Ejercicios de programación IV: Gráficos y estadística

[IMSER 2013]

Archivos incluidos:

El archivo con los ejercicios del práctico debe bajarse y descomprimirse en disco duro, creando la carpeta `rep-X` (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:

- `1.a-genero.R`
- `1.b-hist.R`
- `1.c-cajas.R`
- `1.d-regresion.R`
- `1.e-dispersion.R`

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

- `evaluar.R`
- `datos`
- `INSTRUCCIONES.pdf`
- `hacemagia.R`
- `auxiliar.RData`

Mecanismo de corrección:

Nota: más recomendaciones **importantes** se hacen en el documento Dinámica de los repartidos.

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:

```
source("evaluar.R", encoding = "UTF-8")
```

Nótese que hemos dejado de usar la función `options`, de forma que de ahora en más **no ejecute el comando**:

```
options(encoding = "utf-8")  # No me ejecuten!
```

Este cambio se debe a que hemos detectado que esta elección trae más problemas que soluciones.

Si usted ha ejecutado todos los pasos anteriores correctamente, al usar el comando `ls()` verá que `"evaluar"` figura en su sesión y además en la consola debería ver lo siguiente:

```
Archivo de código fuente cargado correctamente

Chequeo de encoding:
  Los siguientes caracteres deben ser vocales con tilde:
    á - é - í - ó - ú
  Si *no se ven correctamente* corra el siguiente comando:
    source('evaluar.R', encoding = 'UTF-8')
```

Usted trabajará modificando los contenidos de los archivos de los ejercicios 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 servir aún si se modifican los datos originales (i.e.: no use valores fijos si no comandos). 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 trabaje 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 necesario que no utilice el código creado por sus compañeros, si no que debe programar sus propias instrucciones, ya que de lo contrario supone un sabotaje a su propio proceso de aprendizaje. Esto implica también evitar, en la medida de lo posible, exponer el código propio a sus colegas. Como profesores estamos comprometidos a dar nuestro mayor esfuerzo para dar las herramientas y explicaciones adecuadas a fin de que pueda encontrar su propio camino para resolver los ejercicios.

En casos de planteos de dudas a través del foro, en los que considere que es imposible expresar un problema sin exponer su própio código, entonces es aceptable hacerlo. De todas formas en estos casos es preferible que envíe su código por correo electrónico directamente a un profesor, explicando la problemática.

---

1. Campeonato de Magic

El juego Magic: el encuentro, popular entre muchos jóvenes aficionados a los generos literarios de fantasía y los juegos de rol, congrega campeonatos mundiales regularmente. Pensando en estos campeonatos, creamos una data.frame de datos ficticios de los participantes. Para a agregar a su sesión de R una data.frame llamada `magic` con dichos datos, ejecute:

```
source("hacemagia.R")
```

A partir de esta data.frame haremos varios análisis sencillos y algunos gráficos correspondientes.

1.a Variable "genero"

La variable `genero` de la data.frame presenta los valores 1 y 2. Transforme esta variable en factor y luego modifique los nombres de los niveles del mismo a `"mujer"` y `"hombre"` (correspondientes a los valores originales 1 y 2 respectivamente).

Por ejemplo, la variable original:

```
> magic$genero[1:4]
[1] 1 2 1 2
```

... y la variable modificada:

```
> magic$genero[1:4]
[1] mujer  hombre mujer  hombre
Levels: mujer hombre
```

1.b Histogramas

Nota: la corrección de este ejercicio asume que usted ya cambió la columna `genero` según las instrucciones de 1.a.

Aquí debe hacer un gráfico como el de la figura 1. Se trata de dos histogramas de los valores de alturas de los participantes, separados por sexo. En el histograma de arriba se encuentran las alturas de las mujeres, y de los hombres en el de abajo. Para lograr este gráfico hay que comprender al menos dos comandos:

i. función `par` Con la función `par` se pueden determinar la cantidad de gráficos que vamos a tener dentro de cada figura. Para eso se usa alguno de los parámetros `mfcol` o `mfrow` (no al mismo tiempo). Utilice estos parámetros correctamente para lograr dividir la figura en dos gráficos tal como en el ejemplo.

ii. función `hist` Con la función `hist` usted puede graficar rápidamente un histograma en R. En este caso tiene que utilizar los valores de altura, pero recuerde **separar por hombres y mujeres** antes de hacer los dos histogramas. Por otro lado debe fijar la cantidad de celdas/cortes en *25*, utilizando el argumento correcto de la función. Es posible que en la figura final este número no coincida con la cantidad de barras, ya que actúa tan sólo como una sugerencia para `hist`.

Nota: la cantidad de cortes asignada parece una exageración, pero se trata de un número que sirve a los propósitos de la corrección automática.

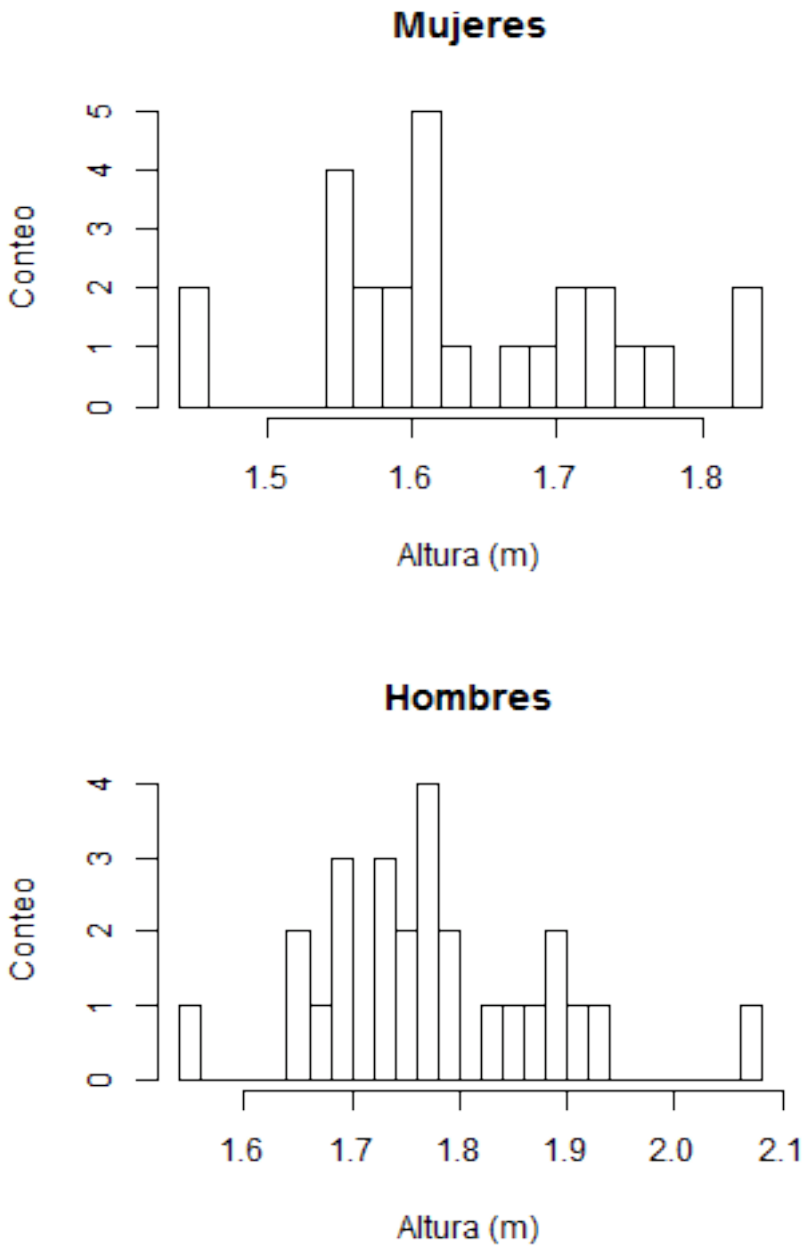

Figure 1: Histograma de las alturas de mujeres y hombres (ej. 1.b)

1.c Gráfico de cajas

En este ejercicio simplemente vamos a graficar utilizando los argumentos que editan los títulos de los gráficos. La idea es hacer un gráfico de cajas del peso de los participantes en función de su género, poniendo textos adecuados para el mismo. Estos textos serán los siguientes:

- Título: "Peso en funcion del genero"
- Etiqueta del eje horizontal: "Genero"
- Etiqueta del eje vertical: "Peso (Kg)"

(Se omiten las comas para evitar problemas de encoding.)

Recuerde que para agregar estos elementos debe utilizar los parámetros gráficos de R, los cuales aparecen listados en la ayuda de la función `par`. Estos parámetros se utilizan como argumentos al momento de llamar a plot, por ejemplo:

```
plot(speed ~ dist, data = cars, col = "red")
```

Aquí se utiliza el parámetro `col` como argumento de `plot`, asignándole el valor "red". Nótese que también se pueden asignar parámetros con comandos del tipo `par(parametro = "valor")`, pero en este ejercicio no está contemplada esta opción, por lo que no debe utilizarla.

Para hacer este gráfico tanto `plot` como `boxplot` son opciones válidas. Recuerde respetar mayúsculas y minúsculas en los textos, así como la ausencia de tildes. La corrección sólo contempla el uso de objetos de clase "formula" para hacer el plot. Es decir, debe nombrar las variables siguiendo el esquema `y ~ x`, como en el ejemplo anterior (use `?plot.formula` para ver la documentación). También es necesario que utilice el nombre del argumento `data` para indicar el data.frame (i.e.: `data = magic`).

Por último, las variables deben ser los nombres de las columnas: no sirven opciones como `cars$dist` o `cars[,2]`; debe usarse directamente `dist`.

Nota: si no tiene hecha la parte 1.a puede cargar una data.frame `magic` auxiliar con el comando:

```
load("auxiliar.RData")
```

1.d Regresiones lineales

En este ejercicio usted hará la regresión lineal entre dos variables: la altura elevada al cuadrado (*variable de respuesta*) en función del peso (*variable explicativa*) de

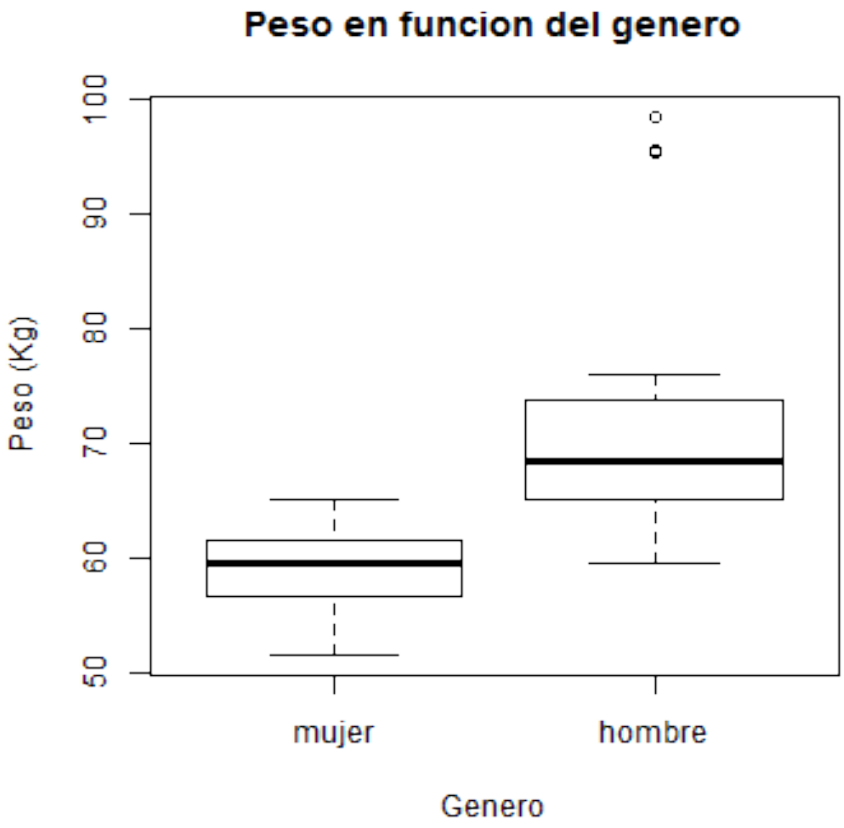

Figure 2: Gráfico de cajas (ej. 1.c)

los participantes del campeonato. Debido a que lo esperable es que a medida que el peso se aproxima a 0 la altura también lo haga, la regresión se hará **sin intercepto**. El objeto `reg.a` será esta regresión, la cual debe crearse con la función `lm`.

Posteriormente hará otra regresión, pero eliminando outliers (`reg.b`; ver fig. 3). Se detectó que hay algunos participantes con un peso mucho mayor al esperable por su altura. Por eso es más razonable eliminar estos valores para obtener una regresión que sirva para hacer predicciones útiles. Haga una regresión idéntica a la anterior, pero de tal forma que **no se usen** las filas de `magic` tales que el peso es igual o mayor a 95 Kg.

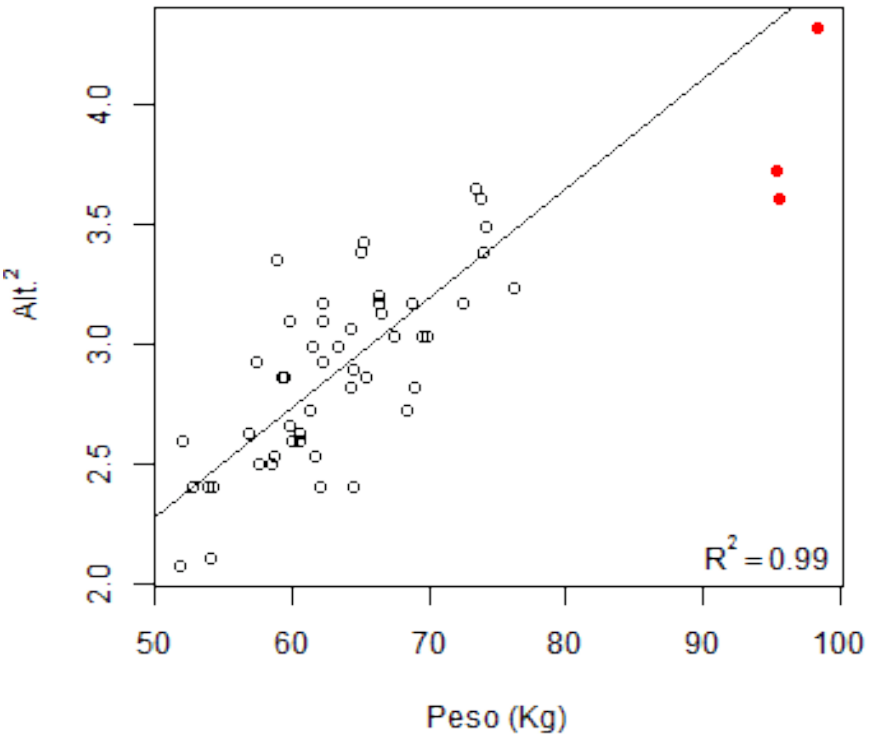

Figure 3: se muestran en rojo los outliers; la recta es reg.b.

Con esta segunda regresión ahora usted va a hacer predicciones de alturas en base un vector de pesos generado aleatoriamente (`p`). Considere que si a es la altura y c es el coeficiente obtenido en la regresión lineal (i.e.: la pendiente de la

recta), entonces se debe cumplir que:

$$a^2 = c \cdot p$$

$$a = \sqrt{c \cdot p}$$

El objeto `ae` deberá ser un vector con las alturas esperadas dado el vector de pesos `p`.

Finalmente guardará el valor del coeficiente de determinación o R^2 en el objeto `r2`. Para obtener este valor recuerde que la función `summary` aplicada a objetos `lm1` devuelve un objeto de tipo lista, el cual incluye el valor del R^2. Para ubicar fácilmente en dónde se encuentra este valor en la lista `str` puede ser una función muy útil. Por supuesto que también puede basarse en un libro de texto para hallar este valor.

1.e Gráfico de dispersión

En este ejercicio se propone reproducir una figura como la 4. Para eso el código ya está a medio hacer, usted sólamente necesita ajustar los objetos que se crean antes de los plots para que se asemeje a lo que se ve en el ejemplo. Tiene libertad para elegir ciertos parámetros, como los colores o el tipo de puntos.

De todas formas, el órden en que se pusieron las funciones no es el correcto, por lo que deberá mover de lugar las líneas de código que se encuentran bajo el marcador `##!`. La clave está en identificar cuáles son funciones de "alto nivel" ("High Level Plot functions") y cuáles no. Una vez que ordena por alto/bajo nivel, el resto puede ordenarlo como prefiera.

También podrá agregar nuevos argumentos a los plots, como

Nota: si no tiene hechas las partes 1.a y/o 1.d puede cargar una data.frame `magic` auxiliar, así como la regresión `reg.b` necesaria, con el comando:

```
load("auxiliar.RData")
```

1.f Leyenda (sin calificación)

(*Este ejercicio no se podrá evaluar con el mecanismo de corrección automática y por lo tanto no entrará en la nota total. Puede considerarlo simplemente como un ejercicio "desafío".*)

El objetivo es sencillo: tratar de lograr un resultado similar a la figura 5. Como este no tiene puntaje es totalmente opcional. Invitamos a todos los interesados

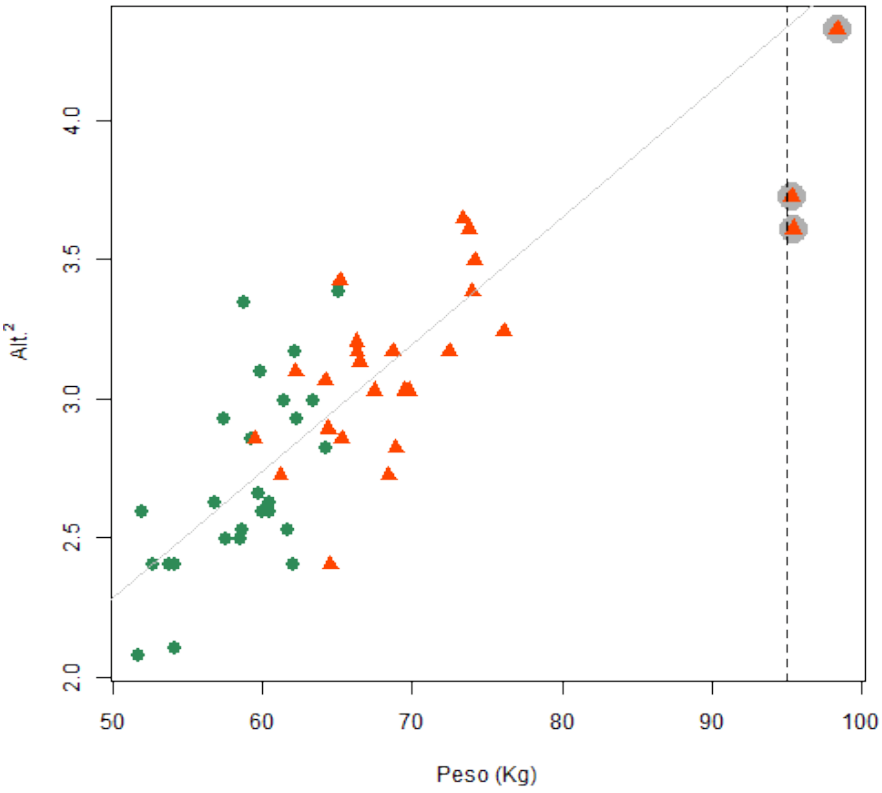

Figure 4: Gráfico de dispersión, ej. 1.e

a que usen el foro y compartan sus códigos a fin de encontrar una solución o posibles mejoras al gráfico en cuestión.

Pistas: para generar el gráfico de ejemplo (fig. 5) se usaron las siguientes funciones: `plot`, `points`, `abline`, `legend`, `text`, `rug`, `format`, `bquote`, `expression`, `summary`, `round`, `paste0` y algunas otras muy comunes.

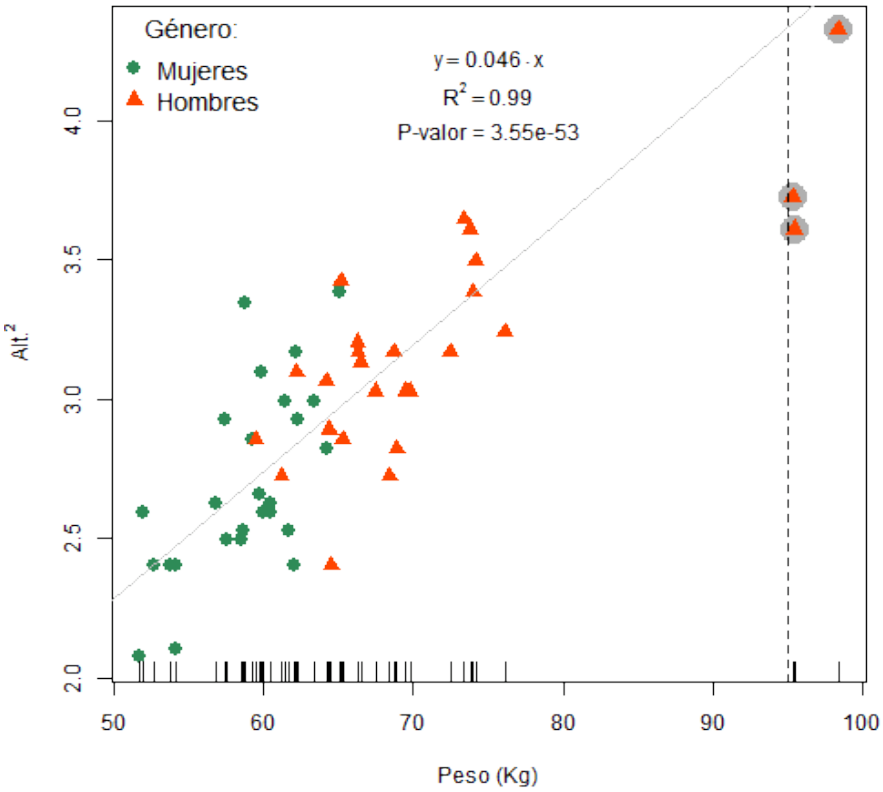

Figure 5: el gráfico 'desafío', ej. 1.f