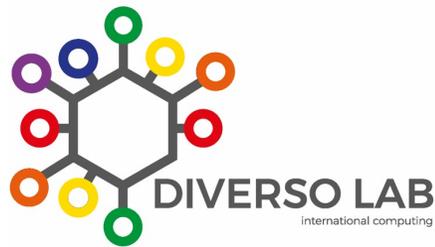
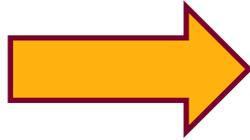


Retos sobre el pensamiento computacional

Jesús Moreno León



Índice



1. Algo de historia

2. Programamos

3. KGBL3

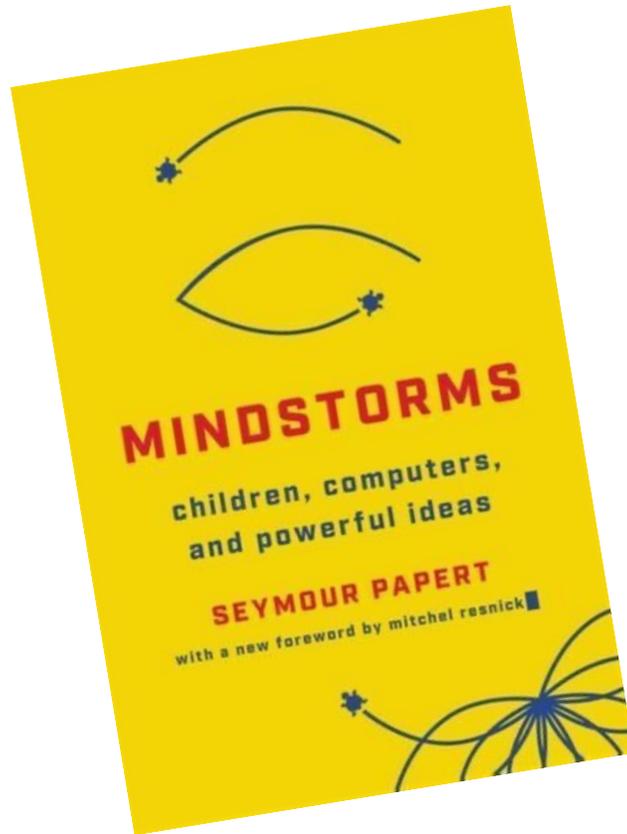
4. Ministerio de Educación

5. Retos futuros

1960s



1980



In most cases, although the experiments have been interesting and exciting, they have failed to make it because they were too primitive. Their computers simply did not have the power needed for the most engaging and shareable kinds of activities. Their visions of how to integrate **computational thinking** into everyday life was insufficiently developed. But there will be more tries, and more and more. And eventually, somewhere, all the pieces will come together and it will “catch.” One can be confident of this because such attempts will not be isolated experiments operated by researchers who may run out of funds or simply become disillusioned and quit. They will be manifestations of a social movement of people interested in personal computation, interested in their own children, and interested in education.

Computational Thinking

It represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use.



Computational thinking builds on the power and limits of computing processes, whether they are executed by a human or by a machine. Computational methods and models give us the courage to solve problems and design systems that no one of us would be capable of tackling alone. Computational thinking confronts the riddle of machine intelligence: What can humans do better than computers? and What can computers do better than humans? Most fundamentally it addresses the question: What is computable? Today, we know only parts of the answers to such questions.

Computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability. Just as the printing press facilitated the spread of the three Rs, what is appropriately incestuous about this vision is that computing and computers facilitate the spread of computational thinking.

Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science.

Having to solve a particular problem, we might ask: How difficult is it to solve? and What's the best way to solve it? Computer science rests on solid theoretical underpinnings to answer such questions pre-

cisely. Stating the difficulty of a problem accounts for the underlying power of the machine—the computing device that will run the solution. We must consider the machine's instruction set, its resource constraints, and its operating environment.

In solving a problem efficiently, we might further ask whether an approximate solution is good enough, whether we can use randomization to our advantage, and whether false positives or false negatives are allowed. Computational thinking is reformulating a seemingly difficult problem into one we know how to solve, perhaps by reduction, embedding, transformation, or simulation.

Computational thinking is thinking recursively. It is parallel processing. It is interpreting code as data and data as code. It is type checking as the generalization of dimensional analysis. It is recognizing both the virtues and the dangers of aliasing, or giving someone or something more than one name. It is recognizing both the cost and power of indirect addressing and procedure call. It is judging a program not just for correctness and efficiency but for aesthetics, and a system's design for simplicity and elegance.

Computational thinking is using abstraction and decomposition when attacking a large complex task or designing a large complex system. It is separation of concerns. It is choosing an appropriate representation for a problem or modeling the relevant aspects of a problem to make it tractable. It is using invariants to describe a system's behavior succinctly and declaratively. It is having the confidence we can safely use, modify, and influence a large complex system without understanding its every detail. It is

2011



2012

Michael Gove to scrap 'boring' IT lessons

Schools to be given freedom to run cutting-edge computer classes under plans for open source curriculum



Ministers are keen to see universities and businesses creating a new computer science GCSE.
Photograph: Frank Baron

The teaching of computer science in school is to be dramatically overhauled, with the existing programme of study scrapped to make way for new lessons designed by industry and universities, [Michael Gove](#) will announce on Wednesday.

In a speech, the education secretary will say the existing curriculum in Information and Communication Technology (ICT) has left children "bored out of their minds being taught how to use Word and Excel by bored teachers".

Instead he will, in effect, create an "open source" curriculum in computer science by giving schools the freedom to use teaching resources designed with input from leading employers and academics, in changes that will come into effect this September.

Índice

1. Algo de historia



2. Programamos

3. KGBL3

4. Ministerio de Educación

5. Retos futuros

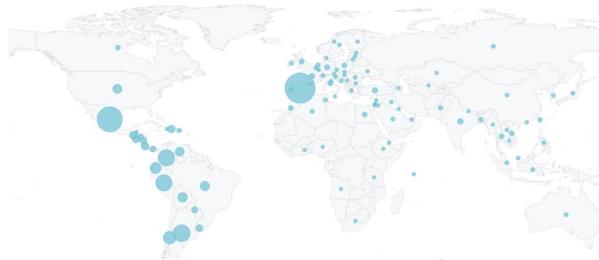
Programamos



Healing with Code

The Spanish non-profit Programamos is bringing Scratch to children in hospitals

The Scratch Team · Follow
Published in The Scratch Team Blog · 6 min read · May 19, 2017



Total de estudiantes
53.524

Valoraciones
4096



Visualizaciones
3,1 M

Tiempo de visualización (horas)
120,7 mil



Administraciones públicas: “¡evidencias!”



Índice

1. Algo de historia

2. Programamos

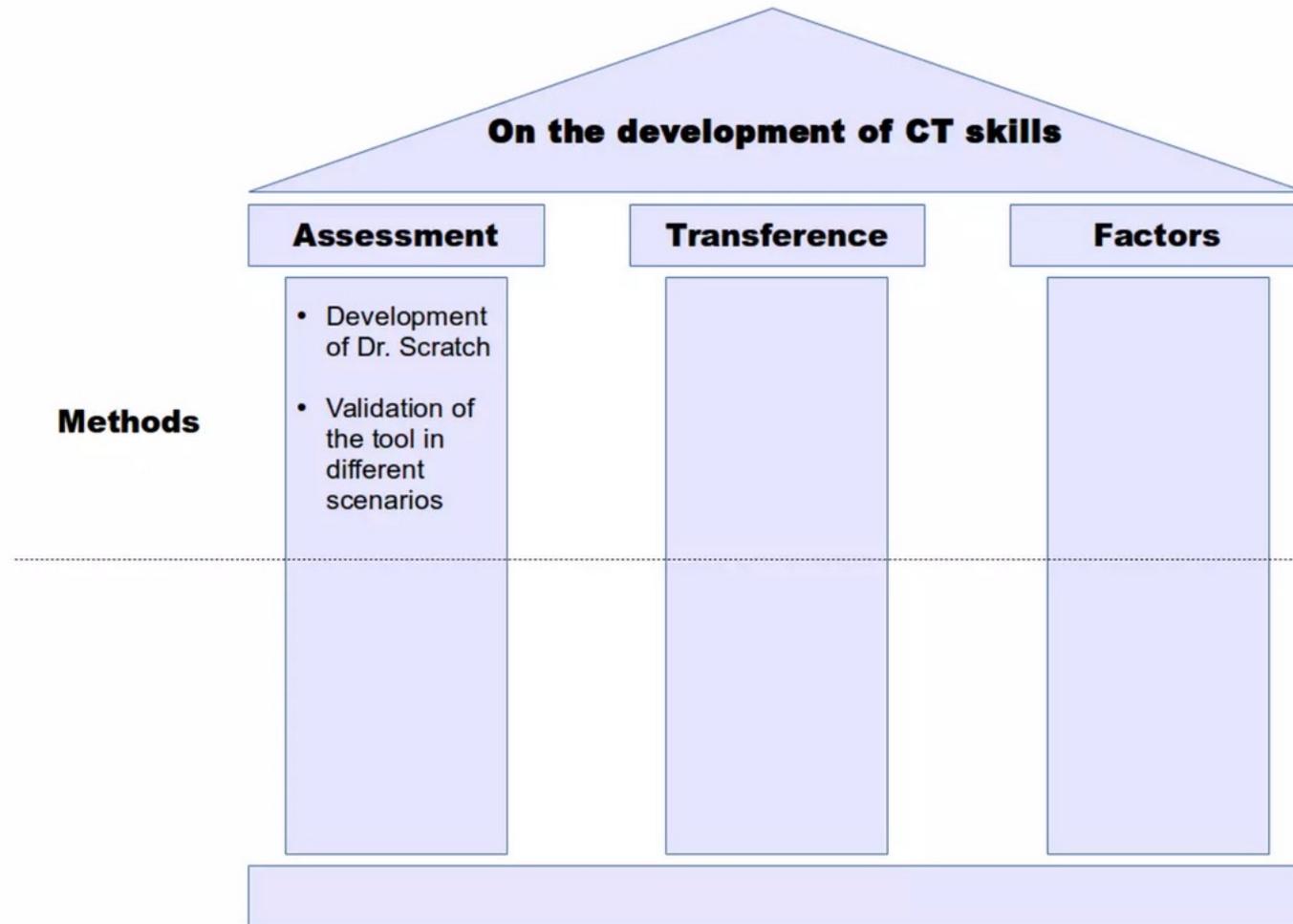


3. KGBL3

4. Ministerio de Educación

5. Retos futuros

Investigación KGBL3



Investigación KGBL3: evaluación

The screenshot shows the GitHub repository for 'hairball'. The repository is public and has 26 stars and 26 forks. It is currently on the 'master' branch with 1 branch and 9 tags. The repository contains several files, including 'hairball', 'test', '.gitignore', 'LICENSE.txt', 'MANIFEST.in', 'NOTES', 'README.md', 'lint.sh', 'setup.cfg', and 'setup.py'. The repository is described as a plugin-able framework useful for static analysis of Scratch projects. The repository is licensed under BSD-2-Clause license and has 26 stars and 8 watchers. The repository is maintained by bboe (Bryce Boe) and has 4 contributors. The repository is written in Python (99.2%) and Shell (0.8%).

hairball Public

1 Branch 9 Tags

Go to file

Add file Code

About

Hairball is a plugin-able framework useful for static analysis of Scratch projects.

Readme

BSD-2-Clause license

Activity

Custom properties

26 stars

8 watching

26 forks

Report repository

Releases

9 tags

Create a new release

Packages

No packages published

Publish your first package

Contributors 4

- charlottehill Charlotte Hill
- bboe Bryce Boe
- pconrad Phill Conrad
- jemole Jesús Moreno León

Languages

- Python 99.2%
- Shell 0.8%

hairball

Hairball is a plugin-able framework useful for static analysis of Scratch projects.

The paper and presentation slides for Hairball can be found at: <http://cs.ucsb.edu/~bboe/p/cv#sigcse13>

A Hairball demo web service is running and available at: <http://hairball.herokuapp.com>

Hairball installation

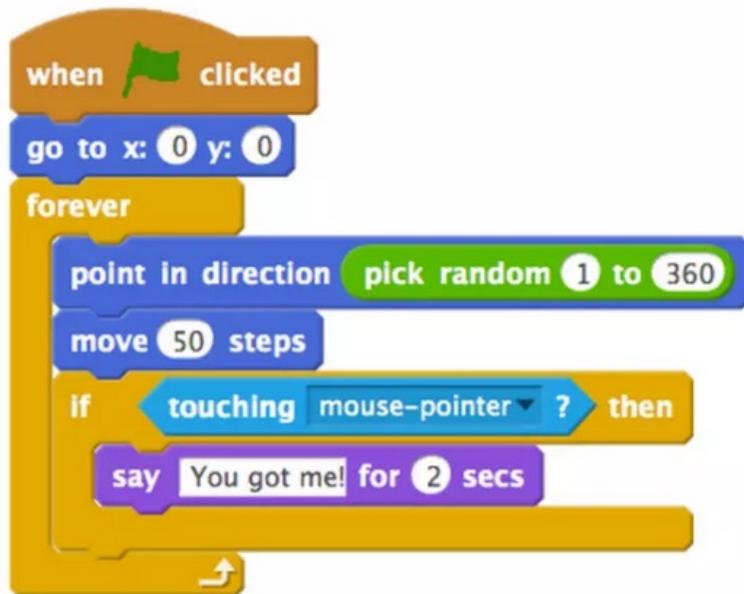
With a proper python environment (one which has `pip` available), installation is as simple as `pip install hairball`. `easy_install` can also be used via `easy_install hairball`.

To install from source, first checkout this project and then navigate your command-line interface to the outer

The image shows two Scratch code blocks for a 'when clicked' event. The first block is a 'hide' block followed by a 'forever' loop containing an 'if touching Sprite14?' block with a 'then' block containing 'go to Sprite12' and 'show'. The second block is a 'hide' block followed by a 'forever' loop containing an 'if touching Sprite14?' block with a 'then' block containing 'go to Sprite' and 'show'. Below the code blocks is a 'Sprites' panel showing a grid of 41 sprites, including various characters and objects.

The image shows two Scratch code blocks for 'when I receive message' events. The first block is 'when I receive message1' followed by 'set rotation style left-right', 'repeat 10' loop containing 'move 2 steps', 'next costume', 'if on edge, bounce', and 'wait 1 secs'. The second block is 'when I receive message2' followed by 'set rotation style left-right', 'repeat 5' loop containing 'move 4 steps', 'next costume', 'if on edge, bounce', and 'wait 0.5 secs'.

Investigación KGBL3: evaluación



CT dimension	Basic	Developing	Proficient
Data representation	modifiers of sprites properties	operations on vars	operations on lists
Logical Thinking	if	if else	logic operations
User interactivity	green flag	key pressed, sprite clicked, ask and wait, mouse blocks	when %s is >%s, video, audio
Algorithmic notions of flow control	sequence of blocks	repeat, forever	repeat until
Abstraction and problem decomposi- tion	more than one script and more than one sprite	def block	when I start as clone
Parallelism	Two scripts on green flag	Two scripts on key pressed, two scripts on sprite clicked on the same sprite	Two scripts on when I receive message, two scripts when %s is >%s, two scripts on when backdrop change to
Synchronization	wait	Broadcast, when I re- ceive message, stop all, stop program, stop programs sprite	wait until, when backdrop change to, broadcast and wait

Investigación KGBL3: evaluación

```
chen@th:~$ hairball -p mastery.Mastery Catch\ me\ if\ you\ can.sb2
Catch me if you can.sb2
{'Abstraction': 0, 'Parallelization': 0, 'Logic': 1, 'Synchronization': 0, 'FlowControl': 2, 'UserInteractivity': 2, 'DataRepresentation': 1}
Total mastery points: 6/21
Average mastery points: 0.86/3
Overall programming competence: Basic
```

Total Physical Source Lines of Code	1,809
Dominant language	Python
Schedule Estimate, Months	4.42
Estimated Average Number of Developers	1.01
Total Estimated Cost to Develop	\$50,345

Table: Estimated effort for the development of the Hairball fork (using the *basic* COCOMO estimation model [Boehm et al., 2000]).

Investigación KGBL3: evaluación

Doctor Scratch (alpha version)

Dr. Scratch (alpha version)

Analyze your Scratch projects here!

Welcome to the Dr. Scratch website, an analytical tool that evaluates your Scratch projects in a variety of computational areas. We provide feedback on aspects such as abstraction, logical thinking, synchronization, parallelization, flow control, user interactivity and data representation.

This analyzer is a helpful tool to evaluate your own projects, or those of your Scratch students.



Help

Slides used in Scratch Conference, MIT 2014



Contact

If you have any questions, comments or ideas to improve Dr. Scratch, please feel free to contact us:

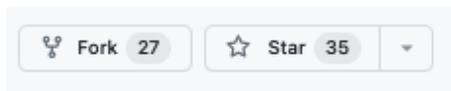
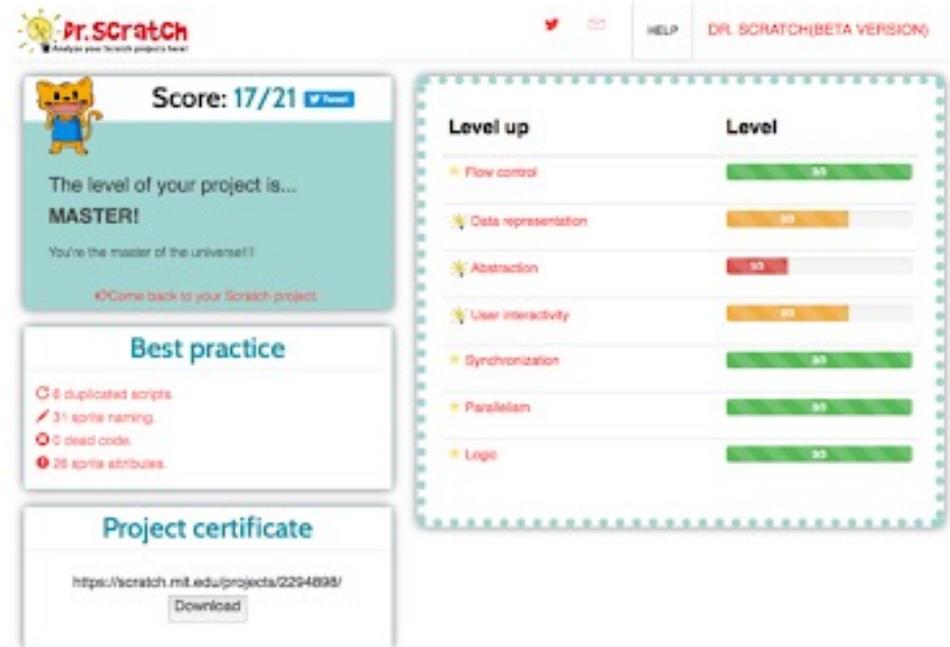
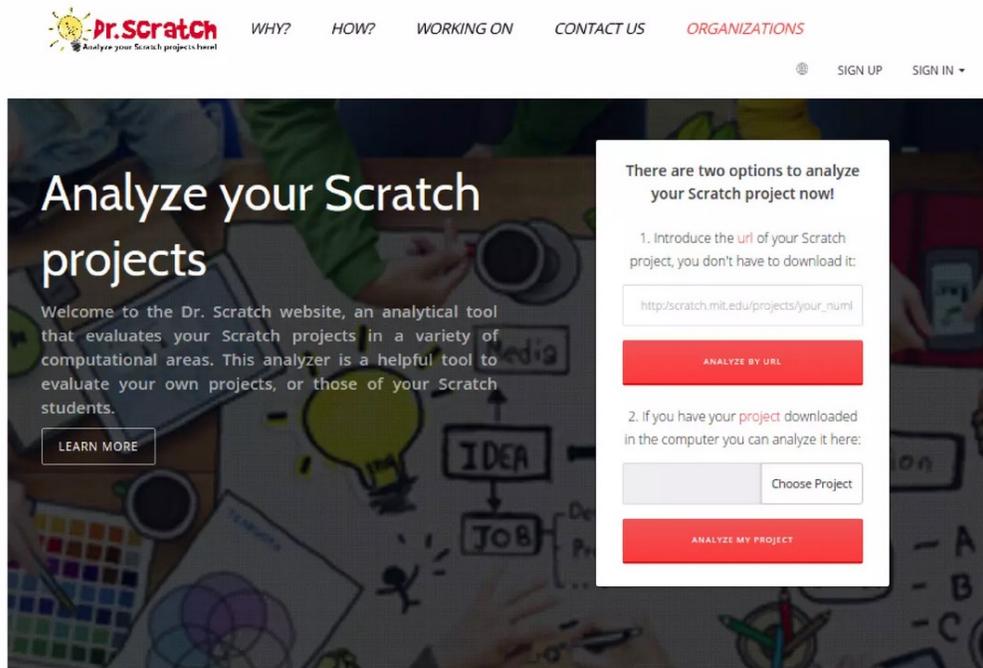
jesus.moreno (at) programamos.es



Community

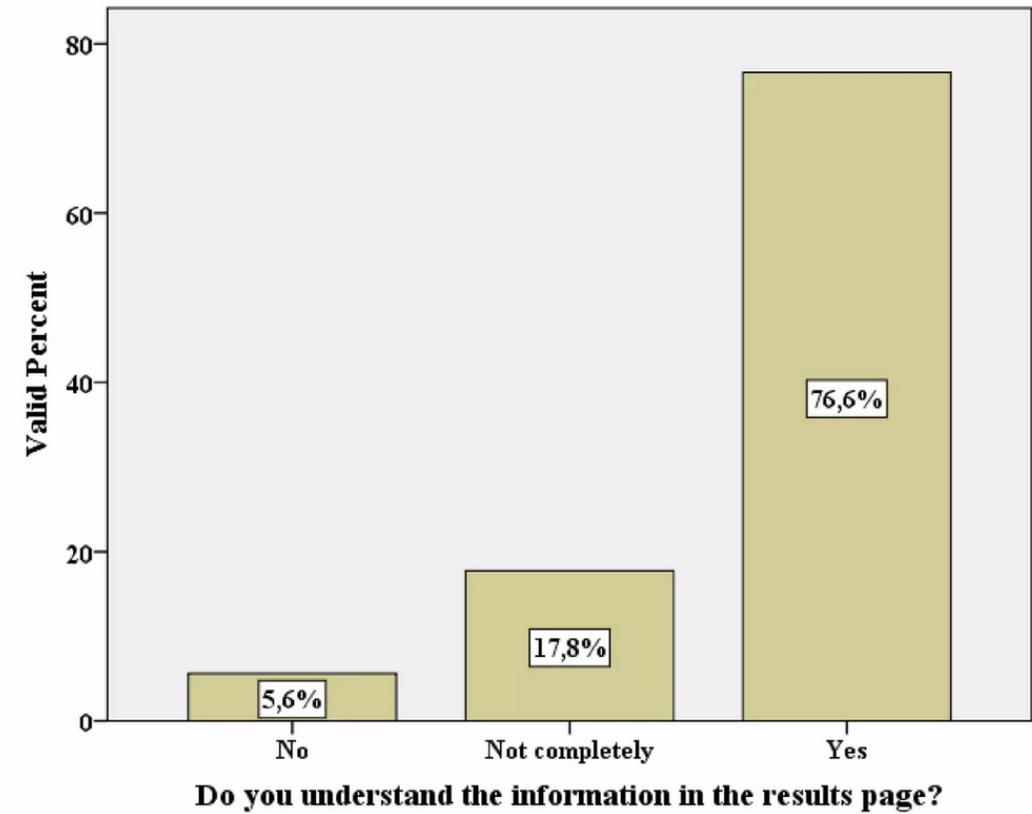
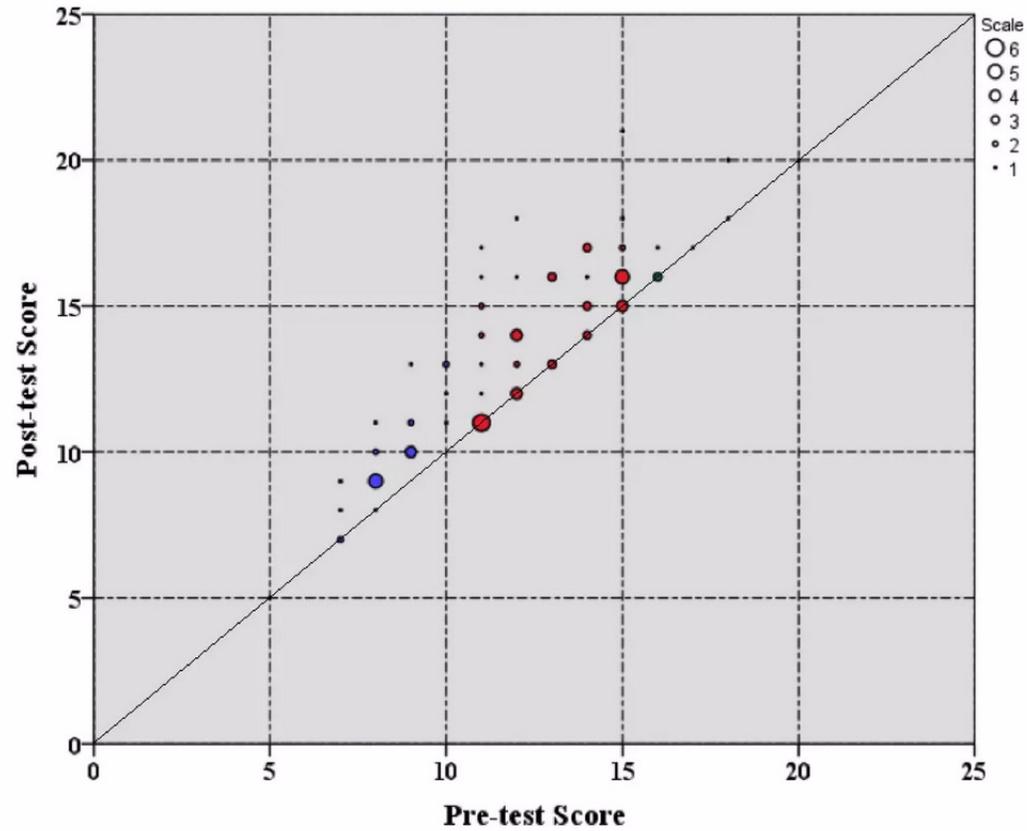
Join us! Meet other users and share experiences

Investigación KGBL3: evaluación

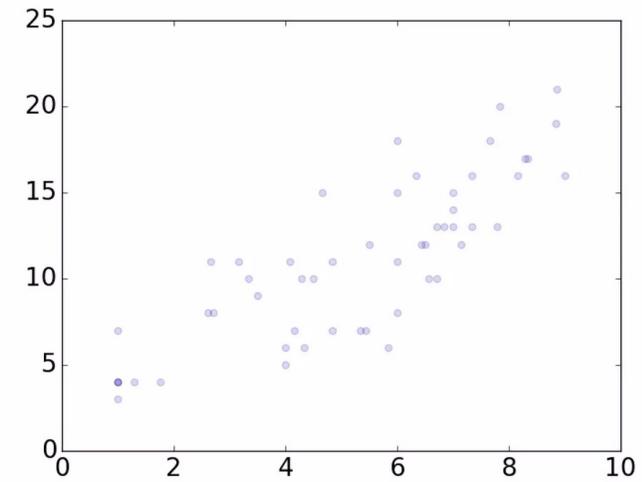


~ 100.000 proyectos analizados/año

Investigación KGBL3: evaluación

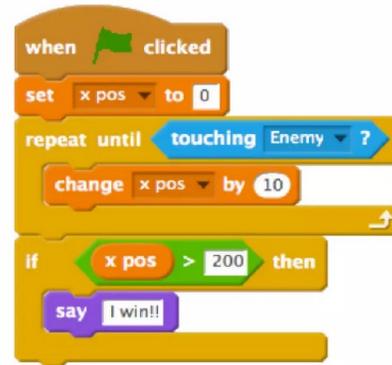


Investigación KGBL3: evaluación

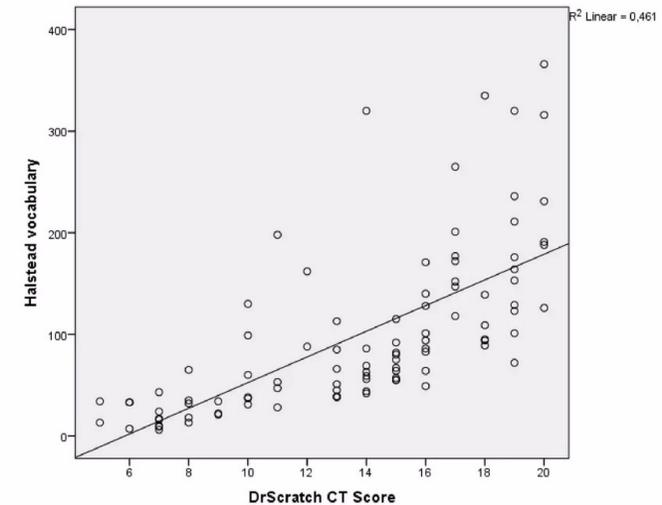
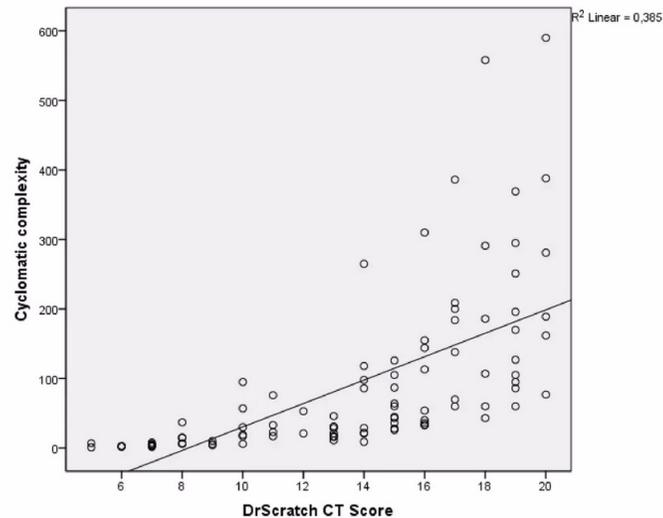


Scatter plot for experts evaluation (x-axis) and Dr. Scratch assessment (y-axis).

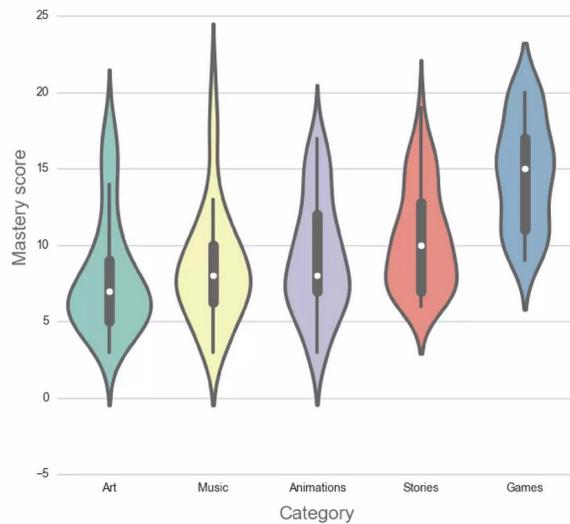
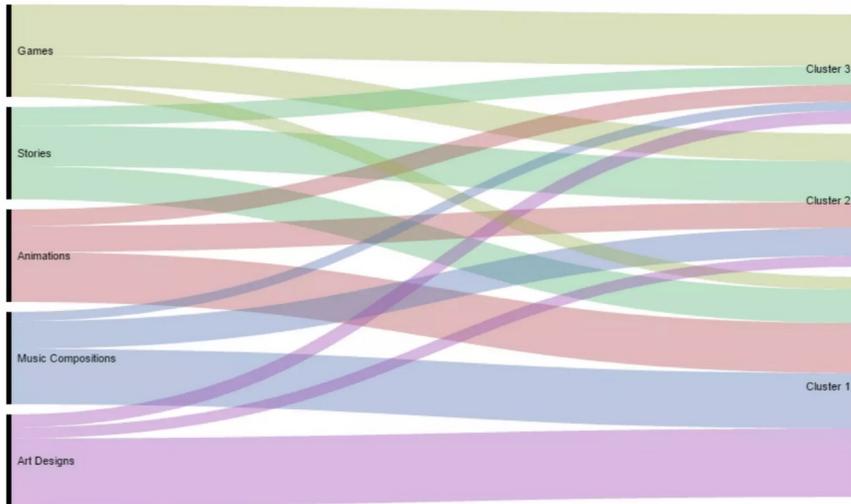
Investigación KGBL3: evaluación



Metric	Value
Cyclomatic complexity	3
Vocabulary	14
Length	14
Volume	66.42
Difficulty	53.30
Effort	293.86



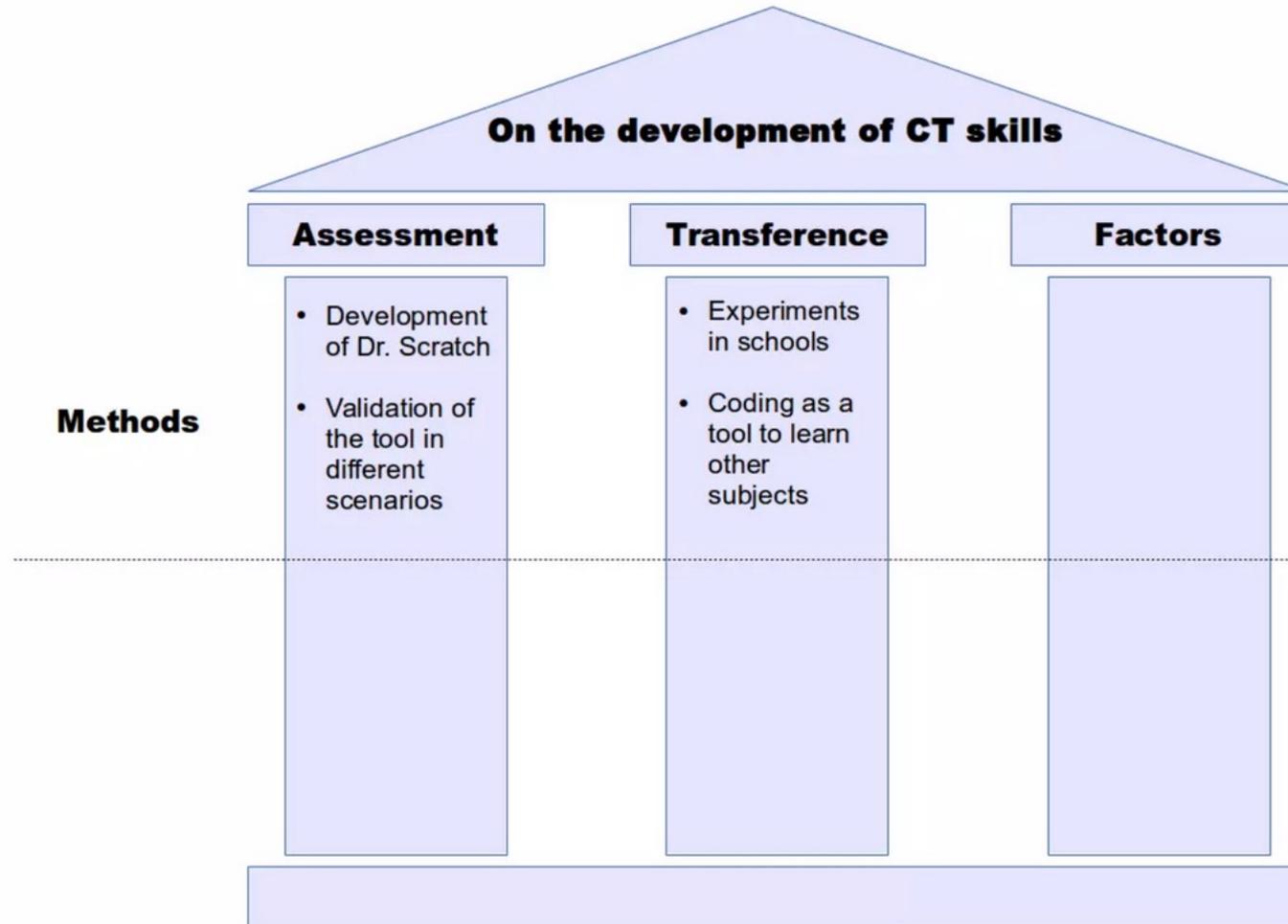
Investigación KGBL3: evaluación



The image shows two screenshots of the CT Paths website. The top screenshot displays the main interface with the title "CT Paths" and the subtitle "Create your own learning path!". It includes a dropdown menu for "What mastery level would you like to get?" set to "16", and radio buttons for "What kind of projects would you like to work?" with "Games" selected. A "Get some new examples" button is visible. The bottom screenshot shows the "Get some inspiration!" section, which includes a table of project URLs and their associated mastery scores and other metrics.

URL	Mastery	Abstraction	Parameters	Logic	Synchronization	Flow Control	Data Representation
https://scratch.mit.edu/projects/124876443/	16	1	3	0	3	2	2
https://scratch.mit.edu/projects/121898238/	15	1	3	0	3	2	2
https://scratch.mit.edu/projects/122980713/	16	1	3	3	3	2	2
https://scratch.mit.edu/projects/123062727/	15	1	3	3	3	2	2
https://scratch.mit.edu/projects/124838474/	15	1	3	0	3	2	2

Investigación KGBL3: transferencia



Investigación KGBL3: transferencia

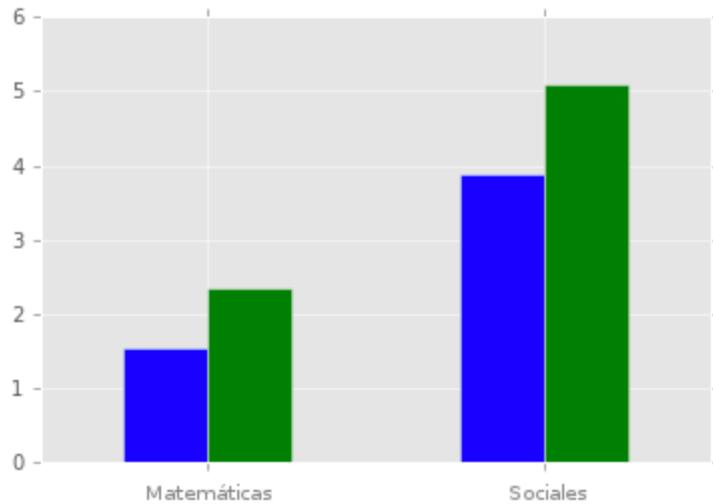


Figura 2: Mejora experimentada entre la prueba inicial y final: a la izquierda, los resultados para la clase de Matemáticas; a la derecha, los de la clase de Ciencias Sociales. Los grupos de control se muestran en color azul; los grupos experimentales en color verde.

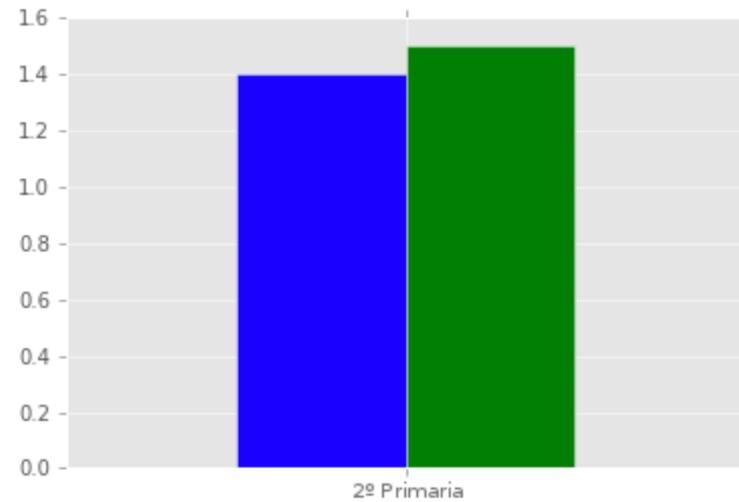


Figura 3: Mejora experimentada entre la prueba inicial y final: resultados para la clase de 2º de Primaria. El grupo de control se muestra en color azul; el grupo experimental en color verde.

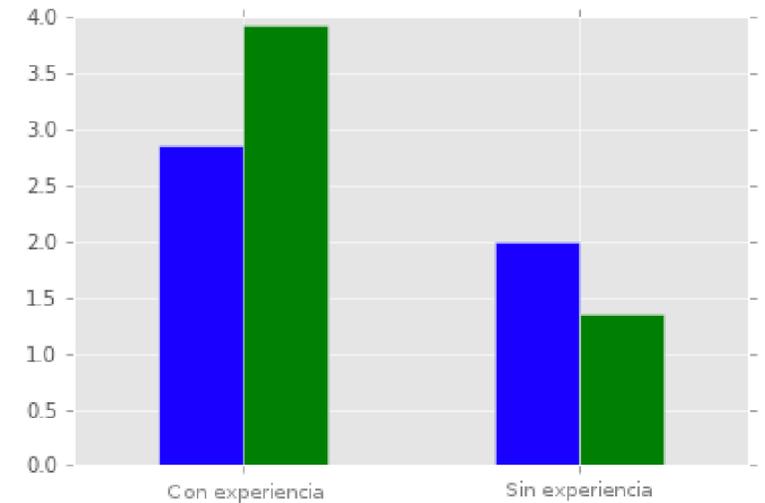
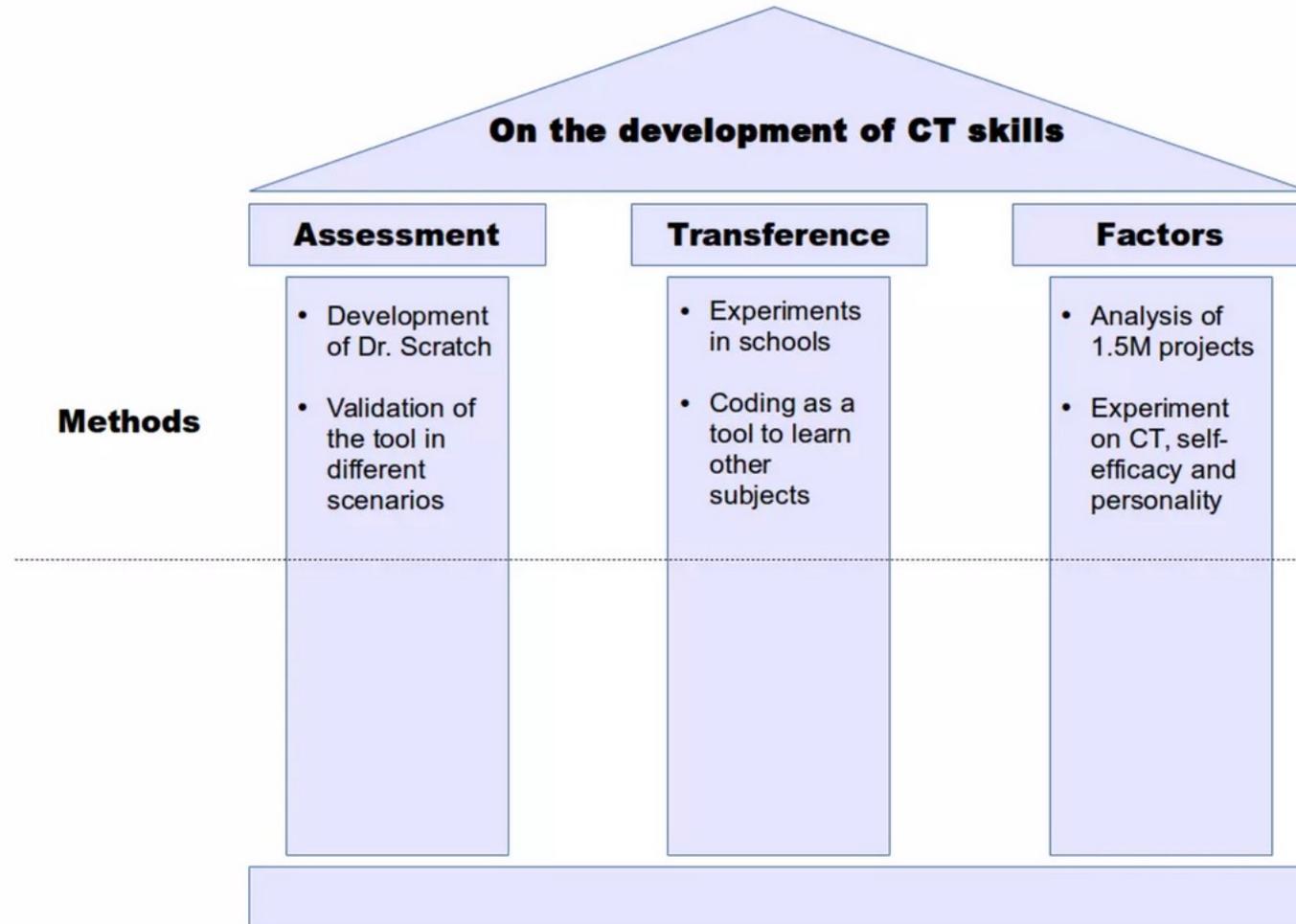
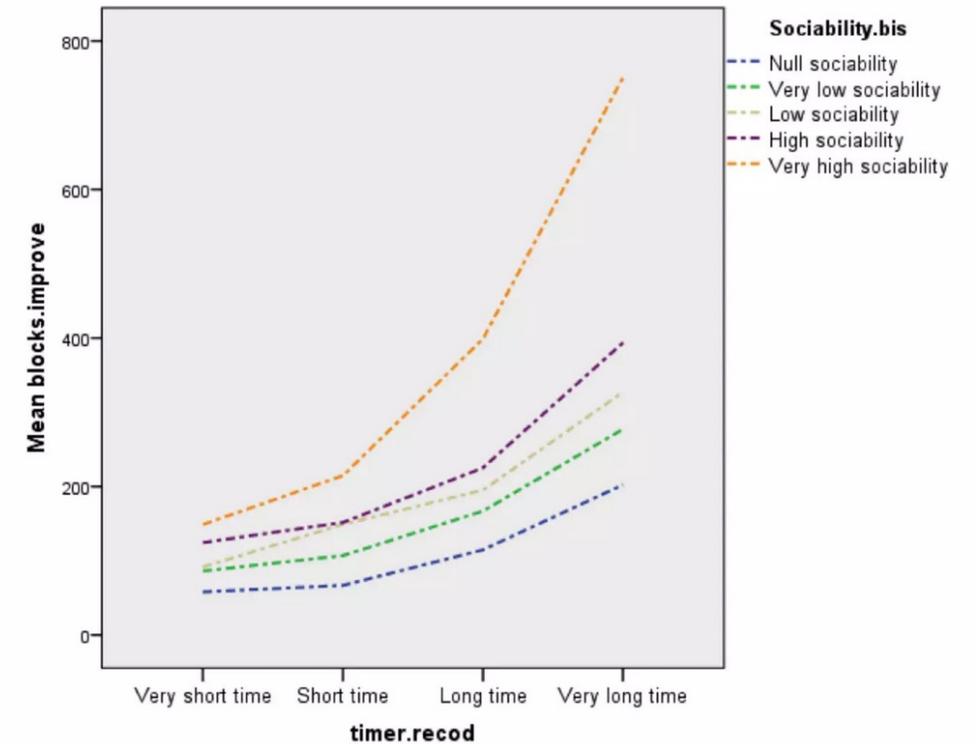
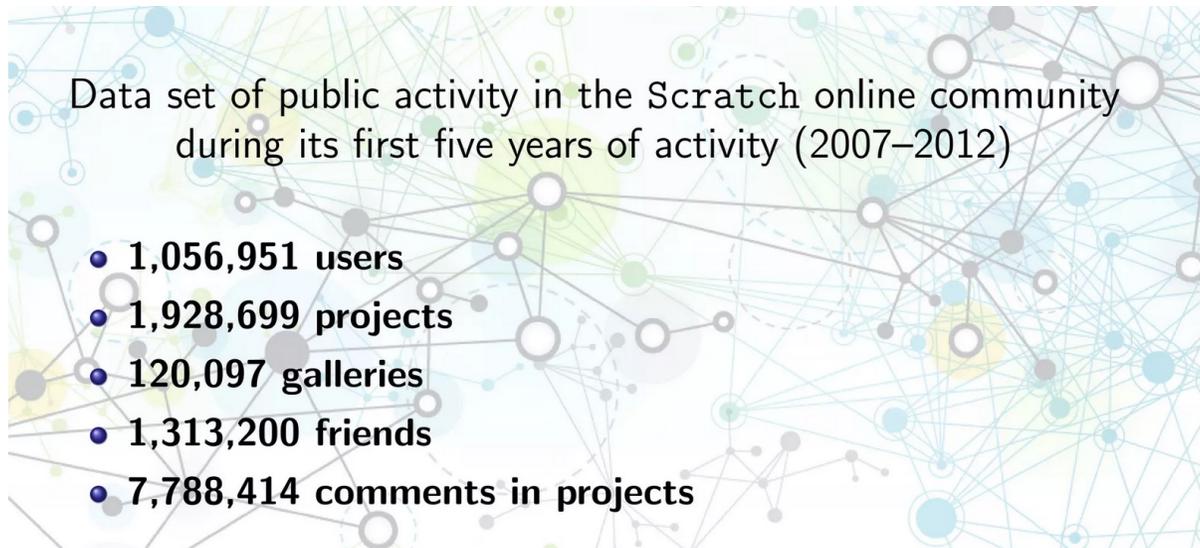


Figura 4: Mejora experimentada entre la prueba inicial y final: a la izquierda, los resultados para la clase del docente con experiencia previa en programación; a la derecha, los de la clase del docente sin experiencia previa. Los grupos de control se muestran en color azul; los grupos experimentales en color verde.

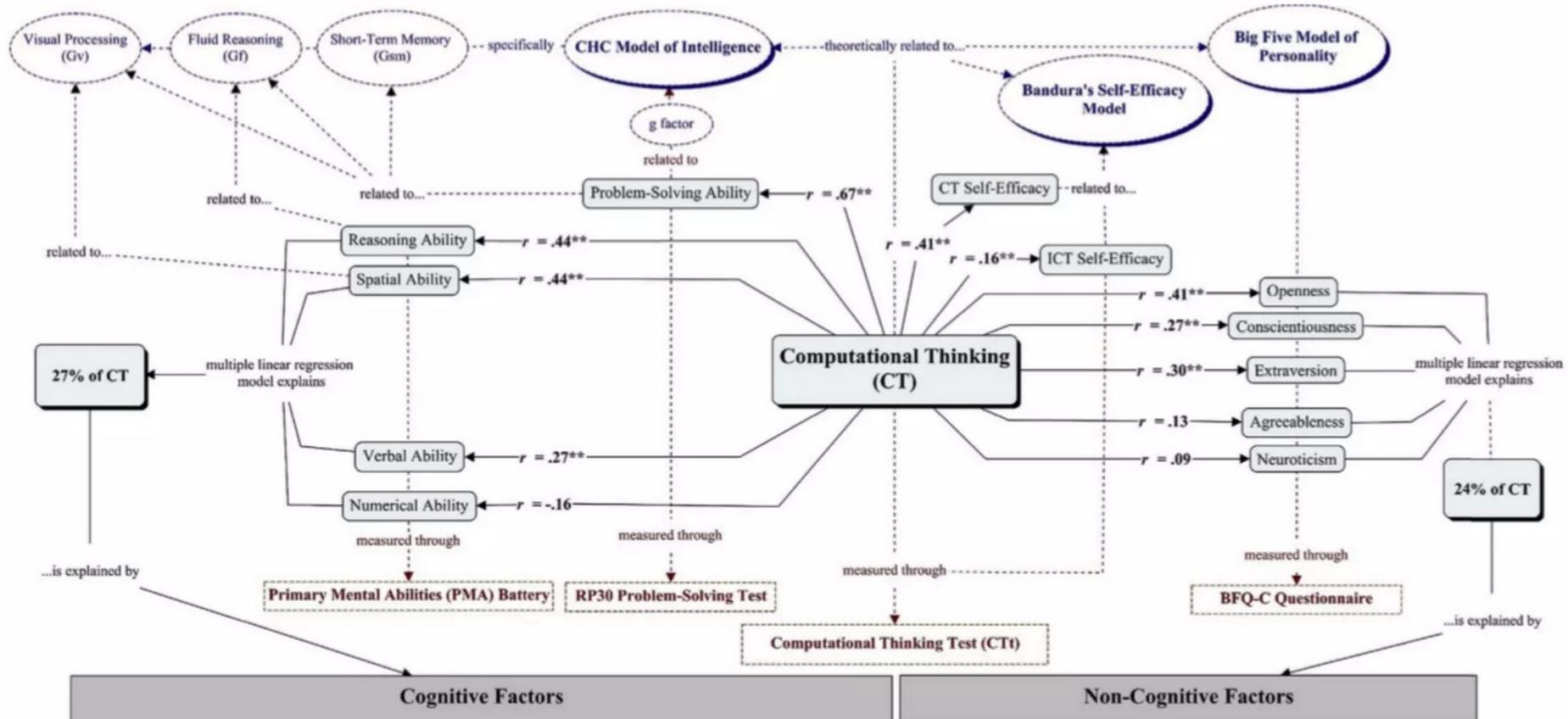
Investigación KGBL3: factores



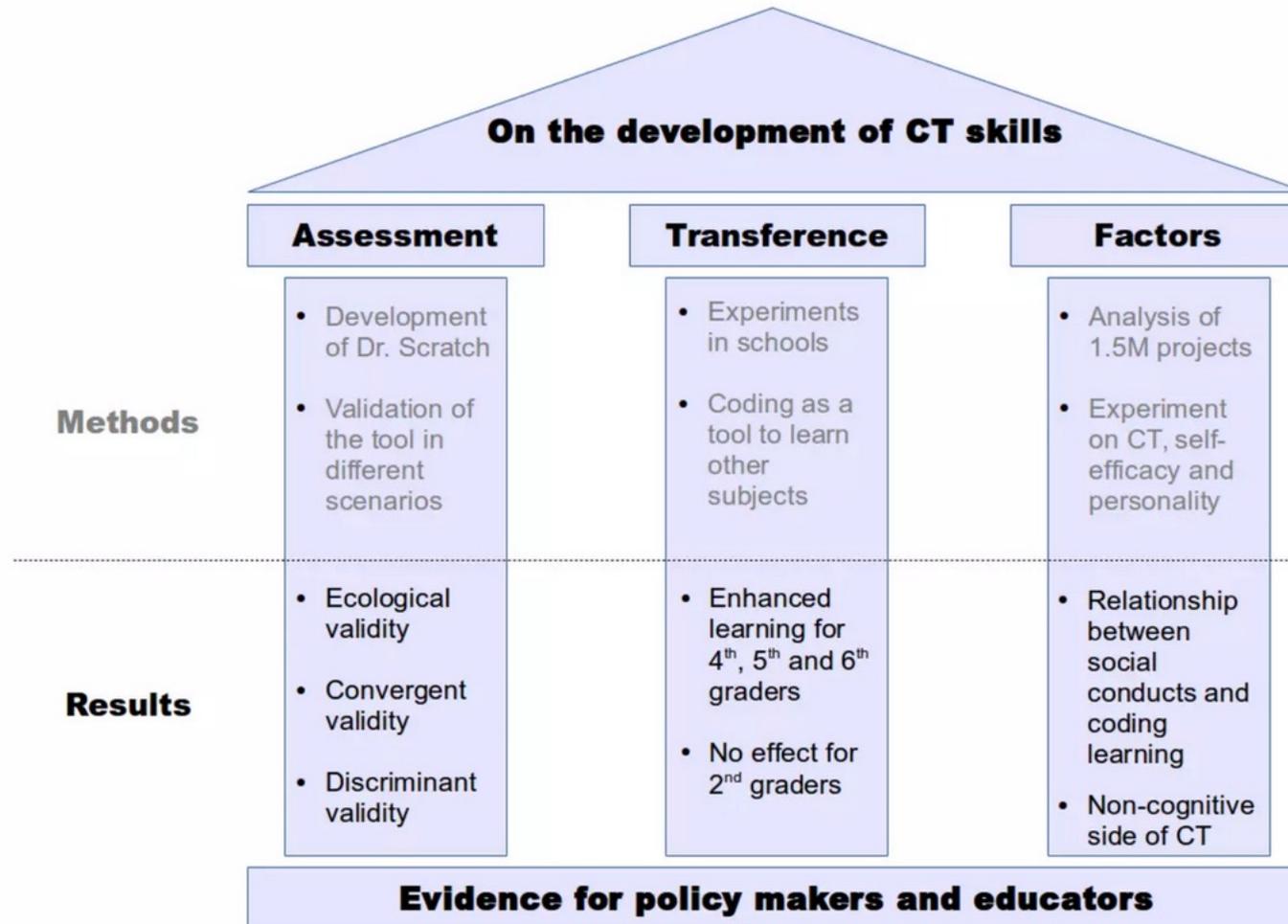
Investigación KGBL3: factores



Investigación KGBL3: factores



Investigación KGBL3

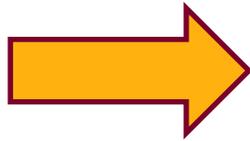


Índice

1. Algo de historia

2. Programamos

3. KGBL3



4. Ministerio de Educación

5. Retos futuros

Ministerio de Educación



Figura 2.2: Comunidades Autónomas que han incluido nuevas asignaturas o contenidos sobre programación, robótica y pensamiento computacional. En color azul aparecen aquellas que lo han hecho en Primaria; verde, en Secundaria; morado, en ambos niveles educativos. Las Comunidades Autónomas que no participan en el estudio aparecen en gris.

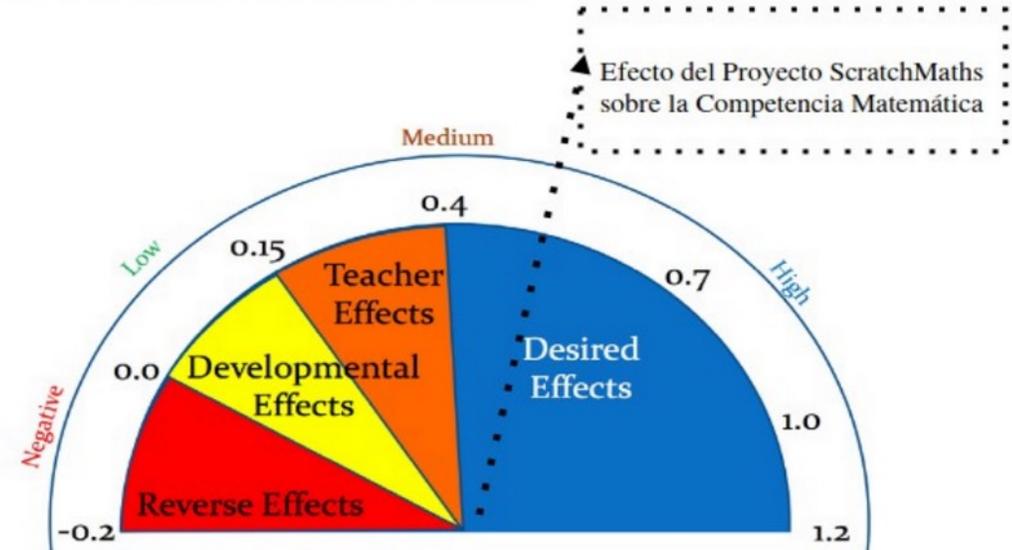
Ministerio de Educación



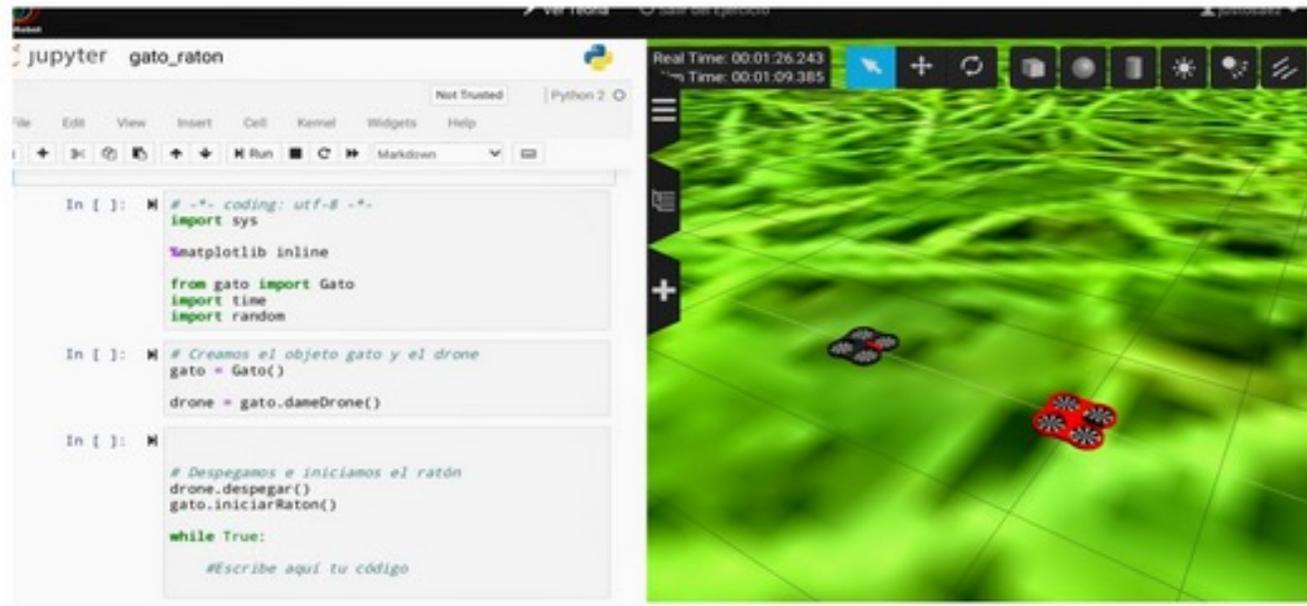
Gráfico 4. Fases y cronograma de la investigación

Diciembre	Enero	Febrero	Marzo	Abril	Mayo	Junio
Formación del profesorado						
		Pre-test				
			Implementación en el aula			
						Post-test

Gráfico 12. "Barómetro de la influencia" (Hattie, 2009).



Ministerio de Educación



Ministerio de Educación



escuela de pensamiento
computacional e
inteligencia artificial



Nivel I Infantil y primaria (1º, 2º y 3º)

En este nivel se propone trabajar haciendo uso de actividades unplugged (desenchufadas o desconectadas), que hacen uso de juegos de lógica, vasos, cuerdas, cartas o movimientos físicos, que se utilizan para representar y comprender diferentes conceptos relacionados con la IA, como algoritmos o representación de datos.

Nivel II Primaria (4º, 5º y 6º) y secundaria (1º y 2º)

En este nivel el alumnado debe reconocer cómo los sistemas informáticos que utilizan en su día a día hacen uso de la IA para percibir el mundo usando sensores, razonar, aprender e interactuar con humanos. Además, los estudiantes deben recapacitar sobre el impacto que la IA puede tener en la sociedad, tanto de modo positivo como negativo. Y el mejor modo de lograr estos objetivos es que construyan sus propias creaciones software, como un videojuego sencillo, que integre soluciones de IA, en especial las relacionadas con el aprendizaje automático o machine learning.

Nivel III Secundaria (3º y 4º), Bachillerato y FP

En este nivel, para alcanzar los mismos objetivos marcados para el nivel II, se realizarán proyectos de desarrollo de aplicaciones para dispositivos móviles que integren soluciones de IA.

Ministerio de Educación

	Al completar este nivel educativo el alumno o la alumna...	Áreas o materias
INFANTIL	Desarrolla, de manera progresiva, las destrezas del pensamiento computacional a través de procesos de observación y manipulación de objetos para iniciarse en la interpretación del entorno y responder de forma creativa a las situaciones y retos que se plantean.	Área 2. Descubrimiento y exploración del entorno (en el 2º ciclo)
PRIMARIA	Se inicia en el desarrollo de soluciones digitales sencillas y sostenibles (reutilización de materiales tecnológicos, programación informática por bloques, robótica educativa...) para resolver problemas concretos o retos propuestos de manera creativa, solicitando ayuda en caso necesario.	Conocimiento del medio natural, social y cultural (en los 3 ciclos) y Matemáticas (en los 3 ciclos)
E.S.O.	Desarrolla aplicaciones informáticas sencillas y soluciones tecnológicas creativas y sostenibles para resolver problemas concretos o responder a retos propuestos, mostrando interés y curiosidad por la evolución de las tecnologías digitales y por su desarrollo sostenible y uso ético.	Tecnología y Digitalización (1º, 2º y 3º), Tecnología (4º), Digitalización (4º), Biología y Geología (1º, 2º, 3º y 4º), Matemáticas (1º, 2º y 3º), Matemáticas A y B (4º), Ámbito Ciencias Aplicadas (CFGB)
BACHI-LLERATO	Desarrolla soluciones tecnológicas innovadoras y sostenibles para dar respuesta a necesidades concretas, mostrando interés y curiosidad por la evolución de las tecnologías digitales y por su desarrollo sostenible y uso ético	Tecnología e Ingeniería I y II, Biología, Geología y Ciencias Ambientales (1º), Geología y Ciencias Ambientales (2º), Matemáticas I y II, Matemáticas Aplicadas a las Ciencias Sociales I y II, Matemáticas Generales

Índice

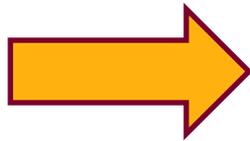
1. Algo de historia

2. Programamos

3. KGBL3

4. Ministerio de Educación

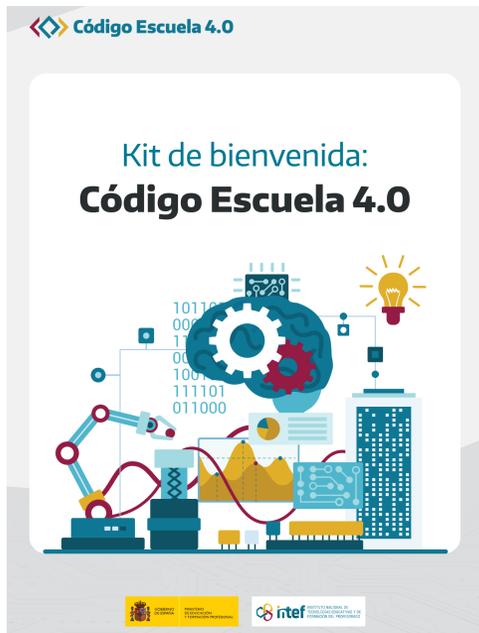
5. Retos futuros



Didáctica del pensamiento computacional

Actuación 2: Acompañamiento al profesorado

CC.AA.	N.º de alumnos 40% 1	N.º de centros 15% 2	N.º de unidades 40% 3	Dispersión población 5% 4	Ponderación (1)= 40% (2)= 15% (3)= 40% (4)= 5%	Total
Andalucía.	759.863	2.526	36.740	1.512.197	19,54	19.422.274,00
Aragón.	109.551	406	5.997	340.351	3,07	3.052.981,00
P. Asturias.	64.727	291	3.580	328.925	1,94	1.932.630,00
I. Balears.	96.865	317	5.701	450.576	2,82	2.807.267,00
Canarias.	152.805	675	7.905	951.857	4,55	4.521.579,00
Cantabria.	45.693	192	2.607	241.570	1,38	1.370.049,00
Castilla y León.	171.653	836	10.314	928.424	5,43	5.396.147,00
Castilla-La Mancha.	183.182	776	10.360	726.581	5,39	5.362.147,00
Cataluña.	671.941	2.368	31.003	1.281.193	17,10	16.993.525,00
C. Valenciana.	417.967	1.376	22.560	710.154	11,16	11.095.096,00
Extremadura.	86.776	483	5.412	427.143	2,83	2.817.328,00
Galicia.	188.477	881	10.454	1.498.134	5,97	5.937.165,00
C. Madrid.	560.503	1.304	30.448	302.046	14,06	13.980.596,00
R. Murcia.	154.920	508	7.449	359.055	3,99	3.968.071,00
La Rioja.	27.752	86	1.488	77.034	0,75	743.145,00
Totales.	3.692.675	13.025	192.018	10.135.240	100,00	99.400.000,00



- Herramientas
- Recursos
- Lenguajes
- Edades
- Enfoques
- ...

Didáctica del pensamiento computacional

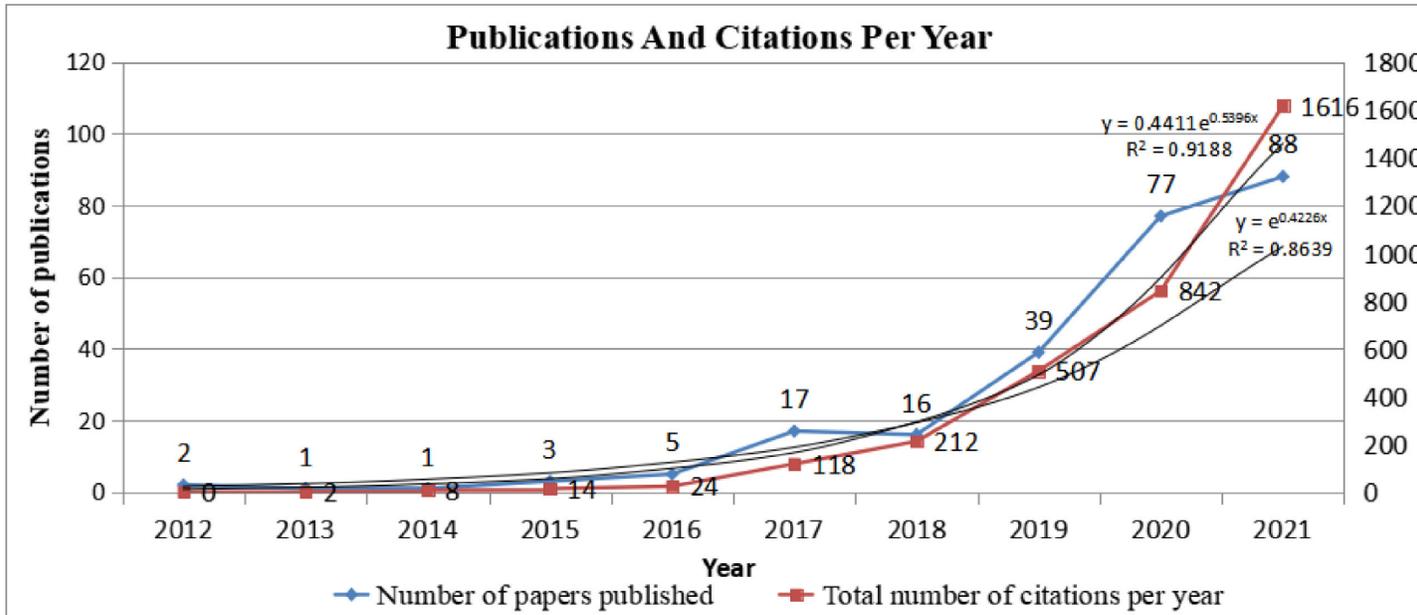


Dinamismo del pensamiento computacional



Todavía quedan muchos retos sobre los que investigar en relación con el pensamiento computacional

¿Pero esto le interesa a alguien?



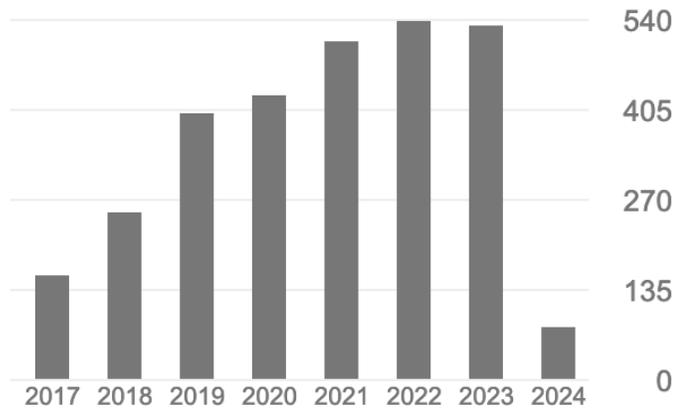
Visualising trends in computational thinking research from 2012 to 2021: A bibliometric análisis

<https://doi.org/10.1016/j.tsc.2022.101224>

¿Pero esto le interesa a alguien?

Citado por [VER TODO](#)

	Total	Desde 2019
Citas	3004	2486
Índice h	27	25
Índice i10	31	31



¡Muchas gracias!



Jesús Moreno León

Dpto. de Lenguajes y Sistemas
Informáticos

jmorenol@us.es

Retos sobre el pensamiento computacional

Jesús Moreno León

