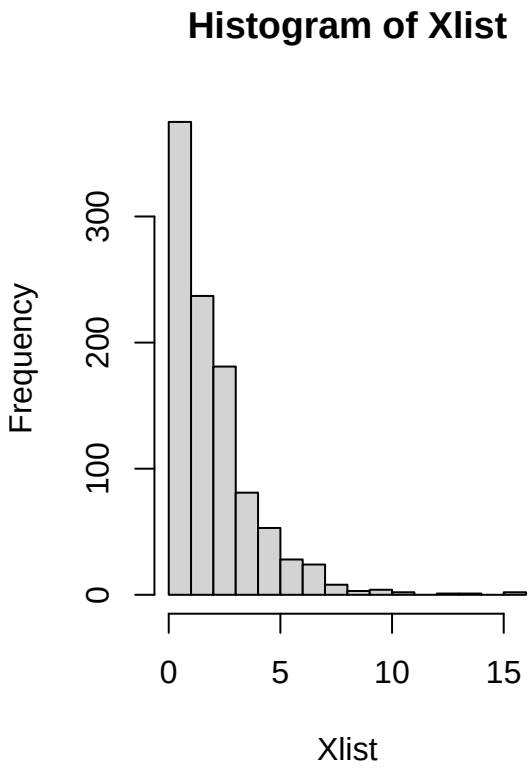
CiS_Akhilesh_HW5

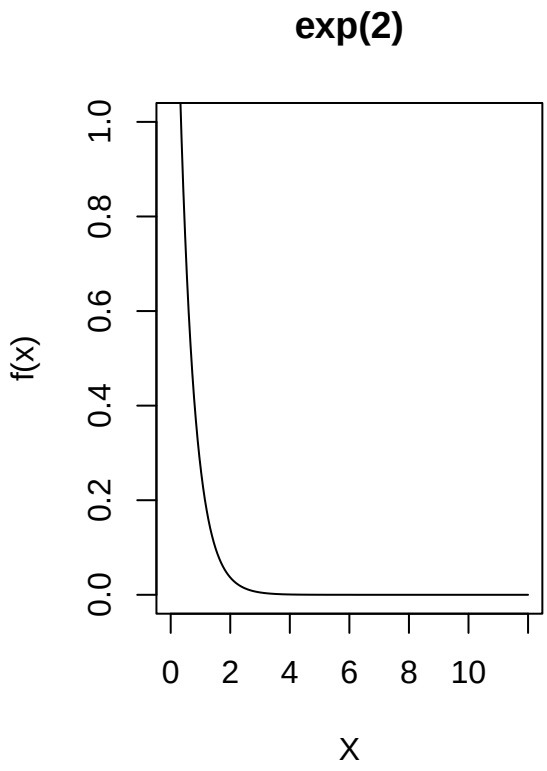
Akhilesh

2024-10-04

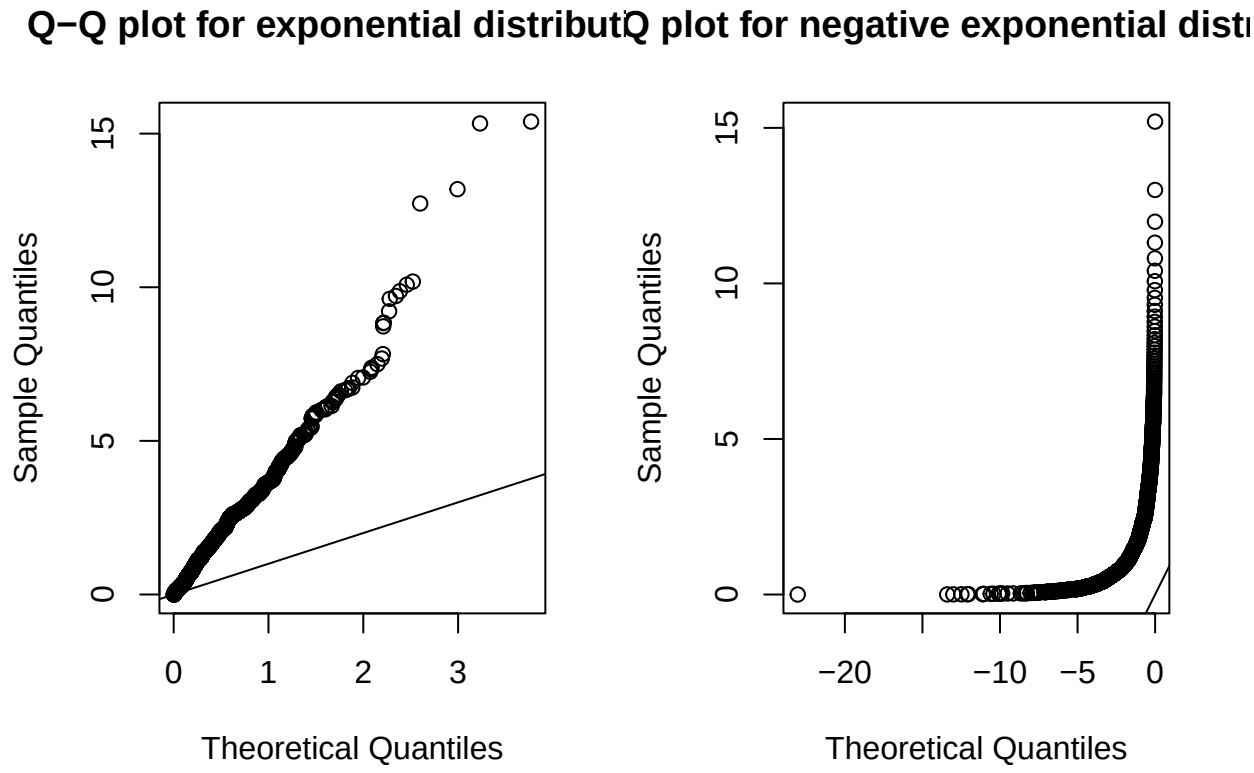
R Markdown

```
set.seed(5400)
Ulist <- runif(1000, 0, 1)
Xlist <- -2 * log(Ulist)
par(mfrow = c(1,2))
hist(Xlist)

x_vals <- seq(0.001, 12, len=200)
y_vals <- dexp(x_vals, rate=2)
plot(x_vals, y_vals, type="l", ylim=c(0, 1), xlab="X", main="exp(2)", ylab="f(x)")
```





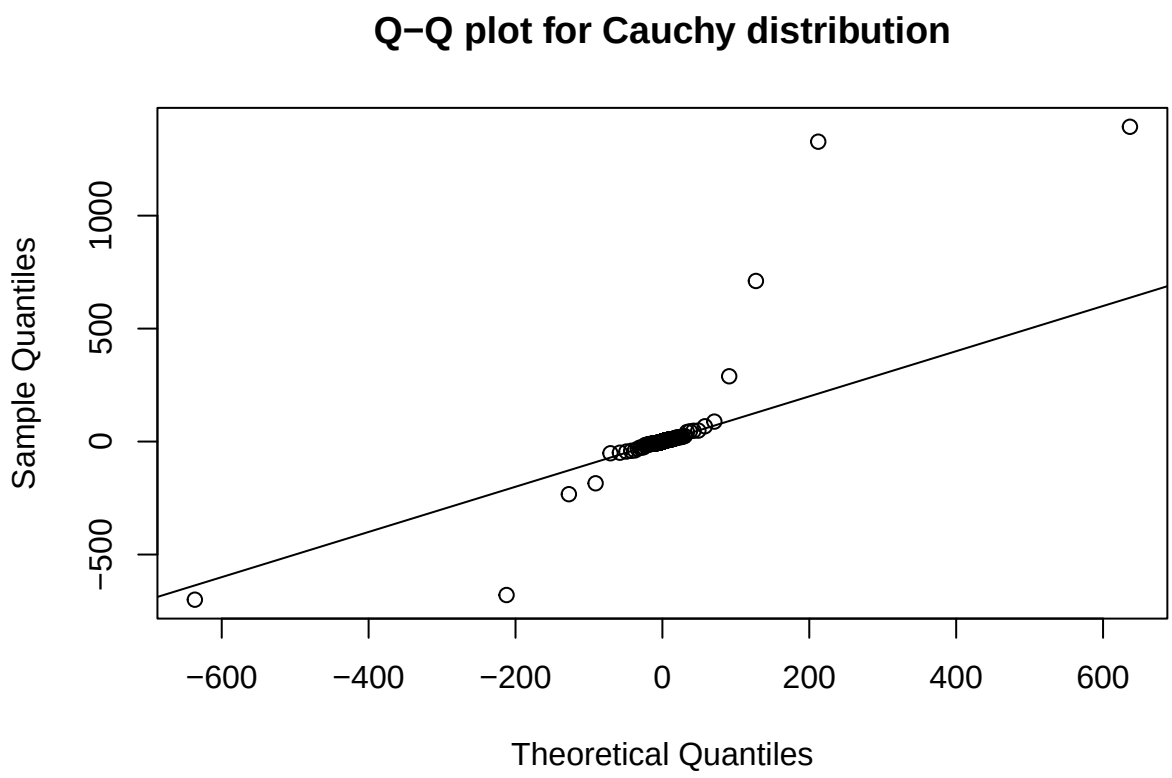
```r
qqplot <- qqplot(rexp(1000, rate=2), Xlist, xlab="Theoretical Quantiles", ylab="Sample Quantiles",main=
abline(0, 1)

negative_exp_sample <- -rexp(1000, rate=1/2)

qqplot(negative_exp_sample, qexp(ppoints(1000), rate=1/2),
       xlab="Theoretical Quantiles", ylab="Sample Quantiles",
       main="Q-Q plot for negative exponential distribution")
abline(0, 1)
```



2)

```r
set.seed(5400)
U <- runif(1000)
X_cauchy <- tan(pi * (U - 0.5))

#setEPS()
#postscript(file="qqcauchy.eps", height=4.5, width=9)

probs <- ppoints(1000)
theoretical_q <- qcauchy(probs)

# Create the Q-Q plot
plot(theoretical_q, sort(X_cauchy),
     xlab="Theoretical Quantiles",
```

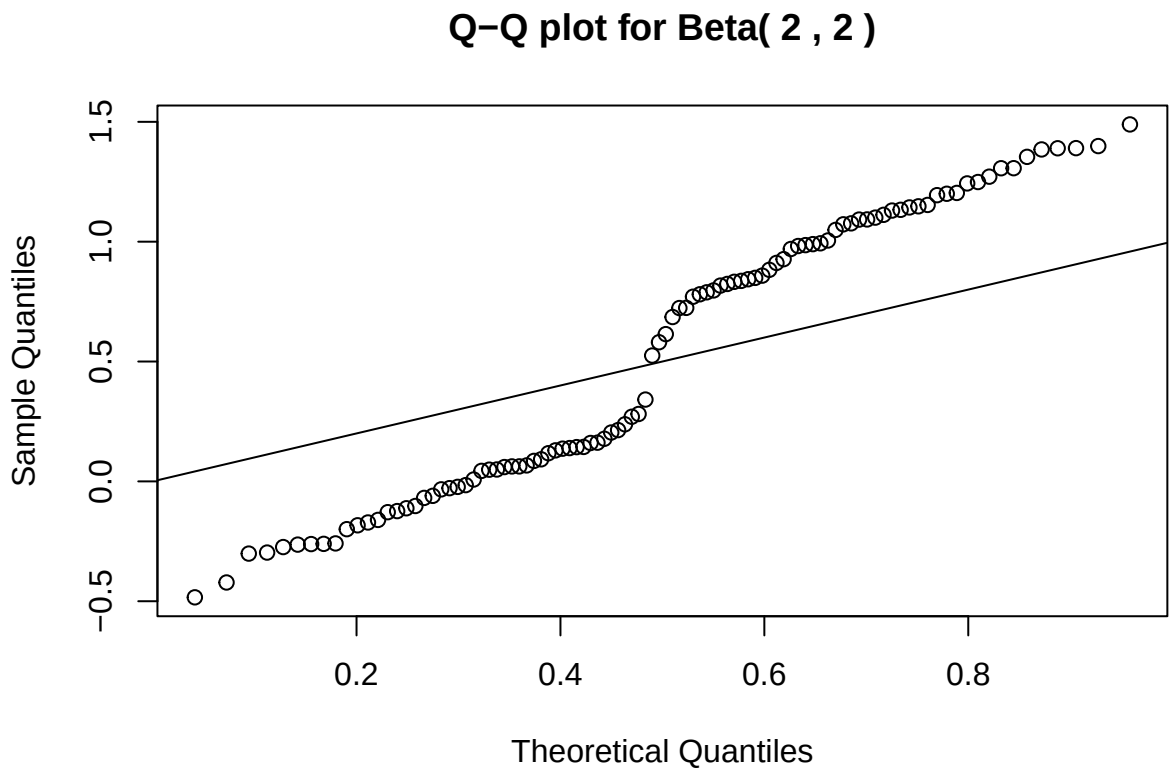
```
     ylab="Sample Quantiles",
     main="Q-Q plot for Cauchy distribution")
abline(0, 1)
```



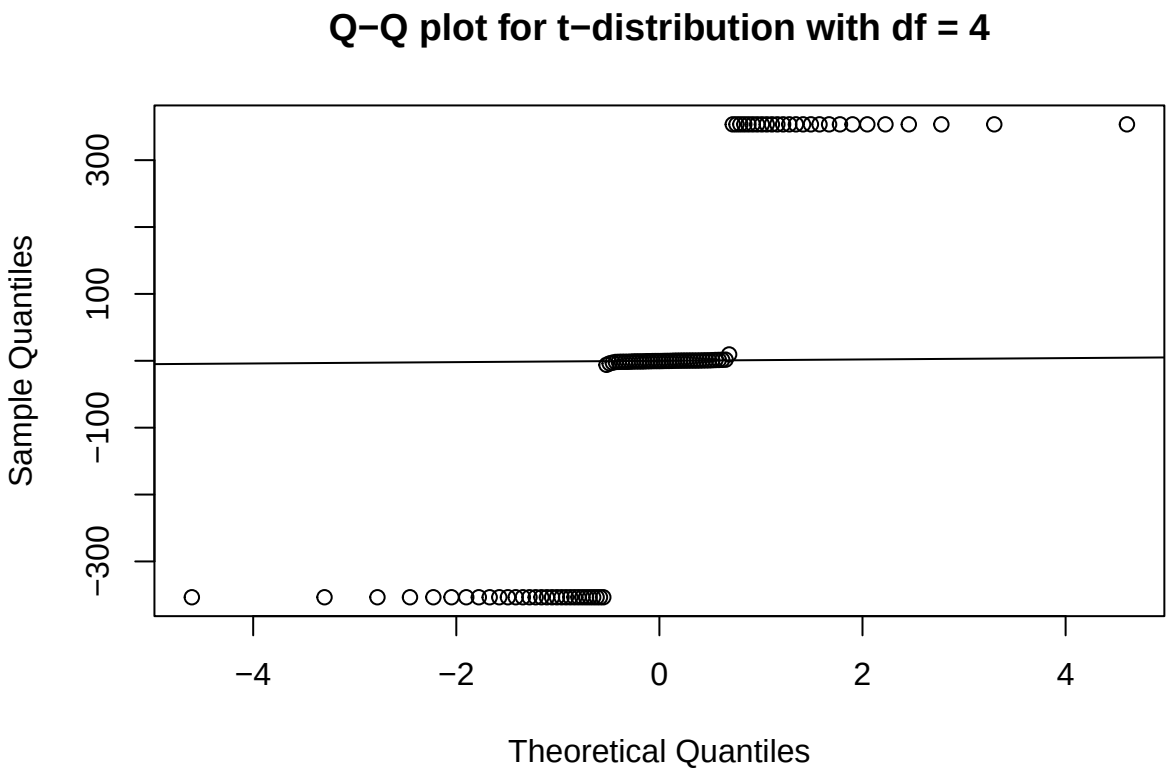

```
#dev.off()
```

```
set.seed(123)
### 1
generate_beta <- function(alpha, n) {
  U1 <- runif(n)
  U2 <- runif(n)
  U3 <- runif(n)

  indicator <- ifelse(U3 <= 1/2, 1, -1)  # Indicator function I(U3 <= 1/2)

  X <- 1/2 + indicator * sqrt(1 + (1 / (alpha^2) - 1) * cos(2 * pi * U2)^2) * (U1^(1/alpha))

  return(X)
}

### 2

alpha <- 2
beta_sample <- generate_beta(alpha, 100)
```

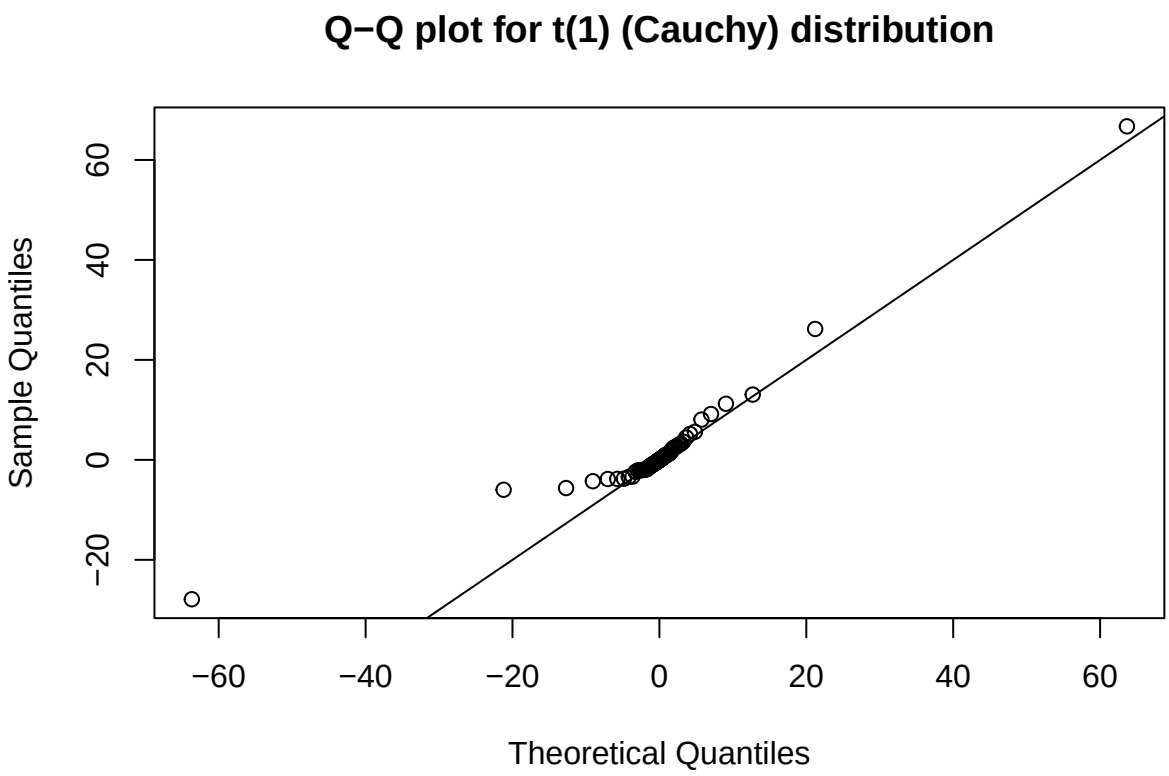
```r
qqplot(qbeta(ppoints(100), alpha, alpha), beta_sample,
       xlab = "Theoretical Quantiles",
       ylab = "Sample Quantiles",
       main = paste("Q-Q plot for Beta(", alpha, ",", alpha, ")"))
abline(0, 1)
```




```r
### 3

n <- 2
Y <- generate_beta(n, 100)

epsilon <- 1e-6
Y <- pmin(pmax(Y, epsilon), 1 - epsilon)
Z <- sqrt(n) * (Y - 1/2) / (2 * sqrt(Y * (1 - Y)))

qqplot(qt(ppoints(100), df = 2 * n), Z,
       xlab = "Theoretical Quantiles",
       ylab = "Sample Quantiles",
       main = paste("Q-Q plot for t-distribution with df =", 2 * n))
abline(0, 1)
```




```r
### 4

cauchy_sample <- rt(100, df = 1)

qqplot(qt(ppoints(100), df = 1), cauchy_sample,
       xlab = "Theoretical Quantiles",
       ylab = "Sample Quantiles",
       main = "Q-Q plot for t(1) (Cauchy) distribution")
abline(0, 1)
```



Both the plots look similar

4)

```r
truncated_normal <- function(n, mu, sigma, a = -Inf, b = Inf) {
  truncated_values <- numeric(0)

  while(length(truncated_values) < n) {
    sample_values <- rnorm(n - length(truncated_values), mean = mu, sd = sigma)

    valid_values <- sample_values[sample_values > a & sample_values < b]

    truncated_values <- c(truncated_values, valid_values)
  }
  return(truncated_values)
}

n <- 100
mu <- 0
sigma <- 1
a <- -1
b <- 1
result <- truncated_normal(n, mu, sigma, a, b)
print(result)
```

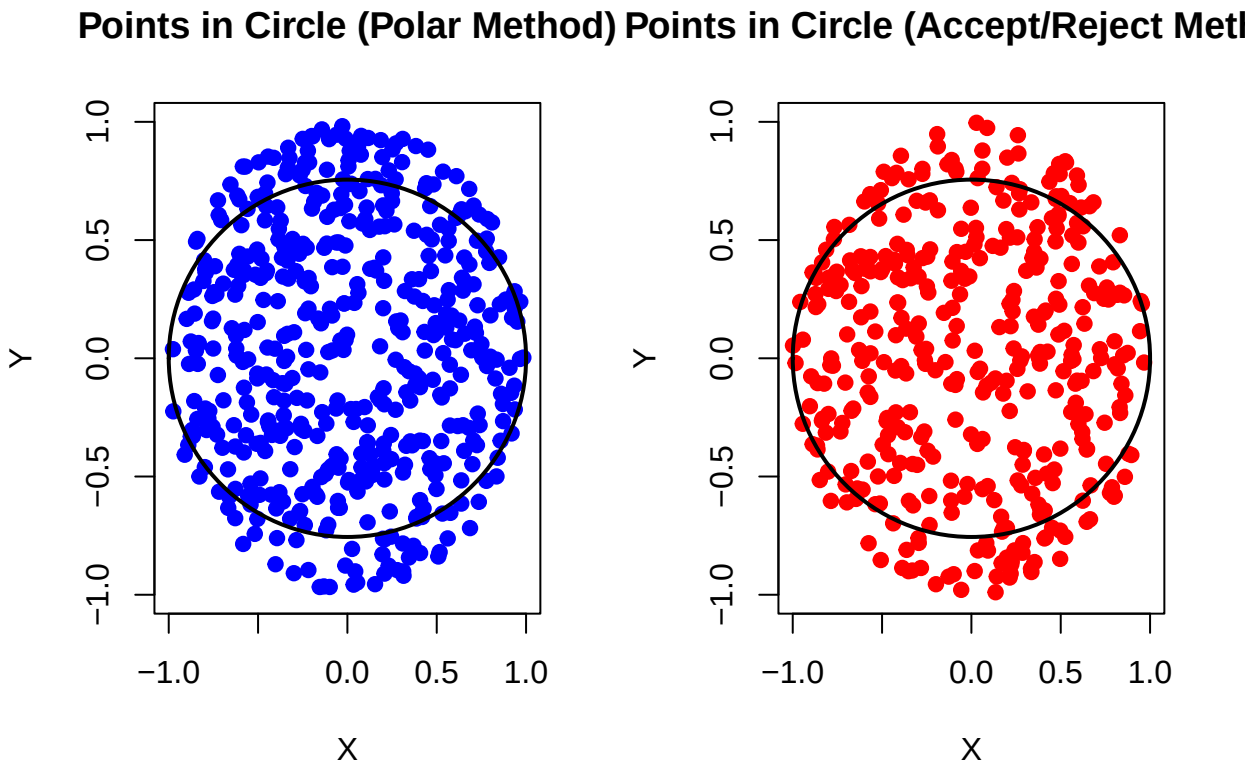
```
##   [1]  0.17273148 -0.86760381 -0.48590322  0.17214443  0.16313641  0.91415585
```

```
##   [7]  0.17064457  0.68220145 -0.55920685 -0.64443693 -0.23948728 -0.23881307
##  [13]  0.32763139  0.25105955 -0.32661712 -0.80642798 -0.81559641 -0.70147842
##  [19] -0.56229473 -0.24865494  0.12060709 -0.26048199  0.34484994 -0.80460704
##  [25]  0.80113042  0.23855062  0.13850186  0.72628030 -0.85899573 -0.29185540
##  [31]  0.57718221  0.10594824  0.01974265  0.89149575  0.84944344  0.05529995
##  [37] -0.75187304  0.07762225 -0.59010514  0.22421828  0.53522776 -0.49253947
##  [43] -0.46282728 -0.15035607 -0.58497042  0.60402496  0.02687699 -0.69167512
##  [49]  0.19570964  0.29225443  0.59844344  0.16666305  0.13719884  0.31370951
##  [55] -0.27359878  0.69380585 -0.59506799  0.12384194  0.03752209  0.06590323
##  [61] -0.51486072  0.62327642  0.17395309 -0.03478088  0.98082357  0.13859221
##  [67] -0.34251909 -0.71639871 -0.65850860 -0.41860392 -0.36620401 -0.29400111
##  [73] -0.02312263  0.36239938 -0.46671438  0.29851361 -0.95270815  0.66754058
##  [79]  0.08723854 -0.10920945 -0.67749220  0.38583100  0.24675703 -0.06130483
##  [85]  0.89525254 -0.02805877 -0.99974702 -0.15960396 -0.52769365  0.21615646
##  [91] -0.91895498 -0.90059465  0.28860879  0.92888479  0.28551442 -0.37258244
##  [97] -0.88446792 -0.72280580  0.46266496 -0.31255353
```

5)

```r
set.seed(5400)

start_time_polar <- Sys.time()
n_points <- 500
theta <- runif(n_points, 0, 2 * pi)
r <- sqrt(runif(n_points))
x_polar <- r * cos(theta)
y_polar <- r * sin(theta)
end_time_polar <- Sys.time()
time_polar <- end_time_polar - start_time_polar

accept_reject_sampling <- function(n) {
  dat <- matrix(runif(n * 2, -1, 1), n, 2)
  accept <- (dat[, 1]^2 + dat[, 2]^2) <= 1
  V <- dat[accept, ]
  return(data.frame(V))
}

start_time_reject <- Sys.time()
points_reject <- accept_reject_sampling(n_points)
end_time_reject <- Sys.time()
time_reject <- end_time_reject - start_time_reject
par(mfrow=c(1, 2))


plot(x_polar, y_polar, pch=19, col="blue",
     xlim=c(-1, 1), ylim=c(-1, 1),
     xlab='X', ylab='Y',
     main='Points in Circle (Polar Method)')
symbols(0, 0, circles=1, inches=FALSE, add=TRUE, lwd=2)

plot(points_reject[,1], points_reject[,2], pch=19, col="red",
     xlim=c(-1, 1), ylim=c(-1, 1),
     xlab='X', ylab='Y',
```

```
     main='Points in Circle (Accept/Reject Method)')
symbols(0, 0, circles=1, inches=FALSE, add=TRUE, lwd=2)
```




```
par(mfrow=c(1, 1))

cat("Time taken for polar coordinates method:", time_polar, "\n")
```

```
## Time taken for polar coordinates method: 0.004334927
```

```
cat("Time taken for accept/reject method:", time_reject, "\n")
```

```
## Time taken for accept/reject method: 0.001635075
```