Exercise 1

As pointed out in the presentation we can use `head()` and `tail()` for getting the first or last few rows of a data frame.

```
head(iris)

#   Sepal.Length Sepal.Width Petal.Length
# 1          5.1         3.5          1.4
# 2          4.9         3.0          1.4
# 3          4.7         3.2          1.3
# 4          4.6         3.1          1.5
# 5          5.0         3.6          1.4
# 6          5.4         3.9          1.7
#   Petal.Width Species
# 1         0.2  setosa
# 2         0.2  setosa
# 3         0.2  setosa
# 4         0.2  setosa
# 5         0.2  setosa
# 6         0.4  setosa

tail(iris)

#     Sepal.Length Sepal.Width Petal.Length
# 145          6.7         3.3          5.7
# 146          6.7         3.0          5.2
# 147          6.3         2.5          5.0
# 148          6.5         3.0          5.2
# 149          6.2         3.4          5.4
# 150          5.9         3.0          5.1
#     Petal.Width   Species
# 145         2.5 virginica
# 146         2.3 virginica
# 147         1.9 virginica
# 148         2.0 virginica
# 149         2.3 virginica
# 150         1.8 virginica
```

Using `summary()` gives us a simple summary for each of the columns.

```
summary(iris)

#   Sepal.Length    Sepal.Width     Petal.Length
#  Min.   :4.300   Min.   :2.000   Min.   :1.000
#  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600
#  Median :5.800   Median :3.000   Median :4.350
#  Mean   :5.843   Mean   :3.057   Mean   :3.758
#  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100
#  Max.   :7.900   Max.   :4.400   Max.   :6.900
#   Petal.Width          Species
#  Min.   :0.100   setosa    :50
#  1st Qu.:0.300   versicolor:50
#  Median :1.300   virginica :50
#  Mean   :1.199
#  3rd Qu.:1.800
#  Max.   :2.500
```

This gives summary statistics for all pooled observations. If we want to get statistics separately for each species, we have to split the data frame first. This yields a list of data frames.

```
split_iris <- split(iris, iris$Species)
str(split_iris)

# List of 3
#  $ setosa    :'data.frame': 50 obs. of  5 variables:
#   ..$ Sepal.Length: num [1:50] 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
#   ..$ Sepal.Width : num [1:50] 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
#   ..$ Petal.Length: num [1:50] 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
#   ..$ Petal.Width : num [1:50] 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
#   ..$ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
#  $ versicolor:'data.frame': 50 obs. of  5 variables:
#   ..$ Sepal.Length: num [1:50] 7 6.4 6.9 5.5 6.5 5.7 6.3 4.9 6.6 5.2 ...
#   ..$ Sepal.Width : num [1:50] 3.2 3.2 3.1 2.3 2.8 2.8 3.3 2.4 2.9 2.7 ...
#   ..$ Petal.Length: num [1:50] 4.7 4.5 4.9 4 4.6 4.5 4.7 3.3 4.6 3.9 ...
#   ..$ Petal.Width : num [1:50] 1.4 1.5 1.5 1.3 1.5 1.3 1.6 1 1.3 1.4 ...
#   ..$ Species     : Factor w/ 3 levels "setosa","versicolor",..: 2 2 2 2 2 2 2 2 2 2 ...
#  $ virginica :'data.frame': 50 obs. of  5 variables:
#   ..$ Sepal.Length: num [1:50] 6.3 5.8 7.1 6.3 6.5 7.6 4.9 7.3 6.7 7.2 ...
#   ..$ Sepal.Width : num [1:50] 3.3 2.7 3 2.9 3 3 2.5 2.9 2.5 3.6 ...
#   ..$ Petal.Length: num [1:50] 6 5.1 5.9 5.6 5.8 6.6 4.5 6.3 5.8 6.1 ...
#   ..$ Petal.Width : num [1:50] 2.5 1.9 2.1 1.8 2.2 2.1 1.7 1.8 1.8 2.5 ...
#   ..$ Species     : Factor w/ 3 levels "setosa","versicolor",..: 3 3 3 3 3 3 3 3 3 3 ...
```

We can now compute summary statistics for each of the elements of the lists. As the hint points out, we can use double brackets to extract one element from a list.

```
head(split_iris[[1]])

#   Sepal.Length Sepal.Width Petal.Length
# 1          5.1         3.5          1.4
# 2          4.9         3.0          1.4
# 3          4.7         3.2          1.3
# 4          4.6         3.1          1.5
# 5          5.0         3.6          1.4
# 6          5.4         3.9          1.7
#   Petal.Width Species
# 1         0.2  setosa
# 2         0.2  setosa
# 3         0.2  setosa
# 4         0.2  setosa
# 5         0.2  setosa
# 6         0.4  setosa

summary(split_iris[[1]])

#   Sepal.Length    Sepal.Width     Petal.Length
#  Min.   :4.300   Min.   :2.300   Min.   :1.000
#  1st Qu.:4.800   1st Qu.:3.200   1st Qu.:1.400
#  Median :5.000   Median :3.400   Median :1.500
#  Mean   :5.006   Mean   :3.428   Mean   :1.462
#  3rd Qu.:5.200   3rd Qu.:3.675   3rd Qu.:1.575
#  Max.   :5.800   Max.   :4.400   Max.   :1.900
#   Petal.Width          Species
#  Min.   :0.100   setosa    :50
#  1st Qu.:0.200   versicolor: 0
#  Median :0.200   virginica : 0
#  Mean   :0.246
#  3rd Qu.:0.300
#  Max.   :0.600
```

We can do this for each of the elements in the list. I will skip the other two species. Interestingly, R still knows the other two species that are not present in subset (see at the summary for the `Species` column). Alternatively, we can use `subset` for computing the summary statistics for each species separately.

```
summary(subset(iris, Species == 'setosa'))

#   Sepal.Length    Sepal.Width     Petal.Length
#  Min.   :4.300   Min.   :2.300   Min.   :1.000
#  1st Qu.:4.800   1st Qu.:3.200   1st Qu.:1.400
#  Median :5.000   Median :3.400   Median :1.500
#  Mean   :5.006   Mean   :3.428   Mean   :1.462
#  3rd Qu.:5.200   3rd Qu.:3.675   3rd Qu.:1.575
#  Max.   :5.800   Max.   :4.400   Max.   :1.900
#   Petal.Width          Species
#  Min.   :0.100   setosa    :50
#  1st Qu.:0.200   versicolor: 0
#  Median :0.200   virginica : 0
#  Mean   :0.246
#  3rd Qu.:0.300
#  Max.   :0.600
```

However, the latter way becomes tedious if the column by which we want to split the data frame has many categories. Splitting the data frame with `split` into a list of data frames has the advantage that the subsets are available as elements of a single list, which make them programatically accessible. For instance, we will see in a few weaks, how we can use functional programming to apply the `summary` function to each element of a list automatically. As a foretaste:

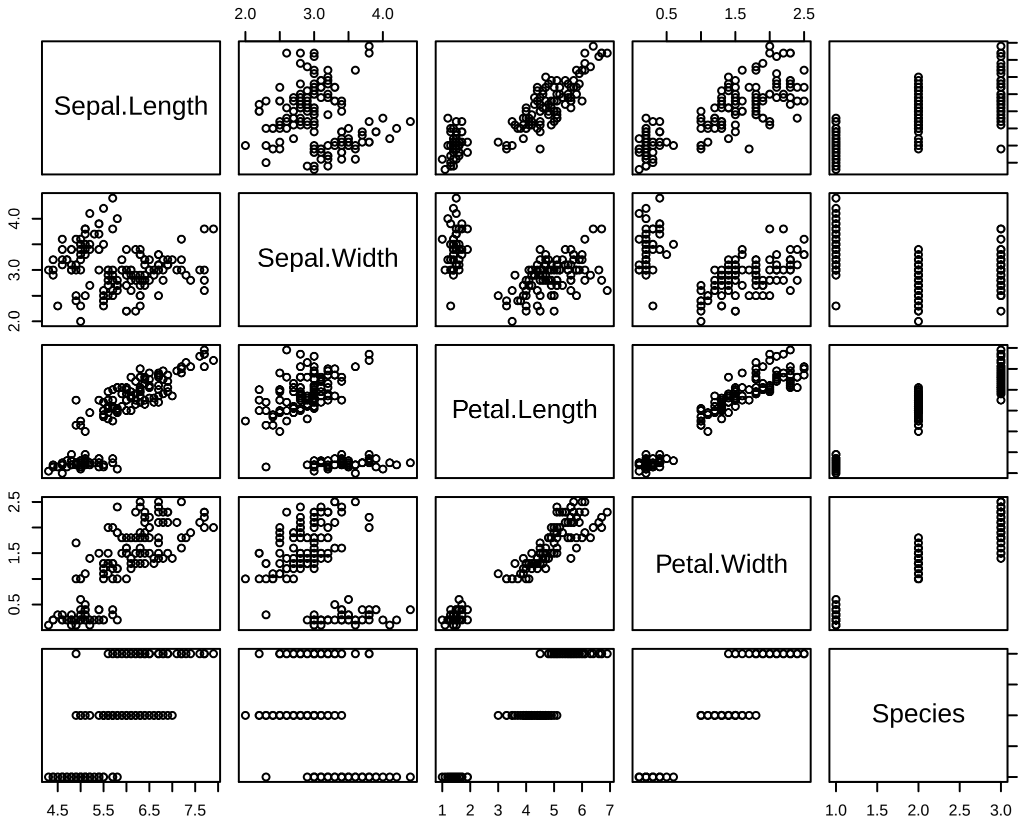
```
lapply(split_iris, summary)

# $setosa
#   Sepal.Length    Sepal.Width     Petal.Length
#  Min.   :4.300   Min.   :2.300   Min.   :1.000
#  1st Qu.:4.800   1st Qu.:3.200   1st Qu.:1.400
#  Median :5.000   Median :3.400   Median :1.500
#  Mean   :5.006   Mean   :3.428   Mean   :1.462
#  3rd Qu.:5.200   3rd Qu.:3.675   3rd Qu.:1.575
#  Max.   :5.800   Max.   :4.400   Max.   :1.900
#   Petal.Width          Species
#  Min.   :0.100   setosa    :50
#  1st Qu.:0.200   versicolor: 0
#  Median :0.200   virginica : 0
#  Mean   :0.246
#  3rd Qu.:0.300
#  Max.   :0.600
#
```

```
# $versicolor
#   Sepal.Length    Sepal.Width     Petal.Length
#  Min.   :4.900   Min.   :2.000   Min.   :3.00
#  1st Qu.:5.600   1st Qu.:2.525   1st Qu.:4.00
#  Median :5.900   Median :2.800   Median :4.35
#  Mean   :5.936   Mean   :2.770   Mean   :4.26
#  3rd Qu.:6.300   3rd Qu.:3.000   3rd Qu.:4.60
#  Max.   :7.000   Max.   :3.400   Max.   :5.10
#   Petal.Width          Species
#  Min.   :1.000   setosa    : 0
#  1st Qu.:1.200   versicolor:50
#  Median :1.300   virginica : 0
#  Mean   :1.326
#  3rd Qu.:1.500
#  Max.   :1.800
#
# $virginica
#   Sepal.Length    Sepal.Width     Petal.Length
#  Min.   :4.900   Min.   :2.200   Min.   :4.500
#  1st Qu.:6.225   1st Qu.:2.800   1st Qu.:5.100
#  Median :6.500   Median :3.000   Median :5.550
#  Mean   :6.588   Mean   :2.974   Mean   :5.552
#  3rd Qu.:6.900   3rd Qu.:3.175   3rd Qu.:5.875
#  Max.   :7.900   Max.   :3.800   Max.   :6.900
#   Petal.Width          Species
#  Min.   :1.400   setosa    : 0
#  1st Qu.:1.800   versicolor: 0
#  Median :2.000   virginica :50
#  Mean   :2.026
#  3rd Qu.:2.300
#  Max.   :2.500
```

What happens if we try to plot a data frame?

```
plot(iris)
```



This creates plots for each column combination. This can be very useful for a first exploratory analysis of the data set. For example, the width and length of petals are positively correlated while this is not so pronounced for sepals. Also the species differ perceptibly from each other (the plots on the bottom and right).

Exercise 2

If we measure several tricellular vertices in one embryo then the observational unit is not any longer the embryo but a single vertex. In that case we have to measure additional properties to make sense out of the data, e.g. the embryo (we can simply number them) and the tissue if we measure in different tissues. So a good data frame would look like the one below.
So what was wrong with the data frames in the exercise? The first one was complete nonsense. Of course we can say in advance that we want to measure in three different embryos for each combination of stage and genotype. But this does not result in three properties for one observation. Actually, the way the data was noted down in the first example suggests that 1.24, 0.23, and 0.98 are all values measured in the very same embryo. This is of course impossible.

Embryo	Stage	Genotype	Intensity enrichment
1	14	y w	1.24
1	14	y w	0.23
2	14	M6[W186*]	0.98
⋮	⋮	⋮	⋮

The second table isn't any better. Several values are stored in one cell. There is not even a possibility to store a structure like this in a data frame in R (except using a string of course...). The third table is a bit better except that two properties are stored in one column. This is not very handy since now we cannot split the data by one of the properties separately. At least we can fix this problem easily by splitting the column by the slash (we will learn soon how).