Homework 4

BMV

2/6/2018

1.Suppose x = 1.1, a = 2.2, and b = 3.3. Assign each expression to the value of the variable zand print the value stored in z.

```
x=1.1
a=2.2
b=3.3

# 1.a
z=x^(a)^(b)
print(z)
```

```
## [1] 3.61714
```

```
# 1.b
z=(x^a)^b
print(z)
```

```
## [1] 1.997611
```

```
# 1.c
z=3*(x^3) + 2*(x^2) +1
print(z)
```

```
## [1] 7.413
```

```
# 1.d
# round z to the first decimal place (which is the second number in z)
# subtract z rounded to the tenths place (so only the second number remains)
# well this gives, 0.4.... so need to get rid of decimal place
# multiply by 10?
z=(round(z, 1)- round(z,0)) * 10
print(z)
```

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

2.Using the rep and seq functions, create the following vectors:

a. (1,2,3,4,5,6,7,8,7,6,5,4,3,2,1)
b. (1,2,2,3,3,3,4,4,4,4,5,5,5,5,5)
c. (5,4,4,3,3,3,2,2,2,2,1,1,1,1,1)

```
# 2.a
vec1=c(seq(from=1, to =8, by=1), seq(from=7, to=1, by=-1))
print(vec1)
```

```
##  [1] 1 2 3 4 5 6 7 8 7 6 5 4 3 2 1
```

```
# 2.b
vec2digits=seq(1:5)
```

```
vec2=c(rep(vec2digits, times=vec2digits))
print(vec2)
```

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

```
# 2.c
vec3digits=seq(from=5, to=1, by=-1)
vec3=c(rep(vec3digits, times=vec2digits))
print(vec3)
```

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

3. Create a vector of two random uniform numbers. In a spatial map, these can be interpreted as x and y coordinates that give the location of an individual (such as a marked forest tree in a plot that has been mapped). Using one of R's inverse trigonometry functions (asin(), acos(), or atan()), convert these numbers into polar coordinates (If you don't know what polar coordinates are, read about them on the web or in your calculus textbook).

POLAR COORDS: each point on a plane is determined by a distance from a reference point and an angle from a reference direction.

From the web: xy is converted to r and theta, respectively. r = sqrt(x2 + y2) theta= tan-1 (y / x)

```
xy=runif(2)
# so first number is z and second number is y

# arc-tangent of two arguments atan2(y, x) returns the angle between the x-axis and the vector from the
r=sqrt(xy[1]^2 + xy[2]^2)
theta=atan(xy[2]/xy[1])

# need to combine the coords again
polarCoords=c(r, theta)
print(polarCoords)
```

```
## [1] 0.7966362 0.9647292
```

4. Suppose that queue <- c("sheep", "fox", "owl", "ant") and that queue represents the animals that are lined up to enter Noah's Ark, with the sheep at the front of the line.

Using R expressions, update the queue successively as

a. the serpent arrives;
b. the sheep enters the ark;
c. the donkey arrives and talks his way to the front of the line;
d. the serpent gets impatient and leaves;
e. the owl gets bored and leaves;
f. the aphid arrives and the ant invites him to cut in line.
g. Finally, determine the position of the aphid in the line.

```
queue <- c("sheep", "fox", "owl", "ant")

# 4.a
```

```
queue=c(queue, "serpent")
print(queue)
```

```
## [1] "sheep"   "fox"     "owl"     "ant"     "serpent"
```

```
# 4.b
# i could have used this subsetting for 4.a with queue[5]..just realized.
queue=queue[2:5]
print(queue)
```

```
## [1] "fox"     "owl"     "ant"     "serpent"
```

```
# 4.c
queue=c("donkey", queue)
print(queue)
```

```
## [1] "donkey"  "fox"     "owl"     "ant"     "serpent"
```

```
# 4.d
queue=queue[1:4] # peace out donkey!
print(queue)
```

```
## [1] "donkey" "fox"    "owl"    "ant"
```

```
# 4.e
queue=queue[c(1,2,4)] # peace out owl
print(queue)
```

```
## [1] "donkey" "fox"    "ant"
```

```
# 4.f it isn't a party without an aphid
queue=c(queue[1:2], "aphid", queue[3])
print(queue)
```

```
## [1] "donkey" "fox"    "aphid"  "ant"
```

```
# 4.g
aphidPos=which(queue=="aphid")
print(aphidPos)
```

```
## [1] 3
```

5.Use R to create a vector of all of the integers from 1 to 100 that are not divisible by 2, 3, or 7.

```
vec5=seq(from=1, to=100, by=1)
# this works: subset vec5 for elements which are not divisible by 2,3,7 separately, then concatenate th
q5sln=vec5[!(!vec5%%2) + (!vec5%%3) + (!vec5%%7)]
print(q5sln)
```

```
##  [1]  1  5 11 13 17 19 23 25 29 31 37 41 43 47 53 55 59 61 65 67 71 73 79
## [24] 83 85 89 95 97
```

6.Create a vector z of 1000 random uniform numbers.

a. create a vector that contains 3 numbers: the proportion of the numbers in z that are less than 0.10, greater than 0.90, and between 0.45 and 0.55.

```
z=runif(1000)
proportions=c((sum(z[which(z<0.1)])/1000), (sum(z[which(z>0.9)])/1000), (sum(z[which(z>0.45 & z<0.55)]),
print(proportions)
```

```
## [1] 0.004419594 0.087676224 0.048378552
```

b. Making successive copies of z, transform your vector of uniform numbers in the following ways:

b.1 log (base 10) of z

b.2 z2

b.3 ez

b.4 square root of z

For each case calculate your vector of 3 numbers to get the new proportions.

```
# b.1
z=log10(z)
proportions=c((sum(z[which(z<0.1)])/1000), (sum(z[which(z>0.9)])/1000), (sum(z[which(z>0.45 & z<0.55)]),
print(proportions)
```

```
## [1] -0.4227511  0.0000000  0.0000000
```

```
# b.2
z=z^2
proportions=c((sum(z[which(z<0.1)])/1000), (sum(z[which(z>0.9)])/1000), (sum(z[which(z>0.45 & z<0.55)]),
print(proportions)
```

```
## [1] 0.01369573 0.19570165 0.01898454
```

```
# b.3
z=exp(z)
proportions=c((sum(z[which(z<0.1)])/1000), (sum(z[which(z>0.9)])/1000), (sum(z[which(z>0.45 & z<0.55)]),
print(proportions)
```

```
## [1]  0.00000 26.96719  0.00000
```

```
z=sqrt(z)
proportions=c((sum(z[which(z<0.1)])/1000), (sum(z[which(z>0.9)])/1000), (sum(z[which(z>0.45 & z<0.55)]),
print(proportions)
```

```
## [1] 0.000000 1.515835 0.000000
```