# S2S Lab 1 Task Solutions

## 1 Welcome to S2S Labs

### 1.1 Mentimeter

## 2 Vectors

### 2.1 Addition

**Create and save the vector $a = \begin{bmatrix} 2 & 2 & -1 \end{bmatrix}^\intercal$ and scalar $b = 2$, then add these together and save the result as a new vector called `c` and print the contents of `c` in the console.**

```
a <- c(2, 2, -1)
b <- 2

c <- a + b
c
```

```
[1] 4 4 1
```

### 2.2 Logical operators

**Write some code to show:**

- **which elements of `s` and `t` are equal to each other**
- **which elements of `s` are less than the corresponding element in `t`**
- **which elements of `s` and `t` are unequal**

When using logical operators with vectors, R will compare the vectors elementwise meaning the first elements of each vector will be compared to each other, then the second elements and so on.

**Equality**:

```
s == t
```

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

This tells us that the first element of `s` is equal to the first element of `t` (they are both the value 2), whereas the second elements of `s` and `t` are not equal and neither are the third elements.

**Less than**:

```
s < t
```

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

The first element of `s` is not less than the first elements of `t` (that is $2 < 2$ is not a true statement) so this returns `FALSE`. The statement `s < t` is `TRUE` for the third elements because the third element of `s` is 1 and for `t` it is 4 which equates to the statement $1 < 4$.

**Inequality**:

```
s != t
```

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

This return the opposite of `s == t`. The first elements of `s` and `t` are both 2 so they are equal meaning `!=` will return `FALSE`. Since the second elements are 4 and 3 respectively, these are unequal so `!=` returns `TRUE`.

Try using some of the other operators listed above to see if they return the vectors you expect.

**Create the vectors $u = \begin{bmatrix}4 & 4 & 1\end{bmatrix}^\intercal$ and $v = \begin{bmatrix}1 & 0 & 5\end{bmatrix}^\intercal$. Write code to check:**

- **whether the first elements of `u+v` and `x+y` (defined in the previous section) are the same.**
- **if the third element of `u+v` is the same as the second element of `x+y`.**

```
u <- c(4, 4, 1)
v <- c(1, 0, 5)
```

**Equality of the first elements:**

```
(u + v)[1] == (x + y)[1]
```

```
## [1] TRUE
```

Here we have wrapped both `u+v` and `x+y` in brackets first so that we can then index the resulting vector. Alternatively, you could save `u+v` and `x+y` as new vectors, called something different, using the `<-` operator.

`u+v` is equal to $\begin{bmatrix}5 & 4 & 6\end{bmatrix}^\intercal$ so its first element is 5. `x+y` is equal to $\begin{bmatrix}5 & 7 & 4\end{bmatrix}^\intercal$ so its first element is also 5. It makes sense then that a `TRUE` statement is returned when checking the equality of the first elements of these two vectors.

**Equality of the third and second elements:**

```
(u + v)[3] == (x + y)[2]
```

```
## [1] FALSE
```

Similarly, we can see that the third entry of `u+v` is 6 and the second entry of `x+y` is 7. Since these are clearly not equal we would expect R to return a `FALSE` statement when checking their equality, which we can see above.

## 2.3 Other types of vectors

**What type of vector is `w`?**

```
typeof(w)
```

```
## [1] "logical"
```

Using `typeof()`, we can see that `w` is stored by R as a logical vector.

**Create a new vector, called `combined`, which is a combination of the logical vector `w` and the character vector `animals`. What type of vector is this?**

```
combined <- c(w, animals)
```

```
## [1] "FALSE" "TRUE"  "FALSE" "dog"   "sheep" "cow"   "horse"
```

```
typeof(combined)
```

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

The resulting vector is $\begin{bmatrix}\text{“FALSE"} & \text{“TRUE"} & \text{“FALSE"} & \text{“dog"} & \text{“sheep"} & \text{“cow"} & \text{“horse"}\end{bmatrix}^\intercal$ which we can see is stored by R as a character vector. Each element is now considered to be a string of text (note the quotation marks around each element).

## 2.4 Indexing vectors

**Write some code that would remove the third and fourth elements from the vector `animals`.**

There are several ways this code could be written. We can remove the third and fourth entries through either of the following lines of code.

```
animals[-c(3, 4)]
```

```
## [1] "dog"   "sheep"
```

```
animals[c(-3, -4)]
```

```
## [1] "dog"   "sheep"
```

Alternatively, because animals has four elements, removing the third and fourth elements is equivalent to extracting the first and second elements. Therefore we can achieve the same results using the following code.

```
animals[c(1, 2)]
```

```
## [1] "dog"   "sheep"
```

**This task has three steps, so you should write three lines of code.**

- **Create the vector $\begin{bmatrix}1 & 1 & 0 & 1\end{bmatrix}^\intercal$ and call it `binary`**
- **Change `binary` to be a logical vector and save this as a new vector called `logical`**
- **Extract the first and third elements of this vector `logical`.**

```
binary <- c(1, 1, 0, 1)
logical <- as.logical(binary)

logical[c(1, 3)]
```

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

In the above code we have chosen to extract the first and third elements. Alternatively, we could remove the second and fourth elements instead.

```
logical[-c(2, 4)]
```

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

## 2.5 Sequences

**Write some code to create the sequence 4.10, 4.15, 4.20, 4.25, 4.30, 4.35, 4.40 using the function `seq()`**

There are three different ways we could generate this sequence using the `seq()` function.

```
seq(from = 4.10, to = 4.40, by = 0.05)
```

```
## [1] 4.10 4.15 4.20 4.25 4.30 4.35 4.40
```

```
seq(from = 4.10, to = 4.40, length.out = 7)
```

```
## [1] 4.10 4.15 4.20 4.25 4.30 4.35 4.40
```

```
seq(from = 4.10, by = 0.05, length.out = 7)
```

```
## [1] 4.10 4.15 4.20 4.25 4.30 4.35 4.40
```

## 2.6 Repeating constants

**Write some code to create each of the following repeating sequences using `rep()`:**

- **3, 4, 5, 3, 4, 5, 3, 4, 5**
- **2, 2, 4, 4, 6, 6**
- **blue, blue, blue, red, red**

```
rep(3:5, times = 3)
```

```
## [1] 3 4 5 3 4 5 3 4 5
```

```
rep(c(2, 4, 6), each = 2)
```

```
## [1] 2 2 4 4 6 6
```

```
rep(c("blue", "red"), times=c(3, 2))
```

```
## [1] "blue" "blue" "blue" "red"  "red"
```

**Write some code to generate the sequence 2, 3, 3, 4, 5, 5, 6, 7, 7 using both the `seq()` and `rep()` functions.**

```
rep(x = 2:7,
    times = rep(x = c(1, 2), times = 3))
```

```
## [1] 2 3 3 4 5 5 6 7 7
```

## 2.7 Filtering vectors

**Write some code to extract the elements of the vector `z` which are greater than 5.**

```
z[z > 5]
```

```
## [1] 8
```

**Extract the elements of the vector `z` which are greater than 5 using the `subset()` function.**

```
subset(x = z, subset = (z > 5))
```

```
## [1] 8
```

## 2.8 Getting help