Homework BI-PST.

Andrey Babushkin

Important note. The code here is not complete, it only shows the most important parts. A complete script is attached with this document.

Prerequisites

Let's calculate K, L and M and choose an appropriate dataset.

K = 28 (28.05.1996)
L = 9 (Babushkin)
M = 47 * 28 * 9 (mod 12) = 47 * 7 * 3 * 12 (mod 12) = 0

Dataset: **case0101**

Tasks

1) First, we read the dataset, split it and convert Extrinsic and Intrinsic datasets into vectors.

```
ds = case0101
ds.split = split(ds, ds$Treatment)
ds.intrinsic = ds.split$Intrinsic$Score
ds.extrinsic = ds.split$Extrinsic$Score
```

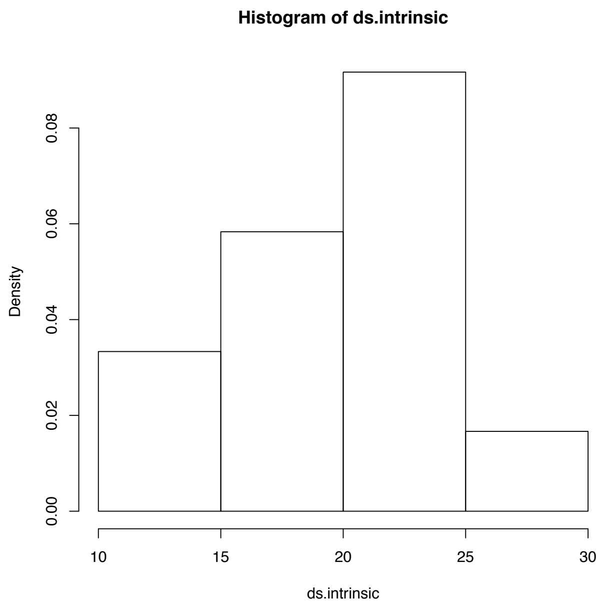
Now we print the summary out about both vectors.

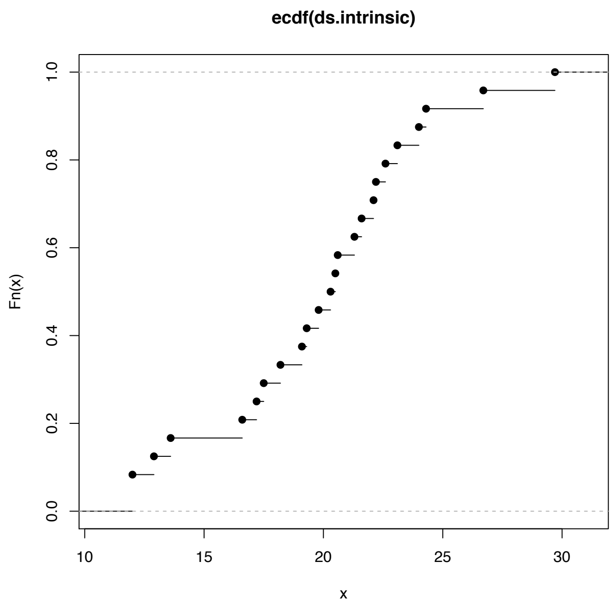
```
> summary(ds.intrinsic)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  12.00   17.43   20.40   19.88   22.30   29.70 
> var(ds.intrinsic)
[1] 19.70928

> summary(ds.extrinsic)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   5.00   12.15   17.20   15.74   18.95   24.00 
> var(ds.extrinsic)
[1] 27.58976
```

So, we have found min, max, mean, median, and variance of both vectors.

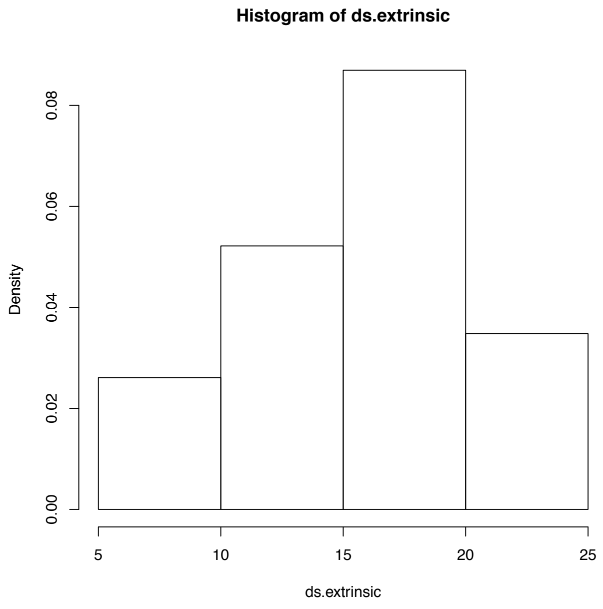
2) Now let have a look at densities and empirical distribution functions.

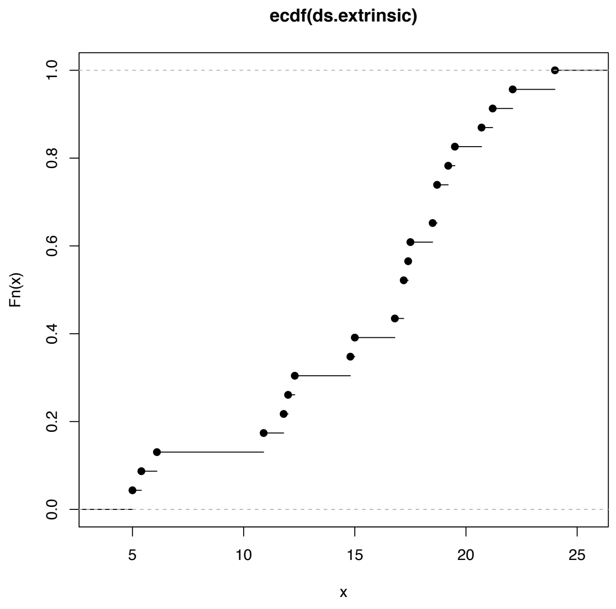
```
> hist(ds.intrinsic, freq=FALSE, breaks = 6)
> plot(ecdf(ds.intrinsic))
```





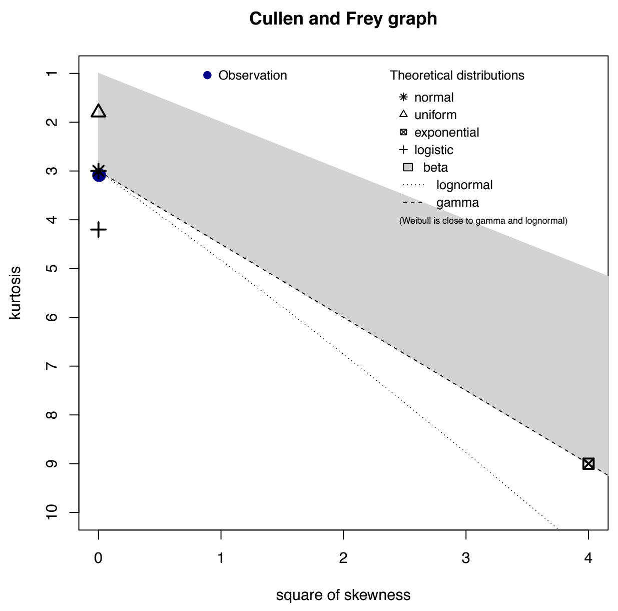
```
> hist(ds.extrinsic, freq=FALSE, breaks = 6)
> plot(ecdf(ds.extrinsic))
```

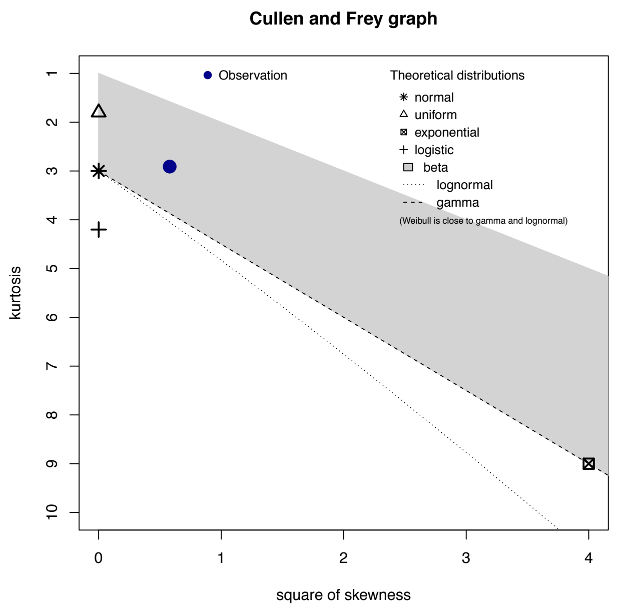




3) We can see that the distributions looks much like a Gaussian. But let's check our hypothesis using the descdist function from the fitdistrplus package.

```
> descdist(ds.intrinsic)
> descdist(ds.extrinsic)
```



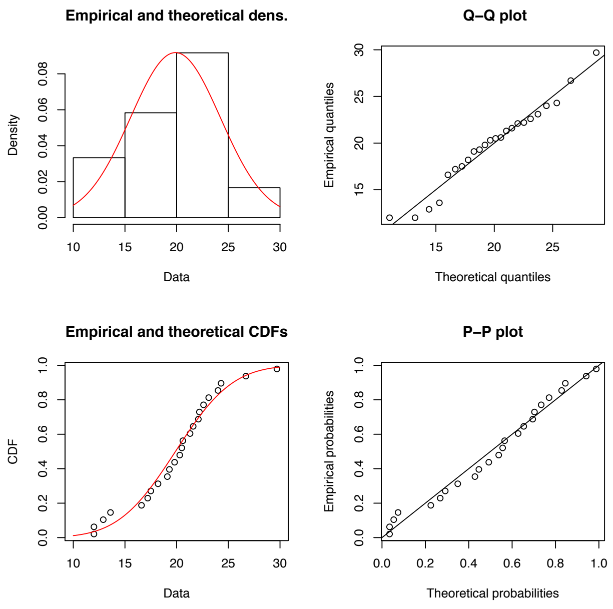


From these plots we can clearly see that the kurtosis and skewness of our distributions are almost the same as for the Gaussian distribution. Nevertheless, let's fit 3 known distributions (normal, exponential and uniform) and see if they approximate our datasets well.

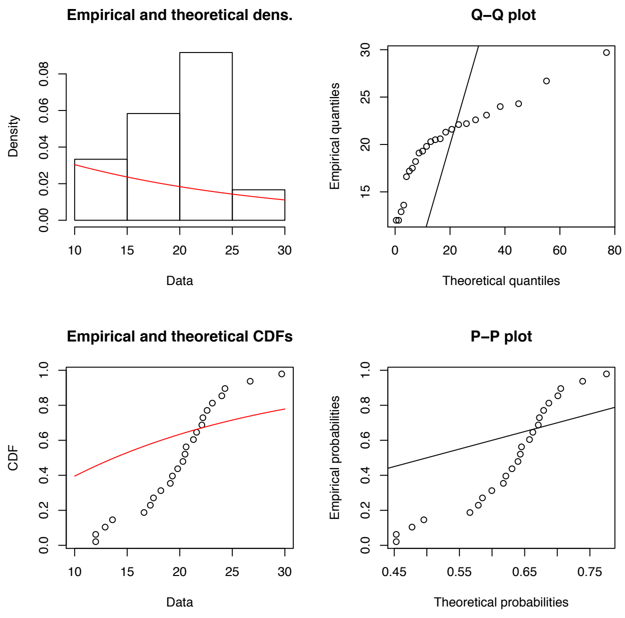
```
> fit.intrinsic.norm = fitdist(ds.intrinsic, "norm")
> fit.intrinsic.exp = fitdist(ds.intrinsic, "exp")
> fit.intrinsic.unif = fitdist(ds.intrinsic, "unif")

> plot(fit.intrinsic.norm)
> plot(fit.intrinsic.exp)
> plot(fit.intrinsic.unif)

> mu.intrinsic = fit.intrinsic.norm$estimate['mean']
> sigma.intrinsic = fit.intrinsic.norm$estimate['sd']

> x.intrinsic.min = min(ds.intrinsic)
> x.intrinsic.max = max(ds.intrinsic)
> x.intrinsic = seq(x.intrinsic.min, x.intrinsic.max, length =
1000)

> y.intrinsic.norm = dnorm(x.intrinsic,
                           mean = mu.intrinsic,
                           sd = sigma.intrinsic)
> y.intrinsic.exp = dexp(x.intrinsic,
                         rate=fit.intrinsic.exp$estimate['rate'])
```

```
> y.intrinsic.unif = dunif(x.intrinsic,
                   min=fit.intrinsic.unif$estimate['min'],
                   max = fit.intrinsic.unif$estimate['max'])

> hist(ds.intrinsic, breaks = 6, freq = FALSE)
> lines(x = x.intrinsic, y = y.intrinsic.norm, col = 'red')
> lines(x = x.intrinsic, y = y.intrinsic.exp, col = 'blue')
> lines(x = x.intrinsic, y = y.intrinsic.unif, col = 'green')
```

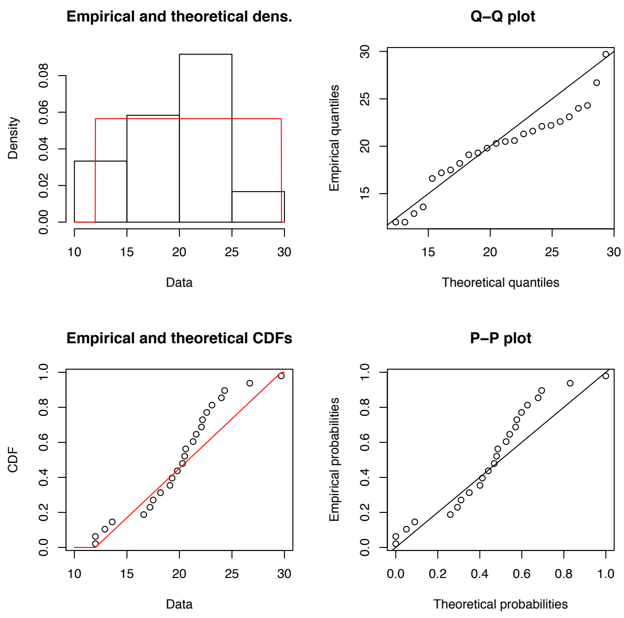
Normal Intrinsic



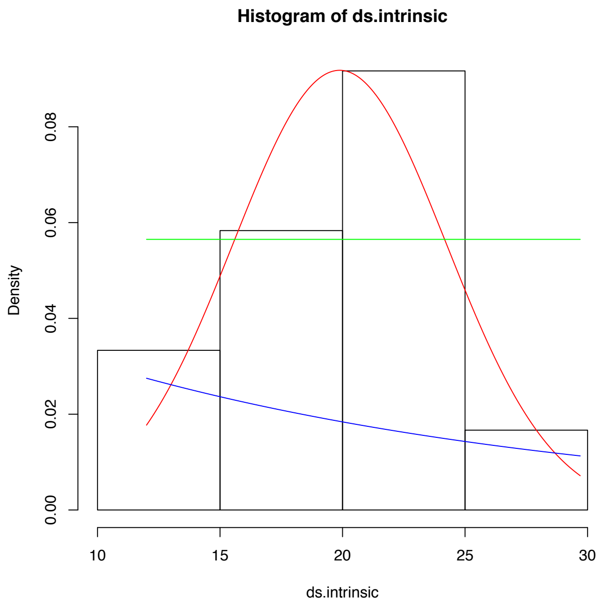
Exponential Intrinsic



Uniform Intrinsic

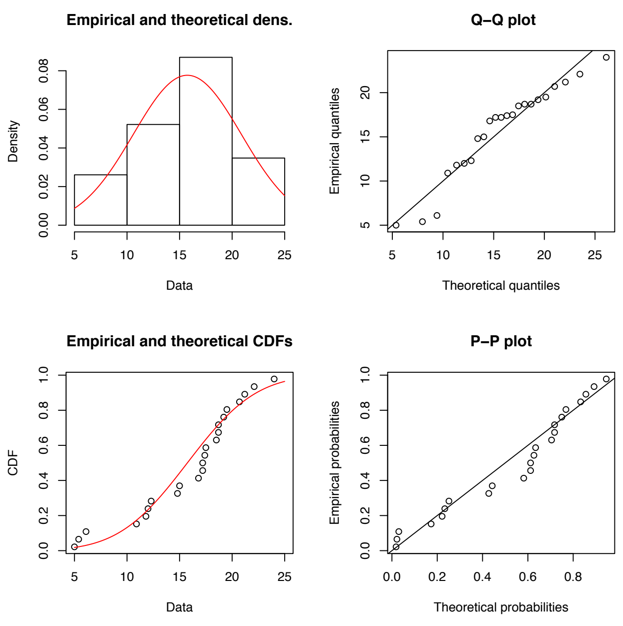


All fitted distributions

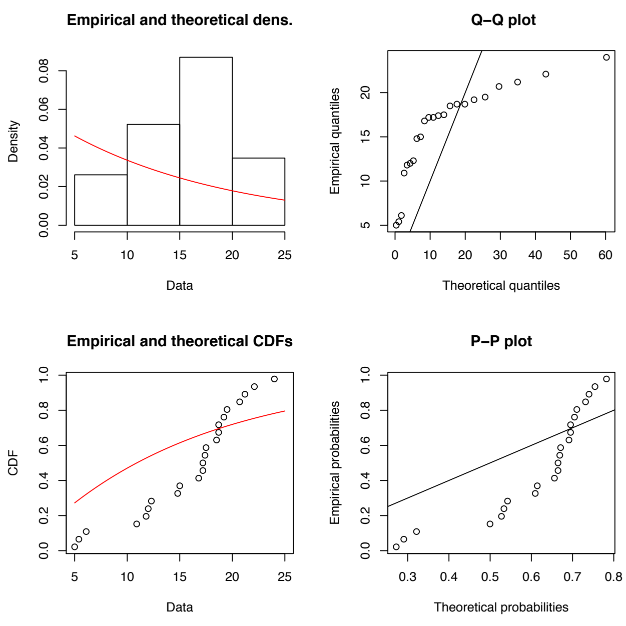


We will do absolutely the same for the Extrinsic dataset with the identical code.

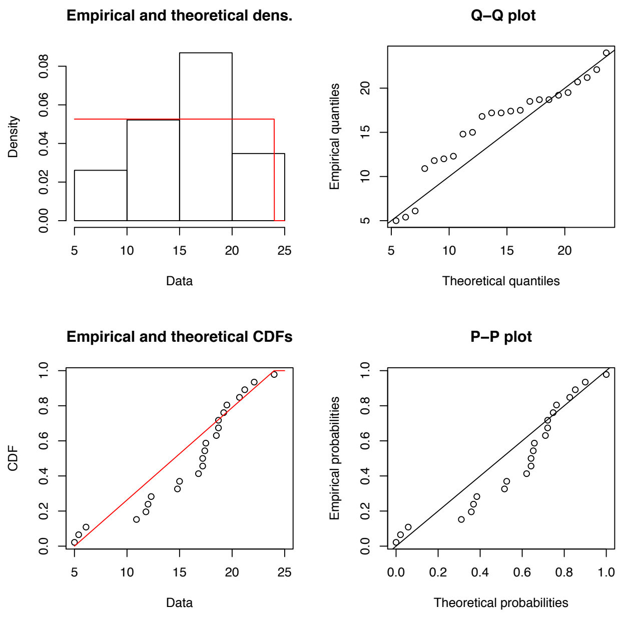
Normal Extrinsic



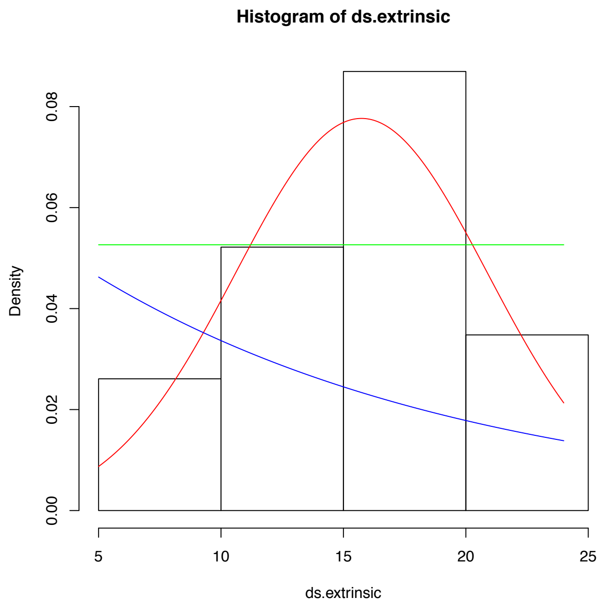
Exponential Extrinsic



Uniform Extrinsic



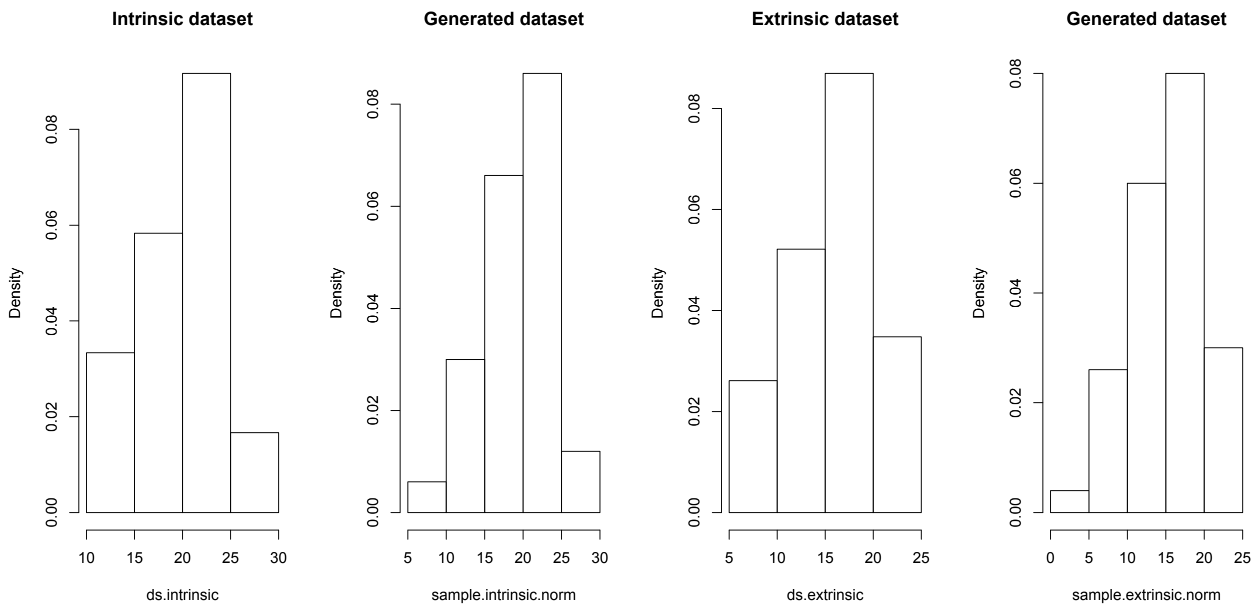
All fitted distributions



From these plot one can clearly see that for both our datasets the best approximation will be a Gaussian distribution.

4) Now we can generate 100 random deviates from the fitted normal distribution and compare histograms of a generated dataset and the actual.

```
> sample.intrinsic.norm = rnorm(100,
                                mean = mu.intrinsic,
                                sd = sigma.intrinsic)

> par(mfrow = c(1, 2))
> hist(ds.intrinsic, breaks = 6,
                     freq = FALSE,
                     main = "Intrinsic dataset")
> hist(sample.intrinsic.norm, breaks = 6,
                              freq = FALSE,
                              main = "Generated dataset")

> sample.extrinsic.norm = rnorm(100,
                                mean = mu.extrinsic,
                                sd = sigma.extrinsic)

> par(mfrow = c(1, 2))
> hist(ds.extrinsic, breaks = 6,
                     freq = FALSE,
                     main = "extrinsic dataset")
> hist(sample.extrinsic.norm, breaks = 6,
                              freq = FALSE,
                              main = "Generated dataset")
```



So, from these histograms it is seen that the fitted normal distributions approximate our samples quiet well.

5) To calculate confidence intervals for both our distributions we'll use a built-in function in R. Actually, we'll use a side effect of this function - calculating a confidence interval.

```
> siglevel = 0.05

> (conf.int.intrinsic = t.test(ds.intrinsic,
                               conf.level = 1-siglevel)$conf.int)
[1] 18.00869 21.75798
attr(,"conf.level")
[1] 0.95

> (conf.int.extrinsic = t.test(ds.extrinsic,
                               conf.level = 1-siglevel)$conf.int)
[1] 13.46774 18.01052
attr(,"conf.level")
[1] 0.95
```

So, our 95% confidence interval for the **Intrinsic** dataset is
 (18.00869, 21.75798),

for the **Extrinsic** dataset is
 (13.46774, 18.01052).

6) Let's check if the mean value of the datasets is equals to the value of K (K = 28). We'll use the 5% significance level interval.

```
> t.test(ds.intrinsic, mu=28)

     One Sample t-test

data:  ds.intrinsic
t = -8.9567, df = 23, p-value = 5.857e-09
alternative hypothesis: true mean is not equal to 28
95 percent confidence interval:
 18.00869 21.75798
sample estimates:
mean of x 
 19.88333 

> t.test(ds.extrinsic, mu=28)

     One Sample t-test

data:  ds.extrinsic
t = -11.195, df = 22, p-value = 1.491e-10
alternative hypothesis: true mean is not equal to 28
95 percent confidence interval:
 13.46774 18.01052
sample estimates:
mean of x 
 15.73913 
```

From the output we see that the p-value for the Intrinsic dataset is 5.857e-09, for the Extrinsic dataset is 1.491e-10, which is less than 0.05. That means that we can reject the null-hypothesis, that is, the mean value of the extrinsic data set is not equal to the value of K, K = 28.

7) The last part of this work will be comparing two datasets or more exactly we will check if the mean values of both dataset are equal on the 95% confidence interval. That will be our null hypothesis (or H_0). An alternative hypothesis will be that mean values are not equal.

First of all, before using the two sample t-test, we should evaluate a Fisher's F-test and verify a homogeneity of variances:

```
> var.test(ds.intrinsic, ds.extrinsic)

     F test to compare two variances

data:  ds.intrinsic and ds.extrinsic
F = 0.71437, num df = 23, denom df = 22, p-value = 0.4289
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.3047427 1.6612045
sample estimates:
ratio of variances
         0.7143691
```

We see that the p-value of 0.4289 is greater than 0.05, thus, variances are homogeneous.

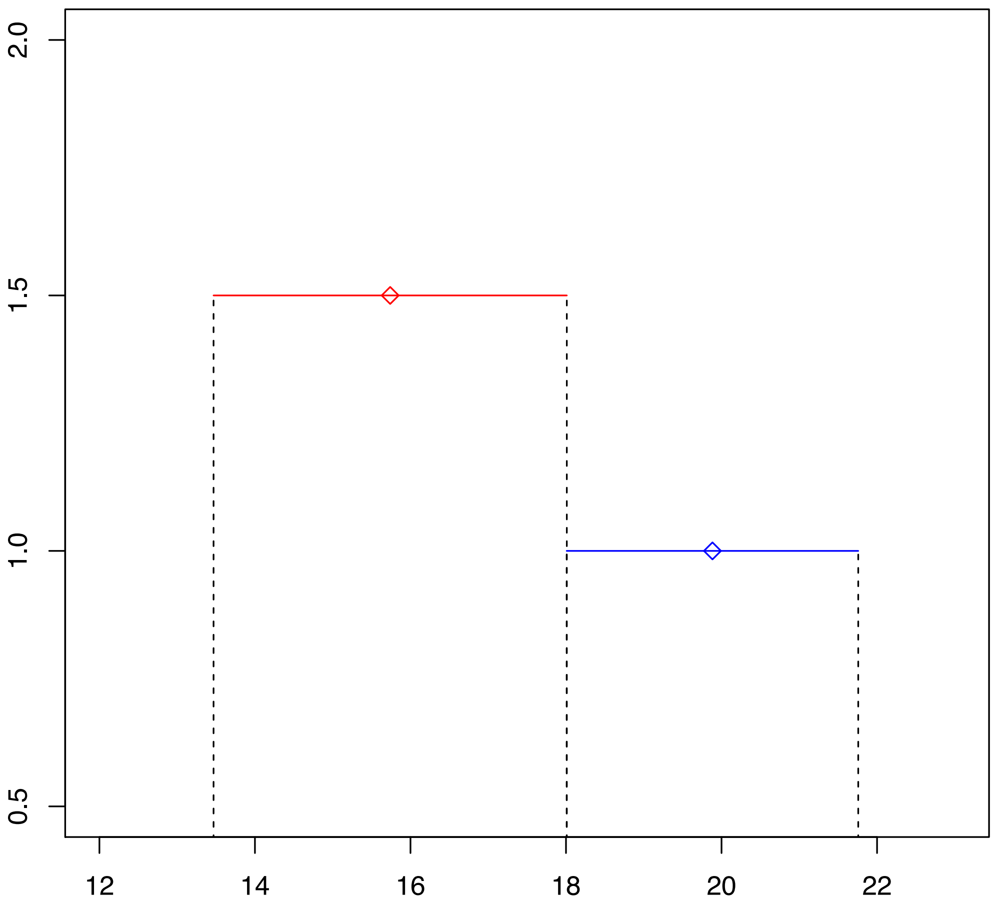
So, the variances are homogeneous, our datasets follow the Gaussian distribution, therefore, finally, we can perform a t-test.

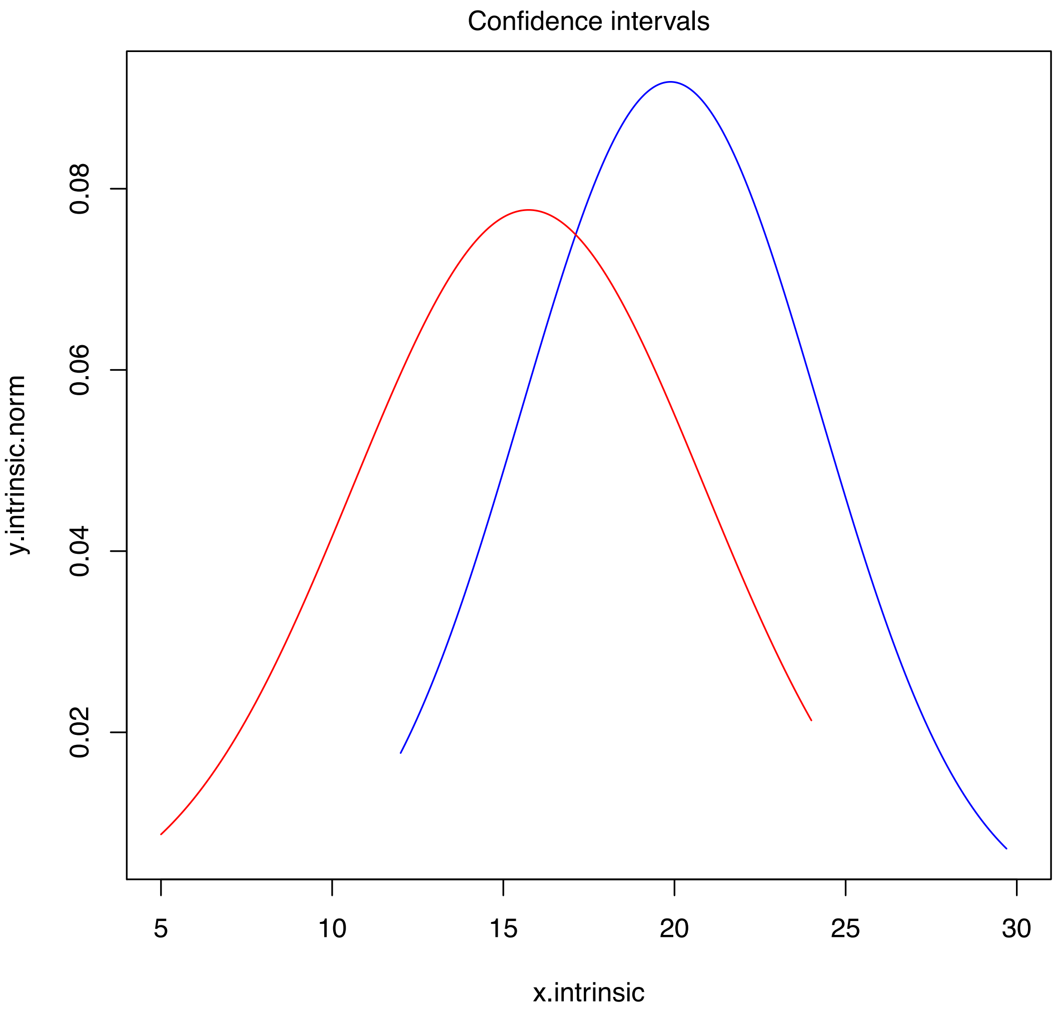
```
> t.test(ds.intrinsic, ds.extrinsic, var.equal=TRUE)

     Two Sample t-test

data:  ds.intrinsic and ds.extrinsic
t = 2.9259, df = 45, p-value = 0.005366
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 1.291432 6.996973
sample estimates:
mean of x mean of y
 19.88333  15.73913
```

We have the p-value of 0.005366, which is less then 0.05. So, the H_0 hypothesis can be rejected, mean values of our distributions are different.

BONUS) Let's now draw, just for fun, our 95% confidence intervals.

```
> plot(c(conf.int.intrinsic[1],
         conf.int.intrinsic[2]),
       c(1, 1),
       type = 'l',
       lty = 'solid',
       xlab = 'Confidence intervals',
       ylab = '',
       xlim=c(12, 23),
       ylim=c(0.5, 2),
       col = 'blue')

> points(x = mu.intrinsic, y = 1, pch = 5, col = 'blue')
> segments(x0 = conf.int.intrinsic[1],
           x1 = conf.int.intrinsic[1],
           y0 = 0,
           y1 = 1,
           lty = 'dashed')
> segments(x0 = conf.int.intrinsic[2],
           x1 = conf.int.intrinsic[2],
           y0 = 0,
           y1 = 1,
           lty = 'dashed')
> segments(x0 = conf.int.extrinsic[1],
           y0 = 1.5,
           x1 = conf.int.extrinsic[2],
           y1 = 1.5,
           col = 'red')
> points(x = mu.extrinsic, y = 1.5, pch = 5, col = 'red')
> segments(x0 = conf.int.extrinsic[1],
           x1 = conf.int.extrinsic[1],
           y0 = 0,
           y1 = 1.5,
           lty = 'dashed')
> segments(x0 = conf.int.extrinsic[2],
           x1 = conf.int.extrinsic[2],
           y0 = 0,
           y1 = 1.5,
           lty = 'dashed')

# At the end we'll print two our fitted distributions on one plot.

> plot(x = x.intrinsic,
       y = y.intrinsic.norm,
       col = 'blue', type = 'l', lty = 'solid', xlim=c(5, 30))
> lines(x = x.extrinsic, y = y.extrinsic.norm, col = 'red')
```


2.0
1.5
1.0
0.5
12
14
16
18
20
22


Confidence intervals
y.intrinsic.norm
0.08
0.06
0.04
0.02
5
10
15
20
25
30
x.intrinsic