

R syntax for SAS programmers

Max Cherny

Agenda



-
- Introduction to R and Tidyverse
 - SAS vs. R syntax examples
 - Conclusion

-
- R is a free software for statistical analysis
 - Approx. 15,000 packages (as of 2020)
 - Tidyverse is a set of R packages
 - Developed by Hadley Wickham, an R Foundation member and RStudio employee
 - Streamlines R code
 - All functions have single purpose
 - More similar to SAS than Base R

The screenshot shows the RStudio interface with the Environment pane active. The top tabs are 'Environment', 'History', and 'Connections'. The top toolbar includes icons for saving, importing datasets, and a search icon. The Environment pane displays the 'Global Environment' with a search bar and a 'List' button. Below the search bar, the 'Data' section lists two objects: 'test' (6 obs. of 3 variables) and 'vitals' (6 obs. of 5 variables). Each object has a corresponding icon (a blue circle with a white 'i' for 'test' and a blue circle with a white 'v' for 'vitals').

Object	Variables	Observations
test	3	6
vitals	5	6

The screenshot displays a Jupyter Notebook interface with a line plot. The plot shows two data series, Group 1 and Group 2, across three x-values: 1.0, 2.0, and 3.0. Group 1 is represented by a solid black line with circular markers, and Group 2 is represented by a dashed black line with triangular markers. The y-axis ranges from 5 to 20, and the x-axis ranges from 1.0 to 3.0. The legend on the right indicates the line styles and markers for each group.

x	Group 1 (y)	Group 2 (y)
1.0	22	8
2.0	11	4
3.0	7	5

Creating a dataset (referred to as DATA FRAME in R)

SAS	R
<pre>data vitals; input subjid visit dia sys wt; datalines; 1 1 67 113 170 1 2 70 114 160 ; run;</pre>	<pre>install.packages("tidyverse") library(tidyverse) vitals<-tibble(subjid=c(1, 1), visit=c(1, 2), dia=c(67, 70), sys=c(113, 114), wt=c(170, 160))</pre>

subjid	visit	dia	sys	wt
1	1	67	113	170
1	2	70	114	160

Viewing data in a data frame

subjid	visit	dia	sys	wt
1	1	67	113	170
2	2	70	114	160

SAS

```
proc print data=vitals;  
run;
```

R

```
print (vitals)
```

```
view(vitals)
```

```
# A tibble: 2 x 5  
  subjid visit    dia    sys    wt  
  <dbl> <dbl> <dbl> <dbl> <dbl>  
1     1     1     67    113    170  
2     1     2     70    114    160
```

	subjid	visit	dia	sys	wt
1	1	1	67	113	170
2	1	2	70	114	160

Copying a data frame

subjid	visit	dia	sys	wt
1	1	67	113	170
2	2	70	114	160

SAS	R
<pre>data vitals2; set vitals; run;</pre>	<pre>vitals2<- vitals</pre>

subjid	visit	dia	sys	wt
1	1	67	113	170
2	2	70	114	160

Deleting a data frame

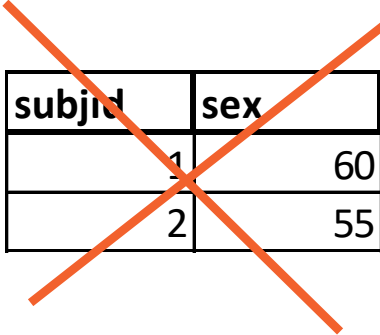
subjid	sex
1	60
2	55

SAS

```
proc datasets nolist;  
  delete demo;  
run;
```

R

```
rm (demo)
```



subjid	sex
1	60
2	55

Labeling variables

subjid	sex
1	F
2	M

SAS

```
data demo2;  
  set demo;  
  label subjid="Subject ID"  
        sex="Gender";  
run;
```

R

```
install.packages("labelled")  
library("labelled")  
var_label(demo) <- list(subjid="Subject ID",  
                        sex="Gender")
```

subjid (Subject ID)	sex (Gender)
1	F
2	M

Dropping a variable

subjid	visit	dia	sys	wt
1	1	67	113	170
2	2	70	114	160

SAS

```
data vitals2;  
  set vitals;  
  drop sys;  
run;
```

R

```
vitals2<-select (vitals, -sys)
```

subjid	visit	dia	wt
1	1	67	170
2	2	70	160

Deriving new variables

subjid	visit	dia	sys	wt
1	1	67	113	170
1	2	70	114	160

SAS

```
data vitals2;  
  set vitals;  
  wt_kg=wt/2.2;  
  unit="kg";  
run;
```

R

```
vitals2<-mutate(vitals,  
  wt_kg=wt/2.2,  
  unit="kg")
```

subjid	visit	dia	sys	wt	wt_kg	unit
1	1	67	113	170	77.27273	kg
1	2	70	114	160	72.72727	kg

Converting data types

char	num
1	9999
222	44

SAS

```
data test2;  
  set test;  
  char_to_num=input(char, 8.);  
  num_to_char=put(num, 8. -L);  
run;
```

R

```
test2<-mutate(test,  
  char_to_num=as.numeric(char),  
  num_to_char=as.character(num))
```

char	num	char_to_num	num_to_char
1	9999	1	9999
222	44	222	44

Creating a flag variable

subjid	visit	dia	sys	wt
1	1	67	113	170
1	2	70	114	160

SAS	R
<pre>data vitals2; set vitals; length sys_flag \$15; if sys > 113 then sys_flag="High"; else sys_flag="Low or Normal"; run;</pre>	<pre>vitals2<-mutate(vitals, sys_flag=cut(sys, c(-Inf, 113, Inf), labels=c("Low or Normal","High")))</pre>

subjid	visit	dia	sys	wt	sys_flag	
1	1	67	113	170	Low or Normal	
1	2	70	114	160	High	

Subsetting a data frame

subjid	visit	dia	sys	wt
1	1	67	113	170
2	2	70	114	160

SAS

```
data vitals2;  
  set vitals;  
  where subjid=2;  
run;
```

R

```
vitals2<-filter (vitals, subjid==2)
```

subjid	visit	dia	sys	wt
2	2	70	114	160

Sorting a data frame

subjid	age
2	67
4	54
3	76
1	75

SAS

```
proc sort data=demo out=demo2;  
  by subjid;  
run;
```

R

```
demo2<-arrange(demo, subjid)
```

subjid	age
1	75
2	67
3	76
4	54

Stacking two data frames together

subjid	age
1	60
2	55

+

subjid	age
3	20
4	30

SAS

```
data alldemo;  
  set demo1  
      demo2;  
run;
```

R

```
alldemo<-bind_rows(demo1, demo2)
```

subjid	age
1	60
2	55
3	20
4	30

Merging two data frames together

subjid	age	+	subjid	visit	dia	sys	wt
1	60		1	1	67	113	170
2	55		3	2	70	114	160

SAS

```
data vitals2;  
  merge demo (in=a)  
        vitals (in=b);  
  by subjid; if a and b;  
run;
```

R

```
vitals2<-inner_join(demo,vitals,by="subjid")
```

subjid	visit	dia	sys	wt	age
1	1	67	113	170	60

Other types of joins include LEFT, RIGHT, FULL_JOIN, NEST and ANTI joins

Concatenating variables

trtgrp	trt_counts
Drug A	50
Drug B	55

SAS

```
data results2;  
  set results;  
  length trt_text $14;  
  trt_text=cat(trtgrp, ' (N=', trt_counts, ')');  
run;
```

R

```
results2<-mutate(results,  
  trt_text=str_c(trtgrp,  
    sep=" ",  
    '(N=',trt_counts, ')'))
```

trtgrp	trt_counts	trt_text
Drug A	50	Drug A (N=50)
Drug B	55	Drug B (N=55)

Transposing a data frame (from long to wide)

subjid	visit	dia
1	1	77
1	2	76
2	1	82
2	2	80

SAS

```
proc transpose data=vitals out=tr_dia;  
  by subjid;  
  var dia;  
run;
```

R

```
tr_dia<-pivot_wider(vitals,  
  names_from=visit,  
  names_prefix = "visit:",  
  values_from=c("dia"))
```

subjid	visit:1	visit:2
1	77	76
2	82	80

Transposing a data frame (from wide to long)

subjid	visit	dia	sys
1	1	67	113
1	2	70	114

SAS

```
proc transpose data=vitals out=vitals2  
(rename=(_name_=paramcd col1=aval)) ;  
by subjid visit;  
run;
```

R

```
vitals2<-pivot_longer(vitals,  
                      cols=c(dia,sys),  
                      names_to = "paramcd")
```

subjid	visit	paramcd	aval
1	1	dia	67
1	1	sys	113
1	2	dia	70
1	2	sys	114

Removing duplicate records

subjid	visit	dia	sys	wt
1	1	67	113	170
1	2	65	113	170
1	2	70	114	160

SAS

```
proc sort data=vitals out=no_dups  
nodupkey;  
  by subjid;  
run;
```

R

```
no_dups<-distinct(vitals, subjid,  
  .keep_all=TRUE)
```

subjid	visit	dia	sys	wt
1	1	67	113	170

Using pipes in R

subjid	wt
13	170
5	160
7	200

SAS

```
proc sort data=vitals_wt out=vitals2;  
  by subjid;  
run;  
data vitals2;  
  set vitals2;  
  wt_kg=wt/2.2;  
  unit='kg';  
run;
```

R

```
vitals2<-vitals_wt %>%  
# sort data  
  arrange(subjid) %>%  
# derive weight in kilograms  
  mutate(wt_kg=wt/2.2,unit="kg")
```

subjid	wt	wt_kg	unit
5	160	72.7	kg
7	200	90.9	kg
13	170	77.27	kg

Summarizing statistics

subjid	visit	dia
1	1	67
1	2	70
2	1	71
2	2	72

SAS

```
proc summary data=vitals noprint ;  
  var dia;  
  output out=max_dia  
    (keep=subjid max) max=max ;  
  by subjid;  
run;
```

subjid	max
1	70
2	72

R

```
max_dia<-group_by(vitals, subjid) %>%  
  summarize( max=max(dia) )
```

Comparing two data frames

subjid	sex
1	60
2	55

subjid	sex
1	61
2	55

SAS

```
proc compare base=demo1  
compare=demo2;  
run;
```

R

```
all.equal(demo1, demo2)
```

Value Comparison Results for Variables

Obs	Base AGE	Compare AGE	Diff.	% Diff
1	60.0000	61.0000	1.0000	1.6667

```
> all.equal(demo1, demo2)  
[1] "Rows in x but not y: 1. Rows in y but not x: 1. "  
> |
```

Comparing two data frames using DIFFDF package

subjid	sex
1	60
2	55

subjid	sex
1	61
2	55

R

```
install.packages("diffdf")  
library(diffdf)  
diffdf(demo1, demo2)
```

Differences found between the objects!

A summary is given below.

Not all values Compared Equal
All rows are shown in table below

Variable	No of Differences
age	1

All rows are shown in table below

VARIABLE	..ROWNUMBER..	BASE	COMPARE
age	1	60	61

Warning message:
In diffdf(demo1, demo2) :
Not all values Compared Equal

> |

Macros and Functions

SAS macro

```
%macro combine_two_datasets(data1, data2);  
data alldata;  
  set &data1  
      &data2;  
run;  
%mend;  
  
%combine_two_datasets(demo1, demo2);
```

R functions

```
combine_two_datasets<-function  
(data1, data2)  
{  
  alldata<-rbind(data1, data2)  
}  
  
combine_two_datasets(demo1, demo2)
```

subjid	sex
1	60
2	55

+

subjid	sex
3	20
4	30



subjid	sex
1	60
2	55
3	20
4	30

Conclusions



-
- The Tidyverse package makes R more similar to SAS and easier to use.
 - Despite differences in language syntax, most of the statistical programmer's typical SAS code can be easily converted into R code.
 - The next step in learning R would be to attempt to recreate more complex SAS programs in R.

ACKNOWLEDGMENTS

Author would like to thank Michael Rimler, Jessica Mandel, David Wade, Eric Simms and Andy Nicholls for their help with this paper.

Questions