`purrr`

Justin Post

`purrr` Package

- Provides a `tidyverse` alternative to the `apply()` family
    - Cheat sheet

NC STATE UNIVERSITY

`purrr` Package

- Provides a `tidyverse` alternative to the `apply()` family
    - Cheat sheet
- Main advantage is more consistency and some helper functions
    - Accepted answer here (by Hadley) gives some good details

map()

- Always returns a `list`
- First arg is the list, second is the function

```
set.seed(10)
my_list <- list(rnorm(100), runif(10), rgamma(40, shape = 1, rate = 1))
map(my_list, mean)
```

```
## [[1]]
## [1] -0.1365489
##
## [[2]]
## [1] 0.5997619
##
## [[3]]
## [1] 1.108209
```

NC STATE UNIVERSITY

map()

- Allows for shorthand
- Suppose we want the second element of each list. Compare:

```
map(my_list, 2)
```

```
## [[1]]
## [1] -0.1842525
##
## [[2]]
## [1] 0.535895
##
## [[3]]
## [1] 1.076614
```

```
lapply(my_list, function(x) x[[2]])
lapply(my_list, `[[`, 2)
```

NC STATE UNIVERSITY

purrr

- `purrr` functions also give a shorthand way to make anonymous functions

```
map(my_list, \(x) mean(x))
```

```
## [[1]]
## [1] -0.1365489
##
## [[2]]
## [1] 0.5997619
##
## [[3]]
## [1] 1.108209
```

```
map(my_list, \(x) max(x)-min(x))
```

```
## [[1]]
## [1] 4.405807
##
## [[2]]
## [1] 0.8494514
##
## [[3]]
## [1] 4.150777
```

map_*()

- Allows you to specify the type of output

```
map_dbl(my_list, mean)
```

```
## [1] -0.1365489  0.5997619  1.1082087
```

- `map_chr()`, `map_lgl()`, ... return vectors

map2()

- Allows you to apply a function to two similar lists (returns a list)

```
my_list_2 <- list(rnorm(100), runif(10), rgamma(40, shape = 1, rate = 1))
map2(my_list, my_list_2, \(x, y) mean(x)-mean(y))
```

```
## [[1]]
## [1] -0.05717766
##
## [[2]]
## [1] 0.03301644
##
## [[3]]
## [1] 0.04992712
```

NC STATE UNIVERSITY

pmap()

- Extends this idea to an arbitrary number of lists

```
my_list_3 <- list(rnorm(100), runif(10), rgamma(40, shape = 1, rate = 1))
pmap(list(my_list, my_list_2, my_list_3),
     \(x, y, z) (mean(x)-mean(y))/mean(z))
```

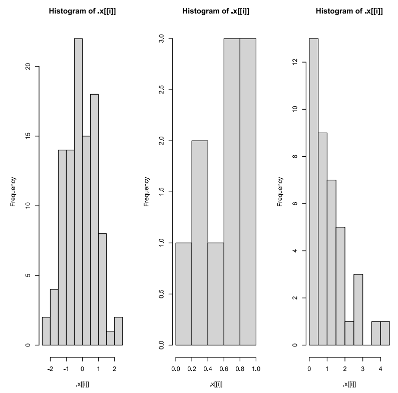
```
## [[1]]
## [1] 0.4895469
##
## [[2]]
## [1] 0.07453712
##
## [[3]]
## [1] 0.0421021
```

NC STATE UNIVERSITY

walk()

- `walk()` allows you to use a side-effect function but return the original data

```
#just apply the function
par(mfrow = c(1, 3))
my_list |>
  map(hist)
```



```
## [[1]]
## $breaks
##  [1] -2.5 -2.0 -1.5 -1.0 -0.5  0.0  0.5  1.0  1.5  2.0  2.5
```

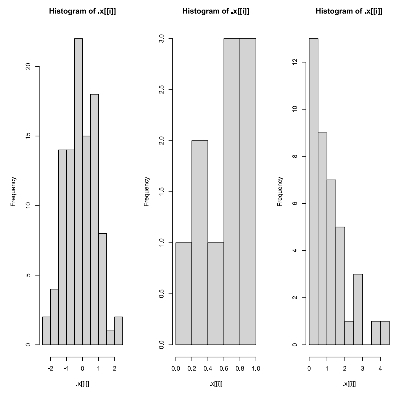
NC STATE UNIVERSITY

```
1   2
##
```

walk()

- `walk()` allows you to use a side-effect function but return the original data

```
par(mfrow = c(1, 3))
#now apply the function but still have the original data
my_list |>
  walk(hist) |>
  map_dbl(mean)
```

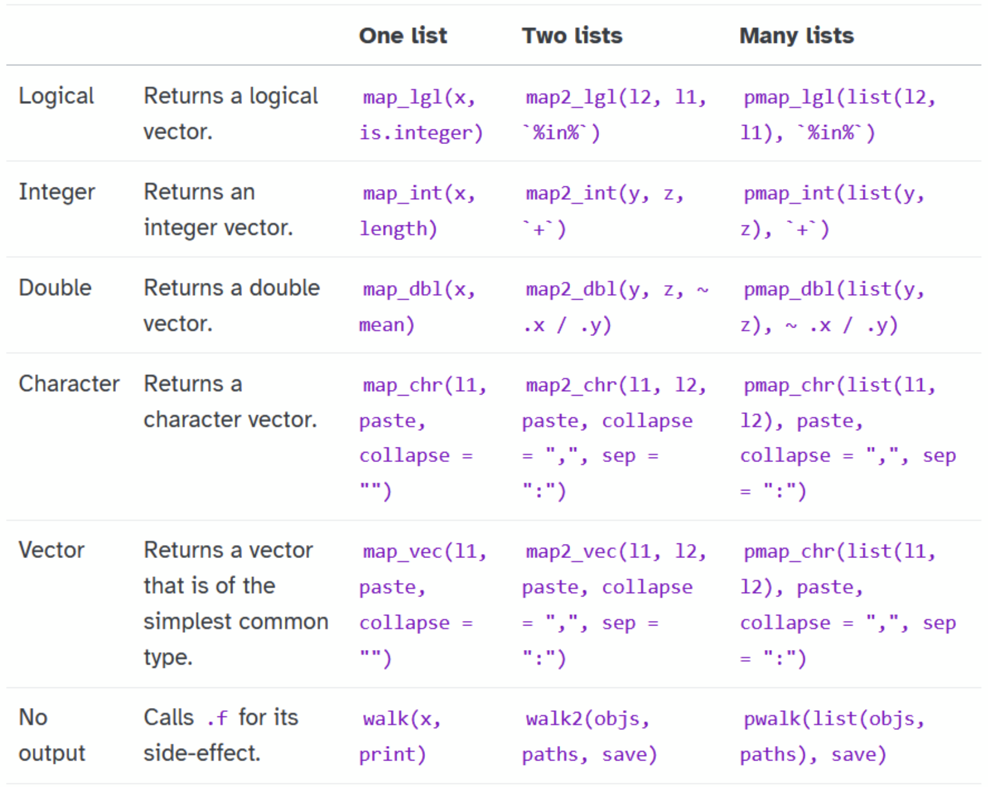


```
## [1] -0.1365489  0.5997619  1.1082087
```

NC STATE UNIVERSITY

Summary of Common `purrr` Functions

- Plenty of other functionality provided (see cheat sheet)



		One list	Two lists	Many lists
Logical	Returns a logical vector.	`map_lgl(x, is.integer)`	``map2_lgl(l2, l1, `%in%`)``	``pmap_lgl(list(l2, l1), `%in%`)``
Integer	Returns an integer vector.	`map_int(x, length)`	``map2_int(y, z, `+`)``	``pmap_int(list(y, z), `+`)``
Double	Returns a double vector.	`map_dbl(x, mean)`	`map2_dbl(y, z, ~ .x / .y)`	`pmap_dbl(list(y, z), ~ .x / .y)`
Character	Returns a character vector.	`map_chr(l1, paste, collapse = "")`	`map2_chr(l1, l2, paste, collapse = ",", sep = ":")`	`pmap_chr(list(l1, l2), paste, collapse = ",", sep = ":")`
Vector	Returns a vector that is of the simplest common type.	`map_vec(l1, paste, collapse = "")`	`map2_vec(l1, l2, paste, collapse = ",", sep = ":")`	`pmap_chr(list(l1, l2), paste, collapse = ",", sep = ":")`
No output	Calls `.f` for its side-effect.	`walk(x, print)`	`walk2(objs, paths, save)`	`pwalk(list(objs, paths), save)`

List Columns

- Recall our connection between lists and data frames:
    - Data frame = list of equal length vectors

```
 typeof(iris)
```

```
## [1] "list"
```

```
 str(iris)
```

```
## 'data.frame':	150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

NC STATE UNIVERSITY

List Columns

- Recall our connection between lists and data frames:
  - Data frame = list of equal length vectors
- A list is a vector… if of appropriate length, it can be the column of a data frame!

```
iris |>
  as_tibble() |>
  mutate(diffs = pmap(list(Sepal.Length, Sepal.Width, Petal.Length, Petal.Width),
                      \(x, y, z, w) list(x-y, x-z, x-w))) |>
  select(diffs, everything())
```

```
## # A tibble: 150 x 6
##   diffs      Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##   <list>            <dbl>       <dbl>        <dbl>       <dbl> <fct>
## 1 <list [3]>          5.1         3.5          1.4         0.2 setosa
## 2 <list [3]>          4.9         3            1.4         0.2 setosa
## 3 <list [3]>          4.7         3.2          1.3         0.2 setosa
## 4 <list [3]>          4.6         3.1          1.5         0.2 setosa
## 5 <list [3]>          5           3.6          1.4         0.2 setosa
## # i 145 more rows
```

NC STATE UNIVERSITY

List Columns

- Recall our connection between lists and data frames:
    - Data frame = list of equal length vectors
- A list is a vector... if of appropriate length, it can be the column of a data frame!

```
 iris |>
   as_tibble() |>
   mutate(diffs = pmap(list(Sepal.Length, Sepal.Width, Petal.Length, Petal.Width),
                       \(x, y, z, w) list(x-y, x-z, x-w))) |>
   pull(diffs)
```

```
## [[1]]
## [[1]][[1]]
## [1] 1.6
##
## [[1]][[2]]
## [1] 3.7
##
## [[1]][[3]]
## [1] 4.9
##
```

```
## [1] 1.9
```

List Columns

- A more interesting example!
- Note: `purrr:pluck()` is a helper function for grabbing a named element or by index number

```
library(httr)
library(jsonlite)
game_info <- GET("https://api-web.nhle.com/v1/score/2024-04-04") |>
  content("text") |>
  fromJSON(flatten = TRUE, simplifyDataFrame = TRUE) |>
  pluck("games")
```

- `pluck()` could be replaced with

```
`[[`("games")
```

NC STATE UNIVERSITY

List-Columns

- Check the `tvBroadcasts` column!

```r
str(game_info, max.level = 1)
```

```
## 'data.frame':	9 obs. of  40 variables:
##  $ id                             : int  2023021202 2023021203 2023021204 2023021205 2023021206 2023021207 20230212
##  $ season                         : int  20232024 20232024 20232024 20232024 20232024 20232024 20232024 20232024 20
##  $ gameType                       : int  2 2 2 2 2 2 2 2 2
##  $ gameDate                       : chr  "2024-04-04" "2024-04-04" "2024-04-04" "2024-04-04" ...
##  $ startTimeUTC                   : chr  "2024-04-04T23:00:00Z" "2024-04-04T23:00:00Z" "2024-04-04T23:00:00Z" "2024
##  $ easternUTCOffset               : chr  "-04:00" "-04:00" "-04:00" "-04:00" ...
##  $ venueUTCOffset                 : chr  "-04:00" "-04:00" "-04:00" "-04:00" ...
##  $ tvBroadcasts                   :List of 9
##  $ gameState                      : chr  "OFF" "OFF" "OFF" "OFF" ...
##  $ gameScheduleState              : chr  "OK" "OK" "OK" "OK" ...
##  $ gameCenterLink                 : chr  "/gamecenter/bos-vs-car/2024/04/04/2023021202" "/gamecenter/nyi-vs-cbj/202
##  $ threeMinRecap                  : chr  "/video/recap-bruins-at-hurricanes-4-4-24-6350293603112" "/video/recap-isl
##  $ neutralSite                    : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ venueTimezone                  : chr  "US/Eastern" "US/Eastern" "America/Montreal" "US/Eastern" ...
##  $ period                         : int  3 3 3 3 3 3 3 3 3
##  $ goals                          :List of 9
##  $ threeMinRecapFr                : chr  NA NA "/fr/video/recap-lightning-at-canadiens-4-4-24-6350292258112" "/fr/v
##  $ condensedGame                  : chr  NA NA NA NA ...
                                      : chr  "PNC Arena" "Nationwide Arena" "Centre Bell" "Canadian Tire Centre" ...
                                      : int  6 2 14 13 5 21 19 20 26
                                      : chr  "BOS" "NYI" "TBL" "FLA" ...
##  $ awayTeam.score                 : int  4 4 7 6 4 5 3 2 2
```

Working with List-Columns

- In this case, our list-column contains a data frame in each list element:

```
game_info$tvBroadcasts
```

```
## [[1]]
##     id market countryCode network sequenceNumber
## 1 375      H          US    BSSO            390
## 2  31      A          US    NESN            396
##
## [[2]]
##     id market countryCode network sequenceNumber
## 1 347      H          US    BSOH            391
## 2 409      A          US   MSGSN            402
##
## [[3]]
##     id market countryCode network sequenceNumber
## 1 131      H          CA    TSN2            112
## 2  33      H          CA     RDS            132
## 3 359      A          US   BSSUN            401
##
## [[4]]
##     id market countryCode network sequenceNumber
## 1 230      H          CA    RDS2            133
                                 N5            135
                                 FL            404

## [[5]]
```

Working with List-Columns

- We can manipulate list-columns with `dplyr::mutate()`
- Since elements are lists, we want to use `map()` functions!

```
game_info |>
  mutate(num_networks = map(tvBroadcasts, nrow)) |>
  select(num_networks, tvBroadcasts, everything())
```

```
##   num_networks
## 1            2
## 2            2
## 3            3
## 4            3
## 5            3
## 6            3
## 7            2
## 8            2
## 9            3
##                                                                  tvBroadcasts
## 1                         375, 31, H, A, US, US, BSSO, NESN, 390, 396
## 2                        347, 409, H, A, US, US, BSOH, MSGSN, 391, 402
## 3  131, 33, 359, H, H, A, CA, CA, US, TSN2, RDS, BSSUN, 112, 132, 401
## 4 230, 294, 353, H, H, A, CA, CA, US, RDS2, TSN5, BSFL, 133, 135, 404
## 5  282, 528, 517, N, A, H, CA, US, US, SN, SN-PIT, MNMT, 21, 375, 387
[illegible], US, US, ALT, BSN, BSWI, 376, 395, 403
[illegible] 391, N, N, US, US, ESPN+, HULU, 16, 17
[illegible], 292, A, H, CA, CA, SNW, TSN3, 34, 134
## 9  282, 355, 314, N, A, H, CA, US, US, SN, BSW, NBCSCA, 101, 379, 384
```

Recap!

`purrr` gives us a bit cleaner/more consistent way to apply functions to objects

- Lots of additional helper functions
- Use `apply()` family or `purrr` to improve your code!

NC STATE UNIVERSITY