Introduction to VIPs
(and to some other variable importance methods)

Illustration on Tecator data

matthieu.lesnoff@cirad.fr

ChemHouse 06/03/2023

Tecator data

NIRS data recorded on a Tecator Infratec Food and Feed Analyzer working in the wavelength range 850 - 1050 nm by the Near Infrared Transmission (NIT) principle.

Each sample contains finely chopped pure meat with different moisture, fat and protein contents.

For each meat sample the data consists of a 100 channel spectrum of absorbances and the contents of moisture (water), fat and protein. The absorbance is -log10 of the transmittance measured by the spectrometer.

The three contents, measured in percent, are determined by analytic chemistry.

All description here:
http://lib.stat.cmu.edu/datasets/tecator

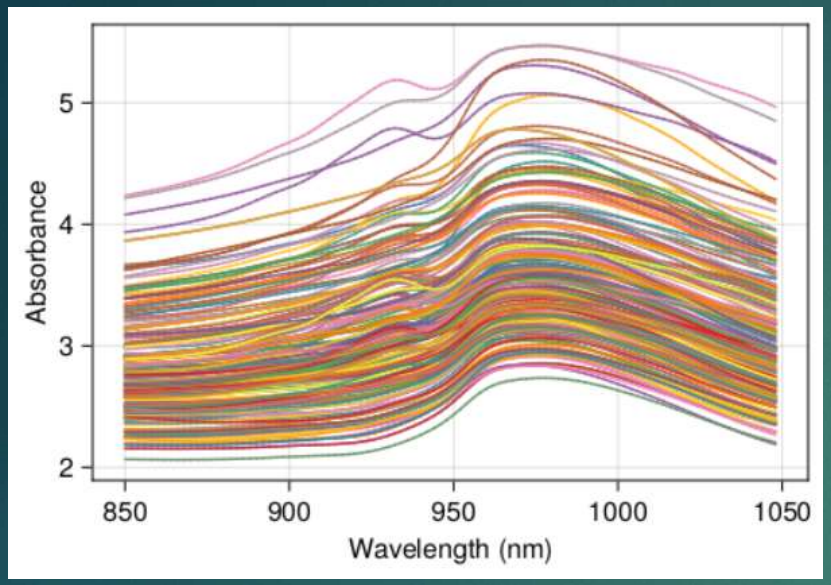
Available in JLD2 format at:
https://github.com/mlesnoff/JchemoData.jl

Warning: The original X-data contains 22 duplicates.
They were removed in the present JLD2 dataset.

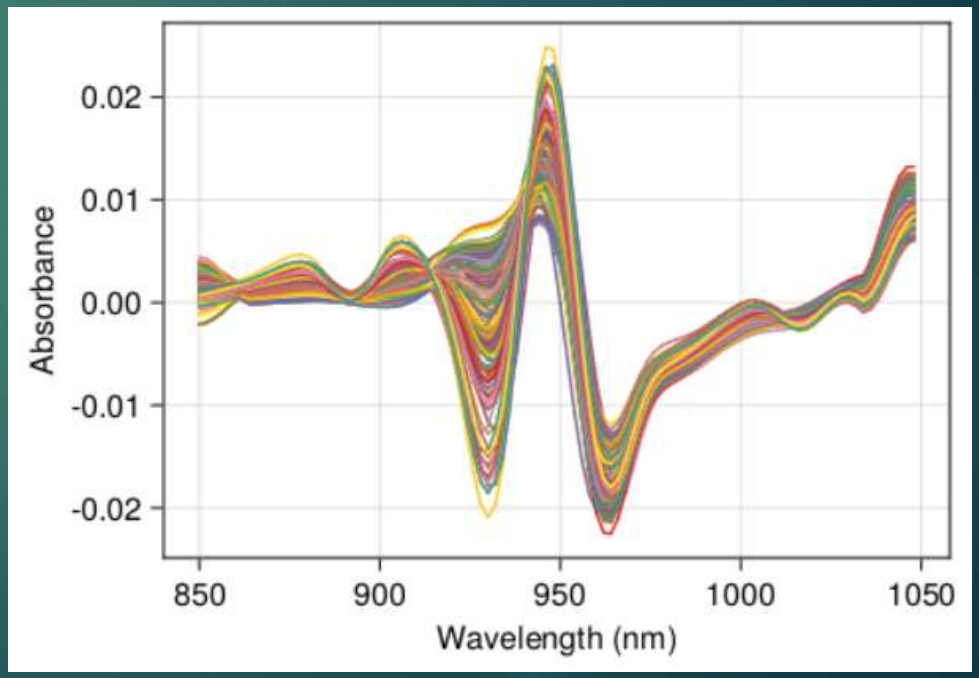
NIR wl = 850:1048 nm (step 2 nm) p = 100

ntot = 193

- ntrain = 120
- ntest = 73



Xp = savgol(snv(X); f = 15, pol = 3, d = 2)

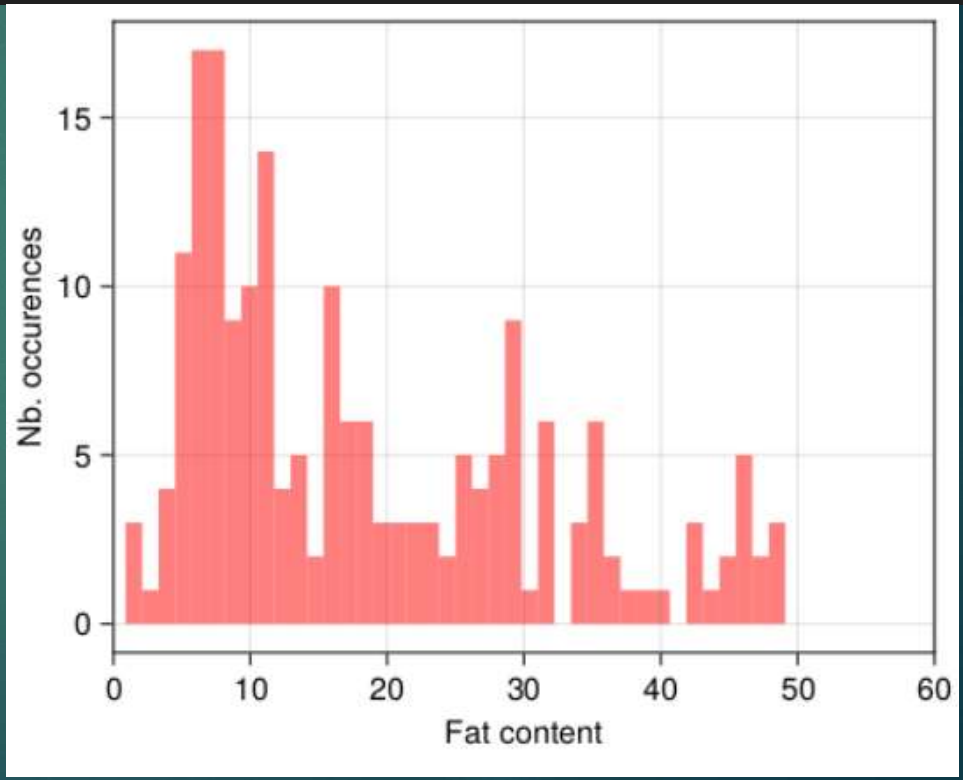


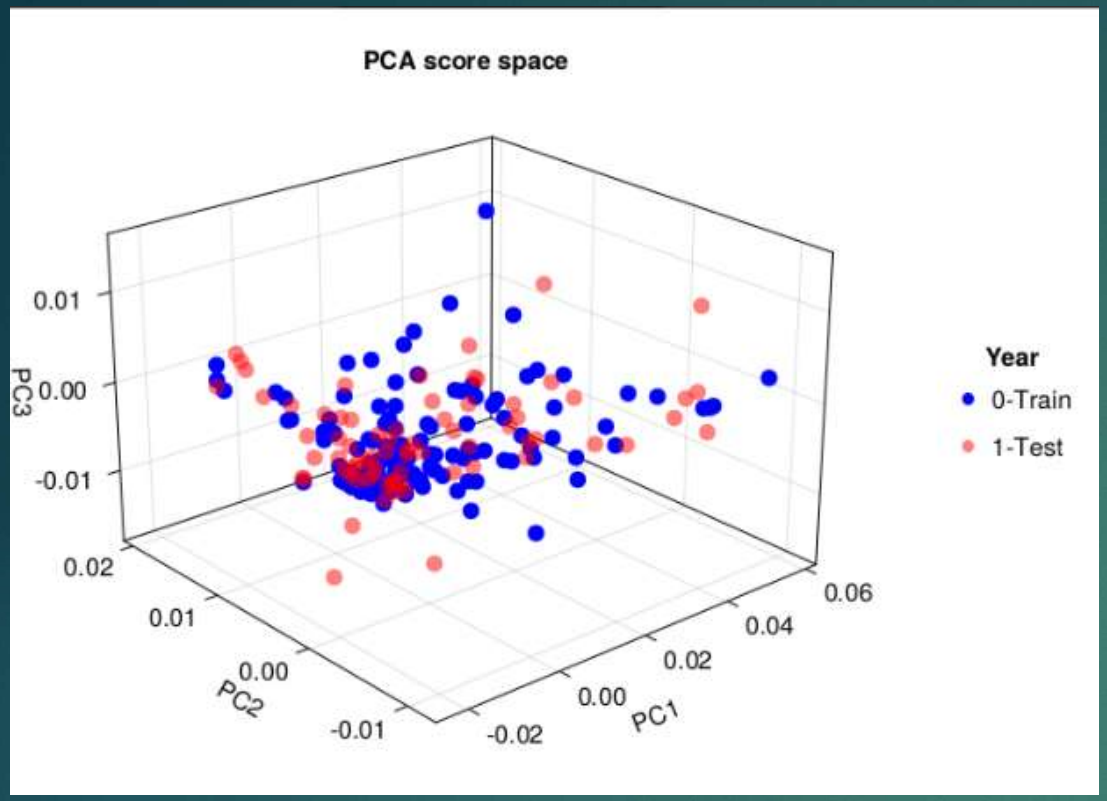
```
julia> Y
193×4 DataFrame
```

Row	water	fat	protein
	Float64	Float64	Float64
1	60.5	22.5	16.7
2	46.0	40.1	13.5
3	71.0	8.4	20.5
4	72.8	5.9	20.7
5	58.3	25.5	15.5
6	44.0	42.7	13.7
⋮	⋮	⋮	⋮
189	55.4	29.2	15.0
190	53.4	31.3	15.3
191	51.6	33.8	13.8
192	50.3	35.5	13.2
193	44.9	42.5	12.0

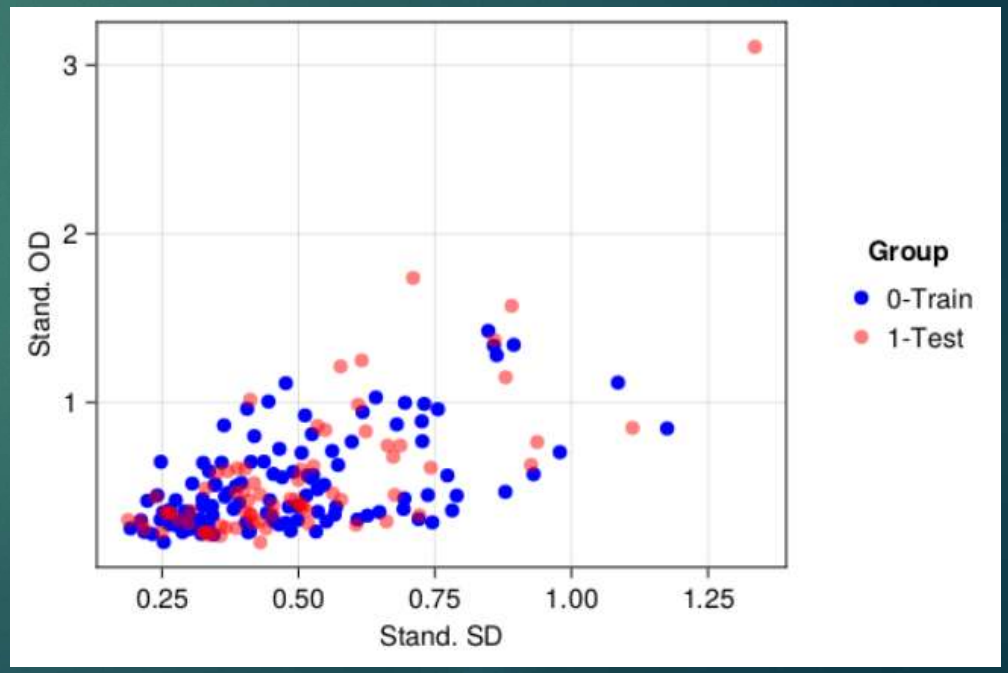
```
julia> summm(Y.fat).res
1×7 DataFrame
```

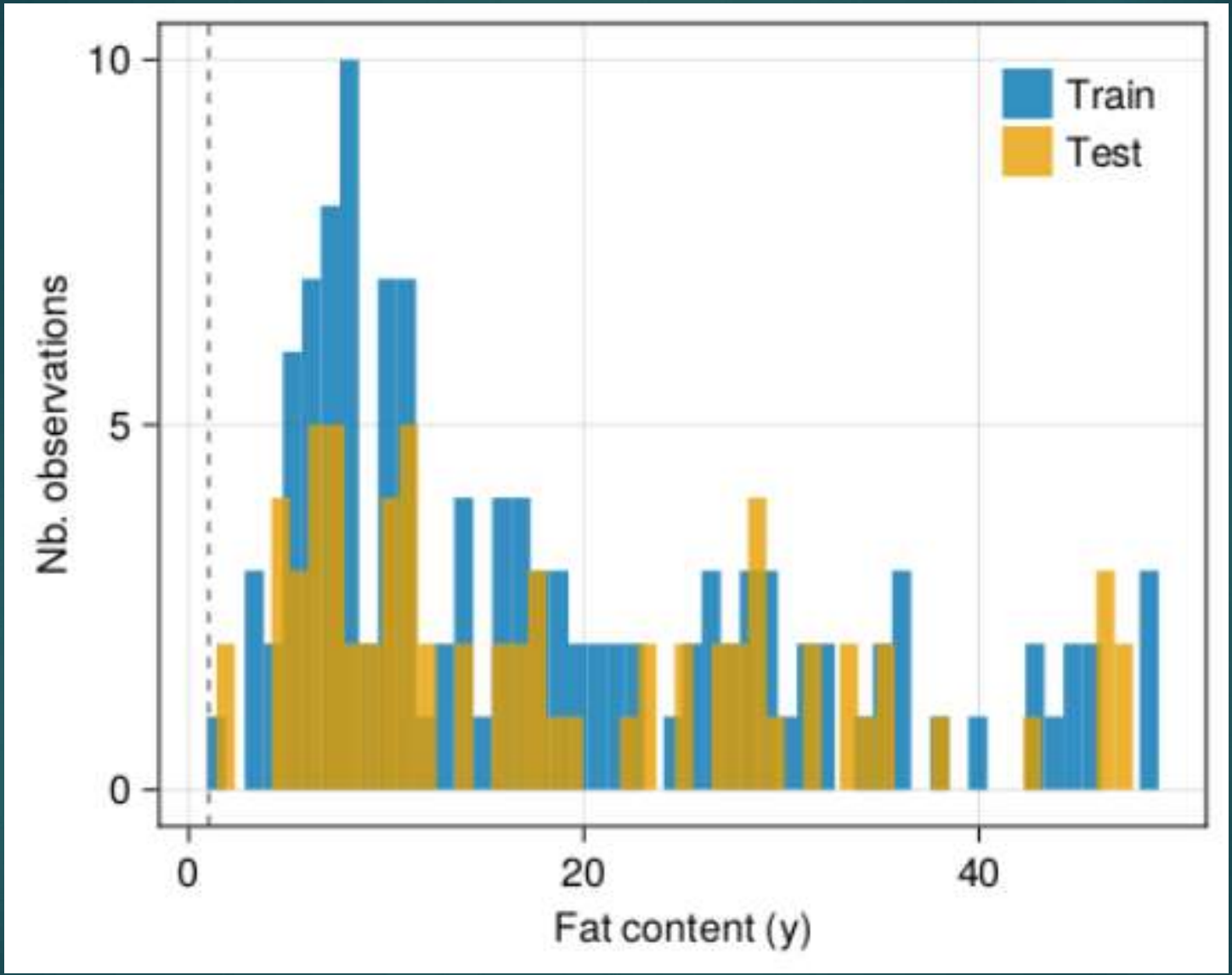
Row	variable	mean	std	min	max	n	nmissing
	Symbol	Float64	Float64	Float64	Float64	Int64	Int64
1	x1	18.512	12.693	0.9	49.1	193	0




PCA score space
PC3
0.01
0.00
-0.01
0.02
0.01
0.00
-0.01
PC2
-0.02
0.00
0.02
0.04
0.06
PC1
Year
0-Train
1-Test

PCA 10 PCs


Stand. OD
3
2
1
0.25
0.50
0.75
1.00
1.25
Stand. SD
Group
0-Train
1-Test


Train
Test
Nb. observations
10
5
0
0
20
40
Fat content (y)

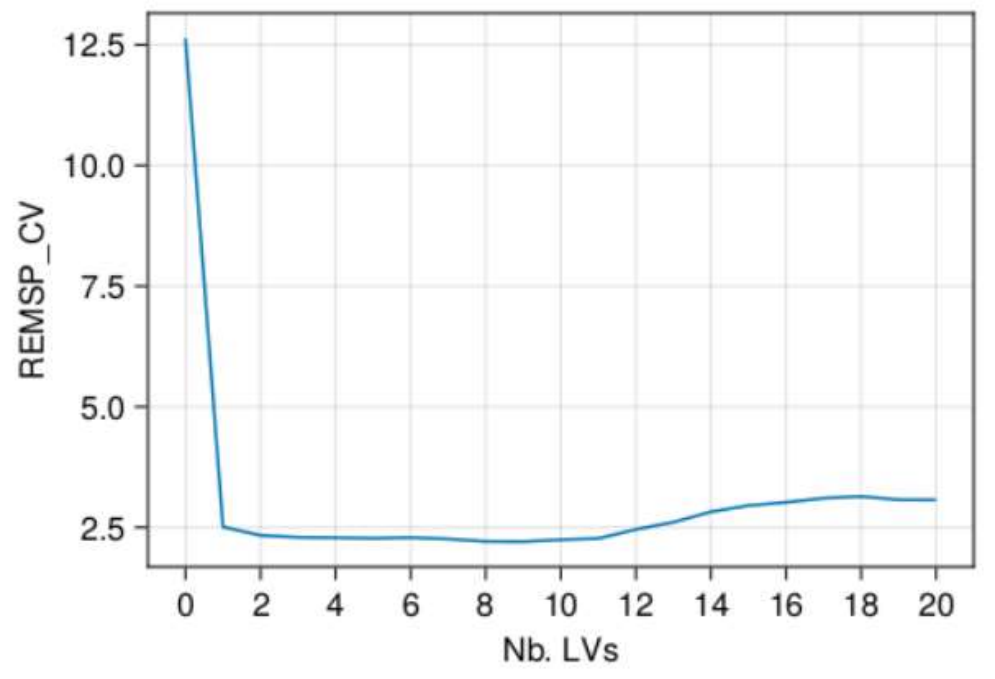
Tuning models and predictions

Training n = 120

- Tuning by replicated K-fold CV

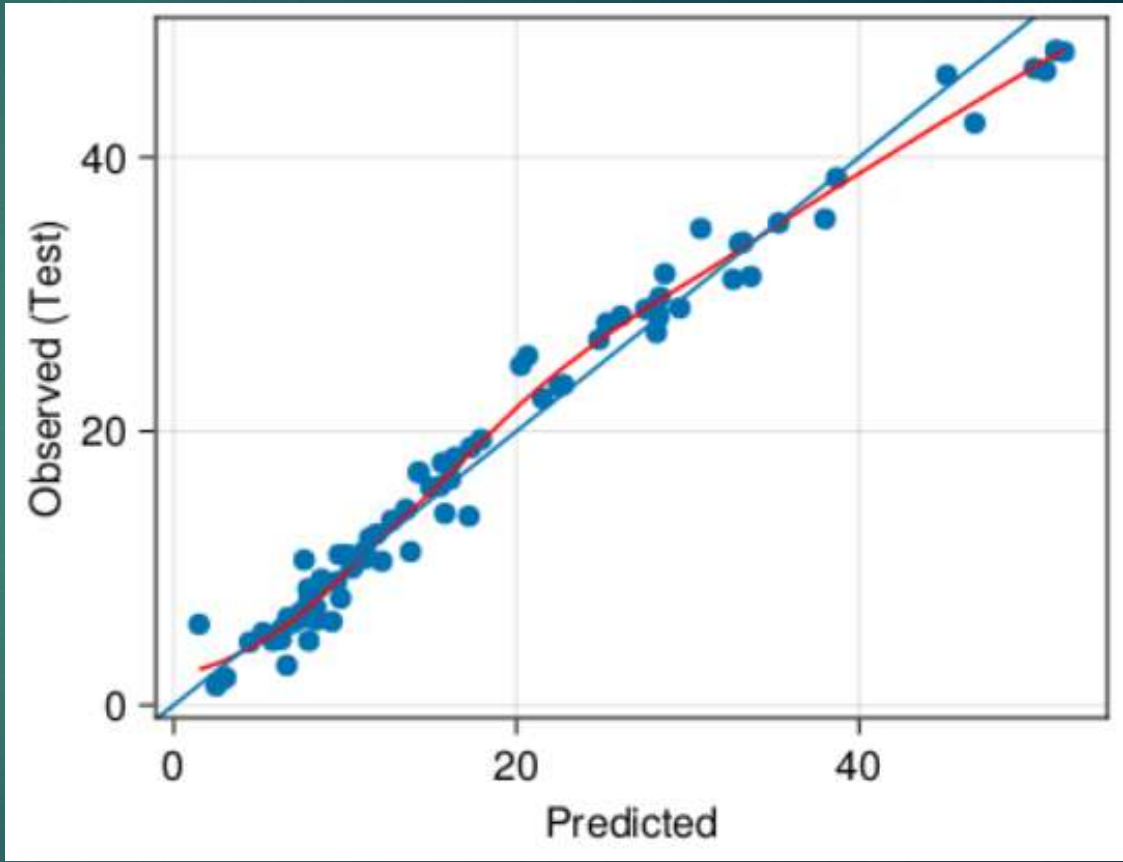
$\Rightarrow$ Prediction on Test n = 73 $\Rightarrow$ RMSEP_Test

PLSR

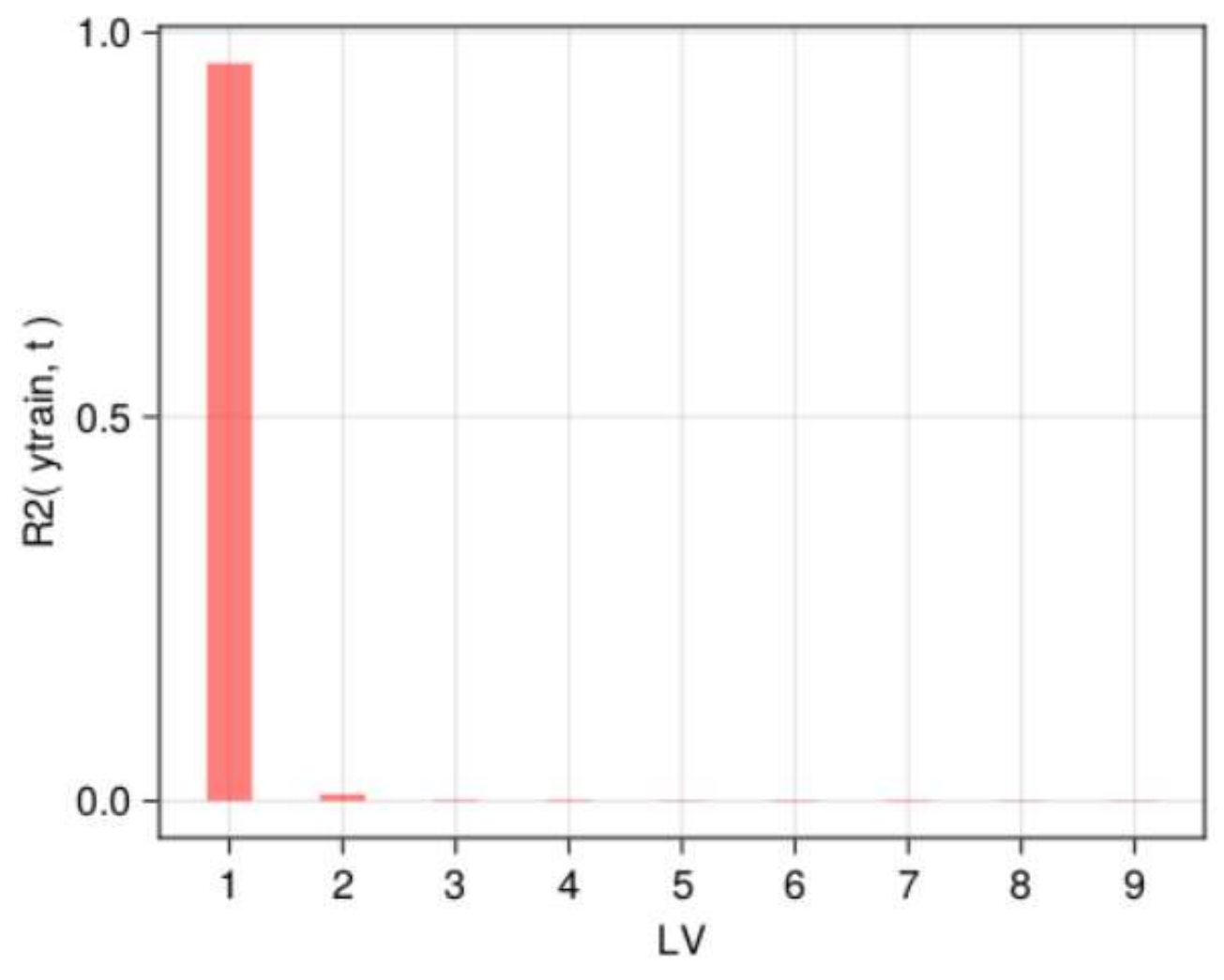


Optimal: nlv = 9

RMSPE_CV = 2.21



RMSPE_Test = 1.94

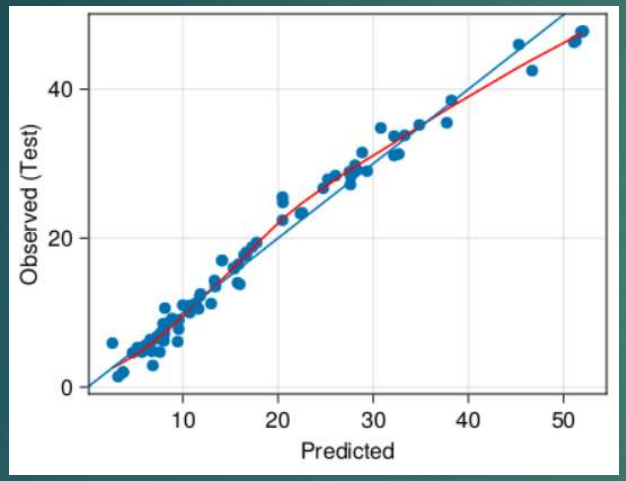


LV1 is highly correlated with **y**

⇒Variables $\mathbf{x}_j$ influent to build LV1 are expected to have a high VIP

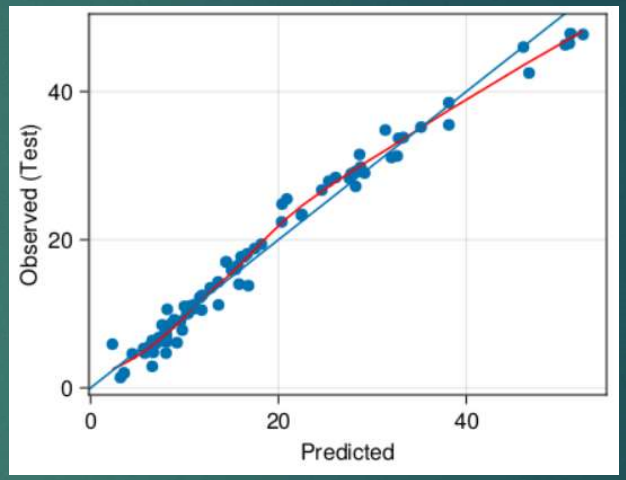
Other models

Ridge regression (RR)



RMSPE_Test = 1.93

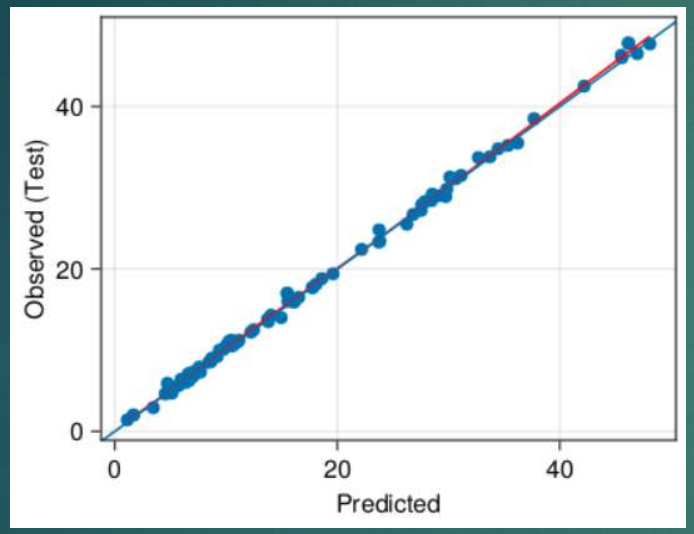
Covsel regression



nvar = 10
RMSPE_Test = 1.88

Kernel regressions Gram matrix $K = \Phi(X)\Phi(X)'$

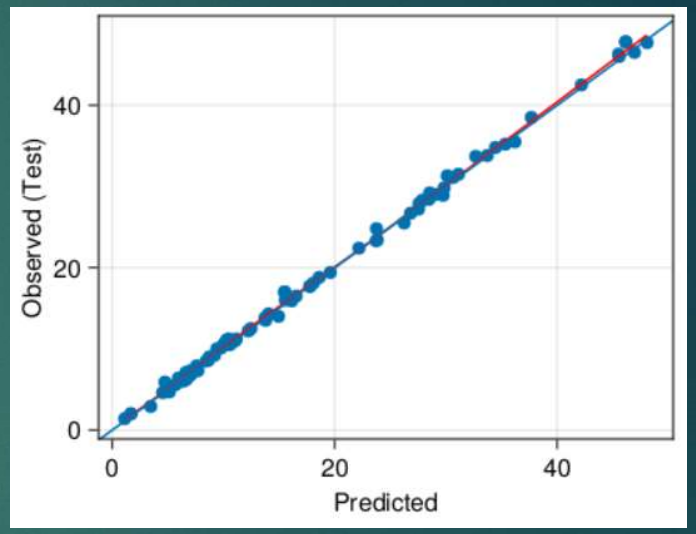
KRR = LS-SVMR



RMSPE_Test = 0.52

```
fm = krr(Xtrain, ytrain; lb = .001,
    gamma = 100) ;
```

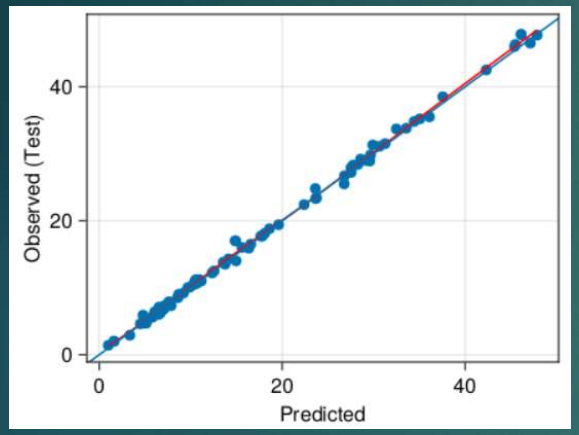
KPLSR



RMSPE_Test = 0.55

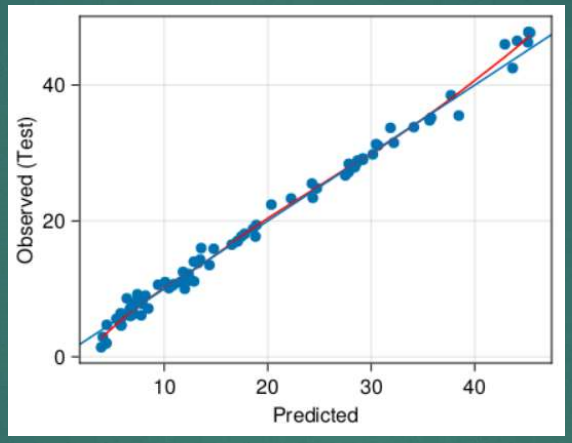
```
fm = kplsr(Xtrain, ytrain; nlv = 14,
    gamma = 100) ;
```

SVMR



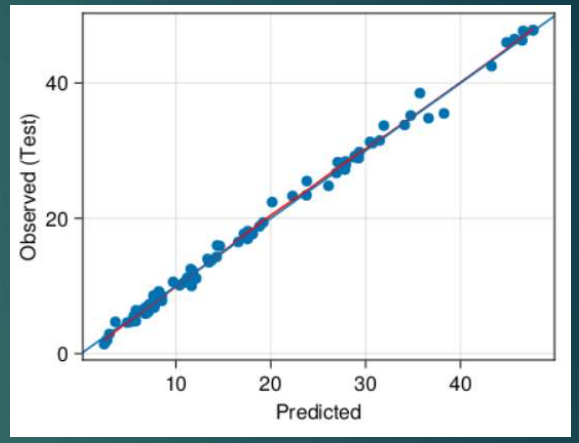
RMSPE_Test = 0.57

RFR



RMSPE_Test = 1.36

XGBOOSTR

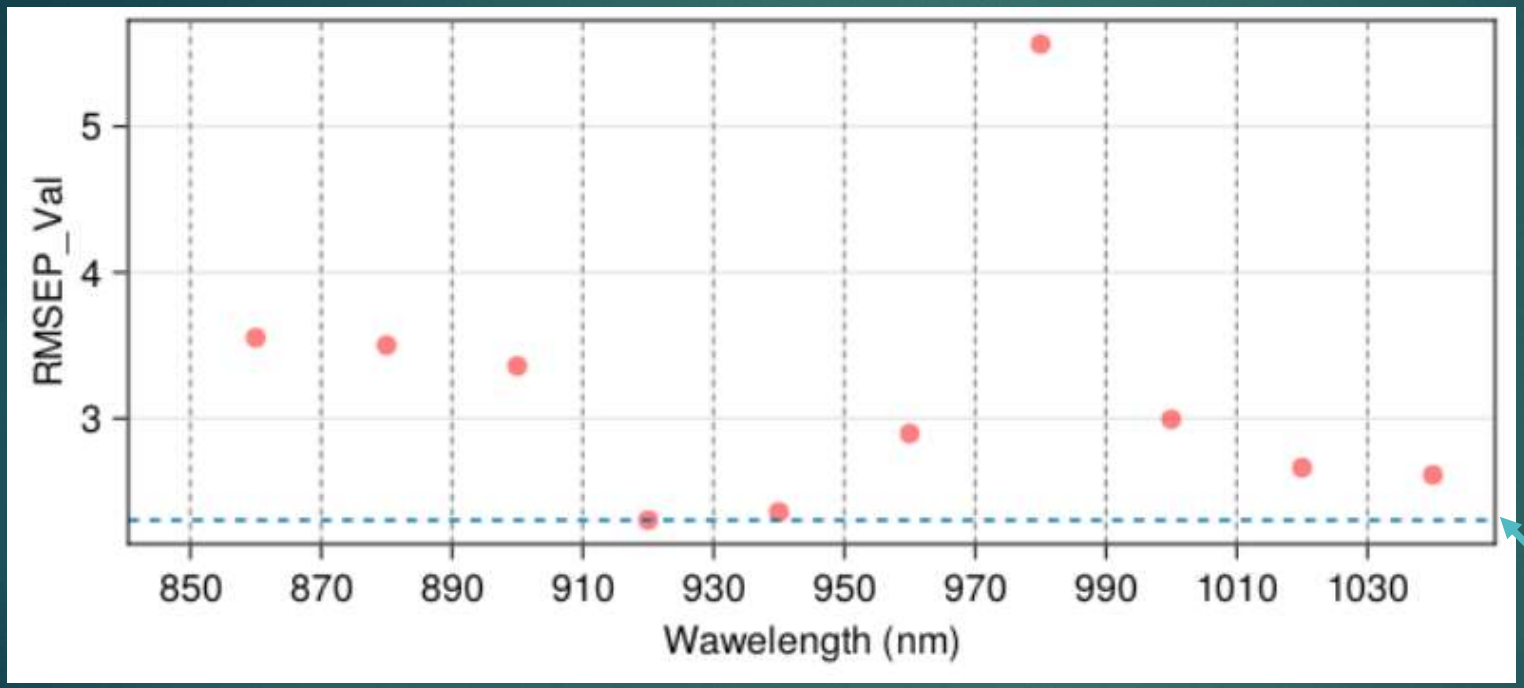


RMSPE_Test = 1.03

Variable importance

Interval methods

iPLSR nint = 10 nlv = 5 rep = 30



1) Xtrain is splitted randomly to Xcal + Xval ⇒ RMSEP_Val
   a) All variables (Ref)
   b) Only 1 interval

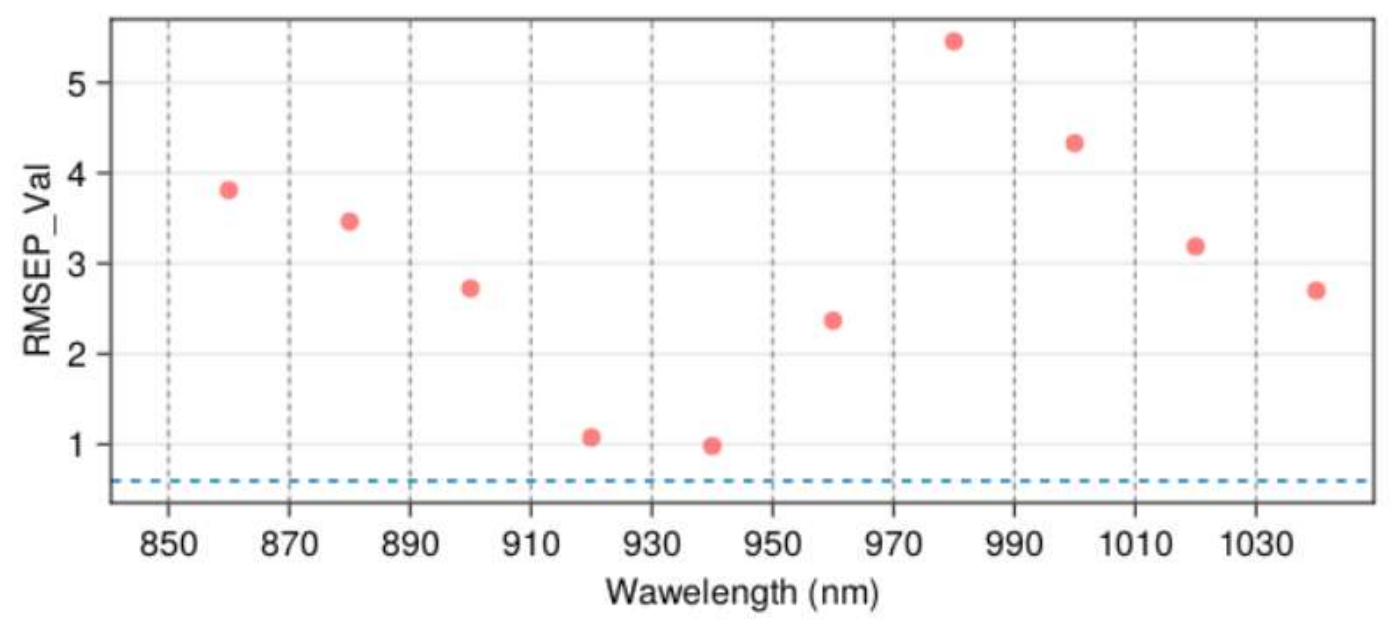
2) This is replicated

Reference (all X-variables)

New tuning (CV) on Train on this interval ⇒ RMSPE_Test = 2.14

The method can use any type of models

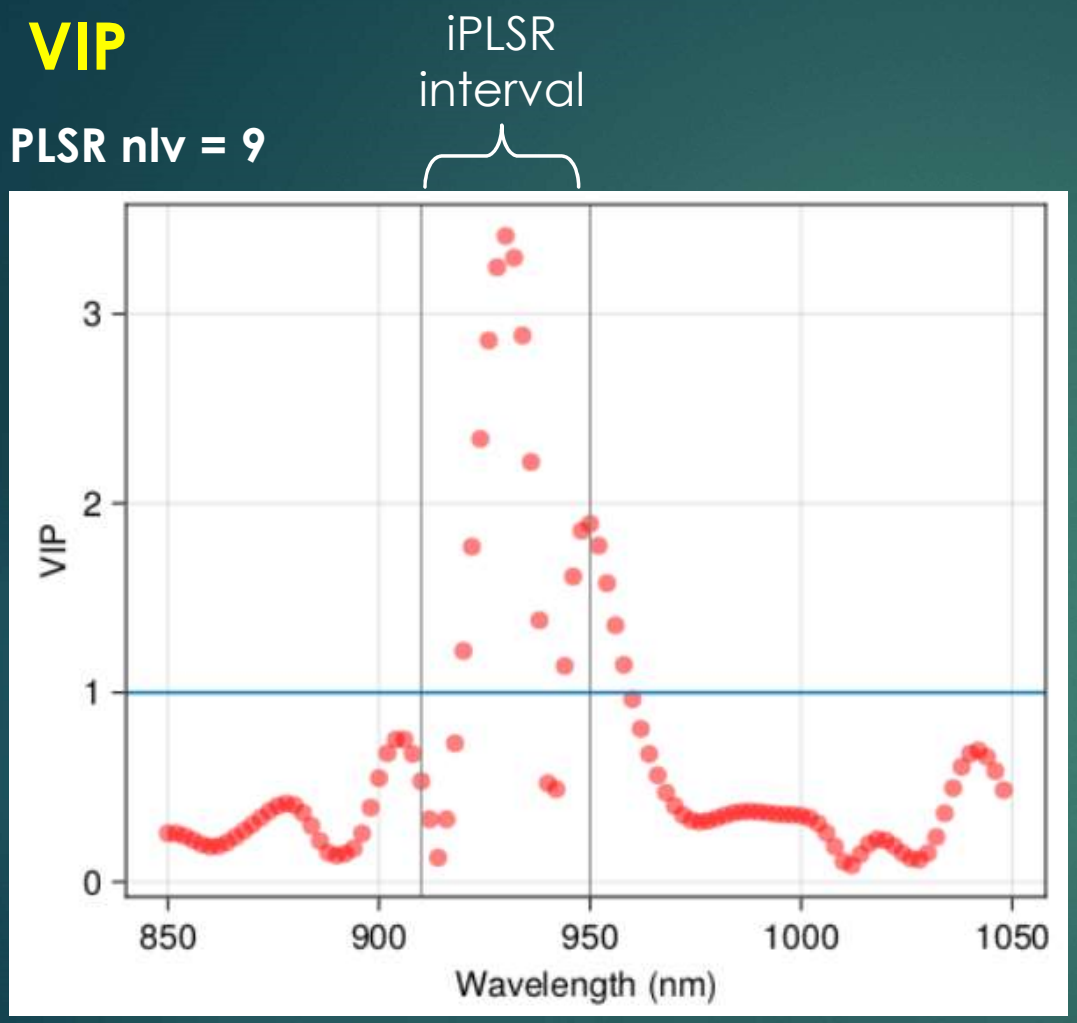
iKRR



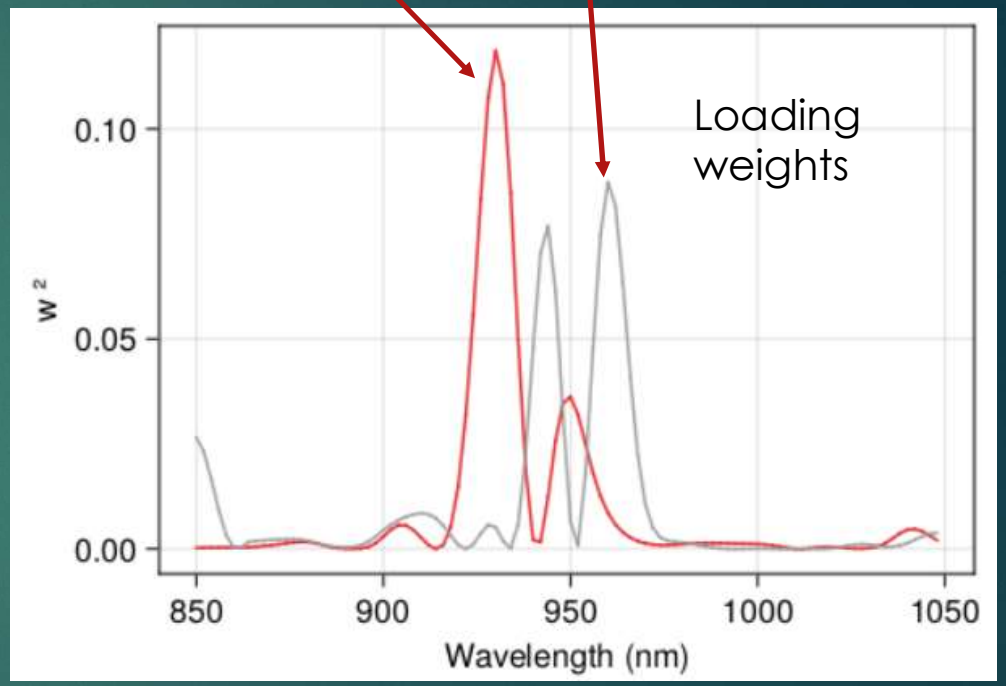
res = isel(Xtrain, ytrain, wl_num; nint = 10,
rep = 30, fun = krr, gamma = 100, lb = .001) ;

VIP

PLSR nlv = 9



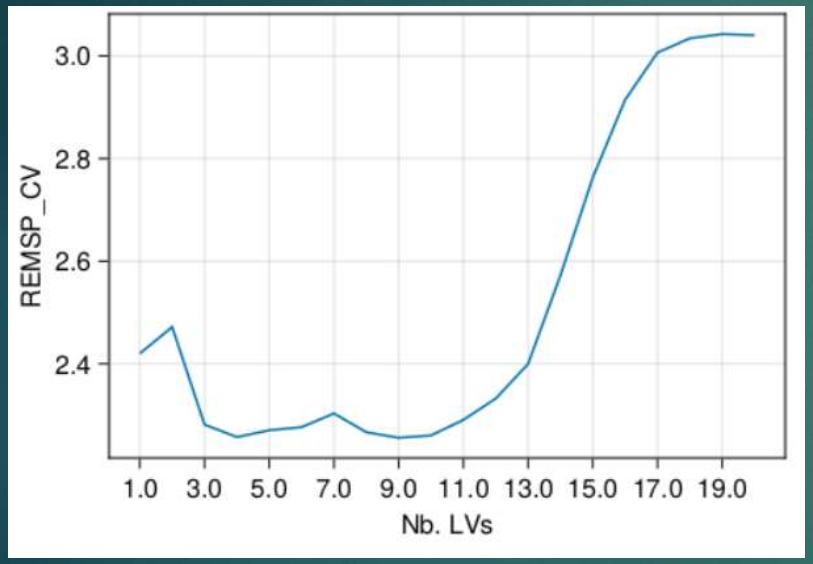
After selecting variables > 1.5
RMSPE_Test = 2.24



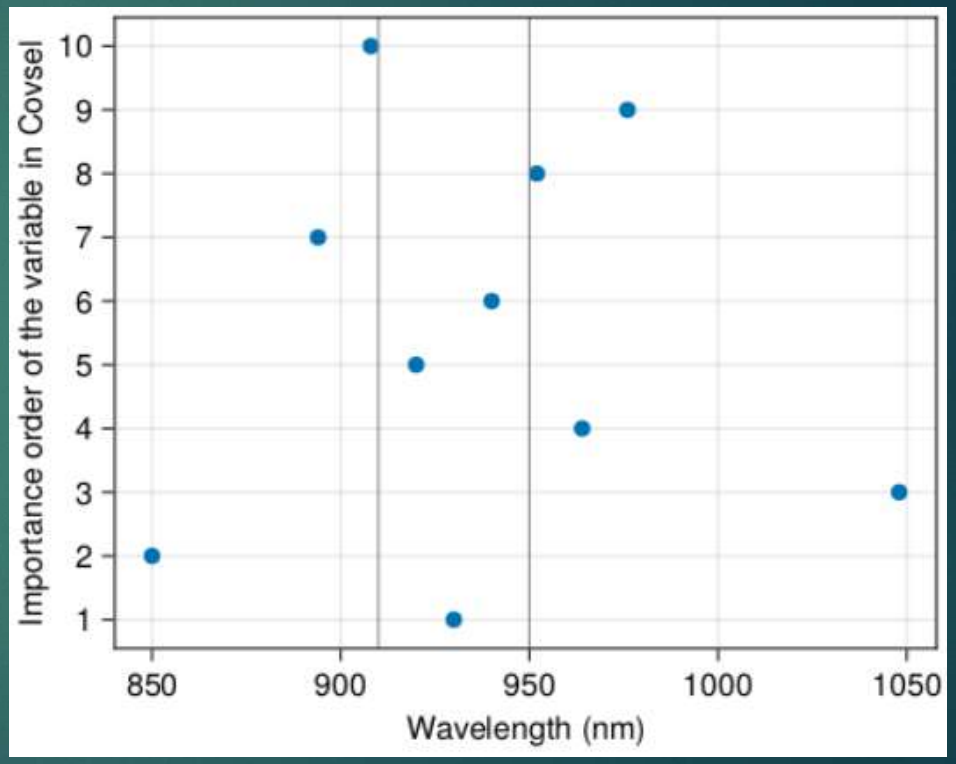
As expected from previous R2s, variables influent to build LV1 have high VIPS

Covsel

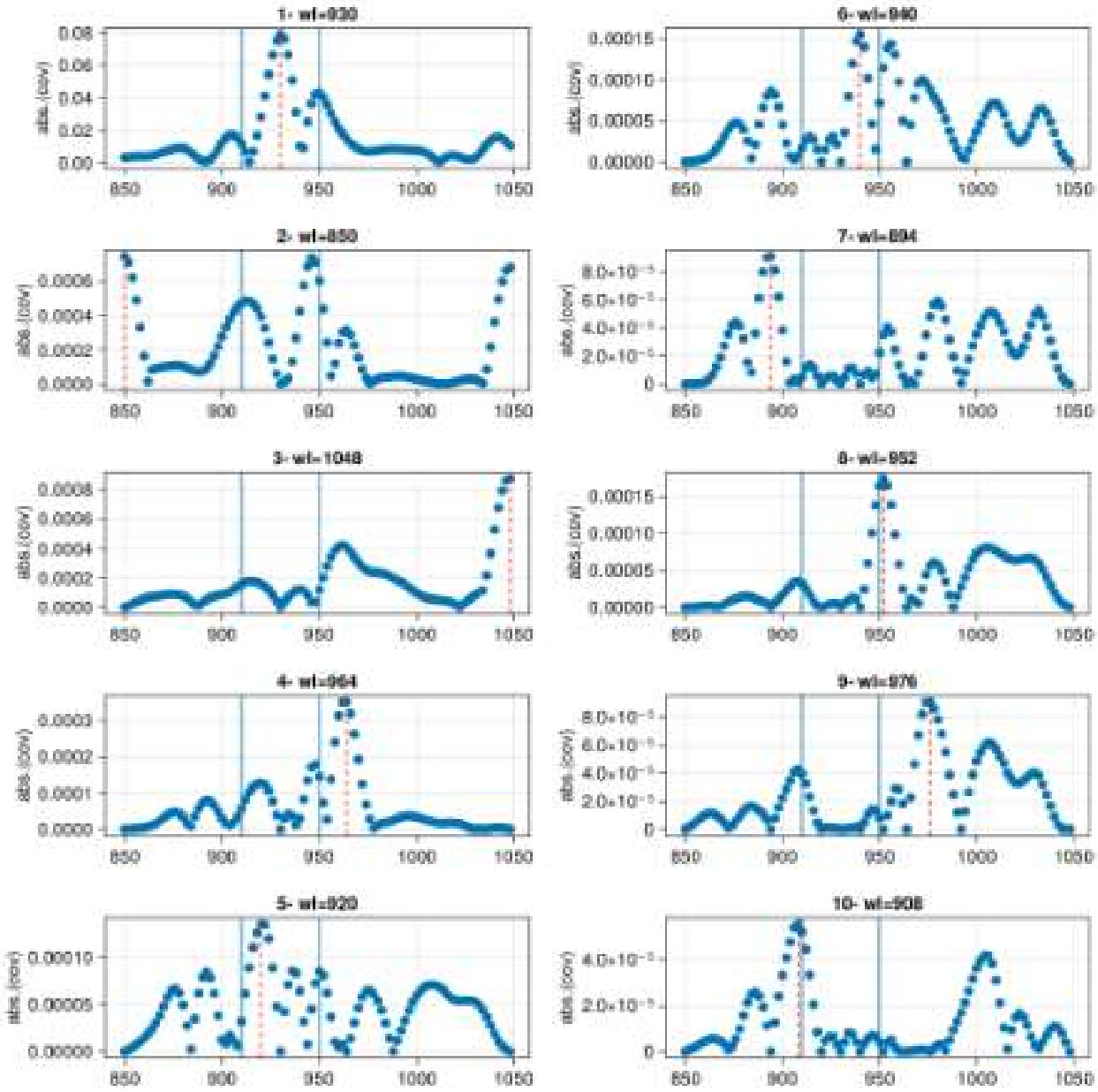
CovselR CV on Train



iPLSR interval



Profiles of partial covariance



Bagging and out-of-box permutations

Bootstrap Train(ntrain)

1) 1 replication

a) Sampling of ntrain samples within Train with replacement
   ⇒ Samples that have been sampled = Xcal
   Samples that have not been sampled (~33% in average) = OOB = Xval

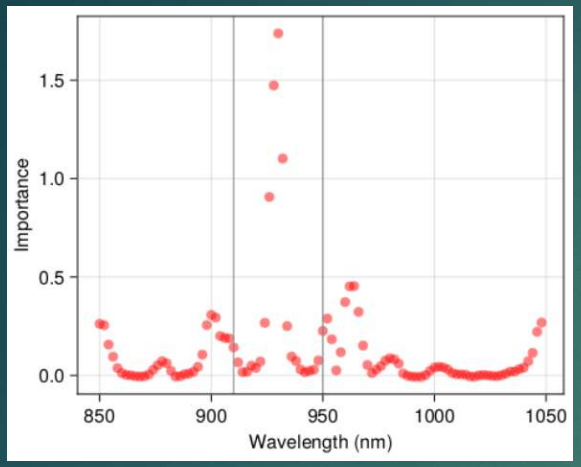
b) The model is fitted on Xcal, and used to predict Xval
   ⇒ Reference error rate Err0

c) Then, for a given column $\mathbf{x}_j$ of Xval, the rows are randomly permuted, and the error rate Err is computed ⇒ $VI(\mathbf{x}_j) = Err - Err0$

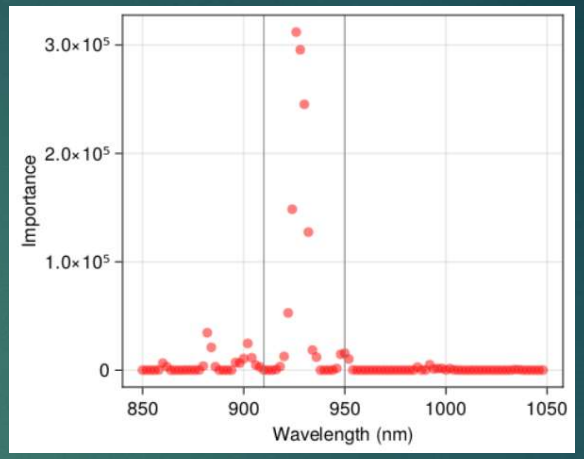
2) The process is replicated
   ⇒ Final VI = Average $VI(\mathbf{x}_j)$

This is the method used in random forests

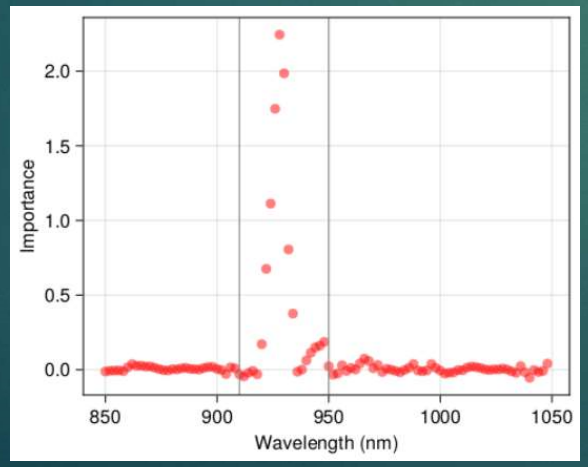
Bagging PLSR



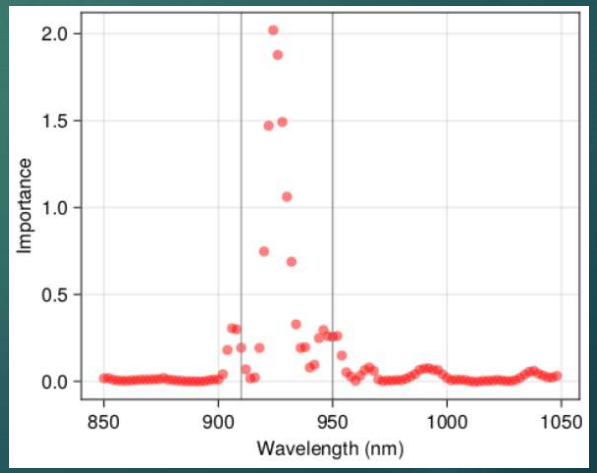
Bagging regression trees = Random forest



Bagging KRR



Bagging KPLSR



Direct permutations

1) 1 replication

a) Xtrain is randomly splitted to Xcal + Xval

b) The model is fitted on Xcal, and used to predict Xval ⇒ Reference error rate Err0

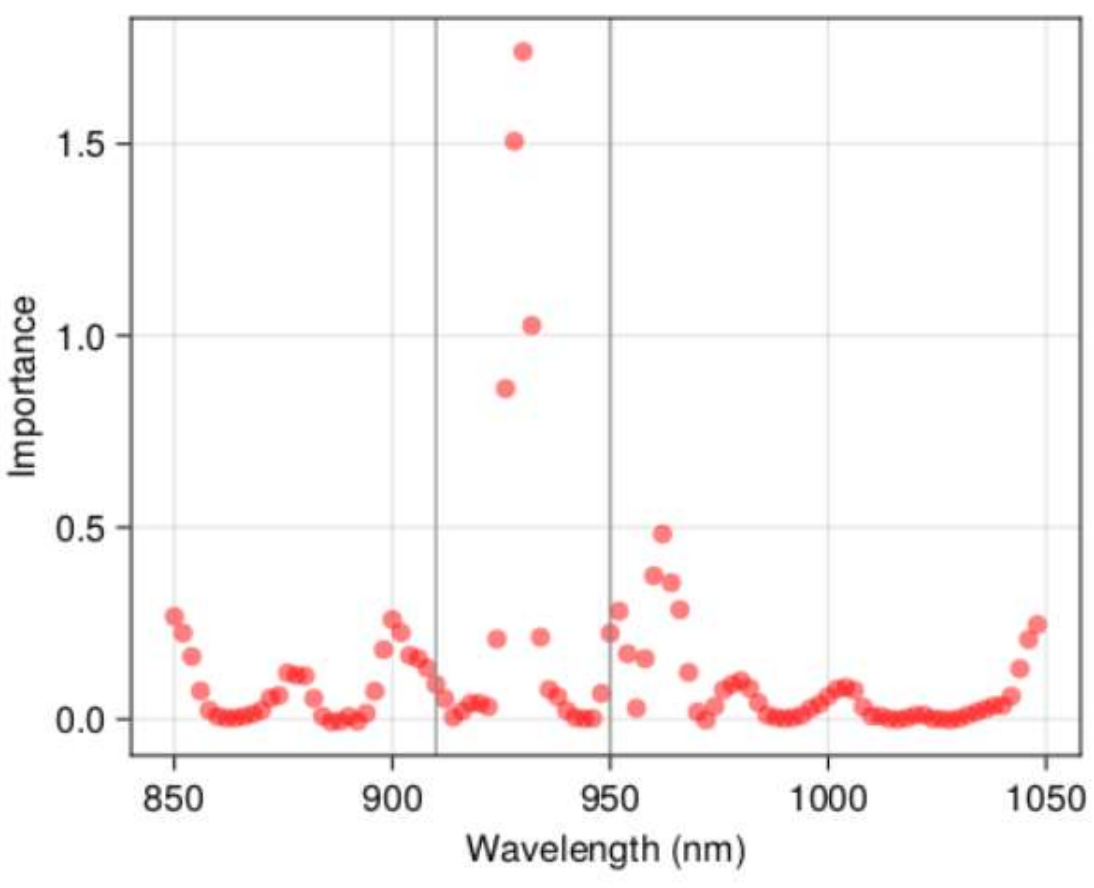
c) Then, for a given column $\mathbf{x}_j$ of Xval, the rows are randomly permuted, and the error rate Err is computed ⇒ $VI(\mathbf{x}_j)$ = Err – Err0

2) The process is replicated

⇒ Final VI = Average $VI(\mathbf{x}_j)$

Better method: permute several variables in the same time?

PLSR



viperm(Xtrain, ytrain; perm = 50,
score = rmsep, fun = plskern, nlv = 9)