Algorisme Random més fiable

Carles Guivernau
Marcel Pujol
Pau Vila

Resum

- **Objectiu:** Observar quin és l'algorisme de nombres "random" més fiable, independentment del tipus de llenguatge de programació emprat, es a dir, l'estudi es centra en el codi de l'algorisme en sí.

- **Mètodes:** Hem utilitzat tres codis generadors de nombres "random" implementats en llenguatges de programació diferents (C++, PHP i JAVA) i els hem executat per obtenir cinquanta nombres aleatoris per cada un d'ells.

- **Resultats:** Realitzant la corresponent anàlisis de variància, hem pogut comprovar que hi ha evidències suficients com per creure que tots tres algorismes estudiats són igual de fiables, entenent com a fiable aquell algorisme "random" que té tantes possibilitats de que surti un número com un altre.

- **Discussió:** Tot i els resultats presentats, ens vàrem veure limitats en quant a la recollida de dades, ja que possiblement ens hàgim quedat curts a la hora d'executar els algoritmes cinquanta vegades per a generar els corresponents cinquanta nombres "random" (entre 0 i 100).

Materials i mètodes

- Tres programes amb **l'algorisme Random, en tres llenguatges** de programació diferents:
    1. rand de PHP
    2. Math.random() de JAVA
    3. rand de C++

- **Variable resposta :** Qualitat.

- **Variable explicativa:** Algorisme utilitzat.

- **Qualificació execució** = $\sum_{i=1}^{100}(\mid (Ni - 10000) \mid)$

Exemple d'execució

- 1: 10,200
- 2: 9,800
- 3: 11,000
- 4: 9,000
- 5: 10,000
- …
- Qualificació = abs(10,200 - 0,000) + abs(9,800 – 10,000) + abs(11,000 -10,000) + abs(9,000 -10,000) + abs(10,000 -10,000)

  1 2 3 4 5

  = 200 + 200 +1000 + 1000 + 0

  = 2400

Annex, Codis:

- **CODI PHP:**

```
<?php
$r = array();
$suma = 0;
for($j = 0; $j < 1000000; ++$j) $r[mt_rand(1,100)]++;
for($j = 1; $j <= 100; ++$j) $suma += abs($r[$j] - 10000);
echo $suma;
?>
```

Anàlisi estadístic

1.Prova de significació (unilateral):

$H0$: μC++ =μPHP = μJAVA $H1$= ∃ i,j tq μi ≠ μJ

2. Comprovació de les premisses:

a) Normalitat dels valors de la variable resposta de cada algorisme.
b) Homoscedasticitat (variància constant entre els tres algorismes)
c) Independència de les dades obtingudes.

3. Estadístic i distribució de referència

$\wedge F = \frac{QME}{QMR}$

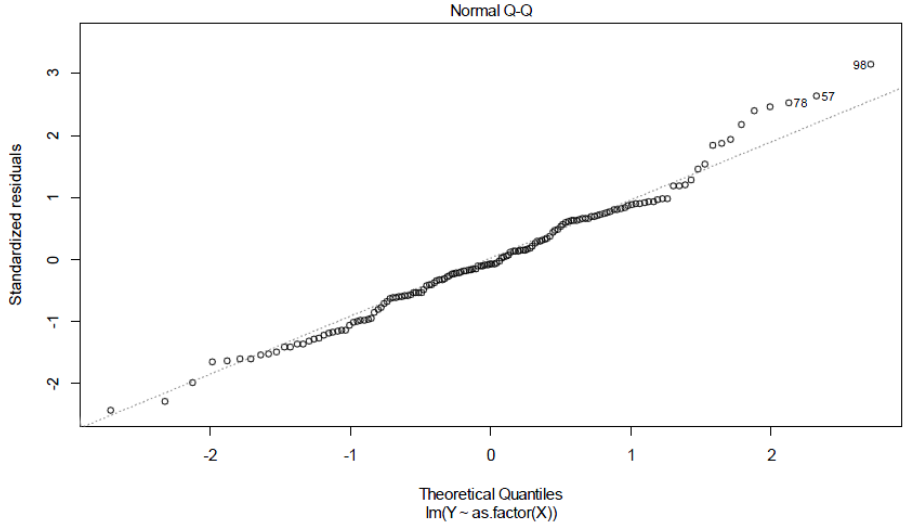
- Sota hipòtesis nul·la, segueix una distribució Fm,n graus de llibertat del numerador (m=k-1) y denominador (n=N-k).
- Amb un 5% de risc alfa (α).

Resultats: Descriptiva

- La taula 1.1 mostra la mitjana i la desviació típica de la qualitat (la nostra variable resposta) de l'algorisme per a cada grup:

Algorisme	Nº d'execucions	Mitjana (qualitat)	Desviació tipus (qualitat)
rand de C++	50	7919	523.4921
Math.random() de JAVA	50	7663	509.7388
rand de PHP	50	7818	607.5983

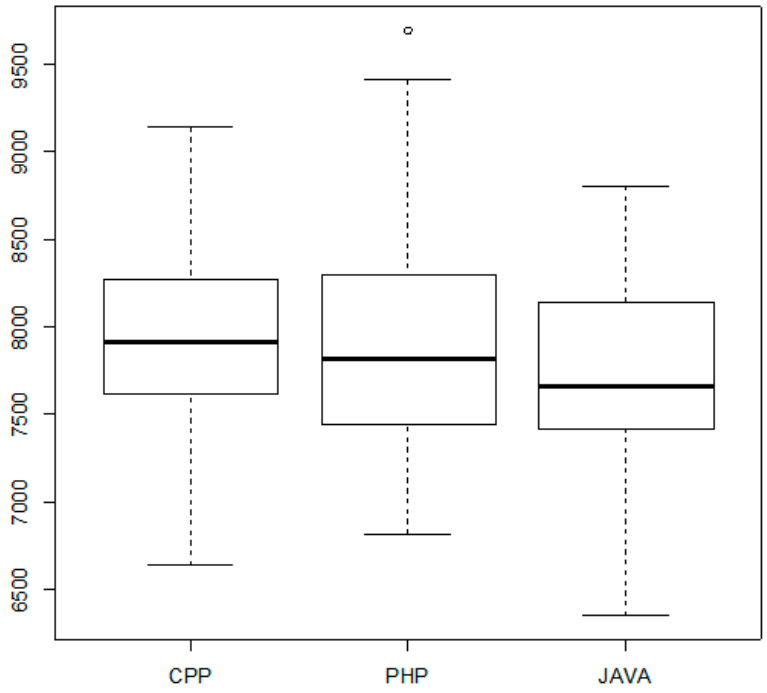
Taula 1.1: Anàlisis descriptiva de la variable qualitat del algorisme



Gràfic 1: Anàlisis de la normalitat de la variable.

Resultats: Descriptiva

- El Boxplot de les tres mostres ens donen una imatge gràfica dels resultats obtinguts. Podem veure clarament (amb la línia més gruixuda) la mitjana dels resultats de cada algorisme, el segon i tercer quartil, i observem en la mostra del algorisme en PHP un "outlier" degut a la aleatorietat dels resultats.



Gràfic 2: Boxplot dels resultat dels tres algorismes.

Resultats: Descriptiva

- Una vegada obtingudes les dades i ben organitzades, hem procedit a l'anàlisi principal de l'estudi. Com que es tracta d'una comparació de mitjanes (qualitat de l'algorisme) entre 3 grups, calia realitzar l'estudi des del punt de vista de la descomposició de la variància d'aquesta variable resposta. Per aquest motiu, hem realitzat una descomposició de la variabilitat (ANOVA, ANalysis Of VAriance), aconseguint les mesures de variabilitat, variàncies explicades i residuals presentades en la següent taula:

Analysis of Variance Table

Response: Y

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
as.factor(x)	2	1430307	715154	2.2168	0.1126
Residuals	147	47422182	322600		

Discussió

- Segons resultats obtinguts en la taula de la descomposició de la variància, hem pogut observar que en la qualitat del algorisme, no podem refutar la hipòtesis nul·la: $H0$: µC++ =µPHP = µJAVA
- Ja que el P-valor associat és exactament: 0.1126, per tant, és major que el risc previ (α = 0.05).
- Aquest fet, ens provoca pensar que les tres mitjanes dels tres algorismes són força iguals, a més si calculem el coeficient de determinació de la següent manera:

$$R^2 = \frac{QM_E}{QM_T} = \frac{1430307}{48852489} = 0,02928$$

- Obtenim que el 2.928% la variable explicativa (algorisme emprat) no aconsegueix justificar la variabilitat total de la variable resposta (qualificació del algorisme). Per tant, podem afirmar que tots tres algorismes són iguals de eficients en quant a variabilitat.
- Finalment, ens disposem a estimar els corresponents intervals de confiança per a cada algorisme estudiat. Pel que fa a la mitjana de la qualitat del algorisme (la nostre mitjana resposta) hem obtingut les següents dades per a cadascun d'ells:
- IC µC++ ,95% = (7760'26, 8077'74)
- IC µPHP ,95% = (7659'26, 7976'74)
- IC µJAVA ,95% = (7504'26, 7821'74)

Limitacions

- No podem assumir que passarà el mateix en tots els algorismes "random" que existeixen actualment, ja que en aquest estudi solament hem seleccionat una petita mostra (els corresponents als llenguatges de programació: C++, PHP i JAVA).

- A més, també considerem que el nombre de dades generades possiblement no es el més adequat, pel que fa a quantitat de dades obtingudes.

- En possibles estudis futurs s'hauria d'obtenir un nombre més elevat de dades i més tipus d'algoritmes per així poder fer més petit el cercle del objectiu del treball.

Annex, Operacions R

- **#interval de confiança 95% en C++**

```
meanCPP = 7919; QMR = 322600;
meanCPP-qt(0.975,150-3)*sqrt(QMR/50);
meanCPP+qt(0.975,150-3)*sqrt(QMR/50);
```

- **#interval de confiança 95% de PHP**

```
meanPHP = 7818;
QMR = 322600;
meanPHP-qt(0.975,150-3)*sqrt(QMR/50);
meanPHP+qt(0.975,150-3)*sqrt(QMR/50);
```

- **#interval de confiança 95% en JAVA**

```
meanJAVA = 7663;
QMR = 322600;
meanJAVA-qt(0.975,150-3)*sqrt(QMR/50);
meanJAVA+qt(0.975,150-3)*sqrt(QMR/50);
```

Annex, Codis:

- **CODI C++:**

```
#include <iostream>
#include <vector>
#include <cmath>
#include <cstdlib>
#include <stdio.h>
#include <stdlib.h>
using namespace std;
int main(){
  srand(time(0));
  int random;
  vector <int> v (101,0);
  int suma = 0;
    for(int j = 0; j < 1000000; ++j){
      random = round(rand() % 100);
      v[random]++;
    }
    for(int j = 0; j < 100; ++j)suma += abs(v[j]-10000);
    cout << suma << endl;
}
```

Annex, Codis:

- **CODI JAVA:**

```
public static void main(String[] args) {
    Vector<Integer> v = new Vector<>() ;
    for(int j=0; j<100; ++j){
        v.add(0);
    }
    int azar;
    int z;
    Random aleatorio = new Random();
    for(int j=0; j<1000000; ++j){
        azar = aleatorio.nextInt(100);
        z = v.elementAt(azar);
        z++;
        v.add(azar,z);
    }
    int suma = 0;
    for(int l=0; l<100; ++l){
        suma += Math.abs(v.elementAt(l)-10000);
    }
    System.out.println(suma);
}
```