import packages

```
import numpy as np
import pandas as pd
import sys
import os
import time as tm
import pickle
from functools import partial
import scipy.stats as st
from scipy.stats import wasserstein_distance
import scipy.stats
import copy
from sklearn.model_selection import KFold
import pandas as pd
import multiprocessing
import matplotlib as mpl
import matplotlib.pyplot as plt
import scanpy as sc
import warnings
import subprocess
import seaborn as sns
from sklearn.metrics import mean_squared_error
from scipy.spatial.distance import jensenshannon
from scipy.stats import pearsonr,ttest_ind,mannwhitneyu
import matplotlib
import time
```

###please make sure that you have changed the directory to 'SpatialBenchmarking'.

```
os.chdir('../')
### please add the your SpatialBenchmarking dir into the pythn path
sys.path.append('~/SpatialBenchmarking/')
```

###Predict gene spatial distribution of undetected genes
###Prepared files

```
### We sssume that you have these files:
   #1): scRNA-seq count files (eg. scRNA_count.txt);
       # genes X cells, each row is a gene and each col is a cell.
   #2): spatial transcriptomics count files (eg. Insitu_count.txt);
       # spots X genes, each col is a gene. Please note that the file has no index
   #3): spatial location files (eg. Locations.txt);
       # each col is a spot coordinates. Please note that the file has no index
       # default: None. It is necessary when you use SpaOTsc or novoSpaRc to integrate datasets.
   #4): CellTypeAnnotate file for scRNA-seq data, when using Tangram for prediction. (Option)
       # if this CellTypeAnnotate is supported, you must set the annotate != None and modes = 'clusters'
```

```
import numpy as np
import pandas as pd
###Import the package "SpatialGenes" to directly predict the gene spatial distribution for any spatial datasets.
import Benchmarking.SpatialGenes as SpatialGenes

### input data directory
PATH = 'DataUpload/Dataset15/'

### scRNA-seq count files (genes X cells, each row is a gene and each col is a cell)
RNA_path = PATH + 'scRNA_count.txt'

### spatial transcriptomics count files (each col is a gene)
Spatial_path =  PATH + 'Insitu_count.txt'

### spatial location files (each col is a spot coordinate)
location_path = PATH + 'Locations.txt'

### Read data
RNA_data = pd.read_table(RNA_path,header = 0,index_col = 0)
Spatial_data = pd.read_table(Spatial_path,sep = '\t',header=0)

### genes for integration and prediction. Please note it must be a list.
train_list = list(RNA_data.index&Spatial_data.columns)
test_list = list(set(RNA_data.index) - set(Spatial_data.columns))[:20]

### Outfile directory
outdir =  'Dataset15/'
if not os.path.exists(outdir):
    os.mkdir(outdir)

### Running
test = SpatialGenes.GenePrediction(RNA_path, Spatial_path, location_path, train_list = train_list, test_list = test_list,
outdir = outdir)

### choose tools you you want use for prediction
Methods = ['SpaGE','novoSpaRc','SpaOTsc','Seurat','LIGER']

### Prediction
Result = test.Imputing(Methods)
```

###GPU Platform gimVI, Tangram, and stPlus

```
import os
import numpy as np
import pandas as pd
import Benchmarking.SpatialGenes as SpatialGenes
import os
from stPlus import *

PATH = 'DataUpload/Dataset15/'
RNA_path = PATH + 'scRNA_count.txt'
Spatial_path =  PATH + 'Insitu_count.txt'
location_path = PATH + 'Locations.txt'
RNA_data = pd.read_table(RNA_path,header = 0,index_col = 0)
Spatial_data = pd.read_table(Spatial_path,sep = '\t',header=0)
train_list = list(RNA_data.index&Spatial_data.columns)
print (train_list)
test_list = list(set(RNA_data.index) - set(Spatial_data.columns))[:20]

outdir =  'Dataset15/'
if not os.path.exists(outdir):
    os.mkdir(outdir)
test = SpatialGenes.GenePrediction(RNA_path, Spatial_path, location_path, train_list = train_list, test_list = test_list,
outdir = outdir)
Methods = ['Tangram', 'gimVI', 'stPlus']
Result = test.Imputing(Methods)
```

###Prediction Celltype deconvolution
###Prepared files:

```
  #You can import the package "DeconvolutionSpot" to directly predict the cell locations for any spatial datasets.

  # Before forecasting, please prepare the following files:
  #1): scRNA count files, h5ad file or h5seurat file;
  #2): spatial count files, h5ad file or h5seurat file;
  # 3): scRNA cell annotation files;
  #4): output dir.

  # For more details, please see the Benchmarking/DeconvolutionSpot.py and Figure Data.

time_start=time.time()
### please add the your SpatialBenchmarking dir into the pythn path
sys.path.append('~/SpatialBenchmarking/')
import Benchmarking.DeconvolutionSpot as DeconvolutionSpot
```

Celltype deconvolution Prediction
for Cell2location, Stereoscope, Tangram, DestVI, you must have .h5ad files as input.
for Cell2location, you must add my_python_path.

```
RNA_h5ad = 'ExampleData/Simulated_STARmap/starmap_sc_rna.h5ad'
Spatial_h5ad = 'ExampleData/Simulated_STARmap/starmap_spatial.h5ad'
celltype_key = 'celltype'
my_python_path = '~/miniconda2/envs/Deconvolution/bin/python'
output_path = 'FigureData/Figure4/Dataset10_STARmap/Result_STARmap/'
if not os.path.exists(output_path):
    os.mkdir(output_path)

test = DeconvolutionSpot.Deconvolutions(RNA_h5ad = RNA_h5ad, Spatial_h5ad = Spatial_h5ad, celltype_key =
celltype_key, my_python_path = my_python_path, output_path = output_path)
Methods = ['Cell2location', 'Stereoscope','Tangram', 'DestVI']
Result = test.Dencon(Methods)
time_end=time.time()
print('STARmap datasets prediction time cost',(time_end-time_start)/60,'minutes')

sys.path.append('~/SpatialBenchmarking/')
import Benchmarking.DeconvolutionSpot as DeconvolutionSpot
```

for SpatialDWLS, RCTD, Seurat, SPOTlight, you must have .h5seurat files as input.

```
RNA_h5Seurat = 'ExampleData/Simulated_STARmap/starmap_sc_rna.h5seurat'
Spatial_h5Seurat = 'ExampleData/Simulated_STARmap/starmap_spatial.h5seurat'
celltype_key = 'celltype'
output_path = 'FigureData/Figure4/Dataset10_STARmap/Result_STARmap/'
if not os.path.exists(output_path):
    os.mkdir(output_path)

test = DeconvolutionSpot.Deconvolutions(RNA_h5Seurat= RNA_h5Seurat, Spatial_h5Seurat = Spatial_h5Seurat,
celltype_key = celltype_key, output_path = output_path)
Methods = ['SpatialDWLS','RCTD', 'Seurat', 'SPOTlight']
Result = test.Dencon(Methods)

sys.path.append('~/SpatialBenchmarking/')
import Benchmarking.DeconvolutionSpot as DeconvolutionSpot
```

for STRIDE, have count matrix files as input.

```
RNA_file = 'ExampleData/Simulated_STARmap/starmap_sc_rna.tsv'
Spatial_file = 'ExampleData/Simulated_STARmap/starmap_spatial.tsv'
celltype_file = 'ExampleData/Simulated_STARmap/starmap_sc_rna_celltype.tsv'
output_path = 'FigureData/Figure4/Dataset10_STARmap/Result_STARmap/'
if not os.path.exists(output_path):
    os.mkdir(output_path)

test = DeconvolutionSpot.Deconvolutions(RNA_file = RNA_file, Spatial_file = Spatial_file, celltype_file = celltype_file,
output_path = output_path)
Methods = ['STRIDE']
Result = test.Dencon(Methods)
```