Mejor código, sin esfuerzos, sin siquiera IA

Maëlle Salmon, Hugo Gruson, Etienne Bacher

Palabras clave: estilo de codigo, buenas practicas, automatización

Abstract

Estamos viviendo una revolución de la programación, con la democratización de la inteligencia artificial…pero también con la creación y mejora de herramientas más de la vieja escuela para mejorar su código: soluciones que son locales, gratuitas, deterministas. En esta charla os presentaré Air, una herramienta para formatear el código R de manera automática y quasi instantánea; lintr, un paquete R que siempre detecta más motivos para mejorar el código; flir, un paquete R que para una parte de los chequeos de lintr no solo los detecta más rápido sino también los repara de forma automática. Esas tres herramientas pueden utilizarse de vez en cuando o de forma automática. Con estas tres herramientas maravillosas, puedes mejorar sin esfuerzo tu código, el código de tus compañeras y compañeros… Y el código que te propone una IA.

Air, para formatear el código R

Leer código con espacios, indentaciones y nuevas líneas donde hace falta es muy agradable y más fácil. Aún mejor es obtener tal código sin tener que pensarlo y sin esperar. Es lo que ofrece Air, una nueva herramienta desarrollada en Posit por Lionel Henry y Davis Vaughan. Una vez que has instalado y configurado Air en RStudio, Positron, VSCode, o hasta GitHub Actions, Air se cargará de la apariencia de tu código. Por ejemplo, Air puede arreglar el código cada vez que guardas un archivo. Os explicaré qué es Air y las formas más usuales de usarlo.

lintr, para mejorar el código R

Nunca usar any(is.na(x)) en vez de anyNA(x), usar !all(x) en vez de any(!x) que es menos legible… Hay muchas reglas que distinguen el código medio del bueno. Puede ser difícil interiorizarlas y tenerlas en cuenta todo el tiempo. Por suerte no hace falta. Con lintr, un paquete creado por Jim Hester y mantenido por Michael Chirico y otros colaboradores, se detectan muchos motivos de código para hacerlos más correctos, más legibles o más usuales. Así se mejora el código, pero también nuestro conocimiento de R: después de recibir la misma alerta una docena de veces, aprendemos cómo evitarla. Os presentaré lintr, y las formas de usarlo y de extenderlo con nuevas reglas para todos o para su propio proyecto.

flir, para mejorar el código R más rápido

El paquete flir de Étienne Bacher es como lintr pero más rápido: usa Rust, por eso detecta problemas a toda prisa. Además, puede corregir el código en lugar de emitir solo una alerta para la mayoría de sus reglas. Su limitación es tener menos reglas implementadas que lintr. Os presentaré flir, y como configurarlo con sus propias reglas, y como compartirlas si quereis.