

Taller de Simulación computacional

Punto 4: Utilizando el generador de números aleatorios del lenguaje de programación, generar N números y realizar la prueba de uniformidad e independencia indicados en los puntos anteriores, ¿Determinar si es un buen generador y por qué?

R//

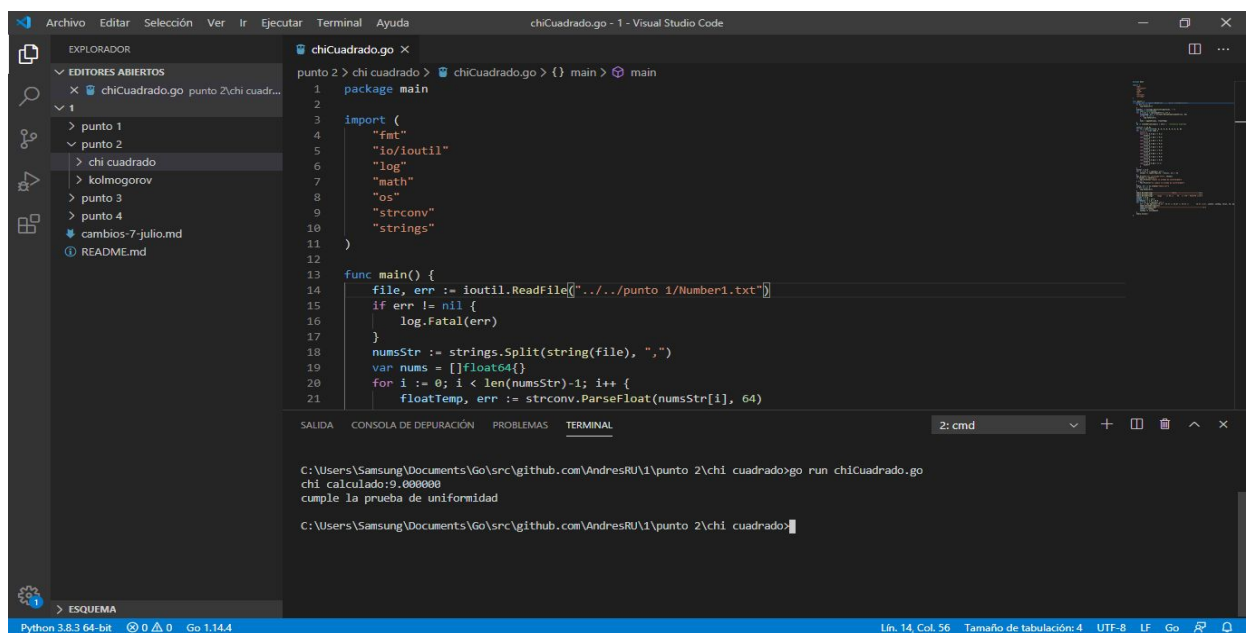
Para este punto utilizaremos el generador de Python para generar una lista de 500 números pseudoaleatorios con una semilla de 0.



```
1 import numpy as np
2
3 np.random.seed(0)
4 random_num = np.random.rand(500)
5 num1 = open("Number1.txt", "w")
6 for i in range(random_num):
7     num1.write(str(i))
8     num1.write("\n")
9 num1.close()
10
11
```

Primero realizaremos dos pruebas de uniformidad, Chi cuadrado y Kolmogorov.

CHI CUADRADO:



```
1 package main
2
3 import (
4     "fmt"
5     "io/ioutil"
6     "log"
7     "math"
8     "os"
9     "strconv"
10    "strings"
11)
12
13 func main() {
14     file, err := ioutil.ReadFile("../punto 1/Number1.txt")
15     if err != nil {
16         log.Fatal(err)
17     }
18     numsStr := strings.Split(string(file), "\n")
19     var nums = []float64{}
20     for i := 0; i < len(numsStr)-1; i++ {
21         floatTemp, err := strconv.ParseFloat(numsStr[i], 64)
22     }
23 }
```

C:\Users\Samsung\Documents\Go\src\github.com\AndresRU\1\punto 2\chi cuadrado>go run chiCuadrado.go
chi calculado:9.000000
cumple la prueba de uniformidad

C:\Users\Samsung\Documents\Go\src\github.com\AndresRU\1\punto 2\chi cuadrado>

KOLMOGOROV:

The screenshot shows a Visual Studio Code editor with a Go file named `kolmogorov.go`. The code implements a Kolmogorov-Smirnov test by reading data from `Number1.txt`, parsing it into a float array, and calculating the maximum difference between the empirical and theoretical cumulative distribution functions. The terminal output shows the test result: `0.023999999999999966` and the message `cumple la prueba de uniformidad`.

```
6  "log"
7  "math"
8  "os"
9  "strconv"
10 "strings"
11 }
12
13 func main() {
14     file, err := ioutil.ReadFile("../punto 1/Number1.txt")
15     if err != nil {
16         log.Fatal(err)
17     }
18     numsStr := strings.Split(string(file), ",")
19     var nums = []float64{}
20     for i := 0; i < len(numsStr)-1; i++ {
21         floatTemp, err := strconv.ParseFloat(numsStr[i], 64)
22         if err != nil {
23             log.Fatal(err)
24         }
25         nums = append(nums, floatTemp)
26     }
27 }
```

```
C:\Users\Samsung\Documents\Go\src\github.com\AndresRU\1\punto 2\kolmogorov>go run kolmogorov.go
0.023999999999999966
cumple la prueba de uniformidad

C:\Users\Samsung\Documents\Go\src\github.com\AndresRU\1\punto 2\kolmogorov>
```

Ahora realizaremos las pruebas de Series, Corridas y Poker con 5 decimales para verificar independencia.

SERIES:

The screenshot shows a Visual Studio Code editor with a Python file named `series.py`. The code implements a Series test by reading data from `Number1.txt`, calculating the chi-squared statistic, and comparing it to a critical value. The terminal output shows the chi-squared value `4517.040000000001` and the message `no cumple la prueba de independencia`.

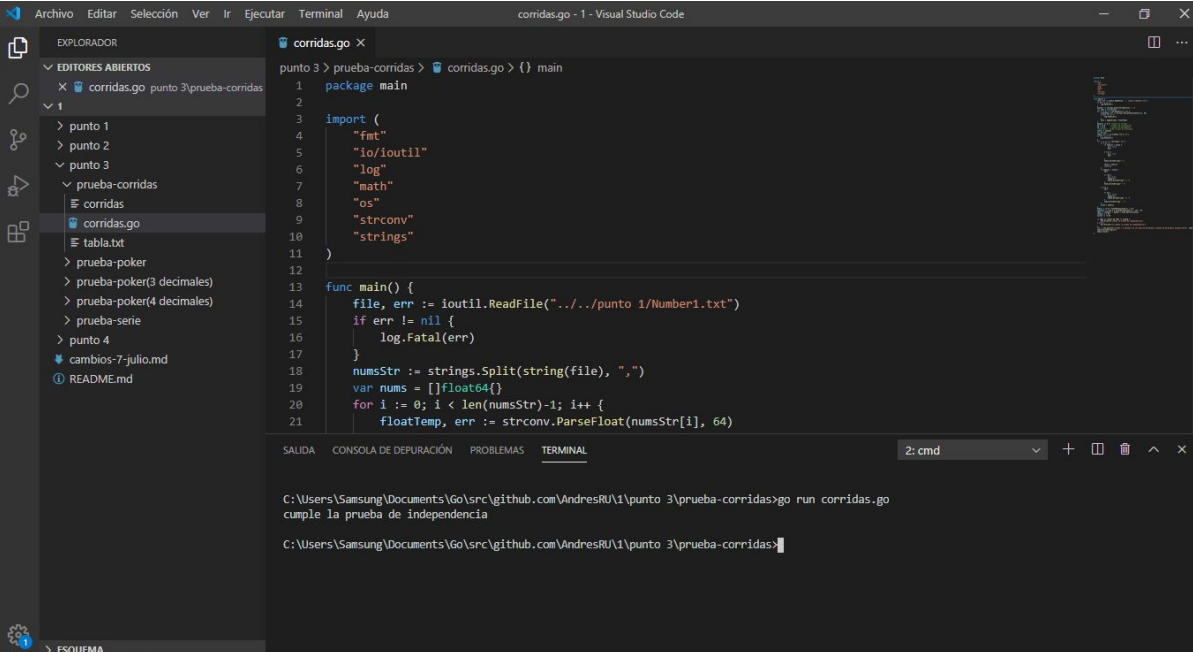
```
37 if pairs[i][0] >= int_inf and pairs[i][0] < int_sup:
38     nums_temp.append(pairs[i])
39
40 while row sup <= 1:
```

```
frequncia obtenida:
0.0-0.1 | 0.1-0.2 | 0.2-0.3 | 0.3-0.4 | 0.4-0.5 | 0.5-0.6 | 0.6-0.7 | 0.7-0.8 | 0.8-0.9 | 0.9-1.0
3.0 | 2.0 | 2.0 | 2.0 | 1.0 | 1.0 | 6.0 | 2.0 | 3.0 | 1.0
2.0 | 3.0 | 3.0 | 3.0 | 1.0 | 3.0 | 5.0 | 2.0 | 2.0 | 4.0
2.0 | 7.0 | 2.0 | 1.0 | 2.0 | 5.0 | 2.0 | 3.0 | 1.0 | 3.0
0.0 | 1.0 | 1.0 | 3.0 | 4.0 | 3.0 | 3.0 | 2.0 | 2.0 | 1.0
2.0 | 1.0 | 4.0 | 3.0 | 0.0 | 2.0 | 3.0 | 2.0 | 4.0 | 2.0
5.0 | 2.0 | 3.0 | 0.0 | 4.0 | 6.0 | 1.0 | 2.0 | 4.0 | 3.0
2.0 | 4.0 | 7.0 | 2.0 | 6.0 | 4.0 | 4.0 | 2.0 | 1.0 | 1.0
2.0 | 4.0 | 3.0 | 3.0 | 3.0 | 2.0 | 1.0 | 0.0 | 2.0 | 2.0
4.0 | 2.0 | 1.0 | 2.0 | 0.0 | 2.0 | 3.0 | 2.0 | 2.0 | 4.0
1.0 | 2.0 | 2.0 | 2.0 | 2.0 | 1.0 | 5.0 | 5.0 | 1.0 | 0.0
(fe-to)/fe

0.0-0.1 | 0.1-0.2 | 0.2-0.3 | 0.3-0.4 | 0.4-0.5 | 0.5-0.6 | 0.6-0.7 | 0.7-0.8 | 0.8-0.9 | 0.9-1.0
44.18 | 46.08 | 46.08 | 46.08 | 48.02 | 48.02 | 38.72 | 46.08 | 44.18 | 48.02
46.08 | 44.18 | 44.18 | 44.18 | 48.02 | 44.18 | 40.5 | 46.08 | 46.08 | 42.32
46.08 | 36.98 | 46.08 | 48.02 | 46.08 | 40.5 | 46.08 | 44.18 | 48.02 | 44.18
50.0 | 48.02 | 48.02 | 44.18 | 42.32 | 44.18 | 44.18 | 46.08 | 46.08 | 48.02
46.08 | 48.02 | 42.32 | 44.18 | 50.0 | 46.08 | 44.18 | 46.08 | 42.32 | 46.08
40.5 | 46.08 | 44.18 | 50.0 | 42.32 | 38.72 | 48.02 | 46.08 | 42.32 | 44.18
46.08 | 42.32 | 36.98 | 46.08 | 38.72 | 42.32 | 42.32 | 46.08 | 48.02 | 48.02
46.08 | 42.32 | 44.18 | 44.18 | 44.18 | 46.08 | 50.0 | 46.08 | 46.08 | 46.08
42.32 | 46.08 | 48.02 | 46.08 | 50.0 | 46.08 | 44.18 | 46.08 | 46.08 | 42.32
48.02 | 46.08 | 46.08 | 46.08 | 46.08 | 48.02 | 40.5 | 40.5 | 48.02 | 50.0

chi calculado: 4517.040000000001
no cumple la prueba de independencia
diego@parrot:~/Desktop/universidad/7-semestre/simulacion computacional/taller/taller1/punto 3/prueba-serie]
```

CORRIDAS:

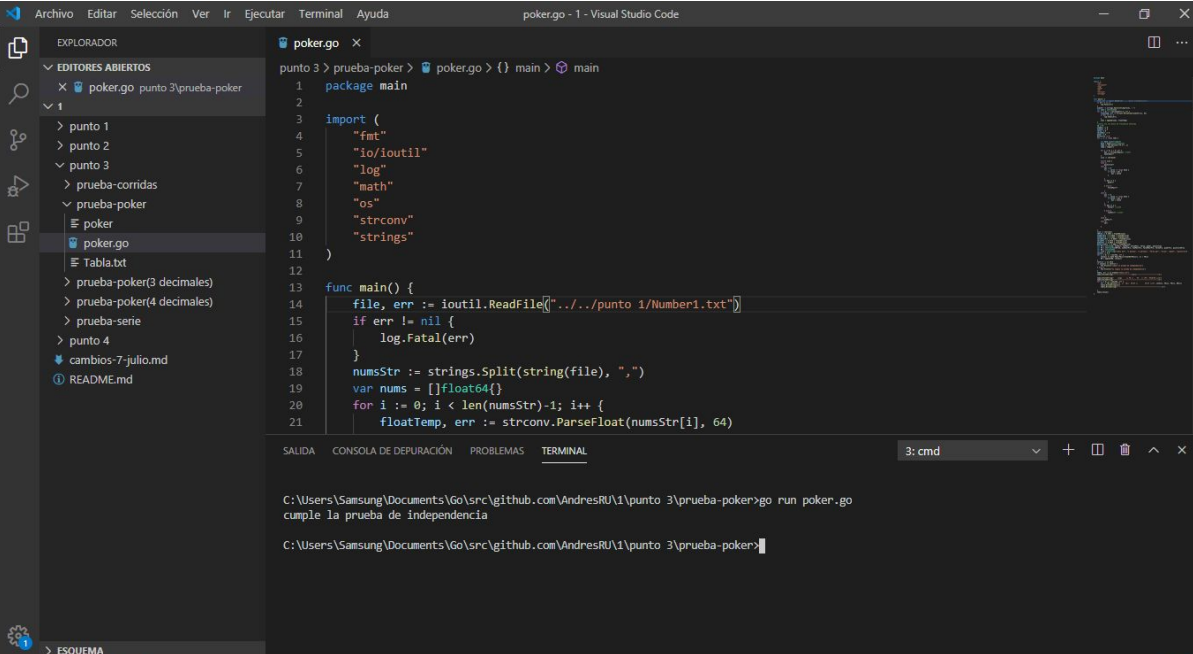


The screenshot shows the Visual Studio Code interface with the 'corridas.go' file open. The file contains the following Go code:

```
1 package main
2
3 import (
4     "fmt"
5     "io/ioutil"
6     "log"
7     "math"
8     "os"
9     "strconv"
10    "strings"
11 )
12
13 func main() {
14     file, err := ioutil.ReadFile("../punto 1/Number1.txt")
15     if err != nil {
16         log.Fatal(err)
17     }
18     numsStr := strings.Split(string(file), ",")
19     var nums = []float64{}
20     for i := 0; i < len(numsStr)-1; i++ {
21         floatTemp, err := strconv.ParseFloat(numsStr[i], 64)
```

The terminal output shows the command 'run corridas.go' being executed, resulting in the message 'cumple la prueba de independencia'.

POKER:



The screenshot shows the Visual Studio Code interface with the 'poker.go' file open. The file contains the following Go code:

```
1 package main
2
3 import (
4     "fmt"
5     "io/ioutil"
6     "log"
7     "math"
8     "os"
9     "strconv"
10    "strings"
11 )
12
13 func main() {
14     file, err := ioutil.ReadFile("../punto 1/Number1.txt")
15     if err != nil {
16         log.Fatal(err)
17     }
18     numsStr := strings.Split(string(file), ",")
19     var nums = []float64{}
20     for i := 0; i < len(numsStr)-1; i++ {
21         floatTemp, err := strconv.ParseFloat(numsStr[i], 64)
```

The terminal output shows the command 'run poker.go' being executed, resulting in the message 'cumple la prueba de independencia'.

Este generador logró pasar todas las pruebas de uniformidad e independencia que se le aplicaron excepto la de series, lo que nos dice que no es un generador excelente, pero tampoco es un pésimo, estas pruebas nos muestran que los números no siguen ningún patrón aparente.