Централна гранична теорема. Нормална pp-plot.

Способността на R да симулира реализации на случайни величини ни дава възможност да провеждаме експерименти, да наблюдаваме резултатите от тях и бързо да отговаряме на въпроси, свързани с тях.

Централна гранична теорема (ЦГТ)

В тази точка ще извършим отново наблюдения върху ЦГТ. Нека този път да я разгледаме в нейния общ вид за суми.

Централна гранична теорема, за суми: Нека $X_1, X_2, \ldots, X_n$ са независими наблюдения върху случайната величина ξ с математическо очакване $E\xi = \mu$ и дисперсия $D\xi = \sigma^2$. [1] Тогава, при голям обем на извадката, тяхната сума $X_1 + X_2 + \ldots + X_n$ е приблизително $N(n\mu, n\sigma^2)$. Като центрираме и нормираме получаваме, че при $n \to \infty$

$$\frac{Sn - n\mu}{\sigma\sqrt{n}} = \frac{X1 + X2 + \ldots + Xn - n\mu}{\sigma\sqrt{n}} \to N(0,1).$$

Когато разделим числителя и знаменателя на n получаваме

Централна гранична теорема, за средното аритметично:

Нека $X_1, X_2, \ldots, X_n$ са независими наблюдения върху случайната величина ξ с математическо очакване $E\xi = \mu$ и дисперсия $D\xi = \sigma^2$. Тогава, при голям обем на извадката, тяхното следно аритметично $\bar{X} = \frac{X1 + X2 + \ldots + Xn}{n}$ е приблизително $N(\mu, \frac{\sigma^2}{n})$. Като центрираме и нормираме получаваме, че при $n \to \infty$

$$\frac{\bar{X} - \mu}{\frac{\sigma}{\sqrt{n}}} \to N(0, 1)$$

Т.е. при големи n, центрираното и нормирано следно аритметично е приблизително стандартно нормално разпределена величина.

Централна гранична теорема за пропорции.

Като си припомним, че:

1. средното аритметично е обобщение на пропорциите и по-точно

$$p_n(A) = \frac{\text{брой сбъдвания на A}}{n} = \frac{I_{A1} + I_{A2} + \cdots + I_{An}}{n}$$

където I_{A1} е 1 ако A се е сбъднало при първия опит и нула ако A не се е сбъднало при първия опит, I_{A2} е 1 ако A се е сбъднало при втория опит и нула ако A не се е сбъднало при втория опит, и т.н. I_{An} е 1 ако A се е сбъднало при n-тия опит и нула ако A не се е сбъднало при n-тия опит;

2. математическото очакване на индикаторите е равно на p = P(A)
3. дисперсията на индикаторите е p(1 - p) = P(A)[1 - P(A)]

центрираме и нормираме сумата от индикаторите с тези величини, прилагаме ЦГТ за средно аритметично на такива индикатори и получаваме ЦГТ за относителни дялове (пропорции).

Централна гранична теорема, за пропорции: Нека имаме независими наблюдения върху сбъдването на събитието A, което е с вероятност p = P(A). Нека $p_n(A)$ е пропорцията на сбъдванията на A спрямо броят на опитите, т.е. броят на сбъдванията на A при първите n опита, разделен на броят на опитите, т.е. средното аритметично на индикаторите на A. Тогава, при голям

[1] На по-теоретичен език това означава, че $X_1, X_2, \ldots, X_n$ и ξ са независими еднакво разпределени случайни величини със средни $EX_1 = EX_2 = \ldots = EX_n = E\xi = \mu$ и дисперсия $DX_1 = DX_2 = \ldots = DX_n = D\xi = \sigma^2$.

обем на извадката, това средно аритметично $p_n(A)$ е приблизително N(p, $\frac{p(1-p)}{n}$). Като го центрираме и нормираме получаваме, че при n → ∞

$$\frac{p_n(A)-p}{\frac{\sqrt{p(1-p)}}{\sqrt{n}}} \to \text{N}(0, 1).$$

Т.е. това е ЦГТ за средното аритметично, но приложена за индикатори и в резултат се получава ЦГТ за пропорции. Ако приложим ЦГТ за суми за сумата от индикаторите на А получаваме

$$\frac{S_n-np}{\sqrt{np(1-p)}} = \frac{I_{A1}+I_{A2}+\cdots+I_{An}-np}{\sqrt{np(1-p)}} = \frac{np_n(A)-np}{\sqrt{np(1-p)}} \to \text{N}(0, 1).$$

Как можем да илюстрираме всички тези формули и да проверим тяхната истинност? Симулации те са един прекрасен начин.

Например: Нека първо симулираме 1 наблюдение върху величината

$$X_n := \frac{S_n - np}{\sqrt{np(1-p)}} = \frac{I_{A1} + I_{A2} + \cdots + I_{An} - np}{\sqrt{np(1-p)}} = \frac{np_n(A) - np}{\sqrt{np(1-p)}}$$

при n = 10 и p = 0.25. Получаваме

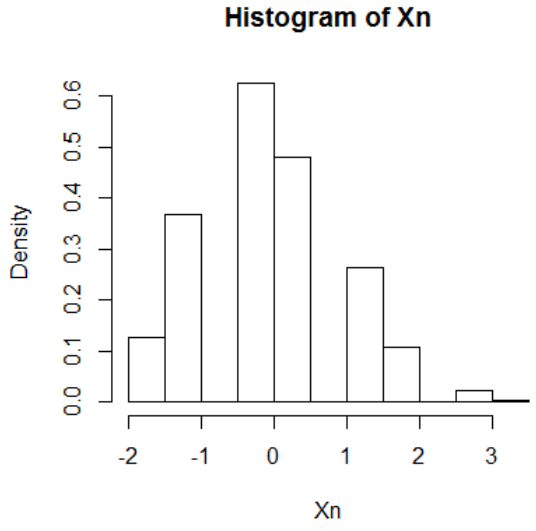
```
> n = 10; p = .25;
> Sn =  rbinom(1, n, p)
> Xn = (Sn - n*p)/sqrt(n*p*(1-p))
[1] -0.3651484
```

За да изчертаем хистограма или да получим оценка на каквато и да е характеристика на тази случайна величина Xn ни трябват повече от едно наблюдения. За това нека сега да симулираме 500 такива наблюдения. В случая това може да стане само с една единствена промяна на първия параметър на функцията ***rbinom***, която симулира толкова наблюдения върху биномно разпределена случайна величина, колкото е първият и параметър. Получаваме

```
> n = 10; p = .25;
>  Sn = rbinom(500, n, p)
> Xn = (Sn - n*p)/sqrt(n*p*(1-p))
```

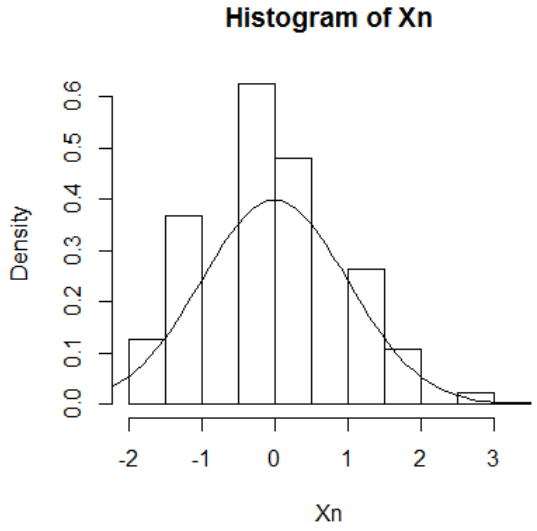
Вече сме готови да начертаем хистограма на симулираните наблюдения върху случайната величина Xn

```
> hist(Xn, prob = T)
```



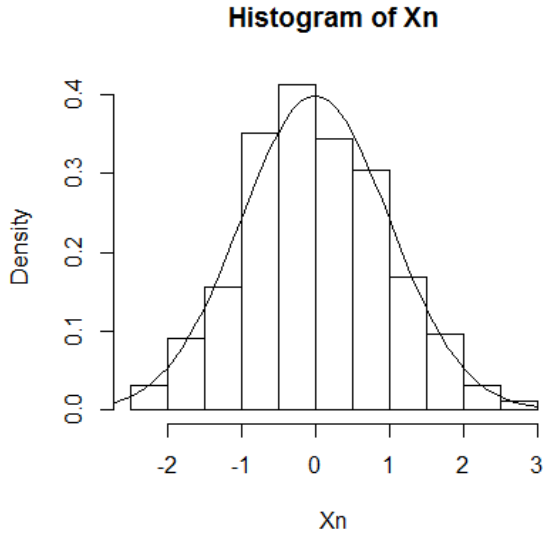
Сега да сравним тази хистограма със стандартната нормална крива, която според централната гранична теорема, би трябвало да се появи при големи n.

```
>  y = dnorm(seq(from = -3, to = 3, by = 0.1), mean = 0, sd = 1)
>  lines(seq(from = -3, to = 3, by = 0.1), y, type = "l")
```



Нека увеличим n = 1000. Според Централната гранична теорема новата хистограма би трябвало много повече да се доближи до стандартната нормална крива. Да проверим. Отново симулираме 500 наблюдения над случайната величина Xn.

```
> n = 1000; p = .25;
> Sn = rbinom(500, n, p)
> Xn = (Sn - n*p)/sqrt(n*p*(1-p))
> hist(Xn, prob = T)
>  y = dnorm(seq(from = -3, to = 3, by = 0.1), mean = 0, sd = 1)
>  lines(seq(from =-3, to = 3, by = 0.1), y, type = "l")
```



Действително виждаме, че вече разпределението на Xn (което е представено с хистограмата) почти съвпада със стандартното нормално разпределение (което е представено с камбанообразната крива). По тази причина, често пъти на практика Биномното разпределение, центрирано и нормирано както в Xn се приближава със стандартното нормално разпределение.

Същите симулации можем да направим и с помощта на цикъла for като симулираме различните реализации на Xn една по една. Общата конструкция на цикъла for е

for(variable in vector) { command(s) }

Т.е.

for (i in множество, в което се изменя i)
{
действия
}

Например за да получим последната графика той би изглеждал така

```
> n = 1000; p = .25;
>Sn = mat.or.vec(500, 1); # задаваме си вида на Sn, в който ще запишем резултата
>Xn = mat.or.vec(500, 1); # задаваме си вида на Xn
>for (i in 1:500)
>{
> Sn[i] = rbinom(1, n, p)
> Xn[i] = (Sn[i] - n*p)/sqrt(n*p*(1-p))
>}
> hist(Xn, prob = T)
>  y = dnorm(seq(from = -3, to = 3, by = 0.1), mean = 0, sd = 1)
>  lines(seq(from = -3, to = 3, by = 0.1), y, type = "l")
```

Тук всяка една симулация на Sn се прави по отделно.

Достатъчно е да зададем типа на първото число от вектора. Цикълът ***for*** автоматично си уголемява вектора. Т.е. резултата от горния скрипт може да бъде получен и по следния начин.

```
> n = 1000; p = .25;
>Sn = numeric(0) ;          # задаваме си типа само на първото число от вектора Sn, в
                            # който ще запишем резултата
>Xn = numeric(0);           # задаваме си типа само на първото число от вектора Xn , в
                            # който ще запишем резултата
>for (i in 1:500)
>{
> Sn[i] = rbinom(1, n, p)
> Xn[i] = (Sn[i] - n*p)/sqrt(n*p*(1-p))
>}
> hist(Xn, prob = T)
>  y = dnorm(seq(from = -3, to = 3, by = 0.1), mean = 0, sd = 1)
>  lines(seq(from = -3, to = 3, by = 0.1), y, type = "l")
```

В цикъла ***for*** фигурните скоби { } не са задължителни ако имаме само едно действие. Т.е. при едно действие те се слагат по подразбиране. Например те не са нужни по-долу, където едно по едно отпечатваме следващите числа:

```
> p=c(2,3,5,7,11);
> for  (i in 1:5)   print(p[i])
[1] 2
[1] 3
[1] 5
[1] 7
[1] 11
```

Не е задължително числата в множеството, обхождано от индекса да са последователни например можем да постигнем същия резултат и с

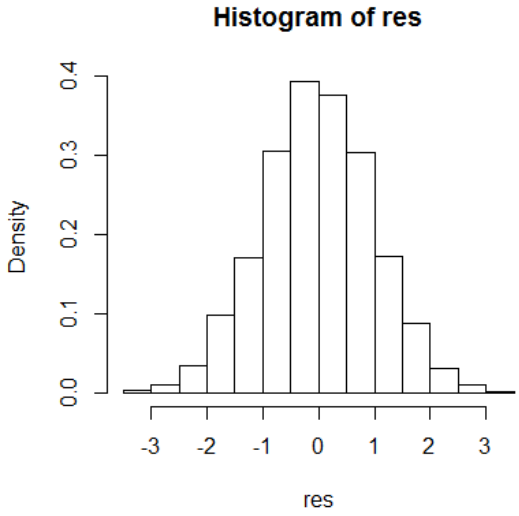
```
> for(i in p)    print(i)
```

Сега да илюстрираме действието на Централната гранична теорема в най-общия й вид, който написахме първо в тази тема, т.е. за произволна сума на независими еднакво разпределени случайни величини.

Първо ще разгледаме случая, когато наблюдаваната величина е точно нормално разпределена. Тогава вместо граница в Централната гранична теорема имаме равенство и по тази причина дори не е нужно n да е голямо, но за да получим добра оценка на наблюдаваната величина отново колкото са повече наблюденията, толкова по-добре.

Например. Симулирайте 5000 извадки, всеки път от по n = 10 наблюдения върху нормално разпределена случайна величина ξ с $\mu = 2$ и $\sigma = 5$. Тук е най-добре да използваме цикъл

```
> res = mat.or.vec(5000, 1);
> n = 10; mu = 2; sigma = 5
> for(i in 1:5000)
+ {
+ X = rnorm(n, mu, sigma)                    # генерира n реализации на ξ
+ res[i] = (mean(X) - mu)/(sigma/sqrt(n))
+ }
> hist(res, prob = T)
```



Всичко това може да бъде направено с **функции**.

Общият формат на функциите в R е

Име на функцията **= function(**входни параметри, които може и да липсват**)**
{
операции
return(име на *изходна променлива, която е обект*)
}

Когато до някои от входните параметри в първия ред на функцията има равенство и посочена стойност, това означава, че това е стойността, която се задава по подразбиране. Не е задължително да имаме ***return***, но е по-добре, т.к. тогава е ясно кой е резултатът от функцията. При това положение е добре в началото на тялото на функцията, изходната променлива да се инициализира и да се определи типа й.

Например същият резултат, както в предния скрипт ще постигнем с:

```
> f = function (m, n, mu = 0, sigma = 1)
+ {
+  res = numeric(0)
+        for(i in 1:m)
```

```
+        {
+        X = rnorm(n,mu,sigma) # генерира n реализации на ξ
+        res[i] = (mean(X) - mu)/(sigma/sqrt(n))
+       }
+ return(res)
+}
>r = f(5000, 10, 2, 5)
> hist(r, prob = T)
```

Така, повтаряйки само последните два реда и сменяйки параметрите можем да изчертаем много хистограми.

Същото може да бъде направено и с помощта на функцията ***simple.sim*** от библиотеката ***UsingR***. Тя е много удобна при повторения на едни и същи симулации. С нейна помощ можем да напишем функция за генериране на една симулация на дадена случайна величина и после да подадем тази функция като параметър на функцията ***simple.sim*** и също като параметър да определим какъв да е броят на симулациите. Това спестява писането на цикли. Нейният общ вид е

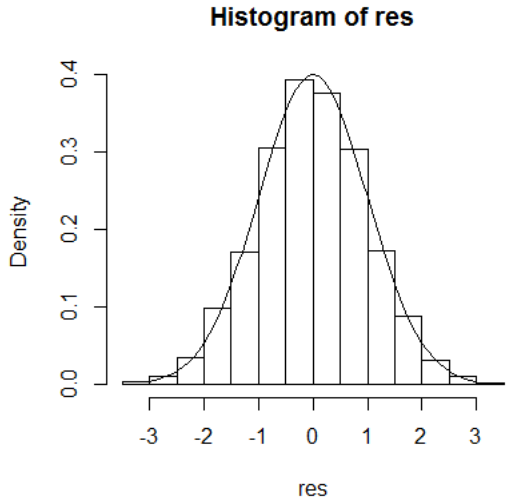
simple.sim(*брой извадки,*
име на функция която генерира една такава извадка,
*параметри на функцията ако има такива****)***

Т.е. ако искаме с помощта на функцията ***simple.sim*** да получим хистограмата по-горе (т.е. да повторим действията от предните два примера), това става например с:

```
> library(UsingR)
> f = function(n, mu, sigma)
+ {
+ X = rnorm(n, mu, sigma)
+ (mean(X) - mu)/(sigma/sqrt(n))
+ }
> r = simple.sim(5000, f, 10, 2, 5)
> hist(r, prob = T)
```

Сега да сравним тази хистограма със стандартната нормална крива

```
>  y = dnorm(seq(from = -3, to = 3, by = 0.1), mean = 0, sd = 1)
>  lines(seq(from = -3, to = 3, by = 0.1), y, type = "l")
```



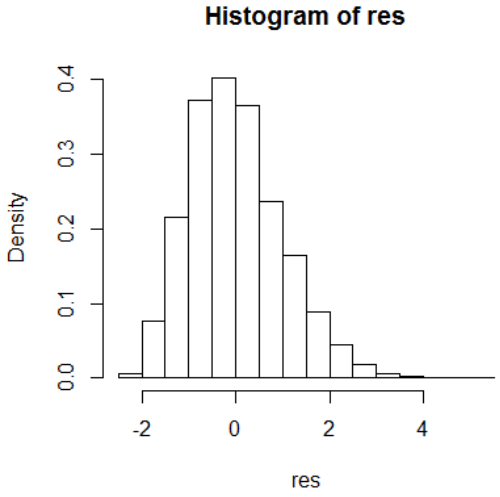
Виждаме, че за да е в сила резултата от ЦГТ, в случая когато наблюдаваната величина има точно нормално разпределение дори не е нужна да имаме голям брой събираеми n.

Когато наблюдаваната величина, обаче има произволно разпределение за да стегнем до гранично стандартно нормално разпределение понякога е нужен голям брой събираеми n.

Например: Нека наблюдаваме експоненциално разпределена случайна величина ξ с параметър λ = 0.1 и нека да видим дали сумата на 10 такива величини, центрирана и нормирана ще е стандартно нормално разпределена.

От предната тема и теорията за експоненциалното разпределение е известно, че в случая $\mu = E\xi = \frac{1}{\lambda} = 10$ и $\sigma^2 = E\xi = \frac{1}{\lambda^2} = 100$. Тогава в този случай

```
> res = mat.or.vec(5000, 1);
> lambda = 0.1; n = 10;
> mu = 1/lambda; sigma = 1/lambda
> for(i in 1:5000)
+ {
+ X = rexp(n, lambda)                    # генерира n реализации на ξ
+ res[i] = (mean(X) - mu)/(sigma/sqrt(n))
+ }
> hist(res, prob = T)
```

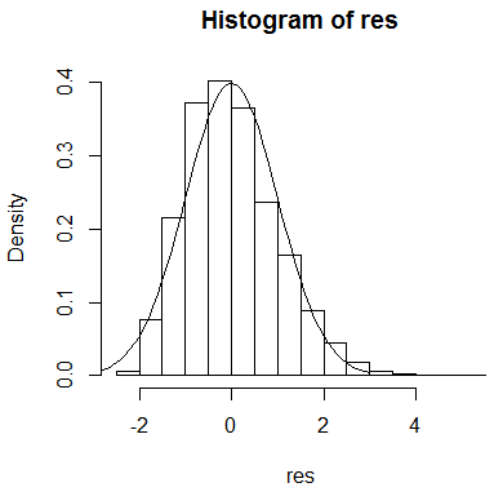


Същото можем да направим с помощта на функцията ***simple.sim*** например с:

```
> library(UsingR)
> f = function(n, lambda)
+ {
+ X = rexp(n, lambda)
+ mu =1/lambda
+ sigma=1/lambda
+ (mean(X) - mu)/(sigma/sqrt(n))
+ }
> r = simple.sim(5000, f, 10, 0.1)
> hist(r, prob = T)
```

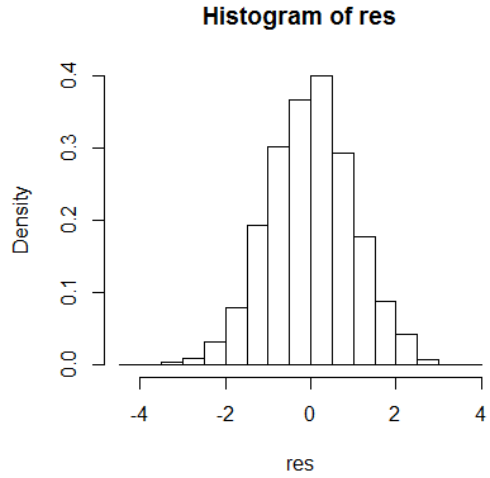
Сега да сравним тази хистограма със стандартната нормална крива

```
>  y = dnorm(seq(from = -3, to = 3, by = 0.1), mean = 0, sd = 1)
>  lines(seq(from = -3, to = 3, by = 0.1), y, type = "l")
```



Наблюдаваме разлика. Т.е. за произволно разпределение, ЦГТ не е в сила при произволни n. Нека сега да увеличим n = 100 000 и да повторим същите действия.

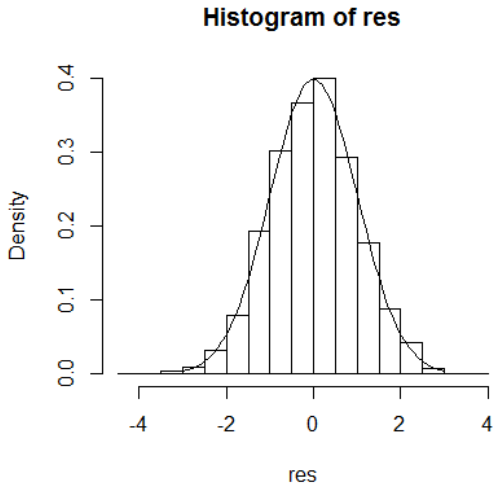
```
> res = mat.or.vec(5000, 1);
> lambda = 0.1; n = 100000;
> mu = 1/lambda; sigma = 1/lambda
> for(i in 1:5000)
+ {
+ X = rexp(n, lambda)                    # генерира n реализации на ξ
+ res[i] = (mean(X) - mu)/(sigma/sqrt(n))
+ }
> hist(res, prob = T)
```



Същото можем да направим с помощта на функцията ***simple.sim*** например с:

```
> library(UsingR)
> f = function(n, lambda)
+ {
+ X = rexp(n, lambda)
+ mu =1/lambda
+ sigma=1/lambda
+ (mean(X) - mu)/(sigma/sqrt(n))
+ }
> r = simple.sim(5000, f, 100000, 0.1)
```

> hist(r, prob = T)
Сега да сравним тази хистограма със стандартната нормална крива
> y = dnorm(seq(from = -3, to = 3, by = 0.1), mean = 0, sd = 1)
> lines(seq(from = -3, to = 3, by = 0.1), y, type = "l")



Наблюдаваме по-голяма прилика със стандартната нормална крива. Това се дължи на действието на ЦГТ.

Още графични методи за сравняване на разпределения pp-plot и qq-plot

До тук сравнявахме разпределенията само чрез графиката на хистограмата и плътността. Има още много графични начини, с които това може да бъде направено. Ние ще разгледаме сега по-подробно техниката на pp-plot и qq-plot. И двете са сравнително точни, само при големи извадки. Съкращението pp идва от факта, че по двете оси на тази графика се нанасят вероятности, т.е. от probability probability графика. Съкращението qq идва от факта, че по двете оси на тази графика се нанасят квантили, т.е. от quantile quantile графика.
И двата типа такива графики са по-добри, когато стане дума за сравняване на разпределения.
В частен случай, тези графики служат за сравняване с нормалното разпределение. В този случай те се наричат съответно normal probability plot и normal quantile plot.

По-нататък се използва ***теоремата на Гливенко-Кантели***, която казва, че ако увеличим обемът на извадката, т.е. n емпиричната функция на разпределението на извадката

$$F_n(x) = \frac{\text{брой на наблюденията със стойност по-малка от x}}{\text{броят на всички наблюдения,т.е.}n}$$

може да доближи теоретичната функция на разпределение на наблюдаваната величина, т.е. F(x) = P(ξ < x) с произволна точност.

Преди да започнем с обяснението да припомним, че емпиричният p квантил е онова число, за което можем да кажем, че p100% от наблюденията са по-малки или равни на него и (1-p)100% от наблюденията са по-големи или равни на него. При това вече знаем, че в много случаи тези квантили не са определени еднозначно и се додефинират.
Нека X(n,1) ≤ X(n,2) ≤ … ≤ X(n,n) са подредените по големина наблюдения върху случайната величина ξ. Та ако при големи извадки, заместим в емпиричната функция на разпределение на наблюдаваната величина, променливата x с X(n,1) ≤ X(n,2) ≤ … ≤ X(n,n) бихме получили приблизително 0, 1/n, 2/n, …, (n-1)/n. Т.е.

$$F_n\left(X_{(n,i)}\right) = \frac{i-1}{n},\ i = 1, 2, \ldots, n.$$

И обратно. Ако пресметнем емпиричните квантили на разпределението в точките 0, 1/n, 2/n, …, (n-1)/n, т.е. ще получим съответно X(n,1) ≤ X(n,2) ≤ … ≤ X(n,n). Т.е.

$$q_n\left(\frac{i-1}{n}\right) = F_n^{\leftarrow}\left(\frac{i-1}{n}\right) = X_{(n,i)},\ i = 1,\ 2,\ \ldots,\ n.$$

На това разсъждение са основани следните две графики.

qq-plot

Основната идея е да се изчертаят точки с координати $(X(n,i), q((i-1)/n))$, i = 1, 2, …, n, където с q(p) сме означили p квантила на тестваното разпределение F(x). Ако тестваното разпределение съвпада с действителното разпределение на наблюдаваната величина, то тези точки ще съвпаднат с ъглополовящата на първи квадрант. Т.е. ако точките от qq-plot съвпаднат с ъглополовящата на първи квадрант, то тестваното и действителното разпределение съвпадат.

В R функциите ***qqnorm*** (или по-общо ***qqplot.das*** от библиотеката ***StatDA***) служат съответно за изчертаване на нормална qq-plot, т.е. за тестване на нормално разпределение (или по-общо ***qqplot.das*** за произволно разпределение)

qqnorm(вектор който тестваме за нормалност, ylim, main = "Normal Q-Q Plot", xlab = "Theoretical Quantiles", ylab = "Sample Quantiles", ...)

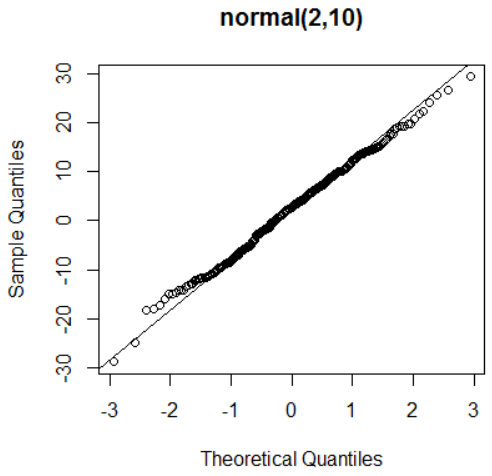
qqplot.das(вектор който тестваме дали има съответното разпределение, "име на разпределнието което тестваме",...)

Когато се говори за тежки десни (леви) опашки се има в предвид разпределения, които имат по-големи вероятности да се случат безкрайно големи(малки) стойности в сравнение със съответното нормално разпределение (в случая, когато се сравняват центрираните и нормирани разпределения).

Пример. Симулирайте 300 наблюдения върху нормално разпределена случайна величина с параметри μ = 2 и σ = 10. После предположете, че не знаете разпределението на наблюдаваната величина и изчертайте нормална qq-plot, за да направите тест за нормалност. Направете няколко подобни опита. Какво наблюдавате?

Отг. Точките лежат почти върху ъглополовящата на първи квадрант, следователно наблюдаваното разпределение е нормално.

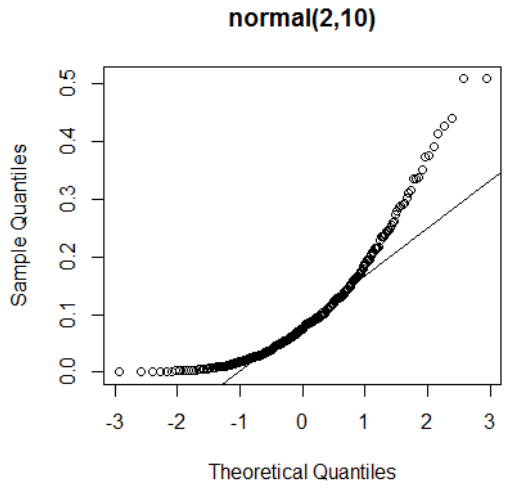
```
> x = rnorm(300, 2, 10);  qqnorm(x, main = 'normal(2, 10)');  qqline(x)
```



Пример. Симулирайте 300 наблюдения върху експоненциално разпределена случайна величина с параметър λ = 10. После предположете, че не знаете разпределението на наблюдаваната величина и изчертайте нормална qq-plot за да направите тест за нормалност. Направете няколко подобни опита. Какво наблюдавате?

Отг. Точките не лежат върху ъглополовящата на първи квадрант следователно наблюдаваното разпределение не е нормално.

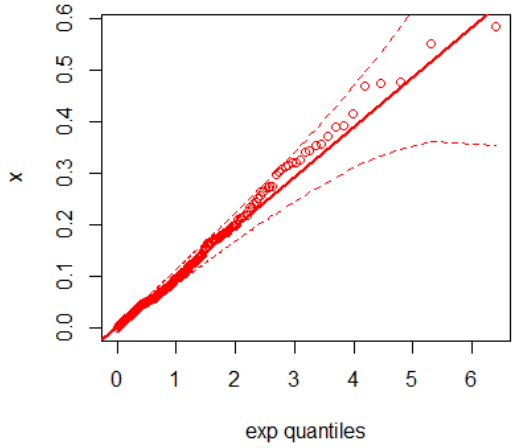
```
> x = rexp(300, 10);  qqnorm(x,  main = 'normal(2, 10)');  qqline(x)
```



Пример: Симулирайте 300 наблюдения върху експоненциално разпределена случайна величина с параметър $\lambda = 10$. После предположете, че не знаете разпределението на наблюдаваната величина и изчертайте експоненциално qq-plot за да направите тест за експоненциалност. Направете няколко подобни опита. Какво наблюдавате?

Отг. Пунктираните линии очертават доверителните интервали, в които хипотезата за експоненциалност се приема. Т.е. след такава графика можем да заключим, че наблюдаваното разпределение е експоненциално.

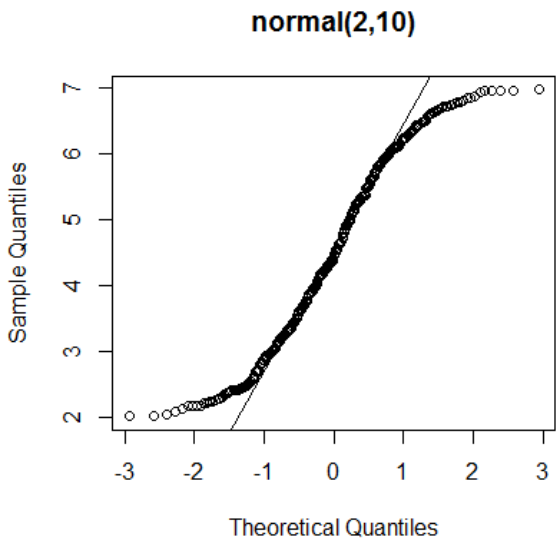
```
> x = rexp(300, 10)
> library(StatDA)
> qqplot.das(x, "exp")
```



Пример: Симулирайте 300 наблюдения върху равномерно разпределена случайна величина с върху интервала [2, 7]. После предположете че не знаете разпределението на наблюдаваната величина и изчертайте нормална qq-plot за да направите тест за нормалност. Направете няколко подобни опита. Какво наблюдавате?

Отг. Точките не лежат върху ъглополовящата на първи квадрант, следователно наблюдаваното разпределение не е нормално.

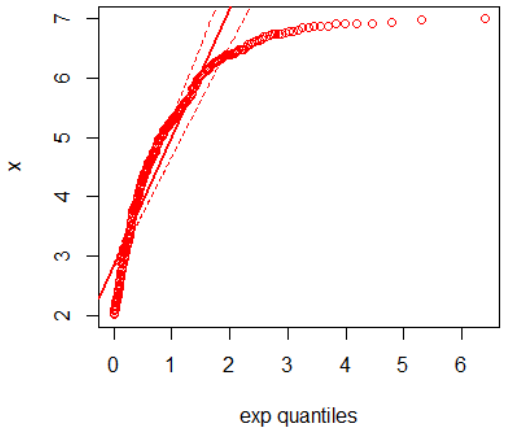
```
> x = runif(300, min = 2, max = 7);  qqnorm(x, main = 'normal(2, 10)');  qqline(x)
```



Пример: Симулирайте 300 наблюдения върху равномерно разпределена случайна величина с върху интервала [2, 7]. После предположете че не знаете разпределението на наблюдаваната величина и изчертайте експопенциална qq-plot за да направите тест за експоненциалност. Направете няколко подобни опита. Какво наблюдавате?

Отг. Точките не лежат върху ъглополовящата на първи квадрант следователно наблюдаваното разпределение не е експоненциално.

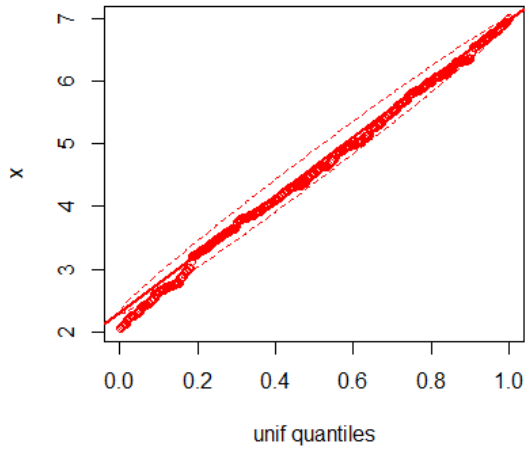
```
> x = runif(300, min = 2, max = 7)
> library(StatDA)
> qqplot.das(x, "exp")
```



Пример: Симулирайте 300 наблюдения върху равномерно разпределена случайна величина с върху интервала [2, 7]. После предположете че не знаете разпределението на наблюдаваната величина и изчертайте равномерна qq-plot за да направите тест за равномерност. Направете няколко подобни опита. Какво наблюдавате?

Отг. Точките лежат върху ъглополовящата на първи квадрант следователно наблюдаваното разпределение е равномерно.

```
> x = runif(300, min = 2, max = 7)
> library(StatDA)
> qqplot.das(x, "unif")
```



По аналогичен начин се работи с pp-plot.
Този път, основната идея е да се изчертаят точки с координати

$$\left(\frac{i-1}{n}, F(\mathrm{X}(\mathrm{n}, \mathrm{i}))\right) \text{ за i = 0, 1, …, n,}$$

където F е тестваната функция на разпределение. В случая и по двете оси имаме оценки на вероятности. Ако сме улучили точно разпределението на наблюдаваната величина, то тези точки биха лежали върху ъглополовящата на първи квадрант. Т.е. ако точките от pp-plot съвпаднат с ъглополовящата на първи квадрант, то тестваното и действителното разпределение съвпадат.

Забележка. Когато тестваме това дали наблюдаваната величина има нормално разпределение съответните pp-plot и qq-plot се наричат още съответно normal pp-plot или normal probability plot и normal qq-plot, т.е. normal quantile plot.