Tutorial Exercises Week 2

Question 1

Use R to calculate the principal cubed root of 64, $\sqrt[3]{64}$.

* Solution

The cubed root of 64 can also be written as $64^{\frac{1}{3}}$. We can calculate this in R with:

```
64^(1/3)
```

```
[1] 4
```

Question 2

Use R to calculate $\ln(e^5)$.

* Solution

We can calculate this with:

```
log(exp(5))
```

```
[1] 5
```

The reason we get 5 (the same as the input value) is because $\log(e^x) = x$ for all x.

Question 3

Use R to calculate $\log_4(64)$.

That is, take the log of 64 to the base 4.

* Solution

We can specify the base using the `base` argument in the `log()` function:

```r
log(64, base = 4)
```

```
[1] 3
```

The reason we get 3 is because we need to multiply 4 by itself exactly 3 times to get 64:

```r
4 * 4 * 4
```

```
[1] 64
```

Question 4

In R we create vectors with the `c()` function (the combine function). In a vector, all elements need to have the same type (like numerical, logical, character). If we combine elements of different types into a vector, the `c()` function will force elements to a common type.

What types are the following vectors?

- `c(1, 2, 3, TRUE, FALSE)`
- `c(1, 2, "3")`
- `c(TRUE, FALSE, "Yes", "No")`

* Solution

Let's create each of the following vector and check their classes:

```r
q4i <- c(1, 2, 3, TRUE, FALSE)
q4i
```

```
[1] 1 2 3 1 0
```

```r
class(q4i)
```

```
[1] "numeric"
```

R coerces all values to numeric because of the presence of both numeric and logical values. It coerces the `TRUE` to 1 and `FALSE` to 0.

```r
q4ii <- c(1, 2, "3")
q4ii
```

```
[1] "1" "2" "3"
```

```
class(q4ii)
```

```
[1] "character"
```

R coerces all to character because of the presence of both numeric and character values. It coerces the `1` to `"1"` and `2` to `"2"`.

```
q4iii <- c(TRUE, FALSE, "Yes", "No")
q4iii
```

```
[1] "TRUE"  "FALSE" "Yes"   "No"
```

```
class(q4iii)
```

```
[1] "character"
```

R coerces all to character because of the presence of both logical and character values. It coerces the `TRUE` to `"TRUE"` and `FALSE` to `"FALSE"`.

The hierarchy for conversion is logical $<$ numeric $<$ character.

Question 5

Consider the sequence $(1.0, 1.2, 1.4, 1.6, ..., 100)$.

- How many numbers are in the sequence?
- What is the 100th number in the sequence?
- What is the median value in the sequence?

* Solution

We can see the sequence is numbers from 1 to 100 in steps of 0.2. We can create this with:

```
x <- seq(from = 1, to = 100, by = 0.2)
```

To find the number of elements in `x` we use the `length()` function:

```
length(x)
```

```
[1] 496
```

To find the 100th element of `x` we can use indexing:

```
x[100]
```

```
[1] 20.8
```

To find the median value of `x` we can use the `median()` function:

```
median(x)
```

```
[1] 50.5
```

Question 6

Create the sequence:

$$(1, 1, 2, 2, 3, 3, 4, 4, 5, 5, ..., 98, 98, 99, 99, 100, 100)$$

Assign this sequence to the variable `x`.

Write a command to get the subset of this sequence with values exceeding 60. The output should be:

$$(61, 61, 62, 62, ..., 99, 99, 100, 100)$$

What is the average of this subsequence?

* Solution

We can this sequence using the `rep()` function with the `each` option:

```
x <- rep(1:100, each = 2)
x
```

```
  [1]   1   1   2   2   3   3   4   4   5   5   6   6   7   7   8   8   9   9
 [19]  10  10  11  11  12  12  13  13  14  14  15  15  16  16  17  17  18  18
 [37]  19  19  20  20  21  21  22  22  23  23  24  24  25  25  26  26  27  27
 [55]  28  28  29  29  30  30  31  31  32  32  33  33  34  34  35  35  36  36
 [73]  37  37  38  38  39  39  40  40  41  41  42  42  43  43  44  44  45  45
 [91]  46  46  47  47  48  48  49  49  50  50  51  51  52  52  53  53  54  54
[109]  55  55  56  56  57  57  58  58  59  59  60  60  61  61  62  62  63  63
[127]  64  64  65  65  66  66  67  67  68  68  69  69  70  70  71  71  72  72
[145]  73  73  74  74  75  75  76  76  77  77  78  78  79  79  80  80  81  81
[163]  82  82  83  83  84  84  85  85  86  86  87  87  88  88  89  89  90  90
[181]  91  91  92  92  93  93  94  94  95  95  96  96  97  97  98  98  99  99
[199] 100 100
```

To get the values exceeding 60 we can use logical indexing:

```
x[x > 60]
```

```
 [1]  61  61  62  62  63  63  64  64  65  65  66  66  67  67  68  68  69  69  70
[20]  70  71  71  72  72  73  73  74  74  75  75  76  76  77  77  78  78  79  79
[39]  80  80  81  81  82  82  83  83  84  84  85  85  86  86  87  87  88  88  89
[58]  89  90  90  91  91  92  92  93  93  94  94  95  95  96  96  97  97  98  98
[77]  99  99 100 100
```

To get the average of this we use the `mean()` function on this subsequence:

```
mean(x[x > 60])
```

```
[1] 80.5
```

Question 7

Using the example logical vectors `a` and `b` from the book:

```
a <- c(TRUE, TRUE, FALSE, FALSE)
b <- c(TRUE, FALSE, TRUE, FALSE)
```

Which command returns the elements where `a` and `b` are not both `TRUE`?

* Solution

We use `!a` to get when `a` is not `TRUE`. Similarly, we use `!b` to get when `b` is not `TRUE`. To see when `a` and `b` are both not `TRUE` we use the logical AND operator `&` with `!a` and `!b`:

```
a <- c(TRUE, TRUE, FALSE, FALSE)
b <- c(TRUE, FALSE, TRUE, FALSE)

!a & !b
```

```
[1] FALSE FALSE FALSE  TRUE
```

`a` and `b` are both not `TRUE` only in the 4th element, when both `a` and `b` are `FALSE`.

Question 8

Download the file rotterdam-airbnb.csv.

This contains data on Rotterdam Airbnb listings.

Read it into R. What is the average price of a night's stay in the data?

If the data is loaded in R as `df`, you can get the vector of prices with the command `df$price`.

* Solution

First we create a folder for our project and put the `rotterdam-airbnb.csv` file into that folder. Then we start a new project in RStudio and point RStudio to where the folder is. RStudio will open a new session and we will be in the directory of the project which has the dataset in it. We should be able to see the `rotterdam-airbnb.csv` file in the "Files" tab in RStudio.

```
df <- read.csv("rotterdam-airbnb.csv")
mean(df$price)
```

```
[1] 162.6658
```