МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

ЗВІТ

до індивідуального завдання №6

з дисципліни «Моделі статистичного навчання»

Виконав

студент групи ПМіМ-12:

Бордун Михайло

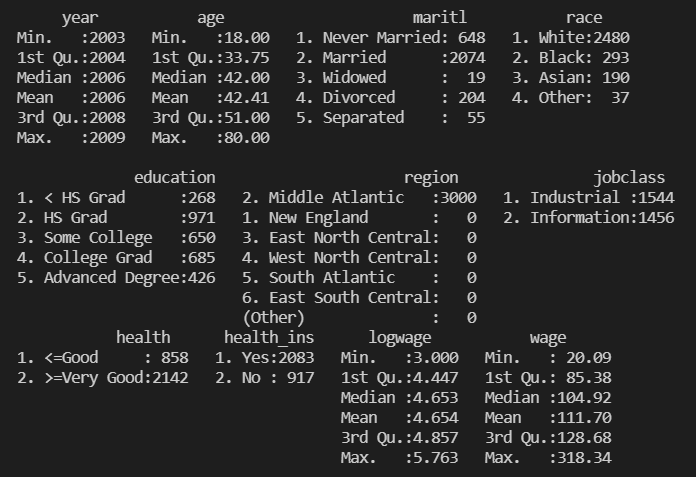
Перевірив:

Проф. Заболоцький Т. М.

Львів – 2021

**Хід виконання**

**1. Додатково проаналізуєте набір даних Wage.**

****

Характеристика даних Wage

**1.1. Використайте поліноміальну регресію для прогнозування wage за age. Використайте перехресну перевірку для вибору оптимального степеня d для полінома. Який степінь було обрано, і як це співвідноситься з результатами перевірки гіпотез з використанням ANOVA? Побудуйте графік отриманого поліному пристосованого до даних.**

deltas = rep(0, 10)

for (i in 1:10) {

    fit = glm(wage ~ poly(age, i), data = Wage)

    deltas[i] = cv.glm(Wage, fit, K = 10)$delta[1]

}

plot(1:10, deltas, xlab = "Degree", ylab = "Test MSE", type = "l")

points(which.min(deltas), deltas[which.min(deltas)],

    col = "blue", cex = 2, pch = 20)

print(paste('Polynom power for min MSE: ', which.min(deltas)))

cat("\n")

fit1 = lm(wage ~ age, data = Wage)

fit2 = lm(wage ~ poly(age, 2), data = Wage)

fit3 = lm(wage ~ poly(age, 3), data = Wage)

fit4 = lm(wage ~ poly(age, 4), data = Wage)

fit5 = lm(wage ~ poly(age, 5), data = Wage)

fit6 = lm(wage ~ poly(age, 6), data = Wage)

fit7 = lm(wage ~ poly(age, 7), data = Wage)

fit8 = lm(wage ~ poly(age, 8), data = Wage)

fit9 = lm(wage ~ poly(age, 9), data = Wage)

fit10 = lm(wage ~ poly(age, 10), data = Wage)

print(anova(fit1, fit2, fit3, fit4, fit5,

 fit6, fit7, fit8, fit9, fit10))

PlotFunc = function(poly\_pow = 0, is\_poly = FALSE, is\_cut = FALSE) {

    plot(wage ~ age, data = Wage, col = "darkgray")

    age.grid = seq(from = range(Wage$age)[1],

     to = range(Wage$age)[2])

    if (is\_poly) {

        fit = lm(wage ~ poly(age, poly\_pow), data = Wage)

        pred = predict(fit, newdata = list(age = age.grid))

        lines(age.grid, pred, col = "blue", lwd = 2)

    }

    if (is\_cut) {

        fit = lm(wage ~ cut(age, 8), data = Wage)

        pred = predict(fit, newdata = list(age = age.grid))

        lines(age.grid, pred, col = "blue", lwd = 2)

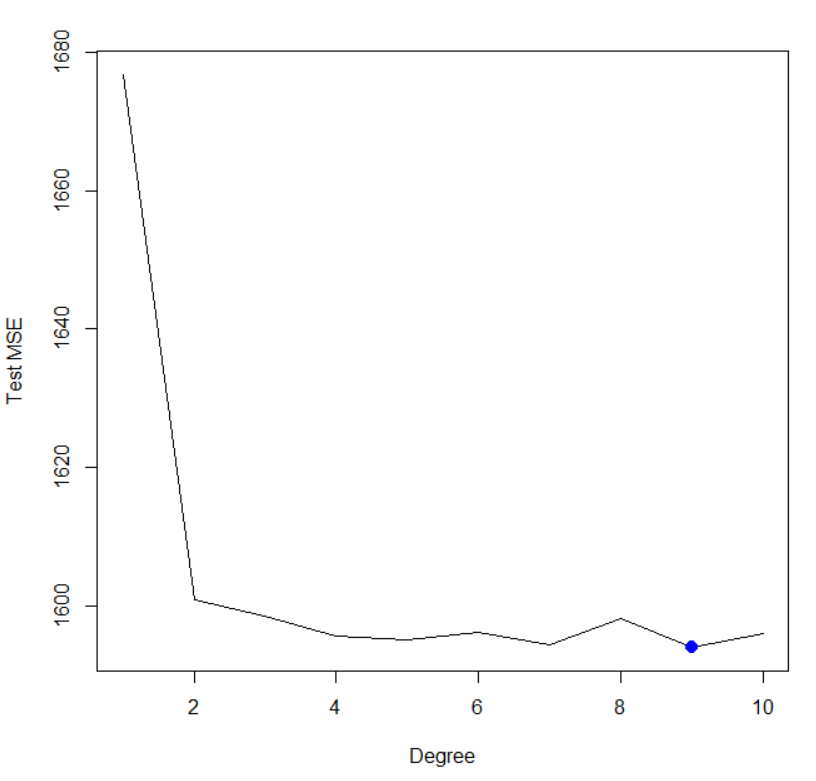
    }

}

par(mfrow = c(1, 2))

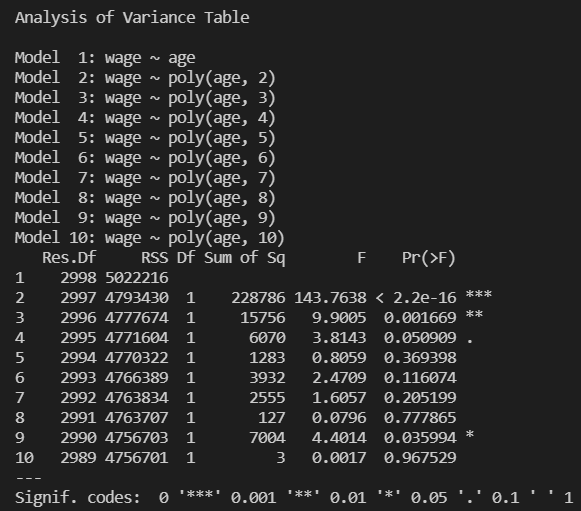
PlotFunc(2, TRUE)

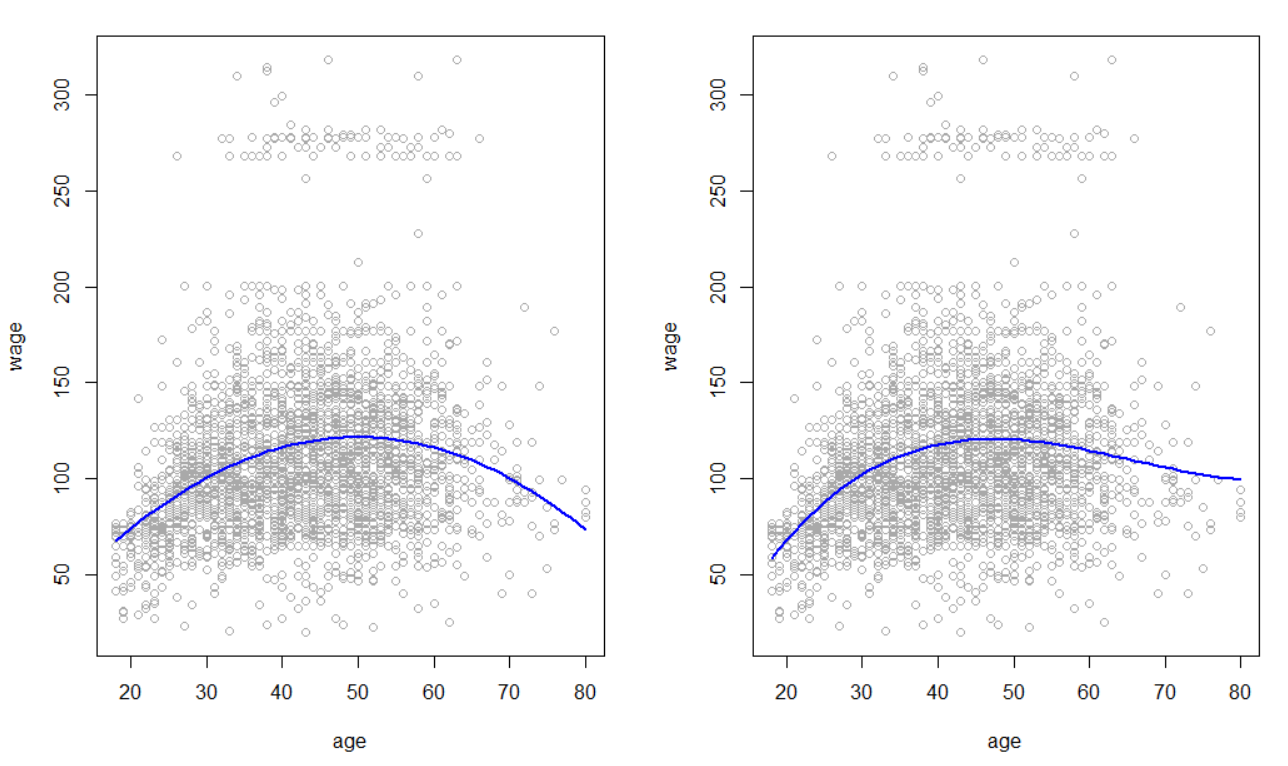
PlotFunc(3, TRUE)

****

****

З огляду на результат перехресної перевірки із значенням k=10 бачимо, що при d=9 в нас є оптимальний степінь для полінома в сенсі мінімальної тестової помилки.

****

****

З результатів перевірки гіпотез з використанням ANOVA видно, що зважаючи на p-значення, ми можемо побачити, що поліном 2 і 3 степенів забезпечує найкращий результат. Проте варто зауважити, що поліном 9-го степеня також має p-значення нижче за 0.05, що також демонструє хорошу кореляцію між залежною змінною.

**1.2. Використайте східчасту функцію для прогнозування wage за age та проведіть перехресну перевірку для вибору оптимальної кількості розрізів. Побудуйте графік з отриманими результатами.**

par(mfrow = c(1, 1))

cvs = rep(Inf, 10)

for (i in 2:10) {

    Wage$age.cut = cut(Wage$age, i)

    fit = glm(wage ~ age.cut, data = Wage)

    cvs[i] = cv.glm(Wage, fit, K = 10)$delta[1]

}

cat("\n")

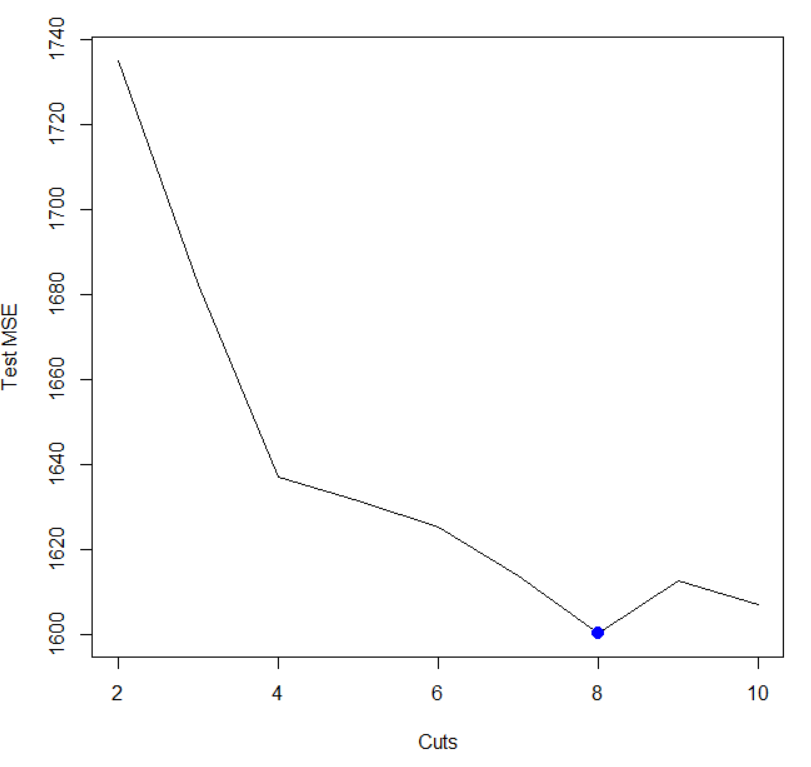
plot(2:10, cvs[-1], xlab = "Cuts", ylab = "Test MSE", type = "l")

points(which.min(cvs), cvs[which.min(cvs)],

    col = "blue", cex = 2, pch = 20)

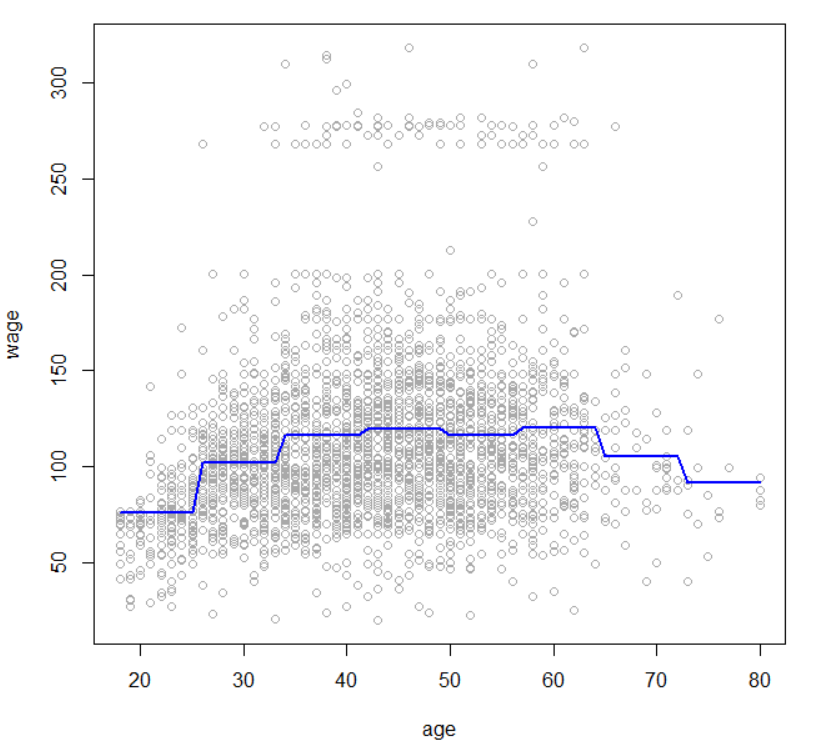
print(paste('Number of cuts for min MSE: ', which.min(cvs)))

PlotFunc(, , TRUE)

****

****

Результат показує, що тестова помилка буде мінімальною з використанням східчастої функції з 8-ма зрізами.

****

**2. Набір даних Wage містить інші змінні такі як, сімейний стан (maritl), робочий клас (jobclass) тощо. Дослідіть зв’язки між деякими з цих інших предикторів та wage, а також використовуючи нелінійні методи пристосуйте гнучкі моделі до даних. Побудуйте графіки отриманих результатів, та підсумуйте свої висновки.**

par(mfrow = c(1, 2))

plot(Wage$maritl, Wage$wage,  xlab='maritl', ylab='wage')

plot(Wage$jobclass, Wage$wage,  xlab='jobclass', ylab='wage')

library(gam)

fit0 = gam(wage ~ lo(year, span = 0.6) +

s(age, df = 2) + education, data = Wage)

fit1 = gam(wage ~ lo(year, span = 0.6) +

s(age, df = 2) + education + jobclass, data = Wage)

fit2 = gam(wage ~ lo(year, span = 0.6) +

s(age, df = 2) + education + maritl, data = Wage)

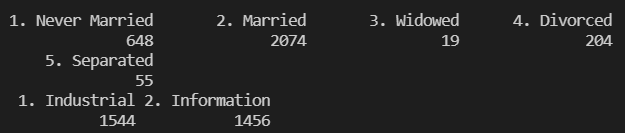
fit3 = gam(wage ~ lo(year, span = 0.6) +

s(age, df = 2) + education + jobclass + maritl, data = Wage)

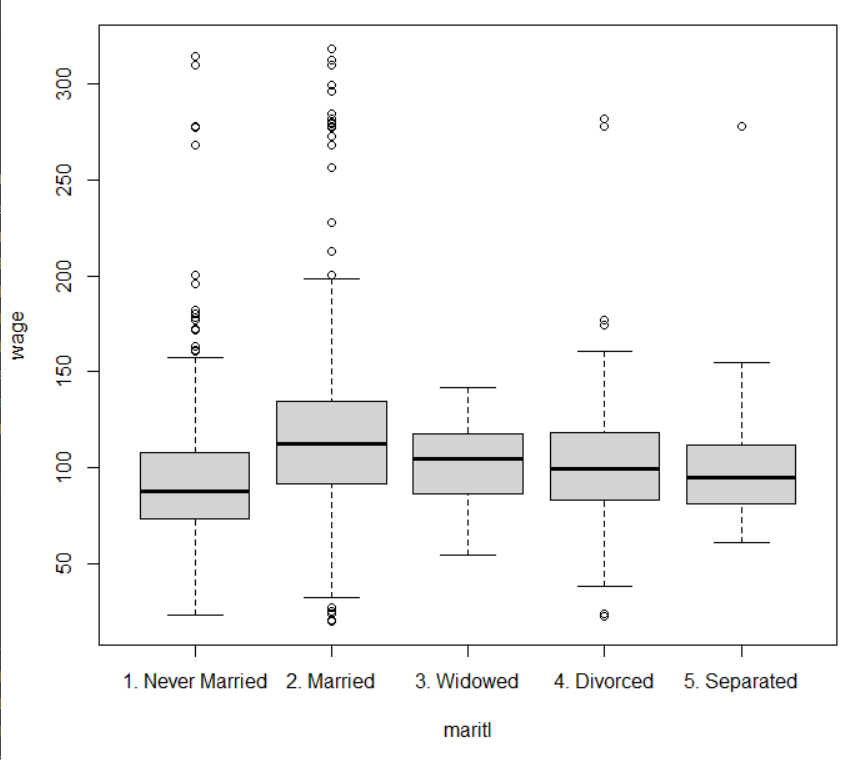
print(anova(fit0, fit1, fit2, fit3))

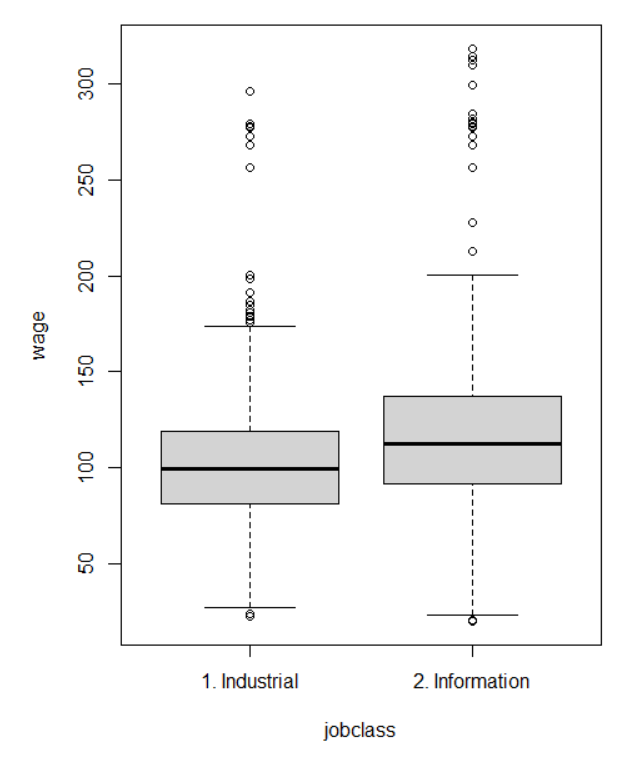
par(mfrow = c(2, 2))

plot(fit2, se = T, col = "blue")

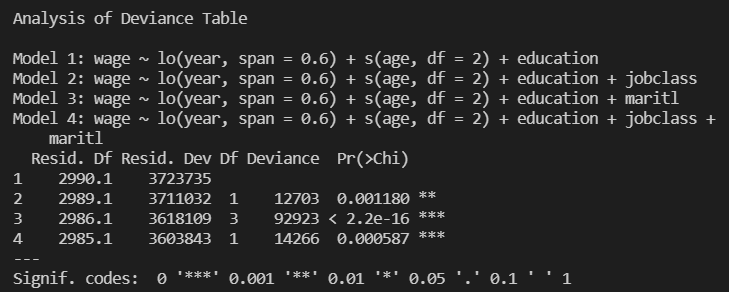
****

Загальна характеристика якісних змінних maritl та jobclass

****

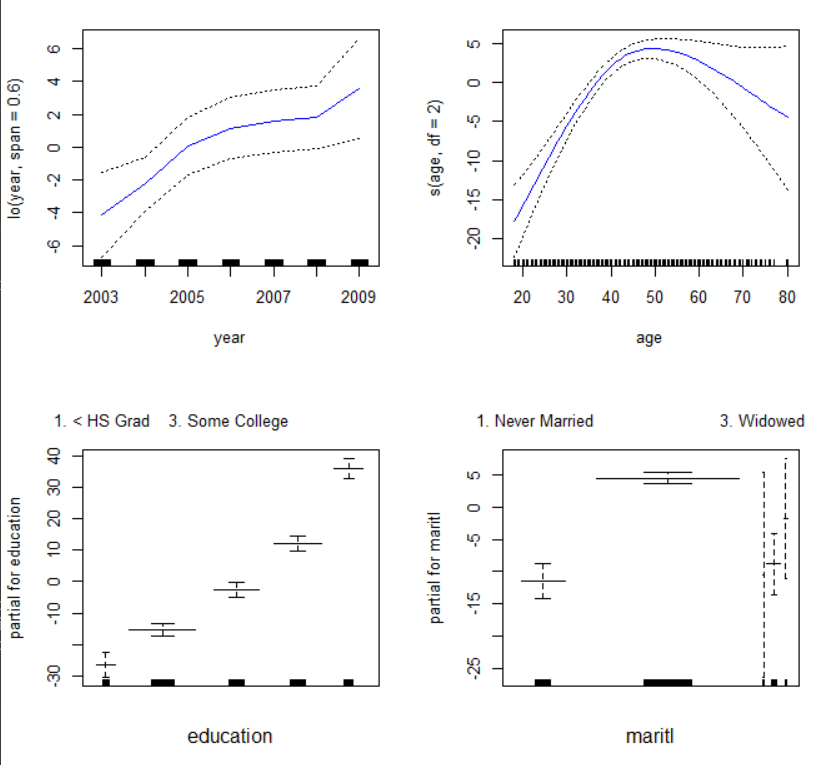
****

З огляду на візуально представлені зв’язки між змінними maritl, jobclass та залежною wage, бачимо, що подружжя в середньому заробляє більше грошей ніж решта категорій, а також, що працівник в інформаційній сфері в середньому заробляє більше ніж в індустріальній.

****

Використавши різні нелінійні методи було пристосовано гнучкі моделі до даних і зроблено перевірку гіпотез з використанням ANOVA.

Для проектування моделей було використано згладжувальні сплайни з функцією s(), з бібліотеки gam. Вказуємо всюди, що функція від age повинна мати 2 ступеня свободи. Також використано всюди локальну регресію для змінної year з інтервалом 0.6. Різниця між моделями полягає у використанні різних додаткових якісних змінних для прогнозування нашої залежної змінної wage.

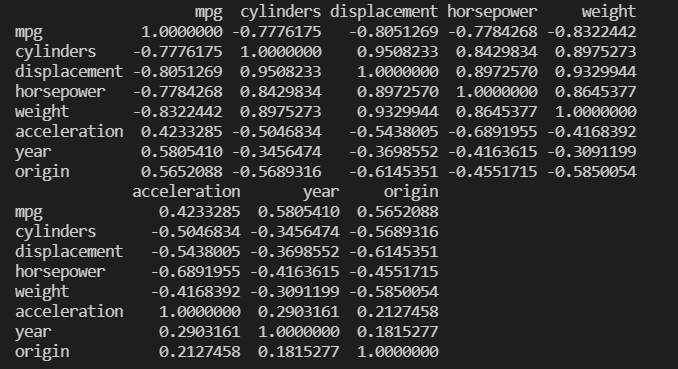


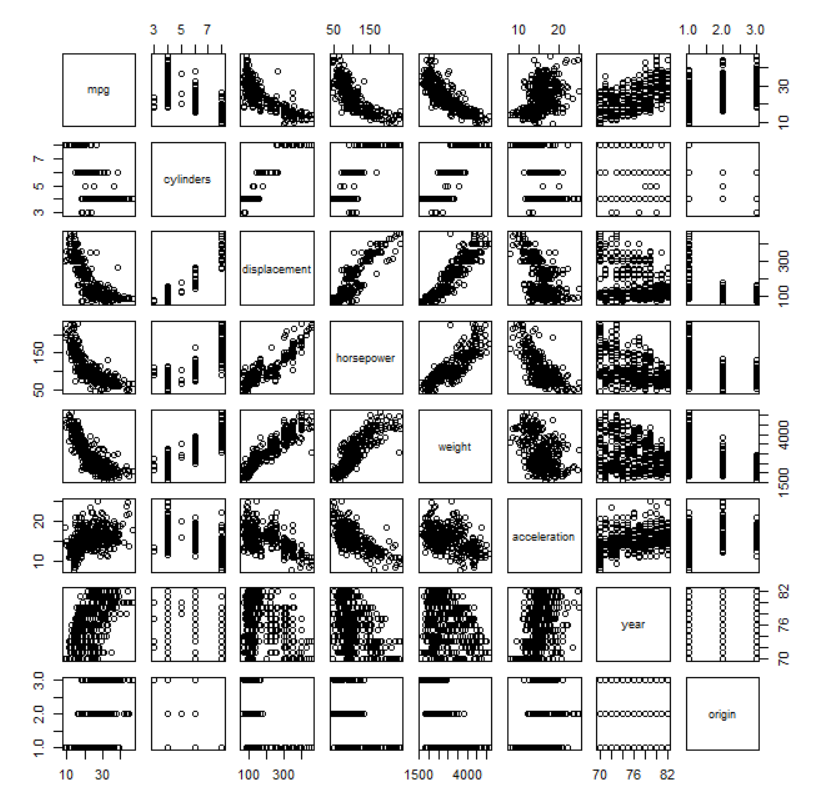
У висновку бачимо, що для прогнозування найкраще підходить третя модель з використанням двох якісних змінних education та maritl.

**3. Пристосуйте деякі нелінійні моделі на наборі даних Auto. Чи є якісь докази нелінійних взаємозв’язків в цьому наборі даних? Побудуйте кілька інформативних графіків, щоб обґрунтувати свою відповідь.**

Дослідимо змінні та їх залежності з датасету Auto.Для початку використано функцію cor(), щоб побачити настільки сильна чи слаба кореляція між змінними.

А також за допомогою функції pairs() виведено графічну залежність всіх змінних.

****

****

Матриця діаграм розсіювання для даних Auto

Розглядаючи mpg як залежну змінну бачимо, що вона має досить сильну залежність між змінними cylinders, displacement, horsepower та weight, але тут варто наголосити що це є від’ємна кореляція.

deltas = rep(0, 15)

for (i in 1:15) {

    fit = glm(mpg ~ poly(displacement, i), data = autos)

    deltas[i] = cv.glm(autos, fit, K = 10)$delta[1]

}

plot(1:15, deltas, xlab = "Degree", ylab = "Test MSE", type = "l")

points(which.min(deltas), deltas[which.min(deltas)],

 col = "blue", cex = 2, pch = 20)

print(paste('Polynom power for min MSE: ', which.min(deltas)))

cat("\n")

cvs = rep(Inf, 10)

for (i in 2:10) {

    autos$dis.cut = cut(displacement, i)

    fit = glm(mpg ~ dis.cut, data = autos)

    cvs[i] = cv.glm(autos, fit, K = 10)$delta[1]

}

plot(2:10, cvs[-1], xlab = "Cuts", ylab = "Test MSE", type = "l")

points(which.min(cvs), cvs[which.min(cvs)],

 col = "blue", cex = 2, pch = 20)

print(paste('Number of cuts for min MSE: ', which.min(cvs)))

cat("\n")

cvs = rep(Inf, 10)

for (i in 3:10) {

    fit = glm(mpg ~ ns(displacement, df = i), data = autos)

    cvs[i] = cv.glm(autos, fit, K = 10)$delta[1]

}

plot(3:10, cvs[-c(1, 2)], xlab = "Degree", ylab = "Test MSE", type = "l")

points(which.min(cvs), cvs[which.min(cvs)],

col = "blue", cex = 2, pch = 20)

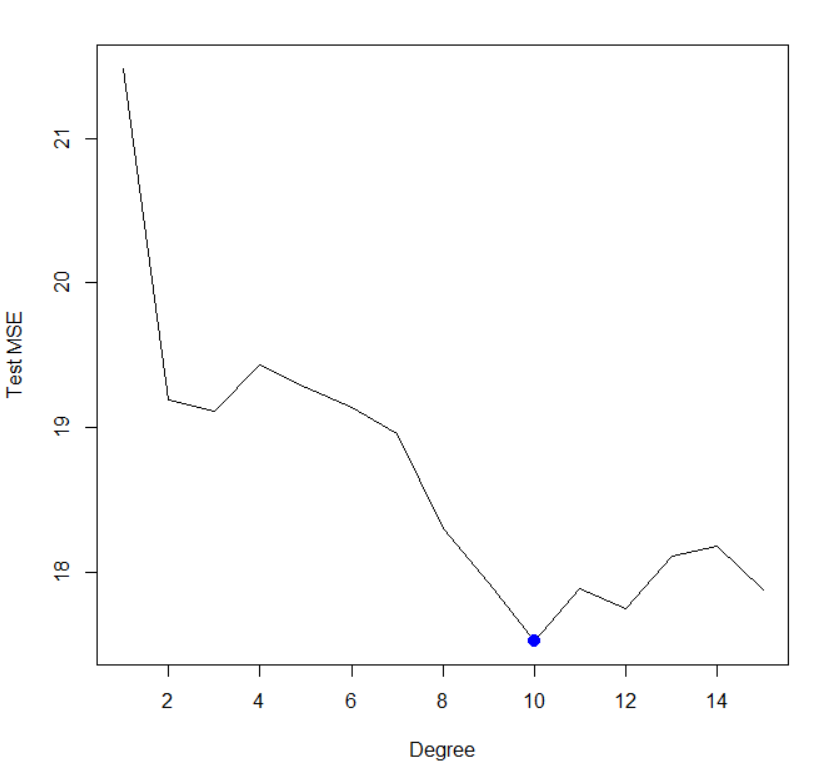
print(paste('Degrees of freedom for min MSE [splines]: ', which.min(cvs)))

cat("\n")

fit = gam(mpg ~ s(displacement, df = 2) +

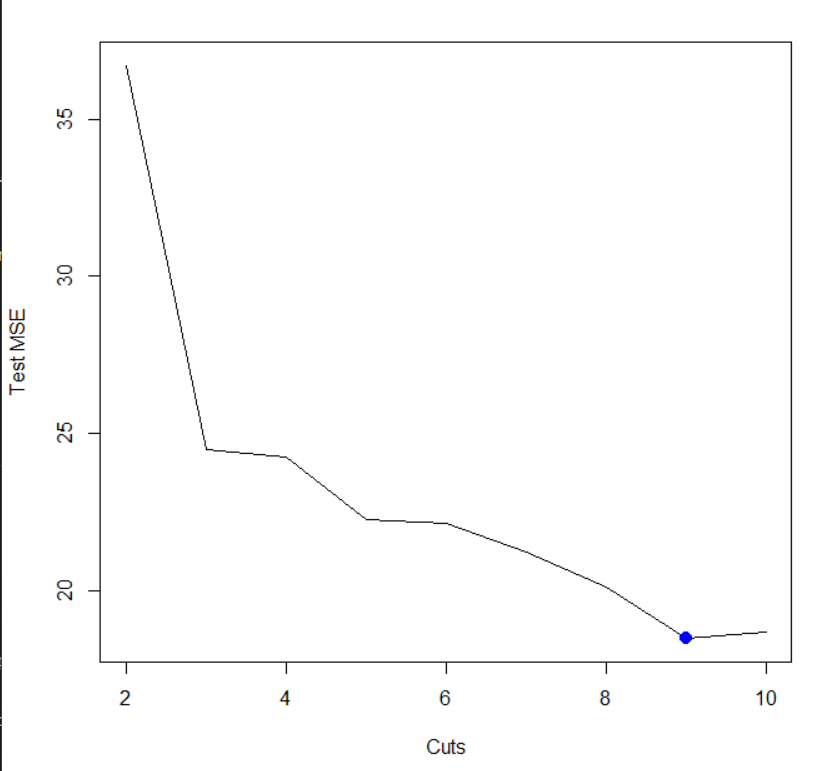
 s(horsepower, df = 2), data = autos)

print(summary(fit))

****

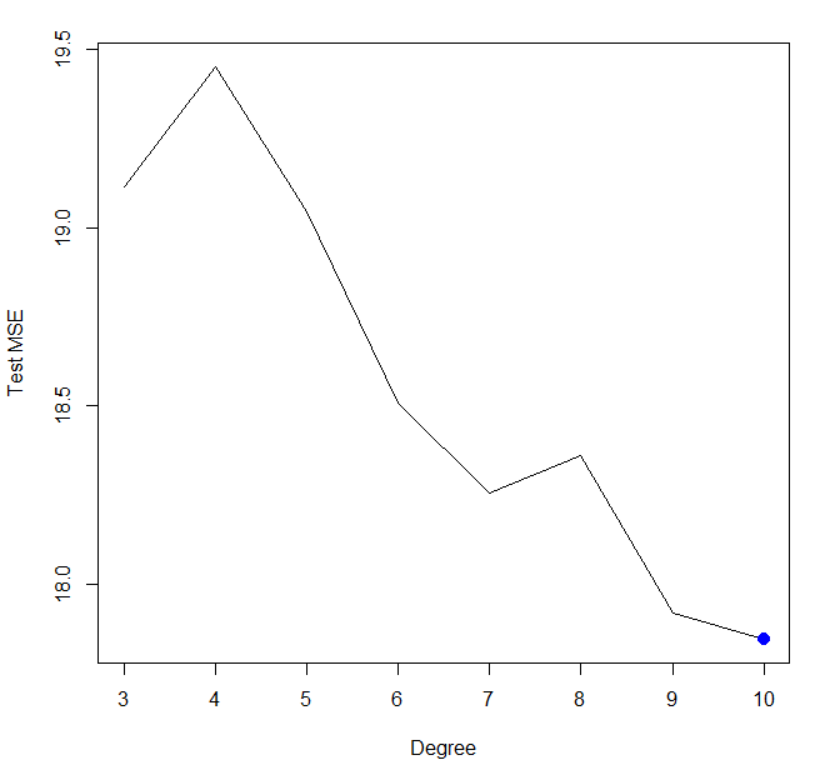
****

З огляду на графік наведений нижче можемо побачити, що d=10 є оптимальним степенем для полінома в сенсі тестової помилки.

****

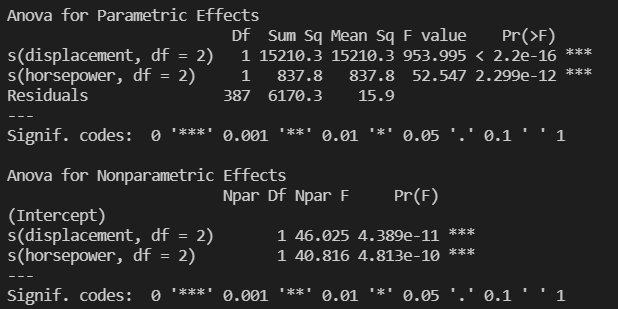
****

Зважаючи на графік наведений нижче можемо побачити, що мінімальна тестова помилка буде досягатися для східчастої функції з 9-ма зрізами.

****

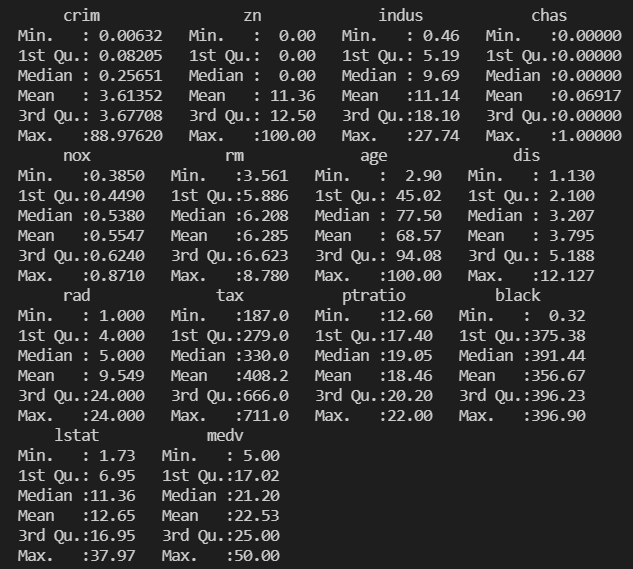
****

Зважаючи на графік наведений нижче, можемо побачити, що тестова похибка мінімальна для 10 ступенів свободи з використанням природного сплайну для змінної displacement.

****

Для проектування УАМ моделі було використано згладжувальні сплайни з функцією s(). Вказуємо, що функція від displacement та horsepower повинна мати 2 ступеня свободи. І загалом модель показує, що всі обидві змінні є значущі для моделювання mpg.

**4. Використаємо змінні dis (зважене середнє відстані до п'яти центрів зайнятості в Бостоні) та nox (концентрація оксидів азоту у частинах на 10 мільйонів) з набору даних Boston. Розглянемо dis як предиктор та nox як залежну змінну.**

****

Характеристика даних Boston

**4.1. Використовуючи функцію poly(), встановіть кубічну поліноміальну регресію для передбачення nox за допомогою dis. Опишіть результати регресії та побудуйте графік даних та поліноміальної регресії.**

fit = lm(nox ~ poly(dis, 3), data = Boston)

print(summary(fit))

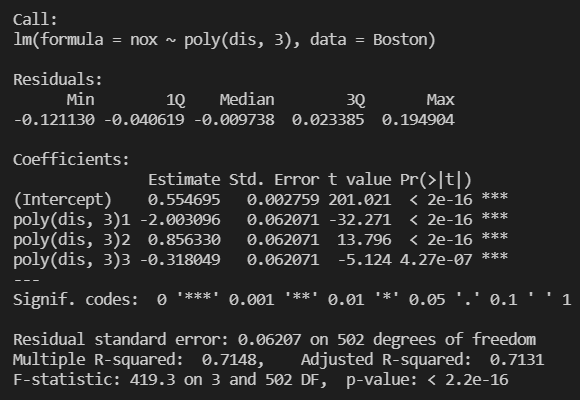
dis.grid = seq(from = range(Boston$dis)[1],

 to = range(Boston$dis)[2], by = 0.1)

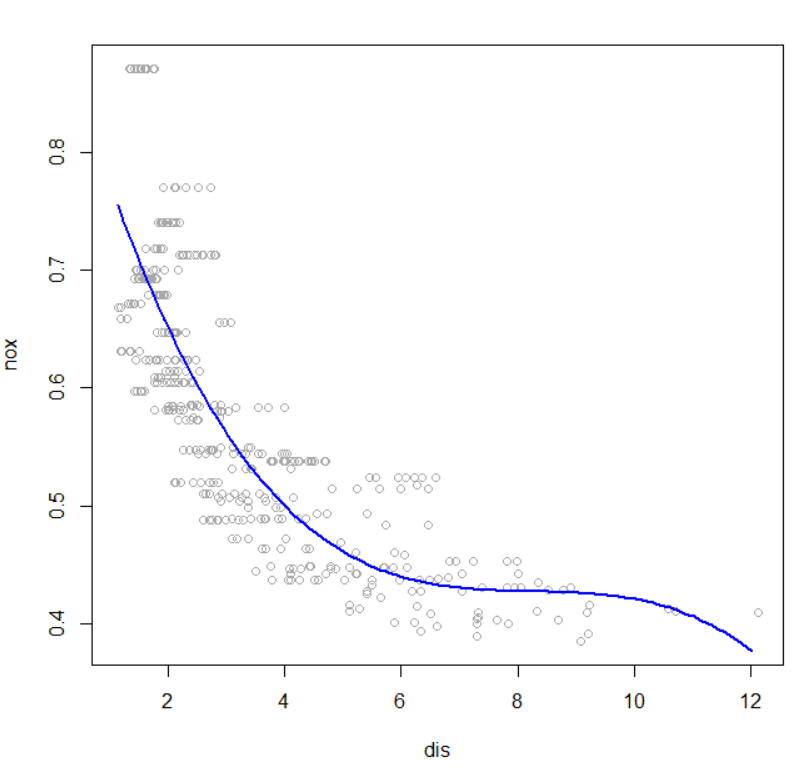
preds = predict(fit, list(dis = dis.grid))

plot(nox ~ dis, data = Boston, col = "darkgrey")

lines(dis.grid, preds, col = "blue", lwd = 2)

****

Результат аналізу нашої регресії показав, що всі степені кубічного поліному є значущими для залежної змінної nox.

****

З графіка даних бачимо обернену залежність наших змінних та графік поліноміальної регресії.

**4.2. Побудуйте поліноміальні моделі для різних степенів (скажімо, від 1 до 10), і наведіть їхні RSS.**

rss = rep(0, 10)

for (i in 1:10) {

    fit = lm(nox ~ poly(dis, i), data = Boston)

    rss[i] = sum(fit$residuals^2)

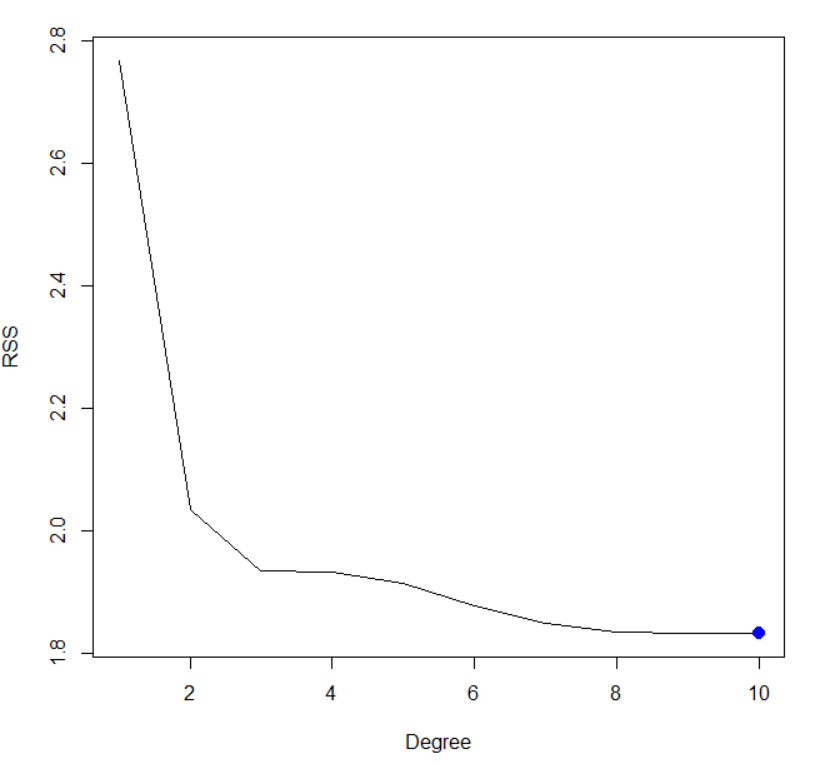
}

plot(1:10, rss, xlab = "Degree", ylab = "RSS", type = "l")

points(which.min(rss), rss[which.min(rss)],

 col = "blue", cex = 2, pch = 20)

print(paste('Polynom power for min RSS: ', which.min(rss)))

****

****

Можна побачити, що RSS (residual sum of squares) зменшується зі збільшенням степеня полінома, відповідно для 10-го степеня RSS буде мінімальним.

**4.3. Використайте перехресну перевірку або інший підхід для вибору оптимального степеня для поліноміальної регресії та поясніть отримані результати.**

deltas = rep(0, 10)

for (i in 1:10) {

    fit = glm(nox ~ poly(dis, i), data = Boston)

    deltas[i] = cv.glm(Boston, fit, K = 10)$delta[1]

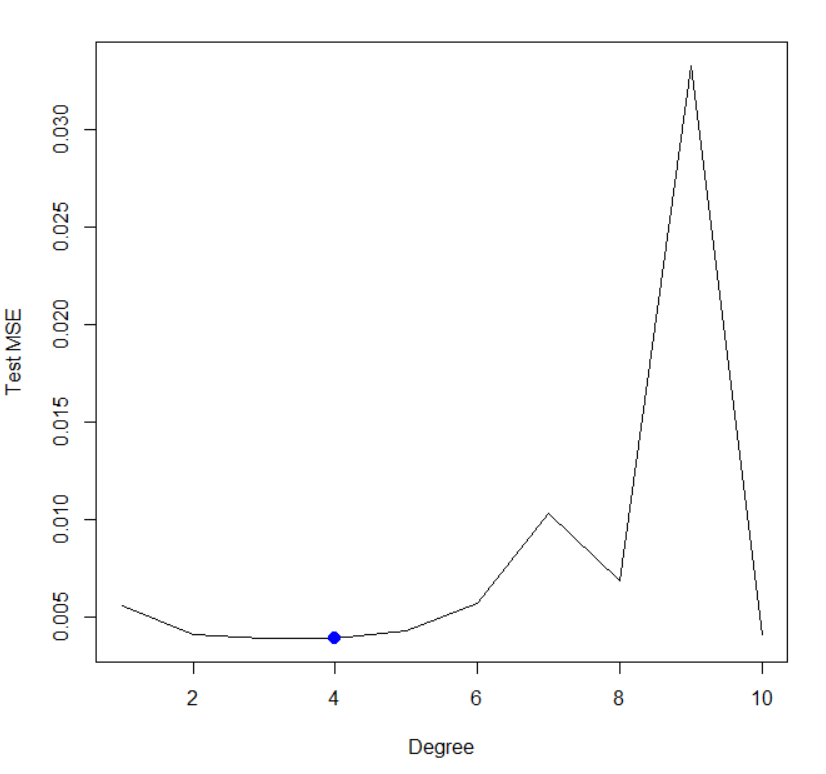
}

plot(1:10, deltas, xlab = "Degree", ylab = "Test MSE", type = "l")

points(which.min(deltas), deltas[which.min(deltas)],

 col = "blue", cex = 2, pch = 20)

print(paste('Polynom power for min MSE: ', which.min(deltas)))

****

****

З результатів перехресної перевірки найменше MSE досягається при степені полінома чотири і досить різко зростає для 9-го степеня.

**4.4. Використовуючи функцію bs (), пристосуйте сплайн регресію для прогнозування nox за допомогою dis. Опишіть результати отримані з використанням чотирьох ступенів свободи. Як ви вибрали вузли? Побудуйте графік отриманої моделі.**

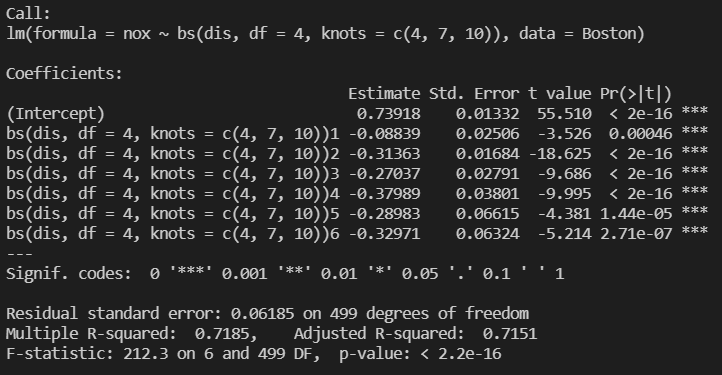
fit = lm(nox ~ bs(dis, df = 4, knots = c(4, 7, 10)), data = Boston)

print(summary(fit))

pred = predict(fit, list(dis = dis.grid))

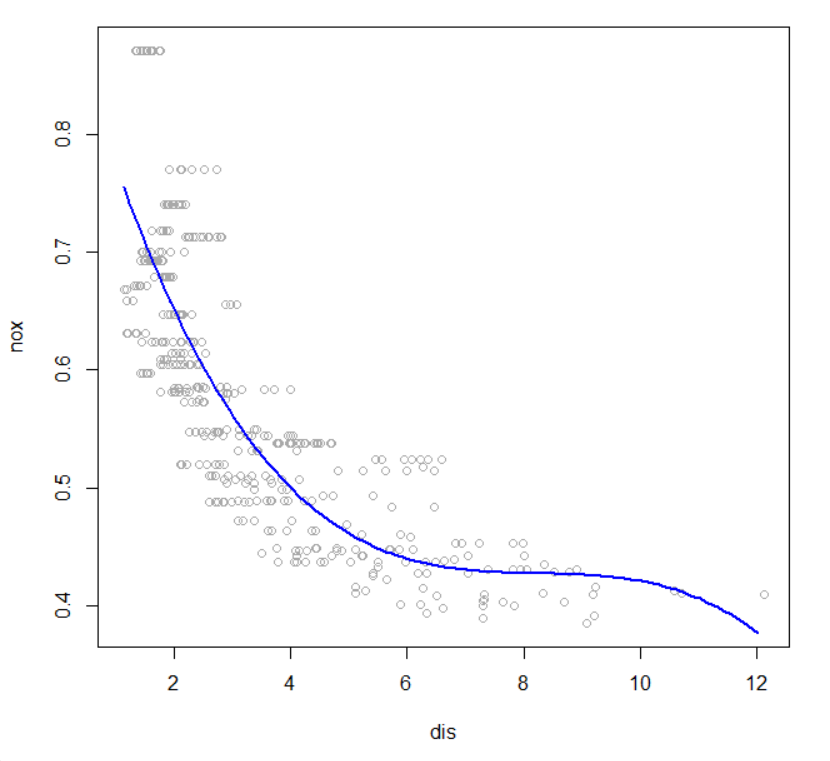
plot(nox ~ dis, data = Boston, col = "darkgrey")

lines(dis.grid, preds, col = "blue", lwd = 2)

****

Для того, щоб застосувати регресійні сплайни, ми використовуємо бібліотеку splines. Функція bs() генерує всю матрицю базисних функцій для сплайнів із заданим набором вузлів. Ступінь свободи вказаний як 4, а вузли у dis 4, 7 і 10, що забезпечувало найкращі результати. Також варто додати, що в нас використовується кубічний сплайн.

Результати показують, що всі базові функції є значущі у сплайні, при чому p-значення всюди є нижче ніж 0.001.

****

**4.5. Пристосуйте сплайн регресію для діапазону ступенів свободи, і побудуйте графік результатів. Наведіть відповідні RSS. Опишіть отримані результати.**

rss = rep(Inf, 15)

for (i in 3:15) {

    fit = lm(nox ~ bs(dis, df = i), data = Boston)

    rss[i] = sum(fit$residuals^2)

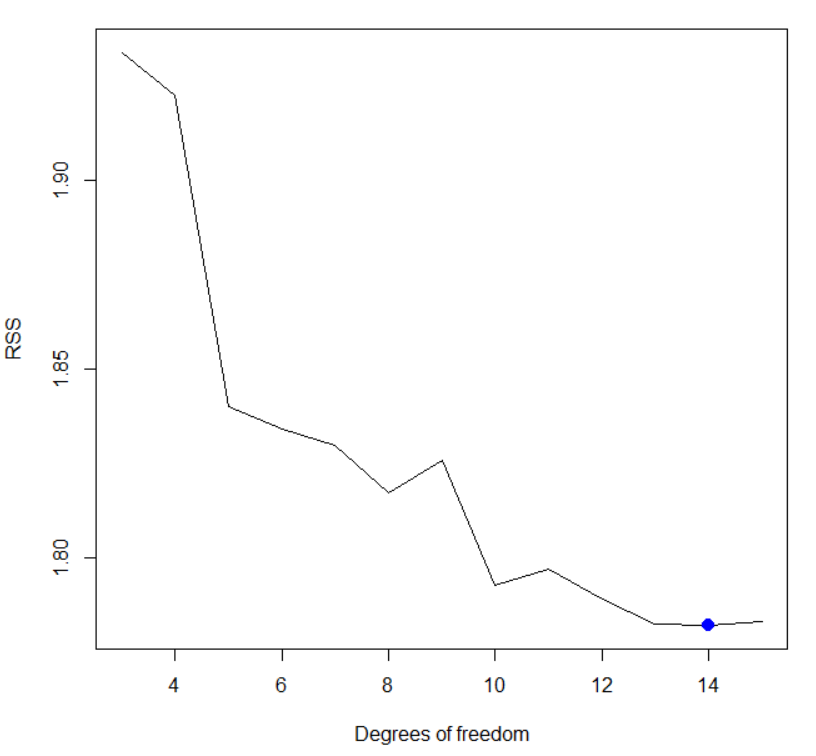
}

plot(3:15, rss[-c(1, 2)], xlab = "Degrees of freedom", ylab = "RSS", type = "l")

points(which.min(rss), rss[which.min(rss)],

 col = "blue", cex = 2, pch = 20)

print(paste('Degrees of freedom for min RSS: ', which.min(rss)))

****

****

Можемо зауважити, що RSS спадає до 14 ступеня свободи, а потім починає зростати.

**4.6. Використайте перехресну перевірку або інший підхід, щоб вибрати найкращий ступінь свободи для сплайн регресії на цих даних. Опишіть свої результати.**

cv = rep(Inf, 15)

for (i in 3:15) {

    fit = glm(nox ~ bs(dis, df = i), data = Boston)

    cv[i] = cv.glm(Boston, fit, K = 10)$delta[1]

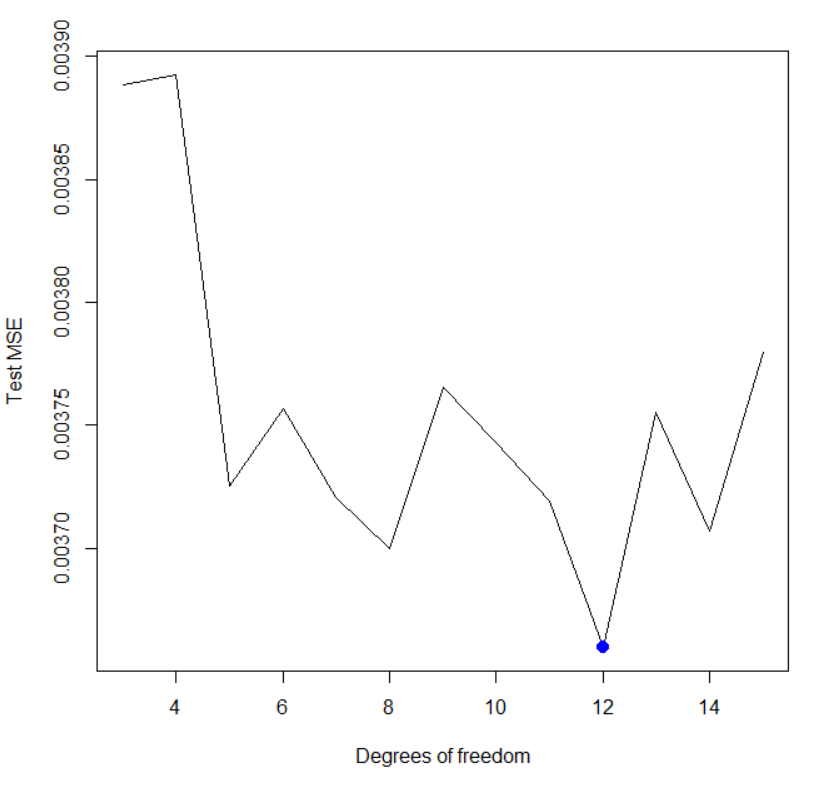
}

plot(3:15, cv[-c(1, 2)], xlab = "Degrees of freedom", ylab = "Test MSE", type = "l")

points(which.min(cv), cv[which.min(cv)],

 col = "blue", cex = 2, pch = 20)

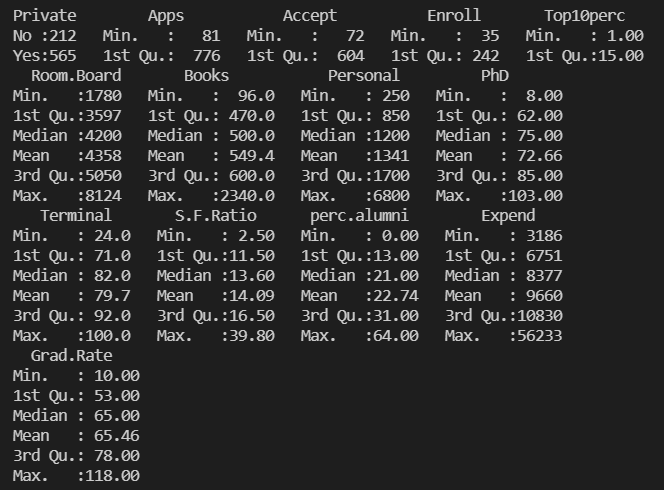
print(paste('Degrees of freedom for min MSE: ', which.min(cv)))

****

****

Результати перехресної перевірки із значенням k=10 показують, що тестова помилка мінімальна для 12-ти ступенів свободи для сплайн регресії. Для більших степенів вона зростає.

**5. Використайте набір даних College.**

****

Характеристика даних College

**5.1. Розбийте дані на навчальний та тестовий набори. Використайте Outstate як залежну змінну, а інші змінні як предиктори. Виконайте покроковий вибір вперед на навчальному наборі, щоб визначити задовільну модель, яка використовує лише підмножину предикторів.**

train = sample(length(Outstate), length(Outstate) / 2)

College.train = College[train, ]

College.test = College[-train, ]

fit = regsubsets(Outstate ~ ., data = College.train,

  nvmax = 14, method = "forward")

fit.summary = summary(fit)

par(mfrow = c(1, 3))

plot(fit.summary$cp, xlab = "Number of variables", ylab = "Cp", type = "l")

points(which.min(fit.summary$cp), fit.summary$cp[which.min(fit.summary$cp)],

col = "blue", cex = 2, pch = 20)

plot(fit.summary$bic, xlab = "Number of variables", ylab = "BIC", type='l')

points(which.min(fit.summary$bic), fit.summary$bic[which.min(fit.summary$bic)],

col = "blue", cex = 2, pch = 20)

plot(fit.summary$adjr2, xlab = "Number of variables", ylab = "Adjusted R2",

 type = "l", ylim = c(0.4, 0.84))

points(which.max(fit.summary$adjr2), fit.summary$adjr2[which.max(fit.summary$adjr2)],

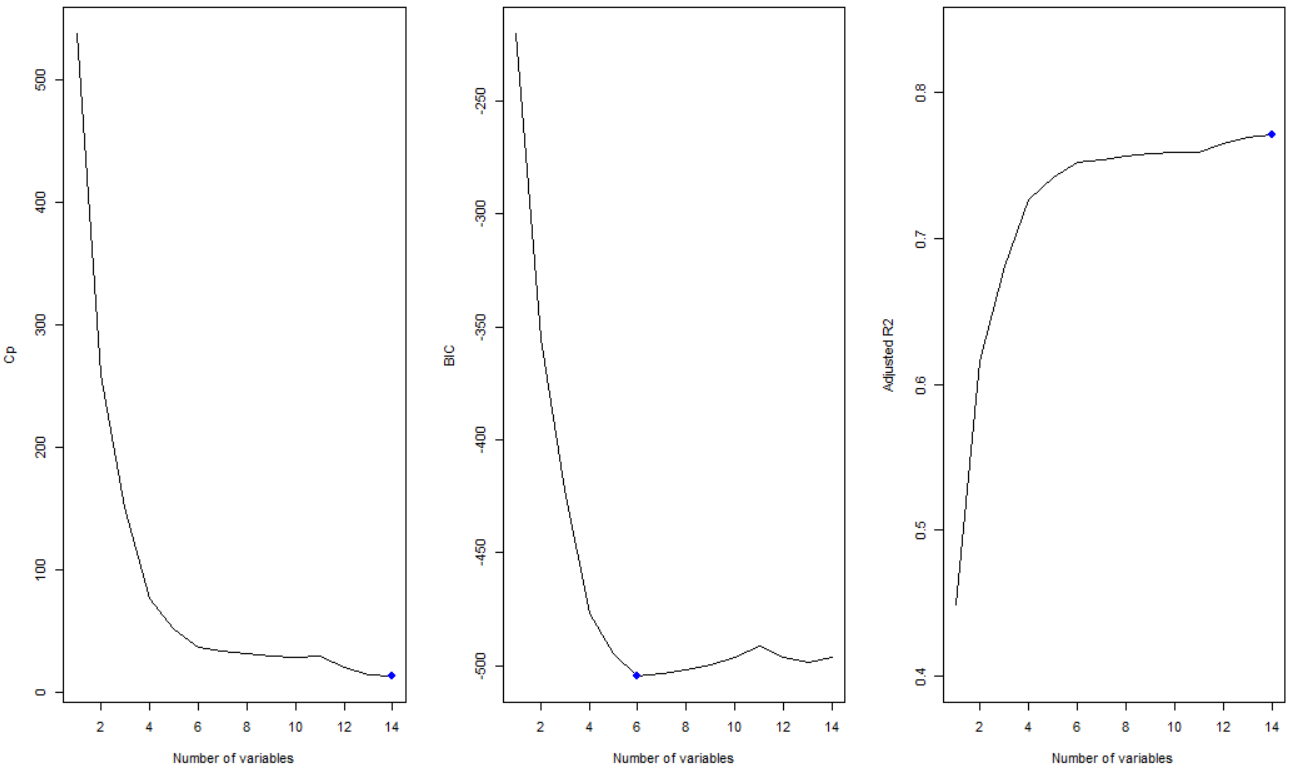
col = "blue", cex = 2, pch = 20)

fit = regsubsets(Outstate ~ ., data = College, method = "forward")

coeffs = coef(fit, id = 6)

cat("\n")

print(names(coeffs))

****

Як можна бачити з графіків методи Cp, BIC і скорегований показують різні результати, проте BIC якраз визначив на навчальному наборі задовільну модель, яка використовує лише 6 предикторів, що і буде мінімальним розміром підножини предикторів.

****

**5.2. Оцініть УАМ модель на навчальних даних, використовуючи Outstate як залежну змінну та ознаки обрані на попередньому кроці як предиктори. Побудуйте графік результатів та поясніть свої висновки.**

fit = gam(Outstate ~ Private + s(Room.Board, df = 2) +

s(PhD, df = 2) + s(perc.alumni, df = 2) +

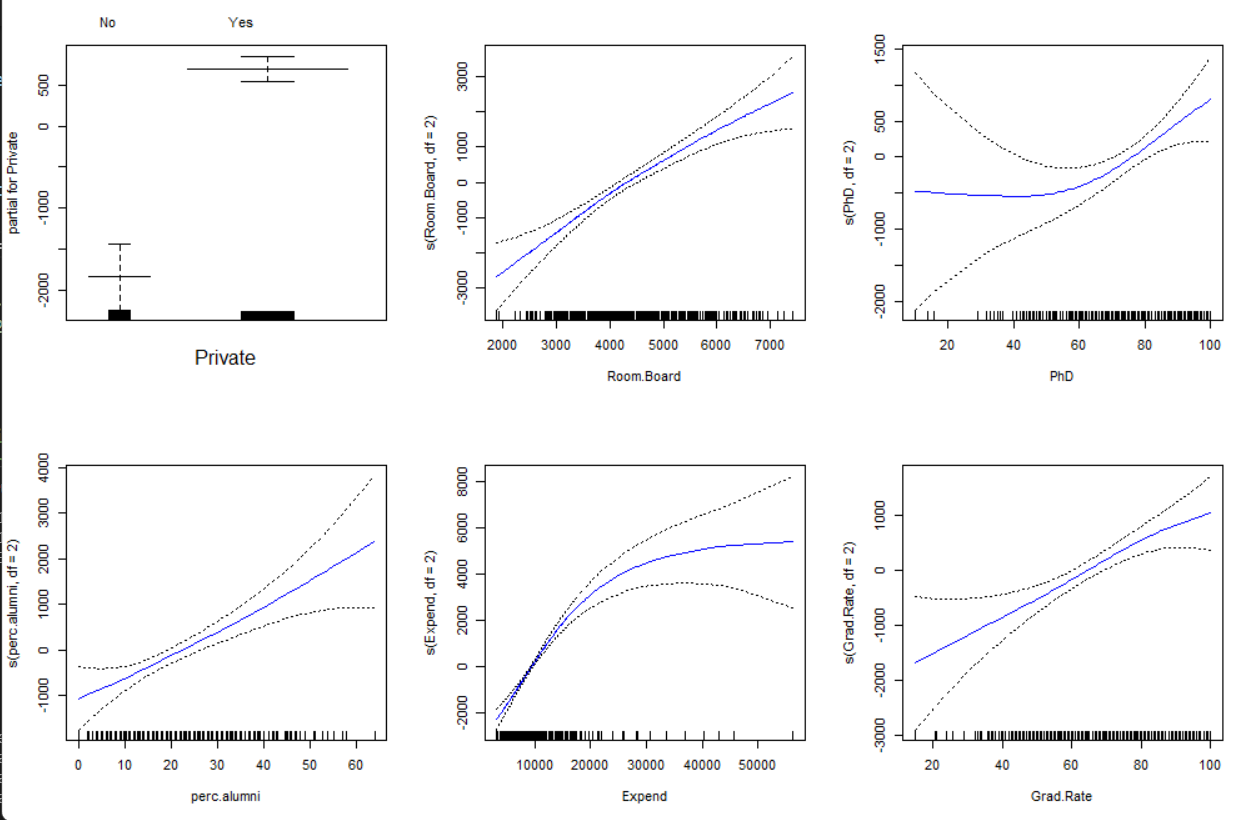
 s(Expend, df = 2) +

  s(Grad.Rate, df = 2),

  data=College.train)

par(mfrow = c(2, 3))

plot(fit, se = T, col = "blue")

****

З графіка бачимо пряму залежність змінних та згладжувальних сплайнів, які використовувалися для прогнозування змінної Outstate.

**5.3. Застосуйте модель на тестовому наборі даних, поясніть отримані результати.**

preds = predict(fit, College.test)

err = mean((College.test$Outstate - preds)^2)

print(paste("Test error :", round(err, 2)))

tss = mean((College.test$Outstate - mean(College.test$Outstate))^2)

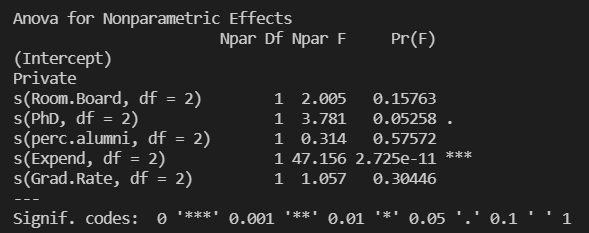
rss = 1 - err / tss

print(paste("RSS :", round(rss, 2)))

****

Ми отримуємо 0,766 для тестових даних, використовуючи УАМ з 6 предикторами.

**5.4. Для яких змінних, якщо такі є, є докази нелінійності взаємозв'язку з залежною змінною?**

****

ANOVA показує чіткий нелінійний зв’язок між Outstate та Expend. Також бачимо слабкий нелінійний зв’язок між Outstate та PhD.

**6. УАМ, як правило, оцінюють на основі методу підгонки. Розглянемо цей метод на основі множинної лінійної регресії. Припустимо, що ми хочемо використати множинну лінійну регресію, але у нас немає програмного забезпечення для цього. Натомість ми маємо лише програмне забезпечення для оцінки простої лінійної регресії. Тому ми використаємо наступний ітераційний підхід: ми фіксуємо всі оцінки коефіцієнтів, крім одного на поточному значенні та оновлюємо лише одну оцінку коефіцієнта за допомогою простої лінійної регресії. Процес продовжується поки не досягнеться збіжність, тобто поки оцінки коефіцієнтів не перестануть змінюватися. Застосуємо цей підхід на штучному прикладі.**

**6.1. Згенеруйте залежну змінну *Y* і два предиктори *X*1 і *X*2, з *n* = 100.**

set.seed(1)

X1 = rnorm(100)

X2 = rnorm(100)

eps = rnorm(100, sd = 0.1)

betas = runif(3, min=-5, max=5)

cat("\n")

print("Beta-values list:")

print(betas)

Y = betas[1] + betas[2] \* X1 +

 betas[3] \* X2 + eps

Як можна побачити визначення значень *β*i відбувається випадково (seed = 1), в межах від -5 до 5 будь-які дробові числа.

****

**6.2. Ініціалізуйте оцінку *β*1 довільним значенням на свій вибір.**

beta0 = rep(0, 1000)

beta1 = rep(0, 1000)

beta2 = rep(0, 1000)

beta1[1] = runif(1, min=-5, max=5)

****

**6.3. Не змінюючи *β*1 оцініть модель**

***Y –* *β*1*X*1 = *β*0 + *β*2*X*2 + *ε.***

**Це можна зробити наступним чином:**

**> a=y-beta1 \*x1**

**> beta2=lm(a*∼*x2)$coef [2]**

**6.4. Зафіксувавши оцінку *β*2, оцініть модель**

***Y −* *β*2*X*2 = *β*0 + *β*1*X*1 + *ε.***

**Це можна зробити наступним чином:**

**> a=y-beta2 \*x2**

**> beta1=lm(a*∼*x1)$coef [2]**

**6.5. Використайте for для організації циклу з повторень кроків 6.3 та 6.4 1,000 разів. Наведіть оцінки параметрів *β*0, *β*1 і *β*2 на кожній ітерації. Побудуйте графіки на яких відображено ці значення для *β*0, *β*1 і *β*2 різними кольорами.**

for (i in 1:1000) {

    a = Y - beta1[i] \* X1

    beta2[i] = lm(a ~ X2)$coef[2]

    a = Y - beta2[i] \* X2

    lm.fit = lm(a ~ X1)

    if (i < 1000) {

        beta1[i + 1] = lm.fit$coef[2]

    }

    beta0[i] = lm.fit$coef[1]

}

plot(1:1000, beta0, type = "l", xlab = "iteration", ylab = "betas",

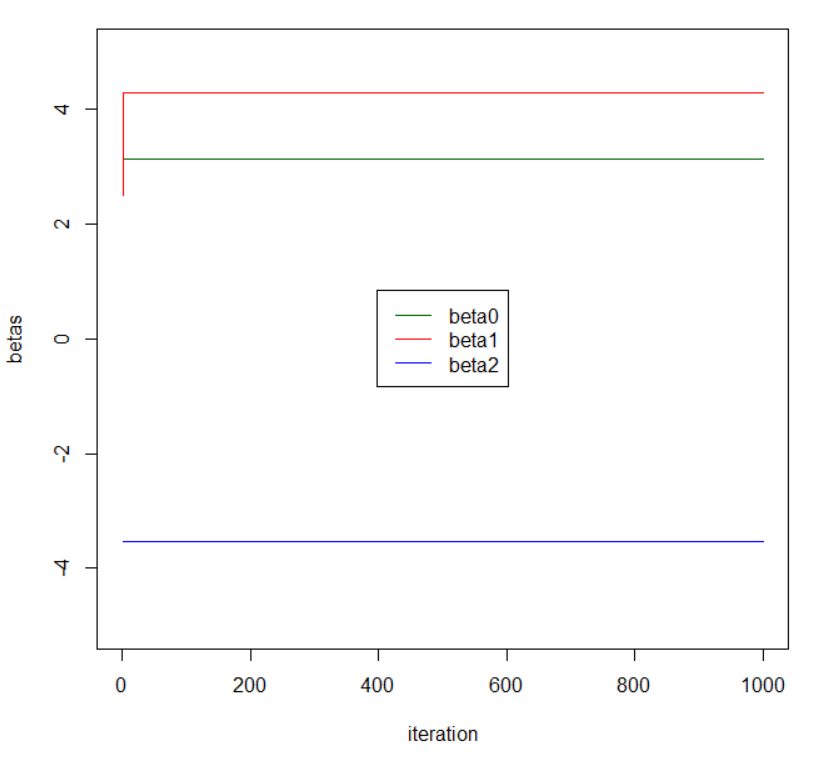
    ylim = c(-5, 5), col = "darkgreen")

lines(1:1000, beta1, col = "red")

lines(1:1000, beta2, col = "blue")

legend("center", c("beta0", "beta1", "beta2"),

    col = c("darkgreen", "red", "blue"))

****

Як бачимо коефіцієнти швидко досягають своїх справжніх значень. Тільки для β1 в нас не є постійно горизонтальна пряма, оскільки ми ініціалізували оцінку β1 і від неї проводили подальші дії.

**6.6. Порівняйте результати 6.5 з результатами оцінки множинної регресії *Y* на *X*1 і *X*2. Використайте функцію abline() для накладання цих значень на графік отриманий в 6.5.**

lm.fit = lm(Y ~ X1 + X2)

plot(1:1000, beta0, type = "l", xlab = "iteration", ylab = "betas",

 ylim = c(-5, 5), col = "darkgreen")

lines(1:1000, beta1, col = "red")

lines(1:1000, beta2, col = "blue")

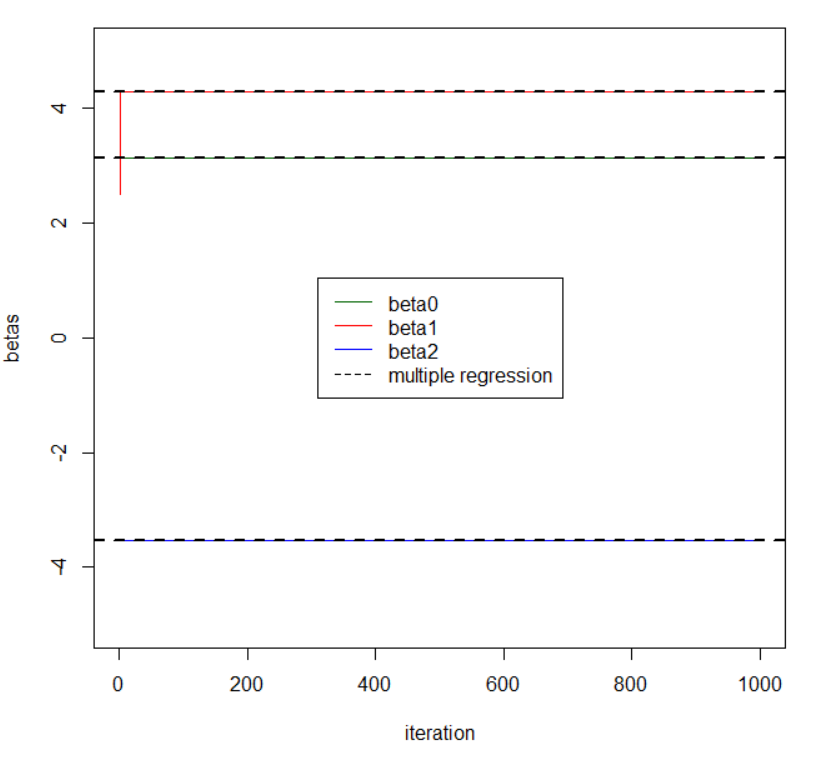
abline(h = lm.fit$coef[1], lty = "dashed", lwd = 2, col = "black")

abline(h = lm.fit$coef[2], lty = "dashed", lwd = 2, col = "black")

abline(h = lm.fit$coef[3], lty = "dashed", lwd = 2, col = "black")

legend("center", c("beta0", "beta1", "beta2", "multiple regression"), lty = c(1,

    1, 1, 2), col = c("darkgreen", "red", "blue", "black"))

****

Пунктирні лінії на графіку показують, що коефіцієнти множинної регресії збігаються з коефіцієнтами отриманими в минулому пункті за допомогою backfitting.

**6.7. Для цього набору даних, скільки ітерацій підгонки потрібно було для отримання «доброго» наближення до оцінок коефіцієнтів множинної регресії?**

Коли зв’язок між залежною змінною та предикторами є лінійний, однієї ітерації достатньо для досягнення хорошого наближення до реальних значень коефіцієнтів регресії.

**7. Продовжимо розгляд попереднього прикладу. Покажіть, що у випадку *p* = 100 можна отримати оцінки коефіцієнтів множинної регресії повторно застосовуючи метод підгонки. Скільки ітерацій підгонки потрібно було для отримання «доброго» наближення до оцінок коефіцієнтів множинної регресії? Побудуйте графік для підтвердження своїх висновків.**

set.seed(1)

p = 100

n = 1000

x = matrix(ncol = p, nrow = n)

coefi = rep(0, p)

for (i in 1:p) {

    x[, i] = rnorm(n)

    coefi[i] = rnorm(1) \* 100

}

y = x %\*% coefi + rnorm(n)

beta = rep(0, p)

max\_iterations = 1000

errors = rep(0, max\_iterations + 1)

iter = 2

errors[1] = Inf

errors[2] = sum((y - x %\*% beta)^2)

threshold = 1e-04

cat("\n")

while (iter < max\_iterations && errors[iter - 1] - errors[iter] > threshold) {

    for (i in 1:p) {

        a = y - x %\*% beta + beta[i] \* x[, i]

        beta[i] = lm(a ~ x[, i])$coef[2]

    }

    iter = iter + 1

    errors[iter] = sum((y - x %\*% beta)^2)

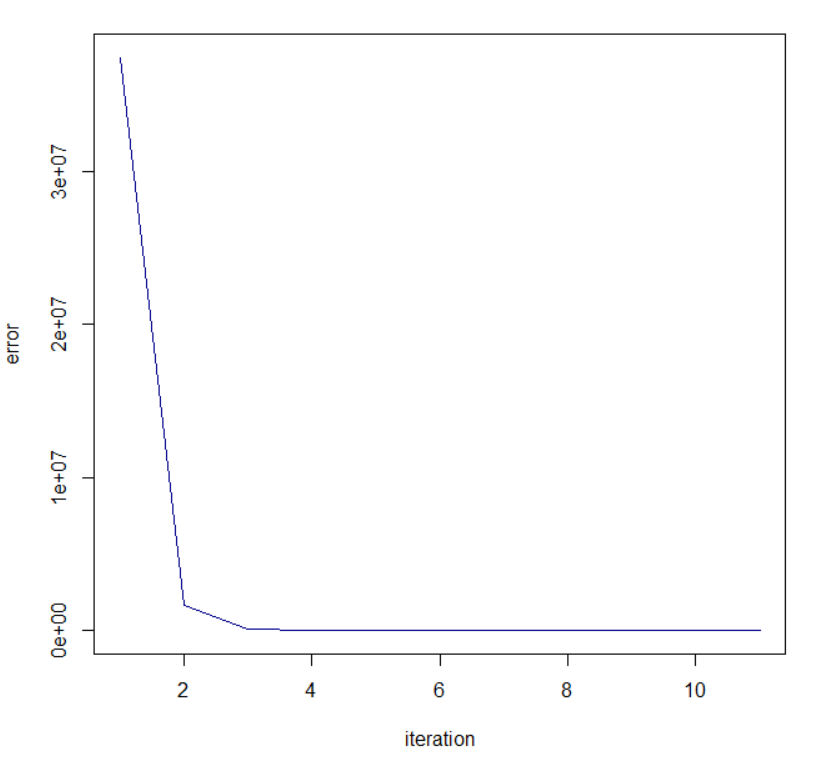
    print(paste(iter - 2, "  ", round(errors[iter - 1], 4), round(errors[iter], 4)))

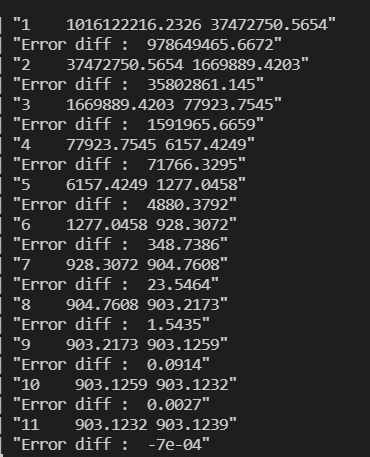
    print(paste('Error diff : ', round(errors[iter - 1] - errors[iter], 4)))

}

plot(1:11, errors[3:13], type = "l",

 xlab = "iteration", ylab = "error", col = "darkblue")

****

****

Десять ітерацій достатньо, щоб отримати хорошу апроксимацію (з точністю до 0.01) визначену пороговим значенням суми квадратичних помилок між наступними ітераціями.