Actas de las XXII Jenui. Almería, 6-8 de julio 2016
ISBN: 978-84-16642-30-4
Páginas: 253-260

De Menos a Distinto: Estudio de la Implantación de R en las Asignaturas del Grado de Estadística

Jaume Baixeries, Marta Fairén, Joaquim Gabarro, Edelmira Pasarella
Departament de Ciències de la Computació
Universitat Politècnica de Catalunya, Barcelona Tech

Manuela Alcañiz
Departament d'Econometria, Estadística i Economia Espanyola
Universitat de Barcelona

Resumen

Los cursos de informática en las titulaciones que no son informáticas siempre pecan de falta de motivación y buenos resultados de los estudiantes. El argumento del estudiante acostumbra a ser: *¿A mí para que me servirá todo esto?*, y esto se acusa aún más cuando al estudiante le cuestan este tipo de asignaturas.

En este artículo analizamos los primeros cursos de informática en el Grado de Estadística de la *Universitat de Barcelona* (UB) y la *Universitat Politècnica de Catalunya* (UPC), grado conjunto de las dos universidades. En estos cursos se intentó inicialmente reducir la complejidad de los contenidos para promover mejores resultados pero no funcionó como se esperaba, así que cambiamos de táctica y decidimos modificar las herramientas para adaptarlas a aquellas que los estudiantes entienden que están más cerca de sus intereses. En este caso concreto se decidió usar el lenguaje de programación R, un lenguaje ampliamente utilizado por los estadísticos, para explicarles los conceptos básicos de la programación. Así pues, modificamos nuestra estrategia pasando de *menos* (contenidos más simples) a *distinto* (contenidos más elaborados y no triviales adaptados a sus expectativas).

Abstract

Teaching computer science in degrees that are not computer science related presents an important challenge: to motivate the students and to achieve good average grades. The student's complaint is always based on his lack of motivation: *What is this subject useful for?*, and this is specially relevant when this subject is not easy to learn by the student.

In this paper we show the case of computer courses in the Statistics degree taught in the *Universitat de Barcelona* (UB) and the *Universitat Politècnica de Catalunya* (UPC) (two Catalan universities). We initially tried to reduce the complexity of their contents in order to obtain better average grades. Yet, it did not work out as expected. Therefore, we changed our strategy and instead of making the contents easier (less complex), we changed the tools that were used to teach and tried to adapt them to the students' interests. In this particular case, we decided to use the R programming language, a language widely used by statisticians, in order to explain the basics of programming. Therefore, we changed our strategy from *less* (simpler contents) to *different* (more elaborated and nontrivial contents adapted to meet their expectations).

Palabras clave

Grado de Estadística, Programación, R , Motivación.

1. Introducción

Los cursos de Informática en titulaciones de grados que no son explícitamente de ciencias de la computación siempre suponen un problema de falta de motivación hacia estas asignaturas por parte de los estudiantes. Esto es así, por ejemplo, en las diferentes titulaciones de Ingeniería Industrial (Química, Mecánica, Electricidad, ...), y también pasa en el caso del Grado de Estadística.

Una gran parte de los estudiantes del Grado de Estadística son estudiantes que provienen del bachillerato social, lo que los hace aún más reticentes a aprender informática, que ellos piensan que no es sujeto de lo que quieren aprender.

En el plan de estudios de este Grado de Estadística la informática se enseña en dos asignaturas que se imparten en los dos primeros semestres de la carrera, incluyendo el contenido habitual:

- *Semestre 1*: Variables, estructuras de control, recorrido y búsqueda, tablas y funciones.
- *Semestre 2*: Estructuras de datos más complejas.

Una primera implantación de estos cursos se hizo usando C++ como lenguaje de programación, como es habitual en cursos de programación en ciencias de la computación. Ante la gran falta de motivación y de buenos resultados del primer año, se intentó reducir el nivel de dificultad, adaptando los contenidos para enseñar únicamente *aquello que era básico*, pero los resultados no mejoraron (se puede ver en el apartado de resultados de los años previos) y la motivación tampoco. La separación profunda entre las expectativas de los estudiantes y el contenido de los cursos se ejemplifica magníficamente en la frase *¿Y a mí para qué me sirve todo esto? ¿Por qué me lo están contando?*

Era evidente que la falta de motivación no venía producida por la complejidad de los contenidos, sino por la motivación de ellos, así que se pensó en modificar la forma en que se enseñaban estos contenidos. Entrando en detalle en la utilidad concreta que tienen estos cursos para un graduado en Estadística, se pensó en modificar las herramientas y utilizar R como lenguaje de programación. Con este cambio se consiguió que los estudiantes viesen como importantes los contenidos de estas asignaturas, ya que el lenguaje R es una de las herramientas más utilizadas entre los estadísticos. También se adaptaron los ejercicios para que tuviesen mayor contenido estadístico.

Esta nueva aproximación (Sección 2) nos ha permitido pasar de *menos* (contenidos más simples) a *distinto* (contenidos más elaborados y no triviales adaptados a sus espectativas). En la Sección 3 entramos en detalle en los objetivos y contenidos de los dos cursos y en la (Sección 4) en la forma en que se imparten. En la Sección 5 discutimos las ventajas e inconvenientes que supone usar este tipo de lenguajes para explicar conceptos básicos de programación. En la Sección 6 vemos cuáles han sido los resultados objetivos que se han obtenido en los dos últimos años en los que se ha aplicado el cambio. Finalmente, en la Sección 7 discutimos las conclusiones y futuros desarrollos.

2. De menos a distinto

La aproximación (o paradigma) de *menos a distinto* aparece tras una reflexión sobre los aspectos esenciales de la enseñanza básica de la Informática para no informáticos. Una aproximación razonable en el diseño de cursos de Informática para no informáticos consiste en "adaptar" el contenido de cursos existentes para informáticos. El problema reside en cḿo tiene lugar la adaptación. Esquematizamos dos modos de proceder.

- *Simplificar Contenido*. En este caso, habitualmente se eliminan los temas más difíciles. De los restantes temas se eliminan las partes (y ejercicios) más complicados y se aumenta la cantidad de ejemplos y ejercicios sencillos. Esto suele dar lugar a un temario "un tanto descafeinado" que quizá no acabe gustando a nadie, ni a docentes ni a estudiantes.
- *De Menos a Distinto*. Se toma el temario original, se mira "lo que es esencial" y se simplifica al máximo lo que no lo es. Por ejemplo, para un informático, que se deban declarar variables y darles un tipo es importante, pero quizá esto no lo es para los no informáticos. Lo que sí es fundamental en ambos casos es la noción de variable y las ideas más básicas de la algorítmica como son la asignación, la descomposición secuencial, el análisis por casos y las iteraciones.

Veamos cómo la aproximación "de menos a distinto" genera situaciones nuevas. La docencia en *Introducción a la Informática* y *Programación* de primer año del Grado de Estadística ha generado problemas de comunicación y comprensión debido a la heterogeneidad de lenguaje entre profesores y alumnos.

Por una parte los docentes provienen de una Universidad Politécnica, la UPC, y por lo tanto están acostumbrados a impartir cursos en los que se supone que los estudiantes tienen una formación matemática aceptable. Se asume una cierta familiaridad con sumatorios, vectores, matrices, subíndices o álgebra de Boole. Por lo tanto, el lenguaje de los profesores, aparece plagado de igualdades matemáticas o expresiones booleanas que muchas veces utilizan sin ser plenamente conscientes de ello. Podríamos decir que es el lenguaje de la "gente de ciencias". Por otra parte, como el curso se imparte en la Facultad de Economía y Empresa de la UB la mayoría de estudiantes provienen del bachillerato social. Dichos estudiantes están mucho menos familiarizados con el lenguaje matemático ya que son "gente de letras". Esta fractura ciencias/letras lleva a veces a que simplemente los estudiantes no sepan de qué están hablando sus profesores. Dualmente los profesores a veces, se quedan literalmente sin habla y piensan *¿Cómo cuento yo esto ahora?*.

Veamos qué dificultades pueden aparecer. Supongamos que se pide a los estudiantes de desarrollen una función para calcular el coeficiente de Gini [4]. Dado que dicho coeficiente se utiliza en ciencias sociales para medir el grado de desigualdad, es un problema en el que pueden estar interesados. Dado $x = (x_1, \ldots, x_n)$, con $n > 1$, el coeficiente de Gini definido por

$$\mathsf{gini}(x) = \frac{1}{\mu \times n \times (n-1)} \sum_i \sum_{j>i} \mid x_i - x_j \mid$$

con $\mu = (\sum_i x_i)/n$. Para los profesores de "ciencias" el siguiente código es claro y necesita de pocas explicaciones.

```
gini <- function(x){
   n <- length(x)
   mu <- mean(x)
   sum <- 0
   for(i in 1:(n-1)){
      for(j in (i+1):n){
         sum <- sum + abs(x[i]-x[j])
   } }
   gini <- sum / (mu * n *(n-1))
   return(gini)
}
```

Sin embargo esto deja de ser cierto para la mayoría de los estudiantes del grado de estadística. Para explicar el recorrido de los `for` es aconsejable proceder en varios pasos. Primero, tomar $n = 5$ (valor ni demasiado pequeño ni demasiado grande) ver los pares (x_i, x_j) como elementos de una matriz $n \times n$. Segundo ver que los índices de los `for` recorren la parte de la matriz por encima de la diagonal.

3. Objetivo y contenidos de los cursos

Cada una de las dos asignaturas de informática del Grado de Estadística (*Introducción a la Informática* y *Programación*) tienen una carga lectiva de 6 créditos ECTS y, tal y como se describe en la guía docente [1] [2], se estima que el tiempo total de dedicación requerido es de 150 horas por semestre.

El *objetivo principal* de estas dos asignaturas es que los estudiantes aprendan las técnicas básicas del diseño e implementación de algoritmos para la resolución de problemas. Este objetivo es el mismo en cualquier otra titulación en la que se den contenidos de programación (incluso en el Grado de Informática). La diferencia pues entre las diferentes titulaciones estriba habitualmente en la dificultad de los contenidos, es decir, las titulaciones que no son de informática acostumbran a reducir esta dificultad respecto a las que sí son de informática, pero utilizan las mismas herramientas o muy parecidas.

En este caso decidimos cambiar de estrategia y basar la diferencia no tanto en la reducción de la dificultad de los contenidos, sino en las herramientas usadas para enseñar la programación, y en este caso concreto en el uso del lenguaje R como lenguaje de programación. El lenguaje R es un lenguaje ampliamente usado y conocido entre los estadísticos. De esta manera se pretendía motivar más a los estudiantes en estas asignaturas que no son directamente propias de su carrera y de paso se les introducía ya desde un principio a un lenguaje que más adelante en su carrera volverán a necesitar.

La primera de las asignaturas, que se imparte en el primer semestre (*Introducción a la Informática*), tiene contenidos prácticamente idénticos a los que tiene cualquier primer semestre de programación.

1. **Introducción a la algorítmica**
   - 1.1 Nociones elementales: arquitectura básica de un ordenador, entorno, variables, estado.
   - 1.2 Estructuras de un lenguaje de programación imperativo: asignación, alternativa e iterativa.
   - 1.3 Tipos de datos simples.
2. **Estructuras de datos no simples (vectores)**
   - 2.1 Conjunto de datos de un mismo tipo.
   - 2.2 Algoritmos de tratamiento secuencial.
   - 3.3 Esquemas básicos de recorrido y búsqueda.
3. **Diseño descendente**
   - 3.1 Funciones y parámetros.
   - 3.2 Diseño de funciones y uso de ellas desde un programa u otra función.

En el caso de la segunda asignatura (*Programación*), que se imparte en el segundo semestre, los contenidos ya no son tan estándares, porque explicar las estructuras de almacenamiento de datos en un programa depende fuertemente del lenguaje de programación que se utilice, y por tanto en este caso la asignatura está directamente basada en las estructuras de datos que el lenguaje R define y utiliza: Matrices, *Lists* y *Data Frames*.

En abstracto se siguen enseñando conceptos muy similares: se introducen las matrices (estructuras de dos dimensiones), se introducen estructuras que nos permiten mezclar en un mismo registro datos que no son del mismo tipo (en C++ y C serían los *structs* y en R se pueden usar los *List*), y finalmente se pueden describir estructuras que combinan otras estructuras (en el caso de R nos encontramos con los *Data Frames* que nos permiten este nivel de complejidad).

En concreto, por contra, se han adaptado los conceptos a las particularidades de las estructuras de datos del lenguaje. Así usamos los *List* como diccionarios y los *Data Frames* como conjunto de datos extraído de ficheros de hoja de cálculo, por ejemplo. Los contenidos son los siguientes:

[1] Web de la asignatura de Introducción a la Informática: `http://www.ub.edu/grad/plae/AccesInformePD?curs=2015&codiGiga=361180&idioma=CAT&recurs=publicacio`

[2] Web de la asignatura de Programación: `http://www.ub.edu/grad/plae/AccesInformePD?curs=2015&codiGiga=361192&idioma=CAT&recurs=publicacio`

1. **Estructura matricial: Matrices**
   1.1 Concepto de matriz. Filas y columnas.
   1.2 Acceso a los datos: un único dato, una fila, una columna.
   1.3 Esquemas de recorrido y búsqueda en matrices.
2. **Combinación de datos de diferentes tipos: *Lists***
   2.1 Introducción del tipo *List*
   2.2 Construcción y operaciones con *Lists*
3. **Estructuras más complejas: *Data Frames***
   3.1 Introducción y conceptos básicos.
   3.2 Construcción y operaciones con *Data Frames*
   3.3 Combinación de diferentes estructuras.

Cabe destacar algunas particularidades del lenguaje que también ayudan a que los estudiantes no se estanquen tanto en el diseño de los algoritmos, como por ejemplo que el lenguaje no está tipado (no hay que declarar las variables y cada variable asume el tipo del valor que se le asigna en cada momento).

Por otro lado, el lenguaje R también ofrece muchas funcionalidades ya implementadas, y en este caso la mayoría de ellas no se las permitimos utilizar directamente en estos cursos. Esto, en parte, es un inconveniente porque da lugar a algunas discusiones con los estudiantes que describimos con algunos ejemplos concretos en la Sección 5.

4. Organización y metodología

Tanto en *Introducción a la Informática* como en *Programación*, el número de estudiantes inscritos oscila entre 80 y 120 repartidos en 2 grupos de teoría y tres grupos de laboratorio. Esto implica que, como recurso físico, se requieren dos aulas docentes para las clases de teoría y tres aulas de informática para los laboratorios.

- Las *clases de teoría* son semanales y se caracterizan por ser clases dinámicas, donde se intercalan la presentación de conceptos, técnicas y herramientas con el análisis, y el desarrollo de problemas prácticos, contando con la participación de los estudiantes.
- Durante las *clases de laboratorio*, los estudiantes deben trabajar individualmente realizando un número preestablecido de problemas que requieran utilizar los conceptos estudiados en la última clase de teoría que hayan tenido o simplemente implementando problemas que ya han hecho. Durante las horas de laboratorio, los profesores o profesoras inician la clase con un repaso del último tema visto en teoría, orientan sobre los aspectos clave de los ejercicios más difíciles, hacen seguimiento del trabajo individual de los estudiantes y, en caso de detectar dudas generales, hacen aclaraciones a todo el grupo.

En relación a los recursos bibliográficos, se sugiere a los estudiantes que utilicen como referencia los libros [3], disponible de manera gratuita en formato electrónico, y [5]. Adicionalmente, los estudiantes disponen de apuntes y listas de problemas adaptados a los contenidos, elaborados por los profesores y profesoras de la asignatura. Estos apuntes y listas de problemas se han ido actualizando en cada curso.

5. Sobre ejercicios y motivación

Desde el punto de vista del estudiante, la limitación que supone la utilización de un conjunto muy reducido de instrucciones del lenguaje R puede representar no solamente una situación paradójica, sino contradictoria, ya que muchos de los ejercicios que se les propone se podrían resolver utilizando instrucciones más sofisticadas del mismo lenguaje. A modo de ejemplo, consideremos el siguiente diálogo socrático e imaginario entre estudiante y profesor.[4] Usamos, para ello un tono ligero y distendido en un intento de situar a profesor y alumno en su día a día. De este modo pretendemos transformar planteamientos (más o menos) abstractos en el sencillo quehacer cotidiano y en la lucha constante del profesor por la credibilidad (docente) ante sus estudiantes. Podemos suponer que, dicho diálogo, aparece después de que el estudiante ha programado el producto de matrices con el triple `for` habitual. Tras esto, el profesor le dice que en R el operador `%*%` multiplica matrices directamente:

> *Estudiante*: ¿Por qué me pide que haga esto? Es decir, ¿Por qué me pide que diseñe un programa con muchas iteraciones si R da la solución?. ¡Si R me lo da hecho es para que pueda utilizarlo y no perder el tiempo precisamente haciéndolo yo! La verdad, no le veo la gracia a la solución que usted propone (con el triple `for`). Piense que soy un estudiante de Estadística y no un informático profesional.

Una posible respuesta es:

[3] Introducción a R: `http://www-eio.upc.es/teaching/best/R/TutorialR.pdf`

[4] Sólo es imaginario en el sentido de que no se verbaliza. Los estudiantes tienen múltiples maneras de hacer notar al docente que lo que les está contando "ni les va ni les viene". A uno de los autores, con más de 30 años de experiencia, una vez una estudiante le preguntó de viva voz: "pero a ver, ¿Para qué me está contando todo esto?".

> *Profesor*: Porque de este modo has aprendido a manipular variables (y estructuras de datos) de distinta granularidad. Una cosa es manipular una matriz `A` globalmente, es decir *como un todo* y otra muy distinta poder tratar sus elementos `A[i,j]` a *nivel individual*. Conocer estas técnicas te permitirá, en caso de ser necesario, llegar a diseñar programas que tratan con los distintos ítems de información de un modo mucho más individualizado. Así aprendes a tratar con estructuras de información a distintos niveles de granularidad.

A lo que el estudiante podría responder:

> *Estudiante*: Entonces, ¿Por qué no me da ejercicios en que esto es realmente necesario?

A esta pregunta la única respuesta decente es:

> *Profesor*: Mira el ejercicio ....

Se puede responder a la pregunta del estudiante de modo alternativo:

> *Profesor*: Porque en otros lenguajes de programación, como `Java` o `C++` es la única solución posible. Es muy probable que en algún momento de tu carrera te encuentres con ellos, así podrás entender el código. En este curso aprendes a programar, por lo tanto te cuento las técnicas generales. No es un curso de introducción a R .

Claro que esta respuesta es un tanto peligrosa, ya que el estudiante podría replicar:

> *Estudiante*: ¡Pero si a mí lo que me interesa es R en concreto! Si quisiera aprender a programar en general ya estaría estudiando informática.

Podríamos seguir el diálogo. En el mismo estilo informal, el lector podría preguntar:

> *Lector*: ¿Qué conclusión se supone que he de sacar?.

Pues que, en la enseñanza de la programación, no hay que dar nunca nada por sentado. Hay que escuchar a los estudiantes y aceptar discutir con ellos (si hace falta). Dificilmente podremos enseñarles algo duradero si no creen que aquello tenga valor. El profesor siempre puede preguntar. ¿Creen que ha valido la pena venir hoy a esta clase?. Si la respuesta son medias sonrisas, soplidos y silencio (o fuerte absentismo) algo va mal. De modo más formal, queremos enfatizar que el contenido del curso depende fuertemente de los estudiantes a los que va dirigido y que hay que hacer un esfuerzo constante de adaptación, que no es nada fácil. Esto plantea un reto claro a la hora de buscar ejemplos interesantes y motivadores. Veamos que esto dista mucho de ser claro como podemos ver en los dos casos siguientes.

5.1. Caso de estudio: producto de matrices

Tratamos el ejercicio de diseñar una función para calcular el producto de dos matrices `A` y `B` (suponiendo que las dimensiones son las correctas)[5]. En R hay distintas soluciones posibles. Utilizamos la instrucción

```
C <- matrix(nrow=nrow(A),ncol=ncol(B))
```

para crear una matriz de las dimensiones necesarias. Empecemos con la solución que traduce a R la aproximación proveniente de `Java` o `C++`. En dichos lenguajes, las variables son los elementos de las matrices y se implementa la fórmula

$$\mathtt{C}[\mathtt{i},\mathtt{j}] = \sum_{\mathtt{k}} \mathtt{A}[\mathtt{i},\mathtt{k}] * \mathtt{B}[\mathtt{k},\mathtt{j}]$$

en que, para cada `C[i,j]` el sumatorio se ejecuta vía la siguiente iteración.

```
for (k in 1:nrow(B)){
   C[i,j] <- C[i,j]+A[i,k]*B[k,j]
}
```

Esta aproximación da lugar al conocido programa del triple `for`:

```
mult_matr <- function (A,B){
   C <- matrix(...)
   for (i in 1:nrow(A)){
      for (j in 1:ncol(B)){
         for (k in 1:nrow(B)){
            C[i,j] <- C[i,j]+A[i,k]*B[k,j]
   }}}
   return(C)
}
```

Consideremos otra solución que utiliza ”vectorización”. R permite trabajar directamente con filas o colummas como variables. La fórmula del sumatorio se traduce directamente como

```
C[i,j] <- A[i, ]%*%B[ ,j]
```

y el programa completo es:

[5] Dicho ejercicio tiene amplia historia. Por ejemplo, ya aparece como Ejercicio 11 del Capítulo 5, *Proposiciones de Iteración (Do)* de [6]

```
mult_matr <- function (A,B){
   C <- matrix(...)
   for (i in 1:nrow(A)){
      for (j in 1:ncol(B)){
         C[i,j] <- A[i, ]%*%B[ ,j]
   }}
   return(C)
}
```

Finalmente no hay que olvidar que R ya nos da programado directamente el producto de matrices con lo que obtenemos:

```
mult_matr<- function (A,B){
   C <- matrix(...)
   C <- A%*%B
   return(C)
}
```

5.2. Data frames

Una estructura fundamental en R son los *Data Frames*. Consideremos el siguiente ejercicio:

> *Ejercicio*: El *Data Frame* `trees` contiene datos sobre la altura, el diámetro i el volumen de 31 cerezos talados. Tomando como entrada `trees`,
>
> ```
> Girth Height Volume
> 1    8.3    70    10.3
> 2    8.6    65    10.3
> 3    8.8    63    10.2
>  ...
> 30  18.0   80    51.0
> 31  20.6   87    77.0
> ```
>
> Diseña una función que retorne el número de árboles que tienen altura (Height) más grande que 85 y diámetro (Girth) más grande que 10.

Como el número de tuplas en el *Data Frame* está dado por `nrow(trees)`, podemos hacer un recorrido por filas, lo que nos da el programa:

```
seleccion <- function(trees){
  total <- 0
  for (i in 1:nrow(trees)){
    if(trees$Height[i] > 85
       & trees$Girth[i] > 10){
            total <- total + 1
   }}
  return (total)
}
```

Este programa es importante porque muestra como acceder a los atributos que nos interesan de cada fila mediante `trees$Height[i]` y `trees$Girth[i]`. Sin embargo es posible obtener la siguiente solución mucho más compacta en la que el recorrido no aparece explícitamente:

```
seleccion <- function(trees){
  total <-
    nrow(
      subset(trees,Height>85&Girth>10)
    )
  return (total)
}
```

6. Valoración de resultados

Esta sección ofrece un análisis cuantitativo de la implantación de la nueva metodología en las dos asignaturas de informática descritas en este artículo. Para ello, ofrecemos la evolución mediante tres parámetros de calidad:

1. *Tasa de éxito*: total de aprobados / (total de matriculados - no presentados).
2. *Tasa de rendimiento*: total de aprobados / total de matriculados.
3. *Tasa de presentados*: estudiantes presentados / total de matriculados.

Hemos calculado la media de las tasas de éxito de las dos asignaturas de informática (*Introducción a la Informática* y *Programación*) y la hemos comparado con la media del resto de asignaturas del primer año. El motivo de comparar únicamente con las asignaturas del primer año es porque existe una diferencia muy significativa entre las tasas de rendimiento de los estudiantes en el primer año (el más crítico) y en los posteriores [3].

Para poder estudiar los resultados que se observan en las gráficas, también debemos resaltar que la implantación del lenguaje de programación R en las dos asignaturas se produjo a partir del curso 2013-14 (valor 2013 en las gráficas), por lo que los valores de las tasas correspondientes a este período se observan en los *dos últimos datos* de cada gráfica. Además, en cada gráfica hemos acompañado las dos líneas a comparar (media de las asignaturas de informática y media del resto de asignaturas de primer curso) con una tercera línea que muestra la diferencia entre las dos y, finalmente, una recta de ajuste de esta diferencia, a fin de ver la tendencia de la diferencia entre las asignaturas de Informática y el resto de asignaturas.

En la Figura 1 se representan los resultados de la tasa de éxito, en la Figura 2 se representan los resultados de la tasa de rendimiento, y en la Figura 3 se representan los resultados de la tasa de presentados. Se puede observar en las tres gráficas que la diferencia entre la tendencia de las asignaturas de informática y el resto de asignaturas de primer curso decrece de manera ostensible a partir de los dos últimos años. Este comportamiento queda corroborado por *las rectas de ajuste* que se incluyen en las gráficas, que tienen, en todos los

casos, una pendiente de signo negativo y estrictamente diferente de cero. Su *significación estadística* se ha comprobado a través del contraste de significación individual sobre la pendiente, que tiene en los tres casos un p-valor inferior a $0{,}05$. Por tanto, podemos afirmar que hay una convergencia entre las tasas de las asignaturas de programación y el resto de asignaturas. Concretamente, las asignaturas de programación han equiparado sus indicadores a los del resto de asignaturas, dejando de ser materias con menos aprobados y menos presentados que el resto.

Así pues, podemos afirmar que la implantación del lenguaje R ha tenido consecuencias beneficiosas en cuanto a los resultados en el rendimiento, teniendo en cuenta que el temario subyacente de las asignaturas ha permanecido básicamente igual.

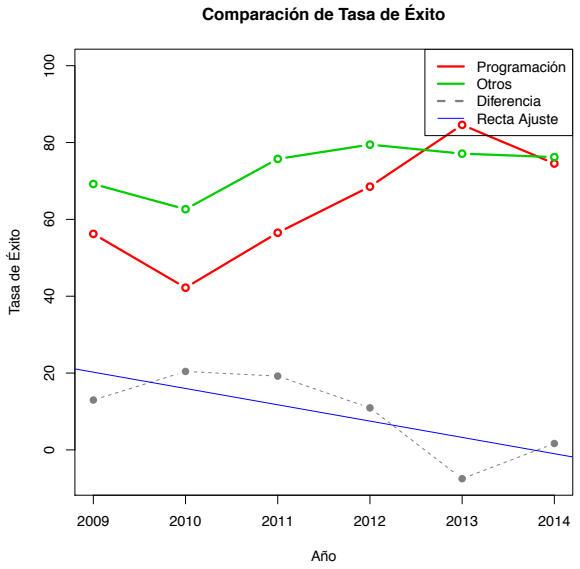

Figura 1: Evolución de la tasa de éxito: $\frac{aprobados}{presentados}$.

7. Conclusiones y futuros desarrollos

A la pregunta sobre qué hay que enseñar respondemos: hay que enseñarlo un poco todo. Una aproximación tentativa consiste en enseñar a programar con las técnicas desarrolladas para `C++` o `Java`. Sin embargo tampoco hay que olvidar que R permite en muchos casos obtener programas mucho más compactos. Por lo tanto hay que llegar a algún tipo de equilibrio. Hay un *peligro evidente* en enseñar a programar basándose en funciones de alto nivel como `A %* %B` . El peligro consiste en que el estudiante te pregunte cuál es la función que resuelve el problema y evite pensar en soluciones

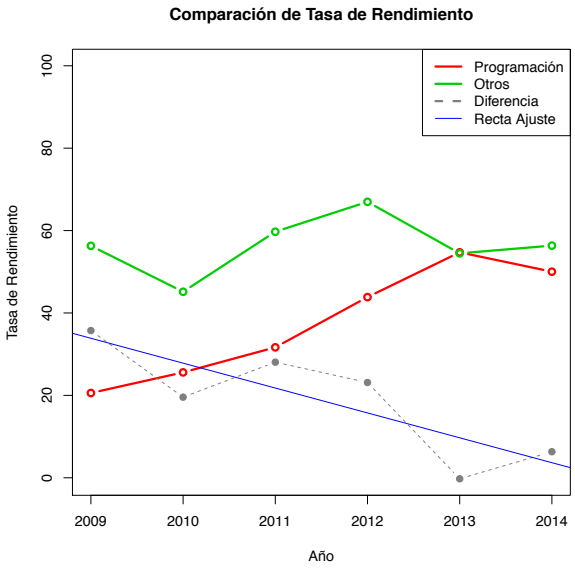

Figura 2: Evolución de la tasa de rendimiento: $\frac{aprobados}{matriculados}$.

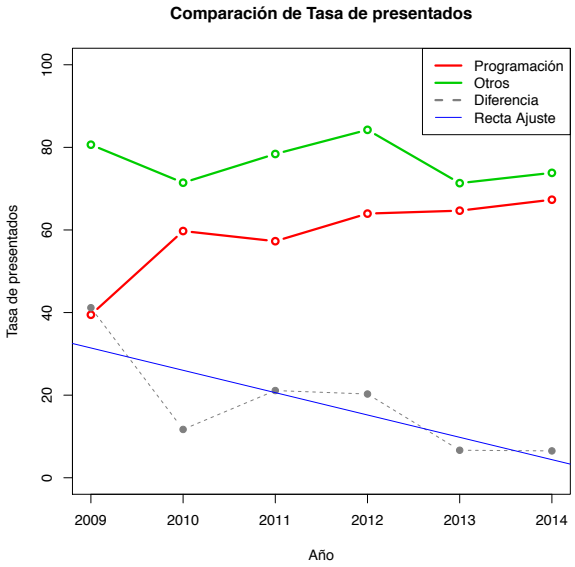

Figura 3: Evolución de la tasa de presentados: $\frac{presentados}{matriculados}$.

algorítmicas.

Muchas de estas cuestiones han aparecido debido al carácter interuniversitario de dicho grado. Esto no quiere decir que valoremos negativamente dicho carácter sino más bien todo lo contrario. Son precisamente las dudas y problemas que han aparecido lo que ha permitido a los profesores discutir, repensar, adaptar, descartar e incluir nuevos tópicos en los cursos. Es decir, en una palabra, ha permitido a los profeso-

res recuperar plenamente su función primigenia como *docentes*. Los esfuerzos han tenido cierto éxito como podemos ver en la evolución de los últimos años en las Figuras 1, 2 y 3.

Dada la distinta tipología de los estudiantes del Grado de Estadística en relación a los estudiantes de Informática, es fundamental publicar apuntes de las asignaturas de informática especialmente pensados para ellos. Actualmente, parte de los docentes involucrados, están redactando apuntes. Dentro de poco, la Facultad de Economía y Empresa publicará una colección de problemas de la asignatura *Introducción a la Informática* dentro de la colección de *Textos Docentes*[2]. En relación a *Programación*, los estudiantes disponen actualmente de una primera versión de los apuntes correspondientes a la asignatura. La publicación de este material docente permitirá dar mayor difusión al esfuerzo y resultados obtenidos. Sin duda, otras titulaciones de Grado de Estadística, tendrán puntos de vista alternativos. Hay que esperar que en un tiempo relativamente corto se llegue a un consenso.

Si bien los estudiantes pueden ejecutar los programas en R en sus ordenadores personales cosa que les proporciona gran autonomía, también sería interesante ofrecer entornos de programación más completos. En particular, sería interesante incorporar en ambos cursos el uso de herramientas de corrección automática [7]. Esto permitiría ofrecer un entorno interactivo de corrección, flexibilizar el trabajo de prácticas y desacoplarlo, al menos en parte, de las clases de laboratorio. Además vía ficheros de `logs` sería posible intentar un análisis de los distintos ejercicios vía *data mining*.

La enseñanza de la programación es algo demasiado serio para no dedicarle todo el tiempo y esfuerzo necesario. Precisamente, en este contexto sería muy interesante ver la evolución a lo largo del tiempo de la ratio coste-beneficio [1].

8. Agradecimientos

Los autores agradecen muy especialmente a Natàlia Pallarès y a Sara Fernández su trabajo en la redacción de la lista de problemas y los apuntes de las asignaturas. También agradecemos a los revisores sus cuidados y lúcidos comentarios.

Varios autores han recibido ayudas del *Ministerio de Economía y Competitividad* (MINECO) y de la *Agència de Gestió d'Ajuts Universitaris i de Recerca* (AGAUR) de la Generalitat de Catalunya. En detalle, Jaume Baixeries está parcialmente financiado por el proyecto SGR2014-890 (MACDA) de la AGAUR y el proyecto MINECO APCOM (TIN2014-57226-P). Marta Fairén está parcialmente financiada por el proyecto TIN2014-52211-C2-1-R co-financiado por el MINECO y fondos FEDER. Joaquim Gabarro y Edelmira Pasarella están parcialmente financiados por el MINECO y los *Fondos Feder* de la Comunidad Europea, referencia TIN2013-46181-C2-1-R (COMMAS), y por el proyecto SGR 2014:1034 (ALBCOM) de la AGAUR. Manuela Alcañiz está parcialmente financiada por el proyecto 2014 SGR1016 (AGAUR) y el proyecto MINECO ECO2015-66314-R.

Referencias

[1] Maria J. Blesa, Amalia Duch, Joaquim Gabarró, Jordi Petit, and Maria J. Serna. Continuous assessment in the evolution of a CS1 course: The pass rate/workload ratio. In *Computer Supported Education - 7th International Conference, CSEDU 2015, Revised Selected Papers*, volume 583 of *Communications in Computer and Information Science*, pages 313–332. Springer, 2015.

[2] Jaume Baixeries (coord), Natàlia Pallarès, and Enrique Romero. *Introducció a la Informàtica. Exercicis*. Col·lecció de Textos Docents (en prensa), 2016.

[3] Teresa Bartual Figueras and M. Cristina Poblet Farrés. Determinantes del rendimiento académico en estudiantes universitarios de primer año de economía. In *Revista de Formación e Innovación Educativa Universitaria*, pages 172 – 181. Vol. 2, N. 3, 2009.

[4] C. Gini. *Variabilità e Mutuabilità. Contributo allo Studio delle Distribuzioni e delle Relazioni Statistiche*. C. Cuppini, 1912.

[5] Norman Matloff. *The art of R Programming*. No Starch Press, 2011.

[6] D. McCracken. *Programación Fortran IV*. Limusa-Wiley, 1967.

[7] Jordi Petit, Omer Giménez, and Salvador Roura. Jutge.org: an educational programming judge. In *Proceedings of the 43rd ACM technical symposium on Computer science education, SIGCSE 2012*, pages 445–450, 2012.