Exercises

2025-03-22

Exercise 1

1.1 Find the natural log, log to the base 10, square root and the exponential of 12.43.

```
log(12.43)              # natural log
```

```
## [1] 2.520113
```

```
log10(12.43)            # log to base 10
```

```
## [1] 1.094471
```

```
sqrt(12.43)             # square root
```

```
## [1] 3.525621
```

```
exp(12.43)              # exponent
```

```
## [1] 250196
```

1.2 Use the concatenate function `c()` to create a vector called `weight` containing the weight (in kg) of 10 children: 69, 62, 57, 59, 59, 64, 56, 66, 67, 66.

```
weight <- c(69, 62, 57, 59, 59, 64, 56, 66, 67, 66)
```

1.3 Calculate the mean, standard deviation, range of weights and the number of children of your weight vector. Next, extract the weights for the first five children using Positional indexes and store these weights in a new variable called first_five. Remember, you will need to use the square brackets `[ ]` to extract elements from a variable.

```
mean(weight)                                # calculate mean
```

```
## [1] 62.5
```

```
sd(weight)                                  # calculate standard deviation
```

```
## [1] 4.552167
```

```r
range(weight)                                # range of weight values
```

```
## [1] 56 69
```

```r
length(weight)                               # number of observations
```

```
## [1] 10
```

```r
first_five <- weight[1:5]                   # extract first 5 weight values
first_five <- weight[c(1, 2, 3, 4, 5)]      # alternative method
```

1.4 Use the `c()` function again to create another vector called `height` containing the height (in cm) of the same 10 children: 112, 102, 83, 84, 99, 90, 77, 112, 133, 112. Next, use the `summary()` function to summarise these data in the `height` object. Now, let's extract all the heights of children less than or equal to 99 cm and assign to a variable called shorter_child.

```r
height <- c(112, 102, 83, 84, 99, 90, 77, 112, 133, 112)

summary(height)   # summary statistics of height variable
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    77.0    85.5   100.5   100.4   112.0   133.0
```

```r
shorter_child <- height[height <= 99]     # extract all heights less than or equal to 99

shorter_child
```

```
## [1] 83 84 99 90 77
```

1.5 Now you can use the information in your weight and height variables to calculate the body mass index (BMI) for each child. The BMI is calculated as weight (in kg) divided by the square of the height (in meters). Store the results of this calculation in a variable called bmi.

```r
bmi <- weight/(height/100)^2    # don't forget to convert height to meters

bmi
```

```
##  [1] 55.00638 59.59246 82.74060 83.61678 60.19794 79.01235 94.45100 52.61480
##  [9] 37.87665 52.61480
```

Exercise 2

Time for a quick description of the 'whaledata.csv' dataset to get your bearings. These data were collected during two research cruises in the North Atlantic in May and October 2003. During these two months the research vessel visited multiple stations (areas) and marine mammal observers recorded the number of whales at each of these stations. The time the vessel spent at each station was also recorded along with other site specific variables such as the latitude and longitude, water depth and gradient of the seabed. The researchers also recorded the ambient level of sub-surface noise with a hydrophone and categorised this variable into 'low', 'medium' or 'high'. The structure of these data is known as a rectangular dataset with no missing cells. Each row is an individual observation and each column a separate variable. The variable names are contained in the first row of the dataset (aka a header)

2.1 Now let's import the 'whaledata.csv' file into R. To do this you will use the workhorse function of data importing, `read.csv()` and assign it a name "whale"

```r
whale <- read.csv('Data/whaledata.csv', header = T)
```

2.2 Use the `head()` function to display the first 5 rows of your dataframe. Again, this is likely to just fill up your console. Another option is to use the `str()` function to display the structure of the dataset and a neat summary of your variables. How many observations does this dataset have? How many variables are in this dataset? What type of variables are month and water.noise?

```r
head(whale)          # display the first 5 rows
```

```
##   month time.at.station water.noise number.whales latitude longitude depth
## 1   May            1344         low             7    60.37     -4.18   520
## 2   May            1633      medium            13    60.38     -4.19   559
## 3   May             743      medium            12    60.54     -4.62  1006
## 4   May            1050      medium            10    60.29     -4.35   540
## 5   May            1764      medium            12    60.41     -5.20  1000
## 6   May             580        high            10    60.38     -5.22  1000
##   gradient
## 1      415
## 2      405
## 3       88
## 4      409
## 5       97
## 6      173
```

```r
names(whale)         # display the variable names
```

```
## [1] "month"           "time.at.station" "water.noise"     "number.whales"
## [5] "latitude"        "longitude"       "depth"           "gradient"
```

```r
str(whale)           # display the structure of the dataframe whale
```

```
## 'data.frame':    100 obs. of  8 variables:
##  $ month          : chr  "May" "May" "May" "May" ...
##  $ time.at.station: int  1344 1633 743 1050 1764 580 459 561 709 690 ...
##  $ water.noise    : chr  "low" "medium" "medium" "medium" ...
##  $ number.whales  : int  7 13 12 10 12 10 5 8 11 12 ...
##  $ latitude       : num  60.4 60.4 60.5 60.3 60.4 ...
##  $ longitude      : num  -4.18 -4.19 -4.62 -4.35 -5.2 -5.22 -5.08 -5 -4.64 -4.84 ...
##  $ depth          : int  520 559 1006 540 1000 1000 993 988 954 984 ...
##  $ gradient       : int  415 405 88 409 97 173 162 162 245 161 ...
```

2.3 You can get another useful summary of your dataframe by using the `summary()` function. This will provide you with some useful summary statistics for each variable. Which variables have missing values and how many?

```r
summary(whale)
```

```
##      month           time.at.station  water.noise        number.whales
##  Length:100         Min.   :  60.0   Length:100         Min.   : 0.00
##  Class :character   1st Qu.: 693.8   Class :character   1st Qu.: 9.00
##  Mode  :character   Median :1077.5   Mode  :character   Median :11.00
##                     Mean   :1064.7                      Mean   :11.53
##                     3rd Qu.:1349.2                      3rd Qu.:14.00
##                     Max.   :2158.0                      Max.   :28.00
##                                                         NA's   :1
##     latitude       longitude          depth         gradient
##  Min.   :60.29   Min.   :-6.330   Min.   : 120   Min.   :  0.00
##  1st Qu.:60.69   1st Qu.:-4.660   1st Qu.: 987   1st Qu.: 99.25
##  Median :61.29   Median :-4.185   Median :1086   Median :132.00
##  Mean   :61.16   Mean   :-3.813   Mean   :1087   Mean   :179.78
##  3rd Qu.:61.59   3rd Qu.:-2.888   3rd Qu.:1242   3rd Qu.:245.25
##  Max.   :62.10   Max.   :-1.530   Max.   :1610   Max.   :671.00
##
```

2.4 Another useful way to manipulated your dataframes is to sort the rows based on the value of a variable. Use the `order()` function to sort all rows in the whale dataframe in ascending order of depth. Store this sorted dataframe in a variable called whale.depth.sort.

```r
whale.depth.sort <- whale[order(whale$depth), ]

head(whale.depth.sort, 10)
```

```
##      month time.at.station water.noise number.whales latitude longitude depth
## 12     May             422         low            11    60.86     -1.53   120
## 23     May            1206         low            13    61.61     -6.33   195
## 22     May             234      medium             4    61.75     -5.07   255
## 21     May            1707      medium             9    61.67     -4.65   400
## 1      May            1344         low             7    60.37     -4.18   520
## 4      May            1050      medium            10    60.29     -4.35   540
## 2      May            1633      medium            13    60.38     -4.19   559
## 76 October            1238      medium             5    61.30     -1.99   647
## 77 October             521         low             9    60.44     -4.26   687
## 75 October            1512         low            16    61.36     -2.00   735
##    gradient
## 12       23
## 23      133
## 22      111
## 21      226
## 1       415
## 4       409
## 2       405
## 76      288
## 77      421
## 75      400
```

2.5 Now for something a little more complicated. Sort all rows in the whale dataframe by ascending order of depth within each level of water noise. The trick here is to remember that you can order by more than one variable when using the `order()` function.

```r
whale.sorted <- whale[order(whale$water.noise, whale$depth), ]

head(whale.sorted, 10)
```

```
##      month time.at.station water.noise number.whales latitude longitude depth
## 6      May             580        high            10    60.38     -5.22  1000
## 48     May            1329        high            28    61.59     -3.06  1425
## 58 October             787        high             8    61.62     -3.00  1455
## 69 October             824        high             9    61.65     -2.96  1484
## 73 October            1164        high            19    61.78     -2.74  1549
## 59 October            1252        high            15    62.01     -2.00  1553
## 57 October            1096        high            10    62.07     -2.01  1584
## 65 October             901        high             8    61.87     -2.58  1587
## 56 October             792        high             1    62.04     -2.12  1592
## 46     May            1216        high            17    61.99     -2.26  1593
##    gradient
## 6       173
## 48      106
## 58      103
## 69       94
## 73       63
## 59      122
## 57      124
## 65       88
## 56      115
## 46      120
```

2.6 Repeat the previous ordering in Question 2.5 but this time order by descending order of depth within each level of water noise.

```r
whale.rev.sorted <- whale[order(whale$water.noise, -whale$depth), ]

head(whale.rev.sorted, 10)
```

```
##      month time.at.station water.noise number.whales latitude longitude depth
## 52 October            1137        high            12    62.10     -2.01  1610
## 47     May             905        high             7    61.94     -2.41  1601
## 53 October             948        high             9    61.91     -2.48  1600
## 54 October            1041        high             9    62.07     -2.06  1600
## 55 October            1454        high            25    62.01     -2.18  1596
## 46     May            1216        high            17    61.99     -2.26  1593
## 56 October             792        high             1    62.04     -2.12  1592
## 65 October             901        high             8    61.87     -2.58  1587
## 57 October            1096        high            10    62.07     -2.01  1584
## 59 October            1252        high            15    62.01     -2.00  1553
##    gradient
## 52      134
## 47      118
## 53      110
## 54      123
## 55      125
## 46      120
```

```
## 56      115
## 65       88
## 57      124
## 59      122
```

2.7 Knowing how many observations are present for each factor level is useful to determine whether you have an adequate sample size. Use the `table()` function to determine the number of observations for each level of water noise. Next use the same function to display the number of observations for each combination of water noise and month.

```r
table(whale$water.noise)
```

```
##
##   high    low medium
##     15     28     57
```

```r
table(whale$water.noise, whale$month)
```

```
##
##          May October
##   high     4      11
##   low     22       6
##   medium  24      33
```

Exercise 3

These data were originally collected as part of a study published in Aquatic Living Resources1 in 2005. The aim of the study was to investigate the seasonal patterns of investment in somatic and reproductive tissues in the long finned squid *Loligo forbesi* caught in Scottish waters. Squid were caught monthly from December 1989 - July 1991 (month and year variables). After capture, each squid was given a unique specimen code, weighed (weight) and the sex determined (sex - only female squid are included here). The size of individuals was also measured as the dorsal mantle length (DML) and the mantle weight measured without internal organs (eviscerate.weight). The gonads were weighed (ovary.weight) along with the accessory reproductive organ (the nidamental gland, nid.weight, nid.length). Each individual was also assigned a categorical measure of maturity (maturity.stage, ranging from 1 to 5 with 1 = immature, 5 = mature). The digestive gland weight (dig.weight) was also recorded to assess nutritional status of the individual.

3.1 Import the 'squid.csv' file into R using the `read.csv()` function and assign it to a variable named squid. Use the str() function to display the structure of the dataset and the `summary()` function to summarise the dataset. How many observations are in this dataset? How many variables? The year, month and maturity.stage variables were coded as integers in the original dataset. Here we would like to code them as factors. Create a new variable for each of these variables in the squid dataframe and recode them as factors. Use the `str()` function again to check the coding of these new variables.

```r
squid <- read.csv('Data/squid.csv', header =TRUE)

str(squid)
```

```
## 'data.frame':    519 obs. of  13 variables:
##  $ sample.no        : int  105128901 105128901 105128901 105128901 105128901 105128901 105128901 105
##  $ specimen         : int  1002 1003 1005 1007 1008 1009 1011 1013 1014 1017 ...
```

```
##  $ year              : int  1989 1989 1989 1989 1989 1989 1989 1989 1989 1989 ...
##  $ month             : int  12 12 12 12 12 12 12 12 12 12 ...
##  $ weight            : num  152 106 138 141 126 ...
##  $ sex               : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ maturity.stage    : int  3 1 2 2 3 1 2 3 3 4 ...
##  $ DML               : int  174 153 169 175 169 116 135 192 170 205 ...
##  $ eviscerate.weight: num  87.5 62.6 79.4 83.1 72.2 ...
##  $ dig.weight        : num  4.648 3.138 0.307 4.123 3.605 ...
##  $ nid.length        : num  39.4 24.1 39 41.4 39.8 20 14 55 44 53 ...
##  $ nid.weight        : num  2.46 0.319 1.169 1.631 2.03 ...
##  $ ovary.weight      : num  1.68 0.103 0.289 0.252 0.86 ...
```

```r
summary(squid)
```

```
##    sample.no           specimen         year          month       
##  Min.   :100039001   Min.   :1001   Min.   :1989   Min.   : 1.000  
##  1st Qu.:105079001   1st Qu.:1009   1st Qu.:1990   1st Qu.: 3.000  
##  Median :113099001   Median :1026   Median :1990   Median : 7.000  
##  Mean   :112499032   Mean   :1028   Mean   :1990   Mean   : 6.803  
##  3rd Qu.:121029101   3rd Qu.:1045   3rd Qu.:1991   3rd Qu.:10.000  
##  Max.   :130039001   Max.   :1076   Max.   :1991   Max.   :12.000  
##      weight           sex    maturity.stage      DML      eviscerate.weight
##  Min.   : 34.0   Min.   :2   Min.   :1.000   Min.   : 88   Min.   : 16.8    
##  1st Qu.:184.5   1st Qu.:2   1st Qu.:2.000   1st Qu.:187   1st Qu.: 97.0    
##  Median :272.0   Median :2   Median :3.000   Median :217   Median :138.0    
##  Mean   :286.8   Mean   :2   Mean   :3.355   Mean   :215   Mean   :149.4    
##  3rd Qu.:360.5   3rd Qu.:2   3rd Qu.:5.000   3rd Qu.:240   3rd Qu.:187.0    
##  Max.   :809.0   Max.   :2   Max.   :5.000   Max.   :323   Max.   :397.0    
##    dig.weight        nid.length       nid.weight      ovary.weight   
##  Min.   :  0.307   Min.   : 10.00   Min.   : 0.031   Min.   : 0.016  
##  1st Qu.:  4.705   1st Qu.: 34.00   1st Qu.: 0.863   1st Qu.: 0.429  
##  Median :  7.321   Median : 65.10   Median : 7.769   Median :10.461  
##  Mean   :  8.118   Mean   : 59.65   Mean   : 9.675   Mean   :12.564  
##  3rd Qu.: 10.028   3rd Qu.: 81.00   3rd Qu.:16.140   3rd Qu.:22.784  
##  Max.   :100.341   Max.   :430.20   Max.   :39.325   Max.   :50.230  
```

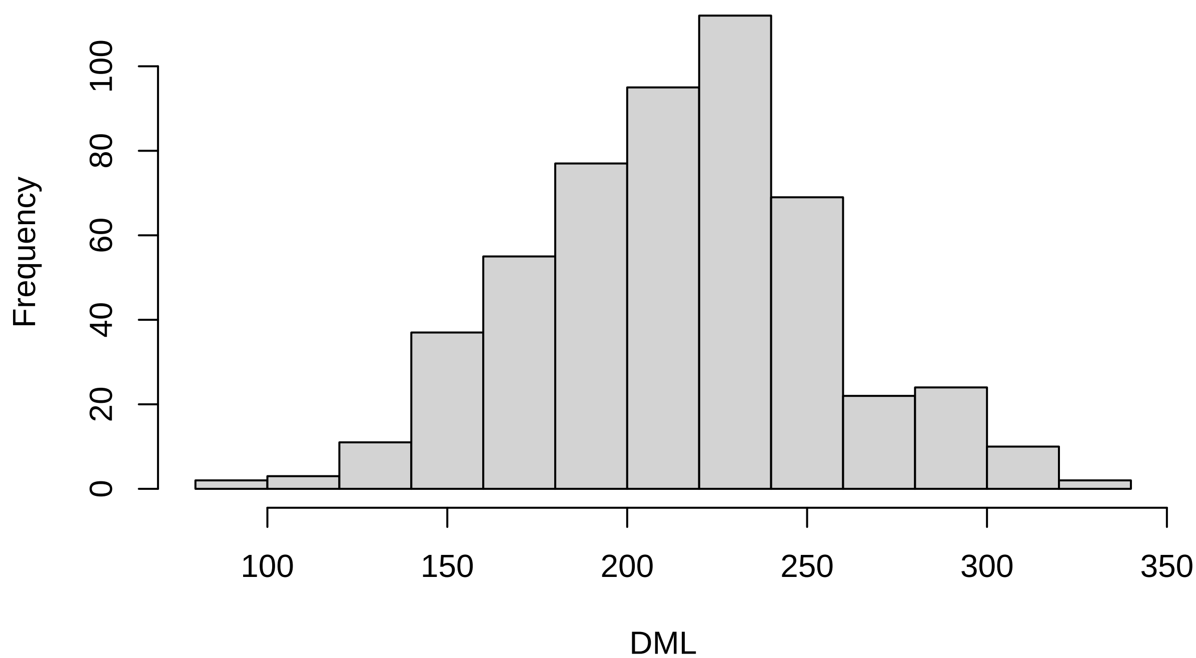
```r
# convert variables to factors
squid$Fmaturity <- factor(squid$maturity.stage)
squid$Fmonth <- factor(squid$month)
squid$Fyear <- factor(squid$year)

str(squid)
```

```
## 'data.frame':    519 obs. of  16 variables:
##  $ sample.no         : int  105128901 105128901 105128901 105128901 105128901 105128901 105128901 105
##  $ specimen          : int  1002 1003 1005 1007 1008 1009 1011 1013 1014 1017 ...
##  $ year              : int  1989 1989 1989 1989 1989 1989 1989 1989 1989 1989 ...
##  $ month             : int  12 12 12 12 12 12 12 12 12 12 ...
##  $ weight            : num  152 106 138 141 126 ...
##  $ sex               : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ maturity.stage    : int  3 1 2 2 3 1 2 3 3 4 ...
##  $ DML               : int  174 153 169 175 169 116 135 192 170 205 ...
##  $ eviscerate.weight: num  87.5 62.6 79.4 83.1 72.2 ...
```

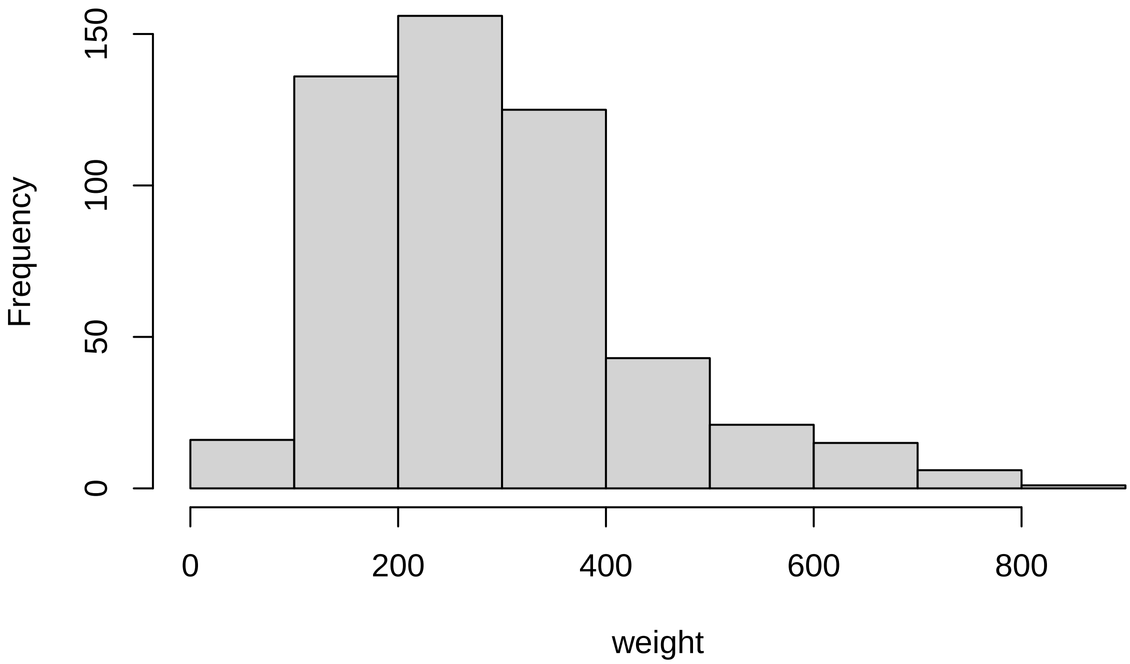
```
##  $ dig.weight       : num  4.648 3.138 0.307 4.123 3.605 ...
##  $ nid.length       : num  39.4 24.1 39 41.4 39.8 20 14 55 44 53 ...
##  $ nid.weight       : num  2.46 0.319 1.169 1.631 2.03 ...
##  $ ovary.weight     : num  1.68 0.103 0.289 0.252 0.86 ...
##  $ Fmaturity        : Factor w/ 5 levels "1","2","3","4",..: 3 1 2 2 3 1 2 3 3 4 ...
##  $ Fmonth           : Factor w/ 12 levels "1","2","3","4",..: 12 12 12 12 12 12 12 12 12 12 ...
##  $ Fyear            : Factor w/ 3 levels "1989","1990",..: 1 1 1 1 1 1 1 1 1 1 ...
```

3.2 When exploring your data it is often useful to visualize the distribution of continuous variables. Create histograms using `hist()` function for the variables; DML, weight, eviscerate.weight and ovary.weight.Then plot all the histograms in only one image using `par()` function.

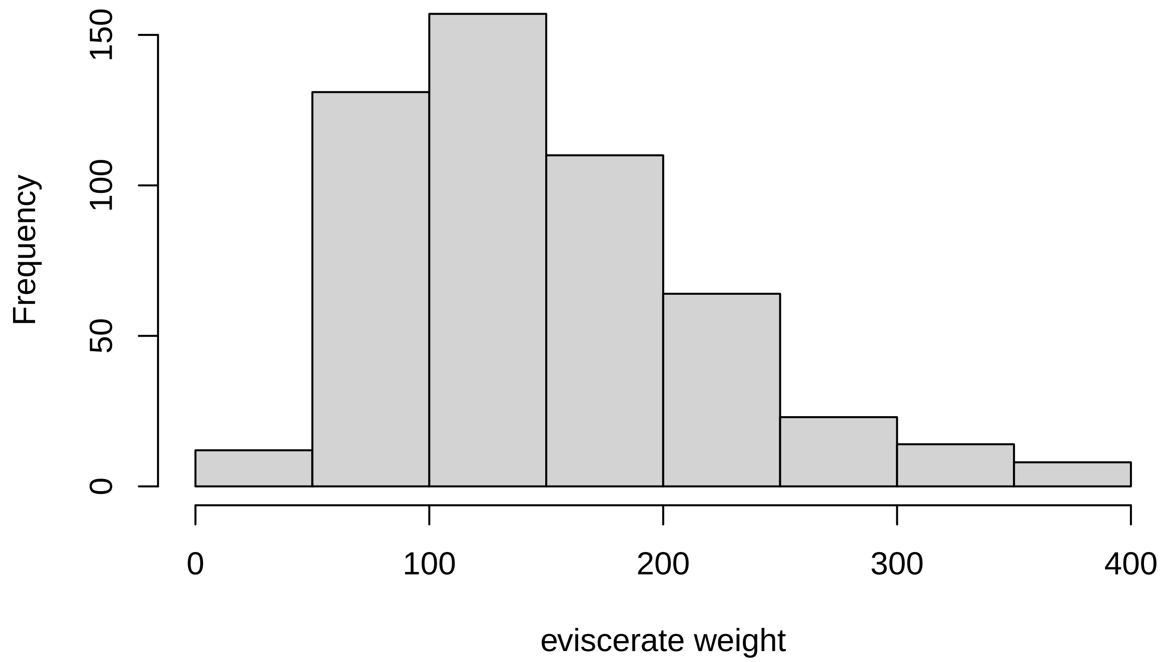
```r
# par(mfrow = c(2,2))
hist(squid$DML, main="", xlab = "DML")
```



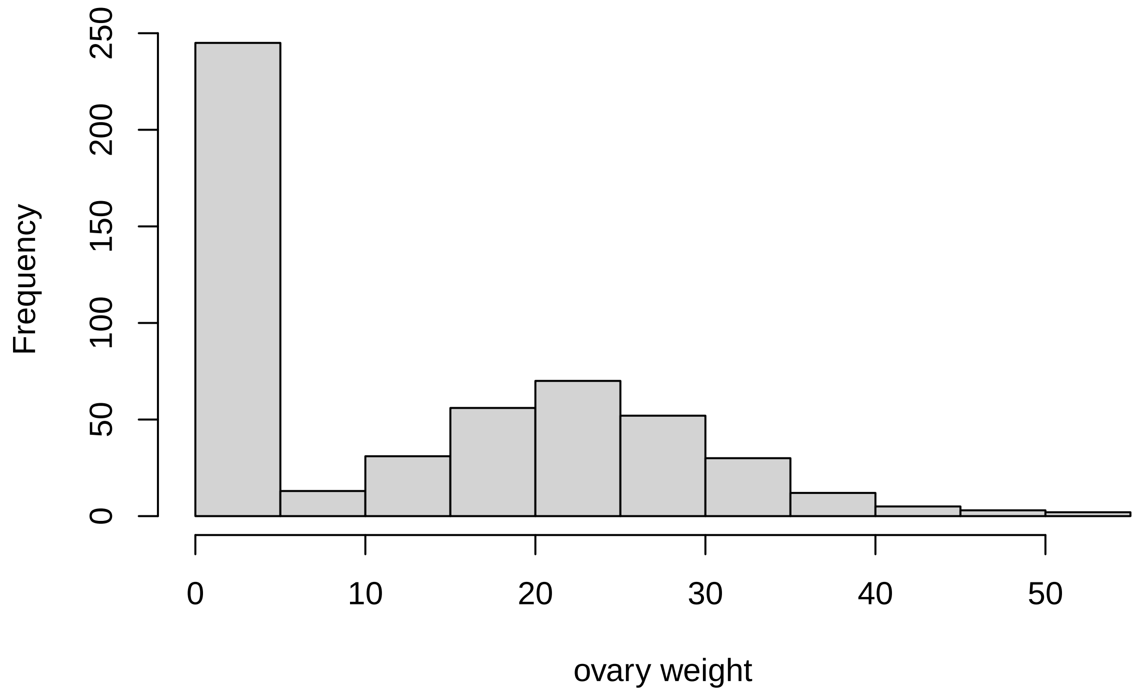

```r
hist(squid$weight, main="", xlab = "weight")
```




```r
hist(squid$eviscerate.weight, main="", xlab = "eviscerate weight")
```

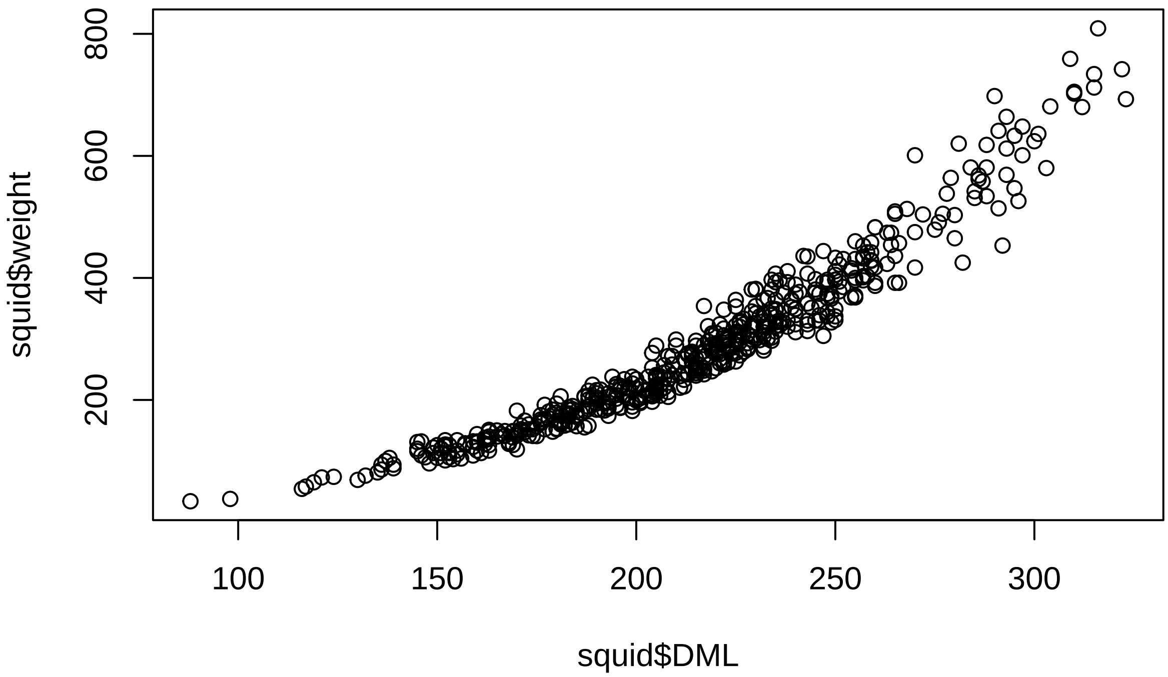



```r
hist(squid$ovary.weight, main="", xlab = "ovary weight")
```

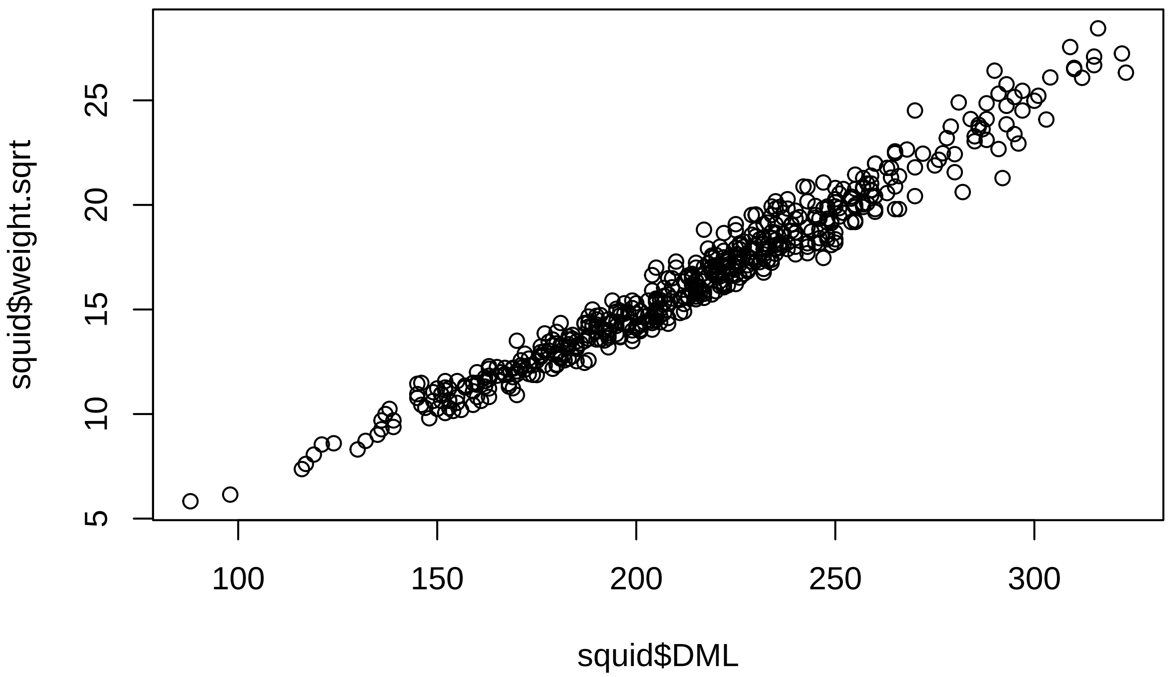


3.3 Scatterplots are great for visualising relationships between two continuous variables. Plot the relationship between DML on the x axis and weight on the y axis. How would you describe this relationship? Is it linear? One approach to linearising relationships is to apply a transformation on one or both variables. Try transforming the weight variable with either a natural log (log()) or square root (sqrt()) transformation. I suggest you create new variables in the squid dataframe for your transformed variables and use these variables when creating your new plots. Which transformation best linearises this relationship?

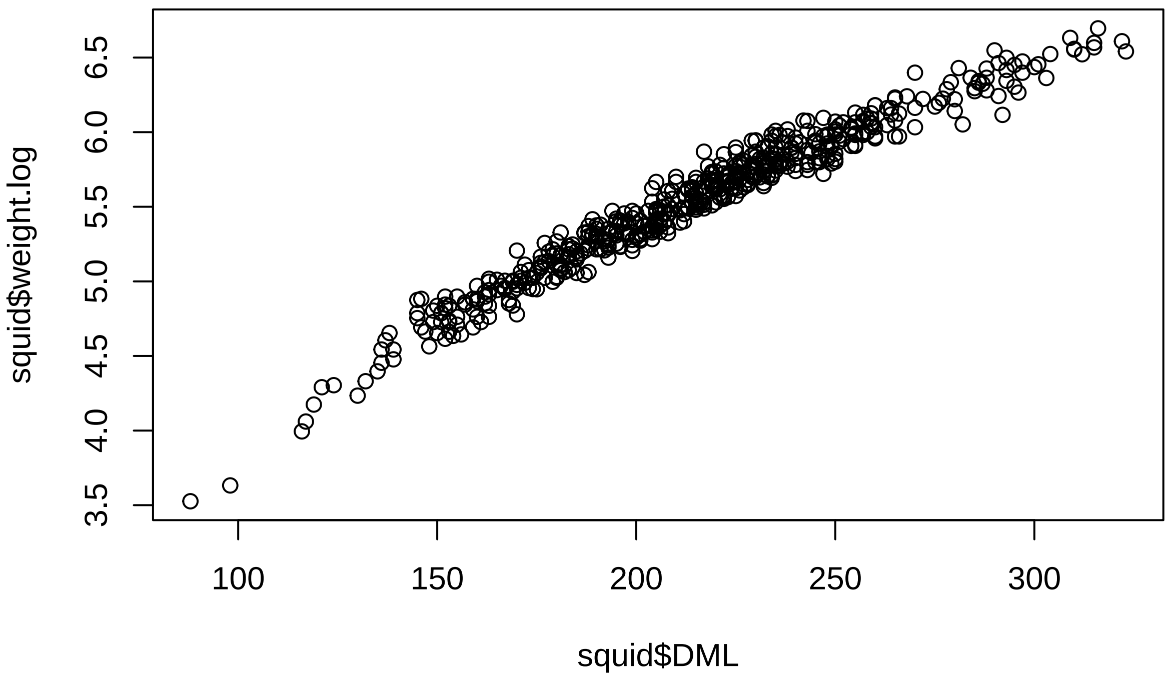
```
# clearly not linear
plot(squid$DML, squid$weight)
```




```
# natural log and sqrt tranform weight
squid$weight.sqrt <- sqrt(squid$weight)
squid$weight.log <- log(squid$weight)

plot(squid$DML, squid$weight.sqrt)
```

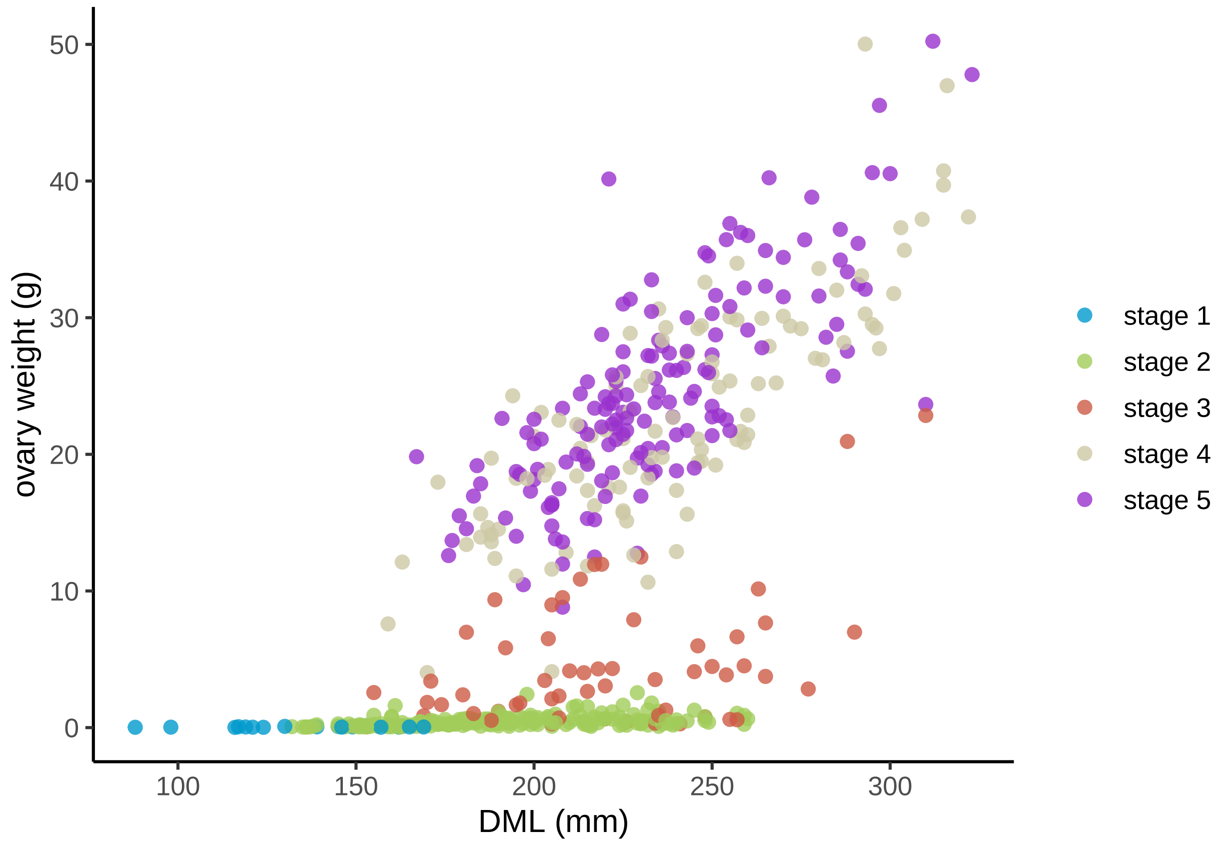



```r
plot(squid$DML, squid$weight.log)
```

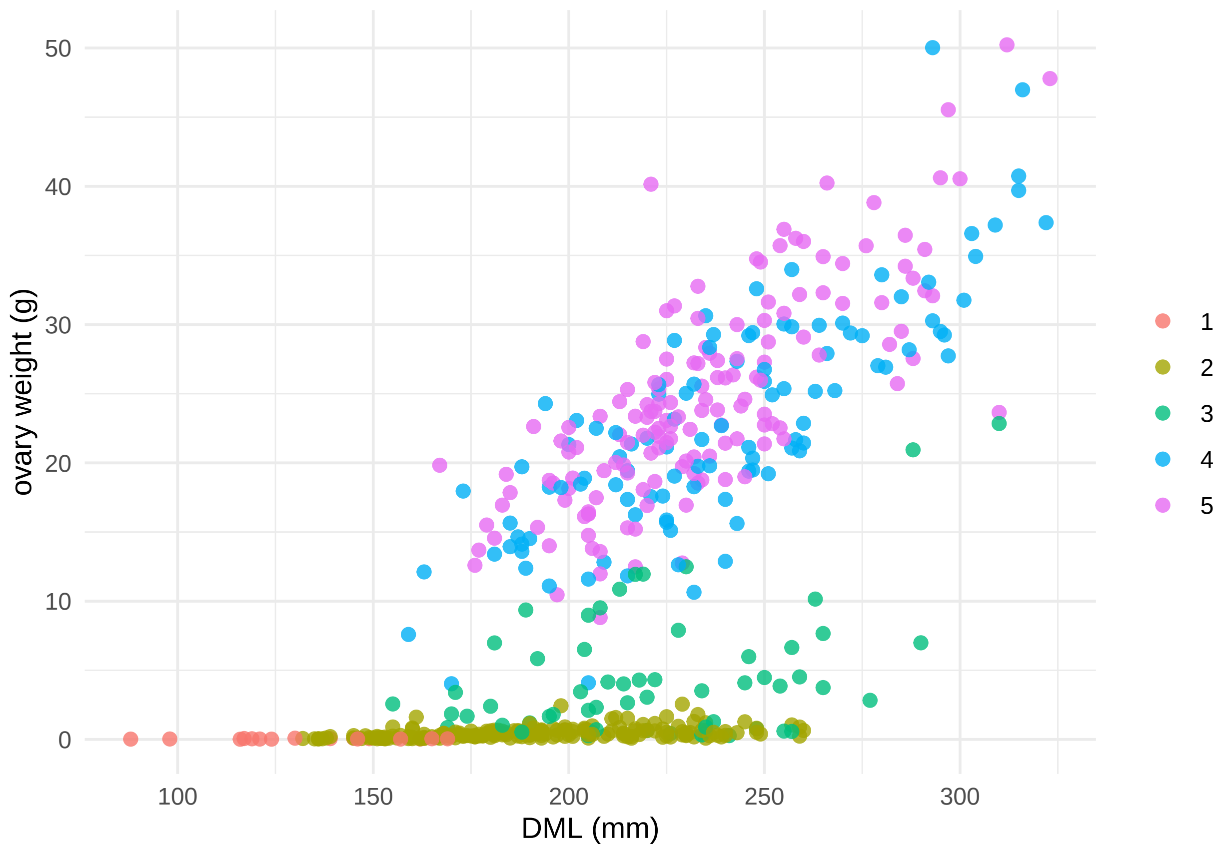


3.4 Almost every aspect of the figures you create in R is customisable. Use the ggplot() function to produce a scatterplot of DML on the x axis and ovary weight on the y axis. Use a different colour to highlight points for each level of maturity stage. Produce a legend explaining the different colours and place it in a suitable position on the plot and add axes labels.

```
library(ggplot2)
ggplot(data = squid) +
  geom_point(aes(x = DML, y = ovary.weight, colour = Fmaturity),
             alpha = 0.8, size = 2) +
  scale_colour_manual(values = c("deepskyblue3", "darkolivegreen3", "coral3",
                                 "lemonchiffon3", "darkorchid3"),
                      labels = c("stage 1", "stage 2", "stage 3",
                                 "stage 4", "stage 5")) +
 theme_classic(base_size = 12) +
  labs(colour = "", x = "DML (mm)", y = "ovary weight (g)")
```




```
# OR
ggplot(data = squid) +
  geom_point(aes(x = DML, y = ovary.weight, colour = Fmaturity),
             alpha = 0.8, size = 2) +
  theme_minimal() +
  labs(colour = "", x = "DML (mm)", y = "ovary weight (g)")
```



Exercise 4

4.1 Import the 'prawn' data into R and assign to a variable with a name. These data were collected from an experiment to investigate the difference in growth rate of the giant tiger prawn, *Penaeus monodon*, fed either an artificial or natural diet. Have a quick look at the structure of this dataset and plot a table of the growth rate versus the diet using an appropriate plot. How many observations are there in each diet treatment?

```
prawn <- read.csv('Data/prawn.csv', stringsAsFactors = T)

str(prawn)
```

```
## 'data.frame':    60 obs. of  2 variables:
##  $ GRate: num  9.77 10.29 10.05 10.08 9.31 ...
##  $ diet : Factor w/ 2 levels "Artificial","Natural": 2 2 2 2 2 2 2 2 2 2 ...
```

```
summary(prawn)
```

```
##      GRate                diet
##  Min.   : 7.395   Artificial:30
##  1st Qu.: 9.550   Natural   :30
##  Median : 9.943
##  Mean   : 9.920
```

```
##   3rd Qu.:10.344
##   Max.   :11.632
```

```r
table(prawn$diet)
```

```
##
## Artificial    Natural
##         30         30
```

4.2 You want to compare the difference in growth rate between the two diets using a two sample t-test. Before you conduct the test, you need to assess the normality and equal variance assumptions. Use the function `shapiro.test()` to assess normality of growth rate for each diet separately (Hint: use your indexing skills to extract the growth rate for each diet GRate[diet==‘Natural’] first) and plot a qq plot to assess the normality. Are your data normally distributed?.

```r
# test normality assumption

# Do not perform test on all data together, i.e.

shapiro.test(prawn$GRate) # this is wrong!!
```

```
##
##  Shapiro-Wilk normality test
##
## data:  prawn$GRate
## W = 0.97067, p-value = 0.1574
```

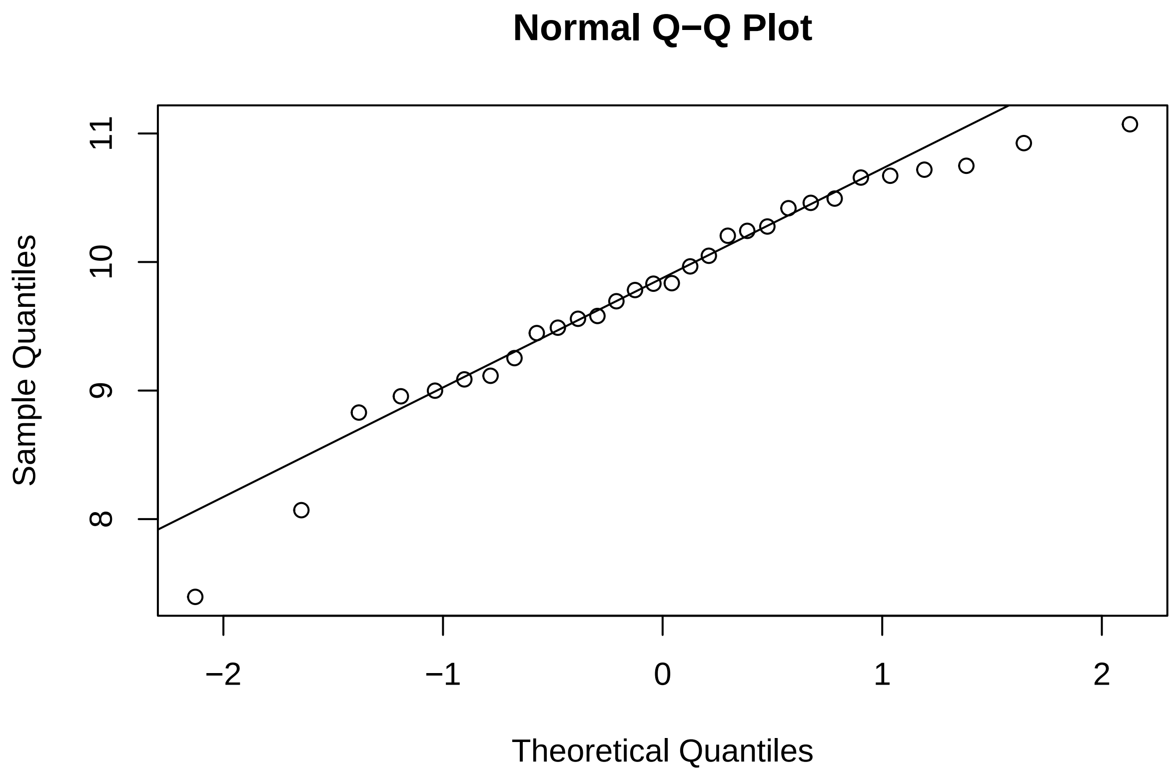
```r
# Need to test for departures from normality for each group

shapiro.test(prawn$GRate[prawn$diet == "Artificial"])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  prawn$GRate[prawn$diet == "Artificial"]
## W = 0.94863, p-value = 0.1553
```

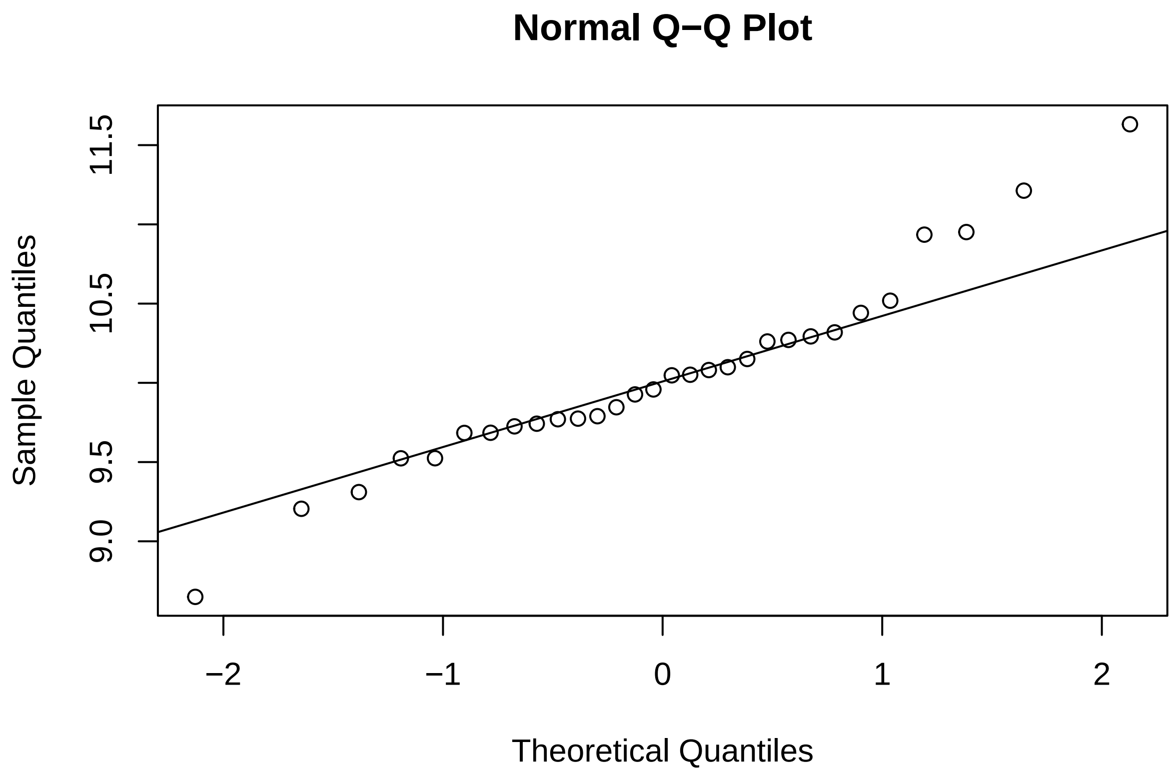
```r
shapiro.test(prawn$GRate[prawn$diet == "Natural"])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  prawn$GRate[prawn$diet == "Natural"]
## W = 0.95985, p-value = 0.307
```

```r
# Therefore no evidence to reject the Null hypothesis and data are normally distributed


# However much better assessment of normality is to use a quantile - quantile plot
# looking for points to lie along the line for normality

qqnorm(prawn$GRate[prawn$diet == "Artificial"])
qqline(prawn$GRate[prawn$diet == "Artificial"])
```




```r
qqnorm(prawn$GRate[prawn$diet == "Natural"])
qqline(prawn$GRate[prawn$diet == "Natural"])
```



4.3 Conduct a two sample t-test using the `t.test()` function. Use the argument var.equal = TRUE to perform the t-test assuming equal variances. What is the null hypothesis you want to test? Do you reject or fail to reject the null hypothesis? What is the value of the t statistic, degrees of freedom and p value? How would you summarise these summary statistics in a report?

```
# Conduct t-test assuming equal variances
# Null hypothesis Ho: no difference in growth rate between prawns fed on artificial diet or Natural die

t.test(GRate ~ diet, var.equal = TRUE, data = prawn)
```

```
##
##  Two Sample t-test
##
## data:  GRate by diet
## t = -1.3259, df = 58, p-value = 0.1901
## alternative hypothesis: true difference in means between group Artificial and group Natural is not eq
## 95 percent confidence interval:
##  -0.6319362  0.1283495
## sample estimates:
## mean in group Artificial    mean in group Natural
##                 9.794133                10.045927
```

```
# No evidence to reject the Null hypothesis, therefore no difference in growth rate of prawns fed on ei
```

4.4 Import the TemoraBR.csv dataset into R and as usual assign it a name. These data are from an experiment that was conducted to investigate the relationship between temperature (temp) and the beat rate (Hz) beat_rate of the copepod *Temora longicornis* which had been acclimatised at three different temperature regimes (acclimitisation_temp). Examine the structure of the dataset. How many variables are there? and what type of variables are they?

```
temora <- read.csv("Data/temora.csv", stringsAsFactors = T)

str(temora)
```

```
## 'data.frame':    45 obs. of  3 variables:
##  $ temp               : num  5 6 7 10 11 12 13 15 16 17 ...
##  $ beat_rate          : num  3.76 5.4 8 9.4 16.6 18.5 19 22.4 24.1 22.7 ...
##  $ acclimitisation_temp: int  5 5 5 5 5 5 5 5 5 5 ...
```

```
summary(temora)
```

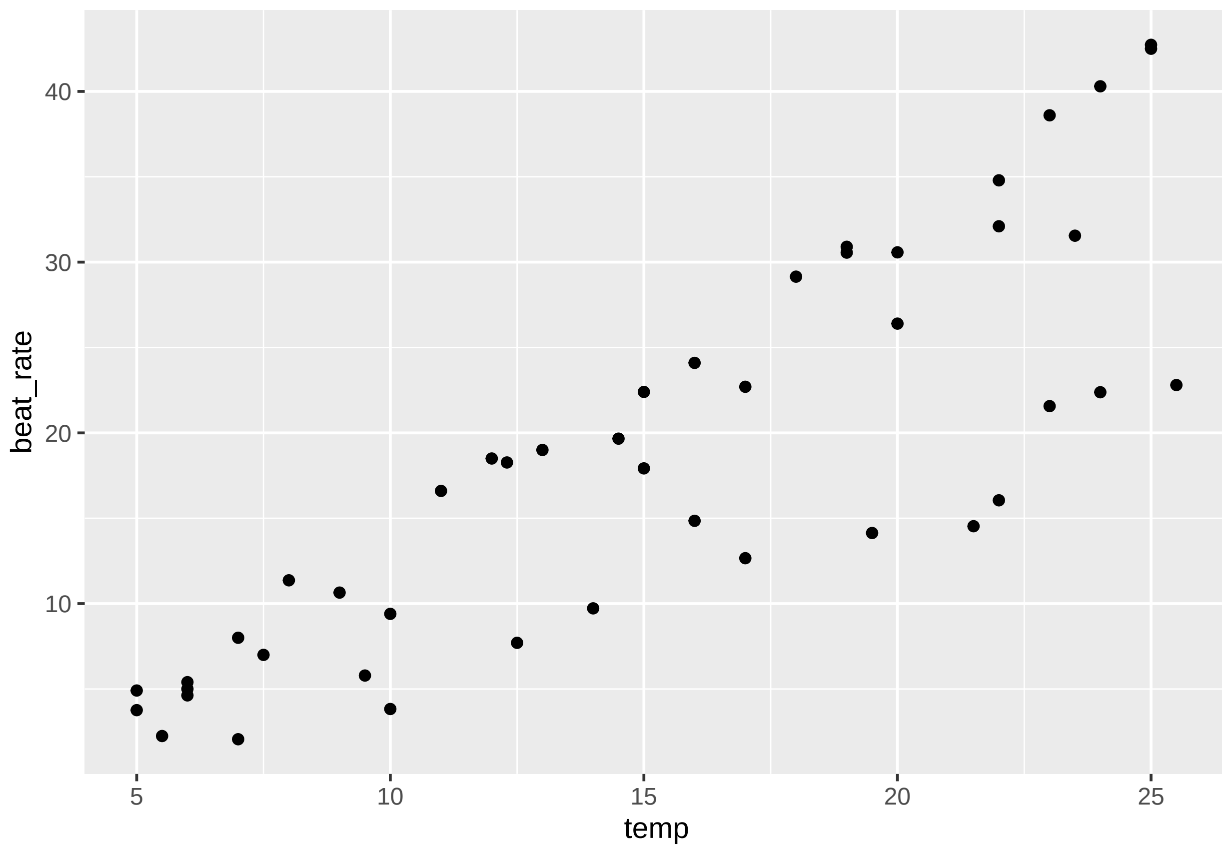
```
##       temp         beat_rate      acclimitisation_temp
##  Min.   : 5.00   Min.   : 2.058   Min.   : 5.00
##  1st Qu.: 9.50   1st Qu.: 8.000   1st Qu.: 5.00
##  Median :15.00   Median :17.922   Median :10.00
##  Mean   :15.17   Mean   :18.439   Mean   :11.67
##  3rd Qu.:21.50   3rd Qu.:26.400   3rd Qu.:20.00
##  Max.   :25.50   Max.   :42.728   Max.   :20.00
```

4.5 Now we want to explore the correlation between the temperature and beat rate using `cor()`function. What relationship do you see from these two variables? Can you visualize these two variables with `ggplot()` function?

```
library(ggplot2)

cor(temora$beat_rate, temora$temp)
```

```
## [1] 0.8476537
```

```
ggplot(mapping = aes(x = temp, y = beat_rate), data = temora) +
  geom_point()
```



4.6 Now, let's fit a simple linear model to these data we will use the `lm()` function and include our model formula beat_rate ~ temp and assign the results to an object called lm_temora. Can you predict how the beat rate will change when there is an increase in one unit of temperature?

```
lm_temora <- lm(beat_rate ~ temp, data = temora)

summary(lm_temora)
```

```
##
## Call:
## lm(formula = beat_rate ~ temp, data = temora)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.3953  -4.1401   0.7217   4.4213   9.5494
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -4.3219     2.3641  -1.828   0.0745 .
## temp          1.5000     0.1432  10.477 2.06e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.254 on 43 degrees of freedom
## Multiple R-squared:  0.7185, Adjusted R-squared:  0.712
## F-statistic: 109.8 on 1 and 43 DF,  p-value: 2.061e-13
```

```r
ggplot(mapping = aes(x = temp, y = beat_rate), data = temora) +
  geom_point() +
  geom_smooth(method = "lm", se = T)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```