STATISTICAL INFRENCE

Moein Karami

23/4/2022

a) Central Limit Theorem

1)

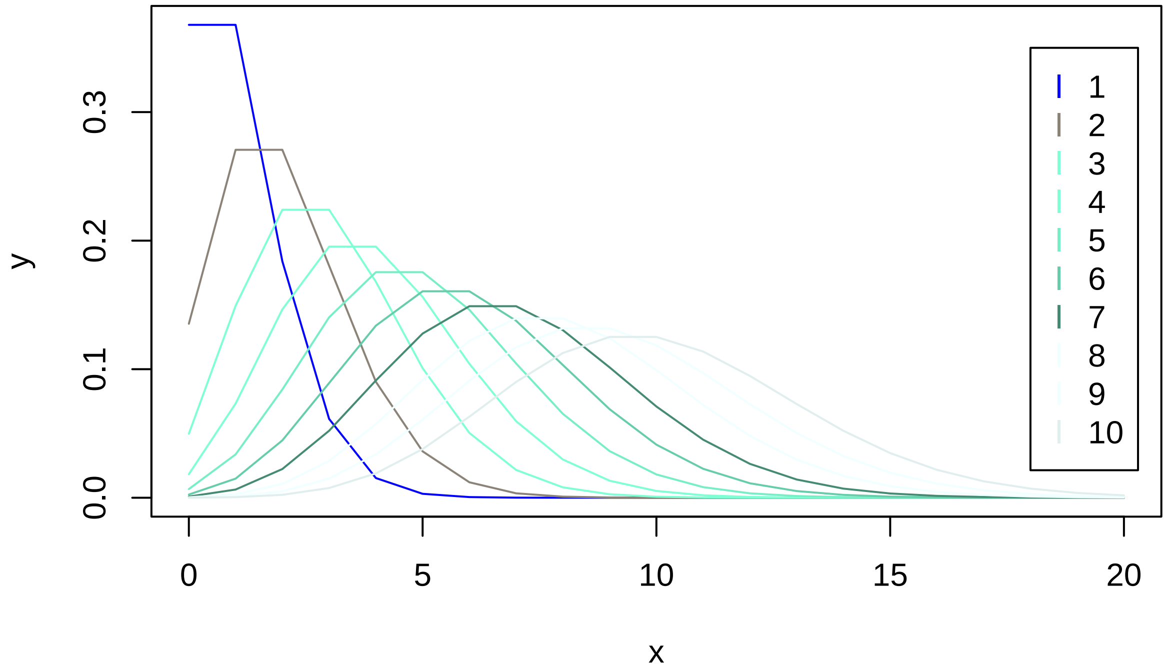
The number of flights departing from an airport, number customers lining up at the store register, the number of earthquakes occurring in a year at a specific region.

2)

As we can see by reducing lamda, left skewness increases.

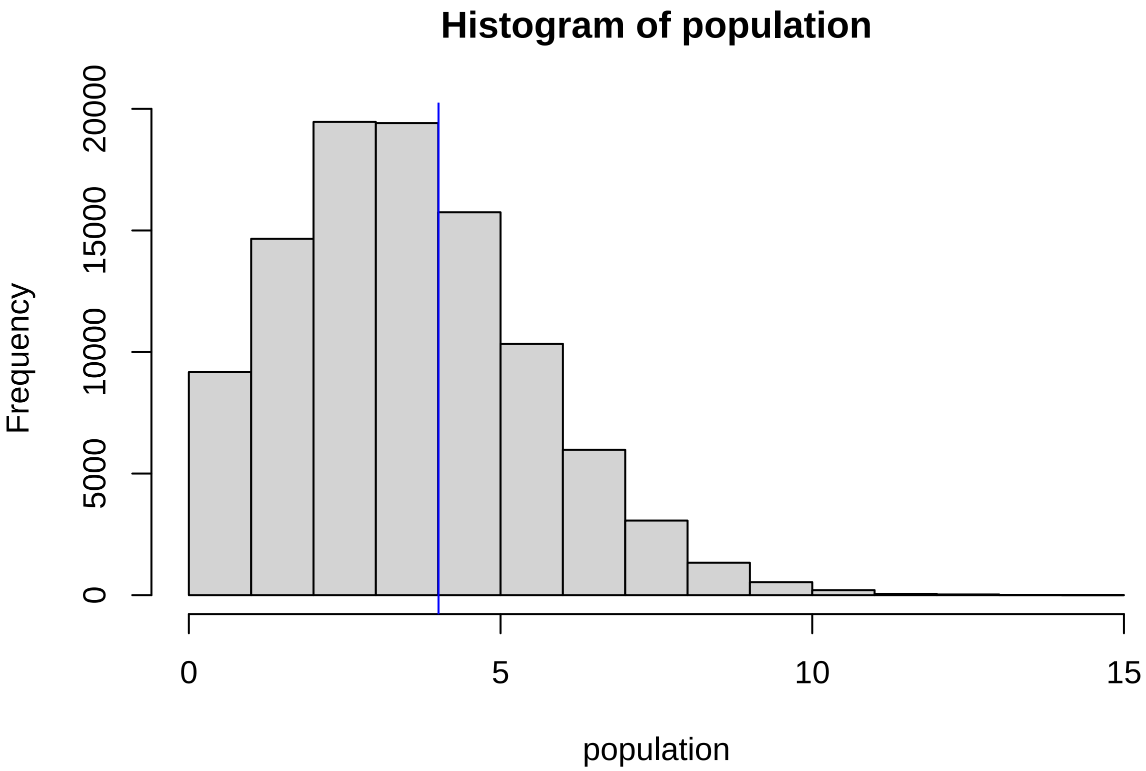
```
x = c(0:20)
y = dpois(x, 1)
plot(x, y, type = 'l', col = 'blue')

for (i in 2:10)
{
  lines(x, dpois(x, i), col = colors()[i+5])
}
legend(x = 18, y = 0.35, legend = c(1:10), col = c('blue', colors()[7:15]), pch = rep('l', 10))
```



3)

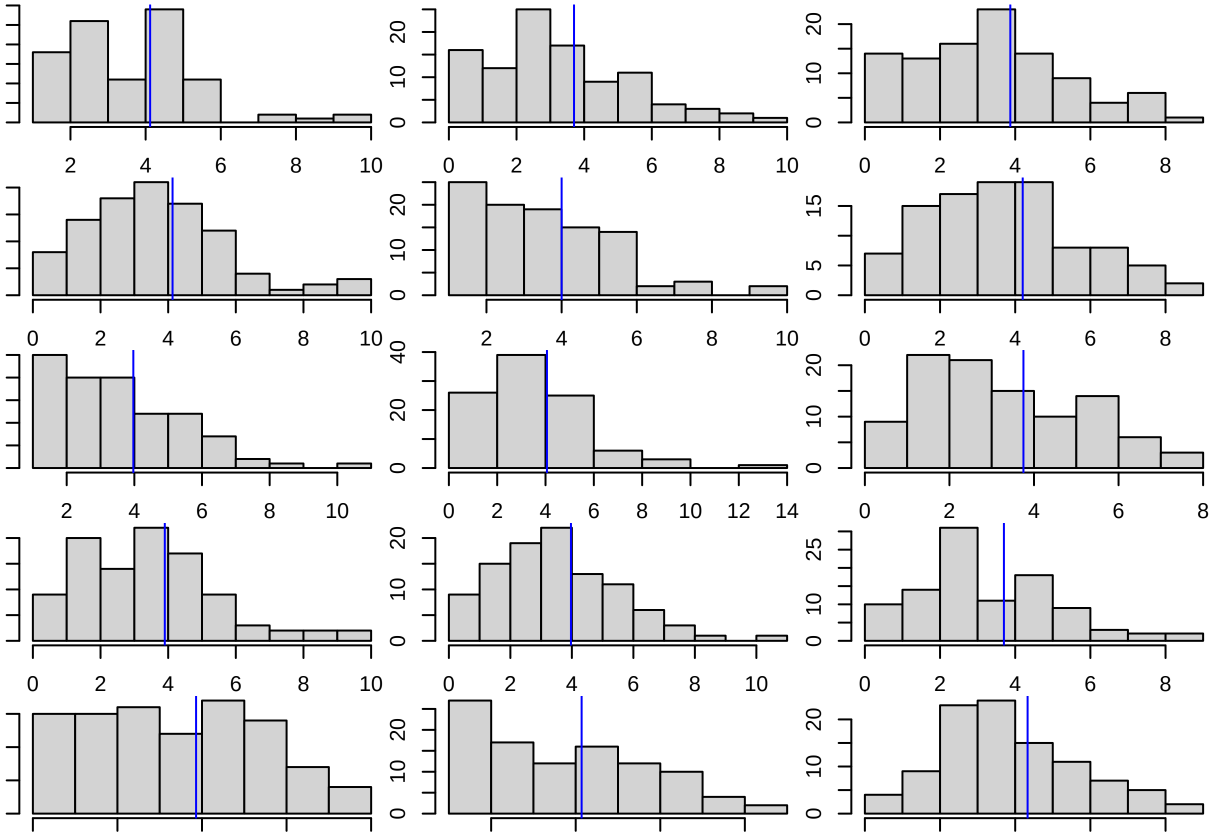
```
population <- rpois(100000, 4)
hist(population)
mean_value = mean(population)
abline(v = mean_value, col = 'blue')
```



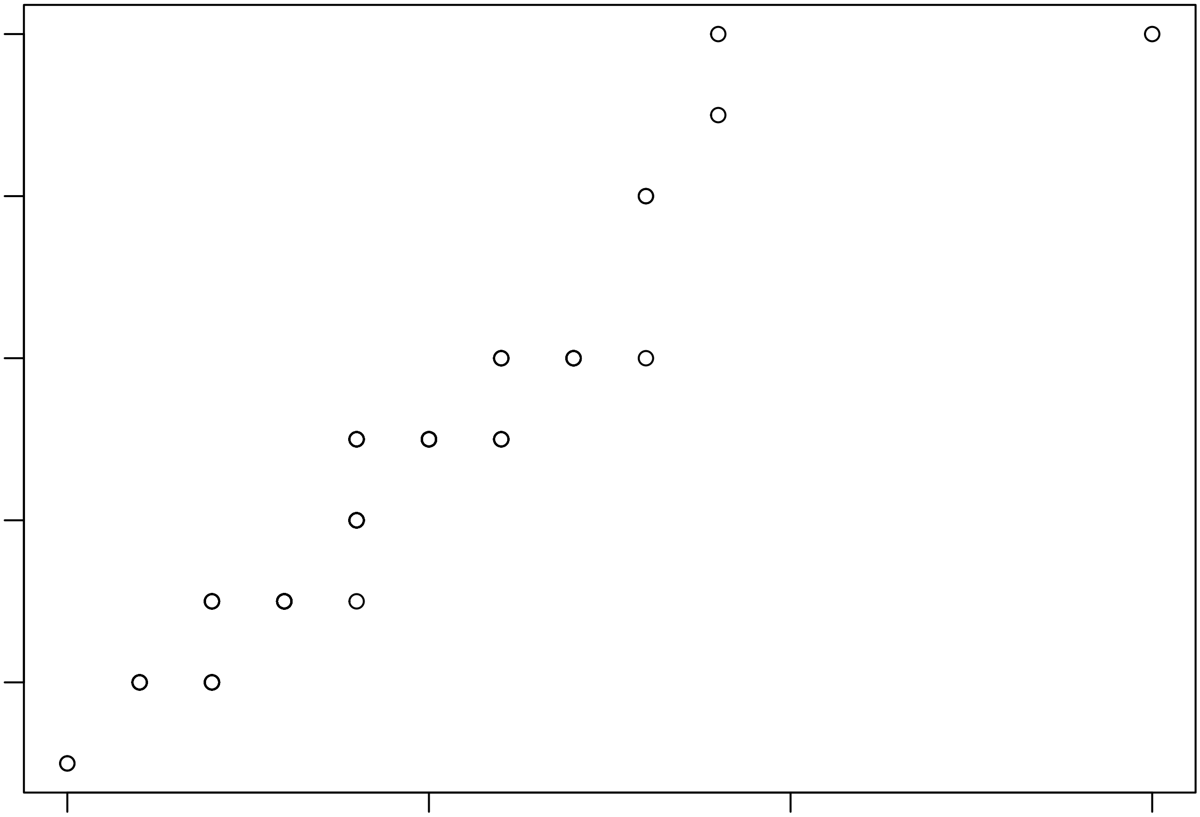
4)

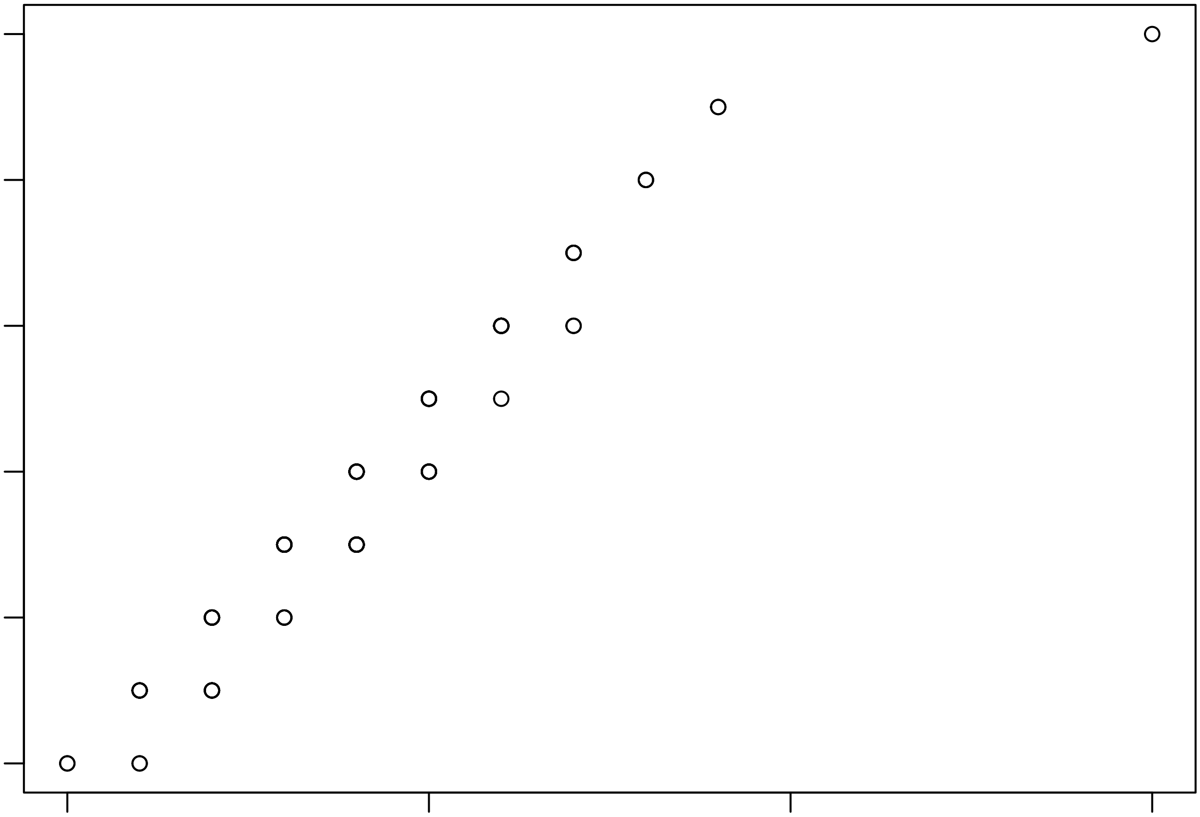
Samples and population have the same distribution.

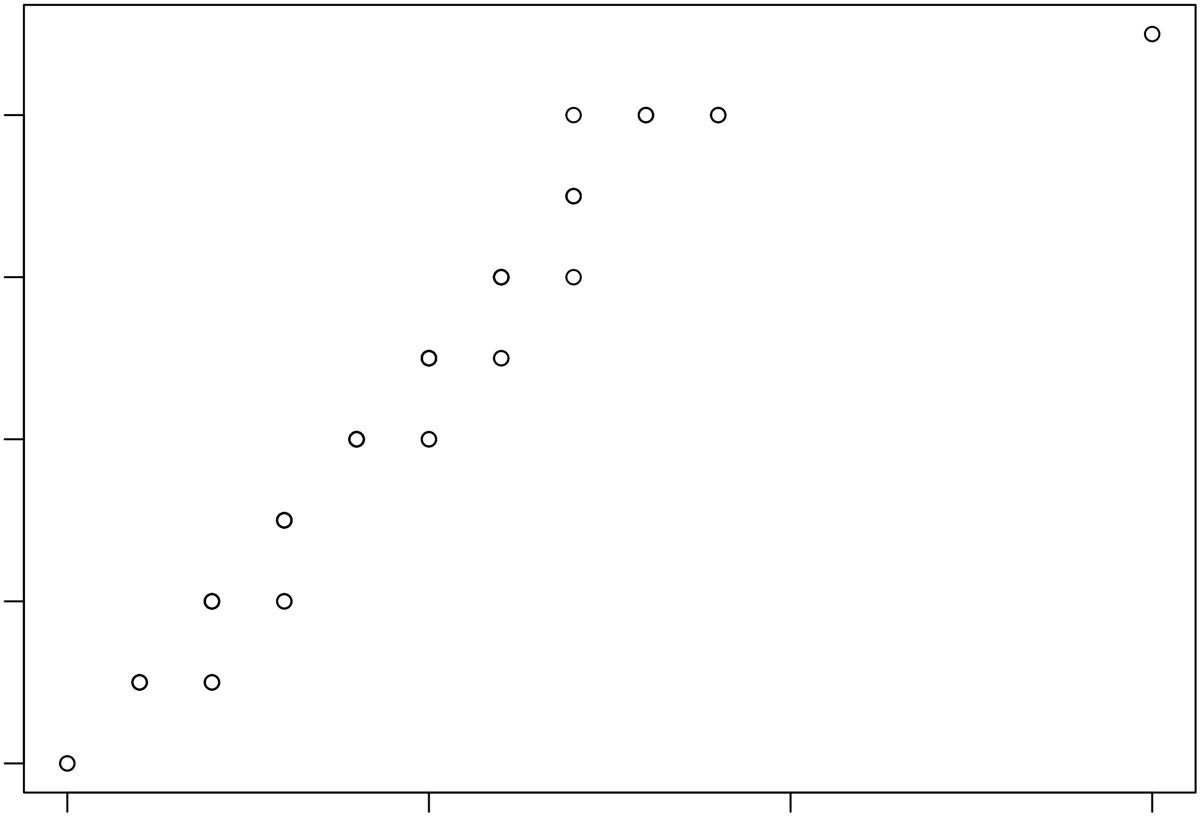
```r
samples <- list()
par(mfrow = c(5, 3), mar = c(1, 1, 1, 1))
for (i in 1 : 15)
{
  data <- sample(population, 100, replace = T)
  hist(data, main = '')
  abline(v = mean(data), col = 'blue')
  #print(data)
  samples[[i]] <-data
}
```

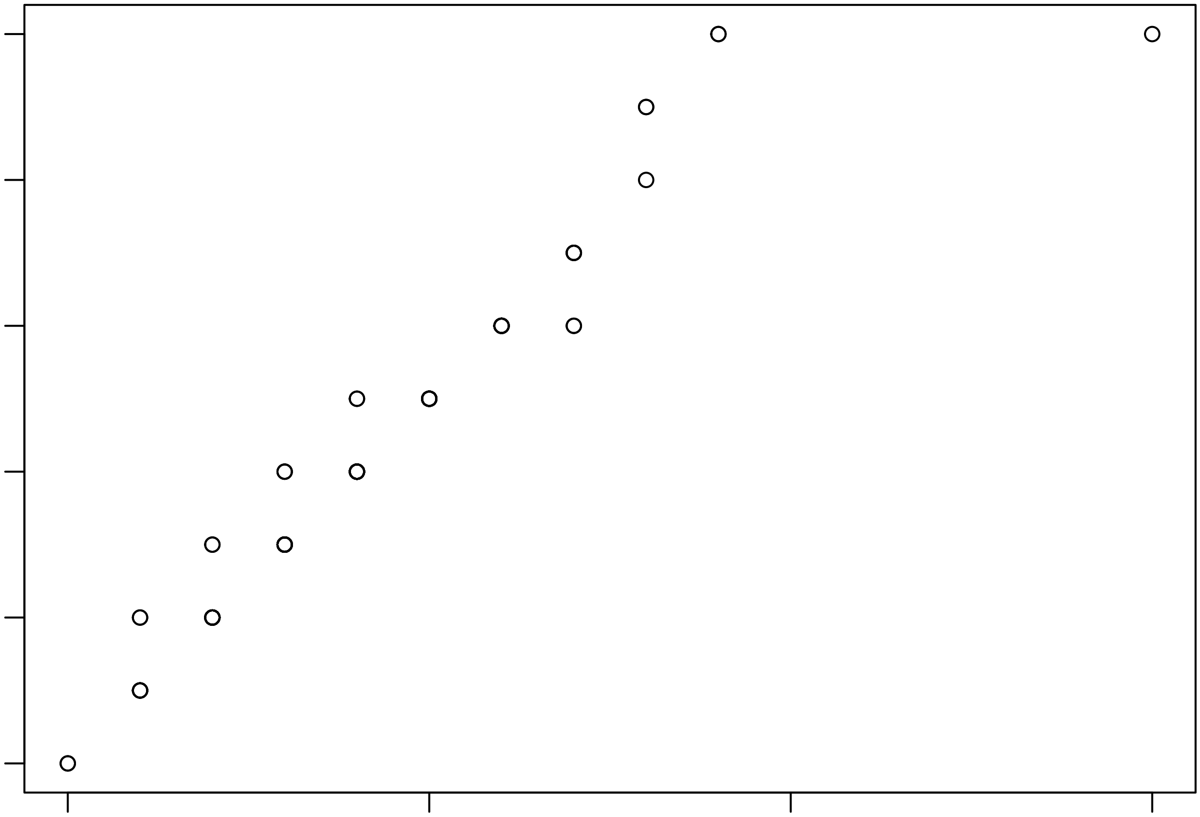


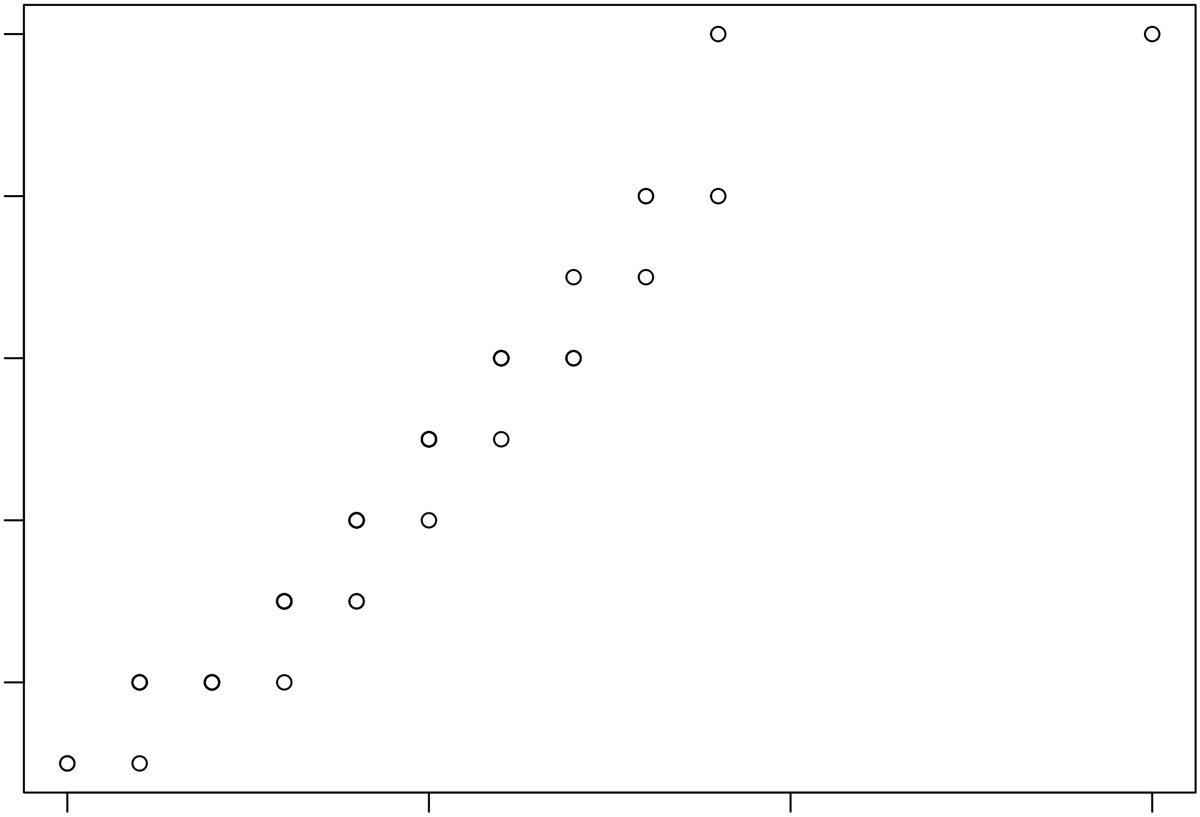
```
par(mfrow = c(1, 1))
#print (samples)
for (i in 1 : 15)
{
  #print(i)
  qqplot(population, samples[[i]])
}
```

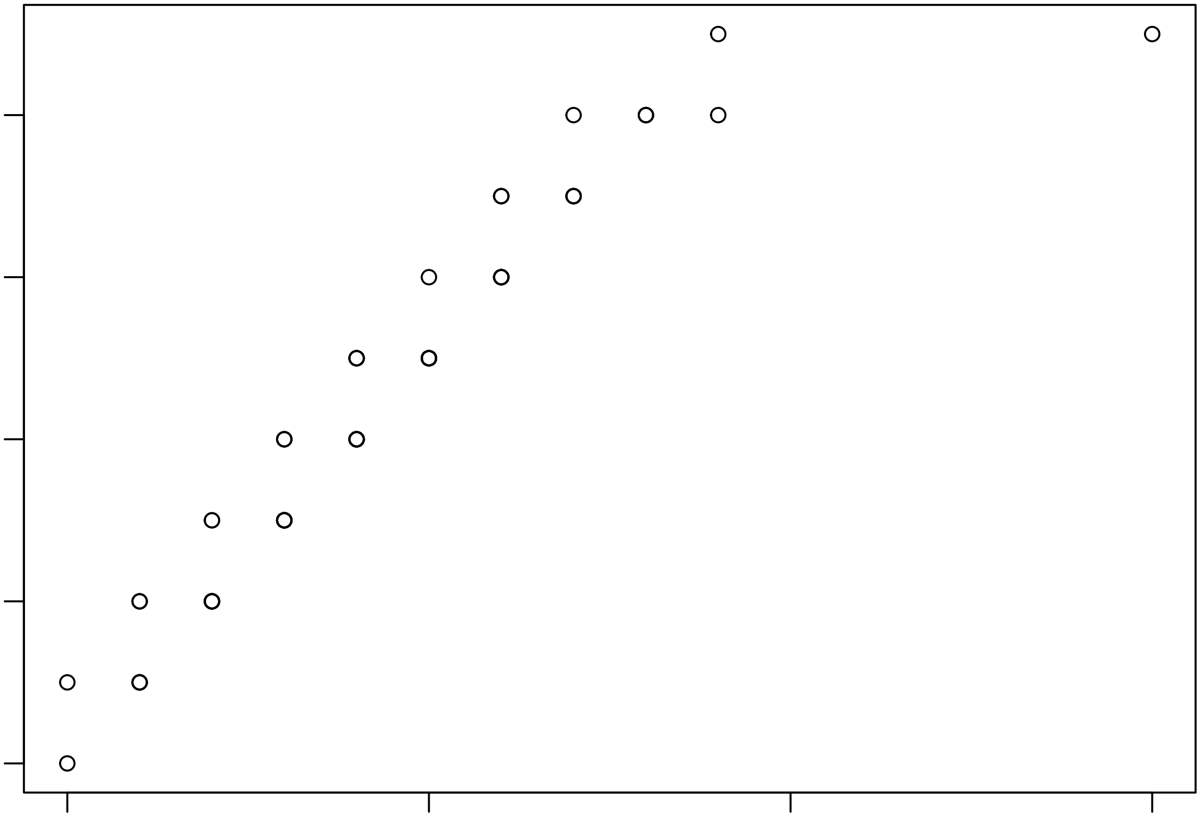


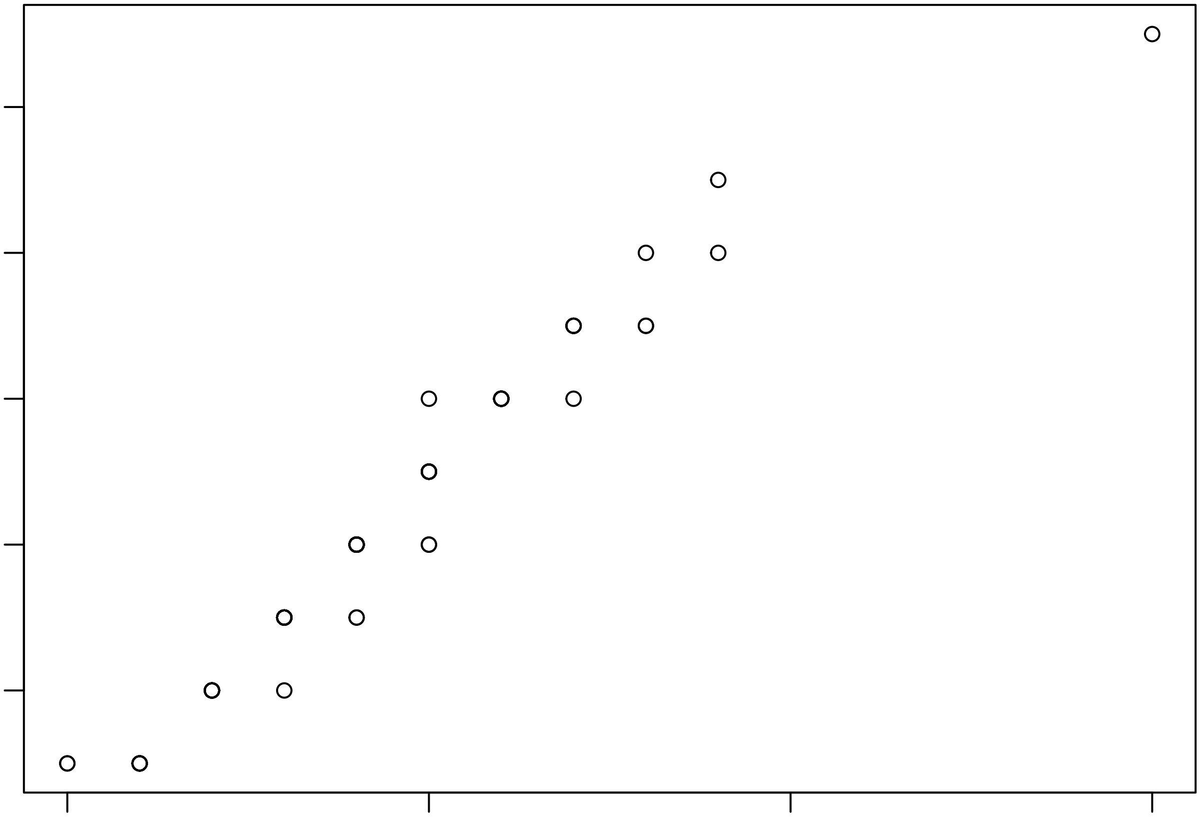


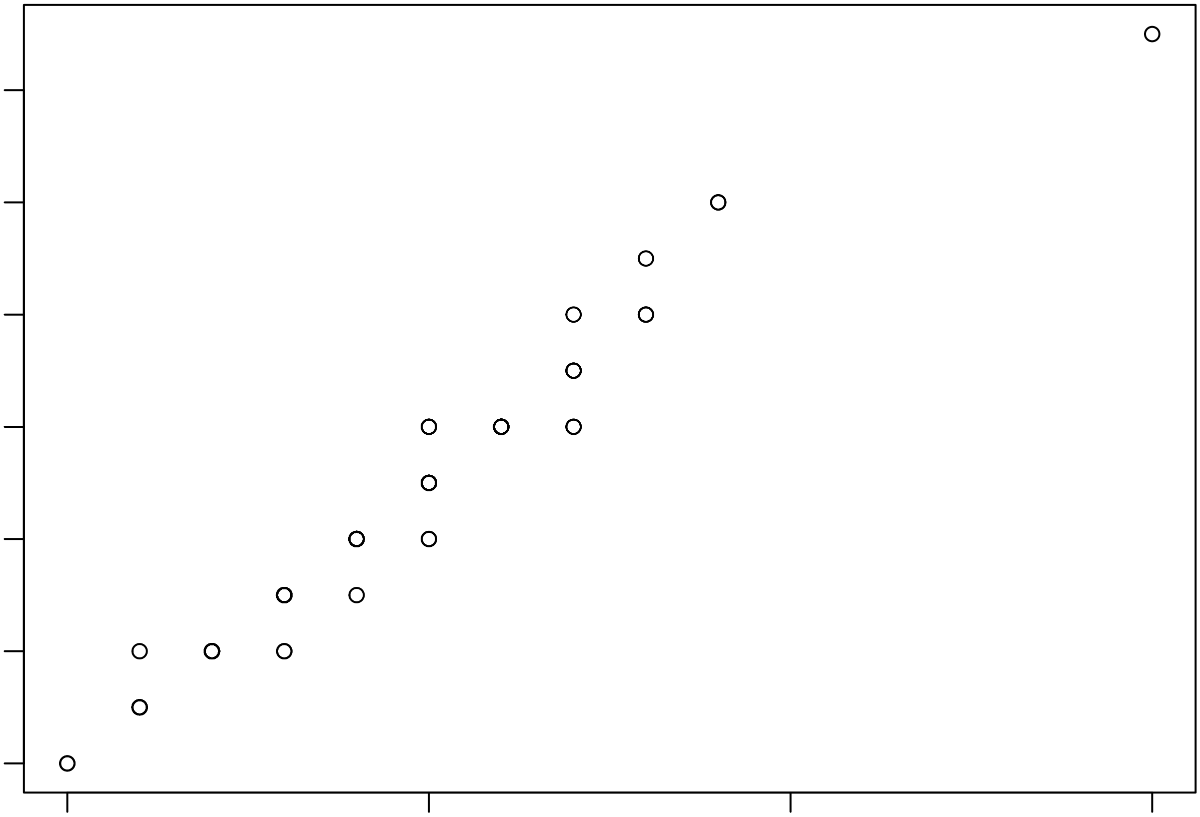


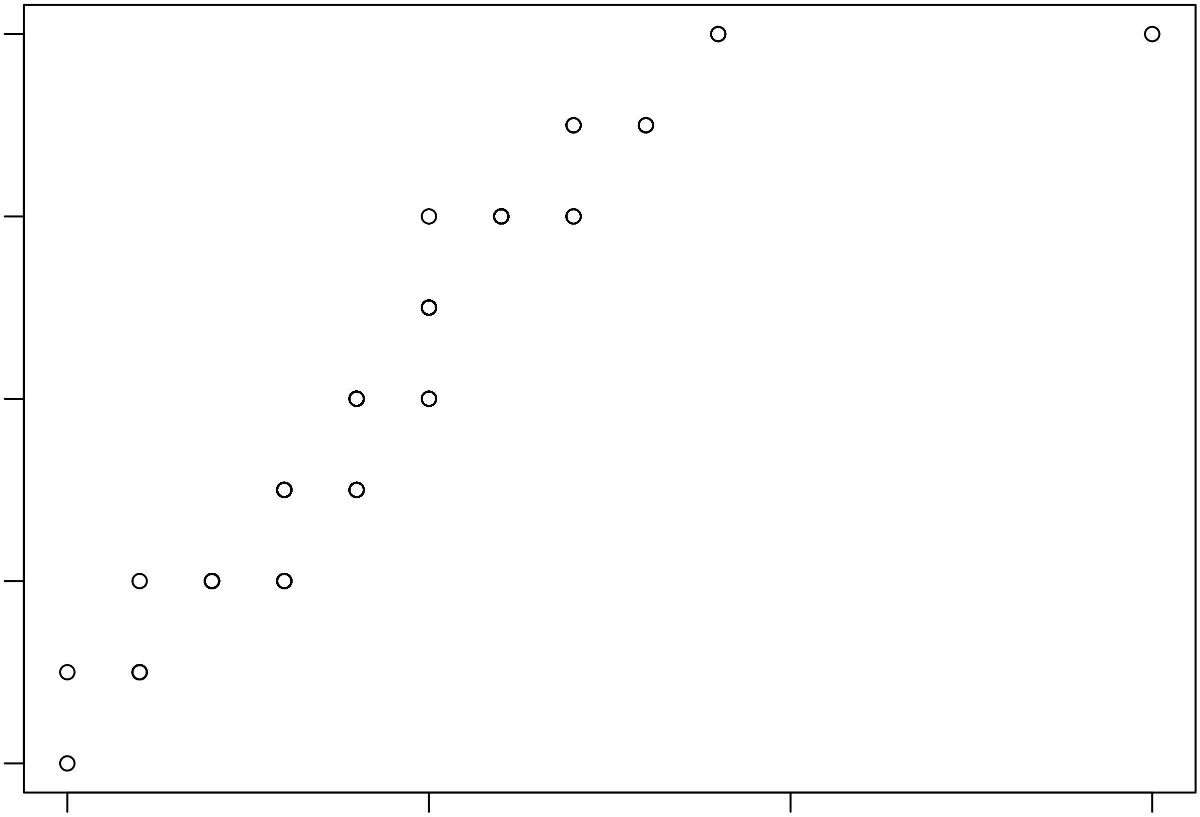


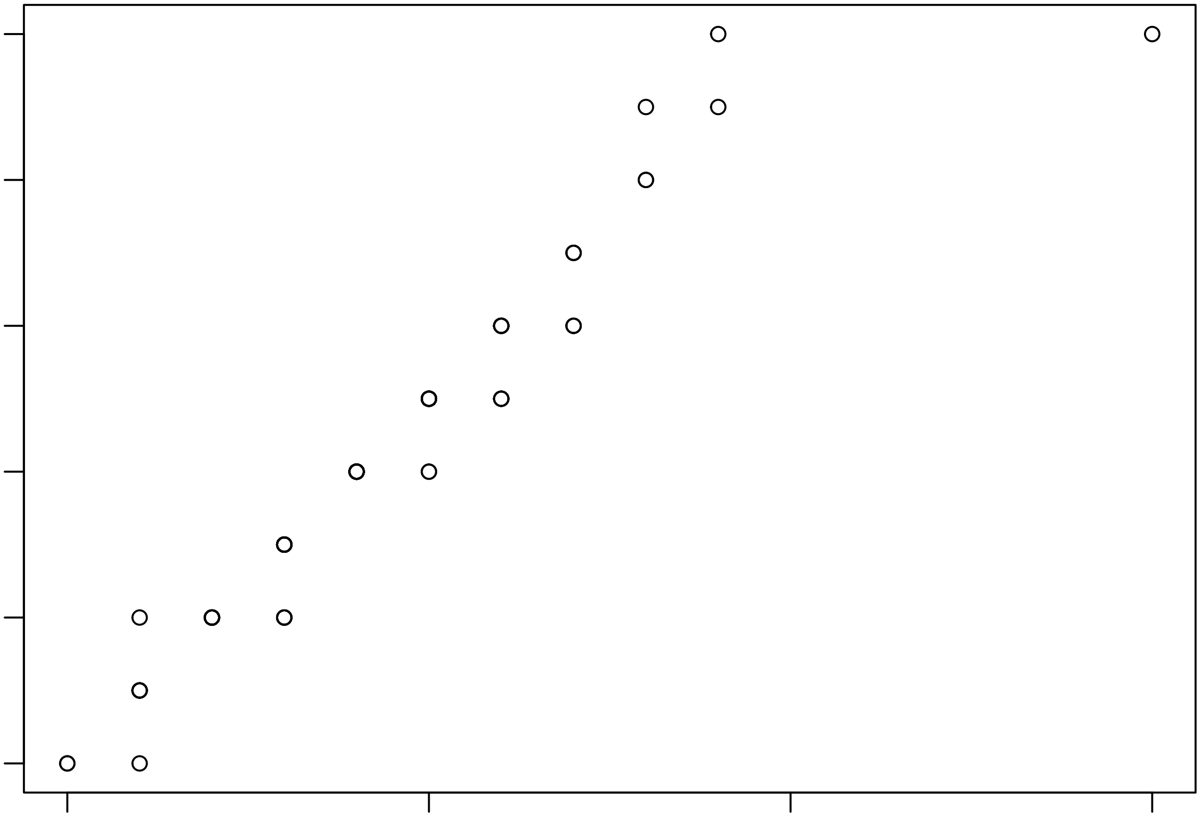


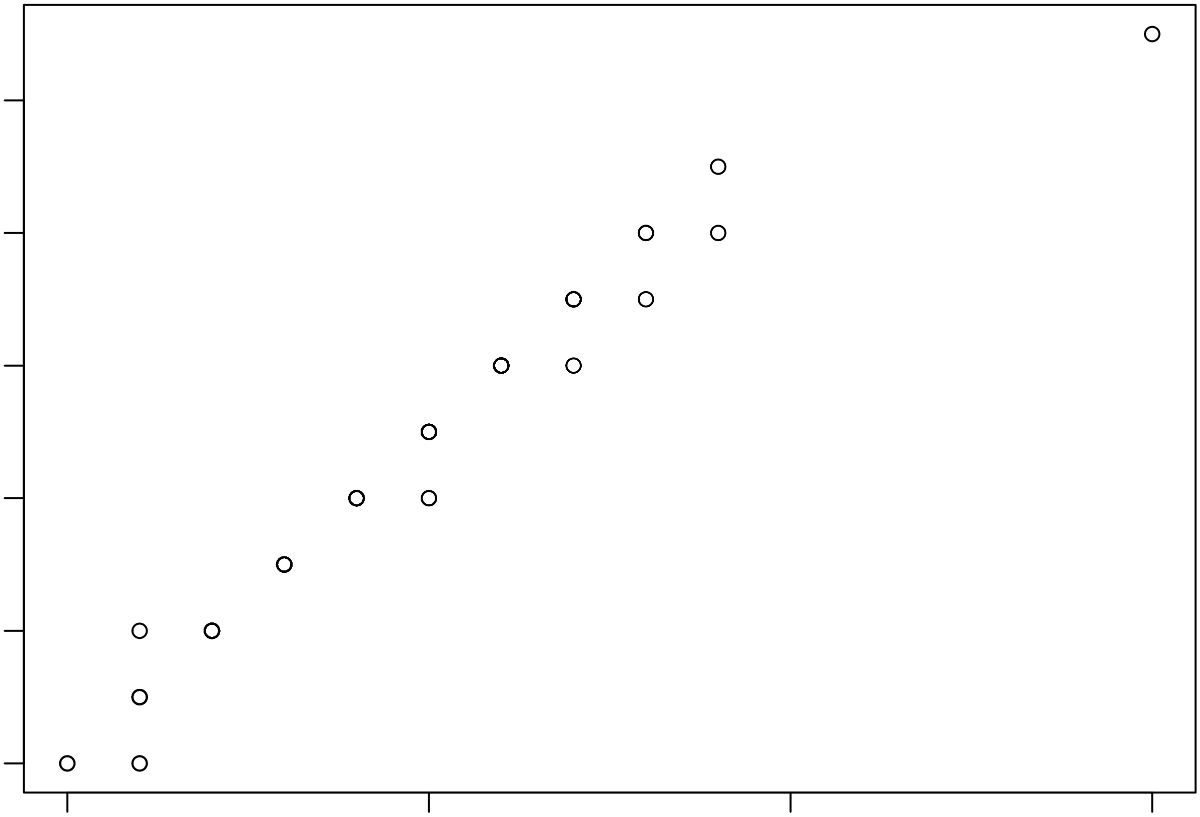


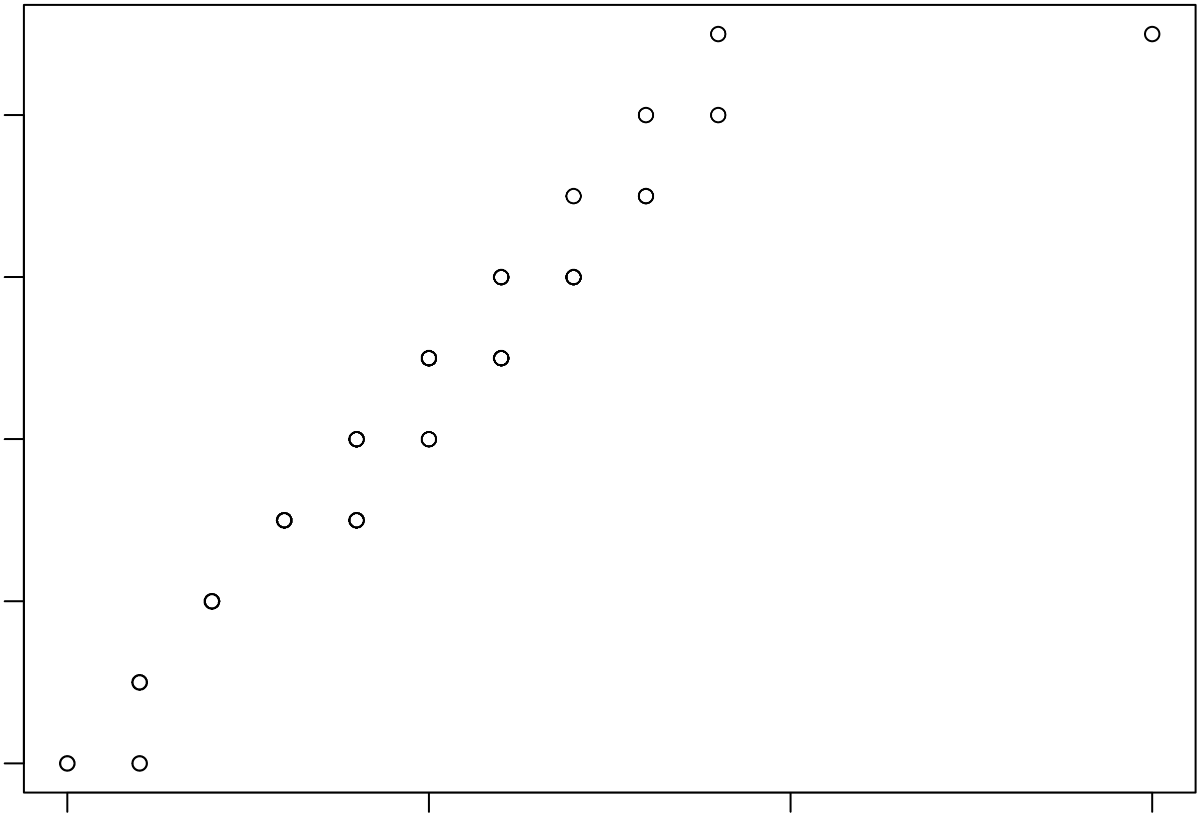


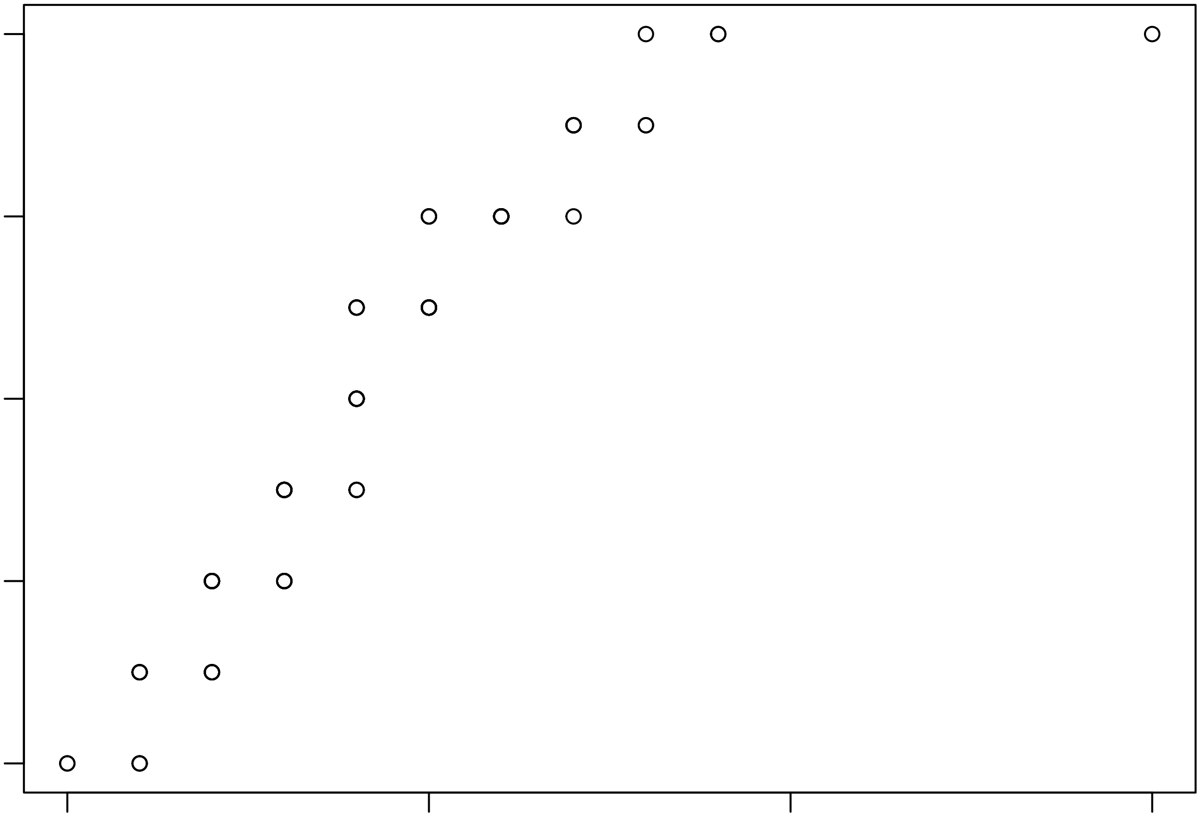


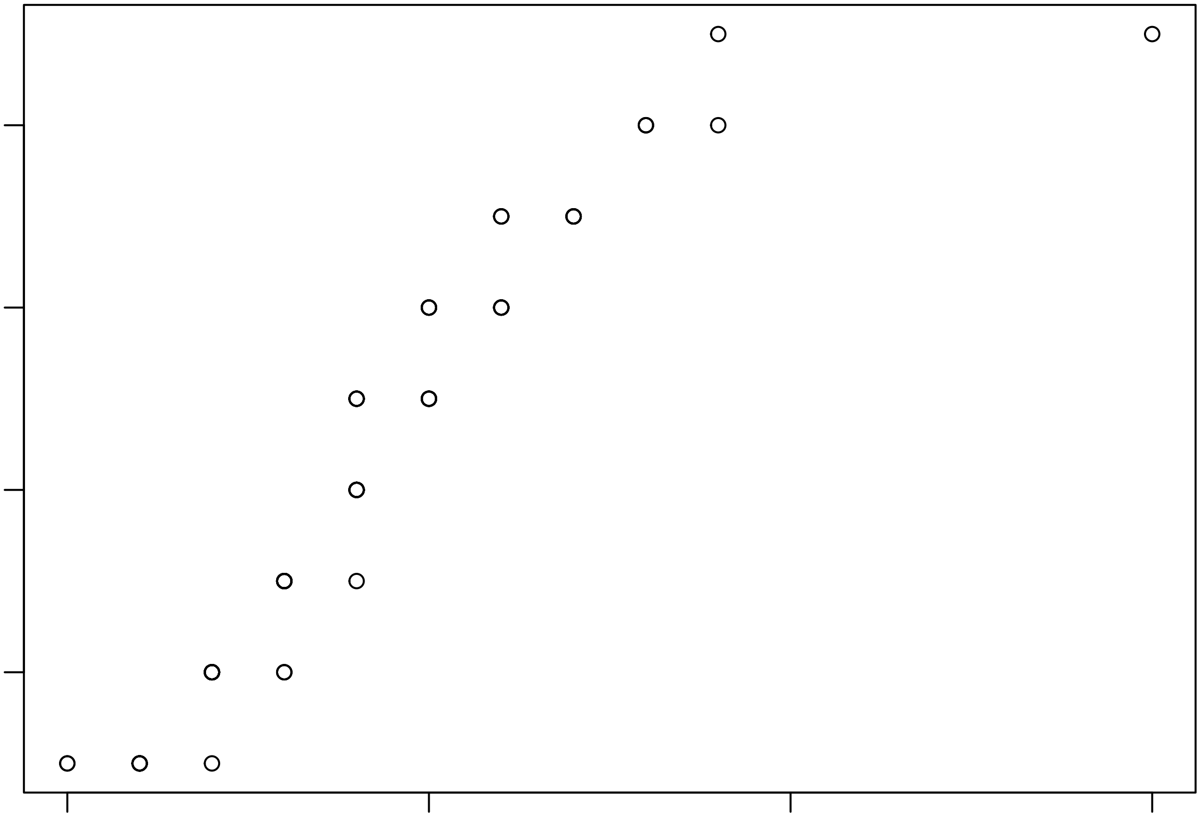


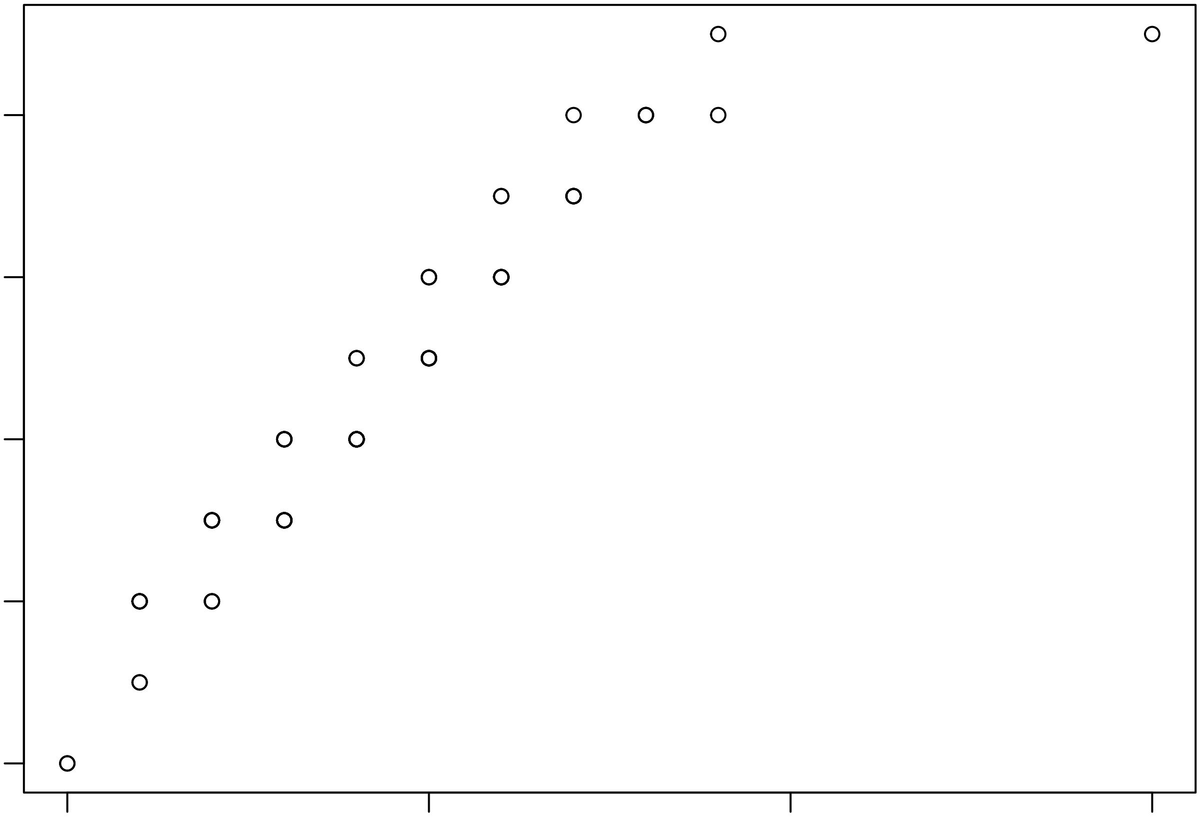








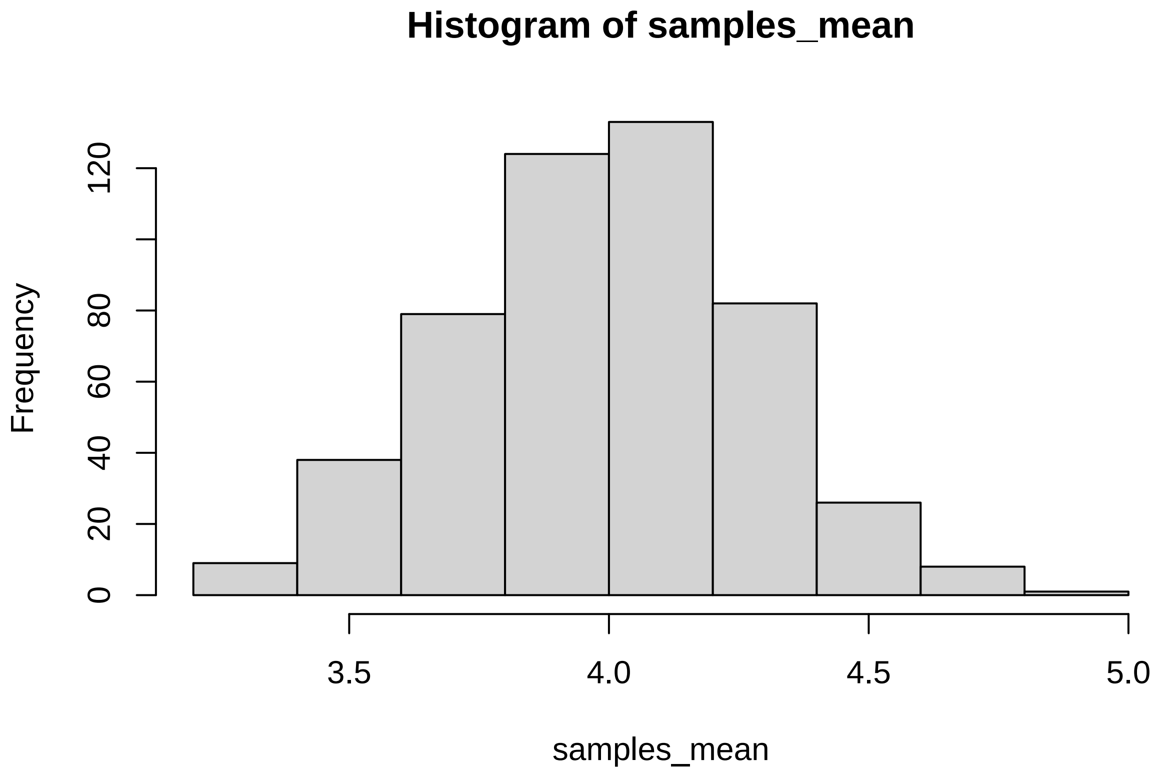




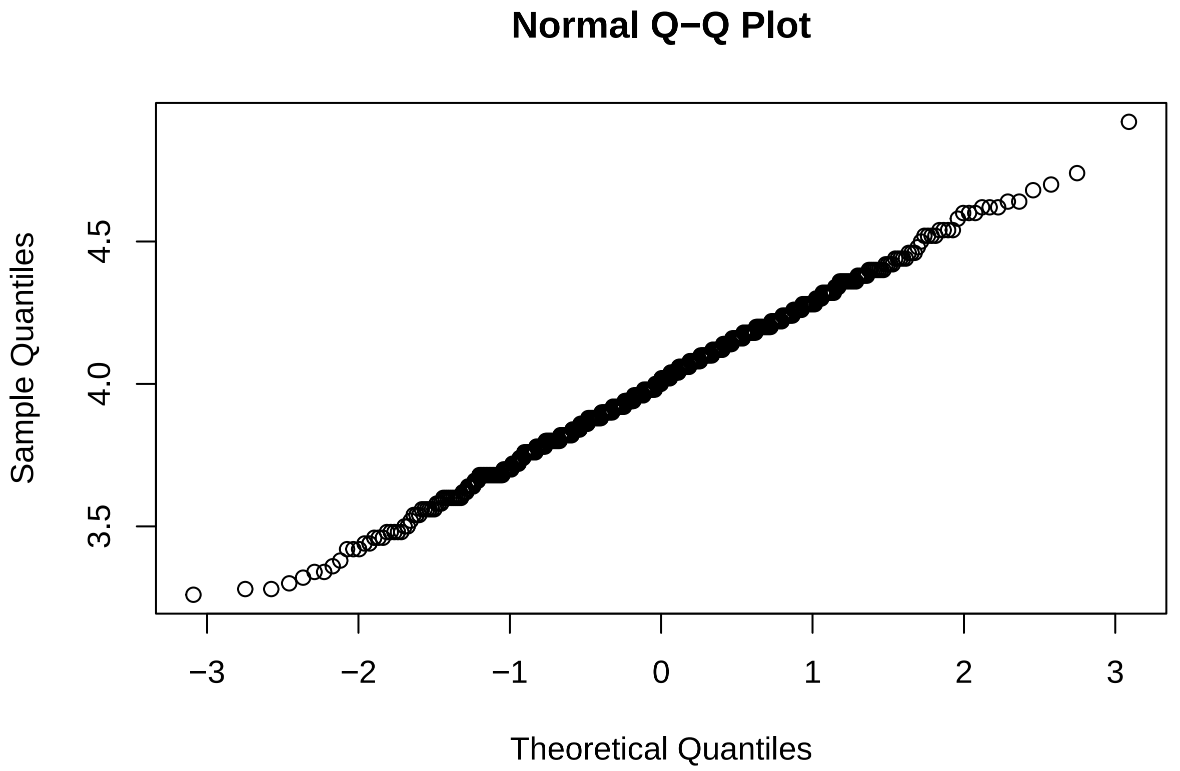
5)

As we know the central limit theorem (CLT) states that the distribution of sample means approximates a normal distribution as the sample size gets larger, regardless of the population's distribution so as we expect the histogram of samples_mean is similar to normal distribution.

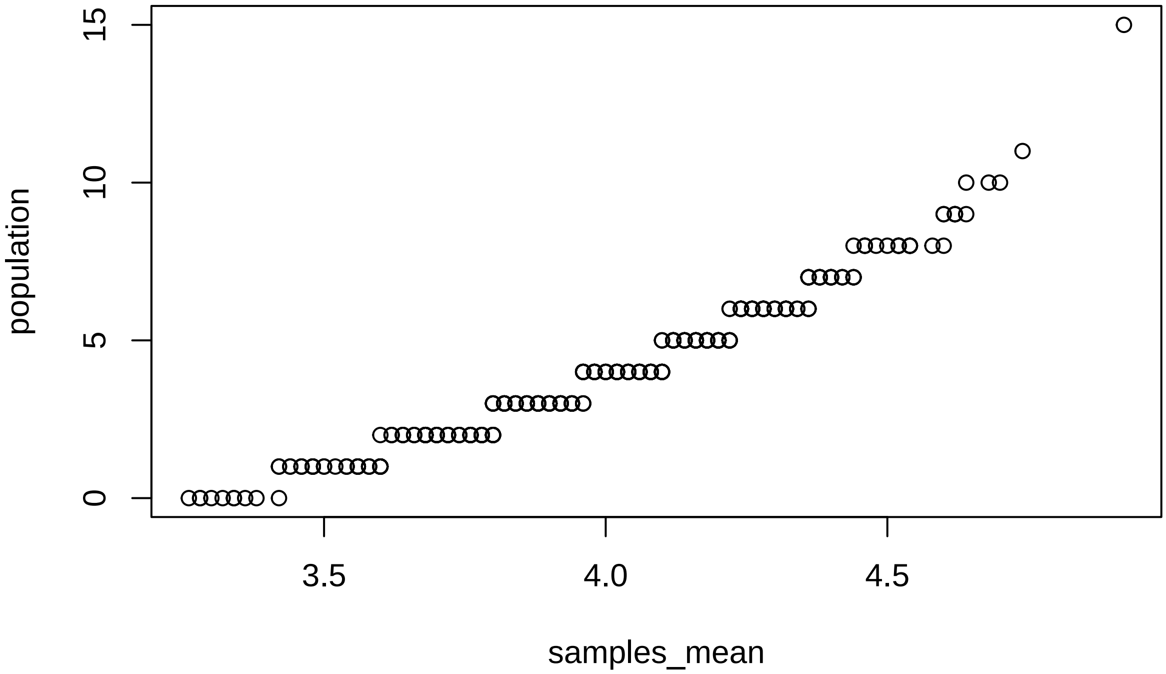
```r
samples_mean = c()
for (i in 1:500)
  samples_mean = c(samples_mean, mean(sample(population, 50, replace = T)))
hist(samples_mean)
```



```
qqnorm(samples_mean)
```



```
qqplot(samples_mean, population)
```



6)

No, based on CLT theorem, sampling distribution will always be similar to normal distribution if samples sizes be large enough.

7)

```
print(paste0("samples mean: ", mean(samples_mean)))
```

```
## [1] "samples mean: 4.00468"
```

```
se = sd(samples_mean)/sqrt(length(samples_mean))
print(paste0("standard error:, ", se))
```

```
## [1] "standard error:, 0.0127986486010057"
```

```
mu = mean(samples_mean)

print(paste0("sigma tilde: ", se * sqrt(length(samples_mean))))
```

```
## [1] "sigma tilde: 0.286186482919812"
```

```
print(paste0("mu tilde: ",  mu))
```

```
## [1] "mu tilde: 4.00468"
```

```
print(paste0("real mean: ", mean(population)))
```

```
## [1] "real mean: 4.0053"
```

```
print(paste0("real sd: ", sqrt(sum((population - mean(population)) ** 2) / length(population))))
```

```
## [1] "real sd: 2.00648745572954"
```

b) Confidence Intervals

8)

```
library(mosaicData)
```

```
## Warning: package 'mosaicData' was built under R version 4.1.3
```

```
heights = Galton$height
```

9)

As we expected 97% of confidence intervals contains the real mean.

```
real_mean = mean(heights)
func <- function(data)
{
  std = sd(data)
  err = qnorm(0.985) * std / sqrt(length(data))
  return (abs(real_mean - mean(data)) <= err)
}

samples = replicate(10000, sample(heights, 50, replace = T), simplify = F)

res <- unlist(lapply(samples, func))
print (sum(res) / length(res) * 100)
```

```
## [1] 96.37
```

10)

Size condition is not satisfied, it should be 30 at least. so we can't get 90% accuraty.

```
func <- function(data)
{
  std = sd(data)
  err = qnorm(0.95) * std / sqrt(length(data))
  return (abs(real_mean - mean(data)) <= err)
}

samples = replicate(10000, sample(heights, 10, replace = T), simplify = F)

res <- unlist(lapply(samples, func))
print (sum(res) / length(res) * 100)
```

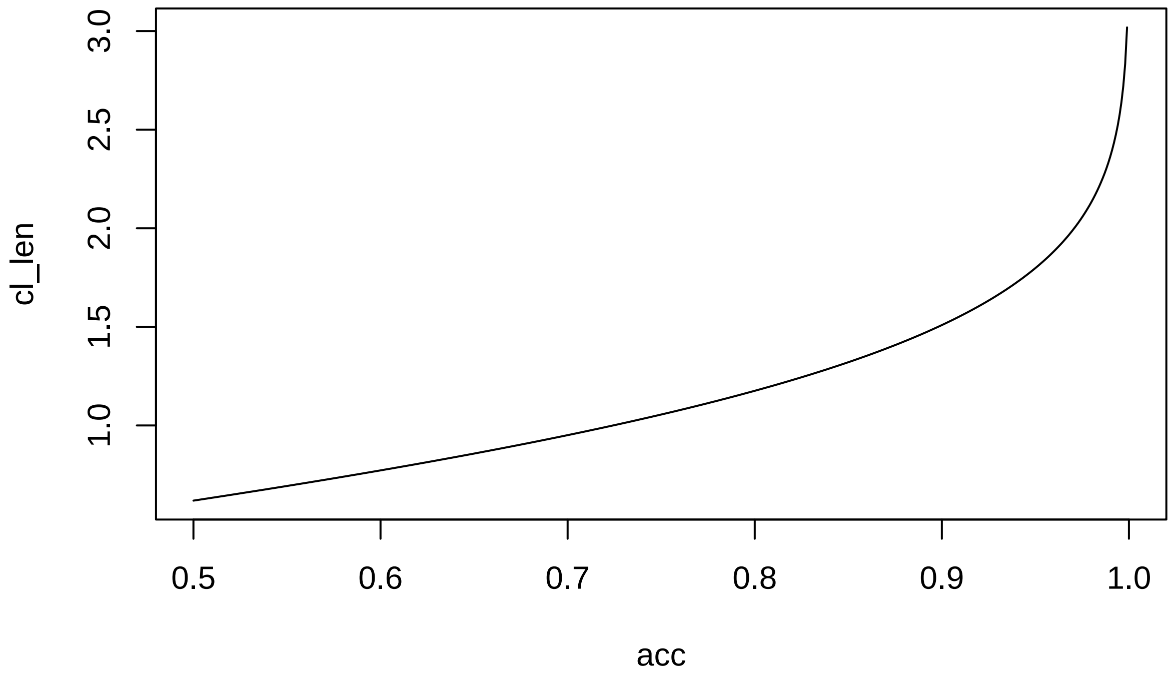
```
## [1] 86.25
```

11)

```
calculate_ci <-function(acc, data)
{
  if (length(data) < 30)
    print("sample size should be at least 30")
  std = sd(data)
  err = qnorm(1 - (1 - acc)/2) * std / sqrt(length(data))
  return (c(mean(data) - err, mean(data) + err))
}
```

12)

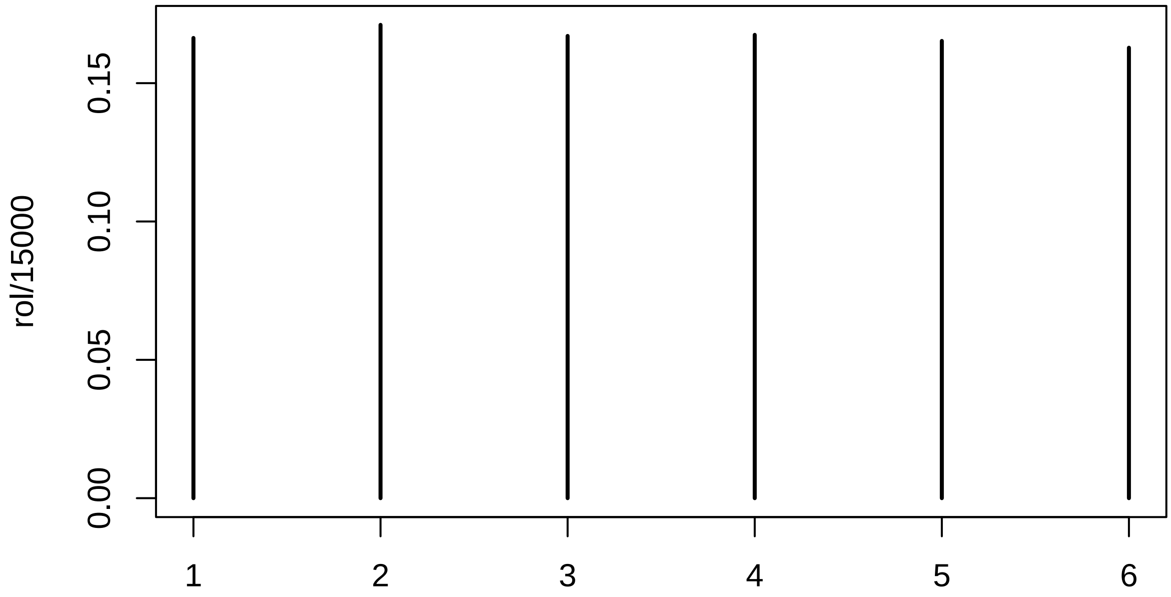
```
data = sample(heights, 60, replace = T)
acc = seq(0.5, 1, by = 0.001)
cl = (lapply(acc, calculate_ci, data = data))
cl_len =unlist(lapply(cl, function(int) int[2] - int[1]))
plot(acc, cl_len, type = 'l')
```



c)

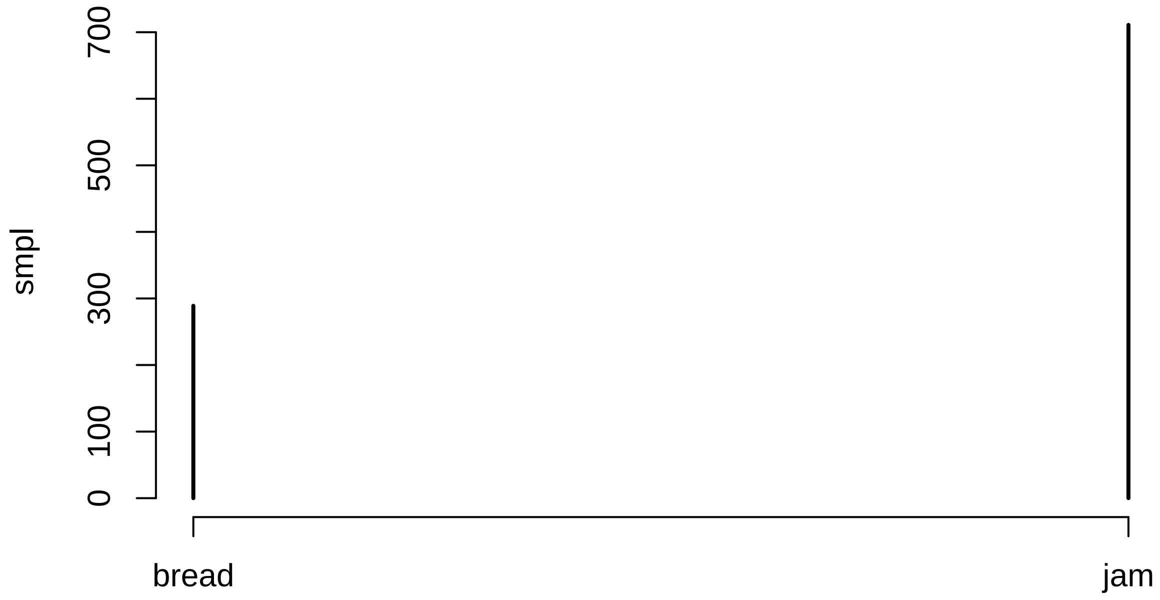
13)

```
dice <- factor(1:6)
rol <- table(sample(dice, 15000, replace = T))
plot(rol / 15000)
```



14)

```
prob = c(22, 10)
prob = prob / sum(prob)
smpl = table(sample(c("jam", "bread"), 1000, prob = prob, replace = T))
plot(smpl)
```



15)

Ii is almost equal to theoretically probablity : 10 / 36 ~ 0.27

```
dice = list()
dice$f <- sample(1:6, 100000, replace = T)
dice$s <- sample(1 : 6, 100000, replace = T)
res <- dice$f + dice$s > 8
print(sum(res) / length(res))
```

```
## [1] 0.27807
```

d)

16)

We can reject p0.

p_0 : average humid is 50%

p_h : average humid > 50%

```
my_city <- Weather[c(Weather$city == 'Beijing'),]
avg_humid = my_city$avg_humidity
mean_humid = mean(avg_humid)
se = sd(avg_humid) / sqrt(length(avg_humid))
p_value = pnorm(mean_humid, mean = 50, sd = se, lower.tail = F)
print(p_value)
```

```
## [1] 2.629439e-05
```

17)

I don't reject the p_0, my inference problem is size of the sample, it is very large.

18)

p_0 : shrimp = 31%

p_h : shrimp < 31%

We can't reject p_0 against p_h because p-value = 0.96 and this mean restaurant recipe claimes false.

```
library(MASS)
mean_shrimp = mean(shrimp)
se = sd(shrimp) / sqrt(length(shrimp))
p_value = pnorm(mean_shrimp, mean = 31, sd = se)
print(p_value)
```

```
## [1] 0.9662565
```

19)

```
two_tail_z_dist_mean_hyp_test <- function(data, null_h, alpha)
{
  if (length(data) < 30)
    print("Warning: sample_size is too small")
  se = sd(data) / sqrt(length(data))
  p_value = pnorm(mean(data), null_h, sd = se)
  p_value = min(p_value, 1 - p_value)
  if (2 * p_value < alpha)
    return (F)
  return (T)
}
```

20)

```
print (two_tail_z_dist_mean_hyp_test(avg_humid, 50, 0.05))
```

```
## [1] FALSE
```

```
print (two_tail_z_dist_mean_hyp_test(shrimp, 31, 0.05))
```

```
## [1] "Warning: sample_size is too small"
## [1] TRUE
```