

Examples of Lists and Named One and Two Dimensional Lists in R

Fan Wang

2020-05-28

Contents

1	Multiple Dimensional List	1
1.1	Named List of Matrixes	1
1.2	One Dimensional Named List	2
1.3	Two Dimensional Unnamed List	3
1.4	Define Two Dimensional Named List	4

1 Multiple Dimensional List

Go to the [RMD](#), [R](#), [PDF](#), or [HTML](#) version of this file. Go back to [fan's REconTools Package](#), [R Code Examples Repository \(bookdown site\)](#), or [Intro Stats with R Repository \(bookdown site\)](#).

- r list tutorial
- r vector vs list
- r initialize empty multiple element list
- r name rows and columns of 2 dimensional list
- r row and colum names of list
- list dimnames
- r named list to string

1.1 Named List of Matrixes

Save a list of matrixes. Retrieve Element of that list via loop.

```
# Define an array to loop over
ar_fl_mean <- c(10, 20, 30)

# store results in named list
ls_mt_res = vector(mode = "list", length = length(ar_fl_mean))
ar_st_names <- paste0('mean', ar_fl_mean)
names(ls_mt_res) <- ar_st_names

# Loop and generat a list of dataframes
for (it_fl_mean in seq(1, length(ar_fl_mean))) {
  fl_mean = ar_fl_mean[it_fl_mean]

  # dataframe
  set.seed(it_fl_mean)
  tb_combine <- as_tibble(
    matrix(rnorm(4,mean=fl_mean,sd=1), nrow=2, ncol=3)
  ) %>%
  rowid_to_column(var = "id") %>%
```

```

    rename_all(~c(c('id', 'var1', 'varb', 'vartheta'))))

ls_mt_res[[it_fl_mean]] = tb_combine
}

# Retrieve elements
print(ls_mt_res[[1]])
print(ls_mt_res$mean10)
print(ls_mt_res[['mean10']])

# Print via Loop
for (it_fl_mean in seq(1, length(ar_fl_mean))) {
  tb_combine = ls_mt_res[[it_fl_mean]]
  print(tb_combine)
}

```

1.2 One Dimensional Named List

1. define list
2. slice list
3. print r named list as a single line string
 - [R Unlist named list into one string with preserving list names](#)

```

# Define Lists
ls_num <- list(1,2,3)
ls_str <- list('1','2','3')
ls_num_str <- list(1,2,'3')

# Named Lists
ar_st_names <- c('e1','e2','e3')
ls_num_str_named <- ls_num_str
names(ls_num_str_named) <- ar_st_names

# Add Element to Named List
ls_num_str_named$e4 <- 'this is added'

```

- r print input as string
- r print parameter code as string
- [How to convert variable \(object\) name into String](#)

The function below ffi_lst2str is also a function in REconTools: [ff_sup_lst2str](#).

```

# list to String printing function
ffi_lst2str <- function(ls_list, st_desc, bl_print=TRUE) {

  # string desc
  if(missing(st_desc)){
    st_desc <- deparse(substitute(ls_list))
  }

  # create string
  st_string_from_list = paste0(paste0(st_desc, ':'),
                               paste(names(ls_list), ls_list, sep="=", collapse=";" ))

  if (bl_print){

```

```

    print(st_string_from_list)
  }
}

# print full
ffi_lst2str(ls_num)

## [1] "ls_num:=1;=2;=3"
ffi_lst2str(ls_str)

## [1] "ls_str:=1;=2;=3"
ffi_lst2str(ls_num_str)

## [1] "ls_num_str:=1;=2;=3"
ffi_lst2str(ls_num_str_named)

## [1] "ls_num_str_named:e1=1;e2=2;e3=3;e4=this is added"
# print subset
ffi_lst2str(ls_num[2:3])

## [1] "ls_num[2:3]:=2;=3"
ffi_lst2str(ls_str[2:3])

## [1] "ls_str[2:3]:=2;=3"
ffi_lst2str(ls_num_str[2:4])

## [1] "ls_num_str[2:4]:=2;=3=NULL"
ffi_lst2str(ls_num_str_named[c('e2', 'e3', 'e4')])

## [1] "ls_num_str_named[c(\"e2\", \"e3\", \"e4\")]:e2=2;e3=3;e4=this is added"

```

1.3 Two Dimensional Unnamed List

Generate a multiple dimensional list:

1. Initiate with an N element empty list
2. Reshape list to M by Q
3. Fill list elements
4. Get list element by row and column number

List allows for different data types to be stored together.

Note that element specific names in named list are not preserved when the list is reshaped to be two dimensional. Two dimensional list, however, could have row and column names.

```

# Dimensions
it_M <- 2
it_Q <- 3
it_N <- it_M*it_Q

# Initiate an Empty MxQ=N element list
ls_2d_flat <- vector(mode = "list", length = it_N)
ls_2d <- ls_2d_flat

```

```

# Named flat
ls_2d_flat_named <- ls_2d_flat
names(ls_2d_flat_named) <- paste0('e',seq(1,it_N))
ls_2d_named <- ls_2d_flat_named

# Reshape
dim(ls_2d) <- c(it_M, it_Q)
# named 2d list can not carry 1d name after reshape
dim(ls_2d_named) <- c(it_M, it_Q)

```

Print Various objects generated above, print list flattened.

```

# display
ffi_lst2str(ls_2d_flat_named)

## [1] "ls_2d_flat_named:e1=NULL;e2=NULL;e3=NULL;e4=NULL;e5=NULL;e6=NULL"

# print(ls_2d_flat_named)
ffi_lst2str(ls_2d_named)

## [1] "ls_2d_named:=NULL;=NULL;=NULL;=NULL;=NULL;=NULL"

print(ls_2d_named)

##      [,1] [,2] [,3]
## [1,] NULL NULL NULL
## [2,] NULL NULL NULL

```

Select element from list:

```

# Select Values, double bracket to select from 2dim list
print('ls_2d[[1,2]]')

## [1] "ls_2d[[1,2]]"

print(ls_2d[[1,2]])

## NULL

```

1.4 Define Two Dimensional Named Llist

For naming two dimensional lists, *rowname* and *colname* does not work. Rather, we need to use *dimnames*. Note that in addition to *dimnames*, we can continue to have element specific names. Both can co-exist. But note that the element specific names are not preserved after dimension transform, so need to be redefined afterwards.

How to select an element of a two dimensional list:

1. row and column names: *dimnames*, `ls_2d_flat_named[['row2','col2']]`
2. named elements: *names*, `ls_2d_flat_named[['e5']]`
3. select by index: *index*, `ls_2d_flat_named[[5]]`
4. converted two dimensional named list to tibble/matrix

Neither *dimnames* nor *names* are required, but both can be used to select elements.

```

# Dimensions
it_M <- 3
it_Q <- 4
it_N <- it_M*it_Q

```

```

# Initiate an Empty MxQ=N element list
ls_2d_flat_named <- vector(mode = "list", length = it_N)
dim(ls_2d_flat_named) <- c(it_M, it_Q)

# Fill with values
for (it_Q_ctr in seq(1,it_Q)) {
  for (it_M_ctr in seq(1,it_M)) {
    # linear index
    ls_2d_flat_named[[it_M_ctr, it_Q_ctr]] <- (it_Q_ctr-1)*it_M+it_M_ctr
  }
}

# Replace row names, note rownames does not work
dimnames(ls_2d_flat_named)[[1]] <- paste0('row',seq(1,it_M))
dimnames(ls_2d_flat_named)[[2]] <- paste0('col',seq(1,it_Q))

# Element Specific Names
names(ls_2d_flat_named) <- paste0('e',seq(1,it_N))

# Convert to Matrix
tb_2d_flat_named <- as_tibble(ls_2d_flat_named) %>% unnest()
mt_2d_flat_named <- as.matrix(tb_2d_flat_named)

```

Print various objects generated above:

```

# These are not element names, can still name each element
# display
print('ls_2d_flat_named')

## [1] "ls_2d_flat_named"

print(ls_2d_flat_named)

##      col1 col2 col3 col4
## row1 1    4    7    10
## row2 2    5    8    11
## row3 3    6    9    12
## attr(,"names")
## [1] "e1" "e2" "e3" "e4" "e5" "e6" "e7" "e8" "e9" "e10" "e11" "e12"

print('tb_2d_flat_named')

## [1] "tb_2d_flat_named"

print(tb_2d_flat_named)
print('mt_2d_flat_named')

## [1] "mt_2d_flat_named"

print(mt_2d_flat_named)

##      col1 col2 col3 col4
## [1,] 1    4    7    10
## [2,] 2    5    8    11
## [3,] 3    6    9    12

```

Select elements from list:

```

# Select elements with with dimnames
ffi_lst2str(ls_2d_flat_named[['row2','col2']])

## [1] "ls_2d_flat_named[["row2\", \"col2\"]]:=5"

# Select elements with element names
ffi_lst2str(ls_2d_flat_named[['e5']])

## [1] "ls_2d_flat_named[["e5\"]]:=5"

# Select elements with index
ffi_lst2str(ls_2d_flat_named[[5]])

## [1] "ls_2d_flat_named[[5]]:=5"

```