Simple-Run

Vignette Author

2023-04-03

Example script

```
library(tidyverse)
library(usethis)
library(devtools)
# run this everytime - it will only install if there are updates to the package.
# if you are getting errors, try using 'force = T'
devtools::install_github("TranscriptionFactory/LassoReg", force = F,
                         dependencies = F, quiet = T)
library(LassoReg, attach.required = T)
library(Matrix)


# load your data so gene names are in the columns and y-values prepended and in the first column
df = load_data

# get average size 6 clusters
cluster_input_avgSize6 = LassoReg::clustered_ppi_avgSize6

# get network modules
modules = LassoReg::getModules(dataframe, cluster_input_avgSize6)

# pass modulePA from modules list
results = LassoReg::LASSO_Grid(modules$modulePA, lambdaValues = c(0.95, 1.0, 1.05))

# get list of the genes in the network modules chosen by lasso
network_names = LassoReg::getNetworkNames(results)

# view plots (omit the outpath argument to get the plots returned)
plots = LassoReg::plotResults(results, outpath = "<path>")
```

Steps explained

```
library(tidyverse)
library(usethis)
library(devtools)

# run this everytime - it will only install if there are updates to the package.
# if you are getting errors, try using 'force = T'
devtools::install_github("TranscriptionFactory/LassoReg", force = F,
                         dependencies = F, quiet = T)

library(LassoReg, attach.required = T)
library(Matrix)
```

Create Network modules

Important If you are creating the modules for your own data, you need to pass your dataframe with the gene features to the getModules function. Input is clustered ppi data generated with ClustalOne. Included in the package are inputs that should work fine for many use cases. There are two cluster inputs that vary based on the average size of the clusters they contain. If you wish to cluster de-duplicated ppi data, the input passed to ClustalOne can be found as well. The getModules function returns a list with the Lasso input matrix stored "modulePA" list and the module results (both the modules that were chosen and the full list of modules).

important: if using network modules, pass the modulePA to the lasso function

```
# get binary ppi data to use for clustering
all_ppis = LassoReg::all_ppis

# get average size 6 clusters
cluster_input_avgSize6 = LassoReg::clustered_ppi_avgSize6

# get average size 8 clusters
cluster_input_avgSize8 = LassoReg::clustered_ppi_avgSize8


# To generate the network modules, call getModules
modules = LassoReg::getModules(dataframe, cluster_input_avgSize6)
```

Load data

First note: lambda here refers to a multiplier on the lambda chosen through cross validation.

Second note: some values of lambda may be too stringent for your data and result in no features being selected and a resulting error in calculating the svm. If this happens, try using a smaller value. You can test multiple values for lambda during a run. Here's some ideas for values to try (I usually increase/decrease in increments of 0.05) 1, 0.95, 1.0, 1.05

If you're getting good classification, use larger values to reduce the number of features chosen.

In general, # Larger alpha/lambda values = fewer features selected and vice versa

```
# these are example network modules
df = as.data.frame(LassoReg::example_data)

lambdaValues = c(0.75)
```

Run Lasso

LASSO_GRID will return: - gridResults: contains the results chosen in each of the 10 runs (each run contains 10-fold cross validation), for each of the lambda values - lambdas: the lambdaValues tested - lasso_input: the dataframe used - vars: the chosen variables

Lasso with network modules

```
# not run
results = LassoReg::LASSO_Grid(df$modulePA, lambdaValues = lambdaValues)
```

Lasso with individual features:

```
results = LassoReg::LASSO_Grid(df, lambdaValues = lambdaValues)

names(results)
#> [1] "gridResults" "lambdas"     "lasso_input" "vars"
```

Analyze features

The chosen features can be found like this:

```
chosen_vars = results$vars$chosen_vars
```

Analyze Network Modules (omit this step if not using network modules)

The names of the network modules; these don't work for the example data because the module information is not included.

```
# not run
network_names = LassoReg::getNetworkNames(results)
```

Analyze Results

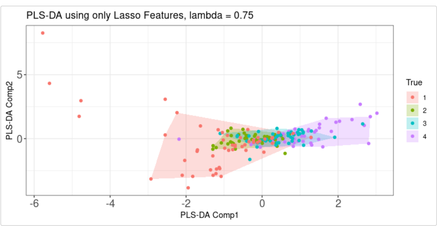
```
plots = LassoReg::plotResults(results)

# multiple plots get returned.
# One PLS-DA plot is returned for each lambda value
```

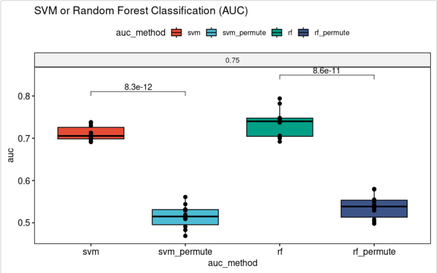
You can save the plots by calling the function like this instead:

```
plots = LassoReg::plotResults(results, outpath = "<path>")
```

```
plots[[1]]
#> [[1]]
```




```
plots[[2]]
```




```
plots[[3]]
```