Jchemo.jl

Chemometrics and machine learning on high-dimensional data with Julia

docs stable | docs dev | CI passing | repo status Active

Lesnoff, M. 2024. *UMR SELMET*, Montpellier, France
https://github.com/mlesnoff/Jchemo

julia Programming

julialang.org
Discourse
JuliaHub

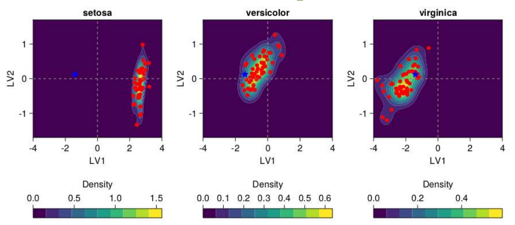
- Fast computing
- Easy to learn/read
- Open source

Multiple dispatch
Inplace functions
Simple multi-threading, GPU
High-quality graphics

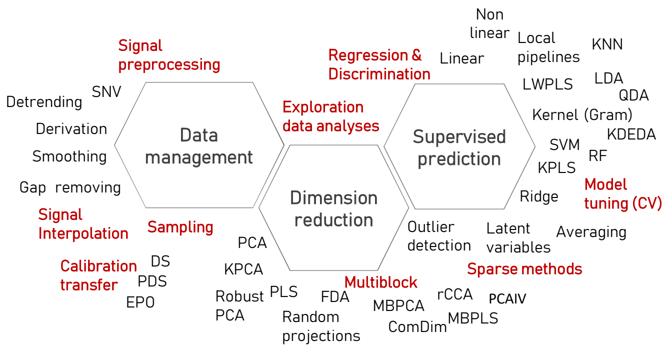
Packages manager
Project environments

```
## SNV preprocessing transformation
function transf!(object::Snv, X::Matrix)
    n, p = size(X)
    centr = object.par.centr
    scal = object.par.scal
    centr ? mu = rowmean(X) : mu = zeros(n)
    scal ? s = rowstd(X) : s = ones(n)
    @inbounds for j = 1:p
        X[:, j] .= (vcol(X, j) .- mu) ./ s
    end
end
```

```
## dmkern() Kernel density estimation, iris data
```



Jchemo?
CHEMOMETRICS and MACHINE LEARNING



See: https://mlesnoff.github.io/Jchemo.jl/dev/domains

Easy to use

Simple installation procedure
- Jchemo ∈ official Julia packages repository (equivalent to R Cran)

Consistent syntax

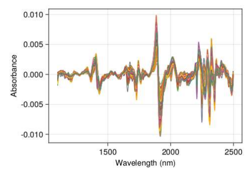
Generic functions of model tuning
- Grid-search cross-validation, validation-set
- Sampling designs (Kennard-Stone, Duplex, Systematic, etc.)

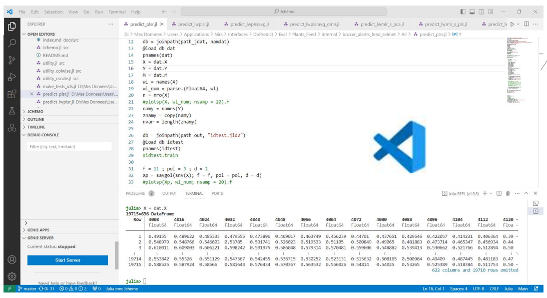
Pipelines
- `model = pip(mod1, mod2, mod3)`

Each function is documented with examples

Training materials are available (Github)

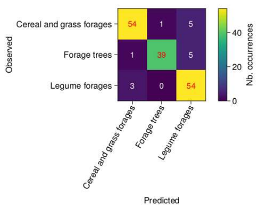
```
## Pipelines
## Ex: SNV :> Savitzky-Golay preprocessing
mod1 = snv(centr = true, scal = true)
mod2 = savgol(npoint = 11, deriv = 2, degree = 3)
model = pip(mod1, mod2)
fit!(model, X)
Xp = transf(model, X)
plotsp(Xp; xlabel = "Wavelength (nm)",
    ylabel = "Absorbance").f
```



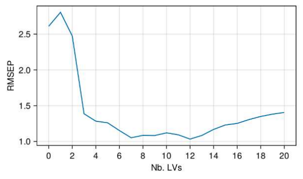


IDE: Visual Studio Code

```
## PLS-QDA
model = plsqda(nlv = 25)
fit!(model, Xtrain, ytrain)
res = predict(mod, Xtest)
errp(res.pred, ytest)
> 0.093
cf = conf(pred, ytest)
plotconf(cf).f
```



```
## Replicated K-fold CV
K = 3        # nb. folds (segments)
segm = segmkf(ntrain, K; rep = 30)
nlv = 0:20
model = plskern()
res = gridcv(model, Xtrain, ytrain; segm,
    score = rmsep, nlv).res
plotgrid(res.nlv, res.y1; step = 2,
    xlabel = "Nb. LVs", ylabel = "RMSEP").f
```



```
## PLS2 with 1e6 observations
Platform Info:
  OS: Windows (x86_64-w64-mingw32)
  CPU: 16 × 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)
```