Mini Project #2

STA302H Summer 2020

Ruo Ning Qiu

University of Toronto

June 8, 2020

Outline

1. Introduction
   - Goal
   - Set up
2. Task 1: Q1-2 - SLR vs MLR
   - Simulation
   - The mean and variance of the estimators for SLR
   - The mean and variance of the estimators for MLR
   - Check whether the estimators are unbiased
3. Task1: Q3-4 - Correlated Predictors
   - Correlated predictors for X_1 and X_2
   - Correlated predictors for X_1 and X_3

Outline

4. Task 2: Q1-2 - Non-Linear Relation
   - Set up a new data-set with a non-linear relationship
   - Diagnostic Checks
   - Prediction Error

5. Task2: Q3 - Transformation
   - Box-Cov and True Transformation
   - Diagnostic Checks
   - Prediction Error

6. Conclusion
   - Key Takeaways

Introduction

In this project, we will examine the properties of multiple linear regression by running several simulations and diagnostic checks.

Our goal is the following:

To observe and understand

1. What happens when the predictor variables are correlated
    1. The effects on estimators, its variance and biasness as correlation increases
2. How model diagnostics work when the true model is non-linear
    1. Check residual plots, influential observations...
    2. Whether appropriate transformation helps to solves the problem of non-linearity

Set up

Define the population parameters under my student number seed, and a helper function to save repetitive codes.

```r
library(MASS)
set.seed(1004079631)
nsample <- 10; nsim <- 100 # Set 100 simulations
sig2 <- rchisq(1, df = 1) ## The true error variance
bet <- c(rnorm(3, 0, 1), 0) ## 4 values of beta that is beta0, beta1, beta2, beta3 = 0
beta0 <- bet[1]; beta1 <- bet[2]; beta2 <- bet[3]; beta3 <- bet[4]  ## the population regression parameters

muvec <- rnorm(3, 0, 1)
sigmat <- diag(rchisq(3, df = 4)) # Assume correlation between the three predictors are zero, so off diagonals
X <- mvrnorm(nsample, mu = muvec, Sigma = sigmat)
Xmat <- cbind(1, X)
sig2hat <- vector()  # saves the sample estimates of error variance sigma^2s

# To save some time, let's define a function that calculates the true varaiance of estimators: Helper function
var_true <- function(X, sig2, nsample, beta){
  sumx <- sum(X); xbar <- sumx/nsample
  sumx2 <- sum(X^2); SXX <- sumx2 - nsample*(xbar^2)
  var_beta_0 <- sig2*(1/nsample + xbar^2/SXX)
  var_beta_1 <- sig2/SXX

  if (beta == 0){
    return(var_beta_0)
  }
  return(var_beta_1)
}
```

Set up simulation for Q1

Define vectors to store each simulation's data.

```
## Task1 : Q1 ##
set.seed(1004079631)
b0_1 <- vector(); b0_2 <- vector(); b0_3 <- vector() # saves the sample estimates of beta_0 for SLR of X1-3 pre
b1 <- vector(); b2 <- vector(); b3 <- vector()  # saves the sample estimates of beta1-3 of SLR with X1-3 predic
var_b0_1 <- vector(); var_b0_2 <- vector(); var_b0_3 <- vector() # save variance of b0 of SLR with X1-3 predict
var_b1 <- vector(); var_b2 <- vector(); var_b3 <- vector()   # save variance of b1, b2, b3 of SLR
# saves the sample estimates of error variance sigma^2 with SLR model on X1-3 predictor
sig2hat1 <- vector(); sig2hat2 <- vector(); sig2hat3 <- vector()

bets <- matrix(NA, ncol = length(bet), nrow = nsim)  # MLR: save sample estimators of beta0, beta1, beta2 and b
vars <- matrix(NA, ncol = length(bet), nrow = nsim)  # MLR: save sample variance of b0, b1, b2, b3
X1 <- X[,1]; X2 <- X[,2]; X3 <- X[,3]  # save X1, X2, X3 from X, the matrix
```

Set up simulation for Q1

Using a for-loop to store data from models of SLR and MLR.

```
for(i in 1:nsim){
  Y <- Xmat%*%bet + rnorm(nsample, 0, sqrt(sig2))  # Y generated by all 3 predictors
  model1 <- lm(Y ~ X1)  # SLR with X1
  b0_1[i] <- coef(model1)[1]  # store regression parameter estimates of beta_0 of SLR with X1
  b1[i] <- coef(model1)[2]  # store regression parameter estimates of beta_1
  sig2hat1[i] <- summary(model1)$sigma^2  # store error var estimates with X1
  var_b0_1[i] <- summary(model1)$coefficient[1,2]^2  # store variance of b0 of SLR with X1
  var_b1[i] <- summary(model1)$coefficient[2,2]^2  # store variance of b1 of SLR with X1

  model2 <- lm(Y ~ X2)  # SLR with X2
  b0_2[i] <- coef(model2)[1]  # store regression parameter estimates of beta_0 of SLR with X2
  b2[i] <- coef(model2)[2]  # store regression parameter estimates of beta_2
  sig2hat2[i] <- summary(model2)$sigma^2  # store error var estimates of SLR with X2
  var_b0_2[i] <- summary(model2)$coefficient[1,2]^2  # store variance of b0 of SLR with X2
  var_b2[i] <- summary(model2)$coefficient[2,2]^2  # store variance of b2 of SLR with X2

  model3 <- lm(Y ~ X3)  # SLR with X3
  b0_3[i] <- coef(model3)[1]  # store regression parameter estimates of beta_0 of SLR with X3
  b3[i] <- coef(model3)[2]  # store regression parameter estimates of beta_3
  sig2hat3[i] <- summary(model3)$sigma^2  # store error var estimates of SLR with X3
  var_b0_3[i] <- summary(model3)$coefficient[1,2]^2  # store variance of b0 of SLR with X3
  var_b3[i] <- summary(model3)$coefficient[2,2]^2  # store variance of b2 of SLR with X3

  model <- lm(Y ~ X)  # MLR with X1, X2, X3
  bets[i,] <- coef(model)  # assign regression parameter estimates for MLR model with X1-3
  sig2hat[i] <- summary(model)$sigma^2  # store error var estimates under MLR
  vars[i, 1] <- summary(model)$coefficient[1,2]^2  # store variance of b0
  vars[i, 2] <- summary(model)$coefficient[2,2]^2  # store variance of b1
  vars[i, 3] <- summary(model)$coefficient[3,2]^2  # store variance of b2
```

The mean and variance of the estimators for SLR with X_1

Let's have a look at the mean of estimators b_0, b_1, s^2 from the simple linear regression model with predictor X_1.

```
mean(b0_1); mean(b1); mean(sig2hat1)  # The mean of esitmators b0, b1, and error variance s^2 for lm(Y ~ X1)

## [1] -0.6188704
## [1] -0.05106455
## [1] 2.647152

beta0; beta1; sig2  # The true population parameters beta_0, beta_1, and error variance sigma^2

## [1] -0.3627086
## [1] 0.3954798
## [1] 0.7116378

# Variance of reg parameter estimators for lm(Y ~ X1), SLR of X1 predictor
var_beta_0_1 <- var_true(X1, sig2, nsample, 0); var_beta_1 <- var_true(X1, sig2, nsample, 1)
var_beta_0_1; var_beta_1  # true variance of b0 with X1 as the predictor and b1

## [1] 0.07322619
## [1] 0.05498306

mean(var_b0_1); mean(var_b1)  # the mean of the sample variances of b0 and b1

## [1] 0.2723869
## [1] 0.2045261
```

The mean and variance of the estimators for SLR with X_2

Let's have a look at the mean of estimators b_0, b_2, s^2 from the simple linear regression model with predictor X_2.

```
mean(b0_2); mean(b2); mean(sig2hat2)  # The mean of esitmators b0, b2, and error variance s^2 for lm(Y ~ X2)

## [1] -0.4228588
## [1] 0.5849439
## [1] 0.9321075

beta0; beta2; sig2  # The true population parameters beta_0, beta_2, and error variance sigma^2

## [1] -0.3627086
## [1] 0.6617895
## [1] 0.7116378

# Variance of reg parameter estimators for lm(Y ~ X2), SLR of X2 predictor
var_beta_0_2 <- var_true(X2, sig2, nsample, 0); var_beta_2 <- var_true(X2, sig2, nsample, 1)
var_beta_0_1; var_beta_2  # true variance of b0 with X2 as the predictor and b2

## [1] 0.07322619
## [1] 0.01775078

mean(var_b0_1); mean(var_b2)  # the mean of the sample variances of b0 with X2 as the predictor and b2

## [1] 0.2723869
## [1] 0.02325008
```

The mean and variance of the estimators for SLR with X_3

And the mean of estimators b_0, b_3, s^2 from the simple linear regression model with predictor X_3.

```
mean(b0_3); mean(b3); mean(sig2hat3)  # The mean of esitmators b0, b2, and error variance s^2 for lm(Y ~ X3)

## [1] -0.6199692
## [1] 0.0220115
## [1] 2.651455

beta0; beta3; sig2  # The true population parameters beta_0, beta_3, and error variance sigma^2

## [1] -0.3627086
## [1] 0
## [1] 0.7116378

# Variance of reg parameter estimators for lm(Y ~ X3), SLR of X3 predictor
var_beta_0_3 <- var_true(X3, sig2, nsample, 0); var_beta_3 <- var_true(X3, sig2, nsample, 1)
var_beta_0_3; var_beta_3  # true variance of b0 with X3 as the predictor and b3

## [1] 0.08787135
## [1] 0.06703791

mean(var_b0_3); mean(var_b3)  # the mean of the sample variances of b0 with X3 as the predictor and b3

## [1] 0.3273954
## [1] 0.2497731
```

The mean of the estimators for MLR with X_1, X_2, X_3

```
# The mean of esitmators b0, b1, b2, b3, and s^2 for lm(Y ~ X), MLR with 3 predictors
mean(bets[,1]); mean(bets[,2]); mean(bets[,3]); mean(bets[,4]); mean(sig2hat)
```

```
## [1] -0.3221902
## [1] 0.385131
## [1] 0.6662959
## [1] -0.0003872324
## [1] 0.7486474
```

```
mean(b0_1); mean(b0_2); mean(b0_3)  # The mean of estimators b0 for SLR with each predictor
```

```
## [1] -0.6188704
## [1] -0.4228588
## [1] -0.6199692
```

```
mean(b1); mean(b2); mean(b3)  # The mean of estimators b1, b2, b3 for SLR with each predictor
```

```
## [1] -0.05106455
## [1] 0.5849439
## [1] 0.0220115
```

```
beta0; beta1; beta2; beta3; sig2 # The true population paramesters
```

```
## [1] -0.3627086
## [1] 0.3954798
## [1] 0.6617895
## [1] 0
## [1] 0.7116378
```

The variance of the estimators for MLR with X_1, X_2, X_3 as predictors

Let's have a look at the variance of estimators b_0, b_1, b_2, b_3, s^2 from the multiple linear regression model with predictor X_1, X_2, X_3.

```
# MLR variance of esitmators
XT <- t(Xmat)  # Transpose of Xmat
XTX <- XT%*%Xmat  # X'X
XTX_inverse <- solve(XTX)  #(X'X)^-1 = C
# var(bj) = sig2*C_{jj}
var_beta_0 <- sig2*XTX_inverse[1,1]; var_beta_1 <- sig2*XTX_inverse[2,2]
var_beta_2 <- sig2*XTX_inverse[3,3]; var_beta_3 <- sig2*XTX_inverse[4,4]

var_beta_0; var_beta_1; var_beta_2; var_beta_3  # true variance of b0, b1, b2, b3


## [1] 0.1030058
## [1] 0.07347805
## [1] 0.02063055
## [1] 0.07807018


mean(vars[,1]); mean(vars[,2]); mean(vars[,3]); mean(vars[,4])  # the mean of the sample variances of b0, b1, b


## [1] 0.1083627
## [1] 0.07729936
## [1] 0.02170347
## [1] 0.08213031
```

Run 1000 (n.sim) simulations

Set up to run a 1000 (n.sim) simulations

```r
set.seed(1004079631)
sig2hat_sim <- vector()  # MLR: save sample estimators for sig2
bets_sim <- matrix(NA, ncol = length(bet), nrow = 1000)  # MLR: save sample estimators of beta0, beta1, beta2 a
vars_sim <- matrix(NA, ncol = length(bet), nrow = 1000)  # MLR: save sample variance of b0, b1, b2, b3

for(i in 1:1000){
  Y <- Xmat%*%bet + rnorm(nsample, 0, sqrt(sig2))  # Y generated by all 3 predictors

  model_sim <- lm(Y ~ X)  # MLR with X1, X2, X3
  bets_sim[i,] <- coef(model_sim)  # assign regression parameter estimates for MLR model with X1-3
  sig2hat_sim[i] <- summary(model_sim)$sigma^2  # store error var estimates under MLR
  vars_sim[i, 1] <- summary(model_sim)$coefficient[1,2]^2  # store variance of b0
  vars_sim[i, 2] <- summary(model_sim)$coefficient[2,2]^2  # store variance of b1
  vars_sim[i, 3] <- summary(model_sim)$coefficient[3,2]^2  # store variance of b2
  vars_sim[i, 4] <- summary(model_sim)$coefficient[4,2]^2  # store variance of b3
}
```

Check whether the estimators are unbiased

Check whether the mean of estimators under 1000 (n.sim) simulations is closer to the population parameters for MLR compared to 100 simulations.

```
# The mean of esitmators b0, b1, b2, b3, and s^2 for lm(Y ~ X) under 100 n.sim
mean(bets[,1]); mean(bets[,2]); mean(bets[,3]); mean(bets[,4]); mean(sig2hat)

## [1] -0.3221902
## [1] 0.385131
## [1] 0.6662959
## [1] -0.0003872324
## [1] 0.7486474

# The mean of esitmators b0, b1, b2, b3, and s^2 for lm(Y ~ X) under 1000 n.sim
mean(bets_sim[,1]); mean(bets_sim[,2]); mean(bets_sim[,3]); mean(bets_sim[,4]); mean(sig2hat_sim)

## [1] -0.3701821
## [1] 0.3971863
## [1] 0.6579387
## [1] -0.00145451
## [1] 0.7149725

beta0; beta1; beta2; beta3; sig2 # The true population paramesters

## [1] -0.3627086
## [1] 0.3954798
## [1] 0.6617895
## [1] 0
## [1] 0.7116378
```

Check whether the estimators are unbiased

Calculate the matrix C

```
# MLR variance of esitmators
XT <- t(Xmat); XTX <- XT%*%Xmat; XTX_inverse <- solve(XTX)
# var(bj) = sig2*C_{jj}
var_beta_0_sim <- sig2*XTX_inverse[1,1]; var_beta_1_sim <- sig2*XTX_inverse[2,2]
var_beta_2_sim <- sig2*XTX_inverse[3,3]; var_beta_3_sim <- sig2*XTX_inverse[4,4]
```

Observe that the difference between true and mean of variance of the estimators *decreases* as the number of simulations *increases*

```
# The difference between true vs mean of variance of esitmators b0, b1, b2, b3 under 100 n.sim
var_beta_0-mean(vars[,1]); var_beta_1-mean(vars[,2]); var_beta_2-mean(vars[,3]); var_beta_3-mean(vars[,4])

## [1] -0.005356931
## [1] -0.003821309
## [1] -0.001072915
## [1] -0.004060128

# The difference between true vs mean of variance of the esitmators b0, b1, b2, b3 under 1000 n.sim
var_beta_0_sim-mean(vars_sim[,1]); var_beta_1_sim-mean(vars_sim[,2]); var_beta_2_sim-mean(vars_sim[,3]); var_be

## [1] -0.0004826674
## [1] -0.0003443055
## [1] -9.667124e-05
## [1] -0.0003658235
```

Set up for making X_1, X_2 correlated where $r_{12} = 0.2$

```
## Q3 ##
r12 <- 0.2  # Assume X1 and X2 are correlated. Set value for correlation r12 = 0.2, our small r12, also call it
sigmat[1,2] <- sigmat[2,1] <- r12*sqrt(sigmat[1,1])*sqrt(sigmat[2,2])
## Simulation for Categorical Variables with Interaction ##
set.seed(1004079631)
X <- mvrnorm(nsample, mu = muvec, Sigma = sigmat); cor(X[,1], X[,2])

## [1] 0.06534178

Xmat <- cbind(1, X)

# Repeat step 1 & 2, for small correlation, r12=0.2, which is named after _r1
# Not saving what b0, the intercept, is since it is not a focus for interaction among predictors
b1_r1 <- vector(); b2_r1 <- vector(); b3_r1 <- vector()  # saves the sample estimates of beta1-3 of SLR with X1
var_b1_r1 <- vector(); var_b2_r1 <- vector(); var_b3_r1 <- vector()   # save variance of b1, b2, b3 of SLR
# saves the sample estimates of error variance sigma^2 with SLR model on X1-3 predictor
sig2hat1_r1 <- vector(); sig2hat2_r1 <- vector(); sig2hat3_r1 <- vector()

sig2hat_r1 <- vector() # saves the sample estimates of error variance sigma^2 with MLR model on X1-3 predictor
bets_r1 <- matrix(NA, ncol = length(bet), nrow = nsim)  # MLR: save sample estimators of beta0, beta1, beta2 an
vars_r1 <- matrix(NA, ncol = length(bet), nrow = nsim)  # MLR: save sample variance of b0, b1, b2, b3
X1 <- X[,1]; X2 <- X[,2]; X3 <- X[,3]  # save X1, X2, X3 from X, the matrix

for(i in 1:nsim){
  Y <- Xmat%*%bet + rnorm(nsample, 0, sqrt(sig2))  # Y generated by all 3 predictors
  model1_r1 <- lm(Y ~ X1)  # SLR with X1
  b1_r1[i] <- coef(model1_r1)[2]  # store regression parameter estimates of beta_1
  sig2hat1_r1[i] <- summary(model1_r1)$sigma^2  # store error var estimates with X1
  var_b1_r1[i] <- summary(model1_r1)$coefficient[2,2]^2  # store variance of b1 of SLR with X1
```

Set up for making X_1, X_2 correlated where $r_{12} = 0.5$

```
## The correlation ##
r12 <- 0.5  # Assume X1 and X2 are correlated. Set value for correlation r12 = 0.5, our medium r12, also call i
sigmat[1,2] <- sigmat[2,1] <- r12*sqrt(sigmat[1,1])*sqrt(sigmat[2,2])
## Simulation for Categorical Variables with Interaction ##
set.seed(1004079631)
X <- mvrnorm(nsample, mu = muvec, Sigma = sigmat); cor(X[,1], X[,2])

## [1] 0.1263526

Xmat <- cbind(1, X)

# Repeat step 1 & 2, for medium correlation, r12=0.5, which is named after _r2
# Not saving what b0, the intercept, is since it is not a focus for interaction among predictors
b1_r2 <- vector(); b2_r2 <- vector(); b3_r2 <- vector()  # saves the sample estimates of beta1-3 of SLR with X1
var_b1_r2 <- vector(); var_b2_r2 <- vector(); var_b3_r2 <- vector()   # save variance of b1, b2, b3 of SLR
# saves the sample estimates of error variance sigma^2 with SLR model on X1-3 predictor
sig2hat1_r2 <- vector(); sig2hat2_r2 <- vector(); sig2hat3_r2 <- vector()

sig2hat_r2 <- vector() # saves the sample estimates of error variance sigma^2 with MLR model on X1-3 predictor
bets_r2 <- matrix(NA, ncol = length(bet), nrow = nsim)  # MLR: save sample estimators of beta0, beta1, beta2 an
vars_r2 <- matrix(NA, ncol = length(bet), nrow = nsim)  # MLR: save sample variance of b0, b1, b2, b3
X1 <- X[,1]; X2 <- X[,2]; X3 <- X[,3]  # save X1, X2, X3 from X, the matrix

for(i in 1:nsim){
  Y <- Xmat%*%bet + rnorm(nsample, 0, sqrt(sig2))  # Y generated by all 3 predictors
  model1_r2 <- lm(Y ~ X1)  # SLR with X1
  b1_r2[i] <- coef(model1_r2)[2]  # store regression parameter estimates of beta_1
  sig2hat1_r2[i] <- summary(model1_r2)$sigma^2  # store error var estimates with X1
  var_b1_r2[i] <- summary(model1_r2)$coefficient[2,2]^2  # store variance of b1 of SLR with X1
```

Set up for making X_1, X_2 correlated where $r_{12} = 0.95$

```
## The correlation ##
r12 <- 0.95  # Assume X1 and X2 are correlated. Set value for correlation r12 = 0.5, our high r12, also call it
sigmat[1,2] <- sigmat[2,1] <- r12*sqrt(sigmat[1,1])*sqrt(sigmat[2,2])
## Simulation for Categorical Variables with Interaction ##
set.seed(1004079631)
X <- mvrnorm(nsample, mu = muvec, Sigma = sigmat); cor(X[,1], X[,2])

## [1] 0.8126207

Xmat <- cbind(1, X)

# Repeat step 1 & 2, for large correlation, r12=0.5, which is named after _r3
# Not saving what b0, the intercept, is since it is not a focus for interaction among predictors
b1_r3 <- vector(); b2_r3 <- vector(); b3_r3 <- vector()  # saves the sample estimates of beta1-3 of SLR with X1
var_b1_r3 <- vector(); var_b2_r3 <- vector(); var_b3_r3 <- vector()   # save variance of b1, b2, b3 of SLR
# saves the sample estimates of error variance sigma^2 with SLR model on X1-3 predictor
sig2hat1_r3 <- vector(); sig2hat2_r3 <- vector(); sig2hat3_r3 <- vector()

sig2hat_r3 <- vector() # saves the sample estimates of error variance sigma^2 with MLR model on X1-3 predictor
bets_r3 <- matrix(NA, ncol = length(bet), nrow = nsim)  # MLR: save sample estimators of beta0, beta1, beta2 an
vars_r3 <- matrix(NA, ncol = length(bet), nrow = nsim)  # MLR: save sample variance of b0, b1, b2, b3
X1 <- X[,1]; X2 <- X[,2]; X3 <- X[,3]  # save X1, X2, X3 from X, the matrix

for(i in 1:nsim){
  Y <- Xmat%*%bet + rnorm(nsample, 0, sqrt(sig2))  # Y generated by all 3 predictors
  model1_r3 <- lm(Y ~ X1)  # SLR with X1
  b1_r3[i] <- coef(model1_r3)[2]  # store regression parameter estimates of beta_1
  sig2hat1_r3[i] <- summary(model1_r3)$sigma^2  # store error var estimates with X1
  var_b1_r3[i] <- summary(model1_r3)$coefficient[2,2]^2  # store variance of b1 of SLR with X1

  model2_r3 <- lm(Y ~ X2)  # SLR with X2
```

The estimators for SLR, $r_{12} = 0.2, 0.5, 0.95$

Let's have a look at what the mean of estimators are for SLR.

```
# For r12 = 0.95
mean(b1_r1); mean(b2_r1); mean(b3_r1)  # r12=0.2: The mean of estimators b1, b2, b3 for each SLR

## [1] 0.4163537
## [1] 0.6912161
## [1] -0.3203985

mean(b1_r2); mean(b2_r2); mean(b3_r2)  # r12=0.5: The mean of estimators b1, b2, b3 for each SLR

## [1] 0.4597044
## [1] 0.714557
## [1] -0.3090667

mean(b1_r3); mean(b2_r3); mean(b3_r3)  # r12=0.95: The mean of estimators b1, b2, b3 for each SLR

## [1] 1.210695
## [1] 0.867522
## [1] -0.3095506

beta1; beta2; beta3 # The true population parameters

## [1] 0.3954798
## [1] 0.6617895
## [1] 0
```

The estimators for SLR, $r_{12} = 0.2, 0.5, 0.95$

Let's have a look at what the variance of estimators are for SLR.

```
mean(var_b1_r1); mean(var_b2_r1); mean(var_b3_r1)  # r12=0.2: The mean of the sample variances of b1, b2, b3 fo

## [1] 0.08689663
## [1] 0.09422064
## [1] 0.04477469

mean(var_b1_r2); mean(var_b2_r2); mean(var_b3_r2)  # r12=0.5: The mean of the sample variances of b1, b2, b3 fo

## [1] 0.1142057
## [1] 0.08578694
## [1] 0.04550887

mean(var_b1_r3); mean(var_b2_r3); mean(var_b3_r3)  # r12=0.95: The mean of the sample variances of b1, b2, b3 f

## [1] 0.2116782
## [1] 0.07136186
## [1] 0.04991448
```

The estimators for MLR, $r_{12} = 0.2, 0.5, 0.95$

Let's have a look at what the mean of estimators are for MLR.

```
# The mean of estimators b1, b2, b3 for MLR
mean(bets_r1[,2]); mean(bets_r1[,3]); mean(bets_r1[,4])  # r12=0.2: The mean of estimators b1, b2, b3 for MLR

## [1] 0.3640337
## [1] 0.6368773
## [1] -0.03337979

mean(bets_r2[,2]); mean(bets_r2[,3]); mean(bets_r2[,4])  # r12=0.5: The mean of estimators b1, b2, b3 for MLR

## [1] 0.3558938
## [1] 0.6456377
## [1] -0.03337979

mean(bets_r3[,2]); mean(bets_r3[,3]); mean(bets_r3[,4])  # r12=0.95: The mean of estimators b1, b2, b3 for MLR

## [1] 0.2985956
## [1] 0.6843456
## [1] -0.03337979

beta1; beta2; beta3  # The true population parameters

## [1] 0.3954798
## [1] 0.6617895
## [1] 0
```

The estimators for MLR, $r_{12} = 0.2, 0.5, 0.95$

Let's have a look at what the variance of estimators are for MLR.

```
mean(vars_r1[,2]); mean(vars_r1[,3]); mean(vars_r1[,4])  # r12=0.2: the mean of the sample variances of b1, b2,

## [1] 0.05739735
## [1] 0.09001812
## [1] 0.0392292

mean(vars_r2[,2]); mean(vars_r2[,3]); mean(vars_r2[,4])  # r12=0.5: the mean of the sample variances of b1, b2,

## [1] 0.08039977
## [1] 0.08125437
## [1] 0.0392292

mean(vars_r3[,2]); mean(vars_r3[,3]); mean(vars_r3[,4])  # r12=0.95: the mean of the sample variances of b1, b2

## [1] 0.5589607
## [1] 0.1994737
## [1] 0.0392292
```

Set up for making X_1, X_3 correlated where $r_{13} = 0.2$

```
## Q4 ##
sigmat[1,2] <- sigmat[2,1] <- 0  # Assume X1, X2 are uncorrelated, set to 0
r13 <- 0.2  # Assume X1 and X3 are correlated. Set value for correlation r13 = 0.2, our small r13, also call it
sigmat[1,3] <- sigmat[3,1] <- r13*sqrt(sigmat[1,1])*sqrt(sigmat[3,3])
## Simulation for Categorical Variables with Interaction ##
set.seed(1004079631)
X <- mvrnorm(nsample, mu = muvec, Sigma = sigmat); cor(X[,1], X[,3])

## [1] 0.4375513

Xmat <- cbind(1, X)

# Repeat step 1 & 2, for small correlation, r13=0.2, which is named after _r4
# Not saving what b0, the intercept, is since it is not a focus for interaction among predictors
b1_r4 <- vector(); b2_r4 <- vector(); b3_r4 <- vector()  # saves the sample estimates of beta1-3 of SLR with X1
var_b1_r4 <- vector(); var_b2_r4 <- vector(); var_b3_r4 <- vector()   # save variance of b1, b2, b3 of SLR
# saves the sample estimates of error variance sigma^2 with SLR model on X1-3 predictor
sig2hat1_r4 <- vector(); sig2hat2_r4 <- vector(); sig2hat3_r4 <- vector()

sig2hat_r4 <- vector() # saves the sample estimates of error variance sigma^2 with MLR model on X1-3 predictor
bets_r4 <- matrix(NA, ncol = length(bet), nrow = nsim)  # MLR: save sample estimators of beta0, beta1, beta2 an
vars_r4 <- matrix(NA, ncol = length(bet), nrow = nsim)  # MLR: save sample variance of b0, b1, b2, b3
X1 <- X[,1]; X2 <- X[,2]; X3 <- X[,3]  # save X1, X2, X3 from X, the matrix

for(i in 1:nsim){
  Y <- Xmat%*%bet + rnorm(nsample, 0, sqrt(sig2))  # Y generated by all 3 predictors
  model1_r4 <- lm(Y ~ X1)  # SLR with X1
  b1_r4[i] <- coef(model1_r4)[2]  # store regression parameter estimates of beta_1
  sig2hat1_r4[i] <- summary(model1_r4)$sigma^2  # store error var estimates with X1
  var_b1_r4[i] <- summary(model1_r4)$coefficient[2,2]^2  # store variance of b1 of SLR with X1
```

Set up for making X_1, X_3 correlated where $r_{13} = 0.5$

```
## The correlation ##
sigmat[1,2] <- sigmat[2,1] <- 0  # Assume X1, X2 are uncorrelated, set to 0
r13 <- 0.5  # Assume X1 and X3 are correlated. Set value for correlation r13 = 0.5, our medium r13, also call i
sigmat[1,3] <- sigmat[3,1] <- r13*sqrt(sigmat[1,1])*sqrt(sigmat[3,3])
## Simulation for Categorical Variables with Interaction ##
set.seed(1004079631)
X <- mvrnorm(nsample, mu = muvec, Sigma = sigmat); cor(X[,1], X[,3])

## [1] 0.6145322

Xmat <- cbind(1, X)

# Repeat step 1 & 2, for medium correlation, r13=0.5, which is named after _r5
# Not saving what b0, the intercept, is since it is not a focus for interaction among predictors
b1_r5 <- vector(); b2_r5 <- vector(); b3_r5 <- vector()  # saves the sample estimates of beta1-3 of SLR with X1
var_b1_r5 <- vector(); var_b2_r5 <- vector(); var_b3_r5 <- vector()   # save variance of b1, b2, b3 of SLR
# saves the sample estimates of error variance sigma^2 with SLR model on X1-3 predictor
sig2hat1_r5 <- vector(); sig2hat2_r5 <- vector(); sig2hat3_r5 <- vector()

sig2hat_r5 <- vector() # saves the sample estimates of error variance sigma^2 with MLR model on X1-3 predictor
bets_r5 <- matrix(NA, ncol = length(bet), nrow = nsim)  # MLR: save sample estimators of beta0, beta1, beta2 an
vars_r5 <- matrix(NA, ncol = length(bet), nrow = nsim)  # MLR: save sample variance of b0, b1, b2, b3
X1 <- X[,1]; X2 <- X[,2]; X3 <- X[,3]  # save X1, X2, X3 from X, the matrix

for(i in 1:nsim){
  Y <- Xmat%*%bet + rnorm(nsample, 0, sqrt(sig2))  # Y generated by all 3 predictors
  model1_r5 <- lm(Y ~ X1)  # SLR with X1
  b1_r5[i] <- coef(model1_r5)[2]  # store regression parameter estimates of beta_1
  sig2hat1_r5[i] <- summary(model1_r5)$sigma^2  # store error var estimates with X1
  var_b1_r5[i] <- summary(model1_r5)$coefficient[2,2]^2  # store variance of b1 of SLR with X1
```

Set up for making X_1, X_3 correlated where $r_{13} = 0.95$

```
## The correlation ##
sigmat[1,2] <- sigmat[2,1] <- 0  # Assume X1, X2 are uncorrelated, set to 0
r13 <- 0.95  # Assume X1 and X3 are correlated. Set value for correlation r13 = 0.95, our large r13, also call
sigmat[1,3] <- sigmat[3,1] <- r13*sqrt(sigmat[1,1])*sqrt(sigmat[3,3])
## Simulation for Categorical Variables with Interaction ##
set.seed(1004079631)
X <- mvrnorm(nsample, mu = muvec, Sigma = sigmat); cor(X[,1], X[,3])
```

```
## [1] 0.9572645
```

```
Xmat <- cbind(1, X)

# Repeate step 1 & 2, for large correlation, r13=0.95, which is named after _r6
# Not saving what b0, the intercept, is since it is not a focus for interaction among predictors
b1_r6 <- vector(); b2_r6 <- vector(); b3_r6 <- vector()  # saves the sample estimates of beta1-3 of SLR with X1
var_b1_r6 <- vector(); var_b2_r6 <- vector(); var_b3_r6 <- vector()   # save variance of b1, b2, b3 of SLR
# saves the sample estimates of error variance sigma^2 with SLR model on X1-3 predictor
sig2hat1_r6 <- vector(); sig2hat2_r6 <- vector(); sig2hat3_r6 <- vector()

sig2hat_r6 <- vector() # saves the sample estimates of error variance sigma^2 with MLR model on X1-3 predictor
bets_r6 <- matrix(NA, ncol = length(bet), nrow = nsim)  # MLR: save sample estimators of beta0, beta1, beta2 an
vars_r6 <- matrix(NA, ncol = length(bet), nrow = nsim)  # MLR: save sample variance of b0, b1, b2, b3
X1 <- X[,1]; X2 <- X[,2]; X3 <- X[,3]  # save X1, X2, X3 from X, the matrix

for(i in 1:nsim){
  Y <- Xmat%*%bet + rnorm(nsample, 0, sqrt(sig2))  # Y generated by all 3 predictors
  model1_r6 <- lm(Y ~ X1)  # SLR with X1
  b1_r6[i] <- coef(model1_r6)[2]  # store regression parameter estimates of beta_1
  sig2hat1_r6[i] <- summary(model1_r6)$sigma^2  # store error var estimates with X1
  var_b1_r6[i] <- summary(model1_r6)$coefficient[2,2]^2  # store variance of b1 of SLR with X1
```

The estimators for SLR, $r_{13} = 0.2, 0.5, 0.95$

Let's have a look at what the mean of estimators are for SLR.

```
mean(b1_r4); mean(b2_r4); mean(b3_r4)  # r13=0.2: The mean of estimators b1, b2, b3 for each SLR

## [1] 0.248157
## [1] 0.5168615
## [1] -0.06139558

mean(b1_r5); mean(b2_r5); mean(b3_r5)  # r13=0.5: The mean of estimators b1, b2, b3 for each SLR

## [1] 0.202194
## [1] 0.4450114
## [1] 0.01201254

mean(b1_r6); mean(b2_r6); mean(b3_r6)  # r13=0.95: The mean of estimators b1, b2, b3 for each SLR

## [1] 0.143383
## [1] 0.4066709
## [1] 0.0990949

beta1; beta2; beta3 # The true population parameters

## [1] 0.3954798
## [1] 0.6617895
## [1] 0
```

The estimators for SLR, $r_{13} = 0.2, 0.5, 0.95$

Let's have a look at what the variance of estimators are for SLR.

```
mean(var_b1_r4); mean(var_b2_r4); mean(var_b3_r4)  # r13=0.2: The mean of the sample variances of b1, b2, b3 fo

## [1] 0.06693242
## [1] 0.0987603
## [1] 0.05683156

mean(var_b1_r5); mean(var_b2_r5); mean(var_b3_r5)  # r13=0.5: The mean of the sample variances of b1, b2, b3 fo

## [1] 0.06051033
## [1] 0.09954239
## [1] 0.05685716

mean(var_b1_r6); mean(var_b2_r6); mean(var_b3_r6)  # r13=0.95: The mean of the sample variances of b1, b2, b3 f

## [1] 0.06798656
## [1] 0.09415776
## [1] 0.04555494
```

The estimators for MLR, $r_{13} = 0.2, 0.5, 0.95$

Let's have a look at what the mean of estimators are for MLR.

```
# The mean of estimators b1, b2, b3 for MLR
mean(bets_r4[,2]); mean(bets_r4[,3]); mean(bets_r4[,4])  # r13=0.2: The mean of estimators b1, b2, b3 for MLR

## [1] 0.378442
## [1] 0.6915293
## [1] 0.03919632

mean(bets_r5[,2]); mean(bets_r5[,3]); mean(bets_r5[,4])  # r13=0.5: The mean of estimators b1, b2, b3 for MLR

## [1] 0.3811229
## [1] 0.6915293
## [1] 0.04290887

mean(bets_r6[,2]); mean(bets_r6[,3]); mean(bets_r6[,4])  # r13=0.95:

## [1] 0.3273616
## [1] 0.6915293
## [1] 0.08330634

beta1; beta2; beta3  # The true population parameters

## [1] 0.3954798
## [1] 0.6617895
## [1] 0
```

The estimators for MLR, $r_{13} = 0.2, 0.5, 0.95$

Let's have a look at what the variance of estimators are for MLR.

```
mean(vars_r4[,2]); mean(vars_r4[,3]); mean(vars_r4[,4])  # r13=0.2: the mean of the sample variances of b1, b2,

## [1] 0.04755201
## [1] 0.09392865
## [1] 0.04656558

mean(vars_r5[,2]); mean(vars_r5[,3]); mean(vars_r5[,4])  # r13=0.5: the mean of the sample variances of b1, b2,

## [1] 0.05834752
## [1] 0.09392865
## [1] 0.05787425

mean(vars_r6[,2]); mean(vars_r6[,3]); mean(vars_r6[,4])  # r13=0.95: the mean of the sample variances of b1, b2

## [1] 0.4951749
## [1] 0.09392865
## [1] 0.3379382
```

Set up a new data-set with a non-linear relationship

Generate 500 random values,
$Y = 4[\sin(\pi x_1 x_2) + 8(x_3 - 0.5)^3 + 1.5x_4 - x_5 - 0.77] + \epsilon$, where
$\epsilon \sim N(0, 1)$ and $X_i \sim \text{Uniform}[0; 1]$ for $i \in \{1, 2, 3, 4, 5\}$

```r
## Task2 ##
# This following function provides a data set with p+1 columns #
gendata <- function(n, p){
  Xmat <- matrix(runif(n*p, 0, 1), nrow = n, ncol = p)
  Y <- 4*( (sin(pi*Xmat[,1]*Xmat[,2])) + 8*(Xmat[,3] - 0.5)^3 +
             1.5*Xmat[,4] - Xmat[,5] - 0.77 ) + rnorm(n, 0, 1)
  dat <- cbind(Xmat, Y)
  return(dat)
}

set.seed(1004079631)
dat <- as.data.frame(gendata(500, 5))
colnames(dat) <- c(paste0("X", 1:5), "Y")

## Q1 ##
# MLR model
model <- lm(dat$Y ~ ., data = dat)  # p=5, n=500
Y_hat <- predict(model)  # the fitted values by the MLR model, Y_hat
```

Check the p-values and R^2_{adjusted}

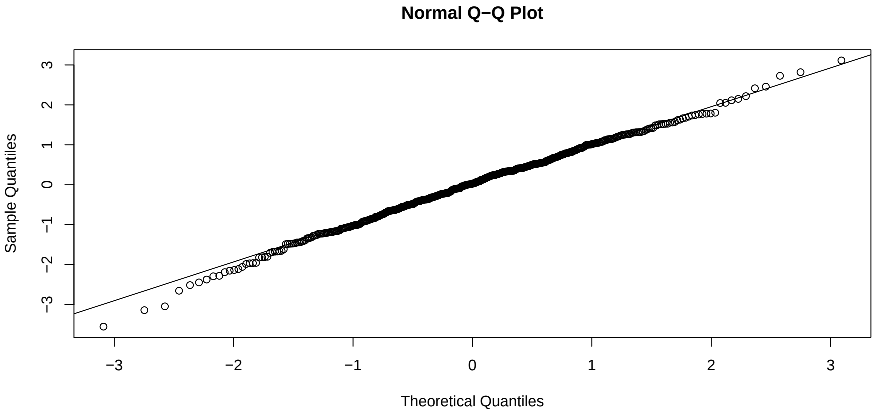
```
summary(model)  # Check p-value, R^2 adjusted

##
## Call:
## lm(formula = dat$Y ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6633 -0.8576  0.0421  0.8871  4.1113
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -6.3194     0.2367  -26.70   <2e-16 ***
## X1            2.7409     0.2041   13.43   <2e-16 ***
## X2            2.9066     0.2043   14.23   <2e-16 ***
## X3            5.0161     0.2064   24.30   <2e-16 ***
## X4            6.0866     0.2085   29.20   <2e-16 ***
## X5           -4.0267     0.2082  -19.34   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.34 on 494 degrees of freedom
## Multiple R-squared:  0.8174,Adjusted R-squared:  0.8155
## F-statistic: 442.2 on 5 and 494 DF,  p-value: < 2.2e-16
```

Check normality

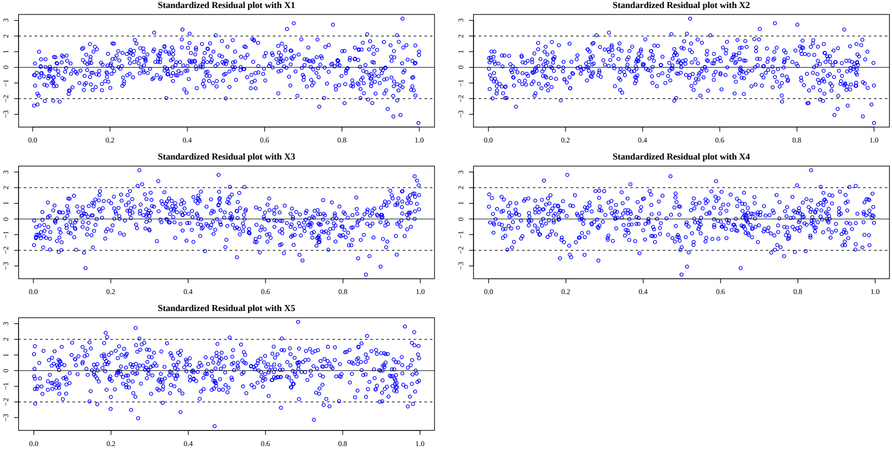
The following is the Normal Q-Q plot:

```
# Normal Q-Q plot
r <- rstudent(model)  # standardized residual for the model
qqnorm(r); qqline(r)
```



Check Residual Plots

The followings are standardized residual plots against each predictor $X_1, X_2, ..., X_5$.



Standardized Residual plot with X1

Standardized Residual plot with X2

Standardized Residual plot with X3

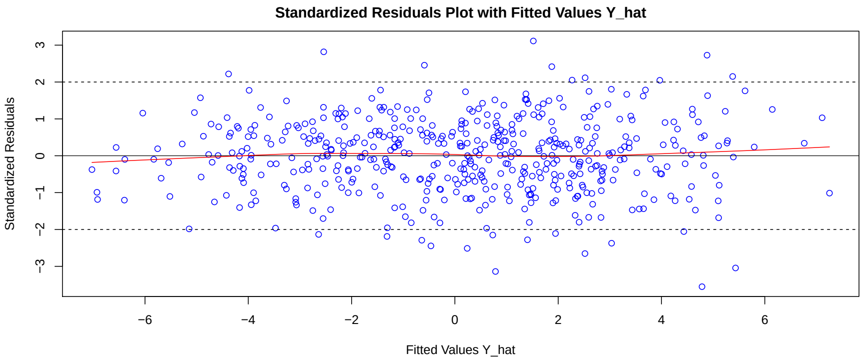
Standardized Residual plot with X4

Standardized Residual plot with X5

Check Residual Plots

Here we have a standardized residual plot against fitted values $\hat{Y}$

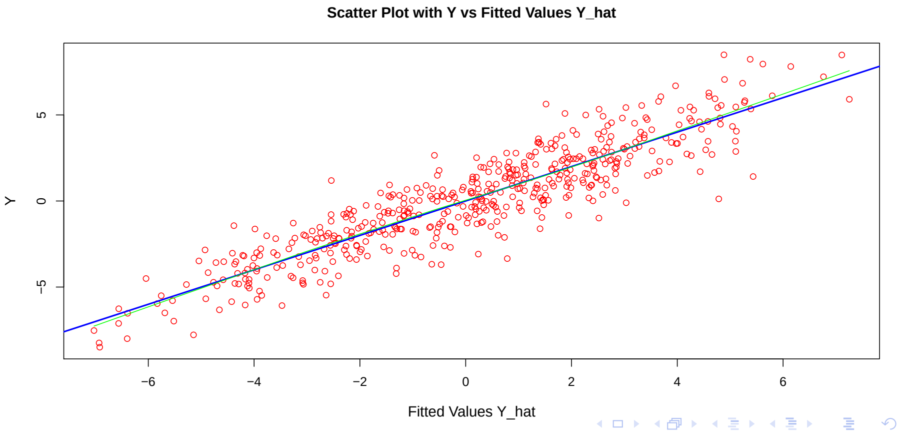
```
par(mfrow = c(1,1), mai = c(1, 1, 0.5, 0.5))  # set up margin for important plots
# Standardized Residual plot for fitted y-values, Y_hat
plot(Y_hat, r, type = "p", xlab = "Fitted Values Y_hat", ylab = "Standardized Residuals", main = "Standardized
abline(h = 2, lty = 2); abline(h = -2, lty = 2); abline(h = 0, lwd = 0.8); lines(lowess(Y_hat, r), col="red")
```



Check Scatter Plot

And the scatter plot of responses Y vs. fitted values $\hat{Y}$.

```r
# response vs fitted values Y_hat
plot(dat$Y ~ Y_hat, type="p", xlab="Fitted Values Y_hat", ylab="Y", main = "Scatter Plot with Y vs Fitted Value
abline(lm(dat$Y ~ Y_hat), lwd=2, col="blue"); lines(lowess(Y_hat, dat$Y), col="green")
```



Check Influential points

Check Hat Values, Cook's Distance, DFFITS, and DFBETAS.

```
# hat values
h <- hatvalues(model); thresh <- 2 * 6/500; which(h > thresh)

## 344 354
## 344 354

# Cook's distance
D <- cooks.distance(model); which(D > qf(0.5, 6, 500-5-1))  # 50th percentile of F with p+1 and n-p-1 df

## named integer(0)

# DFFITS
dfits <- dffits(model); which(abs(dfits) > 2*sqrt(6/500))  # 2*sqrt((p+1)/n)

##  34  55  93  95 106 128 129 146 147 152 171 200 206 223 246 269 283 292 314 346
##  34  55  93  95 106 128 129 146 147 152 171 200 206 223 246 269 283 292 314 346
## 354 381 385 399 404 413 425 437 459 465 477 496
## 354 381 385 399 404 413 425 437 459 465 477 496

# DFBETAS
dfb <- dfbetas(model); which(abs(dfb[,1]) > 2/sqrt(500))  # 2*sqrt(1/n)

##   9  26  55  88  93 105 152 167 171 201 223 232 246 271 283 292 304 313 314 322
##   9  26  55  88  93 105 152 167 171 201 223 232 246 271 283 292 304 313 314 322
## 334 339 341 344 346 347 354 371 381 404 413 425 441 477
## 334 339 341 344 346 347 354 371 381 404 413 425 441 477
```

Prediction Error before transformation

We generate 200 new data to calculate the prediction error of this model, $lm(Y \sim X_1 + X_2 + X_3 + X_4 + X_5)$.

```r
## Q2 ##
dat.new <- as.data.frame(gendata(200, 5)); colnames(dat.new) <- c(paste0("X", 1:5), "Y")
pred.y <- predict(model, newdata = dat.new, type = "response", interval = "none")

PE <- mean((dat.new$Y - pred.y)^2); PE  # Prediction error
```

```
## [1] 1.740713
```

Box-Cov Transformation

Use powerTransform to see if we can apply Box-Cov, assuming we do not know the true relationship.

```
## Q3 ##
library(car) # Box_Cov

## Loading required package:  carData

mult <- lm(cbind(dat$Y, dat$X1, dat$X2, dat$X3, dat$X4, dat$X5) ~ 1); bc <- powerTransform(mult)

## Error in bc1(out[, j], lambda[j]):  First argument must be strictly positive.

summary(bc)

## Error in object[[i]]:  object of type 'closure' is not subsettable
```

Since Y is not strictly positive, we cannot use Box-Cov. Let's perform the true transformation, where $Z_1 = \sin(\pi x_1 x_2), Z_2 = (x_3 - 0.5)^3, Z_3 = x_4, Z_4 = x_5$ such that $Y = 4Z_1 + 32Z_2 + 6Z_3 - 4Z_4 - 4 \cdot 0.77 + \epsilon$.

```
# True Transformation: Know Y = 4sin(pi*x1*x2) + 32(x3-0.5)^3 + 6*x4 - 4*x5 - 0.77*4 + epsilon
Z1 <- sin(pi*dat$X1*dat$X2); Z2 <- (dat$X3-0.5)**3; Z3 <- dat$X4; Z4 <- dat$X5   # X4, X5 no change
# Then Y = 4*Z1 + 32*Z2 + 6*Z3 - 4*Z4 - 0.77*4 + epsilon, a linear relation with predictor variables Z1-4
model_new <- lm(dat$Y ~ Z1 + Z2 + Z3 + Z4)  # p=4, n=500
Y_hat_new <- predict(model_new)  # the fitted values by the MLR model, Y_hat
```

Check the p-values and R^2_{adjusted} for the new model

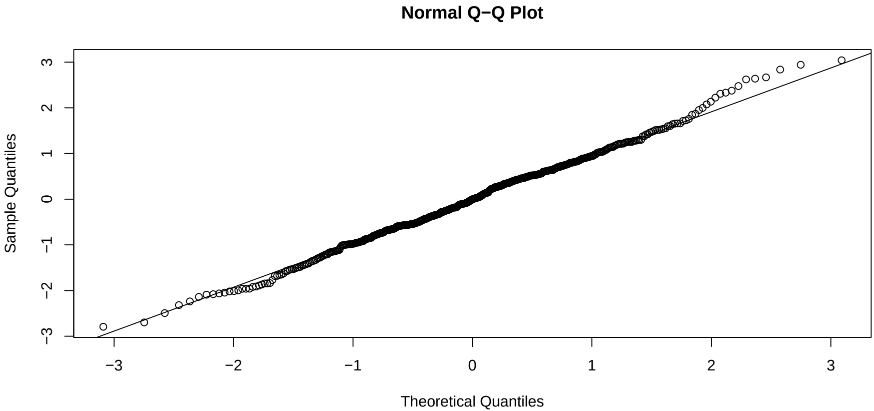
```
summary(model_new); # Check p-value, R^2 adjusted

##
## Call:
## lm(formula = dat$Y ~ Z1 + Z2 + Z3 + Z4)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.60191 -0.61343 -0.00397  0.60283  2.82829
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -3.0591     0.1280  -23.90   <2e-16 ***
## Z1            4.0562     0.1240   32.71   <2e-16 ***
## Z2           33.2675     0.8681   38.32   <2e-16 ***
## Z3            6.0227     0.1469   41.01   <2e-16 ***
## Z4           -4.0181     0.1466  -27.40   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9443 on 495 degrees of freedom
## Multiple R-squared:  0.9091,Adjusted R-squared:  0.9084
## F-statistic:  1238 on 4 and 495 DF,  p-value: < 2.2e-16
```

Check normality for the new model

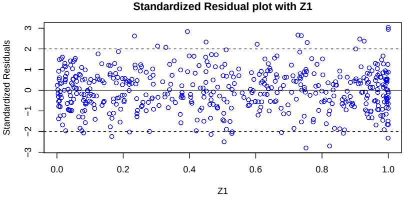
The following is the Normal Q-Q plot after transformation:

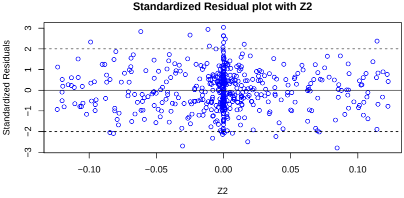
```
# Normal Q-Q plot
r_new <- rstudent(model_new)  # standardized residual for the model
qqnorm(r_new); qqline(r_new)
```

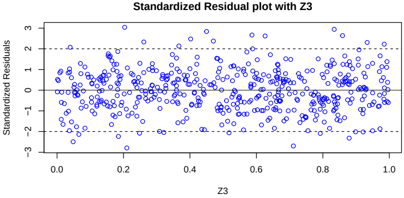


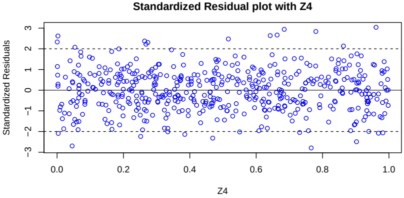
Check Residual Plots for the new model

The followings are standardized residual plots against each predictor Z_1, Z_2, Z_3, Z_4.


Standardized Residual plot with Z1


Standardized Residual plot with Z2

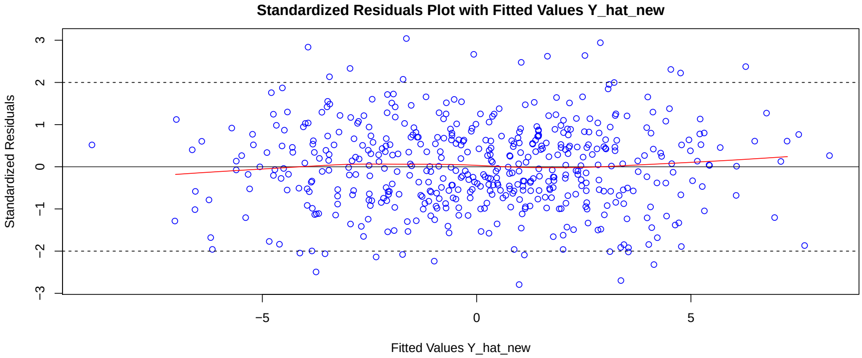

Standardized Residual plot with Z3


Standardized Residual plot with Z4

Check Residual Plots for the new model

Here we have a standardized residual plot against fitted values $\hat{Y}_{new}$ for the new model with predictors Z's.

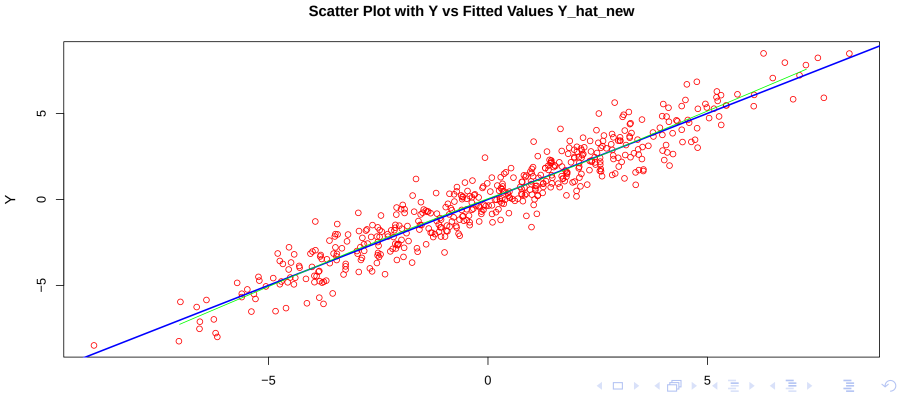
```r
par(mfrow = c(1,1), mai = c(1, 1, 0.5, 0.5))  # set up margin for important plots
# Standardized Residual plot for fitted y-values, Y_hat_new
plot(Y_hat_new, r_new, type = "p", xlab = "Fitted Values Y_hat_new", ylab = "Standardized Residuals", main = "S
abline(h = 2, lty = 2); abline(h = -2, lty = 2); abline(h = 0, lwd = 0.8); lines(lowess(Y_hat, r), col="red")
```



Check Scatter Plot for the new model

And the scatter plot of responses Y vs. fitted values $\hat{Y}_{new}$ for the new model with predictors Z's.

```
# response vs fitted values Y_hat
plot(dat$Y ~ Y_hat_new, type="p", xlab="Fitted Values Y_hat_new", ylab="Y", main = "Scatter Plot with Y vs Fitt
abline(lm(dat$Y ~ Y_hat), lwd=2, col="blue"); lines(lowess(Y_hat, dat$Y), col="green")
```



Check Influential Points for the new model

Check Hat Values, Cook's Distance, DFFITS, and DFBETAS for the new model.

```
# hat values
h <- hatvalues(model_new); thresh <- 2 * 5/500; which(h > thresh)

##  16  34  93 114 115 169 201 271 304 311 332 340 344 354 419 441 459 490 493
##  16  34  93 114 115 169 201 271 304 311 332 340 344 354 419 441 459 490 493

# Cook's distance
D <- cooks.distance(model_new); which(D > qf(0.5, 5, 500-4-1))  # 50th percentile of F with p+1 and n-p-1 df

## named integer(0)

# DFFITS
dfits <- dffits(model_new); which(abs(dfits) > 2*sqrt(5/500))  # 2*sqrt((p+1)/n)

##  26  60  95 114 116 123 129 137 170 171 175 200 202 206 251 291 292 308 314 325
##  26  60  95 114 116 123 129 137 170 171 175 200 202 206 251 291 292 308 314 325
## 334 354 381 394 422 425 426 431 435 444 496
## 334 354 381 394 422 425 426 431 435 444 496

# DFBETAS
dfb <- dfbetas(model_new); which(abs(dfb[,1]) > 2/sqrt(500))  # 2*sqrt(1/n)

##  10  26  44  56  70  72 104 105 114 123 148 171 251 262 283 298 308 314 322 324
##  10  26  44  56  70  72 104 105 114 123 148 171 251 262 283 298 308 314 322 324
## 325 333 334 344 347 354 394 409 429 435 497 500
```

Prediction Error before transformation

We generate 200 new data to calculate the prediction error of this new model, $lm(Y \sim Z_1 + Z_2 + Z_3 + Z_4)$.

```
dat.new_bf_transf <- as.data.frame(gendata(200, 5)); colnames(dat.new_bf_transf) <- c(paste0("X", 1:5), "Y")
dat.new_af_transf <- as.data.frame(cbind(sin(pi*dat.new_bf_transf$X1*dat.new_bf_transf$X2), (dat.new_bf_transf$
colnames(dat.new_af_transf) <- c(paste0("Z", 1:4), "Y")
pred.y_new <- predict(model_new, newdata = dat.new_af_transf, type = "response", interval = "none")

PE_new <- mean((dat.new_af_transf$Y - pred.y_new)^2); PE_new  # New Prediction error after transformation
```

```
## [1] 1.248111
```

```
PE # Prediction error before transformation
```

```
## [1] 1.740713
```

Clearly, the prediction error decreased for the transformed model.

Key Takeaways

From this project, we learn that...

1. Multiple linear regression models better than simple linear regression if the true population relationship is multi-linear.
2. As correlation increases in the predictors, the accuracy of estimators for MLR is affected negatively as the variance of estimators also increases. SLR model is independent of changes in correlation since it only has one predictor variable.
3. Moderate correlation among the predictors does not affect the model as much as strong correlation, and could be ignored for the interest of prediction.
4. Influential observations require multiple tests to determine whether they are outliers or not.
5. Prediction error decreases if one applies appropriate transformation to solve the problem of non-linearity in the model.

Thank You!

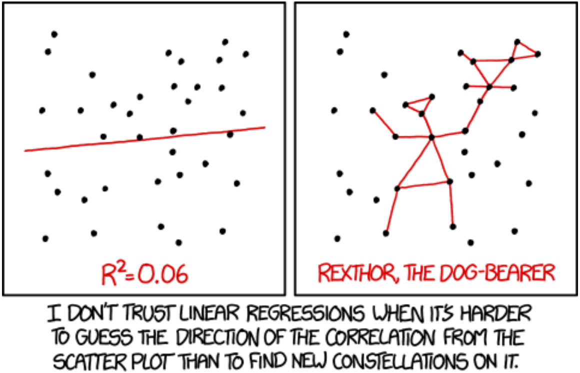

$R^2 = 0.06$
REXTHOR, THE DOG-BEARER
I DON'T TRUST LINEAR REGRESSIONS WHEN IT'S HARDER TO GUESS THE DIRECTION OF THE CORRELATION FROM THE SCATTER PLOT THAN TO FIND NEW CONSTELLATIONS ON IT.

Figure: Source: xkcd