A sparse PCA algorithm

sPCA-rSVD (Shen & Huang 2008)

matthieu.lesnoff@cirad.fr
ChemHouse, Montpellier, 22 January 2025

ELSEVIER

Available online at www.sciencedirect.com
ScienceDirect

Journal of Multivariate Analysis 99 (2008) 1015–1034

Journal of Multivariate Analysis

www.elsevier.com/locate/jmva

Sparse principal component analysis via regularized low rank matrix approximation

Haipeng Shen[a,*], Jianhua Z. Huang[b]

[a] *Department of Statistics and Operations Research, University of North Carolina at Chapel Hill, Chapel Hill, NC 27599, USA*

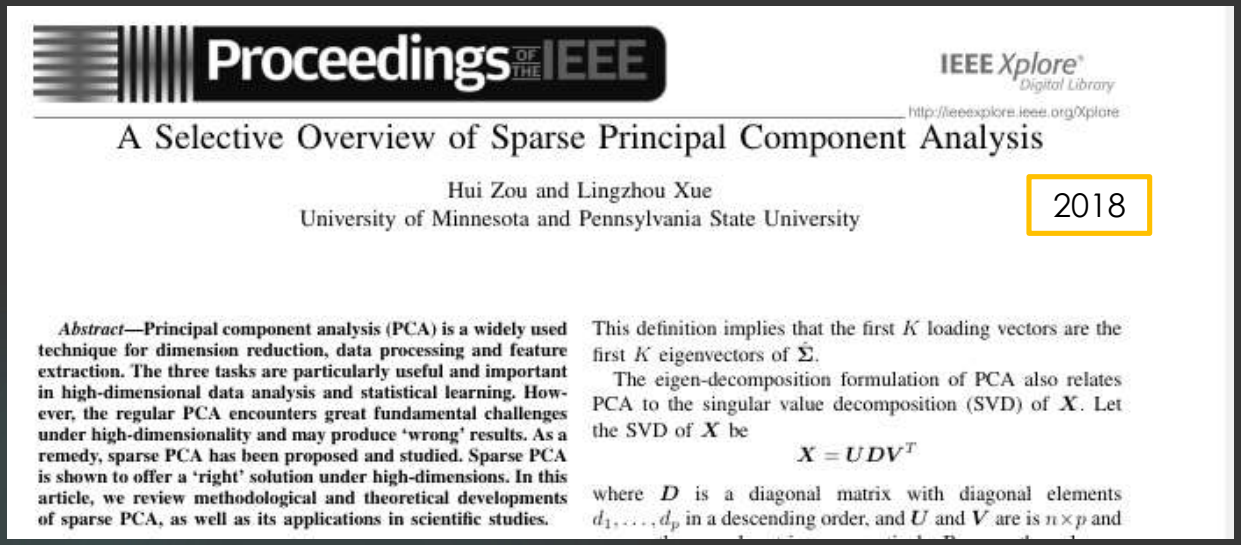
[b] *Department of Statistics, Texas A&M University, College Station, TX 77843, USA*

Received 25 July 2006
Available online 27 June 2007

Abstract

Principal component analysis (PCA) is a widely used tool for data analysis and dimension reduction in applications throughout science and engineering. However, the principal components (PCs) can sometimes be difficult to interpret, because they are linear combinations of all the original variables. To facilitate interpretation, sparse PCA produces modified PCs with sparse loadings, i.e. loadings with very few non-zero elements. In this paper, we propose a new sparse PCA method, namely *sparse PCA via regularized SVD* (sPCA-rSVD). We use the connection of PCA with singular value decomposition (SVD) of the data matrix and extract the PCs through solving a low rank matrix approximation problem. Regularization penalties are introduced to the corresponding minimization problem to promote sparsity in PC loadings. An efficient iterative algorithm is proposed for computation. Two tuning parameter selection methods are discussed. Some theoretical results are established to justify the use of sPCA-rSVD when only the data covariance matrix is available. In addition, we give a modified definition of variance explained by the sparse PCs. The sPCA-rSVD provides a uniform treatment of both classical multivariate data and high-dimension-low-sample-size (HDLSS) data. Further understanding of sPCA-rSVD and some existing alternatives is gained through simulation studies and real data examples, which suggests that sPCA-rSVD provides competitive results.

Many other algorithms than sPCA-rSVD, e.g.



Proceedings of the IEEE

IEEE Xplore Digital Library
http://ieeexplore.ieee.org/Xplore

A Selective Overview of Sparse Principal Component Analysis

Hui Zou and Lingzhou Xue
University of Minnesota and Pennsylvania State University

2018

***Abstract*—Principal component analysis (PCA) is a widely used technique for dimension reduction, data processing and feature extraction. The three tasks are particularly useful and important in high-dimensional data analysis and statistical learning. However, the regular PCA encounters great fundamental challenges under high-dimensionality and may produce 'wrong' results. As a remedy, sparse PCA has been proposed and studied. Sparse PCA is shown to offer a 'right' solution under high-dimensions. In this article, we review methodological and theoretical developments of sparse PCA, as well as its applications in scientific studies.**

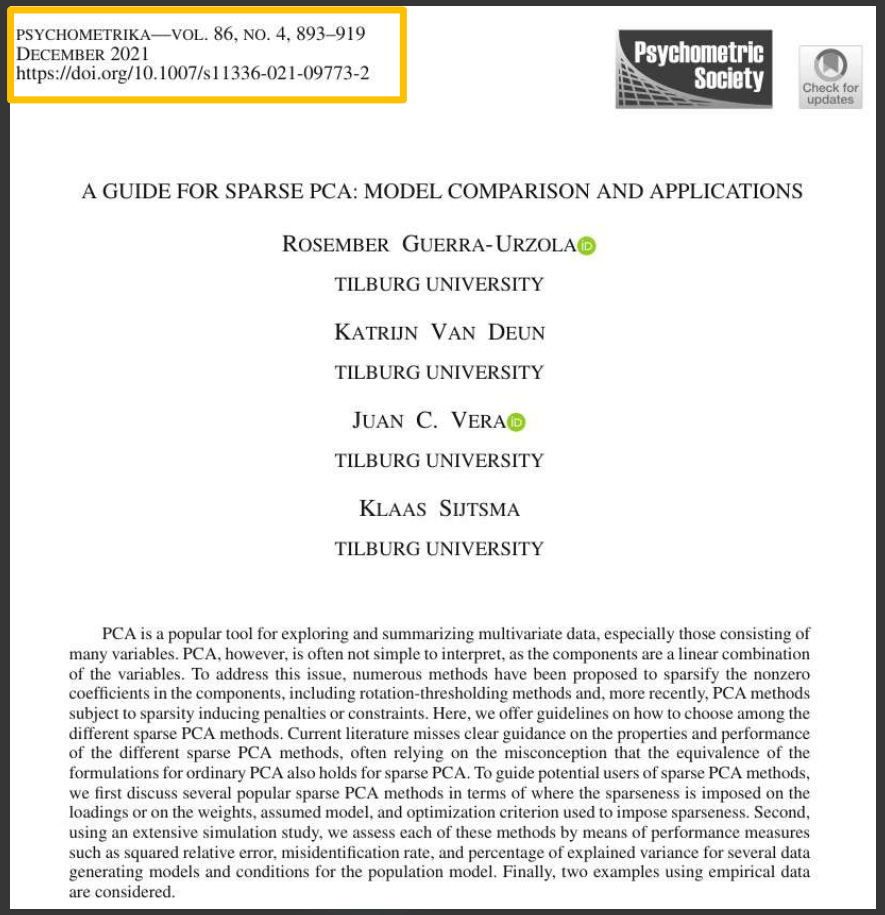
This definition implies that the first K loading vectors are the first K eigenvectors of $\hat{\Sigma}$.

The eigen-decomposition formulation of PCA also relates PCA to the singular value decomposition (SVD) of $\boldsymbol{X}$. Let the SVD of $\boldsymbol{X}$ be

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$$

where $\boldsymbol{D}$ is a diagonal matrix with diagonal elements $d_1, \ldots, d_p$ in a descending order, and $\boldsymbol{U}$ and $\boldsymbol{V}$ are is $n \times p$ and

[not exhaustive]



PSYCHOMETRIKA—VOL. 86, NO. 4, 893–919
DECEMBER 2021
https://doi.org/10.1007/s11336-021-09773-2

Psychometric Society

Check for updates

A GUIDE FOR SPARSE PCA: MODEL COMPARISON AND APPLICATIONS

ROSEMBER GUERRA-URZOLA
TILBURG UNIVERSITY

KATRIJN VAN DEUN
TILBURG UNIVERSITY

JUAN C. VERA
TILBURG UNIVERSITY

KLAAS SIJTSMA
TILBURG UNIVERSITY

PCA is a popular tool for exploring and summarizing multivariate data, especially those consisting of many variables. PCA, however, is often not simple to interpret, as the components are a linear combination of the variables. To address this issue, numerous methods have been proposed to sparsify the nonzero coefficients in the components, including rotation-thresholding methods and, more recently, PCA methods subject to sparsity inducing penalties or constraints. Here, we offer guidelines on how to choose among the different sparse PCA methods. Current literature misses clear guidance on the properties and performance of the different sparse PCA methods, often relying on the misconception that the equivalence of the formulations for ordinary PCA also holds for sparse PCA. To guide potential users of sparse PCA methods, we first discuss several popular sparse PCA methods in terms of where the sparseness is imposed on the loadings or on the weights, assumed model, and optimization criterion used to impose sparseness. Second, using an extensive simulation study, we assess each of these methods by means of performance measures such as squared relative error, misidentification rate, and percentage of explained variance for several data generating models and conditions for the population model. Finally, two examples using empirical data are considered.

Why sPCA-rSVD?

Guerra- Urzola et al. 2018

the sPCA-rSVD approach. Considering that the data generating model may be unknown and that there may be a mismatch in sparsity, sPCA-rSVD is overall the best method for recovering the relevant variables, and GPower performs best in terms of explained variance.

and relatively fast
to compute (Nipals)

[but not exhaustive
comparison]

PCA score and loadings vectors: t and v

$$t = X\,v \qquad \|v\| = 1$$

$(n \times 1) \quad (n \times p) \quad (p \times 1)$

$$v = \begin{pmatrix} v_1 \\ v_2 \\ . \\ . \\ v_{p-1} \\ v_p \end{pmatrix}$$

Columns of X

$$t = v_1 x_1 + v_2 x_2 + \ldots + v_p x_p$$

Sparse: Decrease/remove coefficients v_j of non important variables (penalization)

Objective function (for a given PC)

- PCA $\underset{v}{\operatorname{argmax}} \ \operatorname{var}(t)$

same as: $\underset{v}{\operatorname{argmin}} \ \|X - t\,v'\|^2$ Regression problem

- sPCA-rSVD

$$\underset{v}{\operatorname{argmin}} \ \{ \|X - t\,v'\|^2 + P_\lambda(v) \}$$

Penalization

sPCA-rSVD

$$\underset{v}{\operatorname{argmin}} \; \{ \|X - t\,v'\|^2 + P_\lambda(v) \} \qquad \text{with} \qquad P_\lambda(v) = \sum_{j=1}^{p} p_\lambda(|v_j|) \qquad \lambda > 0$$

Tuning variable

In the final algorithm, represented by a thresholding function f_λ (L1/lasso, hard, etc.)

[3 functions are described in Shen & Huang 2008]

Problem solving: Iterative alternated regressions (Nipals)

PCA

1. Set $u \quad (n \times 1)$
2. Repeat until convergence
   a) $v = X' u$
   b) $u = X v \,/\, \mathrm{norm}(X v)$
3. $v = v \,/\, \mathrm{norm}(v)$
4. $t = X v$

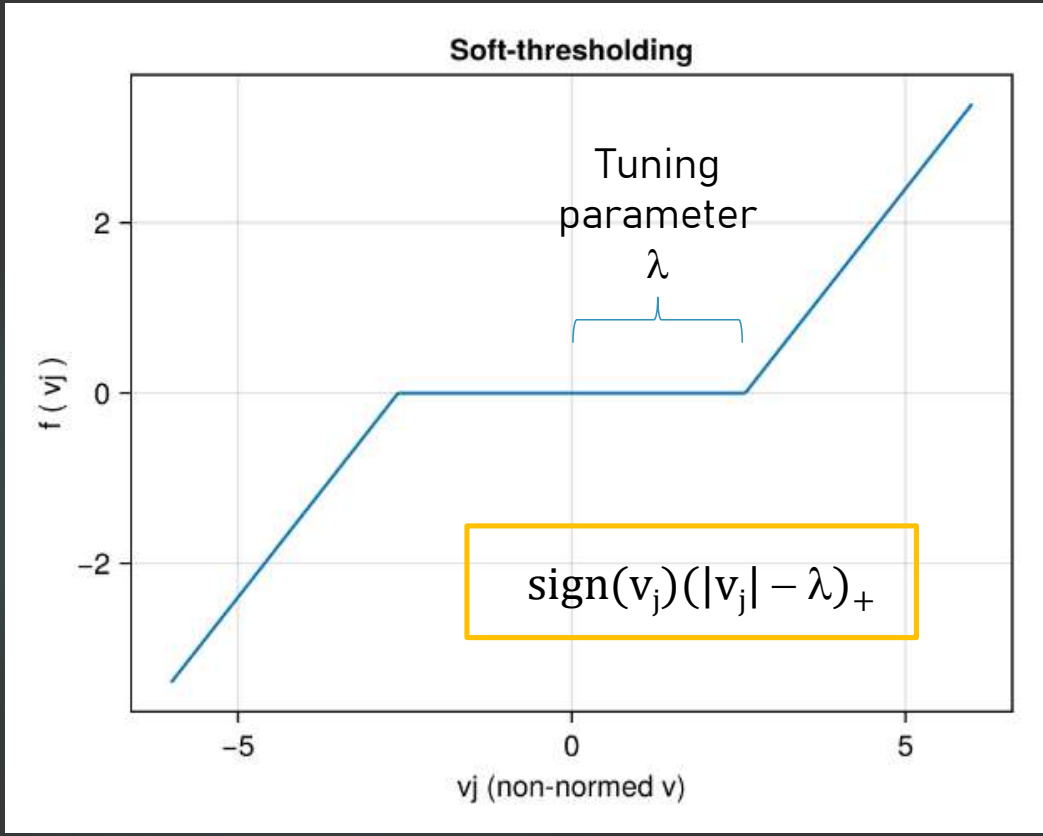
$$\underset{v}{\mathrm{argmin}} \; \{ \|X - t v'\|^2 \}$$

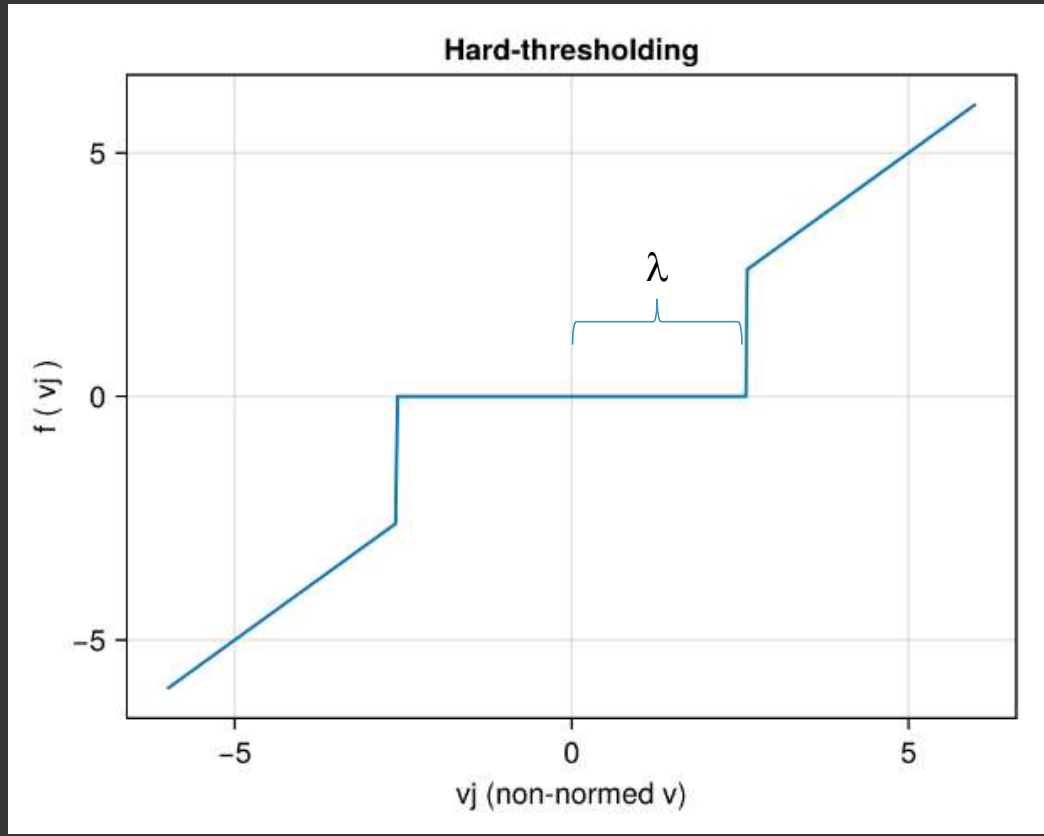
sPCA-rSVD (S&H 2008)

1. Set $u \quad (n \times 1)$
2. Repeat until convergence
   a) $v = f_\lambda(X' u)$ $\quad f_\lambda$: Threshold function
   b) $u = X v \,/\, \mathrm{norm}(X v)$
3. $v = v \,/\, \mathrm{norm}(v)$
4. $t = X v$

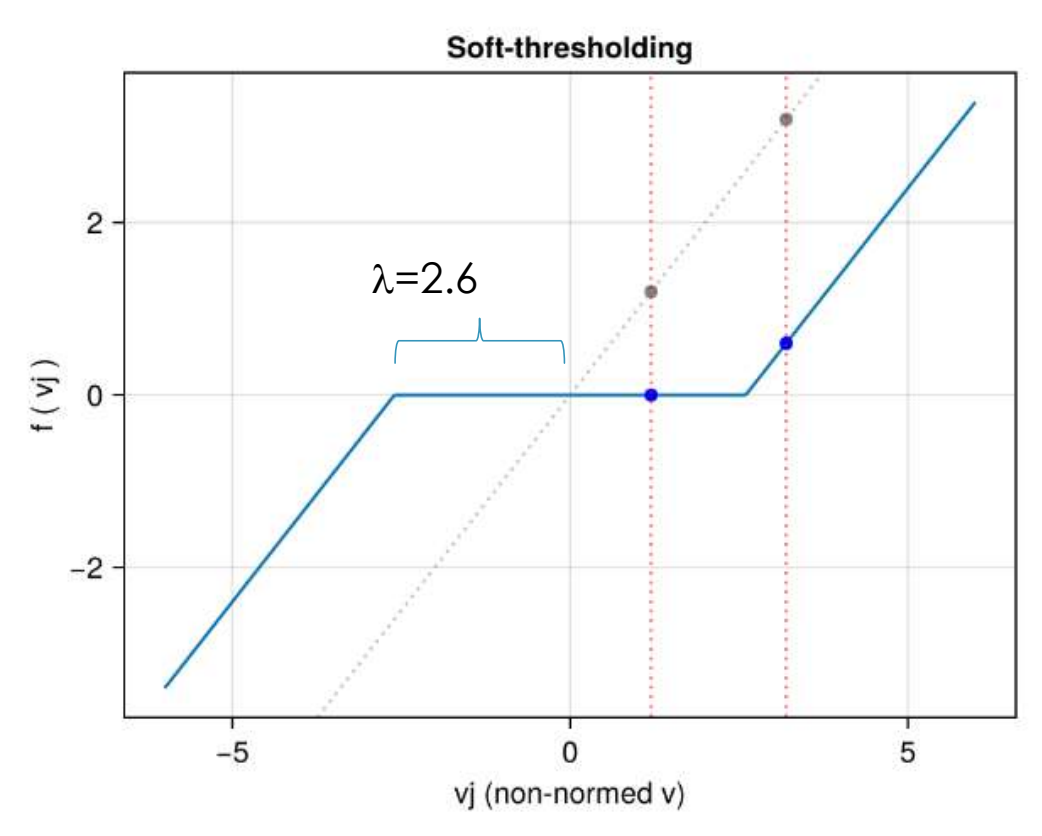
$$\underset{v}{\mathrm{argmin}} \; \{ \|X - t v'\|^2 + P_\lambda(v) \}$$

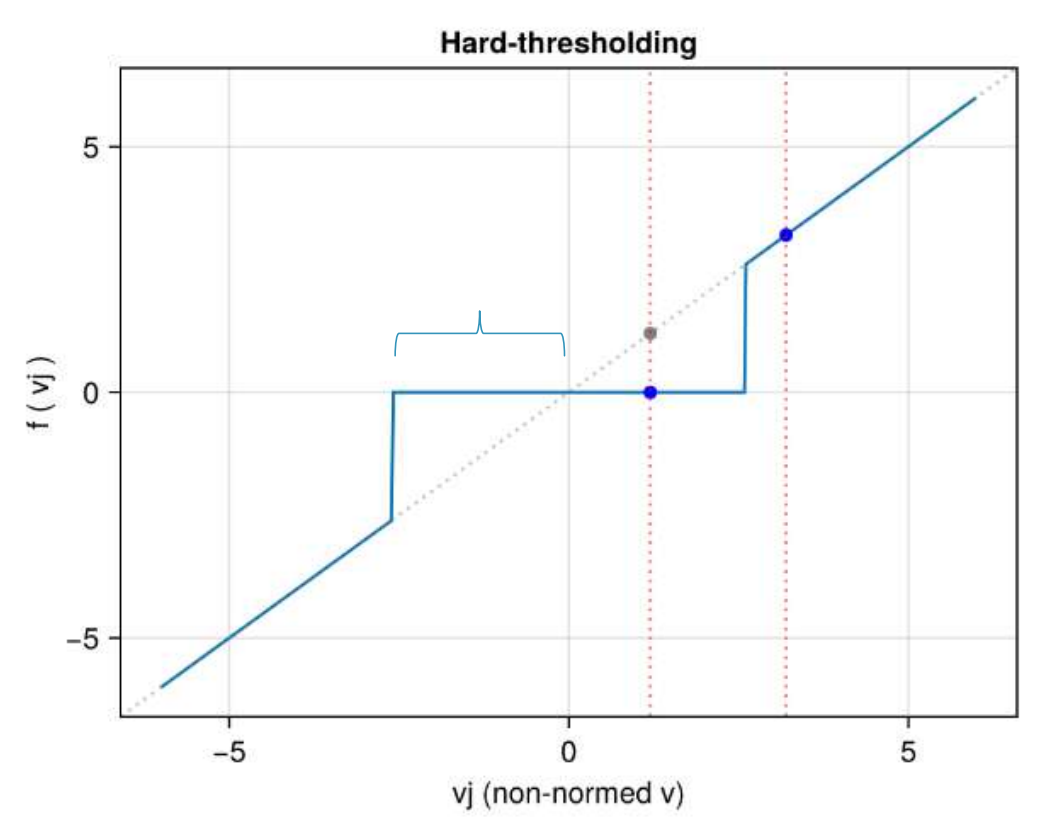
Two usual thresholding functions f_λ





L1 = LASSO


Soft-thresholding
λ=2.6
f (vj)
vj (non-normed v)
2
0
−2
−5
0
5

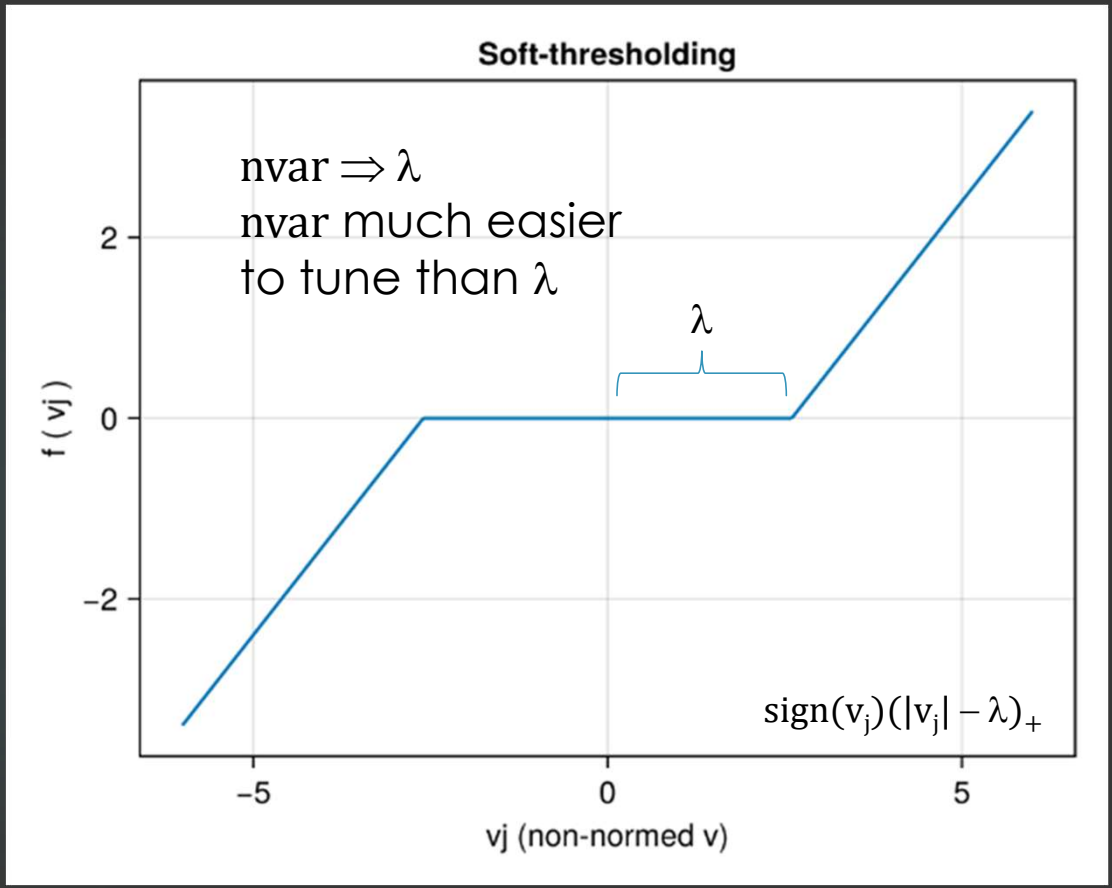

Hard-thresholding
f (vj)
vj (non-normed v)
5
0
−5
−5
0
5

Shen & Huang 2008 section 2.2 p.1020

To tune not directly λ, but:

nb. zero components in v = "degree of sparsity" k

or, equivalently, $nvar = p - k$ nb. selected variables (for the PC)



$\alpha = k / p = 1 - nvar / p$

$\lambda = \text{quantile}_{\alpha}(\{|v_1|, \ldots, |v_p|\})$

```
julia> v
7-element Vector{Float64}:
  1.5692830160261046
  1.2202436854388077
 -1.2464697097541109
 -2.635143873053083
  0.6369786289412306
  0.4045131244852491
  2.8170808401488165
```

```
julia> absv = abs.(v)
       sort(absv)
7-element Vector{Float64}:
 0.4045131244852491
 0.6369786289412306
 1.2202436854388077
 1.2464697097541109
 1.5692830160261046
 2.635143873053083
 2.8170808401488165
```

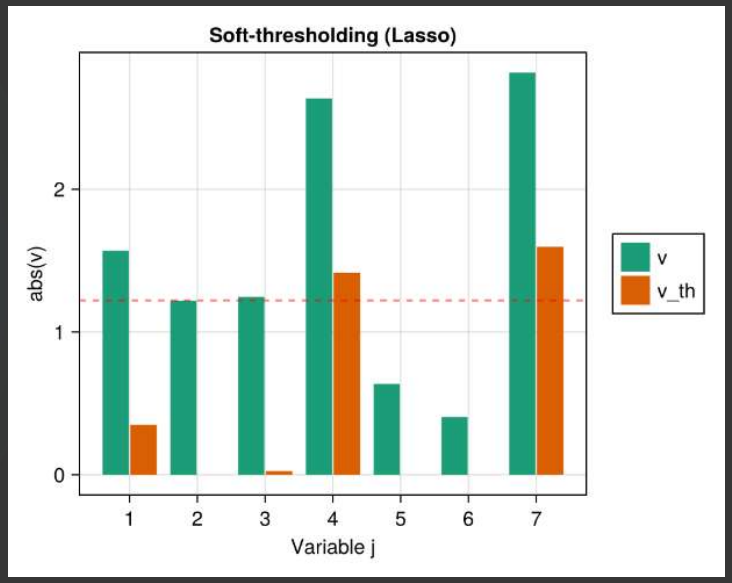
λ

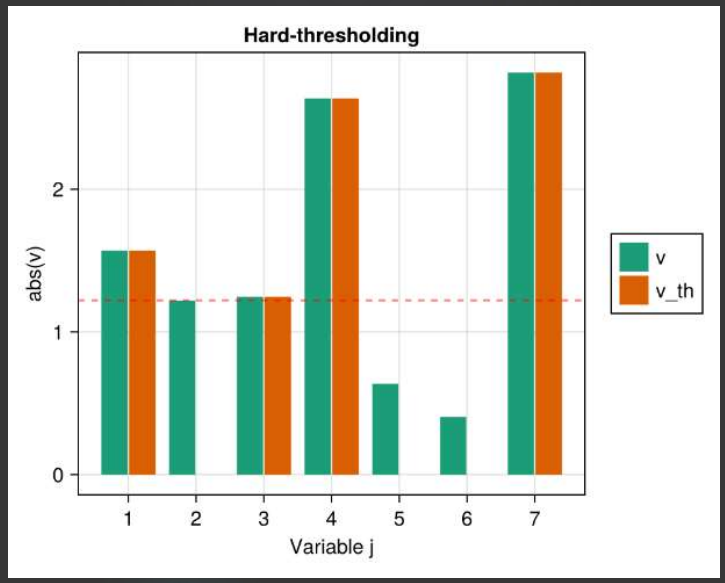
If nvar = 4

$\Rightarrow \quad 1.220 \leq \lambda < 1.246$

Rk: The effective nb. selected variables car vary from the specified parameter nvar if tied values vj

```
julia> v_th
7-element Vector{Float64}:
  0.34903933058729697
  0.0
 -0.026226024315303187
 -1.4149001876142753
  0.0
  0.0
  1.5968371547100089
```





Some variants

sPCA-rSVD (S&H 2008)

1. Set u $(n \times 1)$
2. Repeat until convergence
   a) $v = f_\lambda(X' u)$
   b) $u = X v / \text{norm}(X v)$
3. $v = v / \text{norm}(v)$
4. $t = X v$

sPCA-rSVD mixOmics::spca (*)

1. Set t $(n \times 1)$
2. Repeat until convergence
   a) $v = f_\lambda(X' t)$ Soft (Lasso)
   b) $v = v / \text{norm}(v)$
   c) $t = X v$
3. t, v

Same results

(*) Lê Cao et al. (2016). **mixOmics**: Omics Data Integration Project. R package version 6.1.1. https://CRAN.R-project.org/package=mixOmics

sPCA-rSVD (S&H 2008)

1. Set u $(n \times 1)$
2. Repeat until convergence
   a) $v = f_\lambda(X' u)$
   b) $u = X v / \text{norm}(X v)$
3. $v = v / \text{norm}(v)$
4. $t = X v$

Posterior thresholding after Nipals

1. Set u $(n \times 1)$
2. Repeat until convergence
   a) $v = X' u$
   b) $u = X v / \text{norm}(X v)$
3. $v = f_\lambda(v) / \text{norm}(f_\lambda(v))$
4. $t = X v$

Different results

4. sPCA-rSVD-soft vs. SPCA

(*)

This section provides some remarks on two sparse PCA methods: our sPCA-rSVD-soft and SPCA of Zou et al. [20]. Both approaches relate PCA to regression problems, and then employ a lasso (L_1) penalty to produce sparsity, as well as some iterative algorithm for computation.

Despite these similarities, there are major differences between the two approaches. First of all, they solve different optimization problems. As we discussed in Section 2.3, to get the first loading vector, sPCA-rSVD solves

$$\min_{\tilde{\mathbf{v}}} \{-2\|\mathbf{X}\tilde{\mathbf{v}}\| + \|\tilde{\mathbf{v}}\|^2 + \lambda|\tilde{\mathbf{v}}|_1\},$$

while the same argument yields that SPCA solves

$$\min_{\tilde{\mathbf{v}}}\{-2\|\mathbf{X}^T\mathbf{X}\tilde{\mathbf{v}}\| + \|\mathbf{X}\tilde{\mathbf{v}}\|^2 + \lambda\|\tilde{\mathbf{v}}\|^2 + \lambda_1|\tilde{\mathbf{v}}|_1\}.$$

The objective functions of the two optimization problems are different.

(*) Zou, H., Hastie, T., Tibshirani, R., 2006. Sparse Principal Component Analysis. Journal of Computational and Graphical Statistics 15, 265–286.

Elastic net: Ridge + Lasso

See Camacho et .
All sparse PCA models are wrong, but some are useful.
Parts I-II-III Chemolab 2020, 2021, 2025 for extensive review and improvments of this model

Several PCs $\Rightarrow$ Deflation $X - \hat{X}$

Shen & Huang 2008

Regression of the X'-columns on v

- $\hat{X}' = v\,\hat{\alpha} = v\,(v'\,v)^{-1}\,v'\,X'$

  $\Rightarrow \hat{X} = t\,v'$

After deflations, in general:

- $t_1, t_2, \ldots t_A$ non orthogonals
- $v_1, v_2, \ldots, v_A$ non orthogonals
- X-residuals based on $\Sigma\, t_a v_a'$

Used in mixOmics::spca

Regression of the X-columns on t

- $\hat{X} = t\,\hat{\beta} = t\,\underbrace{(t'\,t)^{-1}\,t'\,X}_{\neq \text{ sparse } v'}$ (in PLSR/PCA, $\hat{\beta} = v'$)

After deflations

- $t_1, t_2, \ldots t_A$ orthogonals
- In general: $v_1, v_2, \ldots, v_A$ non orthogonals

Not the same results!

A simulated example


```
julia> V
7×3 Matrix{Float64}:
 0.0       0.0       0.292589
 0.161406  0.313671  0.0
 0.422263  0.949532  0.0
 0.0       0.0       0.0
 0.245371  0.0       0.0
 0.857575  0.0       0.349319
 0.0       0.0       0.890151
```

Matrix rank = 3

$$X = U\ D\ V' + E$$

```
julia> D
3×3 Matrix{Float64}:
 11.7542  0.0      0.0
  0.0     6.70285  0.0
  0.0     0.0      2.11381
```

```
julia> X
100×7 Matrix{Float64}:
  0.181285   -0.20734    -0.408499   0.179915   0.094679   -0.147017   0.277054
  0.274594    0.555308    0.882288   0.264666   0.233231    0.0656052  0.0945824
  0.195889    0.337897    0.491918   0.34014    0.740921    1.25783    0.181756
 -0.0305505   0.0884222   0.597367   0.412415   0.178265    0.446075   0.0614099
  0.3828      0.155404   -0.452334   0.414874   0.0305074  -0.830222   0.38701
  0.196888    0.0373859   0.273575   0.254252   0.463714    1.30288    0.661211
  0.257969    0.0736727  -0.0882495  0.0591647  0.233551   -0.142516   0.0948209
  0.4452     -0.494221   -1.73284    0.249954  -0.181028   -0.9741     0.418076
  ⋮                                                         ⋮
  0.382795    0.746685    1.2001     0.120311   0.210753    0.59023    0.49458
  0.21954     0.457912    0.909637   0.467529   0.888031    2.24521    0.0462937
  0.153923    0.682237    0.791914   0.36691    0.669263    0.76458    0.748261
  0.100549    0.936789    1.29377    0.486492   0.891741    1.91321    0.0940264
  0.487144    0.147174   -0.925512   0.0173717  0.0808379  -0.104117   0.615556
  0.171246    0.411152    0.0373041  0.16536   -0.195522   -1.29837    0.365166
  0.304981    0.0165791  -0.631723   0.209357   0.408155    0.740491   0.0423148
  0.184323    0.303364    0.0671519  0.0695118  0.138994    0.424231   0.644166
```

Truth

```
julia> V
7×3 Matrix{Float64}:
 0.0       0.0       0.292589
 0.161406  0.313671  0.0
 0.422263  0.949532  0.0
 0.0       0.0       0.0
 0.245371  0.0       0.0
 0.857575  0.0       0.349319
 0.0       0.0       0.890151
```

PCA

```
julia> fitm.V
7×3 Matrix{Float64}:
  0.00438894   0.00649242   0.296274
 -0.204425     0.26658      0.2128
 -0.576727     0.744196    -0.0518993
 -0.0135646   -0.00325929  -0.0621416
 -0.217537    -0.154472    -0.150922
 -0.760308    -0.591813     0.0259066
 -0.0018665   -0.0308385    0.914841
```

sPCA-SVD nvar = 4, 2, 3

```
julia> fitm.V
7×3 Matrix{Float64}:
 -0.0       0.0        0.159027
  0.195288  0.0        0.0629095
  0.576263  0.827631  -0.0
  0.0       0.0       -0.0
  0.209112  -0.0      -0.0
  0.765543  -0.561273 -0.0
  0.0       -0.0       0.985268
```

Extension: Simple pipeline of sPCR

1. sPCA-rSVD($\mathbf{X}$) $\Rightarrow$ scores $\mathbf{T}$

2. Regression of $\mathbf{y}$ on $\mathbf{T}$

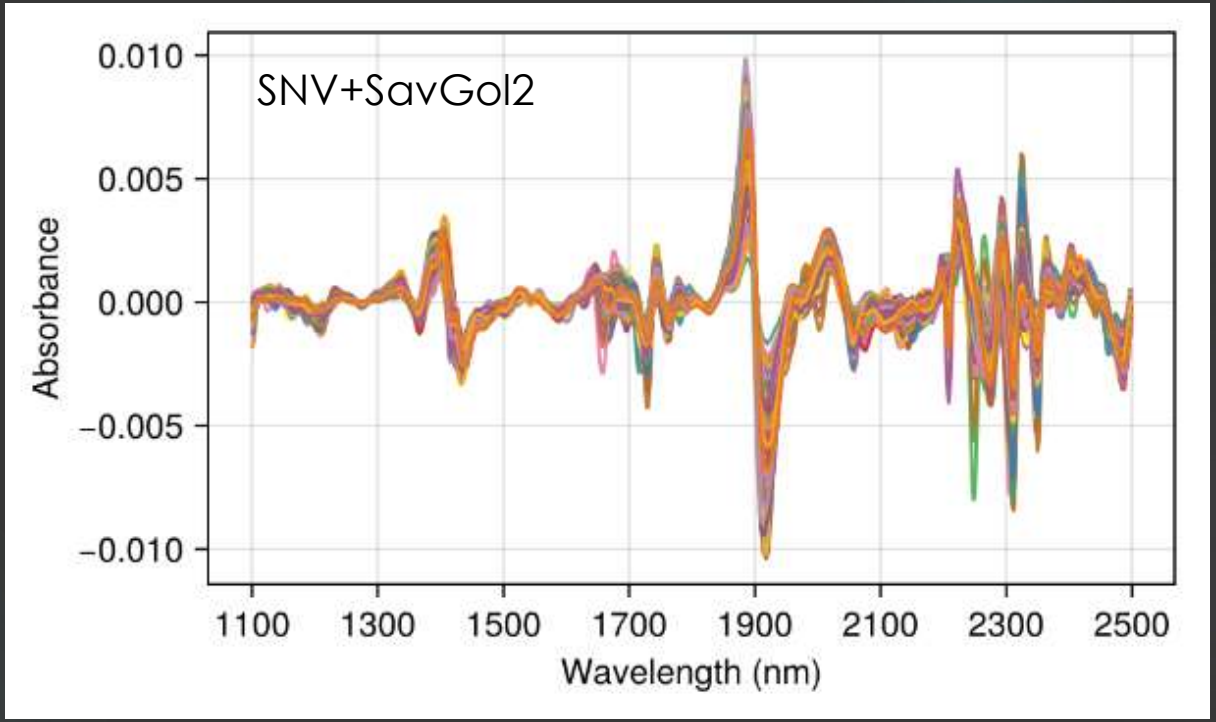
$\Rightarrow$ sPCA-rSVD-MLR

Specific parameters to define: threshold function
deflation mode

A real example

Forage data
(mixed plants,
dried & grounded)

```
julia> n, p = size(X)
(485, 700)
```

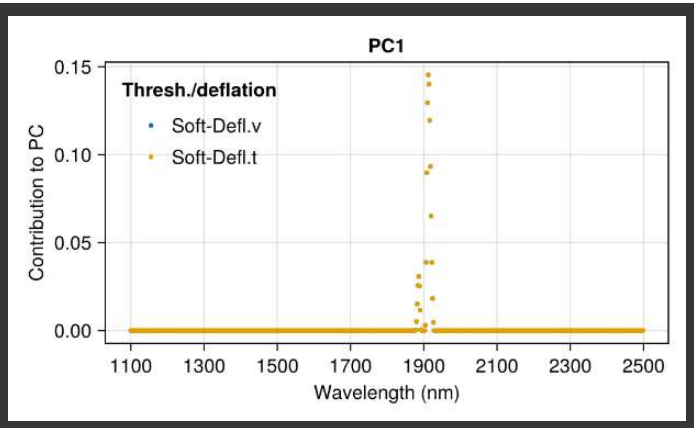


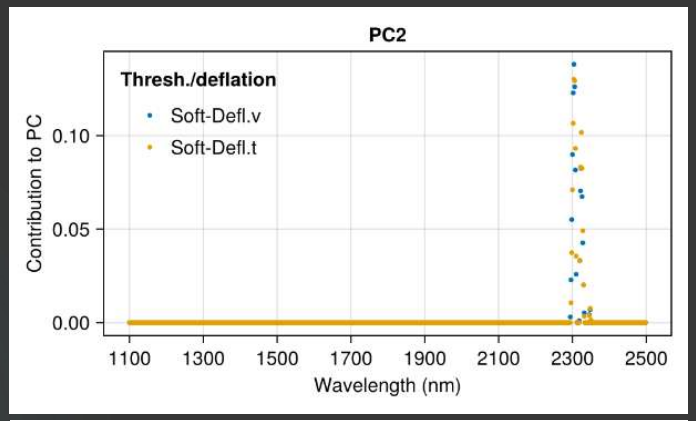
⇒ Prediction of
%Fibers

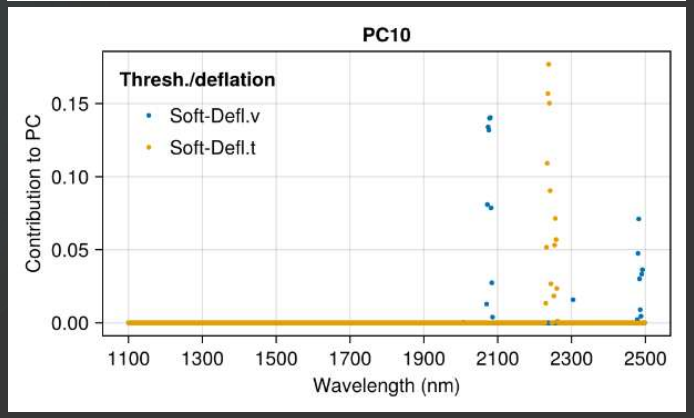
```
julia> summ(y).res
1×7 DataFrame
 Row │ variable  mean     std      min      max      n      nmissing
     │ Symbol    Float64  Float64  Float64  Float64  Int64  Int64
─────┼──────────────────────────────────────────────────────────────
   1 │ x1        50.736   13.204   18.425   85.7235    485         0
```

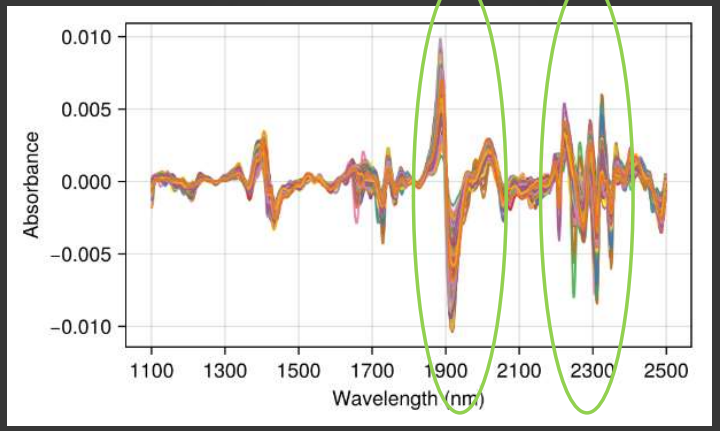
sPCA-rSVD

$\mathrm{nvar} = 20$



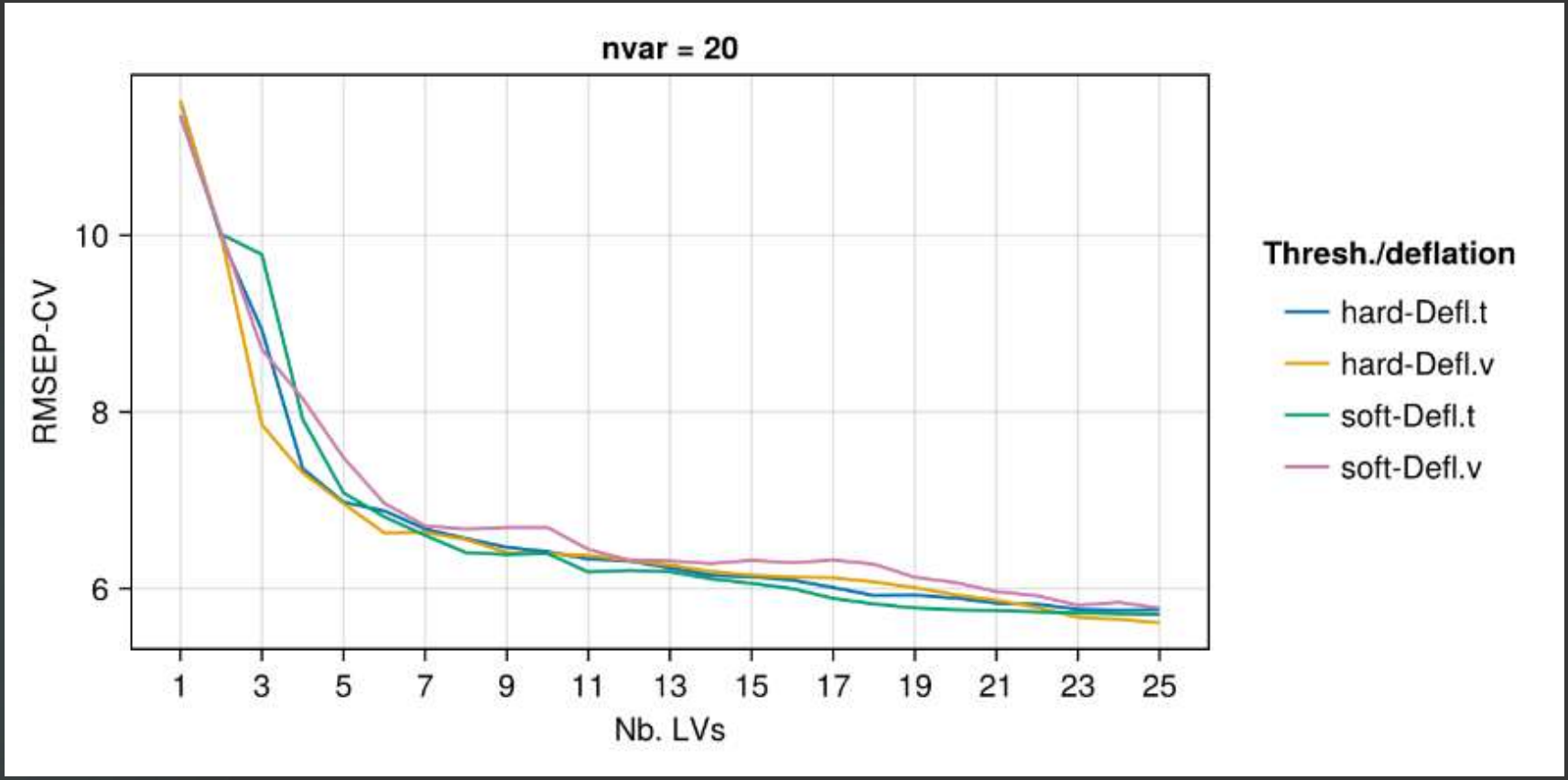






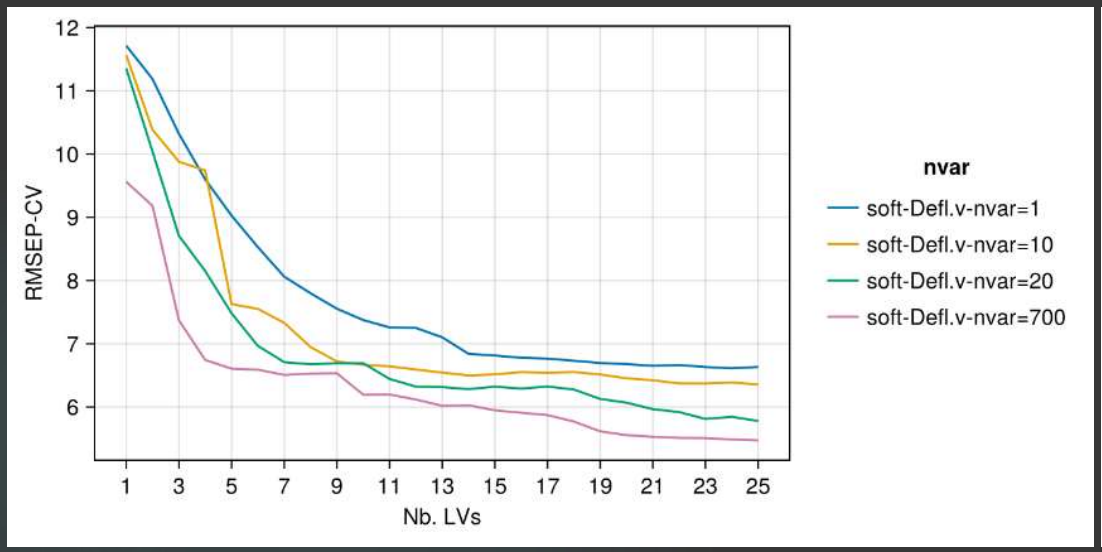
sPCR : sPCA-rSVD-MLR

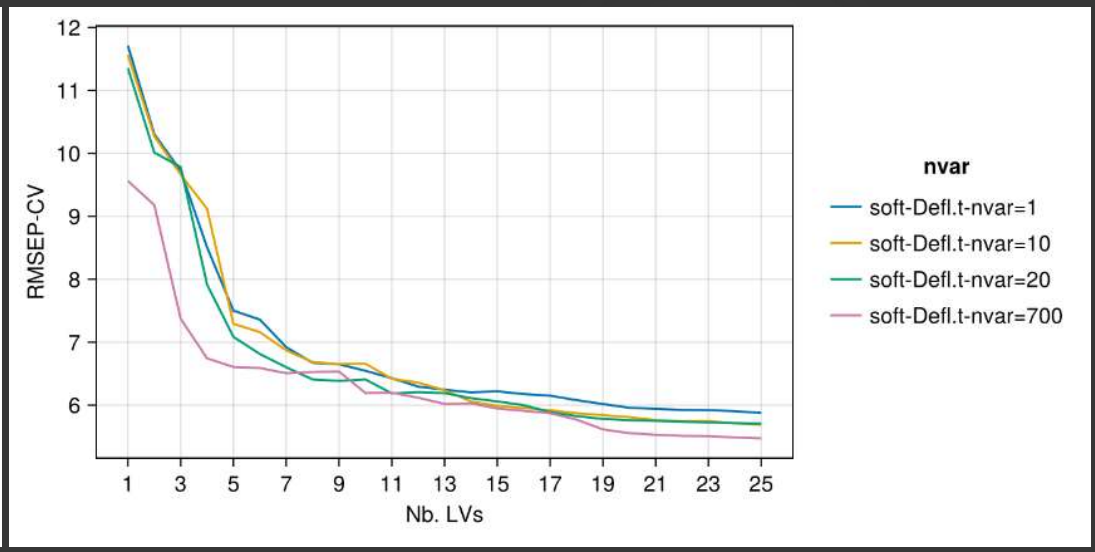
K-Fold CV
(K = 3, repl = 5)



sPCR : sPCA-rSVD-MLR

K-Fold CV
(K = 3, repl = 5)

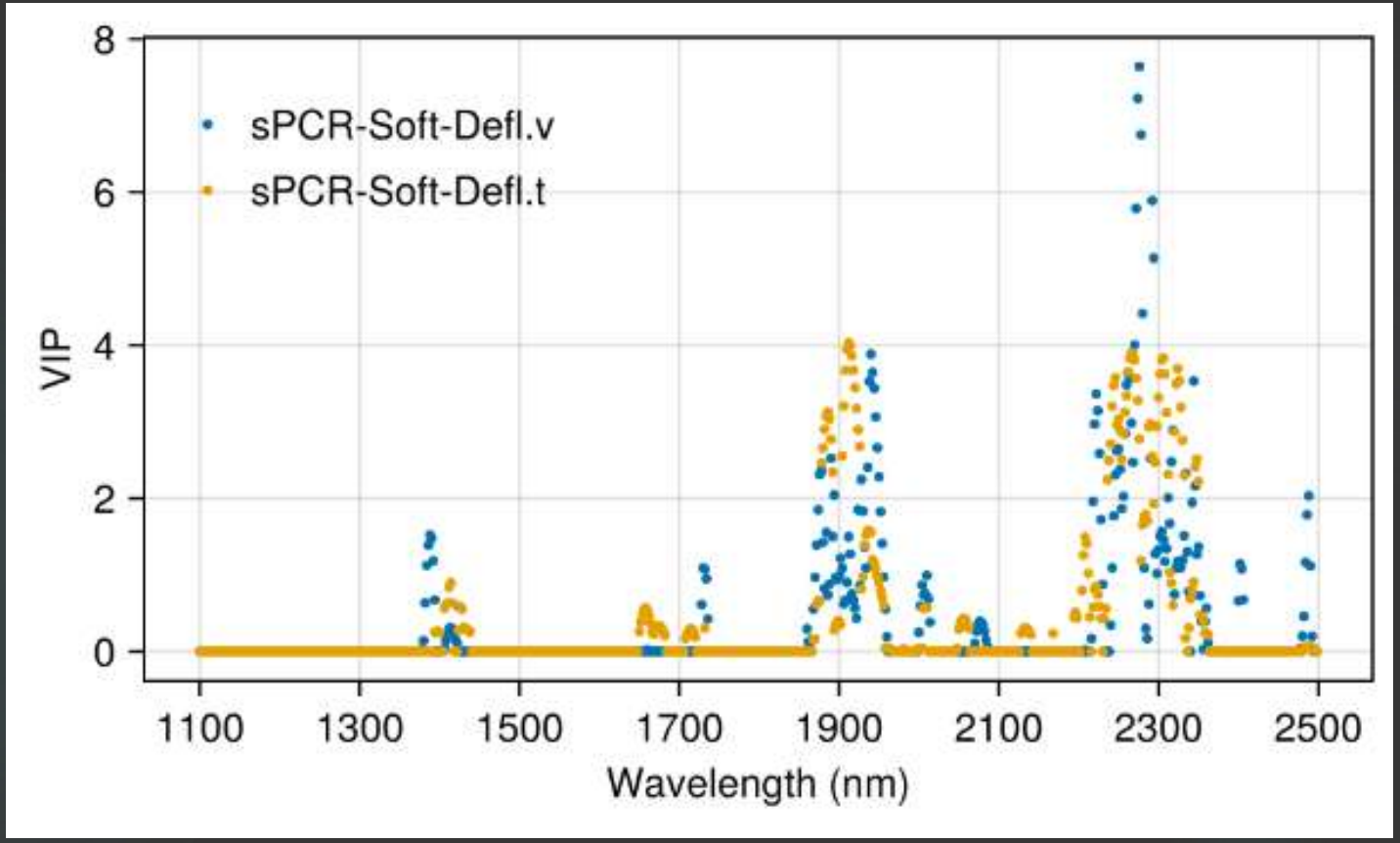




!! Almost same performances from few variables v_j than with all the 700 wavelengths

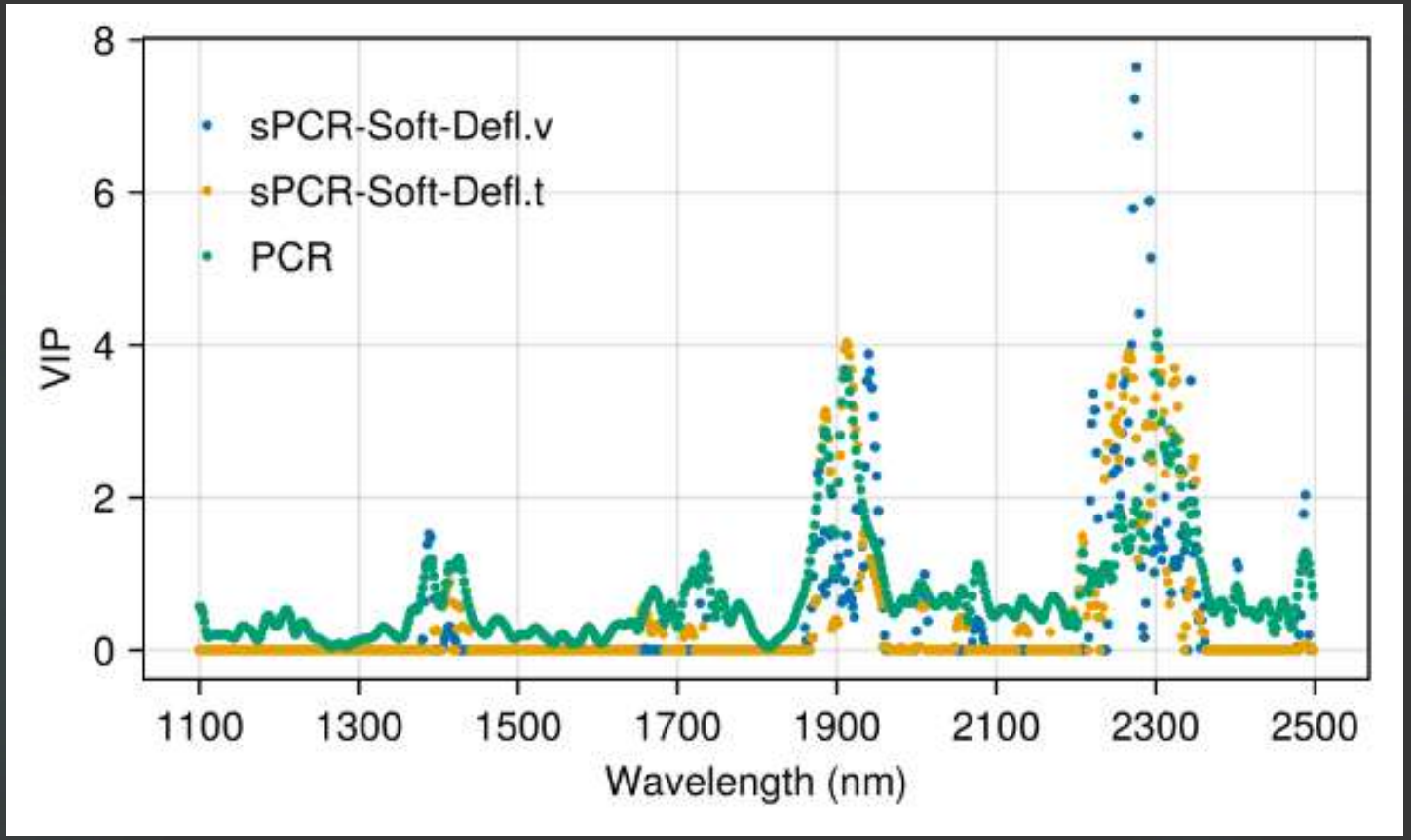
Variable importances for the projections

sPCR nlv = 15 nvar = 20



Variable importances for the projections

sPCR vs. PCR nlv = 15 nvar = 20



Jchemo sparse methods

https://github.com/mlesnoff/Jchemo.jl

```
# sPCA-rSVD

model = spca(nlv = 10, meth = :soft, nvar = 5, defl = :v)
fit!(model, X)

# Sparse PCR  (sPCA-rSVD-MLR)

model = spcr(nlv = 10, meth = :soft, nvar = 5, defl = :t)
fit!(model, X, y)
pred = predict(model, Xnew).pred

# Sparse PLSR          (& SPLSDA)

model = splsr(nlv = 10, meth = :soft, nvar = 5)
fit!(model, X, y)
pred = predict(model, Xnew).pred
```