Determinar l'algorisme de nombres "random" més fiable en els llenguatges de programació:

C++, PHP i JAVA

Carles Guivernau – Marcel Pujol – Pau Vila

Maig 2012

RESUM

Objectiu: Observar en 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 en 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 resultats dels nombres "random" (entre 0 i 100).

INTRODUCCIÓ

En ocasions necessitem utilitzar una sèrie de nombres aleatoris, es a dir, nombres obtinguts a través de l'atzar. Nosaltres com a estudiants, a vegades necessitem usar aquests mètodes de generació de nombres, per tant, vàrem decidir estudiar quin es l'algorisme més "fiable" a la hora de generar-los amb la menor desviació possible entre ells.
Per dur a terme aquest estudi, hem utilitzat tres llenguatges de programació escollits a l'atzar: C++, PHP i JAVA, per tal de obtenir algoritmes programats òbviament de diferent manera.

Objectiu

Determinar quin algoritme de generació de nombres aleatoris "random" és el més fiable.
Entenent com algorisme fiable, un "random" que té tantes possibilitats de que surti un número com un altre.

MATERIAL I MÈTODES

Per dur a terme l'estudi, sempre utilitzarem el mateix ordinador per els algoritmes en C++ i JAVA, en canvi l'algoritme implementat en PHP s'executarà a través d'un servidor.

Executarem els tres algoritmes random un total de cinquanta vegades per cada un d'ells: primer durem a terme totes les cinquanta execucions de l'algorisme en C++, seguidament continuarem amb el de JAVA i finalment el de PHP. També tindrem en compte que entre execucions deixarem un cert temps (uns 20 minuts per evitar que hi hagin residus en el ordinador que influeixin en la recopilació de dades).

VARIABLES

L'estudi presenta la següent **variable explicativa** que definirà **l'algorisme utilitzat**. Aquesta obtindrà els següents valors:

1. rand de PHP
2. Math.random() de JAVA
3. rand de C++

I obtindrem una **variable resposta** que ens servirà per **decidir quin algorisme és millor (qualitat)**: Com que anirem traient números aleatoris del 1 al 100 i ho farem un milió de cops per cada execució de cada algorisme, mirarem per cada número del 1 al 100 quants cops ha sortit i li restarem el valor esperat (1.000.000/100 = 10.000). Un cop tinguem el valor per cada número del 1 al 100, els sumem en valor absolut i aquell serà el resultat de l'execució. Una vegada recollits els valors per totes les execucions d'aquell algorisme, farem la seva mitjana, aquesta serà la qualificació que li posem al algorisme. Com més baixa sigui la qualificació, serà millor:

$$qualificació\ execució = \sum_{i=1}^{100} (|(N_i - 10000)|)$$

On N_i és el número de cops que ha sortit i.

1. Prova de significació (unilateral):

H_0: $\mu_{C++} = \mu_{PHP} = \mu_{JAVA}$

$H_1 = \exists_{i,j}$ tq $\mu_i \neq \mu_j$

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

$$\hat{F} = \frac{QM_E}{QM_R}$$

- Sota hipòtesis nul·la, segueix una distribució $F_{m,n}$ graus de llibertat del numerador (m) y denominador (n).
- Amb un 5% de risc alfa (α).

RESULTATS

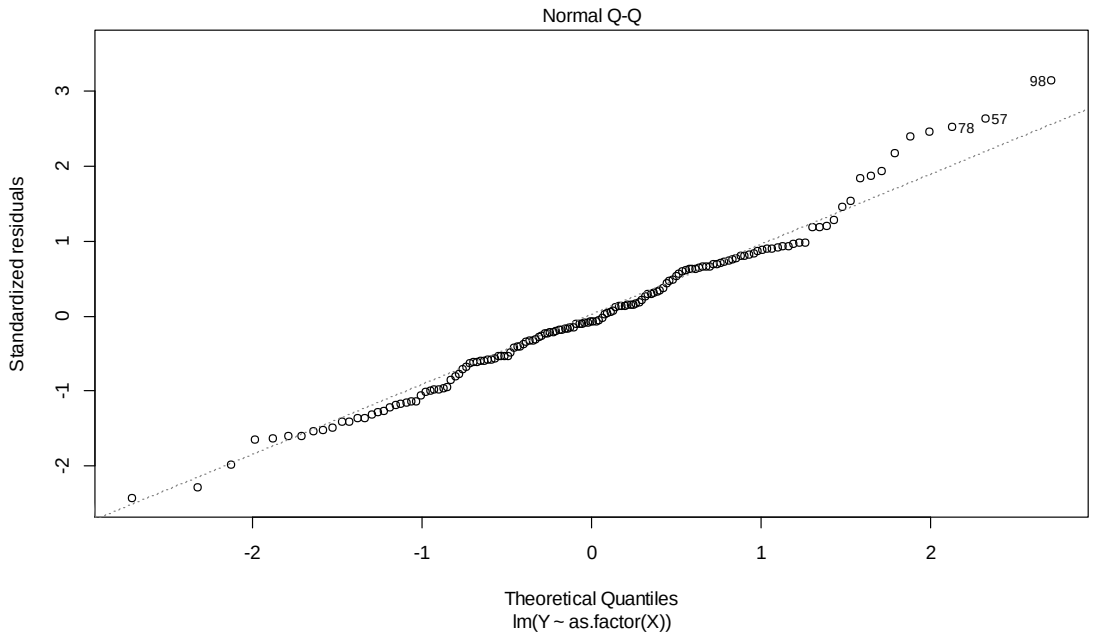
Descriptiva

La taula 1.1 mostra la mitjana i la desviació típica de la qualitat (la nostre 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

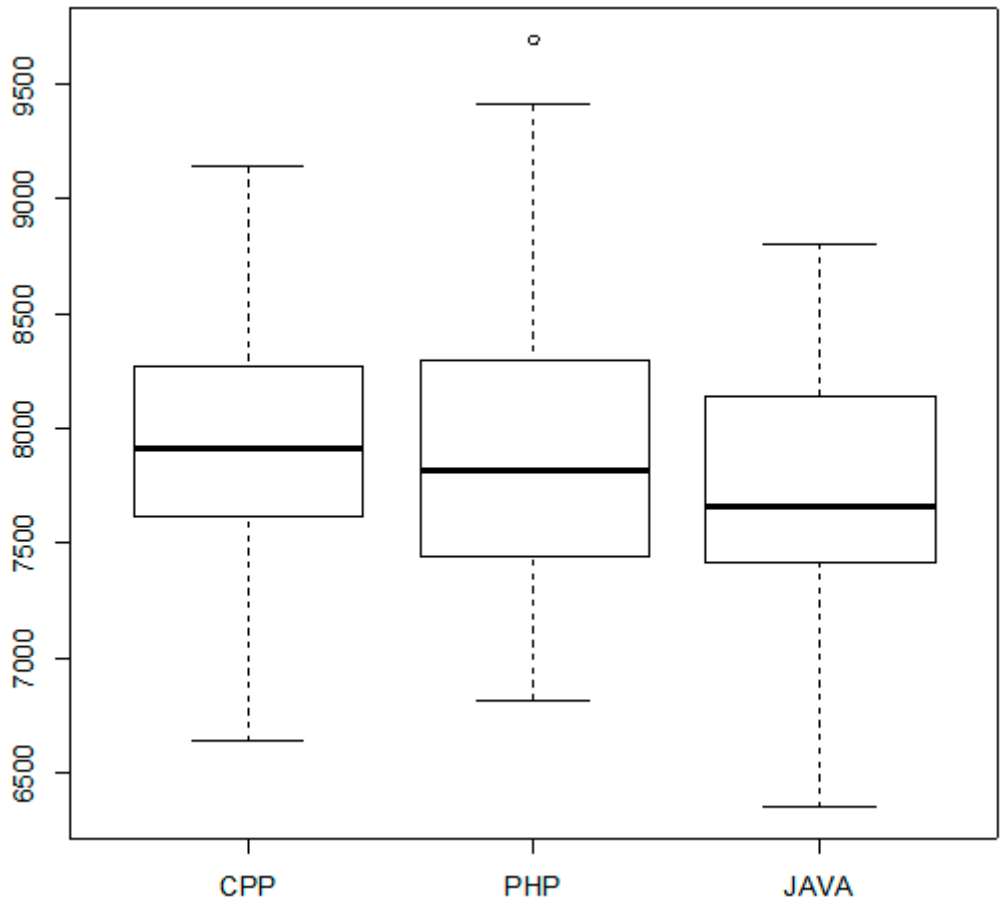
Taula 1: Anàlisis descriptiva de la variable qualitat del algorisme

Observant-la podem veure que, en mitjana, la qualitat de l'algorisme és molt similar entre els tres, ja que presenten poca diferència entre ells. En quant a la desviació estàndard dels tres algoritmes, el Math.random() de JAVA és el que presenta una dispersió major a la resta.



Gràfic 1: Anàlisis de la normalitat de la variable.

Podem veure en aquest plot dels resultats obtinguts (gràfic 1) que tots es mostren sobra la recta indicant la normalitat dels resultats que hem obtingut de les mesures dels algorismes. Si la mostra no fos normal, tindríem que rebutjar la hipòtesi nul·la.



Gràfic 2: Boxplot dels resultat dels tres algorismes.

Els Boxplot de les tres mostres ens donen una imatge gràfica dels resultats obtinguts. Podem veure clarament amb la línia més grossa 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.

ANÀLISIS PRINCIPAL

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		

Taula 2: Anàlisi de la variància en la qualitat de l'algorisme en funció del algorisme emprat.

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:

$$H_0\colon\ \mu_{\mathrm{C++}} = \mu_{\mathrm{PHP}} = \mu_{\mathrm{JAVA}}$$

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:

$$R^2 = \frac{QM_E}{QM_R} = \frac{1430307}{47422182} = 0.03016114$$

Obtenim que el 3.016% 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 en eficiència 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:

$IC\left(\mu_{\mathrm{C++}}, 95\%\right) =$ (7760'26, 8077'74)

$IC\left(\mu_{\mathrm{PHP}}, 95\%\right) =$ (7659'26, 7976'74)

$IC\left(\mu_{\mathrm{JAVA}}, 95\%\right) =$ (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

Codis utilitzats per les mesures i per l'anàlisi dels resultats obtinguts.

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;
}
```

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);
}
```

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;

?>
```

Càlculs realitzats amb 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);

#càlcul anova
> M = lm(Y~as.factor(X))
> anova(M)
```