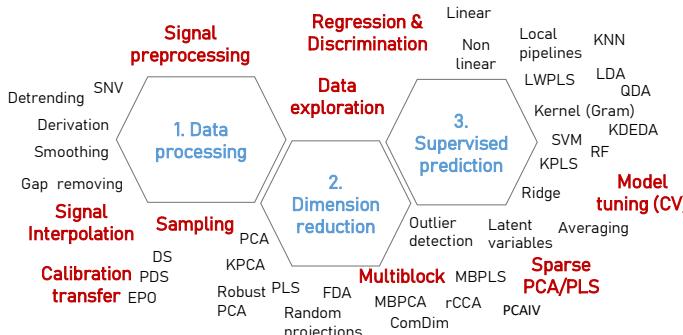


# Chemometrics with julia package Jchemo

Lesnoff, M. 2026. UMR SELMET  
Montpellier, France



## VERSATILE MACHINE LEARNING TOOLBOX FOR CHEMOMETRICS



<https://github.com/mlesnoff/Jchemo>

### Simple installation

- Jchemo ∈ [official repository of Julia](#) (equivalent to the CRAN for R)

### Easy syntax (generic functions `transf`, `fit!`, `predict`,...)

### Easy model tuning

- Grid-search (functions `gridcv`, `gridscore`)
- Sampling designs (Kennard-Stone, Duplex, Systematic, etc.)

### Easy implementation of ad'hoc pipelines

- model = `pip`(model1, model2, model3, ...)

### Documentation with examples (ex: `?pls kern`)

### Training materials (available on Github → `JchemoDemo`, `JchemoData`)

```

## Examples of pipelines

# 1) Preprocessing
#   SNV |> Savitzky-Golay
model1 = snv()
model2 = savgol(npoint = 11, deriv = 2,
                 degree = 3)
model = pip(model1, model2)
fit!(model, X)
Xp = transf(model, X)

plotsp(Xp; xlabel = "Wavelength (nm)",
       ylabel = "Absorbance").f

```

```

# 2) Non linear regression
# Data 'Challenge2018' Chemometrics Paris

# 2.1) SVMR on preliminary PLS scores
nlv = 25
metric = :eucl ; k = 200 ; h = 2

model1 = pcasvd(; nlv)
model2 = lwmrlr(; metric, h, k)
model = pip(model1, model2)
fit!(model, Xtrain, ytrain)
pred = predict(model, Xtest).pred ;

rmsep(pred, ytest)
> 0.813775

```

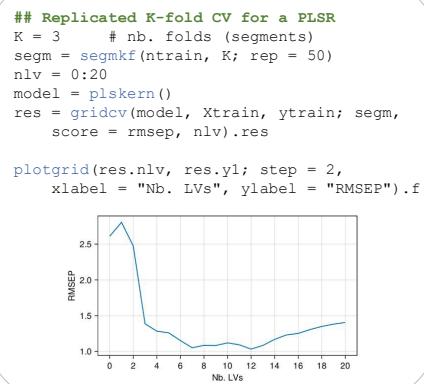
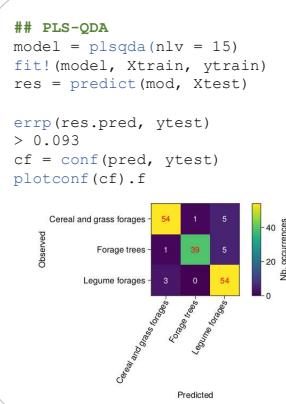
```

## Fast computations
# Ex: PLS2 with 1e6 observations
# Platform Info:
# OS: Windows (x86_64-w64-mingw32)
# CPU: 16 x Intel(R) Core(TM) i9-10885H CPU @ 2.40GHz

n = 10^6 # nb. observations (samples)
p = 500 # nb. X-variables (features)
q = 10 # nb. Y-variables to predict
X = rand(n, p)
Y = rand(n, q)

nlv = 25 # nb. PLS latent variables
model = plskern(; nlv)
@time fit!(model, X, Y)
7.532 seconds (299 allocations: 4.130 GiB, 6.58% gc time)

```



## 2.2) LWR algorithm (Næs et al. 1990 Analytical Chem.)

```

#   = KNN locally weighted MLR on preliminary PCA scores
nlv = 25
metric = :eucl ; k = 200 ; h = 2

model1 = pcasvd(; nlv)
model2 = lwmrlr(; metric, h, k)
model = pip(model1, model2)
fit!(model, Xtrain, ytrain)
pred = predict(model, Xtest).pred

rmsep(pred, ytest)
> 0.734050

```

## 2.3) A direct local PLSR pipeline: KNN-LWPLSR

```

#   Lesnoff et al. 2019 J. Chem.
nlvdis = 15 ; metric = :mah
k = 200 ; h = 2
nlv = 15

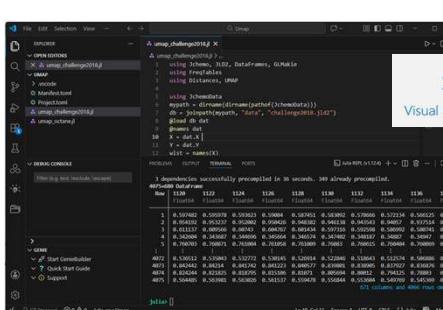
model = lwplsr(; nlvdis, metric, k, h, nlv)
fit!(model, Xtrain, ytrain)
pred = predict(model, Xtest).pred

```

```

rmsep(pred, ytest)
> 0.691477
plotxy(pred, ytest; color = (:red, .5), bisect = true,
       xlabel = "Prediction", ylabel, "Observed").f

```



```

## PLS-KDE-DA
model = plskdeda(nlv = 15)
fit!(model, Xtrain, ytrain)
res = predict(mod, Xtest)

errp(res.pred, ytest)
> 0.01

```

## Kernel density estimation: dmkmern()
# iris data

