BMI 713: Computational Statistical for Biomedical Sciences

Assignment 1

September 9, 2010 (due Sept 16 for Part 1; Sept 23 for Part 2 and 3)

1 Basic

1. Use the `:` operator to create the vector $(1, 2, 3, 4, 5)$. Store this vector in a variable.

   ```
   v <- 1:5
   ```

2. Extend the vector you just created with the values $(6, 7, 8)$. Use the `c` function.

   ```
   v <- c(v, 6, 7, 8)
   ```

3. Shrink the vector you just created by removing the first element. One could also use the `[]` operators with a negative index to remove an element.

   ```
   v <- v[-1]
   ```

4. Set the first value in the remaining vector to 9.

   ```
   v[1] <- 9
   ```

5. Decrease the value of every element in the vector by 1. Do this using a single arithmetic operation.

   ```
   v <- v - 1
   ```

6. Create a vector v of length 20 such that all the elements of v are 1. Hint: use the function `rep()` (Can you do the same thing without using the function `rep()`?).

   ```
   rep(1, 20)
   ```

7. Try `rep(1:4,each=2)` and `rep(1:4,times=2)`. Are the results the same? If not, what is the difference?

   ```
   > rep(1:4, each=2)
   [1] 1 1 2 2 3 3 4 4
   > rep(1:4, times=2)
   [1] 1 2 3 4 1 2 3 4
   ```

8. Use R to calculate $\sqrt{2}$, 2^{10}, $\log(10)$, e^2, $\cos(\pi)$, $\sin(\pi)$ (Note: π in R is called `pi`).

   ```
   > sqrt(2)
   [1] 1.414214
   > 2^10
   [1] 1024
   > log(10)
   [1] 2.302585
   > exp(2)
   [1] 7.389056
   > cos(pi)
   [1] -1
   > sin(pi)
   [1] 1.224606e-16
   ```

9. Given a vector `v = (1,2,3,4,5)`. What happens if you try `1/v`? How about `exp(v)`? How about `v*v)`?

```
> 1/v
[1] 1.0000000 0.5000000 0.3333333 0.2500000 0.2000000
> exp(v)
[1]   2.718282   7.389056  20.085537  54.598150 148.413159
> v*v
[1]  1  4  9 16 25
```

10. Create a vector `v=(2,2,5,7,9,3,4)`. Try `v[v==2]`, `v[v>2]`, `v[v=<5]`. What are the results?

```
> v <- c(2,2,5,7,9,3,4)
> v[v==2]     # The elements in v equal to 2
[1] 2 2
> v[v>2]      # The elements in v greater than 2
[1] 5 7 9 3 4
> v[v<=5]     # The elements in v less than or equal to 5
[1] 2 2 5 3 4
```

11. Create a vector `v = (2,3,6,11,18)`. What happens if you try `diff(v)` and `diff(v,lag=2)`?

```
> v <- c(2,3,6,11,18)
> diff(v)      # The difference between each successive value in v
[1] 1 3 5 7
> diff(v, lag=2) # The difference between values 1 and 3, 2 and 4, etc.
[1]  4  8 12
```

12. Create a vector `v = (15.9,21.4,19.9,21.9,20.0,16.5,17.9,17.5)`. What is the smallest value in v? What is the largest value? How many are greater than 18? Make a new vector with only those bigger than 18.

```
> v <- c(15.9,21.4,19.9,21.9,20.0,16.5,17.9,17.5)
> min(v)
[1] 15.9
> max(v)
[1] 21.9
> length(v[v > 18])
[1] 4
> w <- v[v > 18]
> w
[1] 21.4 19.9 21.9 20.0
```

13. What does R return if you try `1/0`? How about `1/0+1/0`, `0/0`, `1/0-1/0`? (Note: `Inf` means infinity and `NaN` means "not a number".)

```
> 1/0
[1] Inf
> 1/0 + 1/0
[1] Inf
> 0/0
[1] NaN
> 1/0 - 1/0
[1] NaN
```

2 Intermediate

1. Write a program to print out the numbers 1,. . .,10, separated by tab or a new line. Compute the sum of squares of those numbers.

```
> # Print the numbers in 1:10 with a newline
> for (i in 1:10) {
+     print(i)
+ }
> # Sum of squares of 1:10
> (1:10)^2
 [1]   1   4   9  16  25  36  49  64  81 100
> sum((1:10)^2)
[1] 385
```

2. Try `v = rnorm(20)`. What is the length of the vector `v`? What is the summation of `v`? (Hint: Use the function `sum`.) What is the mean? (Hint: Use the function `mean`.) What is the variance and standard deviation? (Use the function `var` and `sd`.)

```
> v <- rnorm(20)
> v
 [1]  1.73005248 -0.97845106  0.83759813  2.37786037  0.09951812  1.30635567
 [7] -1.81987134  0.56947816  0.08173542 -0.39859737  0.03606763  2.14436418
[13]  0.36767422  0.31959704  0.15305834  1.41468246  1.51827840  0.36135462
[19] -1.68922703  0.87783010
> length(v)
[1] 20
> sum(v)
[1] 9.309359
> mean(v)
[1] 0.4654679
> var(v)
[1] 1.284884
> sd(v)
[1] 1.133527
```

3. Try `v = rnorm(20,mean=100)`. What is the mean of `v`? What is the variance? Compare this mean with the mean of the previous problem. Are these two means similar? How about the variances? Are they similar? Try `v = rnorm(20,mean=100,sd = 3)` and compare with previous results. (Note: `rnorm()` can generate random numbers from the normal distribution.)

```
> v <- rnorm(20, mean=100)
> v
 [1]  99.09846 100.39513 100.65857 101.35481 100.60418  99.92690 100.69124
 [8]  97.13165  96.57580 100.48261 100.76007  99.78171 100.79009 100.08776
[15]  98.51875 101.67336  99.38287 100.61369  99.92920 100.03486
> mean(v)       # The mean is much larger than the previous mean
[1] 99.92459
> var(v)
[1] 1.640700    # The variances are quite similar
> v <- rnorm(20, mean=100, sd=3)
> v
 [1] 101.45070 104.69059  96.35249 100.35655 101.57098 101.34621 100.75470
 [8]  99.52002  96.47503  98.49146  98.88748 102.27238 101.64558 101.82560
[15]  97.45174  95.73286 104.48075  96.22113  98.74802  96.80006
```

```
> mean(v)
[1] 99.75372
> var(v)        # Larger variance than the previous 2 cases
[1] 7.337591
```

4. Suppose that you track your commute times for two weeks (ten days), recording the following times (in minutes)

   17 16 20 24 22 15 21 15 17 22.

   Find the maximum, minimum, mean and standard deviation of your commute times. How many days was the commute time less than 20 minutes? How many days was the commute time between 17 and 21 minutes (including 17 and 21)?

```
> v <- c(17, 16, 20, 24, 22, 15, 21, 15, 17, 22)
> max(v)
[1] 24
> min(v)
[1] 15
> mean(v)
[1] 18.9
> sd(v)
[1] 3.28126
> length(v[v < 20])
[1] 5
> length(v[v >= 17 & v <= 21])
[1] 4
```

5. Let `v` be the vector `v <- c(9, 2, 8, 8, 5)`.

   (a) Sort `v` using the `[]` operators. (The `order` function will be useful here.)

```
> v[order(v)]
[1] 2 5 8 8 9
```

   (b) Select a random subsample of `v`. (The `sample` function will be useful here.)

```
> sample(v)     # Random subsample of the same size as v (equivalently: random shuffle of v)
[1] 8 2 9 8 5
> sample(v, 3)  # Random subsample of size 3
[1] 5 2 8
```

   (c) Use the `[]` operators and a boolean operation to select the elements in `v` which are greater than 5.

```
> v[v > 5]
[1] 9 8 8
```

6. Consider the matrix `M <- matrix(1:9, ncol=3)`.

   (a) Use the `[]` operators to select only the first row of `M`.

```
> M[1,]
[1] 1 4 7
```

   (b) Use the `[]` operators to select the second and third columns of `M`.

```
> M[, 2:3]
     [,1] [,2]
[1,]    4    7
[2,]    5    8
[3,]    6    9
```

(c) In the previous section, you used matrix multiplication to compute some rows and columns of a matrix. Now you have used the `[]` operators to *select* rows and columns in a matrix. What is the difference? (*Hint*: what if you wanted to alter values in a matrix?)

Sol'n. *Matrix multiplication forms a* new *matrix to hold the result of the computation. This result is completely separate from the original matrix—changing its values will not alter the original matrix. The `[]` operators can be used with the assignment operator (`<-` or `=`) to alter the values of the original matrix.*

(d) Use the `[]` operators to select 3 copies of the first row of `M`. *I.e.*, create the matrix

$$\begin{pmatrix} 1 & 4 & 7 \\ 1 & 4 & 7 \\ 1 & 4 & 7 \end{pmatrix}$$

using only the `[]` operators.

```
> M[c(1,1,1),]
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    1    4    7
[3,]    1    4    7
> # Typically, rep() is used to create repetitive vectors:
> M[rep(1,3),]
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    1    4    7
[3,]    1    4    7
```

3 Advanced

1. `x = c(2, 10, NA, 5, NA); y = c(NA, 5, 4, 3, NA)` Write a program to get a list of positions where no values are missing. `is.na()` is a useful command here.

```
> x <- c(2, 10, NA, 5, NA)
> y <- c(NA, 5, 4, 3, NA)
> which(!is.na(x) & !is.na(y))
[1] 2 4
```

2. `x = matrix(c(5,3,2,9,3,14,7,6,8),3,3)` Write a program to find the location of the maximum value of the matrix.

```
> # In practice, one should use:
> which(x == max(x), arr.ind=TRUE)
     row col
[1,]   3   2

> # Using a loop, we can determine which row contains the max
> m <- max(x)
> for (i in 1:nrow(x)) {
+     if (any(x[i,] == m))
+         r <- i
+ }
> r
[1] 3
```

```
> # The same method can be used to find the column
> for (i in 1:ncol(x)) {
+     if (any(x[,i] == m))
+         c <- i
+ }
> c
[1] 2
> # The max is located at:
> c(r, c)
[1] 3 2
```

3. Create and sort the matrix

$$M = \begin{pmatrix} 2 & 5 & 8 \\ 3 & 6 & 9 \\ 1 & 4 & 7 \end{pmatrix}$$

by its first column.

```
> M <- matrix(c(2, 3, 1, 5, 6, 4, 8, 9, 7), ncol=3)
> M
     [,1] [,2] [,3]
[1,]    2    5    8
[2,]    3    6    9
[3,]    1    4    7
> M[order(M[,1]),]
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

4. More matrices

   (a) Create a 4x4 zero matrix using the `matrix` function.

```
> Z <- matrix(0, nrow=4, ncol=4)
> Z
     [,1] [,2] [,3] [,4]
[1,]    0    0    0    0
[2,]    0    0    0    0
[3,]    0    0    0    0
[4,]    0    0    0    0
```

   (b) Create a 4x4 identity matrix using the `diag` function.

```
> I <- diag(4)
> I
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    1    0    0
[3,]    0    0    1    0
[4,]    0    0    0    1
```

   (c) Add the two previous matrices.

```
> Z + I   # Equal to I, unsurprisingly
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    1    0    0
```

```
[3,]    0    0    1    0
[4,]    0    0    0    1
```

(d) Use the `lower.tri` function to create a lower triangular matrix (all entries below the diagonal are 1). (Note: the `lower.tri` function creates a matrix of boolean (`TRUE` and `FALSE`) values. Use this boolean matrix in the `[]` operators to select the lower triangular portion of another matrix you create.)

```
> M <- Z    # Z is the zero matrix from part (a)
> M[lower.tri(M)] <- 1
> M
     [,1] [,2] [,3] [,4]
[1,]    0    0    0    0
[2,]    1    0    0    0
[3,]    1    1    0    0
[4,]    1    1    1    0
```

(e) Make a triangular matrix where the entries along the diagonal are also 1.

```
> # One way: add the identity to the lower triangular matrix
> M + I
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    1    1    0    0
[3,]    1    1    1    0
[4,]    1    1    1    1
> # Or use the diag=TRUE option to lower.tri
> M <- Z
> M[lower.tri(M, diag=TRUE)] <- 1
> M
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    1    1    0    0
[3,]    1    1    1    0
[4,]    1    1    1    1
```

(f) Compute the following matrix product:

$$\begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

The standard multiplication operator (`*`) won't work here; you must use the special matrix multiplication operator `%*%`. What product would give the second column? The first row?

```
> M <- matrix(1:9, ncol=3)
> M %*% c(1,0,0)   # Equals the first column of the matrix
     [,1]
[1,]    1
[2,]    2
[3,]    3
> M %*% c(0,1,0)   # Equals the second column of the matrix
     [,1]
[1,]    4
[2,]    5
[3,]    6
> c(1,0,0) %*% M   # Equals the first row of the matrix
     [,1] [,2] [,3]
[1,]    1    4    7
```

5. Assume that you know that the mean retail price of certain product nationwide is \$17. Suppose that you want to study the retail price of the product in Boston. You randomly choose 8 stores in Boston and record their retail prices of the product as `v = (15.9,21.4,19.9,21.9,20.0,16.5,17.9,17.5)` (in dollars). Use the function `t.test()` to calculate the P-value of the following alternative hypotheses (the null hypothesis is that the mean retail price in Boston is the same as that nationwide).

   (a) The mean retail price in Boston is not equal to that nationwide.

   Sol'n.
```
> t.test(v, mu=17, alternative="two.sided")

        One Sample t-test

data:  v
t = 2.3658, df = 7, p-value = 0.04991
alternative hypothesis: true mean is not equal to 17
95 percent confidence interval:
 17.00093 20.74907
sample estimates:
mean of x
   18.875
```
   The probability of selecting more a extreme price set from the null distribution is 0.04991.

   (b) The mean retail price in Boston is less than that nationwide.
```
> t.test(v, mu=17, alternative="less")

        One Sample t-test

data:  v
t = 2.3658, df = 7, p-value = 0.975
alternative hypothesis: true mean is less than 17
95 percent confidence interval:
     -Inf 20.37654
sample estimates:
mean of x
   18.875
```

   (c) The mean retail price in Boston is more than that nationwide.
```
> t.test(v, mu=17, alternative="greater")

        One Sample t-test

data:  v
t = 2.3658, df = 7, p-value = 0.02496
alternative hypothesis: true mean is greater than 17
95 percent confidence interval:
 17.37346      Inf
sample estimates:
mean of x
   18.875
```

   Note: use the `t.test()` with proper `mu` value `alternative` value.

6. Comparison of different normal distributions.

   (a) Generate 100 random numbers from the standard normal distribution. Save these numbers in a vector `v1`.

```
> v1 <- rnorm(100)
```

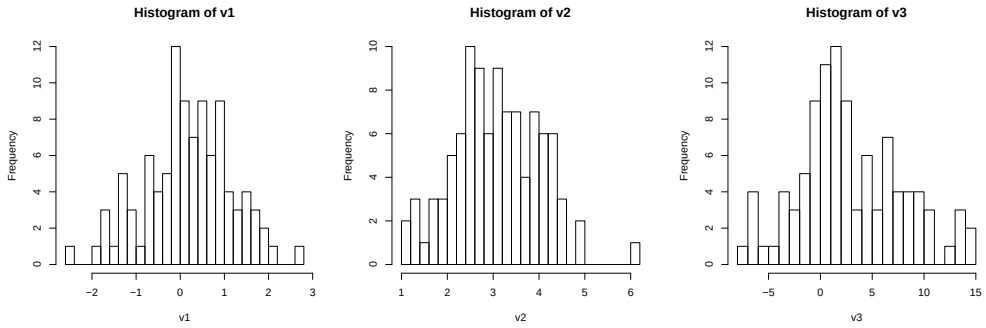
(b) Generate 100 random numbers from the normal distribution with mean 3 and standard deviation 1. Save these numbers in a vector `v2`.

```
> v2 <- rnorm(100, mean=3, sd=1)
```

(c) Generate 100 random numbers from the normal distribution with mean 3 and standard deviation 5. Save these numbers in a vector `v3`.

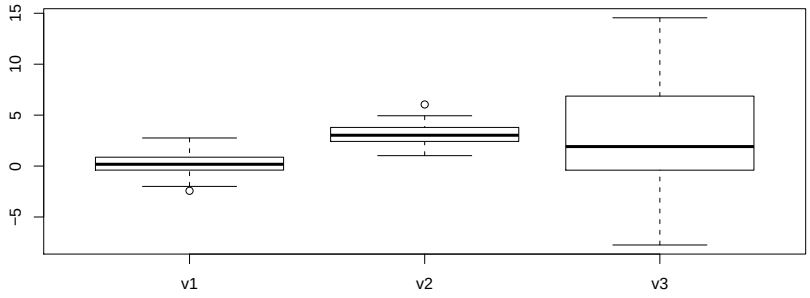
```
> v3 <- rnorm(100, mean=3, sd=5)
```

(d) Plot and compare the histograms of the three vectors `v1`, `v2` and `v3`. Does the shapes of histograms look similar? What looks different?



Sol'n. *The shapes are similar. The first histogram is centered around 0 while the last two are centered around 3. The spread of the first and second histogram is less than that of the third (*-2 *to 3 and 1 to 6 for the first two histograms and* -5 *to 15 for the third).*

(e) Plot and compare the boxplots of the three vectors `v1`, `v2` and `v3`.



(f) Test whether the means of `v1` and `v2` are equal using t test. What are the null hypothesis and the alternative hypothesis (Optional question)?

Sol'n. *Run the t-test on* `v1` *and* `v2`*:*

```
> t.test(v1, v2)

        Welch Two Sample t-test

data:  v1 and v2
```

```
t = -21.2935, df = 197.658, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.175541 -2.637208
sample estimates:
mean of x mean of y
0.1690405 3.0754151
```

The null hypothesis is that the difference between the true means of `v1` *and* `v2` *is 0. The alternative hypothesis is that the difference is not 0.*

7. Suppose you observed an event at the following times: $t = (1, 4, 5, 12, 15, 17, 33)$. You are interested in studying the time between events. Convert the vector t of event times into the vector d of time differences using a single operation. (*I.e.*, create a vector d such that $d_i = t_{i+1} - t_i$. d will be one element shorter than t.)

```
> t <- c(1,4,5,12,15,17,33)
> d <- t[2:length(t)] - t[1:(length(t) - 1)]
> d
[1]  3  1  7  3  2 16
```

8. Let v be a vector of length n. `R` contains a function named `cumsum` which will compute the vector of partial sums of `v`. *E.g.*, if $v = (1, 2, 3, 4, 5)$ then the vector of partial sums is

$$p = (1, 1+2, 1+2+3, 1+2+3+4, 1+2+3+4+5) = (1, 3, 6, 10, 15).$$

Without using `cumsum`, produce the same vector of partial sums using a single arithmetic operation. (*Hint*: Construct a suitable matrix, then use matrix multiplication to produce the vector of partial sums.)

Sol'n. *One way to do this is to construct the lower triangular matrix:*

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Then $M \cdot v$ is the vector of partial sums of v. To generalize to a vector v of any length, make the lower triangular matrix M with dimensions `length(v)` $\times$ `length(v)`*. In* `R`*:*

```
> M <- matrix(0, nrow=5, ncol=5)
> M
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    0    0    0    0
[2,]    0    0    0    0    0
[3,]    0    0    0    0    0
[4,]    0    0    0    0    0
[5,]    0    0    0    0    0
> # Make the lower triangular matrix
> M[lower.tri(M, diag=TRUE)] <- 1
> M
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    0    0    0
[2,]    1    1    0    0    0
[3,]    1    1    1    0    0
```

```
[4,]    1    1    1    1    0
[5,]    1    1    1    1    1
> v <- 1:5
> # One operation produces the vector of partial sums
> M %*% v
     [,1]
[1,]    1
[2,]    3
[3,]    6
[4,]   10
[5,]   15
```

9. Consider the "rotated" vector $v = (5, 6, 7, 8, 9, 1, 2, 3, 4)$. As you can see, every element in the vector is in order—$v_i < v_{i+1}$—for all i except one. Using a single boolean operation, determine the position i such that $v_i \not< v_{i+1}$.

```
> v
[1] 5 6 7 8 9 1 2 3 4
> v[1:(length(v) - 1)] < v[2:length(v)]
[1]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
> # Find the index of the FALSE in the above vector
> which(!(v[1:(length(v) - 1)] < v[2:length(v)]))
[1] 5
```