Modeling rate curves with SAS

Applications of Proc Expand

Matteo M. Costa
November 11, 2024

Abstract

These pages present an application of Proc Expand, a SAS procedure used for points interpolation. The main purpose is to demonstrate a recent use of Proc Expand in a risk modeling context. This procedure simplifies mathematical operations that would otherwise be computationally expensive if coded from scratch.

1 Introduction

A banking perimeter dataset contains cashflow information for each customer at fixed time points. We can view the perimeter dataset P as corresponding to a real matrix with n rows and m columns. For example, if $c_j \in C = \{c_1, \dots, c_p\}$ is a generic customer in the finite set of customers, then for each j we have an associated submatrix of real numbers I_j of P such that $I_j = (t^j | r^j) \subset \mathbb{R}^{s_j} \times \mathbb{R}^2$ where:

- $s_j \in \mathbb{N}$ is the number of records (i.e. rows in P) for c_j;
- $t^j \in \mathbb{R}^{s_j}$ is the column vector of times, where each cashflow is defined at a specific time. In particular, $t_i^j \in \mathbb{R}$ is the i-th time point for the j-th customer;
- $r^j \in \mathbb{R}^{s_j}$ is the column vector of rate values, where $r_i^j \in \mathbb{R}$ is the rate value at time t_i^j for c_j customer.

For example, the first customer c_1 might have three cashflow times defined at the beginning of each month, so we have:

$$I_1 = \begin{pmatrix} t_1^1 & 0.60 \\ t_2^1 & 0.45 \\ t_3^1 & 0.20 \end{pmatrix}$$

The goal is to calculate risk indicators that use rates values r_i^j as inputs. One challenge is presented by the perimeter structure: in general, the perimeter is not complete, meaning that for a fixed t_i^j the corresponding r_i^j might be missing. In such cases, it is necessary to fill in this value. In SAS, this can be done by joining with a second dataset containing rate curves, previously calculated based on the bank's historical data.

Another potential issue is that r_i^j could be missing from the rate curve dataset itself. In this situation, different approaches can be used to determine the missing value; in our application, we opted to apply a linear polynomial interpolation. The following plot provides a clear geometric visualization of the problem.

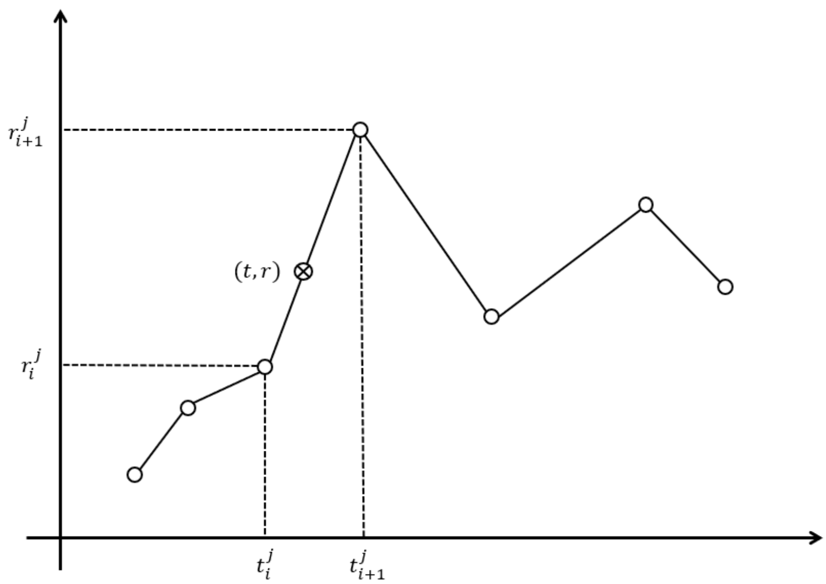

Figure 1 Example of a possible rate curve for a fixed customer c_j. It may occur that r is missing for certain values of t.

How do we perform linear interpolation? For a fixed j, if t_i^j and t_{i+1}^j are consecutive values, we can apply the classical formula

$$r = \frac{r_{i+1}^j - r_i^j}{t_{i+1}^j - t_i^j}\left(t - t_i^j\right) + r_i^j,$$

where:

- $\{t_i^j\}_i$ represents the curve times;
- $\{r_i^j\}_i$ represents the curve rates;
- t is the point at which we want to interpolate, under the condition $t_i^j < t < t_{i+1}^j$;
- r is the resulting interpolated value.

This equation represents the classical formula for the line that connects two distinct fixed points (x_a, y_a) and (x_b, y_b) in the Euclidean plane $\mathbb{R}^2$:

$$\frac{y - y_a}{y_b - y_a} = \frac{x - x_a}{x_b - x_a} \Rightarrow y = \frac{y_b - y_a}{x_b - x_a}(x - x_a) + y_a.$$

Specifically, the real number

$$m = \frac{y_b - y_a}{x_b - x_a} = \frac{r_{i+1}^j - r_i^j}{t_{i+1}^j - t_i^j},$$

is known as the slope of the line. By knowing the slope and one point on the line, we can determine the line's direction and compute any other points that lie along it. Using this formula, we can calculate any missing y-value for a given t but in SAS this process can be simplified by using Proc Expand. In next following sections, we will explore Proc Expand with some examples.

2 Proc Expand

Let's start understanding Proc Expand with a simple example. We begin by defining a dataset *test* which contains two columns, x and y. While x is complete, some values are missing in y. The idea is to populate these missing values using linear interpolation.

Before applying Proc Expand, it is necessary to sort the dataset by the x column. If we don't sort the dataset, an error message will appear in the SAS log. In Proc Expand procedure, the option *method* allows to specify the interpolation method; if we set *method = join*, linear interpolation is performed. Here is an example:

```
data test;
   input x y;
   datalines;
   0.0 1
   5.0 .
   0.7 .
   1.0 3
   1.3 .
   4.3 6
   2.0 4
   2.6 .
   4.6 .
   6.1 3
;
run;

proc sort data=test out=test_output;
   by x;
run;

proc expand data=test_output
   out=Interp(rename=(Linear=Interp));
   convert y=linear / method=join;
   id x;
run;
```

In the ouput, a new column will appear, containing the interpolated values. By default, Proc Expand names this column as *Linear* and so we have renamed it to *Interp*. We have specified the variable and the interpolation method. This is a basic syntax that can be expanded by considering additional options. We recommend consulting the official SAS guide for a deeper understanding.

Oss	x	y	Interp
1	0.0	1	1.00000
2	0.7	.	2.40000
3	1.0	3	3.00000
4	1.3	.	3.30000
5	2.0	4	4.00000
6	2.6	.	4.52174
7	4.3	6	6.00000
8	4.6	.	5.50000
9	5.0	.	4.83333
10	6.1	3	3.00000

Figure 2 In the final output, the *Interp* column contains the linear interpolated values.

In Figure 2, for example, let's consider the second observation where y is missing. By default, SAS adopts the standard format for numeric variables: in the following rows, we use an approximation to two decimal places. To find the rate value, we adopt the formula

$$r = \frac{r_{i+1}^{j} - r_{i}^{j}}{t_{i+1}^{j} - t_{i}^{j}}\left(t - t_{i}^{j}\right) + r_{i}^{j},$$

with $(t_i^j, r_i^j) = (0, 1)$ and $(t_{i+1}^j, r_{i+1}^j) = (1, 3)$. We obtain

$$y = \frac{3-1}{1-0}(0.7 - 0) + 1 = 2.4.$$

Moreover, if we have two consecutive missing rate values, the *join* method considers the last found interpolated value as the starting point. For example, in Figure 2, the interpolated value 4.83 in line nine was determined using the previously calculated interp value 5.50 at line eight. Specifically, considering $(t_i^j, r_i^j) = (4.30, 6)$ and $(t_{i+1}^j, r_{i+1}^j) = (6.10, 3)$, we start calculating line eight as follows:

$$\frac{3-6}{6.10-4.30}(4.60 - 4.30) + 6 = 5.50,$$

and then we use this new last term $(t_i^j, r_i^j) = (4.60, 5.50)$ and $(t_{i+1}^j, r_{i+1}^j) = (6.10, 3)$ to determine line nine:

$$\frac{3 - 5.50}{6.10 - 4.60}(5 - 4.60) + 5.50 = 4.83.$$

3 Rate curve modeling

In this section, we apply the procedure to a real case. We summarize the resolution of the problem in the following steps:

- left join between the perimeter and rate curve datasets: the purpose is to add rate values into the perimeter;
- sorting data: Proc Expand may generate an error message in the log if the variable to be interpolated is not sorted;
- Proc Expand: application of the method.

For privacy reasons, we use fake datasets in this application to simulate a real situation. We generate these datasets using data steps. In the next section, we will begin to construct the perimeter by considering some customers with different cash flows and rate curves.

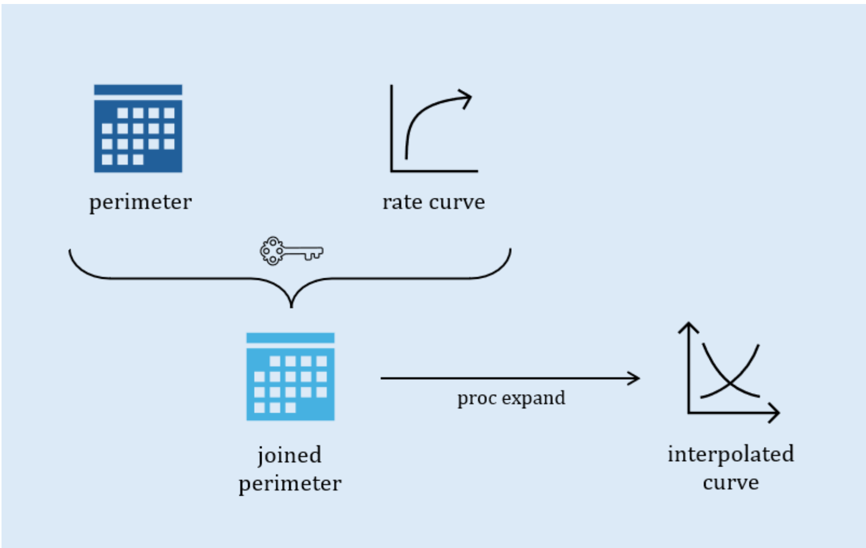

Figure 3 A diagram that summarizes the dataset creation, the sorting operation, and the final application of Proc Expand.

3.1 The perimeter

The fake perimeter has been constructed considering five variables:

- **ID**: in real cases, a bank customer is typically represented by an alphanumeric code. In this dataset, we consider three customers, so $C = \{c_1, c_2, c_3\}$;
- **progressivo_cf**: the cashflow of each customer is described over time, date by date. According to Section 1, for each c_j we can identify the respective vector t^j. For example, the first customer $c_1 =$ 0000CKDO has $t^1 = (t_1^1, t_2^1, t_3^1, t_4^1, t_5^1) = (1, 2, 3, 4, 5)$;
- **cashflow**: the flow of money at fixed time for each customer;
- **curve_id**: this variable identifies the rate curve to be extracted from the rate curve dataset. Additionally, the bank models different curves based on various economic scenarios, resulting in different types of curves;
- **date**: a real date value is associated with each t_i^j.

Here is the code:

```
data perimeter;
   infile datalines delimiter = ',';
   input ID $ progressivo_cf cashflow curve_id $ date $;
   datalines;
   0000CKDO,1,1920.25,curve3M,01/03/2024
   0000CKDO,2,1820.10,curve3M,01/04/2024
   0000CKDO,3,1929.00,curve3M,01/05/2024
   0000CKDO,4,1755.00,curve3M,01/06/2024
   0000CKDO,5,1921.78,curve3M,01/07/2024
   0000ALFO,1,1919.59,curve3M,01/06/2024
   0000ALFO,2,1899.60,curve3M,01/07/2024
   0000ALFO,3,1920.25,curve3M,01/08/2024
   0000ASJJ,1,1300.10,curveM,01/03/2024
   0000ASJJ,2,1330.00,curveM,01/04/2024
   0000ASJJ,3,1250.00,curveM,01/05/2024
   0000ASJJ,4,1320.65,curveM,01/06/2024
   0000ASJJ,5,1220.75,curveM,01/07/2024
   0000ASJJ,6,1120.85,curveM,01/08/2024
;
run;
```

We have a dataset with three customers: the first two share the same rate curve, while the third has a different one. In a real-world scenario, the number of cash flows may vary for each customer.

Oss	ID	progressivo_cf	cashflow	curve_id	date
1	0000CKDO	1	1920.25	curve3M	01/03/20
2	0000CKDO	2	1820.10	curve3M	01/04/20
3	0000CKDO	3	1929.00	curve3M	01/05/20
4	0000CKDO	4	1755.00	curve3M	01/06/20
5	0000CKDO	5	1921.78	curve3M	01/07/20
6	0000ALFO	1	1919.59	curve3M	01/06/20
7	0000ALFO	2	1899.60	curve3M	01/07/20
8	0000ALFO	3	1920.25	curve3M	01/08/20
9	0000ASJJ	1	1300.10	curveM	01/03/20
10	0000ASJJ	2	1330.00	curveM	01/04/20
11	0000ASJJ	3	1250.00	curveM	01/05/20
12	0000ASJJ	4	1320.65	curveM	01/06/20
13	0000ASJJ	5	1220.75	curveM	01/07/20
14	0000ASJJ	6	1120.85	curveM	01/08/20

Figure 4 The perimeter: for each customer we have a cashflow described at the beginning of each month. We also have the variable *curve_id*, which serves as one of the join keys.

3.2 The rate curve dataset

In general, each type of curve models an economic scenario, and the values are updated annually by the bank. Additionally, the term "curve" should not imply a smooth function. Creating a plot may help in understanding the trend. For example, a useful way to visualize a curve is as a linear piecewise function:

$$r(t) = \sum_{i=1}^{N} (m_i t + q_i) \mathbb{I}_{[t_{i-1}, t_i)}(t),$$

where $m_i t + q_i$ represents the i-th linear component defined on the time subset $[t_{i-1}, t_i)$ and $\mathbb{I}_{[t_{i-1}, t_i)}(t)$ is the corresponding indicator function for this interval:

$$\mathbb{I}_{[t_{i-1}, t_i)}(t) = \begin{cases} 1 & t \in [t_{i-1}, t_i) \\ 0 & t \notin [t_{i-1}, t_i) \end{cases}$$

In our example, the rate curves dataset contains the following variables:

- **key_curve**: each curve is identified by a unique code;
- **date**: for each curve, the rate value is specified at particular time. In our example, we have two curves defined within a year, called *curveM* and *curve3M*;
- **rate**: the rate value at the specified date.

We have created the dataset as follows:

```
data rate_curves;
   infile datalines delimiter=',';
   input key_curve $ date $ rate;
   datalines;
   curveM,01/01/2024,0.20
   curveM,01/02/2024,
   curveM,01/03/2024,0.32
   curveM,01/04/2024,0.42
   curveM,01/05/2024,
   curveM,01/06/2024,0.56
   curveM,01/07/2024,
   curveM,01/08/2024,0.20
   curveM,01/09/2024,
   curveM,01/10/2024,
   curveM,01/11/2024,0.80
   curveM,01/12/2024,0.90
   curve3M,01/01/2024,0.12
   curve3M,01/02/2024,0.56
   curve3M,01/03/2024,0.30
   curve3M,01/04/2024,
   curve3M,01/05/2024,
   curve3M,01/06/2024,0.49
   curve3M,01/07/2024,0.30
   curve3M,01/08/2024,0.95
   curve3M,01/09/2024,0.32
   curve3M,01/10/2024,0.15
   curve3M,01/11/2024,0.15
   curve3M,01/12/2024,0.20
;
run;
```

We have some missing values for different curves. In this case we have a time distribution over a year; in general, we could have longer periods. In the next section, we will use the Proc Expand and the final procedures.

3.3 Join and Proc Expand

We begin by joining the datasets using a left join on the keys. In general, these keys are linked to the customer and the time. In our example, we consider two keys: *date* and *key_curve*. After the join, we sort the results by *ID* and *progressivo_cf* and then we apply Proc Expand:

```
proc sql;
   create table joined as
   select a.*, b.rate
   from perimeter as a
   left join
   rate_curves as b
   on a.curve_id=b.key_curve and
      a.date=b.date
   order by ID, progressivo_cf;
quit;

proc sort data=joined;
   by ID progressivo_cf;
quit;


 proc expand data=joined
              out=interp(rename=(Linear=interp));
       convert rate=linear / method=join;
       by ID;
       id progressivo_cf;
run;
```

There we can observe the joined dataset and the final output.

Oss	ID	progressivo_cf	cashflow	curve_id	date	rate	interp
1	0000ALFO	1	1919.59	curve3M	01/06/20	0.49	0.49000
2	0000ALFO	2	1899.60	curve3M	01/07/20	0.30	0.30000
3	0000ALFO	3	1920.25	curve3M	01/08/20	0.95	0.95000
4	0000ASJJ	1	1300.10	curveM	01/03/20	0.32	0.32000
5	0000ASJJ	2	1330.00	curveM	01/04/20	0.42	0.42000
6	0000ASJJ	3	1250.00	curveM	01/05/20	.	0.49000
7	0000ASJJ	4	1320.65	curveM	01/06/20	0.56	0.56000
8	0000ASJJ	5	1220.75	curveM	01/07/20	.	0.38000
9	0000ASJJ	6	1120.85	curveM	01/08/20	0.20	0.20000
10	0000CKDO	1	1920.25	curve3M	01/03/20	0.30	0.30000
11	0000CKDO	2	1820.10	curve3M	01/04/20	.	0.36333
12	0000CKDO	3	1929.00	curve3M	01/05/20	.	0.42667
13	0000CKDO	4	1755.00	curve3M	01/06/20	0.49	0.49000
14	0000CKDO	5	1921.78	curve3M	01/07/20	0.30	0.30000

Figure 5 Using the keys *date* and *curve_id*, we add the rate values associated with each cashflow (column *rate*). Then we obtain the final dataset with interpolated values (column *interp*).

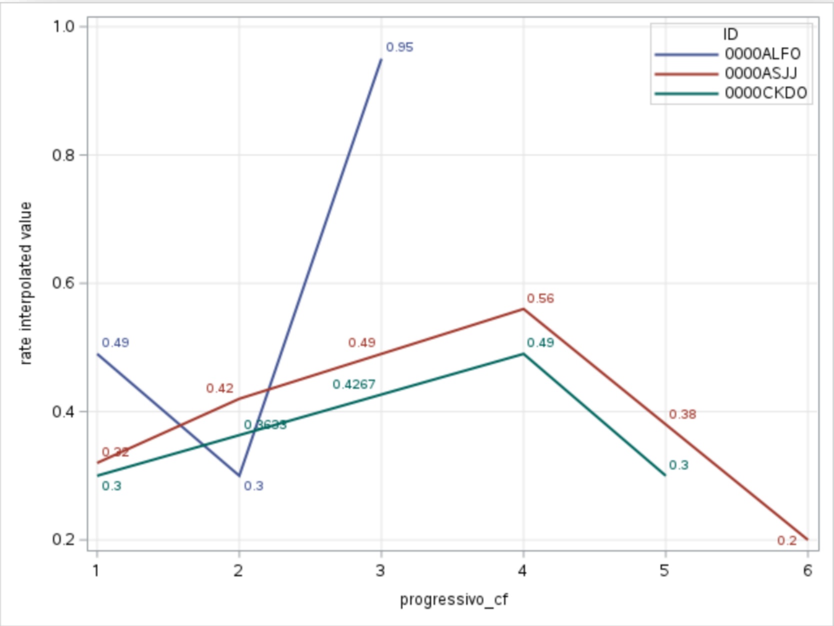

Figure 6 Using Proc Sgplot, we can plot the three rate curves for each customer, allowing us to observe different changes in slope for each curve. By using the *group* option, we can display all three curves simultaneously. To adjust the legend, we can use the *position* and *across* options.

4 Conclusion

Proc Expand is an extremely useful tool within the SAS programming language for performing various types of interpolations. In our initial example, we focused on the linear interpolation case, but it is important to highlight that this procedure offers a wide range of methods and options to suit different analytical needs.

Proc Expand is not limited to linear interpolation; it also allows for more sophisticated methods such as splines, kernel smoothing, and other advanced algorithms for handling timestamped data.

These methods can be particularly useful when working with time series that have missing data points or when there is a need to transform high-frequency data into lower-frequency data, or vice versa. To illustrate a practical application, we created simulated datasets.

However, for a detailed analysis of its full functionalities and numerous available options, consulting the official SAS documentation is essential. This documentation provides comprehensive explanations and practical examples that cover all aspects of the procedure.

References

[1] A. Quarteroni, R. Sacco, F. Saleri, *Matematica Numerica*, Springer, 2008

[2] *SAS/ETS® 13.2 User's Guide The EXPAND Procedure*, SAS Institute Inc., 2014