ARTÍCULOS DE INVESTIGACIÓN DOI: 10.24844/EM3001.05

# Aprendizaje de las matemáticas a través del lenguaje de programación R en Educación Secundaria

## Learning Mathematics through the R Programming Language in Secondary Education

Álvaro Briz Redón
Ángel Serrano Aroca[1]

**Resumen.** El aprendizaje de la programación por medio de los ordenadores constituye una gran ventaja a nivel de competencias en la época actual. Además, en un sentido estrictamente educacional, la programación puede dotar a los alumnos que la estudian y practican de una mayor capacidad de razonamiento lógico, pensamiento estructurado o incluso una mayor imaginación. Así pues, el primer objetivo de este trabajo es revisar algunos estudios que señalan las múltiples ventajas que puede suponer para el alumnado el aprendizaje de la programación durante su educación secundaria.

El segundo y principal objetivo es plantear el uso de uno de los lenguajes de programación más populares del momento, el R, como una herramienta para tratar contenidos propios de la asignatura de Matemáticas. Esto es especialmente interesante debido a la presencia de contenidos en el currículo que se prestan claramente al diseño de algoritmos y a una experimentación mayor que la que permite la enseñanza tradicional.

Esta metodología de aprendizaje fue puesta en práctica con 33 alumnos españoles de entre 14 y 15 años de edad, los cuales utilizaron el lenguaje R para

**Fecha de recepción:** 24 de mayo de 2017. **Fecha de aceptación:** 25 de noviembre de 2017

[1] Departamento de Ciencias Aplicadas y Tecnológicas, Facultad de Veterinaria y Ciencias Experimentales, Universidad Católica de Valencia San Vicente Mártir, Valencia, España, *angel.serrano@ucv.es*

tratar cuestiones relativas a la resolución de ecuaciones polinómicas. La experiencia permitió comprobar grandes ventajas de la metodología, aunque también algunas desventajas para ciertos alumnos, debido a la complejidad intrínseca de la programación, como se desprendió del análisis correlacional de la encuesta realizada a los mismos. En cualquier caso, estas desventajas podrían subsanarse mediante una aplicación de la metodología más prolongada en el tiempo.

**Palabras clave:** *educación matemática; pensamiento computacional; educación secundaria; lenguaje de programación.*

**Abstract.** Learning to program using computers constitutes a great advantage for students in order to gain competencies nowadays. Furthermore, programming can help them to develop skills such as logical reasoning, structured and/or even creative thinking from an educational point of view. Therefore, the first objective of this work is to review several studies that discuss the multiple benefits that students can obtain from learning to program in secondary education.

The second and main purpose of this study seeks to show the possibility of learning mathematical concepts with the aid of one of the most popular programming languages currently available, R. This is especially interesting because some mathematical contents in the curriculum are easy to teach via algorithmic design and experimentation. Besides, these contents are unable to be taught in this way when using traditional teaching methods.

This learning methodology was tested with 33 students from Spain aged 14-15 years, which used the R programming language to study polynomial equations. The experience provided sufficient information for observing great advantages and yet some disadvantages for certain students due to the intrinsic complexity of programming, as it was revealed through the correlational analysis of the survey that was taken by the participants. In any case, these disadvantages would likely be solved by a longer implementation of the methodology.

**Keywords:** *mathematics education; computational thinking; secondary education; programming language.*

## 1. INTRODUCCIÓN

La programación según McCracken (1957) puede definirse como la rama dentro del campo de la Informática que consiste básicamente en traducir el lenguaje humano al lenguaje que comprende un ordenador. Otros autores, como Booth (1958), dotan de una clara componente matemática a este proceso, ya que primero se formula un problema, luego se propone una solución que pueda resolverse con el apoyo de la tecnología digital y posteriormente se realiza su traducción al lenguaje de la máquina para que lo resuelva. De este modo, se puede definir un programa como el conjunto de órdenes ejecutadas en lenguaje del ordenador que permiten que una máquina realice una serie de operaciones de forma automática (Wilkes, 1956). Algunos de estos autores afirman que la programación a cierto nivel constituye una actividad de pensamiento matemático. Sin embargo, es importante tener en cuenta tanto las ventajas cognitivas que puede suponer la práctica de la programación como las posibles dificultades que puede acarrear su enseñanza y aprendizaje (Saeli *et al.,* 2011). Soloway (1993) argumenta que el aprendizaje de la programación proporciona poderosas estrategias de pensamiento, diseño y resolución de problemas mediante varias fases: primero se obtiene la solución del problema, pero luego dicha solución debe replantearse de forma alternativa y precisa para que pueda ser trasladada al lenguaje que comprende el ordenador (Hromkovivc, 2006; Szlavi, 2006). Además, la programación es capaz de proporcionar ciertas oportunidades que pocas actividades mentales están en disposición de hacerlo, ya que según Papert (1980) programar no solo permite al alumno comunicarse con el ordenador, sino también explorar las dinámicas que rigen sus propios pensamientos, asentar el razonamiento lógico a través de la sistematización y aumentar su capacidad de autocrítica por medio de la corrección de sus errores.

Por otra parte, existen pocas discrepancias sobre la declaración de la programación como una actividad compleja (Govender, 2009), por lo cual es lógico que resulte difícil de asimilar para un alumno recién iniciado. En esta línea Du Boulay (1986) distingue cinco tipos de dificultades que tienden a solaparse continuamente en la práctica: falta de orientación en el alumno sobre la utilidad de la programación, desconocimiento de la máquina en la que se programa, dificultad de adaptación al lenguaje, problemas de comprensión y ausencia de método a la hora de programar. Atendiendo a todas estas dificultades, resulta clave la elección del lenguaje de programación y del tipo de estrategia de enseñanza a llevar cabo. Así, la simplificación y división de los problemas en

subproblemas más sencillos, la definición y explicación de un modelo basado en el entorno máquina-lenguaje donde se está trabajando (Du Boulay, 1986), la ligera modificación de conjuntos de instrucciones para resolver problemas parecidos (Linn y Dalbey, 1989), o el planteamiento constante de problemas aplicados constituyen algunos ejemplos metodológicos de enseñanza de la programación que se pueden considerar exitosos.

A principios de los años 70 se concluye en un estudio que el uso del lenguaje Logo puede potenciar la capacidad lectora de algunos alumnos, además de aumentar el interés por aprender y el nivel de autoconfianza en muchos de ellos (Feurzeig y Lukas, 1972). Por este motivo Papert (1972) defendió el aprendizaje de las matemáticas de un modo activo y práctico, en contra de lo que proponía la enseñanza tradicional. Según este autor, conocer conceptos sin saber de qué manera se construyen es insuficiente, motivo por el cual también propuso el aprendizaje mediante la ayuda de la programación. Diversos estudios realizados con el lenguaje Logo han demostrado que la programación puede facilitar la conceptualización de variables matemáticas como la algebraica (Noss, 1986), contribuye muy significativamente al desarrollo de la competencia en la resolución de problemas (Battista y Clements, 1987) y afecta positivamente ciertas áreas del funcionamiento y logro cognitivo (Clements, 1987).

Otros autores, como Du Boulay (1978), pusieron a prueba la enseñanza de las matemáticas mediante la programación con profesores de Primaria en formación que mostraban dificultades y hasta un cierto desagrado hacia las mismas. De este modo, según Howe *et al.* (1982), la razón principal de enseñar matemáticas mediante la programación se debe a que el alumno necesita replantearse los conceptos matemáticos para poder traducirlos a un lenguaje que el ordenador comprenda, pudiendo favorecer la adquisición de estrategias en la resolución de problemas. Además, el conocimiento ganado en programación puede también ser utilizado para discutir conceptos y problemas clásicos de matemáticas (Feurzeig *et al.*, 2011). Otros lenguajes que se están incorporando con fuerza durante los últimos años son el Scratch (Resnick, 2009), R (Team, 2000) o Python (Van Rossum y Drake, 2003), ya que los lenguajes como Logo, Basic o Pascal, que son referentes históricos, están siendo desplazados por estos programas más modernos. Por ejemplo, Misfeldt y Ejsing-Duun (2015) elaboraron una experiencia con niños de 11 años de edad cuya tarea era diseñar y programar juegos sencillos para iPad en el lenguaje Hopscotch, lo cual garantizó desde el primer momento la elevada motivación de los alumnos, al mismo tiempo que les permitió aprender lenguaje algebraico sencillo, ángulos y sistemas de coordenadas.

Akpinar y Aslan (2015) aplicaron el lenguaje Scratch para la creación de un entorno de aprendizaje que permitiese el estudio de la probabilidad. Pese a que Scratch no dispone de elementos específicos para el tratamiento de cuestiones probabilísticas, la puesta en práctica de situaciones experimentales y simuladas pudo ayudar a mejorar la destreza de los alumnos hacia esta rama de las matemáticas. Respecto al Scratch, es importante mencionar el proyecto ScratchMaths (Benton *et al.*, 2016) que se desarrolló, inicialmente, en cuatro escuelas primarias de Londres con estudiantes de 9 a 11 años y se centra en el aprendizaje de los alumnos por medio de este lenguaje sobre cuatro objetivos básicos: explorar, explicar, predecir y compartir, los cuales garantizan un aprendizaje íntegro en los alumnos.

Otro de los muchos trabajos recientes acerca del lenguaje Scratch en educación es el estudio de Foerster *et al.* (2016) aplicado a una clase de alumnos, también de unos 11 años, con la idea de que aprendieran sobre congruencias de polígonos y construcción de teselaciones, lo cual permitió demostrar, meses después, los posibles efectos positivos a largo plazo que puede producir una metodología de estas características. De forma más específica, algunos autores han comprobado que el uso de la programación mejora el rendimiento de estudiantes de todos los niveles, como Wang *et al.* (2016) que analizaron los efectos del aprendizaje de la programación sobre dos grupos: estudiantes especialmente dotados para las matemáticas y alumnos de un nivel considerado medio. En este estudio, si bien los niños especialmente hábiles en matemáticas se beneficiaron aún más con la metodología, los resultados fueron significativamente buenos en ambos grupos, en términos de capacidad de resolución de problemas y motivación. Sin embargo, Lye y Koh (2014) concluyen que para el aprovechamiento de cualquier nueva metodología de este tipo es necesario el establecimiento de un entorno de aprendizaje basado en la resolución de problemas de forma constructiva. Esto debe incluir, básicamente, el procesado de datos y la reflexión continua, aunque siempre de forma creciente en cuanto a dificultad y que el docente actúe como guía del proceso.

De acuerdo con todos los estudios previos, la hipótesis del presente trabajo se basa en que la enseñanza de las matemáticas puede resultar aún más efectiva en potenciar el desarrollo de contenidos y destrezas propios de la asignatura en la etapa de educación secundaria si elegimos un lenguaje de programación como el R, que es uno de los más populares del momento y ofrece grandes ventajas.

## 2. METODOLOGÍA

### 2.1 Participantes

El presente estudio fue realizado con dos grupos de estudiantes de entre 14 y 15 años de edad: un primer grupo de 15 individuos y un segundo grupo de 18 alumnos españoles de tercer curso de educación secundaria de la clase de Informática del colegio San Pedro Pascual, de Valencia, España, quienes ya habían sido iniciados en la programación a través del lenguaje Scratch. De entre los 33 alumnos, 9 eran chicas (3 y 6 en cada grupo, respectivamente) y 24 eran chicos (12 y 12). El estudio fue llevado a cabo de la misma manera en ambos grupos, sin realizar ninguna distinción de sexo ni calificaciones entre los participantes. Debido al reducido número de alumnas que participaron en el estudio, no pueden obtenerse conclusiones significativas en cuanto al efecto del sexo del alumno en los resultados.

### 2.2 Lenguaje de programación

El lenguaje de programación elegido para las ocho sesiones fue el lenguaje R (Team, 2000) debido a que es uno de los 10 más populares del momento, según el índice PYPL (*The PYPL Popularity of Programming Language Index,* ver http://pypl.github.io/PYPL.html); es de libre uso, de fácil instalación y actualización, potente a nivel gráfico, dispone de una sintaxis agradable en comparación con otros lenguajes, pues destaca porque no es necesario definir los tipos de las variables, o la facilidad para trabajar con estructuras dinámicas, y dispone de un entorno de desarrollo integrado, RStudio, que facilita su uso y permite, simultáneamente, escribir instrucciones en consola, guardar documentos de trabajo, controlar las variables que están siendo utilizadas y visualizar gráficas, entre otras capacidades.

### 2.3 Tema de trabajo

El tema de la asignatura de Matemáticas que se escogió para el presente trabajo mediante el lenguaje de programación R fue la resolución de ecuaciones polinómicas, un elemento capital dentro del bloque de *Números y álgebra*, perteneciente al currículo español durante toda la etapa de educación secundaria.

Desde el punto de vista de contenidos matemáticos, el objetivo consistió en simplemente obtener todas las soluciones enteras de una ecuación polinómica de coeficientes enteros, y alcanzar este objetivo principal requirió del uso de diferentes técnicas y métodos de programación, al mismo tiempo que permitió el aprendizaje y/o refuerzo de variados conceptos matemáticos.

### 2.4 Sesiones

Respecto a la temporalidad del estudio, se impartieron cuatro sesiones de 50 minutos a cada uno de los dos grupos, realizando un total de ocho sesiones de implantación de esta nueva metodología. Las dos primeras sesiones consistieron en una introducción básica a los elementos y operaciones principales que pueden realizarse mediante el lenguaje R. Las dos últimas se dedicaron a poner en práctica, mediante este mismo lenguaje, la resolución de ecuaciones polinómicas de coeficientes enteros. El proceso de obtención de las soluciones de este tipo de ecuaciones requiere de dos pasos: obtener todos los divisores del término independiente del polinomio y, seguidamente, comprobar cuáles de estos divisores son solución, pues son los únicos candidatos enteros.

En la Figura 1 se muestra un esquema que representa el orden en que se plantearon los diferentes contenidos y problemas a lo largo de las cuatro sesiones.

Las elipses contienen los conceptos matemáticos básicos que fueron tratados de acuerdo con los problemas propuestos, los cuales se muestran dentro de rectángulos redondeados.

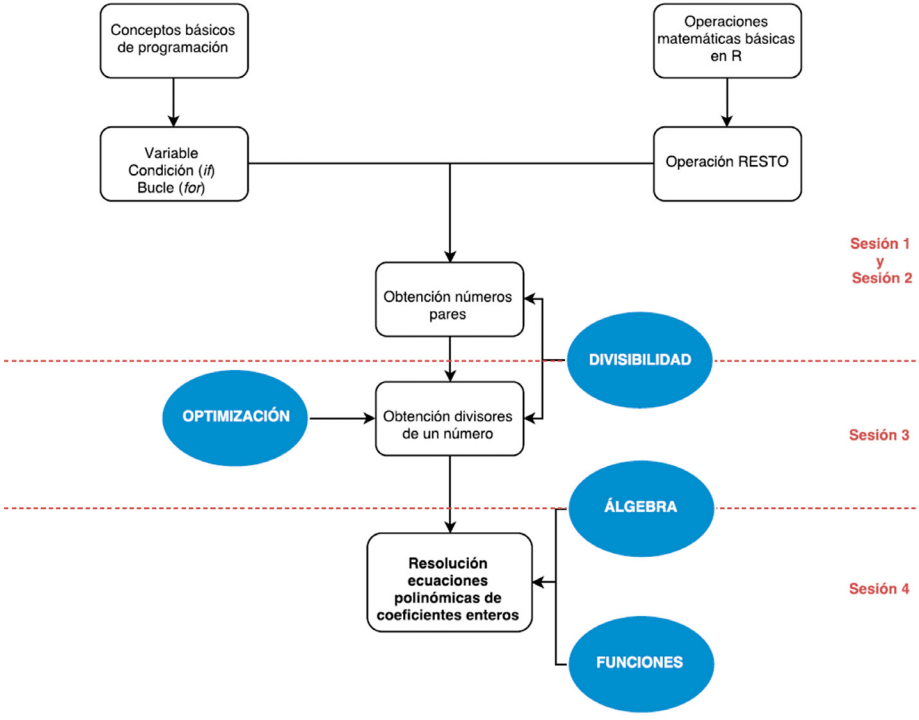


**Figura 1.** Esquema de la secuencia didáctica seguida durante las cuatro sesiones.

## 2.5 Encuesta

Con el fin de estudiar lo exitosa que resultó la metodología empleada, se pasó a los alumnos una encuesta con diversas preguntas en relación a las sesiones recibidas. Para medir con cierta precisión los efectos que sobre ellos tuvo la nueva metodología, las preguntas referentes a las sesiones fueron acompañadas de otras, totalmente generales, que valoraban su aptitud y actitud hacia las Matemáticas, la Informática y la combinación de ambas. Las preguntas que se hicieron al respecto son las que forman parte del test mtas (*Mathematics and Technology Attitudes Scale*) diseñado por Pierce *et al.* (2007). El test consta de 20 preguntas divididas en cinco categorías (cuatro preguntas para cada una) que podrían traducirse, a partir de su denominación original, de la forma

siguiente: *actitud hacia las matemáticas, confianza con la tecnología, confianza con las matemáticas, afición por las matemáticas y uso de la tecnología para aprender matemáticas.*

El test mtas, por tanto, mide la autopercepción de los alumnos sobre las cinco cuestiones mencionadas, las cuales involucran gusto y capacidad por las Matemáticas y la tecnología (Informática, básicamente) y su conjunción, en ciertos casos. Para ello, cada pregunta tiene asociada una respuesta en la escala 1-5, donde 1 indica *totalmente en desacuerdo* y 5 *totalmente de acuerdo*. Sin embargo, en el test que hicieron los alumnos que forman parte de este estudio se modificó la escala a una del tipo 1-6, con la intención de evitar respuestas neutras, que sí pueden darse en el test original (donde se asocian con el valor 3). Hicimos esta modificación para evitar un exceso de respuestas neutras, que podrían haber tenido como consecuencia una escasa variabilidad en los resultados de algunas de las preguntas en la muestra (33 alumnos) de la que se dispuso para el estudio. Por tanto, al haber elegido utilizar para la respuesta una escala 1-6, el valor central de la misma será 3,5 y servirá para conocer si la respuesta media de los alumnos ha sido positiva (>3,5) o negativa (<3,5) en cualquiera de las preguntas o categorías. De este manera se realizaron 10 preguntas sobre las propias sesiones correspondientes a varias categorías e incluyendo aspectos como el gusto de los alumnos por las sesiones o las dificultades experimentadas durante las mismas; en resumen, la encuesta utilizada para valorar el éxito de la metodología estuvo formado por 30 preguntas (Tabla 1) clasificadas por categorías (Figura 2).

| Código | Pregunta |
|---|---|
| P1 | Me concentro fácilmente en las clases de matemáticas. |
| P2 | Intento responder cuando el profesor de matemáticas hace preguntas. |
| P3 | Si cometo errores en matemáticas, trabajo hasta que los corrijo. |
| P4 | Si no soy capaz de resolver un problema, sigo intentándolo de diferentes maneras. |
| P5 | Soy bueno usando los ordenadores. |
| P6 | Soy bueno usando dispositivos como móviles, tablets, etc. |
| P7 | Soy capaz de arreglar numerosos problemas informáticos. |

| | |
|---|---|
| P8 | Soy capaz de manejar cualquier programa que necesito para el colegio. |
| P9 | Tengo una mente buena para las matemáticas. |
| P10 | Soy capaz de obtener buenos resultados en matemáticas. |
| P11 | Soy capaz de manejar las dificultades en matemáticas. |
| P12 | Tengo confianza en mí mismo sobre mi dominio de las matemáticas. |
| P13 | Me interesa aprender cosas nuevas sobre matemáticas. |
| P14 | En matemáticas eres recompensado por tus esfuerzos. |
| P15 | Disfruto aprendiendo matemáticas. |
| P16 | Siento satisfacción cuando resuelvo problemas de matemáticas. |
| P17 | Me gusta utilizar aplicaciones informáticas para aprender matemáticas. |
| P18 | Usar aplicaciones informáticas para aprender matemáticas vale la pena. |
| P19 | Las matemáticas son más interesantes con aplicaciones informáticas. |
| P20 | Las aplicaciones informáticas me ayudan a aprender matemáticas mejor. |
| P21 | Las clases en las que hemos usado R me han resultado interesantes. |
| P22 | Las clases en las que hemos usado R me han resultado divertidas. |
| P23 | He comprendido los programas y códigos que el profesor ha escrito en el lenguaje R. |
| P24 | He sido capaz de modificar algunos códigos en R para resolver nuevos problemas. |
| P25 | He sido capaz de escribir por mí mismo algunas instrucciones en el lenguaje R. |
| P26 | Me ha resultado fácil adaptarme a la sintaxis del lenguaje R. |
| P27 | He descubierto la necesidad de usar el ordenador para resolver ciertas ecuaciones. |
| P28 | He reforzado o ampliado mis conocimientos sobre ecuaciones polinómicas. |
| P29 | He adquirido nuevos conocimientos matemáticos que desconocía. |
| P30 | He repasado las notas elaboradas por el profesor antes de cada nueva clase. |

**Tabla 1.** Códigos y preguntas de la encuesta a valorar por los alumnos con la escala utilizada de 1 para *totalmente en desacuerdo* y 6 para *totalmente de acuerdo.*

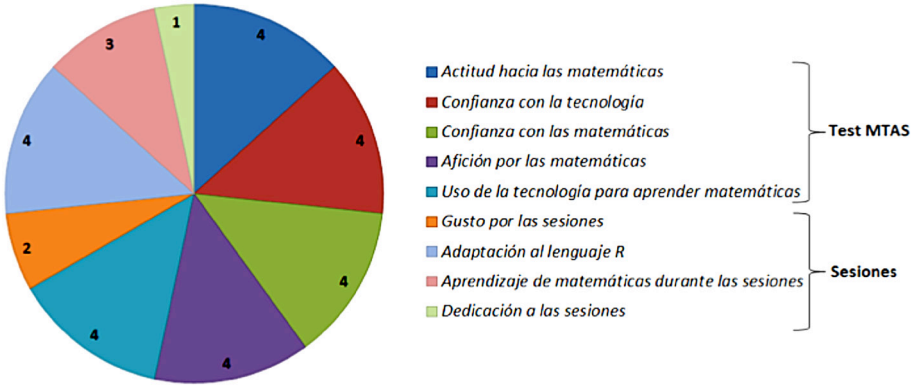


**Figura 2.** Encuesta con las 30 preguntas clasificadas por categorías en 9 bloques (5 para el test mtas, 4 para las preguntas directas sobre las sesiones). Dentro de cada uno de los sectores de este gráfico se indica el número de preguntas incluidas en la categoría correspondiente.

## 3. RESULTADOS

### 3.1 Sesiones

Como parte inicial de la primera sesión, se practicaron las operaciones matemáticas básicas, tratando la consola del lenguaje como una simple calculadora. Inmediatamente después los alumnos comenzaron a crear variables numéricas y a realizar operaciones sencillas entre ellas. Posteriormente introdujimos las listas de números y se enseñó a trabajar con ellas por medio de las operaciones básicas que se les asocia: extracción o sustitución de uno o varios elementos, reducción o ampliación de listas, etcétera. Para terminar con la primera sesión, se explicó el concepto de variable lógica o booleana (de tipo *true* o *false*) mediante diversos ejemplos.

La segunda sesión se dedicó principalmente a la implementación de bucles en el lenguaje R. Se partió de los más simples, cuyo objetivo suele ser mostrar una cierta cantidad de números ordenados por pantalla. Se propuso también la construcción de bucles que requiriesen del uso de variables lógicas (explicándose la condición de tipo *Si*) o del editado de una lista de números.

Se propuso a los alumnos el diseño de un bucle que almacenase todos los números pares (positivos) inferiores o iguales a 10. Como ya se habían explicado los operadores lógicos básicos, no les fue demasiado difícil obtener tal bucle, el

cual se muestra en la Figura 3 tal y como fue implementado por buena parte de los estudiantes.

```
pares=c()
for (i in c(1:10)){
  if (i==2 | i==4 | i==6 | i==8 | i==10){
    pares=c(pares,i)
  }
}
pares

## [1]  2  4  6  8 10
```

**Figura 3.** Código en lenguaje R para obtener los números pares menores o iguales a 10.

Si bien este bucle resuelve perfectamente el problema inicialmente planteado, es evidente que no resulta eficiente si se desea cambiar el número 10 por un número natural cualquiera, *n*, significativamente mayor. Para mejorar la resolución de este problema, los alumnos necesitaron revisar su idea de paridad y formalizarla debidamente a través del resto que se produce al dividir cualquier número natural entre 2, el cual solo puede ser 0 (número par) o 1 (número impar). De este modo, el bucle que finalmente implementaron fue el que aparece en la Figura 4, donde el símbolo %% representa el operador resto de una división entera en el lenguaje R.

```
pares=c()
for (i in c(1:10)){
  if (i%%2==0){
    pares=c(pares,i)
  }
}
pares

## [1]  2  4  6  8 10
```

**Figura 4.** Código en lenguaje R para obtener los números pares menores o iguales a 10. Este código, el cual sí hace uso del concepto preciso de paridad, supone una mejora respecto al de la Figura 3.

Una vez resuelto el problema de hallar los números pares inferiores a uno dado, el siguiente paso fue obtener todos los divisores de un número cualquiera, lo que supuso el núcleo de la tercera sesión. Así pues, el problema de encontrar y almacenar números pares se planteó como ejemplo básico para servir de

referencia a la hora de obtener divisores, lo cual constituía un objetivo fundamental para cubrir el tema de resolución de ecuaciones polinómicas.

De esta forma, los alumnos no tardaron en adivinar que bastaba con modificar el código previo sobre números pares para resolver este nuevo problema, aunque algunos se encontraron con dificultades. La falta de atención sobre la manera en que estaba construido el bucle relativo a los número pares llevó a la modificación de la condición de paridad (*i %% 2 == 0*) por una nueva del tipo *i %% n == 0*, como puede verse en la Figura 5. Lo que evalúa esta condición no son los posibles divisores de *n*, sino los números de los que *n* es divisor. Obviamente, la ejecución del bucle anterior con esta modificación solo puede producir el mayor divisor de *n*, que no es otro que el propio *n*, como pudieron apreciar los estudiantes que llevaron a cabo este cambio.

```
divisores=c()
for (i in c(1:100)){
  if (i%%100 == 0){
    divisores=c(divisores,i)
  }
}
divisores

## [1] 100
```

**Figura 5.** Código en lenguaje R que pretende, erróneamente, producir los divisores del número 100.

Tras unos minutos, la mayoría de los alumnos consiguió darse cuenta de que la modificación requerida respecto del bucle anterior era ligeramente superior, siendo imprescindible invertir los papeles de *i* y *n* para plantear la condición *n %% i == 0*, como puede verse en la Figura 6.

```
divisores=c()
for (i in c(1:100)){
  if (100%%i == 0){
    divisores=c(divisores,i)
  }
}
divisores

## [1]   1   2   4   5  10  20  25  50 100
```

**Figura 6.** Código en R para poder obtener correctamente todos los divisores positivos del número 100.

Este último código, que ahora sí es correcto, es un ejemplo de cómo la modificación del código que resuelve un problema puede permitir la fácil resolución de otro de naturaleza similar, si bien es necesario ser cuidadoso y realizar todos los cambios necesarios.

Sobre este mismo problema fue planteada la posibilidad de optimizar el proceso que se estaba implementando, reduciendo la cantidad de números intervinientes en el bucle. Por ser este asunto de mayor complejidad, se aconsejó a los alumnos que sustituyeran el valor de 100 *(en i in c(1:100))* por números inferiores, para ver en qué medida el resultado se veía afectado. Tras varios intentos, algunos ya comenzaron a intuir que a partir de cierto valor límite no se producían cambios en cuanto a los divisores encontrados, develándose finalmente que solo era necesario ejecutar el proceso repetitivo hasta $i = n / 2$, es decir, hasta $i = 50$ para el caso del número 100. De esta forma, los estudiantes pudieron reforzar su conocimiento sobre los divisores de un número y apreciar la necesidad de reducir el coste de los programas siempre que sea posible.

Por último, para tener completamente resuelto el problema de obtener los divisores de un número, se especificó la necesidad de considerar también los divisores negativos del número dado. Normalmente, cuando se piensa en los divisores de un número positivo, son excluidos los posibles candidatos inferiores a 0, pero para la resolución de ecuaciones polinómicas de coeficientes enteros es necesario que también sean considerados.

Una fácil modificación del código que genera los divisores de 100, para que también se almacenen los divisores negativos, puede apreciarse en la Figura 7.

```
divisores=c()
for (i in c(1:100)){
  if (100%%i == 0){
    divisores=c(divisores,i,-i)
  }
}
divisores
```

```
##  [1]    1   -1    2   -2    4   -4    5   -5   10  -10   20  -20   25
-25
## [15]   50  -50  100 -100
```

**Figura 7.** Código en R para poder obtener todos los divisores, positivos y negativos, del número 100.

Aunque esta forma fue la considerada por gran parte del alumnado, algunos estudiantes intentaron hacer lo que se muestra en la Figura 8.

```
divisores=c()
for (i in c(-100:100)){
  if (100%%i == 0){
    divisores=c(divisores,i)
  }
}

## Error in if (100%%i == 0) {: valor ausente donde TRUE/FALSE es
necesario

divisores

## [1] -100  -50  -25  -20  -10   -5   -4   -2   -1
```

**Figura 8.** Código en R para poder obtener todos los divisores, positivos y negativos, de 100.

En este caso, la diferencia del código presentado en la Figura 7, el resultado no es satisfactorio. Como puede intuirse por el mensaje que devuelve el programa y por el estado de la lista de divisores, el error tiene que ver con el caso $i = 0$.

Esta posibilidad se basaba en modificar el vector de números que son recorridos en el bucle, pasando a contener también todos los enteros negativos correspondientes, en lugar de modificarse la instrucción contenida en la condición (como se hace en el código anterior). Si bien es una buena idea, pues demuestra una cierta familiaridad con el lenguaje, este código genera un error en el programa cuando la variable *i* toma el valor 0.

Este hecho permitió que los alumnos que tomaron esta decisión valoraran la razón del error, concluyendo que el problema se debía a la imposibilidad de obtener el resto de una división entre 0, pues tal división no está definida. En cualquier caso, no fue difícil retocar esta opción alternativa a partir del operador lógico *and*, excluyendo directamente el 0 como posible divisor (Figura 9) y resolviendo correctamente el problema.

```
divisores=c()
for (i in c(-100:100)){
  if (100%%i == 0 & i!=0){
    divisores=c(divisores,i)
  }
}
divisores

##  [1] -100  -50  -25  -20  -10   -5   -4   -2   -1    1    2    4    5
10
## [15]   20   25   50  100
```

**Figura 9.** Modificación necesaria del código contenido en la Figura 8 para obtener todos los divisores.

Una vez resuelto completamente el problema de obtener todos los divisores de un número dado, para llegar al objetivo final de implementar la resolución de ecuaciones polinómicas solo quedaba diseñar otro bucle, el cual fue tratado durante la cuarta y última sesión.

En primer lugar se debían obtener los divisores de su término independiente mediante el código construido en la sesión anterior, partiendo de una ecuación polinómica cualquiera. Enseguida, un bucle debía recorrer el conjunto de divisores (previamente almacenados en una lista de números) para seleccionar aquellos que verificasen la ecuación. Por tanto, la condición a incluir en este nuevo bucle valoraba directamente si cada divisor era solución, o no, de la ecuación planteada. Un primer intento consistió en escribir explícitamente la relación correspondiente a la ecuación dentro de la condición. Si, por ejemplo, en la ecuación polinómica de grado 5 con expresión algebraica $x^5 - 35x^4 + 470x^3 - 3010x^2 + 9129x - 10395 = 0$, la condición que debía implementarse era *x^5 − 35*x^4 + 470*x^3 − 3010*x^2 + 9129*x − 10395 == 0*, tomando *x* el valor de cada uno de los divisores del término independiente.

Con la intención de refinar la evaluación de la condición de ser solución, se dedicaron unos minutos a repasar el concepto de función matemática y a explicar cómo pueden ser implementadas y utilizadas en el lenguaje R. De esta manera, si la función que representa a un polinomio vale 0 para cierto divisor, tal divisor es una solución de la ecuación polinómica. Con todo esto, las figuras 10 y 11 contienen los bloques de código que llevaron a cabo los estudiantes para obtener las soluciones de la ecuación de quinto grado previamente enunciada, tratando al polinomio como una función matemática, *p(x)*, cuya definición se realiza al inicio del código incluido en la Figura 11.

```
n=10395
divisores=c()
for (i in c(1:n)){
  if (n%%i == 0){
    divisores=c(divisores,i,-i)
  }
}
```

**Figura 10.** Código en R para obtener los divisores del número 10395.

No se muestran en la imagen los divisores que se obtienen por ser un total de 64.

```
p=function(x) x^5-35*x^4+470*x^3-3010*x^2+9129*x-10395
num_divisores=length(divisores)
soluciones=c()
for (i in c(1:num_divisores)){
  divisor=divisores[i]
  if (p(divisor) == 0){
    soluciones=c(soluciones,divisor)
  }
}
```

**Figura 11.** Resolución de la ecuación $x^5 - 35x^4 + 470x^3 - 3010x^2 + 9129x - 10395 = 0$ en lenguaje R.

## 3.2 Encuestas

Los resultados de los cuatro bloques dedicados a las sesiones de aplicación de la metodología se muestran en la Figura 12 y exceptuando los resultados para la variable *dedicación a las sesiones*, las otras tres variables implicadas de esta parte de la encuesta presentan valores medios claramente superiores al central de 3,5.

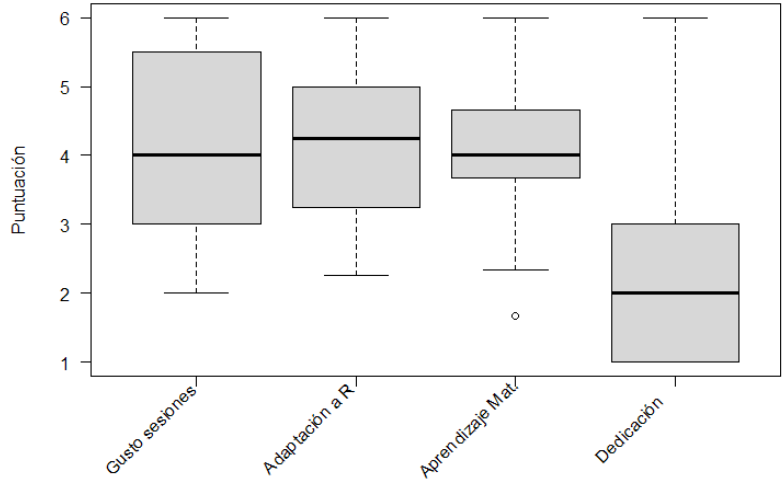


**Figura 12.** Diagramas de cajas para las respuestas de los alumnos en las cuatro categorías con las preguntas sobre las sesiones impartidas.

La puntuación media de *dedicación a las sesiones* muestra que el compromiso de trabajo que adquirieron los estudiantes fue más bien bajo. Ante la falta de

evaluaciones anunciadas sobre el contenido de las sesiones, los alumnos, salvo algunas excepciones, acudieron a cada nueva sesión sin repasar convenientemente lo visto en las anteriores y, evidentemente, esto debió traer como consecuencia mayor dificultad para asumir los nuevos problemas que se les iban planteando, lo que en parte se reflejó en la variable *adaptación al lenguaje R*.

A continuación se presentan los resultados obtenidos mediante el análisis por bloques de preguntas, por preguntas, por tipo de alumno y, finalmente, el análisis individual.

### 3.2.1 *Análisis por bloques de preguntas*

El primer paso consiste en calcular el nivel de relación existente entre todas las variables disponibles mediante el llamado coeficiente de correlación, el cual oscila entre -1 y 1. El valor -1 indica una relación completamente negativa y el 1 justo lo contrario. En la Figura 13 se muestran las correlaciones obtenidas entre todas las variables de interés.

En esta gráfica no se distingue ninguna relación fuertemente negativa entre las variables, es decir, no existe ninguna cuyo mayor valor influya negativamente en otra. Al contrario, lo que se observan son varias relaciones claramente positivas entre algunas de ellas, como las fuertes correlaciones existentes entre los bloques *gusto por las sesiones*, *adaptación al lenguaje R* y *aprendizaje de matemáticas durante las sesiones*. La presencia clara de esta relación entre las tres variables parece revelar que los alumnos que disfrutaron de las sesiones comprendieron adecuadamente el lenguaje y absorbieron los contenidos matemáticos, fueron en buena medida los mismos para las tres facetas. Esto es en parte positivo, ya que es un indicador de que algunos fueron capaces de aprovechar las sesiones de la manera más completa posible. Pero, a su vez, indica que otros que tuvieron dificultades para adquirir el lenguaje fueron poco capaces de disfrutar o de adquirir/reforzar los conocimientos matemáticos que se trataron. Por último, debe destacarse la mayor correlación que puede observarse entre las variables *confianza con las matemáticas* y *aprendizaje de matemáticas durante las sesiones*. Independientemente de los gustos hacia las matemáticas o la tecnología, los estudiantes con mejor opinión sobre sí mismos en cuanto a dominio matemático fueron también quienes consideraron que aprendieron más matemáticas durante las sesiones, probablemente debido a que su verdadero mayor nivel matemático permitió el correcto aprovechamiento de las clases en

ese aspecto, sin importar tanto su mayor o menor desempeño e interés por el lenguaje de programación R.

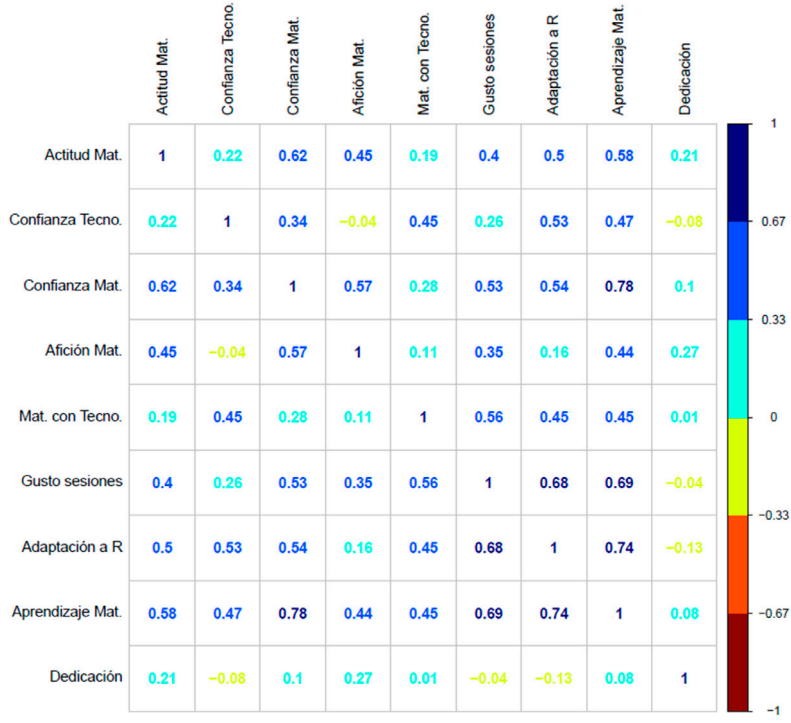


**Figura 13.** Gráfica de correlaciones entre los 9 bloques de preguntas del encuesta. El intervalo [-1,1] se representa en escala de colores: el azul más intenso representa una mayor correlación positiva, y el rojo justo lo contrario.

### 3.2.2 *Análisis por preguntas*

El análisis de las respuestas siguiendo los bloques/categorías determinados, tanto los del mtas como los propios del desarrollo de las sesiones, permite llegar a alguna conclusión sobre cómo transcurrieron las mismas y los efectos que pudieron producir en los alumnos. De este modo, se han analizado directamente las posibles relaciones entre todas y cada una de las 30 preguntas, en busca de un conocimiento más completo.

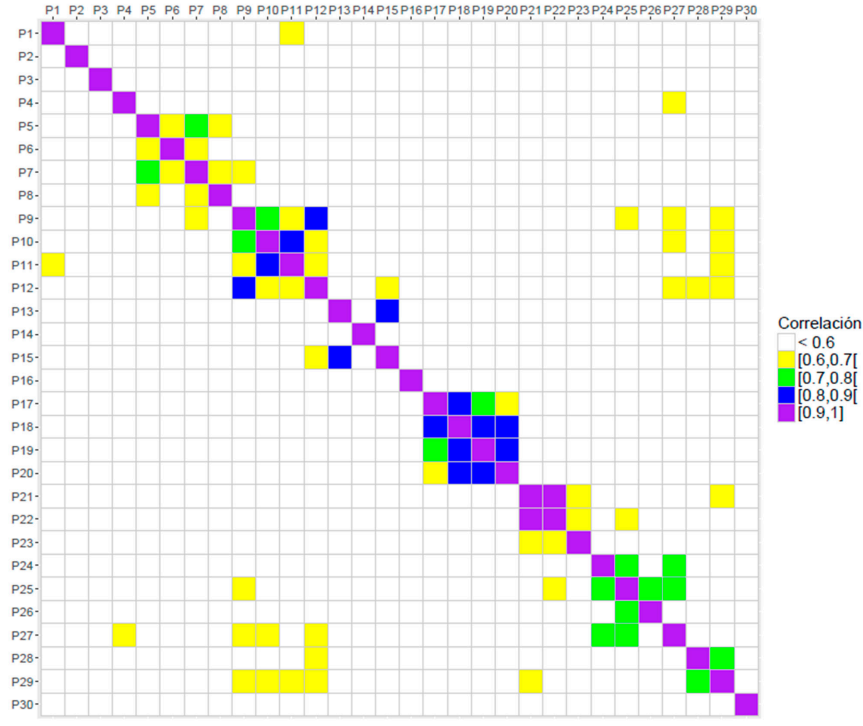


**Figura 14.** Correlaciones entre las preguntas, desde la P1 hasta la P30.
Solo se colorean las casillas correspondientes a aquellas correlaciones con valor superior a 0,6, que son las más destacables.

La Figura 14 muestra las correlaciones de mayor relevancia producidas dentro del conjunto de las 30 preguntas. En esta gráfica se distinguen algunas agrupaciones de preguntas consecutivas muy correlacionadas, por corresponderse con los propios bloques previamente definidos como los de *confianza con la tecnología* (P5-P8), *confianza con las matemáticas* (P9-P12) y *uso de la tecnología para aprender matemáticas* (P17-P20). La fuerte relación entre preguntas de un mismo bloque es esperable, pero no todos los bloques o categorías muestran el mismo comportamiento. De hecho, dentro del de *confianza con la tecnología* (P5-P8), altamente correlacionado internamente, las preguntas P6 y P8 tienen poco que ver entre sí, lo cual indica que el buen uso de los dispositivos digitales no siempre implica un buen manejo de los programas informáticos que se utilizan en las aulas, y viceversa.

Un ejemplo de bloque con muy poca correlación interna es el de *actitud hacia las matemáticas* (P1-P4). De tal manera, cuestiones como la capacidad para concentrarse en las clases de matemáticas, el nivel de participación en las mismas, o la insistencia de cara a resolver las dificultades y problemas que originan parecen tener poco en común, al menos en el conjunto de alumnos aquí estudiado. Ocurre algo similar en el bloque de *afición por las matemáticas*, ya que no se dan coincidencias destacables entre algunas de las preguntas, lo cual resulta sorprendente; por ejemplo, la poca correlación entre el interés por aprender cosas nuevas sobre matemáticas y la satisfacción que se siente al resolver un problema.

Es importante también analizar las fuertes relaciones que se hayan podido producir entre las respuestas de algunas de las preguntas del test mtas y las que trataban de forma particular sobre las sesiones que se han llevado a cabo, o incluso entre preguntas de estas últimas que pertenecen a categorías distintas (ver Tabla 2).

| Pregunta A | Pregunta B | Corr. |
|---|---|---|
| P27 | P25 | 0,77 |
| P27 | P24 | 0,72 |
| P29 | P12 | 0,70 |
| P11 | P1 | 0,69 |
| P27 | P9 | 0,67 |
| P29 | P9 | 0,67 |
| P27 | P4 | 0,65 |
| P29 | P21 | 0,64 |
| P25 | P22 | 0,64 |
| P27 | P10 | 0,64 |
| P27 | P12 | 0,64 |
| P25 | P9 | 0,62 |
| P23 | P21 | 0,62 |
| P29 | P11 | 0,62 |
| P15 | P12 | 0,61 |
| P23 | P22 | 0,61 |
| P29 | P10 | 0,60 |
| P28 | P12 | 0,60 |
| P27 | P22 | 0,60 |

**Tabla 2.** Preguntas de bloques diferentes que muestran mayor correlación (Corr.) entre sí.

Uno de los objetivos transversales de las sesiones impartidas era que los alumnos apreciaran la necesidad de utilizar el ordenador para la resolución de ciertas ecuaciones, inabordables para una persona sin ayuda Informática. Este objetivo fue generalmente conseguido (con una media de 4,18 para la P27), pero la capacidad de adaptación al lenguaje resultó determinante para que los estudiantes se dieran cuenta de ello. Pese a ser algo que parecía fácil de conseguir, pues se plantearon ecuaciones con coeficientes muy superiores a las vistas en clase de Matemáticas, algunos alumnos no valoraron la utilidad del ordenador ni del lenguaje de programación para resolverlas.

La adquisición de nuevos conocimientos matemáticos, objetivo capital de la experiencia realizada, también mostró una aceptación general, pero de forma algo más discreta (3,97 para la P29). Además, la percepción de haber aprendido matemáticas mantuvo una fuerte relación con las cuestiones relacionadas con la categoría *confianza con las matemáticas*, especialmente con las preguntas P12, P9 y P10. Del mismo modo que los resultados obtenidos en el análisis por bloques, los estudiantes con buena opinión sobre sí mismos en relación al conocimiento matemático, también consideraron que aprendieron nuevas matemáticas durante las sesiones.

Respecto a las preguntas de la categoría *gusto por las sesiones*, la P21 y la P22, ambas muestran su mayor correlación con las referentes a la capacidad del alumno para comprender, modificar y utilizar la sintaxis del lenguaje empleado durante las sesiones. A su vez, este tipo de habilidades se relaciona de forma positiva, aunque no demasiado fuerte, con la pregunta P9 (*tengo una mente buena para las matemáticas*). De nuevo, los datos parecen indicar que el manejo del lenguaje R tuvo un peso demasiado elevado para el ideal aprovechamiento de las sesiones.

### 3.2.3 *Análisis por tipo de alumno*

Puede tratarse de agrupar a los participantes en el estudio en función de su nivel de matemáticas, para luego analizar los resultados de cada grupo en las preguntas referentes a las sesiones. Esta tarea no es nada trivial a partir de meros datos, pero puede realizarse una aproximación en este sentido, que permita reforzar lo comentado. De este modo, utilizando la técnica de clasificación de datos conocida como *clustering* sobre las preguntas del test mtas, es posible efectuar una división de este tipo. Concretamente, se aplicó el *algoritmo de las*

*k-medias* (con $k = 3$) utilizando las preguntas incluidas en los bloques *actitud hacia las matemáticas*, *confianza con las matemáticas* y *afición por las matemáticas*. Mediante la aplicación de este algoritmo de clasificación fue posible dividir al total del alumnado (33 alumnos) en tres grupos de diferente nivel: bajo (8 alumnos), medio (17 alumnos) y alto (8 alumnos). Esta forma de agruparlos solo tiene en cuenta las variables más íntimamente relacionadas con sus sensaciones hacia las matemáticas (obviando las equivalentes con la tecnología).

Aun siendo conscientes de las limitaciones existentes en esta separación de los estudiantes por niveles, y teniendo en cuenta que los tamaños muestrales dentro de cada nivel son muy reducidos, la Tabla 3 proporciona una idea para comprobar cómo se comportan los promedios en cada grupo y reafirma lo mencionado de que los alumnos especialmente inclinados hacia las matemáticas disfrutaron de las sesiones, pero los que presentan menor interés o mayores dificultades no reaccionaron de forma tan positiva. En cualquier caso, es muy destacable que el grupo clasificado como de nivel medio (el más numeroso) aceptara notablemente la metodología planteada.

| Categoría | Nivel bajo | Nivel medio | Nivel alto |
|---|---|---|---|
| *Actitud hacia las matemáticas* | 3,63 | 4,54 | 5,38 |
| *Confianza con las matemáticas* | 2,66 | 4,43 | 5,59 |
| *Afición por las matemáticas* | 3,19 | 4,09 | 5,09 |
| *Gusto por las sesiones* | 3,56 | 4,06 | 5,06 |
| *Adaptación al lenguaje R* | 3,34 | 4,21 | 4,66 |
| *Aprendizaje de matemáticas durante las sesiones* | 3,08 | 4,08 | 4,92 |

**Tabla 3.** Promedios de las respuestas para las tres categorías del test mtas referentes a matemáticas y para los tres bloques principales de preguntas sobre las sesiones.

Se distinguen tres tipos de alumnos en función de su actitud, confianza y afición hacia las matemáticas.

### 3.2.4 *Análisis individual*

Por último, se realiza un análisis que consiste en seguir el patrón de respuestas de cada estudiante de forma individualizada. Esto nos puede llevar a descubrir

ciertas situaciones indetectables por los procedimientos estadísticos anteriores. El estudio por casos detecta varios alumnos para los que la aplicación de la metodología fue todo un éxito. Ellos, pese a sus dificultades o desagrado por las matemáticas, o incluso hacia la idea de usar Informática para aprenderlas, aprobaron de forma muy positiva la metodología que se aplicó durante las sesiones.

| Test mtas | | | | | |
|---|---|---|---|---|---|
| Act. Mat. | Conf. Tecno. | Conf. Mat. | Af. Mat. | Mat. Tecno. | Gusto sesiones |
| 4,75 | 5,5 | 5,75 | 2,75 | 3,5 | **6** |
| 3,25 | 4,5 | 3,25 | 3,25 | 2,75 | **5** |
| 4 | 5,25 | 4,5 | 3,5 | 5 | **6** |

**Tabla 4.** Promedios de las respuestas para las cinco categorías del test mtas y para la de *gusto por las sesiones* en tres alumnos con quienes se podría considerar un éxito la aplicación de la metodología.

En la Tabla 4 pueden verse las puntuaciones para el test mtas y para la variable g*usto por las sesiones* de tres estudiantes para los que el método resultó muy efectivo. Es especialmente interesante ver cómo la metodología fue positiva para los dos primeros alumnos de la tabla, a pesar de su poco afecto por las matemáticas y por el uso de la Informática para su estudio.

## 4. DISCUSIÓN

Combinando la información que aportan las encuestas efectuadas y las apreciaciones que pudieron realizarse durante el desarrollo de las sesiones, pueden resumirse los siguientes puntos, relacionados con las dificultades detectadas y los aprendizajes realizados.

### 4.1 Dificultades en el uso del lenguaje R

Como todo lenguaje de programación, la sintaxis del lenguaje R es estricta, lo cual significa que el uso de palabras y símbolos debe hacerse con total precisión para evitar errores. Además, el uso de cualquier lenguaje supone una adaptación

importante, pues obliga a que un principiante razone de forma a veces poco natural y contra intuitiva. Estas dificultades han sido valoradas en las preguntas P23-P26 de la encuesta.

Los resultados de la encuesta reflejan que el nivel de comprensión del lenguaje y de las explicaciones asociadas fue bastante bueno, como también la capacidad para modificar ciertos códigos para resolver nuevos problemas, en líneas generales (promedios de 4,42 y 4,21 para las preguntas P23 y P24). En cambio, sí se observa una peor puntuación sobre la adaptación al lenguaje y la capacidad para escribir con él de manera autónoma (promedios de 3,88 y 3,91 para las preguntas P25 y P26). Esto se corresponde totalmente con lo observado durante las sesiones. No resulta fácil adaptarse a un nuevo lenguaje de programación, especialmente si la única referencia que se posee en ese sentido es con un lenguaje cuya sintaxis es mucho más relajada, donde la forma de elaborar los programas es en buena parte gráfica, como ocurre con el Scratch. En un lenguaje como el R, al igual que en cualquier otro lenguaje de programación avanzado, es necesario escribir cuidadosamente cualquier proceso que se quiera implementar, lo cual complica el aprendizaje inicial. Aunque, al mismo tiempo, esta rigidez es la base del desarrollo de habilidades de tipo matemático a través de la programación.

Ahora bien, ¿sería posible superar dicha dificultad en alumnos de educación secundaria? En primer lugar, haría falta dedicarle tiempo. En la experiencia que aquí se analiza solo se dispuso de cuatro sesiones, por lo cual era esperable que parte de los alumnos no se acostumbrara al uso del lenguaje al finalizar el proceso.

Además de la inversión en tiempo, pueden proponerse algunas actividades concretas que ayuden a acelerar el proceso de adaptación del estudiante al lenguaje de programación. Una primera opción –que ya se llevó a cabo durante las sesiones objeto de estudio– fue plantear modificaciones del código para resolver nuevos problemas. Esto permite que el alumno solamente se centre en ciertas partes de un programa hasta conseguir que resuelva un problema ligeramente diferente al de partida, lo que disminuye la dificultad general del proceso.

Esta actividad puede plantearse de dos maneras, indicando los puntos concretos que requieren modificación u omitiendo esta información, lo cual supone una mayor dificultad. En las sesiones que se realizaron se empezaba planteando la modificación sin aclarar el lugar donde había que introducir cambios, pero normalmente se acababa indicando dónde había que mirar para facilitar la resolución de los problemas.

Otro ejercicio de gran utilidad sería proporcionar a los alumnos trozos de código con errores de sintaxis, para que ellos los detectaran y corrigieran. Según lo experimentado, cuando los estudiantes cometían errores sintácticos, lo cual ocurría con frecuencia, tenían serios problemas para hallarlos y subsanarlos. Además, no demostraban demasiado interés en encontrarlos por sí mismos, pues tendían a pedir ayuda con excesiva prontitud. En relación a este punto, cabría preguntarse en qué medida pudo ser un problema para los alumnos (españoles) el hecho de que las palabras clave del lenguaje y, especialmente los mensajes de error que la consola del lenguaje muestra en la pantalla, estén en lengua inglesa. En general, esto podría resultar una dificultad añadida, pero en alumnos con tan poca experiencia en programación, incluso si los mensajes de error hubieran aparecido en su propia lengua, su capacidad para corregir errores habría sido presumiblemente igual de baja. En cualquier caso, el planteamiento de ejercicios de detección de errores podría ser clave para que los estudiantes aumentasen su autonomía y comprensión del lenguaje de programación.

En definitiva, pese a las dificultades que existieron durante las sesiones, una mayor dedicación temporal al aprendizaje del lenguaje, un diseño de actividades específicas para suplir ciertas carencias y un mayor compromiso por parte de los alumnos (que no se dio en este caso, como ha demostrado la variable *dedicación a las sesiones*) hacen suponer que los resultados podrían ser bastante mejores a mediano y largo plazo.

### 4.2 Aprendizajes sobre matemáticas

Como se ha comentado, el tema de la asignatura de Matemáticas que se trató a través del lenguaje R fue la resolución de ecuaciones polinómicas de coeficientes enteros. Aunque las dos primeras sesiones fueron dedicadas a cuestiones generales del programa, necesario ante la inexperiencia de los alumnos, las dos últimas consistieron en aplicarlo a este tema concreto. El objetivo final era elaborar un código en el lenguaje R que, dada una ecuación polinómica de coeficientes enteros, proporcionase todas sus soluciones enteras en una estructura de tipo lista de números. Como paso previo, siguiendo lo que se estudia en la asignatura de Matemáticas, era necesario obtener todos los divisores del término independiente del polinomio. Estos divisores, como es bien sabido, son los únicos candidatos a soluciones enteras de la ecuación.

La consecución de este objetivo final supuso el repaso de los contenidos sobre resolución de polinomios vistos en Matemáticas, pero también permitió la incorporación de algunas ideas complementarias que les eran poco conocidas.

De cara a la obtención de los divisores de un número, fue necesario trabajar con los restos de divisiones enteras. Aunque esto pueda parecer muy básico, algunos alumnos mostraron dificultades para enlazar el resto de una división con el hecho de que un número sea divisor de otro, o también para asociar la paridad de un número con su resto al dividir entre 2 para un ejemplo previo al de la obtención de los divisores. El repaso de estos conceptos sirvió para recordar cuestiones básicas, ya casi olvidadas.

Los estudiantes fueron cuestionados sobre la necesidad de buscar divisores entre todos los números menores a uno dado, *n*, para resolver este problema, y pocos consiguieron resolver este asunto, aunque algunos sí observaron que era suficiente con probar hasta *n / 2*, ahorrando muchas operaciones al ordenador.

Por otro lado, también fue imprescindible repasar el concepto de función matemática. Como paso previo a la obtención de las raíces enteras del polinomio, resultó conveniente su definición como función matemática, y también se aprovechó para realizar la representación gráfica de algunos polinomios básicos (recordatorio de rectas y parábolas). Con todo esto, parece evidente que las clases en las cuales se inició a los alumnos en el lenguaje R también sirvieron para enseñar y repasar numerosos contenidos matemáticos. Es especialmente interesante observar cómo la completa implementación de un contenido puramente algebraico obligó a la revisión de cuestiones de divisibilidad, optimización y funciones.

En vista de los contenidos tratados, y de la forma en que se hizo durante las sesiones, parece sensato afirmar que los estudiantes reforzaron contenidos matemáticos y aprendieron otros nuevos. Esto puede resultar un poco contradictorio con sus respuestas en la encuesta al respecto, que no son tan altas como podría esperarse. Es posible que el hecho de no seguir el esquema de enseñanza habitual, sino el haber ido aportando ideas según era necesario, provocase que los alumnos fuesen menos conscientes de su proceso de aprendizaje.

## 5. CONCLUSIONES

La aplicación de la metodología de aprendizaje propuesta, basada en la utilización del lenguaje de programación R para la enseñanza de las matemáticas durante toda la etapa de educación secundaria, puede resultar muy positiva,

como se demuestra en este estudio con la puesta en práctica realizada con alumnos del tercer curso de educación secundaria.

El uso de esta potente herramienta docente para el tratamiento de contenidos algebraicos propios de la asignatura de Matemáticas de su curso, además de otros contenidos matemáticos de carácter transversal, reforzó los conocimientos de programación adquiridos en la asignatura de Informática por los estudiantes, mejorando al mismo tiempo sus conocimientos matemáticos. Además, durante las sesiones se observó que esta metodología docente favoreció que su interés fuese superior al habitual en la asignatura de Matemáticas.

Pese a los buenos resultados obtenidos, la puesta en práctica de esta metodología no resultó eficiente en un cierto porcentaje de los alumnos debido a dificultades típicamente asociadas al aprendizaje inicial de un lenguaje de programación y relacionadas, sobre todo, con el manejo de su sintaxis. Se considera que es muy probable que una aplicación más prolongada en el tiempo de la metodología pudiera haber disipado gran parte de estas dificultades.

En cualquier caso, debido a la complejidad intrínseca a la programación, resulta imprescindible tener en cuenta la atención a la diversidad a la hora de aplicar una metodología de este tipo sobre un grupo heterogéneo de alumnos.

### Agradecimientos

Los autores agradecen la colaboración del alumnado y profesorado del colegio San Pedro Pascual de Valencia (España) implicados en el presente estudio.

## REFERENCIAS

Akpinar, Y. & Aslan, Ü. (2015). Supporting Children's Learning of Probability through Video Game Programming. *Journal of Educational Computing Research, 53*(2), 228–259.

Battista, M. T. & Clements, D. H. (1986). The Effects of Logo and CAI Problem-Solving Environments on Problem-Solving Abilities and Mathematics Achievement. *Computers in Human Behavior, 2*, 183–193.

Benton, L., Hoyles, C., Kalas, I. & Noss, R. (2016). Building Mathematical Knowledge with Programming: Insights from the ScratchMaths Project. In *Constructionism in Action 2016: Conference Proceedings* (pp. 26–33).

Booth, K. H. V. (1958). *Programming for an Automatic Digital Calculator*. London: Butterworths.

Clements D. H. (1987). Longitudinal Study of the Effects of Logo Programming on Cognitive Abilities and Achievement. *Journal of Educational Computing Research, 3*(1), 73–94.

Du Boulay, B. (1986). Some Difficulties of Learning to Program. *Journal of Educational Computing Research, 2*(1), 57–73.

Du Boulay, J. B. H. (1978). *Learning Primary Mathematics through Computer Programming,* PhD. Thesis, University of Edinburgh.

Feurzeig, W. & Lukas, G. (1972). Logo - A Programming Language for Teaching Mathematics. *Educational Technology, 12*(3), 39–46.

Feurzeig, W., Papert, S. A. & Lawler, B. (2011). Programming-languages as a Conceptual Framework for Teaching Mathematics. *Interactive Learning Environments, 19*(5), 487–501.

Foerster, K. T. (2016). Integrating Programming into the Mathematics Curriculum: Combining Scratch and Geometry in Grades 6 and 7. In *Proceedings of the 17th Annual Conference on Information Technology Education* (pp. 91–96).

Govender, I. (2009). *Learning to Program, Learning to Teach Programming: Pre-and in Service Teachers' Experiences of an Object-oriented Language,* PhD. Thesis, University of South Africa.

Howe, J. A. M., Ross, P. M., Johnson, K. R., Plane, F. & Inglis, R. (1982). Teaching Mathematics through Programming in the Classroom. *Computers & Education, 6*(1), 85–91.

Hromkovič, J. (2006). Contributing to General Education by Teaching Informatics. In *International Conference on Informatics in Secondary Schools-Evolution and Perspectives* (pp. 25–37). Springer, Berlin.

Linn, M. C. & Dalbey, J. (1989). Cognitive Consequences of Programming Instruction. *Studying the Novice Programmer,* 57–81.

Lye, S. Y. & Koh, J. H. L. (2014). Review on Teaching and Learning of Computational Thinking through Programming: What is Next for K-12? *Computers in Human Behavior, 41*, 51–61.

McCracken, D. D. (1957). *Digital Computer Programming*. John Wiley & Sons, New York.

Misfeldt, M. & Ejsing-Duun, S. (2015). Learning Mathematics through Programming: An Instrumental Approach to Potentials and Pitfalls. In *CERME 9-Ninth Congress of the European Society for Research in Mathematics Education* (pp. 2524–2530).

Noss, R. (1986). Constructing a Conceptual Framework for Elementary Algebra through Logo Programming. *Educational Studies in Mathematics, 17*, 335–357.

Papert, S. (1972). Teaching Children Thinking. *Programmed Learning and Educational Technology, 9*(5), 245–255.

Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, Inc.

Pierce, R., Stacey, K. & Barkatsas, A. (2007). A Scale for Monitoring Students' Attitudes to Learning Mathematics with Technology. *Computers & Education*, *48*(2), 285–300.

Resnick, M. *et al.* (2009). Scratch: Programming for All. *Communications of the ACM*, *52*(11), 60–67.

Saeli, M., Perrenet, J., Jochems, W. M. G., Zwaneveld, B. & others. (2011). Teaching Programming in Secondary School: A Pedagogical Content Knowledge Perspective. *Informatics in Education-An International Journal*, (vol. 10-1), 73–88.

Soloway, E. (1993). Should We Teach Students to Program? *Communications of the ACM*, *36*(10), 21–25.

Szlávi, P. & Zsakó, L. (2006). Programming versus Application. In *International Conference on Informatics in Secondary Schools-Evolution and Perspectives* (pp. 48–58).

Team, R. C. (2000). *R Language Definition*. Vienna, Austria: R Foundation for Statistical Computing.

Van Rossum, G. & Drake, F. L. (2003). *Python Language Reference Manual*. Network Theory.

Wang, H. Y., Huang, I. & Hwang, G. J. (2016). Comparison of the Effects of Project-based Computer Programming Activities between Mathematics-gifted Students and Average Students. *Journal of Computers in Education*, *3*(1), 33–45.

Wilkes, M. V. (1956). *Automatic Digital Computers*. Methuen & Co.