



FOUNDATIONS OF FUNCTIONAL PROGRAMMING WITH PURRR

# The power of iteration

Auriel Fournier  
Instructor



# Iteration without purrr

```
USbirds <- read_csv("us_data.csv")
CANbirds <- read_csv("can_data.csv")
MEXbirds <- read_csv("mex_data.csv")

birdfiles <- list.files(pattern=".csv")
birdfiles
```

```
[1] "can_data.csv"
[2] "mex_data.csv"
[3] "us_data.csv"
```

```
list_of_birdfiles <- list()

for(i in birdfiles){
  list_of_birdfiles[[i]] <- read_csv(i)
}
```



# Iteration without purrr

```
files <- list.files()
```

```
d <- list()

# Loop through the values 1 through 10, to add them to d
for(i in 1:10){
  d[[i]] <- read_csv(files[i])
}
```



# Iteration with purrr

```
map(object, function)
```

`object` - can be a vector or a list

`function` - any function in R that takes the input offered by the object

```
d <- map(files, read_csv)
```



# Let's work through an example

```
bird_counts
```

```
[[1]]  
[1] 3 1  
  
[[2]]  
[1] 3 8 1 2  
  
[[3]]  
[1] 8 3 9 9 5 5  
  
[[4]]  
[1] 8 9 7 9 5 4 1 5
```



# Example time!

```
# Create bird_sum list, loop over and sum elements of bird_counts
bird_sum <- list()

for(i in seq_along(bird_counts)){
  bird_sum[[i]] <- sum(bird_counts[[i]])
}

# sum each element of bird_counts, and put it in bird_sum
bird_sum <- map(bird_counts, sum)
bird_sum
```

```
[[1]]
[1] 8
```

```
[[2]]
[1] 28
```

```
[[3]]
[1] 44
```

```
[[4]]
[1] 47
```



## FOUNDATIONS OF FUNCTIONAL PROGRAMMING WITH PURRR

**Let's purrr-actice!**



## FOUNDATIONS OF FUNCTIONAL PROGRAMMING WITH PURRR

# Subsetting lists

Auriel Fournier  
Instructor





# Let's talk about lists!

```
lo[["data"]] <- data.frame(bird = c("robin", "sparrow", "jay"),  
                           weight = c(76, 14, 100),  
                           wing_length = c(100, 35, 130))  
  
lo[["model"]] <- lm(weight ~ wing_length,  
                    data = lo[["data"]])  
  
lo[["plot"]] <- ggplot(  
  data = lo[["model"]],  
  aes(x = weight, y = wing_length)) +  
  geom_point()
```



# Indexing dataframes and lists

## Dataframes

```
mtcars[1, "wt"]
```

```
[1] 2.62
```

```
mtcars$wt
```

```
[1] 2.620 2.875 2.320 3.215 3.440  
[6] 3.460 3.570 3.190 3.150 3.440  
[11] 3.440 4.070 3.730 3.780 5.250  
[16] 5.424 5.345 2.200 1.615 1.835  
[21] 2.465 3.520 3.435 3.840 3.845  
[26] 1.935 2.140 1.513 3.170 2.770  
[31] 3.570 2.780
```

## Lists

```
lo[[2]]
```

```
Call:
```

```
lm(formula = weight ~ wing_length,  
    data = lo[["data"]])
```

```
Coefficients:
```

```
(Intercept)  wing_length  
-17.3216      0.9131
```

```
lo[["model"]]
```

```
Call:
```

```
lm(formula = weight ~ wing_length,  
    data = lo[["data"]])
```

```
Coefficients:
```

```
(Intercept)  wing_length  
-17.3216      0.9131
```



# Calculate something on each element without purrr

```
# Create a dataframe to place the results in
df_rows <- data.frame(names = names(survey_data), rows = NA)

# Loop over survey_data to determine how many rows are in each element
for(i in 1:length(survey_data)){
  df_rows[i, 'rows'] <- nrow(survey_data[[i]])
}
```

```
# Print out survey_rows
df_rows
```

```
   names rows
1 LakeErieS  14
2 LakeErieN  14
3 LakeErieW  14
4 LakeErieE  15
```

# Calculate something on each element with purrr

```
# Get a summary of survey_data  
summary(survey_data)
```

	Length	Class	Mode
LakeErieS	2	data.frame	list
LakeErieN	2	data.frame	list
LakeErieW	2	data.frame	list
LakeErieE	2	data.frame	list

```
# Determine row num in survey_data  
map(survey_data, ~nrow(.x))
```

```
$`LakeErieS`  
[1] 14
```

```
$`LakeErieN`  
[1] 14
```

```
$`LakeErieW`  
[1] 14
```

```
$`LakeErieE`  
[1] 15
```



## FOUNDATIONS OF FUNCTIONAL PROGRAMMING WITH PURRR

**Let's purrr-actice!**



FOUNDATIONS OF FUNCTIONAL PROGRAMMING WITH PURRR

# The many flavors of map()

Auriel Fournier  
Instructor



# Non-list outputs

```
# Map over survey_data and determine number of rows
map(survey_data, ~nrow(.x))

$`LakeErieS`
[1] 14

$`LakeErieN`
[1] 14

$`LakeErieW`
[1] 14

$`LakeErieE`
[1] 15
```



# purrr::map\_variants

## List output

```
# Determine row number
map(survey_data, ~nrow(.x))

$`LakeErieS`
[1] 14

$`LakeErieN`
[1] 14

$`LakeErieW`
[1] 14

$`LakeErieE`
[1] 15
```

## Double, a type of numeric

```
# Determine row number
map_dbl(survey_data, ~nrow(.x))

[1] 14 14 14 15
```





# map\_lgl()

## List

```
# Determine row number
map(survey_data, ~nrow(.x))

$`LakeErieS`
[1] 14

$`LakeErieN`
[1] 14

$`LakeErieW`
[1] 14

$`LakeErieE`
[1] 15
```

## Logical

```
# Determine if elements have 14 rows
map_lgl(survey_data, ~nrow(.x)==14)

[1] TRUE TRUE TRUE FALSE
```



# map\_chr()

## List output

```
# Map over species_names list
map(species_names, ~.x)

$`LakeErieS`
[1] "Green Frog"

$`LakeErieN`
[1] "American Bullfrog"

$`LakeErieW`
[1] "Gray Treefrog"

$`LakeErieE`
[1] "Mudpuppy"
```

## Character

```
# Map over species_names list
map_chr(species_names, ~.x)

LakeErieS      LakeErieN
"Green Frog"   "American Bullfrog"
LakeErieW      LakeErieE
"Gray Treefrog" "Mudpuppy"
```



# Example time!

```
# Create a dataframe called survey_rows
survey_rows <- data.frame(names = names(survey_data),
                          rows = NA)

# Map over survey_data to determine row number in each element
survey_rows$rows <- map_dbl(survey_data, ~nrow(.x))

# Print out the survey_rows dataframe
survey_rows
```

	names	rows
1	LakeErieS	14
2	LakeErieN	14
3	LakeErieW	14
4	LakeErieE	15



## FOUNDATIONS OF FUNCTIONAL PROGRAMMING WITH PURRR

**Let's purrr-actice!**