Vectorization Slides

Adam Kaplan

March 30, 2022

Vectorization: apply and purrr

Vectorization: apply and purrr

- `apply`, `lapply`, `sapply`, ... come built-in with `R` and allow you to apply *any* function in a vectorized way to a matrix/dataframe, list, or vector.

Vectorization: apply and purrr

- `apply`, `lapply`, `sapply`, ... come built-in with `R` and allow you to apply *any* function in a vectorized way to a matrix/dataframe, list, or vector.
- `purrr`'s `map`, `map_dbl`, `map_dfc`, ... are the `tidyverse` equivalents and extensions to the `apply()` family.

```r
library(tidyverse)
d <- list(
  data.frame(quant = c("danny", "insong", "teppei")),
  data.frame(schools = c("MIT", "Harvard"))
)
# We want a list as an output
lapply(d, function(x) nrow(x))

# default map returns a list
map(d, function(x) nrow(x))

# The tidy version
d %>% map(~ nrow(.x))
```

```r
library(tidyverse)
v <- 1:10

# We want a list as an output
sapply(v, function(x) x * x)

# The tidy version
v %>% map_dbl(~ .x * .x)
```

Vectorization using purrr

- `map_*` return different object type and will fail if it is not appropriate.

Vectorization using purrr

- `map_*` return different object type and will fail if it is not appropriate.
- Makes the code more predictable, and thus easier to debug.

Vectorization using purrr

- `map_*` return different object type and will fail if it is not appropriate.
- Makes the code more predictable, and thus easier to debug.
- Of particular interest are `map_dfr` and `map_dfc`, which run a function on the input and then row/column bind the outputs together.

Vectorization using purrr

- `map_*` return different object type and will fail if it is not appropriate.
- Makes the code more predictable, and thus easier to debug.
- Of particular interest are `map_dfr` and `map_dfc`, which run a function on the input and then row/column bind the outputs together.
- Inspired popular parallelization package `furrr` which we will introduce you to shortly.