MANGO SOLUTIONS

Functional Programming with purrr

Doug Ashton
Principal Data Scientist
dashton@mango-solutions.com
@dougashton

Agenda

- Functional Programming
- Lists
- purrr

What is Functional Programming?

Disclaimer: I'm not a Computer Scientist

Emphasise the **result**

not

Exactly **how** to do it

```r
maximum <- max(vec)
```

```r
maximum <- -Inf
for i in seq_along(vec) {
  if (vec[i] > maximum) {
    maximum <- vec[i]
  }
}
```

Expression Based

$$E = mc^2$$

$$c = 299792458$$

$$m = 80$$

Expression Based

```
E <- expression({
  m * c^2
})

c <- 299792458
m <- 80

> eval(E)
[1] 7.190041e+18
```

Some Guiding Principles

1. Do reuse expressions
> functions

```
rest_energy <- function(m, c) {
  m * c^2
}

E0 <- rest_energy(m0, c)
E1 <- rest_energy(m1, c)
```

2. Don’t reassign variables

```
E <- m
E <- E * c
E <- E * c
E
```

imperative

3. No side-effects

```
side_effect_energy <- function(m, c) {
  # update E in parent frame
  E <<- m * c^2
}
```

Classic side-effects

- Changing values in a database (I/O)
- Changing files on disk (I/O)
- Changing global variables

4. Stateless

```
stateful_energy <- function() {
  # m and c come from parent frame
  m * c^2
}
```

Stateless function returns the same value for the same inputs. Always.

Functional Style

- Variables defined once
- Functions don't cause side-effects
- Functions are stateless
- Functions are reusable
  - Resist copy/paste
- Separate your I/O

} pure

Functional Programming for Data

- Easier to read
- Easier to test
- Easier to debug
- Good for Big Data
  - Parallelises well
- Tidyverse largely satisfies FP

Recap on Lists