PP09

Dynamic Lookups in R

duong.tran@ext.elderbrooksolutions.com

elderbrook
linking minds
www.elderbrooksolution.com
+49 7142 339 2919

1 Introduction

The concept of creating or storing lookup tables in a catalog is a powerful technique. It simplifies program code, and makes it more readable. So, what is a lookup table? It is a key-value pair table of "one-to-one" or "many-to-one" mappings that assign data values to data labels. Such mappings are often a requirement prior to the analysis, summarization and presentation of the data. In SAS, lookups can be implemented via the PROC FORMAT and managed by the PROC CATALOG procedures. In Base R and Tidyverse, there is no such equivalent. There is such a concept, however, in the "**fmtr**" package. The purpose of this presentation is to introduce the "**fmtr**" package, and bring awareness of this concept of dynamic lookups to R programmers. In doing so, I will explain the process, and contrast the R equivalent to its counter-part in SAS. In doing so, I will demonstrate that R programmers too, now have this highly productive tool at their disposal.

Key words: fmtr, fapply, fcat, value, condition, factor

2 Demo Data

Here are the snippets of data used in some of the examples in this presentation.

sashelp.class - dataset

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0

mtcars - dataframe

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1

3 Lookups with SAS

The following is a brief summary of how SAS does lookups with IF/THEN ELSE statements, by associating a variable with a format and by using the PUT function.

```
proc format;
 value AGEFMT
     0 - <14 = "0-<14"
     14 - <16 = "14-<16"
     other    = ">=16"
     ;
 value $SEX "M" = "Male"
            "F"  = "Female"
            ;
run;
```

```
data class1;
 set sashelp.class;
* by using the IF/THEN ELSE statement;
      if ( 0 <= age <14) then Agegrp1 = "0-<14";
 else if (14 <= age <16) then Agegrp1 = "14-<16";
 else if (age >=16)      then Agegrp1 = ">=16";
* by  using the PUT function;
 Agegrp2 = put(age, AGEFMT.);
run;
```

```
data class2;
 * associating a format to the Age variable;
 format Age AGEFMT.;
 set sashelp.class;
run;
```

class1

Obs	Name	Sex	Age	Height	Weight	Agegrp1	Agegrp2
1	Alfred	M	14	69.0	112.5	14-<16	14-<16
2	Alice	F	13	56.5	84.0	0-<14	0-<14
3	Barbara	F	13	65.3	98.0	0-<14	0-<14

class2

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14-<16	69.0	112.5
2	Alice	F	0-<14	56.5	84.0
3	Barbara	F	0-<14	65.3	98.0

These examples are commonly seen in programs. However, there are other more powerful tools for creating lookups from a lookup-table and storing them in a catalog that can be recalled by different programs. For example, we can convert the lookup-table below into a stored format catalog with PROC FORMAT and the CNTLIN option and the reverse is also possible with CNTLOUT.

fmtdat (lookup-table)

FMTNAME	START	END	LABEL	TYPE	HLO
AGEFMT	0	14	0<14	N	
AGEFMT	14	16	14<16	N	
AGEFMT	**OTHER**	**OTHER**	>=16	N	O
SEX	F	F	Female	C	
SEX	M	F	Female	C	

```
*Creating formats from  a lookup table;
libname MYFMTDAT ".";
PROC FORMAT CNTLIN=MYFMTDAT.fmtdat;
RUN;

*And the reverse is done with the CLNTOUT;
PROC FORMAT CNTLOUT=MYFMTDAT.fmtdat;
RUN;
```

Furthermore, to add and delete a format and to make a copy of the catalog, we can use PROC CATALOG. This presentation is mainly about **fmtr** so refer to the SAS documentation for more details.

4 Lookups with BASE R and Tidyverse

Here are some examples of how to use **ifelse**, **if_else** and **case_when** functions to do lookup.

```
df <- mtcars[, c("mpg", "cyl")]  # select the necessary variables for demonstrate purpose
```

a. BASE R method

```
dfa <- df
dfa$cylcat1 = ifelse(dfa$cyl>4,"More than 4 cylinders","4 or less than 4 cylinders")
```

b. Tidyverse method

```
library(dplyr)
dfb <- df |> mutate(cylcat2 = if_else(cyl > 4, "More than 4 cylinders", "4 or less than 4 cylinders"),
                    cylcat3 = case_when(cyl > 4 ~ "More than 4 cylinders",.default =  "4 or less than 4 cylinders"))
```

dfa

mpg	cyl	cylcat1
21.0	6	More than 4 cylinders
21.0	6	More than 4 cylinders
22.8	4	4 or less than 4 cylinders

dfb

cylcat2	cylcat3
More than 4 cylinders	More than 4 cylinders
More than 4 cylinders	More than 4 cylinders
4 or less than 4 cylinders	4 or less than 4 cylinders

As you can see, these methods are rather cumbersome, especially when there are many categories to map. However there is now a much better way with **fmtr.**

5 Factors

Before we go on to **fmtr**, an important concept to understand is factors in R. A factor is a special type of vector that typically stores character or integer category values. It also stores customizable information about the characteristics of these values. These are the levels (the distinct values), their labels and whether they are ordered or not. Factors are often necessary for statistical modeling, presentation purposes in tables and use in plots. Below are examples of creating factors and their results.

a.

```
v <- c("L", "H")
v <- factor(v, ordered = TRUE)
```

```
str(v)      : Ord.factor w/ 2 levels "H"<"L": 2 1
levels(v)   : "H" "L"
v[1] < v[2]: FALSE
table(v)    : H L
              1 1
```

b.

```
v <- c("L", "H")
v <- factor(v, levels = c("L", "M", "H"),
      label = c("Low", "Med", "High"), ordered = TRUE)
```

```
str(v)      : Ord.factor w/ 3 levels "Low"<"Med"<"High": 1 3
levels(v)   : "Low" "Med" "High"
v[1] < v[2]: TRUE
table(v)    : Low Med High
              1   0   1
```

Note the subtle differences in the two examples - in (b) when the **levels** and **label** parameters are specified, this gives a desirable result. That is, the levels are ordered with a zero count forced for the "Med" category as intended. Whereas in (a) the categories are ordered alphabetically when no **levels** are specified.

6 Lookups with fmtr

6.1 Defining lookups

fmtr provides two functions to define lookups that are analogous to PROC FORMAT. The **condition** function maps an R expression to a label and the **value** function accepts one or more condition arguments. The **condition** function takes an expression and so is more flexible than how PROC FORMAT defines mappings. Once a mapping is defined, it can be used by the **fapply** function to decode, similar to the SAS **put** and **input** functions. Here is an outline of the process in contrast to SAS. Take note of how the **as.factor** and the **order** parameters affect the results' presentation.

value(..., log = TRUE, **as.factor** = FALSE); **condition**(expr, label, **order** = NULL); **fapply**(x, format = NULL, ..)

a. fmtr

```
library (sassy)
library (dplyr)

df1 <- data.frame(SEX = c("F", "M"))
fmt1 <- value(condition(x == "M", "Male",1),
  condition(x == "F", "Female",2),
  condition(TRUE, "Unknown", 3), as.factor = TRUE)

df2 <- df1 |> mutate(SEX  = fapply(SEX, fmt1))
proc_freq(df2, tables = "SEX")
```

Table of SEX

SEX	N	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Male	2	1	50.00	1	50.00
Female	2	1	50.00	2	100.00
Unknown	2	0	0.00	2	100.00

b. SAS proc format

```
proc format;
 value $ sex 'M' = 'Male'  'F' = 'Female' other = 'Unknown';
run;
data ds1; Sex = 'M'; output; Sex = 'F'; output; run;
data ds2; set ds1; Sex = put(Sex, $sex.); run;
proc summary data=ds2 nway completetypes;
 format Sex $sex.;
 class Sex / preloadfmt;
 output out=ds2(rename=(_FREQ_=Frequency));
run;
proc print data=ds2;
 var Sex Frequency;
run;
```

Obs	Sex	Frequency
1	Male	1
2	Female	1
3	Unknown	0

Note, **fapply** returns mapped-data to either a numeric or character type automatically, so there is no need for the equivalent of SAS-Informat.

6.2 Creating lookups from data

Defining too many lookups within a program can be messy, as mentioned previously. Hence, like PROC FORMAT, **fmtr** allows you to define and create lookups from a data-table. This means you can manage lookups external from the program. It also means you can share and re-use them more easily. Below is a lookup-table in Excel and steps to create and use lookups.

NAME	TYPE	EXPRESSION	LABEL	ORDER	FACTOR	ETYPE
SEX	U	F	Female	2	TRUE	Char
SEX	U	M	Male	1	TRUE	Char
AESEV	U	1	Mild	1	FALSE	Num
AESEV	U	2	Moderate	2	FALSE	Num
AESEV	U	3	Severe	3	FALSE	Num
LABRNG	U	x < 500	Low	1	TRUE	Exp
LABRNG	U	x >= 500	High	2	TRUE	Exp
Excel File: lookup.xlsx; Sheet: CONTERM						

a. Read lookup table

```
library(sassy)
library (readxl)
luptab <- read_excel("I:/Phuse2024/lookup.xlsx", sheet="CONTERM")
datastep(luptab,
   keep=v(Name, Type, Expression, Label, Order, Factor),{
  Name   = NAME
  Type   = "U"
  Label  = LABEL
  Order  = as.integer(ORDER)
  Factor = FACTOR
  if (ETYPE == "Char") {Expression = paste0("x == ", "'", EXPRESSION, "'")}
  else if (ETYPE == "Num"){Expression = paste0("x == ", EXPRESSION)}
  else if (ETYPE == "Exp"){Expression = EXPRESSION}}) -> fdat
```

The lookup **fdat** dataframe can then be converted to a lookup catalog with the **as.fcat** function.

b. Create a lookup calatog object: fc <- as.fcat(**fdat**)

fdat

Name	Type	Expression	Label	Order	Factor
SEX	U	X == "F"	Female	2	TRUE
SEX	U	X == "M"	Male	1	TRUE
AESEV	U	X == 1	Mild	1	FALSE
AESEV	U	X == 2	Moderate	2	FALSE
AESEV	U	X == 3	Severe	3	FALSE

c. View the SEX lookup definition: fc$SEX

Name	Type	Expression	Label	Order
x	U	x == "F"	Female	2
x	U	x == "M"	Male	1

d. Apply the SEX lookup: fapply(<var>, fc$SEX)

```
df1 <- data.frame(USUBJID = c("100"), SEX = c("M"))
df2 <- df1 |> mutate(SEX  = fapply(SEX, fc$SEX))
```

df2

USUBJID	SEX
100	Male

```
table(df2$SEX)
Male Female
   1      0
```

```
str(df2)
USUBJID: chr "100"
SEX    : Ord.factor w/ 2 levels "Male"<"Female": 1
```

6.3 Store and Read lookups catalogs

Again similar to SAS, you can store lookup definitions in multiple catalogs and read them with **fmtr**.

a. Convert the **fdat** dataframe into a catalog object

```
fmtdf <- as.data.frame(fmt1);  fdat <- rbind(fdat, fmtdf)
fc <- as.fcat(fdat)
```

b. Save the catalog object (fc) to a file

```
write.fcat(fc, dir_path = "I:/Phuse2024",  file_name = "cterm")
```

c. Read the catalog file

```
cterm <- read.fcat("I:/Phuse2024/cterm.fcat")
```

d. Use lookup within the catalog

```
v <- c("M", "F")
fapply(v, cterm$SEX)
```

6.4 Edit Catalogs and other Functionalities

To edit a catalog you can convert it to a dataframe with the **as.data.frame** function, edit it, and convert it back to a catalog with the **as.fcat** function. **fmtr** also has some other useful functions as shown with two examples here.

a. Edit catalogs

```
cterm <- read.fcat("I:/Phuse2024/cterm.fcat") |>
  as.data.frame() |>  filter(Name != "SEX")

fc <- as.fcat(cterm)
```

b. Some other useful functions

```
fmt_mean_sd(mtcars$mpg)
[1] "20.1 (6.0)"

fmt_range(mtcars$mpg, format = "%s", sep = "-")
[1] "10.4 - 33.9"
```

7 A Practical Application

In clinical trial reporting, there are many control terminologies (CT) to handle. Many of these are similar and shared among different studies. One such example shown below is the lab CT, which is commonly managed by Excel. We can make use of such lookup-tables and **fmtr** to map codes to labels with the technique mentioned above. For example, we can combine the necessary columns in the SDTM.LB to act as a unique key and map this to our chosen unique PARAMCD in the ADaM.ADLB and then map this to PARAM. This is not only efficient but also ensures the mappings are consistent among studies.

SDTM.LB				ADaM.ADLB	
LBCAT	LBTESTCD	LBTEST	LBUNIT	PARAMCD	PARAM
HEAMOTOLOGY	HGB	Heameglobin	g/dL	HGB	Heameglobin (g/dL)
CHEMISTRY	ALT	Alanine Aminotransferase	U/L	ALT	Alanine Aminotransferase (U/L)

LB key variables = LBCAT || LBTESTCD || LBUNIT => ADLB.PARAMCD => ADLB.PARAM

8 Conclusion

Dynamic lookups is a powerful technique in programming and, like SAS, we can now implement this in R also with **fmtr**. This presentation only introduces **fmtr** fundamentals, but there are other functionalities in this package for you to explore.

9 References

[1] https://www.lexjansen.com/pharmasug/2024/AP/PharmaSUG-2024-AP-229.pdf
[2] https://www.lexjansen.com/pharmasug/2024/AP/PharmaSUG-2024-AP-295.pdf