PP09

Dynamic Lookups in R

duong.tran@ext.elderbrooksolutions.com



The lookup **fdat** dataframe can then be converted to a

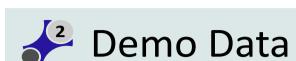
SEX : Ord.factor w/ 2 levels "Male"<"Female": 1

lookup catalog with the as.fcat function.

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



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

sashe	elp.class	- datas	set			mtcars - datafr	ame										
0bs	Name	Sex	Age	Height	Weight		mpg cy	1	disp	hp	drat	wt	qsec	vs	am	gear	carb
1	Alfred	M	14	69.0	112.5	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
2	Alice	F	13	56.5	84.0	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
3	Barbara	F	13	65.3	98.0	Datsun 710	22.8										

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;
                                                                                                data class2;
                                        data class1;
 value AGEFMT
                                        set sashelp.class;
                                                                                                 * associating a format to the Age variable;
    0 - <14 = "0-<14"
                                        * by using the IF/THEN ELSE statement;
                                                                                                 format Age AGEFMT.;
    14 - <16 = "14-<16"
                                             if (0 <= age <14) then Agegrp1 = "0-<14";
                                                                                                 set sashelp.class;
    other = ">=16"
                                         else if (14 <= age <16) then Agegrp1 = "14-<16";
                                         else if (age >=16) then Agegrp1 = ">=16";
 value $SEX "M" = "Male"
                                        * by using the PUT function;
             "F" = "Female'
                                         Agegrp2 = put(age, AGEFMT.);
run;
class1
                                                                                      class2
                                                                                                                                Weight
                                     112.5
                           69.0
                                              14-<16
                                                        14-<16
                                                                                                            14-<16 69.0
                                                                                                                                 84.0
                                              0-<14
                                                         0-<14
                                      98.0
                    13 65.3
                                                                                                                    65.3
                                              0-<14
                                                        0-<14
                                                                                                             0-<14
                                                                                                                                 98.0
                                                                                        3 Barbara
```

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.

	lookup-table	• •				*Creating formats from a lookup table;
FMTNAME	START	END	LABEL	TYPE	HLO	libname MYFMTDAT ".";
AGEFMT	0	14	0<14	N		PROC FORMAT CNTLIN=MYFMTDAT.fmtdat;
AGEFMT	14	16	14<16	N		RUN;
AGEFMT	**OTHER**	**OTHER**	>=16	N	0	*And the reverse is done with the CLNTOUT;
SEX	F	F	Female	С		PROC FORMAT CNTLOUT=MYFMTDAT.fmtdat;
SEX	M	F	Female	С		RUN;

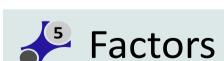
Furthermore, to add and delete a format and to make a copy of the catalog, we can use PROC CATALOG. This presentation is mainly about **fmtr** so refer to the SAS documentation for more details.

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 \sim "More than 4 cylinders",.default = "4 or less than 4 cylinders"))
dfa
                                                                                           cylcat3
                                                       cylcat2
                cylcat1
                More than 4 cylinders
                                                       More than 4 cylinders
                                                                                           More than 4 cylinders
                More than 4 cylinders
                                                       More than 4 cylinders
                                                                                           More than 4 cylinders
22.8
                4 or less 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.**



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.

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.

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.

(, log = TRUE, as.factor = FALSE); condition (expr, label, o	rder = NULL); fapply(x, format = NULL,)
a. fmtr	b. SAS proc format
library (sassy)	proc format;
library (dplyr)	value \$ sex 'M' = 'Male' 'F' = 'Female' other = 'Unknowr
	run;
df1 <- data.frame(SEX = c("F", "M"))	data ds1; Sex = 'M'; output; Sex = 'F'; output; run;
fmt1 <- value(condition(x == "M", "Male",1),	data ds2; set ds1; Sex = put(Sex, \$sex.); run;
condition(x == "F", "Female", 2),	proc summary data=ds2 nway completetypes;
condition(TRUE, "Unknown", 3), as.factor = TRUE)	format Sex \$sex.;
	class Sex / preloadfmt;
df2 <- df1 > mutate(SEX = fapply(SEX, fmt1))	output out=ds2(rename=(_FREQ_=Frequency));
proc_freq(df2, tables = "SEX")	run;
Table of SEX	proc print data=ds2; Obs Sex Frequency 1 Male 1
SEX N Frequency Percent Cumulative Cumulative Frequency Percent	var Sex Frequency; 2 Female 1
Male 2 1 50.00 1 50.00 Female 2 1 50.00 2 100.00	run; 0
Unknown 2 0 0.00 2 100.00	

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

EXPRESSION LABEL

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.

TRUE

JLA	l o	' ' '	iviaic	1*	IIIOL	Citai		•	•				
AESEV	U	1	Mild	1	FALSE	Num		b. Create a	lookur	calatog obje	ct: fc <- a	as.fcat(fdat)
AESEV	U	2	Moderate	2	FALSE	Num		fdat	•	g ,		,	,
AESEV	U	3	Severe	3	FALSE	Num			П	Warmana a a i a m	Tabal	Omdom	Es at an
LABRNG	U	x < 500	Low	1	TRUE	Exp		Name	Type	Expression	Label	Order	Factor
LABRNG	U	x >= 500	High	2	TRUE	Exp		SEX	U	X == "F"	Female	2	TRUE
Excel File: I	ookup.xlsx;	Sheet: CONTE	ERM	_		_		SEX AESEV	U	X == "M" X == 1	Male Mild	1	TRUE
a. Read I	ookun ta	hle						AESEV AESEV	IJ	X == 2	Moderat	e 2	FALSE FALSE
	•	DIC						AESEV	Ü	X == 3	Severe	3	FALSE
library(sa	assy)							c. View the	SEX lo	okup definitio	on: fc\$SE	Χ	
library (r	eadxl)							Name	Type	Expression		bel	Order
luptab <-	read_ex	cel("I:/Phus	se2024/loc	kup.xls	x", shee	t="CONTER	RM")	X	U	x == "F"	Fe	male	2
datastep	(luptab,							X	U	x == "M"	Mai	le	1
keep=\	/(Name,	Type, Expre	ssion, Labe	el, Orde	r, Factoi	r),{		d Apply the	SEV I	ookup: fapply((<var> fo</var>	-¢¢EY)	
Name	= NAME	=						u. Apply the	S JLX IC	окир. таррту	,~vai~, ic	,JJLK)	
		-						df1 <- da	ata.frar	ne(USUBJID =	c("100"), SEX =	c("M")
Type	= "U"									•	•		` .
Label	= LABEL							012 <- 01	T > m	utate(SEX = f	apply(SE	.Χ, ΙϹϸϽ	EXII
Order	= as.inte	ger(ORDER)					df2			table(df2	2\$SEX)	
		•	,					USUBJID	SEX		Male	Female	
Factor	= FACTO	ĸ						100	Male		1	0	
if (ETY	PE == "Cl	nar") {Expre	ssion = pa	ste0("x	== ", "'"	, EXPRESSION	ON, "'")}	/ 150)					

6.3 Store and Read lookups catalogs

else if (ETYPE == "Num"){Expression = paste0("x == ", EXPRESSION)}

else if (ETYPE == "Exp"){Expression = EXPRESSION}}) -> fdat

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

ne catalogo alla read them with milit
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)

str(df2)

USUBJID: chr "100"

6.4 Edit Catalogs and other Functionalities

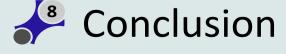
To edit a catalog you can convert it to a dataframe with the **as.data.frame** function, edit it, and convert it back to a catalog with the **as.fcat** function. **fmtr** also has some other useful functions as shown with two examples here.

<pre>a. Edit catalogs cterm <- read.fcat("I:/Phuse2024/cterm.fcat") ></pre>	<pre>b. Some other useful functions fmt_mean_sd(mtcars\$mpg)</pre>
as.data.frame() > filter(Name != "SEX")	[1] "20.1 (6.0)"
fc <- as.fcat(cterm)	<pre>fmt_range(mtcars\$mpg, format = "%s", sep = "-") [1] "10.4 - 33.9"</pre>

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.

BCAT	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)



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.



- [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