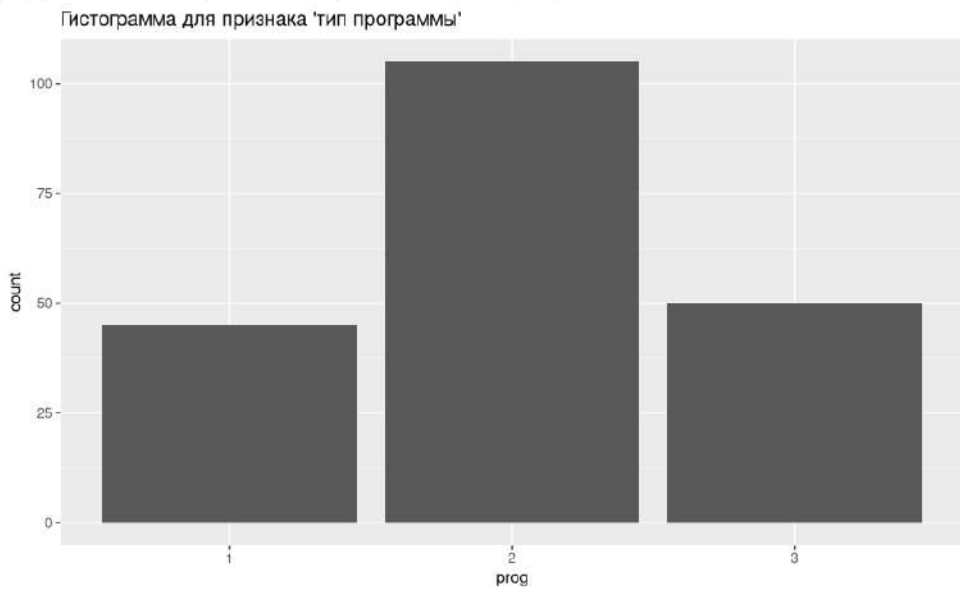
```
# task 1
#1
library(ggplot2)
library(data.table)
library(mltools)
# загрузка данных
df = read.csv("https://stats.idre.ucla.edu/stat/data/poisson_sim.csv")
head(df)
```

```
   id num_awards prog math
1  45          0    3   41
2 108          0    1   41
3  15          0    3   44
4  67          0    3   42
5 153          0    3   40
6  51          0    1   42
```

```
# есть столбец id, который нам не дает никакой полезной информации в рамках
# данной задачи
df = subset(df, select=-id)
dim(df) #200 записей

unique(df$num_awards) # 0 1 3 2 5 4 6
unique(df$prog)       # 3 1 2
# сделаем эту переменную категориальной
df$prog = factor(df$prog)
ggplot(df, aes(prog)) +
  geom_bar() +
  ggtitle("Гистограмма для признака 'тип программы'")
```




```
summary(df$math)
# минимум - 33, максимум - 75, отрицательных значений нет
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  33.00   45.00   52.00   52.65   59.00   75.00 
```

```
# строим обобщенную линейную модель
G = glm(num_awards ~.,data=df, family="poisson")

# проверим значимость факторов
summary(G)
```

```
Call:
glm(formula = num_awards ~ ., family = "poisson", data = df)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-2.2043  -0.8436  -0.5106   0.2558   2.6796  

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) -5.24712    0.65845  -7.969 1.60e-15 ***
prog2        1.08386    0.35825   3.025  0.00248 ** 
prog3        0.36981    0.44107   0.838  0.40179    
math         0.07015    0.01060   6.619 3.63e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 287.67  on 199  degrees of freedom
Residual deviance: 189.45  on 196  degrees of freedom
AIC: 373.5

Number of Fisher Scoring iterations: 6
```

```r
# признак prog3 оказался незначимым. Правда, если просто удалить его, то
# может пропасть информация о первом признаке.

# модель, где просто уберем столбец с признаком prog3
ohdf = one_hot(as.data.table(df), cols = 'prog')
ohdf = subset(ohdf, select=-prog_3)
G_trucated = glm(num_awards ~.,data=ohdf, family="poisson")
summary(G_trucated)
```

```
Call:
glm(formula = num_awards ~ ., family = "poisson", data = ohdf)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2043  -0.8436  -0.5106   0.2558   2.6796

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -4.87732    0.62818  -7.764 8.21e-15 ***
prog_1      -0.36981    0.44107  -0.838   0.4018
prog_2       0.71405    0.32001   2.231   0.0257 *
math         0.07015    0.01060   6.619 3.63e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 287.67  on 199  degrees of freedom
Residual deviance: 189.45  on 196  degrees of freedom
AIC: 373.5

Number of Fisher Scoring iterations: 6
```

```r
# убираем и prog_1
ohdf2 = subset(ohdf, select=-prog_1)
G_trucated2 = glm(num_awards ~.,data=ohdf2, family="poisson")
summary(G_trucated2)
```

```
Call:
glm(formula = num_awards ~ ., family = "poisson", data = ohdf2)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2020  -0.8346  -0.5115   0.2589   2.6793

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -5.04179    0.60783  -8.295  < 2e-16 ***
prog_2       0.89129    0.25662   3.473 0.000514 ***
math         0.06995    0.01068   6.548 5.83e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 287.67  on 199  degrees of freedom
Residual deviance: 190.16  on 197  degrees of freedom
AIC: 372.22

Number of Fisher Scoring iterations: 6
```

```r
# теперь все признаки значимые.

# сравним с моделью, где попробуем перенумеровать признаки
df$prog = as.numeric(as.character(df$prog))
df$prog[df$prog == 1] = 4
df$prog = factor(df$prog)
G_changed = glm(num_awards ~.,data=df, family="poisson")
summary(G_changed)
```

```
Call:
glm(formula = num_awards ~ ., family = "poisson", data = df)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2043  -0.8436  -0.5106   0.2558   2.6796

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -4.16327    0.66288  -6.281 3.37e-10 ***
prog3       -0.71405    0.32001  -2.231  0.02566 *
prog4       -1.08386    0.35825  -3.025  0.00248 **
math         0.07015    0.01060   6.619 3.63e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 287.67  on 199  degrees of freedom
Residual deviance: 189.45  on 196  degrees of freedom
AIC: 373.5

Number of Fisher Scoring iterations: 6
```

```
# теперь все признаки значимые.
# осталось переопределить кодировку:
# 4 - профессиональная (прикладная), 2 - общая, 3 - академическая

# показатели моделей практически идентичны.

#2
# Протестируем гипотезу о том, что модель на самом деле является тривиальной
# (то есть с одинаковым значением параметра в каждой точке)

# берем показатели для нашей модели и вычитаем из них показатели для тривиальной
# модели (null model). со степенями свободы наоборот.

# У нас есть 2 модели, сравним их результаты

pchisq(G_changed$null.deviance - G_changed$deviance,
       df = G_changed$df.null - G_changed$df.residual)
# получили 1, то есть отклоняем гипотезу о том, что модель является тривиальной.

pchisq(G_trucated2$null.deviance - G_trucated2$deviance,
       df = G_trucated2$df.null - G_trucated2$df.residual)
# получили 1, то есть отклоняем гипотезу о том, что модель является тривиальной.
# остановим выбор в итоге на усеченной модели, так как она проще, и AIC у нее
# немного меньше. А в целом это прекрасно, что при помощи некоторой перестановки
# можно сделать фактор значимым на некотором уровне.

#3
# функция, возвращающая по заданным значениям типа программы и балла за финальный
# экзамен вероятности получения 0, 1, 2, ..., 6 наград (сколько наград получит
# студент с данной программы и данным баллом за финальный экзамен)

f = function(prog, math){
  data = data.frame(prog_2 = as.numeric(prog == 2), math = max(min(math, 100), 0))
  as.numeric(round(predict(G_trucated2, data, type = "response")))
}

f(1, 10)  # 0
f(2, 50)  # 1
f(3, 80)  # 2
f(1, 87)  # 3
f(2, 80)  # 4
f(2, 83)  # 5
f(3, 98)  # 6

# заодно проверим значения, которые прогнозирует модель на исходных данных
z = predict(G_trucated2, ohdf2, type="response")
res = as.data.frame(cbind(z, round(z), ohdf$num_awards))
res['diff'] = res$V2 - res$V3
length(which(res$diff !=0)) / length(df)
# ошиблись с количеством наград в почти в 25% случаев
```

Код для задания 1

task 1
#1
library(ggplot2)

```
library(data.table)
library(mltools)
# загрузка данных
df = read.csv("https://stats.idre.ucla.edu/stat/data/poisson_sim.csv")
head(df)

# есть столбец id, который нам не дает никакой полезной информации в рамках
# данной задачи
df = subset(df, select=-id)
dim(df) #200 записей

unique(df$num_awards) # 0 1 3 2 5 4 6
unique(df$prog)      # 3 1 2
# сделаем эту переменную категориальной
df$prog = factor(df$prog)
ggplot(df, aes(prog)) +
  geom_bar() +
  ggtitle("Гистограмма для признака 'тип программы'")

summary(df$math)
# минимум - 33, максимум - 75, отрицательных значений нет

# строим обобщенную линейную модель
G = glm(num_awards ~.,data=df, family="poisson")

# проверим значимость факторов
summary(G)

# признак prog3 оказался незначимым. Правда, если просто удалить его, то
# может пропасть информация о первом признаке.

# модель, где просто уберем столбец с признаком prog3
ohdf = one_hot(as.data.table(df), cols = 'prog')
ohdf = subset(ohdf, select=-prog_3)
G_trucated = glm(num_awards ~.,data=ohdf, family="poisson")
summary(G_trucated)

# убираем и prog_1
ohdf2 = subset(ohdf, select=-prog_1)
G_trucated2 = glm(num_awards ~.,data=ohdf2, family="poisson")
summary(G_trucated2)
# теперь все признаки значимые.

# сравним с моделью, где попробуем перенумеровать признаки
df$prog = as.numeric(as.character(df$prog))
df$prog[df$prog == 1] = 4
df$prog = factor(df$prog)
G_changed = glm(num_awards ~.,data=df, family="poisson")
summary(G_changed)
# теперь все признаки значимые.
# осталось переопределить кодировку:
# 4 - профессиональная (прикладная), 2 - общая, 3 - академическая

# показатели моделей практически идентичны.

#2
# Протестируем гипотезу о том, что модель на самом деле является тривиальной
# (то есть с одинаковым значением параметра в каждой точке)

# берем показатели для нашей модели и вычитаем из них показатели для тривиальной
# модели (null model). со степенями свободы наоборот.

# У нас есть 2 модели, сравним их результаты

pchisq(G_changed$null.deviance - G_changed$deviance,
       df = G_changed$df.null - G_changed$df.residual)
# получили 1, то есть отклоняем гипотезу о том, что модель является тривиальной.

pchisq(G_trucated2$null.deviance - G_trucated2$deviance,
       df = G_trucated2$df.null - G_trucated2$df.residual)
# получили 1, то есть отклоняем гипотезу о том, что модель является тривиальной.
# остановим выбор в итоге на усеченной модели, так как она проще, и AIC у нее
# немного меньше. А в целом это прекрасно, что при помощи некоторой перестановки
```

можно сделать фактор значимым на некотором уровне.

#3
функция, возвращающая по заданным значениям типа программы и балла за финальный
экзамен вероятности получения 0, 1, 2, ..., 6 наград (сколько наград получит
студент с данной программы и данным баллом за финальный экзамен)

```
f = function(prog, math){
  data = data.frame(prog_2 = as.numeric(prog == 2), math = max(min(math, 100), 0))
  as.numeric(round(predict(G_trucated2, data, type = "response")))
}

f(1, 10)  # 0
f(2, 50)  # 1
f(3, 80)  # 2
f(1, 87)  # 3
f(2, 80)  # 4
f(2, 83)  # 5
f(3, 98)  # 6

# заодно проверим значения, которые прогнозирует модель на исходных данных
z = predict(G_trucated2, ohdf2, type="response")
res = as.data.frame(cbind(z, round(z), ohdf$num_awards))
res['diff'] = res$V2 - res$V3
length(which(res$diff !=0)) / length(df)
# ошиблись с количеством наград в почти в 25% случаев
```

(библиотека np оказалась не совместима с моей ОС, так что задание 2 делала в колабе)

```
#task2
install.packages('np')
install.packages('scatterplot3d')
install.packages('fANCOVA')
install.packages('Hmisc')
install.packages('corrplot')
```

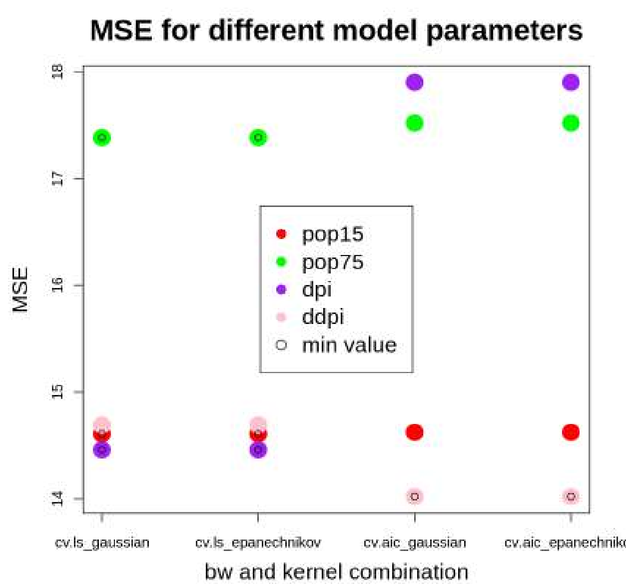
```
library(np)

#1, 2.1
# датасет
head(LifeCycleSavings)
```

	sr	pop15	pop75	dpi	ddpi
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
Australia	11.43	29.35	2.87	2329.68	2.87
Austria	12.07	23.32	4.41	1507.99	3.93
Belgium	13.17	23.80	4.43	2108.47	3.82
Bolivia	5.75	41.89	1.67	189.13	0.22
Brazil	12.88	42.19	0.83	728.47	4.56
Canada	8.79	31.72	2.85	2982.88	2.43

```
# построим матрицу значений среднеквадратичных ошибок для разных методов для
# всех четырех объясняющих переменных
M = data.frame('var' = NULL, 'bw_ker' = NULL, 'MSE' = NULL)

y = LifeCycleSavings$sr
bws = c("cv.ls", "cv.aic")
kernels = c("gaussian","epanechnikov")
```

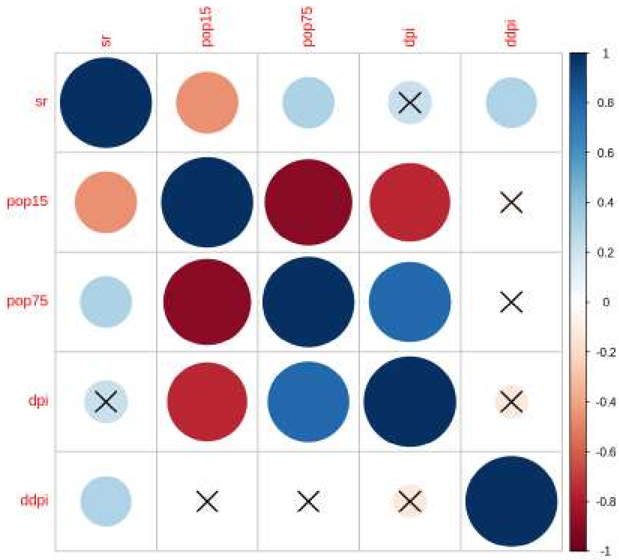
```r
# заполняем матрицу

for (i in 2:5){
  x = LifeCycleSavings[, i]
  for (bwmethod in bws){
    for (ker in kernels){
      pair = sprintf("%s_%s", bwmethod, ker)
      colname = colnames(LifeCycleSavings[i])
      model = npreg(txdat = x, tydat = y, bwmethod = bwmethod, ker = ker)
      MSE = mean((y - fitted(model)) ** 2)
      M = rbind(M, data.frame('var' = colname, 'bw_ker' = pair, 'MSE' = MSE))
    }
  }
}
M['pair_number'] = rep(c(1, 2, 3, 4), 4)
M['color'] = c(rep('red', 4), rep('green', 4), rep('purple', 4), rep('pink', 4))
```

```r
# визуализируем результаты

plot(M$pair_number, M$MSE, col = M$color, xaxp = c(-1, -1, 1),
      main = 'MSE for different model parameters', cex.main = 2,
      xlab = 'bw and kernel combination', ylab = 'MSE', lwd = 12, cex.lab = 1.5)
for (v in M$var[c(4, 8, 12, 16)]){
  min_val = min(M[M$var == v, ]$MSE)
  M_min = M[M$MSE == min_val, 3:4]
  points(M_min$pair_number, M_min$MSE, col = 'black')
}

legend(col = c(M$color[c(4, 8, 12, 16)], 'black'), x = "center", cex = 1.5,
        legend = c(M$var[c(4, 8, 12, 16)], 'min value'), pch = c(rep(16, 4), 1))
axis(1, labels = M$bw_ker[1:4], at = 1:4, padj = 0.3)
```



Как видно, метод выбора ядра не влияет на среднеквадратичную ошибку, в то время как для 3 из 4 переменных лучшим в смысле СКО методом выбора параметра bandwidth оказался обобщённый метод кросс-проверки cv.ls

```r
#2.2
# Выберем 2 переменные, которые наилучшим образом объясняют коэффициент
# персональных сбережений

# посмотрим на корреляцию величин в датасете
library(corrplot)
library(dplyr)
library(Hmisc)

matr = rcorr(as.matrix(LifeCycleSavings))
par(mfcol = c(1, 1), pty = 'm', mar = c(1, 1, 1, 1))
corrplot(matr$r, p.mat=as.matrix(mutate_all(as.data.frame(matr$P), ~if_else(is.na(.), 0, .))))
```




```
# Проверим вывод линейной модели для всех переменных
L_all = lm(sr ~ ., data = LifeCycleSavings)
summary(L_all)
```

```
[9] Call:
    lm(formula = sr ~ ., data = LifeCycleSavings)

    Residuals:
        Min      1Q  Median      3Q     Max
    -8.2422 -2.6857 -0.2488  2.4280  9.7509

    Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
    (Intercept) 28.5660865  7.3545161   3.884 0.000334 ***
    pop15       -0.4611931  0.1446422  -3.189 0.002603 **
    pop75       -1.6914977  1.0835989  -1.561 0.125530
    dpi         -0.0003369  0.0009311  -0.362 0.719173
    ddpi         0.4096949  0.1961971   2.088 0.042471 *
    ---
    Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

    Residual standard error: 3.803 on 45 degrees of freedom
    Multiple R-squared:  0.3385,    Adjusted R-squared:  0.2797
    F-statistic: 5.756 on 4 and 45 DF,  p-value: 0.0007904
```

Выбираем pop15 и ddpi в качестве двух переменных, которые наилучшим образом объясняют коэффициент персональных сбережений. Во-первых, это две значимые переменные в линейной регрессии. Во-вторых, значимая корреляция есть между sr и тремя переменными (pop75, pop15 и ddpi), но две из них сами между собой очень сильно коррелируют (pop75, pop15), поэтому из них выбираем более значимую (pop15) и оставшуюся ddpi. Также по результатам вычисления MSE pop75 показала в среднем худший результат.

```
#3
library(fANCOVA)

set.seed(888)
V1 = 2
V2 = 5

dt = sort(sample(nrow(LifeCycleSavings), nrow(LifeCycleSavings) * 0.8))
train = LifeCycleSavings[dt, c(1, V1, V2)]
test = LifeCycleSavings[-dt, c(1, V1, V2)]

# построим модели
L_2 = lm(sr ~ ., data = train)
Loess = loess.as(train[, 2:3], train$sr, criterion = "gcv")
```

```
# оценим параметры модели на обучающем наборе данных

sprintf("MSE на обучающей выборке для линейной модели равен %.2f",
        mean((train$sr - fitted(L_2)) ** 2))
sprintf("MSE на обучающей выборке для модели LOESS равен %.2f",
        mean((train$sr - fitted(Loess)) ** 2))

# проверим качество моделей на тестовом наборе данных

sprintf("MSE на тестовой выборке для линейной модели равен %.2f",
        mean((train$sr - predict(L_2, test[, 2:3], type = 'response')) ** 2))
colnames(test) = c('sr', 'x1', 'x2')
sprintf("MSE на тестовой выборке для модели LOESS равен %.2f",
        mean((train$sr - predict(Loess, test[, 2:3], type = 'response')) ** 2))

# выясняем, какая из построенных моделей является более точной

# И на обучающей, и на тестовой выборке лучшее качество по MSE показала модель
# LOESS. Однако стоит иметь в виду, что выборка у нас достаточно маленькая, и
# и для другого разбиения результат мог получиться совсем другой.
```

```
fANCOVA 0.6-1 loaded
```

'MSE на обучающей выборке для линейной модели равен 13.16'
'MSE на обучающей выборке для модели LOESS равен 8.53'
'MSE на тестовой выборке для линейной модели равен 29.04'
'MSE на тестовой выборке для модели LOESS равен 19.71'

Код

```
#task2
install.packages('np')
install.packages('scatterplot3d')
install.packages('fANCOVA')
install.packages('Hmisc')
install.packages('corrplot')

library(np)

#1, 2.1
# датасет
head(LifeCycleSavings)

# построим матрицу значений среднеквадратичных ошибок для разных методов для
# всех четырех объясняющих переменных
M = data.frame('var' = NULL, 'bw_ker' = NULL, 'MSE' = NULL)

y = LifeCycleSavings$sr
bws = c("cv.ls", "cv.aic")
kernels = c("gaussian","epanechnikov")

# заполняем матрицу

for (i in 2:5){
  x = LifeCycleSavings[, i]
  for (bwmethod in bws){
    for (ker in kernels){
      pair = sprintf("%s_%s", bwmethod, ker)
      colname = colnames(LifeCycleSavings[i])
      model = npreg(txdat = x, tydat = y, bwmethod = bwmethod, ker = ker)
      MSE = mean((y - fitted(model)) ** 2)
      M = rbind(M, data.frame('var' = colname, 'bw_ker' = pair, 'MSE' = MSE))
    }
  }
}
M['pair_number'] = rep(c(1, 2, 3, 4), 4)
M['color'] = c(rep('red', 4), rep('green', 4), rep('purple', 4), rep('pink', 4))

# визуализируем результаты
```

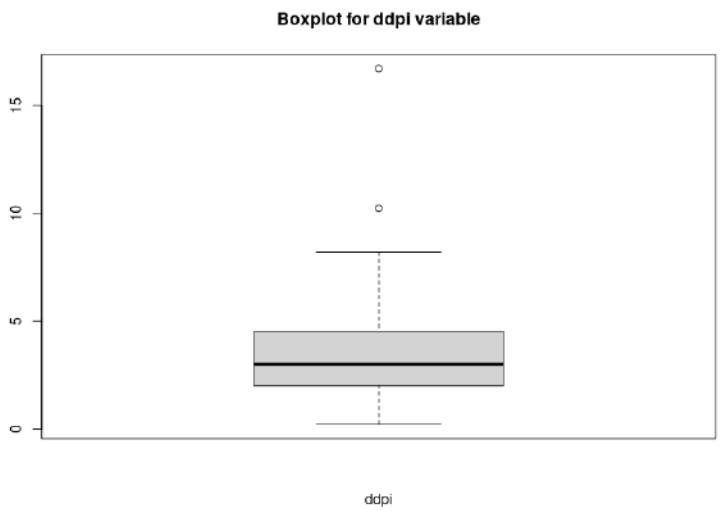
```
plot(M$pair_number, M$MSE, col = M$color, xaxp = c(-1, -1, 1),
     main = 'MSE for different model parameters', cex.main = 2,
     xlab = 'bw and kernel combination', ylab = 'MSE', lwd = 12, cex.lab = 1.5)
for (v in M$var[c(4, 8, 12, 16)]){
  min_val = min(M[M$var == v, ]$MSE)
  M_min = M[M$MSE == min_val, 3:4]
  points(M_min$pair_number, M_min$MSE, col = 'black')
}

legend(col = c(M$color[c(4, 8, 12, 16)], 'black'), x = "center", cex = 1.5,
       legend = c(M$var[c(4, 8, 12, 16)], 'min value'), pch = c(rep(16, 4), 1))
axis(1, labels = M$bw_ker[1:4], at = 1:4, padj = 0.3)

# Как видно, метод выбора ядра не влияет на среднеквадратичную ошибку, в то время как
# для 3 из 4 переменных лучшим в смысле СКО методом выбора параметра bandwidth оказался
# обобщённый метод кросс-проверки cv.ls

#2.2
# Выберем 2 переменные, которые наилучшим образом объясняют коэффициент
# персональных сбережений

# посмотрим на корреляцию величин в датасете
library(corrplot)
library(Hmisc)
library(dplyr)

matr = rcorr(as.matrix(LifeCycleSavings))
par(mfcol = c(1, 1), pty = 'm', mar = c(1, 1, 1, 1))
corrplot(matr$r, p.mat=as.matrix(mutate_all(as.data.frame(matr$P), ~if_else(is.na(.), 0, .))))

# Проверим вывод линейной модели для всех переменных
L_all = lm(sr ~ ., data = LifeCycleSavings)
summary(L_all)

# Выбираем pop15 и ddpi в качестве двух переменных, которые наилучшим образом объясняют
# коэффициент персональных сбережений. Во-первых, это две значимые переменные в линейной
# регрессии. Во-вторых, значимая корреляция есть между sr и тремя переменными (pop75, pop15
# и ddpi), но две из них сами между собой очень сильно коррелируют (pop75, pop15), поэтому
# из них выбираем более значимую (pop15) и оставшуюся ddpi. Также по результатам вычисления
# MSE pop75 показала в среднем худший результат.

#3
library(fANCOVA)

set.seed(888)
V1 = 2
V2 = 5

dt = sort(sample(nrow(LifeCycleSavings), nrow(LifeCycleSavings) * 0.8))
train = LifeCycleSavings[dt, c(1, V1, V2)]
test = LifeCycleSavings[-dt, c(1, V1, V2)]

# построим модели
L_2 = lm(sr ~ ., data = train)
Loess = loess.as(train[, 2:3], train$sr, criterion = "gcv")

# оценим параметры модели на обучающем наборе данных

sprintf("MSE на обучающей выборке для линейной модели равен %.2f",
        mean((train$sr - fitted(L_2)) ** 2))
sprintf("MSE на обучающей выборке для модели LOESS равен %.2f",
        mean((train$sr - fitted(Loess)) ** 2))

# проверим качество моделей на тестовом наборе данных

sprintf("MSE на тестовой выборке для линейной модели равен %.2f",
        mean((train$sr - predict(L_2, test[, 2:3], type = 'response')) ** 2))
colnames(test) = c('sr', 'x1', 'x2')
sprintf("MSE на тестовой выборке для модели LOESS равен %.2f",
        mean((train$sr - predict(Loess, test[, 2:3], type = 'response')) ** 2))

# выясняем, какая из построенных моделей является более точной
```

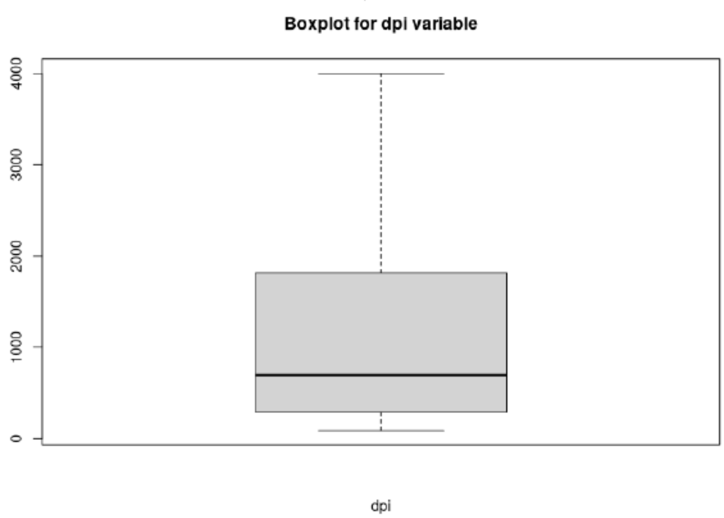
И на обучающей, и на тестовой выборке лучшее качество по MSE показала модель
LOESS. Однако стоит иметь в виду, что выборка у нас достаточно маленькая, и
и для другого разбиения результат мог получиться совсем другой.

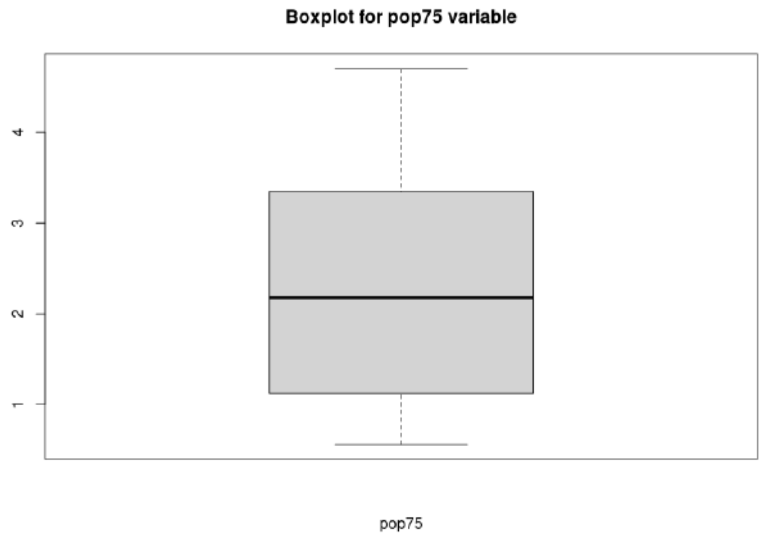
```
#task 3
# линейная модель на всем датасете
L_all = lm(sr ~ ., data = LifeCycleSavings)
summary(L_all)
# p-value: 0.0007904
```

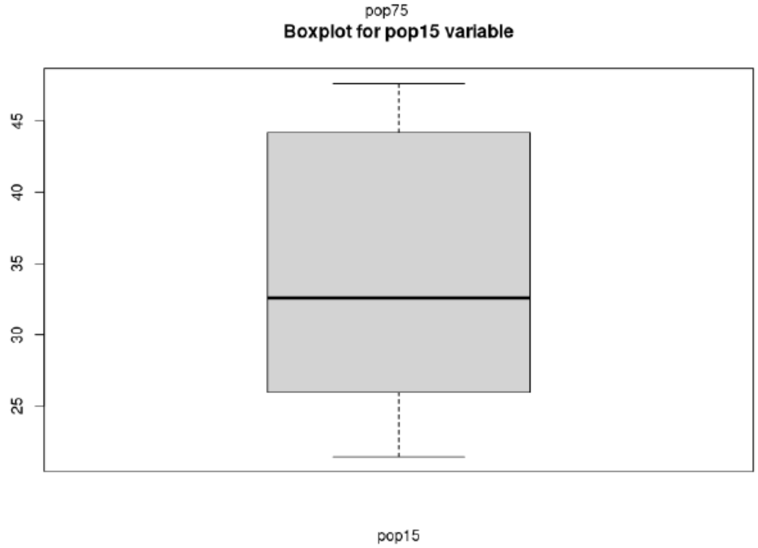
```
Call:
lm(formula = sr ~ ., data = LifeCycleSavings)

Residuals:
    Min      1Q  Median      3Q     Max 
-8.2422 -2.6857 -0.2488  2.4280  9.7509 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) 28.5660865  7.3545161   3.884 0.000334 ***
pop15       -0.4611931  0.1446422  -3.189 0.002603 ** 
pop75       -1.6914977  1.0835989  -1.561 0.125530    
dpi         -0.0003369  0.0009311  -0.362 0.719173    
ddpi         0.4096949  0.1961971   2.088 0.042471 *  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.803 on 45 degrees of freedom
Multiple R-squared:  0.3385,	Adjusted R-squared:  0.2797 
F-statistic: 5.756 on 4 and 45 DF,  p-value: 0.0007904
```

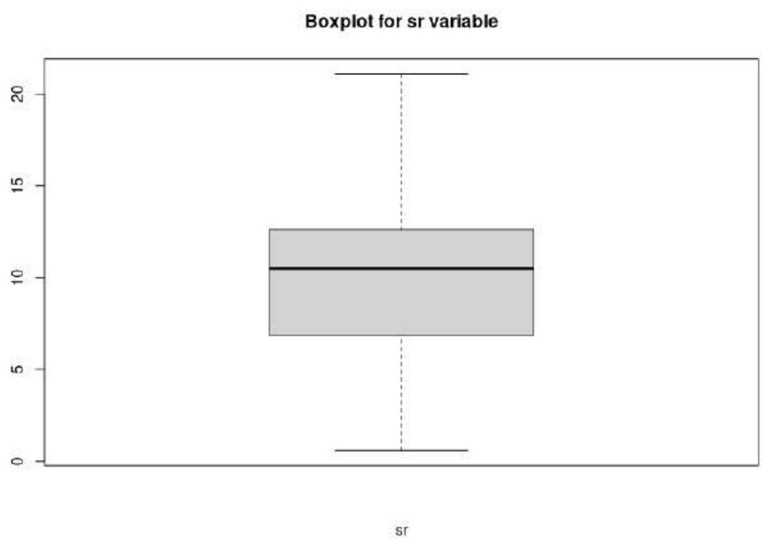
```
#a
# строим ящики с усами  отдельно для каждой из 5 переменных
for (i in 1:5){
  name = colnames(LifeCycleSavings[i])
  main = sprintf("Boxplot for %s variable", name)
  boxplot(LifeCycleSavings[, i], main=main, xlab=name)
}
```












```
# выбросы нашлись только для одной переменной - ddpi. находим индексы выбросов .
ddpi = LifeCycleSavings$ddpi
del_inds_bp = which(ddpi %in% boxplot(ddpi, plot=FALSE)$out)
del_inds_bp
# 47 49

# линейная модель на датасете без выбросов, удаленных при помощи боксплотов
L_bp = lm(sr ~ ., data = LifeCycleSavings[-del_inds_bp, ])
summary(L_bp)
# p-value: 0.0002821
```

```
Call:
lm(formula = sr ~ ., data = LifeCycleSavings[-del_inds_bp, ])

Residuals:
    Min      1Q  Median      3Q     Max 
-7.9884 -2.4159  0.1353  2.2927  8.8034 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)  
(Intercept) 20.5778361  8.5573223   2.405   0.0206 *
pop15       -0.3222471  0.1629443  -1.978   0.0544 .
pop75       -0.9168151  1.1578170  -0.792   0.4328  
dpi         -0.0002958  0.0009177  -0.322   0.7487  
ddpi         0.8394509  0.3082606   2.723   0.0093 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.747 on 43 degrees of freedom
Multiple R-squared:  0.3834,	Adjusted R-squared:  0.3261 
F-statistic: 6.685 on 4 and 43 DF,  p-value: 0.0002821
```

```r
#b
#  подсчёт параметров leverages (регрессия от 4-х факторов у нас уже построена)
sum(hatvalues(L_all))
# 5 - все верно

del_inds_l = which(hatvalues(L_all) > 2 * mean(hatvalues(L_all)))
del_inds_l
#      Ireland         Japan United States         Libya 
#           21            23            44            49 

# линейная модель на датасете без выбросов, удаленных при помощи leverages
L_l = lm(sr ~ ., data = LifeCycleSavings[-del_inds_l, ])
summary(L_l)
# p-value: 0.005315
```

```
Call:
lm(formula = sr ~ ., data = LifeCycleSavings[-del_inds_l, ])

Residuals:
    Min      1Q  Median      3Q     Max 
-7.9632 -2.6323  0.1466  2.2529  9.6687 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)  
(Intercept)  2.221e+01  9.319e+00   2.384   0.0218 *
pop15       -3.403e-01  1.798e-01  -1.893   0.0655 .
pop75       -1.124e+00  1.398e+00  -0.804   0.4258  
dpi         -4.499e-05  1.160e-03  -0.039   0.9692  
ddpi         5.273e-01  2.775e-01   1.900   0.0644 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.805 on 41 degrees of freedom
Multiple R-squared:  0.2959,	Adjusted R-squared:  0.2272 
F-statistic: 4.308 on 4 and 41 DF,  p-value: 0.005315
```

```r
# в итоге, лучшее качество (по p-уровням тестов для коэффициентов детерминации R^2)
# показала модель с удалением выбросов по диаграмме размаха. Второй стала модель
# на всех данных. На порядок больший p-value у модели с удалением наблюдений, для
# которых leverages превосходят более чем в 2 раза среднее значение этого параметра
# по всем наблюдениям.
# при этом Libya дважды попадала в список наблюдений-выбросов.
# интересно было бы предсказать значения sr для наблюдений-выбросов всеми тремя
# моделями, и сравнить отклонение от реального значения (так как если модели будут
# использоваться для предсказания, то могут прийти и такие значения-выбросы).
del_inds = unique(c(del_inds_bp, del_inds_l))
L_all_pred = predict(L_all, LifeCycleSavings[del_inds, ], type = "response")
L_bp_pred = predict(L_bp, LifeCycleSavings[del_inds, ], type = "response")
L_l_pred = predict(L_l, LifeCycleSavings[del_inds, ], type = "response")
real_sr = LifeCycleSavings$sr[del_inds]

# отклонения
sqrt(sum((L_all_pred - real_sr) ** 2) / 5)
# 3.398448
sqrt(sum((L_bp_pred - real_sr) ** 2) / 5)
# 5.720109
sqrt(sum((L_l_pred - real_sr) ** 2) / 5)
# 4.168431

# вот тут уже интереснее. Общая модель показала самый хороший результат, и это
# логично, так как она обучалась на всех данных, в том числе на этих 5 объектах.
# теперь на 2 месте модель с удалением объектов по leverages, и это несмотря на
# то, что она не видела 4 из 5 переданных значений. на последнем месте модель с
# удалением выбросов по по диаграмме размаха, она была построена с учетом 4 из 5
# значений из списка.
```

Код

```
#task 3
# линейная модель на всем датасете
L_all = lm(sr ~ ., data = LifeCycleSavings)
summary(L_all)
# p-value: 0.0007904

#a
# строим ящики с усами  отдельно для каждой из 5 переменных
for (i in 1:5){
  name = colnames(LifeCycleSavings[i])
  main = sprintf("Boxplot for %s variable", name)
  boxplot(LifeCycleSavings[, i], main=main, xlab=name)
}

# выбросы нашлись только для одной переменной - ddpi. находим индексы выбросов .
ddpi = LifeCycleSavings$ddpi
del_inds_bp = which(ddpi %in% boxplot(ddpi, plot=FALSE)$out)
del_inds_bp
# 47 49

# линейная модель на датасете без выбросов, удаленных при помощи боксплотов
L_bp = lm(sr ~ ., data = LifeCycleSavings[-del_inds_bp, ])
summary(L_bp)
# p-value: 0.0002821

#b
#  подсчёт параметров leverages (регрессия от 4-х факторов у нас уже построена)
sum(hatvalues(L_all))
# 5 - все верно

del_inds_l = which(hatvalues(L_all) > 2 * mean(hatvalues(L_all)))
del_inds_l
#      Ireland        Japan United States         Libya
#           21           23            44            49

# линейная модель на датасете без выбросов, удаленных при помощи leverages
L_l = lm(sr ~ ., data = LifeCycleSavings[-del_inds_l, ])
summary(L_l)
# p-value: 0.005315

# в итоге, лучшее качество (по p-уровням тестов для коэффициентов детерминации R^2)
# показала модель с удалением выбросов по диаграмме размаха. Второй стала модель
# на всех данных. На порядок больший p-value у модели с удалением наблюдений, для
# которых leverages превосходят более чем в 2 раза среднее значение этого параметра
# по всем наблюдениям.
# при этом Libya дважды попадала в список наблюдений-выбросов.
# интересно было бы предсказать значения sr для наблюдений-выбросов всеми тремя
# моделями, и сравнить отклонение от реального значения (так как если модели будут
# использоваться для предсказания, то могут прийти и такие значения-выбросы).
del_inds = unique(c(del_inds_bp, del_inds_l))
L_all_pred = predict(L_all, LifeCycleSavings[del_inds, ], type = "response")
L_bp_pred = predict(L_bp, LifeCycleSavings[del_inds, ], type = "response")
L_l_pred = predict(L_l, LifeCycleSavings[del_inds, ], type = "response")
real_sr = LifeCycleSavings$sr[del_inds]

# отклонения
sqrt(sum((L_all_pred - real_sr) ** 2) / 5)
# 3.398448
sqrt(sum((L_bp_pred - real_sr) ** 2) / 5)
# 5.720109
sqrt(sum((L_l_pred - real_sr) ** 2) / 5)
# 4.168431

# вот тут уже интереснее. Общая модель показала самый хороший результат, и это
# логично, так как она обучалась на всех данных, в том числе на этих 5 объектах.
# теперь на 2 месте модель с удалением объектов по leverages, и это несмотря на
# то, что она не видела 4 из 5 переданных значений. на последнем месте модель с
# удалением выбросов по по диаграмме размаха, она была построена с учетом 4 из 5
# значений из списка.
```

N4 1) $h = 1/2$

$K(x) = (1-|x|)\cdot \mathbb{I}\{|x|\le 1\}$

$n = 6$; $x_i = i$, $\forall i = 1 \dots 6$

$$\hat{r}(x) = \frac{\sum_{i=1}^{n} y_i K((x-x_i)/h)}{\sum_{i=1}^{n} K((x-x_i)/h)}$$ ← (переобозначим для удобства x_i как $\tilde{x}_i$)

$\hat{\vec{y}} = H\vec{y}$, $\vec{y} = (y_1, \dots, y_n)^T$, $\hat{\vec{y}} = (\hat{r}(x_1), \dots, \hat{r}(x_n))^T$

Найдём вектор $\hat{\vec{y}}$ (все значение $\hat{r}(x_i)$)

$$\hat{r}(x_1) = \hat{r}(1) = \frac{\sum_{i=1}^{n} y_i K((1-\tilde{x}_i)/0{,}5)}{\sum_{i=1}^{n} K((1-\tilde{x}_i)/0{,}5)} = \frac{\sum_{i=1}^{n} y_i K(2(1-\tilde{x}_i))}{\sum_{i=1}^{n} K(2(1-\tilde{x}_i))} =$$

= [для значений $|x|>1$ $K(x) = 0$ (по индикатору), т.е. для выражения $2(1-\tilde{x}_i)$ для $\tilde{x}_i > 1{,}5$ и $\tilde{x}_i < 0{,}5$ $K(x)$ будет равно 0. Значит, только для $\tilde{x}_1 = 1$ есть ненулевое значение] =

$$= \frac{y_1 \cdot K(2(1-1))}{K(2(1-1))} = \left[K(0) = (1-0)\cdot \mathbb{I}\{|0|\le 1\} = 1 \right] = y_1$$

Аналогично для $\hat{r}(x_2)$, $\hat{r}(x_3)$, $\hat{r}(x_4)$, $\hat{r}(x_5)$ и $\hat{r}(x_6)$ единственным значением, для которого $K(x)$ не равно 0, будет само значение $\tilde{x}_i$ в $\hat{r}(x_i)$.

$$\hat{r}(x_2 = 2) = \frac{\sum_{i=1}^{n} y_i K(2(2-\tilde{x}_i))}{\sum_{i=1}^{n} K(2(2-\tilde{x}_i))} = \frac{y_2 \cdot K(2\cdot(2-2))}{K(2\cdot(2-2))} = y_2$$

$\hat{r}(x_3 = 3) = y_3$; $\hat{r}(x_4 = 4) = y_4$; $\hat{r}(x_5 = 5) = y_5$,

$\hat{r}(x_6 = 6) = y_6$ => $\hat{\vec{y}} = (y_1, \dots, y_6)^T = \vec{y}$

Найдём H.

По свойству единичной матрицы $X \cdot E = E \cdot X = X$

Тогда

$$\vec{\hat{y}} = H \cdot \vec{y} \quad \text{при} \quad \vec{\hat{y}} = \vec{y}:$$

$$\vec{y} = H \cdot \vec{y} \quad \Rightarrow \quad \boxed{H = E = \begin{pmatrix} 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&0&1 \end{pmatrix}}$$

Тогда $\underline{tr H = 6 \cdot 1 = 6}$

2) $h = 3/2$

Вычислим $\hat{r}(x_i)$ для $i = 1 \ldots 6$

$$\hat{r}(x_1 = 1) = \frac{\sum_{i=1}^{n} y_i \cdot K((1-\tilde{x}_i)/1.5)}{\sum_{i=1}^{n} K((1-\tilde{x}_i)/1.5)} = \frac{\sum_{i=1}^{n} y_i \cdot K(2/3 \cdot (1-\tilde{x}_i))}{\sum_{i=1}^{n} K(2/3 \cdot (1-\tilde{x}_i))} =$$

$= \Big[$ чтобы $K(x) \neq 0$ по индикатору, значения $\tilde{x}_i$ должны быть $-1 < 2/3(1-\tilde{x}_i) < 1 \Rightarrow -1,5 < 1-\tilde{x}_i < 1,5 \Rightarrow -2,5 < -\tilde{x}_i < 0,5 \Rightarrow 0,5 < \tilde{x}_i < 2,5 \Rightarrow \tilde{x}_i = 1$ и $\tilde{x}_i = 2 \Big] =$

$$= \frac{y_1 \cdot K(2/3(1-1)) + y_2 \cdot K(2/3(1-2))}{K(2/3(1-1)) + K(2/3(1-2))} = \left[\begin{array}{l} K(-2/3) = K(2/3) = \\ = (1-2/3) \cdot \mathbb{I}\{|x| \geq 1\} = 1/3 \end{array} \right] =$$

$$= \frac{y_1 \cdot 1 + y_2 \cdot 1/3}{1 + 1/3} = \frac{y_1 + y_2 \cdot 1/3}{4/3} = \frac{3(y_1 + y_2 \cdot 1/3)}{4} = \frac{3y_1 + y_2}{4}$$

Аналогично для остальных $\hat{r}(x_i)$: $K(x) \neq 0$ по индикатору для таких $\tilde{x}_i$, что лежат в окрестности $\pm 1,5$ от x_i. В нашем случае для крайних значений подходящими будут два значения: само $\tilde{x}_i = x_i$ и $\tilde{x}_i = x_{i+1}$ для $i = 1$; $\tilde{x}_i = x_{i-1}$ для $i = 6$. Для остальных i подходящими будут $\tilde{x}_i = x_i$; $\tilde{x}_i = x_{i-1}$; $\tilde{x}_i = x_{i+1}$

Считаем:

$$\hat{r}(x_2 = 2) = \frac{\sum_{i=1}^{n} y_i \cdot K(2/3 \cdot (2-\tilde{x}_i))}{\sum_{i=1}^{n} K(2/3 \cdot (2-\tilde{x}_i))} =$$

$$= \frac{y_1 \cdot K(2/3 \cdot (2-1)) + y_2 \cdot K(2/3(2-2)) + y_3 \cdot K(2/3(2-3))}{K(2/3 \cdot (2-1)) + K(2/3 \cdot (2-2)) + K(2/3 \cdot (2-3))} =$$

$$= \frac{y_1 \cdot K(2/3) + y_2 \cdot K(0) + y_3 \cdot K(-2/3)}{K(2/3) + K(0) + K(-2/3)} = \frac{1/3\, y_1 + y_2 + 1/3 \cdot y_3}{1/3 + 1 + 1/3} =$$

$$= \frac{3(1/3 y_1 + y_2 + 1/3 y_3)}{5} = \frac{y_1 + 3y_2 + y_3}{5}$$

$$\hat{r}(x_3 = 3) = \frac{y_2 + 3y_3 + y_4}{5}, \quad \hat{r}(x_4 = 4) = \frac{y_3 + 3y_4 + y_5}{5},$$

$$\hat{r}(x_5 = 5) = \frac{y_4 + 3y_5 + y_6}{5}$$

$$\hat{r}(x_6 = 6) = \frac{y_5 \cdot K(2/3 \cdot (6-5)) + y_6 \cdot K(2/3(6-6))}{K(2/3 \cdot (6-5)) + K(2/3 \cdot (6-6))} = \frac{y_5 \cdot K(2/3) + y_6 \cdot K(0)}{K(2/3) + K(0)} =$$

$$= \frac{1/3\, y_5 + y_6}{4/3} = \frac{y_5 + 3y_6}{4}$$

Собираем векторы.

$$\vec{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}, \quad \hat{\vec{y}} = \begin{pmatrix} 3/4\, y_1 + 1/4\, y_2 \\ 1/5\, y_1 + 3/5\, y_2 + 1/5\, y_3 \\ 1/5\, y_2 + 3/5\, y_3 + 1/5\, y_4 \\ 1/5\, y_3 + 3/5\, y_4 + 1/5\, y_5 \\ 1/5\, y_4 + 3/5\, y_5 + 1/5\, y_6 \\ 1/4\, y_5 + 3/4\, y_6 \end{pmatrix},$$

тогда для $\hat{\vec{y}} = H \cdot \vec{y}$ найдём матрицу H подбором, пользуясь св-вом перемножения матриц

$$H = \begin{pmatrix} \smile & \smile & \smile & \smile & \smile & \smile \\ \smile & \smile & \smile & \smile & \smile & \smile \\ \smile & \smile & \smile & \smile & \smile & \smile \\ \smile & \smile & \smile & \smile & \smile & \smile \\ \smile & \smile & \smile & \smile & \smile & \smile \\ \smile & \smile & \smile & \smile & \smile & \smile \end{pmatrix} = [\text{подбираем коэффициенты}] =$$

$$= \begin{pmatrix} 3/4 & 1/4 & 0 & 0 & 0 & 0 \\ 1/5 & 3/5 & 1/5 & 0 & 0 & 0 \\ 0 & 1/5 & 3/5 & 1/5 & 0 & 0 \\ 0 & 0 & 1/5 & 3/5 & 1/5 & 0 \\ 0 & 0 & 0 & 1/5 & 3/5 & 1/5 \\ 0 & 0 & 0 & 0 & 1/4 & 3/4 \end{pmatrix}, \quad tr\, H = 2 \cdot 3/4 + 4 \cdot 3/5 = 3(0{,}5 + 0{,}8) = 3{,}9$$

Проверка

$$H \cdot \vec{y} = \begin{pmatrix} 3/4 & 1/4 & 0 & 0 & 0 & 0 \\ 1/5 & 3/5 & 1/5 & 0 & 0 & 0 \\ 0 & 1/5 & 3/5 & 1/5 & 0 & 0 \\ 0 & 0 & 1/5 & 3/5 & 1/5 & 0 \\ 0 & 0 & 0 & 1/5 & 3/5 & 1/5 \\ 0 & 0 & 0 & 0 & 1/4 & 3/4 \end{pmatrix} \cdot \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix} = \begin{pmatrix} 3/4\, y_1 + 1/4\, y_2 + 0 + 0 + 0 + 0 \\ 1/5\, y_1 + 3/5\, y_2 + 1/5\, y_3 + 0 + 0 + 0 \\ 0 + 1/5\, y_2 + 3/5\, y_3 + 1/5\, y_4 + 0 + 0 \\ 0 + 0 + 1/5\, y_3 + 3/5\, y_4 + 1/5\, y_5 + 0 \\ 0 + 0 + 0 + 1/5\, y_4 + 3/5\, y_5 + 1/5\, y_6 \\ 0 + 0 + 0 + 0 + 1/4\, y_5 + 3/4\, y_6 \end{pmatrix} \equiv \hat{\vec{y}}$$

№5 Модель линейной регрессии

$$\begin{pmatrix} y_1 \\ \dots \\ y_n \end{pmatrix} = \begin{pmatrix} x_{11} & \dots & x_{1m} \\ \dots & \dots & \dots \\ x_{n1} & \dots & x_{nm} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \dots \\ \beta_m \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \dots \\ \varepsilon_n \end{pmatrix} = X\vec{\beta} + \vec{\varepsilon}$$

$n \geqslant m$

1) $x_{ij} = u_i^{j-1}$, $i = 1 \dots n$, $j = 1 \dots m$, $u_1, \dots, u_n$ – различны

Выберем m строк и m столбцов (для простоты первые m), чтобы получить минор порядка m.

Это максимальный порядок минора для матрицы $m \times n$, т.е. все миноры большего порядка равны нулю (не существуют).

Посмотрим, является ли данный минор ненулевым.

Подставим значения x_{ij} в определитель.

$$\begin{vmatrix} 1 & u_1 & u_1^2 & \dots & u_1^{m-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & u_m & u_m^2 & \dots & u_m^{m-1} \end{vmatrix} = \left[\begin{matrix} \text{это определитель} \\ \text{Вандермонда,} \\ \text{он равен (см wiki)} \end{matrix} \right] = \prod_{1 \leq j < i \leq n} (u_i - u_j)$$

Этот определитель равен нулю тогда и только тогда, когда существует хотя бы одна пара (u_i, u_j) такая, что $u_i = u_j$, $i \neq j$, чего по условию задачи быть не может.

Таким образом, данный минор является базисным, а это значит что столбцы, входящие в его состав, линейно независимы, как и m строк. Значит, ранг матрицы X равен m, и у неё есть m линейно независимых столбцов.

Всего в матрице X m столбцов, а значит, все столбцы матрицы X линейно независимы

2) $Q = X^T X$, вектор $\vec{v} \in R^m$ - ненулевой

$\vec{v}^T Q \vec{v} = \vec{v}^T X^T X \vec{v} > 0$ - доказать

Обратим внимание, что Q является матрицей Грама:

$$\begin{pmatrix} x_{11} & \dots & x_{n1} \\ \dots & \dots & \dots \\ x_{1m} & \dots & x_{nm} \end{pmatrix} \cdot \begin{pmatrix} x_{11} & \dots & x_{1m} \\ \dots & \dots & \dots \\ x_{n1} & \dots & x_{nm} \end{pmatrix} = \begin{pmatrix} e_1 \\ \dots \\ e_m \end{pmatrix} \cdot (e_1 \dots e_m) =$$

$$= \begin{pmatrix} \langle e_1, e_1 \rangle & \dots & \langle e_1, e_m \rangle \\ \dots & \dots & \dots \\ \langle e_m, e_1 \rangle & \dots & \langle e_m, e_m \rangle \end{pmatrix}$$

Значит, сама матрица Q является положительно определённой (св-во положительно определённой матрицы, wiki, п.4), так как по условию задачи она образована линейно независимыми векторами.