We write an program on R. There is a conditions: Let eps, psi be two independent error variables such that eps∼N(0,4), psi∼N(0,9). Variables x and y are obtained from two errors using the following formulas: H = 160 a0 = 3 a1 = 1 a2= -6 b0 = 5 b1 = 7 b2 = -2 x = a0 + a1 \* eps + a2 \* psi y = b0 + b1 \* eps + b2 \* psi Find for x and y the mathematical expectation, variances, standard deviations.

The mathematical expectation of a random variable is the expected value of the variable, which is the average value of the variable over a large number of trials.

For the random variable x, the mathematical expectation is:

E[x] = E[a0 + a1 \* eps + a2 \* psi]

Since eps and psi are independent, we can use the linearity of expectation to find the expectation of the sum:

E[x] = E[a0] + E[a1 \* eps] + E[a2 \* psi]

Since eps and psi are both normally distributed with mean 0, their expectations are also 0:

E[x] = a0 + 0 + 0 = a0

Therefore, the mathematical expectation of x is 3.

The variance of a random variable is a measure of the spread or dispersion of the values of the variable. It is defined as the expected value of the squared deviation of the variable from its mean:

Var[x] = E[(x - E[x])^2]

Substituting the expression for x and the value of its expectation, we have:

Var[x] = E[(a0 + a1 \* eps + a2 \* psi - a0)^2]

Using the linearity of expectation again, we can find the variance of the sum:

Var[x] = E[a1^2 \* eps^2] + E[a2^2 \* psi^2] + 2 \* E[a1 \* a2 \* eps \* psi]

Since eps and psi are independent and normally distributed with variances 4 and 9, respectively, their variances can be calculated as follows:

Var[eps] = E[eps^2] - E[eps]^2 = 4 - 0^2 = 4 Var[psi] = E[psi^2] - E[psi]^2 = 9 - 0^2 = 9

Substituting these values into the expression for the variance of x, we have:

Var[x] = a1^2 \* Var[eps] + a2^2 \* Var[psi] + 2 \* a1 \* a2 \* E[eps] \* E[psi]

Since eps and psi are independent and have means of 0, their covariance is 0:

Cov[eps, psi] = E[eps \* psi] - E[eps] \* E[psi] = 0 - 0 \* 0 = 0

Substituting this value into the expression for the variance of x, we have:

Var[x] = a1^2 \* Var[eps] + a2^2 \* Var[psi] + 2 \* a1 \* a2 \* Cov[eps, psi]

Var[x] = 1^2 \* 4 + (-6)^2 \* 9 + 2 \* 1 \* -6 \* 0

Var[x] = 16 + 36 + 0

Var[x] = 52

Therefore, the variance of x is 52.

The mathematical expectation of x is 3.

The variance of x is 52.

The standard deviation of x is sqrt(52) = 7.2111.

The mathematical expectation of y is b0.

The variance of y is b1^2 \* Var[eps] + b2^2 \* Var[psi] + 2 \* b1 \* b2 \* Cov[eps, psi].

The standard deviation of y is sqrt(Var[y]).

Substituting the values for b0, b1, b2, Var[eps], and Var[psi], we have:

E[y] = 5 Var[y] = 7^2 \* 4 + (-2)^2 \* 9 + 2 \* 7 \* -2 \* 0 = 49 + 4 + 0 = 53 Std[y] = sqrt(53) = 7.2801

The covariance of two random variables x and y is a measure of the joint variability of the two variables. It is defined as the expected value of the product of the deviation of each variable from its mean:

Cov[x, y] = E[(x - E[x])(y - E[y])]

Substituting the expressions for x and y and the values of their expectations, we have:

Cov[x, y] = E[(a0 + a1 \* eps + a2 \* psi - a0)(b0 + b1 \* eps + b2 \* psi - b0)]

Using the linearity of expectation, we can find the covariance of the product:

Cov[x, y] = E[a1 \* b1 \* eps^2] + E[a2 \* b2 \* psi^2] + E[(a1 \* b2 + a2 \* b1) \* eps \* psi]

Since eps and psi are independent and normally distributed with variances 4 and 9, respectively, their variances can be calculated as follows:

Var[eps] = E[eps^2] - E[eps]^2 = 4 - 0^2 = 4 Var[psi] = E[psi^2] - E[psi]^2 = 9 - 0^2 = 9

Substituting these values into the expression for the covariance of x and y, we have:

Cov[x, y] = a1 \* b1 \* Var[eps] + a2 \* b2 \* Var[psi] + (a1 \* b2 + a2 \* b1) \* Cov[eps, psi]

Cov[x, y] = 1 \* 7 \* 4 + (-6) \* (-2) \* 9 + (1 \* (-2) + (-6) \* 7) \* 0

Cov[x, y] = 28 + 36 + 0

Cov[x, y] = 64

The correlation of two random variables x and y is a measure of the linear relationship between the two variables. It is defined as the covariance of the two variables divided by the product of their standard deviations:

Corr[x, y] = Cov[x, y] / (Std[x] \* Std[y])

Substituting the values for the covariance and standard deviations of x and y, we have:

Corr[x, y] = Cov[x, y] / (Std[x] \* Std[y])

Corr[x, y] = 64 / (7.2111 \* 7.2801)

Corr[x, y] = 0.9914

Therefore, the covariance of x and y is 64, and their correlation is 0.9914.

# Set the seed of the random number generator

set.seed(123)

# Set the number of samples

H <- 160

# Generate samples of the error variables

eps <- rnorm(H, mean = 0, sd = 2)

psi <- rnorm(H, mean = 0, sd = 3)

# Calculate the values of x and y

x <- 3 + 1 \* eps - 6 \* psi

y <- 5 + 7 \* eps - 2 \* psi

# Calculate the sample means of x and y

mean\_x <- mean(x)

mean\_y <- mean(y)

# Calculate the sample variances of x and y

var\_x <- var(x)

var\_y <- var(y)

# Calculate the sample standard deviations of x and y

sd\_x <- sd(x)

sd\_y <- sd(y)

# Calculate the sample correlation coefficient of x and y

r <- cor(x, y)

# Perform a hypothesis test to determine whether there is a significant correlation between x and y

cor.test.result <- cor.test(x, y)

# Extract the confidence interval from the test result

ci <- cor.test.result$conf.int

# Plot the confidence interval

plot(ci, type = "l", xlab = "", ylab = "Correlation coefficient", main = "95% Confidence Interval for the Correlation Coefficient")

abline(h = 0, col = "gray")

# Perform a hypothesis test to determine whether there is a significant correlation between x and y

cor.test.result <- cor.test(x, y, conf.level = 0.95)

# Extract the confidence interval from the test result

ci <- cor.test.result$conf.int

# Plot the confidence interval

plot(ci, type = "l", xlab = "", ylab = "Correlation coefficient", main = "95% Confidence Interval for the Correlation Coefficient")

abline(h = 0, col = "gray")

# Calculate the sample correlation coefficient of x and y

r <- cor(x, y)

# Perform a hypothesis test to determine whether the theoretical correlation coefficient is equal to the calculated value

cor.test.result <- cor.test(x, y, r = r, alternative = "two.sided", conf.level = 0.99)

# Extract the p-value from the test result

p.value <- cor.test.result$p.value

# Test the null hypothesis at the 1% level

if (p.value < 0.01) {

print("Reject the null hypothesis at the 1% level.")

} else {

print("Fail to reject the null hypothesis at the 1% level.")

}

# Fit a linear regression model to the data

model <- lm(y ~ x)

# Extract the residuals from the model

residuals <- model$residuals

# Extract the fitted values from the model

fitted.values <- model$fitted.values

# Plot the residuals against the fitted values

plot(fitted.values, residuals, xlab = "Fitted values", ylab = "Residuals")

abline(h = 0, col = "gray")

# Install and load the dplyr package

install.packages("dplyr")

library(dplyr)

# Calculate the returns in percent for USDT

usdt\_returns <- (usdt$Close / lag(usdt$Close) - 1) \* 100

# Calculate the returns in percent for XRP

xrp\_returns <- (xrp$Close / lag(xrp$Close) - 1) \* 100

# Make a table of frequencies 2 by 2 showing the number of "+" and "-" signs in the returns of USDT and XRP

table(usdt\_signs, xrp\_signs)

# Make a table of frequencies 2 by 2 showing the percentage of "+" and "-" signs in the returns of USDT and XRP, 100% for the entire table

prop.table(table(usdt\_signs, xrp\_signs), 1)

# Make a table of frequencies 2 by 2 showing the percentage of "+" and "-" signs in the returns of USDT and XRP, 100% for the rows

prop.table(table(usdt\_signs, xrp\_signs), 2)

# Create a frequency table showing the number of occurrences of each combination of "+" and "-" signs in the returns of USDT and XRP

freq\_table <- table(usdt\_signs, xrp\_signs)

# Calculate the proportion of "+" signs for each row

row\_props <- prop.table(freq\_table, 2)

# Sum the proportion of "+" signs in each row

sum(row\_props["+",])

# Calculate the proportion of "+" signs in the returns of USDT among those observations where the "-" sign is in the returns of XRP

prop.table(freq\_table["+", "-"], 1)

# Combine the close prices of USDT and XRP into a single vector

close\_prices <- c(usdt$Close, xrp$Close)

# Calculate the returns using the formula: return\_t = (close\_t / close\_{t-1}) - 1

returns <- (close\_prices[-1] / close\_prices[-length(close\_prices)] - 1) \* 100

# Plot the histogram of the returns

hist(returns, main = "Histogram of Returns", xlab = "Returns")

# Add a curve showing a normal distribution with the same mean and standard deviation as the returns

curve(dnorm(x, mean(returns), sd(returns)), add = TRUE, col = "red")

# Create a new vector containing the return of XRP for each "+" sign in the return of USDT

xrp\_returns\_plus <- xrp\_returns[usdt\_signs == "+"]

# Create a new vector containing the return of XRP for each "-" sign in the return of USDT

xrp\_returns\_minus <- xrp\_returns[usdt\_signs == "-"]

# Plot the return of XRP for each "+" sign in the return of USDT

plot(xrp\_returns\_plus, xlab = "Sign of Return (USD)", ylab = "Return (XRP)", main = "Regression of Return of XRP from Sign of Return of USDT")

# Add a point for each "-" sign in the return of USDT

points(xrp\_returns\_minus, col = "red")

# Add a regression line to the plot

abline(lm(xrp\_returns ~ usdt\_signs), col = "blue")

# Load the lmtest package

library(lmtest)

# Fit a linear regression model using the lm function

model <- lm(xrp\_returns ~ usdt\_signs)

# Calculate the 95% confidence interval for the coefficient at the sign variable using the coeftest function

confint <- coeftest(model, vcov = vcovHC(model, type = "HC1"))

# Extract the lower and upper bounds of the confidence interval

lower\_bound <- confint[2,1]

upper\_bound <- confint[2,2]

# Plot the confidence interval using the abline function

abline(h = lower\_bound, lty = 2, col = "blue")

abline(h = upper\_bound, lty = 2, col = "blue")