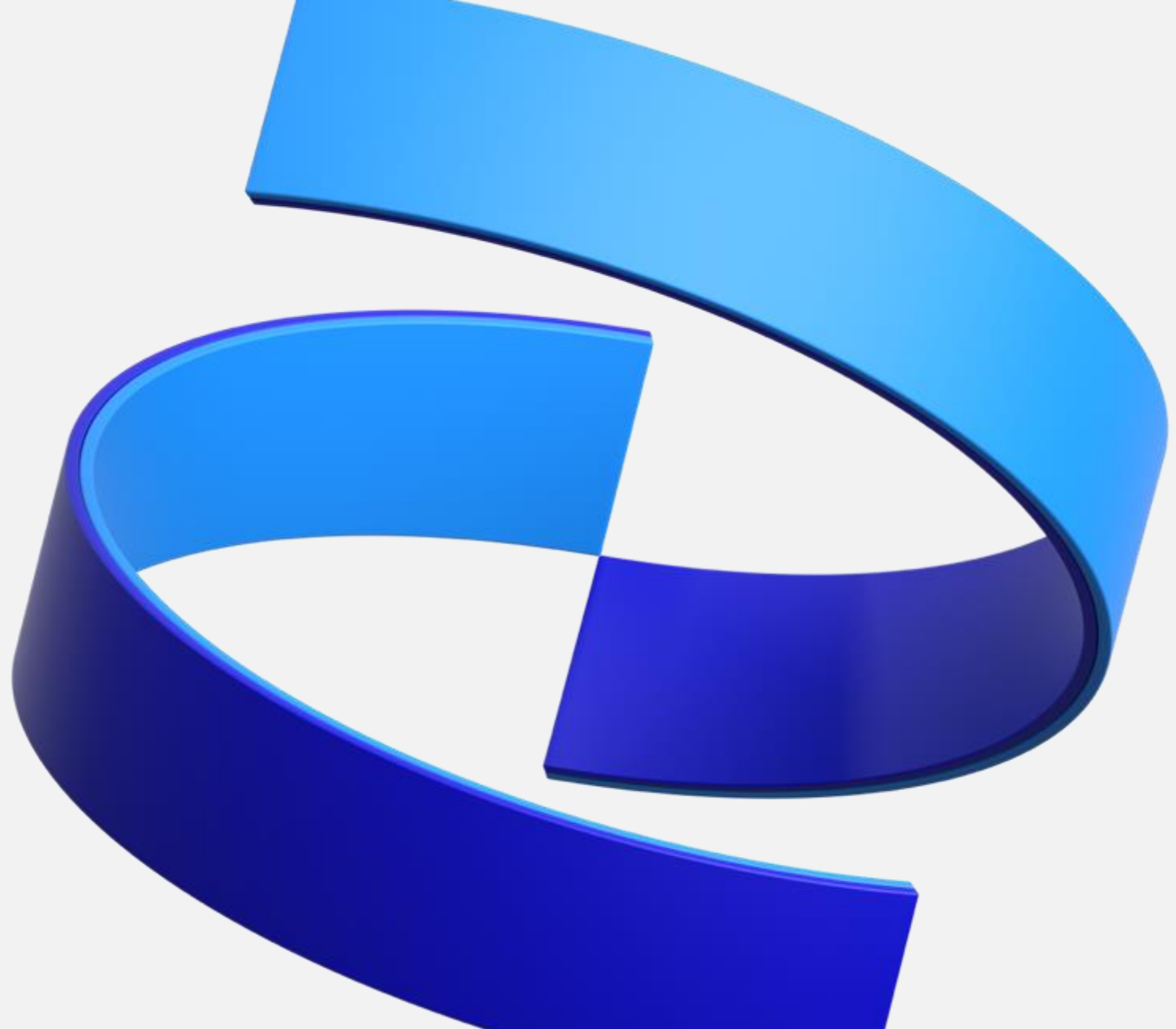


SS02: A Beginner's Roadmap to Map Functions for Clinical Reporting in R

Samuel Curlee
Gabriela Piasecki
Wenjian Xu
Xiaoqing Tang

7 March 2023
PHUSE US Connect 2023
samuel.curlee@pfizer.com
l.gabriela.piasecki@pfizer.com





Iterations in R

- A for loop evaluates statements inside the loop multiple times, iterating over (in R) a vector of values that controls the execution of the loop
- The process of iterating over an object and applying a function to each element is very common in programming – this process can itself be made into a function
- Map functions in the “purrr” package can automate the looping process, leading to shorter and clearer code

Map Functions and the purrr Package

- The most basic function in purrr is called map()

`map(.x, .f, ...)` – returns a list the same length as `.x`.
`map_dfr(.x, .f, ..., .id = NULL)` – returns a data frame.

`...` – a list of arguments passed on to the Map functions.

`.x` – a list or atomic vector.

`.f` – a function, formula, or vector.

`.id` – either a string or NULL.

- `map_dfr()` has the same input structure as `map()` but returns the output in a data frame rather than a list

Basic Example: For loop VS `map()`

First input is the object whose elements R will iterate over

Second input is the function that will be applied to each element

Output is the container (for `map()`, this is always a list) that stores the results of the function applied to each element

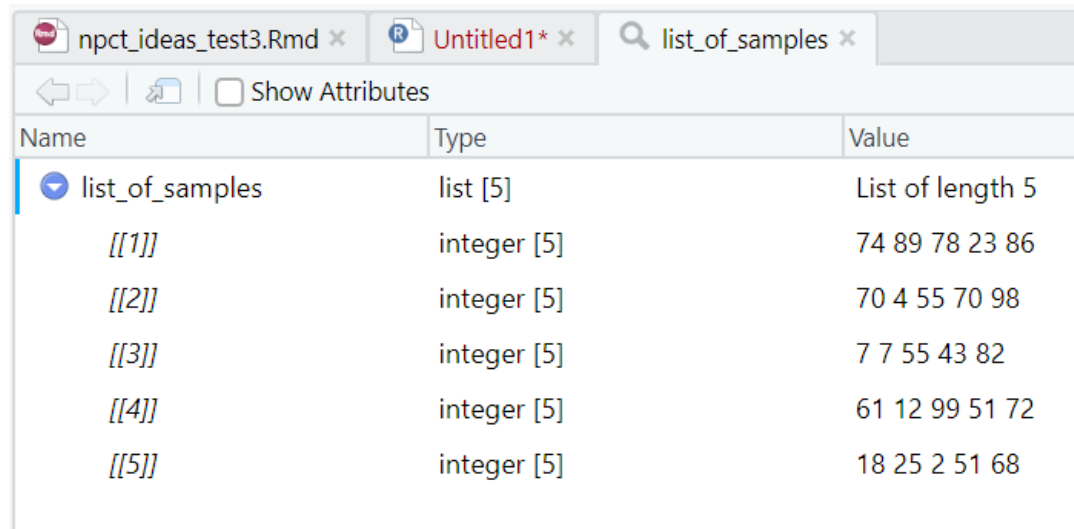
```
library(purrr)

#Create a list of random vectors
set.seed(100)
list_of_samples <- (replicate(5, sample(1:100, 5,
replace=TRUE), simplify=FALSE) )

Method 1:
#Get the mean of each vector in the list using a For loop
list_of_means <- list()
for (i in seq_along(list_of_samples)) {
  list_of_means[[i]] <- mean(list_of_samples[[i]])
}

Method 2:
#Get the mean of each vector in the list using map()
list_of_means2 <- map(list_of_samples, mean)
```

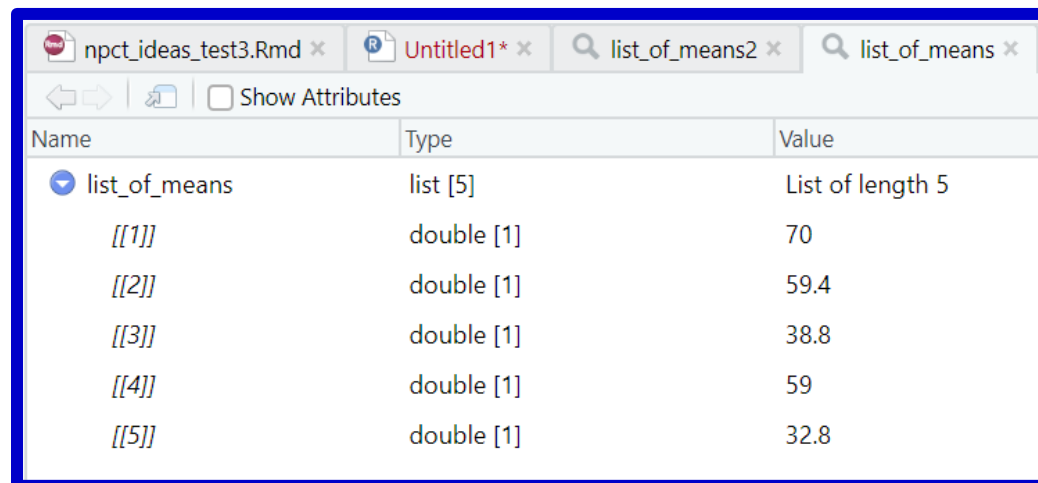
R output



The screenshot shows the R Studio Environment pane. The active tab is 'list_of_samples'. The table below represents the data shown in the pane.

Name	Type	Value
list_of_samples	list [5]	List of length 5
[[1]]	integer [5]	74 89 78 23 86
[[2]]	integer [5]	70 4 55 70 98
[[3]]	integer [5]	7 7 55 43 82
[[4]]	integer [5]	61 12 99 51 72
[[5]]	integer [5]	18 25 2 51 68

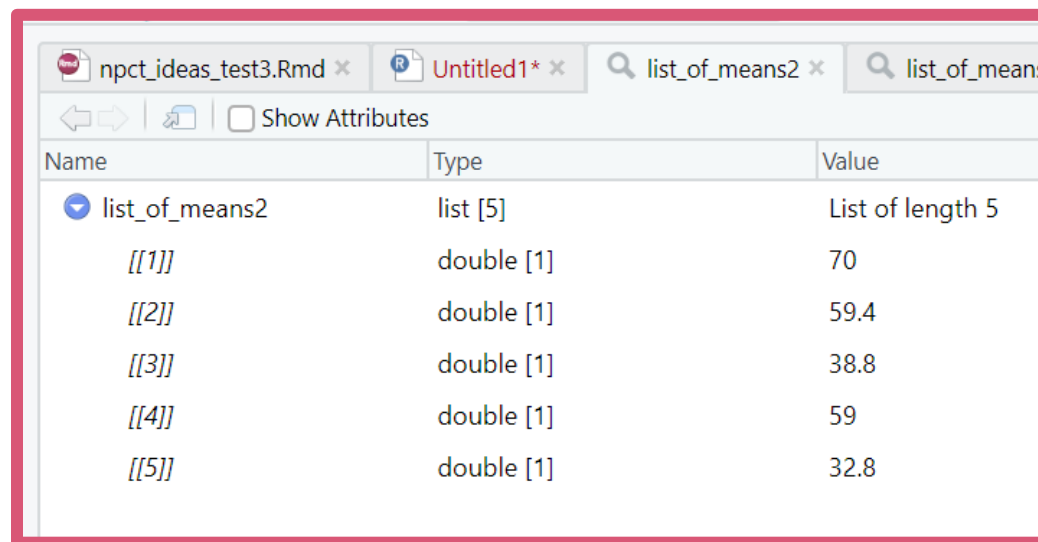
Method 1:



The screenshot shows the R Studio Environment pane. The active tab is 'list_of_means'. The table below represents the data shown in the pane.

Name	Type	Value
list_of_means	list [5]	List of length 5
[[1]]	double [1]	70
[[2]]	double [1]	59.4
[[3]]	double [1]	38.8
[[4]]	double [1]	59
[[5]]	double [1]	32.8

Method 2:



The screenshot shows the R Studio Environment pane. The active tab is 'list_of_means2'. The table below represents the data shown in the pane.

Name	Type	Value
list_of_means2	list [5]	List of length 5
[[1]]	double [1]	70
[[2]]	double [1]	59.4
[[3]]	double [1]	38.8
[[4]]	double [1]	59
[[5]]	double [1]	32.8

Using map() for Clinical Reporting

- Case study: demographic characteristics tables
- Good candidate for a loop-based approach since the process for obtaining an age group summary is the same as for a race summary and so on

Demographic Characteristics		
	Drug A (N=xxx)	Drug B (N=xxx)
Age (years)		
< 18	xx (xx.x)	xx (xx.x)
18 - 44	xx (xx.x)	xx (xx.x)
45 - 65	xx (xx.x)	xx (xx.x)
>65	xx (xx.x)	xx (xx.x)
Mean (SD)	xx.x (xx.xx)	xx.x (xx.xx)
Median (Range)	xx.x (xx, xx)	xx.x (xx, xx)
Gender		
Male	xx (xx.x)	xx (xx.x)
Female	xx (xx.x)	xx (xx.x)
Race		
White	xx (xx.x)	xx (xx.x)
Black	xx (xx.x)	xx (xx.x)
Asian	xx (xx.x)	xx (xx.x)
Other	xx (xx.x)	xx (xx.x)
Ethnicity		
Hispanic or Latino	xx (xx.x)	xx (xx.x)
Not Hispanic or Latino	xx (xx.x)	xx (xx.x)
Not reported	xx (xx.x)	xx (xx.x)

Using map() for Clinical Reporting

```
npct <- function(var) {  
  adsl %>%  
    group_by(trtan, trta, !!sym(var)) %>%  
    summarise(n_pat = n()) %>%  
    group_by(trtan, trta) %>%  
    mutate(N_pat = sum(n_pat),  
           pct = round(n_pat / N_pat*100,digits = 1),  
           txt = paste0(n_pat, " (", pct, "%)",  
           trta = paste0(trta, "\\n (N=", N_pat, ")")) %>%  
    ungroup() %>%  
    select( trta, !!sym(var), txt) %>%  
    pivot_wider(values_from = txt, names_from = trta) %>%  
    arrange(!!sym(var)) %>%  
    mutate(Label = case_when(!!var == "agegr1n" ~ "Age (years)",  
                             !!var == "sexn" ~ "Gender",  
                             !!var == "racen" ~ "Race",  
                             !!var == "ethn1c" ~ "Ethnicity"),  
           Subgroup=paste0(" ", case_when(!!var == "agegr1n"~ age_map[!!sym(var)],  
                                           !!var == "sexn"~ sex_map[!!sym(var)],  
                                           !!var == "racen"~ race_map[!!sym(var)],  
                                           !!var== "ethn1c" ~  
ethnic_map[!!sym(var)])),  
           order=case_when(!!var == "agegr1n" ~ 1,  
                             !!var == "sexn" ~ 2,  
                             !!var == "racen" ~ 3,  
                             !!var == "ethn1c" ~ 4)) %>%  
    select(Label, Subgroup, starts_with("Drug"), order) %>%  
    replace(is.na(.), "0")  
}
```

← Creating our function called npct

← We want the counts to be calculated by treatment group

← Gathering small n, big N, and the percentages per group per treatment group

← Arguments to dictate what the label should be based on which vars we choose as input

← This will paste in the correct subgroup values for each var

← Creating the order in which we want the vars to be displayed

```
tbl <- map_dfr(c("agegr1n", "sexn", "racen", "ethn1c"), function(var) npct(var))
```

←

Creating a data frame called tbl that calls map_dfr() to iterate our function for each categorical vector we want to generate the statistics for with row and column binding

npct_ideas_test3.Rmd x tbl x adsl x R Untitled1* x R Untitled2* x

Filter

	usubjid	trtan	trta	age	agegr1n	agegr1	sexn	sex	race	racen	ethnic
1	ABC123-001	1	Drug_A	42	2	18-44	2	F	Black	2	Hispanic or Latino
2	ABC123-002	2	Drug_B	35	2	18-44	2	F	White	1	Hispanic or Latino
3	ABC123-003	1	Drug_A	37	2	18-44	1	M	Asian	3	Not Hispanic or Lat
4	ABC123-004	1	Drug_A	67	4	>65	2	F	Other	4	Not Hispanic or Lat
5	ABC123-005	2	Drug_B	17	1	<18	1	M	Black		
6	ABC123-006	2	Drug_B	51	3	45-64	2	F	White		

ADSL before

tbl data frame
after (will be
used as input
prior to
generating final
table)



npct_ideas_test3.Rmd* x tbl x R basic example.R x

Filter

	Label	Subgroup	Drug_A\ (N=3)	Drug_B\ (N=3)	order
1	Age (years)	< 18	0	1 (33.3%)	1
2	Age (years)	18 - 44	2 (66.7%)	1 (33.3%)	1
3	Age (years)	45 - 64	0	1 (33.3%)	1
4	Age (years)	> 65	1 (33.3%)	0	1
5	Gender	Male	1 (33.3%)	1 (33.3%)	2
6	Gender	Female	2 (66.7%)	2 (66.7%)	2
7	Race	White	0	2 (66.7%)	3
8	Race	Black	1 (33.3%)	1 (33.3%)	3
9	Race	Asian	1 (33.3%)	0	3
10	Race	Other	1 (33.3%)	0	3
11	Ethnicity	Hispanic or Latino	1 (33.3%)	1 (33.3%)	4
12	Ethnicity	Not Hispanic or Latino	2 (66.7%)	1 (33.3%)	4
13	Ethnicity	Not reported	0	1 (33.3%)	4

Demographic Characteristics

Final Output

	Drug A (N=3)	Drug B (N=3)
Age (years)		
< 18	0	1 (33.3%)
18 - 44	2 (66.7%)	1 (33.3%)
45 - 64	0	1 (33.3%)
> 65	1 (33.3%)	0
Mean(SD)	48.7 (16.10)	34.3 (17.00)
Median(Range)	42.0 (37, 67)	35.0 (17, 51)
Gender		
Male	1 (33.3%)	1 (33.3%)
Female	2 (66.7%)	2 (66.7%)
Race		
White	0	2 (66.7%)
Black	1 (33.3%)	1 (33.3%)
Asian	1 (33.3%)	0
Other	1 (33.3%)	0
Ethnicity		
Hispanic or Latino	1 (33.3%)	1 (33.3%)
Not Hispanic or Latino	2 (66.7%)	1 (33.3%)
Not reported	0	1 (33.3%)



Conclusions

- The family of map functions in the purrr package can enable programmers to generate reporting datasets through efficient and clear iteration
- While not every type of table is best programmed in this way, tables with repeated blocks of similar calculations on different variables can benefit greatly from an iteration-based approach
- Other extensions of map() include map_chr(), map_int(), map_dbl(), and map_dfc() (amongst others)