



**USMP**  
SAN MARTÍN DE PORRES

FACULTAD DE  
INGENIERÍA Y ARQUITECTURA



**Asignatura:**

ESTADÍSTICA Y PROBABILIDADES II

**Tema:**

“Uso de R para hallar intervalos de confianza”

**Integrantes del equipo:**

- Barrera Arche, Manuel Alberto
- Ontiveros Ordinola, James
- Quichiz Santome, José Carlos
- Ozusky Castro, Ivan Alexander

**Sección:**

28D

**Profesor:**

Prof. Celso Gonzales Chavesta.

**Lima, Perú  
2013**

# INTRODUCCIÓN

El uso de la estadística hoy en día es fundamental para tomar decisiones entre otras cosas, pero el hacer estadísticas a mano lápiz y papel no es muy agradable, es por eso que existen numerosos programas estadísticos y entre ellos esta R, que más que un lenguaje estadístico es un lenguaje de programación orientado a objetos diseñado exclusivamente para hacer estadísticas.

En esta oportunidad usaremos este lenguaje para hallar intervalos de confianza, y veremos como hallar cada una de las variables (media, varianza, z, t, etc), con este lenguaje.

## **OBJETIVOS GENERALES**

- Comprender el uso de R.
- Utilizar R para hallar: tamaño de muestra, una muestra aleatoria, desviación estándar o típica, valores de tabla normal y de t de student.
- En base a los valores hallados previamente hallar los intervalos de confianza de 2 casos.

## ENUNCIADO DEL PROBLEMA

La comunidad de desarrolladores de PHP lanzo una mejora al publico del famoso comando `mysql`, que permite conectarse con bases de datos MySQL y realizar consultas .La mejora es una clase, que hace lo mismo pero de una manera mucho mas segura, la clase se llama `mysqli`.

Se desea comprobar que el tiempo promedio en el que se demora en hacer una consulta a MySQL (0.130 segundos aprox.), se encuentra dentro del intervalo de confianza de la clase `mysqli` y del comando `mysql`, hablando de la demora promedio de cada uno de estos.

Usar para esto un nivel de confianza del 95% y un margen de error del 5%.

# Marco teórico

## 1. Concepto de Intervalo de Confianza.

En el contexto de estimar un parámetro poblacional, un intervalo de confianza es un rango de valores (calculado en una muestra) en el cual se encuentra el verdadero valor del parámetro, con una probabilidad determinada.

La probabilidad de que el verdadero valor del parámetro se encuentre en el intervalo construido se denomina **nivel de confianza**, y se denota  $1-\alpha$ . La probabilidad de equivocarnos se llama **nivel de significancia** y se simboliza  $\alpha$ . Generalmente se construyen intervalos con confianza  $1-\alpha=95\%$  (o significancia  $\alpha=5\%$ ). Menos frecuentes son los intervalos con  $\alpha=10\%$  o  $\alpha=1\%$ .

### 1.1 Formula del intervalo de confianza para la media poblacional ( $\sigma^2$ desconocido)

$$\left\langle \bar{x} - t_{(1-\frac{\alpha}{2}, n-1)} \frac{s}{\sqrt{n}}; \bar{x} + t_{(1-\frac{\alpha}{2}, n-1)} \frac{s}{\sqrt{n}} \right\rangle$$

Donde:

a.  $n$  = tamaño de la muestra

b.  $s$  = desviación muestral.

### 1.2 Formula para calcular $n$

$$n = \frac{N\sigma^2 Z^2}{(N-1)e^2 + \sigma^2 Z^2}$$

## 2. Lenguaje R

### 2.1 Introduccion

R es un lenguaje Orientado a Objetos: bajo este complejo termino se esconde la simplicidad y flexibilidad de R. El hecho que R es un lenguaje de programacion puede desaminar a muchos usuarios que piensan que no tienen “alma de programadores”. Esto no es necesariamente cierto por dos razones. Primero R es un lenguaje interpretado (como Java) y no compilado (como C, C++, Fortran, Pascal, . . . ), lo cual significa que los comandos escritos en el teclado son ejecutados directamente sin necesidad de construir ejecutables

### 2.2 Comandos útiles

sample(variable, tamaño, reemplazo) => Genera una muestra aleatoria

mean(variable) => Calcula la media del conjunto de datos en la variable

sd(variable) => Calcula la desviacion estandar

var(x) => Calcula la varianza

qnorm(prob) => Calcula el valor de Z (Tabla, de adentro hacia afuera)

qt(prob, GL) => Probabilidad en t de student.

## 2.3 Otros comandos

`pnorm(z)` => Probabilidad en Z (Tabla de afuera hacia adentro)

`paste(elem1, elem2, ...)` => Concatena elementos

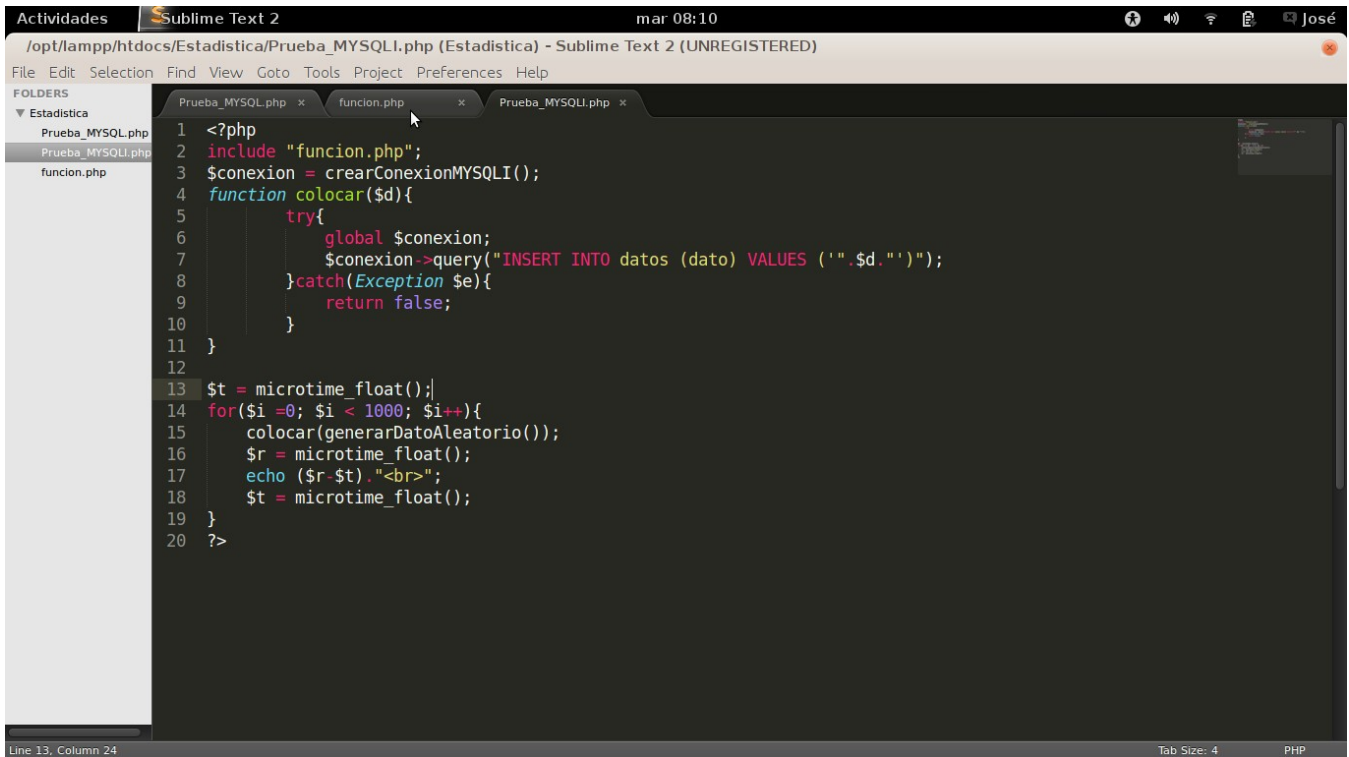
`ceiling(var)` => Redondear por exceso la variable

`scan()` => Leer datos y almacenarlos en una variable

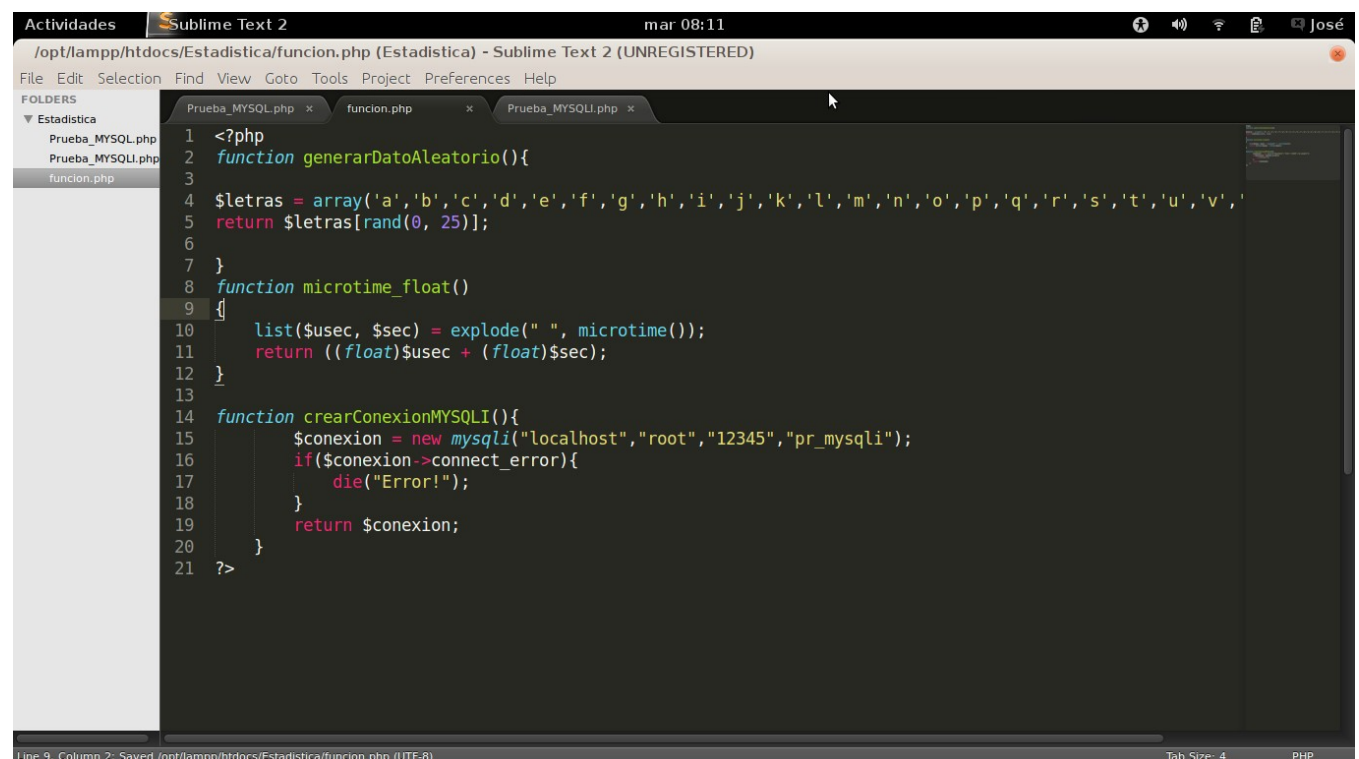
`sqrt(numero)` => Raiz cuadrada de un numero

# PROCEDIMIENTO REALIZADO

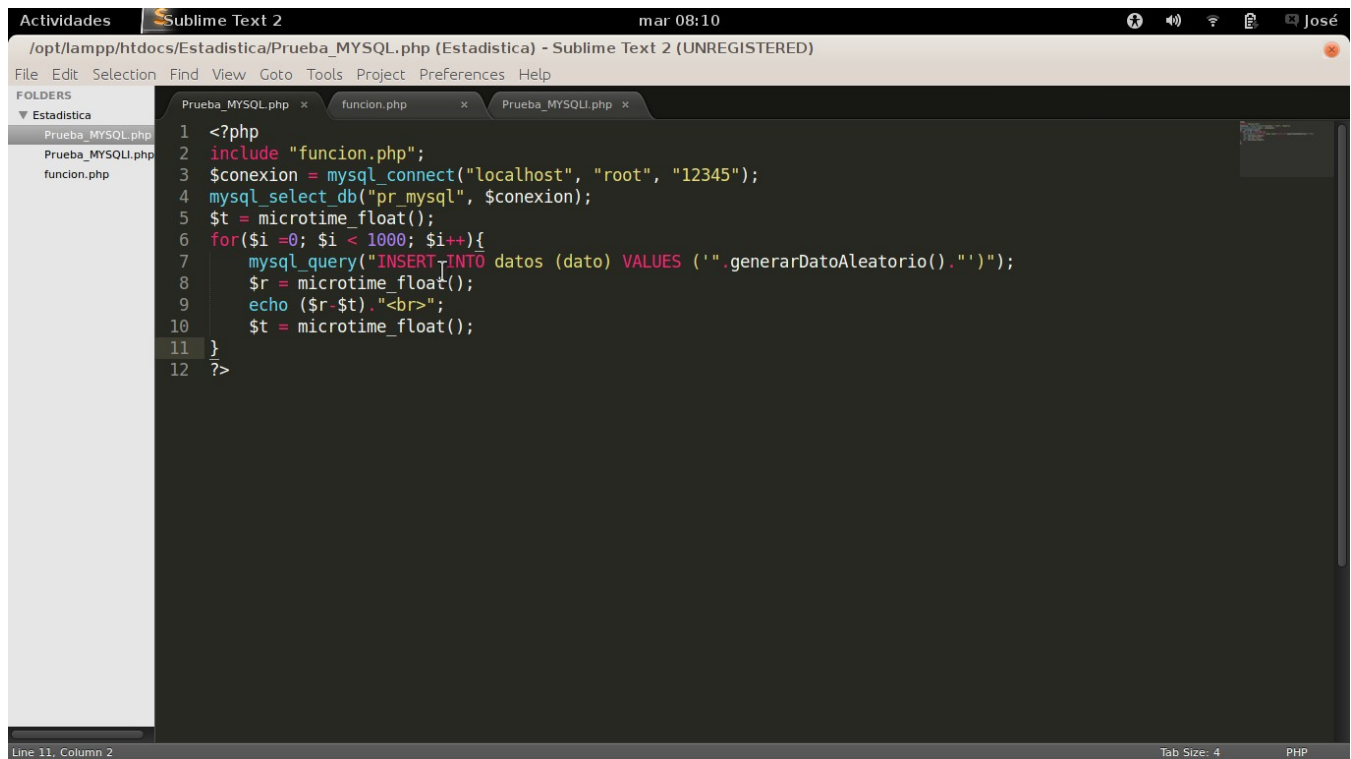
Se realizaron 1000 consultas a la base de datos, más específicamente registros y por cada registro un carácter (1 byte), y se obtuvo el tiempo que demoró cada consulta. La simulación se realizó tanto para el comando mysql como para la clase mysqli, este es el código utilizado:



```
1 <?php
2 include "funcion.php";
3 $conexion = crearConexionMYSQLI();
4 function colocar($d){
5     try{
6         global $conexion;
7         $conexion->query("INSERT INTO datos (dato) VALUES ('".$d."')");
8     }catch(Exception $e){
9         return false;
10    }
11 }
12
13 $t = microtime_float();
14 for($i =0; $i < 1000; $i++){
15     colocar(generarDatoAleatorio());
16     $r = microtime_float();
17     echo ($r-$t)."<br>";
18     $t = microtime_float();
19 }
20 ?>
```



```
1 <?php
2 function generarDatoAleatorio(){
3
4     $letras = array('a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z');
5     return $letras[rand(0, 25)];
6 }
7
8 function microtime_float()
9 {
10     list($usec, $sec) = explode(" ", microtime());
11     return ((float)$usec + (float)$sec);
12 }
13
14 function crearConexionMYSQLI(){
15     $conexion = new mysqli("localhost","root","12345","pr_mysql");
16     if($conexion->connect_error){
17         die("Error!");
18     }
19     return $conexion;
20 }
21 ?>
```



```
1 <?php
2 include "funcion.php";
3 $conexion = mysql_connect("localhost", "root", "12345");
4 mysql_select_db("pr_mysql", $conexion);
5 $t = microtime_float();
6 for($i = 0; $i < 1000; $i++){
7     mysql_query("INSERT INTO datos (dato) VALUES ('.generarDatoAleatorio().')");
8     $r = microtime_float();
9     echo ($r-$t)."<br>";
10    $t = microtime_float();
11 }
12 ?>
```

Para ejecutar el código PHP se usó el motor Apache 2.4.4, como motor de base de datos MySQL 5.5.32 y la versión de PHP usada fue PHP 5.4.19.

El script está disponible en el siguiente enlace: <http://goo.gl/FP9z0g>

Se corrió el programa y los datos fueron recopilados, se pueden descargar del siguiente enlace: <http://goo.gl/RMMvIU>

Luego, procedimos a enviar los datos a R.

Como ya dijimos R es un lenguaje orientado a objetos, esto quiere decir que los datos los vamos a almacenar en objetos o variables.

El modo de asignar un dato a una variable es el siguiente:

```
variable <- 2 o 2 -> variable
```

Esto crea el objeto llamado “variable” y le asigna el valor 2, los valores a asignar pueden ser de 4 tipos: numéricos reales, numéricos complejos, lógicos (TRUE y FALSE), y texto.

Voyendo a la resolución, para mandarle los 1000 registros a R creamos una variable llamada `res_mysql` y otra llamada `res_mysqlI`, usando el método `scan` mandamos los datos de cada simulación a sus respectivas variables, del siguiente modo:

```
res_mysql <- scan()
```



Al hacer esto se nos piden los datos como esta en la captura:

```
pepe@ubuntu: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
> res_mysql <- scan()  
1: █
```

The terminal window shows the following output:

```
981: 0.1458420753479  
982: 0.12228202819824  
983: 0.12216901779175  
984: 0.12229585647583  
985: 0.12232708930969  
986: 0.12240195274353  
987: 0.12254309654236  
988: 0.12193512916565  
989: 0.14491820335388  
990: 0.12286305427551  
991: 0.12205195426941  
992: 0.12218308448792  
993: 0.15382981300354  
994: 0.13476800918579  
995: 0.18886208534241  
996: 0.13284301757812  
997: 0.1333514068604  
998: 0.13362789154053  
999: 0.13325715065002  
1000: 0.13305997848511  
1001:  
Read 1000 items  
> █
```

The background spreadsheet (LibreOffice Calc) shows a column of numbers in column A, starting from row 1. The numbers are the same as those in the terminal output, but only the first few are visible in the screenshot.

luego copiamos los datos almacenados en la tabla Excel, los pegamos en la consola y le damos a enter para que los datos se guarden en la variable:  
se uso el mismo procedimiento para crear res\_mysql.

Una vez guardamos los datos, se procedio a hallar la desviacion poblacional de cada caso y guardarlas en las variables desvP\_mysql y en desvP\_mysql, luego se calculo el valor de z, se asigno la variable error y se asigno el alfa:

```

> desvP_mysql <- sd(res_mysql)
> sd(res_mysql)
> alfa <- 1-0.95
> z_val <- qnorm(0.95)
> error <- 0.05

```

Luego de esto, se creo una funcion dentro de R que nos calcule el tamaño de muestra, adaptando la formula antes mencionada a un modelo lineal:

```

> hallarMuestra <- function(z, o, e, N){(N*o*o*z*z)/((N-1)*e*e+o*o*z*z)}

```

donde:

- z=El valor z
- o = desviacion estandar
- e=error
- N=Poblacion

Ahora que tenemos la funcion, se procedio a hallar n

```

> n <- hallarMuestra(z=z_val, o=desvP_mysql, e=error, N=1000)

```

Como n sale decimal, de redondeo por exceso:

```

> n
[1] 9.047087
> n <- ceiling(n)
> n
[1] 10

```

Se procedio seleccionando las muestras aleatorias usando la funcion sample, y las almacenamos en muestra\_mysql y en muestra\_mysqli (ambas sin reemplazo):

```

> muestra_mysql <- sample(res_mysql, n, replace=FALSE)
> muestra_mysqli <- sample(res_mysqli, n, replace=FALSE)

```

Creamos las funciones hallarLimiteInferior y hallarLimiteSuperior

**\*Aclaración:** Para hallar la formula del tamaño de muestra se tomo en cuenta la formula del error con una varianza conocida y para hallar los limites se usara la formula de intervalo de confianza de la media poblacional con una varianza desconocida, con el fin de poder apreciar el uso de la funcion qnorm y la funcion qt.

```
> hallarLimiteInferior <- function(a,n,muestra){mean(muestra)-qt(1-(a/2), n-1)*(sd(muestra)/sqrt(n))}  
> hallarLimiteSuperior <- function(a,n,muestra){mean(muestra)+qt(1-(a/2), n-1)*(sd(muestra)/sqrt(n))}
```

Teniendo las funciones procedemos a generar los intervalos:

```
> intervalo_mysql<-paste("<",hallarLimiteInferior(alfa,n,muestra_mysql),"",hallarLimiteSuperior(alfa,n,muestra_mysql),">")  
> intervalo_mysql<-paste("<",hallarLimiteInferior(alfa,n,muestra_mysql),"",hallarLimiteSuperior(alfa,n,muestra_mysql),">")
```

y ahora, para terminar, procedemos a mostrar ambos intervalos:

```
> intervalo_mysql  
[1] "< 0.121016422513914 , 0.150613578554204 >"  
> intervalo_mysql  
[1] "< 0.12228735598091 , 0.160743297018966 >"  
>
```

## INTERPRETACION DE LOS RESULTADOS

Como podemos apreciar, 0.130 se encuentra dentro de ambos intervalos, por lo que es posible afirmar con un nivel de confianza del 95% que los intervalos contengan a 0.130.

## **CONCLUSIONES**

1. La funcion mysql y la clase mysqli tienen casi el mismo tiempo de respuesta pues los limites de sus respectivos intervalos no varían abismalmente.
2. Los programas estadísticos hacen más fácil el trabajo de desarrollar estadísticas.
3. R es muy recomendable, dado que funciona bajo varias plataformas (Windows, Linux, iOS)
4. Al ser la desviación estándar no muy grande, no se necesita una muestra demasiado grande

## **RECOMENDACIONES**

1. Se debe desarrollar una GUI para R, si es que no existe ya, pues aunque facilita el trabajo, resulta un poco tedioso comprender la sintaxis y estructura de R.

## BIBLIOGRAFÍA

- [http://cran.r-project.org/doc/contrib/rdebuts\\_es.pdf](http://cran.r-project.org/doc/contrib/rdebuts_es.pdf), Disponible
- <http://www.monografias.com/trabajos87/calculo-del-tamano-muestra/calculo-del-tamano-muestra.shtml> , Disponible
- [http://www.dm.uba.ar/materias/modelo\\_lineal/2009/1/guiar.pdf](http://www.dm.uba.ar/materias/modelo_lineal/2009/1/guiar.pdf) , Disponible
- [http://www.uam.es/personal\\_pdi/ciencias/ggarrigo/ccaa2008/comandos\\_de\\_R.pdf](http://www.uam.es/personal_pdi/ciencias/ggarrigo/ccaa2008/comandos_de_R.pdf) , Disponible
- <http://stat.ethz.ch/R-manual/R-devel/library/base/html/save.html> , Disponible
- <http://unbarquero.blogspot.com/2009/08/r-funciones.html>, Disponible