

# Trabajo práctico para R y Estadística

Ramón Díaz-Uriarte

2014-11-19 (Release 1.1: Rev: b2ab80f)

## 1 INSTRUCCIONES PARA LOS DOS ÚLTIMOS DÍAS DE CLASE

Esos dos días cubriremos la preparación de un paquete de R e iremos sobre dudas generales de R y el trabajo. Cuánto más y mejor hayáis pensado en el trabajo, más provechosas serán las sesiones.

## 2 Introducción

El trabajo consiste en que escribáis unas funciones que realicen un trabajo útil (o que podamos pretender que es útil ;-)). Si todo va bien, preparareis un paquete en R. Si no conseguimos terminarlo, preparareis las funciones sin que formen parte de un paquete. Estas instrucciones asumen que será un paquete, pero esto ya lo decidiremos.

Supongamos que decidimos preparar un paquete. Será un paquete de R básico, elemental. El objetivo de preparar este paquete es familiarizarnos con los paquetes de R, acostumbrarnos a interpretar mensajes de R, y utilizar las funcionalidades de R para testeo básico de código.

Este trabajo lo haréis en grupos de tres o cuatro. Si existen esos grupos, bien. Si no, los creamos el lunes.

Durante la presentación (4 de Abril de 2014), mostrareis que el paquete pasa los checks, que se instala, y mostrareis la ayuda y su funcionalidad. Los ejemplos tienen que correr. Tendréis que responder a preguntas mías y de vuestros compañeros, si las hubiera o hubiese. (De nuevo: si no preparáis un paquete, el 4 de Abril haremos lo de arriba, salvo lo específico del paquete.)

El paquete tiene que contener funciones para resolver el siguiente problema. Lo más sencillo es que haya una “top-level function”.

## 3 El problema del "Selection bias"



### 3.1 Introducción

Mencionamos, al hablar de clasificación, que si seleccionamos genes en problemas de clasificación, la validación cruzada ha de incorporar la selección de genes.

El objetivo de esta parte es ilustrar lo que ocurre cuando hacemos las cosas bien y cuando las hacemos mal.

- Simularemos un conjunto de datos, sin señal, para un número de sujetos y un número de genes, en un problema con dos clases (ej., sanos y enfermos).
- Usaremos como algoritmo de clasificación `randomForest`. Pero seleccionaremos genes antes de usarlos en el clasificador. Los genes seleccionados serán aquellos con el menor p-valor en un test de la t (un procedimiento de “filtrado”).
- Evaluaremos el funcionamiento del clasificador con “cross-validation”. Sin embargo, en un caso la selección de genes se habrá hecho antes de la partición para la validación cruzada (o sea, mal) y en el otro la selección de genes se hará en cada partición. Compararemos resultados.

## 3.2 Detalles y pistas

- Usaremos `randomForest` como clasificador. Tendréis que instalar este paquete.
- El número de genes a seleccionar es un argumento de la función.
- Como arriba, tendréis que simular datos (y número de genes y sujetos son argumentos de la función).
- Para la validación cruzada, las siguientes líneas de código muestran cómo seleccionar los casos de una forma muy compacta:

```
## De Venables y Ripley, "S programming", 2000,
##           Springer, p. 175
## x2 es un vector con datos
x2 <- rnorm(27)
N <- length(x2)
knumber <- 10 ## el k-fold
## en la k-vez, dejamos en testing set
## los que tienen index.select = k
## O sea, index.select es el vector de índices
index.select <- sample(rep(1:knumber,
                           length = N),
                      N, replace = FALSE)

rep(1:knumber, length = N)
table(index.select)

sum(index.select != 1)
sum(index.select != 10)
sum(index.select != 6)
```

Con eso, podríamos hacer cosas como


```
hace.algo.con.train.y.test <- function(train, test) {
```

```

## esto hace algo con train and test
## blablabla
}

for(sample.number in 1:knumber) {
  x2.train <- x2[index.select != sample.number]
  x2.test <- x2[index.select == sample.number]
  hace.algo.con.train.y.test(x2.train, x2.test)
}

```

- Tenemos que entrenar un modelo con unos datos, y testarlo con otros. En el caso concreto de `randomForest`, lo primero lo haremos con la función `randomForest` y lo segundo con `predict` (realmente, `predict.randomForest`).
- ¿Cómo obtenéis el error de clasificación? Comparando observado con predicho. ¿Y eso cómo se hace?
- Además del error de clasificación, usad los votos o probabilidades que da `randomForest` cuando predice algo, para obtener el score de Brier y/o el índice de concordancia. 
- Ojo: `randomForest` (y la mayoría de las funciones para modelos estadísticos) esperan estructuras con sujetos en filas y variables en columnas. Algo tendréis que hacer.
- Querremos repetir todo el procedimiento unas cuantas veces, para hacernos una idea de variabilidad, etc.
- Los siguientes deberían ser argumentos de la función (en paréntesis, una sugerencia sobre valores por defecto): número de genes (1000), número de sujetos (50), el “k” en el k-fold cross-validation (10), número de genes que seleccionamos (10), número de veces que repetimos todo el procedimiento (10).
- Querréis mostrar gráficos y algún tipo de output de texto. Por ejemplo, el error de clasificación medio cuando se hace como es debido (selección de genes dentro de la validación) y cuando se hace mal, las diferencias, etc.
- Una vez que vuestro código funcione, podréis pensar preguntas como: ¿es el selection bias más o menos severo si hay muchos genes en relación con el número de sujetos? ¿y qué ocurre a medida que el número de genes seleccionados aumenta o disminuye? Tres o cuatro escenarios básicos pueden ser parte de los ejemplos en la ayuda.

## 4 Instrucciones para preparar el paquete

1. Usar la función `package.skeleton` para crear el armazón del paquete.
2. Editar las funciones de ayuda, para que el paquete se pueda crear. La mayor parte de las secciones de la ayuda pueden quedar vacías, pero el título y la descripción deben llevar algo. Y los ejemplos también, claro.

3. ¿Qué hay en el subdirectorio R? ¿Está bien así? ¿Alguna cosa que sobre o que sea mejor unificar?
4. Asegurarse de incluir ejemplos relevantes.
5. Asegurarse de que pasa el R CMD check el-nombre-del-paquete.
6. Crear el “tar.gz”: R CMD build el-nombre-del-paquete.
7. Instalar (a ser posible en otra máquina) el tar.gz:  
R CMD INSTALL el-nombre-del-paquete-y-version.tar.gz.
8. Los warnings que se emiten: ¿nos molestan? No hay ningún problema con los warnings, siempre y cuando podáis entender y explicarlos todos.
9. Por favor, la documentación y los comentarios del código **en inglés**.

