Quality Test of random number generators

David Zambrano Lizarazo

March 11, 2016

Abstract

Your abstract.

1 Introduction

1.1 ¿Porqué estudiar generadores de números aleatorios?

Actualmente existen multiples generadores de números aleatorios en diferentes entornos y compiladores lo cual supondría para un usuario de la Simulación que no es necesario su estudio. Sin embargo, estudios sobre algunos generadores comerciales sugieren que debemos actuar con cuidado con el uso de ellos. Incluso, el uso progresivo de modelos de simulación cada vez más detallados exige generadores de números aleatorios de mayor calidad.

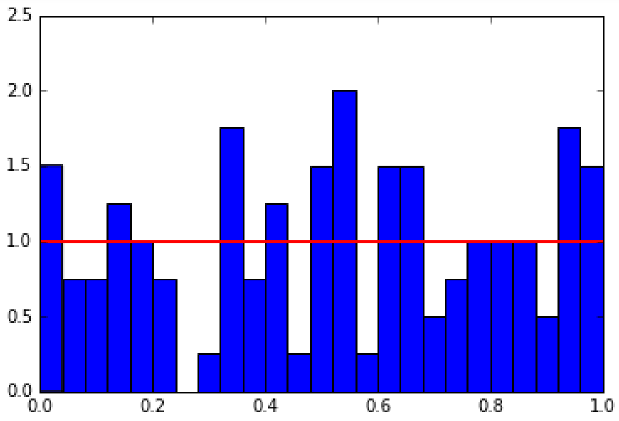
2 Se evaluara la calidad de los siguientes generadores de números aleatorios

- Randu
$$x_i + 1 = 65539 x_i mod 2^{31}$$
- Sinclair ZX81
$$x_i + 1 = 75 x_i mod 2^{16} + 1$$
- Numerical reciepes
$$x_i + 1 = 1664525 x_i + 1013904223 mod 2^{32}$$
- Borland C/C++
$$x_i + 1 = 22695477 x_i + 1 mod 2^{32}$$

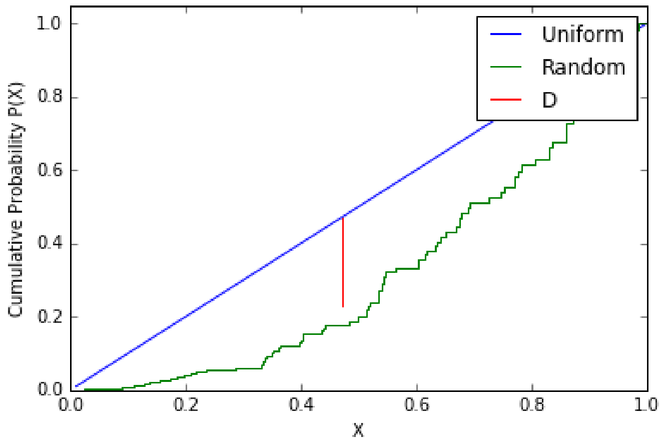
2.1 RANDU

$$x_i + 1 = 65539 x_i mod 2^{31}$$

2.1.1 Test Kolmogorov-Smirnov



('De = ', 0.24122449960362033)



```
('D = ', 0.24122449960362005)
('p-value = ', 1.3178895129861701e-05)
```

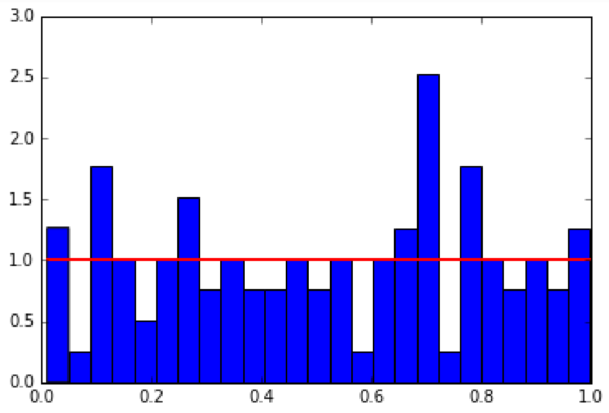
Comments can be added to the margins of the document using the

.

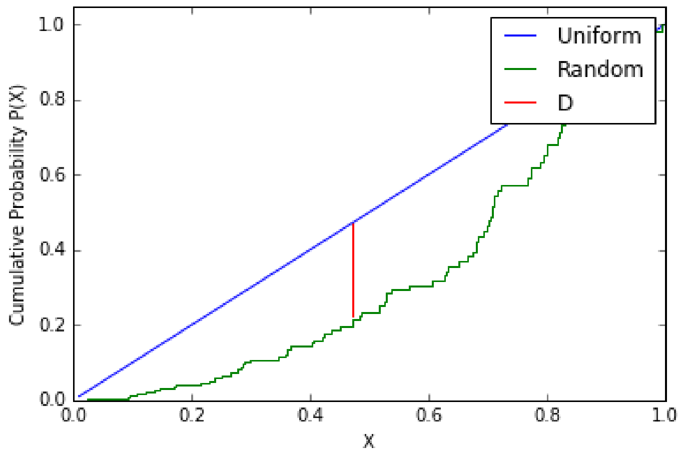
2.2 Sinclair ZX81

$$x_i + 1 = 75 x_i mod 2^{16} + 1$$

2.2.1 Test Kolmogorov-Smirnov



```
('De = ', 0.24707387596121735)
```



```
('D = ', 0.24707387596121708)
('p-value = ', 7.3061291787634985e-06)
```

LaTeX is great at typesetting mathematics. Let $X_1, X_2, \ldots, X_n$ be a sequence of independent and identically distributed random variables with $\mathrm{E}[X_i] = \mu$ and $\mathrm{Var}[X_i] = \sigma^2 < \infty$, and let

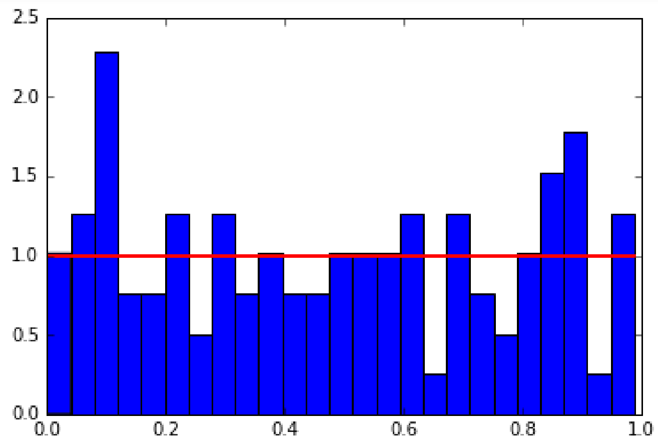
$$S_n = \frac{X_1 + X_2 + \cdots + X_n}{n} = \frac{1}{n} \sum_i^n X_i$$

denote their mean. Then as n approaches infinity, the random variables $\sqrt{n}(S_n - \mu)$ converge in distribution to a normal $\mathcal{N}(0, \sigma^2)$.

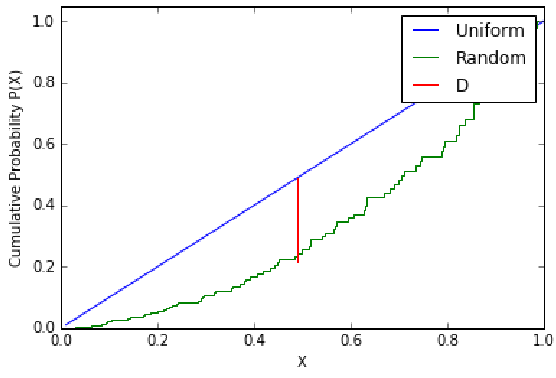
2.3 Numerical reciepes

$$x_i + 1 = 1664525x_i + 1013904223 mod 2^{32}$$

2.3.1 Test Kolmogorov-Smirnov



```
('De = ', 0.27530983606204029)
```



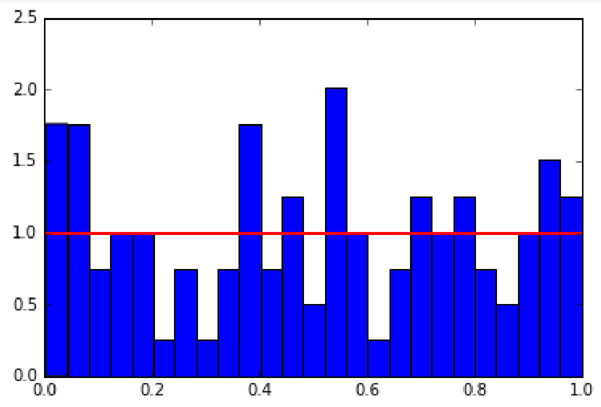
```
('D = ', 0.27530983606203996)
('p-value = ', 3.4328976838970959e-07)
```

Use section and subsection commands to organize your document. LaTeX handles all the formatting and numbering automatically. Use ref and label commands for cross-references.

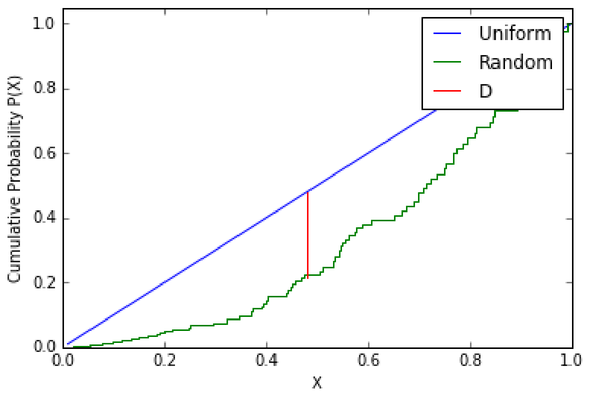
2.4 Borland C/C++

$$x_i + 1 = 22695477x_i + 1 mod 2^{32}$$

2.4.1 Test Kolmogorov-Smirnov



```
('De = ', 0.26390738787818702)
```



```
('D = ', 0.26390738787818679)
('p-value = ', 1.2310481680710694e-06)
```

Use section and subsection comment

2.5 Anexo Codigo

Codigo fuente para los Generadores aleatorios ...

```
def Randu(Maxgenerator=1):
    nums= []
    Xn = 9
    a = 65539
    m = np.power(2,31)
    for iterator in range(Maxgenerator):
        Xn_1 = a*Xn  % m
        out = float(Xn_1)/float(m)
        nums.append(out)
        Xn = Xn_1
    return nums

def Sinclair_ZX81(Maxgenerator=1):
    nums= []
    Xn = 9
    a = 75
    m =  np.power(2,16)+1

    for iterator in range(Maxgenerator):
        Xn_1 = a*Xn  % m
        out = float(Xn_1)/float(m)
        nums.append(out)
        Xn = Xn_1
    return nums

def Numerical_reciepes(Maxgenerator=1):
    nums= []
    Xn = 9
    a = 1664525
    m =  np.power(2,32)
    c= 1013904223
    for iterator in range(Maxgenerator):
        Xn_1 = (a*Xn + c ) % m
        out = float(Xn_1)/float(m)
        nums.append(out)
        Xn = Xn_1
    return nums

# Borland C/C++
def Borland_C(Maxgenerator=1):
    nums= []
    Xn = 9
    a = 22695477
    m =  np.power(2,32)
    c= 1
    for iterator in range(Maxgenerator):
        Xn_1 = (a*Xn + c )  % m
        out = float(Xn_1)/float(m)
        nums.append(out)
        Xn = Xn_1
    return nums
```

1. Like this,
2. and like this.

. . . or bullet points . . .

- Like this,
- and like this.

. . . or with words and descriptions . . .

Word Definition

Concept Explanation

Idea Text