Lab 07

## setup

setwd("C:/Users/22700/Desktop")  
library(data.table)  
library(sandwich)  
library(lmtest)

## Loading required package: zoo

##   
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

library(stargazer)

##   
## Please cite as:

## Hlavac, Marek (2018). stargazer: Well-Formatted Regression and Summary Statistics Tables.

## R package version 5.2.2. https://CRAN.R-project.org/package=stargazer

## Violation of Simple Linear Regression Assumption 2

## SLR.2: We have a random sample of size n, (xi,yi): i = 1,2,. . . ,n following the population model laid out in SLR.1

# Create data for a population of 10,000 individuals

set.seed(1984)  
x1 <- rnorm(n = 10000, mean = 0 , sd = 3) # create indep. var. 1  
x2 <- rnorm(n = 10000, mean = 0, sd = 4) # create indep. var. 2  
e <- rnorm(n = 10000, mean = 0, sd = 2) # create error  
y <- 2 + 3\*x1 + 4\*x2 + e # create y according to population model

# Create a dataframe named “dt.population” putting together the population data:

dt.population <- data.table( y, x1, x2) # creates tables  
dt.population <- dt.population[order(y)]  
dt.population # shows first and last entries of table

## y x1 x2  
## 1: -65.51931 -8.9425248 -10.925706  
## 2: -64.08313 0.3982382 -16.079049  
## 3: -62.40373 -7.5047649 -10.113220  
## 4: -58.28314 -3.0228161 -12.841153  
## 5: -57.83996 -6.7402440 -9.934788  
## ---   
## 9996: 63.23389 5.1128133 10.935504  
## 9997: 63.99458 7.7187598 10.025892  
## 9998: 67.18546 6.1780560 11.427169  
## 9999: 67.34564 5.1032742 11.854979  
## 10000: 67.38101 3.9659545 13.253237

# Extract a random sample from the population using the function sample.

#help(sample) # when in doubt use help

##Use the function sample to extract a random sample of size 100, choosing from all the rows of our population dataset. This function takes a vector, and randomly selects the specified number of observations. In this case, as the population is 10000, the sample function selects 100 random numbers between 1 and 10000.

r.sample.rows <- sample(1:nrow(dt.population), size = 100)  
r.sample.rows # shows the vector of 100 randomly selected row numbers

## [1] 9312 6165 6458 6477 5188 406 2820 572 652 5549 9072 3968 2815 7752 4184  
## [16] 9956 3045 6106 1921 3925 6653 9426 9697 8545 7052 2314 8746 5711 4654 7588  
## [31] 3315 4286 2542 4340 4799 186 2661 4487 5296 2770 4991 8833 5568 4317 7128  
## [46] 3859 5196 1072 6460 3063 8471 2182 7634 727 8534 8819 597 4368 4776 4845  
## [61] 4483 9155 3458 4571 7679 4408 2644 4230 7691 1020 4918 4983 6298 5278 3204  
## [76] 976 758 8705 2609 4244 6084 5799 5596 9210 8573 6431 7388 388 6286 7904  
## [91] 2573 3262 6379 3460 8385 9613 8025 7927 4496 2157

#Now that we have selected the sample of rows we have to get those observations from our dataset.

r.sample <- dt.population[r.sample.rows,] # select the rows according to random sample  
head(r.sample) # show selection (note that the row numbers are the ones that were randomly selected above

## y x1 x2  
## 1: 29.432379 -2.4933338 8.455335  
## 2: 7.430384 -1.7219022 3.029325  
## 3: 8.966529 0.9625825 1.160981  
## 4: 9.036588 -1.4417106 2.091823  
## 5: 3.144875 1.9221459 -0.930215  
## 6: -30.386939 -0.6415410 -7.397989

# Run regression on a random sample:

summary(lm( y ~ x1 + x2, data = r.sample))

##   
## Call:  
## lm(formula = y ~ x1 + x2, data = r.sample)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -5.1047 -1.4439 -0.0498 1.1808 5.9282   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.90817 0.20551 9.285 4.69e-15 \*\*\*  
## x1 3.00848 0.07517 40.025 < 2e-16 \*\*\*  
## x2 4.09883 0.05554 73.796 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2.049 on 97 degrees of freedom  
## Multiple R-squared: 0.9841, Adjusted R-squared: 0.9838   
## F-statistic: 3008 on 2 and 97 DF, p-value: < 2.2e-16

#You can see that the estimated betas are very close to the population betas.

## Extract a non-random sample from the population:

nr.sample <- dt.population[1:100,] # select a subset of the first 100 rows  
head(nr.sample)

## y x1 x2  
## 1: -65.51931 -8.9425248 -10.925706  
## 2: -64.08313 0.3982382 -16.079049  
## 3: -62.40373 -7.5047649 -10.113220  
## 4: -58.28314 -3.0228161 -12.841153  
## 5: -57.83996 -6.7402440 -9.934788  
## 6: -57.68156 -5.4035347 -10.977426

#Run regression on a non-random sample:

summary(lm( y ~ x1 + x2, data = nr.sample))

##   
## Call:  
## lm(formula = y ~ x1 + x2, data = nr.sample)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -4.4988 -1.2392 -0.0186 1.2096 3.9580   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -5.7973 1.5566 -3.724 0.000329 \*\*\*  
## x1 2.5723 0.1078 23.868 < 2e-16 \*\*\*  
## x2 3.4089 0.1294 26.352 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1.744 on 97 degrees of freedom  
## Multiple R-squared: 0.8814, Adjusted R-squared: 0.8789   
## F-statistic: 360.3 on 2 and 97 DF, p-value: < 2.2e-16

#You can see that the estimated beta coefficients are now further away from the true betas

## Violation of Simple Linear Regression Assumption 3

## SLR.3: The sample outcomes x ,xi, i = 1,. . . ,n are not all the same value.

# Colinearity: The variable x2 is a constant.

set.seed(1984)  
x1 <- rnorm(n = 1000, mean = 0 , sd = 2) # create indep. var. 1  
x2 <- rep(3, times=1000) # create indep. var. 2 - a constant  
e <- rnorm(n = 1000, mean = 0, sd = 1) # create error  
y <- 2 + 3\*x1 + 4\*x2 + e # create y according to population model

# Run regression including constant.

summary(lm( y ~ x1 + x2 ))

##   
## Call:  
## lm(formula = y ~ x1 + x2)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3.3114 -0.6618 0.0197 0.6507 3.6707   
##   
## Coefficients: (1 not defined because of singularities)  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 13.98124 0.03209 435.7 <2e-16 \*\*\*  
## x1 2.99665 0.01620 185.0 <2e-16 \*\*\*  
## x2 NA NA NA NA   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1.014 on 998 degrees of freedom  
## Multiple R-squared: 0.9717, Adjusted R-squared: 0.9716   
## F-statistic: 3.423e+04 on 1 and 998 DF, p-value: < 2.2e-16

# x2 is dropped from the model. Notice that the intercept is not 2 but 2+β2∗x2=14

## Very high colinearity: x1 and x2 are very highly correlated.

set.seed(1984)  
x1 <- rnorm(n = 1000, mean = 0 , sd = 2) # create indep. var. 1  
x2 <- 0.4\*x1 + rnorm(n=1000, mean = 0, sd =0.01) # create indep. var. 2 - mean = x1  
e <- rnorm(n = 1000, mean = 0, sd = 1) # create error  
y <- 2 + 3\*x1 + 4\*x2 + e # create y according to population model

In statistics, multicollinearity (also collinearity) is a phenomenon in which two or more predictor variables in a multiple regression model are highly correlated, meaning that one can be linearly predicted from the others with a non-trivial degree of accuracy. In this situation the coefficient estimates of the multiple regression may change erratically in response to small changes in the model or the data. Multicollinearity does not reduce the predictive power or reliability of the model as a whole, at least within the sample data set; it only affects calculations regarding individual predictors. That is, a multiple regression model with correlated predictors can indicate how well the entire bundle of predictors predicts the outcome variable, but it may not give valid results about any individual predictor, or about which predictors are redundant with respect to others. In case of perfect multicollinearity the predictor matrix is singular and therefore cannot be inverted. Under these circumstances, the ordinary least-squares estimator does not exist. Note that in statements of the assumptions underlying regression analyses such as ordinary least squares, the phrase “no multicollinearity” is sometimes used to mean the absence of perfect multicollinearity, which is an exact (non-stochastic) linear relation among the regressors.

# Multicollinearity affects your standard errors and thus the significance of your results:

cor.test(x2,x1)

##   
## Pearson's product-moment correlation  
##   
## data: x2 and x1  
## t = 2469.3, df = 998, p-value < 2.2e-16  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## 0.9999074 0.9999277  
## sample estimates:  
## cor   
## 0.9999182

summary(lm( y ~ x1 + x2 ))

##   
## Call:  
## lm(formula = y ~ x1 + x2)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -2.84702 -0.68167 -0.01766 0.69449 2.81105   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 2.02596 0.03151 64.300 <2e-16 \*\*\*  
## x1 2.18183 1.24299 1.755 0.0795 .   
## x2 6.02423 3.10749 1.939 0.0528 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.9954 on 997 degrees of freedom  
## Multiple R-squared: 0.9882, Adjusted R-squared: 0.9882   
## F-statistic: 4.169e+04 on 2 and 997 DF, p-value: < 2.2e-16

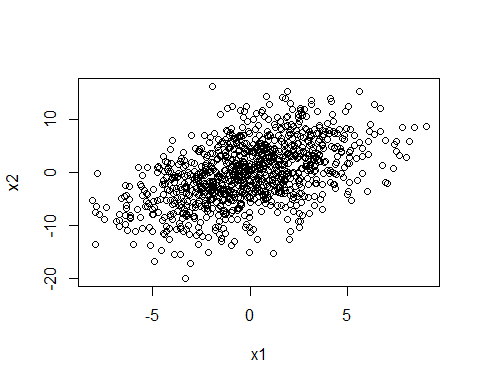
## Violation of Simple Linear Regression Assumption 4

## SLR.4: E(u|x)=0 (the error u has an expected value of zero given any value of the explanatory variable)

# In this case we have variables hidden in the error term that should be in our model.

## If x1 and x2 are positively correlated and β2 is positive, then if we omit x2 from the model βˆ1 will be positively biased.

set.seed(1984)  
x1 <- rnorm(n = 1000, mean = 0 , sd = 3) # create indep. var. 1  
x2 <- rnorm(n = 1000, mean = x1 , sd = 5) # create indep. var. 2  
plot(x1,x2) # plot x1 against x2



cor.test(x = x1, y = x2) # get correlation coefficient between x1 and x2

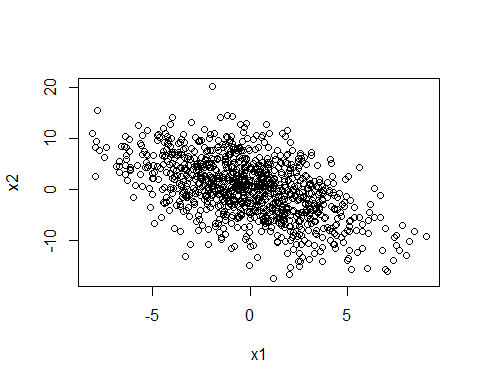
##   
## Pearson's product-moment correlation  
##   
## data: x1 and x2  
## t = 18.314, df = 998, p-value < 2.2e-16  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## 0.4536584 0.5465459  
## sample estimates:  
## cor   
## 0.5015462

e <- rnorm(n = 1000, mean = 0, sd = 1) # create error  
y <- 2 + 3\*x1 + 4\*x2 + e # create y according to population model  
out.y.full <- lm ( y ~ x1 + x2) # full model  
out.y.x1.om <- lm ( y ~ x1) # model with x2 ommitted  
stargazer(out.y.full, out.y.x1.om, type="text")

##   
## ===========================================================================  
## Dependent variable:   
## -------------------------------------------------------  
## y   
## (1) (2)   
## ---------------------------------------------------------------------------  
## x1 2.990\*\*\* 6.950\*\*\*   
## (0.012) (0.216)   
##   
## x2 4.004\*\*\*   
## (0.006)   
##   
## Constant 2.026\*\*\* 1.650\*\*   
## (0.032) (0.643)   
##   
## ---------------------------------------------------------------------------  
## Observations 1,000 1,000   
## R2 0.999 0.508   
## Adjusted R2 0.999 0.508   
## Residual Std. Error 0.995 (df = 997) 20.324 (df = 998)   
## F Statistic 422,429.000\*\*\* (df = 2; 997) 1,030.926\*\*\* (df = 1; 998)  
## ===========================================================================  
## Note: \*p<0.1; \*\*p<0.05; \*\*\*p<0.01

## If x1 and x2 are negatively correlated and β2 is negative, then if we omit x2 from the model βˆ1 will be positively biased.

set.seed(1984)  
x1 <- rnorm(n = 1000, mean = 0 , sd = 3) # create indep. var. 1  
x2 <- rnorm(n = 1000, mean = -x1 , sd = 5) # create indep. var. 2  
plot(x1,x2) # plot x1 against x2



cor.test(x = x1, y = x2) # get correlation coefficient between x1 and x2

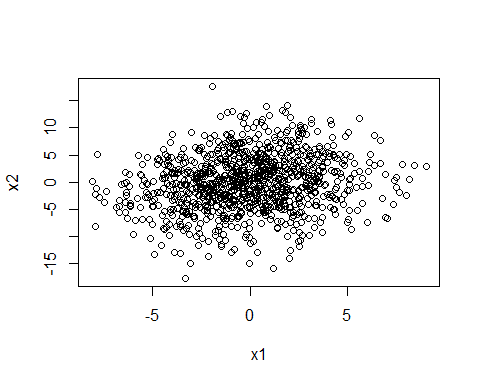
##   
## Pearson's product-moment correlation  
##   
## data: x1 and x2  
## t = -18.728, df = 998, p-value < 2.2e-16  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## -0.5544216 -0.4625868  
## sample estimates:  
## cor   
## -0.5099558

e <- rnorm(n = 1000, mean = 0, sd = 1) # create error  
y <- 2 + 3\*x1 - 4\*x2 + e # create y according to population model  
out.y.full <- lm ( y ~ x1 + x2) # full model  
out.y.x1.om <- lm ( y ~ x1) # model with x2 ommitted  
stargazer(out.y.full, out.y.x1.om, type="text")

##   
## ===========================================================================  
## Dependent variable:   
## -------------------------------------------------------  
## y   
## (1) (2)   
## ---------------------------------------------------------------------------  
## x1 2.998\*\*\* 7.039\*\*\*   
## (0.012) (0.216)   
##   
## x2 -3.996\*\*\*   
## (0.006)   
##   
## Constant 2.026\*\*\* 2.401\*\*\*   
## (0.032) (0.642)   
##   
## ---------------------------------------------------------------------------  
## Observations 1,000 1,000   
## R2 0.999 0.516   
## Adjusted R2 0.999 0.515   
## Residual Std. Error 0.995 (df = 997) 20.283 (df = 998)   
## F Statistic 427,153.100\*\*\* (df = 2; 997) 1,061.892\*\*\* (df = 1; 998)  
## ===========================================================================  
## Note: \*p<0.1; \*\*p<0.05; \*\*\*p<0.01

## If x1 and x2 are positively correlated and β2 is negative, then if we omit x2 from the model βˆ1 will be negatively biased.

set.seed(1984)  
x1 <- rnorm(n = 1000, mean = 0 , sd = 3) # create indep. var. 1  
x2 <- rnorm(n = 1000, mean = x1/3 , sd = 5) # create indep. var. 2  
plot(x1,x2) # plot x1 against x2



cor.test(x = x1, y = x2) # get correlation coefficient between x1 and x2

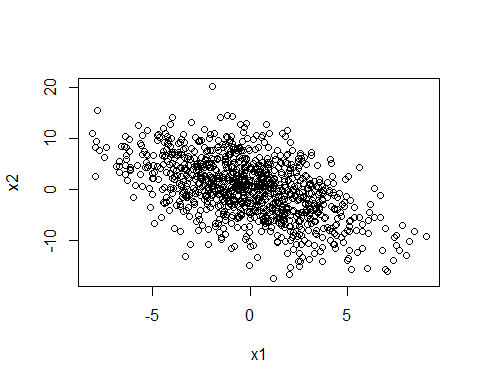
##   
## Pearson's product-moment correlation  
##   
## data: x1 and x2  
## t = 5.9669, df = 998, p-value = 3.355e-09  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## 0.1250425 0.2447737  
## sample estimates:  
## cor   
## 0.1855969

e <- rnorm(n = 1000, mean = 0, sd = 1) # create error  
y <- 2 + 3\*x1 - 4\*x2 + e # create y according to population model  
out.y.full <- lm ( y ~ x1 + x2) # full model  
out.y.x1.om <- lm ( y ~ x1) # model with x2 ommitted  
stargazer(out.y.full, out.y.x1.om, type="text")

##   
## ========================================================================  
## Dependent variable:   
## ----------------------------------------------------  
## y   
## (1) (2)   
## ------------------------------------------------------------------------  
## x1 2.993\*\*\* 1.706\*\*\*   
## (0.011) (0.216)   
##   
## x2 -3.996\*\*\*   
## (0.006)   
##   
## Constant 2.026\*\*\* 2.401\*\*\*   
## (0.032) (0.642)   
##   
## ------------------------------------------------------------------------  
## Observations 1,000 1,000   
## R2 0.998 0.059   
## Adjusted R2 0.998 0.058   
## Residual Std. Error 0.995 (df = 997) 20.283 (df = 998)   
## F Statistic 219,639.600\*\*\* (df = 2; 997) 62.351\*\*\* (df = 1; 998)  
## ========================================================================  
## Note: \*p<0.1; \*\*p<0.05; \*\*\*p<0.01

## If x1 and x2 are negatively correlated and β2 is positive, then if we omit x2 from the model βˆ1 will be negatively biased.

set.seed(1984)  
x1 <- rnorm(n = 1000, mean = 0 , sd = 3) # create indep. var. 1  
x2 <- rnorm(n = 1000, mean = -x1 , sd = 5) # create indep. var. 2  
plot(x1,x2) # plot x1 against x2



cor.test(x = x1, y = x2) # get correlation coefficient between x1 and x2

##   
## Pearson's product-moment correlation  
##   
## data: x1 and x2  
## t = -18.728, df = 998, p-value < 2.2e-16  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## -0.5544216 -0.4625868  
## sample estimates:  
## cor   
## -0.5099558

e <- rnorm(n = 1000, mean = 0, sd = 1) # create error  
y <- 2 + 3\*x1 + 4\*x2 + e # create y according to population model  
out.y.full <- lm ( y ~ x1 + x2) # full model  
out.y.x1.om <- lm ( y ~ x1) # model with x2 ommitted  
stargazer(out.y.full, out.y.x1.om, type="text")

##   
## ========================================================================  
## Dependent variable:   
## ----------------------------------------------------  
## y   
## (1) (2)   
## ------------------------------------------------------------------------  
## x1 2.998\*\*\* -1.050\*\*\*   
## (0.012) (0.216)   
##   
## x2 4.004\*\*\*   
## (0.006)   
##   
## Constant 2.026\*\*\* 1.650\*\*   
## (0.032) (0.643)   
##   
## ------------------------------------------------------------------------  
## Observations 1,000 1,000   
## R2 0.998 0.023   
## Adjusted R2 0.998 0.022   
## Residual Std. Error 0.995 (df = 997) 20.324 (df = 998)   
## F Statistic 212,442.600\*\*\* (df = 2; 997) 23.551\*\*\* (df = 1; 998)  
## ========================================================================  
## Note: \*p<0.1; \*\*p<0.05; \*\*\*p<0.01

## The bias will be:

out.y.full <- lm ( y ~ x1 + x2)  
coeffs.full <- coefficients(out.y.full)  
b2\_hat <- coeffs.full[3]  
b1\_hat <- coeffs.full[2]  
out.part.x2 <- lm ( x2 ~ x1)  
coeffs.part <- coefficients(out.part.x2)  
delta <- coeffs.part[2]  
bias <- delta\*b2\_hat  
bias

## x1   
## -4.048779

b1\_tilda = b1\_hat + bias  
b1\_tilda

## x1   
## -1.050385

## Violation of Simple Linear Regression Assumption 5

## SLR.5: V (u|x) = σ2 (homoskedasticity)

# The variance of the errror term depends on the x - there is heteroskedasticity in the data. Generate random variables x1, x2, and e.

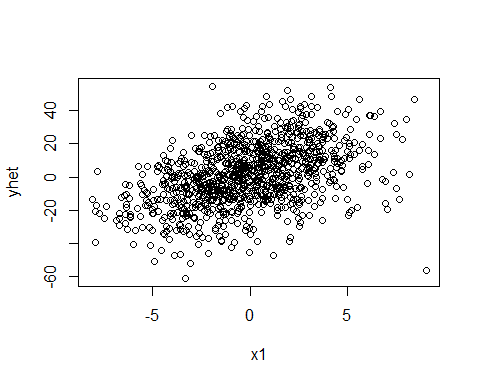
set.seed(1984)  
x1 <- rnorm(n = 1000, mean = 0 , sd = 3) # create indep. var. 1  
x2 <- rnorm(n = 1000, mean = 0 , sd = 4) # create indep. var. 1  
e <- rnorm(n = 1000, mean = 0, sd = 1) # create error  
s <- exp(0.4 \* x1 ) # create function of x1

# Create a model with heteroskedasticity: the error term is a function of a dependent variable (in this case x1).

yhet <- 2 + 3\*x1 + 4\*x2 + s\*e # model with heteroskedasticity

# You cannot detect heteroskedasticity just by looking at the plot.

plot(x1, yhet)

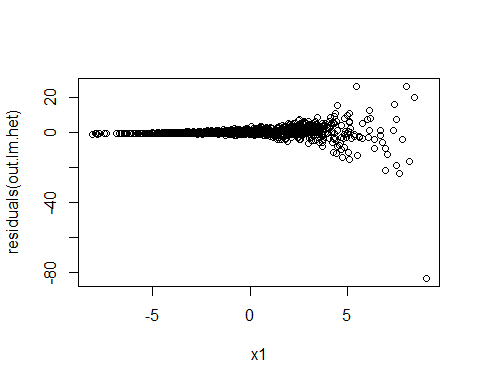


# But, you can identify the presence of heteroskedasticity by looking at the residuals plot. If there is a relationship between the x and the residuals then we have heteroskedasticity.

summary(out.lm.het <- lm( yhet ~ x1 + x2))

##   
## Call:  
## lm(formula = yhet ~ x1 + x2)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -83.484 -0.660 -0.143 0.716 26.171   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.87087 0.12831 14.58 <2e-16 \*\*\*  
## x1 2.86007 0.04317 66.25 <2e-16 \*\*\*  
## x2 4.01248 0.03164 126.83 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 4.054 on 997 degrees of freedom  
## Multiple R-squared: 0.9533, Adjusted R-squared: 0.9532   
## F-statistic: 1.018e+04 on 2 and 997 DF, p-value: < 2.2e-16

plot(y = residuals(out.lm.het), x=x1) # residual plot with heteroskedasticity

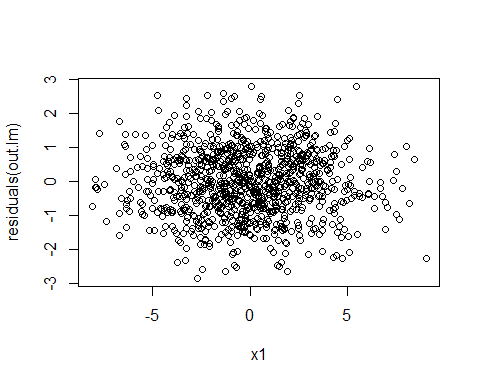


# Create the same model without heteroskedasticity.

ynorm <- 2 + 3\*x1 + 4\*x2 + e # model with no heteroskedasticity  
summary(out.lm <- lm( ynorm ~ x1 + x2))

##   
## Call:  
## lm(formula = ynorm ~ x1 + x2)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -2.84702 -0.68167 -0.01766 0.69449 2.81105   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 2.025961 0.031508 64.3 <2e-16 \*\*\*  
## x1 2.994346 0.010601 282.5 <2e-16 \*\*\*  
## x2 4.005061 0.007769 515.5 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.9954 on 997 degrees of freedom  
## Multiple R-squared: 0.9971, Adjusted R-squared: 0.9971   
## F-statistic: 1.718e+05 on 2 and 997 DF, p-value: < 2.2e-16

plot(y = residuals(out.lm), x=x1) # residual plot without heteroskedasticity



# Heteroskedasticity does not affect the coefficients but if affects the standard errors.

# How can we know whether or not we have heteroskedasticity in our data? Use Breusch-Pagan test.

help(bptest)

## starting httpd help server ... done

bptest( ynorm ~ x1 + x2 ) # high p-value, can’t reject null of constant variance

##   
## studentized Breusch-Pagan test  
##   
## data: ynorm ~ x1 + x2  
## BP = 0.1777, df = 2, p-value = 0.915

bptest( yhet ~ x1 + x2 ) # low p-value, reject null of constant variance

##   
## studentized Breusch-Pagan test  
##   
## data: yhet ~ x1 + x2  
## BP = 29.487, df = 2, p-value = 3.953e-07

# If we have heteroskedasticity in our data, what can we do about it? We can adjust the way R calculates the variances of the regression coefficients. The usual way

vcov(out.lm)

## (Intercept) x1 x2  
## (Intercept) 9.927650e-04 1.363781e-05 4.528209e-06  
## x1 1.363781e-05 1.123784e-04 5.393789e-07  
## x2 4.528209e-06 5.393789e-07 6.035322e-05

diag(vcov(out.lm))

## (Intercept) x1 x2   
## 9.927650e-04 1.123784e-04 6.035322e-05

sqrt( diag(vcov(out.lm)) ) # standard errors without adjustment

## (Intercept) x1 x2   
## 0.031508173 0.010600868 0.007768733

sqrt( diag(vcovHC(out.lm)) ) # standard errors with adjustment

## (Intercept) x1 x2   
## 0.031558230 0.010320489 0.007555949

# With heteroskedasticity:

vcovHC(out.lm.het)

## (Intercept) x1 x2  
## (Intercept) 0.0198862619 0.0145904657 -0.0007972898  
## x1 0.0145904657 0.0118946088 -0.0006158823  
## x2 -0.0007972898 -0.0006158823 0.0004453495

diag(vcovHC(out.lm.het))

## (Intercept) x1 x2   
## 0.0198862619 0.0118946088 0.0004453495

# If you apply the correction to a model with no heteroskedasticity the SEs become higher

sqrt( diag(vcov(out.lm.het)) ) # standard errors without adjustment

## (Intercept) x1 x2   
## 0.12831348 0.04317084 0.03163729

sqrt( diag(vcovHC(out.lm.het)) ) # standard errors with adjustment

## (Intercept) x1 x2   
## 0.14101866 0.10906241 0.02110331

# How do you apply it to your regression results?

coeftest(out.lm.het) # before correction

##   
## t test of coefficients:  
##   
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.870872 0.128313 14.581 < 2.2e-16 \*\*\*  
## x1 2.860071 0.043171 66.250 < 2.2e-16 \*\*\*  
## x2 4.012475 0.031637 126.827 < 2.2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

coeftest(out.lm.het, vcov. = vcovHC(out.lm.het)) # after correction

##   
## t test of coefficients:  
##   
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.870872 0.141019 13.267 < 2.2e-16 \*\*\*  
## x1 2.860071 0.109062 26.224 < 2.2e-16 \*\*\*  
## x2 4.012475 0.021103 190.135 < 2.2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

## Extras

# Select rows by inserting the row number, a vector of row numbers or a sequence of row numbers inside the square brackets and before the comma.

dt.population[1,]

## y x1 x2  
## 1: -65.51931 -8.942525 -10.92571

dt.population[1:5,]

## y x1 x2  
## 1: -65.51931 -8.9425248 -10.925706  
## 2: -64.08313 0.3982382 -16.079049  
## 3: -62.40373 -7.5047649 -10.113220  
## 4: -58.28314 -3.0228161 -12.841153  
## 5: -57.83996 -6.7402440 -9.934788

c <- c(1,7,8)  
dt.population[c,]

## y x1 x2  
## 1: -65.51931 -8.942525 -10.92571  
## 2: -55.92612 -3.082815 -11.64722  
## 3: -55.54762 -5.394655 -10.38010

# Select columns by inserting the columns number, a vector of column numbers or a sequence of columns numbers inside the square brackets and before the comma.

head(dt.population[,1])

## y  
## 1: -65.51931  
## 2: -64.08313  
## 3: -62.40373  
## 4: -58.28314  
## 5: -57.83996  
## 6: -57.68156

head(dt.population[,1:2])

## y x1  
## 1: -65.51931 -8.9425248  
## 2: -64.08313 0.3982382  
## 3: -62.40373 -7.5047649  
## 4: -58.28314 -3.0228161  
## 5: -57.83996 -6.7402440  
## 6: -57.68156 -5.4035347

# Select both rows and columns:

dt.population[1,1]

## y  
## 1: -65.51931

dt.population[1:2,1:2]

## y x1  
## 1: -65.51931 -8.9425248  
## 2: -64.08313 0.3982382