Task 3

Generate a 500 element sample from $\mathcal{N}(1, 1)$. Find $\overline{X}, S^2$, confint

In [223]:
```
mean <- 1
sd <- 1
sample_size <- 500
```

In [224]:
```
our_sample <- rnorm(sample_size, mean = mean, sd = sd)
```

In [225]:
```
mean(our_sample) # Mean
```

0.952332253627821

In [226]:
```
var(our_sample) # Variance
```

1.01900682121439

Воспользуемся правилом двух сигм, которое дает нам 95(почти) процентный доверительный интервал для X

In [227]:
```
conf_int <- c(mean - 2 * sd ** 2, mean + 2 * sd ** 2)
```

In [228]:
```
conf_int
```

-1 3

Проеверим, сколько элементов выборки не попали в данный интервал

In [229]:
```
out_number <- 0
for(elem in our_sample){
    if ((elem < -1) || (3 < elem))
        out_number <- out_number + 1
}
```

In [230]:
```
out_number
```

22

In [231]:
```
out_number / sample_size # 0.044 не попадает, то что мы и хотели)
```

0.044

Task4

Open given data, choose 3 regressors and build a linear regression. find R^2 , and draw Q-Q plot on residuals from the model

In [76]:
```
data4 <- read.table("./housing.data.txt")
```

In [356]:
```
head(data4)
```

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14
0.00632	18	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
0.02729	0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
0.06905	0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2
0.02985	0	2.18	0	0.458	6.430	58.7	6.0622	3	222	18.7	394.12	5.21	28.7

In [80]:
```
data4_for_linreg <- data4[c('V3', 'V6', 'V8', 'V14')]
```

In [357]:
```
head(data4_for_linreg)
```

V3	V6	V8	V14
2.31	6.575	4.0900	24.0
7.07	6.421	4.9671	21.6
7.07	7.185	4.9671	34.7
2.18	6.998	6.0622	33.4
2.18	7.147	6.0622	36.2
2.18	6.430	6.0622	28.7

In [89]:
```
data4.lr1 <- lm(V14 ~ V3 + V6 + V8, data = data4_for_linreg)
data4.lr2 <- lm(V14 ~ V6 + V7 + V8, data = data4[c('V6', 'V7', 'V8'
, 'V14')])
data4.lr3 <- lm(V14 ~ V5 + V11 + V13, data = data4[c('V5', 'V11', '
V13', 'V14')])
```

In [91]:
```
summary(data4.lr1)
print('-----------')
summary(data4.lr2)
print('-----------')
summary(data4.lr3)
```

```
Call:
lm(formula = V14 ~ V3 + V6 + V8, data = data4_for_linreg)

Residuals:
    Min      1Q  Median      3Q     Max
-17.676  -3.207  -0.365   2.686  39.548

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -19.41680    3.29971  -5.884 7.31e-09 ***
V3           -0.43458    0.06142  -7.075 5.03e-12 ***
V6            7.71008    0.43272  17.818  < 2e-16 ***
V8           -0.43888    0.18813  -2.333     0.02 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.248 on 502 degrees of freedom
Multiple R-squared:  0.5413,    Adjusted R-squared:  0.5385
F-statistic: 197.4 on 3 and 502 DF,  p-value: < 2.2e-16

[1] "-----------"

Call:
lm(formula = V14 ~ V6 + V7 + V8, data = data4[c("V6", "V7", "V8",
    "V14")])

Residuals:
    Min      1Q  Median      3Q     Max
-21.179  -2.808  -0.215   2.166  40.505

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -21.87279    3.17796  -6.883 1.76e-11 ***
V6            8.44063    0.41047  20.563  < 2e-16 ***
V7           -0.09942    0.01511  -6.582 1.17e-10 ***
V8           -0.48038    0.20028  -2.399   0.0168 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.286 on 502 degrees of freedom
Multiple R-squared:  0.5356,    Adjusted R-squared:  0.5328
F-statistic:   193 on 3 and 502 DF,  p-value: < 2.2e-16

[1] "-----------"
```

```
Call:
lm(formula = V14 ~ V5 + V11 + V13, data = data4[c("V5", "V11",
    "V13", "V14")])

Residuals:
     Min       1Q   Median       3Q      Max
-12.2428  -3.6412  -0.8629   1.8791  26.9049

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 54.05255    2.58783  20.887   <2e-16 ***
V5          -0.01226    2.75640  -0.004    0.996
V11         -1.14528    0.12834  -8.924   <2e-16 ***
V13         -0.82006    0.04736 -17.315   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.785 on 502 degrees of freedom
Multiple R-squared:  0.6067,    Adjusted R-squared:  0.6043
F-statistic: 258.1 on 3 and 502 DF,  p-value: < 2.2e-16
```

In [100]:
```
print(c(summary(data4.lr1)$r.squared, 'fst_model'))
print(c(summary(data4.lr2)$r.squared, 'snd_model'))
print(c(summary(data4.lr3)$r.squared, 'thrd_model'))
```

```
[1] "0.54127176772878" "fst_model"
[1] "0.535597869878533" "snd_model"
[1] "0.60665463084166" "thrd_model"
```

In [110]:
```
resid1 <- summary(data4.lr1)$residual
resid2 <- summary(data4.lr2)$residual
resid3 <- summary(data4.lr3)$residual
```

In [113]:
```
shapiro.test(resid1)
shapiro.test(resid2)
shapiro.test(resid3)
```

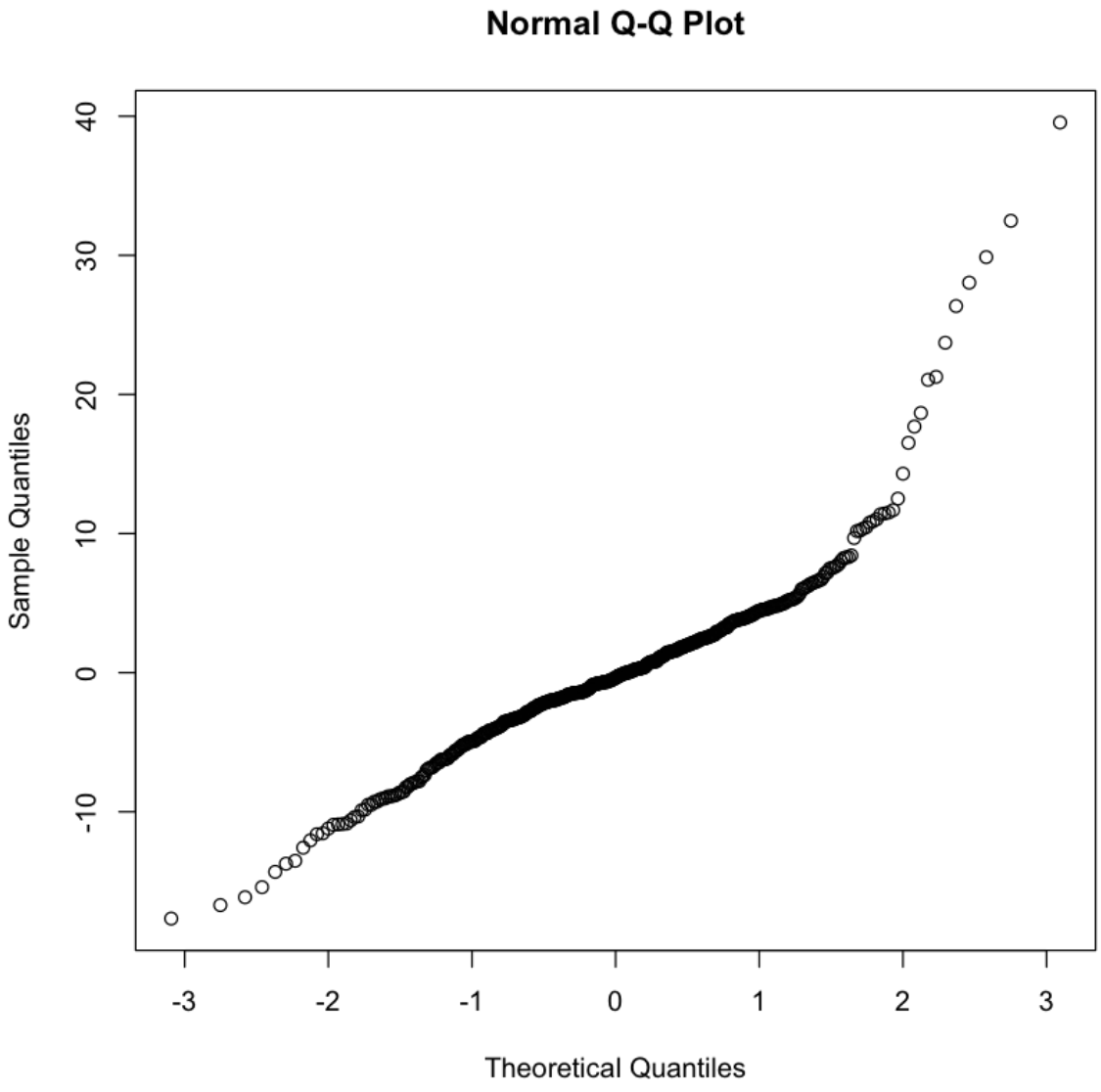
```
        Shapiro-Wilk normality test

data:  resid1
W = 0.89827, p-value < 2.2e-16

        Shapiro-Wilk normality test

data:  resid2
W = 0.88723, p-value < 2.2e-16

        Shapiro-Wilk normality test

data:  resid3
W = 0.89576, p-value < 2.2e-16
```

Остатки не распределены нормально. так как гипотеза отвергается

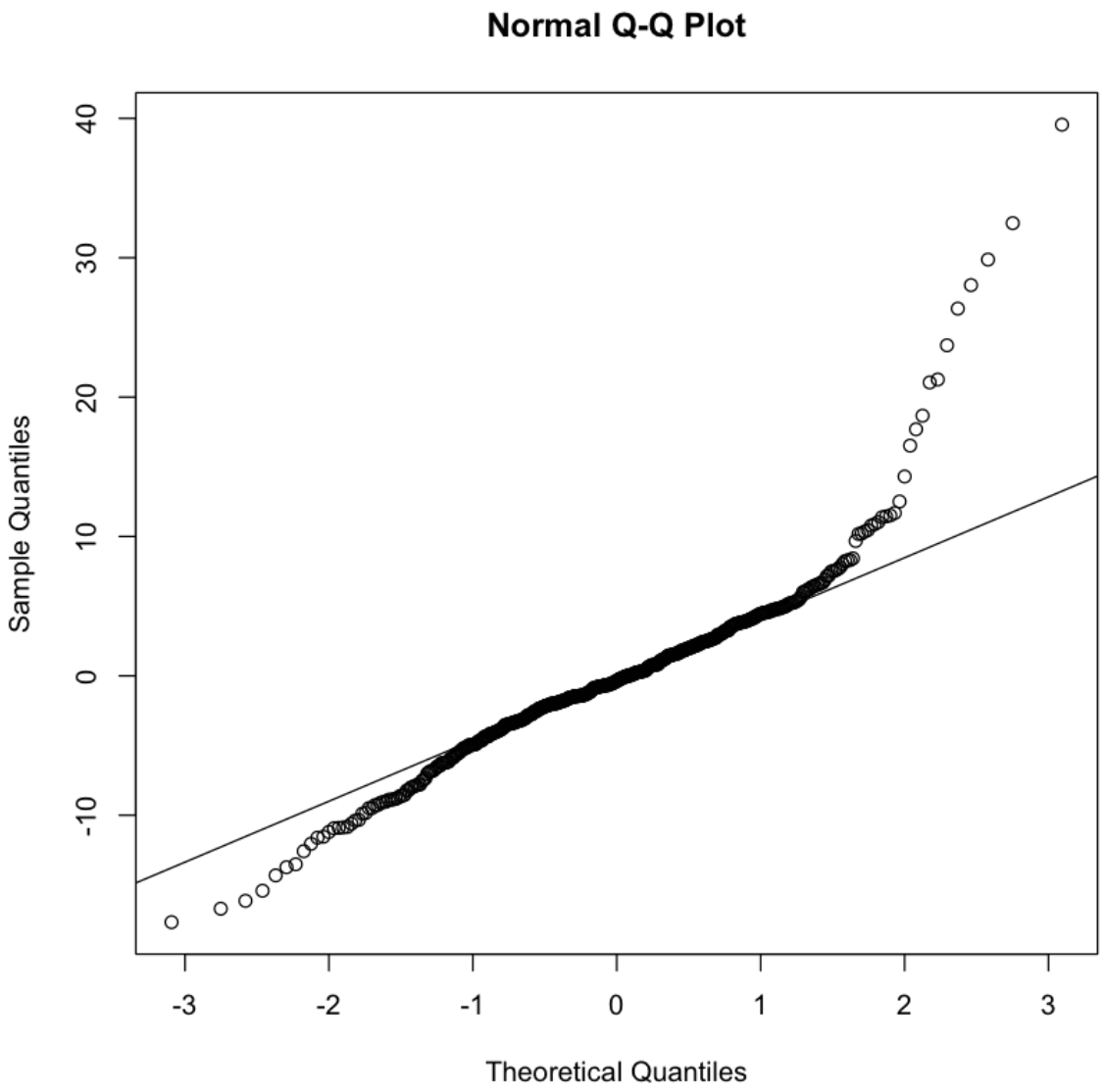
In [121]:
```
qqplot
```

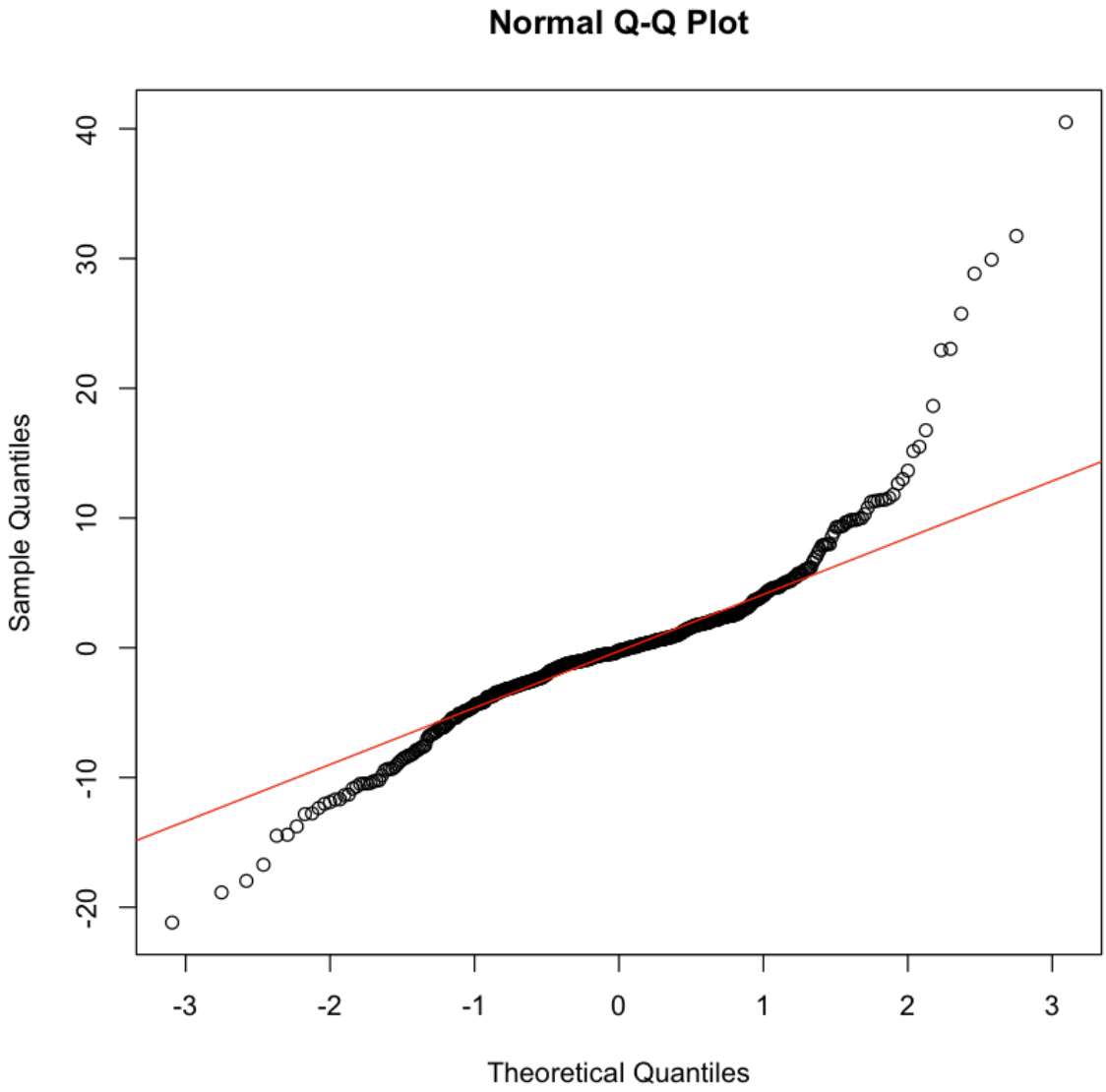
```
function (x, y, plot.it = TRUE, xlab = deparse(substitute(x)),
    ylab = deparse(substitute(y)), ...)
{
    sx <- sort(x)
    sy <- sort(y)
    lenx <- length(sx)
    leny <- length(sy)
    if (leny < lenx)
        sx <- approx(1L:lenx, sx, n = leny)$y
    if (leny > lenx)
        sy <- approx(1L:leny, sy, n = lenx)$y
    if (plot.it)
        plot(sx, sy, xlab = xlab, ylab = ylab, ...)
    invisible(list(x = sx, y = sy))
}
```

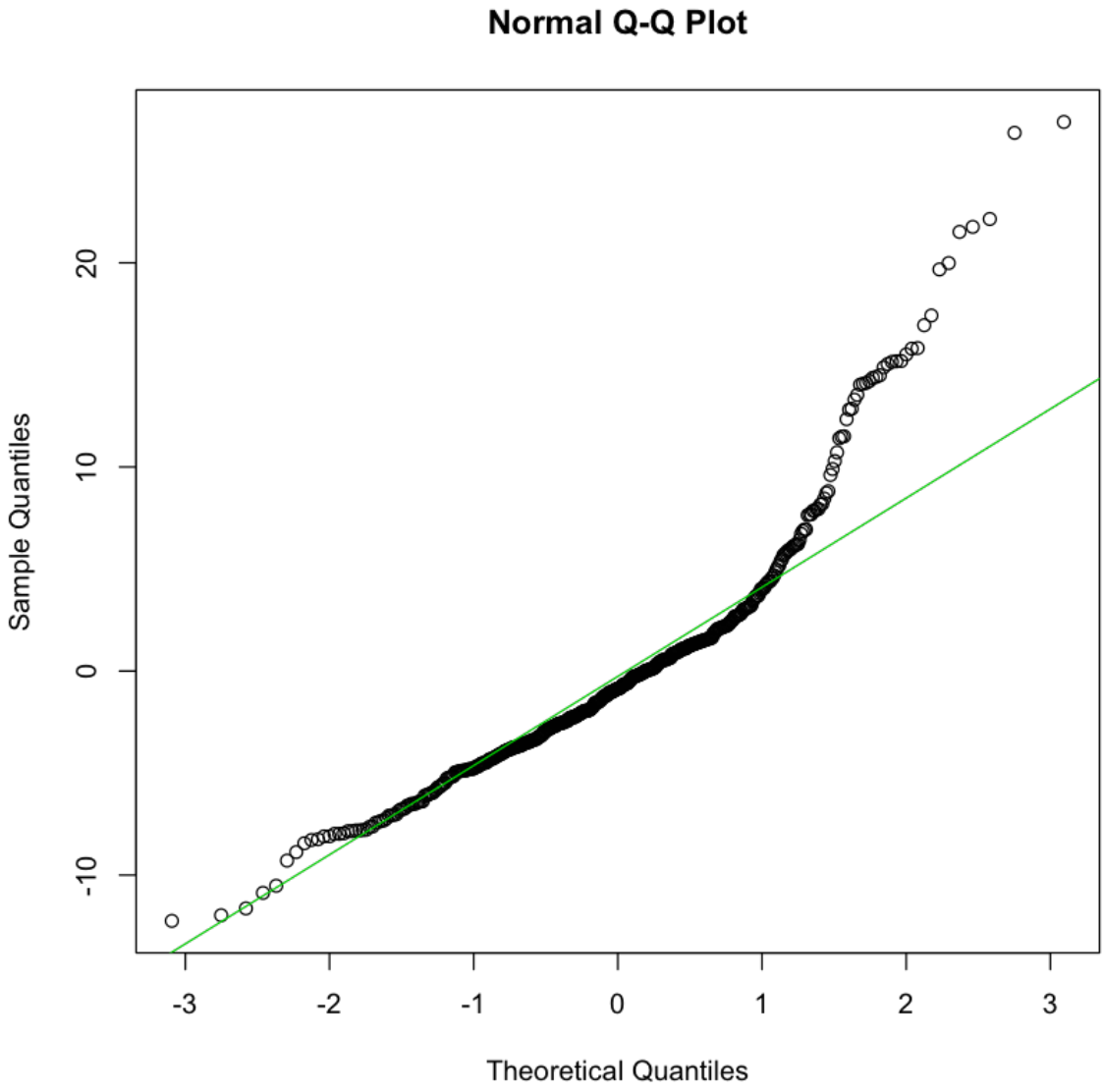
In [123]:
```
qqqq <= qqnorm(resid1)
```



In [127]:
```
qqnorm(resid1); qqline(resid1, col = 1)
qqnorm(resid2); qqline(resid1, col = 2)
qqnorm(resid3); qqline(resid1, col = 3)
```




Normal Q-Q Plot
Sample Quantiles
Theoretical Quantiles
40
30
20
10
0
-10
-20
-3
-2
-1
0
1
2
3



Да и по графику видно, что они не распределены нормально, слишком уж сильно они отклоняются от прямой

Task 5

$X \sim f(x), Y \sim g(x). \quad \frac{f(x)}{g(x)} \leq c$

Algo: 1) Generate y from Y

2) assign value from first step as x, return to the step 1 otherwise

$f(x) = 20x(1-x)^3, x \in [0,1]$, выбрав в качестве g(x) равномерную плотность на $U[0,1]$.

In [140]:
```
seq.gen <- function(c_val) {
  c = c_val
  function() {
    cur_val <- -1
    while (cur_val < runif(1)) {
      new_y <- runif(1)
      cur_val <- (20 * new_y * (1 - new_y) ** 3.) / c
    }
    return(cur_val)
  }
}
```

In [359]:
```
ksi <- seq.gen(12)
```

In [365]:
```
ksi()
```

0.136843833830102

Докажем, что действуя таким алгоритмом мы все же получим величину с нужной плотностью

$$P(X \leq x) = P(Y < x \mid U \leq \frac{f(Y)}{cg(Y)}) = \frac{P(Y<x, U \leq \frac{f(Y)}{cg(Y)})}{P(U \leq \frac{f(Y)}{cg(Y)})}$$

А это то же самое что и:

$$\frac{\int_{-\infty}^{x} \int_{0}^{\frac{f(y)}{cg(y)}} g(y) du dx}{\int_{-\infty}^{+\infty} \int_{0}^{\frac{f(y)}{cg(y)}} g(y) du dx}$$

где g - равномерное распределение. Тогда это превращается в

$$c \cdot \int_{-\infty}^{x} \frac{f(y)}{c} dy = \int_{-\infty}^{x} f(y) dy$$

О, так это и есть нужное нам распределение!

Task6 AIC & BIC

Прямой алгоритм отбора признаков по AIC (BIC) производится так: начинаем с 0 признаков в модели, далее подбираем модель с одним признаком, у которой наи- больший AIC (BIC), потом варьируем второй признак, считая первый признак уже включенным в модель, и так далее. Откройте выданные вам данные в R и реализуй- те на них (использовать bestglm нельзя) прямой алгоритм отбора признаков по AIC и BIC. Одинаковые ли модели получились? Как вы это объясните?

$$AIC = 2k + n(ln\frac{RSS}{n} + 1)$$
$$BIC = 2klnn + n(ln\frac{RSS}{n} + 1)$$
$$RSS = \sum(y_i - \hat{y_i})^2$$

In [198]:
```
remove(append(c(1,2,3,4), 78.1))
```

```
Error in remove(append(c(1, 2, 3, 4), 78.1)): ... must contain nam
es or character strings
Traceback:

1. remove(append(c(1, 2, 3, 4), 78.1))
2. stop("... must contain names or character strings")
```

In [339]:
```
epsilon <- 1e-6
prev_aic <- 1e7
cur_aic <- 1e6
top_columns <- c('V14')
cols_to_test <- c('V14')
remaining_columns <- colnames(data4)[-14]
while(prev_aic - cur_aic > epsilon || length(remaining_columns) > 0
) {
    best_column <- ''
    best_aic <- 1e7
    for(column in remaining_columns) {
        cols_to_test <- append(cols_to_test, column)

        coeffs <- lm(data4[cols_to_test])$coefficients
        y_hat <- c()
        for(i in 1:nrow(data4[cols_to_test])) {
            row <- data4[cols_to_test][i,]
            y_hat <- append(y_hat, coeffs[1] + sum(coeffs[-1] * row
[-1]))
        }
        rss <- sum((data4['V14'] - y_hat) ** 2)
        k <- length(cols_to_test)
        n <- length(y_hat)
        aic <- 2 * k + n * (log(rss / n) + 1)
        if(aic < best_aic) {
            best_aic <- aic
            best_column <- column
        }
        cols_to_test <- head(cols_to_test, -1)
    }
    top_columns <- append(top_columns, best_column)
    remaining_columns <- setdiff(remaining_columns, best_column)
    cols_to_test <- top_columns
    prev_aic <- cur_aic
    cur_aic <- best_aic
}
```

In [340]:
```
top_columns # for AIC
```

'V14' 'V13' 'V6' 'V11' 'V8' 'V5' 'V4' 'V12' 'V2' 'V1' 'V9' 'V10'
'V3' 'V7'

In [341]:
```
length(top_columns)
```

14

Чего и следовало ожидать, ведь у нас признаков не так уж и много, они все важны для нас

The same for BIC Model

In [344]:
```
remaining_columns
```

'V1' 'V2' 'V3' 'V4' 'V5' 'V6' 'V7' 'V8' 'V9' 'V10' 'V11' 'V12'
'V13'

In [349]:
```
cols_to_test
```

'V14' '' 'V1'

In [347]:
```
cols_to_test[2]
```

''

In [353]:
```
epsilon <- 1e-6
prev_bic <- 1e20
cur_bic <- 1e20
top_columns_bic <- c('V14')
cols_to_test <- c('V14')
remaining_columns <- colnames(data4)[-14]
while(prev_bic - cur_bic > epsilon || length(remaining_columns) > 0
) {
    best_column <- ''
    best_bic <- 1e15
    for(column in remaining_columns) {
        cols_to_test <- append(cols_to_test, column)

        coeffs <- lm(data4[cols_to_test])$coefficients
        y_hat <- c()
        for(i in 1:nrow(data4[cols_to_test])) {
            row <- data4[cols_to_test][i,]
            y_hat <- append(y_hat, coeffs[1] + sum(coeffs[-1] * row
[-1]))
        }
        rss <- sum((data4['V14'] - y_hat) ** 2)
        k <- length(cols_to_test)
        n <- length(y_hat)
        bic <- 2 * k * log(n) + n * (log(rss / n) + 1)
        if(bic < best_bic) {
            best_bic <- bic
            best_column <- column
        }
        cols_to_test <- head(cols_to_test, -1)
    }
    top_columns_bic <- append(top_columns_bic, best_column)
    remaining_columns <- setdiff(remaining_columns, best_column)
    cols_to_test <- top_columns_bic
    prev_bic <- cur_bic
    cur_bic <- best_bic
}
```

In [355]:
```
length(top_columns_bic)
```

14

Все же модели оказались одинаковыми в силу важности всех признаков, так как их не так уж и много при таком размере выборки

In [ ]: