Lab 2

Suixin Jiang

09/07/2019

Exercise 1

Take the absolute value of a vector of 100 random draws from the standard normal distribution, then take the square root of these values. Use pipes. Useful functions: `sqrt()`, `rnorm()`, `abs()`.

```
## Solution

library(magrittr)
```

```
## Warning: package 'magrittr' was built under R version 3.5.3
```

```
rnorm(100) %>%
  abs() %>%
  sqrt()
```

```
##   [1] 0.61799867 1.10037967 1.44823375 1.01618168 0.75860254 0.87266318
##   [7] 0.83909791 1.41430228 0.30399350 0.99827187 0.92233915 1.24999705
##  [13] 0.19759295 1.50599957 0.31521385 0.42628895 0.35815312 0.38851723
##  [19] 0.15174188 0.48949492 0.69131475 0.80666641 0.32379168 0.85042405
##  [25] 0.08311237 1.19004960 1.05681914 0.48852193 0.54976205 1.54679580
##  [31] 0.36169416 0.89143225 1.18507326 1.01753289 0.93529026 1.09491954
##  [37] 1.47804513 0.79075621 0.60346758 0.38636773 0.21153077 1.19032771
##  [43] 0.25608593 0.89656699 0.86778370 1.22736837 0.40702523 0.94716209
##  [49] 1.01299146 0.74261262 0.54759120 0.53649204 0.68123312 0.70699937
##  [55] 0.56688995 1.34980212 0.80107259 0.70398172 0.89930317 1.13706701
##  [61] 1.02338283 1.27903961 1.55133918 0.21474445 0.54872154 0.41033043
##  [67] 1.34535484 0.37717061 1.40435250 0.98127899 0.20490027 0.56160920
##  [73] 0.29928939 0.80946770 0.74869823 0.67165128 1.29302633 1.33535420
##  [79] 1.27803826 0.49036061 1.20237079 0.44651285 0.52689201 1.02646157
##  [85] 0.70444287 0.76159867 0.78876552 0.73375107 1.20369408 1.14694233
##  [91] 1.31823484 1.23517647 0.66952025 0.57561873 0.76757980 0.54679028
##  [97] 1.13755544 1.01556639 0.94264002 0.46746189
```

Exercise 2

Create a function that takes a numeric vector (say `x`) and a numeric scalar (say `y`) as arguments. It should return a numeric vector where the first half of the returned vector is the same as the first half of `x`, and the second half of the returned vector is the same as the second half of `x + y`. If the length of `x` is odd, then it randomly chooses the middle value of the returned vector to be the same as the middle value `x` or the same as the middle value of `x + y`. Think up a good name and write documentation. Check the inputs of the function for correctness. Useful functions: `length()`, `%%` (to check for even vs odd), `floor()`, `sample()`, `c()`. You will also need to use conditionals and logicals to write this function.

```
## Solution

# Function Documentation

# The creation of a new numeric vector from an old numeric vector and a numeric
```

```r
# scalar

# x: the old numeric vector ( number of elements in vector could be even or odd)
# y: the numeric scalar

# returns: the new numeric vector with different conditions

new_vec <- function(x,y) {
  stopifnot(length(y) == 1)
  if ( length(x) %% 2 == 0 ) {
    return ( new_vec <- c(head(x,(length(x)/2)), tail((x+y),(length(x)/2))))
  } else {
    return ( new_vec <- c(head(x,((length(x)-1)/2)),
      sample(M<-c(x[(length(x)+1)/2],(x+y)[(length(x)+1)/2]),1),
      tail((x+y),((length(x)-1)/2))))
  }
}
x <- c(2,4,6,8)
y <- 5
new_vec(x,y)

x <- c(1,3,5,7,9)
new_vec(x,y)
```

```
## [1]  1  3 10 12 14
```

Question?

What happens if `x <- c(1, 1, 1, 1)` and `y <- c(1, 2)`?

```r
## Solution

# I restrict the length(y) equal to 1 in the function, so when y <- c(1, 2) the
# function would not run since 'length(y) == 1 is not TRUE'.
```