CHAPTER
30
The RANK Procedure

Overview

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.

Output 30.1 on page 833 shows the results of ranking the values of one variable with a simple PROC RANK step. In this example, the new ranking variable shows the order of finish of five golfers over a four-day competition. The player with the lowest number of strokes finishes in first place. The following statements produce the output:

```
proc rank data=golf out=rankings;
   var strokes;
   ranks Finish;
run;

proc print data=rankings;
run;
```

Output 30.1 Assignment of the Lowest Rank Value to the Lowest Variable Value

```
                          The SAS System                          1

                OBS    Player    Strokes    Finish

                 1     Jack         279         2
                 2     Jerry        283         3
                 3     Mike         274         1
                 4     Randy        296         4
                 5     Tito         302         5
```

In Output 30.2 on page 834, the candidates for city council are ranked by district according to the number of votes that they received in the election and according to the number of years that they have served in office.

This example shows how PROC RANK can

- reverse the order of the rankings so that the highest value receives the rank of 1, the next highest value receives the rank of 2, and so on
- rank the observations separately by values of multiple variables
- rank the observations within BY groups
- handle tied values.

For an explanation of the program that produces this report, see Example 2 on page 842.

Output 30.2 Assignment of the Lowest Rank Value to the Highest Variable Value within Each BY Group

```
                        Results of City Council Election                        1
---------------------------------- District=1 ----------------------------------

                                                        Vote    Years
                OBS    Candidate    Vote    Years      Rank     Rank

                 1     Cardella     1689       8         1        1
                 2     Latham       1005       2         3        2
                 3     Smith        1406       0         2        3
                 4     Walker        846       0         4        3

                                N = 4


---------------------------------- District=2 ----------------------------------

                                                        Vote    Years
                OBS    Candidate      Vote    Years      Rank     Rank

                 5     Hinkley         912       0         3        3
                 6     Kreitemeyer    1198       0         2        3
                 7     Lundell        2447       6         1        1
                 8     Thrash          912       2         3        2

                                N = 4
```

Procedure Syntax

Reminder: You can use the ATTRIB, FORMAT, LABEL, and WHERE statements. See Chapter 3, "Statements with the Same Function in Multiple Procedures," for details. You can also use any global statements as well. See Chapter 2, "Fundamental Concepts for Using Base SAS Procedures," for a list.

PROC RANK <*option(s)*>;
BY <DESCENDING> *variable-1*
<…<DESCENDING> *variable-n*>
<NOTSORTED>;
VAR *data-set-variables(s)*;
RANKS *new-variables(s)*;

PROC RANK Statement

PROC RANK <*option(s)*>;

To do this	Use this option
Specify the input data set	DATA=
Create an output data set	OUT=
Specify the ranking method	
Compute fractional ranks	FRACTION or NPLUS1
Partition observations into groups	GROUPS=
Compute normal scores	NORMAL=
Compute percentages	PERCENT
Compute Savage scores	SAVAGE
Reverse the order of the rankings	DESCENDING
Specify how to rank tied values	TIES=

Note: You can specify only one ranking method in a single PROC RANK step. △

Options

DATA=*SAS-data-set*
specifies the input SAS data set.

Main discussion: “Input Data Sets” on page 18

Restriction: You cannot use PROC RANK with an engine that supports concurrent access if another user is updating the data set at the same time.

DESCENDING
reverses the direction of the ranks. With DESCENDING, the largest value receives a rank of 1, the next largest value receives a rank of 2, and so on. Otherwise, values are ranked from smallest to largest.

Featured in: Example 1 on page 840 and Example 2 on page 842

FRACTION
computes fractional ranks by dividing each rank by the number of observations having nonmissing values of the ranking variable.

Alias: F

Interaction: TIES=HIGH is the default with the FRACTION option. With TIES=HIGH, fractional ranks can be considered values of a right-continuous empirical cumulative distribution function.

See also: NPLUS1 option

GROUPS=*number-of-groups*
assigns group values ranging from 0 to *number-of-groups* minus 1. Common specifications are GROUPS=100 for percentiles, GROUPS=10 for deciles, and GROUPS=4 for quartiles. For example, GROUPS=4 partitions the original values into four groups, with the smallest values receiving, by default, a quartile value of 0 and the largest values receiving a quartile value of 3.

The formula for calculating group values is

$$\mathrm{FLOOR}\,(rank * k/(n+1))$$

where FLOOR is the FLOOR function, *rank* is the value's order rank, k is the value of GROUPS=, and n is the number of observations having nonmissing values of the ranking variable.

If the number of observations is evenly divisible by the number of groups, each group has the same number of observations, provided there are no tied values at the boundaries of the groups. Grouping observations by a variable that has many tied values can result in unbalanced groups because PROC RANK always assigns observations with the same value to the same group.

Tip: Use DESCENDING to reverse the order of the group values.

Featured in: Example 3 on page 844

NORMAL=BLOM | TUKEY | VW
computes normal scores from the ranks. The resulting variables appear normally distributed. The formulas are

BLOM	$y_i=\Phi^{-1}(r_i-3/8)/(n+1/4)$
TUKEY	$y_i=\Phi^{-1}(r_i-1/3)/(n+1/3)$
VW	$y_i=\Phi^{-1}(r_i)/(n+1)$

where Φ^{-1} is the inverse cumulative normal (PROBIT) function, r_i is the rank of the ith observation, and n is the number of nonmissing observations for the ranking variable.

VW stands for van der Waerden. With NORMAL=VW, you can use the scores for a nonparametric location test. All three normal scores are approximations to the exact expected order statistics for the normal distribution, also called *normal scores*. The BLOM version appears to fit slightly better than the others (Blom 1958; Tukey 1962).

NPLUS1
computes fractional ranks by dividing each rank by the denominator $n+1$, where n is the number of observations having nonmissing values of the ranking variable.

Aliases: FN1, N1

Interaction: TIES=HIGH is the default with the NPLUS1 option.

See also: FRACTION option

OUT=*SAS-data-set*
names the output data set. If *SAS-data-set* does not exist, PROC RANK creates it. If you omit OUT=, the data set is named using the DATA*n* naming convention.

PERCENT
divides each rank by the number of observations having nonmissing values of the variable and multiplies the result by 100 to get a percentage.

Alias: P

Interaction: TIES=HIGH is the default with the PERCENT option.

Tip: You can use PERCENT to calculate cumulative percentages, but use GROUPS=100 to compute percentiles.

SAVAGE
computes Savage (or exponential) scores from the ranks by the following formula (Lehman 1975):

$$y_i = \left[\sum_{j=n-r_i+1} \left(\frac{1}{j} \right) \right] - 1$$

TIES=HIGH | LOW | MEAN
specifies the rank for tied values.

HIGH
assigns the largest of the corresponding ranks.

LOW
assigns the smallest of the corresponding ranks.

MEAN
assigns the mean of the corresponding ranks.

Default: MEAN (unless the FRACTION or PERCENT option is in effect)

Featured in: Example 1 on page 840 and Example 2 on page 842

BY Statement

Produces a separate set of ranks for each BY group.

Main discussion: "BY" on page 68

Featured in: Example 2 on page 842 and Example 3 on page 844

BY <DESCENDING> *variable-1*
<…<DESCENDING> *variable-n*>
<NOTSORTED>;

Required Arguments

variable
specifies the variable that the procedure uses to form BY groups. You can specify more than one variable. If you do not use the NOTSORTED option in the BY statement, the observations in the data set must either be sorted by all the variables that you specify, or they must be indexed appropriately. Variables in a BY statement are called *BY variables*.

Options

DESCENDING
specifies that the observations are sorted in descending order by the variable that immediately follows the word DESCENDING in the BY statement.

NOTSORTED
specifies that observations are not necessarily sorted in alphabetic or numeric order. The observations are grouped in another way, for example, chronological order.

The requirement for ordering or indexing observations according to the values of BY variables is suspended for BY-group processing when you use the NOTSORTED option. In fact, the procedure does not use an index if you specify NOTSORTED. The procedure defines a BY group as a set of contiguous observations that have the same values for all BY variables. If observations with the same values for the BY variables are not contiguous, the procedure treats each contiguous set as a separate BY group.

RANKS Statement

Creates new variables for the rank values.

Requirement: If you use the RANKS statement, you must also use the VAR statement.

Default: If you omit the RANKS statement, the rank values replace the original variable values in the output data set.

Featured in: Example 1 on page 840 and Example 2 on page 842

RANKS *new-variables(s)*;

Required Arguments

new-variable(s)
specifies one or more new variables that contain the ranks for the variable(s) listed in the VAR statement. The first variable listed in the RANKS statement contains the ranks for the first variable listed in the VAR statement, the second variable listed in the RANKS statement contains the ranks for the second variable listed in the VAR statement, and so forth.

VAR Statement

Specifies the input variables.

Default: If you omit the VAR statement, PROC RANK computes ranks for all numeric variables in the input data set.

Featured in: Example 1 on page 840, Example 2 on page 842, and Example 3 on page 844

VAR *data-set-variables(s)*;

Required Arguments

data-set-variable(s)
specifies one or more variables for which ranks are computed.

Using the VAR Statement with the RANKS Statement

The VAR statement is required when you use the RANKS statement. Using these statements together creates the ranking variables named in the RANKS statement that correspond to the input variables specified in the VAR statement. If you omit the RANKS statement, the rank values replace the original values in the output data set.

Concepts

Computer Resources

PROC RANK stores all values in memory of the variables for which it computes ranks.

Statistical Applications

Ranks are useful for investigating the distribution of values for a variable. The ranks divided by n or $n+1$ form values in the range 0 to 1, and these values estimate the cumulative distribution function. You can apply inverse cumulative distribution functions to these fractional ranks to obtain probability quantile scores, which you can compare to the original values to judge the fit to the distribution. For example, if a set of data has a normal distribution, the normal scores should be a linear function of the original values, and a plot of scores versus original values should be a straight line.

Many nonparametric methods are based on analyzing ranks of a variable:

- A two-sample t–test applied to the ranks is equivalent to a Wilcoxon rank sum test using the t approximation for the significance level. If you apply the t–test to the normal scores rather than to the ranks, the test is equivalent to the van der Waerden test. If you apply the t–test to median scores (GROUPS=2), the test is equivalent to the median test.

- A one-way analysis of variance applied to ranks is equivalent to the Kruskal-Wallis k-sample test; the F–test generated by the parametric procedure applied to the ranks is often better than the X^2 approximation used by Kruskal-Wallis. This test can be extended to other rank scores (Quade 1966).
- You can obtain a Friedman's two-way analysis for block designs by ranking within BY groups and then performing a main-effects analysis of variance on these ranks (Conover 1980).
- You can investigate regression relationships by using rank transformations with a method described by Iman and Conover (1979).

Results

Missing Values

Missing values are not ranked and are left missing when ranks or rank scores replace the original values in the output data set.

Output Data Set

The RANK procedure creates a SAS data set containing the ranks or rank scores but does not create any printed output. You can use PROC PRINT, PROC REPORT, or another SAS reporting tool to print the output data set.

The output data set contains all the variables from the input data set plus the variables named in the RANKS statement. If you omit the RANKS statement, the rank values replace the original variable values in the output data set.

Examples

Example 1: Ranking Values of Multiple Variables

Procedure features:

PROC RANK statement options:

DESCENDING
TIES=

RANKS statement

VAR statement

Other features:

PRINT procedure

This example

- reverses the order of the ranks so that the highest value receives the rank of 1
- assigns tied values the best possible rank
- creates ranking variables and prints them with the original variables.

Program

```
options nodate pageno=1 linesize=80 pagesize=60;
```

The data set CAKE contains each participant's last name, score for presentation, and score for taste in a cake-baking contest.

```
data cake;
   input Name $ 1-10 Present 12-13 Taste 15-16;
   datalines;
Davis      77 84
Orlando    93 80
Ramey      68 72
Roe        68 75
Sanders    56 79
Simms      68 77
Strickland 82 79
;
```

DESCENDING reverses the order of the ranks so that the high score receives the rank of 1. TIES=LOW gives tied values the best possible rank. OUT= creates the output data set ORDER.

```
proc rank data=cake out=order descending ties=low;
```

The new variables PresentRank and TasteRank contain the respective ranks for the variables Present and Taste.

```
   var present taste;
   ranks PresentRank TasteRank;
run;
```

PROC PRINT prints the ORDER data set.

```
proc print data=order;
   title "Rankings of Participants' Scores";
run;
```

Output

Rankings of Participants' Scores 1

OBS	Name	Present	Taste	Present Rank	Taste Rank
1	Davis	77	84	3	1
2	Orlando	93	80	1	2
3	Ramey	68	72	4	7
4	Roe	68	75	4	6
5	Sanders	56	79	7	3
6	Simms	68	77	4	5
7	Strickland	82	79	2	3

Example 2: Ranking Values within BY Groups

Procedure features:

PROC RANK statement options:

DESCENDING
TIES=

BY statement

RANKS statement

VAR statement

Other features:

PRINT procedure

This example

- ranks observations separately within BY groups
- reverses the order of the ranks so that the highest value receives the rank of 1
- assigns tied values the best possible rank
- creates ranking variables and prints them with the original variables.

Program

```
options nodate pageno=1 linesize=80 pagesize=60;
```

The data set ELECT contains each candidate's last name, district number, vote total, and number of years experience on the city council.

```
data elect;
   input Candidate $ 1-11 District 13 Vote 15-18 Years 20;
   datalines;
Cardella    1 1689 8
Latham      1 1005 2
Smith       1 1406 0
```

```
Walker      1  846 0
Hinkley     2  912 0
Kreitemeyer 2 1198 0
Lundell     2 2447 6
Thrash      2  912 2
;
```

DESCENDING reverses the order of the ranks so that the highest vote total receives the rank of 1. TIES=LOW gives tied values the best possible rank. OUT= creates the output data set RESULTS.

```
proc rank data=elect out=results ties=low descending;
```

The BY statement separates the rankings by values of District.

```
   by district;
```

The new variables VoteRank and YearsRank contain the respective ranks for the variables Vote and Years.

```
   var vote years;
   ranks VoteRank YearsRank;
run;
```

PROC PRINT prints the RESULTS data set. The N option prints the number of observations in each BY group.

```
proc print data=results n;
   by district;
   title 'Results of City Council Election';
run;
```

Output

In the second district, Hinkley and Thrash tied with 912 votes. They both receive a rank of 3 because TIES=LOW.

Results of City Council Election 1

District=1

OBS	Candidate	Vote	Years	Vote Rank	Years Rank
1	Cardella	1689	8	1	1
2	Latham	1005	2	3	2
3	Smith	1406	0	2	3
4	Walker	846	0	4	3

N = 4

District=2

OBS	Candidate	Vote	Years	Vote Rank	Years Rank
5	Hinkley	912	0	3	3
6	Kreitemeyer	1198	0	2	3
7	Lundell	2447	6	1	1
8	Thrash	912	2	3	2

N = 4

Example 3: Partitioning Observations into Groups Based on Ranks

Procedure features:

PROC RANK statement option:

GROUPS=

BY statement

VAR statement

Other features:

PRINT procedure

SORT procedure

This example

- partitions observations into groups on the basis of values of two input variables
- groups observations separately within BY groups
- replaces the original variable values with the group values.

Program

```
options nodate pageno=1 linesize=80 pagesize=60;
```

The data set SWIM contains swimmers' first names and their times, in seconds, for the backstroke and the freestyle. This example groups the swimmers into pairs, within male and female classes, based on times in both strokes so that every swimmer is paired with someone who has a similar time for each stroke.

```
data swim;
   input Name $ 1-7 Gender $ 9 Back 11-14 Free 16-19;
   datalines;
Andrea  F 28.6 30.3
Carole  F 32.9 24.0
Clayton M 27.0 21.9
Curtis  M 29.0 22.6
Doug    M 27.3 22.4
Ellen   F 27.8 27.0
Jan     F 31.3 31.2
Jimmy   M 26.3 22.5
Karin   F 34.6 26.2
Mick    M 29.0 25.4
Richard M 29.7 30.2
Sam     M 27.2 24.1
Susan   F 35.1 36.1
;
```

PROC SORT sorts the data set by Gender. This is required to obtain a separate set of ranks for each group.

```
proc sort data=swim out=pairs;
   by gender;
run;
```

GROUPS=3 assigns one of three possible group values (0,1,2) to each swimmer for each stroke.

```
proc rank data=pairs out=rankpair groups=3;
```

The BY statement ranks the times separately by Gender.

```
   by gender;
```

The VAR statement specifies the input variables. With no RANKS statement, PROC RANK replaces the original variable values with the group values in the output data set.

```
   var back free;
run;
```

PROC PRINT prints the RANKPAIR data set. The N option prints the number of observations in each BY group.

```
proc print data=rankpair n;
   by gender;
```

```
   title 'Pairings of Swimmers for Backstroke and Freestyle';
run;
```

Output

The group values pair up swimmers with similar times to work on each stroke. For example, Andrea and Ellen work together on the backstroke because they have the fastest times in the female class. The groups of male swimmers are unbalanced because there are seven male swimmers; for each stroke, one group has three swimmers.

```
              Pairings of Swimmers for Backstroke and Freestyle              1

---------------------------------- Gender=F ----------------------------------

                          OBS    Name      Back    Free

                            1    Andrea       0       1
                            2    Carole       1       0
                            3    Ellen        0       1
                            4    Jan          1       2
                            5    Karin        2       0
                            6    Susan        2       2

                                      N = 6


---------------------------------- Gender=M ----------------------------------

                          OBS    Name       Back    Free

                            7    Clayton      0       0
                            8    Curtis       2       1
                            9    Doug         1       0
                           10    Jimmy        0       1
                           11    Mick         2       2
                           12    Richard      2       2
                           13    Sam          1       1

                                      N = 7
```

References

Blom, G. (1958), *Statistical Estimates and Transformed Beta Variables*, New York: John Wiley & Sons, Inc.

Conover, W.J. (1980), *Practical Nonparametric Statistics*, Second Edition, New York: John Wiley & Sons, Inc.

Conover, W.J. and Iman, R.L. (1976), "On Some Alternative Procedures Using Ranks for the Analysis of Experimental Designs," *Communications in Statistics*, A5, 14, 1348–1368.

Conover, W.J. and Iman, R.L. (1981), "Rank Transformations as a Bridge between Parametric and Nonparametric Statistics," *The American Statistician*, 35, 124–129.

Iman, R.L. and Conover, W.J. (1979), "The Use of the Rank Transform in Regression," *Technometrics*, 21, 499–509.

Lehman, E.L. (1975), *Nonparametrics: Statistical Methods Based on Ranks*, San Francisco: Holden-Day.

Quade, D. (1966), "On Analysis of Variance for the *k*-Sample Problem," *Annals of Mathematical Statistics*, 37, 1747–1758.

Tukey, John W. (1962), "The Future of Data Analysis," *Annals of Mathematical Statistics*, 33, 22.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS® Procedures Guide, Version 8,* Cary, NC: SAS Institute Inc., 1999. 1729 pp.

SAS® Procedures Guide, Version 8

Copyright © 1999 by SAS Institute Inc., Cary, NC, USA.

ISBN 1–58025–482–9

All rights reserved. Printed in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

U.S. Government Restricted Rights Notice. Use, duplication, or disclosure of the software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227–19 Commercial Computer Software-Restricted Rights (June 1987).

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, October 1999

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries.® indicates USA registration.

IBM® and DB2® are registered trademarks or trademarks of International Business Machines Corporation. ORACLE® is a registered trademark of Oracle Corporation. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

The Institute is a private company devoted to the support and further development of its software and related services.