hat{\varepsilon}_i = y_i - \hat{y}_i = y_i - \sum_k^p \hat{b}_k x_{i,k}$, but doesn't adjust for the fact that we are using residuals and not the true errors.
$$\frac{1}{n}\sum_i \hat{\varepsilon}_i^2$$
- An estimate which uses the residuals $\hat{\varepsilon}_i = y_i - \sum_k^p \hat{b}_k x_{i,k}$, but does adjust for the fact that we are using residuals and not the true errors by dividing by $n - p - 1$
$$\frac{1}{n-p-1}\sum_i \varepsilon_i^2$$

```
#### Change the code below ####

# number of observations in the sample
n <- 20
# number of covariates (must be less than n)
p <- 3
# standard deviation of the X values
x.sd <- 1
# covariance of the X values (choose a positive value less than x.sd)
rho <- 0

# distribution of the errors (choose either "normal" or "gamma" or "T")
# gamma distribution is skewed, T distribution has outliers
errDist <- "gamma"

# Standard deviation of the epsilon terms
err.sd <- 1



#### Don't change the code below ####
# Number of times we will simulate a new data set
sim.size <- 10000
# True coefficients
beta <- rep(1, p)
rec <- matrix(0, sim.size, 5)
# Covariance matrix of X
cov.mat <- matrix(rho, p,p); diag(cov.mat) <- x.sd^2

# Helper function to draw errors
drawErrors <- function(n, sd = 1, type = "normal"){
  if(type == "normal"){
    err <- rnorm(n)
```

```
  } else if (type == "gamma"){
    err <- rgamma(n, 1, 1) - 1
  } else if (type == "T"){
    err <- rt(n, df = 3)
  }
  return(err * sd)
}

for(i in 1:sim.size){
  X <- mvtnorm::rmvnorm(n, sigma = cov.mat)
  errs <- drawErrors(n, sd = err.sd, type = errDist)
  Y <- X %*% beta + errs

  mod1 <- lm(Y ~ X)
  # RSS(b) / n: we can calculate this using the true errors, which we know because
  # it's a simulation, but in practice we would need to know the true coefficients
  # to calculate the true errors
  true_errors <- sum(errs^2) / n

  # RSS(b hat) / n: we can calculate this using the residuals, but we don't
  # adjust for the fact that we are using residuals and not the true errors
  resid_unadjust <- sum(mod1$res^2) / n

  # RSS(b hat) / (n-p): we can calculate this using the residuals, and now we
  # adjust for the fact that we are using residuals and not the true errors
  resid_adjust <- sum(mod1$res^2) / (n-p - 1)

  # record each of the estimators
  rec[i, ] <- c(true_errors,
                resid_unadjust,
                resid_adjust, mean(errs^2), mean(mod1$residuals^2))
}
```

We can first compare the RSS using the true coefficients to the RSS using the estimated coefficients. In the plot below, each point represents the outcome of one replication.

```
plot(rec[,4], rec[,5], xlab = "RSS/n using True coefficients", ylab = "RSS/n using estimated coefficien
abline(a = 0, b = 1, lwd= 2, col = "Red")
```

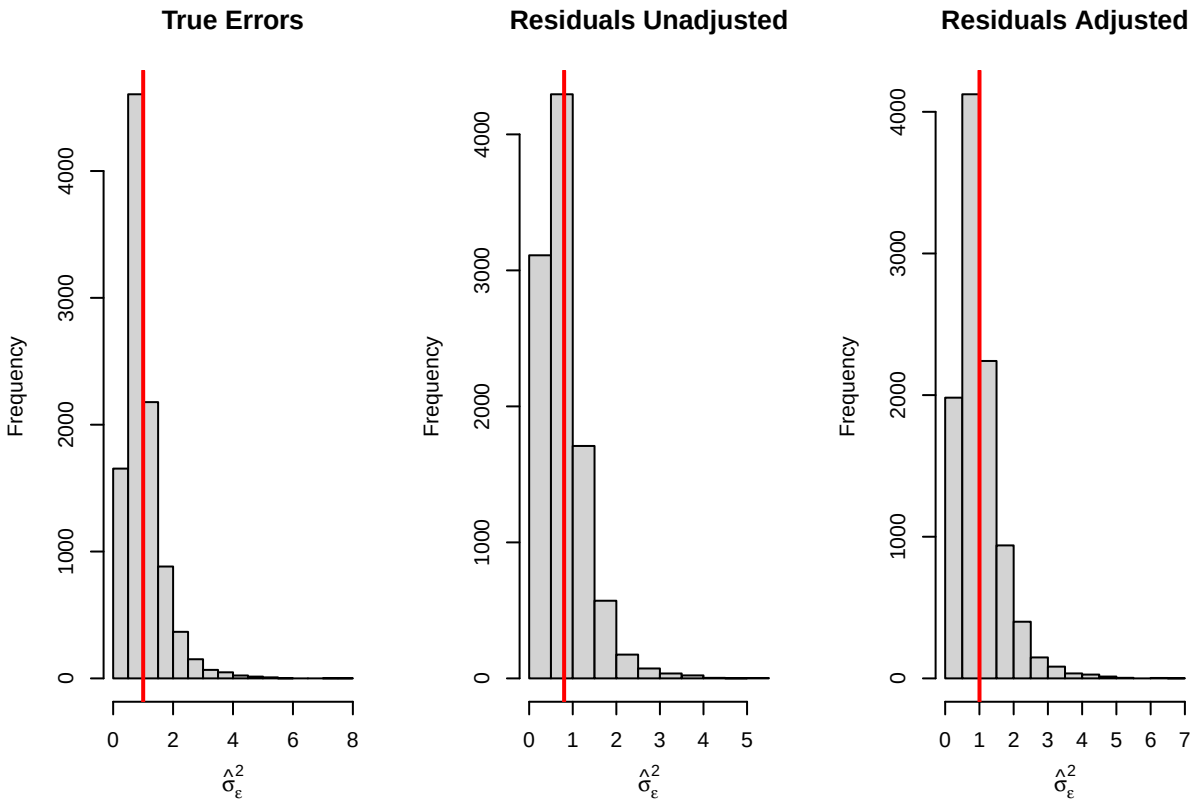


Questions

- Given the plot, what can you conclude about the RSS using the true coefficients vs the RSS using the estimated coefficients?
- How would this change if you changed p and n?

We can plot histograms of each of the estimators of $\hat{\sigma}^2_\varepsilon$.

```
par(mfrow = c(1,3))
# We're using some fancy code to label the axis
# We won't cover this because of time, but the following is a good tutorial
# if you are interested in learning more:
# https://www.dataanalytics.org.uk/axis-labels-in-r-plots-using-expression/

# Histogram of estimator using true errors
hist(rec[, 1], main = "True Errors", xlab = expression(hat(sigma)[epsilon]^2))
# draw a red vertical line at the mean
abline(v = mean(rec[, 1]), col = "red", lwd = 2)

# Histogram of estimator using residuals, but unadjusted
hist(rec[, 2], main = "Residuals Unadjusted", xlab = expression(hat(sigma)[epsilon]^2))
# draw a red vertical line at the mean
abline(v = mean(rec[, 2]), col = "red", lwd = 2)

# Histogram of estimator using residuals, but adjusted
hist(rec[, 3], main = "Residuals Adjusted", xlab = expression(hat(sigma)[epsilon]^2))
# draw a red vertical line at the mean
abline(v = mean(rec[, 3]), col = "red", lwd = 2)
```



We can calculate the mean and variance of each of the estimators. As we can see, the mean of the estimators using the true errors and the mean of the estimator which uses the residuals and adjusts for them are pretty close to the actual value of $\sigma^2_\varepsilon = 1$. However, the mean of the estimator using the residuals and not adjusting is further from the true value.

```
# mean and variance of the estimator using the true errors
print("Using true coefficients")
```

```
## [1] "Using true coefficients"
```

```
mean(rec[, 1])
```

```
## [1] 1.002261
```

```
var(rec[, 1])
```

```
## [1] 0.3978981
```

```
# mean and variance of the estimator using the residuals but not adjusting
print("Using estimated coefficients without adjusting")
```

```
## [1] "Using estimated coefficients without adjusting"
```

```
mean(rec[, 2])
```

```
## [1] 0.8019886
```

```
var(rec[, 2])
```

```
## [1] 0.2687509
```

```
# mean and variance of the estimator using the residuals and adjusting
print("Using estimated coefficients with adjusting")
```

```
## [1] "Using estimated coefficients with adjusting"
```

```
mean(rec[, 3])
```

```
## [1] 1.002486
```

```
var(rec[, 3])
```

```
## [1] 0.4199233
```

Questions:

- Keep $n = 20$ but increase p to be $5, 10, 15$. What happens to the mean of each of the estimators? What happens to the variance of each of the estimators? Keep $p = 3$ but increase n to be $50, 100, 150$. What happens to the mean of each of the estimators? What happens to the variance of each of the estimators?

Confidence Intervals

We will start by seeing how to calculate a confidence interval in R.

```
# number of observations in the sample
n <- 20
# number of covariates (must be less than n)
p <- 5

# Draw p covariates which are all independent of each other
X <- mvtnorm::rmvnorm(n, sigma = diag(p))

# The true coefficients are 1
beta <- rep(1, p)
# draw Y with no Y intercept and Gaussian errors
Y <- X %*% beta + rnorm(n)

# fit the model
mod1 <- lm(Y~X1 + X2 + X3 + X4 + X5, data = data.frame(X))

# The confidence interval for each coefficient can be calculated using confint
confint(mod1, level = .95)
```

```
##                   2.5 %   97.5 %
## (Intercept) -0.73227717 1.046222
## X1           0.56825536 2.078015
## X2           0.35946896 1.749718
## X3           0.41708777 1.939599
## X4          -0.21646720 1.890987
## X5          -0.02732268 1.555570
```

The confidence interval you calculate will be different from your neighbor's interval (it will even change each time you knit this document), and it either does or does not contain the true parameter (which happens to be 1). So it doesn't make sense to say "there's a 95% chance that $[.49, 1.5]$ (or whatever numbers you actually got) contains 1."

Questions

- Check around the room and see how many people got a confidence interval that does not contain 1. What proportion of the students have a confidence interval that contains the truth? Hopefully it's close to .95.

Instead of making a statement about a specific interval, the 95% chance describes the probability that the procedure will contain the truth when applied to a new set of data. Or thought another way, when repeating the procedure an infinite number of times with new data each time, 95% of the resulting confidence intervals will contain the true parameter. We use the phrase:

"We are 95% confident that the true coefficient is between $[.49, 1.5]$ (or whatever numbers you actually got)"

as a shorthand way of encoding that longer explanation.

We can also get Confidence intervals for the conditional mean and prediction intervals

```
## The confidence interval for conditional mean
# covariate values at which to get confidence interval for conditional mean
newdf <- data.frame(X1 = 1, X2 = 5, X3 = .5, X4 = 2, X5 = .5)
predict(mod1, newdata = newdf, interval = "confidence")
```

```
##        fit      lwr      upr
## 1 9.398827 5.467471 13.33018
```

```
## The prediction interval
# covariate values at which to get prediction interval
newdf <- data.frame(X1 = 1, X2 = 5, X3 = .5, X4 = 2, X5 = .5)
predict(mod1, newdata = newdf, interval = "prediction")
```

```
##        fit      lwr      upr
## 1 9.398827 4.774565 14.02309
```

Questions

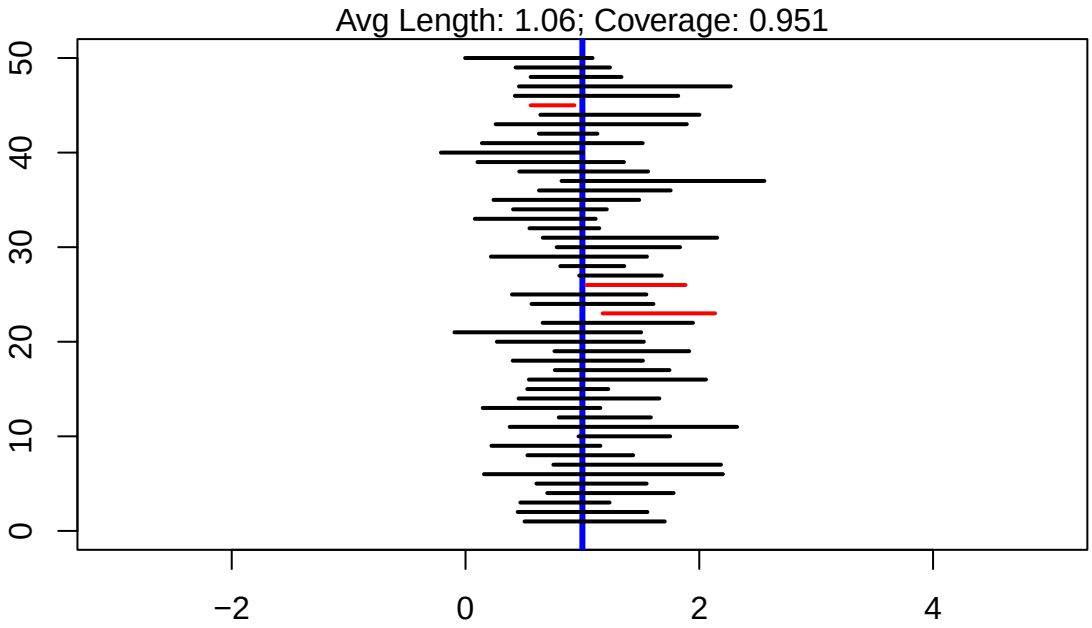
- Compare the length of the confidence interval for the conditional mean to the length of the prediction interval. Which is longer? Explain why it would be longer.

CI simulations

We go back to the same code as before, where we looked at the sampling distribution of $\hat{b}_1$. But this time, we will look at confidence intervals and how the length of confidence intervals change depending on characteristics of the population and the confidence level of the CI.

```
#### Change the code below ####

# number of observations in the sample
n <- 20
# number of covariates (must be less than n)
p <- 3
# standard deviation of the X values
x.sd <- 1
# covariance of the X values (choose a positive value less than x.sd)
rho <- .2

# distribution of the errors (choose either "normal" or "gamma" or "T")
# gamma distribution is skewed, T distribution has outliers
errDist <- "gamma"

# Standard deviation of the epsilon terms
err.sd <- 1

# The target proportion of times that the confidence interval should contain the truth
```

```r
ci_level <- .95


#### Don't change the code below ####
# Number of times we will simulate a new data set
sim.size <- 5000
# True coefficients
beta <- rep(1, p)
ci <- matrix(0, sim.size, 2)
# Covariance matrix of X
cov.mat <- matrix(rho, p,p); diag(cov.mat) <- x.sd^2

# Helper function to draw errors
drawErrors <- function(n, sd = 1, type = "normal"){
  if(type == "normal"){
    err <- rnorm(n)
  } else if (type == "gamma"){
    err <- rgamma(n, 1, 1) - 1
  } else if (type == "T"){
    err <- rt(n, df = 3)
  }
  return(err * sd)
}

for(i in 1:sim.size){
  X <- mvtnorm::rmvnorm(n, sigma = cov.mat)
  Y <- X %*% beta + drawErrors(n, sd = err.sd, type = errDist)

  mod1 <- lm(Y~X)

  ### use confint
  confidence_interval <- confint(mod1, level = ci_level)
  ci[i, ] <- confidence_interval[2, ]
}


avgLength <- mean(ci[,2] - ci[,1])
coverage <- mean(ci[,1] < 1 & ci[,2] > 1)
plot(-5, 5, xlim = c(-3, 5), ylim = c(0, 50), xlab = "", ylab = "")
abline(v = 1, col = "blue", lwd = 3)
# only plot 50 of the 5000 replications
for(i in 1:50){
  segments(ci[i, 1], i, ci[i, 2], i, col = ifelse(ci[i,1] < 1 & ci[i,2] > 1, "black", "red"), lwd =2)
}
mtext(paste("Avg Length: ", round(avgLength, 2), "; Coverage: ", round(coverage,3), sep = ""), cex = 1)
```

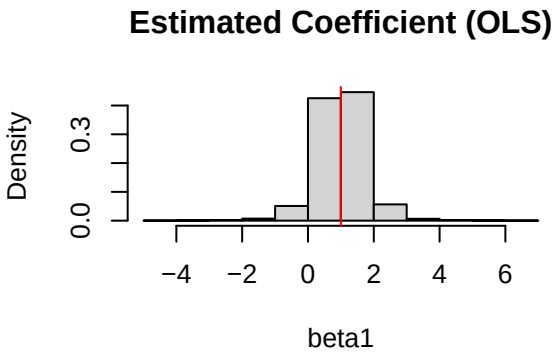


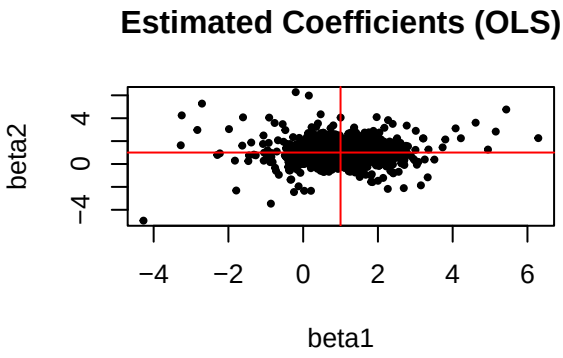
Quantile Regression

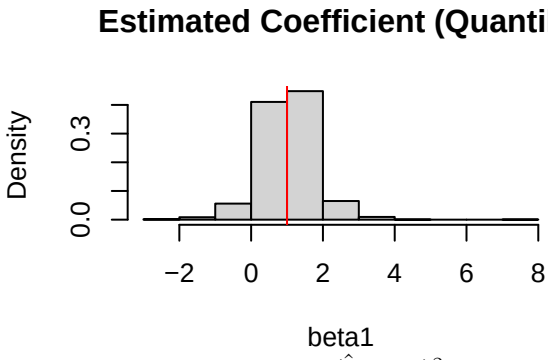
It is not as easy to describe the sampling distribution of the least absolute deviation estimator; i.e., picking $\hat{b}$ by minimizing $\sum_i |y_i - \hat{y}_i|$ instead of $\sum_i (y_i - \hat{y}_i)^2$. Nonetheless, simulations allow us to approximate it well. Below, we compare the (ordinary) least squares estimator with the least absolute deviation estimator.

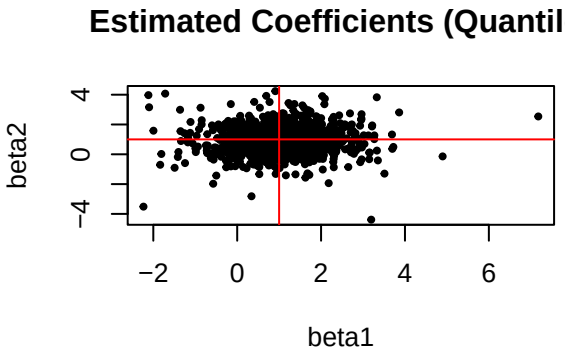
```
#### Change the code below ####

# number of observations in the sample
n <- 10
# number of covariates (must be less than n)
p <- 3
# standard deviation of the X values
x.sd <- 1
# covariance of the X values (choose a positive value less than x.sd)
rho <- 0

# distribution of the errors (choose either "normal" or "gamma" or "T")
# gamma distribution is skewed, T distribution has outliers
errDist <- "T"

# Standard deviation of the epsilon terms
err.sd <- 1



#### Don't change the code below ####
# Number of times we will simulate a new data set
sim.size <- 2000
# True coefficients
beta <- rep(1, p)
recOLS <- recQR <- matrix(0, sim.size, (p + 1))
# Covariance matrix of X
cov.mat <- matrix(rho, p,p); diag(cov.mat) <- x.sd^2
```

```r
# Helper function to draw errors
drawErrors <- function(n, sd = 1, type = "normal"){
  if(type == "normal"){
    err <- rnorm(n)
  } else if (type == "gamma"){
    err <- rgamma(n, 1, 1) - 1
  } else if (type == "T"){
    err <- rt(n, df = 3)
  }
  return(err * sd)
}

for(i in 1:sim.size){
  X <- mvtnorm::rmvnorm(n, sigma = cov.mat)
  Y <- X %*% beta + drawErrors(n, sd = err.sd, type = errDist)

  mod1 <- lm(Y~X)
  mod2 <- quantreg::rq(Y~X)
  recOLS[i, ] <- mod1$coefficients
  recQR[i, ] <- mod2$coefficients
}

par(mfrow = c(2,2))
hist(recOLS[, 2], main = "Estimated Coefficient (OLS)", xlab = "beta1", freq = F)
abline( v = 1, col = "red")
if(p > 1){
  plot(recOLS[ ,2], recOLS[ ,3], xlab = "beta1", ylab = "beta2", main = "Estimated Coefficients (OLS)",
  abline(v = 1, h = 1, col = "red")
}

hist(recQR[, 2], main = "Estimated Coefficient (Quantile)", xlab = "beta1", freq = F)
abline( v = 1, col = "red")
if(p > 1){
  plot(recQR[ , 2], recQR[ , 3], xlab = "beta1", ylab = "beta2", main = "Estimated Coefficients (Quanti
  abline(v = 1, h = 1, col = "red")
}
```









On average we can see that $(\hat{b}_1 - b_1)^2$ for Least squares and Quantile Regression are:

```
cat("OLS: ")
```

```
## OLS:
```

```
mean((recOLS[, 2] - 1)^2)
```

```
## [1] 0.5637601
```

```
cat("QR: ")
```

```
## QR:
```

```
mean((recQR[, 2] - 1)^2)
```

```
## [1] 0.5478441
```

Questions

- Play around with the parameters. Are there settings where OLS is preferred to quantile regression? Are there settings where quantile regression is preferred to OLS?