Random Number Generators, Special Functions LAG and RETAIN, FIRST./LAST.variable

STAT 3505

Week 9 (March 14, 2024)

Gunes Fleming Ph.D.

FIRST.variable and LAST.variable

- The FIRST.variable and LAST.variable automatic variables are available when you are using a BY statement in a DATA step.
- These are are particularly useful when dealing with sorted data or when you want to perform certain calculations or actions only on the first or last observation within a group.
- The FIRST.variable will have a value of 1 when SAS is processing an observation with the first occurrence of a new value for that variable and a value of 0 for the other observations.
- The LAST.variable will have a value of 1 for an observation with the last occurrence of a value for that variable and the value 0 for the other observations.

FIRST.variable and LAST.variable: Example

- SchoolData contains Subject ID, Test Dates, Score1, Score2, Score3 and Term variables.
- To extract last term records for each subject:

```
proc sort data=SchoolData;
  by subjid term;
run;
data SchoolData2;
  set SchoolData;
  by subjid term;
  if last.subjid;
run;
```

Obs	SUBJID	test_date	score1	score2	score3	term
1	1	01OCT2020	73	55	68	1
2	1	09JAN2021	94	72	66	2
3	1	19APR2021	82	54	54	3
4	2	01OCT2020	135	57	54	1
5	2	09JAN2021	81	93	84	2
6	2	19APR2021	135	96	74	3
7	3	01OCT2020	115	84	94	1
8	3	09JAN2021	124	81	91	2
9	3	19APR2021	81	94	86	3
10	4	01OCT2020	127	55	55	1
11	4	09JAN2021	75	98	75	2
12	4	19APR2021	77	69	65	3
13	5	01OCT2020	96	59	92	1
14	5	09JAN2021	134	107	64	2
15	5	19APR2021	138	82	65	3

LAG Function

- In SAS, the LAG function returns the value of its argument from the last time the LAG function executed.
- Often, LAG is used to retrieve the value of a variable from the previous observation within a BY group.
- It is useful fpr calculating differences between consecutive values, identifying trends, or detecting changes in a variable over time etc.

LAG Function: Example

- Get the difference between DBP value at each visit between the previous visit.

```
proc sort data=VitalsData; by subjid visit; run;
data VitalsData3;
  set VitalsData;
  by subjid;
  difference = dbp - lag(dbp);
  if not first.subjid;
run;
```

RETAIN Statement

- When SAS reads the DATA statement at the beginning of each iteration of the DATA step, SAS places missing values in the program data vector for variables that were assigned by either an INPUT statement or an assignment statement within the DATA step.
- A RETAIN statement effectively overrides this default.
- That is, a RETAIN statement tells SAS not to set variables whose values are assigned by an INPUT or assignment statement to missing when going from the current iteration of the DATA step to the next. Instead, SAS retains the values.
- You can specify as few or as many variables as you want in a RETAIN statement.

Random Number Generators

- Random number generators (pseudorandom number generators) in SAS are functions or procedures used to generate pseudo-random numbers.
- These numbers are typically used in different statistical and simulation tasks, such as Monte Carlo simulations, random sampling, and random assignment.
- Random number generators require an initial number, called a **seed**, that is used to calculate the first random number.

Random Number Generators

- The seed must be a nonnegative number less than 2,147,483,647.
- A given seed always produces the same results. That is, using the same seed, a random number generator would select the same observations.
- If you choose 0 as the seed, then the computer clock time at execution is used. In this case, it is very unlikely that a random number generator would produce the same results.
- Note that it is common practice when conducting research to use a non-zero seed, so that the results could be reproduced if necessary.

Random Number Generators

- **RANUNI** function generates uniform random numbers in the range from 0 to 1.
  - Syntax: RANUNI(seed)
- **RANNOR** function generates a series of random numbers from a standard normal distribution. You can scale these values to any mean and standard deviation.
  - Syntax: RANNOR(seed)
- **RAND** function can be used to generate random numbers from various distributions.
  - Syntax: RAND(distribution, parameter1, .., parameterk)

Random Number Generators: **RAND** Function

Distribution	Function Call	Parameter Values
Bernoulli	`RAND('BERNoulli', prob)`	where (0 <= prob <= 1)
Beta	`RAND('BETA', alpha, beta)`	where (0.01 <= alpha <= 1.5e6) and (0.20 <= beta <= 1.5e6)
Binomial	`RAND('BINOmial', prob, trials)`	where (0 <= prob <= 1) and (0 <= trials)
Cauchy	`RAND('CAUChy')`	
Chi-Square(d)	`RAND('CHISquare', df)`	where 0.03 <= df <= 4e9
Erlang	`RAND('ERLAng', alpha)`	where 1 <= alpha <= 2e9
Exponential	`RAND('EXPOnential', scale )`	where 0 <= scale <= 1e300

SAS documentation

Random Number Generators

MORE ON RANDOM NUMBER GENERATORS NEXT WEEK!