Random numbers, sampling and simulations: II

M P Gururajan, Hina A Gokhale and Dayadeep Monder

Indian Institute of Technology Bombay, Mumbai

In this session, we will continue to learn some R commands for sampling and simulations. One of the good resources for what follows is *simpleR: using R for introductory statistics* by John Verzani which is available, among other places, at the following URL:

`https://cran.r-project.org/doc/contrib/Verzani-SimpleR.pdf`.

1 Sampling from different distributions

In the last session, we learnt the R commands for several random variables. In this session, we will learn some more. For the sake of completion, I am using the earlier table, and am extending it by including the new commands: the table consists of the special random variables and the R commands for generating the density, distribution function, quantile function and the random deviate. This will serve as a ready-reckoner and the help command can be used to generate more information – such as the inputs to these function calls and the parameters.

Random Variable	**Density**	**Distribution function**	**Quantile function**	**Random Variate**
Binomial	dbinom	pbinom	qbinom	rbinom
Poisson	dpois	ppois	qpois	rpois
Hypergeometric	dhyper	phyper	qhyper	rhyper
Uniform	dunif	punif	qunif	runif
Normal	dnorm	pnorm	qnorm	rnorm
Exponential	dexp	pexp	qexp	rexp
Gamma	dgamma	pgamma	qgamma	rgamma
Chi-Square	dchisq	pchisq	qchisq	rchisq
Student t	dt	pt	qt	rt
F	df	pf	qf	rf
Cauchy	dcauchy	pcauchy	qcauchy	rcauchy
Log normal	dlnorm	plnorm	qlnorm	rlnorm
Logistic	dlogis	plogis	qlogis	rlogis

Of course, this table is not complete. Using the `help` command, one can get more information on the available distributions:

```
help(distribution)
```

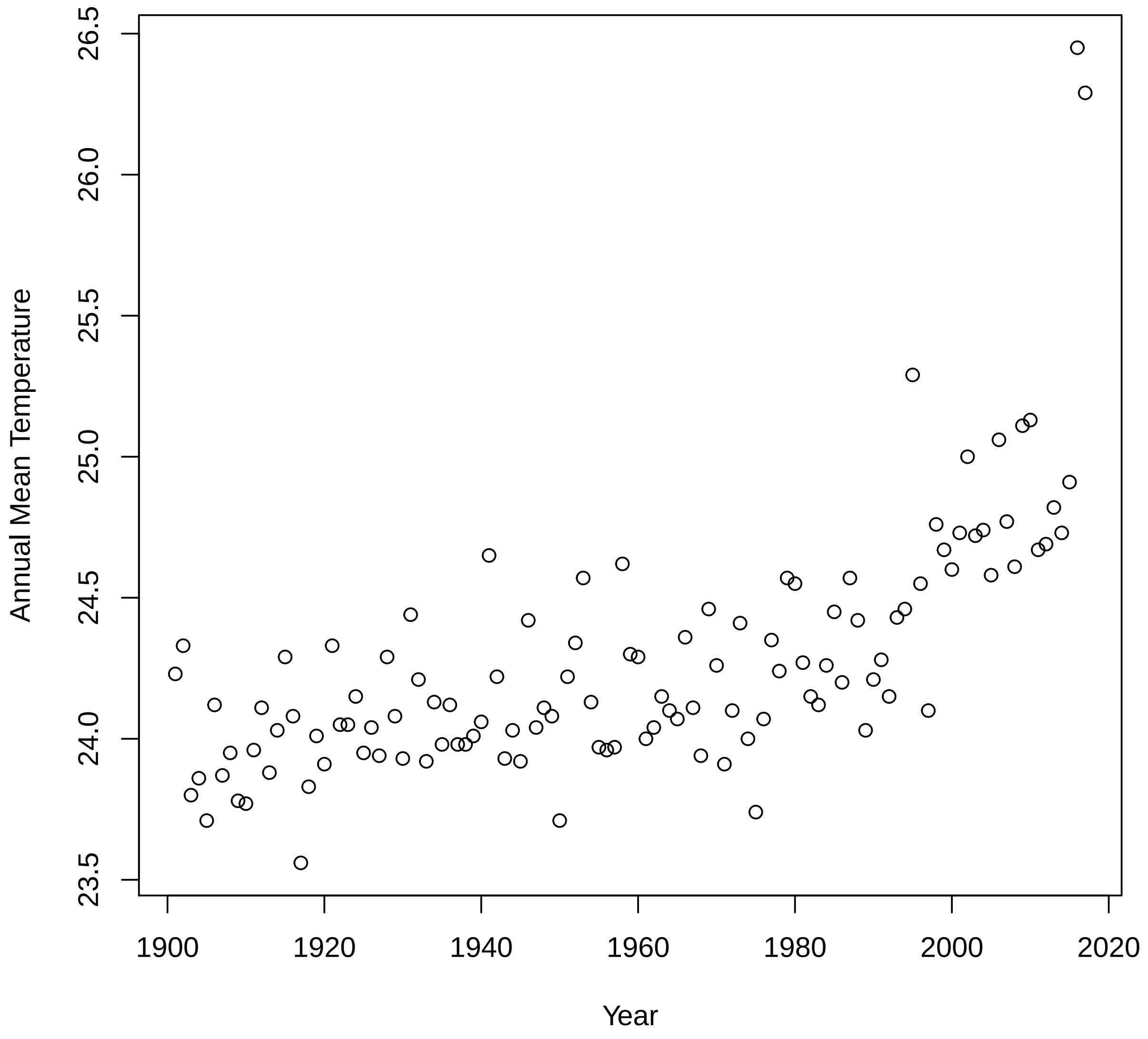
As you can see from the help, it is also possible to use R to work with beta, geometric, multinomial, negative binomial and weibull distributions. The help command also indicates some less common distributions of test statistics available in R; we will not use them in this course; but, it is useful for you to be aware of their availability.

2 Sampling from a given data

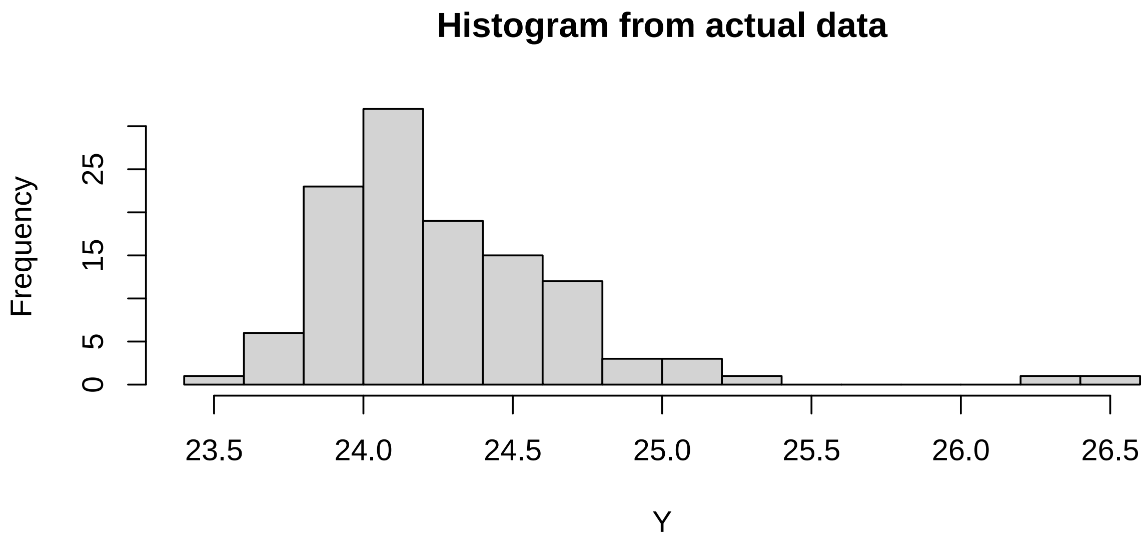
In the last tutorial, we saw how to use the `sample` command to sample with and without replacement. The same command can be used to sample from a given data. This is useful to carry out what is known as bootstrapping that we will learn later.

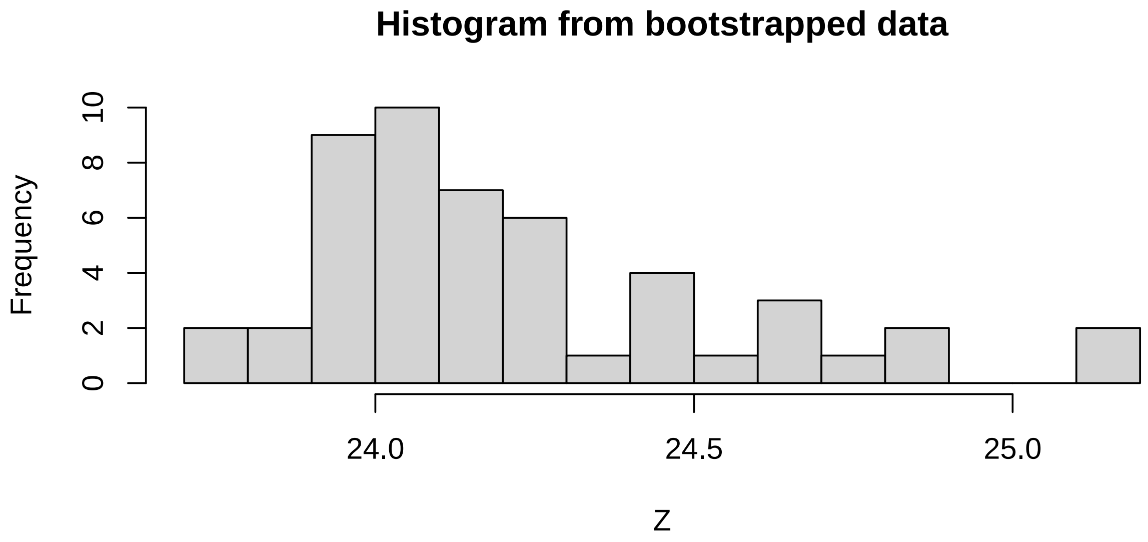
Let us consider an example. The data file `IndiaMeanTemperature.csv` contains the monthly, seasonal and annual mean temperature of the Indian continent from the year 1901 to 2017. Let us load this data and plot the annual mean temperature. Then, we plot two histograms: one is from the actual data and the other is by sampling the given data – we pick 50 data points with replacement.

```r
setwd("/home/guru/Desktop/Teaching/MM217/RTutorial5")
X <- read.csv("IndiaMeanTemperature.csv")
plot(X$YEAR,X$ANNUAL,xla="Year",ylab="Annual Mean Temperature")
Y <- X$ANNUAL
Z <- sample(Y,50,replace=TRUE)
par(mfrow=c(2,1))
hist(Y,breaks=20,main = "Histogram from actual data")
hist(Z,breaks=20,main = "Histogram from bootstrapped data")
```



Annual Mean Temperature
23.5
24.0
24.5
25.0
25.5
26.0
26.5
1900
1920
1940
1960
1980
2000
2020
Year



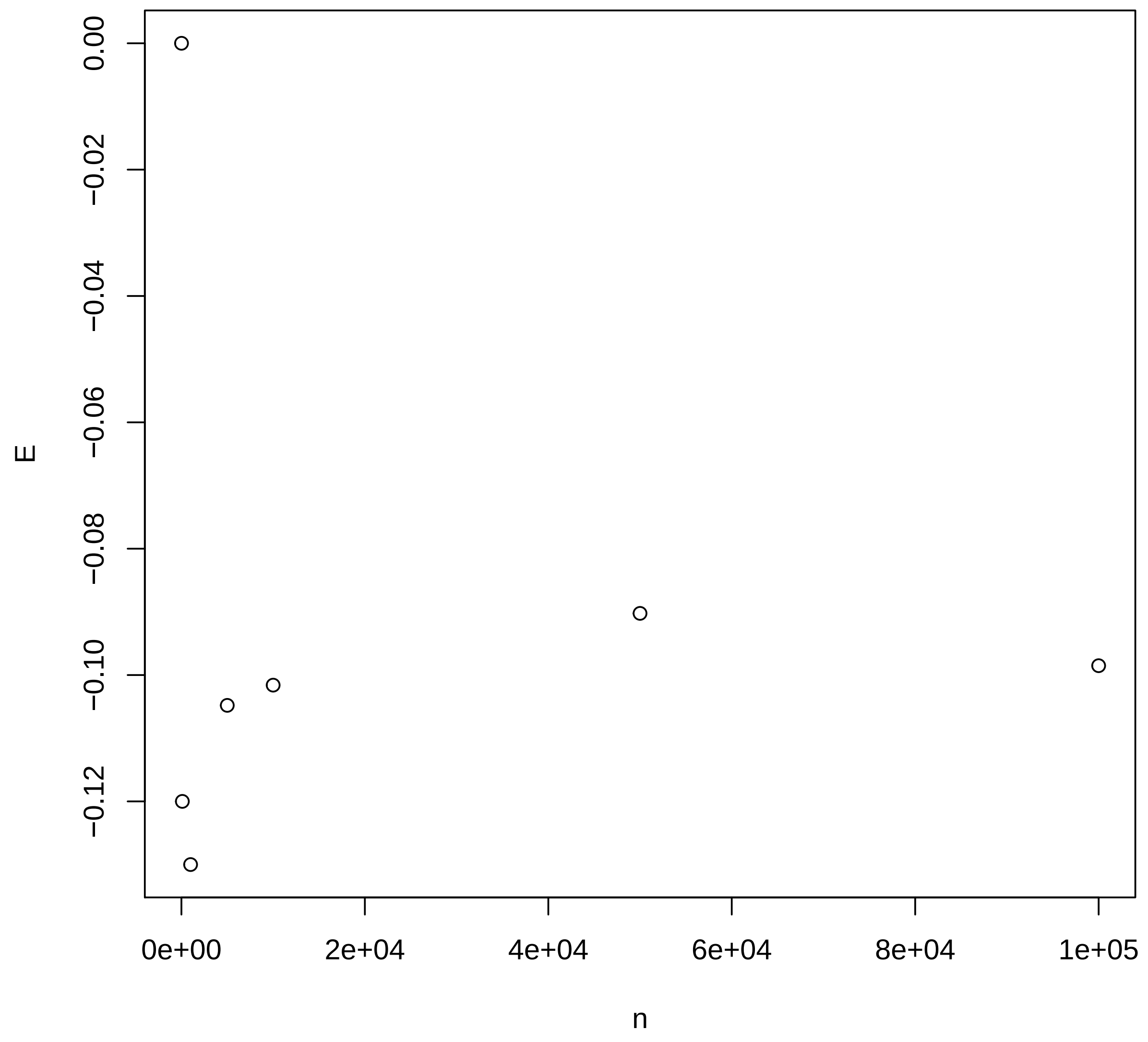


3 Simulations

You have already seen one example of simulations when you used the R script to test the central limit theorem. In simulations, we generate random numbers and use them to carry out virtual experiments. Hence, this becomes a cost effective way to study problems. Simulations such as Monte Carlo simulations can also be used to carry out integrations (typically multi-dimensional and are otherwise hard to carry out).

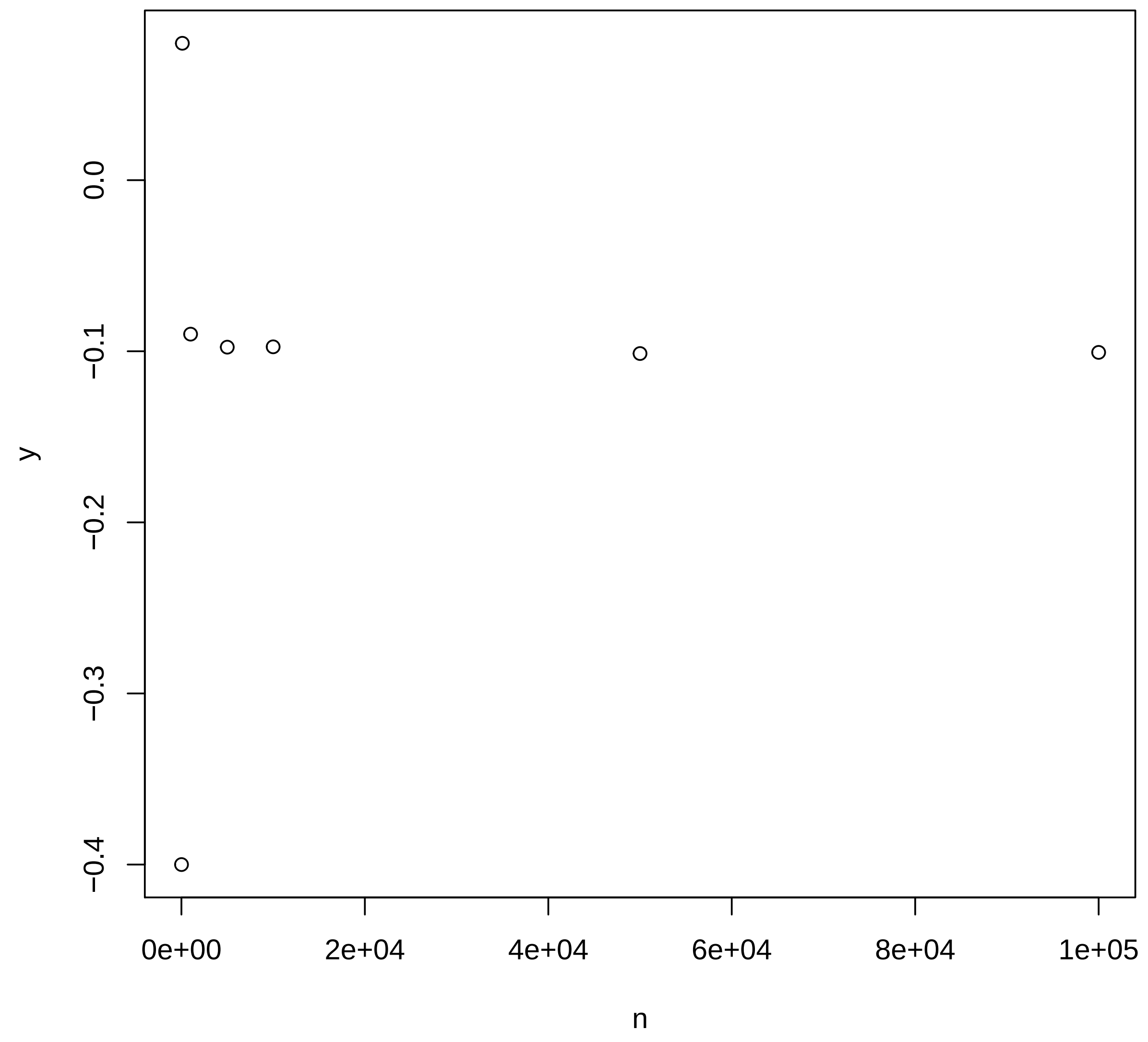
Let us consider the tossing of an unfair coin. We want to calculate the expectation value in such a scenario assuming -1 for head and +1 for tail. Simulations is one way of doing it. The following script carries out the simulation of expectation value of the toss of an unfair coin – which returns head 55% of the times instead of 50. Note that if the toss is made N times, then, the expected value is $-0.1N$. In the simulation, as the n value increases, we do see that the simulation performs better and by N = 10000, converges to -0.1.

```
n = c(10,100,1000,5000,10000,50000,100000)
E <- c(0,0,0,0,0,0,0)
for(j in 1:7){
x <- runif(n[j])
y <- c(1,n[j])
for(i in 1:n[j])
if(x[i] > 0.55) {
  y[i] <- 1.0
} else {
  y[i] <- -1.0
}
sum(y)
E[j] <- sum(y)
E[j] <- E[j]/n[j]
}
par(mfrow=c(1,1))
plot(n,E)
```



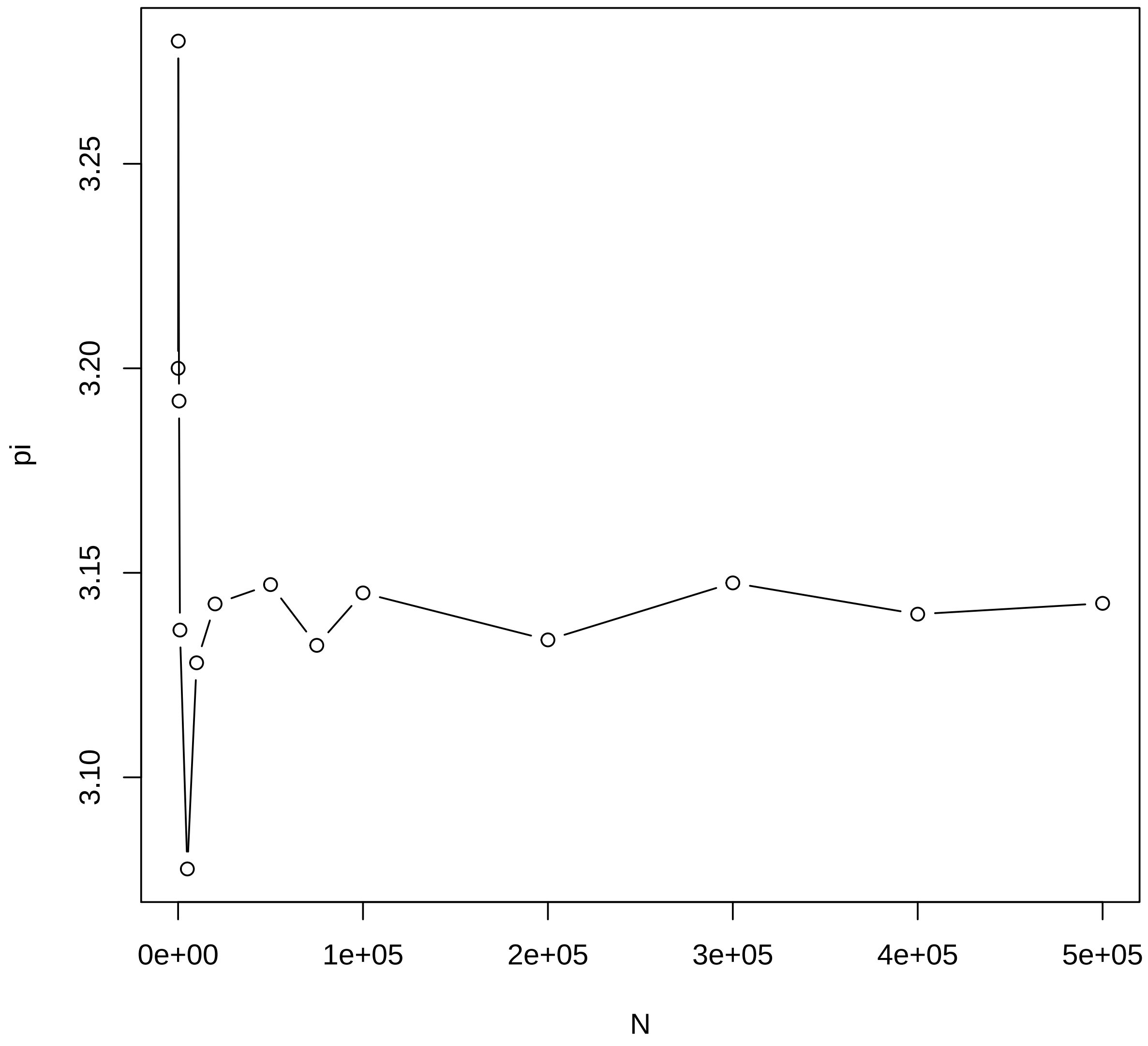
This is not the only way to simulate this problem. Here is another solution which is a bit more elegant:

```
p <- c(0.55,0.45)
R <- c(-1,1)
n = c(10,100,1000,5000,10000,50000,100000)
y = c(0,0,0,0,0,0,0)
for(j in 1:7){
  y[j] <- mean(sample(R,n[j],replace=TRUE,prob = p))
}
plot(n,y)
```



Let us do a different simulation. In this problem, we are interested in calculating the value of π. In order to do so, we throw random darts at a unit square in which a circle of unit diameter is transcribed. The circle area is given by the fraction of darts that fall within the circle multiplied by 4; since the radius of the circle is 0.5 and the area of of the square is unity, the ratio of the areas is given by $\frac{\pi}{4}$. So, by multiplying the fraction of darts that fall within the circle by 4, we obtain the value of π.

```
N <- c(10,100,500,1000,5000,10000,20000,
       50000,75000,100000,200000,300000,400000,500000)
pi = c(0,0,0,0,0,0,0,0,0,0,0,0,0,0)
for(j in 1:14){
x <- runif(N[j])
y <- runif(N[j])
for(i in 1:N[j])
if(x[i]*x[i] + y[i]*y[i] <= 1){pi[j] = pi[j]+1}
pi[j] = (4*pi[j])/N[j]
}
plot(N,pi,type="b")
```

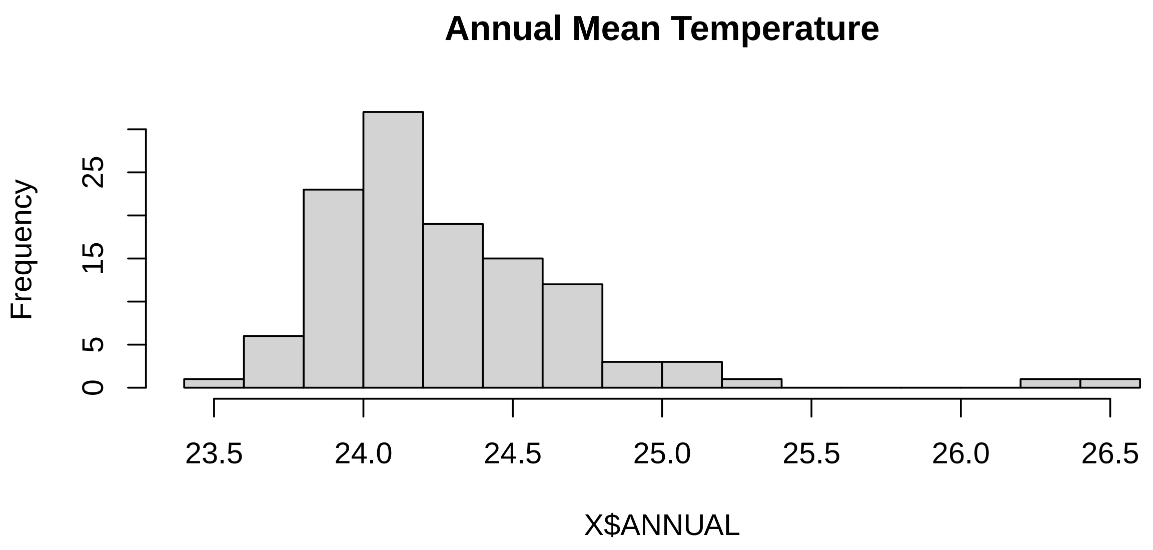


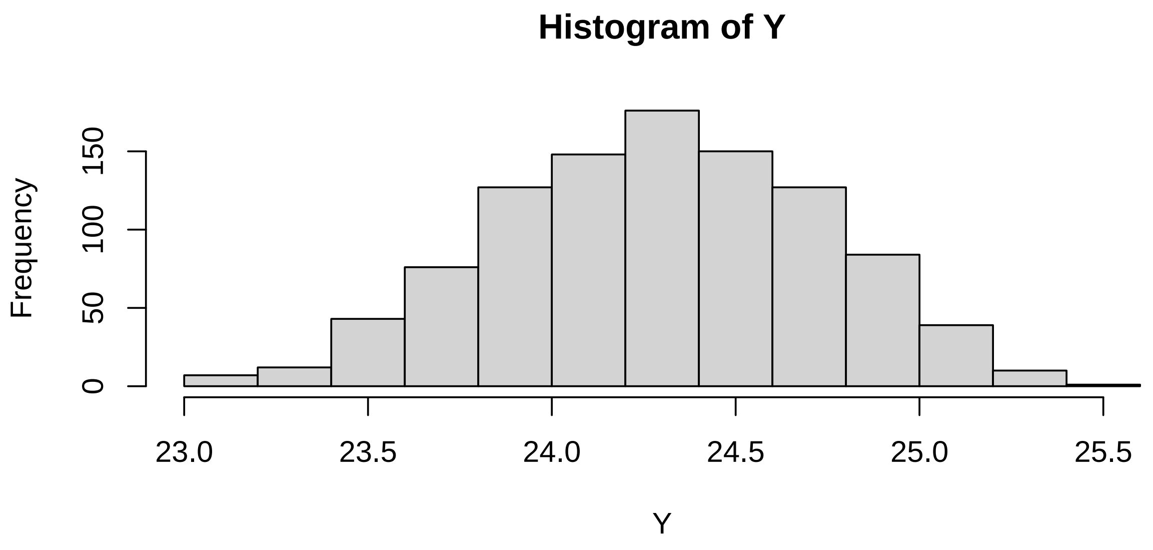
Here again, we see that as we increase N, the π value is calculated more and more accurately.

4 Normal plots

Let us consider the annual mean temperature data that we plotted earlier. Can we say that this data is normally distributed? Let us plot the data and a normal curve with the same mean and standard deviation.

```
setwd("/home/guru/Desktop/Teaching/MM217/RTutorial5")
X <- read.csv("IndiaMeanTemperature.csv")
par(mfrow=c(2,1))
hist(X$ANNUAL,breaks=20,main = "Annual Mean Temperature")
a = mean(X$ANNUAL)
b = sd(X$ANNUAL)
Y <- rnorm(1000,a,b)
hist(Y)
```

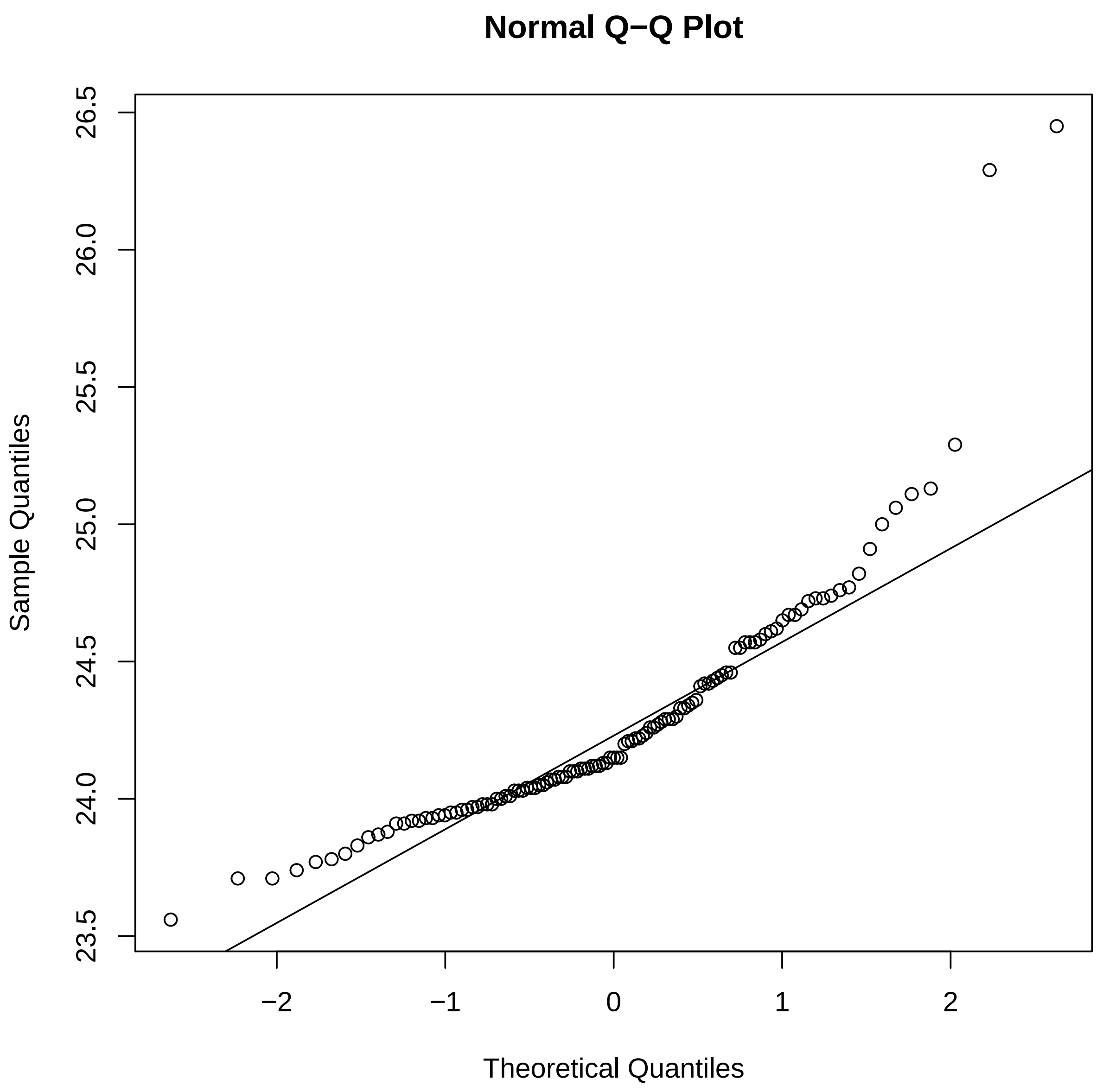




It is not very clear though the actual data seem to have some outliers as well as a bit skewed.

There is another way to check if the given data is normally distributed. We can use `qqplot` to check the normality.

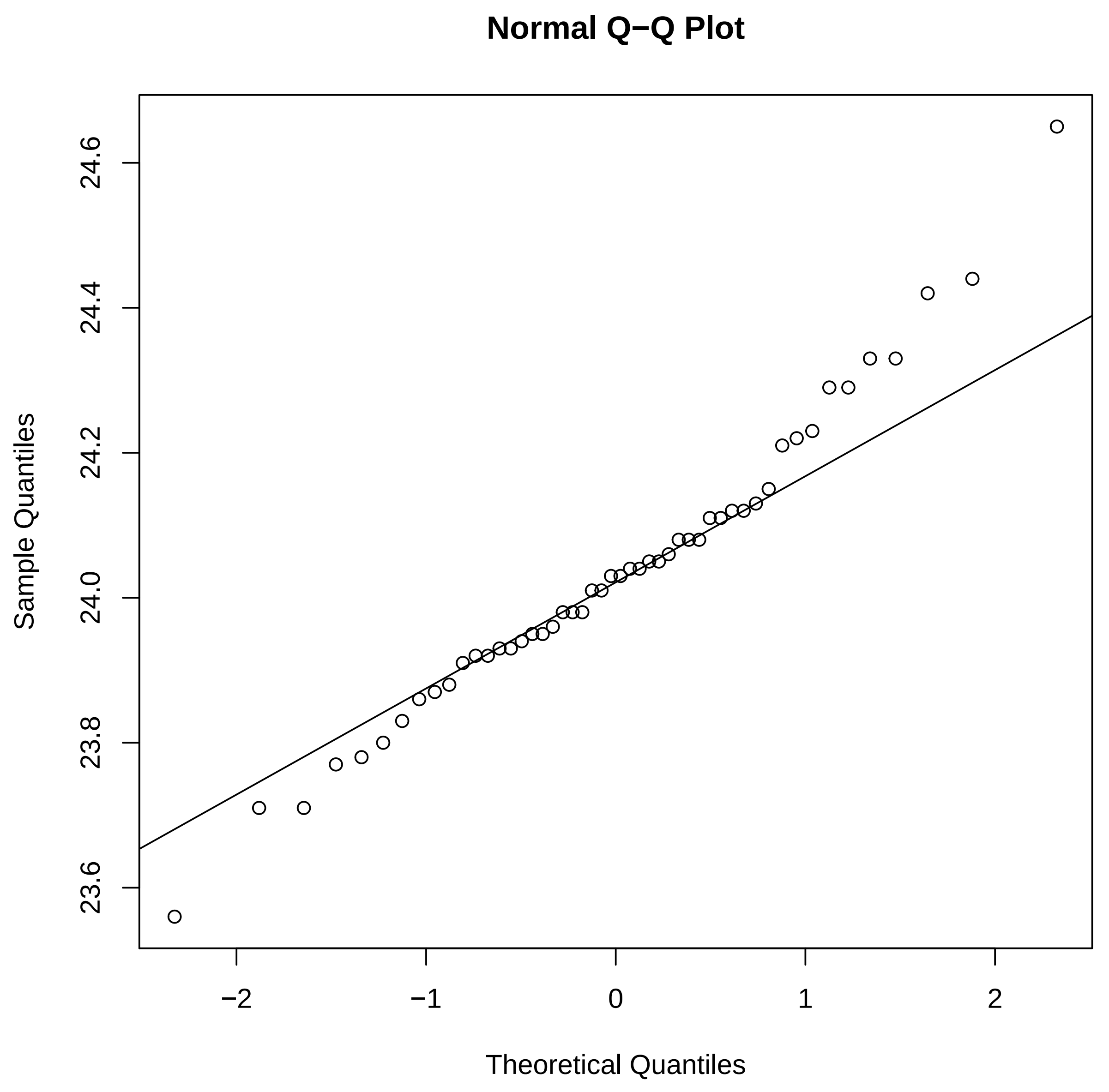
```r
qqnorm(X$ANNUAL)
qqline(X$ANNUAL)
```



From the theoretical quantile versus sample quantile plot, it is clear that the data is not normally distributed.

We can also choose the data from 1901 to 1950 and see if they are normally distributed:

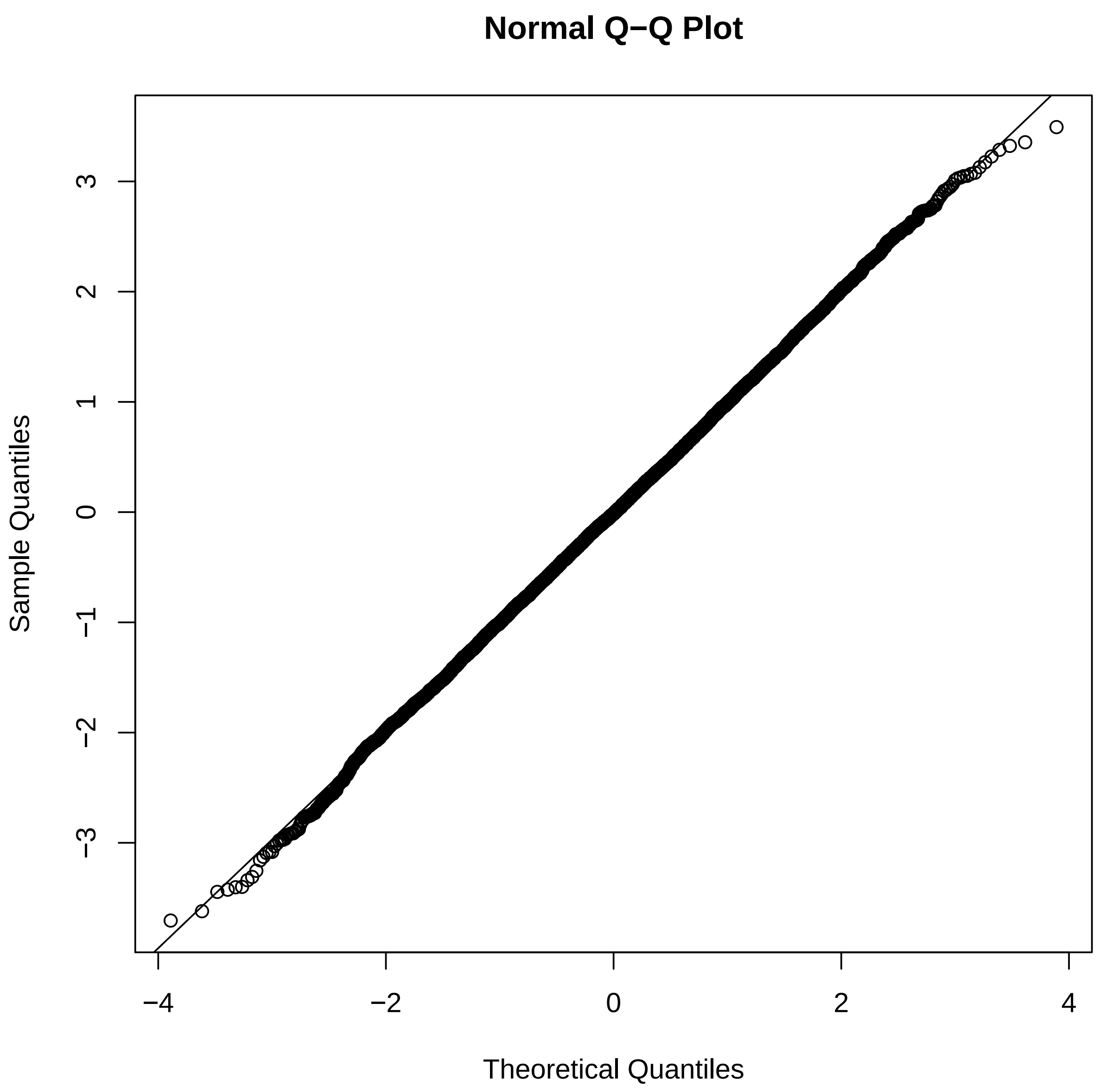
```r
qqnorm(X$ANNUAL[1:50])
qqline(X$ANNUAL[1:50])
```



Even though this is better, it still does not look like a normal distribution.

Of course, to see how the plot will look for actual normal data, we do the following exercise – simulate the normal data and plot:

```
X <- rnorm(10000)
qqnorm(X)
qqline(X)
```


Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles
−4
−2
0
2
4
−3
−2
−1
0
1
2
3