# Линейни модели

# Линейна регресия

NN <- 300

set.seed(73391)

x1 <- round(runif(NN, 0, 5), 1)

x2 <- round(runif(NN, 2, 6), 1)

y <- 3 + 2.5\*x1 + rnorm(NN)

DF <- data.frame(x1, x2, y)

# Какво представлява линейната регресия

# Линейната регресия е статистически метод, който ни позволява да проучим и обобщим връзките

# между две множества от непрекъснати променливи - X и y:

# - в множеството X се намират обясняващите променливи (наречени още предиктори или независими

# променливи) и на върху тях се основават нашите прогнози;

# - в множеството y се съдържа една променлива (вектор), наричаща се зависима променлива

# променливаи или резултат, която искаме да прогнозираме.

# Ако приемем, че размерът на вектора y e N, a размерът на множеството X е N x p, то

# връзката между двете множества е y = b(0) + b(1)\*x(1) + ... + b(p)\*x(p) + error, където

# b(0) е константа, а b(1), ..., b(p) са параметрите, които обясняват влиянието на X над y.

# Линейната регресия ни позволява да оценим стойностите на тези p+1 коефцициенти.

# Има няколко начина за намирането на тези коефициенти, но най-често използваният е OLS

# (метод на най-малките квадрати)

# В R, функцията за линейна регресия е lm()

# Два начина за извикване на линейна регресия. Първият, ако виждаме нужните ни променливите

# в средата на R. Лесно можем да проверим дали променливите са заредени в среда на R с функцията

# ls().

model1 <- lm(y ~ x1)

model1

rm(list = c("x1", "x2", "y"))

# Вторият начин е като посочим data frame-а или матрицата, който(която) съдържа необходимите

# променливи.

model2 <- lm(y ~ x1, data = DF)

model2

rm(list = "model2")

# Когато изследваме връзка между една зависима и една обясняваща променлива, тогава линейната

# регресия е едномерна (или проста). При наличието на повече предиктори (обясняващи променливи),

# тогава имаме многомерна линейна регресия.

# Горните два модела са пример за проста линейна регресия

# Многомерната линейна регресия ще бъде разгледана по-подробно по-нататък в курса.

222

# След като сме построили линееен модел, следващата стъпка е да проверим до колко този модел

# описва добре данни и какви са оценките на коефициенти му.

summary(model1)

# -------

# Хипотези и проверка на хипотези

# Накратко, статистическата хипотеза е предположение за параметър на извадката/популацията.

# Това предположение може да бъде вярно или невярно. Ето защо съществуват две взаимоизключващи се

# хипотези - нулева (H0) и алтернативна (H1).

# Имаме три типа хипотези:

# 1. H0: параметър = число, H1: параметър != число

# 2. H0: параметър <= число, H1: параметър > число

# 3. H0: параметър >= число, H1: параметър < число

# Проверката на хипотезите става с помощта на тестове. На база вида на теста, искаме да

# отхвърлим или не нулевата хипотеза H0. Дали нулевата хипотеза е отхвърлена се определя от

# стойност, наречена "p-value". Стандартно, една H0 се отхвърля при стойност на p-value < 0.05.

# При отхвърляне на нулева хипотеза, за вярна се приема алтернативната H1.

# -------

# Първо ще проверим дали коефициентите са статистически значими, тоест дали е необходимо да

# участват в анализа. За всеки един коефициент проверяваме хипотезата дали коефициентът е равен

# на 0 (b(i) ?= 0). За да бъде един коефициент значим, то трябва за него да отхвърлим горната

# хипотеза. Както беше споменато по-горе, за да се отхвърли H0, то стойността на p-vaue трябва да

# бъде по-малка от 0.05. Стойностите p-value се намират в колоната "Pr(>|t|)". Стойностите на

# p-value за двата параметъра е 2e-16 << 0,05 и следователно двата параметъра са статистически

# значими.

# Преди да продължим с изследването на регресията, нека да видим случай, когато коефициентите

# не са значими.

summary(model3 <- lm(y ~ x2, data = DF))

rm("model3")

# Да разгледаме оценките пред коефицнета x2. Оценката на коефициента е -0,1505. Но въпреки, че

# стойността му е различна от 0, то той е статистически незначим. Защо? Защото стойността на

# p-value e 0,414 > 0,05. Тоест, този коефициент може да отпадне от анализа.

# Следващата стъпка е да проверим до колко модела описва добре данните. За целта ще използваме

# статистиките "Multiple R-squared" или "Adjusted R-squared". Статистиката "Multiple R-squared"

# приема стойности в интервала [0-1]. Колкото тази статистика се приближава до единица, толкова

# моделът е по-добър. И обратното, колкото стойността на R2 клони към 0, толкова моделът не се

# справя с описването на данните. Моделите, които имат стойности за R2 под 0.5, ги приемаме за

# слаби.

# Препоръчително е да се използва обаче статистиката Adjusted R-squared, защото тя "наказва",

# когато използваме ненужни променливи. По принцип, тази статистика също приема стойности в

# интервала [0-1], но когато използваме само статистически незначими, тогава Adjusted R-squared

# може да приеме и отрицателни стойности.

summary(model1)

# Какво можем да кажем за model2? Стойността на Adjusted R2 е 0.9284. Тоест моделът описва

# много добре данните.

# - Какво представлява R2 и как можем да го изчислим?

# - За целта първо ще разгледаме начините да изчисляваме прогнози и остатъците (residuals).

# Регресионното уравнение придобива вида y = 3.027 + 2.247\*x1.

# Линейната регресия позволява не само да се оценят връзките между отделните обясняващи

# променливи и резулатата, но както споменахме по-горе, позволява да се правят прогнози. В R

# използваме функцията "predict" за прогнозиране. Функцията съдържа два основни параметъра object

# (построения модел) и newdata (данни, за които искаме да направим прогноза)

model1.predictions <- predict(object = model1, newdata = DF)

model1.predictions.alt <- model1$coefficients[1] + model1$coefficients[2]\*DF$x1 # Алтернативен начин

all(model1.predictions == model1.predictions.alt)

rm("model1.predictions.alt")

# Нека да видим на графика как изглеждат прогнозите спрямо реалните стойности

plot(model1.predictions, DF$y)

abline(a = 0, b = 1, col = "red", lwd = 2)

# От графиката се вижда, че прогнозите и реалните стойности се движат около ъглополовящата на

# първи квадрант (x = y)

# Остатъците са разликата между наблюдаваната стойност и направената прогноза. За целта ще

# използваме функцията residuals(). Параметърът object приема стойността на модела, за когото

# желаем да оценим остатъците.

res <- residuals(object = model1)

res.alt <- DF$y - model1.predictions

all(round(res, 10) == round(res.alt, 10))

rm("res.alt")

# Нека сега се върнм към Multiple R-squared. В своята същност R2 представлява

# 1 - съотношението на вариацията на остатъците и общата вариация. Колкото един модел е по-добър,

# толкова остатъците му следва да бъдат по-малки, а от там и вариацията им. Тъй като общата

# вариация е константа, то можем да използваме тази статистика при сравняването на моделите и

# избора на по-добрия.

summary(model1)$r.squared

1 - var(res)/var(DF$y)

# Условия

# Не на последно място, остава да се проверят дали линейанта регресия (а и всички останали

# линейни модели или ML алгоритми) отговарят на три необходими условия

# 1. Константна вариация на грешките (Хомоскедастичност)

# Това е най-важното условие при линейните модели. Целта на константната вариация е около

# регресионната линия да се изгради "тунел" и да може да се определи (с някаква вероятност), в

# какви граници се намира прогнозата. При нарушение на това условие, за определени интервали

# грешката ще бъде по-малка от очакваното, а за други - по-голяма. Проблемът е, че ако очакваме

# определени неблагоприятни сценарии, те може да се окажат още по-лоши.

# Това условие най-лесно се проверява графично. По остта X изобразяваме прогнозите, а по Y -

# остатъците

plot(model1.predictions, res)

abline(h = 1.96\*c(-1, 1)\*round(sd(res, 2)), col = "red", lty = 4)

# За да имаме хомоскедастичност, то остатъците трябва да бъдат разпръснати равномерно по

# цялата графика. Между двете червени

# линии хипотетично се намират 95% от остатъците.

# Примери за хетероскедастичност (неконстантна вариация на грешките)

sigmaFunction <- function(x) {

thresholds <- unname(quantile(x, prob = seq(0, 1, by = 0.1)))

thresholds[length(thresholds)] <- thresholds[length(thresholds)] + 0.001

findInterval(x, thresholds)

}

NN <- 400

set.seed(6335)

a <- 0.1; b <- 4

predictions <- 4 + 5\*runif(NN, a, b)

noise <- rnorm(NN, sd = 0.25)

SI <- sigmaFunction(predictions)

r <- cbind(SI, (11 - SI), (1 + 2\*abs(mean(SI) - SI)), (11 - (1 + 2\*abs(mean(SI) - SI))))\*noise

par(mfrow = c(2, 2))

for(i in 1:4) {

plot(predictions, r[, i], xlab = "Predictions", ylab = "Residuals")

#abline(h = 0, col = "red", lwd = 2)

abline(h = 1.96\*c(-1, 1)\*round(sd(r[, i]), 2), col = "red", lty = 4)

}

par(mfrow = c(1, 1))

rm(list = c("a", "b", "i", "noise", "predictions", "r", "SI", "sigmaFunction"))

# На графиката са показани четирите основни типа хетероскедастичност. Между двете червени

# линии хипотетично се намират 95% от остатъците.

# 2. Липса на автокорелация на грешките

# Следващото важно условие е между остатъците да нямаме наличие на автокорелация. Тоест

# всяка следваща грешка да не зависи от предходната грешка. Най-лесно е да проверим с теста

# на Durbin-Watson. Този тест се намира в пакета "lmtest", който трябва да го инсталираме и

# заредим.

# Функцията за теста на Durbin-Watson е dwtest(). Теста приема като параметър самия модел.

# Нулевата хипотеза e, че не съществува автокорелация. Тоест, целта ни е ДА НЕ отхвърлим H0.

install.packages("lmtest")

library(lmtest)

dwtest(model1)

# Стойността на p-value е 0.767 > 0.05. Следователно няма да отхвърлим хипотезата.

# Следователно нямаме автокорелация при грешките.

# Пример за автокорелация при грешките на линеен модел

NN <- 300

set.seed(6621)

x1 <- runif(NN, 1, 5)

noise <- rnorm(NN)

rho <- 0.8

for(i in 2:NN) { noise[i] <- rho\*noise[i-1] + sqrt(1 - rho^2)\*noise[i] }

# Задаваме автокорелация равна на 0.8

y1 <- 3 + 2\*x1 + noise

model4 <- lm(y1 ~ x1)

summary(model4)

dwtest(model4)

# Стойността на p-value e 0, следователно имаме наличие на автокорелация. Както и очаквахме

rm(list = c("i", "model4", "noise", "rho", "x1", "y1"))

# 3. нормално разпределение на грешките

# Последното условие е грешката да има нормално разпределение. Когато това условие е

# изпълнено, тогава имаме най-добрите оценки на коефициентите на линейната регресия. Проверката

# на това условие става с помощта на теста на Shapiro-Wilk (H0: нормално разпределение) и

# Q-Q plot.

shapiro.test(res)

# Стойността на p-value e 0.632 => грешката е нормално разпределена.

qqnorm(res); qqline(res)

# На тази графика търсим за тежки опашки (стойностите в краищата са на голямо разстояние от

# линията). Както се вижда, няма тежки опашки