S2S Lab 2 Task Solutions

1 Welcome!

2 Packages

3 Data Structures

3.1 Arrays

3.2 Matrices

Creating Matrices

Use the `array()` function and the `letters` vector to create a 5 × 5 matrix containing the letters of the alphabet in column-major order, up to "y".

```
array(data = letters, dim = c(5, 5))
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,] "a"  "f"  "k"  "p"  "u"
## [2,] "b"  "g"  "l"  "q"  "v"
## [3,] "c"  "h"  "m"  "r"  "w"
## [4,] "d"  "i"  "n"  "s"  "x"
## [5,] "e"  "j"  "o"  "t"  "y"
```

Create a 5 × 5 matrix containing the first 25 letters of the alphabet using `matrix()`. Fill in the elements in row-major order.

```
matrix(data = letters, nrow = 5, ncol = 5, byrow = TRUE)
```

```
## Warning in matrix(data = letters, nrow = 5, ncol = 5, byrow = TRUE): data
## length [26] is not a sub-multiple or multiple of the number of rows [5]
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,] "a"  "b"  "c"  "d"  "e"
## [2,] "f"  "g"  "h"  "i"  "j"
## [3,] "k"  "l"  "m"  "n"  "o"
## [4,] "p"  "q"  "r"  "s"  "t"
## [5,] "u"  "v"  "w"  "x"  "y"
```

Here, we have to use both `nrow =` and `ncol =` to force the matrix to be 5 × 5. Since there are 26 letters in the alphabet, `matrix()` will first try to repeat them so that they are all included if only one of the number of rows or columns is specified (e.g. resulting in 6 columns if only 5 rows is specified), rather than leaving letters out.

This is why we also see the warning telling us that the length of `letters` is not a multiple of 5.

Naming rows and columns

```
##           2017 2018 2019
## Edinburgh 5033 4899 4683
## Glasgow   6852 6548 6553
## Aberdeen  2402 2337 2260
## Dundee    1493 1488 1417
```

What is the code you would use to show the number of births in Glasgow in all 3 years?

There are several ways we could extract the data for these years. For example, we can specify that we want the row relating to Glasgow using either `"Glasgow"` or simply `2`.

To tell R that we want to see 2017, 2018 and 2019, we can simply leave the column entry blank (make sure to still have a comma between rows and columns within the square brackets i.e. `["Glasgow", ]`). R will then just return the entire row relating to Glasgow.

Alternatively, you could specify columns 1 to 3 using `1:3` or `c(1, 2, 3)`, or you could use the column names; `c("2017", "2018", "2019")`.

That is, any of the following lines of code can be used to show the number of births in Glasgow.

```r
births["Glasgow", ]
births[2, ]
births["Glasgow", 1:3]
births[2, 1:3]
births["Glasgow", c(1, 2, 3)]
births[2, c(1, 2, 3)]
births["Glasgow", c("2017", "2018", "2019")]
births[2, c("2017", "2018", "2019")]
```

```
## 2017 2018 2019
## 6852 6548 6553
```

3.2.1 Dimension reduction

3.2.2 Calculating Statistics

What is the standard deviation for the number of births in 2019?

We can calculate the standard deviation in all three years using the `apply()` function. Because we want to find the standard deviation in each column, we need to set `MARGIN = 2`. The `FUN`ction we should use is `sd`. That is;

```r
apply(X = births, MARGIN = 2, FUN = sd)
```

```
##     2017     2018     2019
## 2451.363 2326.924 2337.263
```

It is then easy to see that the standard deviation for 2019 is 2337.263.

Vector/Matric Multiplication

3.3 Factors

The results from a survey asking students whether statistics is the best subject are shown below. They were given a choice of "Agree", "Disagree" and "Unsure".

Student	Answer
Student 1	Agree
Student 2	Agree
Student 3	Agree
Student 4	Unsure
Student 5	Disgaree

Create and print a factor, called `survey`, which contains the answers of these five students as well as the levels of response they could have given.

There are several ways you could create this factor. Here, we first store the answers given in the vector `answers` and use the encoding 1="Agree", 2="Disagree" and 3="Unsure" (to save on typing!).

```
answers <- c(1, 1, 1, 3, 2)
```

Then we can create the factor `survey` using the `factor()` function.

```
survey <- factor(x = answers, levels = 1:3, labels = c("Agree", "Disagree", "Unsure"))
survey
```

```
## [1] Agree    Agree    Agree    Unsure   Disagree
## Levels: Agree Disagree Unsure
```

3.4 Data frames

```
percentage <- c(84, 76, 90, 53, 6, 67)
grade <- c("A", "A", "A", "C", "H", "B")
pass <- c(TRUE, TRUE, TRUE, TRUE, FALSE, TRUE)
ids <- c("ST002", "ST014", "ST089", "ST060", "ST034", "ST056")

performance <- data.frame(percentage, grade, pass,
                          stringsAsFactors = TRUE, row.names = ids)
```

Write code to extract only the percentage and the associated grade for the student with ID `ST014`?

There are many different ways we could specify the elements we want to extract. Any of the following lines of code will return the same output.

```
performance["ST014", c("percentage", "grade")]
performance["ST014", c(1, 2)]
performance["ST014", 1:2]
performance["ST014", -3]
performance[2, c("percentage", "grade")]
performance[2, c(1, 2)]
performance[2, 1:2]
performance[2, -3]
```

```
##       percentage grade
## ST014         76     A
```

The package `PASWR2` contains a data set called `WAIT`. What do the wait times saved in this data set relate to?

To find out more about what information a data set contains, we can use the `help()` function. Here, running the following code in R tells us that the wait times in `WAIT` are how long a statistician has had to wait for the bus each morning.

```
help("WAIT")
```

Write code to first view the top 5 rows of the data frame `WAIT` and then load it into your Environment.

To view the top 5 rows, we use the `head()` function and specify how many rows we want to see using the argument `n = 5`.

```
head(WAIT, n = 5)
```

```
##   minutes
## 1     8.0
## 2     2.1
## 3     3.8
## 4     8.6
## 5     7.3
```

To then load the data frame into our **Environment** tab, we use the function `data()`.

```
data("WAIT")
```

3.5 Lists