R exercise 1: Introduction

The pairs plot for the iris data

To visualise the relationship between variables, we can use `pairs( )` to create a pairs plot. When the response variable is categorical, we can label data points from different classes using different symbols. Which variables are good for classification task?

We take the iris data as an example. Iris is contained in `R`, so we do not have to load it from files. Use the following command to see all the datasets in `R`.

```
data()
```

To see the summary of the iris data:

```
help(iris) # description of the iris data
str(iris) # displays the structure of the iris data
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  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 ...
```

```
table(iris$Species)  # gives you a table of the species variable
```

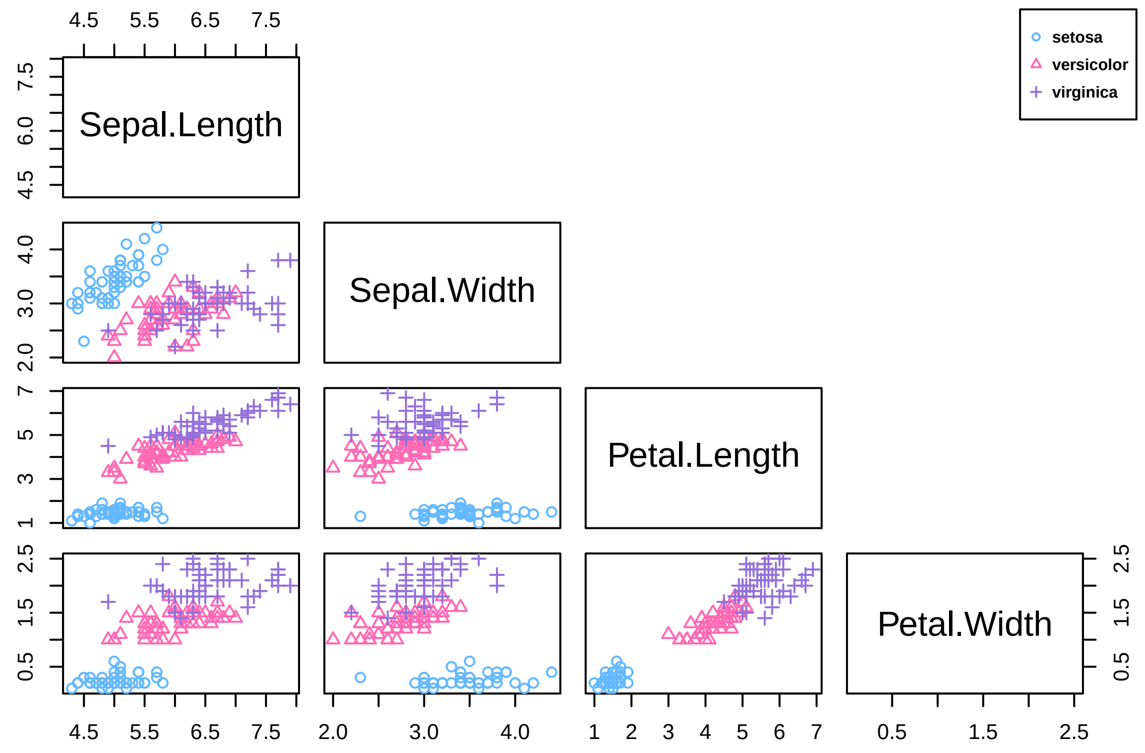
```
## 
##     setosa versicolor  virginica 
##         50         50         50
```

```
summary(iris) # five-number summary statistics of the variables
```

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

Now we can draw a pairs plot for the iris data with different species labelled wit different symbols.

```
# set the colours for the three classes
cols<- c("steelblue1", "hotpink", "mediumpurple")
# set the symbols for the three classes
pchs<-c(1,2,3)
# use pairs() to create the pairs plot, with different colours and symbols
# for different classes
pairs(iris[,1:4], pch = pchs[iris$Species],  cex = 1,
      col = cols[iris$Species],
      upper.panel=NULL)
# create a legend for the plot
par(xpd = TRUE) # enables you to add a legend outside of the plot region
legend("topright",legend=c("setosa","versicolor","virginica"),
       col=cols,pch=pchs,cex=0.5,text.font=2)
```



Use `?function_name` to understand all the commands. Change the values of `cex` and `text.font` in `legend()`, to make the plot fit to your window.

We can save the plot by clicking `Export`.

Alternatively, we can do the following:

```
# Open a new plot device
dev.new(width=5, height=4, unit="in")
# use pairs() to create the pairs plot, with different colours and symbols
# for different classes
```

```r
pairs(iris[,1:4], pch = pchs[iris$Species],  cex = 1,
      col = cols[iris$Species],
      lower.panel=NULL)
# create a legend for the plot
par(xpd = TRUE) # enables you to add a legend outside of the plot region
legend("bottomleft",legend=c("setosa","versicolor","virginica"),
       col=cols,pch=pchs,cex=1,text.font=4)
```

`dev.new()` will open a new plot device in `R`. We can change the size of the plot in the new window and save it manually.

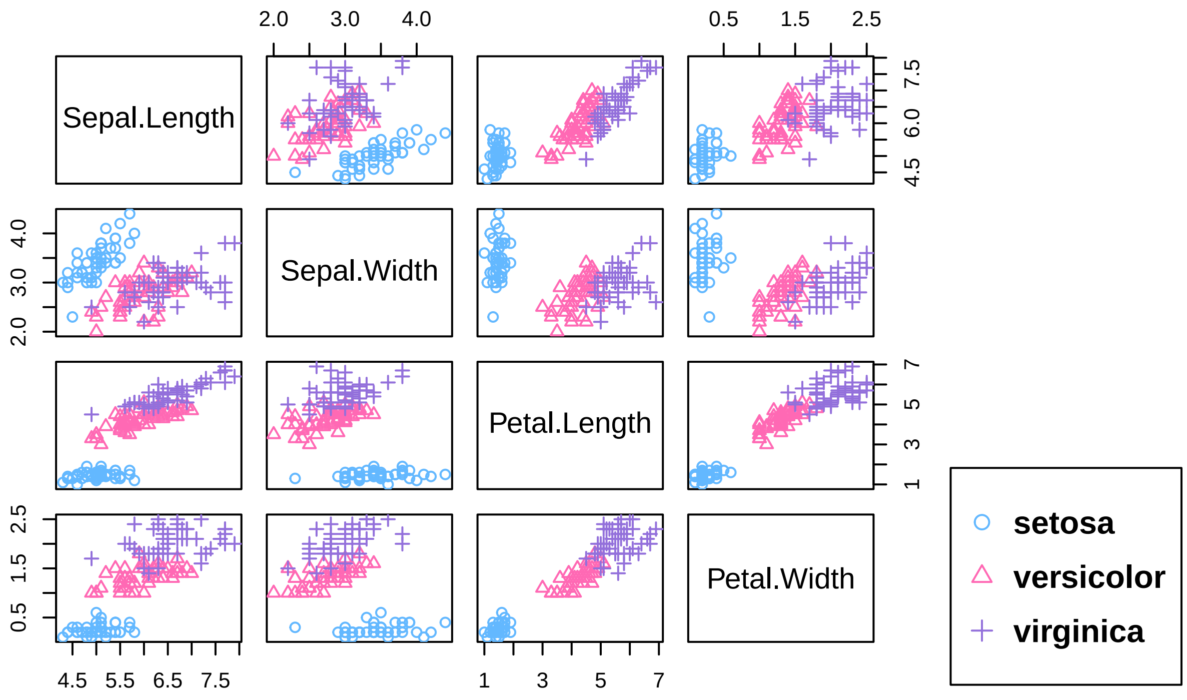
Another way to save a plot in pdf format automatically:

```r
# use pdf() to open a new pdf device
pdf("iris-plot.pdf",width=7, height=5)
# use pairs() to create the pairs plot, with different colours and symbols
# for different classes
pairs(iris[,1:4], pch = pchs[iris$Species],  cex = 1,
      col = cols[iris$Species],
      lower.panel=NULL)
# create a legend for the plot
par(xpd = TRUE) # enables you to add a legend outside of the plot region
legend("bottomleft",legend=c("setosa","versicolor","virginica"),
       col=cols,pch=pchs,cex=1,text.font=4)
# close the device
dev.off()
```

Play with different values for `pch`, `col`, `cex`, `text.font` etc. to see the changes of the plot.

Here is an alternative way to add a legend. Type `?par` and find what `oma` controls.

```r
# oma allows you to create some spare space for the legend
#  here we make the right margin quite large
pairs(iris[,1:4], pch = pchs[iris$Species],  cex = 1,
      col = cols[iris$Species],oma=c(4,4,6,12))
# create a legend for the plot
par(xpd = TRUE) # enables you to add a legend outside of the plot region
legend(0.85, 0.3,legend=c("setosa","versicolor","virginica"),
       col=cols,pch=pchs,cex=1,text.font=2)
```



There are many ways to do the same thing in R. Explore different ways and choose the way you like.

Create subsets from data

This is very important in classification tasks, because we need to create training set and test set.

Load the Default data:

```
Default=read.table("Default.txt",header=TRUE)
str(Default) # gives you the structure of the Default data
```

```
## 'data.frame':    10000 obs. of  4 variables:
##  $ default: chr  "No" "No" "No" "No" ...
##  $ student: chr  "No" "Yes" "No" "No" ...
##  $ balance: num  730 817 1074 529 786 ...
##  $ income : num  44362 12106 31767 35704 38463 ...
```

```
table(Default$default)  # gives you a table of the default variable
```

```
##
##   No  Yes
## 9667  333
```

```
summary(Default) # five-number summary statistics of the variables
```

```
##    default            student              balance           income     
##  Length:10000       Length:10000       Min.   :   0.0   Min.   :  772  
##  Class :character   Class :character   1st Qu.: 481.7   1st Qu.:21340  
##  Mode  :character   Mode  :character   Median : 823.6   Median :34553  
##                                        Mean   : 835.4   Mean   :33517  
##                                        3rd Qu.:1166.3   3rd Qu.:43808  
##                                        Max.   :2654.3   Max.   :73554
```

This dataset is large with 10,000 observations and 4 variables. The first variable `default` indicates whether the customer is default or not, the second variable `student` indicates whether the customer is a student or not, the third variable `balance` is the balance remaining in one customer's credit card after monthly payment and the fourth variable `income` is the customer's income.

Suppose we want to create a subsets of the Default data with 100 observations (50 of them are `default=No` and 50 of them are `default=Yes`):

```
# get the first 50 default=No
default_no_all=Default[Default$default=="No",]
default_no_50_1=default_no_all[1:50,]
# alternatively, try
default_no_50_2=Default[Default$default=="No",][1:50,]
# check whether default_no_50_1 is equivalent to
# default_no_50_2 by yourself
sum(default_no_50_1[,3]!=default_no_50_2[,3])
```

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

```
sum(default_no_50_1[,3]==default_no_50_2[,3])
```

```
## [1] 50
```

```
default_no_50 = default_no_50_2
# get the first 50 default=Yes
default_yes_50 = Default[Default$default=="Yes",][1:50,]
# combine default_no_50 and default_yes_50
# to get the subset
default_50 = rbind(default_no_50,default_yes_50)
?rbind
```

Instead of using the first 50 observations of `default=No` and `default=Yes`, we would like to randomly sample 50 observations from `default=No` and 50 observations from `default=Yes`:

```
?set.seed
set.seed(375) # make the random sampling reproducible
default_no_rand=Default[sample(which(Default$default=="No"),50),]
default_yes_rand=Default[sample(which(Default$default=="Yes"),50),]
?which
which(Default$default=="Yes")
```

```
##   [1]  137  174  202  207  210  242  244  264  342  346  350  358  407  440  441
##  [16]  488  541  546  577  582  642  652  714  741  762  804  834  868  921  933
##  [31]  975  982 1000 1019 1024 1044 1099 1137 1143 1144 1161 1210 1216 1256 1360
##  [46] 1362 1396 1446 1448 1485 1488 1497 1503 1549 1591 1610 1626 1648 1703 1710
##  [61] 1749 1781 1829 1833 1845 1863 1877 1897 1940 1942 2003 2011 2034 2050 2065
##  [76] 2092 2097 2127 2160 2174 2181 2202 2206 2218 2241 2284 2324 2341 2355 2449
##  [91] 2461 2539 2551 2605 2654 2695 2753 2785 2801 2890 2930 3041 3054 3104 3118
## [106] 3124 3157 3163 3182 3190 3212 3239 3249 3286 3298 3307 3315 3324 3377 3380
## [121] 3386 3392 3438 3487 3491 3563 3598 3613 3642 3715 3719 3785 3856 3882 3914
## [136] 3922 3958 3970 4050 4061 4074 4078 4080 4112 4145 4160 4168 4211 4225 4232
## [151] 4309 4372 4409 4418 4431 4464 4478 4510 4514 4569 4590 4591 4595 4682 4684
## [166] 4698 4709 4710 4727 4775 4938 4949 4953 4965 4983 5015 5039 5050 5054 5092
## [181] 5162 5190 5210 5286 5355 5397 5432 5440 5462 5507 5575 5653 5666 5675 5783
## [196] 5821 5858 5891 5982 6020 6033 6076 6078 6098 6122 6138 6181 6192 6243 6250
## [211] 6262 6334 6335 6372 6400 6422 6450 6462 6475 6521 6542 6544 6597 6602 6613
## [226] 6635 6644 6687 6756 6783 6795 6804 6831 6848 6851 6873 6883 6924 6970 6986
## [241] 7015 7047 7094 7105 7141 7200 7247 7336 7338 7363 7429 7438 7487 7490 7525
## [256] 7553 7662 7732 7785 7799 7812 7814 7816 7828 7870 7904 8121 8141 8191 8195
## [271] 8244 8265 8310 8343 8357 8365 8400 8411 8428 8457 8460 8463 8464 8480 8490
## [286] 8496 8543 8575 8706 8725 8738 8793 8833 8836 8908 8920 8993 9041 9046 9071
## [301] 9080 9085 9180 9257 9272 9296 9329 9370 9433 9438 9449 9459 9478 9486 9501
## [316] 9511 9516 9523 9539 9611 9655 9784 9787 9797 9814 9857 9884 9894 9913 9922
## [331] 9950 9952 9979
```

```r
?sample
default_rand = rbind(default_no_rand,default_yes_rand)
```

Usually we want to reproduce the random sampling results, so we have to add `set.seed( )` before any random sampling function. The number in the bracket can be any integer. Next time if you want to obtain the exactly the same random sampling result, use the same integer.

Sometimes we want to change the sample size: change the two 50s in the above codes. Imagine there are a lot of 50s in long codes. It is tedious to do this and easy to miss changing some values. It is easier to modify your codes if you write like this.

```r
n=50
set.seed(375) # make the random sampling reproducible
default_no_rand=Default[sample(which(Default$default=="No"),n),]
default_yes_rand=Default[sample(which(Default$default=="Yes"),n),]
default_rand = rbind(default_no_rand,default_yes_rand)
```

You just have to change the value of `n` in the first line.

Play with `sample( )` and `which( )` with different inputs, e.g. `which(Default$balance>1000)`…