R Tutorial: Data

Arturo Valdivia

2/16/2023

Table of Contents

In this lab:

---

to clear global environment use this code:

```
rm(list = ls())
```

libraries needed

```
library(ggplot2)
```

1.1. Vectors using scan()

Store the data in the same folder as your R file and you can run the following code:

```r
vec = scan('heights.txt')
```

Let's use the author's website

```r
x <- scan("https://mtrosset.pages.iu.edu/StatInfeR/Data/pulses.dat")
```

1.2. Pseudo-random generation from known distributions

```r
rnorm(30)
```

```
##  [1]  1.26555487  0.64179700 -0.70087140  0.15356195 -1.30199732  0.62743898
##  [7] -1.63255489 -0.09345663 -0.97320899 -0.19382644  1.04569911 -1.24209613
## [13] -0.51693087  0.45069548 -0.67649292 -0.15410102  0.60734849 -0.97417464
## [19] -0.19010198 -2.11648823 -0.54777696 -0.41982145  0.07285659 -0.92189558
## [25]  0.54172169  0.46634178 -1.42146465 -1.16859733 -0.39043863 -0.19567071
```

```r
rnorm(30, mean = 10, sd = 2)
```

```
##  [1]  5.383257  9.349245 12.983536 10.625004 10.774730  6.044033 13.284397
##  [8] 10.371585 12.239562 10.828398  8.718485 11.233119  7.598851 14.051787
## [15]  6.587761 12.284939  8.082135  9.664722 10.105101  9.118470 10.060498
## [22]  9.974995  8.955884 12.795069 12.623904 12.321269  9.217056 10.602889
## [29]  5.997723  6.875901
```

```r
runif(20)
```

```
##  [1] 0.663395855 0.267400002 0.161850887 0.897001715 0.298963255 0.856659650
##  [7] 0.490228195 0.039142269 0.061110056 0.965351193 0.186579692 0.761262779
## [13] 0.954761658 0.593755346 0.003693574 0.222246209 0.523668747 0.439587001
## [19] 0.263153397 0.117877308
```

```r
rbinom(25, size = 10, prob = 0.3)
```

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

1.3. A random sample from a vector

```r
vec
```

```
##   [1] 179.0 195.1 183.7 178.7 171.5 181.8 172.5 181.6 174.6 190.4 173.8 172.6
##  [13] 185.2 178.4 177.6 183.5 178.1 177.0 172.9 180.2 188.4 169.4 180.2 189.0
##  [25] 182.4 185.8 180.7 178.7 183.4 183.0 177.0 175.0 179.9 160.9 175.1 176.8
##  [37] 186.3 180.4 176.4 174.0 181.0 170.1 175.2 170.3 185.1 182.7 169.4 172.8
##  [49] 194.3 181.1 181.3 167.0 180.7 174.5 192.8 176.4 180.8 174.1 171.0 188.0
##  [61] 180.8 172.6 183.7 180.7 178.8 176.4 158.9 166.0 162.2 167.8 170.9 164.9
##  [73] 168.1 164.0 163.3 183.2 167.0 163.8 174.0 163.0 167.1 168.1 163.0 154.6
```

```
##  [85] 170.3 170.6 175.1 156.5 160.3 170.8 156.5 165.2 169.8 171.2 160.4 163.8
##  [97] 169.6 172.7 162.4 166.8 157.1 181.1 158.4 165.6 166.7 156.5 168.1 165.3
## [109] 163.7 173.7 163.9 169.2 170.1 166.0 164.2 176.0 170.9 169.2 172.0 163.0
## [121] 154.5 172.5 175.6 167.2 164.0 162.1 161.6 153.6 177.5 169.8 173.5 166.8
## [133] 166.2 162.8 168.6 169.2
```

```
sample(x = vec, size = 10, replace = T)
```

```
##  [1] 162.1 162.8 179.0 180.8 165.2 178.7 182.4 163.9 180.8 178.7
```

or simply

```
sample(vec, 10, T)
```

```
##  [1] 175.1 169.2 188.4 166.8 166.0 163.8 167.2 156.5 178.8 175.0
```

```
set.seed(10) # seed to replicate the random sample
sample(vec, 10, T)
```

```
##  [1] 164.0 169.2 164.9 156.5 177.6 164.0 189.0 185.2 160.4 169.2
```

---

2. Plug-in Estimates

Let's use

```
x <- scan("https://mtrosset.pages.iu.edu/StatInfeR/Data/pulses.dat")
```

Let's construct the empirical probability distribution

pmf

```
n=length(x)
n
```

```
## [1] 39
```

```
fx = rep(1,n) / n
fx
```

```
##  [1] 0.02564103 0.02564103 0.02564103 0.02564103 0.02564103 0.02564103
##  [7] 0.02564103 0.02564103 0.02564103 0.02564103 0.02564103 0.02564103
## [13] 0.02564103 0.02564103 0.02564103 0.02564103 0.02564103 0.02564103
## [19] 0.02564103 0.02564103 0.02564103 0.02564103 0.02564103 0.02564103
## [25] 0.02564103 0.02564103 0.02564103 0.02564103 0.02564103 0.02564103
## [31] 0.02564103 0.02564103 0.02564103 0.02564103 0.02564103 0.02564103
## [37] 0.02564103 0.02564103 0.02564103
```

plug-in estimate for the expected value (mu.hat)

```r
EX = sum(x * fx)
EX
```

```
## [1] 70.30769
```

```r
mean(x)# mean(): the mean or sample average
```

```
## [1] 70.30769
```

```r
median(x)# median(): median, 2nd quartile, or 0.5-quantile
```

```
## [1] 72
```

```r
quantile(x)# The 5-number summary
```

```
##   0%  25%  50%  75% 100%
##   52   64   72   76   92
```

```r
quantile(x, probs = 0.25) #q1 or 0.25-quantile
```

```
## 25%
##  64
```

```r
summary(x) # 5-number summary plus the mean
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   52.00   64.00   72.00   70.31   76.00   92.00
```

```r
IQR(x) # IQR
```

```
## [1] 12
```

plug-in estimate for the variance

```r
VarX = sum ((x - EX)^2 * fx)
VarX
```

```
## [1] 87.90533
```

```r
mean((x - mean(x))^2) # alt. formula 1
```

```
## [1] 87.90533
```

```r
mean(x^2) - mean(x)^2 # alt. formula 2
```

```
## [1] 87.90533
```

```
sqrt(VarX) # Standard Deviation
```

```
## [1] 9.375784
```

Empirical CDF

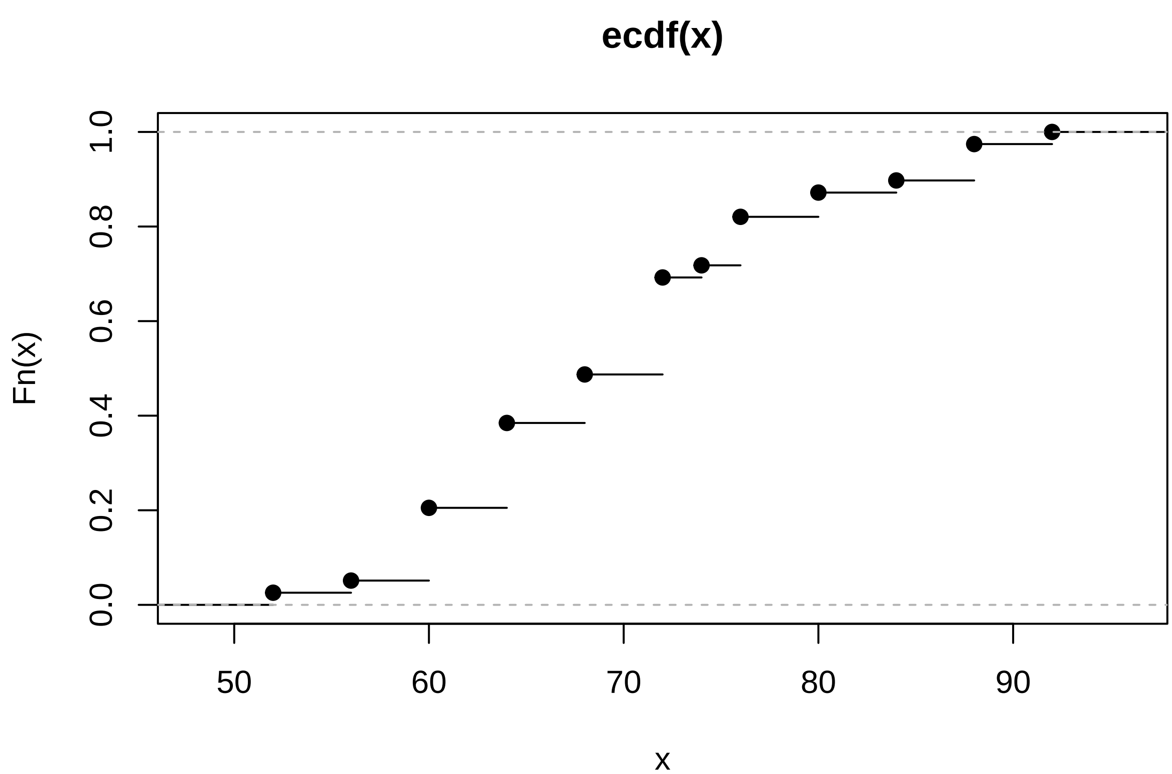
```
EF = ecdf(x)
EF(60)
```

```
## [1] 0.2051282
```

```
mean(x <= 60) #using logical operators
```

```
## [1] 0.2051282
```

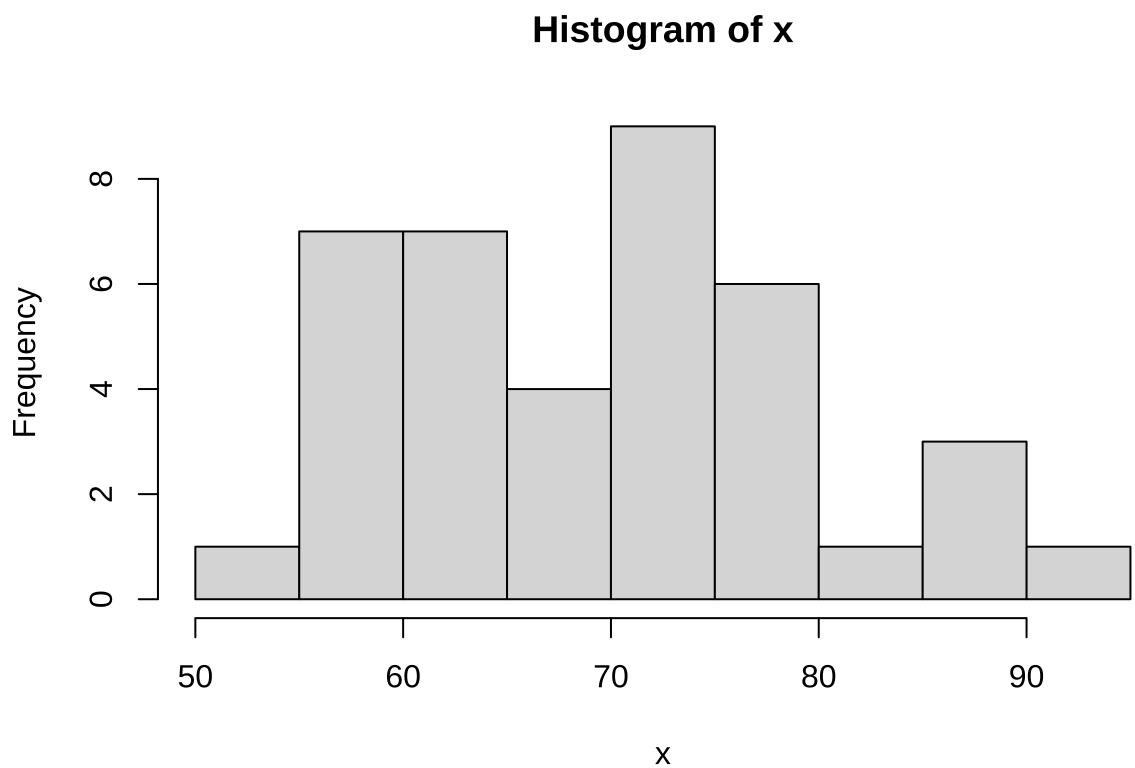
```
plot(ecdf(x))
```



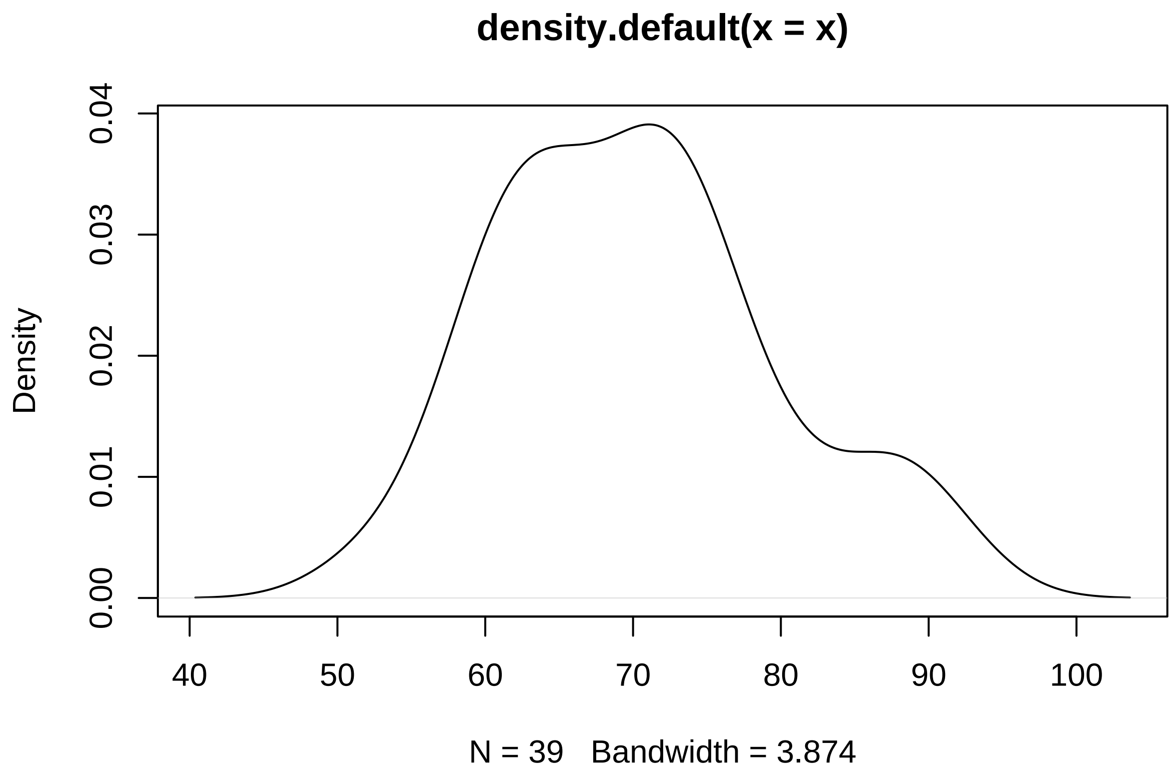
3. Plots

3.1. Using R base graphics

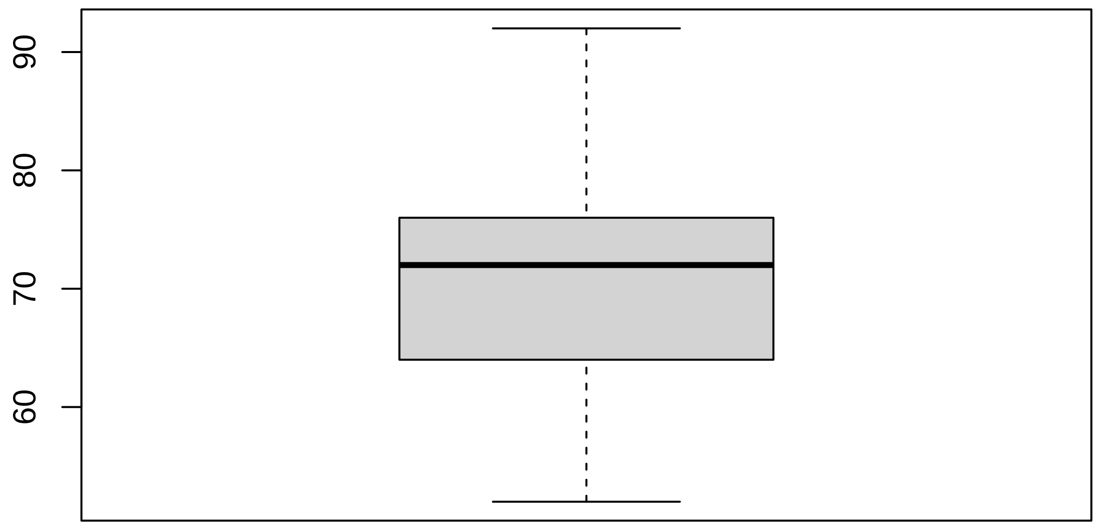
```
hist(x) # histogram
```



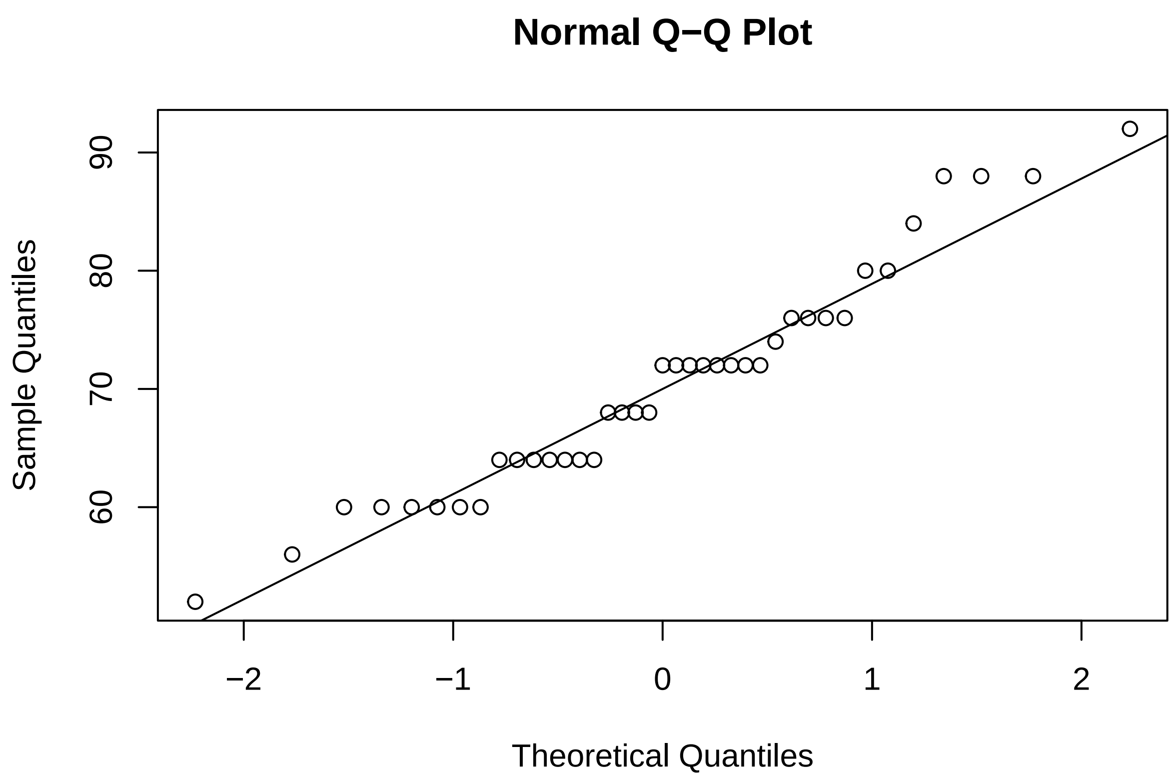

```
plot(density(x)) # kernel density estimate
```




```
boxplot(x) # boxplot
```




```
qqnorm(x) # Normal probability or QQ plot
qqline(x)
```

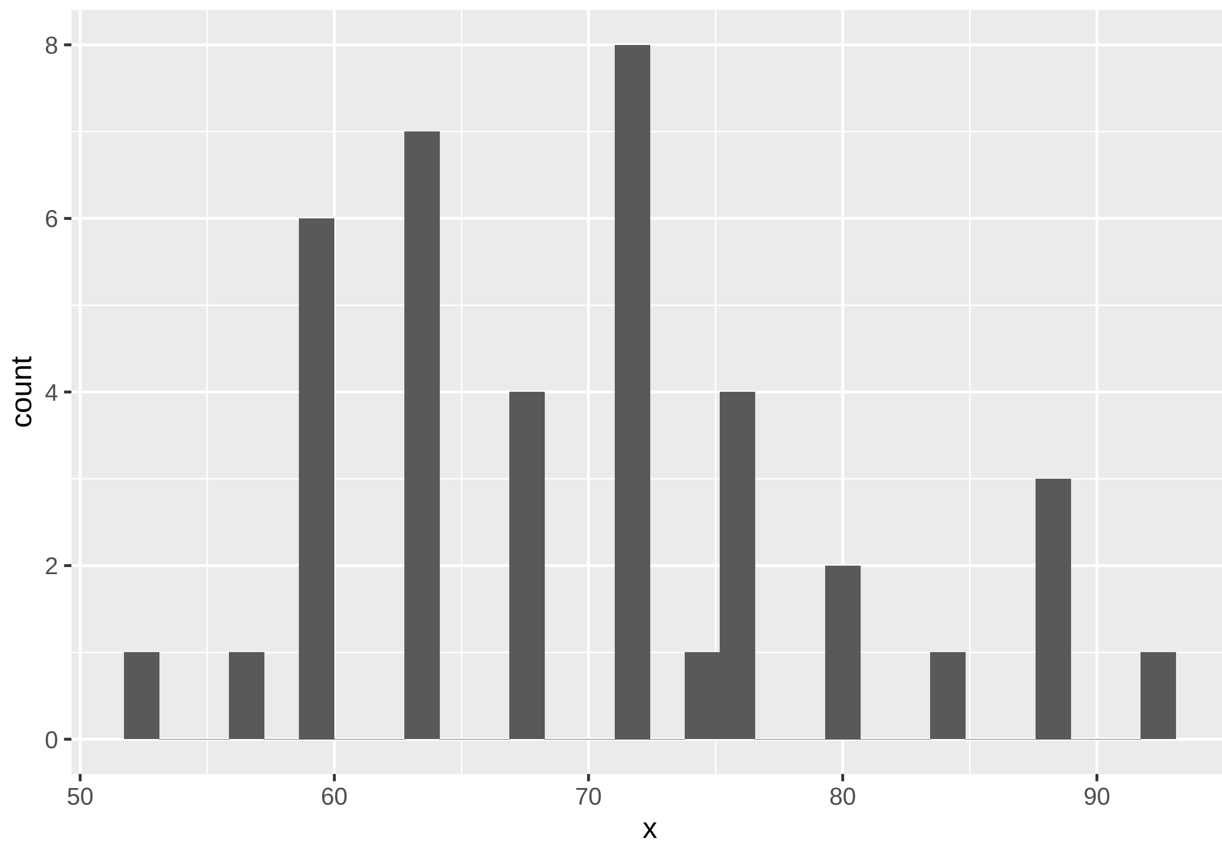


3.2. Using ggplot2

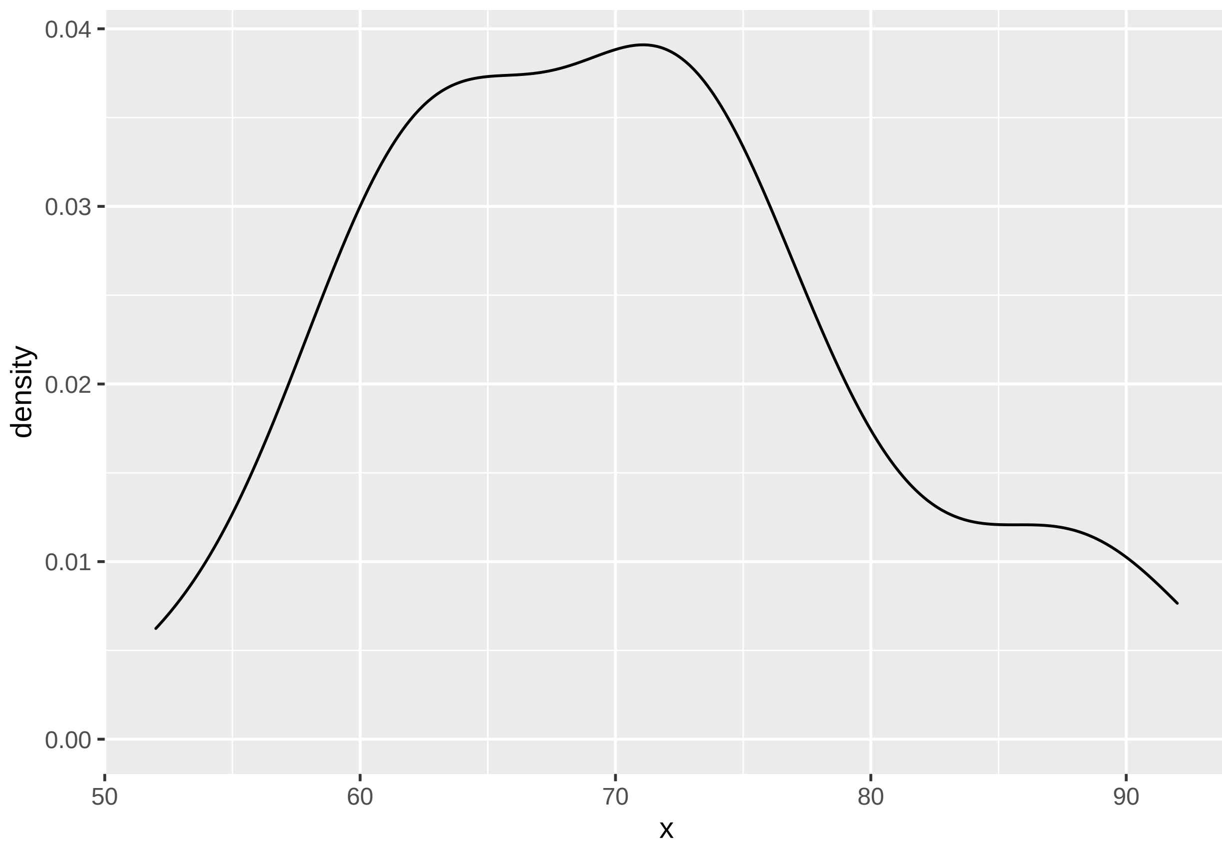
We need x to be a variable in a data frame

```
data1 = data.frame(x = scan("https://mtrosset.pages.iu.edu/StatInfeR/Data/pulses.dat"))
#View(data1)

g1 = ggplot(data = data1, mapping = aes(x = x))

g1 + geom_histogram()
```

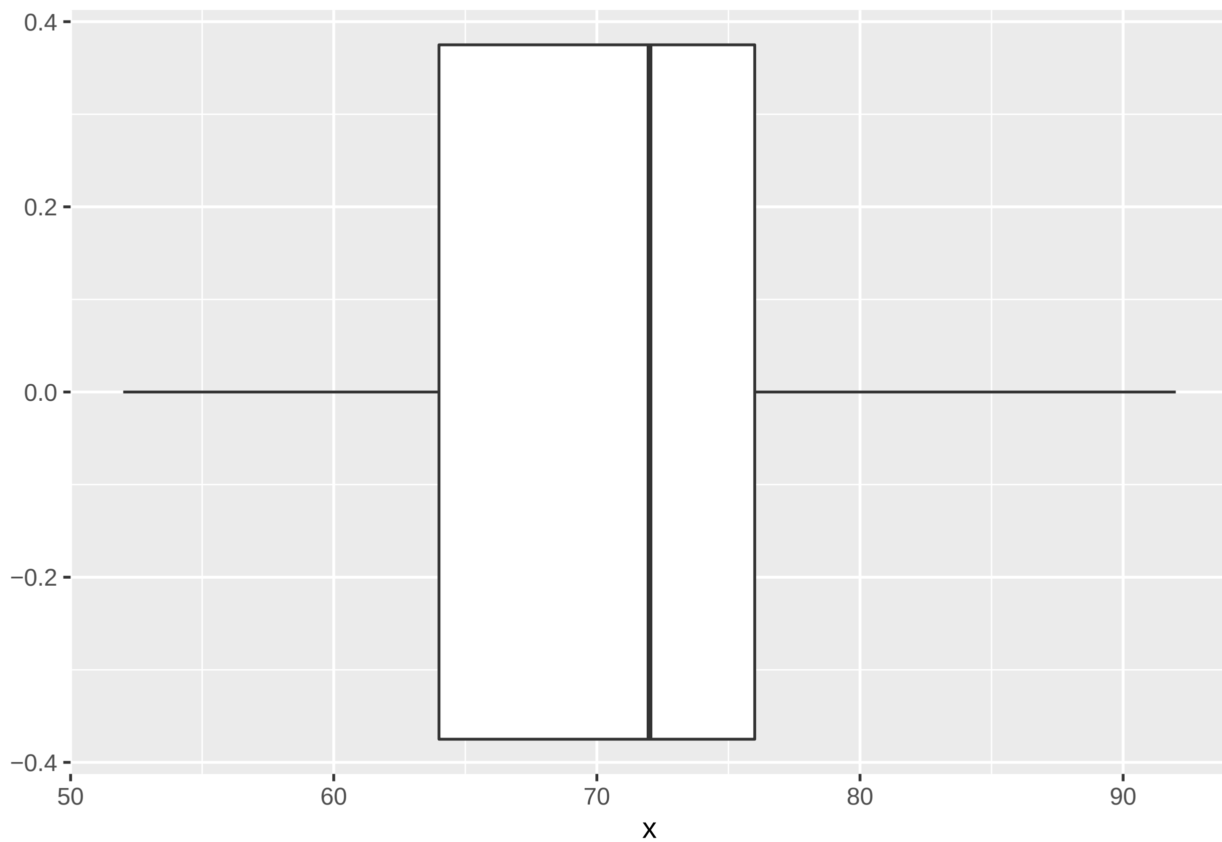
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



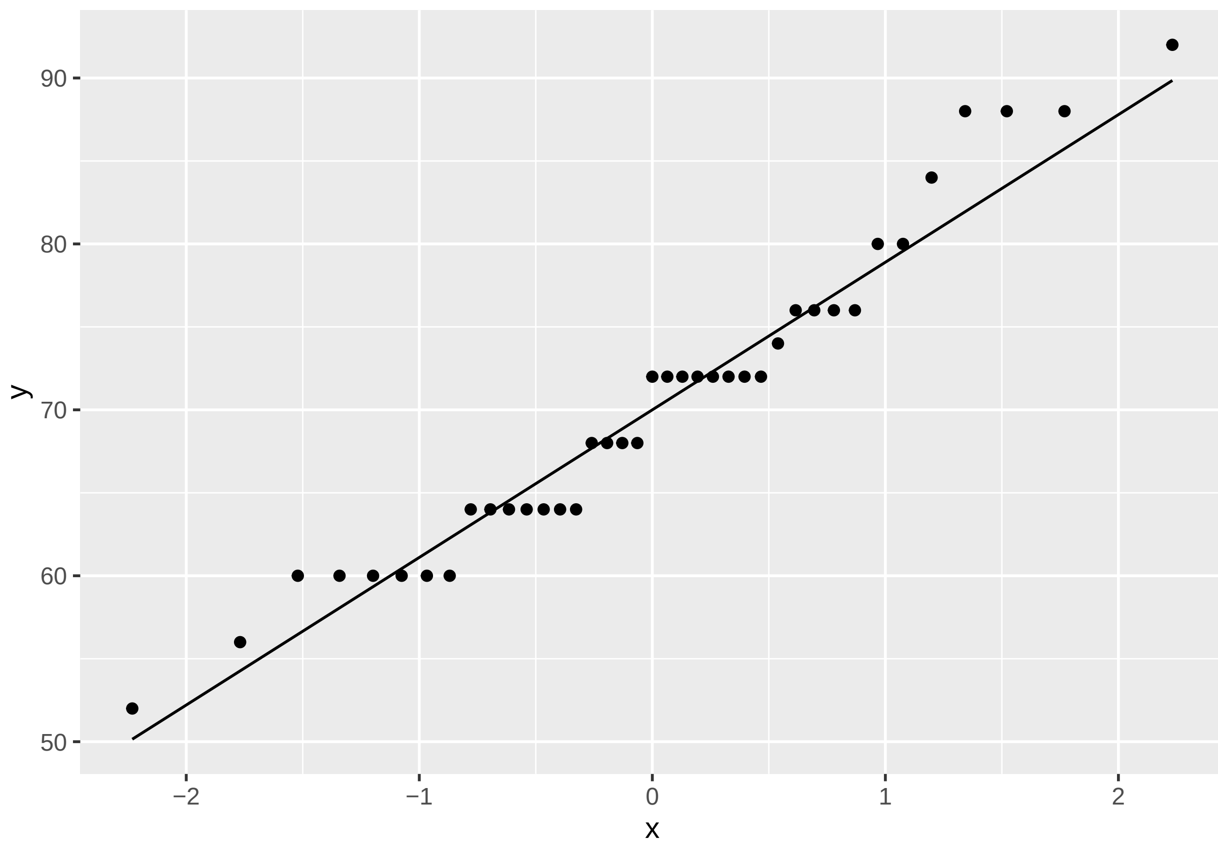

```
g1 + geom_density()
```




```
g1 + geom_boxplot()
```




```
g2 = ggplot(data = data1, mapping = aes(sample = x))
g2 + geom_qq() + geom_qq_line()
```



---

4. Simulations

```
my.vec = 1:20
```

We find the average of 30 numbers samples from my.vec

```
mean(sample(my.vec,30,replace = T))
```

```
## [1] 11.43333
```

Let's replicate this 15 times

```
replicate(n = 15,expr = mean(sample(my.vec,30,replace = T)))
```

```
##  [1]  9.90000 11.23333 10.56667 10.00000  8.70000 11.03333 12.53333 10.23333
##  [9] 10.33333 11.76667 10.00000 11.76667  9.80000 12.03333 10.83333
```

We can store this average in a single vector

```r
mean.vec = replicate(n = 15,expr = mean(sample(my.vec,30,replace = T)))
mean.vec
```

```
##  [1]  9.933333  9.066667 10.966667 10.700000  9.766667  9.666667 11.400000
##  [8] 10.300000 10.366667 10.666667  9.466667 10.966667  9.233333 10.266667
## [15] 10.866667
```

---

5. Functions

Let's construct a function to add two numbers

```r
my.sum = function(a,b) a+b

my.sum(3,5)
```

```
## [1] 8
```

Let's construct a function to generate a random sample of n numbers from a normal distribution with mean=mu and variance=sigma2 and find the ratio iqr/σ

```r
my.ratio = function(n, mu, sigma2){
  sigma = sqrt(sigma2)
  x = rnorm(n, mu, sigma2)
  IQR(x)/sigma
}
my.ratio(n = 50, mu = 10, sigma2 = 25)
```

```
## [1] 6.946945
```