

STA 517 3.0 Programming and Statistical Computing with R

Tutorial 1

1 Creating vectors

1. Create the following vectors:

(a) 1, 2, 3, ..., 100

```
1:100
```

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

```
seq(1, 100)
```

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

(b) 2, 4, 6, 8, ..., 100

```
seq(2, 100, by=2)
```

```
[1]  2  4  6  8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38
[20] 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76
[39] 78 80 82 84 86 88 90 92 94 96 98 100
```

(c)

```
[1] 1.000000  1.497487  1.994975  2.492462  2.989950  3.487437
[7] 3.984925  4.482412  4.979899  5.477387  5.974874  6.472362
[13] 6.969849  7.467337  7.964824  8.462312  8.959799  9.457286
[19] 9.954774 10.452261 10.949749 11.447236 11.944724 12.442211
[25] 12.939698 13.437186 13.934673 14.432161 14.929648 15.427136
[31] 15.924623 16.422111 16.919598 17.417085 17.914573 18.412060
```

```

[37] 18.909548 19.407035 19.904523 20.402010 20.899497 21.396985
[43] 21.894472 22.391960 22.889447 23.386935 23.884422 24.381910
[49] 24.879397 25.376884 25.874372 26.371859 26.869347 27.366834
[55] 27.864322 28.361809 28.859296 29.356784 29.854271 30.351759
[61] 30.849246 31.346734 31.844221 32.341709 32.839196 33.336683
[67] 33.834171 34.331658 34.829146 35.326633 35.824121 36.321608
[73] 36.819095 37.316583 37.814070 38.311558 38.809045 39.306533
[79] 39.804020 40.301508 40.798995 41.296482 41.793970 42.291457
[85] 42.788945 43.286432 43.783920 44.281407 44.778894 45.276382
[91] 45.773869 46.271357 46.768844 47.266332 47.763819 48.261307
[97] 48.758794 49.256281 49.753769 50.251256 50.748744 51.246231
[103] 51.743719 52.241206 52.738693 53.236181 53.733668 54.231156
[109] 54.728643 55.226131 55.723618 56.221106 56.718593 57.216080
[115] 57.713568 58.211055 58.708543 59.206030 59.703518 60.201005
[121] 60.698492 61.195980 61.693467 62.190955 62.688442 63.185930
[127] 63.683417 64.180905 64.678392 65.175879 65.673367 66.170854
[133] 66.668342 67.165829 67.663317 68.160804 68.658291 69.155779
[139] 69.653266 70.150754 70.648241 71.145729 71.643216 72.140704
[145] 72.638191 73.135678 73.633166 74.130653 74.628141 75.125628
[151] 75.623116 76.120603 76.618090 77.115578 77.613065 78.110553
[157] 78.608040 79.105528 79.603015 80.100503 80.597990 81.095477
[163] 81.592965 82.090452 82.587940 83.085427 83.582915 84.080402
[169] 84.577889 85.075377 85.572864 86.070352 86.567839 87.065327
[175] 87.562814 88.060302 88.557789 89.055276 89.552764 90.050251
[181] 90.547739 91.045226 91.542714 92.040201 92.537688 93.035176
[187] 93.532663 94.030151 94.527638 95.025126 95.522613 96.020101
[193] 96.517588 97.015075 97.512563 98.010050 98.507538 99.005025
[199] 99.502513 100.000000

```

```
seq(1, 100, length=200)
```

```

[1] 1.000000 1.497487 1.994975 2.492462 2.989950 3.487437
[7] 3.984925 4.482412 4.979899 5.477387 5.974874 6.472362
[13] 6.969849 7.467337 7.964824 8.462312 8.959799 9.457286
[19] 9.954774 10.452261 10.949749 11.447236 11.944724 12.442211
[25] 12.939698 13.437186 13.934673 14.432161 14.929648 15.427136
[31] 15.924623 16.422111 16.919598 17.417085 17.914573 18.412060
[37] 18.909548 19.407035 19.904523 20.402010 20.899497 21.396985
[43] 21.894472 22.391960 22.889447 23.386935 23.884422 24.381910
[49] 24.879397 25.376884 25.874372 26.371859 26.869347 27.366834
[55] 27.864322 28.361809 28.859296 29.356784 29.854271 30.351759
[61] 30.849246 31.346734 31.844221 32.341709 32.839196 33.336683
[67] 33.834171 34.331658 34.829146 35.326633 35.824121 36.321608
[73] 36.819095 37.316583 37.814070 38.311558 38.809045 39.306533
[79] 39.804020 40.301508 40.798995 41.296482 41.793970 42.291457
[85] 42.788945 43.286432 43.783920 44.281407 44.778894 45.276382
[91] 45.773869 46.271357 46.768844 47.266332 47.763819 48.261307
[97] 48.758794 49.256281 49.753769 50.251256 50.748744 51.246231
[103] 51.743719 52.241206 52.738693 53.236181 53.733668 54.231156
[109] 54.728643 55.226131 55.723618 56.221106 56.718593 57.216080
[115] 57.713568 58.211055 58.708543 59.206030 59.703518 60.201005
[121] 60.698492 61.195980 61.693467 62.190955 62.688442 63.185930
[127] 63.683417 64.180905 64.678392 65.175879 65.673367 66.170854
[133] 66.668342 67.165829 67.663317 68.160804 68.658291 69.155779

```

```
[139] 69.653266 70.150754 70.648241 71.145729 71.643216 72.140704
[145] 72.638191 73.135678 73.633166 74.130653 74.628141 75.125628
[151] 75.623116 76.120603 76.618090 77.115578 77.613065 78.110553
[157] 78.608040 79.105528 79.603015 80.100503 80.597990 81.095477
[163] 81.592965 82.090452 82.587940 83.085427 83.582915 84.080402
[169] 84.577889 85.075377 85.572864 86.070352 86.567839 87.065327
[175] 87.562814 88.060302 88.557789 89.055276 89.552764 90.050251
[181] 90.547739 91.045226 91.542714 92.040201 92.537688 93.035176
[187] 93.532663 94.030151 94.527638 95.025126 95.522613 96.020101
[193] 96.517588 97.015075 97.512563 98.010050 98.507538 99.005025
[199] 99.502513 100.000000
```

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

```
seq(from=1, to=10, by=1)
```

```
[1] 1 2 3 4 5 6 7 8 9 10
```

```
1:10
```

```
[1] 1 2 3 4 5 6 7 8 9 10
```

```
c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

```
[1] 1 2 3 4 5 6 7 8 9 10
```

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

```
rep(1:4, 3)
```

```
[1] 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.

```
sqrt(1:100)
```

```
[1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490 2.645751
[8] 2.828427 3.000000 3.162278 3.316625 3.464102 3.605551 3.741657
[15] 3.872983 4.000000 4.123106 4.242641 4.358899 4.472136 4.582576
[22] 4.690416 4.795832 4.898979 5.000000 5.099020 5.196152 5.291503
[29] 5.385165 5.477226 5.567764 5.656854 5.744563 5.830952 5.916080
[36] 6.000000 6.082763 6.164414 6.244998 6.324555 6.403124 6.480741
[43] 6.557439 6.633250 6.708204 6.782330 6.855655 6.928203 7.000000
[50] 7.071068 7.141428 7.211103 7.280110 7.348469 7.416198 7.483315
[57] 7.549834 7.615773 7.681146 7.745967 7.810250 7.874008 7.937254
[64] 8.000000 8.062258 8.124038 8.185353 8.246211 8.306624 8.366600
[71] 8.426150 8.485281 8.544004 8.602325 8.660254 8.717798 8.774964
[78] 8.831761 8.888194 8.944272 9.000000 9.055385 9.110434 9.165151
[85] 9.219544 9.273618 9.327379 9.380832 9.433981 9.486833 9.539392
[92] 9.591663 9.643651 9.695360 9.746794 9.797959 9.848858 9.899495
[99] 9.949874 10.000000
```

5. 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

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

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

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

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

```
[1] "double"
```

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

```
[1] "integer"
```

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

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

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

```
[1] "complex"
```

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

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

7. 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)

```

8. Consider the vector

```

set.seed(32020)
st_normal <- rnorm(100)
st_normal

```

```

 [1]  0.18183635 -0.92262020  2.06110995 -1.50040396 -1.69529463  2.45410426
 [7]  0.16552699 -2.20702891 -0.21274657 -0.69387976 -0.67516314  1.03136276
[13]  0.77649171  0.60913641 -1.06664784  0.34027083 -0.47879695 -0.40281847
[19] -1.12500580 -0.79235873 -0.89371755 -2.72593829  0.99052081 -0.53966792
[25]  2.44848942  1.82337921 -0.52409631 -2.52099047 -0.01338390 -0.67771367
[31] -0.26224412 -1.96067034  0.03172268 -0.83045197  1.60051305  0.04106971
[37]  0.93303006 -1.31390340 -0.25427286 -0.61430209 -0.09897693  0.33713741
[43]  0.45989743 -0.79752346 -0.77387974 -0.57871649 -1.24023942 -1.74035257
[49] -0.02742062 -2.21931034  0.23715755 -0.47101092 -0.22116294 -1.45243410
[55]  0.27650330 -1.76656058  0.01328862 -1.30263545  1.20788668  1.47504605
[61] -2.19540879  0.44796633  0.39314554 -3.15206211 -0.32687439 -0.54550496
[67]  1.39978830 -2.19770996  1.46683852 -1.19686302  0.87487978 -0.83723410
[73]  1.37510059 -0.80996752  0.56198382  0.40264681  0.13343941 -0.05576293
[79]  1.66654211 -0.78997663  0.29758171  0.36613867  0.80338650 -1.43640458
[85] -0.56015981 -0.12409835 -0.75476839  0.32283051  1.46941104 -0.30940270
[91] -1.14718708 -0.93229533  0.06524165 -0.20590515 -0.69251943  0.93134043
[97]  0.28856808  1.04544874  0.24806814  0.22931507

```

Drop the elements corresponds to the positions multiply of 10 (10, 20, 30, ...)

```

st_normal[-seq(1, 100, by=10)]

```

```

 [1] -0.92262020  2.06110995 -1.50040396 -1.69529463  2.45410426  0.16552699
 [7] -2.20702891 -0.21274657 -0.69387976  1.03136276  0.77649171  0.60913641
[13] -1.06664784  0.34027083 -0.47879695 -0.40281847 -1.12500580 -0.79235873
[19] -2.72593829  0.99052081 -0.53966792  2.44848942  1.82337921 -0.52409631
[25] -2.52099047 -0.01338390 -0.67771367 -1.96067034  0.03172268 -0.83045197
[31]  1.60051305  0.04106971  0.93303006 -1.31390340 -0.25427286 -0.61430209
[37]  0.33713741  0.45989743 -0.79752346 -0.77387974 -0.57871649 -1.24023942
[43] -1.74035257 -0.02742062 -2.21931034 -0.47101092 -0.22116294 -1.45243410
[49]  0.27650330 -1.76656058  0.01328862 -1.30263545  1.20788668  1.47504605
[55]  0.44796633  0.39314554 -3.15206211 -0.32687439 -0.54550496  1.39978830
[61] -2.19770996  1.46683852 -1.19686302 -0.83723410  1.37510059 -0.80996752
[67]  0.56198382  0.40264681  0.13343941 -0.05576293  1.66654211 -0.78997663
[73]  0.36613867  0.80338650 -1.43640458 -0.56015981 -0.12409835 -0.75476839
[79]  0.32283051  1.46941104 -0.30940270 -0.93229533  0.06524165 -0.20590515
[85] -0.69251943  0.93134043  0.28856808  1.04544874  0.24806814  0.22931507

```

9. Create a vector with elements from 1 to 100 incrementing by 0.4

```
seq(1, 100, by=0.4)
```

```
[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
```

10. 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)]
```

3 Filtering vectors based on conditions

11. Consider the vector

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

Write an R code to extract non-missing values in x

```
x[!is.na(x)]
```

```
[1] 80 39 51 51 11 100 80 70
```

Write an R code to extract missing values and odd-numbers in x

```
y <- x%%2  
y
```

```
[1] 0 1 NA 1 1 1 NA NA NA 0 0 0
```

```
x[y==1]
```

```
[1] 39 NA 51 51 11 NA NA NA
```

Write an R code to extract odd numbers on x

```
z <- x[y==1]  
z[!is.na(z)]
```

```
[1] 39 51 51 11
```

Which values of x are NOT in the set 1:50

```
which(x %in% 1:50)
```

```
[1] 2 6
```

4 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)
```

i. Convert all negative values to ``NA``.

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

15. Consider the following vector of 100 random numbers generated from the standard normal distribution.

- i. Change the first five values in the vector to 1.
- ii. Change the last five values in the vector to 0.
- iii. Assign all values greater than 0.5 to 1 and all values less than 0.5 to 0.
- iv. Recode the 0 values to "MALE" and others to "FEMALE"