INTERMEDIATE FUNCTIONAL PROGRAMMING WITH PURRR

Why cleaner code?

Colin Fay
Data Scientist & R Hacker

Where's Waldo?

```
library(broom)
library(dplyr)
lm(Sepal.Length ~ Species, data=iris) %>% tidy() %>% filter(p.value < 0.05)
lm(Pepal.Length ~ Species, data=iris) %>% tidy() %>% filter(p.value < 0.05)
lm(Sepal.Width ~ Species, data=iris) %>% tidy() %>% filter(p.value < 0.05)
lm(Sepal.Length ~ Species, data=iris) %>% tidy() %>% ilter(p.value < 0.05)
```

Finding Waldo

```
library(purrr)

tidy_iris_lm <- compose(
  as_mapper(~ filter(.x, p.value < 0.05)),
  tidy,
  partial(lm, data=iris, na.action = na.fail)
)

list(
  Petal.Length ~ Petal.Width,
  Petal.Width ~ Sepal.Width,
  Sepal.Width ~ Sepal.Length
) %>% map(tidy_iris_lm)
```

What is clean code?

Clean code is:

- Light
- Readable
- Interpretable
- Maintainable

compose()

Composing functions:

```
library(purrr)

rounded_mean <- compose(round, mean)
rounded_mean(1:2811)
[1] 1406
```

Composing cleaner code

```
# FROM
round(mean(1:10))
round(mean(1:100))
round(mean(1:1000))
round(mean(1:10000))
```

```
#TO
round(median(1:10))
round(median(1:100))
round(median(1:1000))
round(median(1:10000))
```

-> 4 changes

```
# FROM
my_stats <- compose(round, mean)
my_stats(1:10)
my_stats(1:100)
my_stats(1:1000)
my_stats(1:10000)
```

```
#TO
my_stats <- compose(round, median)
my_stats(1:10)
my_stats(1:100)
my_stats(1:1000)
my_stats(1:10000)
```

-> 1 change

INTERMEDIATE FUNCTIONAL PROGRAMMING WITH PURRR

Let's practice!

INTERMEDIATE FUNCTIONAL PROGRAMMING WITH PURRR

Building functions with compose() and negate()

Colin Fay
Data Scientist & R Hacker at ThinkR

compose() at will

Limitless compositions:

```
library(broom)
library(purrr)

clean_aov <- compose(tidy, anova, lm)
```

```
clean_aov(Sepal.Length ~ Sepal.Width, data = iris)
# A tibble: 2 x 6
  term           df  sumsq meansq statistic p.value
  <chr>       <int>  <dbl>  <dbl>     <dbl>   <dbl>
1 Sepal.Width     1   1.41  1.41       2.07   0.152
2 Residuals     148 101.    0.681     NA     NA
```

negate()

Flip the logical:

```
is_not_na <- negate(is.na)

x <- c(1,2,3,4, NA)

is.na(x)
```

```
[1] FALSE FALSE FALSE FALSE  TRUE
```

```
is_not_na(x)
```

```
[1]  TRUE  TRUE  TRUE  TRUE FALSE
```

With mappers

`negates()` & mappers:

```
under_hundred <- as_mapper(~ mean(.x) < 100)

not_under_hundred <- negate(under_hundred)

map_lgl(98:102, under_hundred)
```

```
[1]  TRUE  TRUE FALSE FALSE FALSE
```

```
map_lgl(98:102, not_under_hundred)
```

```
[1]  FALSE FALSE  TRUE  TRUE  TRUE
```

Digression on http codes

HTTP Status Codes

Code	Description	Code	Description
200	OK	400	Bad Request
201	Created	401	Unauthorized
202	Accepted	403	Forbidden
301	Moved Permanently	404	Not Found
303	See Other	410	Gone
304	Not Modified	500	Internal Server Error
307	Temporary Redirect	503	Service Unavailable

From Hypertext Transfer Protocol (HTTP) Status Code Registry , fixerrorcode.com

Test if in

Is x `%in%` y?

```
status <- 201

good_status <- c(200, 201, 202, 203)

status %in% good_status
[1] TRUE
```

INTERMEDIATE FUNCTIONAL PROGRAMMING WITH PURRR

Let's practice!

INTERMEDIATE FUNCTIONAL PROGRAMMING WITH PURRR

Prefilling functions

Colin Fay
Data Scientist & R Hacker at ThinkR

Using partial()

Prefill a function:

```
mean_na_rm <- partial(mean, na.rm = TRUE)
mean_na_rm( c(1,2,3,NA) )
[1] 2
```

```
lm_iris <- partial(lm, data = iris)
lm_iris(Sepal.Length ~ Sepal.Width)
Call:
lm(formula = ..1, data = iris)

Coefficients:
(Intercept)  Sepal.Width
     6.5262      -0.2234
```

Prefilling the mean() function

```
mean(airquality$Ozone,
     na.rm = TRUE)
mean(mtcars$mpg,
     na.rm = TRUE)
mean(iris$Petal.Width,
     na.rm = TRUE)
```

```
mean_na_rm <- partial(mean,
                      na.rm = TRUE)
mean_na_rm(airquality$Ozone)
mean_na_rm(mtcars$mpg)
mean_na_rm(iris$Sepal.Length)
```

Using partial() and compose()

`partial()` & `compose()`

```
rounded_mean <- compose(
  partial(round, digits = 2),
  partial(mean, na.rm = TRUE)
)
rounded_mean(airquality$Ozone)
[1] 42.13
```

rvest

- `read_html()`
- `html_nodes()`
- `html_text()`
- `html_attr()`

INTERMEDIATE FUNCTIONAL PROGRAMMING WITH PURRR

Let's practice!

INTERMEDIATE FUNCTIONAL PROGRAMMING WITH PURRR

List columns

Colin Fay
Data Scientist at ThinkR

What is a list column?

A data.frame with a list for a column:

```
library(tidyverse)

df <- tibble(
  classic = c("a", "b","c"),
  list = list(
    c("a", "b","c"),
    c("a", "b","c", "d"),
    c("a", "b","c", "d", "e")
  )
)
df
```

```
# A tibble: 3 x 2
  classic list
  <chr>   <list>
1 a       <chr [3]>
2 b       <chr [4]>
3 c       <chr [5]>
```

Why list columns?

```
library(tidyverse)
library(rvest)
a_node <- partial(html_nodes, css = "a")
href <- partial(html_attr, name = "href")
get_links <- compose( href, a_node,  read_html )

urls_df <- tibble(
  urls = c("https://thinkr.fr", "https://colinfay.me",
           "https://datacamp.com", "http://cran.r-project.org/")
)
urls_df %>%
  mutate(links = map(urls, get_links))
```

```
# A tibble: 4 x 2
  urls                     links
  <chr>                    <list>
1 https://thinkr.fr        <chr [106]>
2 https://colinfay.me      <chr [33]>
3 https://datacamp.com     <chr [93]>
4 http://cran.r-project.org/ <chr [1]>
```

Unnesting nested data.frame

```
urls_df %>%
  mutate(links = map(urls, get_links)) %>%
  unnest()
```

```
# A tibble: 233 x 2
   urls            links
   <chr>           <chr>
 1 https://think… https://thinkr.fr/
 2 https://think… https://thinkr.fr/
 3 https://think… https://thinkr.fr/formation-au-logiciel-r/
 4 https://think… https://thinkr.fr/formation-au-logiciel-r/intr…
 5 https://think… https://thinkr.fr/formation-au-logiciel-r/stat…
 6 https://think… https://thinkr.fr/formation-au-logiciel-r/prog…
 7 https://think… https://thinkr.fr/formation-au-logiciel-r/r-et…
 8 https://think… https://thinkr.fr/formation-au-logiciel-r/r-po…
 9 https://think… https://thinkr.fr/formation-au-logiciel-r/inte…
10 https://think… https://thinkr.fr/formation-au-logiciel-r/form…
# ... with 223 more rows
```

nest() a standard data.frame

```
library(dplyr)
library(tidyr)

iris_n <- iris %>%
  group_by(Species) %>%
  nest()

iris_n
```

```
# A tibble: 3 x 2
  Species    data
  <fct>      <list>
1 setosa     <tibble [50 × 4]>
2 versicolor <tibble [50 × 4]>
3 virginica  <tibble [50 × 4]>
```

A new list to map on

```
iris_n %>%
  mutate(lm = map(data, ~ lm(Sepal.Length ~ Sepal.Width, data = .x)))
```

```
# A tibble: 3 x 3
  Species    data              lm
  <fct>      <list>            <list>
1 setosa     <tibble [50 × 4]> <S3: lm>
2 versicolor <tibble [50 × 4]> <S3: lm>
3 virginica  <tibble [50 × 4]> <S3: lm>
...
```

nest() and unnest()

```
summary_lm <- compose(summary, lm)

iris %>%
  group_by(Species) %>%
  nest() %>%
  mutate(data = map(data, ~ summary_lm(Sepal.Length ~ Sepal.Width,
                                       data = .x)),
         data = map(data, "r.squared")) %>%
  unnest()
```

```
# A tibble: 3 x 2
  Species     data
  <fct>      <dbl>
1 setosa     0.551
2 versicolor 0.277
3 virginica  0.209
```

INTERMEDIATE FUNCTIONAL PROGRAMMING WITH PURRR

Let's practice!