

Katas de aprendizaje de programación con Haskell mediante serendipia



The screenshot shows a Haskell development environment with three windows:

- git diff**: Shows the differences between two versions of `MultiplySpec.hs`. The changes include adding a `spec` function and a `multiply` function.
- Kata: Multiply**: A code editor displaying the `MultiplySpec.hs` file. The code defines a module `MultiplySpec` containing a `Spec` type and several test cases for the `Multiply` function using QuickCheck properties.
- stack build**: A terminal window showing the build process of the `multiply` library. It lists dependencies and build artifacts, ending with the message "multiply> Test suite multiply-test passed".



esLibre 2022 – Viernes 24 de junio – Vigo

Reynaldo Cordero

SS.II. Universidad de Alcalá // HaskellMAD

20
22

HaskellKatas

- Programar: enseña a pensar... y a **hacer**
- **Método científico** real
- **Matemáticas** avanzadas (y prácticas)
- Exprimir la **expresividad** en el lenguaje
- Primar la capacidad de **observación**
- **Redacción** de comentarios eficaz
- Cuidado e importancia de los **detalles**
- Sentido artístico y **estilo**
- Filosofía Unix: Minimalista, modular, **hackable**
- Software **libre**
 - <https://gitlab.com/HaskellKatas/katas--proof-of-concept>

Serendipia

La **serendipia** es un descubrimiento afortunado **no planificado**.

La serendipia es un hecho común a lo largo de la historia de la invención de productos y los descubrimientos científicos. La serendipia también se considera un potencial principio de diseño para las actividades en línea que presentarían una amplia gama de información y puntos de vista, **en lugar de limitarse a reforzar la opinión de un usuario.** (Wikipedia)

La clave de la serendipia está en privilegiar la **observación**.

Kataficación

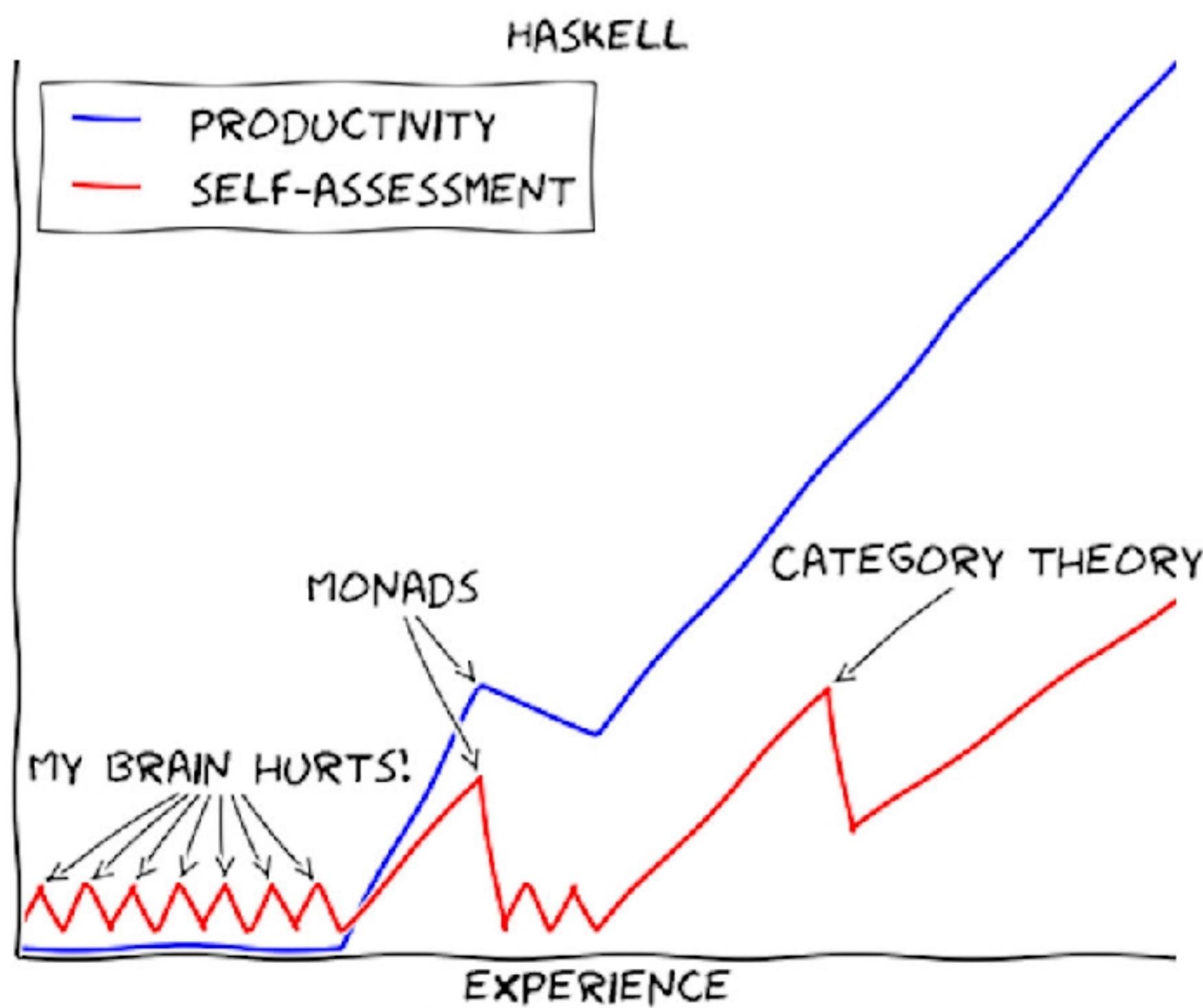
La **kataficación del código** consiste en **reformatear** y **manipular** con intención código informático en **Haskell** (*), tanto propio como de otros programadores, para encontrar un **estilo propio** que aflore las estructuras y las haga fácilmente **relacionables**, y prestas para el **uso**, estimulando la **experimentación**.

(*) Se necesita un lenguaje informático de propósito **general**, **conciso**, **limpio**, fácil de **leer** (mínimo *boilerplate*) diseñado para la **enseñanza**, la **investigación** y las aplicaciones **industriales**, de tipo **funcional puro**, **tipado fuerte** y **estático**, **inferencia** de tipos, **clases** de tipos (sobrecarga), transparencia **referencial**, evaluación **perezosa** (para hacerlo **simple**, **matemático** y reducir las categorías de **errores** al mínimo)

Entrena

LEARNING CURVE "STUDYING APPROACH"

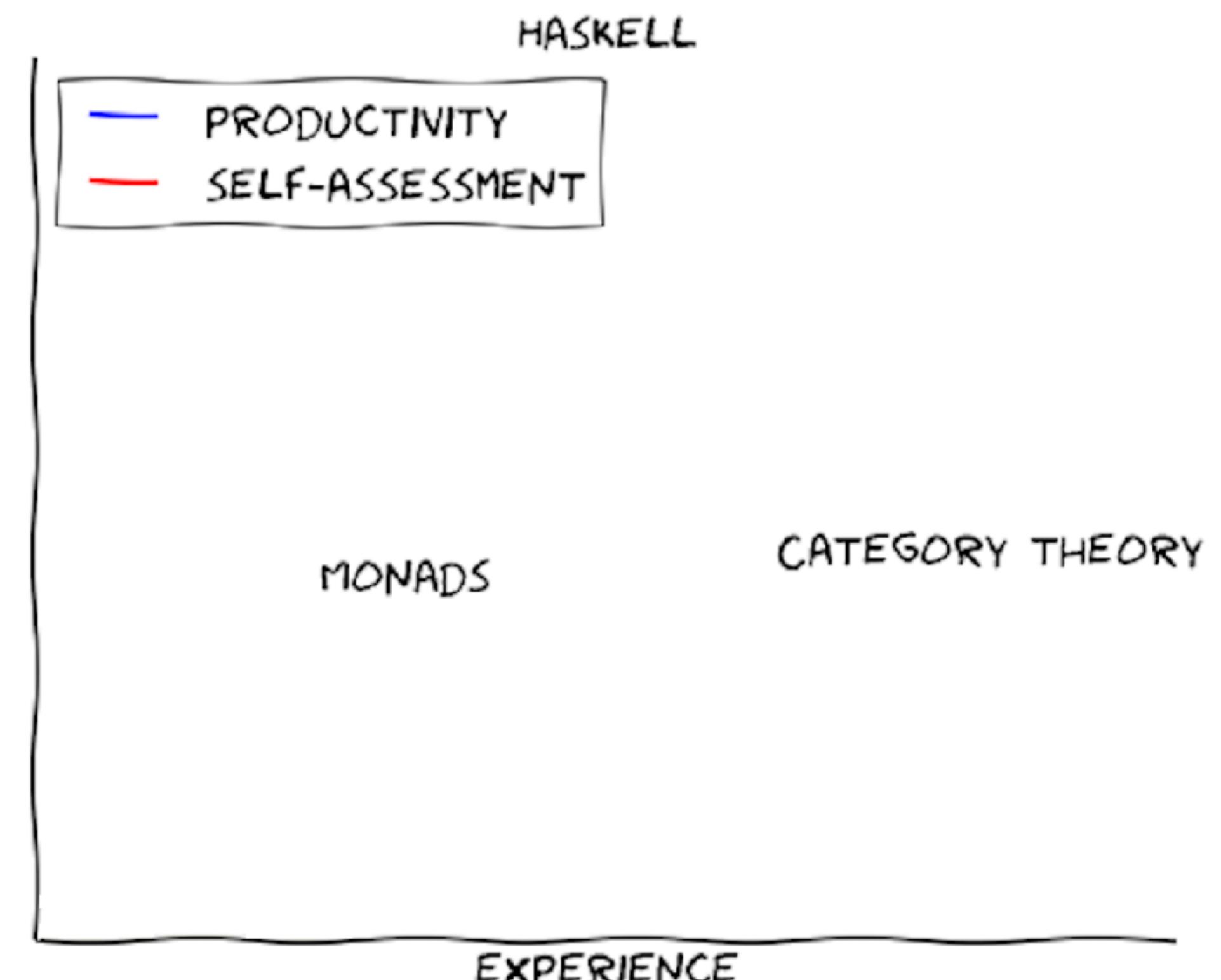
BOOKS / TUTORIALS / EXERCISES / TEACHERS



https://github.com/Dobiasd/articles/blob/master/programming_language_learning_curves.md

LEARNING CURVE "KATAFICATION-GYM"

EXPRESSIVE-ARTISTIC FORMATTING / FAST REFACTORING
EXPERIMENTS ON THE FLY



¡Todo tuyo!



Katas -proof of concept

<https://gitlab.com/HaskellKatas/katas--proof-of-concept>



Kata Library: Haskell Practice

<https://www.codewars.com/kata/search/haskell?q=&&beta=false>



Kata Training: Multiply

<https://www.codewars.com/kata/50654ddff44f800200000004/train/haskell>

Preguntas

Katas -proof of concept

<https://gitlab.com/HaskellKatas/katas--proof-of-concept>

Reynaldo Cordero

<https://matrix.to/#/@naldoco:matrix.org>

HaskellMAD

<https://www.meetup.com/haskell-mad/>

