HW_a

Shamil Alburin BEC172

Загрузим данные

In [1]:
```
data <- read.csv('forestfires.csv')
```

Задание 1

Отфильтруем, отсортируем по показателю area

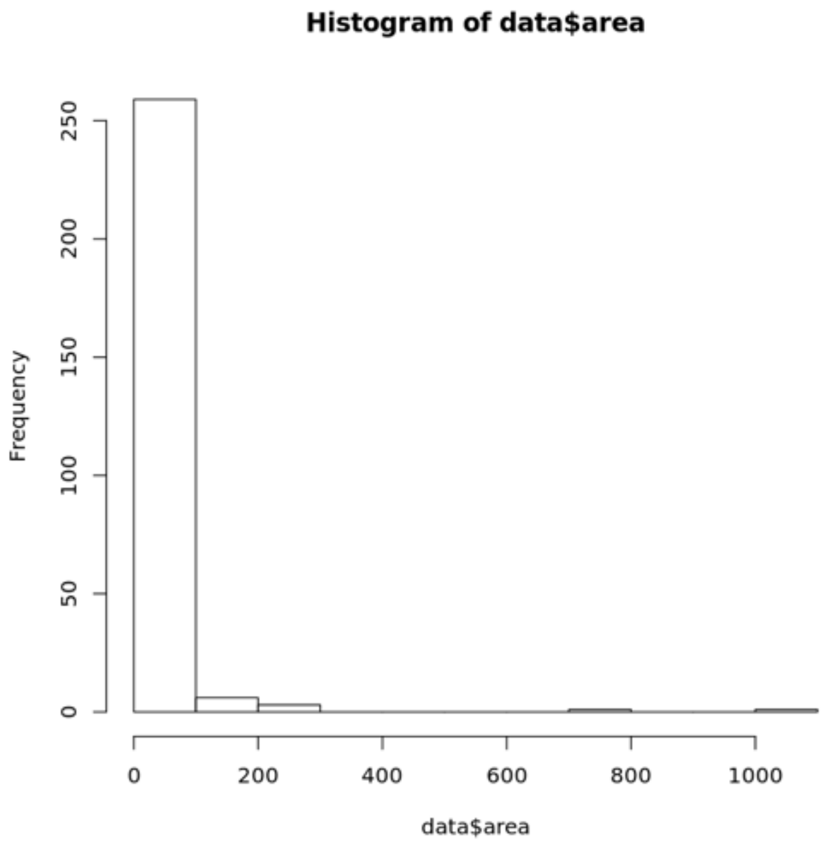
In [2]:
```
data <- subset(data, area > 0)
data <- data[order(data$area),]
head(data)
```

A data.frame: 6 × 13

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain
	<int>	<int>	<fct>	<fct>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<int>	<dbl>	<dbl>
247	5	4	aug	sun	91.8	175.1	700.7	13.8	25.7	39	5.4	0
267	6	5	aug	tue	94.3	131.7	607.1	22.7	19.4	55	4.0	0
253	6	5	aug	wed	93.1	157.3	666.7	13.5	22.1	37	3.6	0
252	8	5	aug	wed	93.1	157.3	666.7	13.5	24.0	36	3.1	0
440	1	3	sep	fri	91.1	91.3	738.1	7.2	19.1	46	2.2	0
139	9	9	jul	tue	85.8	48.3	313.4	3.9	18.0	42	2.7	0

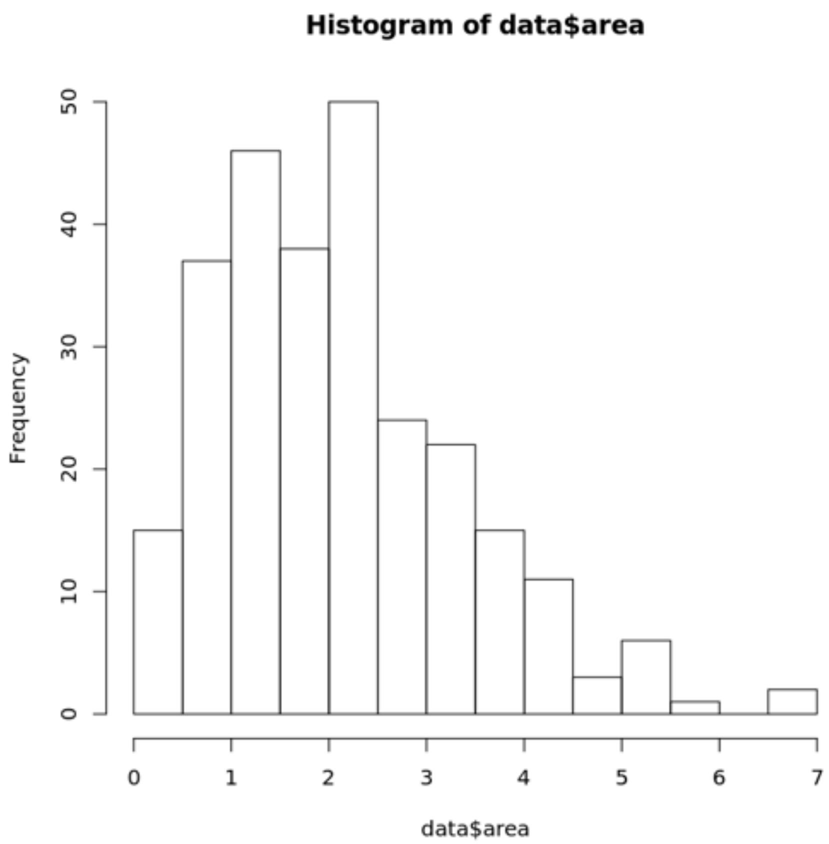
Посмотрим на объясняемую переменную

In [3]:
```
hist(data$area)
```



Она слишком сильно смещена к 0, попробуем прологарифмировать

```r
In [4]: data$area = log(1 + data$area)
        hist(data$area)
```



Все стало отлично))))

Дни недели нам сильно не важны, но выходные - да, так как пожары зачастую вызваны людьми, а они более свободны в выходные.

```r
In [5]: weekend <- as.numeric(data$day == 'sat' | data$day == 'sun')
        data$weekend <- weekend
        head(data)
```

A data.frame: 6 × 14

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain
	<int>	<int>	<fct>	<fct>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<int>	<dbl>	<dbl>
247	5	4	aug	sun	91.8	175.1	700.7	13.8	25.7	39	5.4	0
267	6	5	aug	tue	94.3	131.7	607.1	22.7	19.4	55	4.0	0
253	6	5	aug	wed	93.1	157.3	666.7	13.5	22.1	37	3.6	0
252	8	5	aug	wed	93.1	157.3	666.7	13.5	24.0	36	3.1	0
440	1	3	sep	fri	91.1	91.3	738.1	7.2	19.1	46	2.2	0
139	9	9	jul	tue	85.8	48.3	313.4	3.9	18.0	42	2.7	0

Задание 2

Так как мы добавили дамми для выходных, дни недели нам не нужны. Месяцы и температура сильно связаны, поэтому откажемся от месяцев, так как именно температура важней. В данном исследовании мы также не будем смотреть на местоположение, хотя можно было сделать дамми всех квадратиков кроме одного, но это долго и будет много переменных((

Считаю, что у всех выбранных регрессоров, кроме дождя (rain), будет положительный коэффициент

In [6]:
```
data <- subset(data, select = - c(X, Y, month, day))
```

In [7]:
```
summary(data)
```

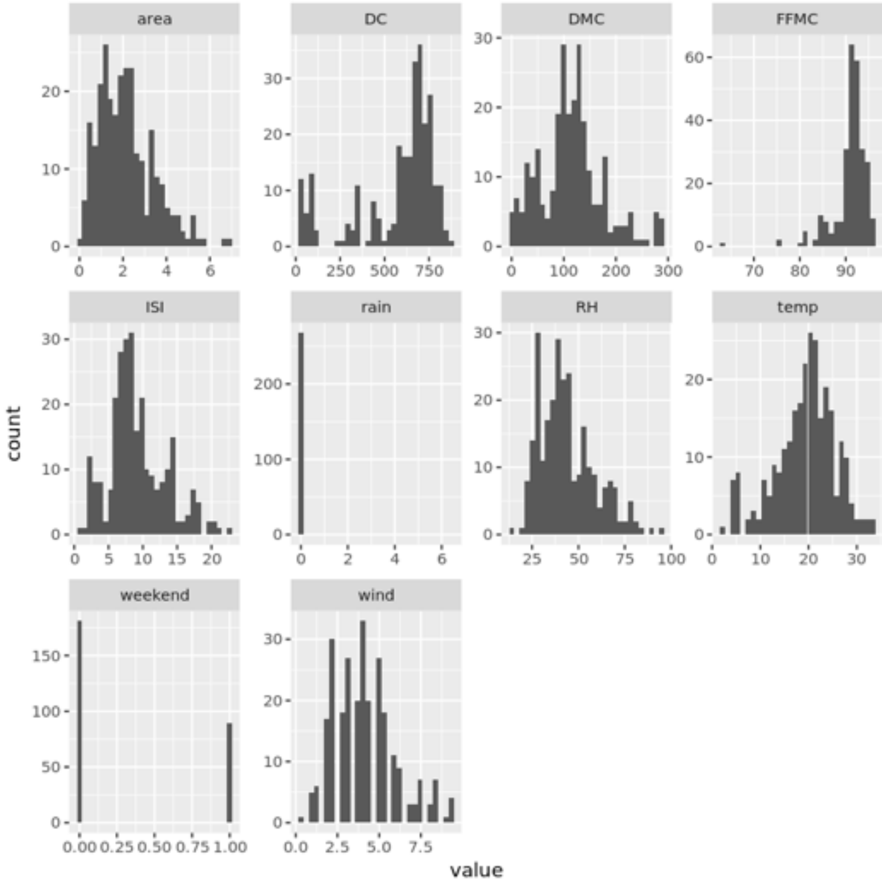
```
      FFMC            DMC              DC             ISI        
 Min.   :63.50   Min.   :  3.2   Min.   : 15.3   Min.   : 0.800  
 1st Qu.:90.33   1st Qu.: 82.9   1st Qu.:486.5   1st Qu.: 6.800  
 Median :91.70   Median :111.7   Median :665.6   Median : 8.400  
 Mean   :91.03   Mean   :114.7   Mean   :570.9   Mean   : 9.177  
 3rd Qu.:92.97   3rd Qu.:141.3   3rd Qu.:721.3   3rd Qu.:11.375  
 Max.   :96.20   Max.   :291.3   Max.   :860.6   Max.   :22.700  
      temp             RH             wind            rain        
 Min.   : 2.20   Min.   :15.00   Min.   :0.400   Min.   :0.00000  
 1st Qu.:16.12   1st Qu.:33.00   1st Qu.:2.700   1st Qu.:0.00000  
 Median :20.10   Median :41.00   Median :4.000   Median :0.00000  
 Mean   :19.31   Mean   :43.73   Mean   :4.113   Mean   :0.02889  
 3rd Qu.:23.40   3rd Qu.:53.00   3rd Qu.:4.900   3rd Qu.:0.00000  
 Max.   :33.30   Max.   :96.00   Max.   :9.400   Max.   :6.40000  
      area             weekend      
 Min.   :0.08618   Min.   :0.0000  
 1st Qu.:1.14422   1st Qu.:0.0000  
 Median :1.99742   Median :0.0000  
 Mean   :2.12741   Mean   :0.3296  
 3rd Qu.:2.79865   3rd Qu.:1.0000  
 Max.   :6.99562   Max.   :1.0000  
```

In [8]:
```r
install.packages('purrr')
install.packages('tidyr')
install.packages('ggplot2')

library(purrr)
library(tidyr)
library(ggplot2)
```

```
Installing package into '/srv/rlibs'
(as 'lib' is unspecified)

Installing package into '/srv/rlibs'
(as 'lib' is unspecified)

also installing the dependencies 'generics', 'dplyr', 'stringi', '
tidyselect'


Installing package into '/srv/rlibs'
(as 'lib' is unspecified)

also installing the dependencies 'ps', 'processx', 'callr', 'prett
yunits', 'withr', 'backports', 'desc', 'pkgbuild', 'rprojroot', 'p
kgload', 'praise', 'colorspace', 'testthat', 'farver', 'labeling',
'munsell', 'RColorBrewer', 'viridisLite', 'gtable', 'isoband', 'sc
ales'
```

In [9]:
```
data %>%
  keep(is.numeric) %>%
  gather() %>%
  ggplot(aes(value)) +
    facet_wrap(~ key, scales = "free") +
    geom_histogram()
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Посмотрим на хвост выборки

In [10]:
```
tail(data)
```

A data.frame: 6 × 10

	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area	weekend
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<int>	<dbl>	<dbl>	<dbl>	<dbl>
236	91.4	142.4	601.4	10.6	19.6	41	5.8	0	5.285637	1
237	92.5	121.1	674.4	8.6	18.2	46	1.8	0	5.307971	1
238	91.0	129.5	692.6	7.0	18.8	40	2.2	0	5.365415	0
480	89.2	103.9	431.6	6.4	22.6	57	4.9	0	5.633110	0
416	94.8	222.4	698.6	13.9	27.5	27	4.9	0	6.616440	0
239	92.5	121.1	674.4	8.6	25.1	27	4.0	0	6.995620	1

Сильно выделяются два последних значения, можем их просто убрать для репрезентативной выборки

In [11]:
```
data <- data[1 : (dim(data)[1] - 2),]
```

In [12]:
```
tail(data)
```

A data.frame: 6 × 10

	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area	weekend
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<int>	<dbl>	<dbl>	<dbl>	<dbl>
378	93.7	231.1	715.1	8.4	21.9	42	2.2	0	5.168380	1
421	91.7	191.4	635.9	7.8	26.2	36	4.5	0	5.229824	0
236	91.4	142.4	601.4	10.6	19.6	41	5.8	0	5.285637	1
237	92.5	121.1	674.4	8.6	18.2	46	1.8	0	5.307971	1
238	91.0	129.5	692.6	7.0	18.8	40	2.2	0	5.365415	0
480	89.2	103.9	431.6	6.4	22.6	57	4.9	0	5.633110	0

Задание 3

In [13]:
```
install.packages('usdm')
library(usdm)
```

```
Installing package into '/srv/rlibs'
(as 'lib' is unspecified)

also installing the dependencies 'sp', 'raster'


Loading required package: sp

Loading required package: raster


Attaching package: 'raster'


The following object is masked from 'package:tidyr':

    extract
```

In [14]:
```
vif(data)
```

A data.frame: 10 × 2

Variables	VIF
<fct>	<dbl>
FFMC	2.700478
DMC	2.357480
DC	2.014470
ISI	2.408070
temp	2.832106
RH	1.745069
wind	1.288590
rain	1.059300
area	1.043546
weekend	1.099523

С VIF все нормально, критическое значение - 5

In [15]:
```
cond(data)
```

```
Error in cond(data): could not find function "cond"
Traceback:
```

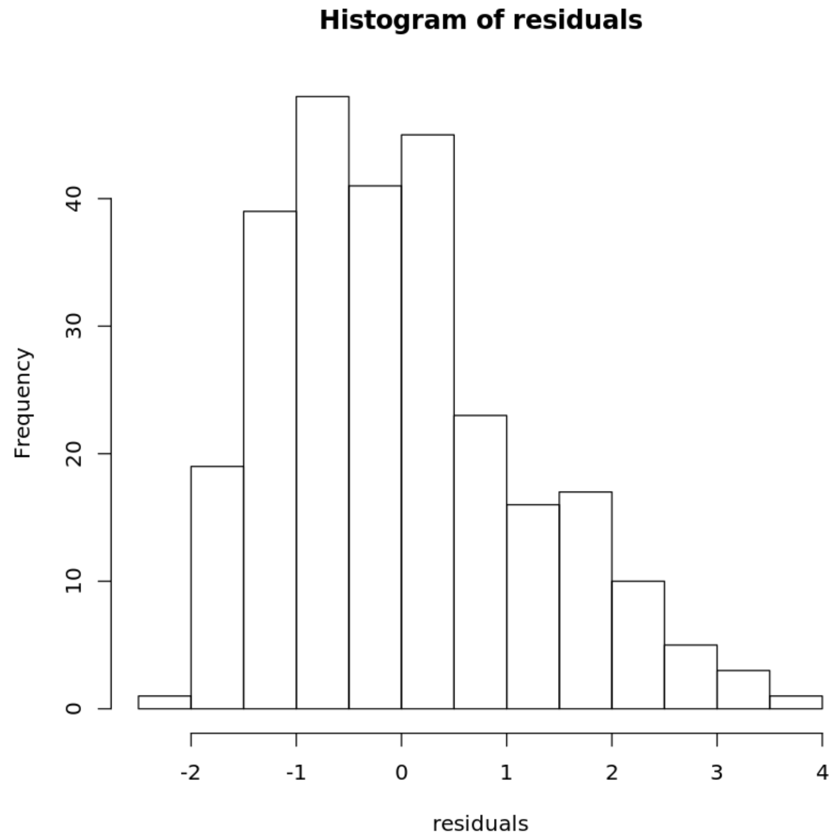
Задание 4

In [16]:
```
model <- lm(data = data, area ~ FFMC + DMC + DC + ISI + temp + RH +
wind + rain + weekend)
```

In [17]:
```r
summary(model)
```

```
Call:
lm(formula = area ~ FFMC + DMC + DC + ISI + temp + RH + wind +
    rain + weekend, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-2.1555 -0.9411 -0.1463  0.5962  3.7106

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.3891016  2.8596312   0.835   0.4042
FFMC         0.0039933  0.0322107   0.124   0.9014
DMC          0.0023040  0.0018073   1.275   0.2035
DC          -0.0001972  0.0004479  -0.440   0.6601
ISI         -0.0344672  0.0271245  -1.271   0.2050
temp        -0.0169655  0.0198256  -0.856   0.3929
RH          -0.0094861  0.0063567  -1.492   0.1368
wind         0.0342085  0.0436574   0.784   0.4340
rain         0.0923050  0.1873218   0.493   0.6226
weekend      0.3176834  0.1610659   1.972   0.0496 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.19 on 258 degrees of freedom
Multiple R-squared:  0.04173,	Adjusted R-squared:  0.008301
F-statistic: 1.248 on 9 and 258 DF,  p-value: 0.2659
```

In [18]:
```
residuals <- residuals(model)
hist(residuals)
```



In [19]:
```
shapiro.test(residuals)
```

```
	Shapiro-Wilk normality test

data:  residuals
W = 0.9583, p-value = 5.678e-07
```

Да, гипотеза о нормальности отвергается

In [20]:
```
library(boot)
```

In [21]:
```
bs <- function(formula, data, indices) {
  d <- data[indices, ] # allows boot to select sample
  fit <- lm(formula, data=d)
  return(coef(fit))
}

# bootstrapping with 1000 replications
results <- boot(data = data, statistic = bs,
   R = 10000, formula = area ~ FFMC + DMC + DC + ISI + temp + RH +
wind + rain + weekend)
```

In [22]:
```
results
```

```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = data, statistic = bs, R = 10000, formula = area ~
    FFMC + DMC + DC + ISI + temp + RH + wind + rain + weekend)


Bootstrap Statistics :
          original        bias     std. error
t1*   2.3891016453 -3.279011e-01 2.4912518672
t2*   0.0039932967  3.675016e-03 0.0285457247
t3*   0.0023040380  3.991860e-05 0.0017551774
t4*  -0.0001972343 -1.571603e-05 0.0004297877
t5*  -0.0344671727 -2.559207e-03 0.0271107012
t6*  -0.0169655348  2.105647e-04 0.0219021458
t7*  -0.0094860969  2.379058e-04 0.0062379488
t8*   0.0342085450  2.016246e-03 0.0428626451
t9*   0.0923050382 -1.795901e-01 0.3440301532
t10*  0.3176833880 -6.864369e-03 0.1741821045
```

Далее для каждого коэффициента будет отдельный доверительный интервал

In [23]:
```
# intercept
boot.ci(results, type="bca", index=1)
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 10000 bootstrap replicates

CALL :
boot.ci(boot.out = results, type = "bca", index = 1)

Intervals :
Level       BCa
95%   (-2.964,  7.078 )
Calculations and Intervals on Original Scale
```

In [24]:
```
# FFMC
boot.ci(results, type="bca", index=2)
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 10000 bootstrap replicates

CALL :
boot.ci(boot.out = results, type = "bca", index = 2)

Intervals :
Level       BCa
95%   (-0.0494,  0.0665 )
Calculations and Intervals on Original Scale
```

In [25]:
```
# DMC
boot.ci(results, type="bca", index=3)
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 10000 bootstrap replicates

CALL :
boot.ci(boot.out = results, type = "bca", index = 3)

Intervals :
Level       BCa
95%   (-0.0011,  0.0058 )
Calculations and Intervals on Original Scale
```

In [26]:
```
# DC
boot.ci(results, type="bca", index=4)
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 10000 bootstrap replicates

CALL :
boot.ci(boot.out = results, type = "bca", index = 4)

Intervals :
Level       BCa
95%   (-0.0010,  0.0007 )
Calculations and Intervals on Original Scale
```

In [27]:
```
# ISI
boot.ci(results, type="bca", index=5)
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 10000 bootstrap replicates

CALL :
boot.ci(boot.out = results, type = "bca", index = 5)

Intervals :
Level       BCa
95%   (-0.0837,  0.0223 )
Calculations and Intervals on Original Scale
```

In [28]:
```
# temp
boot.ci(results, type="bca", index=6)
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 10000 bootstrap replicates

CALL :
boot.ci(boot.out = results, type = "bca", index = 6)

Intervals :
Level       BCa
95%   (-0.0592,  0.0269 )
Calculations and Intervals on Original Scale
```

In [29]:

```r
# RH
boot.ci(results, type="bca", index=7)
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 10000 bootstrap replicates

CALL :
boot.ci(boot.out = results, type = "bca", index = 7)

Intervals :
Level       BCa
95%   (-0.0210,  0.0036 )
Calculations and Intervals on Original Scale
```

In [30]:

```r
# wind
boot.ci(results, type="bca", index=8)
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 10000 bootstrap replicates

CALL :
boot.ci(boot.out = results, type = "bca", index = 8)

Intervals :
Level       BCa
95%   (-0.0503,  0.1175 )
Calculations and Intervals on Original Scale
```

In [31]:

```r
# rain
boot.ci(results, type="bca", index=9)
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 8622 bootstrap replicates

CALL :
boot.ci(boot.out = results, type = "bca", index = 9)

Intervals :
Level       BCa
95%   (-0.7201,  0.2262 )
Calculations and Intervals on Original Scale
```

In [32]:

```r
# weekend
boot.ci(results, type="bca", index=10)
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 10000 bootstrap replicates

CALL :
boot.ci(boot.out = results, type = "bca", index = 10)

Intervals :
Level       BCa
95%   (-0.0208,  0.6618 )
Calculations and Intervals on Original Scale
```

Задание 5

Точечный прогноз

In [33]:
```
x <- subset(data, select = - c(area))
predict(model, x)
```

247: 2.2417117571881 **267:** 1.45293301497938 **253:** 1.92372663572059 **252:** 1.88387394402561 **440:** 1.88436396028392 **139:** 2.03534369574793 **307:** 2.31285900147554 **140:** 2.01962486357565 **511:** 2.03169549780106 **141:** 1.9869539275478 **271:** 1.69075666149419 **363:** 1.56529956003179 **437:** 2.07631088687311 **142:** 1.90900029610311 **143:** 1.92339513982428 **251:** 1.94071751230564 **258:** 2.17114742316508 **144:** 2.22640907657217 **424:** 2.15506908439699 **248:** 1.59816685522571 **246:** 2.26332345634165 **145:** 1.92181912236642 **266:** 1.75978484039553 **146:** 2.06337185655452 **297:** 2.37865257870622 **147:** 2.1909282275394 **148:** 2.09940967597045 **323:** 2.20702568108167 **149:** 1.82971598861642 **361:** 1.89692999398864 **284:** 2.26207893400676 **150:** 1.71226571318023 **151:** 1.97941872277591 **442:** 2.17238693583495 **357:** 1.98982577591173 **483:** 2.20615379474707 **152:** 2.28583351590792 **260:** 2.13843249388714 **153:** 2.12352531426364 **154:** 1.6579378442466 **155:** 2.32735574224474 **340:** 1.96681033283308 **254:** 1.98924210390002 **156:** 1.94169438072757 **264:** 2.13495184679553 **315:** 1.90660935097731 **157:** 2.32721700449968 **158:** 1.93730801725489 **413:** 2.0559088773348 **159:** 2.01022478100663 **350:** 1.80598213613381 **160:** 2.03101339835434 **354:** 1.8176993491207 **161:** 2.16739558878573 **367:** 2.02120399215978 **477:** 1.85727460108559 **162:** 2.00359131277142 **163:** 2.63419124075187 **460:** 2.60416833056525 **473:** 1.90062187048631 **164:** 2.15411888986201 **504:** 1.65003878908781 **487:** 1.76361348779723 **165:** 2.10567665601089 **353:** 1.87285899933042 **486:** 1.56729504786869 **334:** 1.77600565701266 **166:** 2.05581095358145 **465:** 1.99822562361639 **510:** 2.10160642763348 **324:** 2.17360516103889 **344:** 2.03694751246368 **479:** 1.79148529793281 **167:** 1.85651071451856 **428:** 2.10740868290198 **419:** 2.4959707887439 **250:** 1.7925872149223 **168:** 1.82797455666491 **169:** 2.2416711557807 **170:** 1.89728533031861 **171:** 1.97014081699112 **293:** 1.95315515392573 **172:** 1.95882451318309 **173:** 2.38965971315605 **481:** 1.99139635143162 **497:** 1.77436880332461 **245:** 2.23066536784494 **364:** 1.94022408103753 **420:** 2.48202736443573 **174:** 2.07189259158049 **273:** 1.8340640795221 **467:** 2.11859996166587 **476:** 1.91119320167688 **430:** 2.40953169638662 **262:** 1.86722719251523 **459:** 2.4227166486514 **330:** 2.35494371387886 **443:** 1.97576144089719 **175:** 2.17827181581696 **302:** 2.13848970781741 **398:** 2.50528500443545 **397:** 2.04981834885036 **351:** 1.74744201437705 **463:** 2.59236707439844 **318:** 2.15169523506195 **341:** 1.97069754846108 **362:** 1.94870917265132 **407:** 2.66179816708083 **263:** 1.93865635404056 **270:** 1.75133847798203 **320:** 2.20427076775547 **325:** 1.83215006288887 **176:** 2.31363389956756 **177:** 2.21933359527599 **412:** 2.29164618097815 **178:** 2.30406161559004 **179:** 1.91901580425272 **402:**

1.83937275471105 **425:** 2.55556245886622 **308:** 2.30836428124566 **180:**
1.96602617905284 **181:** 2.454634248669 **276:** 2.15184604723316 **464:**
1.99006559714573 **182:** 2.07853777844973 **468:** 2.13798719679252 **365:**
2.03985731651561 **506:** 1.91015316874893 **346:** 2.04967534314099 **383:**
1.82240224832761 **355:** 1.79079887422641 **405:** 2.04783453344292 **345:**
1.97111195598287 **423:** 2.70170937289357 **432:** 2.39650896105766 **183:**
2.30752081754585 **439:** 2.41653801035122 **451:** 1.83726563979351 **513:**
2.28367629642071 **265:** 1.80220055953461 **295:** 1.77982154028896 **331:**
2.31228014901943 **469:** 2.14098668546857 **184:** 1.78919546119275 **466:**
2.36055457319855 **185:** 1.9327699174565 **417:** 2.00694265884718 **186:**
2.04408247164141 **187:** 2.46388950011221 **322:** 2.20702568108167 **188:**
1.98744803622234 **352:** 1.95858188293418 **478:** 1.99133888514546 **189:**
2.28272215721363 **339:** 1.99838960213006 **381:** 2.04493104687056 **409:**
2.3451598301311 **292:** 2.14038250774358 **257:** 2.14008760436035 **360:**
1.87958134074282 **401:** 1.69084520961594 **190:** 2.33512552717733 **191:**
2.03974238534494 **495:** 2.2191421402338 **192:** 2.01627387802583 **193:**
1.83151337361894 **261:** 1.73836568050448 **274:** 2.47102053588676 **272:**
1.74851783988268 **281:** 2.14860942962548 **194:** 2.09954330135725 **452:**
1.64658419457567 **280:** 2.64358738641539 **391:** 2.01116750809774 **445:**
2.29223122502809 **195:** 1.83151337361894 **196:** 1.88509505234646 **475:**
1.8773873616536 **243:** 2.05324491253717 **255:** 2.1297794010357 **278:**
2.64358738641539 **500:** 2.26244539078389 **197:** 2.21933359527599 **198:**
2.14304502496553 **371:** 2.59475685186024 **515:** 2.1720113247219 **275:**
2.14446773285803 **333:** 2.303982760849 **199:** 2.09300303498059 **200:**
2.02626612497292 **201:** 2.068751901429 **202:** 2.28906963313524 **385:**
2.65164903383708 **369:** 2.2886533130149 **203:** 2.26406297421709 **356:**
1.98982577591173 **204:** 1.92795988979467 **205:** 2.16890814473151 **312:**
2.2040039094029 **206:** 1.98341755399631 **498:** 1.83600188840706 **434:**
2.31845813940648 **207:** 2.14818269301154 **332:** 2.43133332080892 **386:**
2.35381221272921 **382:** 2.05720627864323 **488:** 1.7176045581434 **208:**
2.0460764952183 **277:** 2.64358738641539 **375:** 2.21628936065878 **209:**
2.04742444688165 **366:** 1.98412806792329 **279:** 2.64358738641539 **210:**
1.72494580169848 **211:** 2.35958207505235 **285:** 2.23042720953249 **387:**
2.06352951440928 **282:** 2.60664632840576 **212:** 1.78568095193935 **213:**
1.6579378442466 **485:** 2.12230417958924 **214:** 2.24180465069025 **347:**
2.0184315211683 **215:** 2.49375757960153 **216:** 2.49375757960153 **389:**
2.12591345021819 **217:** 2.12305613639752 **392:** 1.87432337716858 **218:**
1.88234096514951 **219:** 2.1472279314022 **220:** 2.20163676382696 **221:**
2.22635018477209 **321:** 2.22394882089762 **222:** 1.80988016267012 **223:**
2.24180465069025 **224:** 1.93891789837999 **225:** 1.93306234613107 **472:**
2.04643521526451 **376:** 2.21754409293594 **499:** 1.84562672302744 **384:**
2.15576821533675 **494:** 2.03393950725122 **489:** 1.62221189722685 **226:**
2.41041545234947 **227:** 1.79345315989282 **505:** 1.66461454639333 **396:**
2.23717932663427 **514:** 2.11986166298716 **338:** 2.00278603857519 **228:**
2.24885157850644 **470:** 2.4509098609007 **229:** 2.38054544294051 **474:**
2.0512283293609 **393:** 2.65671255191667 **230:** 2.20914491275526 **458:**
2.12692975309962 **294:** 2.17516111155346 **231:** 2.11896433835953 **232:**
2.37954211344885 **233:** 2.04019525601341 **234:** 2.00472007497876 **235:**

2.49435555500593 **378:** 2.48815110403963 **421:** 2.16995651283235 **236:** 2.39285370479898 **237:** 2.2421936447083 **238:** 2.04985272065026 **480:** 1.92247027243306

Прогноз для среднего

In [34]:
```
x_median <- lapply(x, median, na.rm = T, USE.NAMES = FALSE)
predict(model, x_median)
```

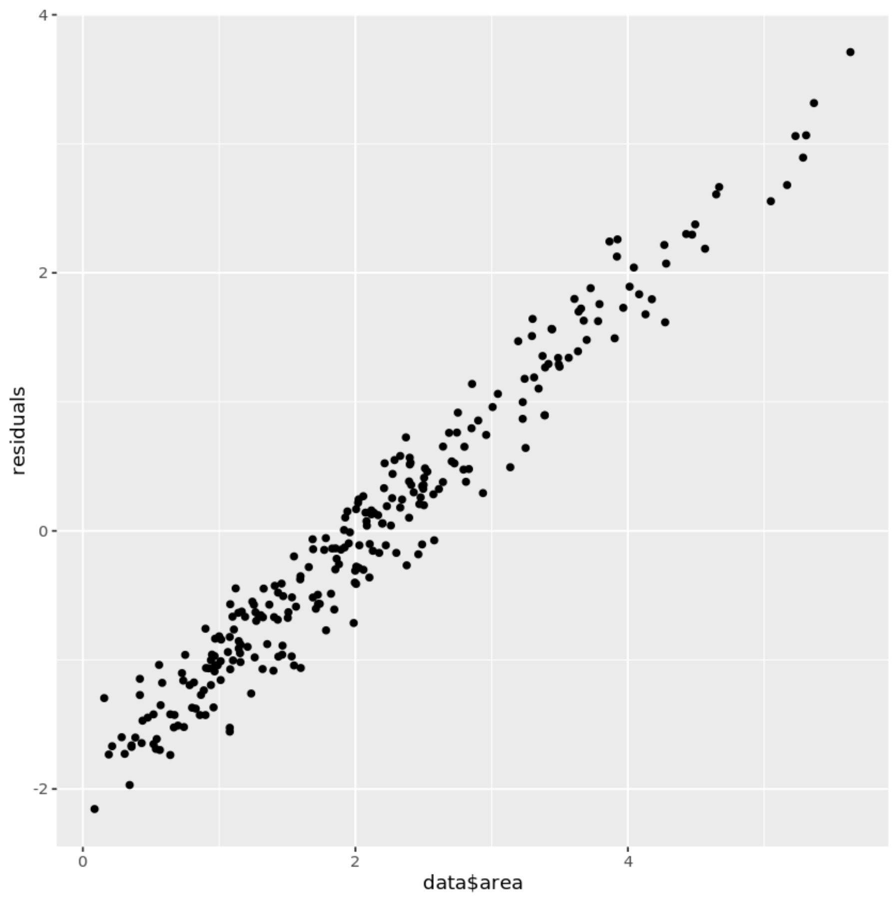
1: 1.99857144665025

Задание 6

Считаю, что ветер может порождать гетероскедастичность, так как при большой скорости ветра, область горения может сильно расширяться, если пожар не потушить. Следовательно, дисперсия площади пожара увеличивается при увеличении скорости ветра.

Задание 7

In [35]:
```
qplot(data$area, residuals)
```



очень ярко выраженная гетероскедастичность

Проведем тест Голдфельда-Квандта!!!

Проверил все переменные. Оказалось, что наименьшее p value оказалось в данных случаях:

In [39]:
```
install.packages('lmtest')
library(lmtest)
gqtest(model, order.by = ~ rain, data = data, fraction = 0.2)
```

```
Installing package into '/srv/rlibs'
(as 'lib' is unspecified)

also installing the dependency 'zoo'


Loading required package: zoo


Attaching package: 'zoo'


The following objects are masked from 'package:base':

    as.Date, as.Date.numeric



        Goldfeld-Quandt test

data:  model
GQ = 4.983, df1 = 98, df2 = 97, p-value = 2.581e-14
alternative hypothesis: variance increases from segment 1 to 2
```

In [40]:
```
gqtest(model, order.by = ~ weekend, data = data, fraction = 0.2)
```

```
        Goldfeld-Quandt test

data:  model
GQ = 4.7993, df1 = 98, df2 = 97, p-value = 8.744e-14
alternative hypothesis: variance increases from segment 1 to 2
```

Моя предпосылка не верна, логически результаты описать не могу(

Задание 8

С помощью дополнительной регрессии посчитаем веса каждого наблюдения в модели

In [47]:
```
residuals_model <- lm(data = data, abs(residuals) ~ FFMC + DMC + DC
+ ISI + temp + RH + wind + rain + weekend)
```

In [48]:
```
weights <- 1/(residuals_model$fitted.values) ^ 2
```

In [50]:
```
weighted_model <-lm(data = data, area ~ FFMC + DMC + DC + ISI + tem
p + RH + wind + rain + weekend, weights = weights)
```

In [51]:
```
summary(weighted_model)
```

```
Call:
lm(formula = area ~ FFMC + DMC + DC + ISI + temp + RH + wind +
    rain + weekend, data = data, weights = weights)

Weighted Residuals:
    Min      1Q  Median      3Q     Max 
-2.0776 -0.9583 -0.1829  0.7065  3.9224 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)   
(Intercept)  2.3635416  2.1533651   1.098  0.27340   
FFMC         0.0028211  0.0241535   0.117  0.90711   
DMC          0.0027224  0.0017826   1.527  0.12793   
DC          -0.0000538  0.0004078  -0.132  0.89514   
ISI         -0.0298989  0.0238893  -1.252  0.21186   
temp        -0.0237452  0.0176068  -1.349  0.17864   
RH          -0.0101663  0.0054703  -1.858  0.06424 . 
wind         0.0544546  0.0392185   1.388  0.16618   
rain         0.1276734  0.0542543   2.353  0.01936 * 
weekend      0.4318989  0.1583978   2.727  0.00684 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.244 on 258 degrees of freedom
Multiple R-squared:  0.07279,	Adjusted R-squared:  0.04044 
F-statistic:  2.25 on 9 and 258 DF,  p-value: 0.01945
```

Как мы видим, результат улучшился, значимых переменных стало больше

Задание 9

In [42]:
```r
install.packages('plm')
library(plm)
```

```
Installing package into ‘/srv/rlibs’
(as ‘lib’ is unspecified)

also installing the dependencies ‘stringr’, ‘miscTools’, ‘bibtex’,
‘gbRd’, ‘bdsmatrix’, ‘sandwich’, ‘maxLik’, ‘Rdpack’, ‘Formula’
```

Оценки в форме Уайта можно получить, использовав стандартную формулу дисперсии коэффициентов, однако вместо $\sigma^2 I$ в формуле будет диагональная матрица с квадратами посчитанных остатков!!!

In [52]:
```r
coeftest(model, .vcov = vcovHC(model, type = 'HC0'))
```

```
t test of coefficients:

               Estimate  Std. Error t value Pr(>|t|)
(Intercept)  2.38910165  2.85963116  0.8355  0.40423
FFMC         0.00399330  0.03221068  0.1240  0.90143
DMC          0.00230404  0.00180733  1.2748  0.20352
DC          -0.00019723  0.00044795 -0.4403  0.66008
ISI         -0.03446717  0.02712455 -1.2707  0.20498
temp        -0.01696553  0.01982564 -0.8557  0.39294
RH          -0.00948610  0.00635674 -1.4923  0.13684
wind         0.03420855  0.04365737  0.7836  0.43401
rain         0.09230504  0.18732176  0.4928  0.62260
weekend      0.31768339  0.16106585  1.9724  0.04963 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

t-статистики везде увеличились

In [56]:
```r
coeftest(model, .vcov = vcovHC(model, type = 'HC3'))
```

```
t test of coefficients:

               Estimate  Std. Error t value Pr(>|t|)
(Intercept)  2.38910165  2.85963116  0.8355  0.40423
FFMC         0.00399330  0.03221068  0.1240  0.90143
DMC          0.00230404  0.00180733  1.2748  0.20352
DC          -0.00019723  0.00044795 -0.4403  0.66008
ISI         -0.03446717  0.02712455 -1.2707  0.20498
temp        -0.01696553  0.01982564 -0.8557  0.39294
RH          -0.00948610  0.00635674 -1.4923  0.13684
wind         0.03420855  0.04365737  0.7836  0.43401
rain         0.09230504  0.18732176  0.4928  0.62260
weekend      0.31768339  0.16106585  1.9724  0.04963 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Задание 10

In [59]:
```
x.pca <- prcomp(x, scale = TRUE)
```

In [60]:
```
pca1 <- x.pca$x[, 1]
pca2 <- x.pca$x[, 2]
```

In [61]:
```
summary(x.pca)
```

```
Importance of components:
                          PC1    PC2    PC3    PC4    PC5     PC6
PC7
Standard deviation     1.7680 1.1260 1.0982 1.0041 0.9524 0.82588
0.56082
Proportion of Variance 0.3473 0.1409 0.1340 0.1120 0.1008 0.07579
0.03495
Cumulative Proportion  0.3473 0.4882 0.6222 0.7342 0.8350 0.91078
0.94573
                           PC8     PC9
Standard deviation     0.55334 0.42692
Proportion of Variance 0.03402 0.02025
Cumulative Proportion  0.97975 1.00000
```

Две первые компоненты объясяют 48 процентов дисперсии

In [62]:
```
model_pca <- lm(data = data, area ~ pca1 + pca2)
```

In [63]:
```
summary(model_pca)
```

```
Call:
lm(formula = area ~ pca1 + pca2, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-1.9665 -0.9536 -0.1058  0.7181  3.5083

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.09249    0.07314  28.609   <2e-16 ***
pca1         0.03431    0.04145   0.828    0.409
pca2        -0.02426    0.06508  -0.373    0.710
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.197 on 265 degrees of freedom
Multiple R-squared:  0.0031,    Adjusted R-squared:  -0.004424
F-statistic: 0.4121 on 2 and 265 DF,  p-value: 0.6627
```

нет, главные компоненты оказались не значимы