PharmaSUG 2022 - Paper SA-104

How RANK are your deciles? Using PROC RANK and PROC MEANS to create deciles based on observations and numeric values

Lisa Mendez, PhD, Emerge Solutions Group

ABSTRACT

For many cases using PROC RANK to create deciles works sufficiently, but occasionally, you find that it does not work for your needs. PROC RANK uses number of observations to produce a rank; however, if you need weighted percentiles then PROC RANK will not work. Instead, you can use Proc Means to successfully create weighted percent groups. This paper will illustrate the basic usage of PROC RANK and how to use PROC MEANS for the alternative. The paper will utilize BASE SAS® 9.4 code and will use a fictional dataset that provides the total number of prescriptions written by providers for two years. All levels of SAS users may benefit from the information provided in this paper.

INTRODUCTION

According to Investopedia.com, a decile is a quantitative method of splitting up a set of ranked data into 10 equal subsections. This type of data ranking is performed as part of many statistical studies in many different industries. The data may be ranked from smallest to largest, or vice versa. Sometimes a decile, which has 10 categorial buckets, may also be contrasted with percentiles (100), quartiles (4), or quintiles (5). Proc Rank computes ranks from one or more numeric variables across observations in a data set (Bilenas, 2009).

The difference in methods to create a list of the "top 10 percent of prescribers" came about while I was doing analysis for two different program managers. Each method provides information pertinent to the program manager, but from different perspectives. Each method is valid but must be explained to ensure the output is used appropriately. This paper will look at each method and will ensure the question is answered properly.

THE QUESTION

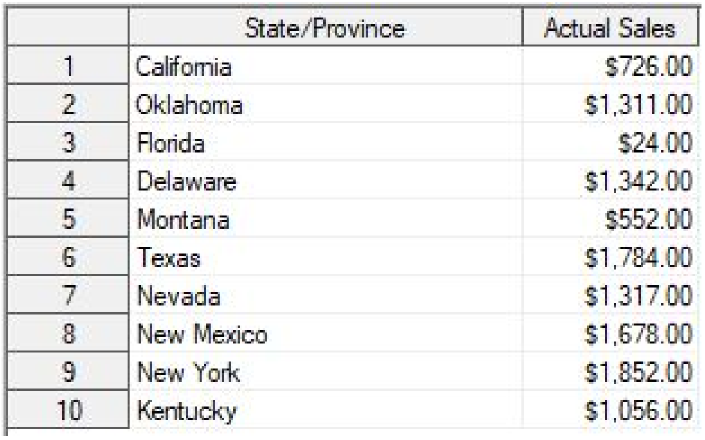
How many providers prescribed the top 10% of prescriptions for a particular drug for 2021? List the providers and total number of prescriptions prescribed for each provider for the year.

This question seems straightforward; however, the way to calculate the "top 10%" was interpreted differently by two different project managers for two different clients. The same data was used, but one project manager didn't want the "same number of prescribers in each decile". He stated that "last year we didn't give them the data like that." After much discussion, it was determined that he wanted weighted percentiles. It was determined that we could use PROC RANK for traditional deciles but would have to find another way to deal with weighted percentiles.

PROC RANK

How does PROC Rank work? SASSupport.com states "The RANK procedure computes ranks for one or more numeric variables across the observations of a SAS data set and outputs the ranks to a new SAS data set. PROC RANK by itself produces no printed output." The RANK procedure creates a separate data set with a new variable that captures the rank of the observation. PROC RANK has many options, such as specifying the order of ranks, handling ties in variable values, and it can generate variable bins or groupings based on the specification of the GROUPS option (Bilenas, 2009).

Let's take a quick look at an example of PROC Rank SAS Code and its output. Figure 1 is an example of a data set named *pr_sales*.



	State/Province	Actual Sales
1	California	$726.00
2	Oklahoma	$1,311.00
3	Florida	$24.00
4	Delaware	$1,342.00
5	Montana	$552.00
6	Texas	$1,784.00
7	Nevada	$1,317.00
8	New Mexico	$1,678.00
9	New York	$1,852.00
10	Kentucky	$1,056.00

Figure 1. pr_sales data set example

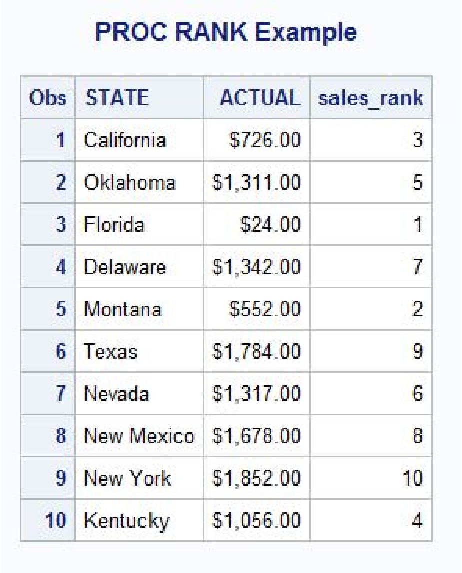
This file is just an extract of 10 observations for 10 different states and their corresponding actual sales for a furniture company. We will use PROC RANK to rank the sales:

```
proc rank data = pharmsug.pr_sales /* input data set */
              out = pr_sales_rank; /* output data set */
      var actual; /* variable to base rank */
      ranks sales_rank; /* new variable name of the rank */

run;

title 'PROC RANK Example';  /* title for proc print */
proc print data = pr_sales_rank; /* input data set */
run;
```

Figure 2 shows the Results of the Proc Print. The default in SAS is to replace the Actual_Sales value with the rank assignment, but we can use the Ranks statement to create a new variable to keep the original data.



PROC RANK Example

Obs	STATE	ACTUAL	sales_rank
1	California	$726.00	3
2	Oklahoma	$1,311.00	5
3	Florida	$24.00	1
4	Delaware	$1,342.00	7
5	Montana	$552.00	2
6	Texas	$1,784.00	9
7	Nevada	$1,317.00	6
8	New Mexico	$1,678.00	8
9	New York	$1,852.00	10
10	Kentucky	$1,056.00	4

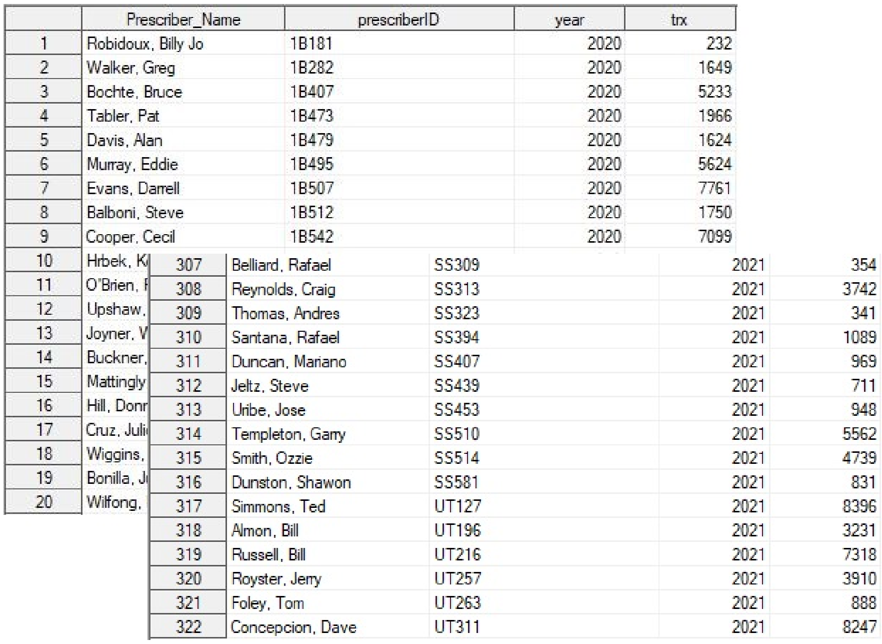
Figure 2. Results of PROC RANK example

You can also rank observations by descending order if needed by adding the option in the PROC RANK statement:

```
proc rank data = pharmsug.pr_sales /* input data set */
             out = pr_sales_rank /* output data set */ descending;
      var actual; /* variable to base rank */
      ranks sales_rank; /* new variable name of the rank */

run;
```

Now that we have the basic idea, let's continue by looking at PROC RANK using a dummy dataset that mimics the dataset of prescribers of a specific drug for a specific year. Figure 3 is a sample of data of a dummy data set. It has four variables: Prescriber_Name, prescriberID, Year, and TRX. The prescriberID is a unique ID for each prescriber and TRX represents the total number of prescriptions written for a particular drug (not listed in the dataset) by the prescriber for two years. Note, this is a fictional dataset. The dummy data set has 322 observations.



	Prescriber_Name	prescriberID	year	trx
1	Robidoux, Billy Jo	1B181	2020	232
2	Walker, Greg	1B282	2020	1649
3	Bochte, Bruce	1B407	2020	5233
4	Tabler, Pat	1B473	2020	1966
5	Davis, Alan	1B479	2020	1624
6	Murray, Eddie	1B495	2020	5624
7	Evans, Darrell	1B507	2020	7761
8	Balboni, Steve	1B512	2020	1750
9	Cooper, Cecil	1B542	2020	7099
10	Hrbek, K			
11	O'Brien,			
12	Upshaw,			
13	Joyner, W			
14	Buckner,			
15	Mattingly			
16	Hill, Donn			
17	Cruz, Juli			
18	Wiggins,			
19	Bonilla, J			
20	Wilfong,			
307	Belliard, Rafael	SS309	2021	354
308	Reynolds, Craig	SS313	2021	3742
309	Thomas, Andres	SS323	2021	341
310	Santana, Rafael	SS394	2021	1089
311	Duncan, Mariano	SS407	2021	969
312	Jeltz, Steve	SS439	2021	711
313	Uribe, Jose	SS453	2021	948
314	Templeton, Garry	SS510	2021	5562
315	Smith, Ozzie	SS514	2021	4739
316	Dunston, Shawon	SS581	2021	831
317	Simmons, Ted	UT127	2021	8396
318	Almon, Bill	UT196	2021	3231
319	Russell, Bill	UT216	2021	7318
320	Royster, Jerry	UT257	2021	3910
321	Foley, Tom	UT263	2021	888
322	Concepcion, Dave	UT311	2021	8247

Figure 3. Sample dummy prescriber data

The first part of the question is: How many providers prescribed the top 10% of prescriptions for a particular drug for 2021? We can use PROC Rank to rank each prescriber's TRX for the year 2021:

```
proc rank data=PharmSUG.Provider_trx_dummy /* input data set */
             out = Provider_trx_rank /* output data set */
             ties=low /* option to deal with ties */
             groups = 10; /* option to group ranks */
      by year;
      var trx;
      ranks trx_rank; /* new variable name of the rank assignment */
run;
```

The SAS code above uses two additional options: TIES and GROUPS. TIES= specifies how to compute normal scores or ranks for tied data values. GROUPS= assigns group values ranging from 0 to number-of-groups minus 1.

Option TIES

Default is MEANS	the mean rank is returned for tied values
Ties = low	the tied values are assigned to the lower rank
Ties = high	tied values are assigned to the higher rank
Ties = dense	the ranks are consecutive integers that begin with 1 end with the number of unique values of the VAR variable

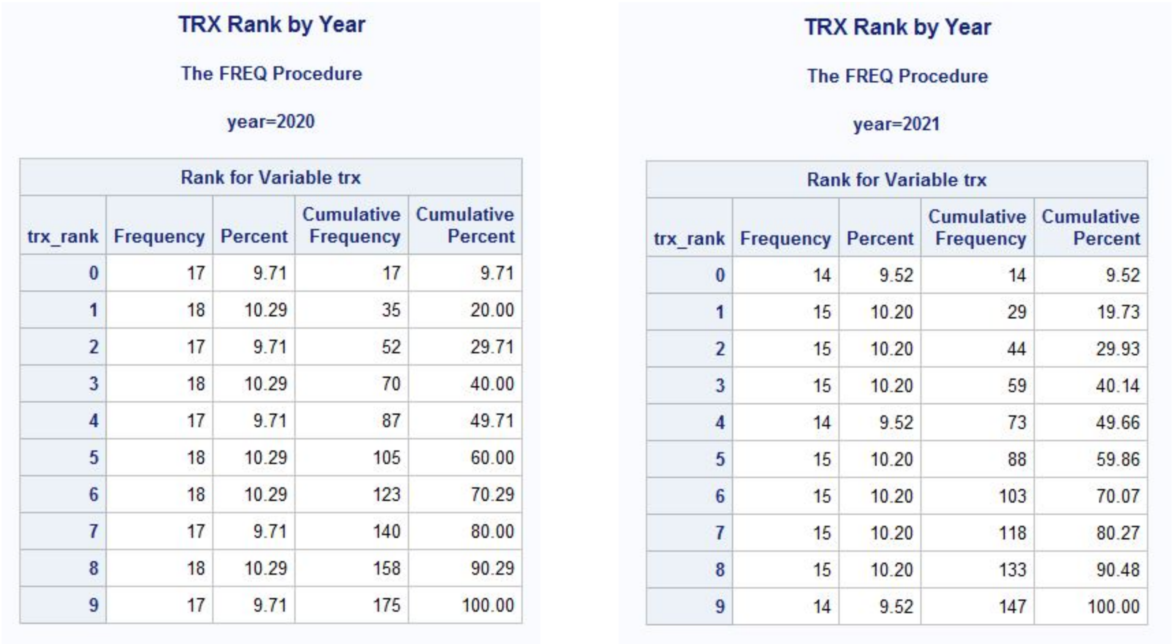
Option GROUPS

Default	none
Groups=100	Percentiles
Groups=10	deciles
Groups=4	quartiles

After we run PROC RANK, the data set will have an additional variable with the rank assigned. We can then run a frequency distribution to understand how the observations are partitioned into deciles:

```
proc freq data = Provider_trx_rank;
      by year;
      tables trx_rank / out=FreqCount outexpect sparse;
      title 'TRX Rank by Year';
run;
```

Figure 4 shows the results of the frequency distribution by year.



TRX Rank by Year

The FREQ Procedure

year=2020

Rank for Variable trx

trx_rank	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	17	9.71	17	9.71
1	18	10.29	35	20.00
2	17	9.71	52	29.71
3	18	10.29	70	40.00
4	17	9.71	87	49.71
5	18	10.29	105	60.00
6	18	10.29	123	70.29
7	17	9.71	140	80.00
8	18	10.29	158	90.29
9	17	9.71	175	100.00

TRX Rank by Year

The FREQ Procedure

year=2021

Rank for Variable trx

trx_rank	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	14	9.52	14	9.52
1	15	10.20	29	19.73
2	15	10.20	44	29.93
3	15	10.20	59	40.14
4	14	9.52	73	49.66
5	15	10.20	88	59.86
6	15	10.20	103	70.07
7	15	10.20	118	80.27
8	15	10.20	133	90.48
9	14	9.52	147	100.00

Figure 4. Frequency of rank by year

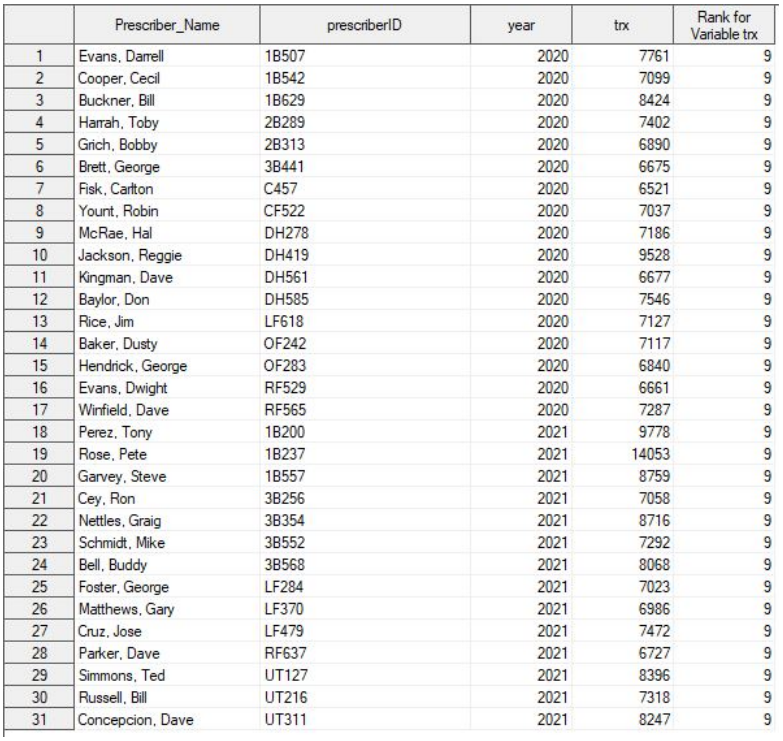
You can see by the frequency distribution that the number of observations in each decile is approximately the same size. The numbers are not exact due to the TIES option. Each decile is close to 10% of the total number of observations.

The second part of the question states to list the providers and total number of prescriptions prescribed for each provider for the year. Now that the ranks are assigned, you can create another dataset with only those prescribers whose rank = 9:

```
/* create data set of prescribers in the top 10 percent rank=9 */
data Prescriber_trx_decile10;
      set Prescriber_trx_rank;
      where trx_rank = 9;
run;

proc print data = Prescriber_trx_decile10;
run;
```

Figure 5 is a screen shot of the Prescriber_trx_decile10 dataset.



	Prescriber_Name	prescriberID	year	trx	Rank for Variable trx
1	Evans, Darrell	1B507	2020	7761	9
2	Cooper, Cecil	1B542	2020	7099	9
3	Buckner, Bill	1B629	2020	8424	9
4	Harrah, Toby	2B289	2020	7402	9
5	Grich, Bobby	2B313	2020	6890	9
6	Brett, George	3B441	2020	6675	9
7	Fisk, Carlton	C457	2020	6521	9
8	Yount, Robin	CF522	2020	7037	9
9	McRae, Hal	DH278	2020	7186	9
10	Jackson, Reggie	DH419	2020	9528	9
11	Kingman, Dave	DH561	2020	6677	9
12	Baylor, Don	DH585	2020	7546	9
13	Rice, Jim	LF618	2020	7127	9
14	Baker, Dusty	OF242	2020	7117	9
15	Hendrick, George	OF283	2020	6840	9
16	Evans, Dwight	RF529	2020	6661	9
17	Winfield, Dave	RF565	2020	7287	9
18	Perez, Tony	1B200	2021	9778	9
19	Rose, Pete	1B237	2021	14053	9
20	Garvey, Steve	1B557	2021	8759	9
21	Cey, Ron	3B256	2021	7058	9
22	Nettles, Graig	3B354	2021	8716	9
23	Schmidt, Mike	3B552	2021	7292	9
24	Bell, Buddy	3B568	2021	8068	9
25	Foster, George	LF284	2021	7023	9
26	Matthews, Gary	LF370	2021	6986	9
27	Cruz, Jose	LF479	2021	7472	9
28	Parker, Dave	RF637	2021	6727	9
29	Simmons, Ted	UT127	2021	8396	9
30	Russell, Bill	UT216	2021	7318	9
31	Concepcion, Dave	UT311	2021	8247	9

Figure 5. Prescriber_trx_decile10 dataset

PROC MEANS

After realizing that the other PM wanted weighted deciles, I tried to figure out how to complete the task with PROC RANK; however, I learned that PROC RANK does not have a weight statement (Bilenas, 2009). Why add weights to percentiles? A weight variable changes the computation of a statistic by giving more weight to some observations than to others (Wicklin, 2016). PROC Univariate, PROC Summary, or PROC MEANS can all be used to create weighted percentiles. I chose PROC MEANS because I am more familiar with it.

The PROC MEANS procedure "provides data summarization tools to compute descriptive statistics for variables across all observations and within groups of observations" (SASSupport.com). PROC MEANS can calculate descriptive statistics (based on moments), estimate quantiles, calculate confidence limits for the mean, identify extreme values, and perform a t test. The default displays output, so if you do not want output, you need to use the option NOPRINT.

When using PROC Means, there will be additional steps as the datasets that are created will drop variables and if you need them, you will need to merge datasets.

The first step is to sort the data, since we will want to use a BY statement to group our data by Year:

```
/* ---------------------------------------------------- */
/*   Sort by Year, PRESCRIBER_ID                        */
/* ---------------------------------------------------- */
proc sort data = PharmSUG.Prescriber_trx_dummy;
      by year prescriberID;
run;
```

Next, let's look at the PROC MEANS code and break it down:

```
/* -------------------------------------------------------------- */
/*  Get weighted percentiles and save output                      */
/* -------------------------------------------------------------- */
proc means data = ❶PharmSUG.prescriber_trx_dummy
                  ❷p10 p20 p30 p40 p50 p60 p70 p80 p90;
    ❸var trx;
      ❹weight trx;
    ❺by year;

    ❻output out = prescriber_trx_percents (❼drop = _freq_ _type_)
    ❽p10 = P_10
      p20 = P_20
      p30 = p_30
      p40 = p_40
      p50 = P_50
      p60 = p_60
      p70 = p_70
      p80 = p_80
      p90 = P_90;

run;
```

❶ Input dataset – input comes from the permanent library named “PharaSUG” and the dataset name is “prescriber_trx_dummy”.

❷ PROC MEANS Options for percentiles – list all percentiles that you want. Note: the Enhanced editor in SAS will turn the standard percentiles blue. Even though some percentiles do not turn blue, you can still use them.

❸ var TRX – TRX is the numeric variable in which the percentiles are based.

❹ weight TRX – TRX is the numeric variable that is weighted.

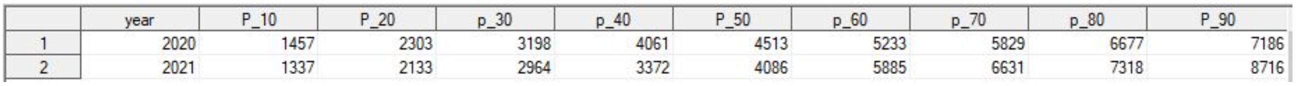
❺ by YEAR – the by statement tells SAS to calculate the percentiles by the variable Year.

❻ output *(keyword)* out = – the output dataset name for the percentiles by year.

❼ drop = – PROC MEANS calculates the frequency and type for each year. If those are not needed, you can drop them by using the drop statement.

❽ Assignment statements – these statements assign a variable name to the percentiles for the output dataset. If you do not list the percentiles and a name, SAS will not write the values to the output dataset.

Figure 6 is a screen shot of the output dataset.



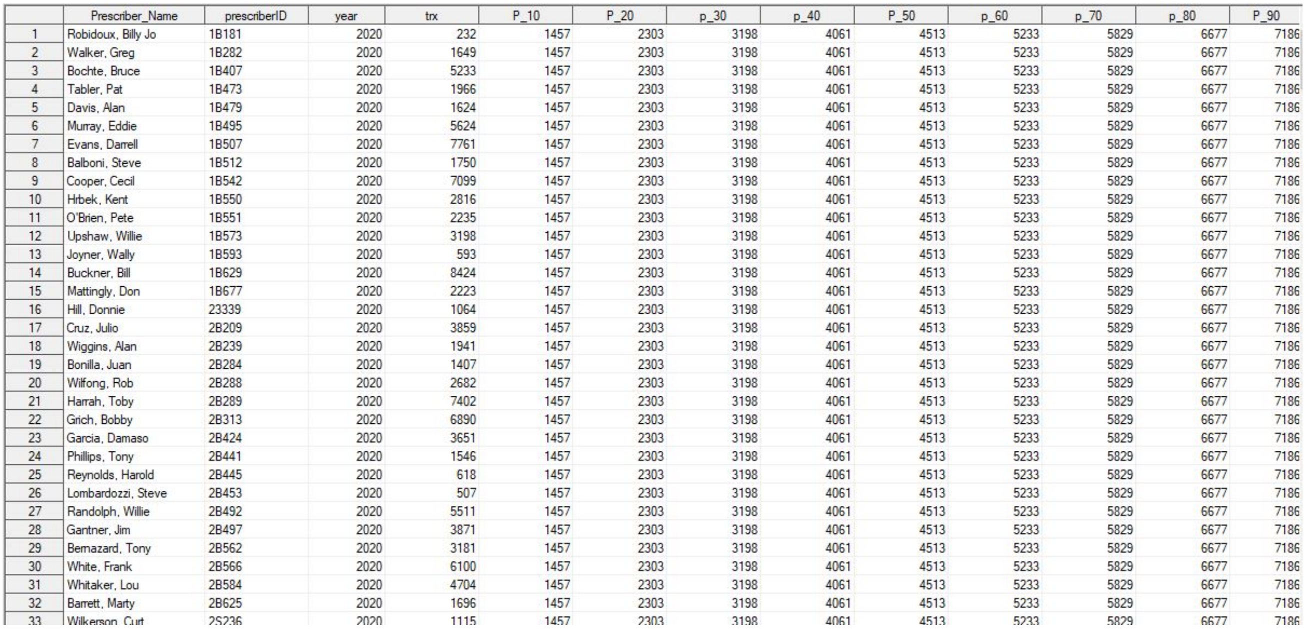
	year	P_10	P_20	p_30	p_40	P_50	p_60	p_70	p_80	P_90
1	2020	1457	2303	3198	4061	4513	5233	5829	6677	7186
2	2021	1337	2133	2964	3372	4086	5885	6631	7318	8716

Figure 6. PROC_MEANS output dataset

To assign the percentile to the observations, you can choose various methods. You can use a lookup, an array, or merge/join the percentiles onto the observation. I chose to join the percentiles onto the observations using PROC SQL, and then use the data step to code the “rank”. Choose the method you are most comfortable with. The following code uses PROC SQL to join the percentiles onto each observation based on the year:

```
/* ------------------------------------------------------- */
/*  Merge back on the provider details                     */
/*  This file will be used for the decile buckets          */
/* ------------------------------------------------------- */
proc sql;
      create table prescriber_TRX_percents2 (drop=_NAME_) as
      select a.*, b.*
      from PharmSUG.prescriber_trx_dummy as a LEFT JOIN prescriber_trx_percents as b
      on a.year = b.year;
      ;
quit;
```

The resulting dataset is the original data, with ALL percentiles for the year added to the observations. Figure 7 is a screen shot of part of the output dataset.



	Prescriber_Name	prescriberID	year	trx	P_10	P_20	p_30	p_40	P_50	p_60	p_70	p_80	P_90
1	Robidoux, Billy Jo	1B181	2020	232	1457	2303	3198	4061	4513	5233	5829	6677	7186
2	Walker, Greg	1B282	2020	1649	1457	2303	3198	4061	4513	5233	5829	6677	7186
3	Bochte, Bruce	1B407	2020	5233	1457	2303	3198	4061	4513	5233	5829	6677	7186
4	Tabler, Pat	1B473	2020	1966	1457	2303	3198	4061	4513	5233	5829	6677	7186
5	Davis, Alan	1B479	2020	1624	1457	2303	3198	4061	4513	5233	5829	6677	7186
6	Murray, Eddie	1B495	2020	5624	1457	2303	3198	4061	4513	5233	5829	6677	7186
7	Evans, Darrell	1B507	2020	7761	1457	2303	3198	4061	4513	5233	5829	6677	7186
8	Balboni, Steve	1B512	2020	1750	1457	2303	3198	4061	4513	5233	5829	6677	7186
9	Cooper, Cecil	1B542	2020	7099	1457	2303	3198	4061	4513	5233	5829	6677	7186
10	Hrbek, Kent	1B550	2020	2816	1457	2303	3198	4061	4513	5233	5829	6677	7186
11	O'Brien, Pete	1B551	2020	2235	1457	2303	3198	4061	4513	5233	5829	6677	7186
12	Upshaw, Willie	1B573	2020	3198	1457	2303	3198	4061	4513	5233	5829	6677	7186
13	Joyner, Wally	1B593	2020	593	1457	2303	3198	4061	4513	5233	5829	6677	7186
14	Buckner, Bill	1B629	2020	8424	1457	2303	3198	4061	4513	5233	5829	6677	7186
15	Mattingly, Don	1B677	2020	2223	1457	2303	3198	4061	4513	5233	5829	6677	7186
16	Hill, Donnie	23339	2020	1064	1457	2303	3198	4061	4513	5233	5829	6677	7186
17	Cruz, Julio	2B209	2020	3859	1457	2303	3198	4061	4513	5233	5829	6677	7186
18	Wiggins, Alan	2B239	2020	1941	1457	2303	3198	4061	4513	5233	5829	6677	7186
19	Bonilla, Juan	2B284	2020	1407	1457	2303	3198	4061	4513	5233	5829	6677	7186
20	Wilfong, Rob	2B288	2020	2682	1457	2303	3198	4061	4513	5233	5829	6677	7186
21	Harrah, Toby	2B289	2020	7402	1457	2303	3198	4061	4513	5233	5829	6677	7186
22	Grich, Bobby	2B313	2020	6890	1457	2303	3198	4061	4513	5233	5829	6677	7186
23	Garcia, Damaso	2B424	2020	3651	1457	2303	3198	4061	4513	5233	5829	6677	7186
24	Phillips, Tony	2B441	2020	1546	1457	2303	3198	4061	4513	5233	5829	6677	7186
25	Reynolds, Harold	2B445	2020	618	1457	2303	3198	4061	4513	5233	5829	6677	7186
26	Lombardozzi, Steve	2B453	2020	507	1457	2303	3198	4061	4513	5233	5829	6677	7186
27	Randolph, Willie	2B492	2020	5511	1457	2303	3198	4061	4513	5233	5829	6677	7186
28	Gantner, Jim	2B497	2020	3871	1457	2303	3198	4061	4513	5233	5829	6677	7186
29	Bernazard, Tony	2B562	2020	3181	1457	2303	3198	4061	4513	5233	5829	6677	7186
30	White, Frank	2B566	2020	6100	1457	2303	3198	4061	4513	5233	5829	6677	7186
31	Whitaker, Lou	2B584	2020	4704	1457	2303	3198	4061	4513	5233	5829	6677	7186
32	Barrett, Marty	2B625	2020	1696	1457	2303	3198	4061	4513	5233	5829	6677	7186
33	Wilkerson, Curt	2S236	2020	1115	1457	2303	3198	4061	4513	5233	5829	6677	7186

Figure 7. PROC SQL output dataset

To illustrate how to assign a "rank" (I called it percent_group since it's not really a rank), I created a new dataset and created a new variable by comparing the TRX value to each percentile range. I used the Keep statement to keep on the variables I wanted:

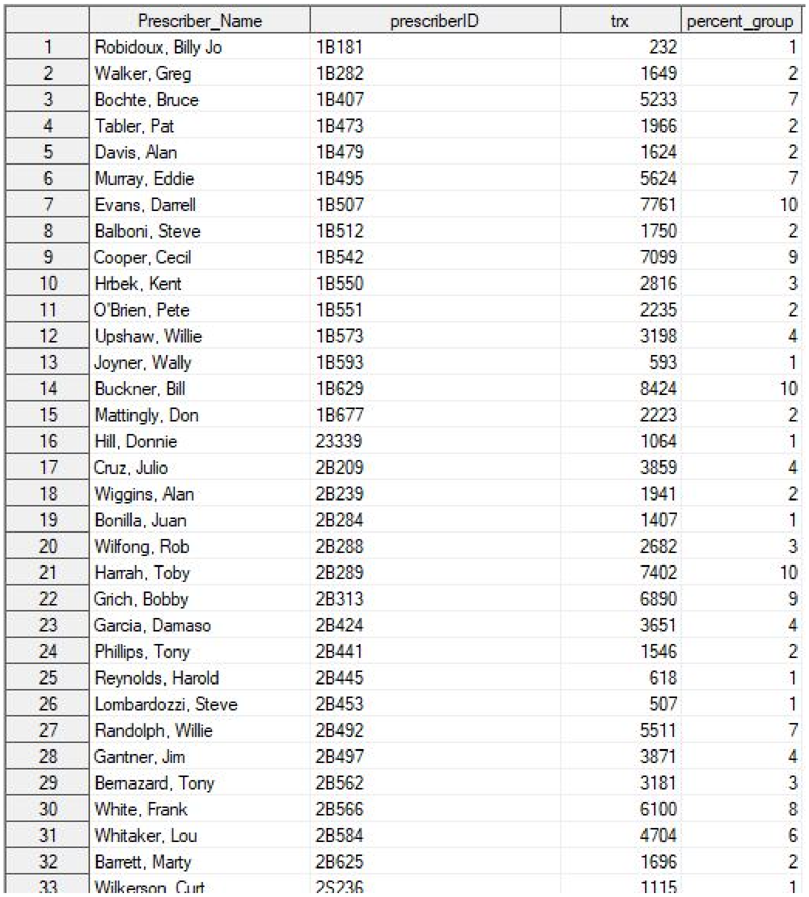
```
/* ---------------------------------------------------------------------- */
/*  calculate which percent group observations belongs in                 */
/* ---------------------------------------------------------------------- */
data prescriber_TRX_percents3;
      set prescriber_TRX_percents2;

      if trx < p_10 then percent_group = 1;
      if trx >=p_10 and trx <p_20 then percent_group = 2;
      if trx >=p_20 and trx <p_30 then percent_group = 3;
      if trx >=p_30 and trx <p_40 then percent_group = 4;
      if trx >=p_40 and trx <p_50 then percent_group = 5;
      if trx >=p_50 and trx <p_60 then percent_group = 6;
      if trx >=p_60 and trx <p_70 then percent_group = 7;
      if trx >=p_70 and trx <p_80 then percent_group = 8;
      if trx >=p_80 and trx <p_90 then percent_group = 9;
      if trx >=p_90 then percent_group = 10;

      Keep prescriber_name prescriberID trx year percent_group;

run;
```

Figure 8 is a screen shot of part of the output dataset with the percent_group assignments.

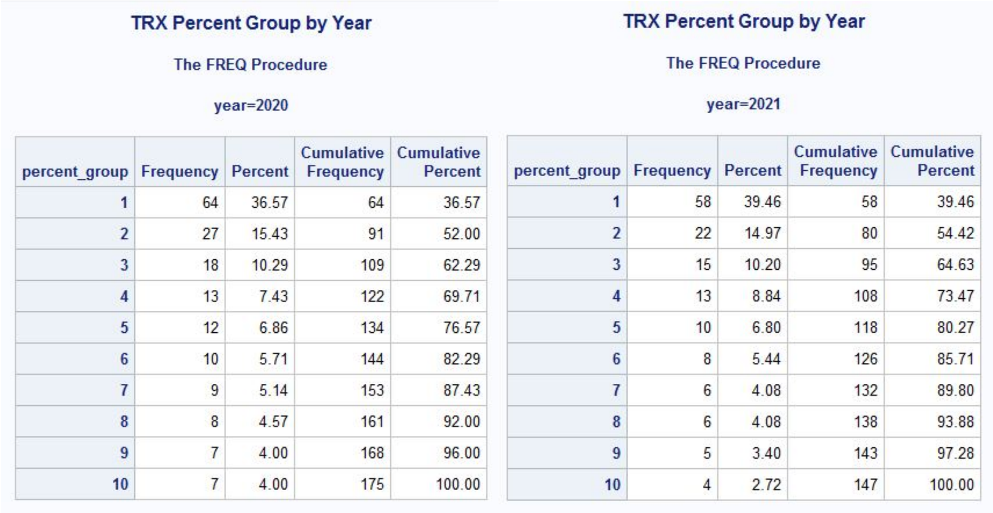


	Prescriber_Name	prescriberID	trx	percent_group
1	Robidoux, Billy Jo	1B181	232	1
2	Walker, Greg	1B282	1649	2
3	Bochte, Bruce	1B407	5233	7
4	Tabler, Pat	1B473	1966	2
5	Davis, Alan	1B479	1624	2
6	Murray, Eddie	1B495	5624	7
7	Evans, Darrell	1B507	7761	10
8	Balboni, Steve	1B512	1750	2
9	Cooper, Cecil	1B542	7099	9
10	Hrbek, Kent	1B550	2816	3
11	O'Brien, Pete	1B551	2235	2
12	Upshaw, Willie	1B573	3198	4
13	Joyner, Wally	1B593	593	1
14	Buckner, Bill	1B629	8424	10
15	Mattingly, Don	1B677	2223	2
16	Hill, Donnie	23339	1064	1
17	Cruz, Julio	2B209	3859	4
18	Wiggins, Alan	2B239	1941	2
19	Bonilla, Juan	2B284	1407	1
20	Wilfong, Rob	2B288	2682	3
21	Harrah, Toby	2B289	7402	10
22	Grich, Bobby	2B313	6890	9
23	Garcia, Damaso	2B424	3651	4
24	Phillips, Tony	2B441	1546	2
25	Reynolds, Harold	2B445	618	1
26	Lombardozzi, Steve	2B453	507	1
27	Randolph, Willie	2B492	5511	7
28	Gantner, Jim	2B497	3871	4
29	Bernazard, Tony	2B562	3181	3
30	White, Frank	2B566	6100	8
31	Whitaker, Lou	2B584	4704	6
32	Barrett, Marty	2B625	1696	2
33	Wilkerson, Curt	2S236	1115	1

Figure 8. percent_group assignment output dataset

I am sure there are better ways (more streamlined ways) to do this; however, I wanted to be able to easily see how the percent group was determined (and non-programmers could easily decipher). During this step, you can drop variables if needed, or write out a separate dataset with only those observations that are in group 10. For the sake of this paper, I wanted to show a Frequency Distribution of the observations in each percent group for weighted variable and how it differs from PROC RANK deciles, so I kept all observations.

Figure 9 is a screen shot of the frequency distribution results for years 2020 and 2021.



TRX Percent Group by Year

The FREQ Procedure

year=2020

percent_group	Frequency	Percent	Cumulative Frequency	Cumulative Percent
1	64	36.57	64	36.57
2	27	15.43	91	52.00
3	18	10.29	109	62.29
4	13	7.43	122	69.71
5	12	6.86	134	76.57
6	10	5.71	144	82.29
7	9	5.14	153	87.43
8	8	4.57	161	92.00
9	7	4.00	168	96.00
10	7	4.00	175	100.00

TRX Percent Group by Year

The FREQ Procedure

year=2021

percent_group	Frequency	Percent	Cumulative Frequency	Cumulative Percent
1	58	39.46	58	39.46
2	22	14.97	80	54.42
3	15	10.20	95	64.63
4	13	8.84	108	73.47
5	10	6.80	118	80.27
6	8	5.44	126	85.71
7	6	4.08	132	89.80
8	6	4.08	138	93.88
9	5	3.40	143	97.28
10	4	2.72	147	100.00

Figure 9. Frequency distribution of weighted percent groups

Since the data are weighted, the groups do not have close to equal number of observations.

CONCLUSION

PROC RANK is a very useful feature to compute the ranks from one or more numeric variables across observations in a dataset. It has many useful options to specify the order of ranks, handling ties, and generating bins or groupings. It does not produce printed output, nor does it allow for weighted percentiles. Utilizing other methods, such as PROC MEANS will allow you to create weighted percentiles. Each method provides an easy way to calculate deciles and percentiles.

REFERENCES

Bilenas, Jonas V. 2009. “Using PROC RANK and PROC UNIVARIATE to Rank or Decile Variables.” *Proceedings of NESUG Conference*.

SAS Blogs. 2016. “The Do Loop.” Accessed April 2022. https://blogs.sas.com/content/iml/2016/08/29/weighted-percentiles.html

ACKNOWLEDGMENTS

I want to thank Lizette Alonzo for the paper review.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Lisa Mendez
sasebmendez@gmail.com