SLIDE

SLIDE Pipeline

The SLIDE pipeline is consisted of below two steps. We recommend to run both steps on a computational cluster for optimal computational time.

1. Calculating and select latent factors (LFs) for multiple input parameter combinations.
2. Reviewing the output of step 1 and choose the optimal parameters for rigorous k-fold CV.

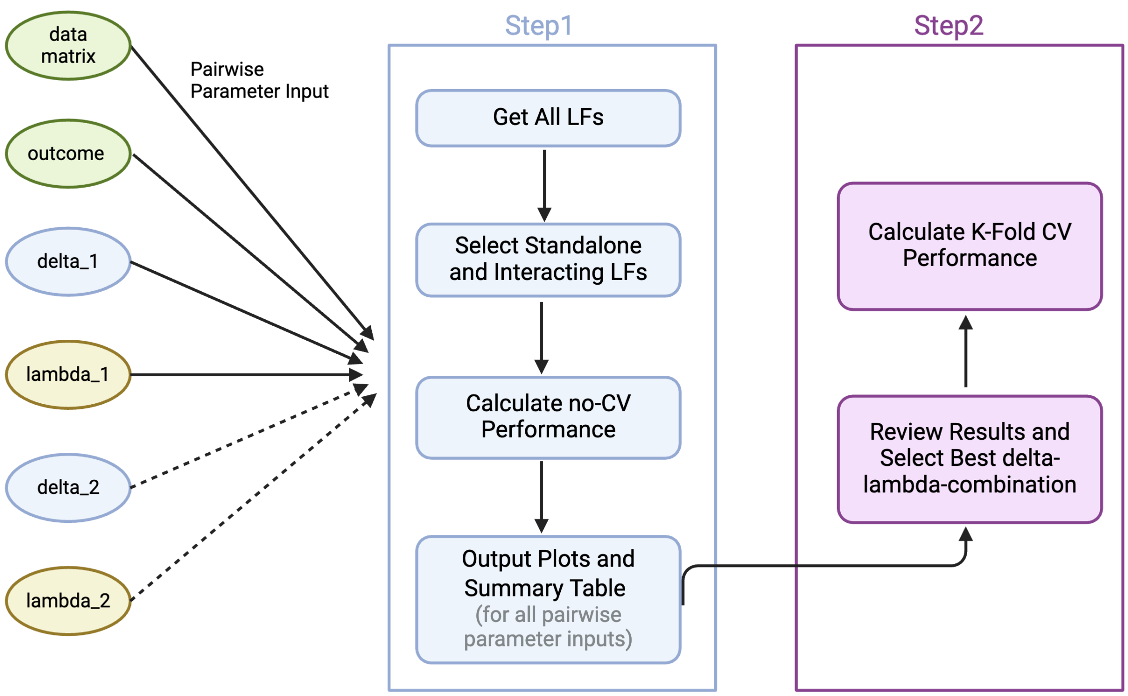

Figure 1: SLIDE workflow

Step 1: Parameter Tuning

1-1 Pre-Processing and Checking Input Data

Due to the assumption-free-nature of SLIDE, there are no data modality limit to the input of SLIDE.

The key input to SLIDE are just two csv files, the data matrix x and the response vector y file post pre-processing such as batch effect correction and/or normalization (for example, for scRN-seq data, first process with the standard Seurat pipeline).

The x file contains your data in a sample by feature format, such as single cell transcriptomics (**cell by gene**), or spatial proteomics (**region by protein**). The y file contains the responses of the data, such as severity of disease, spatial regions or clonal expansion. Since SLIDE is a regression method, when the response vector has multiple unique values (not just two classes), please make sure there are an ordinal relation between the y values. **Please make sure both csv files have row names and column names.**

In this tutorial, we are going to use the example Systemic Sclerosis dataset we have used in the SLIDE paper. This dataset is a human skin cell scRNA-seq dataset that we have transformed into the pseudo-bulk format.

If you have human single-cell data, our recommended workflow is to pseudobulk your dataset since the cell-to-cell variability is high. If you have mouse single cell data, with the reduced cell-to-cell variability, you can consider each cell as an sample. For single-cell datasets, the sparsity might be extremely high with high feature numbers. In this case, please see below for how to reduce the number of features and samples with large amount of zeros.

1-2 Parameters

The input to SLIDE is the path to a YAML file which documents the parameters.

SLIDE accepts many parameters from user input to give user as much freedom to tweak the method to their own data as possible. We have set default values to many parameters that works well with majority of the data moralities. Here, we explain what each of these parameters mean.

x_path: a string of the path to the data matrix (x) in the csv format with row names and column names where each row is samples (cells, patients, regions…) and each column is a features (genes, proteins…).

y_path: a string of the path of the response vector (y) in the csv format with row names and column names where each row is a sample and the column would be the outcome of interest..

out_path: a string of the path of a folder to store all output files (please see below section to interpret the outputs).

delta: control the number of all latent factors. The higher the delta, the less number of latent factors will be found. Default as 0.01 and 0.1.

lambda: control the sparsity of all latent factors. The higher the lambda, the less number of features will be in a latent factor. Default as 0.5 and 1.

spec: control the number of significant latent factors. The higher the spec, the less number of significant latent factors will be outputted. The desired number of output should be between 5 to 12 LFs. Default as 0.1.

y_factor: set to false if not binary and true if binary.

y_levels: null if y is continuous or ordinal. If y is binary, input a list of the correct order relationship such as [0, 1] or [1, 2].

eval_type: the performance evaluation metric used. corr for continuous Y and auc for binary Y.

SLIDE_iter: the number of times to repeat the SLIDE latent factor selection algorithm. The higher the iteration, the more stable the performance would be. Default as

```yaml
---
x_path: test/Ssc_x.csv
y_path: test/SkinScore.csv
out_path: test/out
delta:
  - 0.01
  - 0.1
lambda:
  - 0.5
  - 1.0
spec: 0.1
y_factor: false
y_levels: NULL
eval_type: corr
SLIDE_iter: 500
SLIDE_top_feats: 10
```

Figure 2: Example Yaml (Continuous Y)

```yaml
---
x_path: path/to/x.csv
y_path: path/to/y.csv
out_path: path/to/output/folder
delta:
  - 0.01
  - 0.1
lambda:
  - 0.5
  - 1
spec: 0.1
y_factor: yes
y_levels:
  - 0
  - 1
eval_type: auc
SLIDE_iter: 100
SLIDE_top_feats: 10
```

Figure 3: Example Yaml (Binary Y)

SLIDE_top_feats: the number of top features to plot from each latent factor. If set as n, a union of the top n weighted features and top n correlated (with y) features will be outputted.

do_interacts (optional): set to false if don't want interacting latent factors. Default as TRUE

thresh_fdr: set to lower if co-linearity of the features in the data matrix is high. Default as 0.2.

2-1 Step1 of SLIDE Framework

Once your YAML file is ready, we first recommend using a YAML validator website to ensure your YAML file is correctly formatted.

We can then check if the YAML file is read in correctly by reading it in as an variable.

```r
library(SLIDE)
yaml_path = "examples/test.yaml"
input_params = yaml::read_yaml(yaml_path)

knitr::kable(data.frame(arg = unlist(input_params)))
```

	arg
x_path	examples/Ssc_x.csv
y_path	examples/SkinScore.csv
out_path	examples/out
delta1	0.01
delta2	0.1
lambda1	0.5
lambda2	1
spec	0.3
y_factor	FALSE
thresh_fdr	0.2
eval_type	corr
SLIDE_iter	500
SLIDE_top_feats	10
CViter	10
do_interacts	TRUE

If you have a sparse dataset such as scRNA-seq datasets, we recommend filtering out samples and features that have too many zeros. zeroFiltering function will remove samples with more than g_thresh number of zeros and features with more than c_thresh number of zeros. An appropriate data matrix should at most have around 3-4k features. See `Preprocessing-and-Filtering` vignette for more information.

We then check if your data files are formatted correctly.

```r
checkDataParams(yaml_path)
#> Checking the format and dimensions of input data and response matrices...
#> Checking na values in the input data and response matrices...
#> Checking if yaml file is correct for the input data and response matrices...
```

If everything is formatted correctly, you can now run Step1 of SLIDE. If sink_file set to TRUE, all print statement will be printed to a txt file.

```r
optimizeSLIDE(input_params, sink_file = FALSE)
#> Populating all outputs to  examples/out .
#> Setting sigma as Null.
#> Getting latent factors for delta,  0.01 , and lambda,  0.5 .
#> Setting alpha_level at  0.05 .
#> Setting thresh_fdr at  0.2 .
#> Setting spec at  0.3 .
#> Setting eval_type as  corr .
#> Setting SLIDE_iter at  500 .
#> Setting SLIDE_top_feats as  10 .
#> Setting do_interacts as  TRUE .
#> Setting CViter as  10 .
#> f_size is set as  24
#> Loading required package: dplyr
#>
#> Attaching package: 'dplyr'
#> The following objects are masked from 'package:stats':
#>
#>     filter, lag
#> The following objects are masked from 'package:base':
#>
#>     intersect, setdiff, setequal, union
#>          final marginal spec: 0.3
#>       starting interaction selection . . .
#> [1] "Before doing interaction SLIDE"
#> [1] "z19"  "z20"  "z46"  "z129"
#> [1] "printig the yhat of each maginals:"
#> NULL
#>       running knockoffs on marginal/interaction submodels . . .
#> [1] "upsilon colnames:"
#> Getting real performance:
#> Getting Full  Random:
#> No id variables; using all as measure variables
#> Scale for x is already present.
#> Adding another scale for x, which will replace the existing scale.
#> Saving 7 x 7 in image
#> Getting latent factors for delta,  0.01 , and lambda,  1 .
#> Setting alpha_level at  0.05 .
#> Setting thresh_fdr at  0.2 .
#> Setting spec at  0.3 .
#> Setting eval_type as  corr .
#> Setting SLIDE_iter at  500 .
#> Setting SLIDE_top_feats as  10 .
#> Setting do_interacts as  TRUE .
#> Setting CViter as  10 .
#> f_size is set as  24
#>          final marginal spec: 0.3
#>     no interaction terms . . . no marginals
#>     no interaction terms . . . no marginals
#> Getting latent factors for delta,  0.1 , and lambda,  0.5 .
#> Setting alpha_level at  0.05 .
#> Setting thresh_fdr at  0.2 .
#> Setting spec at  0.3 .
```

```
#> Setting eval_type as  corr .
#> Setting SLIDE_iter at  500 .
#> Setting SLIDE_top_feats as  10 .
#> Setting do_interacts as  TRUE .
#> Setting CViter as  10 .
#> f_size is set as  24
#>          final marginal spec: 0.3
#>       starting interaction selection . . .
#> [1] "Before doing interaction SLIDE"
#> [1] "z26" "z45" "z70" "z77"
#> [1] "printig the yhat of each maginals:"
#> NULL
#>       running knockoffs on marginal/interaction submodels . . .
#> [1] "upsilon colnames:"
#> Getting real performance:
#> Getting Full  Random:
#> Getting partial random:
#> No id variables; using all as measure variables
#> Scale for x is already present.
#> Adding another scale for x, which will replace the existing scale.Saving 7 x 7 in image
#> Getting latent factors for delta,  0.1 , and lambda,  1 .
#> Setting alpha_level at  0.05 .
#> Setting thresh_fdr at  0.2 .
#> Setting spec at  0.3 .
#> Setting eval_type as  corr .
#> Setting SLIDE_iter at  500 .
#> Setting SLIDE_top_feats as  10 .
#> Setting do_interacts as  TRUE .
#> Setting CViter as  10 .
#> f_size is set as  24
#>          final marginal spec: 0.3
#>       starting interaction selection . . .
#> [1] "Before doing interaction SLIDE"
#> [1] "z45" "z70" "z77"
#> [1] "printig the yhat of each maginals:"
#> NULL
#>       running knockoffs on marginal/interaction submodels . . .
#> [1] "upsilon colnames:"
#> Getting real performance:
#> Getting Full  Random:
#> Getting partial random:
#> No id variables; using all as measure variables
#> Scale for x is already present.
#> Adding another scale for x, which will replace the existing scale.Saving 7 x 7 in image
#>   delta lambda f_size Num_of_LFs Num_of_Sig_LFs Num_of_Interactors
#> 1  0.01    0.5     24        172              4                  0
#> 2  0.01      1     24        172             NA                 NA
#> 3   0.1    0.5     24         88              4                  4
#> 4   0.1      1     24         88              3                  4
#>   sampleCV_Performance
#> 1    0.639539552525526
#> 2                   NA
#> 3    0.711411428383196
```

```
#> 4      0.838183538044026
```

The function above will generate a summary of each `delta` and `lambda` parameter, stored in `summary_table.csv`

```
summary_table = read.csv(paste0(input_params$out_path, "/summary_table.csv"), row.names = 1)

knitr::kable(data.frame(summary_table))
```

delta	lambda	f_size	Num_of_LFs	Num_of_Sig_LFs	Num_of_Interactors	sampleCV_Performance
0.01	0.5	24	172	4	0	0.6395396
0.01	1.0	24	172	NA	NA	NA
0.10	0.5	24	88	4	4	0.7114114
0.10	1.0	24	88	3	4	0.8381835

The `sampleCV_Performance` column will tell us which of our parameters gives latent factors that perform well in short cross validation run. We generally want to pick the parameters that have the highest value for `sampleCV_Performance`. Once we pick parameters, we will run a more rigorous cross validation with more iterations.

Finally, we can generate correlation plots for the top features in each significant latent factor.

```
plotCorrelationNetworks(input_params)
```