Hierarchical Clustering exercises (beginner)

Grouping objects into clusters is a frequent task in data analysis. In this set of exercises we will use **hierarchical clustering** to cluster European capitals based on their latitude and longitude. Before trying out this exercise please make sure that you are familiar with the following functions: dist, hlcust, cutree, rect.hclust

We will be using a custom-made dataset. Before starting the exercise please run the following code to obtain the capital locations for Europe (note that you will need to have ggmap library installed):

```
library(ggmap)
capitals <- c("Albania, Tirana", "Andorra, Andorra la Vella", "Armenia, Yerevan", "Austria, Vienna", "Azerbaijan, Baku", "Belarus, Minsk", "Belgium, Brussels", "Bosnia and Herzegovina, Sarajevo", "Bulgaria, Sofia", "Croatia, Zagreb", "Cyprus, Nicosia", "Czech Republic, Prague", "Denmark, Copenhagen", "Estonia, Tallinn", "Finland, Helsinki", "France, Paris", "Germany, Berlin", "Greece, Athens", "Georgia, Tbilisi", "Hungary, Budapest", "Iceland, Reykjavik", "Italy, Rome", "Latvia, Riga", "Kazakhstan, Astana", "Liechtenstein, Vaduz", "Lithuania, Vilnius", "Luxembourg, Luxembourg", "Macedonia, Skopje", "Malta, Valletta", "Moldova, Chişinău", "Monaco, Monaco-Ville", "Montenegro, Podgorica", "Netherlands, Amsterdam", "Norway, Oslo", "Poland, Warsaw", "Portugal, Lisbon", "Republic of Ireland, Dublin", "Romania, Bucharest", "Russia, Moscow", "San Marino, San Marino", "Serbia, Belgrade", "Slovakia, Bratislava", "Slovenia, Ljubljana", "Spain, Madrid", "Sweden, Stockholm", "Switzerland, Bern",
```

```
"Turkey, Ankara", "Ukraine, Kiev", "United Kingdom, London",
"Vatican City, Vatican City")
theData <- geocode(capitals)
rownames(theData) <- capitals
```

Answers to the exercises are available here.

If you obtained a different (correct) answer than those listed on the solutions page, please feel free to post your answer as a comment on that page.

Exercise 1
Calculate the Euclidean latitude/longitude distances between all pairs of capital cities.

Exercise 2
Use the obtained distances to produce the hierarchical clustering dendrogram object. Use all the default parameters. NOTE: By default the clusters will be merged together using the maximum possible distance between all pairs of their elements (this fact will be useful later).

Exercise 3
Visualize the obtained hierarchical clustering dendrogram.

Exercise 4
In the previous step the leaves of our dendrogram were placed at different heights. Let's redo the plot so that all capital names are written at the same level.

Exercise 5
Hierarchical clustering procedure builds a hierarchy of clusters. One advantage of this method is that we can use the same dendrogram to obtain different numbers of groups.
Cluster the European capitals into 3 groups.

Exercise 6
Instead of specifying the wanted number of groups we can select the dendrogram height where the tree will be partitioned into groups. Since we used the maximum linkage function (default in exercise 2) this height has a useful interpretation – it ensures that all elements within one cluster are not more than the selected distance apart.
a) Cluster the European capitals by cutting the tree at height=20.
b) Plot the dendrogram and visualize the height at which the tree was cut into groups using a line.

Exercise 7
Now visualize the clustering solution obtained in the 5th exercise on the dendrogram plot. This should be done by drawing a rectangle around all capitals that fall in the same group. Use different colors for different groups.

Exercise 8
Visualize the dendrogram again but this time present both cluster versions obtained in exercise 5 and exercise 6 on the same plot. Use red color to represent exercise 5 clusters and blue to represent clusters from exercise 6.

Exercise 9
The hclust function has 8 implemented different linkage methods – methods used to merge two clusters when building the dendrogram. We want to experiment with all of them.
Produce a dendrogram, obtain 5 groups and vizualize them using different color rectangles. Repeat this for all available linkage methods.

Exercise 10
Design your own clustering solution based on what you learned in this exercise and visualize it as a map.
Plot capital coordinates with longitude on the x-axis and latitude on the y-axis and color them based on the groups obtained using your hierarchical clustering version.

Hierarchical Clustering solutions (beginner)

Below are the solutions to these exercises on hierarchical clustering.

Learn more about clustering in the online courses Applied Multivariate Analysis with R and Foundations of strategic business analytics

```
# Prepare dataset
library(ggmap)
capitals <- c("Albania, Tirana", "Andorra, Andorra la Vella",
"Armenia, Yerevan",
              "Austria, Vienna", "Azerbaijan, Baku", "Belarus,
Minsk",
               "Belgium, Brussels", "Bosnia and Herzegovina,
Sarajevo",
               "Bulgaria, Sofia", "Croatia, Zagreb", "Cyprus,
Nicosia",
              "Czech Republic, Prague", "Denmark, Copenhagen",
"Estonia, Tallinn",
              "Finland, Helsinki", "France, Paris", "Germany,
Berlin",
              "Greece, Athens", "Georgia, Tbilisi", "Hungary,
Budapest",
               "Iceland, Reykjavik", "Italy, Rome", "Latvia,
Riga",
               "Kazakhstan, Astana", "Liechtenstein, Vaduz",
"Lithuania, Vilnius",
               "Luxembourg, Luxembourg", "Macedonia, Skopje",
"Malta, Valletta",
                "Moldova, Chişinău", "Monaco, Monaco-Ville",
"Montenegro, Podgorica",
                 "Netherlands, Amsterdam", "Norway, Oslo",
```

```
"Poland, Warsaw",
               "Portugal, Lisbon", "Republic of Ireland,
Dublin",
              "Romania, Bucharest", "Russia, Moscow", "San
Marino, San Marino",
               "Serbia, Belgrade", "Slovakia, Bratislava",
"Slovenia, Ljubljana",
                "Spain, Madrid", "Sweden, Stockholm",
"Switzerland, Bern",
               "Turkey, Ankara", "Ukraine, Kiev", "United
Kingdom, London",
             "Vatican City, Vatican City"
             )
theData <- geocode(capitals)
rownames(theData) <- capitals

###############
#             #
# Exercise 1  #
#             #
###############

distances <- dist(theData)

###############
#             #
# Exercise 2  #
#             #
###############

dendrogram <- hclust(distances)

###############
#             #
# Exercise 3  #
#             #
###############

plot(dendrogram)

###############
```

```
#             #
# Exercise 4  #
#             #
###############
#
plot(dendrogram, hang=-1)

###############
#             #
# Exercise 5  #
#             #
###############

cutree(dendrogram, k=3)

##                 Albania, Tirana        Andorra, Andorra la
Vella
##                                                          1
2
##                Armenia, Yerevan                 Austria,
Vienna
##                                                          1
1
##                Azerbaijan, Baku                 Belarus,
Minsk
##                                                          1
1
##                Belgium, Brussels Bosnia and Herzegovina,
Sarajevo
##                                                          2
1
##                 Bulgaria, Sofia                 Croatia,
Zagreb
##                                                          1
1
##                 Cyprus, Nicosia           Czech Republic,
Prague
##                                                          1
1
##              Denmark, Copenhagen                Estonia,
Tallinn
```

```
##                                                                 1
1
##                Finland, Helsinki                    France,
Paris
##                                                                 1
2
##                  Germany, Berlin                    Greece,
Athens
##                                                                 1
1
##                  Georgia, Tbilisi                 Hungary,
Budapest
##                                                                 1
1
##                                    Iceland,  Reykjavik
Italy, Rome
##                                                                 2
1
##                     Latvia, Riga                Kazakhstan,
Astana
##                                                                 1
3
##             Liechtenstein, Vaduz                 Lithuania,
Vilnius
##                                                                 1
1
##           Luxembourg, Luxembourg                 Macedonia,
Skopje
##                                                                 2
1
##                  Malta, Valletta                 Moldova,
Chişinău
##                                                                 1
1
##              Monaco, Monaco-Ville              Montenegro,
Podgorica
##                                                                 1
1
##                                  Netherlands,  Amsterdam
Norway, Oslo
##                                                                 2
```

```
1
##                  Poland, Warsaw                 Portugal,
Lisbon
##                                                         1
2
##      Republic of Ireland, Dublin                 Romania,
Bucharest
##                                                         2
1
##                  Russia, Moscow           San Marino, San
Marino
##                                                         1
1
##                 Serbia, Belgrade              Slovakia,
Bratislava
##                                                         1
1
##              Slovenia, Ljubljana                   Spain,
Madrid
##                                                         1
2
##                                    Sweden,  Stockholm
Switzerland, Bern
##                                                         1
1
##                                       Turkey,  Ankara
Ukraine, Kiev
##                                                         1
1
##            United Kingdom, London       Vatican City,
Vatican City
##                                                         2
1

###############
#             #
# Exercise 6  #
#             #
###############

cutree(dendrogram, h=20)
```

```
##                  Albania, Tirana        Andorra, Andorra la
Vella
##                                                          1
2
##                Armenia, Yerevan                  Austria,
Vienna
##                                                          3
1
##                Azerbaijan, Baku                  Belarus,
Minsk
##                                                          3
4
##                Belgium, Brussels Bosnia and Herzegovina,
Sarajevo
##                                                          2
1
##                 Bulgaria, Sofia                  Croatia,
Zagreb
##                                                          1
1
##                 Cyprus, Nicosia          Czech Republic,
Prague
##                                                          3
1
##              Denmark, Copenhagen                 Estonia,
Tallinn
##                                                          5
5
##                Finland, Helsinki                  France,
Paris
##                                                          5
2
##                  Germany, Berlin                  Greece,
Athens
##                                                          1
1
##                  Georgia, Tbilisi                Hungary,
Budapest
##                                                          3
1
##                                   Iceland,  Reykjavik
```

```
Italy, Rome
##                                                                  6
1
##                    Latvia, Riga                Kazakhstan,
Astana
##                                                                  5
7
##             Liechtenstein, Vaduz                Lithuania,
Vilnius
##                                                                  1
4
##           Luxembourg, Luxembourg                Macedonia,
Skopje
##                                                                  2
1
##                 Malta, Valletta                  Moldova,
Chişinău
##                                                                  1
4
##              Monaco, Monaco-Ville             Montenegro,
Podgorica
##                                                                  1
1
##                                Netherlands,   Amsterdam
Norway, Oslo
##                                                                  2
5
##                  Poland, Warsaw                Portugal,
Lisbon
##                                                                  5
2
##      Republic of Ireland, Dublin                Romania,
Bucharest
##                                                                  2
4
##                  Russia, Moscow           San Marino, San
Marino
##                                                                  4
1
##                 Serbia, Belgrade             Slovakia,
Bratislava
```

```
##                                                                1
1
##              Slovenia, Ljubljana                       Spain,
Madrid
##                                                                1
2
##                                      Sweden,  Stockholm
Switzerland, Bern
##                                                                5
1
##                                         Turkey,  Ankara
Ukraine, Kiev
##                                                                3
4
##            United Kingdom, London       Vatican City,
Vatican City
##                                                                2
1
```

```
plot(dendrogram)
abline(h=20, col="red", lty=2)
```

```
###############
#             #
# Exercise 7  #
#             #
###############
```

```
plot(dendrogram, hang=-1)
rect.hclust(dendrogram, k=3, border=1:3)
```

```
###############
#             #
# Exercise 8  #
#             #
###############
```

```
plot(dendrogram, hang=-1)
rect.hclust(dendrogram, k=3, border="red")
rect.hclust(dendrogram, h=20, border="blue")
```

```
###############
#             #
# Exercise 9  #
#             #
###############

# ward.D
plot(hclust(distances, method="ward.D"), main="ward.D")
rect.hclust(hclust(distances,   method="ward.D"),   k=5,
border=1:5)
# ward.D2
plot(hclust(distances, method="ward.D2"), main="ward.D2")
rect.hclust(hclust(distances,   method="ward.D2"),   k=5,
border=1:5)
# single
plot(hclust(distances, method="single"), main="single")
rect.hclust(hclust(distances,   method="single"),   k=5,
border=1:5)
# complete
plot(hclust(distances, method="complete"), main="complete")
rect.hclust(hclust(distances,   method="complete"),   k=5,
border=1:5)
# average
plot(hclust(distances, method="average"), main="average")
rect.hclust(hclust(distances,   method="average"),   k=5,
border=1:5)
# mcquitty
plot(hclust(distances, method="mcquitty"), main="mcquitty")
rect.hclust(hclust(distances,   method="mcquitty"),   k=5,
border=1:5)
# median
plot(hclust(distances, method="median"), main="median")
rect.hclust(hclust(distances,   method="median"),   k=5,
border=1:5)
# centroid
plot(hclust(distances, method="centroid"), main="centroid")
rect.hclust(hclust(distances,   method="centroid"),   k=5,
border=1:5)

###############
#             #
```

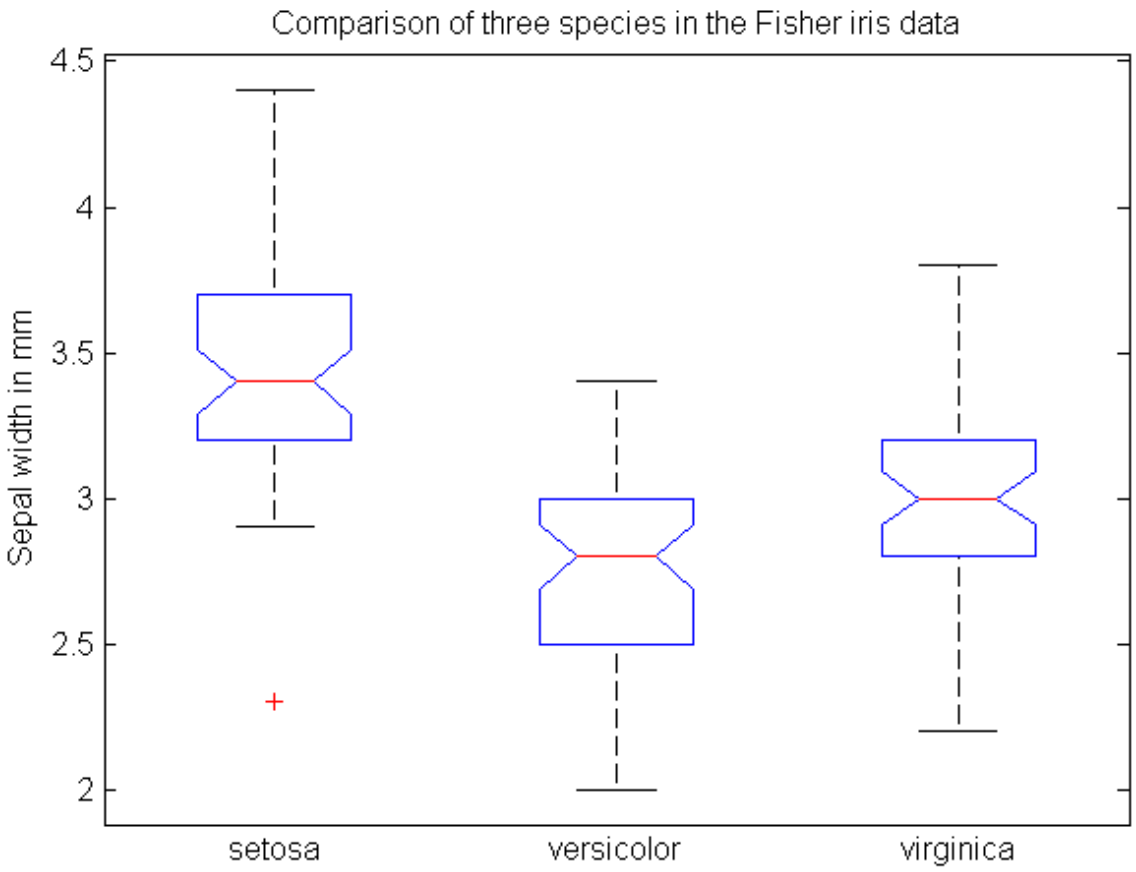
```
# Exercise 10  #
#              #
################

myVersion <- hclust(distances, method="complete")
groups    <- cutree(myVersion, 7)
plot(theData, cex=0, xlim=c(-30,75))
text(theData, rownames(theData), cex=0.6, col=groups)
```

Replicating Plots – Boxplot Exercises

R’s boxplot function has a lot of useful parameters allowing us to change the behaviour and appearance of the boxplot graphs. In this exercise we will try to use those parameters in order to replicate the visual style of Matlab’s boxplot. Before trying out this exercise please make sure that you are familiar with the following functions: bxp, boxplot, axis, mtext

Here is the plot we will be replicating:



We will be using the same **iris** dataset which is available in R by default in the variable of the same name – iris. The exercises will require you to make incremental changes to the default boxplot style.

Answers to the exercises are available here.

If you obtained a different (correct) answer than those listed on the solutions page, please feel free to post your answer as a comment on that page.

Exercise 1
Make a default boxplot of Sepal.Width stratified by Species.

Exercise 2
Change the range of the y-axis so it starts at 2 and ends at 4.5.

Exercise 3

Modify the boxplot function so it doesn’t draw ticks nor labels of the x and y axes.

Exercise 4
Add notches (triangular dents around the median representing confidence intervals) to the boxes in the plot.

Exercise 5
Increase the distance between boxes in the plot.

Exercise 6
Change the color of the box borders to blue.

Exercise 7
a. Change the color of the median lines to red.
b. Change the line width of the median line to 1.

Exercise 8
a. Change the color of the outlier points to red.
b. Change the symbol of the outlier points to “+”.
c. Change the size of the outlier points to 0.8.

Exercise 9
a. Add the title to the boxplot (try to replicate the style of matlab’s boxplot).
b. Add the y-axis label to the boxplot (try to replicate the style of matlab’s boxplot).

Exercise 10
a. Add x-axis (try to make it resemble the x-axis in the matlab’s boxplot)
b. Add y-axis (try to make it resemble the y-axis in the matlab’s boxplot)
c. Add the y-axis ticks on the other side.

NOTE: You can use format(as.character(c(2, 4.5)), drop0trailing=TRUE, justify="right") to obtain the text for y-axis labels.

Replicating Plots – Boxplot Solutions

Below are the solutions to these exercises on boxplot parameters.

```
###############
#             #
# Exercise 1  #
#             #
###############

boxplot(Sepal.Width ~ Species, data=iris)

###############
#             #
# Exercise 2  #
#             #
###############

boxplot(Sepal.Width ~ Species, data=iris, ylim=c(2, 4.5))

###############
#             #
# Exercise 3  #
#             #
###############

boxplot(Sepal.Width ~ Species, data=iris, ylim=c(2, 4.5),
xaxt='n', yaxt='n')

###############
#             #
# Exercise 4  #
#             #
###############
```

```
boxplot(Sepal.Width ~ Species, data=iris, ylim=c(2, 4.5),
xaxt='n', yaxt='n',
        notch=TRUE
        )

###############
#             #
# Exercise 5  #
#             #
###############

boxplot(Sepal.Width ~ Species, data=iris, ylim=c(2, 4.5),
xaxt='n', yaxt='n',
        notch=TRUE, boxwex=0.5
        )

###############
#             #
# Exercise 6  #
#             #
###############

boxplot(Sepal.Width ~ Species, data=iris, ylim=c(2, 4.5),
xaxt='n', yaxt='n',
        notch=TRUE, boxwex=0.5, boxcol="blue"
        )

###############
#             #
# Exercise 7  #
#             #
###############

boxplot(Sepal.Width ~ Species, data=iris, ylim=c(2, 4.5),
xaxt='n', yaxt='n',
         notch=TRUE, boxwex=0.5, boxcol="blue", medcol="red",
medlwd=1
        )

###############
#             #
```

```
# Exercise 8  #
#             #
###############

boxplot(Sepal.Width ~ Species, data=iris, ylim=c(2, 4.5),
xaxt='n', yaxt='n',
         notch=TRUE, boxwex=0.5, boxcol="blue", medcol="red",
medlwd=1,
        outcol="red", outpch=3, outcex=0.8
        )

###############
#             #
# Exercise 9  #
#             #
###############

boxplot(Sepal.Width ~ Species, data=iris, ylim=c(2, 4.5),
xaxt='n', yaxt='n',
         notch=TRUE, boxwex=0.5, boxcol="blue", medcol="red",
medlwd=1,
        outcol="red", outpch=3, outcex=0.8
        )
mtext("Comparison of three species in the Fisher iris data",
3, cex=0.9)
mtext("Sepal width in mm", 2, cex=0.9, line=2)

################
#              #
# Exercise 10  #
#              #
################

boxplot(Sepal.Width ~ Species, data=iris, ylim=c(2, 4.5),
xaxt='n', yaxt='n',
         notch=TRUE, boxwex=0.5, boxcol="blue", medcol="red",
medlwd=1,
        outcol="red", outpch=3, outcex=0.8
        )
mtext("Comparison of three species in the Fisher iris data",
3, cex=0.9)
```

```
mtext("Sepal width in mm", 2, cex=0.9, line=2)

lab    <-    format(as.character(pretty(c(2,4.5))),
drop0trailing=TRUE, justify="right")
axis(2, tck=0.02, at=pretty(c(2,4.5)), labels=lab, las=1,
hadj=0.3)
axis(4, tck=0.02, labels=FALSE)
axis(1, at=1:3, labels=unique(iris$Species), tck=0, padj=-1)
```

Combinations Exercises

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    2    4    6    8   10
[3,]    3    6    9   12   15
[4,]    4    8   12   16   20
[5,]    5   10   15   20   25
```

When doing data analysis it happens often that we have a set of values and want to obtain various possible combinations of them. For example, taking 5 random samples from a dataset of 20. How many possible 5-sample sets are there and how to obtain all of them? R has a bunch of functions that help with tasks like these: expand.grid, combn, outer and choose.

Answers to the exercises are available here.

If you obtained a different (correct) answer than those listed on the solutions page, please feel free to post your answer as a comment on that page.

Exercise 1
You first throw a coin with 2 possible outcomes: it either lands heads or tails. Then you throw a dice with 6 possible outcomes: 1,2,3,4,5 and 6.
Generate all possible results of your action.

Exercise 2
Generate a multiplication table for numbers ranging from 1 to 10.

Exercise 3
You have a set of card values:
values <- c("A", 2, 3, 4, 5, 6, 7, 8, 9, 10, "J", "Q", "K")
and a set of suits representing diamonds, clubs, spades and hearths:
suits <- c("d", "c", "s", "h")
Generate a deck of playing cards. (e.g. King of spades should be represented as Ks).

Note: function paste(..., sep="") can be used to combine characters.

Exercise 4
Think about a game of poker using a standard deck of cards like the one generated earlier. Starting hand in poker consists of 5 cards and their order does not matter. How many different starting hands are there in total?

Exercise 5
You have a set of colors to choose from:
colors <- c("red", "blue", "green", "white", "black", "yellow")
You have to pick 3 colors and you cant' pick the same color more than once. List all possible combinations.

Exercise 6
Using the same set of colors – pick 3 without picking the same more than once, just like in the previous exercise.
List all possible combinations but this time sort each combination alphabetically.

Exercise 7
You have the same choices of colors and have to pick 3 but this time you can pick the same color more than once.
List all possible combinations.

Exercise 8
You have the same set of colors but this time instead of having to pick 3 you can choose to pick either 1, 2 or 3.

How many different choices can you make?

Exercise 9
You have the same set of colors and you can choose to pick either 1, 2 or 3.
Make a list of all possible choices.

Exercise 10
There are 3 color palletes: the first one has 4 colors, the second has 6 colors and the third has 8 colors. You have to pick a pallete and then choose up to 5 (1, 2, 3, 4 or 5) colors from the chosen color pallete. How many different possibilities are there?

Combinations Solutions

Below are the solutions to these exercises on combinations.

```
###############
#             #
# Exercise 1  #
#             #
###############
```

```
expand.grid(c("H","T"), 1:6)
```

```
##    Var1 Var2
## 1     H    1
## 2     T    1
## 3     H    2
## 4     T    2
## 5     H    3
## 6     T    3
## 7     H    4
## 8     T    4
```

```
## 9     H    5
## 10    T    5
## 11    H    6
## 12    T    6

###############
#             #
# Exercise 2  #
#             #
###############

outer(1:10, 1:10)

##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]    1    2    3    4    5    6    7    8    9    10
##  [2,]    2    4    6    8   10   12   14   16   18    20
##  [3,]    3    6    9   12   15   18   21   24   27    30
##  [4,]    4    8   12   16   20   24   28   32   36    40
##  [5,]    5   10   15   20   25   30   35   40   45    50
##  [6,]    6   12   18   24   30   36   42   48   54    60
##  [7,]    7   14   21   28   35   42   49   56   63    70
##  [8,]    8   16   24   32   40   48   56   64   72    80
##  [9,]    9   18   27   36   45   54   63   72   81    90
## [10,]   10   20   30   40   50   60   70   80   90   100

###############
#             #
# Exercise 3  #
#             #
###############

values <- c("A", 2, 3, 4, 5, 6, 7, 8, 9, 10, "J", "Q", "K")
suits  <- c("d", "c", "s", "h")

outer(values, suits, FUN="paste", sep="")

##       [,1]  [,2]  [,3]  [,4]
##  [1,] "Ad"  "Ac"  "As"  "Ah"
##  [2,] "2d"  "2c"  "2s"  "2h"
##  [3,] "3d"  "3c"  "3s"  "3h"
```

```
##  [4,] "4d"  "4c"  "4s"  "4h"
##  [5,] "5d"  "5c"  "5s"  "5h"
##  [6,] "6d"  "6c"  "6s"  "6h"
##  [7,] "7d"  "7c"  "7s"  "7h"
##  [8,] "8d"  "8c"  "8s"  "8h"
##  [9,] "9d"  "9c"  "9s"  "9h"
## [10,] "10d" "10c" "10s" "10h"
## [11,] "Jd"  "Jc"  "Js"  "Jh"
## [12,] "Qd"  "Qc"  "Qs"  "Qh"
## [13,] "Kd"  "Kc"  "Ks"  "Kh"

# Comment: Another possible alternative is to use expand.grid(values, suits)
# But in that case we would later have to combine the characters with paste manually.

###############
#             #
# Exercise 4  #
#             #
###############

choose(52, 5)

## [1] 2598960

###############
#             #
# Exercise 5  #
#             #
###############

colors <- c("red", "blue", "green", "white", "black", "yellow")

combn(colors, 3)

##      [,1]    [,2]    [,3]    [,4]     [,5]    [,6]    [,7]    [,8]
## [1,] "red"   "red"   "red"   "red"    "red"   "red"   "red"
```

```
"red"
## [2,] "blue"  "blue"  "blue"  "blue"   "green" "green"
"green"  "white"
## [3,] "green" "white" "black" "yellow" "white" "black"
"yellow" "black"
##      [,9]     [,10]    [,11]   [,12]   [,13]    [,14]
[,15]    [,16]
## [1,] "red"    "red"    "blue"  "blue"  "blue"   "blue"
"blue"   "blue"
## [2,] "white"  "black"  "green" "green" "green"  "white"
"white"  "black"
## [3,] "yellow" "yellow" "white" "black" "yellow" "black"
"yellow" "yellow"
##      [,17]   [,18]    [,19]    [,20]
## [1,] "green" "green"  "green"  "white"
## [2,] "white" "white"  "black"  "black"
## [3,] "black" "yellow" "yellow" "yellow"

###############
#             #
# Exercise 6  #
#             #
###############

combn(colors, 3, FUN=sort)

##      [,1]    [,2]    [,3]    [,4]     [,5]    [,6]    [,7]
[,8]
## [1,] "blue"  "blue"  "black" "blue"   "green" "black"
"green"  "black"
## [2,] "green" "red"   "blue"  "red"    "red"   "green" "red"
"red"
## [3,] "red"    "white" "red"    "yellow" "white" "red"
"yellow" "white"
##      [,9]     [,10]    [,11]   [,12]   [,13]    [,14]
[,15]    [,16]
## [1,] "red"    "black"  "blue"  "black" "blue"   "black"
"blue"   "black"
## [2,] "white"  "red"    "green" "blue"  "green"  "blue"
"white"  "blue"
## [3,] "yellow" "yellow" "white" "green" "yellow" "white"
```

```
"yellow" "yellow"
##      [,17]   [,18]    [,19]    [,20]
## [1,] "black" "green"  "black"  "black"
## [2,] "green" "white"  "green"  "white"
## [3,] "white" "yellow" "yellow" "yellow"

###############
#             #
# Exercise 7  #
#             #
###############

expand.grid(colors, colors, colors)

##       Var1   Var2   Var3
## 1      red    red    red
## 2     blue    red    red
## 3    green    red    red
## 4    white    red    red
## 5    black    red    red
## 6   yellow    red    red
## 7      red   blue    red
## 8     blue   blue    red
## 9    green   blue    red
## 10   white   blue    red
## 11   black   blue    red
## 12  yellow   blue    red
## 13     red  green    red
## 14    blue  green    red
## 15   green  green    red
## 16   white  green    red
## 17   black  green    red
## 18  yellow  green    red
## 19     red  white    red
## 20    blue  white    red
## 21   green  white    red
## 22   white  white    red
## 23   black  white    red
## 24  yellow  white    red
## 25     red  black    red
## 26    blue  black    red
```

```
## 27   green  black    red
## 28   white  black    red
## 29   black  black    red
## 30  yellow  black    red
## 31     red yellow    red
## 32    blue yellow    red
## 33   green yellow    red
## 34   white yellow    red
## 35   black yellow    red
## 36  yellow yellow    red
## 37     red    red   blue
## 38    blue    red   blue
## 39   green    red   blue
## 40   white    red   blue
## 41   black    red   blue
## 42  yellow    red   blue
## 43     red   blue   blue
## 44    blue   blue   blue
## 45   green   blue   blue
## 46   white   blue   blue
## 47   black   blue   blue
## 48  yellow   blue   blue
## 49     red  green   blue
## 50    blue  green   blue
## 51   green  green   blue
## 52   white  green   blue
## 53   black  green   blue
## 54  yellow  green   blue
## 55     red  white   blue
## 56    blue  white   blue
## 57   green  white   blue
## 58   white  white   blue
## 59   black  white   blue
## 60  yellow  white   blue
## 61     red  black   blue
## 62    blue  black   blue
## 63   green  black   blue
## 64   white  black   blue
## 65   black  black   blue
## 66  yellow  black   blue
## 67     red yellow   blue
```

```
## 68    blue yellow   blue
## 69   green yellow   blue
## 70   white yellow   blue
## 71   black yellow   blue
## 72  yellow yellow   blue
## 73     red    red  green
## 74    blue    red  green
## 75   green    red  green
## 76   white    red  green
## 77   black    red  green
## 78  yellow    red  green
## 79     red   blue  green
## 80    blue   blue  green
## 81   green   blue  green
## 82   white   blue  green
## 83   black   blue  green
## 84  yellow   blue  green
## 85     red  green  green
## 86    blue  green  green
## 87   green  green  green
## 88   white  green  green
## 89   black  green  green
## 90  yellow  green  green
## 91     red  white  green
## 92    blue  white  green
## 93   green  white  green
## 94   white  white  green
## 95   black  white  green
## 96  yellow  white  green
## 97     red  black  green
## 98    blue  black  green
## 99   green  black  green
## 100  white  black  green
## 101  black  black  green
## 102 yellow  black  green
## 103    red yellow  green
## 104   blue yellow  green
## 105  green yellow  green
## 106  white yellow  green
## 107  black yellow  green
## 108 yellow yellow  green
```

```
## 109    red    red  white
## 110   blue    red  white
## 111  green    red  white
## 112  white    red  white
## 113  black    red  white
## 114 yellow    red  white
## 115    red   blue  white
## 116   blue   blue  white
## 117  green   blue  white
## 118  white   blue  white
## 119  black   blue  white
## 120 yellow   blue  white
## 121    red  green  white
## 122   blue  green  white
## 123  green  green  white
## 124  white  green  white
## 125  black  green  white
## 126 yellow  green  white
## 127    red  white  white
## 128   blue  white  white
## 129  green  white  white
## 130  white  white  white
## 131  black  white  white
## 132 yellow  white  white
## 133    red  black  white
## 134   blue  black  white
## 135  green  black  white
## 136  white  black  white
## 137  black  black  white
## 138 yellow  black  white
## 139    red yellow  white
## 140   blue yellow  white
## 141  green yellow  white
## 142  white yellow  white
## 143  black yellow  white
## 144 yellow yellow  white
## 145    red    red  black
## 146   blue    red  black
## 147  green    red  black
## 148  white    red  black
## 149  black    red  black
```

```
## 150 yellow    red  black
## 151    red   blue  black
## 152   blue   blue  black
## 153  green   blue  black
## 154  white   blue  black
## 155  black   blue  black
## 156 yellow   blue  black
## 157    red  green  black
## 158   blue  green  black
## 159  green  green  black
## 160  white  green  black
## 161  black  green  black
## 162 yellow  green  black
## 163    red  white  black
## 164   blue  white  black
## 165  green  white  black
## 166  white  white  black
## 167  black  white  black
## 168 yellow  white  black
## 169    red  black  black
## 170   blue  black  black
## 171  green  black  black
## 172  white  black  black
## 173  black  black  black
## 174 yellow  black  black
## 175    red yellow  black
## 176   blue yellow  black
## 177  green yellow  black
## 178  white yellow  black
## 179  black yellow  black
## 180 yellow yellow  black
## 181    red    red yellow
## 182   blue    red yellow
## 183  green    red yellow
## 184  white    red yellow
## 185  black    red yellow
## 186 yellow    red yellow
## 187    red   blue yellow
## 188   blue   blue yellow
## 189  green   blue yellow
## 190  white   blue yellow
```

```
## 191  black   blue yellow
## 192 yellow   blue yellow
## 193    red  green yellow
## 194   blue  green yellow
## 195  green  green yellow
## 196  white  green yellow
## 197  black  green yellow
## 198 yellow  green yellow
## 199    red  white yellow
## 200   blue  white yellow
## 201  green  white yellow
## 202  white  white yellow
## 203  black  white yellow
## 204 yellow  white yellow
## 205    red  black yellow
## 206   blue  black yellow
## 207  green  black yellow
## 208  white  black yellow
## 209  black  black yellow
## 210 yellow  black yellow
## 211    red yellow yellow
## 212   blue yellow yellow
## 213  green yellow yellow
## 214  white yellow yellow
## 215  black yellow yellow
## 216 yellow yellow yellow

###############
#             #
# Exercise 8  #
#             #
###############

choose(length(colors), 1) + choose(length(colors), 2) +
choose(length(colors), 3)

## [1] 41

###############
#             #
# Exercise 9  #
```

```
#             #
###############

c(combn(colors,  1,  simplify=FALSE),  combn(colors,  2,
simplify=FALSE), combn(colors, 3, simplify=FALSE))

## [[1]]
## [1] "red"
##
## [[2]]
## [1] "blue"
##
## [[3]]
## [1] "green"
##
## [[4]]
## [1] "white"
##
## [[5]]
## [1] "black"
##
## [[6]]
## [1] "yellow"
##
## [[7]]
## [1] "red"  "blue"
##
## [[8]]
## [1] "red"   "green"
##
## [[9]]
## [1] "red"   "white"
##
## [[10]]
## [1] "red"   "black"
##
## [[11]]
## [1] "red"    "yellow"
##
## [[12]]
## [1] "blue"  "green"
```

```
##
## [[13]]
## [1] "blue"  "white"
##
## [[14]]
## [1] "blue"  "black"
##
## [[15]]
## [1] "blue"   "yellow"
##
## [[16]]
## [1] "green" "white"
##
## [[17]]
## [1] "green" "black"
##
## [[18]]
## [1] "green"  "yellow"
##
## [[19]]
## [1] "white" "black"
##
## [[20]]
## [1] "white"  "yellow"
##
## [[21]]
## [1] "black"  "yellow"
##
## [[22]]
## [1] "red"   "blue"  "green"
##
## [[23]]
## [1] "red"   "blue"  "white"
##
## [[24]]
## [1] "red"   "blue"  "black"
##
## [[25]]
## [1] "red"    "blue"   "yellow"
##
## [[26]]
```

```
## [1] "red"   "green" "white"
##
## [[27]]
## [1] "red"   "green" "black"
##
## [[28]]
## [1] "red"    "green"  "yellow"
##
## [[29]]
## [1] "red"   "white" "black"
##
## [[30]]
## [1] "red"    "white"  "yellow"
##
## [[31]]
## [1] "red"    "black"  "yellow"
##
## [[32]]
## [1] "blue"  "green" "white"
##
## [[33]]
## [1] "blue"  "green" "black"
##
## [[34]]
## [1] "blue"   "green"  "yellow"
##
## [[35]]
## [1] "blue"  "white" "black"
##
## [[36]]
## [1] "blue"   "white"  "yellow"
##
## [[37]]
## [1] "blue"   "black"  "yellow"
##
## [[38]]
## [1] "green" "white" "black"
##
## [[39]]
## [1] "green"  "white"  "yellow"
##
```

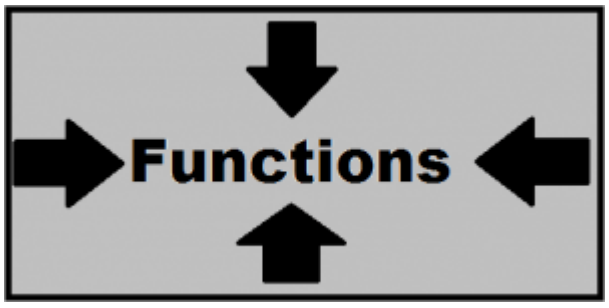
```
## [[40]]
## [1] "green"  "black"  "yellow"
##
## [[41]]
## [1] "white"  "black"  "yellow"

################
#              #
# Exercise 10  #
#              #
################

sum(outer(c(4,6,8), 1:5, choose))

## [1] 295
```

Higher Order Functions Exercises



Higher order functions are functions that take other functions as arguments or return functions as their result. In this set of exercises we will focus on the former. R has a set of built-in higher order functions: Map, Reduce, Filter, Find, Position, Negate. They enable us to complete complex operations by using simple single-purpose functions as their building blocks. In R this is especially helpful in cases where we cannot depend on vectorization and have to utilize control statements like for loops. In such scenarios higher order functions help us by: a) simplifying and shortening the syntax, b) getting rid of counter indices and c) getting rid

of temporary storage values.

Exercises in this section will have to be solved by using one or more of the higher order functions mentioned above. It might be useful reading their help page before continuing.

Answers to the exercises are available here.

If you obtained a different (correct) answer than those listed on the solutions page, please feel free to post your answer as a comment on that page.

Exercise 1
You are working on 3 datasets all at once:
multidata <- list(mtcars, USArrests, rock)
summary(multidata[[1]]) will return the summary information for a single dataset.
Obtain summary information for every dataset in the list.

Exercise 2
cumsum(1:100) returns the cumulative sums of a vector of numbers from 1 to 100.
Do the same using sum and an appropriate higher order function.

Exercise 3
You have a vector of numbers from 1 to 10. You want to multiply all those numbers first by 2 and then by 4. Why the following line does not work and how to fix it?
Map(`*`, 1:10, c(2,4))

Exercise 4
Expression sample(LETTERS, 5, replace=TRUE) obtains 5 random letters.
Generate a list with 10 elements, where first element contains 1 random letter, second element 2 random letters and so on.

Note: use a fixed random seed: set.seed(14)

Exercise 5

Library spatstat has a function is.prime() that checks if a given number is a prime.
Find all prime numbers between 100 and 200.

Exercise 6
We have a vector containing all the words of the English language –
words <- scan("http://www-01.sil.org/linguistics/wordlists/english/wordlist/wordsEn.txt", what="character")
a. Using a function that checks if a given words contains any vowels:
containsVowel <- function(x) grepl("a|o|e|i|u", x)
find all words that do not contain any vowels.
b. Using a function is.colour() from the spatstat library find the index of the first word inside the words vector corresponding to a valid R color.

Exercise 7
a. Find the smallest number between 10000 and 20000 that is divisible by 1234.
b. Find the largest number between 10000 and 20000 that is divisible by 1234.

Exercise 8
Consider the babynames dataset from the babynames library.
Start with a list containing the used names for each year:
library(babynames); namesData <- split(babynames$name, babynames$year)
a. Obtain a set of names that were present in every year.
b. Obtain a set of names that are only present in year 2014

Exercise 9
Using the same babynames dataset and a function that checks if word has more than 3 letters: moreThan3 <- function(x) nchar(x) > 3
Inside each year list leave only the names that have 3 letters or less.

Exercise 10
Using the same babynames dataset:
a. Split each name to a list of letters.
b. Join each list of letters by inserting an underscore “_” after each letter.

Note: if you have a word x <- "exercise" you can split it with x2 <- strsplit(x, "")
and join using underscores with paste(x2[[1]], collapse="_")

Higher Order Functions Solutions

Below are the solutions to these exercises on higher order functions.

```
###############
#             #
# Exercise 1  #
#             #
###############

multidata <- list(mtcars, USArrests, rock)

Map(summary, multidata)

## [[1]]
##       mpg             cyl             disp             hp       
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0
```

```
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean
:146.7
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd
Qu.:180.0
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.
:335.0
##       drat             wt             qsec             vs
##  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.
:0.0000
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st
Qu.:0.0000
##  Median :3.695   Median :3.325   Median :17.71   Median
:0.0000
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean
:0.4375
##  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd
Qu.:1.0000
##  Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.
:1.0000
##       am              gear            carb
##  Min.   :0.0000   Min.   :3.000   Min.   :1.000
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000
##  Median :0.0000   Median :4.000   Median :2.000
##  Mean   :0.4062   Mean   :3.688   Mean   :2.812
##  3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000
##  Max.   :1.0000   Max.   :5.000   Max.   :8.000
##
## [[2]]
##      Murder          Assault         UrbanPop          Rape
##  Min.   : 0.800   Min.   : 45.0   Min.   :32.00   Min.   :
7.30
##  1st Qu.: 4.075   1st Qu.:109.0   1st Qu.:54.50   1st
Qu.:15.07
##  Median : 7.250   Median :159.0   Median :66.00   Median
:20.10
##  Mean   : 7.788   Mean   :170.8   Mean   :65.54   Mean
:21.23
##  3rd Qu.:11.250   3rd Qu.:249.0   3rd Qu.:77.75   3rd
Qu.:26.18
##  Max.   :17.400   Max.   :337.0   Max.   :91.00   Max.
:46.00
```

```
##
## [[3]]
##          area                peri                shape
perm
##  Min.   : 1016   Min.   : 308.6   Min.   :0.09033   Min.
:   6.30
##  1st Qu.: 5305   1st Qu.:1414.9   1st Qu.:0.16226   1st
Qu.:  76.45
##  Median : 7487   Median :2536.2   Median :0.19886   Median
: 130.50
##  Mean   : 7188   Mean   :2682.2   Mean   :0.21811   Mean
: 415.45
##  3rd Qu.: 8870   3rd Qu.:3989.5   3rd Qu.:0.26267   3rd
Qu.: 777.50
##  Max.   :12212   Max.   :4864.2   Max.   :0.46413   Max.
:1300.00

###############
#             #
# Exercise 2  #
#             #
###############

Reduce(sum, 1:100, accumulate=TRUE)

##   [1]    1    3    6   10   15   21   28   36   45   55
66   78   91  105
##  [15]  120  136  153  171  190  210  231  253  276  300
325  351  378  406
##  [29]  435  465  496  528  561  595  630  666  703  741
780  820  861  903
##  [43]  946  990 1035 1081 1128 1176 1225 1275 1326 1378
1431 1485 1540 1596
##  [57] 1653 1711 1770 1830 1891 1953 2016 2080 2145 2211
2278 2346 2415 2485
##  [71] 2556 2628 2701 2775 2850 2926 3003 3081 3160 3240
3321 3403 3486 3570
##  [85] 3655 3741 3828 3916 4005 4095 4186 4278 4371 4465
4560 4656 4753 4851
##  [99] 4950 5050
```

```
###############
#             #
# Exercise 3  #
#             #
###############

Map(`*`, list(1:10), c(2,4))


## [[1]]
##  [1]  2  4  6  8 10 12 14 16 18 20
##
## [[2]]
##  [1]  4  8 12 16 20 24 28 32 36 40

# Map(`*`, 1:10, c(2,4)) didn't work because Map treated 1:10
as 10 separate
# elements and tried to iterate through all of them
multiplying every second
# element by 2 and others by 4.

###############
#             #
# Exercise 4  #
#             #
###############

set.seed(14)
Map(sample, list(LETTERS), 1:10, replace=TRUE)


## [[1]]
## [1] "G"
##
## [[2]]
## [1] "Q" "Y"
##
## [[3]]
## [1] "O" "Z" "N"
##
## [[4]]
## [1] "Y" "L" "M" "J"
##
```

```
## [[5]]
## [1] "X" "E" "M" "W" "W"
##
## [[6]]
## [1] "T" "J" "R" "W" "P" "J"
##
## [[7]]
## [1] "K" "Q" "R" "T" "Q" "K" "N"
##
## [[8]]
## [1] "R" "C" "V" "D" "Z" "R" "Y" "D"
##
## [[9]]
## [1] "O" "M" "W" "Q" "C" "Z" "K" "L" "O"
##
## [[10]]
##  [1] "F" "K" "F" "T" "R" "F" "D" "Y" "A" "U"

###############
#             #
# Exercise 5  #
#             #
###############

library(spatstat)

Filter(is.prime, 100:200)

##  [1] 101 103 107 109 113 127 131 137 139 149 151 157 163
167 173 179 181
## [18] 191 193 197 199

###############
#             #
# Exercise 6  #
#             #
###############

library(spatstat)
words                                                    <-
scan("http://www-01.sil.org/linguistics/wordlists/english/word
```

```
list/wordsEn.txt", what="character")
containsVowel <- function(x) grepl("a|o|e|i|u", x)

# a.
Filter(Negate(containsVowel), words)
```

```
##   [1] "bb"      "bbl"     "bdrm"    "bks"     "bldg"
"blvd"    "bps"
##   [8] "br"      "by"      "cc"      "cd"      "cgs"
"chm"     "cl"
##  [15] "cmdg"    "cpl"     "cps"     "cr"      "crc"
"cry"     "crypt"
##  [22] "crypts"  "cs"      "csp"     "cst"     "ct"
"ctg"     "ctrl"
##  [29] "cts"     "cwt"     "cyst"    "cysts"   "db"
"dbl"     "dbms"
##  [36] "dc"      "dp"      "dry"     "dryly"   "drys"
"dx"      "fly"
##  [43] "flyby"   "flybys"  "fps"     "fry"     "fwd"
"gds"     "glyph"
##  [50] "glyphs"  "gym"     "gyms"    "gyp"     "gyps"
"gypsy"   "hdqrs"
##  [57] "hp"      "hr"      "hrs"     "hts"     "hwy"
"hymn"    "hymns"
##  [64] "jct"     "kb"      "kl"      "kwhr"    "lbs"
"lf"      "lh"
##  [71] "ll"      "lpm"     "lr"      "lymph"   "lymphs"
"lynch"   "lynx"
##  [78] "mb"      "mc"      "md"      "mf"      "mfd"
"mfg"     "mg"
##  [85] "mkt"     "mktg"    "mn"      "mpg"     "mph"
"mr"      "ms"
##  [92] "msg"     "mss"     "mw"      "my"      "myrrh"
"myrrhs"  "myth"
##  [99] "myths"   "nj"      "nm"      "nth"     "ny"
"nymph"   "nymphs"
## [106] "pbx"     "pc"      "pct"     "pf"      "phys"
"pkg"     "pkwy"
## [113] "pl"      "ply"     "pm"      "pp"      "ppd"
"prs"     "pry"
## [120] "psf"     "psst"    "psych"   "psychs"  "pts"
```

```
"pygmy"   "pyx"
## [127] "qts"     "qty"     "rcpt"    "rd"      "rf"
"rh"      "rhythm"
## [134] "rhythms" "rn"      "rpm"     "sc"      "sd"
"sh"      "shy"
## [141] "shyly"   "sky"     "skys"    "sly"     "slyly"
"sn"      "sp"
## [148] "spry"    "spryly"  "spy"     "sr"      "ss"
"st"      "sty"
## [155] "stymy"   "styx"    "sylph"   "sylphs"  "sylphy"
"sync"    "synch"
## [162] "synchs"  "syncs"   "syzygy"  "tbs"     "tbsp"
"th"      "thy"
## [169] "thymy"   "tm"      "tmh"     "tnpk"    "tnt"
"tpk"     "try"
## [176] "tryst"   "trysts"  "tsp"     "tty"     "tv"
"tx"      "vc"
## [183] "vs"      "vt"      "why"     "whys"    "wk"
"wkly"    "wpm"
## [190] "wry"     "wryly"   "xx"      "xxv"     "xxx"
"xysts"   "yds"
## [197] "yr"     "yrs"    "zn"

# b.
Position(is.colour, words)

## [1] 4562

###############
#             #
# Exercise 7  #
#             #
###############

# a.
Find(function(x) x %% 1234 == 0, 10000:20000)

## [1] 11106

# b.
```

```
Find(function(x) x %% 1234 == 0, 10000:20000, right=TRUE)

## [1] 19744

###############
#             #
# Exercise 8  #
#             #
###############

library(babynames)
namesData <- split(babynames$name, babynames$year)

# a.
commonNames <- Reduce(intersect, namesData)

# b.
newestNames   <-   Reduce(setdiff,   namesData[-135],
init=namesData[[135]])

###############
#             #
# Exercise 9  #
#             #
###############

library(babynames)
namesData <- split(babynames$name, babynames$year)
moreThan3 <- function(x) nchar(x) > 3

namesLessThan4  <-  Map(Filter,  list(Negate(moreThan3)),
namesData)

################
#              #
# Exercise 10  #
#              #
################

library(babynames)
namesData <- split(babynames$name, babynames$year)
```

```
# a.
namesAfterSplit <- Map(strsplit, namesData, "")

# b.
namesAfterJoin   <-  Map(Map,  list(paste),  namesAfterSplit,
collapse="_")
```