Inverse Transform method

Otherwise known as inverse CDF method. The continuous case:

Suppose that we have a continuous random variable X having Cumulative Density Function (CDF) F_X. Then, the random variable

$$U = F_X(X)$$

has a Uniform distribution $U \sim U_{[0,1]}$. So to generate a random variate x from the distribution of X, we can use the following transformation

$$F_X^{-1}(U) = x$$

where $F_X^{-1}(.)$ is the inverse CDF or quantile function.

Weibull example

Example We want to generate a sample of $10,000$ random realizations from a Weibull distribution $W(5,2)$ using the Inverse CDF method.

If $X \sim \mathcal{W}ei(\alpha, \beta)$ having PDF and CDF defined respectively as

PDF $$f_X(x) = \frac{\alpha}{\beta}\left(\frac{x}{\beta}\right)^{\alpha-1} exp\left(-\frac{x}{\beta}\right)^{\alpha} \mathbb{1}_{\mathbb{R}^+}(x)$$

CDF $$F_X(x) = 1 - exp\left(-\frac{x}{\beta}\right)^{\alpha}$$

Weibull example solved

Then by the Inverse CDF method, we can generate realizations of X by equating $U = F_X(x)$ and solving for x. We have

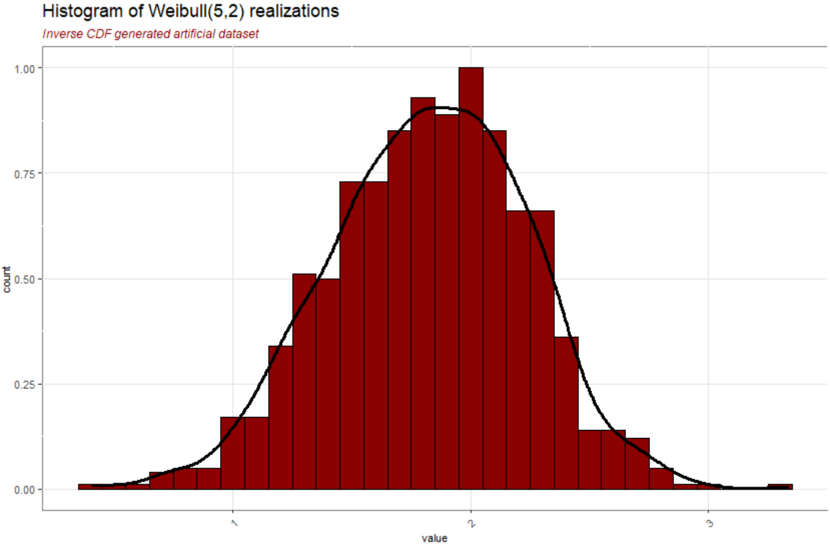
$$\begin{aligned} U &= 1 - exp\Big(- \frac{x}{2}\Big)^5 \\ 1 - U &= exp\Big(- \frac{x}{2}\Big)^5 \\ -ln(1-U) &= \Big(\frac{x}{2}\Big)^5 \\ \big(- ln(1-U)\big)^{1/5} &= \frac{x}{2} \\ 2\big(- ln(1-U)\big)^{1/5} &= x \end{aligned}$$

Code for Weibull realizations using the inverse CDF in R

```r
# 1. Define Inverse CDF function
Inverse_CDF_Weibull <-function(n, alpha, beta) {
   u <- runif(n)                                         # generate uniform numbers
   data <- beta*((-log(1-u))^(1/alpha))                  # formula derived
   return(data.frame(data))                              # return the data
}

# 2. Generate realizations of the desired distribution
set.seed(2023)
dataset <- Inverse_CDF_Weibull(n = 1000, alpha = 5, beta = 2)
head(dataset)
#        data
# 1 1.8226044
# 2 1.6719235
# 3 1.4157089
# 4 1.7441408
# 5 0.9975117
# 6 1.3275159
```

Histogram in R



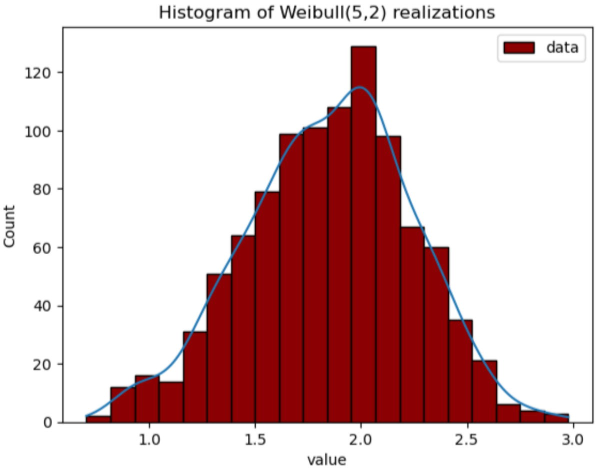
Code for Weibull realizations using the inverse CDF in Python

```
# 1. Define Inverse CDF function
def Inverse_CDF_Weibull(n, alpha, beta) :
        u = np.random.uniform(low=0.0, high=1.0, size=n)          # generate uniform numbers
        data = beta*((-np.log(1-u))**(1/alpha))                    # forumla derived

        return pd.DataFrame(data = data, columns = ['data'])       # return a data frame instead of an arry

# 2. Generate realizations of the desired distribution
np.random.seed(2023)
dataset = Inverse_CDF_Weibull(n = 1000, alpha = 5, beta = 2)
dataset.head()
#      data
#0  1.655497
#1  2.343973
#2  1.952544
#3  1.340683
#4  1.372833
#...  ...
```

Histogram in Python



References

Rizzo, M.L. (2019). Statistical Computing with R, Second Edition (2nd ed.). Chapman and Hall/CRC. https://doi.org/10.1201/9780429192760

The R Project for Statistical Computing: https://www.r-project.org/

Python: https://www.python.org/