Ejercicios de programación III: Trabajo con datos

[IMSER 2012]

Archivos incluidos:

El archivo con los ejercicios del práctico debe bajarse y descomprimirse en disco duro, creando la carpeta `rep-2` (nota: no debe dentro de ningún disco, partición o carpeta protegida a la escritura, como puede ser un disco duro externo de backup). Usted deberá abrir el RStudio y seleccionar dicha carpeta como su directorio de trabajo con `setwd` o en RStudio la combinación **Ctrl + Shift + K**. En esta carpeta se encuentran algunos archivos que usted deberá modificar:

- `importar.R`
- `parche.R`
- `filtrado.R`
- `est.R`
- `transformar.R`
- `nuevo-factor.R`
- `exportar.R`

Adicionalmente los siguientes archivos son necesarios, pero **no deben ser modificados** para que el método de calificación automático funcione correctamente.

- `evaluar.R`
- `notas.csv`
- `datos`
- `INSTRUCCIONES.pdf`
- `est.rda`

Mecanismo de corrección:

Lo primero que debe hacer es cargar el archivo evaluar.R con la función `source` y la codificación de caracteres "UTF-8" (lo cual afecta a la función `evaluar` en particular), de la siguiente manera:

```
options(encoding = "utf-8")
source("evaluar.R")
```

Si usted ha ejecutado todos los pasos anteriores correctamente, la siguiente frase debería verse en la consola:

```
Archivo de codigo fuente cargado correctamente
```

En caso de que ocurra un error o se vea otro mensaje en la consola, verifique que los archivos se descomprimieron correctamente y que usted está trabajando en la carpeta correspondiente con el comando `getwd()`.

Usted trabajará modificando los contenidos de dichos archivos con RStudio (u otro programa de su preferencia) según las consignas que se describen a continuación. Luego de terminar cada ejercicio y **guardando el archivo** correspondiente en el disco duro, usted podrá verificar rápidamente si su respuesta es correcta ejecutando el comando:

```
evaluar()
```

y además podrá en todo momento verificar su puntaje con la función `verNotas()`. Tenga siempre en cuenta que, a **menos que sea indicado** por la letra del ejercicio, las soluciones deben ser genéricas y por lo tanto deben obtenerse con el código de los scripts en lugar de ser valores fijos. Usualmente se utilizan valores generados de forma aleatoria para las correcciones automáticas. Los objetos que son evaluados en la corrección automática estarán indicados con un asterísco en las instrucciones de cada script. Nótese además que en los archivos **se indica claramente en dónde se inicia y dónde finaliza su código** y que debe respetar esta organización para que la corrección de los ejercicios funcione bien.

Al finalizar

Una vez terminados y guardados los archivos de los ejercicios del repartido, usted deberá ejecutar `evaluar()` y seleccionar la última opción ("Todos") y luego subir el archivo "datos" (sin extensión), incluido en la carpeta "rep-1", a la sección de entregas de la portada del curso en la plataforma EVA. Este archivo se podrá reemplazar con uno más nuevo, en caso de que desee corregir algún error; en caso de querer que el archivo sea corregido antes de la fecha de entrega, puede cambiarle el nombre a "datos-finalizado", pero en ese caso la nota no se cambiará de ahí en adelante.

Código de Honor

Si bien animamos a que los estudiantes trabajen en equipos y que haya un intercambio fluido en los foros del curso, es fundamental que las respuestas a los cuestionarios y ejercicios de programación sean fruto del trabajo individual. En particular, consideramos importante que los estudiantes no miren el código creado por sus compañeros ya que esto supone un sabotaje a su propio proceso de aprendizaje. Como profesores estamos comprometidos a pedir tareas para las cuales hayamos dado las herramientas correctas y las explicaciones adecuadas como para que todos puedan encontrar su propio camino para resolver los ejercicios.

---

1. Datos de EEUU

En la carpeta del repartido 'rep-3' se encuentra una planilla de cálculo en formato xls, llamada "usa.xls". Esta planilla tiene una serie de variables medidas para los cincuenta estados de EE.UU., durante los años setenta. Para mayor información de estos datos, ejecute el comando `?state`.

A partir de esta base de datos vamos a trabajar a lo largo de todo el repartido, ejercitando las habilidades y conocimientos necesarios para modificar y manipular data.frames. Esta es la principal clase de objeto para trabajar con datos que tiene R y por lo tanto nos centramos principalmente en esta.

1.a Importar los datos

Script: "importar.R"

Lo primero que se debe hacer es importar los datos a R. Para ello usted deberá exportar la planilla de cálculo desde excell u otro programa capaz de trabajar con ella. Dicha exportación deberá hacerse en un archivo de texto plano (cuyas extensiones suelen ser .txt o .csv). En la evaluación automática no se corregirá dicho archivo, pero es necesario que se encuentre en la carpeta del repartido para que pueda ejecutarse el script "importar.R".

En dicho archivo usted deberá crear el código necesario para importar los datos a R usando alguna de las variantes de `read.table`. La única condición importante es que la primer columna (los nombres de los estados de EE.UU.) debe importarse como los nombres de filas de la data.frame resultante en R. Dicha data.frame deberá llamarse `usa` (ver instrucciones en el arcihvo "importar.R").

Pero además de importar los datos a su área de trabajo, el script debe también cambiarle los nombres de las variables al español. Específicamente, los nombres de las variables de la data.frame deben ser (en el mismo orden y respetando mayúsculas y minúsculas):

`Abrev, Poblacion, Ingresos, Analf, Esp.Vida, Homicidio, Sec.Grad, Heladas, Area, Division`

Por último, en la variable `Division` también queremos cambiar los nombres de los 9 niveles (se trata de un factor) a una versión en español. Específicamente (en el mismo orden y respetando mayúsculas y minúsculas):

`Noreste Central, Sudeste Central, Atlantico Central, Montania, Nueva Inglaterra, Pacifico, Atlantico Sur, Noroeste Central, Sudoeste Central`

Nótese que no se usan tildes ni eñes en los nombres para evitar problemas de codificación de caracteres y que se deben respetar mayúsculas y minúsculas.

(Pista: considere usar la función `levels` para esta tarea.)

1.b Corregir datos de analfabetismo

Script: "parche.R"

Este ejercicio parte de la base que usted logró importar con éxito los datos de "usa.xls" como se indica en el ejercicio anterior. Si usted ejecuta ahora el comando

```
summary(usa)
```

podrá notar que para las columnas `Ingresos` y `Analf` figura un conteo de la cantidad de NA's. Esto quiere decir que hay datos faltantes ("Not Available"; ver en el glosario para mayor información). Esto suele ser un problema para trabajar con datos en general y por lo tanto es importante encontrar la forma sortear este tipo de obstáculos.

Afortunadamente en este caso tenemos una tabla de datos auxiliar que nos permite completar lo que nos falta para la columna `Analf` (tasa de analfabetismo). Esta tabla auxiliar está en el archivo "usa-extra.csv". Para completar el ejercicio, usted deberá completar todos los pasos necesarios para

1. importar estos datos,
2. seleccionar los valores de analfabetismo de los estados correctos y
3. sustituir los NA de la data.frame `usa2`, columna `Analf`, por estos datos seleccionados.

Nótese que la data.frame `usa` debe permanecer incambiada y se debe crear el objeto `usa2` para hacer estas modificaciones. Nótese también que "usa-extra.csv" es una tabla muy distinta a la planilla original, incluyendo sólo 2 columnas y menor cantidad de filas, por lo que la única manera de determinar la ubicación de los valores correctos es a través del uso de los nombres de los estados como referencia. En este sentido es bueno recordar que el operador lógico `%in%` puede ser de mucha utilidad; también es importante recordar que este no es un operador conmutativo; es decir no es lo mismo

```
x %in% y
```

que

```
y %in% x.
```

1.c Eliminar filas sin datos de ingresos

Script: "filtrado.R"

Así como para el analfabetismo tuvimos una forma de llenar un vacío de datos, para el caso de la columna "Ingresos" no tenemos la misma suerte. Por lo tanto, considerando que lo mejor es dejar de lado los casos en que hay ausencia de datos, en esta parte del ejercicio vamos a eliminar las filas correspondientes de "usa2".

Para esto usted deberá escribir el código necesario en el archivo "filtrado.R". Este código debe asumir la existencia de una data.frame llamada `usa2`, y servirá para obtener finalmente una data.frame `usa3` a través de la eliminación de las observaciones de `usa2`, columna `Ingresos`, en las que ocurren valores NA's.

(Pista: considere usar la función `subset` para esta tarea).

1.d Extra: función para estandarizar valores de un vector

(*Este ejercicio es opcional, aunque puede sumar puntos en su calificación final del repartido*)

Script: "est.R"

Muchas veces es útil al analizar datos transformar variables usando distintas fórmulas. Una de ellas es la estandarización de datos, utilizando la fórmula:

$$Z = \frac{X - \mu}{\sigma}$$

En donde X representa a los datos originales, μ es el valor promedio de los X, *sigma* es el desvío estándar de los X y los Z son los valores estandarizados.

En este ejercicio usted deberá crear una función llamada `est` (puede tomar como ejemplo las realizadas en el primer repartido u otras mostradas en las lecciones) que tome como entrada **un sólo argumento**: un vector numérico cualquiera y devuelva otro vector numérico con los valores estandarizados del original. Para esto deberá escribir el código necesario en el archivo "est.R".

Aconsejamos utilizar las funciones `mean` y `sd` para obtener μ y σ respectivamente. Además es deseable que las normalizaciones de datos no se vean afectadas por la ocurrencia de NA's. Por lo tanto, es necesario utilizar el argumento `na.rm` de dichas funciones para que `est` maneje correctamente los NA's. En caso de que la haya construido bien, debería obtener resultados como el siguiente:

```
x <- c(4.5, 12.3, 5.8, 9.4, 7.9, NA)
est(x)
```

```
## [1] -1.13584  1.41000 -0.71153  0.46347 -0.02611       NA
```

Nótese que si la función `est` no maneja correctamente los NA's, entonces el resultado sería igual a `rep(NA, 6)`.

1.e Estandarizar los datos

Script: "transformar.R"

La estandarización o normalización es una tranformación común en el análisis de datos. En general para cualquier tipo de transformación, si se trata de un trabajo con matrices o data.frames, es común en R el uso de las funciónes del tipo `apply`, ya que permiten modificar varias columnas en un sólo comando y además pueden ser más eficientes (en particular `lapply` o `sapply`) cuando se trabaja con grandes cantidades de datos.

En este ejercicio el objetivo es usar la función `est` creada en el ejercicio anterior, en conjunción con `apply`, para transformar las columnas de clase "numeric" de nuestra data.frame `usa3`. Específicamente, debe usar `apply` para transformar el objeto `datosNumericos`, el cual están en el script "trasnformar.R". En caso de **no haber hecho** el ejercicio 1.d, puede cargar una función `est` hecha de antemano con el comando:

```
load("est.rda")
```

En el script "transformar.R" se indica específicamente en qué línea debe utilizarse `apply` para que la evaluación del ejercicio funcione correctamente (i.e.: la línea en que se crea el objeto `datosTrans`). Si usted ha creado correctamente el objeto `datosTrans`, entonces este será de clase "matrix" y los promedios de las columnas `Poblacion` y `Area` serán respectivamente:

```
colMeans(datosTrans[, c('Poblacion', 'Area')])
    Poblacion          Area
-6.745250e-17 -2.678736e-17
```

Finalmente, tome en cuenta también que el objeto final que usted debe crear, llamado `usaNorm`, debe ser de clase 'data.frame' y debe tener las mismas columnas de clase "factor" del objeto inicial `usa3`. También deben coincidir los nombres de filas y columnas. Para esto recomendamos primero coercionar `usaNorm` en una data.frame con la función correspondiente y luego unir el objeto resultante con las columnas "factor" de `usa3` (siempre manteniendo el orden de columnas).

1.f Extra: un nuevo factor

(*Este ejercicio es opcional, aunque puede sumar puntos en su calificación final del repartido*)

Script: "nuevo-factor.R"

En este ejercicio se propone crear una nueva columna de clase "factor" en la data.frame `usa3`, utilizando la función `cut`. Dicho factor deberá llamarse `Ing.Cat` (como se ilustra en el script), tener 4 niveles y ser construido en base a la columna `Ingresos` de `usa3`. Esta variable representará entonces las 4 categorías de ingreso (promedio, por estado) de EE.UU. Si el ejercicio fue hecho correctamente, el conteo de ocurrencias de cada nivel del factor será:

```
> tabulate(usa3$Ing.Cat)
[1] 12 19 11  1
```

En segundo lugar, se utilizará la función `tapply`, una variante bastante especializada de `apply` (y muy similar a la función `by`), para analizar los valores de analfabetismo correspondientes a estas categorías de ingresos. La función `tapply` se usa con la siguiente sintaxis:

```
tapply(x, f, fu)
```

En dónde `x` es típicamente un vector numérico, `f` es un factor cuya longitud equivale a la de `x` y `fu` es una función de R (p.ej.: `mean`). Aquí lo que haría este comando es ejecutar la función `fu` tantas veces como niveles tiene `f` y en cada vez lo hace usando como entrada los elementos de `x` que se corresponden con las ocurrencias de dicho nivel. Es decir, primero ejecuta

```
fu(x[f == levels(f)[1]])
```

luego

```
fu(x[f == levels(f)[2]])
```

y así sucesivamente hasta llegar al último nivel de `f`. Como resultado devuelve una lista en la que cada elemento se corresponde con la salida de uno de estos comandos.

Lo que usted deberá ejecutar aquí es la función `tapply` sobre el vector `Analf`, con `Ing.Cat` como referencia (ambas columnas de `usa3`) y la función `summary`. El resultado, tal como se muestra en el archivo de código fuente, debe guardarse en el objeto `salidaTapply`.

Finalmente debe hacer algo similar con la función `boxplot`, cuya sintaxis es mucho más sencilla que `tapply`, por ejemplo:

```
boxplot(y ~ f, d)
```

Aquí `d` es una data.frame, mientras que `y` y `f` son columnas de `d` de las clases "numeric" y "factor" respectivamente. Nuevamente las columnas a utilizar son `Analf` e `Ing.Cat`, de la data.frame `usa3`. La salida de esta función es doble, por un lado un objeto (el cual deberá guardar bajo el nombre `salidaBoxplot`) y por otro una gráfica similar a la Figura 1 de este repartido.

1.g Exportar

Script: "exportar.R"

Finalmente se deberá exportar la data.frame `usaNorm` a un archivo de texto plano. Dicho archivo se llamará "usa-norm.csv" y deberá cumplir con las siguientes condiciones:

1. Deberá guardar correctamente los nombres de las filas y columnas.
2. El separador de columnas deberá ser el caracter ";".
3. El punto decimal debe estar indicado con el caracter ","

Consulte las lecciones o la ayuda de R en `?write.table` para determinar el comando adecuado para realizar esta operación.

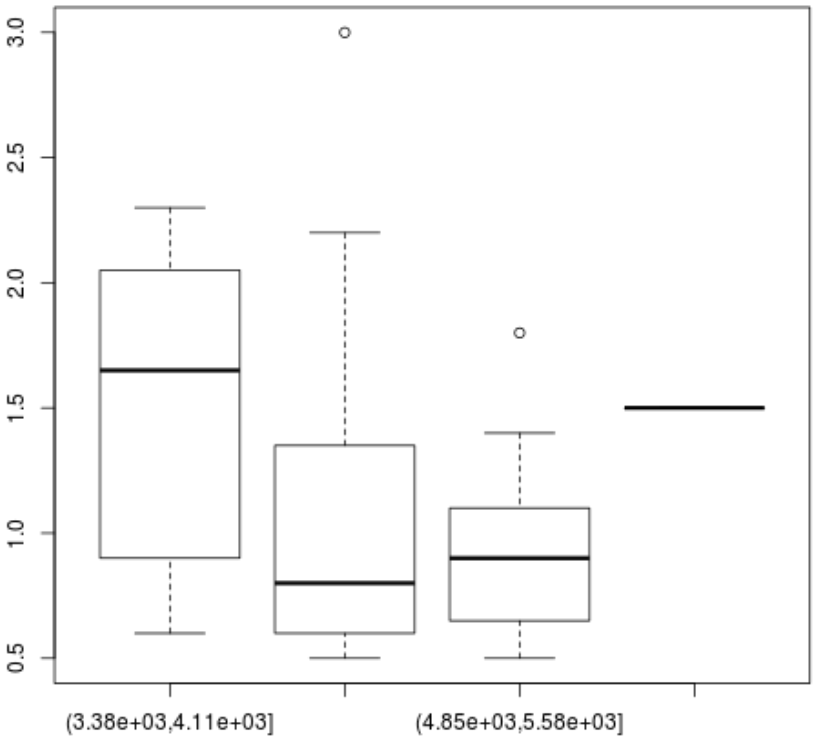

Figure 1: Ejemplo de boxplot