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('../')
```

###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): count files containing the number of cells in each spot for Tangram(option) (eg. count.txt).
        # The col represents a spot coordinates. Please note that has no index and the file columns must be 'cell_counts'
        # default: None. It is necessary when you use Tangram_seq functions to integrate datasets.
        # Please note that Tangram can be used in two ways : Tangram_image or Tangram_seq.
        # Only when  you have information file that containing the number of cells in each spot, you can use Tangram_seq.
```

```
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 = 'FigureData/Figure2/Dataset2_osmFISH/Rawdata/'

### 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 =  'FigureData/Figure2/Dataset2_osmFISH/Test/'
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','gimVI','Tangram_image','Seurat','LIGER']

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

###Prediction Cell Locations
###Prepared 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. combined_spatial_count.txt);
      # spots X genes, each col is a gene. Please note that the file has no index.
  #3): spatial location files (eg. combined_Locations.txt);
      # spots X coordinates, 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): scRNA cell annotation files (scRNA_annotate.txt)
      # cells X celltype, each row is a cell and each col is a cell annotation.
      # Please note that the file must have a columns named 'celltype'
  #5): count files containing the number of cells in each spot for Tangram (option) (eg. combined_
cell_counts.txt);
      # spots X numbers. each row is a spot index and the col represents the cell count in each spot.
      # Please note that the file columns must be 'cell_count'
      # default: None. It is necessary when you use Tangram to integrate datasets.

```
#import the package "CellAssigment" to directly predict the cell locations.
import Benchmarking.CellAssigment as CellAssigment

time_start=time.time()
### input data directory
PATH = 'FigureData/Figure4/Dataset7_STARmap/'

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

### spatial transcriptomics count files (each col is a gene)
spatial_count = PATH + 'Simulated_STARmap/combined_spatial_count.txt'

###spatial location file (spots X coordinates, each col is a spot coordinates. Please note that the file has no index).
location = PATH + 'Simulated_STARmap/combined_Locations.txt'

### count files containing the number of cells in each spot for Tangram (each row is a spot index and each col is a
cell)
cell_counts = PATH + 'Simulated_STARmap/combined_cell_counts.txt'

### scRNA cell annotation files (each row is a cell and each col is a cell annotation)
scrna_meta = PATH + 'Rawdata/scRNA_annotate.txt'

### In scRNA cell annotation files,for one cell  you may have different cell annotation, choose a columns as a annota-
tion for input
annotatetype = 'subclass'

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

### Running
MC = CellAssigment.MappingCell(RNA_path = scRNA, Spatial_path = spatial_count, location_path = location,
                           count_path = cell_counts, scrna_annotation = scrna_meta,
                           annotatetype = annotatetype,outdir = outdir)
### choose tools you you want use for prediction
Tools = ['novoSpaRc','SpaOTsc','Seurat','Tangram']
### prediction
MC.workstart(Tools)
time_end=time.time()
print('STARmap datasets prediction time cost',(time_end-time_start)/60,'minutes')
```