```
#Assignment part 1


#data check through model diagnostics (11)
library(psych) # for describe
library(car) # for residualPlots, vif, pairs.panels, ncvTest
library(lmtest) # bptest
library(sandwich) # for coeftest vcovHC estimator
library(boot) # for bootstrapping
library(lmboot) # for wild bootsrapping
library(tidyverse) # for tidy code
# tidyverse should be the last one to run

#The dataset upload:
data_sample_1 = read_csv("https://raw.githubusercontent.com/kekecsz/PSYP14-Advanced-Scientific-
Methods/main/Home%20assignment/home_sample_1.csv")
View(data_sample_1)


# ## Custom functions
# We will use these custom functions to get bootstrapped confidence intervals.

# function to obtain regression coefficients
# source: https://www.statmethods.net/advstats/bootstrapping.html
bs_to_boot <- function(model, data, indices) {
  d <- data[indices,] # allows boot to select sample
  fit <- lm(formula(model), data=d)
  return(coef(fit))
}

# function to obtain adjusted R^2
# source: https://www.statmethods.net/advstats/bootstrapping.html (partially modified)
adjR2_to_boot <- function(model, data, indices) {
  d <- data[indices,] # allows boot to select sample
  fit <- lm(formula(model), data=d)
  return(summary(fit)$adj.r.squared)
}

# Computing the booststrap BCa (bias-corrected and accelerated) bootstrap confidence intervals
by Elfron (1987)
# This is useful if there is bias or skew in the residuals.

confint.boot <- function(model, data = NULL, R = 1000){
  if(is.null(data)){
    data = eval(parse(text = as.character(model$call[3])))
  }
  boot.ci_output_table = as.data.frame(matrix(NA, nrow = length(coef(model)), ncol = 2))
  row.names(boot.ci_output_table) = names(coef(model))
  names(boot.ci_output_table) = c("boot 2.5 %", "boot 97.5 %")
  results.boot = results <- boot(data=data, statistic=bs_to_boot,
                              R=1000, model = model)

  for(i in 1:length(coef(model))){
    boot.ci_output_table[i,] = unlist(unlist(boot.ci(results.boot, type="bca", index=i))
[c("bca4", "bca5")])
  }

  return(boot.ci_output_table)
}

# Computing the booststrapped confidence interval for a linear model using wild bottstrapping as
descibed by Wu (1986) <doi:10.1214/aos/1176350142>
# requires the lmboot pakcage

wild.boot.confint <- function(model, data = NULL, B = 1000){
  if(is.null(data)){
    data = eval(parse(text = as.character(model$call[3])))
  }
  wild_boot_estimates = wild.boot(formula(model), data = data, B = B)

  result = t(apply(wild_boot_estimates[[1]], 2, function(x) quantile(x,probs=c(.025,.975))))

  return(result)

}


#explore the data data
data_sample_1 %>%
  summary()

sex = "column 3"
age = "column 4"

####
#Data analysis

#explore extreme cases Pain and Age
data_sample_1 %>%
  mutate(rownum = row.names(data_sample_1)) %>%
  ggplot() + aes(x = pain, y = age, label = rownum) +
  geom_point() + geom_text()

data_sample_1 %>%
  mutate(rownum = row.names(data_sample_1)) %>%
  ggplot() + aes(x = pain, y = age, label = rownum) +
  geom_label()

#Identifying extreme cases with high leverage
data_sample_1 %>%
  ggplot() + aes(x = pain, y = age) + geom_point() +
  geom_smooth(method = "lm")
## `geom_smooth()` using formula 'y ~ x'

#explore extreme cases Pain and sex
data_sample_1 %>%
  mutate(rownum = row.names(data_sample_1)) %>%
  ggplot() + aes(x = pain, y = sex, label = rownum) +
  geom_point() + geom_text()

data_sample_1 %>%
  mutate(rownum = row.names(data_sample_1)) %>%
  ggplot() + aes(x = pain, y = sex, label = rownum) +
  geom_label()

#Identifying extreme cases with high leverage
data_sample_1 %>%
  ggplot() + aes(x = pain, y = sex) + geom_point() +
  geom_smooth(method = "lm")

## `geom_smooth()` using formula 'y ~ x'

#explore extreme cases Age and sex
data_sample_1 %>%
  mutate(rownum = row.names(data_sample_1)) %>%
  ggplot() + aes(x = age, y = sex, label = rownum) +
  geom_point() + geom_text()

data_sample_1 %>%
  mutate(rownum = row.names(data_sample_1)) %>%
  ggplot() + aes(x = age, y = sex, label = rownum) +
  geom_label()

#Identifying extreme cases with high leverage
data_sample_1 %>%
  ggplot() + aes(x = age, y = sex) + geom_point() +
  geom_smooth(method = "lm")
## `geom_smooth()` using formula 'y ~ x'


#Build the models
model1 = lm(pain ~ age + sex, data = data_sample_1)
model1

model2 = lm(pain ~ age + sex + STAI_trait + pain_cat + mindfulness + cortisol_serum, data =
data_sample_1)
model2


###
#model diagnostics

#residuals vs Leverage
model2 %>%
  plot(which = 5)

#cook´s distance
model2 %>%
  plot(which = 4)

#Assumptions of linear regression
#Normality
# QQ plot
model2 %>%
  plot(which = 2)

# histogram
residuals_model2 = enframe(residuals(model2))
residuals_model2 %>%
  ggplot() + aes(x = value) + geom_histogram()

# skew and kurtosis
describe(residuals(model2))

# show models
summary(model1)
summary(model2)


#excluding outliers
data_sample_2 = data_sample_1 %>% #dont need to call it anything else, just equal the same name
  slice(-c(8, 34, 47, 88))
View(data_sample_2)


#Build model
model1b = lm(pain ~ age + sex, data = data_sample_2) #correct, use new names!!
model1b

model2b = lm(pain ~ age + sex + STAI_trait + pain_cat + mindfulness + cortisol_serum, data =
data_sample_2)
model2b

# recheck the assumption of normality of residuals
describe(residuals(model1b))
residuals_model1b = enframe(residuals(model1b))
residuals_model1b %>%
  ggplot() + aes(x = value) + geom_histogram()

describe(residuals(model2b))
residuals_model2b = enframe(residuals(model2b))
residuals_model2b %>%
  ggplot() + aes(x = value) + geom_histogram()

#
```

```
# Check the model
#residuals vs Leverage
model1b %>%
  plot(which = 5)
model2b %>%
  plot(which = 5)

#cook´s distance
model1b %>%
  plot(which = 4)
model2b %>%
  plot(which = 4)

# QQ plot
model1b %>%
  plot(which = 2)
model2b %>%
  plot(which = 2)

# histogram
residuals_model1b = enframe(residuals(model1b))
residuals_model1b %>%
  ggplot() + aes(x = value) + geom_histogram()
residuals_model2b = enframe(residuals(model2b))
residuals_model2b %>%
  ggplot() + aes(x = value) + geom_histogram()

# skew and kurtosis
describe(residuals(model1b))
describe(residuals(model2b))
#between -1 and 1 = no violation of normality

shapiro.test(residuals(model1b))
shapiro.test(residuals(model2b))


#Linearity
model1b %>%
  residualPlots()
model2b %>%
  residualPlots()


#Homoscedasticty
model1b %>%
  plot(which = 3)
model1b %>%
  ncvTest() # NCV test

model2b %>%
  plot(which = 3)
model2b %>%
  ncvTest() # NCV test


model1b %>%
  vif()
model2b %>%
  vif()


# comparing the models on data with and without the outliers
summary(model1b)
summary(model2b)

# Comparison
#run summary function and 95% confidence intervals
coef_table = function(model){
  require(lm.beta)
  mod_sum = summary(model)
  mod_sum_p_values = as.character(round(mod_sum$coefficients[,4], 3))
  mod_sum_p_values[mod_sum_p_values != "0" & mod_sum_p_values != "1"] =
substr(mod_sum_p_values[mod_sum_p_values != "0" & mod_sum_p_values != "1"], 2,
nchar(mod_sum_p_values[mod_sum_p_values != "0" & mod_sum_p_values != "1"]))
  mod_sum_p_values[mod_sum_p_values == "0"] = "<.001"


  mod_sum_table = cbind(as.data.frame(round(cbind(coef(model), confint(model), c(0,
lm.beta(model)$standardized.coefficients[c(2:length(model$coefficients))])), 2)),
mod_sum_p_values)
  names(mod_sum_table) = c("b", "95%CI lb", "95%CI ub", "Std.Beta", "p-value")
  mod_sum_table["(Intercept)","Std.Beta"] = "0"
  return(mod_sum_table)
}
coef_table (model1b) #raport in table and put in appendix (ex: 95% [-0.08, -0.01])
coef_table (model2b)


#AIC
AIC(model1b)
AIC(model2b)

anova(model1b, model2b)
```