“Pure, Predictable, Pipeable: Creating Fluent Interfaces with R”

A fluent interface lets you easily express yourself in code. Over time a fluent interface retreats to your subconscious. You don’t need to bring it to mind; the code just flows out of your fingers. I strive for this fluency in all the packages I write, and while I don’t always succeed, I think I’ve learned some valuable lessons along the way.

In this talk, I’ll discuss three guidelines that make it easier to develop fluent interfaces:

• **Pure functions.** A pure function only interacts with the world through its inputs and outputs; it has no side-effects. Pure functions make great building blocks because they’re easy to reason about and can be easily composed.

• **Predictable interfaces.** It’s easier to learn a function if its consistent, because you can learn the behaviour of a whole group of functions at once. I’ll highlight the benefits of predictability with some of my favourite R "WAT"s (including `c()`, `sapply()` and `sample()`).

• **Pipes.** Pure predictable functions are nice in isolation but are most powerful in combination. The pipe, `%>%`, is particularly in important when combining many functions because it turns function composition on its head so you can read it from left-to-right. I’ll show you how this has helped me build dplyr, rvest, ggvis, lowliner, stringr and more.

This talk will help you make best use of my recent packages, and teach you how to apply the same principles to make your own code easier to use.