NIPALS_PCA

A package for calculating PCA and PLS using the NIPALS implementation. Both models handles missing values

The package contains data structures for models and datasets

Installation

In Julia add https://gitlab.moffitt.usf.edu:8000/Bios2Projects/NIPALS_PCA.as unregistered package

Loading package

```
using NIPALS_PCA
```

Tutorial

PCA

1. Load package

```
using NIPALS_PCA
```

1. Load dataset from .csv file to DataFrame

```
x_df = loadIrisData()
```

1. Create dataset and apply normalize to mean center data

```
xdataset = parseDataFrame(x_df) |> normalize
```

1. Calculate PCA model

```
pca = calcPCA(xdataset, 3)
```

1. Calculate variances for model

```
calcVariances(xdataset,pca)
```

PLS modelling

Structures

`NIPALS_PCA.Dataset` — Type

```
struct Dataset
```

- `X::Array{Union{Missing, Float64},2}`
- `means::Array{Float64,1}`
- `stdevs::Array{Float64,1}`
- `value_columns::Array{String,1}`
- `xmask::BitArray{2}`
- `mv::Bool`

`NIPALS_PCA.PCA` — Type

```
struct PCA <: NIPALS_PCA.MultivariateModel
```

- `T::DataFrames.DataFrame`
- `P::DataFrames.DataFrame`

`NIPALS_PCA.PLS` — Type

```
struct PLS <: NIPALS_PCA.MultivariateModel
```

- `T::DataFrames.DataFrame`
- `P::DataFrames.DataFrame`
- `C::DataFrames.DataFrame`
- `W::DataFrames.DataFrame`
- `U::DataFrames.DataFrame`

Functions

General functionality

`NIPALS_PCA.calcPCA` — Function

calcPCA(dataset::Dataset, comps::Int64; normalize::Bool = false)

```
Calculates a PCA model
```

Examples

```
julia> calcPCA(datset,3,normalize=true)
```

`NIPALS_PCA.calcPLS` — Function

calcPLS(xdataset::Dataset,ydataset::Dataset,comps::Int64,incsamples::Array{Int64,1} = collect(1:size(xdataset.X)[1]))

```
Calculates a PLS model
```

`NIPALS_PCA.calcVariances` — Function

calcVariances(dataset::Dataset, model::PCA)::NamedTuple

```
Calculates r2x, r2x_cum and eigenvalues for all components in PCA model
```

Examples

```
julia> r2x,r2x_cum,eigenvalues = calcPCA(datset,model)
```

`NIPALS_PCA.loadmodel` — Function

loadmodel(path::String)::Tuple{MultivariateModel,Array{Float64,1},Array{Float64,1}}

```
Load PCA or PLS model from JLD2 file into a tuple containing the model, variable standard d
```

`NIPALS_PCA.savemodel` — Function

savemodel(model::T, dataset::Dataset, name::String) where T <: MultivariateModel

```
Save PCA or PLS model as JLD2 file
```

PLS normalization

`NIPALS_PCA.correct` — Function

correct(model::T, dataset::Dataset, name::String) where T <: MultivariateModel

```
Save PCA or PLS model as JLD2 file
```

`NIPALS_PCA.calibrate_model` — Function

calibrate_model(x::DataFrame,y::DataFrame,A::Int64, modelfile::String)

```
Calibrates PLS model based on datatypes in DataFrame for y

Columns of type CategoricalArray is handled by one-hot precedure

The calibrated model is saved to specified locations
```

`NIPALS_PCA.predict_xres` — Function

predict_xres(modelfile::String,xfile::String, outfile::String)

```
Loads model from jld2 file, predicts using xfile and exports residual matrix into .csv file
```