# STA 326 2.0/ ASP 454 2.0 Programming and Data Analysis with R
# Tutorial 1

## 1 Creating vectors

1. Write R codes to create following vectors

   `(a) 5, 12, 32, 50, 100000`

   `(b) 1, 2, 3, ..., 100.`

   `(c) 2, 4, 6, 8, ..., 100.`

   `(d) vector (3, 3, 3, . . . 3, 6, 6, . . . 6, 9, 9, 9, . . . 9), where there are 10 occurrences of`

2. Generate a sequence using the code `seq(from=1, to=10, by=1)`. What other ways can you generate the same sequence?

3. Using the function `rep()` , create the below sequence `1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4`

4. Create a vector that shows the square root the integers from 1 to 100.

5. Create a vector with elements from 1 to 100 incrementing by 0.4. Your output should look like the sample output given below.

```
  [1]  1.0  1.4  1.8  2.2  2.6  3.0  3.4  3.8  4.2  4.6  5.0  5.4  5.8  6.2  6.6
 [16]  7.0  7.4  7.8  8.2  8.6  9.0  9.4  9.8 10.2 10.6 11.0 11.4 11.8 12.2 12.6
 [31] 13.0 13.4 13.8 14.2 14.6 15.0 15.4 15.8 16.2 16.6 17.0 17.4 17.8 18.2 18.6
 [46] 19.0 19.4 19.8 20.2 20.6 21.0 21.4 21.8 22.2 22.6 23.0 23.4 23.8 24.2 24.6
 [61] 25.0 25.4 25.8 26.2 26.6 27.0 27.4 27.8 28.2 28.6 29.0 29.4 29.8 30.2 30.6
 [76] 31.0 31.4 31.8 32.2 32.6 33.0 33.4 33.8 34.2 34.6 35.0 35.4 35.8 36.2 36.6
 [91] 37.0 37.4 37.8 38.2 38.6 39.0 39.4 39.8 40.2 40.6 41.0 41.4 41.8 42.2 42.6
[106] 43.0 43.4 43.8 44.2 44.6 45.0 45.4 45.8 46.2 46.6 47.0 47.4 47.8 48.2 48.6
[121] 49.0 49.4 49.8 50.2 50.6 51.0 51.4 51.8 52.2 52.6 53.0 53.4 53.8 54.2 54.6
[136] 55.0 55.4 55.8 56.2 56.6 57.0 57.4 57.8 58.2 58.6 59.0 59.4 59.8 60.2 60.6
[151] 61.0 61.4 61.8 62.2 62.6 63.0 63.4 63.8 64.2 64.6 65.0 65.4 65.8 66.2 66.6
[166] 67.0 67.4 67.8 68.2 68.6 69.0 69.4 69.8 70.2 70.6 71.0 71.4 71.8 72.2 72.6
[181] 73.0 73.4 73.8 74.2 74.6 75.0 75.4 75.8 76.2 76.6 77.0 77.4 77.8 78.2 78.6
[196] 79.0 79.4 79.8 80.2 80.6 81.0 81.4 81.8 82.2 82.6 83.0 83.4 83.8 84.2 84.6
[211] 85.0 85.4 85.8 86.2 86.6 87.0 87.4 87.8 88.2 88.6 89.0 89.4 89.8 90.2 90.6
[226] 91.0 91.4 91.8 92.2 92.6 93.0 93.4 93.8 94.2 94.6 95.0 95.4 95.8 96.2 96.6
[241] 97.0 97.4 97.8 98.2 98.6 99.0 99.4 99.8
```

6. Observe the differences in running the following codes.

```
vec1 <- 1.8:20.8
vec1
```

```
 [1]  1.8  2.8  3.8  4.8  5.8  6.8  7.8  8.8  9.8 10.8 11.8 12.8 13.8 14.8 15.8
[16] 16.8 17.8 18.8 19.8 20.8
```

```
vec2 <- 1.8:30
vec2
```

```
 [1]  1.8  2.8  3.8  4.8  5.8  6.8  7.8  8.8  9.8 10.8 11.8 12.8 13.8 14.8 15.8
[16] 16.8 17.8 18.8 19.8 20.8 21.8 22.8 23.8 24.8 25.8 26.8 27.8 28.8 29.8
```

## 2 Object classes and type of objects

7. Use `typeof` to identify the storage mode of the following objects and `class` to identify object classes.

```
a <- c("MON", "TUES", "WED", "THUR", "FRI")
```

```
b <- c(1, 2, 3, 4, 5)
```

```
c <- c(1L, 2L, 3L, 4L, 5L)
```

```
d <- c(TRUE, FALSE, TRUE, TRUE)
```

```
e <- c(2+3i, 1+2i, 5+3i)
```

```
f <- c("MON", TRUE, 1, 1L)
```

8. Explore comment on the output of following vector functions.

```
a1 <- vector("numeric", 8)
a2 <- vector("complex", 8)
a3 <- vector("logical", 8)
a4 <- vector("character", 8)

b1 <- numeric(8)
b2 <- complex(8)
b3 <- logical(8)
b4 <- character(8)
```

## 3 Subsetting of vectors

9. Consider the vector `x`.

```
x <- 1:10
```

What does each of the following codes do?

```
x[3]

x[c(2, 4)]

x[-1]

x[c(2, -4)]

x[c(2.4, 3.54)]
```

10. Consider the following vector

```
  [1] -0.97448987 -1.27461307  0.48801800  0.49141295 -0.67692483  1.34014254
  [7]  2.18583063 -0.17553252 -1.63483137  0.03024727 -2.11800235  0.18654600
 [13]  1.39701400  0.14609591 -1.83088596 -2.36168175 -2.02598608  0.67525542
 [19]  1.15826732 -0.37223351 -0.08631292 -0.46437630 -0.78819679  0.04924904
 [25] -0.78357858  1.05806879  0.13313826  0.76659792  0.12350731 -0.55793273
 [31]  1.03673831  1.38740632 -1.21604544 -1.27585619 -1.27849850 -0.46260147
 [37]  1.11404898 -0.60577065 -0.05419039  0.82942191 -0.11990169 -1.14907057
 [43] -1.06150768 -0.26883482  1.43786263 -0.76171195 -0.14256348  0.79500907
 [49] -0.09093005  1.68983869  0.48986626 -0.49533952 -0.76212444 -0.45888922
 [55] -0.64235313  0.32436572 -0.86661285  0.12504993 -1.30756345 -1.20334635
 [61] -0.59564966  1.62625440  0.59231311  1.42604105 -0.23446921 -0.82578278
 [67] -0.79852124  0.88041040  0.65535406  0.20698931 -0.99832265 -2.12806683
 [73]  1.43832441 -0.18321060 -0.34885211 -0.76699277  0.75013339  0.29268751
 [79] -1.08067845  0.87199830 -1.11618678 -0.60850387 -0.30167388 -0.89210264
 [85] -1.64258734 -0.67589617  0.45925549  0.54833515 -0.17458499 -1.01837339
 [91]  1.26295194  0.07384001  0.75025707 -0.33222339 -0.14743703 -0.46438730
 [97] -1.24863294  1.48870732  0.71412848 -0.71444642
```

You can use the following command to create the above vector.

```
set.seed(1762021)
st_normal <- rnorm(100)
st_normal
```

Write an R code to drop the elements corresponds to the positions multiply of 10 (10, 20, 30, …).

# 4 Filtering vectors based on conditions

11. Consider the vector

```
x <- c(80, 39, NA, 51, 51, 11, NA, NA, NA, 100, 80, 70)
```

```
(a) Write an R code to extract non-missing values in `x`.
```

```
(b) Write an R code to extract missing values and odd-numbers in `x`
```

(c) Write an R code to extract odd numbers on `x`

(d) Which values of `x` are NOT in the set `1:50`

## 5 Modify a vector

12. Consider the following vector age which includes the age of 10 individuals

```
age <- c(20, 30, 40, 41, 32, 32, 25,  NA, NA, -4, -6, 9999, 10000)
```

(a) Convert all negative values to `NA`.

(b) Convert all values of `age` that are NOT from 10 to 100 and calculate the mean of valid responses.

13. Consider the following vector of 100 random numbers generated from the standard normal distribution. You can use the code given below to reproduce the vector b.

```
set.seed(17620212)
b <- rnorm(100)
b
```

```
  [1]  0.589528488 -0.662937204  0.238279278  0.183757174 -0.002364399
  [6]  0.289002107  0.258796402  0.982174159  0.378628085  0.015035037
 [11] -1.203312799  1.510436562  0.219368378 -0.642429444 -0.373969124
 [16] -0.239829685 -0.186344734  0.517975563 -1.256355393  1.067433297
 [21]  1.035128935 -1.016002843  0.830122365  0.427420672  0.170429825
 [26] -0.001345883 -1.022893025 -0.908602635  0.502535054  0.315929086
 [31]  1.294309571 -0.303323749 -0.322819573 -1.377566743  2.714915313
 [36] -0.512573266  1.342424819 -0.457104082 -1.593015886 -0.202338403
 [41]  1.079678527  0.456102666  1.504041202  0.378318229 -0.289765109
 [46]  1.019989890  0.665591385 -1.076213455  0.272375584  0.545493842
 [51]  0.052391342 -0.402364688  0.152662598 -1.486745812  0.102018231
 [56] -0.024357072  0.068276667  0.075642814  0.379600455 -0.988308679
 [61]  0.701330674 -0.491165150  1.494498791 -1.773934043 -0.460454009
 [66]  0.752256616  0.039189614 -0.939203562 -0.419716046  0.084067026
 [71] -1.081303093  0.780827145  0.207575277  0.733234796 -0.660969465
 [76]  1.649316796  0.491550464 -0.864054075 -0.919522275 -0.727913488
 [81]  1.197400462 -1.645388340  1.704924934 -1.650667045  0.377823148
 [86]  1.436377659 -1.143144414  0.789086285  1.049357974  2.163786809
 [91]  1.626306920  1.317779758  1.647449733  0.588881226 -0.177613835
 [96]  0.081191404  0.093051240  1.202918954 -1.783424334  0.725816313
```

(a) Change the first five values in the vector to 1.

(b) Change the last five values in the vector to 0.

(c) Assign all values grater than 0.5 to 1 and all values less than 0.5 to 0.

(d) Recode the 0 values to "MALE" and others to "FEMALE"