PharmaSUG 2026 Paper AS-190

Implementing Dynamic Time Warping in SAS 9.4 Using PROC IML: An Alternative Approach for Time-Series Model Evaluation

Yida Bao, University of Wisconsin Stout;
Philippe Gaillard, Augusta University;
Wei Yao, University of Wisconsin Stout
Zheng Zhang, Murray State University
Rui Wang, Datapelago

ABSTRACT

Dynamic Time Warping (DTW) is a useful method for comparing time-series signals when timing shifts or delays are present. Common evaluation metrics such as Mean Squared Error (MSE) focus on point-by-point differences and often fail to reflect similarity in overall shape or timing. As a result, models that visually match the behavior of a biological signal may still receive poor numerical scores. In this paper, we implement DTW in SAS 9.4 using PROC IML and provide a practical way to compute dynamic distances between time-series signals directly within SAS. Using both simple examples and real ECG data, we show that DTW captures waveform similarity more effectively than traditional error-based measures when temporal misalignment exists. This work provides SAS users with a straightforward and interpretable tool for evaluating time-dependent data.

INTRODUCTION

In pharmaceutical research, time-series data are everywhere, for example, concentration curves, vital signs, or pharmacodynamic responses that change over time. Evaluating such data is more than just checking whether a model's predictions are close to the observed values.

A good evaluation should tell us *how well* a model captures the shape, timing, and overall behavior of the biological signal, not just the average level.

Figure 1 shows an example based on real pharmacy data. The first is a linear regression model, which focuses mainly on fitting the overall trend. The second is a prediction model designed to reproduce the observed response, including its natural fluctuations.

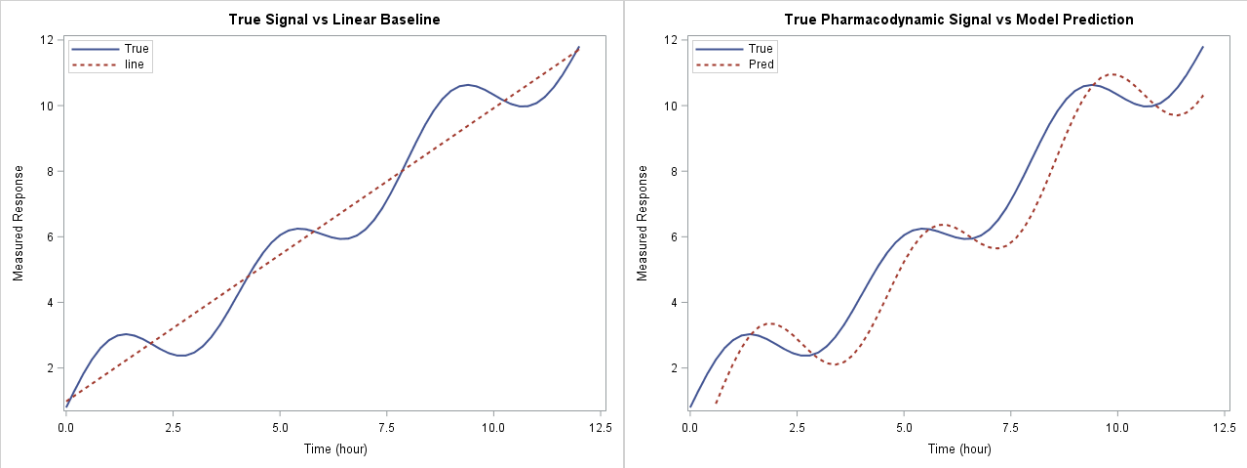

Figure 1. Comparing Model Prediction and Linear Baseline Against the True Signal

The right panel shows the model prediction (Pred), which successfully reproduces the oscillatory shape of the true signal, though with a slight phase delay. Such lagging behavior is common in biological systems where absorption, distribution, or regulatory mechanisms cause temporal shifts in response. On the other

hand, the left panel presents a linear baseline model (Line) that fits the overall upward trend well but entirely fails to represent the cyclical pattern of the true signal.

From a qualitative standpoint, I believe that most analysts would agree that the right model captures the biological reality far better.

Then, when we attempt to validate this intuition using numerical evaluation metrics, the story becomes more complicated. The most widely used performance measure in model fitting is the Mean Squared Error (MSE), defined as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where y_i is the observed value, y_i is the model-predicted value, and n is the number of time points. MSE quantifies the average squared difference between observed and predicted responses, rewarding models that minimize point-by-point deviation.

Applying MSE to the two models in Figure 1 yields the following results:

Model	MSE
Linear Regression Model (Line)	0.55
Predicted Model (Pred)	1.67

Table 1. MSE Result

Numerically, the linear model appears superior because of its smaller MSE value. However visually, the prediction model is clearly more faithful to the true signal's behavior, and it captures the rise and fall of the system, not just its mean trajectory.

This inconsistency reveals a fundamental limitation of pointwise error measures such as MSE: they evaluate magnitude accuracy but fail to recognize temporal or shape similarity between two time-dependent curves.

Therefore, in the remainder of this study, we introduce an additional evaluation metric tailored for time-dependent signals: **Dynamic Time Warping (DTW)**. Unlike pointwise measures, DTW provides a more structure assessment of how closely two time series resemble each other across their entire temporal shape.

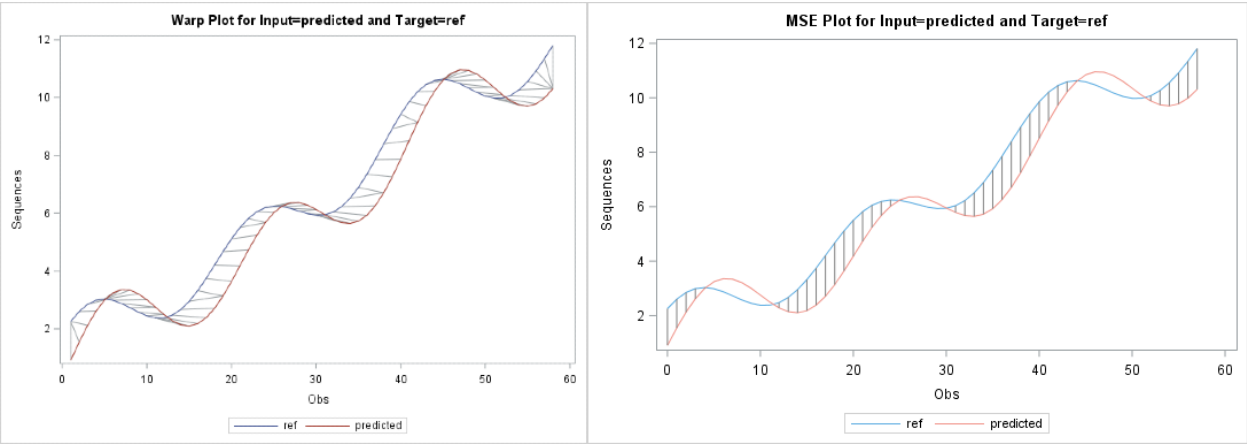

Figure 2. Visual Contrast Between Shape-Based (DTW) and Pointwise (MSE) Evaluations

To illustrate this problem, Figure 2 presents a comparison of the MSE-based discrepancy and the DTW-based discrepancy between the true signal and the predicted curve. In the MSE panel, the discrepancy between the two curves reflects only their vertical distances at each time point. This point-by-point

comparison ignores how the signals move over time. In contrast, the DTW panel emphasizes the overall shape: the predicted curve follows the general rise and fall of the true signal, even when there are slight timing shifts. The figure clearly shows that DTW values structural similarity, whereas MSE focuses on local vertical error.

To compare the two models under this new metric, we compute DTW on the same dataset. The results are summarized below:

Model	DTW
Linear Regression Model (Line)	12.49
Predicted Model (Pred)	4.51

Table 2. DTW Result

The DTW values show a clear gap between the two models. The predicted model produces a much smaller DTW distance, which means its overall shape is closer to the true signal. This result is consistent with the visual plots: the predicted curve follows the main rises and drops of the reference, while the linear model stays closer to a straight trend. DTW captures this difference directly, so the lower DTW value for the predicted model aligns with what we expected from the earlier comparison.

DYNAMIC TIME WARPING ALGORITHM

Dynamic Time Warping (DTW) computes the similarity between two time-series sequences by allowing flexible alignment along the time axis. Given two sequences

$$X = (x_1, x_2, \dots, x_n), Y = (y_1, y_2, \dots, y_m),$$

DTW constructs an $n \times m$cost matrix where each entry stores the local distance:

$$d(i,j) = | x_i - y_j |.$$

A cumulative distance matrix $D$is then built using the following recursion:

$$D(i,j) = d(i,j) + \min \{D(i-1,j), D(i,j-1), D(i-1,j-1)\}.$$

The DTW distance is the value in the bottom-right corner, $D(n,m)$, representing the minimum cumulative distortion required to align the two sequences. This formulation allows local stretching or compression in time, which is critical for signals where delays, shifts, or biological phase differences naturally occur.

Consider two sequences with identical shape but a one-step temporal shift:

X= (1,2,3), Y= (0,1,2),

the series share the same trend and shape; the only difference is a one-step temporal delay in Y.

This is exactly the type of situation where DTW provides a more meaningful similarity measure than ordinary pointwise metrics.

Then, we use DTW computes a local cost matrix $d(i,j) = \left|x_i - x_j\right|$

$$d = \begin{pmatrix} 1 & 0 & 1 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \end{pmatrix}$$

Then we use the DTW recursion, $D(1,1) = d(1,1) = 1$, $D(1,2) = d(1,2) + D(1,1) = 0 + 1 = 1$, $D(1,3) = 1 + 1 = 2$, $D(2,1) = d(2,1) + D(1,1) = 2 + 1 = 3$ $D(3,1) = 3 + 3 = 6$,

Then we try to find the value for interior cells,

$D(2,2) = d(2,2) + \min\{D(1,2), D(2,1), D(1,1)\} = 1 + min\{1,3,1\} = 2$

$D(2,3) = d(2,3) + \min\{D(1,3), D(2,2), D(1,2)\} = 0 + min\{2,2,1\} = 1$

$D(3,2) = d(3,2) + \min\{D(2,2), D(3,1), D(2,1)\} = 2 + min\{2,6,3\} = 4$

$D(3,3) = d(3,3) + \min\{D(2,3), D(3,2), D(2,2)\} = 1 + min\{1,4,2\} = 2$

Then the cumulative distance matrix is

$$D = \begin{pmatrix} 1 & 1 & 2 \\ 3 & 2 & 1 \\ 6 & 4 & 2 \end{pmatrix}$$

So the DTW distance is DTW (X,Y) = D(3,3) = 2

This example illustrates how the DTW algorithm accumulates local distances through a dynamic programming recursion to obtain the minimal warping cost. Each entry $D(i,j)$ represents the minimum cumulative distance required to align the prefixes $X_{1:i}$and $Y_{1:j}$, considering all valid warping paths up to that point. The final value $D(n,m)$therefore corresponds to the globally optimal alignment cost between the two sequences. By computing the cumulative matrix from the local cost matrix and applying the recursive minimum rule, DTW effectively identifies the least-distortion warping path without exhaustively enumerating all possible alignments.

SAS CODE

To compute the DTW distance in SAS, we implement the algorithm in PROC IML and wrap it in a reusable module called DTW. The module takes two numeric vectors x and y. It returns a single scalar: the DTW distance between the two series.

Below is the core code:

```
/* PROC IML: DTW distance  */
proc iml;
/* DTW module: x, y are column vectors; window is the Sakoe-Chiba band (. means no limit)
*/
start DTW(x, y, window);
   x = colvec(x);  y = colvec(y);

   /* Remove missing values */
   ix = loc(^missing(x)); if type(ix)='U' then return(.);
   iy = loc(^missing(y)); if type(iy)='U' then return(.);
   x = x[ix];  y = y[iy];

   n = nrow(x);  m = nrow(y);
   w = ifn(missing(window), max(n,m), window);

   big = 1e30;
```

```
   cost = j(n+1, m+1, big);
   cost[1,1] = 0;

   do i = 2 to n+1;
      /* Restrict alignment to a diagonal band to avoid full matrix computation */
      lo = max(2, i - w + 1);
      hi = min(m+1, i + w - 1);

      do j = lo to hi;
         d = abs( x[i-1] - y[j-1] );    /* L1 distance, can switch to (x-y)**2 for L2 */
         a = cost[i-1, j  ];
         b = cost[i,   j-1];
         c = cost[i-1, j-1];
         mmin = min(a,b);  mmin = min(mmin,c);
         cost[i,j] = d + mmin;
      end;
   end;

   return( cost[n+1, m+1] );
finish;
```

The first part of the module reshapes the inputs into column vectors and drops any missing values. If all entries are missing in either series, the function simply returns a missing value. We then set up the cost matrix cost with a very large initial value big, and initialize cost[1,1] = 0. The double loop over i and j fills this matrix only within the user defined window w. The local cost d is the absolute difference between x[i-1] and y[j-1]. Users can switch to squared differences if they prefer an L2 type distance. The final DTW distance is taken from the bottom right corner of the matrix.

To apply this module, we first read the two time series from a SAS data set into IML vectors, then call DTW and optionally save the result into a new data set:

```
/* Read example data: two columns as vectors */
use Ts_final;
   read all var {ref}       into x;
   read all var {predicted} into y;
close;

/* Compute DTW distance, third argument is the window (. means no limit) */
dist = DTW(x, y, .);
print dist[label="DTW Distance"];

/* Export the result to a data set if needed */
DTW_Distance = dist;
create dtw_out var {"DTW_Distance"};
   append;
close dtw_out;
quit;

/* Inspect the output data set */
proc print data=dtw_out;
run;
```

In this example, Ts_final contains two series: ref is the reference sequence and predicted is the sequence we want to compare to it. The call DTW(x, y, .) uses the widest possible window so the alignment is unconstrained. If we want to force the alignment path to stay close to the diagonal, we can replace . with a positive integer, for example DTW(x, y, 20).

Once dtw_out is created, the DTW distance is available like any other scalar metric in SAS and can be merged with other results, used inside macros, or passed into further modeling steps.

EXAMPLE: A SIMPLE DTW USAGE DEMONSTRATION IN PROC IML

Here's a minimal working example to illustrate how the DTW module can be used in practice.

We first construct a toy time-series dataset consisting of two sequences: an observed signal (signal_A) and a comparison signal (signal_B). The two series follow a similar upward trend and share the same

length, but signal_B is slightly shifted in time, creating a mild misalignment between corresponding observations.
This type of situation is common in real applications, where two signals exhibit similar shapes but do not align perfectly point-by-point.

```
data Ts_example;
  input time signal_A signal_B;
  datalines;
1   102  98
2   108  101
3   115  109
4   121  114
5   127  120
6   133  126
;
run;

proc print data=Ts_example noobs;
run;
```

time	signal_A	signal_B
1	102	98
2	108	101
3	115	109
4	121	114
5	127	120
6	133	126

Figure 3. Dataset used to demonstrate DTW.

After the dataset is constructed, the DTW module is executed in PROC IML. Specifically, the DTW module is run by executing the PROC IML block that defines the function, from proc iml; through return(cost[n+1, m+1]); finish;. Once the module is available, it can be applied to compute the DTW distance between signal_A and signal_B in Figure 3.

```
/* Read the example data */
use Ts_example;
   read all var {signal_A} into x;
   read all var {signal_B} into y;
close;

/* Compute DTW distance */
dist = DTW(x, y, .);
print dist[label="DTW Distance"];

DTW_Distance = dist;
create dtw_out var {"DTW_Distance"};
   append;
close dtw_out;

quit;

proc print data=dtw_out noobs;
run;run;
```

After the DTW module is defined, the following code computes the DTW distance for a given pair of time-series variables. To apply the DTW module to a different dataset, make sure to change signal_A and signal_B, which are the names of the input vectors read into PROC IML.

DTW_Distance
16

Figure 4. DTW distance for the example dataset.

To apply the DTW module to a different dataset, make sure to change signal_A and signal_B, which are the names of the input vectors read into DTW function.

SAS 9.4 provides PROC SIMILARITY for computing a range of similarity measures between ordered sequences, including options such as sliding subsequence matching. These methods are effective when comparing sequences under fixed or locally shifted index alignments. However, PROC SIMILARITY does not implement the dynamic programming framework that defines Dynamic Time Warping (DTW), nor does it search over the full set of admissible warping paths.

DTW, by contrast, is specifically designed to handle temporal misalignment. It allows local stretching and compression along the time axis and computes the distance between two sequences as the minimum cumulative cost over all valid warping paths, obtained via dynamic programming. This property enables DTW to recognize two signals with similar shapes but different temporal phases as being close, whereas pointwise or fixed-alignment measures may indicate substantial dissimilarity.

For this reason, the proposed PROC IML implementation should not be viewed as a replacement for PROC SIMILARITY. Instead, it serves as a complementary tool that provides an alignment-aware distance measure, which is often more appropriate for evaluating dynamic biological or pharmacometric processes. While PROC SIMILARITY and related metrics capture certain aspects of similarity, they are not equivalent to DTW's global warping alignment and cannot substitute for a true DTW-based measure.

REAL ECG SERIES APPLICATION

To demonstrate the practical applicability of the proposed DTW implementation, we conduct a real-data experiment using an electrocardiogram (ECG) dataset from the **UCR Time Series Classification Archive** (Hoang Anh Dau et al., 2019). We select the file *TwoLeadECG_TEST.tsv*, which is part of a well-known benchmark collection widely used in time-series classification research.

The TwoLeadECG dataset consists of ECG signals recorded under two different lead configurations and is originally designed for supervised classification tasks. Each observation is represented as a univariate time series with **82 time points**, capturing the temporal morphology of the ECG waveform. Although the dataset is primarily intended for classification, its structured and fixed-length time-series format makes it particularly suitable for illustrating similarity measures based on temporal alignment.

In this study, we do not focus on classification performance. Instead, we use the dataset to examine how DTW measures similarity between ECG signals at the waveform level. ECG signals often exhibit local temporal distortions due to physiological variability, noise, or measurement conditions, even when their overall shapes are similar. Such characteristics make ECG data a natural and meaningful application domain for DTW-based analysis.

The TwoLeadECG dataset contains more than 1,000 ECG time-series samples, each represented by 82 time points. In our analysis, we select sample 1 as the reference (query) series and compare it against other ECG signals in the dataset. Specifically, we examine sample 7 and sample 4 as two comparison cases.

For each pairwise comparison, we compute both the DTW distance and MSE. This setup enables a direct and interpretable comparison between a traditional point-wise distance measure and a time-warping-based similarity metric, using real biomedical time-series data.

Sample Pair	DTW	MSE
Sample 1 & 4	16.7701	0.11834
Sample 1 & 7	9.551	0.43868

Table 3. Comparison of DTW and MSE distances for selected ECG sample pairs.

The results in Table 3 reveal a clear and interesting contrast between DTW and MSE. For the comparison between Sample 1 and Sample 4, the MSE value is relatively small, suggesting a good point-wise fit under strict alignment. However, the corresponding DTW distance is larger, indicating that the two ECG signals require more temporal warping to achieve optimal alignment.

In contrast, when comparing Sample 1 and Sample 7, the DTW distance is substantially smaller, while the MSE is notably larger. This discrepancy suggests that, although the two signals are not well aligned point-by-point, their overall waveform shapes are highly similar once temporal misalignment is accounted for. In this case, MSE penalizes local shifts in time, whereas DTW captures the underlying morphological similarity between the ECG signals.

To further illustrate the differences observed in Table 3, we next use PROC SGPLOT to visualize the ECG waveforms.

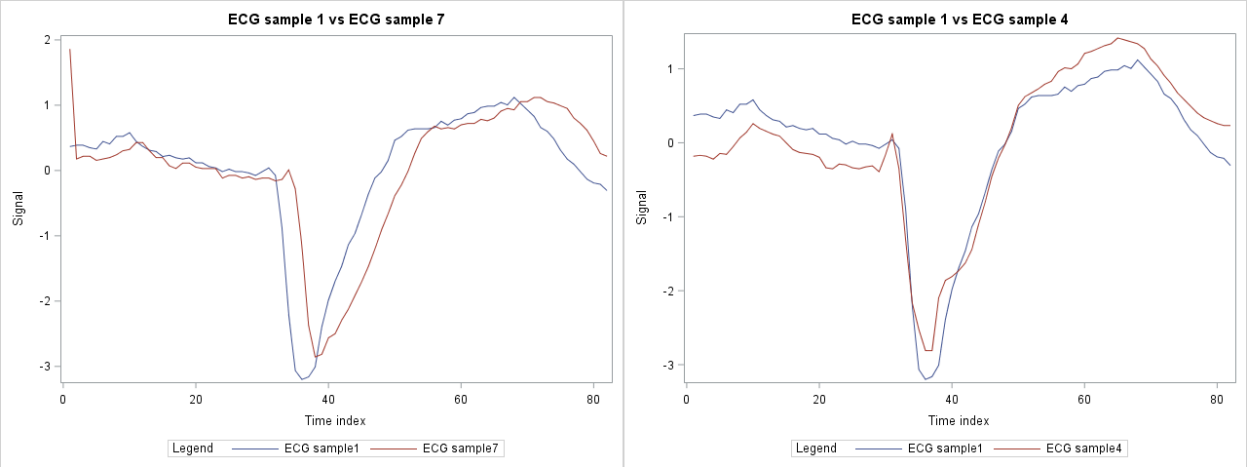

Figure 5. Waveform comparison for ECG Sample 1 versus Samples 7 and 4 (TwoLeadECG).

From the plots, the left panel (Sample 1 vs Sample 7) shows a noticeable time shift around the main trough and the recovery segment, but the overall waveform shape is highly consistent once the misalignment is ignored. This explains why DTW reports a smaller distance even though the point-wise MSE is larger. In contrast, the right panel (Sample 1 vs Sample 4) aligns more closely point-by-point, especially near the sharp drop, which leads to a smaller MSE. However, the two signals diverge more in overall morphology across the broader segment (for example, the baseline trend and the peak region), so the shape similarity is weaker, which is consistent with a larger DTW distance.

By applying DTW to the 82-dimensional time-series representations in the TwoLeadECG dataset, we highlight the ability of DTW to handle temporal misalignment and shape-based similarity, independent of class labels. This example illustrates how DTW can be effectively used as a foundational similarity metric in real-world biomedical time-series analysis, beyond its role in classification pipelines.

CONCLUSION

In this study, we demonstrated that commonly used pointwise error measures such as MSE can be inadequate for evaluating time-series models when temporal misalignment is present, even if the overall signal shape is well captured. Through both simulated examples and a real ECG application from the UCR Time Series Classification Archive, we showed that DTW provides a complementary and often more meaningful assessment by focusing on shape-based similarity across time.

We presented a clear and reusable implementation of the DTW algorithm in **PROC IML**, making DTW readily accessible within the SAS 9.4 environment. By applying the method to real ECG signals, we illustrated how DTW can distinguish between waveform-level similarity and strict pointwise agreement, explaining cases where MSE and DTW lead to conflicting conclusions. The visualization results further reinforced the interpretability of DTW in capturing biologically meaningful patterns that are obscured by traditional error metrics.

The primary contribution of this work lies in bridging a practical methodological gap for SAS users: while SAS provides several similarity measures through procedures such as PROC SIMILARITY, a full dynamic-programming-based DTW implementation has not been readily available. Our approach is not intended to replace existing SAS procedures, but rather to complement them by enabling alignment-aware distance analysis for dynamic biological and pharmacometric signals.

All code presented in this paper will be made available to the community. By sharing a transparent and extensible DTW implementation, we aim to support reproducible research and expand the analytical toolkit available to SAS 9.4 users, particularly in pharmaceutical, biomedical, and time-series modeling applications. We hope this work encourages broader adoption of shape-aware evaluation methods and contributes to the continued growth and vitality of the SAS analytics community.

REFERENCES

Senin, P. 2008. “Dynamic Time Warping Algorithm Review.” Information and Computer Science Department, University of Hawaii at Manoa, Honolulu, HI, Technical Report 855:1–23.

SAS Institute Inc. “The SIMILARITY Procedure.” SAS/ETS 13.2 Documentation. Accessed November 24, 2025. Available at https://support.sas.com/documentation/onlinedoc/ets/132/similarity.pdf.

Bao, Yida, Zheran Rachel Wang, Jingping Guo and Philippe Gaillard. 2020. “Special Plots Methods with Diabetes Disease Data.” [PharmaSUG 2020], 10–13.

Dau, Hoang Anh, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana and Eamonn Keogh. 2019. “The UCR Time Series Archive.” *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305.

Zhang, Zheng, Liangliang Xu, Yida Bao and Sanjeev Baskiyar. 2023. “Towards the Diagnosis of Heart Disease Using an Ensemble Learning Approach.” Proceedings of the 2023 International Conference on Machine Learning and Applications (ICMLA), 1901–1906. Jacksonville, FL: IEEE.

Zhang, Zheng, Amit Das, Mostafa Rahgouy, Yida Bao and Sanjeev Baskiyar. 2023. “Multi-Label Classification of CS Papers Using Natural Language Processing Models.” Proceedings of the 2023 International Conference on Machine Learning and Applications (ICMLA), 1907–1912. Jacksonville, FL: IEEE.

Bao, Yida, and Philippe Gaillard. 2019. “Summarizing Some Conventional Methods to Classify a Binary Target.” *Proceedings of the SouthEast SAS Users Group Conference (SESUG)*. Williamsburg, VA: LexJansen.

Guo, Jingping, Yida Bao, Robert Davis, Ash Abebe, Alan E. Wilson and Donald Allen Davis. 2020. “Application of Meta-Analysis towards Understanding the Effect of Adding a Methionine Hydroxy Analogue in the Diet on Growth Performance and Feed Utilization of Fish and Shrimp.” *Reviews in Aquaculture*, 12(4):2316–2332.

Bao, Yida and Philippe Gaillard. 2022. “Potential Variables Analysis Affects Covid-19 Spread.” 22–25.

Ayine, Priscilla, Vaithinathan Selvaraju, Chandra M. K. Venkatapoorna, Yida Bao, Philippe Gaillard and Thangiah Geetha. 2021. “Eating Behaviors in Relation to Child Weight Status and Maternal Education.” Children, 8(1):32.

Bao, Yida, and Philippe Gaillard. 2020. “Application and Comparison with Several Types of Ensemble Algorithms.” *Proceedings of the SouthEast SAS Users Group Conference (SESUG)*, 25–27.

Holland, Merrilee, Judith Hudson, Yida Bao and Philippe Gaillard. 2020. “Aortic to Caudal Vena Cava Ratio Measurements Using Abdominal Ultrasound Are Increased in Dogs with Confirmed Systemic Hypertension.” Veterinary Radiology and Ultrasound, 61(2):206–214.

Bao, Yida, Zheng Zhang, Jiafeng Ye, Liangliang Xu, Tao Jiang, Yongli Zhao, and Philippe Gaillard. 2024. “Evaluating Questionnaire Data Reliability with SAS.” *Proceedings of the Midwest SAS Users Group Conference (MWSUG)*, 17–19.

CONTACT INFORMATION

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

Name : Yida Bao
Work: University of Wisconsin Stout
Position: Assistant Professor
Email: baoy@uwstout.edu
Web: https://www.uwstout.edu/directory/baoy

APPENDIX: CODE FOR ECG PROJECT

Dataset : https://github.com/yzb0010/PharmaSUG_2026_Yida_Bao

```
proc import datafile="C:\Users\Yida\pharmasug 2026\TwoLeadECG_TEST.tsv"
    out=twolead_raw
    dbms=dlm
    replace;
    delimiter='09'x;   /* Tab */
    getnames=no;
    guessingrows=max;
run;


data twolead;
  set twolead_raw;
  class = var1;

  array v{82} var2-var83;
  array x{82} x1-x82;

  do i = 1 to 82;
    x{i} = v{i};
  end;

  drop i var:;
run;

/* add id = row index */
data twolead;
  set twolead;
  id = _n_;
run;

proc contents data=twolead; run;
```

```
/*============================================================
  1) Compute MSE + DTW for pairs (1,4) and (1,7)
  DTW module: unchanged
============================================================*/
proc iml;

/* ====== DTW module (UNCHANGED) ====== */
start DTW(x, y, window);
   x = colvec(x);  y = colvec(y);

   ix = loc(^missing(x)); if type(ix)='U' then return(.);
   iy = loc(^missing(y)); if type(iy)='U' then return(.);
   x = x[ix];  y = y[iy];

   n = nrow(x);  m = nrow(y);
   w = ifn(missing(window), max(n,m), window);

   big = 1e30;
   cost = j(n+1, m+1, big);
   cost[1,1] = 0;

   do i = 2 to n+1;
      lo = max(2, i - w + 1);
      hi = min(m+1, i + w - 1);

      do j = lo to hi;
         d = abs( x[i-1] - y[j-1] );
         a = cost[i-1, j  ];
         b = cost[i,   j-1];
         c = cost[i-1, j-1];
         mmin = min(a,b);  mmin = min(mmin,c);
         cost[i,j] = d + mmin;
      end;
   end;

   return( cost[n+1, m+1] );
finish;
/* ====== end DTW module ====== */

/* read X (1139 x 82) using the row-vector v trick */
use work.twolead;
v="x1"; do i=2 to 82; v = v || ("x"+strip(char(i))); end;
read all var v into X;
close work.twolead;

/* define the two pairs */
pairs = {1 4,
         1 7};

np = nrow(pairs);
out = j(np, 4, .);  /* q_id n_id mse82 dtw */

window = .;         /* keep as your default (no window) */

do k=1 to np;
  q = pairs[k,1];
  n = pairs[k,2];

  xq = X[q,]`;
  xn = X[n,]`;

  d2  = (xq-xn)##2;
  mse = mean(d2);
  dtw = DTW(xq,xn,window);

  out[k,1]=q;
```

```
  out[k,2]=n;
  out[k,3]=mse;
  out[k,4]=dtw;
end;

create pair_scores from out[colname={"q_id" "n_id" "mse82" "dtw"}];
append from out;
close pair_scores;

quit;

/* add classes (optional but nice) */
proc sql;
  create table pair_scores2 as
  select a.q_id, a.n_id,
         q.class as q_class,
         n.class as n_class,
         a.mse82, a.dtw
  from pair_scores a
  left join twolead q on a.q_id=q.id
  left join twolead n on a.n_id=n.id;
quit;

title "Pair distances (MSE82 and DTW)";
proc print data=pair_scores2 noobs; run;
title;


  2) Plot 2 figures:
     (1 vs 4) and (1 vs 7)

%let QID=1;
%let MSEID=4;

/* ---- Plot A: ECG sample 1 vs ECG sample 4 ---- */
data plotA_wide;
  set work.twolead;
  if id in (&QID, &MSEID);

  length role $40;
  if id=&QID then role=cats("ECG sample ", &QID);
  else role=cats("ECG sample ", &MSEID);
run;

data plotA_long;
  set plotA_wide;
  array x{82} x1-x82;
  do t=1 to 82;
    value=x{t};
    output;
  end;
  keep role id t value;
run;

title "ECG sample &QID vs ECG sample &MSEID ";
proc sgplot data=plotA_long;
  series x=t y=value / group=role name="s";
  keylegend "s" / title="Legend";
  xaxis label="Time index";
  yaxis label="Signal";
run;
title;

%let QID=1;
%let DTWID=7;

/* ---- Plot B: ECG sample 1 vs ECG sample 7 ---- */
```

```
    data plotB_wide;
      set work.twolead;
      if id in (&QID, &DTWID);

      length role $40;
      if id=&QID then role=cats("ECG sample ", &QID);
      else role=cats("ECG sample ", &DTWID);
    run;

    data plotB_long;
      set plotB_wide;
      array x{82} x1-x82;
      do t=1 to 82;
        value=x{t};
        output;
      end;
      keep role id t value;
    run;

    title "ECG sample &QID vs ECG sample &DTWID ";
    proc sgplot data=plotB_long;
      series x=t y=value / group=role name="s";
      keylegend "s" / title="Legend";
      xaxis label="Time index";
      yaxis label="Signal";
    run;
title;
```