Hints and answers for some problems in Assignment 2

Haviland Wright

September 18, 2016

This document reiterates information we discussed in class and applies it to a selection of the problems in assignment 2.

As you do the problems in this assignment, keep in mind that there are many solutions to the problems. This assignment does not challenge you to get the "right answer." It challenges you to see that some answers are better than others because they are better designed and use R more efficiently.

Keeping in mind the distnction between answers that produce the correct result and answers are produced with well-designed code, review your answers with respect to:

- How they translate mathematical expression into working code, and
- how they use the vector and matrix structures and commands in R to produce results without using ANY loops (for, while, until, and so on).

Exercise 1 (Problem 1)

In this problem, parts a, b, c are the warm up. Part d is the setup. Parts e, f, g are the point.

part e

```
# In part d we created tmp

tmp <- c(4, 6, 3)

# using temp to get 10 occurrences of 4

ans.1e <- rep(tmp,10)
ans.1e
```

```
##  [1] 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3
```

part f

```
# you could just "glue" a 4 onto ans.1e

ans.1f <- c(ans.1e,4)
ans.1f
```

```
##  [1] 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4
```

```
# Or you could recognize that if you are repeating tmp iteratively and
# set a length that is not a multiple of 3, you will get an incomplete instance
# of tmp at the end fo the vector, making the number of 4s, 6s, and 3s different

ans.1fa <- rep(tmp, length = 31)
ans.1fa
```

```
##  [1] 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4 6 3 4
```

Exercise 1 (Problems 2, 3, 4, 6) and Exercise 2 (problems 3, 4, 5)

Termwise vector operations in R do not look like their algebraic equivalents, but keep in mind:

These examples:

```
  a + 1
  a + b
  b * 2
  a * b
  a^b
  b/a
  b%%a
  a%*%b
  a%o%b
  outer(a,b,"*")
  outer(a,b,"+")
  outer(a^2,b,"<")
  a%o%a%%2
  1:4%o%1:4%%5
```

Make sure you know how they work.

```
a <- c(1, 2, 3)
b <- c(5, 7, 11)

a + 1
```

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

```
a + b
```

```
## [1]  6  9 14
```

```
b * 2
```

```
## [1] 10 14 22
```

```
a * b
```

```
## [1]  5 14 33
```

```
a^b
```

```
## [1]      1    128 177147
```

```
b/a
```

```
## [1] 5.000000 3.500000 3.666667
```

```
b%%a
```

```
## [1] 0 1 2
```

```
a%*%b
```

```
##      [,1]
## [1,]   52
```

```
a%o%b
```

```
##      [,1] [,2] [,3]
## [1,]    5    7   11
## [2,]   10   14   22
## [3,]   15   21   33
```

```
outer(a,b,"*")
```

```
##      [,1] [,2] [,3]
## [1,]    5    7   11
## [2,]   10   14   22
## [3,]   15   21   33
```

```
outer(a,b,"+")
```

```
##      [,1] [,2] [,3]
## [1,]    6    8   12
## [2,]    7    9   13
## [3,]    8   10   14
```

```
outer(a^2,b,"<")
```

```
##       [,1]  [,2] [,3]
## [1,]  TRUE  TRUE TRUE
## [2,]  TRUE  TRUE TRUE
## [3,] FALSE FALSE TRUE
```

```
a%o%a%%2
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    1
## [2,]    0    0    0
## [3,]    1    0    1
```

```
1:4%o%1:4%%5
```

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

Exercise 1 (Problems 4, 6d) and Exercise 2 (problem 8)

To program a summation, it helps to write out the first few terms so the pattern is clear.

Exercise 1 (Problems 6 and 7)

I demonstrated plots for this problem in class using the built-in functions for plot (initial plot) and points (adding data points to an exiting plot).

Start by creating short versions of vectors xVec and yVec.

```
set.seed(50)
xVec <- sample(0:99, 15, replace=TRUE)
yVec <- sample(0:99, 15, replace=TRUE)
```

Now, set x = xVec (for simplicity) and make the vector

$$(x_1^2 - 5x_4, ..., x_{n-3}^2 - 5x_n)$$

```
# here is xVec the first element is x[1]
xVec
```

```
##  [1] 70 43 20 76 51  4 69 64  4 10 39 26 64  7 27
```

```
# so the specified vector is
xVec[c(-13, -14, -15)] + 5*xVec[c(-1, -2, -3)]
```

```
##  [1] 450 298  40 421 371  24 119 259 134 330  74 161
```

Now, try out order() and sort()

```
ord <- order(xVec, decreasing = FALSE)
ord
```

```
##  [1]  6  9 14 10  3 12 15 11  2  5  8 13  7  1  4
```

```
# these are index numbers
# here's how they make sense --
# they are the indexes that will order xVec

xVec[ord]
```

```
##  [1]  4  4  7 10 20 26 27 39 43 51 64 64 69 70 76
```

```
# That's exactly what sort does

sort(xVec, decreasing=FALSE)
```

```
##  [1]  4  4  7 10 20 26 27 39 43 51 64 64 69 70 76
```

Special note for MSSP students

MSSP students will want to pay careful attention to order() and rank() even though rank() was not included in these exercises.

To see why, generate a set of 40 random variates which are generated as the sum of 5 draws from the uniform distribution U(0,20).

```
set.seed(5)
options(digits=1)
data <- 20*(runif(40) + runif(40) + runif(40) + runif(40) + runif(40))

# data

ord <- order(data)

# here is the data in increasing order
data[ord]
```

```
##  [1] 27 29 38 40 40 42 42 43 44 44 44 45 47 48 48 49 50 50 51 52 52 52 52
## [24] 53 53 54 55 56 56 57 59 59 61 61 61 63 66 69 70 74
```

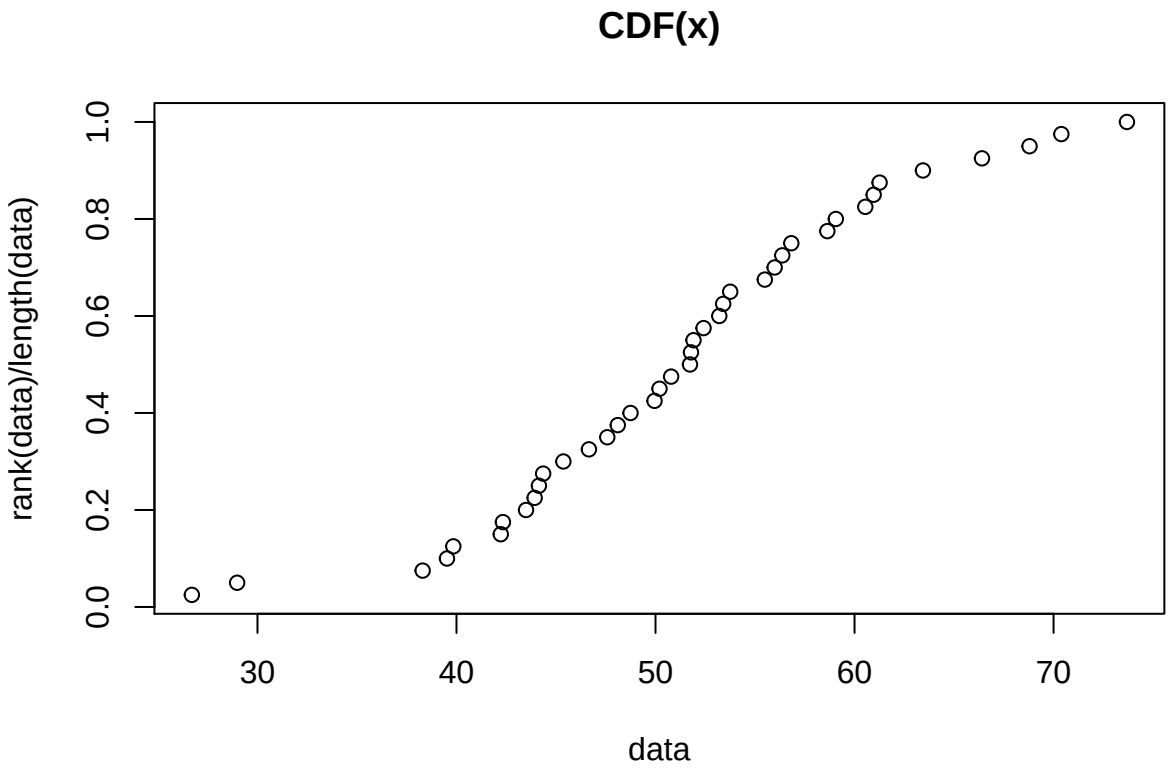
```
# here is the data as ranks
rank(data)
```

```
##  [1] 27 35 28 26 23  5 38 22 34 25  4 17 16 29 11 12  2 36  3  7 40 21 10
## [24] 39 37 20 31 18 33 32 19 24 15  8  1 30  6 13  9 14
```

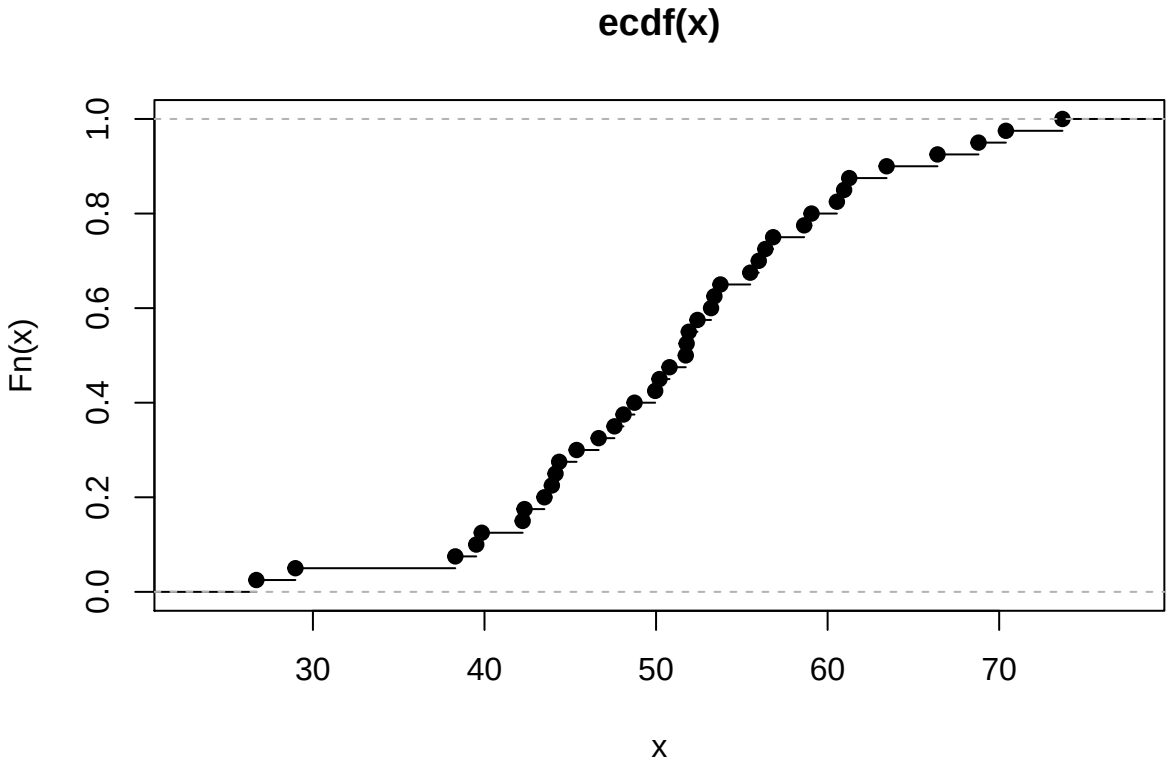
Now we can plot the empirical distribution using the built in plotting functions. Note the build-in ecdf function that produces a plot showing the CDF as right-continuous. Also note the plots of the CDF as a simple line chart. I threw in the qqnorm() plot to reinforce what you already know.

What do you expect the variance to be? What is it?

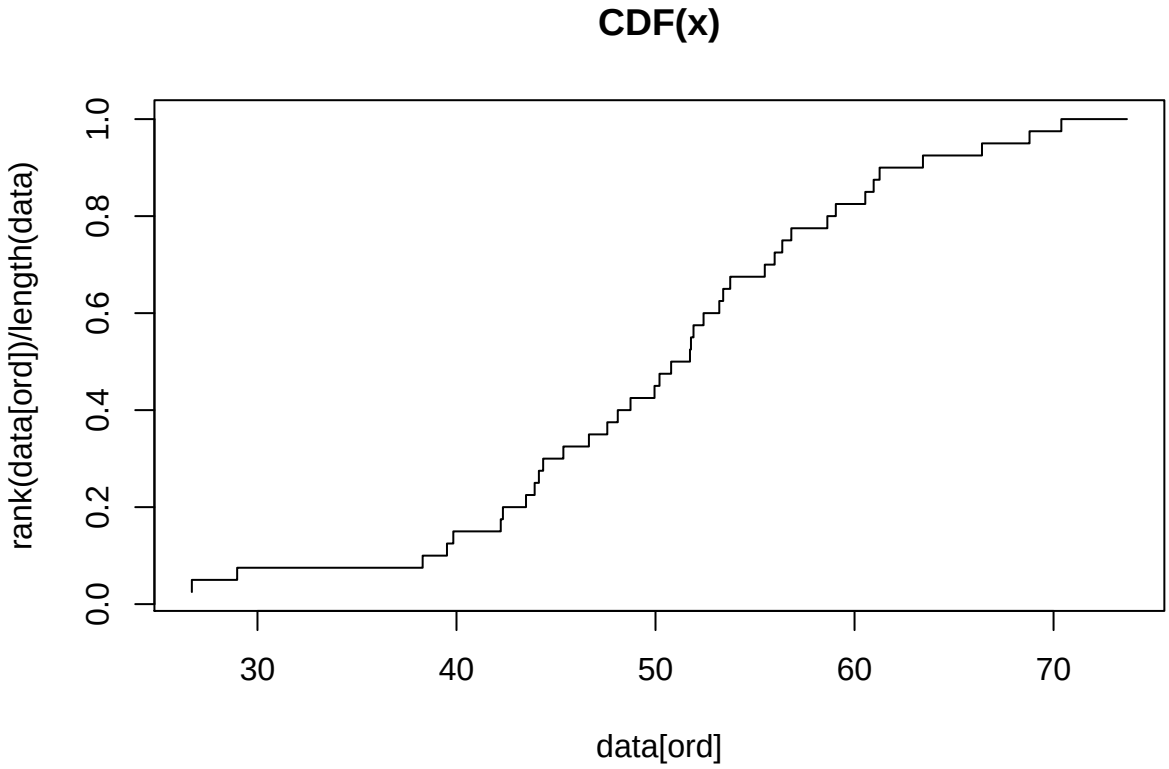
```r
plot(data, rank(data)/length(data),main = "CDF(x)")
```



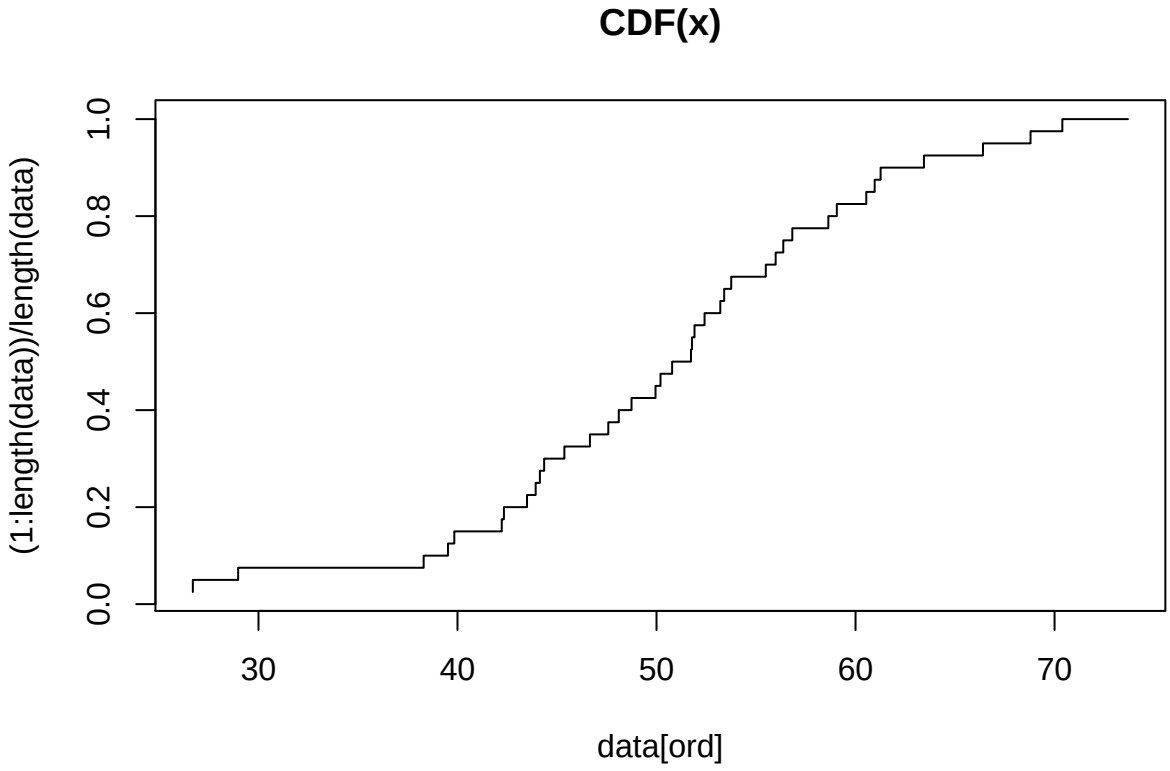
```
plot.ecdf(data)
```



```r
plot(data[ord], rank(data[ord])/length(data), type="S", main = "CDF(x)")
```



```r
plot(data[ord],(1:length(data))/length(data), type="S", main = "CDF(x)")
```



```r
qqnorm(data)
```