Test Plan

Tram Chau

2023-09-29

1. Introduction

'dtrans' package's main purpose is transforming data by three techniques: PCA, NMF and KPCA. The main object is 'transformer' defined as an S3 class. Thus, the object is created by one of three techniques above. There are 3 creator functions are redeveloped based on stat, NMFN, and kernlab ([1] [2] [3]).

There are six S3 methods are new development, they are defined for the 'transformer' class including print, summary, plot, plottran, transform, and inverse. These S3 methods will support user to perform different tasks relating to transforming data.

2. Objectives & Tasks

a. Objectives

Implementing package testing provides different scenarios for code to tackle and ensures the code runs robustly. Testing checks the code behaves appropriately to different inputs by covering different cases with predefined expected outputs. Most of the tests are developed automatically, thus, these tests can be rerun to ensure the consistency of the code whenever the code is updated, maintained, or developed the new features. Testing is another piece of document for the code.

b. Tasks

- Defining the test categories.
- Defining function for common checkings.
- Preparing test script.
- Updating the code to react accordingly to the test scripts and vice versa.
- Run test and fix if any fail test.
- Push package to GitHub and set GitHub Action for platform testing.

3. scope and prerequisites.

a. In scope:

- Test the inputs of all functions.
- Test the output of all functions.
- Test the accuracy of the 3 algorithms.
- System and performance testing.

b. Out of scope

- The feature 'handle_discrete' has not been implemented yet, currently it is a place holder for later developing.
- 'plot3d' function is not developed yet.
- RCpp function is not implemented yet.

c. Prerequisite:

Install packages NMFN and kernlab to cross check the accuracy of nmf and kernel pca algorithms.

4. Testing strategy

As mentioned in the Scope section, there are 4 categories of testing including Input, Output, Algorithm, and System testing. The Input, Output and Algorithm testing are set by 'testthat' package which supports automated unit testing. System testing is conducted by installing the package by different modes and setting the GitHub Action.

a. Input testing

Purpose: test different scenarios with invalid inputs and provide reasonable information to user of what being wrong and try to catch the error before the invalid inputs propagate to a deeper layer which may confuse user and take time to figure out.

Methodology:

- Develop automated unit tests with 'testthat' package.
- Update the script to include input validating with predefined warning or stopping message.
- For creator functions: transformer.pca, transformer.nmf, transformer.kpca
  - The common inputs of these 3 creator functions (i.e. x, components, center, scaling, handle_discrete) are validated in function .validate_instantiate_input().
  - testing extra parameters inside each function for specific algorithm.
- For S3 methods: print, summary, plot, plottrans, transform, inverse
  - the 'transformer' object is validated in function .validate_object().
  - testing extra parameters inside each function accordingly.

b. Output testing

Purpose: test the expected output of specific function. This test includes testing the output structure and the output value(s).

Methodology:

- Write the unit test to compare the executed output vs predefined output by 'testthat' package.
- Test the effect of different valid input values to the output value. e.g. center, scale parameters...
- Function and their outputs:
  - transformer.pca, transformer.nmf, transformer.kpca: have output as the 'transformer' object.

  - print, summary: have the output as text.
  - inverse, transform: have the output as dataset.
  - plot, plottran: have the output as visualizations.
- Test the output structure of S3 class 'transformer'. The object is validated by function .validate_object().

c. Algorithm testing

Purpose: to ensure the newly developed function works correctly based on the referenced packages (stat, NMFN, kerlab)

Methodology:

- For redeveloped functions (creator functions), they will be compared to the other packages based on the transformed data. Specifically, comparing attribute 'x' of the object 'Transformer' with corresponding packages as below:
  - transformer_pca: by comparing with attribute 'x' of the object 'prcomp' created by stat::prcomp.
  - transformer_nmf: by comparing with attribute 'W' of the list created by NMFN::nnmf.
  - transformer_kpca: by comparing with the output of calling rotated() on the object 'kpca' created by kernlab::kpca.
- Compare the output of function with the object's attribute or previous variable:
  - inverse function: undo transformed data back to the original data, then compare this output with the data used to create the transformer object. These data are not matched perfectly, due to the the leave-out characteristic of the transformed data, a certain threshold should be set for comparison. Furthermore, call 'inverse' on the transformed new_data cause greater mismatched than the transformed fit data, so the threshold should be higher.
  - transform function: transform same data (which is used to created the Transformer object) and check this with the object's attribute x, etc...

d. System and performance test

Purpose: Test whether the package can be installed by different methods. Test whether the package works probably on different platforms. Test the performance of the package on big dataset.

Methodology:

- Install from source code of package.
- Install from GitHub.
- Setting GitHub action for the platform testing.
- Set up the performance test on big dataset.

5. References

1. R Core Team contributors worldwide The r stats package

2. Suhai (TImothy) Liu Non-negative matrix factorization package

3. Alexandros Karatzoglou Kernlab: Kernel-based machine learning lab package

6. Appendix

Script for referenced packages:

```
library(NMFN)
library(kernlab)
data(iris)
pca_obj <- prcomp(iris[,1:4], rank. = 2, center=FALSE)
nmf_obj <- nnmf(iris[,1:4], k=2)
kpca_obj <- kpca(~., iris[,1:4], features=2)
```