R Examples Generate Tibble Dataframes

Fan Wang

2023-04-11

Contents

1 Generate Dataframe

Go to the **RMD**, **R**, **PDF**, or **HTML** version of this file. Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

1.1 Simple Dataframe, Name Columns

```
# 5 by 3 matrix
mt_rnorm_a <- matrix(rnorm(4,mean=0,sd=1), nrow=5, ncol=3)

# Column Names
ar_st_varnames <- c('id','var1','varb','vartheta')

# Combine to tibble, add name col1, col2, etc.
tb_combine <- as_tibble(mt_rnorm_a) %>%
  rowid_to_column(var = "id") %>%
  rename_all(~c(ar_st_varnames))

# Display
kable(tb_combine) %>% kable_styling_fc()
```

1.2 Dataframe with Row and Column Names and Export

First, we generate an empty matrix. Second, we compute values to fill in matrix cells.

```
# an NA matrix
it_nrow <- 5
it_ncol <- 3
```

id	var1	varb	vartheta
1	1.5587083	0.0705084	0.1292877
2	0.0705084	0.1292877	1.7150650
3	0.1292877	1.7150650	1.5587083
4	1.7150650	1.5587083	0.0705084
5	1.5587083	0.0705084	0.1292877

3	5	7
5	8	11
7	11	15
9	14	19
11	17	23

```
mt_na <- matrix(NA, nrow=it_nrow, ncol=it_ncol)

# array of nrow values
ar_it_nrow <- seq(1, it_nrow)
ar_it_ncol <- seq(1, it_ncol)

# Generate values in matrix
for (it_row in ar_it_nrow) {
  for (it_col in ar_it_ncol) {
    print(glue::glue("row={it_row} and col={it_col}"))
    mt_na[it_row, it_col] = it_row*it_col + it_row + it_col
  }
}
```

```
## row=1 and col=1
## row=1 and col=2
## row=1 and col=3
## row=2 and col=1
## row=2 and col=2
## row=2 and col=3
## row=3 and col=1
## row=3 and col=2
## row=3 and col=3
## row=4 and col=1
## row=4 and col=2
## row=4 and col=3
## row=5 and col=1
## row=5 and col=2
## row=5 and col=3
```

```
# Display
kable(mt_na) %>% kable_styling_fc()
```

Third, we label the rows and the columns. Note that we will include the column names as column names, but the row names will be included as a variable.

```
# Column Names
ar_st_col_names <- paste0('colval=', ar_it_ncol)
ar_st_row_names <- paste0('rowval=', ar_it_nrow)

# Create tibble, and add in column and row names
```

row_name	colval=1	colval=2	colval=3
rowval=1	3	5	7
rowval=2	5	8	11
rowval=3	7	11	15
rowval=4	9	14	19
rowval=5	11	17	23

```
tb_row_col_named <- as_tibble(mt_na) %>%
  rename_all(~c(ar_st_col_names)) %>%
  mutate(row_name = ar_st_row_names) %>%
  select(row_name, everything())

# Display
kable(tb_row_col_named) %>% kable_styling_fc()
```

Finally, we generate a file name for exporting this tibble to a CSV file. We create a file name with a time stamp.

```
# Create a file name with date stamp
st_datetime <- base::format(Sys.time(),  "%Y%m%d-%H%M%S")
# Copying a fixed date to avoid generating multiple testing files
# The date string below is generated by Sys.time()
st_snm_filename <- paste0("tibble_out_test_", st_datetime, '.csv')

# Create a file name with the time stamp.
spn_file_path = file.path(
  "C:", "Users", "fan",
  "R4Econ", "amto", "tibble", "_file",
  st_snm_filename,
  fsep = .Platform$file.sep)

# Save to file
write_csv(tb_row_col_named, spn_file_path)
```

1.3 Generate Tibble given Matrixes and Arrays

Given Arrays and Matrixes, Generate Tibble and Name Variables/Columns

- naming tibble columns
- tibble variable names
- dplyr rename tibble
- dplyr rename tibble all variables
- dplyr rename all columns by index
- dplyr tibble add index column
- see also: SO-51205520

```
# Base Inputs
ar_col <- c(-1,+1)
mt_rnorm_a <- matrix(rnorm(4,mean=0,sd=1), nrow=2, ncol=2)
mt_rnorm_b <- matrix(rnorm(4,mean=0,sd=1), nrow=2, ncol=4)

# Combine Matrix
mt_combine <- cbind(ar_col, mt_rnorm_a, mt_rnorm_b)
colnames(mt_combine) <- c('ar_col',
```

```
                          paste0('matcolvar_grpa_', seq(1,dim(mt_rnorm_a)[2])),
                          paste0('matcolvar_grpb_', seq(1,dim(mt_rnorm_b)[2])))

# Variable Names
ar_st_varnames <- c('var_one',
                    paste0('tibcolvar_ga_', c(1,2)),
                    paste0('tibcolvar_gb_', c(1,2,3,4)))

# Combine to tibble, add name col1, col2, etc.
tb_combine <- as_tibble(mt_combine) %>% rename_all(~c(ar_st_varnames))

# Add an index column to the dataframe, ID column
tb_combine <- tb_combine %>% rowid_to_column(var = "ID")

# Change all gb variable names
tb_combine <- tb_combine %>%
                  rename_at(vars(starts_with("tibcolvar_gb_")),
                            funs(str_replace(., "_gb_", "_gbrenamed_")))

# Tibble back to matrix
mt_tb_combine_back <- data.matrix(tb_combine)

# Display
kable(mt_combine) %>% kable_styling_fc_wide()
```

ar_col	matcolvar_grpa_1	matcolvar_grpa_2	matcolvar_grpb_1	matcolvar_grpb_2	matcolvar_grpb_3	matcolvar_grpb_4
-1	0.4609162	-0.6868529	1.2240818	0.4007715	1.2240818	0.4007715
1	-1.2650612	-0.4456620	0.3598138	0.1106827	0.3598138	0.1106827

```
kable(tb_combine) %>% kable_styling_fc_wide()
```

ID	var_one	tibcolvar_ga_1	tibcolvar_ga_2	tibcolvar_gbrenamed_1	tibcolvar_gbrenamed_2	tibcolvar_gbrenamed_3	tibcolvar_gbrenamed_4
1	-1	0.4609162	-0.6868529	1.2240818	0.4007715	1.2240818	0.4007715
2	1	-1.2650612	-0.4456620	0.3598138	0.1106827	0.3598138	0.1106827

```
kable(mt_tb_combine_back) %>% kable_styling_fc_wide()
```

ID	var_one	tibcolvar_ga_1	tibcolvar_ga_2	tibcolvar_gbrenamed_1	tibcolvar_gbrenamed_2	tibcolvar_gbrenamed_3	tibcolvar_gbrenamed_4
1	-1	0.4609162	-0.6868529	1.2240818	0.4007715	1.2240818	0.4007715
2	1	-1.2650612	-0.4456620	0.3598138	0.1106827	0.3598138	0.1106827

1.4 Generate a Table from Lists

We run some function, whose outputs are named list, we store the values of the named list as additional rows into a dataframe whose column names are the names from named list.

First, we generate the function that returns named lists.

```
# Define a function
ffi_list_generator <- function(it_seed=123) {
  set.seed(it_seed)
  fl_abc <- rnorm(1)
  ar_efg <- rnorm(3)
  st_word <- sample(LETTERS, 5, replace = TRUE)
  ls_return <- list("abc" = fl_abc, "efg" = ar_efg, "opq" = st_word)
  return(ls_return)
```

```
}
# Run the function
it_seed=123
ls_return <- ffi_list_generator(it_seed)
print(ls_return)
```

```
## $abc
## [1] -0.5604756
##
## $efg
## [1] -0.23017749  1.55870831  0.07050839
##
## $opq
## [1] "K" "E" "T" "N" "V"
```

Second, we list of lists by running the function above with different starting seeds. We store results in a two-dimensional list.

```
# Run function once to get length
ls_return_test <- ffi_list_generator(it_seed=123)
it_list_len <- length(ls_return_test)

# list of list frame
it_list_of_list_len <- 5
ls_ls_return <- vector(mode = "list", length = it_list_of_list_len*it_list_len)
dim(ls_ls_return) <- c(it_list_of_list_len, it_list_len)

# Fill list of list
ar_seeds <- seq(123, 123 + it_list_of_list_len - 1)
it_ctr <- 0
for (it_seed in ar_seeds) {
  print(it_seed)
  it_ctr <- it_ctr + 1
  ls_return <- ffi_list_generator(it_seed)
  ls_ls_return[it_ctr,] <- ls_return
}
```

```
## [1] 123
## [1] 124
## [1] 125
## [1] 126
## [1] 127
```

```
# print 2d list
print(ls_ls_return)
```

```
##      [,1]       [,2]      [,3]
## [1,] -0.5604756 numeric,3 character,5
## [2,] -1.385071  numeric,3 character,5
## [3,] 0.933327   numeric,3 character,5
## [4,] 0.366734   numeric,3 character,5
## [5,] -0.5677337 numeric,3 character,5
```

Third, we convert the list to a tibble dataframe. Prior to conversion we add names to the 1st and 2nd dimensions of the list. Then we print the results.

abc	efg	opq
-0.5604756	-0.23017749, 1.55870831, 0.07050839	K, E, T, N, V
-1.385071	0.03832318, -0.76303016, 0.21230614	J, A, O, T, N
0.933327	-0.52503178, 1.81443979, 0.08304562	C, T, M, S, K
0.366734	0.3964520, -0.7318437, 0.9462364	Z, L, J, Y, P
-0.5677337	-0.814760579, -0.493939596, 0.001818846	Y, C, O, F, U

```
# get names from named list
ar_st_names <- names(ls_return_test)
dimnames(ls_ls_return)[[2]] <- ar_st_names
dimnames(ls_ls_return)[[1]] <- paste0('seed_', ar_seeds)

# Convert to dataframe
tb_ls_ls_return <- as_tibble(ls_ls_return)

# print
kable(tb_ls_ls_return) %>% kable_styling_fc()
```

Fourth, to export list to csv file, we need to unlist list contents. See also Create a tibble containing list columns

```
# Unlist
tb_unlisted <- tb_ls_ls_return %>%
  rowwise() %>%
  mutate_if(is.list,
    funs(paste(unlist(.), sep='', collapse=', ')))
# print on screen, can see values
print(tb_unlisted)
```

1.5 Rename Tibble with Numeric Column Names

After reshaping, often could end up with variable names that are all numeric, intgers for example, how to rename these variables to add a common prefix for example.

```
# Base Inputs
ar_col <- c(-1,+1)
mt_rnorm_c <- matrix(rnorm(4,mean=0,sd=1), nrow=5, ncol=10)
mt_combine <- cbind(ar_col, mt_rnorm_c)

# Variable Names
ar_it_cols_ctr <- seq(1, dim(mt_rnorm_c)[2])
ar_st_varnames <- c('var_one', ar_it_cols_ctr)

# Combine to tibble, add name col1, col2, etc.
tb_combine <- as_tibble(mt_combine) %>% rename_all(~c(ar_st_varnames))

# Add an index column to the dataframe, ID column
tb_combine_ori <- tb_combine %>% rowid_to_column(var = "ID")

# Change all gb variable names
tb_combine <- tb_combine_ori %>%
                  rename_at(
                    vars(num_range('',ar_it_cols_ctr)),
                    funs(paste0("rho", . , 'var'))
```

```
                    )

# Display
kable(tb_combine_ori) %>% kable_styling_fc_wide()
```

ID	var_one	1	2	3	4	5	6	7	8	9	10
1	-1	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199
2	1	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199	0.1335086
3	-1	0.1335086	-0.1059632	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199	0.1335086	-0.1059632
4	1	-0.1059632	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472
5	-1	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199

```
kable(tb_combine) %>% kable_styling_fc_wide()
```

ID	var_one	rho1var	rho2var	rho3var	rho4var	rho5var	rho6var	rho7var	rho8var	rho9var	rho10var
1	-1	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199
2	1	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199	0.1335086
3	-1	0.1335086	-0.1059632	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199	0.1335086	-0.1059632
4	1	-0.1059632	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472
5	-1	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199	0.1335086	-0.1059632	-0.1255472	0.5646199

1.6 Tibble Row and Column and Summarize

Show what is in the table: 1, column and row names; 2, contents inside table.

```
tb_iris <- as_tibble(iris)
print(rownames(tb_iris))
```

```
##   [1] "1"   "2"   "3"   "4"   "5"   "6"   "7"   "8"   "9"   "10"  "11"  "12"  "13"
##  [14] "14"  "15"  "16"  "17"  "18"  "19"  "20"  "21"  "22"  "23"  "24"  "25"  "26"
##  [27] "27"  "28"  "29"  "30"  "31"  "32"  "33"  "34"  "35"  "36"  "37"  "38"  "39"
##  [40] "40"  "41"  "42"  "43"  "44"  "45"  "46"  "47"  "48"  "49"  "50"  "51"  "52"
##  [53] "53"  "54"  "55"  "56"  "57"  "58"  "59"  "60"  "61"  "62"  "63"  "64"  "65"
##  [66] "66"  "67"  "68"  "69"  "70"  "71"  "72"  "73"  "74"  "75"  "76"  "77"  "78"
##  [79] "79"  "80"  "81"  "82"  "83"  "84"  "85"  "86"  "87"  "88"  "89"  "90"  "91"
##  [92] "92"  "93"  "94"  "95"  "96"  "97"  "98"  "99"  "100" "101" "102" "103" "104"
## [105] "105" "106" "107" "108" "109" "110" "111" "112" "113" "114" "115" "116" "117"
## [118] "118" "119" "120" "121" "122" "123" "124" "125" "126" "127" "128" "129" "130"
## [131] "131" "132" "133" "134" "135" "136" "137" "138" "139" "140" "141" "142" "143"
## [144] "144" "145" "146" "147" "148" "149" "150"
```

```
colnames(tb_iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

```
colnames(tb_iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

```
summary(tb_iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

Species	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
setosa	4.3	3.0	1.1	0.1
setosa	4.4	2.9	1.4	0.2
setosa	4.4	3.0	1.3	0.2
setosa	4.4	3.2	1.3	0.2
setosa	4.5	2.3	1.3	0.3
setosa	4.6	3.1	1.5	0.2
setosa	4.6	3.4	1.4	0.3
setosa	4.6	3.6	1.0	0.2
setosa	4.6	3.2	1.4	0.2
setosa	4.7	3.2	1.3	0.2

Species	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
virginica	7.9	3.8	6.4	2.0
virginica	7.7	3.8	6.7	2.2
virginica	7.7	2.6	6.9	2.3
virginica	7.7	2.8	6.7	2.0
virginica	7.7	3.0	6.1	2.3
virginica	7.6	3.0	6.6	2.1
virginica	7.4	2.8	6.1	1.9
virginica	7.3	2.9	6.3	1.8
virginica	7.2	3.6	6.1	2.5
virginica	7.2	3.2	6.0	1.8

1.7 Sorting and Rank

1.7.1 Sorting

- dplyr arrange desc reverse
- dplyr sort

```
# Sort in Ascending Order
tb_iris %>% select(Species, Sepal.Length, everything()) %>%
  arrange(Species, Sepal.Length) %>% head(10) %>%
  kable() %>% kable_styling_fc()
```

```
# Sort in Descending Order
tb_iris %>% select(Species, Sepal.Length, everything()) %>%
  arrange(desc(Species), desc(Sepal.Length)) %>% head(10) %>%
  kable() %>% kable_styling_fc()
```

1.7.2 Create a Ranking Variable

We use dplyr's ranking functions to generate different types of ranking variables.

The example below demonstrates the differences between the functions row_number(), min_rank(), and dense_rank().

- *row_number*: Given 10 observations, generates index from 1 to 10, ties are given different ranks.
- *min_rank*: Given 10 observations, generates rank where second-rank ties are both given "silver", and the 4th highest ranked variable not given medal.
- *dense_rank*: Given 10 observations, generates rank where second-rank ties are both given "silver" (2nd rank), and the 4th highest ranked variable is given "bronze" (3rd rank), there are no gaps between ranks.

Note that we have "desc(var_name)" in order to generate the variable based on descending sort of the the

Ranking variable

Species	Sepal.Length	row_number	min_rank	dense_rank
setosa	5.1	2	2	2
setosa	4.9	5	5	4
setosa	4.7	7	7	5
setosa	4.6	8	8	6
setosa	5.0	3	3	3
setosa	5.4	1	1	1
setosa	4.6	9	8	6
setosa	5.0	4	3	3
setosa	4.4	10	10	7
setosa	4.9	6	5	4

"var_name" variable.

```
tb_iris %>%
  select(Species, Sepal.Length) %>% head(10) %>%
  mutate(row_number = row_number(desc(Sepal.Length)),
         min_rank = min_rank(desc(Sepal.Length)),
         dense_rank = dense_rank(desc(Sepal.Length))) %>%
  kable(caption = "Ranking variable") %>% kable_styling_fc()
```

1.8 REconTools Summarize over Tible

Use R4Econ's summary tool.

```
df_summ_stats <- REconTools::ff_summ_percentiles(tb_iris)
kable(t(df_summ_stats)) %>% kable_styling_fc_wide()
```

stats	n	unique	NAobs	ZEROobs	mean	sd	cv	min	p01	p05	p10	p25	p50	p75	p90	p95	p99	max
Petal.Length	150	43	0	0	3.758000	1.7652982	0.4697441	1.0	1.149	1.300	1.4	1.6	4.35	5.1	5.80	6.100	6.700	6.9
Petal.Width	150	22	0	0	1.199333	0.7622377	0.6355511	0.1	0.100	0.200	0.2	0.3	1.30	1.8	2.20	2.300	2.500	2.5
Sepal.Length	150	35	0	0	5.843333	0.8280661	0.1417113	4.3	4.400	4.600	4.8	5.1	5.80	6.4	6.90	7.255	7.700	7.9
Sepal.Width	150	23	0	0	3.057333	0.4358663	0.1425642	2.0	2.200	2.345	2.5	2.8	3.00	3.3	3.61	3.800	4.151	4.4