# Solutions

```
# Plotting
library(ggplot2) # visualisation
library(ggpubr) # fancy plotting, qqplot
```

```
## Loading required package: magrittr
```

```
library(cowplot) # plot grid
```

```
##
## ************************************************************
## Note: As of version 1.0.0, cowplot does not change the
##   default ggplot2 theme anymore. To recover the previous
##   behavior, execute:
##   theme_set(theme_cowplot())
## ************************************************************
##
## Attaching package: 'cowplot'
## The following object is masked from 'package:ggpubr':
##
##     get_legend
```

```
library(corrplot) # correlation plot
```

```
## corrplot 0.84 loaded
```

```
# Data manipulation
library(dplyr) # data frame manupulation
```

```
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr) # data frame manupulation
```

```
##
## Attaching package: 'tidyr'
## The following object is masked from 'package:magrittr':
##
##     extract
```

```
# Stats
library(Rmisc) # confidence interval
```

```
## Loading required package: lattice
```

```
## Loading required package: plyr
```

```
## ------------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## ------------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
## The following object is masked from 'package:ggpubr':
##
##     mutate
```

```
library(car) # Levene's test
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

# Part 1

1. Generate 100 observations from normal distribution with mean=5, sd=2.

```
sample <- rnorm(n = 100, mean=5, sd=2)
sample
```

```
##   [1] 6.1074123 5.6234703 3.8333731 5.9806748 7.0777543 6.3160490 6.2705000
##   [8] 2.6253844 5.9638472 7.8604739 4.8281559 6.0103646 3.5262507 4.5379304
##  [15] 6.4955469 6.6405225 3.5894763 6.3845810 4.7033055 0.3534076 4.0989776
##  [22] 2.2796185 4.1389767 1.0768108 1.9950216 2.4325862 9.9421001 7.4638048
##  [29] 7.1538860 7.7582782 4.0015100 4.9895885 4.0216793 5.5180563 7.8608688
##  [36] 4.9132774 3.8279053 3.7414554 7.0478317 7.6074955 6.7188995 7.2661220
##  [43] 5.1050481 4.8607844 4.8622638 5.5636720 3.2492385 6.6989687 5.1481994
##  [50] 4.9308300 5.9524574 4.0237570 3.3817621 2.6503298 6.9404279 8.7511005
##  [57] 3.1352526 1.5451747 7.7358023 5.3418626 4.4089418 3.7086788 5.9631020
##  [64] 5.8970918 1.9758198 5.7439541 8.9209654 5.9990292 3.7009866 5.0614989
##  [71] 8.4628607 7.9035009 3.0554094 6.4291056 7.8014786 1.4229818 4.7424551
##  [78] 8.3459092 3.9764183 4.1753290 2.8923689 7.1473029 7.9460138 4.6852818
##  [85] 5.2043464 3.4552516 6.2494531 4.1225589 4.3558694 5.0602106 1.0792614
##  [92] 3.9484086 7.6797460 3.4860360 7.4779707 2.4301167 5.1159681 5.9570598
##  [99] 3.4521608 7.2405549
```

2. Use this vector to calculate mean, median, sd, variance.

```r
mean(sample)
```

```
## [1] 5.171416
```

```r
# [1] 5.057534

median(sample)
```

```
## [1] 5.083273
```

```r
# [1] 5.046119

sd(sample)
```

```
## [1] 1.987743
```

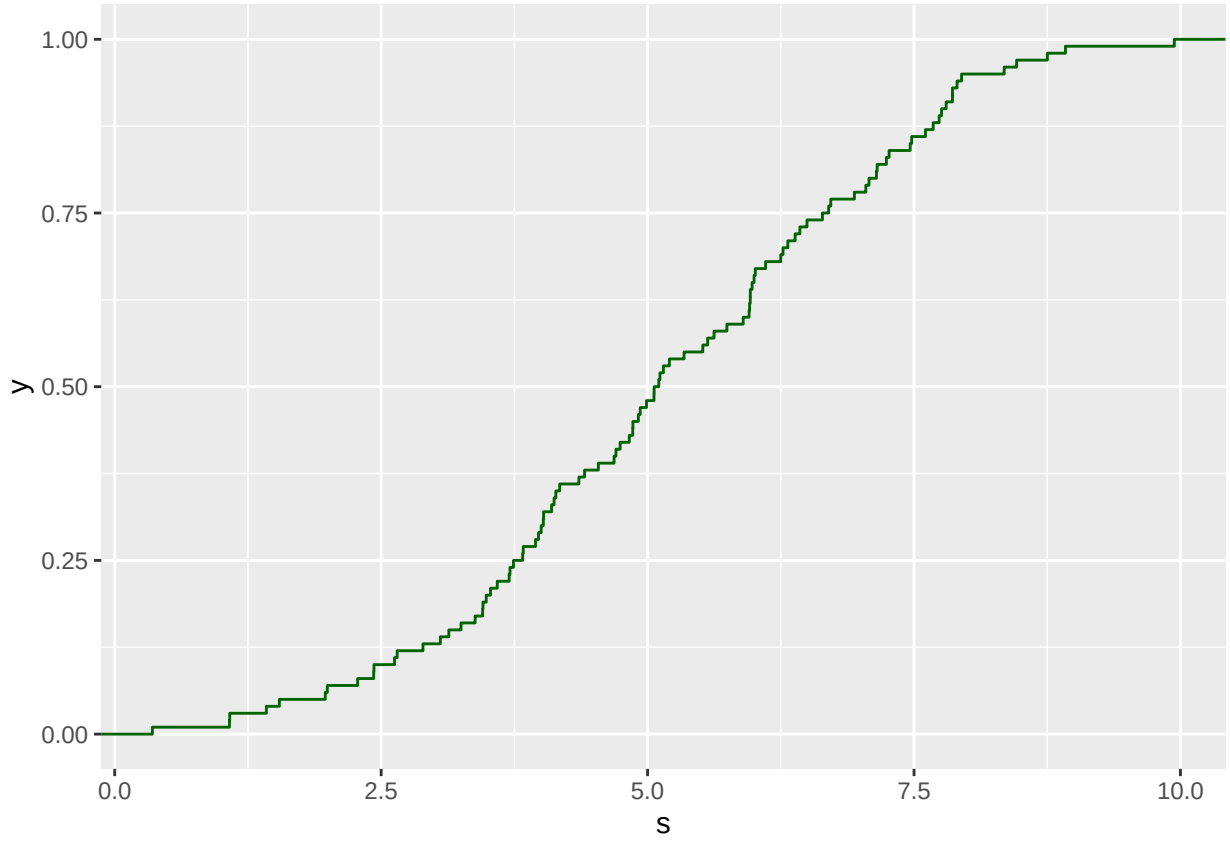
```r
# [1] 1.979544

var(sample)
```

```
## [1] 3.951122
```

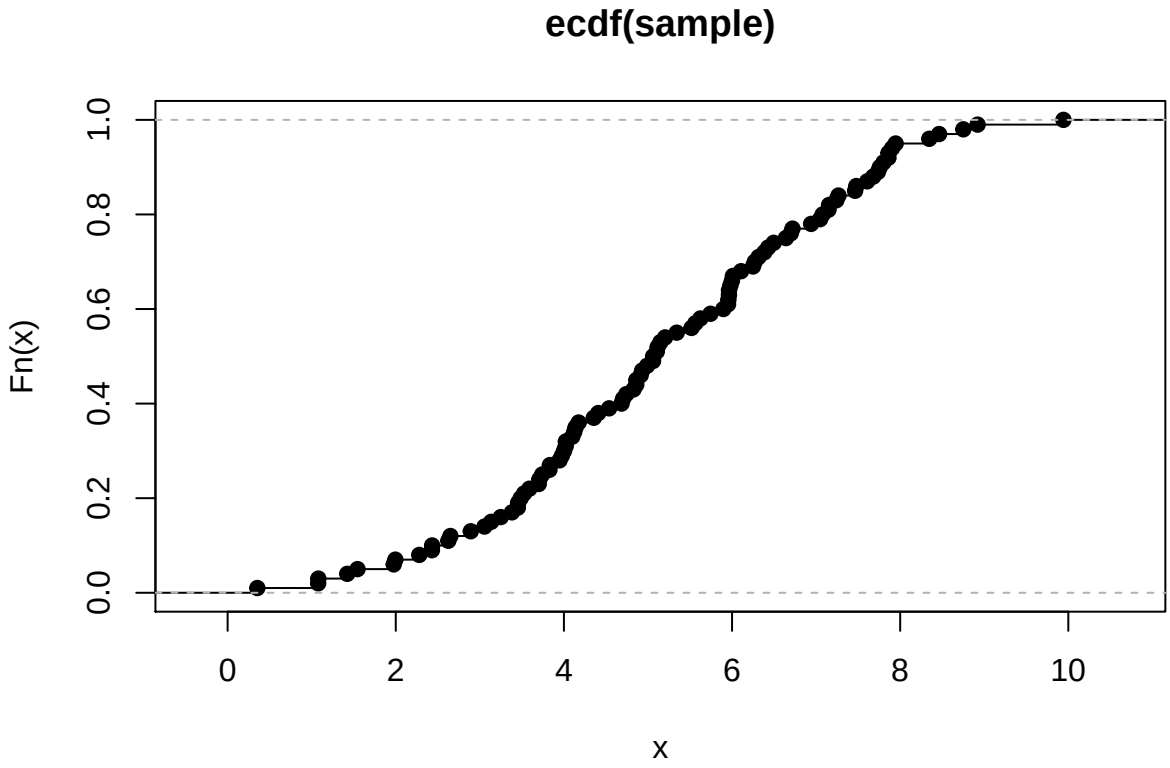
```r
# [1] 3.918594
```

3. Use this vector to plot Cumulative Distribution Function. You can use either basic R or ggplot.

```r
ggplot(data=data.frame(s=sample), aes(x=s)) +
  stat_ecdf(color = 'darkgreen')
```

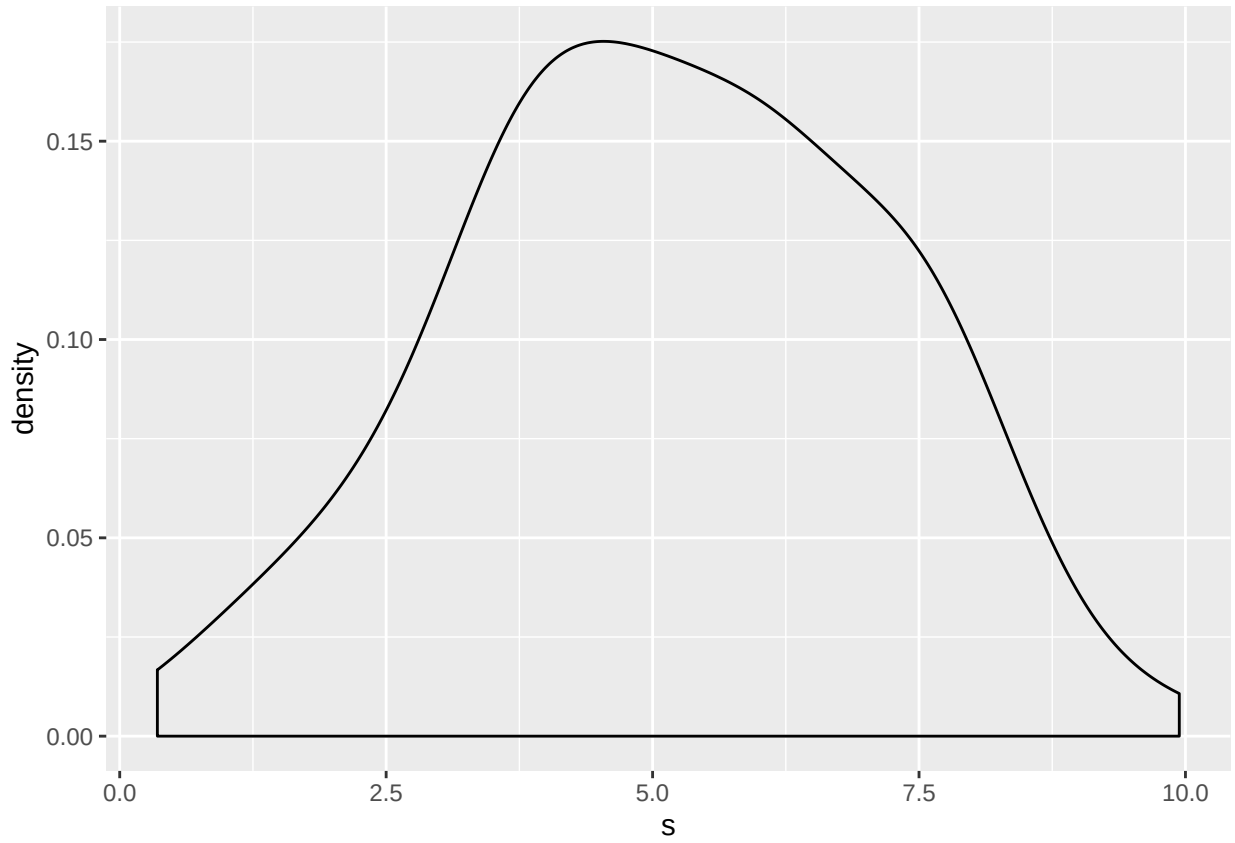


```
plot(ecdf(sample))
```

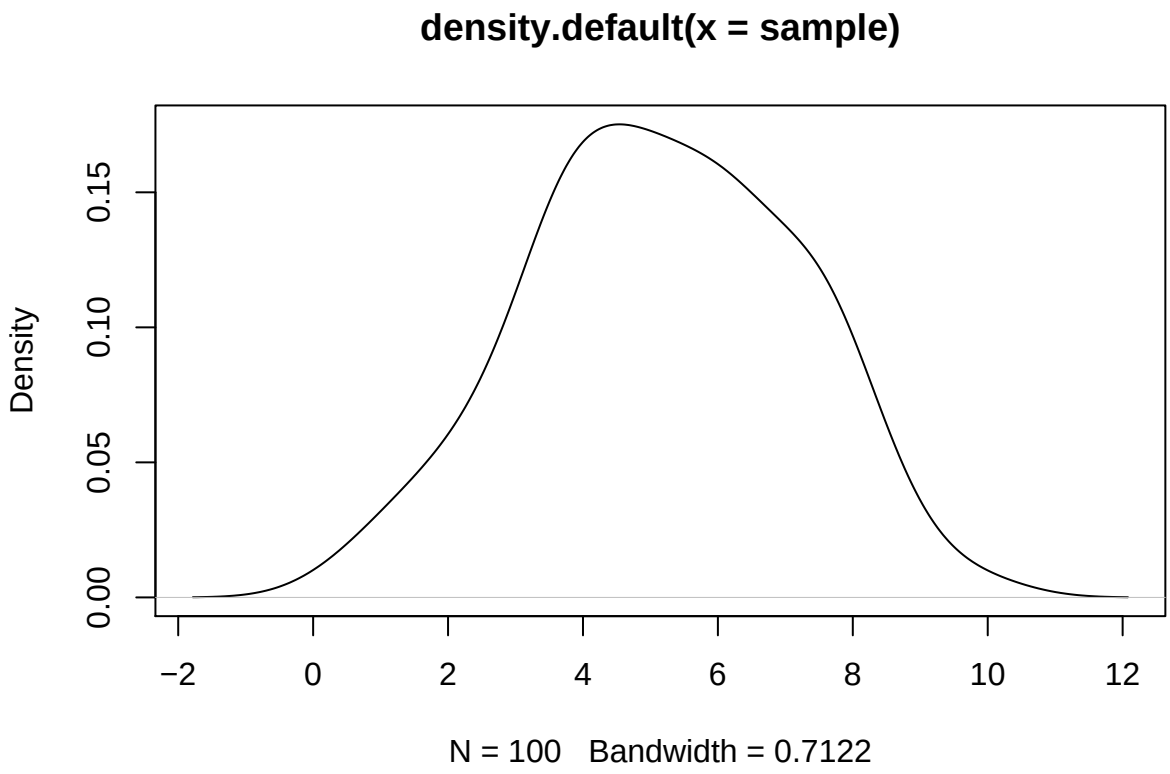



4. Use this vector to plot density function. You can use either basic R or ggplot.

```
ggplot(data=data.frame(s=sample), aes(x=s)) +
  geom_density()
```




```
plot(density(sample))
```



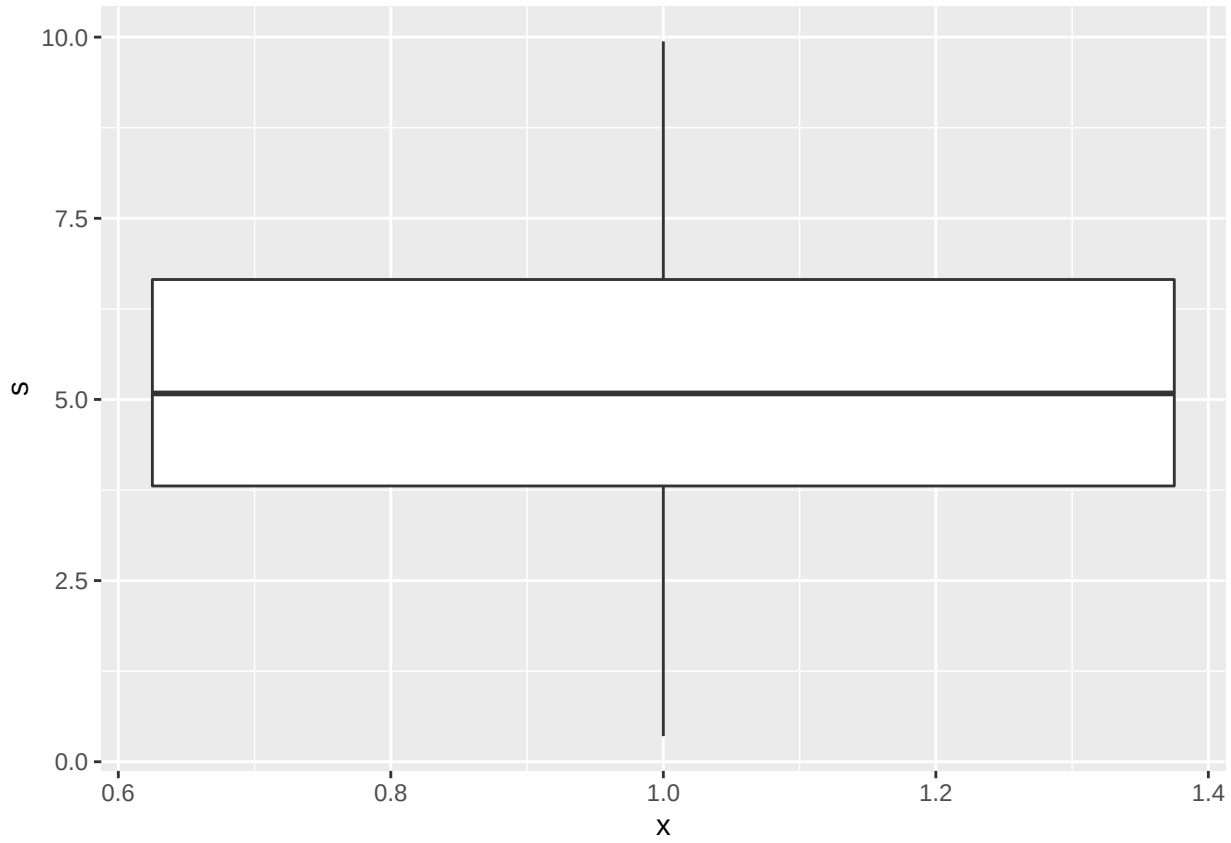

5. Find the 42nd, 77th and 99th percentiles of this vector.

```
quantile(sample, c(.42, .77, .99))
```

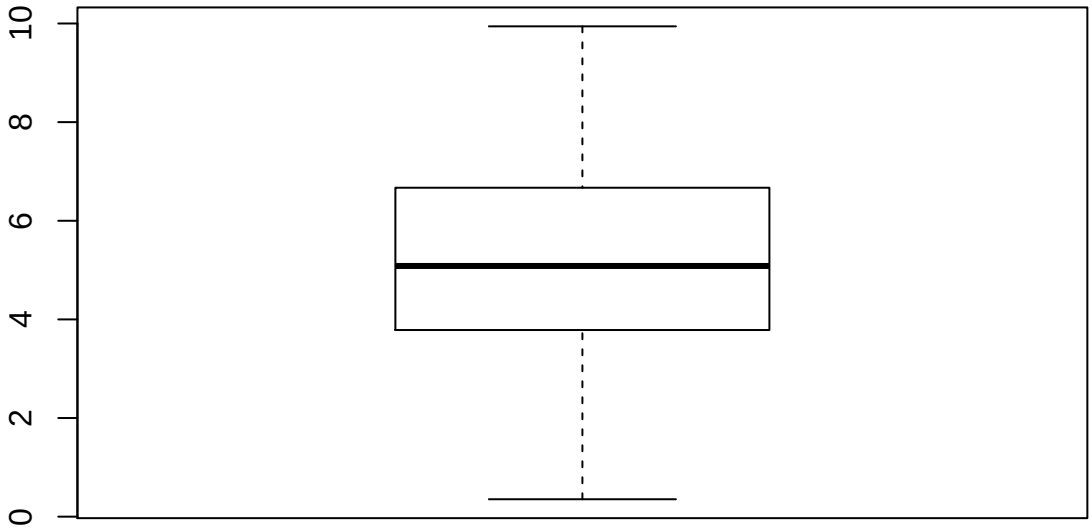
```
##      42%      77%      99%
## 4.792162 6.769851 8.931177
```

6. Plot boxplot using this vector. You can use either basic R or ggplot.

```
ggplot(data=data.frame(s=sample), aes(y=s, x = 1)) +
  geom_boxplot()
```




```
boxplot(sample)
```




7. Find interquartile range using this vector.

```
IQR(sample)
```

```
## [1] 2.848841
```

```
# [1] 1.34361
```

8. Compute 20th percentile, upper and lower quartile of Poisson distribution with $= 14$.

```
qpois(c(0.2, 0.25, 0.75), lambda=14)
```

```
## [1] 11 11 16
```

```
# [1] 11 11 16
```

9. There are 20 multiple choice questions in a test. Each question has 5 possible answers and only one of them is correct.

- Find the probability of having exactly 1 correct answer if a student attempts to answer every question at random.
- Find the probability of having 5 or less correct answers if a student attempts to answer every question at random.

Looks like we can use Binomial distribution here, random variable X is number of correct answers in 20 trials (questions) where the probability of success is the probability of answering one question correctly by random is $1/5 = 0.2$.

```
# prob of having exactly 1 answer
dbinom(x=1, size=20, prob=0.2)
```

```
## [1] 0.05764608
```

```
# [1] 0.05764608

# you can always look at the distribution in order to check sanity
# plot(dbinom(x=seq(0, 20, 1), size=20, prob=0.2), type='line')

# prob of having 5 or less correct answers
sum(dbinom(x=c(0, 1, 2, 3, 4, 5), size=20, prob=0.2))
```

```
## [1] 0.8042078
```

```
# [1] 0.8042078

# alternatively, we can use the cumulative probability function:
# it's just a probability of observing q or less number of successes
pbinom(q = 5, size=20, prob=0.2)
```

```
## [1] 0.8042078
```

```
# [1] 0.8042078
```

10. (this example does not reflect reality!) There are 5 transcription-factor binding sites (TF BSs) on average in a bin of length 2000nt. Find the probability of having more than 15 TF BSs in a particular bin of the same length.

Seems like we can apply Poisson distribution here: we definitely have src="https://render.githubusercontent.com/render/math?math=/lambda"> = 5 (average number of TF BSs per bin).

```
# basically this function gives you P(X <= x)
# so we need to subtract it from 1 in order to get the right value
1 - ppois(q = 15, lambda = 5)
```

```
## [1] 6.900824e-05
```

```
# or we can ask the function to calculate the right tail using lower.tail=F:
ppois(q = 15, lambda = 5, lower.tail = F)
```

```
## [1] 6.900824e-05
```

11. Generate 10 observations from t-distribution.

```
n <- 10
df <- n - 1
rt(n=n, df=df)
```

```
##  [1] -0.363071984 -0.003634448  0.327046200 -0.074744191 -1.644744332
##  [6] -0.955505798 -0.535909916  1.182891078 -1.829635528  1.122645070
```

```
# [1] -0.3149978 -0.3176453  0.8626783  0.6758479  1.2885294
# [6] -0.3888994 -4.4362755  0.9111234  1.1605962 -0.9399259
```

12. Find 1st, 4th and 99th percentile of uniform distribution with parameters: min=1, max=6.

```
qunif(c(0.01, 0.04, 0.99), min=1, max=6)
```

```
## [1] 1.05 1.20 5.95
```

```
# [1] 1.05 1.20 5.95
```

13. Find 68%, 75% and 95% confidence intervals for a population mean, if you have this sample:

```
sample <- c(-2.14, 7.21, -0.98, 2.14, 2.66, -2.48, -4.64, 3.08, -2.82, 5.84,
            3.17, 8.71, 6.5, 4.97, 6.08, 13.2, 10.29, 3.78, 7.2, 5.6, 3.34,
            7.67, 10.88, 5.01, 14.37, 7.64, 11.42, 10.64, 9.02, 7.9, 6.05, 11.25)

CI(sample, ci=c(0.68, 0.75, 0.95))
```

```
##   upper1   upper2   upper3     mean   lower1   lower2   lower3
## 6.555943 6.692047 7.422193 5.705000 4.854057 4.717953 3.987807
```

```
# [1] upper1   upper2   upper3     mean   lower1   lower2   lower3
# [1] 6.555943 6.692047 7.422193 5.705000 4.854057 4.717953 3.987807
```

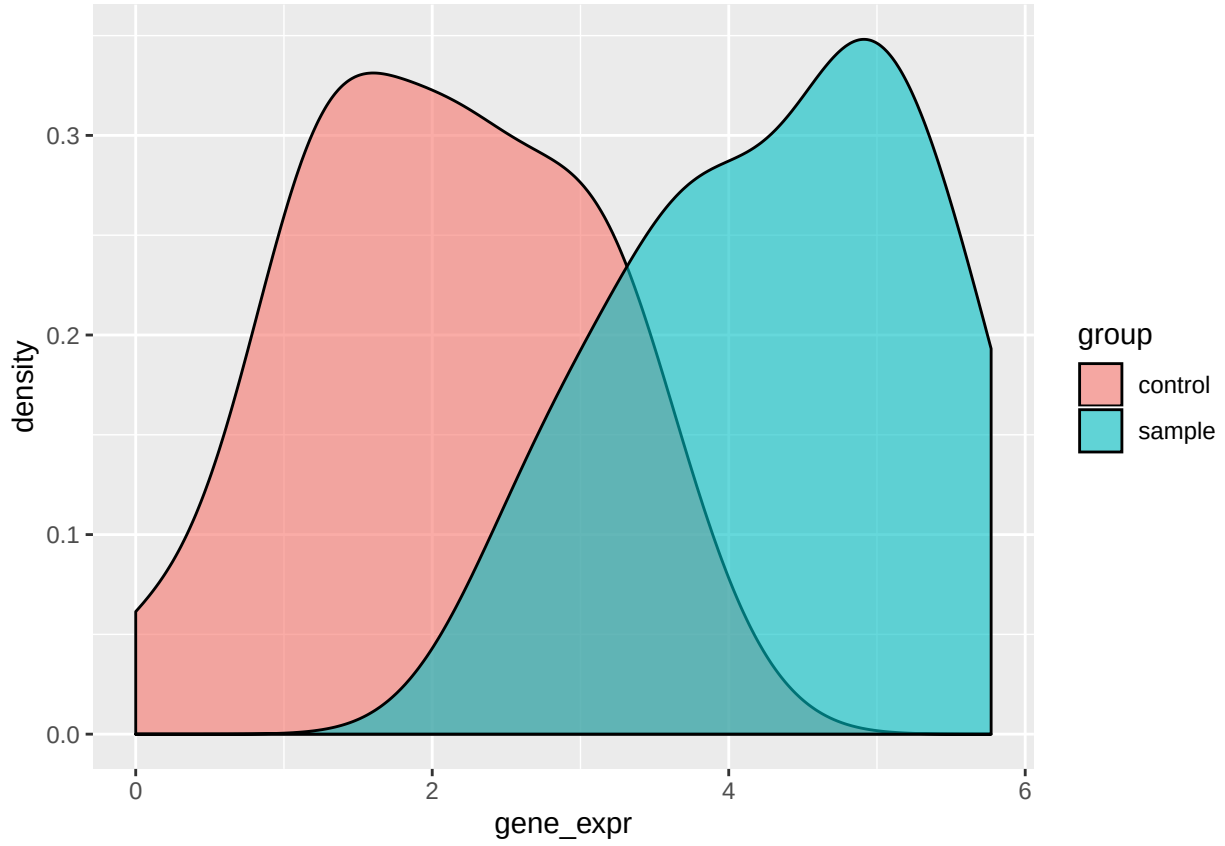
# Part 2

1. Compare 2 samples means using appropriate test.

Suppose, we have got gene expression data for some gene in a samples under stress condition (e.g. starvation) and control samples (untreated). We want to identify whether expression of the gene changes during stress (in both directions, it could be up-regulated or down-regulated).

- Plot the data, spot the difference between 2 groups if any:

```
df <- data.frame(sample = c(4.63, 3.72, 3.81, 5.22, 5.19, 4.86, 5.49, 2.46, 4.43, 3.87, 4.88, 3.2, 3.64
                 control = c(2.96, 3.07, 1.13, 3.76, 1.33, 3.06, 2.19, 1.32, 2.23, 0, 0.76, 2.52, 2.18, 


df %>% tidyr::gather(group, gene_expr) %>%
  ggplot(aes(x= gene_expr, fill=group))+
  geom_density(alpha=0.6)
```




- What kind of test should be applied here, paired or unpaired?

Unpaired, there are 2 different samples (objects are independent).

- What are hypotheses, one-sided or two-sided?

H0: mean_condition = mean_control H1: mean_condition != mean_control

- Check whether data distributed normally in order to choose between parametric and non-parametric test.

For both groups p-value $> 0.05$ implying that the distribution of the data are not significantly different from normal distribution:

```
shapiro.test(df$sample)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df$sample
## W = 0.95819, p-value = 0.5083
```

```
# [1] Shapiro-Wilk normality test

# [1] data:  df$sample
# [1] W = 0.95819, p-value = 0.5083

shapiro.test(df$control)
```

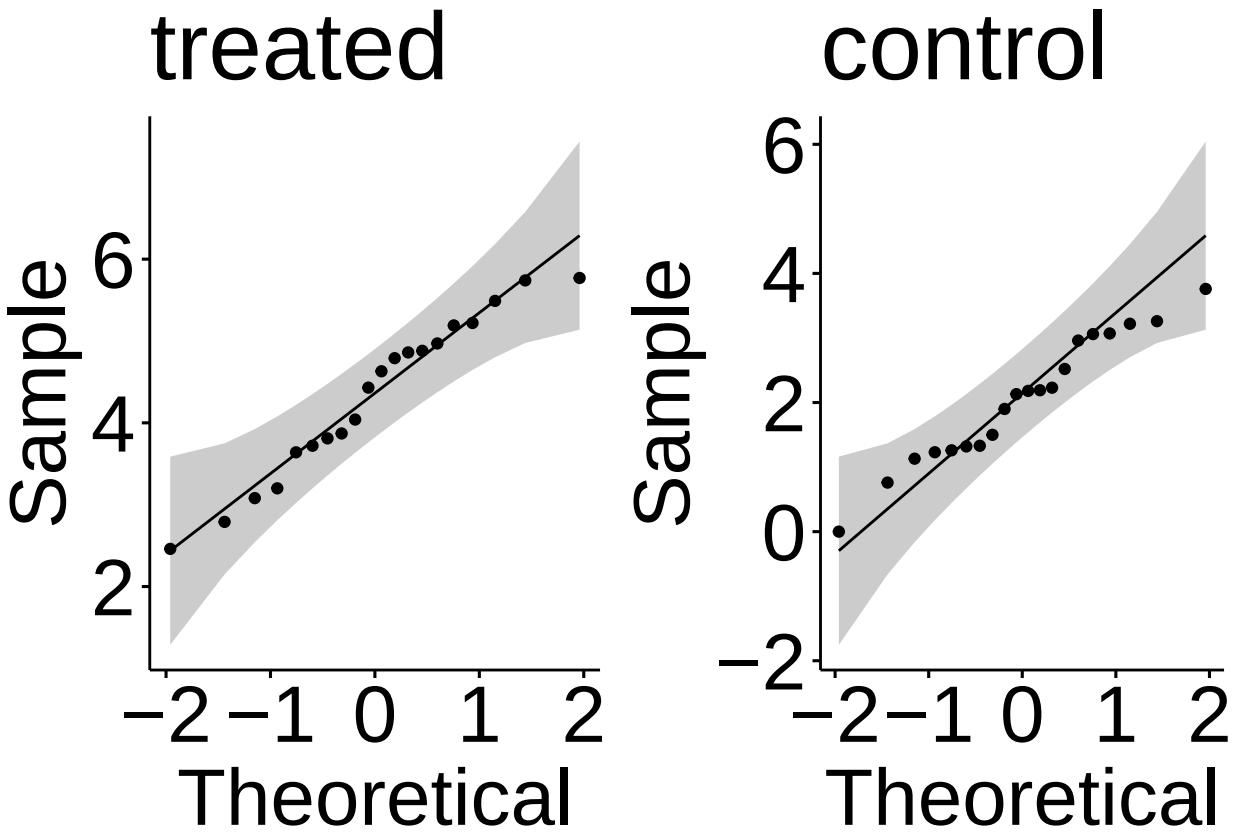
```
##
```

```
##  Shapiro-Wilk normality test
##
## data:  df$control
## W = 0.96717, p-value = 0.6944
```

```
# [1] Shapiro-Wilk normality test

# [1] data:  df$control
# [1] W = 0.96717, p-value = 0.6944
```

- Look at qqplot:

```
p1 <- ggqqplot(df$sample) + labs(title='treated') + theme(text = element_text(size=30))
p2 <- ggqqplot(df$control) + labs(title='control') + theme(text = element_text(size=30))

plot_grid(p1, p2, nrow = 1)
```




- The variances of the groups should be compared (whether they are equal). P-value >= 0.05, we can't reject H0 that variance are the same.

```
bartlett.test(list(df$sample, df$control))
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  list(df$sample, df$control)
## Bartlett's K-squared = 1.9944e-05, df = 1, p-value = 0.9964
```

```
# [1] Bartlett test of homogeneity of variances

# [1] data:  list(df$sample, df$control)
# [1] Bartlett's K-squared = 1.9944e-05, df = 1, p-value = 0.9964

df %>% gather(group, count) %>% mutate(group = factor(group)) -> tmp
leveneTest(tmp$count, tmp$group)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value Pr(>F)
## group  1  0.0308 0.8617
##       38
```

```
# [1] Levene's Test for Homogeneity of Variance (center = median)
# [1]     Df F value Pr(>F)
# [1] group  1  0.0308 0.8617
# [1]     38
```

Finally, we can use unpaired T-test with equal variances and two-sided alternative.

p-value = 8.77e-09, t = 7.3324. Confidence interval for the mean difference does not contain 0 and strictly positive: (1.649432, 2.907568). So we have to reject H0 that means are equal.

```
t.test(x = df$sample,
       y =  df$control,
       alternative = 'two.sided',
       var.equal = T,
       paired = F)
```

```
##
##  Two Sample t-test
##
## data:  df$sample and df$control
## t = 7.3324, df = 38, p-value = 8.77e-09
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  1.649432 2.907568
## sample estimates:
## mean of x mean of y
##    4.3290    2.0505
```

```
# [1] Two Sample t-test

# [1] data:  df$sample and df$control
# [1] t = 7.3324, df = 38, p-value = 8.77e-09
# [1] alternative hypothesis: true difference in means is not equal to 0
# [1] 95 percent confidence interval:
# [1] 1.649432 2.907568
# [1] sample estimates:
# [1] mean of x mean of y
# [1]   4.3290    2.0505
```

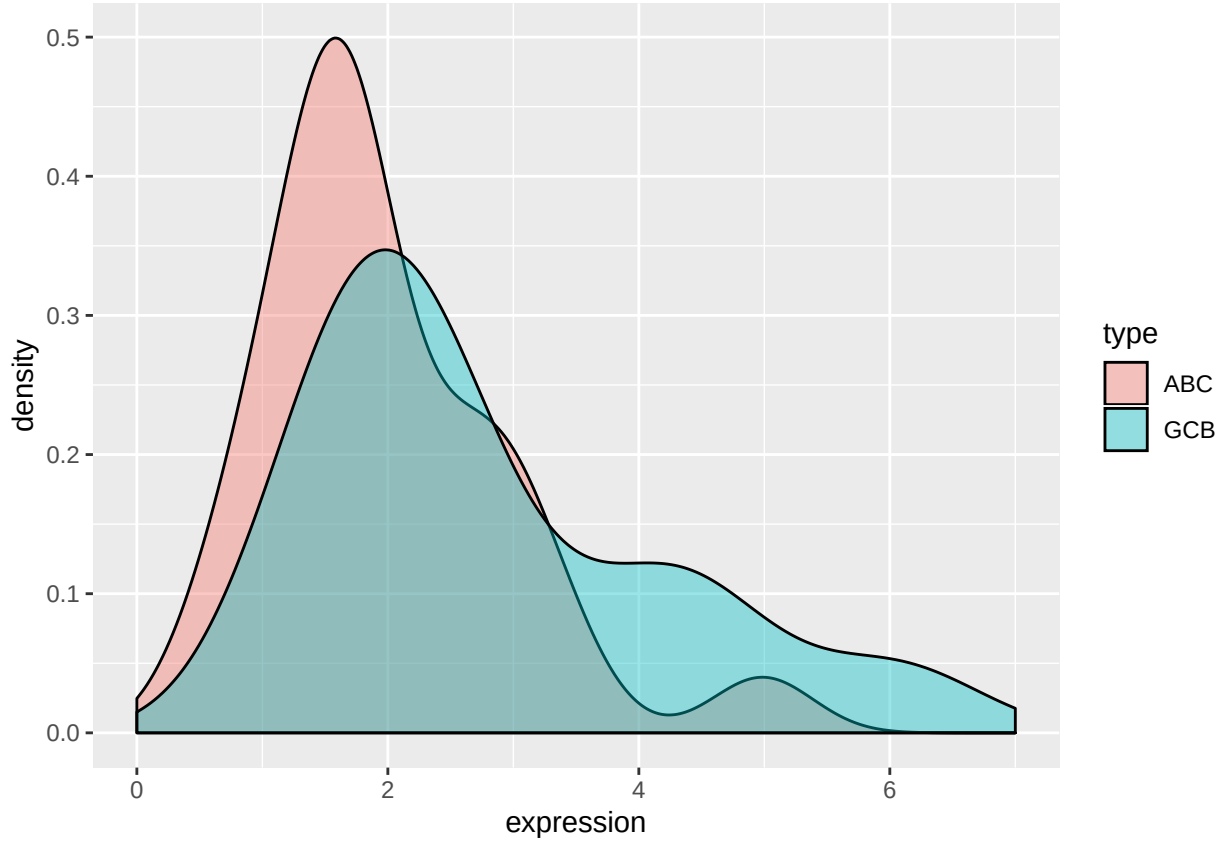
2. Compare 2 samples means using appropriate test.

Suppose that we want to test whether gene BCL-2 plays an important role with respect to discriminating DLBCL ABC from DLBCL GCB patients. We are interested whether an expression of BCL-2 in patients with GCB type is higher than in patients with ABC.

In dataframe there is BCL-2 expression for patients with DLBCL ABC and DLBCL GCB. (this example as any others is artifical)

- plot it first

```
df <- data.frame(ABC = c(1.736, 3.408, 2.54, 1.501, 1.405, 2.057, 2.924, 3.147, 2.309, 2.774,
                         1.929, 1.695, 1.467, 1.61, 4.986, 1.684, 0.926, 1.163, 2.8, 1.125,
                         0.8, 0.56, 1.408, 1.704, 1.724),
                 GCB = c(1.605, 1.662, 2.468, 2.231, 2.163, 1.673, 2.536, 2.41, 1.205, 4.508,
                         1.475, 1.617, 1.906, 2.55, 1.55, 3.756, 6.132, 4.455, 4.448, 1.688,
                         2.091, 2.312, 5.972, 4.213, 3.11))

df %>% gather(type, expression) %>%
  ggplot(aes(x =  expression, fill = type)) +
  geom_density(alpha = 0.4) +
  xlim(0, 7)
```




- Are data paired or not?

No, these are different patients.

- What are hypotheses, one-sided or two-sided?

one-sided: H0: mean_GCB = mean_ABC H1: mean_GCB > mean_ABC

- Check normality of the data. p-value $< 0.05$, we have to reject H0. Data are not normally distributed.

```
shapiro.test(df$ABC)
```

```
##
```

```
##  Shapiro-Wilk normality test
##
## data:  df$ABC
## W = 0.90649, p-value = 0.02547
```

```
# [1] Shapiro-Wilk normality test

# [1] data:  df$ABC
# [1] W = 0.90649, p-value = 0.02547
```

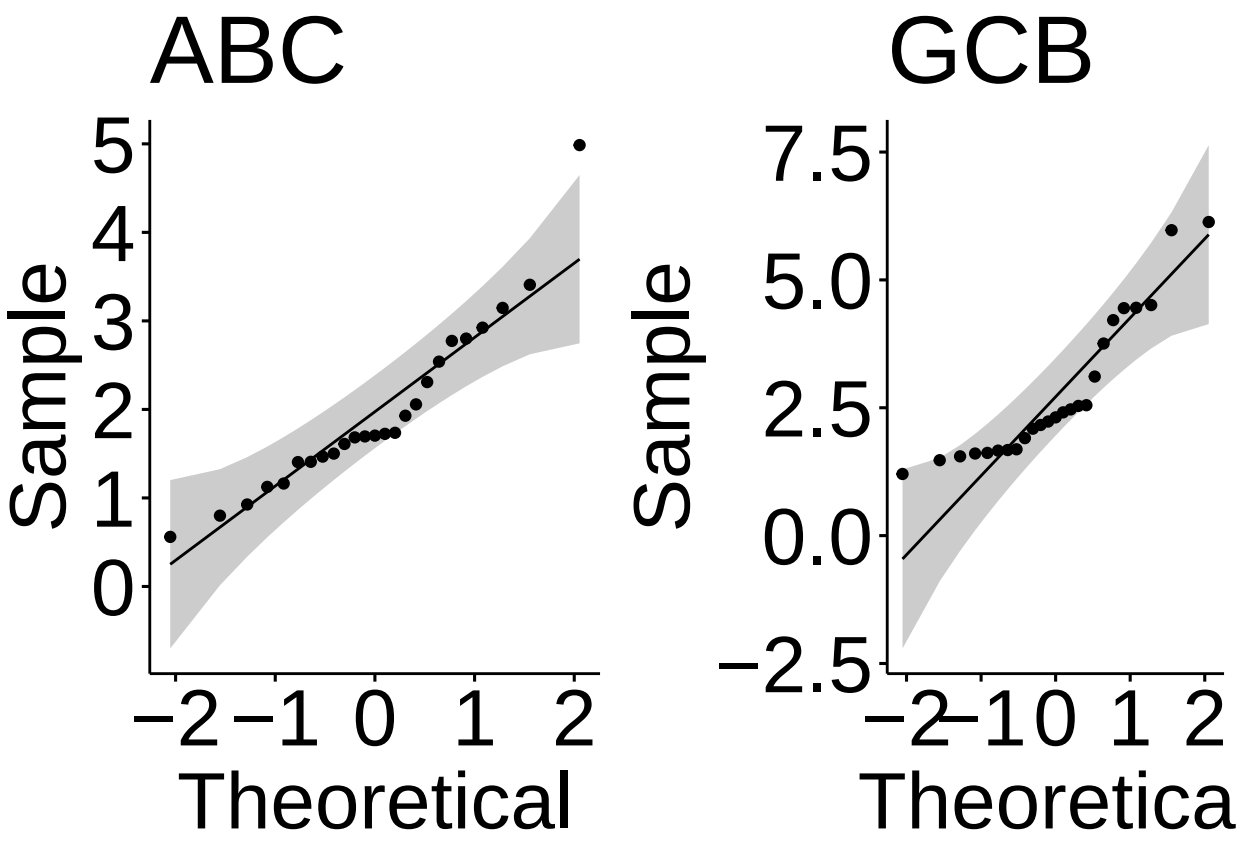
```
shapiro.test(df$GCB)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df$GCB
## W = 0.84899, p-value = 0.001686
```

```
# [1] Shapiro-Wilk normality test

# [1] data:  df$GCB
# [1] W = 0.84899, p-value = 0.001686
```

- Look at qqplot as well:

```
p1 <- ggqqplot(df$ABC) + labs(title='ABC') + theme(text = element_text(size=30))
p2 <- ggqqplot(df$GCB) + labs(title='GCB') + theme(text = element_text(size=30))

plot_grid(p1, p2, nrow = 1)
```




- We have to apply non-parametric test to compare 2 groups.

p-value = 0.01765, it's < 0.05, so we can reject H0 => BCL-2 expression is higher in GCB patients than in ABC patients.

```
wilcox.test(df$GCB,
            df$ABC,
            alternative = "greater",
            paired = F)
```

```
##
##  Wilcoxon rank sum test
##
## data:  df$GCB and df$ABC
## W = 421, p-value = 0.01765
## alternative hypothesis: true location shift is greater than 0
```

```
# [1] Wilcoxon rank sum test

# [1] data:  df$GCB and df$ABC
# [1] W = 421, p-value = 0.01765
# [1] alternative hypothesis: true location shift is greater than 0
```

3. Suppose you performed genome wide association study (GWAS) for n=20 SNPs. You have got p-values. Apply any multiple adjustment correction. How many significant p-values do you still observe?

```
p_values <- c(0.6082, 0.0266, 0.0174, 0.5522, 0.9615, 0.3277, 0.7874, 0.2051, 0.4608, 0.0472, 0.0164, 0
```

```r
pvalues_adj <- p.adjust(p_values, method="BH")

sum(pvalues_adj[pvalues_adj<0.05])
```

```
## [1] 0
```

```r
# sad but true, there is no 'significant' pvalues left after correction :(
```

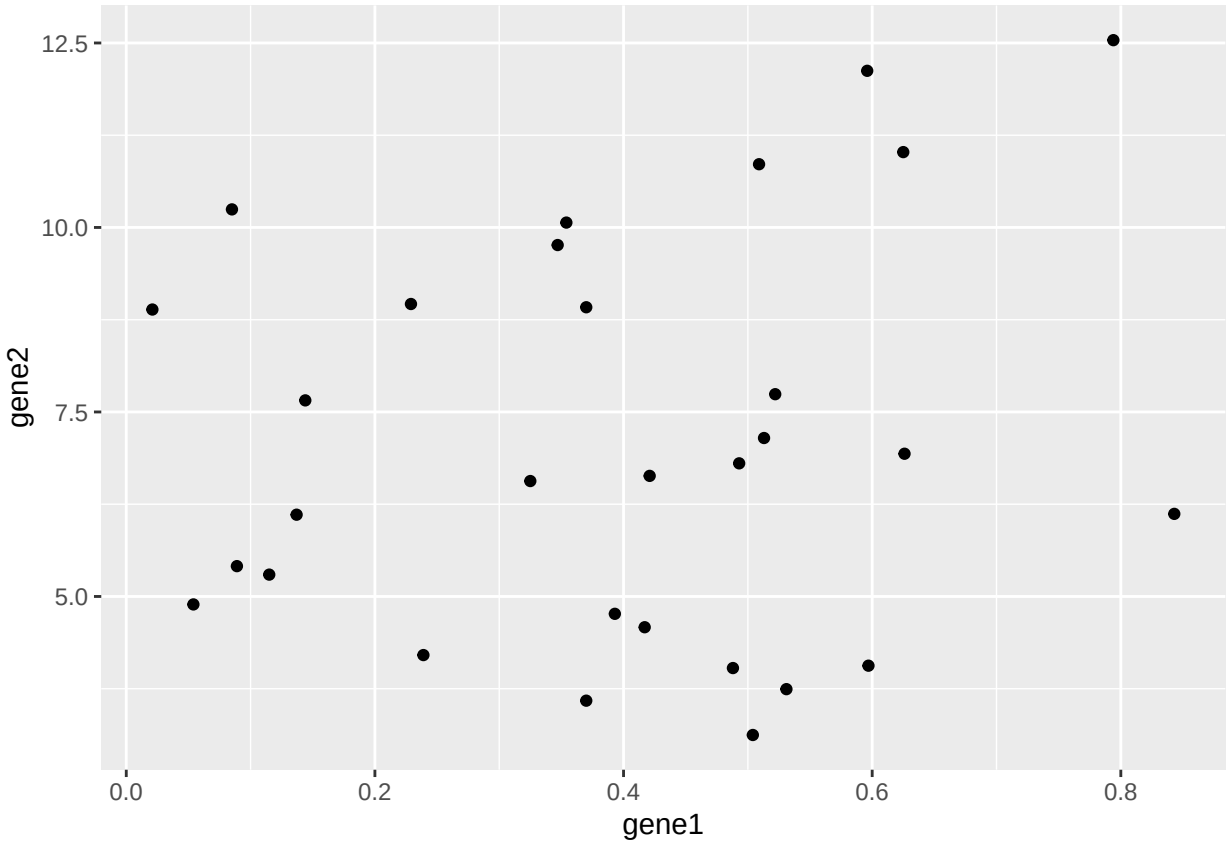
4. Suppose you want to check whether there is a correlation between 2 gene expressions. You have 30 samples. Choose an appropriate test for that.

```r
two_genes <- data.frame(gene1 = c(0.089, 0.239, 0.531, 0.054, 0.625, 0.488, 0.522, 0.37, 0.347,
                                  0.393, 0.513, 0.794, 0.354, 0.085, 0.144, 0.493, 0.021, 0.596,
                                  0.417, 0.504, 0.597, 0.229, 0.137, 0.843, 0.37, 0.421, 0.509,
                                  0.626, 0.325, 0.115),
                        gene2 = c(5.411, 4.206, 3.744, 4.892, 11.021, 4.03, 7.741, 3.588, 9.762,
                                  4.765, 7.147, 12.538, 10.066, 10.245, 7.657, 6.804, 8.888, 12.123,
                                  4.583, 3.123, 4.062, 8.963, 6.108, 6.119, 8.919, 6.634, 10.857, 6.934
                                  6.564, 5.296))
```

- Plot it first to be sure that it looks like there is a linear dependency in the data

  It looks more like a cloud rather than it has clear linear.

```r
ggplot(two_genes, aes(x = gene1, y = gene2)) +
  geom_point()
```




- Check normality of the data

P-value > 0.05, we can't reject H0, so data seem to be normally distributed.

```
shapiro.test(two_genes$gene1)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  two_genes$gene1
## W = 0.96377, p-value = 0.3852
```

```
# [1] Shapiro-Wilk normality test

# [1] data:  two_genes$gene1
# [1] W = 0.96377, p-value = 0.3852


shapiro.test(two_genes$gene2)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  two_genes$gene2
## W = 0.94914, p-value = 0.1603
```

```
# [1] Shapiro-Wilk normality test

# [1] data:  two_genes$gene1
# [1] W = 0.94914, p-value = 0.1603
```

- Use an approriate test

We can use parametric correlation test - Pearson. P-value = 0.4454, so we can't reject H0. 95 percent confidence interval = (-0.2273846, 0.4799819) does contain 0. So we can claim that 2 genes are not correlated.

```
cor.test(two_genes$gene1, two_genes$gene2, method = 'pearson')
```

```
##
##  Pearson's product-moment correlation
##
## data:  two_genes$gene1 and two_genes$gene2
## t = 0.77405, df = 28, p-value = 0.4454
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.2273846  0.4799819
## sample estimates:
##       cor
## 0.1447418
```

```
# [1] Pearson's product-moment correlation

# [1] data:  two_genes$gene1 and two_genes$gene2
# [1] t = 0.77405, df = 28, p-value = 0.4454
# [1] alternative hypothesis: true correlation is not equal to 0
# [1] 95 percent confidence interval:
# [1]  -0.2273846  0.4799819
# [1] sample estimates:
# [1]       cor
# [1] 0.1447418
```

# Part 3

We are going to predict the phenotype feature based on expression of several genes. Build linear regression models and compare them.

```
df <- read.table('datasets/linreg_task.txt', sep='\t')

head(df, 3)
```

```
##   feature  geneA  geneB  geneC  geneD
## 1   5.752 16.643  1.108 34.244 28.418
## 2  15.766 72.700 19.274 57.923 74.064
## 3   8.180 24.745 -9.423 32.620 31.759
```

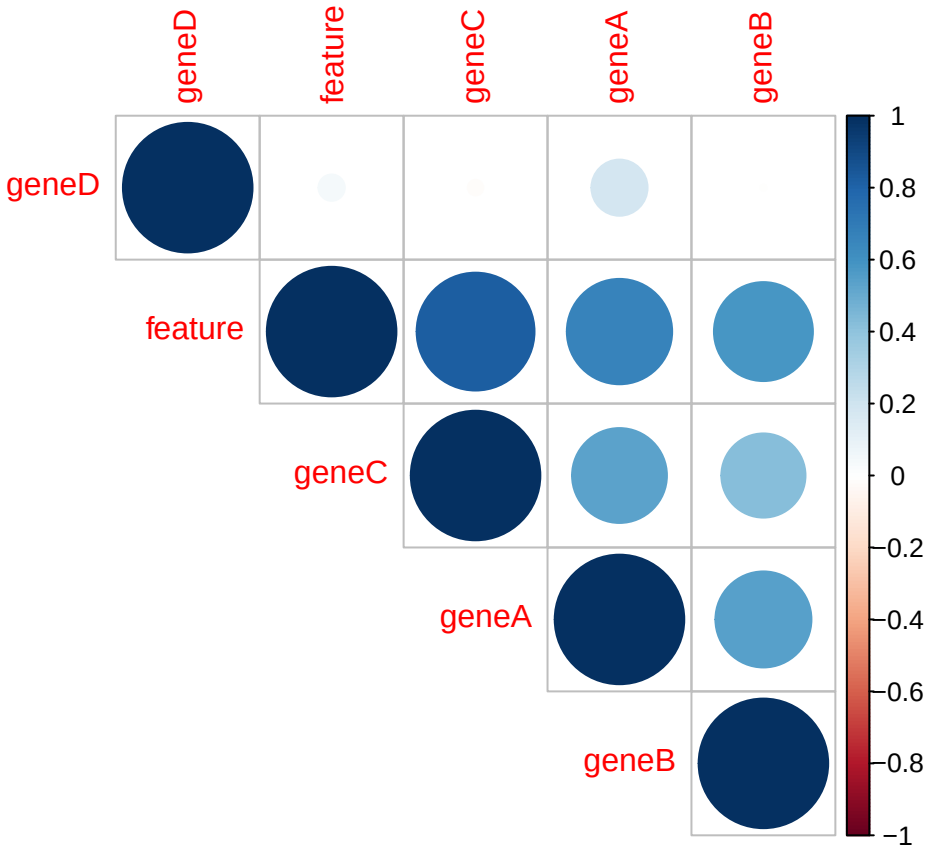
```
# [1]     feature    geneA     geneB     geneC   geneD
# [1]1      5.752     16.643       1.108      34.244    28.418
# [1]2     15.766     72.700      19.274      57.923    74.064
# [1]3      8.180     24.745      -9.423      32.620    31.759
```

- Plot scatter plots and correlation plot:

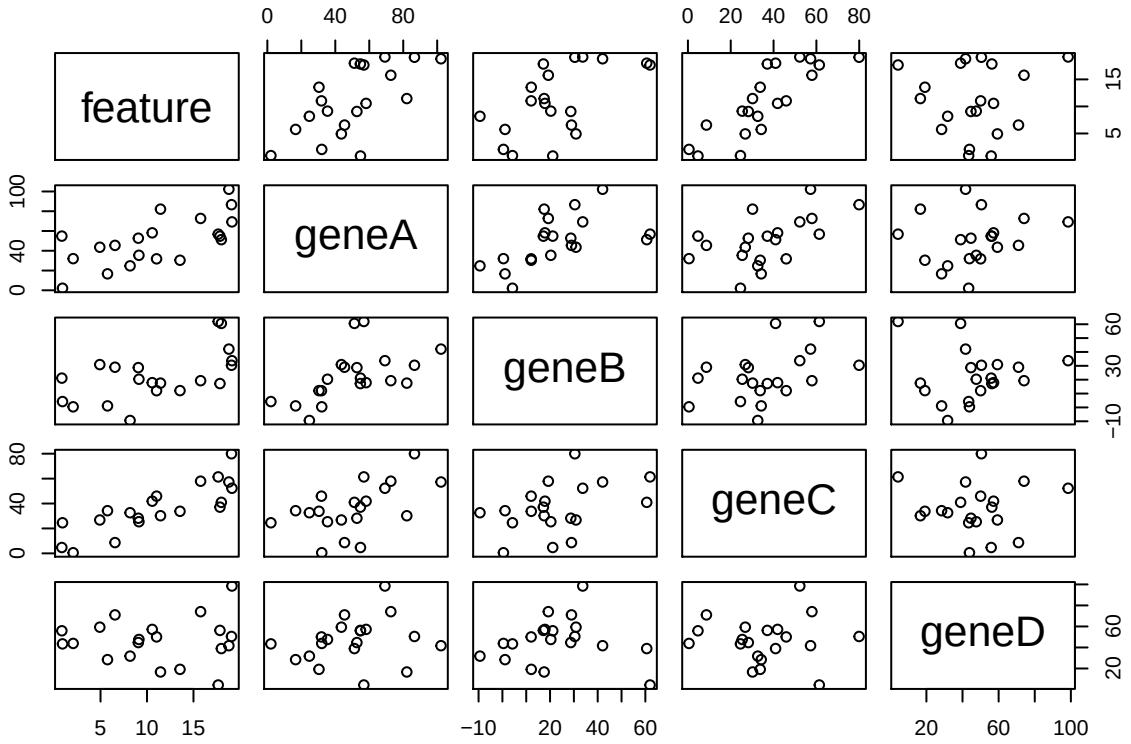
```
# plot the correlation
corr_mat <- cor(df)
corr_mat
```

```
##           feature     geneA        geneB      geneC        geneD
## feature 1.0000000 0.6611209  0.583259589  0.8277919  0.041253996
## geneA   0.6611209 1.0000000  0.549246516  0.5362803  0.187782812
## geneB   0.5832596 0.5492465  1.000000000  0.4218939 -0.002098192
## geneC   0.8277919 0.5362803  0.421893929  1.0000000 -0.014213600
## geneD   0.0412540 0.1877828 -0.002098192 -0.0142136  1.000000000
```

```
corrplot(corr_mat, type="upper", order="hclust")
```




```
# the highest correlation between feature and geneC - 0.8277919
# we will start to build a model with this one predictor

# plot scatterplots
plot(df)
```




- build a simple model with geneC.

Distribution of residuals appears to be approximately normal (symmetrical values for Min and Max, 1Q and 3Q in comparison to median).

Pr(>|t|) for geneC is 6.65e-06, so this coefficient is significant.

Adjusted R-squared is 0.6678, it's close to 1, which means that this model explained a considerable part of variation in the data.

Residual standard error: 3.614 on 18 degrees of freedom

F-statistic: 39.19 on 1 and 18 DF, p-value: 6.648e-06

```
model1 <- lm(feature ~ geneC, data=df)

model1
```

```
##
## Call:
## lm(formula = feature ~ geneC, data = df)
##
## Coefficients:
## (Intercept)        geneC
##      1.4696       0.2636
```

```
summary(model1)
```

```
##
## Call:
```

```
## lm(formula = feature ~ geneC, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.0287 -2.0992  0.0847  2.3805  6.6011
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.46962    1.72587   0.852    0.406
## geneC        0.26365    0.04212   6.260 6.65e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.614 on 18 degrees of freedom
## Multiple R-squared:  0.6852, Adjusted R-squared:  0.6678
## F-statistic: 39.19 on 1 and 18 DF,  p-value: 6.648e-06
```

```
# [1] Call:
# [1] lm(formula = feature ~ geneC, data = df)

# [1] Residuals:
# [1]     Min      1Q  Median      3Q     Max
# [1] -7.0287 -2.0992  0.0847  2.3805  6.6011

# [1] Coefficients:
# [1]             Estimate Std. Error t value Pr(>|t|)
# [1] (Intercept)  1.46962    1.72587   0.852    0.406
# [1] geneC        0.26365    0.04212   6.260 6.65e-06 ***
# [1] ---
# [1] Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

# [1] Residual standard error: 3.614 on 18 degrees of freedom
# [1] Multiple R-squared:  0.6852,  Adjusted R-squared:  0.6678
# [1] F-statistic: 39.19 on 1 and 18 DF,  p-value: 6.648e-06
```

- incorporate into the model all genes at once.

Distribution of residuals seems to be less symmetrical as it was in the model1.

Only geneC is still significant.

Adjusted R-squared = 0.7198, which is higher than was for model1.

But do we really need to include so many predictors in the model? Basically, when building linear regression models or any model, we just want to be able to interpret it and many predictors are not much helpful.

```
model2 <- lm(feature ~ ., data=df)

summary(model2)
```

```
##
## Call:
## lm(formula = feature ~ ., data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.4693 -1.5765 -0.3973  1.3025  6.7507
```

```
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.654055   2.390219  -0.274 0.788095
## geneA        0.053545   0.041792   1.281 0.219572
## geneB        0.069361   0.051012   1.360 0.194010
## geneC        0.200644   0.046922   4.276 0.000663 ***
## geneD        0.003285   0.036966   0.089 0.930359
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.319 on 15 degrees of freedom
## Multiple R-squared:  0.7788, Adjusted R-squared:  0.7198
## F-statistic:  13.2 on 4 and 15 DF,  p-value: 8.346e-05
```

```
# [1] Call:
# [1] lm(formula = feature ~ ., data = df)

# [1] Residuals:
# [1]     Min      1Q  Median      3Q     Max
# [1] -4.4693 -1.5765 -0.3973  1.3025  6.7507

# [1] Coefficients:
# [1]              Estimate Std. Error t value Pr(>|t|)
# [1] (Intercept) -0.654055   2.390219  -0.274 0.788095
# [1] geneA        0.053545   0.041792   1.281 0.219572
# [1] geneB        0.069361   0.051012   1.360 0.194010
# [1] geneC        0.200644   0.046922   4.276 0.000663 ***
# [1] geneD        0.003285   0.036966   0.089 0.930359
# [1] ---
# [1] Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

# [1] Residual standard error: 3.319 on 15 degrees of freedom
# [1] Multiple R-squared:  0.7788,  Adjusted R-squared:  0.7198
# [1] F-statistic:  13.2 on 4 and 15 DF,  p-value: 8.346e-05
```

Let's just exclude geneD as a predictor because it has the lowest correlation with feature.

Distribution of residuals still seems to be less symmetrical as it was in the model1 and geneC is the only one significant coefficient here. Adjusted R-squared (0.7371) is slightly higher than in the model2.

```
model3 <- lm(feature ~ geneA + geneB + geneC, data=df)

summary(model3)
```

```
##
## Call:
## lm(formula = feature ~ geneA + geneB + geneC, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.4221 -1.5993 -0.4022  1.2999  6.7757
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.51928    1.78938  -0.290 0.775389
```

```
## geneA        0.05449    0.03915   1.392 0.183103
## geneB        0.06888    0.04913   1.402 0.179987
## geneC        0.20015    0.04512   4.436 0.000415 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.214 on 16 degrees of freedom
## Multiple R-squared:  0.7787, Adjusted R-squared:  0.7371
## F-statistic: 18.76 on 3 and 16 DF,  p-value: 1.724e-05
```

```
# [1] Call:
# [1] lm(formula = feature ~ geneA + geneB + geneC, data = df)

# [1] Residuals:
# [1]     Min      1Q  Median      3Q     Max
# [1] -4.4221 -1.5993 -0.4022  1.2999  6.7757

# [1] Coefficients:
# [1]             Estimate Std. Error t value Pr(>|t|)
# [1] (Intercept) -0.51928    1.78938  -0.290 0.775389
# [1] geneA        0.05449    0.03915   1.392 0.183103
# [1] geneB        0.06888    0.04913   1.402 0.179987
# [1] geneC        0.20015    0.04512   4.436 0.000415 ***
# [1] ---
# [1] Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

# [1] Residual standard error: 3.214 on 16 degrees of freedom
# [1] Multiple R-squared:  0.7787,  Adjusted R-squared:  0.7371
# [1] F-statistic: 18.76 on 3 and 16 DF,  p-value: 1.724e-05
```

- compare models.

Model 3 and model 1 might have a trend of difference (P-value = 0.05981) , model 3 is slightly better than model 1.

```
anova(model1, model3)
```

```
## Analysis of Variance Table
##
## Model 1: feature ~ geneC
## Model 2: feature ~ geneA + geneB + geneC
##   Res.Df    RSS Df Sum of Sq      F  Pr(>F)
## 1     18 235.06
## 2     16 165.30  2     69.76 3.3761 0.05981 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(model3, model2)
```

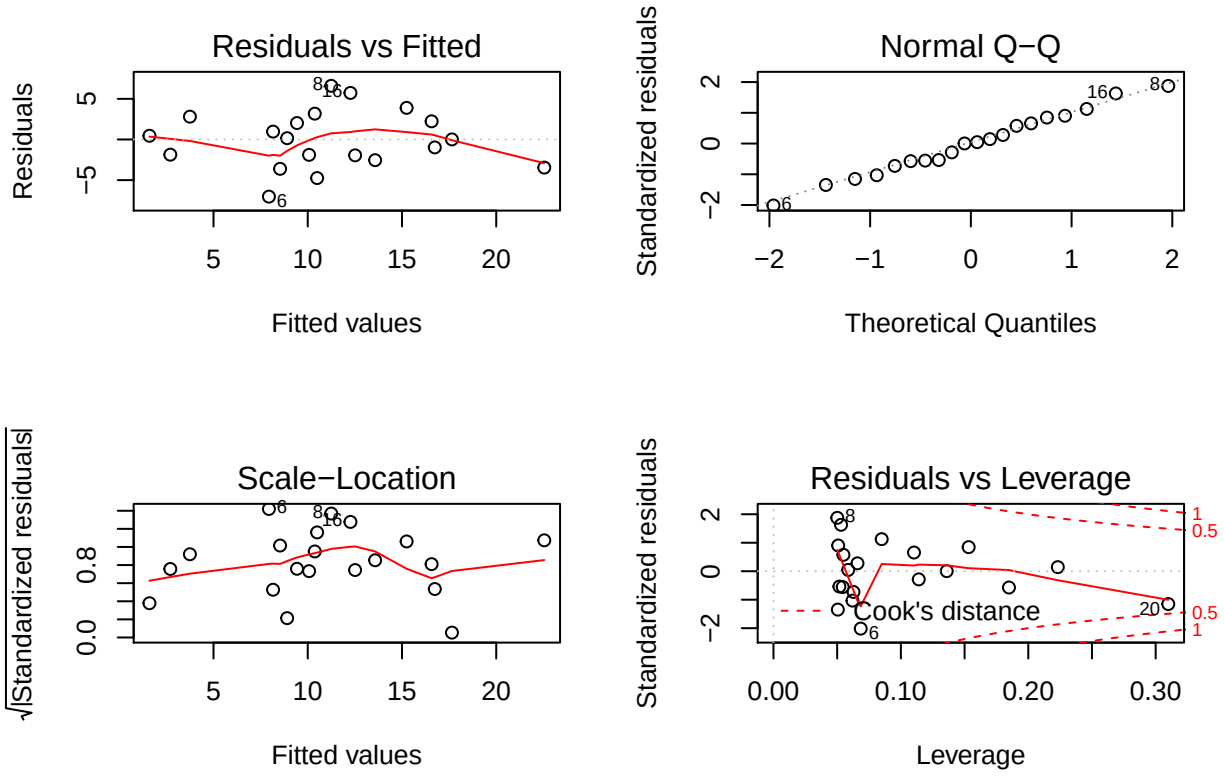
```
## Analysis of Variance Table
##
## Model 1: feature ~ geneA + geneB + geneC
## Model 2: feature ~ geneA + geneB + geneC + geneD
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     16 165.30
## 2     15 165.21  1  0.086995 0.0079 0.9304
```

Let's just summarise linear regression assumptions (that was not fully covered in the tutorial).

- Linearity of the data. The relationship between the predictorand the outcome is assumed to be linear (we used visual inspection for that - just plotted it)
- Normality of residuals. The residual errors are assumed to be normally distributed (we looked at the symmetry of the ditribution of residuals in a summary of the model)
- Homogeneity of residuals variance. The residuals are assumed to have a constant variance (homoscedasticity)

To test this we can draw diagnostic plots:

```
par(mfrow = c(2, 2))
plot(model1)
```




- Residuals vs Fitted plot is used to test the linear relationship between predictor and outcomes (approximately horizontal line tells us that the linear relationship is good)
- Normal QQ-plot is used to show whether residuals are normally distributed.
- Scale-Location plot is used to check the homogeneity of variance of the residuals. Horizontal line with equally spread points is a good indication of homoscedasticity. If it's not the case, it's possible to use a log or square root transformation of the outcome variable to reduce this problem.
- Residuals vs Leverage plot is used to identify extreme values that might influence the regression results (they are annotated with numbers). It might be usefull to know about them and probably even get rid of them.

---

Tricky question 1

As you might recall, T-statistic in 2-sample T-test follows T-distribution under null hypothesis. But how p-values are distributed under the null hypothesis?

Check it using R. Here some way to achive this:

- use function replicate to repeat multiple t-tests - 1000 t-tests for 2 samples disributed normally with the same parameters
- plot density of p-value distribution, quess by the form which distribution could it be (you can use base R functions here)
- check if the probabilities follow a particular distribution with a QQ plot (qqplot)

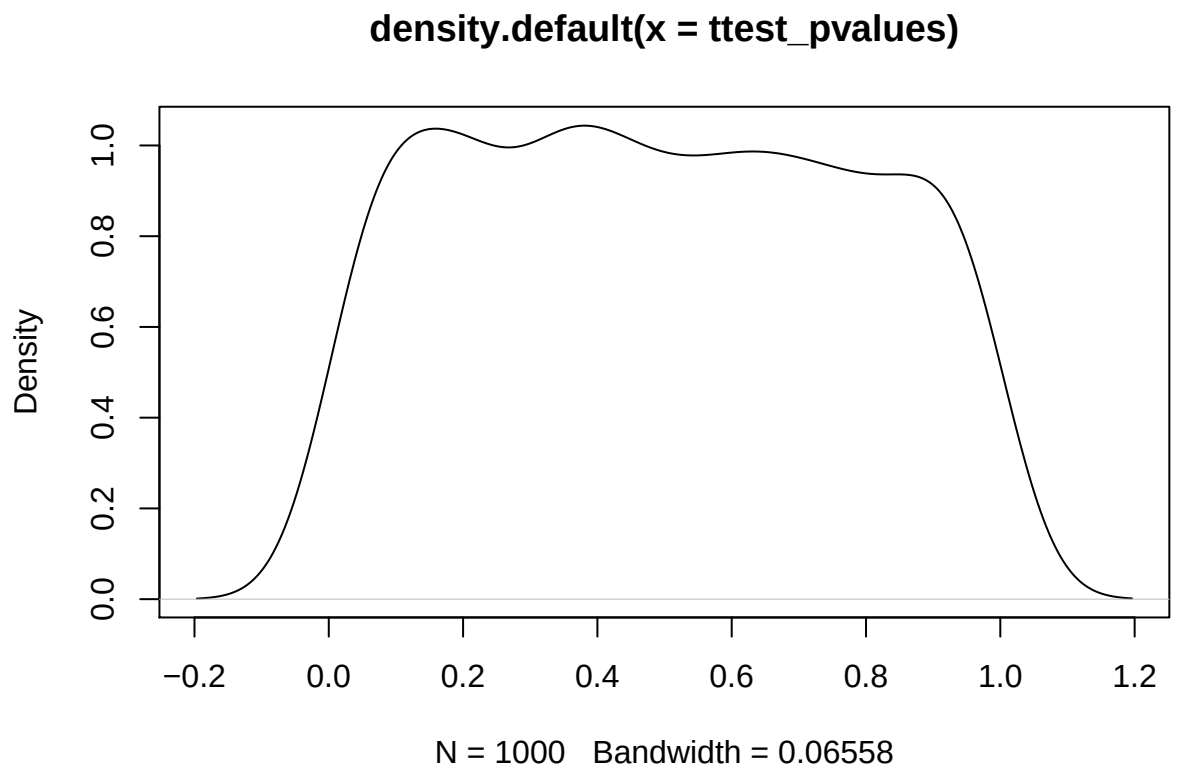
```
set.seed(47)

# it gives 1 p-value
t.test(rnorm(10, mean = 0.5, sd =5), rnorm(10, mean = 0.5, sd =5))$p.value
```

```
## [1] 0.9925137
```

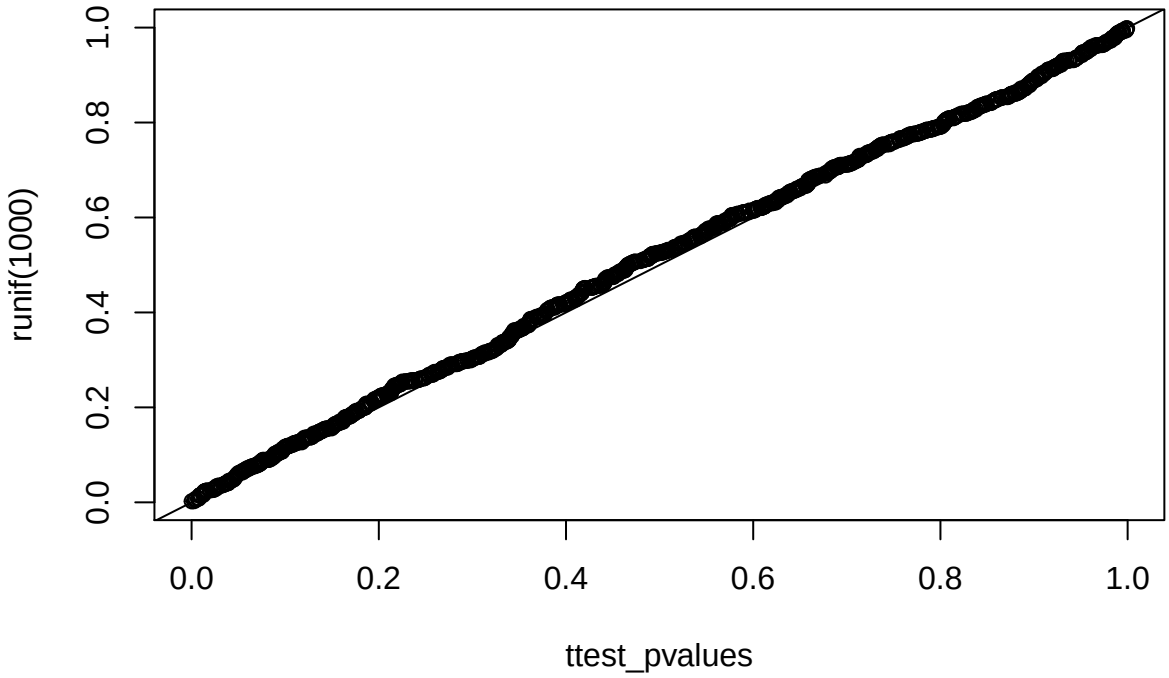
```
# replacte it 1000 times
ttest_pvalues <- replicate(1000, t.test(rnorm(10, mean = 0.5, sd =5),
                                        rnorm(10, mean = 0.5, sd =5))$p.value)
ttest_pvalues[0:4]
```

```
## [1] 0.01551859 0.84395420 0.48302413 0.36830314
```

```
plot(density(ttest_pvalues))
```




```
qqplot(ttest_pvalues, runif(1000))
abline(0,1)
```




The p-values for any statistical test should follow a uniform distribution between 0 and 1. Any value in the interval 0 to 1 is just as likely to occur as any other value.

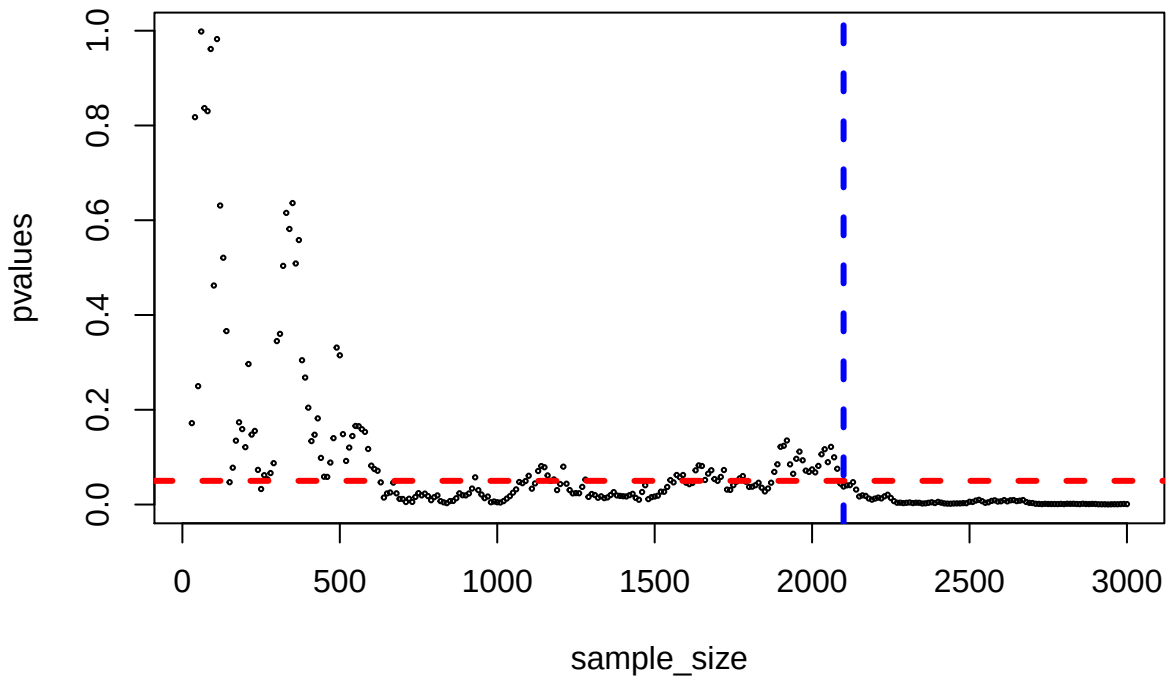
Tricky question 2

How big should be samples to be able to distinguish the very subtle difference in means?

If we have only 10 observations, we obviously can't expect great performance. Let's start with 30 observations following normal distribution with mean1 = 0 and mean2 = 0.1.

Calculate p-values of 2-sample T-test for different sample sizes and plot it. Identify the order of sample size you need to reject H0 if the difference between sample means is 0.1.

```
sample_size <- seq(30, 3000, 10)

min_sample_size <- function(sample_size) {
  set.seed(123)
  pvalue <- t.test(rnorm(sample_size, 0, 1),
                   rnorm(sample_size, 0.1, 1), var.equal = T)$p.value
  return (pvalue)
}

# as we can see here, we need at least 2100 observations to be able to reject H0 hypothesis, if the dif
# this example shows that it's better to have more observations if you are tracing subtle differences
pvalues <- sapply(sample_size, min_sample_size)

plot(x = sample_size, y = pvalues, cex=0.3)
abline(h = 0.05, col='red', lwd=3, lty=2)
abline(v = 2100, col='blue', lwd=3, lty=2)
```


pvalues
0.0
0.2
0.4
0.6
0.8
1.0
0
500
1000
1500
2000
2500
3000
sample_size