R Tricks For Computational Stats

Chenguang Pan

Jan 31, 2023

2.0 Week 2's in class notes

2.1 rnorm: generate data from normal distribution

`rnorm``to genenrate dataset from normal distribution. Using the`rnorm(n, mean =, sd=)‘ function, like

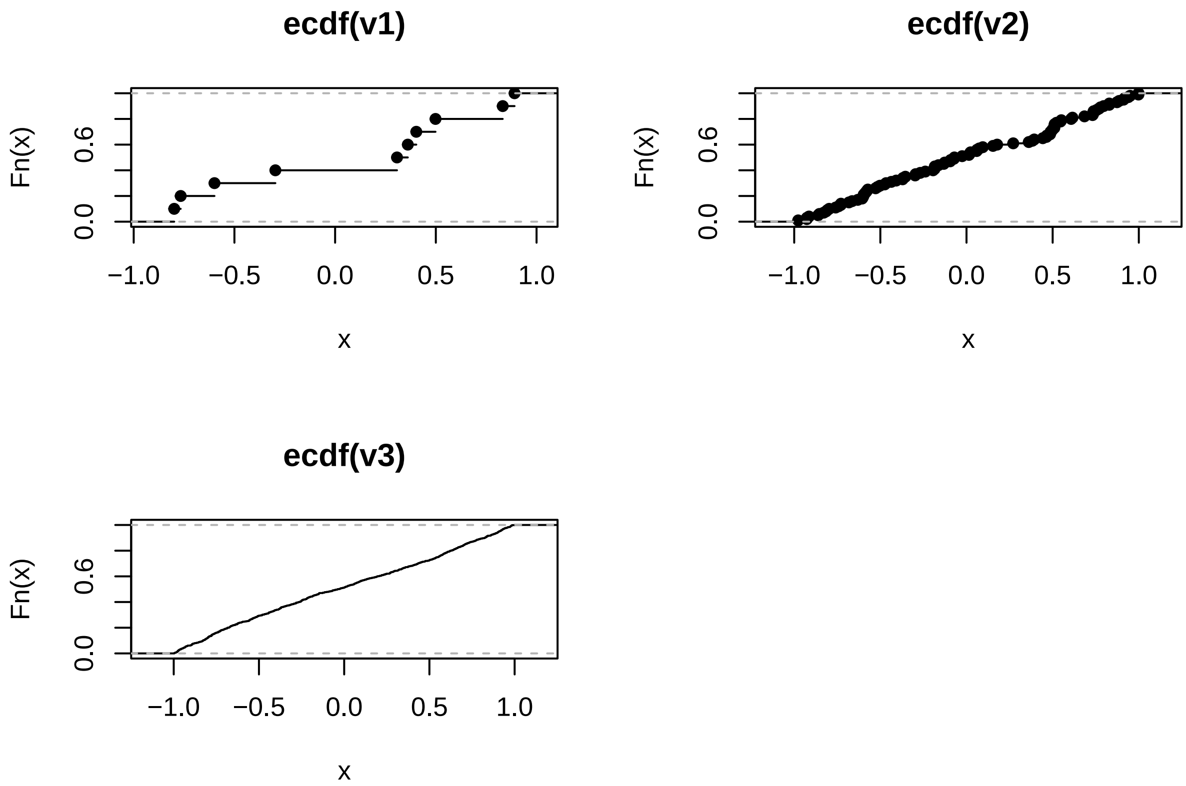
```
> sample_01 <- rnorm(n = 50, mean = 100, sd = 15)
> var(sample_01)
[1] 139.7867
> # this sample's sd and mean is close to the population's
> sd(sample_01)
[1] 11.82314
> mean(sample_01)
[1] 99.84552
```

Note it is not a vector. It is just `numeric` array.

2.2 ECDF generate data from uniform and ECDF distribution

ECDF is empirical cumulative density function. For a given sample of observations from x1,x2,x3…xn and sort it from smallest to largest. The sample size is `n`. We give each datum a value of 1/n. The ECDF is to count the number of xi above a certain level.
Note ECDF is a function, which returns the sum of i*(1/n) when x <xi. Here we use the `runif()` function to generate a dataset from uniform distribution. Note, here `runif()` is quiet like ‘rnorm()“

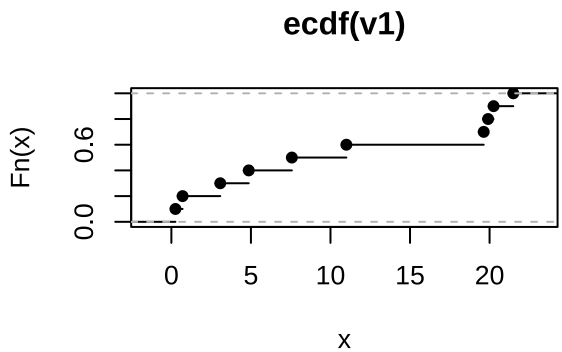
```
> set.seed(8289)
> v1 <- runif(n=10, min = -1, max = 1)
> v1
 [1] -0.2966062 -0.7667650 -0.7989303  0.3608965 -0.5989284  0.3072380
 [7]  0.8908116  0.4029667  0.4984355  0.8320636
> par(mfrow=c(2,2))
> plot(ecdf(v1))
> v2 <- runif(n = 100, min = -1, max = 1)
> plot(ecdf(v2))
> v3 <- runif(n = 1000, min = -1, max = 1)
> plot(ecdf(v3))
```

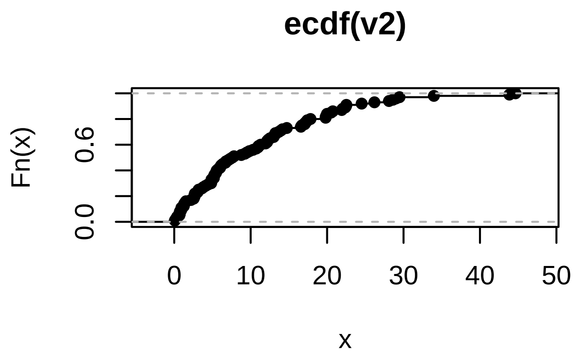


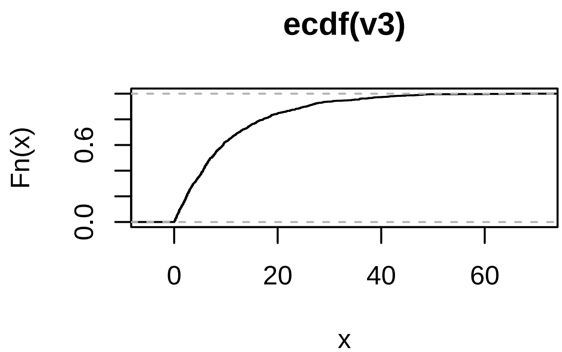
2.3 Generate data from exponential distribution

TODO: to summarize every prob. distribution

```
> set.seed(8289)
> par(mfrow=c(2,2))
> v1 <- rexp(n = 10, rate = .1)
> plot(ecdf(v1))
> v2 <- rexp(n = 100, rate = .1)
> plot(ecdf(v2))
> v3 <- rexp(n = 1000, rate = .1)
> plot(ecdf(v3))
```



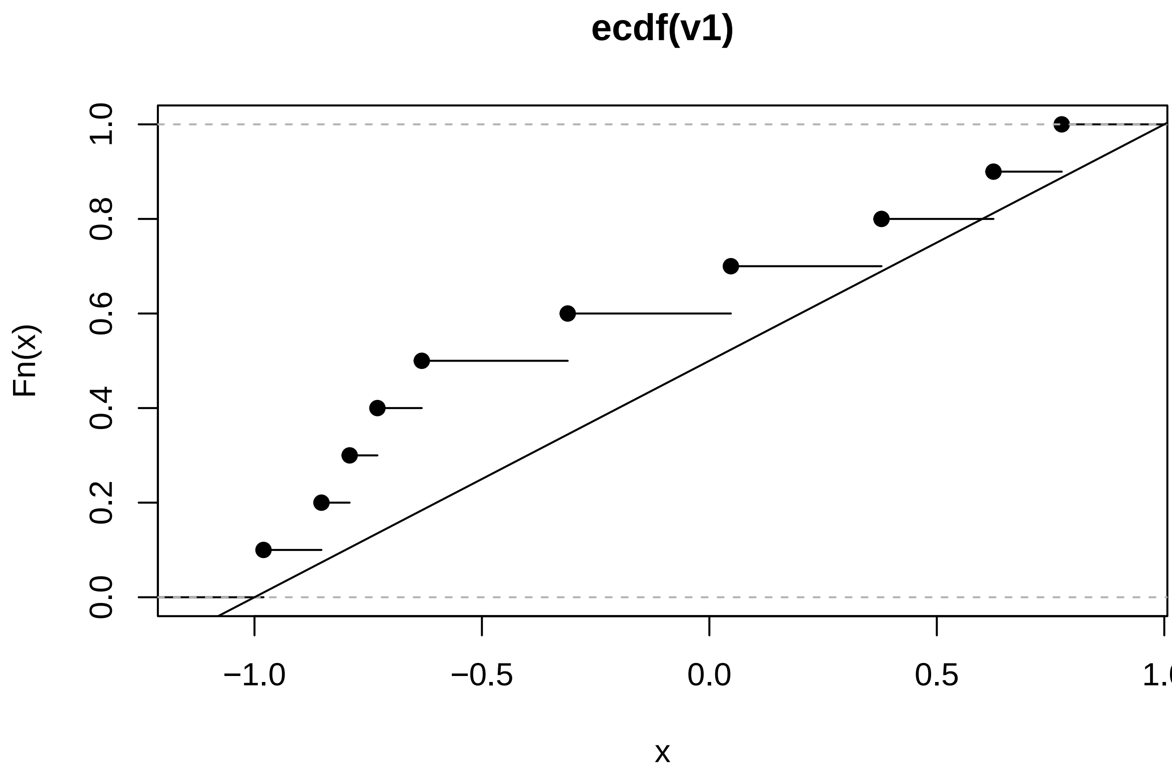




2.4 Quantitle Function

Quantile function is actually a inverse function of CDF. That is for a given probability, to find the greatest lower bound of value.
Now, let's estimate the .25 quantile for a sample from uniform -1~1. Since the properties of ecdf is to give each datum a 1/n value and then to count the culmulative value of 1/n, and since this is a uniform distribution which means each two data point shares the same interval. Therefore the heigh change in step is 1/n

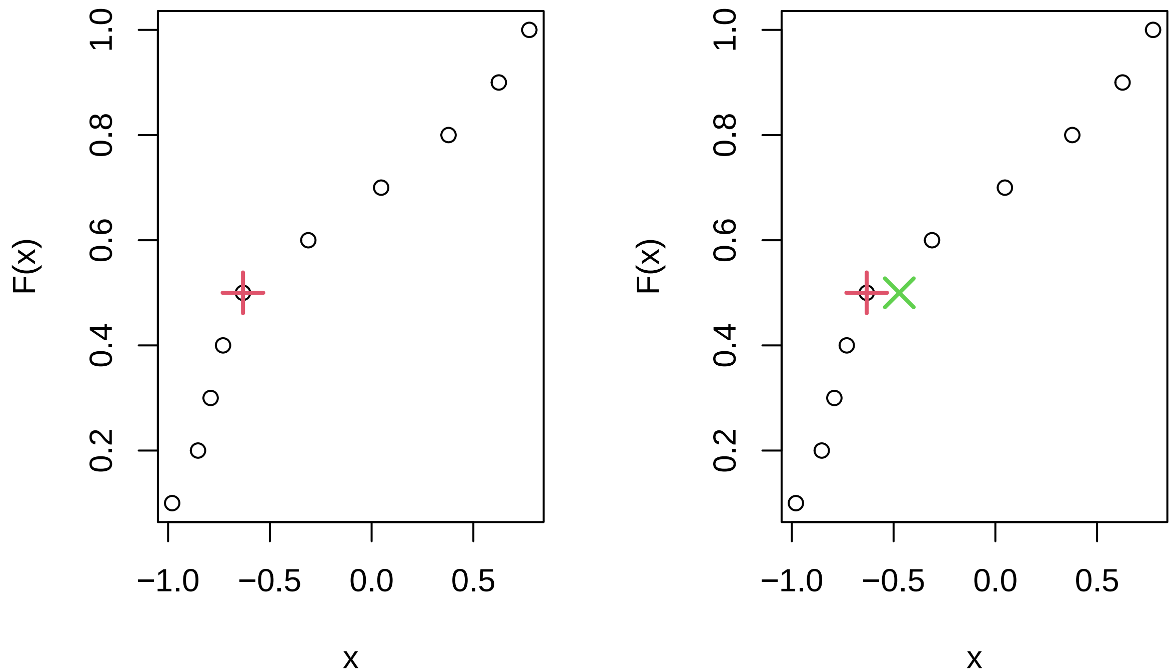
```
> set.seed(9832)
> v1 <- runif(n = 10, min= -1, max= 1)
> # note ecdf is a function based on given observations
> ecdf1<- ecdf(v1)
> plot(ecdf1)
> abline(a = 1/2, b = 1/2)
> # note the following is not a quantile function!!
> ecdf1(0.5)
[1] 0.8
```



Next, let's estimate .5 quantile via ecdf. The quantile is greatest lower bound of set of x values for which the ecdf is greater or equal to .5.

```
> # using cbind() funciton to combine tow columns of data
> # the seq is to create a set of data with certain interval
> ecdf_tab <- round(cbind(sort(v1), seq(.1,1,.1)),3)
> colnames(ecdf_tab) <- c("x", "F(x)")
> ecdf_tab
           x F(x)
 [1,] -0.980  0.1
 [2,] -0.853  0.2
 [3,] -0.791  0.3
 [4,] -0.730  0.4
 [5,] -0.632  0.5
 [6,] -0.311  0.6
 [7,]  0.047  0.7
 [8,]  0.378  0.8
 [9,]  0.625  0.9
[10,]  0.775  1.0
> par(mfrow=c(1,2))
> plot(ecdf_tab)
> points(ecdf_tab[5,1], ecdf_tab[5,2], pch = 3, col = 2, cex = 2, lwd = 2)
> # next, lets draw a quantile function using R's defualt
> quantile(x = v1, prob = 0.5)
       50%
-0.4718142
> plot(ecdf_tab)
```

```
> points(ecdf_tab[5,1], ecdf_tab[5,2], pch = 3, col = 2, cex = 2, lwd = 2)
> points(quantile(x = v1, probs = .5), .5, pch = 4, col = 3, cex = 2, lwd = 2)
```



By default, the `quantile()` function in R will return a different result. Here there are n = 10 observations.R uses type = 7 which specifies that the kth order statistic is assigned cumulative probability of (k-1)/(n-1). Furthermore, R help notes that "All sample quantiles are defined as weighted averages of consecutive order statistics."

```
> type7_tab <- round(cbind(sort(v1), seq(0, 1, 1/9)), 3)
> colnames(type7_tab) <- c("x", "F(x)")
> type7_tab
           x  F(x)
 [1,] -0.980 0.000
 [2,] -0.853 0.111
 [3,] -0.791 0.222
 [4,] -0.730 0.333
 [5,] -0.632 0.444
 [6,] -0.311 0.556
 [7,]  0.047 0.667
 [8,]  0.378 0.778
 [9,]  0.625 0.889
[10,]  0.775 1.000
> # Weighted avg of consecutive order statistics.
> mean(type7_tab[5:6,1])
[1] -0.4715
> # the value is same to the quantile function
```

2.5 Multivariate Normal Distribution

```
> # Generate multivariate normal
> mvn_gen <- function(n, mu, sigma, factorization = "Cholesky") {
+   # Generate a sample of size n from multivariate normal with
+   # mean vector mu and covariance matrix sigma.
+   # Argument factorization can be either "Cholesky" or "Spectral".
+   d <- length(mu)
+   Z <- matrix(rnorm(n*d), nrow = n, ncol = d)
+   if(factorization == "Cholesky") { Q <- chol(sigma) } else {
+     if(factorization == "Spectral") {
+       ev <- eigen(sigma)
+       lambda <- ev$values
+       P <- ev$vectors
+       Q <- P %*% diag(sqrt(lambda)) %*% t(P) } else {
+         stop("Arg factorization must be 'Cholesky' or 'Spectral'.")
+       } }
+   mu <- matrix(mu, nrow = d, ncol = 1)
+   J = matrix(1, nrow = n, ncol = 1)
+   X <- Z %*% Q + J %*% t(mu)
+   return(data.frame(X))
+ }
>
> sig <- matrix(c(1, .5, .8, .5, 1, 0, .8, 0, 1), 3, 3, byrow = TRUE)
>
> set.seed(2381)
> mv1 <- mvn_gen(n = 1000, mu = c(0, 1, 4), sigma = sig, factorization = "Cholesky")
> plot(mv1)
> mv2 <- mvn_gen(n = 1000, mu = c(0, 1, 4), sigma = sig, factorization = "Spectral")
> plot(mv2)
```