The Many Faces of Partial Least Squares Regression

An Investigation of two Common Implementations

by Aidan J. Wagner

---

1 Introduction

The modern field of statistics is the cumulative sum of a history spanning centuries—a history that eagerly anticipated the development of the electronic computer. As such, since the earliest conception of computers, their development has been intimately intertwined with the development of statistical methods. As statisticians we do not stand on the shoulders' of giants; rather, we stand proudly atop a of pyramid comprised of our predecessors. No where is this more evident than in the case of Partial Least Squares (PLS). Originally synthesized from existing methods in the 1960s, over the next half century PLS has grown into a sprawling and flexible analytic tool that ought to be in any competent statistician's tool box. Over the years, PLS has been expanded and refined both conceptually and computationally. Researchers have been constantly deriving ways to apply PLS to a variety of problems; at the same time the algorithm used to compute PLS has undergone significant development: refining both its efficiency, accuracy, and flexibility.

The PLS umbrella includes a variety of methods that solve an assortment of multivariate hurdles. Of the so called "*Soft Modeling*" methods, PLS regression is particularly useful when the goal is prediction based on a large number of collinear predictors. As digital computers have developed, the number of predictors in question seems to grow according to Moore's Law. As a result of this, and the elegant yet robust solution PLS offers, it is no surprise that it has captured the attention of researchers since its conception. It seems that as the statistical landscape shifts so too does the implementation of PLS regression. While many of these refinements to the technique seek to address the needs of a specific use case; the general procedure has also been optimized over the years to leverage the computational power of modern computers. Of these optimizations, perhaps the most fascinating is the use of singular value decomposition (SVD) which allows all components to be calculated in the same decomposition, rather than calculating them iteratively.

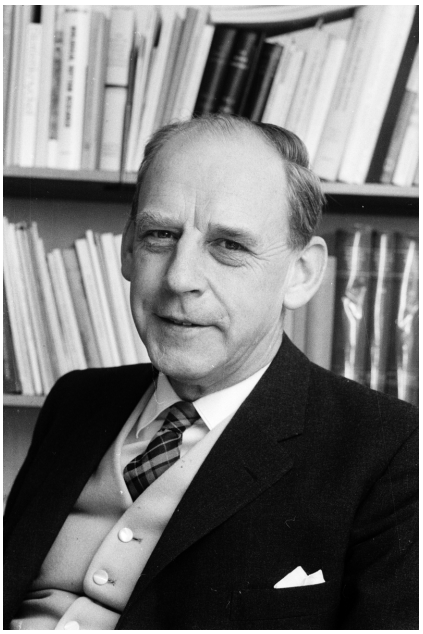
In order to fully appreciate the pros and

cons of SVD algorithms in calculating PLS regression it is important to understand where it came from. The following will provide an overview of the historical development of PLS, before investigating the initial iterative algorithm use to compute the components. From here, the fundamentals of SVD will be introduced, after which a revised algorithm that uses SVD will be explored and compared to the earlier iterative algortithm.

2 The Origins of PLS

The story of PLS begins in the snowy mountains of Upsala, Sweden. In the early 1960s, Herman Wold was working on a method to analyze the relationship between a set of predictors and a set of responses. He was particularly interested in the case where the predictors were highly collinear, which made traditional regression methods unreliable. Wold's solution was to use a latent variable approach, where he would extract a small number of latent variables from the predictors and use them to predict the responses. This method was later named Partial Least Squares (PLS). This initial algorithm was called "NILES" standing for "Nonlinear Iterative Least Squares", which was later updated to "NIPALS" (Nonlinear Iterative Partial Least Squares).

Coming from a background in econometrics, in which the cut and dry independent predictors demanded by OLS are rarely found, Wold was no stranger to the challenges introduced by multicollinearity. He first laid the groundwork for what would later become known as PLS in a 1966 paper in which he introduces a iterative least squares approach that can be used to estimate the principal components of a set of predictors [11]. Less than three years later, he and a group of coauthors published furter applications of these kinds of iterative least squares algorithms to solve a number of problems in econometrics (including canonical correlation analysis and and fix-point estimation)[15]. The emergence of PLS from principal component analysis is no coencidence. The two methods share a common goal: to reduce the dimensionality of a dataset while preserving as much information as possible. However, PLS takes this a step further by also considering the relationship between the predictors and the responses, making it a more powerful tool for regression analysis. However, it would take a few more years before the method would fully galvanize into a cohesive algorithm. [12]



Herman Wold;
the father of PLS [10]

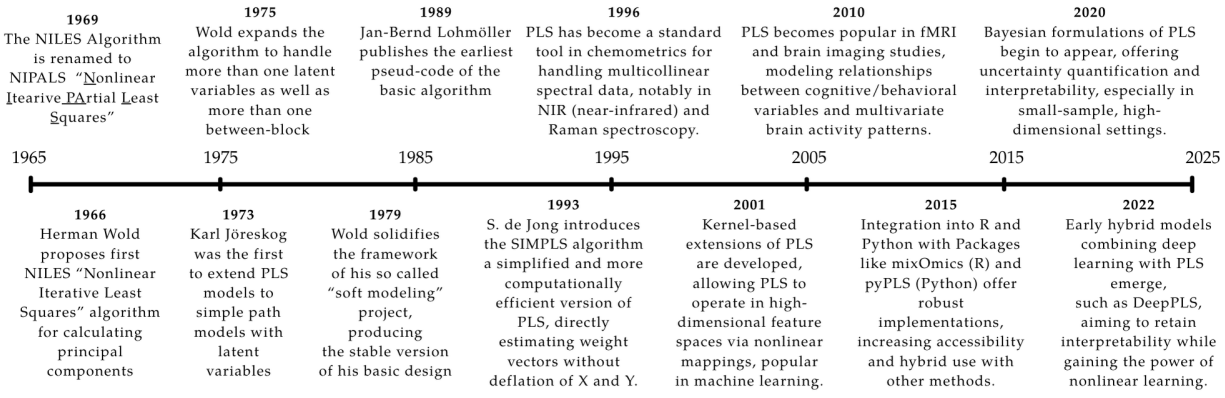
In the years following Wold's initial work, interest in PLS, which was still in its infancy, continued to grow. In 1975, Wold

and his colleagues published a paper that provided a more detailed description of the PLS algorithm and its applications, expanding the models to include an increasing number of predictors[13]. This paper laid the foundation for many of the subsequent developments in PLS, including the introduction of cross-validation techniques to assess the performance of PLS models. By the end of the 70s the research had solidifed enough that Wold and his colleagues were able to iron out the details of the method and publish the first stable version of the so called "Basic Design" of PLS, in which a more formal treatment of the convergence of the algorithm was provided [14].

In the 1980s, PLS gained popularity in the chemometrics community (having acceptance in the field as common practice by the 1990s), where it was used to analyze complex data sets in fields such as spectroscopy and chromatography. The iterative NIPALS approach was particularly well-suited for this because it could handle large numbers of varibles, as well as missing data [16]. Concurrently, a number of software packages were developed to implement PLS, making it more accessible to researchers in various fields, but there was relatively little theoretical development of the method itself.

However, this brief stall in the theoretical development of PLS was short lived. In the 1990s, PLS experienced a resurgence in popularity, thanks in part to the increasing availability of powerful computers and software packages. Of particular interest to us is the publication of the first SVD-based PLS algorithm by Sijmen de Jong in 1993 [3]. This paper introduced a new approach to PLS that used singular value decomposition (SVD) to compute the components of the model, rather than the iterative least squares approach used in earlier versions of PLS. This new method was more efficient and allowed for the computation of all components in a single step, making it particularly useful for large data sets. Researchers began to explore new applications of PLS in fields such as genomics, proteomics, and metabolomics, where the method was used to analyze high-dimensional data sets with many more predictors than observations. [7] This led to the development of new variants of PLS, such as sparse PLS and kernel PLS, which were designed to handle specific challenges in these fields.

At the dawn of the 21st century, PLS was firmly established as a powerful and versatile tool for multivariate data analysis. PLS



found yet another application in the imaging of the brain. In this context, PLS was used to analyze functional magnetic resonance imaging (fMRI) data, where it was used to identify brain regions that were activated during specific tasks or conditions.[5] The method has further been adapted for use in machine learning and data mining, where it was used for tasks such as classification and clustering. [9] This was accomplished via kernel methods, which allowed PLS to be applied to non-linear data sets of exeedingly high dimension. Furthermore, more people than ever are using PLS thanks in large part to the increasingly wide spread availability of comptational programs like R and python through community driven implentations of the methods. [8]

In recent years, PLS has continued to evolve and adapt to new challenges in data analysis. Researchers have developed new algorithms and software packages to improve the efficiency and accuracy of PLS, as well as new methods for assessing the performance of PLS models. The method has also been integrated into other statistical techniques, such as Bayesian methods and ensemble learning, further expanding its applicability. [4] With the recent surge in interest in machine learning and artificial intelligence, PLS has found new applications in these fields as well, where it is used to analyze large and complex data sets. The method has also been adapted for use in deep learning, where it is used to extract features from high-dimensional data sets and improve the performance of neural networks. [6] As we move forward into the future, it is clear that PLS will continue to be a valuable tool for researchers and practitioners in a wide range of fields.

3 The NIPALS Algorithm

So, how does one actually compute PLS regression? The NIPALS algorithm is a simple and elegant solution to the problem of estimating the PLS components, perhaps that is why the earliest formulations of PLS regression use this itterative approach. [13] The algorithm works by iteratively estimating the weights and scores of the components, using a two-step process. In the first step, the weights are estimated by regressing the response variable onto the predictor variables. In the second step, the scores are estimated by projecting the predictor variables onto the weights. This process is repeated until convergence, at which point the final weights and scores are obtained.

The initial residual matrices are set to the scaled inputs: $\mathbf{E} = \mathbf{X}_{\text{scaled}}$ and $\mathbf{F} = \mathbf{Y}_{\text{scaled}}$. Before iterating through components, memory is allocated for the key matrices: the scores $\mathbf{T}$ and $\mathbf{U}$ (of dimensions $n \times H$), the loadings $\mathbf{P}$ and $\mathbf{Q}$ (of dimensions $p \times H$ and $q \times H$, respectively), the weights $\mathbf{W}$ and $\mathbf{C}$ (also $p \times H$ and $q \times H$), and vectors to track variance explained by each component. Where n denotes the number of observations, p is the number of predictor variables, q is the number of response variables, and H is the number of latent components (or dimensions) to extract during the iterative PLS process. Additionally, the total sum of squares for $\mathbf{X}$ and $\mathbf{Y}$ is computed to normalize variance explained metrics.

For each component $h = 1, \ldots, H$, the algorithm performs the following steps. First, a response score vector $\mathbf{u}$ is initialized randomly, typically with a fixed seed to ensure reproducibility. Then, an iterative procedure begins to extract latent variables:

- The predictor weights are updated as

$\mathbf{w} = \mathbf{E}^\top \mathbf{u}$ and normalized.
- These weights yield the predictor score $\mathbf{t} = \mathbf{E}\mathbf{w}$, which is also normalized.
- The response loading vector is computed as $\mathbf{q} = \mathbf{F}^\top \mathbf{t}$ and normalized.
- A new response score vector is computed as $\mathbf{u} = \mathbf{F}\mathbf{q}$.

These steps are repeated until the predictor score $\mathbf{t}$ converges—typically measured by the relative change in successive iterations falling below a specified tolerance threshold.

Once convergence is achieved, a scalar regression coefficient is computed as $b = \mathbf{t}^\top \mathbf{u}$, representing the correlation between the score vectors. The predictor loading is computed as $\mathbf{p} = \mathbf{E}^\top \mathbf{t}$.

Next, the residual matrices are deflated. The predictor matrix is updated via $\mathbf{E} \leftarrow \mathbf{E} - \mathbf{t}\mathbf{p}^\top$, and the response matrix is similarly deflated using $\mathbf{F} \leftarrow \mathbf{F} - b\,\mathbf{t}\mathbf{q}^\top$. This removes the information captured by the current component, ensuring subsequent components capture new variation.

The resulting component vectors are stored:

$$\begin{aligned} \mathbf{T}[, h] &= \mathbf{t} \\ \mathbf{U}[, h] &= \mathbf{u} \\ \mathbf{W}[, h] &= \mathbf{w} \\ \mathbf{Q}[, h] &= \mathbf{q} \end{aligned}$$

and the normalized response weight $\mathbf{C}[, h] = \frac{\mathbf{q}}{\|\mathbf{q}\|}$. The variance explained by this component in both $\mathbf{X}$ and $\mathbf{Y}$ is also recorded.

After extracting all H components, the algorithm performs post-processing. The cumulative variance explained is computed for both predictors and responses. The final scaled regression coefficient matrix is calculated as:

$$\mathbf{B}_{\text{scaled}} = \mathbf{W} \left(\mathbf{P}^\top \mathbf{W} \right)^{-1} \text{diag}(\mathbf{b}) \mathbf{Q}^\top,$$

where $\mathbf{b}$ is the vector of scalar coefficients from each component. This matrix is then rescaled to the original units of the data to obtain $\mathbf{B}_{\text{original}}$, the regression coefficients in the original space.

The algorithm returns all component matrices ($\mathbf{T}$, $\mathbf{U}$, $\mathbf{W}$, $\mathbf{C}$, $\mathbf{P}$, $\mathbf{Q}$), the regression coefficients, intercept, and the variance explained (both individual and cumulative) for each component.

4 Singular Value Decomposition

Before explaining the SVD-based PLS algorithm, it is important to understand the concept of singular value decomposition (SVD). SVD is a powerful mathematical technique used in linear algebra to decompose a matrix into three simpler matrices. Given a matrix $\mathbf{A}$ of dimensions $m \times n$, SVD can be expressed as:

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$$

where:

$\mathbf{U}$ is an $m \times m$ orthogonal matrix whose columns are the left singular vectors of $\mathbf{A}$.

$\mathbf{S}$ is an $m \times n$ diagonal matrix containing the singular values of $\mathbf{A}$, which are the square roots of the eigenvalues of $\mathbf{A}^\top \mathbf{A}$.

$\mathbf{V}$ is an $n \times n$ orthogonal matrix whose columns are the right singular vectors of $\mathbf{A}$.

The singular values in $\mathbf{S}$ are ordered from largest to smallest, and they provide information about the importance of each corresponding singular vector. The left singular vectors in $\mathbf{U}$ represent the directions of maximum variance in the data, while the right

singular vectors in $\mathbf{V}$ represent the directions of maximum covariance between the predictors and responses. *(Adapted from [2])*

5 SVD-based PLS Algorithm

The algorithm begins by standardizing the data: the predictor matrix $\mathbf{X}$ and response matrix $\mathbf{Y}$ are both centered and scaled to unit variance. The initial residual matrices are then defined as $\mathbf{E} = \mathbf{X}_{\text{scaled}}$ and $\mathbf{F} = \mathbf{Y}_{\text{scaled}}$. Prior to the component extraction loop, memory is allocated for all core matrices. These include the score matrices $\mathbf{T}$ and $\mathbf{U}$ (of size $n \times H$), the loading matrices $\mathbf{P}$ and $\mathbf{Q}$ (dimensions $p \times H$ and $q \times H$ respectively), the weight matrices $\mathbf{W}$ and $\mathbf{C}$ (also $p \times H$ and $q \times H$), and vectors to record variance explained at each component. Additionally, the total sum of squares for $\mathbf{X}$ and $\mathbf{Y}$ is computed to facilitate variance decomposition.

For each latent component $h = 1, \ldots, H$, the following procedure is executed. First, the cross-covariance matrix is calculated as

$$\mathbf{R} = \mathbf{E}^\top \mathbf{F}.$$

This matrix captures the linear association between predictor and response residuals. The algorithm then performs a singular value decomposition (SVD) on $\mathbf{R}$:

$$\mathbf{R} = \mathbf{U}_{\text{svd}} \mathbf{D} \mathbf{V}^\top.$$

From this decomposition, the first left and right singular vectors are extracted, denoted $\mathbf{w} = \mathbf{U}_{\text{svd},:,1}$ and $\mathbf{q} = \mathbf{V}_{:,1}$.

These vectors are then used to compute the latent score vectors. The predictor score is computed as $\mathbf{t} = \mathbf{E}\mathbf{w}$ and then normalized to unit length. The response score is calculated as $\mathbf{u} = \mathbf{F}\mathbf{q}$.

Following this, the predictor loading vector is computed:

$$\mathbf{p} = \mathbf{E}^\top \mathbf{t}.$$

To measure the alignment between predictor and response scores, a scalar regression coefficient is calculated:

$$b = \mathbf{t}^\top \mathbf{u}.$$

The residual matrices are then deflated to remove the structure explained by the current component:

$$\mathbf{E} \leftarrow \mathbf{E} - \mathbf{t}\mathbf{p}^\top, \qquad \mathbf{F} \leftarrow \mathbf{F} - b\,\mathbf{t}\mathbf{q}^\top.$$

This ensures that subsequent components capture new, orthogonal sources of variation.

The computed component vectors are stored as follows:

$$\mathbf{T}[, h] = \mathbf{t}$$
$$\mathbf{U}[, h] = \mathbf{u}$$
$$\mathbf{W}[, h] = \mathbf{w}$$
$$\mathbf{Q}[, h] = \mathbf{q}$$

with the normalized response weights recorded as $\mathbf{C}[, h] = \frac{\mathbf{q}}{\|\mathbf{q}\|}$. The variance explained in both $\mathbf{X}$ and $\mathbf{Y}$ by this component is computed and stored.

After all H components have been extracted, the algorithm performs final post-processing. Cumulative variance explained is computed for both matrices, and any components with negligible scalar regression coefficients are removed. The Moore-Penrose pseudoinverse [2] of the predictor loading matrix $\mathbf{P}$ is computed via the SVD-derived factorization:

$$\mathbf{P}^\dagger = \mathbf{V}\ \text{diag}\left(\frac{1}{d}\right) \mathbf{U}_{\text{svd}}^\top.$$

We cannot use the standard inverse of $\mathbf{P}$ here because $\mathbf{P}$ is not guaranteed to be square,

and as such the inverse is not gurranteed to be defined. The Moore-Penrose pseudoinverse is merely a generalization of the inverse that can be applied to non-square matrices, and those without full rank. Using this, the scaled regression coefficient matrix is assembled:

$$\mathbf{B}_{\text{scaled}} = \mathbf{P}^{\dagger} \operatorname{diag}(\mathbf{b}) \mathbf{Q}^{\top},$$

which is finally rescaled back to the original units of the input data, yielding $\mathbf{B}_{\text{original}}$.

The algorithm returns all relevant matrices—scores, loadings, weights—as well as the regression coefficients, intercept term, and detailed variance decomposition across components.

6 NIPALS vs. SVD-based PLS

SVD is particularly useful in PLS regression because it allows for the simultaneous computation of all components in a single step, rather than iteratively. This is achieved by applying SVD to the concatenated matrix of predictors and responses, which captures the relationships between them. This approach is more efficient and can lead to more accurate estimates of the components, especially in high-dimensional data sets. In usecases where the amount of data is so large that the iterative approach would be computationally expensive, SVD-based PLS can provide a more efficient solution. [3] Additionally, In applications where the number of predictors is much larger than the number of observations, SVD-based PLS can be particularly advantageous, as it avoids the numerical instability that can arise from the iterative approach. As a result of these properties, SVD-based PLS has become the preferred method for computing PLS regression in many applications. [5, 6, 7]

However, it is important to note that SVD-based PLS is not without its limitations. One potential drawback is that it can be sensitive to the scaling of the data, particularly when the predictors and responses are measured on different scales. This can lead to biased estimates of the components and regression coefficients. Furthermore, SVD-based PLS is not able to handle missing data in the same way as the iterative approach, which can be a limitation in some applications. In these cases, it may be necessary to use imputation techniques or other methods to handle missing data before applying SVD-based PLS. [17] Imputation techniques can introduce additional complexity and potential bias into the analysis, which means that the SVD algorithm is not necessarily a universally better implementation.

Oftentimes, the two algorithms will yield similar results, especially when the data is well-conditioned and the number of predictors is not excessively large. However, in cases where the data is ill-conditioned or the number of predictors is very large, the SVD-based PLS algorithm may provide more stable and accurate estimates of the components and regression coefficients. In these cases, it is important to carefully consider the choice of algorithm and its particular strengths and weaknesses.

In the following small examples, both implementations produced identical results due to the small size of these datasets; for the sake of brevity I will only list the output of one implementation per data set here, but the full results of each can be found in the appendix. The first dataset contains a number of predictors (e.g., price, sugar, alcohol content) and responses (e.g., hedonic rating, food pairing)

Note that this data is from [1] and this analysis produces nearly identical results to those obtained there, as this is how I verified the accuracy of my implementations. The data is shown below:

Table 1: Wine Dataset: Predictors and Responses

	Price	Sugar	Alcohol	Acidity	Hedonic	Goes with meat	Goes with dessert
Wine1	7	7	13	7	14	7	8
Wine2	4	3	14	7	10	7	6
Wine3	10	5	12	5	8	5	5
Wine4	16	7	11	3	2	4	7
Wine5	13	3	10	3	6	2	4

For this analysis we will use the first four predictors (Price, Sugar, Alcohol, and Acidity) to predict the remaining three (Hedonic, Goes with meat, Goes with dessert). The following is the output resulting from the implementation of the NIPALS algorithm discussed above on this dataset:

Table 2: X Weights (W)

Comp 1	Comp 2	Comp 3
0.5136996	-0.3379159	-0.3491784
-0.2010128	-0.9400125	0.1611530
-0.5704780	-0.0187754	-0.8210983
-0.6084912	0.0428572	0.4217840

Table 3: Y Weights (C)

Comp 1	Comp 2	Comp 3
-0.6092803	0.0517961	0.9672011
-0.7024173	-0.2683721	-0.2181410
-0.3679503	-0.9619218	-0.1301400

Table 4: X Loadings (P)

Comp 1	Comp 2	Comp 3
1.8706265	-0.6844684	-0.1796093
-0.0468215	-1.9977328	0.0828934
-1.9546910	0.0282876	-0.4223541
-1.9874206	0.0555821	0.2169560

Table 5: Y Loadings (Q)

Comp 1	Comp 2	Comp 3
-0.6092803	0.0517961	0.9672011
-0.7024173	-0.2683721	-0.2181410
-0.3679503	-0.9619218	-0.1301400

Table 6: X Scores (T)

Comp 1	Comp 2	Comp 3
-0.4538071	-0.4662144	0.5715796
-0.5399032	0.4940055	-0.4631016
0.0000000	0.0000000	0.0000000
0.4303963	-0.5326520	-0.5301329
0.5633140	0.5048609	0.4216549

Table 7: Y Scores (U)

Comp 1	Comp 2	Comp 3
-1.9451050	-0.7611282	0.6190969
-0.9347239	0.5305490	-0.5388495
0.2327121	0.6083728	0.0823078
0.9158460	-1.1575299	-0.6138664
1.7312708	0.7797364	0.4513112

Table 8: Regression Scalars (b)

Component	Estimate
1	2.756789
2	1.627162
3	1.119134

Table 9: Regression Coefficients (Original Scale)

	Hedonic	Goes_with_meat	Goes_with_dessert
Price	-1.00	-0.0333333	0.0416667
Sugar	0.75	0.2750000	0.5937500
Alcohol	-4.00	1.0000000	0.5000000
Acidity	2.75	0.1750000	0.0937500

Table 10: Variance Explained by Components (X)

Latent Vector	Explained Variance	Cumulative
1	70.4506%	70.4506%
2	27.8958%	98.3464%
3	1.6536%	100.0000%

Table 11: Variance Explained by Components (Y)

Latent Vector	Explained Variance	Cumulative
1	63.3324%	63.3324%
2	22.0638%	85.3962%
3	10.4372%	95.8333%

The second dataset is Fisher's Iris dataset, which contains measurements of various features of iris flowers e.g., sepal length, sepal width, petal length, and petal width (for the sake of this analysis the species variable has been omitted)

Table 12: Iris Dataset (First 6 Observations): Predictors and Responses

Sepal Length	Sepal Width	Petal Length	Petal Width
5.1	3.5	1.4	0.2
4.9	3	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5	3.6	1.4	0.2
5.4	3.9	1.7	0.4
⋮	⋮	⋮	⋮

The goal of the following analysis is to predict the petal length and width based on the sepal length and width. The following is the output resulting from the implementation of the SVD algorithm discussed above on this dataset:

Table 13: X Weights (W)

Comp 1	Comp 2
-0.9046320	-0.4261936
0.4261936	-0.9046320

Table 14: Y Weights (C)

Comp 1	Comp 2
-0.7350032	-0.5400163
-0.6780636	-0.8416546

Table 15: X Loadings (P)

Comp 1	Comp 2
-11.159218	-4.946903
6.224581	-10.500219

Table 16: Y Loadings (Q)

Comp 1	Comp 2
-0.7350032	-0.5400163
-0.6780636	-0.8416546

Table 17: Regression Scalars (b)

Component	Estimate
1	15.444539
2	1.203207

Table 18: Regression Coefficients (Original Scale)

	Petal.Length	Petal.Width
Sepal.Length	1.775592	0.7232920
Sepal.Width	-1.338623	-0.4787213

Table 19: Variance Explained by Components (X)

Latent Vector	Explained Variance	Cumulative
1	54.7898%	54.7898%
2	45.2102%	100.0000%

Table 20: Variance Explained by Components (Y)

Latent Vector	Explained Variance	Cumulative
1	80.0449%	80.0449%
2	0.4858%	80.5307%

7 Conclusion

Partial Least Squares (PLS) regression is a powerful subset of a broader family of statistical methods that continue to be of statistical interest. The NIPALS and SVD-based algorithms provide two different approaches to computing PLS regression, each with its own strengths and weaknesses; the NIPALS algorithm is simple and easy to implement, while the SVD-based algorithm is more efficient and can handle larger data sets. The choice of which to use will depend on the specific characteristics of the data set being analyzed, with neither algorithm universally outclassing the other. As the landscape of statiscial analysis evolves, the solutions it demands are likely to be increasingly multivariate, all but ensuring that it is only a matter of time before PLS finds its way into more and more applications. As researchers and practitioners continue to explore new ways to analyze increasingly complex data sets, PLS will undoubtedly linger, cementing Wold and his collegues in the continually generating history of statistics.

8 References

[1] Abdi, H. (2010). Partial least squares regression and projection on latent structure regression (PLS regression). *Technical report*. University of Texas at Dallas. https://www.utdallas.edu/~herve/Abdi-PLS-pretty.pdf

[2] Abdi, H., & Williams, L. J. (2013). Partial least squares methods: Partial least squares correlation and partial least square regression. *Methods in Molecular Biology (Clifton, N.J.)*, *930*, 549–579. https://doi.org/10.1007/978-1-62703-059-5_23

[3] de Jong, S. (1993). SIMPLS: An alternative approach to partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, *18*(3), 251–263. https://doi.org/10.1016/0169-7439(93)85002-X

[4] Ghosh, S., & Doshi-Velez, F. (2017). Model selection in Bayesian neural networks via horseshoe priors. *Journal of Machine Learning Research*, *20*(1), 1–46.

[5] Krishnan, A., Williams, L. J., McIntosh, A. R., & Abdi, H. (2011). Partial least squares (PLS) methods for neuroimaging: A tutorial and review. *NeuroImage*, *56*(2), 455–475. https://doi.org/10.1016/j.neuroimage.2010.07.034

[6] Kong, X., & Ge, Z. (2023). Deep PLS: A lightweight deep learning model for interpretable and efficient data analytics. *IEEE Transactions on Neural Networks and Learning Systems*, *34*(11), 8923–8937. https://doi.org/10.1109/TNNLS.2022.3154090

[7] Nguyen, D. V., & Rocke, D. M. (2002). Tumor classification by partial least squares using microarray gene expression data. *Bioinformatics*, *18*(1), 39–50. https://doi.org/10.1093/bioinformatics/18.1.39

[8] Rohart, F., Gautier, B., Singh, A., & Lê Cao, K.-A. (2017). mixOmics: An R package for 'omics feature selection and multiple data integration. *PLOS Computational Biology*, *13*(11), e1005752. https://doi.org/10.1371/journal.pcbi.1005752

[9] Rosipal, R., & Trejo, L. J. (2001). Kernel partial least squares regression in reproducing kernel Hilbert space. *Journal of Machine Learning Research*, *2*, 97–123.

[10] Wikipedia contributors. (2025, March 22). Herman Wold. Wikipedia. Wikimedia Foundation. https://en.wikipedia.org/wiki/Herman_Wold#/media/File:Professor_Herman_Wold,_Uppsala,_1969.jpg

[11] Wold, H. (1966). Estimation of principal components and related models by iterative least squares. In P. R. Krishnajah (Ed.), *Multivariate analysis* (pp. 391–420). New York: Academic Press.

[12] Wold, H. (1973). Nonlinear iterative partial least squares (NIPALS) modelling: Some current developments. In P. R. Krishnaiah (Ed.), *Multivariate analysis III* (pp. 383–407). Academic Press. https://doi.org/10.1016/b978-0-12-426653-7.50032-6

[13] Wold, H. (1975). Soft modelling by latent variables: The non-linear iterative partial least squares (NIPALS) approach. In *Perspectives in probabil-*

ity and statistics (pp. 117–142). Academic Press.

[**14**] Wold, H. (1979). Model construction and evaluation when theoretical knowledge is scarce: The PLS approach to latent variables. In K. G. Jöreskog & H. Wold (Eds.), *Systems under indirect observation: Causality, structure, prediction* (Vol. 2, pp. 47–74). North-Holland.

[**15**] Wold, H., & Lyttkens, E. (1969). Nonlinear iterative partial least squares (NIPALS) estimation procedures. *Bulletin of the International Statistical Institute*, *43*, 29–51.

[**16**] Wold, S., Sjöström, M., & Eriksson, L. (1996). PLS-regression: A basic tool of chemometrics. *Chemometrics and Intelligent Laboratory Systems*, *58*(2), 109–130. https://doi.org/10.1016/S0169-7439(01)00155-1

[**17**] Wright, K. (2017, October 27). The NIPALS algorithm. *R-Project.org*. https://cran.r-project.org/web/packages/nipals/vignettes/nipals_algorithm.html