Positive_pattern_regex

May 17, 2019

[146]:
```
import time
import gzip
import shutil
import pandas as pd
import numpy as np
import re
import twobitreader as tbr
from Bio import SeqIO
from numpy import array
from numpy import argmax
import matplotlib.pyplot as plt
from itertools import islice
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
```

[147]:
```
#Creating Reverese Complements
def reverseComp(Seq):
    seq = Seq.upper()
    d = {'A':'T', 'T':'A', 'G':'C', 'C':'G'}
    try:
        seq = seq[::-1]
        rc_seq = "".join([d[nuc] for nuc in seq])
    except KeyError:
        return "Not Viable DNA Seq"
    return rc_seq
```

[148]:
```
motif = '[AGC]TGA[CTG][ATCG][GCAT][AGT]GC[ATCG]'
regBS = re.compile(motif)
motifDF = []
motifQuant = []
genome = tbr.TwoBitFile('/Users/kalyanidhusia/Downloads/hg19.2bit')
with open('/Users/kalyanidhusia/Desktop/nrf2_R/ENCFF126HBJ.bed') as f:
    for line in f:
        if line.startswith('track') == False:
            peak = list(line.split())
            seq = (genome[peak[0]][int(peak[1]):int(peak[2])]).upper()
            rSeq = reverseComp(seq)
```

```
            sequences = []
            sequences.extend(re.findall(regBS, seq))
            #import pdb;pdb.set_trace() code used for debugging
            sequences.extend(re.findall(regBS, rSeq))
            if len(sequences) >= 0:
                seqs = pd.DataFrame({'binding':sequences, 'chrom':peak[0],␣
 ↪'chromstart':peak[1], 'chromend':peak[2], 'NR':'NRF2'})
                motifDF.append(seqs)
                motifQuant.append([peak[0], peak[1], peak[2], len(seqs),␣
 ↪len(seq)])
search_reg = pd.concat(motifDF)
names = ['chrom', 'chromstart', 'chromend', 'numOfMatches', 'lenSeq']
dist_reg = pd.DataFrame(motifQuant, columns=names)
dist_reg.head()
n = 5
x = [len(i[6+n:-6-n]) for i in search_reg['binding']]
```

[149]:
```
motifDF
search_reg.head()
```

[149]:

	binding	chrom	chromstart	chromend	NR
0	ATGACTCAGCA	chr10	5113978	5114231	NRF2
0	ATGACGGAGCA	chr20	48909147	48909439	NRF2
0	ATGACTCAGCA	chr6	143730457	143730733	NRF2
1	ATGAGTGGGCT	chr6	143730457	143730733	NRF2
0	GTGACTCAGCG	chr22	35767953	35768232	NRF2

[150]:
```
search_reg.head()
```

[150]:

	binding	chrom	chromstart	chromend	NR
0	ATGACTCAGCA	chr10	5113978	5114231	NRF2
0	ATGACGGAGCA	chr20	48909147	48909439	NRF2
0	ATGACTCAGCA	chr6	143730457	143730733	NRF2
1	ATGAGTGGGCT	chr6	143730457	143730733	NRF2
0	GTGACTCAGCG	chr22	35767953	35768232	NRF2

[151]:
```
dist_reg.head()
```

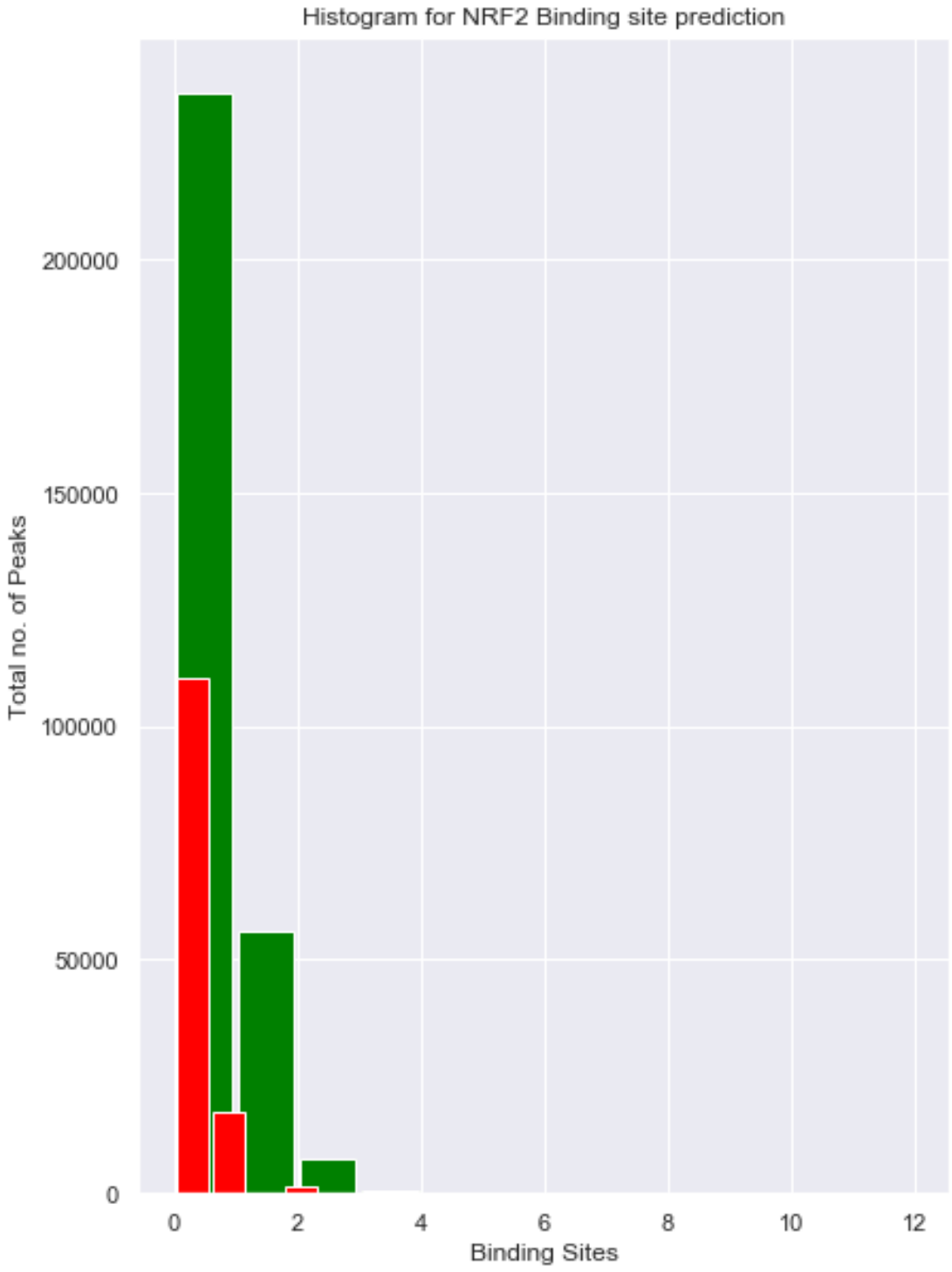
[151]:

	chrom	chromstart	chromend	numOfMatches	lenSeq
0	chr10	5113978	5114231	1	253
1	chr20	48909147	48909439	1	292
2	chr6	143730457	143730733	2	276
3	chr22	35767953	35768232	4	279
4	chr16	73054395	73054618	1	223

[152]:
```
motif = '[AGC]TGA[CTG][ATCG][GCAT][AGT]GC[ATCG]'
regBS = re.compile(motif)
motifDF = []
motifQuant = []
genome = tbr.TwoBitFile('/Users/kalyanidhusia/Downloads/hg19.2bit')
```

```
with open('/Users/kalyanidhusia/Desktop/nrf2_R/search2/min_score70/DNAse/
 ↪negativenrf2/sudin_negative_nrf2.bed') as f:
    for line in f:
        if line.startswith('track') == False:
            peak = list(line.split())
            seq = (genome[peak[0]][int(peak[1]):int(peak[2])]).upper()
            rSeq = reverseComp(seq)
            sequences = []
            sequences.extend(re.findall(regBS, seq))
            sequences.extend(re.findall(regBS, rSeq))
            if len(sequences) >= 0:
                seqs = pd.DataFrame({'binding':sequences,'chrom':
 ↪peak[0],'chromstart':peak[1],'chromend':peak[2],'NR':'NRF2'})
                motifDF.append(seqs)
                motifQuant.append([peak[0], peak[1], peak[2], len(seqs),␣
 ↪len(seq)])
zsearch_reg = pd.concat(motifDF)
names = ['chrom', 'chromstart', 'chromend', 'numOfMatches', 'lenSeq']
zdist_reg = pd.DataFrame(motifQuant, columns=names)
dist_reg.head()
n = 5
x = [len(i[6+n:-6-n]) for i in zsearch_reg['binding']]
```

[165]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(15,10))
plt.subplot(121)
plt.title('Histogram for NRF2 Binding site prediction')
plt.xlabel('Binding Sites')
plt.ylabel('Total no. of Peaks')
plt.grid(True)
plt.hist(dist_reg['numOfMatches'],rwidth = 0.9, bins= 12, color ="green")
plt.hist(zdist_reg['numOfMatches'], rwidth = 0.9, bins= 12, color ="red")
#plt.subplot(122)
#plt.title('Histogram for Negative NRF2 Binding site prediction')
#plt.xlabel('Binding Sites')
#plt.ylabel('Total no. of Peaks')
#plt.grid(True)
#plt.hist(zdist_reg['numOfMatches'], rwidth = 0.5, bins= 12, color ="red")
#plt.tight_layout()
plt.show()
#plt.savefig('compare_hist')
```



[154]:
```
#One Hot Encoding Sequence
def oheSeq(DNAString):
    seq = DNAString.upper()
    nuc = 'ATGC'
```

```
    char2int = dict((c, i) for i, c in enumerate(nuc))
    int2char = dict((i, c) for i, c in enumerate(nuc))
    integer_encoded = [char2int[char] for char in seq]
    OHE = []
    for value in integer_encoded:
        letter = [0 for _ in range(len(nuc))]
        letter[value] = 1
        OHE.append(letter)
    seq_ohe = np.asarray(OHE)
    return seq_ohe
```

[155]:
```
#modified OHE for interspaced regions
def oheSeqMod(DNAString, flank_len):
    seq = DNAString.upper()
    flanks = seq[:6+flank_len] + seq[-6-flank_len:]
    nuc = 'ATGC'
    char2int = dict((c, i) for i, c in enumerate(nuc))
    int2char = dict((i, c) for i, c in enumerate(nuc))
    integer_encoded = [char2int[char] for char in flanks]
    OHE = []
    for value in integer_encoded:
        letter = [0 for _ in range(len(nuc))]
        letter[value] = 1
        OHE.extend(letter)
    OHE.append(len(seq[6+flank_len:-6-flank_len]))
    return OHE
```

[156]:
```
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import umap
from mpl_toolkits.mplot3d import Axes3D
from sklearn.datasets import fetch_mldata
```

[157]:
```
#Dimensionality Reductions
#euc for euclidean
#Dimensionality reduction using PCA takes ~30m mins
Peuc_ohe =  np.array([oheSeqMod(i, 6) for i in search_reg['binding']])

pca = PCA(n_components=3)
pca_result = pca.fit_transform(Peuc_ohe)
print('Explained variation per principal component: {}'.format(pca.
 ↪explained_variance_ratio_))
pca_result.size
```

```
Explained variation per principal component: [0.13439471 0.09280625 0.08533555]
```

[157]: 220107

[158]:
```
#to make the data of same size in case we need it while plotting
pospca_result = pca_result[:20640]
pospca_result.size
```

[158]:
```
61920
```

[159]:
```
#Dimensionality Reductions
#euc for euclidean
#Dimensionality reduction using PCA takes ~30m mins
Neuc_ohe =  np.array([oheSeqMod(i, 6) for i in zsearch_reg['binding']])

pca = PCA(n_components=3)
negpca_result = pca.fit_transform(Neuc_ohe)
print('Explained variation per principal component: {}'.format(pca.
 ↪explained_variance_ratio_))
negpca_result.size
```

```
Explained variation per principal component: [0.11961577 0.09545381 0.09407146]
```

[159]:
```
61920
```

[202]:
```
#X = pospca_result
#y = negpca_result
#print(X.shape, y.shape)
a = np.array([[ -2.11448   ,  -0.02819252,  -2.3632686 ],
       [ 22.176752  ,  -9.464946  ,  -7.922308  ],
       [ -2.6372638 ,   2.6327784 ,  -1.5447168 ],
       [-10.927417  , -25.546335  ,  13.456126  ],
       [ -4.7346973 , -24.675365  ,   0.42249298],
       [  5.0015845 ,   6.3349667 ,  17.936752  ]])
a
b = np.array([[-13.174172  ,   3.2923772 ,  31.345163  ],
       [-13.853187  , -19.450788  ,  12.798331  ],
       [-13.853195  , -19.451141  ,  12.798458  ],
       [ 21.645144  ,   2.8262284 ,  -4.3555226 ],
       [ -5.321156  ,  -0.73769414, -28.301693  ],
       [ -9.092838  ,  -9.796622  ,   2.7244713 ]])
b
cat = np.concatenate((pca_result, negpca_result), axis=0)
cat

#d = np.concatenate((a, b), axis=1)
#d

#e = np.concatenate((a, b), axis=None)
#e
```

[202]:
```
array([[ 2.44178574,  0.13151673, -0.6782166 ],
       [ 1.33817814, -0.19947998,  0.18855484],
```

```
       [ 2.44178574,  0.13151673, -0.6782166 ],
       ...,
       [ 2.28422557,  0.26635381, -0.52823908],
       [-0.46431435, -0.02691042, -0.96319262],
       [-0.89730458, -0.99381163, -0.34084207]])
```

[212]:
```
dim1 = cat[:,0]
dim2 = cat[:,1]
plt.figure(figsize=(5,5))
plt.scatter(cat[:,0], cat[:,1], c=dim1,label = '-ve', s=1)
plt.title("PCA for Negative data")
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.show()
```

[161]:
```
# 10 nice colors
col = np.array(['#a6cee3','#1f78b4','#b2df8a','#33a02c','#fb9a99',
                '#e31a1c','#fdbf6f','#ff7f00','#cab2d6','#6a3d9a'])
# Subsampling
#Showcase of various other options
```

```
plt.figure(figsize=(10,10))
plt.scatter(pca_result[:,0], pca_result[:,1], c='#b2df8a', label = '+ve')
plt.scatter(negpca_result[:,0], negpca_result[:,1], color='red', label = '-ve',␣
↪s=1)
plt.title('Comparative PCA', fontsize=16)
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.legend()
plt.tight_layout()
```

[162]:
```
#Dimensionality Reductions
#euc for euclidean
```

```
#euc_ohe =  np.array([oheSeqMod(i, 6) for i in search_reg['binding']])
#from sklearn.decomposition import PCA
#from sklearn.manifold import TSNE
#import umap
#Be careful with umap installation pip install umap will cause you to install␣
 ↪the wrong ver of umap
#(and will also break the real module)

#pca = PCA(n_components=2)
#pca.fit(euc_ohe)
#print(pca.explained_variance_ratio_)

#Takes a bit with larger datasets (scales n^2 in both compute time and memory)

X_pembedded = TSNE(n_components=3).fit_transform(Peuc_ohe)
X_pembedded
```

[162]:
```
array([[ -2.11448   ,  -0.02819252,  -2.3632686 ],
       [ 22.176752  ,  -9.464946  ,  -7.922308  ],
       [ -2.6372638 ,   2.6327784 ,  -1.5447168 ],
       ...,
       [-10.927417  , -25.546335  ,  13.456126  ],
       [ -4.7346973 , -24.675365  ,   0.42249298],
       [  5.0015845 ,   6.3349667 ,  17.936752  ]], dtype=float32)
```

[166]:
```
#t-Distributed stochastic neighbor embedding (t-SNE) minimizes the divergence␣
 ↪between two distributions:
#a distribution that measures pairwise similarities of the input objects and a␣
 ↪distribution that measures pairwise similarities of the corresponding␣
 ↪low-dimensional points in the embedding

X_nembedded = TSNE(n_components=3).fit_transform(Neuc_ohe)
X_nembedded
```

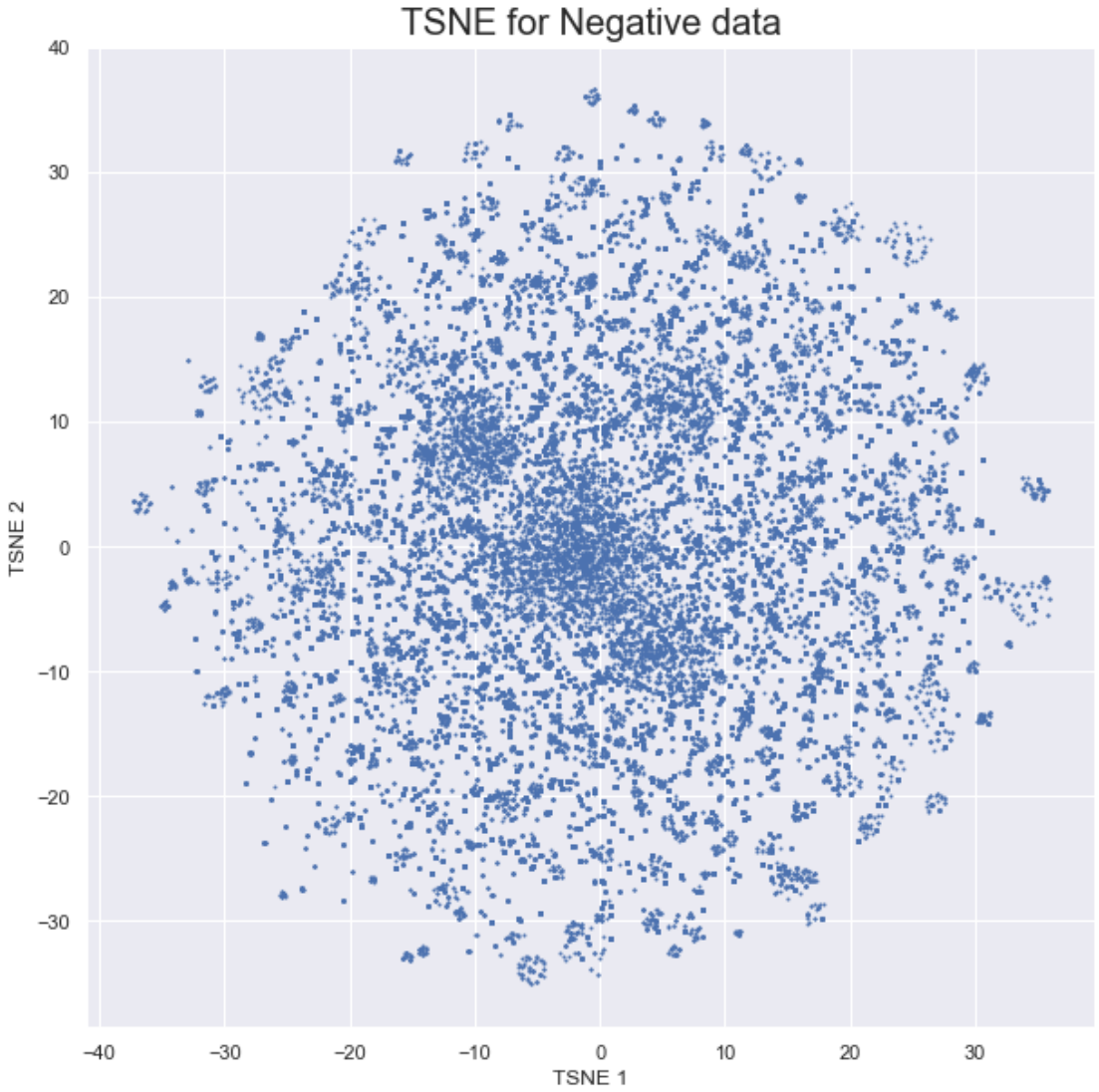
[166]:
```
array([[-13.174172  ,   3.2923772 ,  31.345163  ],
       [-13.853187  , -19.450788  ,  12.798331  ],
       [-13.853195  , -19.451141  ,  12.798458  ],
       ...,
       [ 21.645144  ,   2.8262284 ,  -4.3555226 ],
       [ -5.321156  ,  -0.73769414, -28.301693  ],
       [ -9.092838  ,  -9.796622  ,   2.7244713 ]], dtype=float32)
```

[213]:
```
cat = np.concatenate((X_pembedded, X_nembedded), axis=0)
cat
```

[213]:
```
array([[-2.1144800e+00, -2.8192522e-02, -2.3632686e+00],
       [ 2.2176752e+01, -9.4649458e+00, -7.9223080e+00],
       [-2.6372638e+00,  2.6327784e+00, -1.5447168e+00],
       ...,
```

```
        [ 2.1645144e+01,  2.8262284e+00, -4.3555226e+00],
        [-5.3211560e+00, -7.3769414e-01, -2.8301693e+01],
        [-9.0928383e+00, -9.7966223e+00,  2.7244713e+00]], dtype=float32)
```

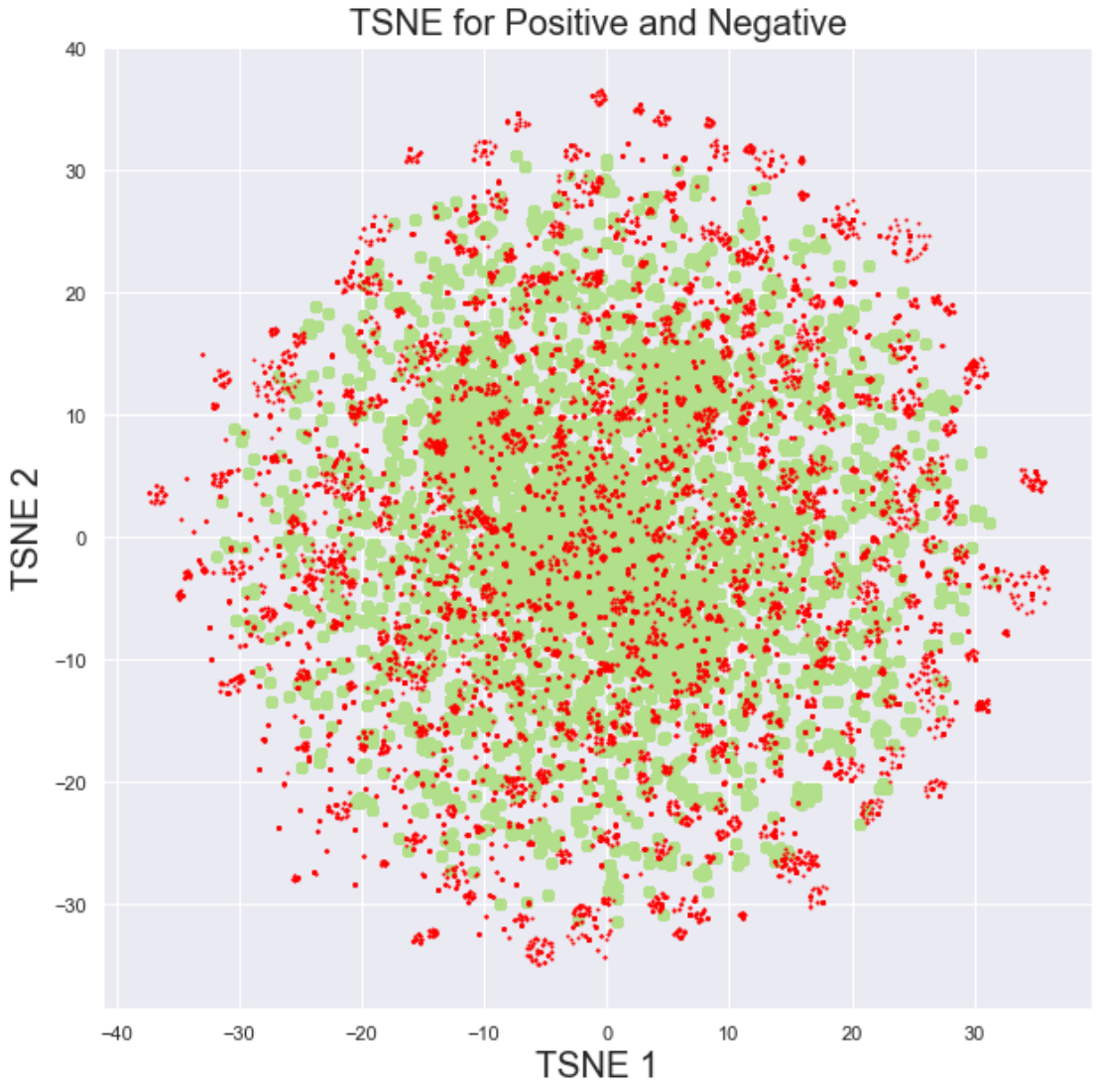
[215]:
```
#Graphing represntation for normal T-SNE
#Currently is takes ~15 to 20mins
dim1 = cat[:, 0]
dim2 = cat[:, 1]
plt.figure(figsize=(10,10))
plt.scatter(dim1, dim2, s=1)
plt.title("TSNE for Negative data", fontsize=20)
plt.xlabel('TSNE 1')
plt.ylabel('TSNE 2')
plt.show()
plt.savefig('TSNE_NRF2')
```



```
<Figure size 432x288 with 0 Axes>
```

[170]:
```
#Graphing represntation for normal T-SNE
#Currently is takes ~15 to 20mins

plt.figure(figsize=(10,10))
plt.scatter(X_pembedded[:, 0], X_pembedded[:, 1], c='#b2df8a', label = '+ve')
plt.scatter(X_nembedded[:, 0], X_nembedded[:, 1], color='red', label = '-ve',␣
 ↪s=1)
plt.title("TSNE for Positive and Negative", fontsize=20)
plt.xlabel('TSNE 1', fontsize=20)
plt.ylabel('TSNE 2', fontsize=20)
plt.show()
#plt.savefig('TSNE_NRF2_Positive')
```

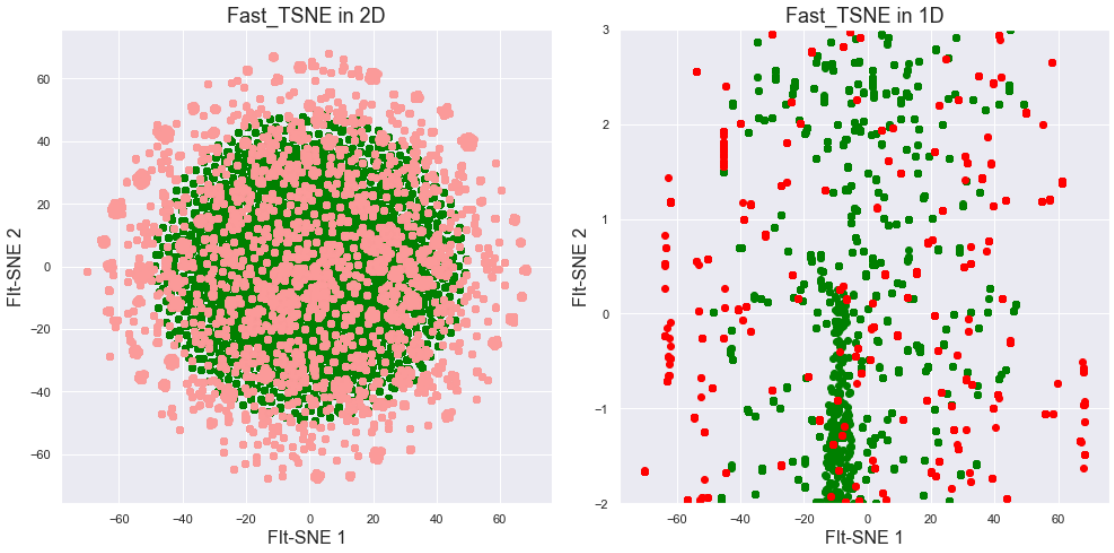


[171]:
```
import numpy as np
import pylab as plt
import seaborn as sns; sns.set()

# change the path for fast-TSNE!
import sys; sys.path.append('/Users/kalyanidhusia/Downloads/FIt-SNE-master/')
from fast_tsne import fast_tsne
```

[179]:
```
Z1 = fast_tsne(X_pembedded)

Z2 = fast_tsne(X_nembedded)
```

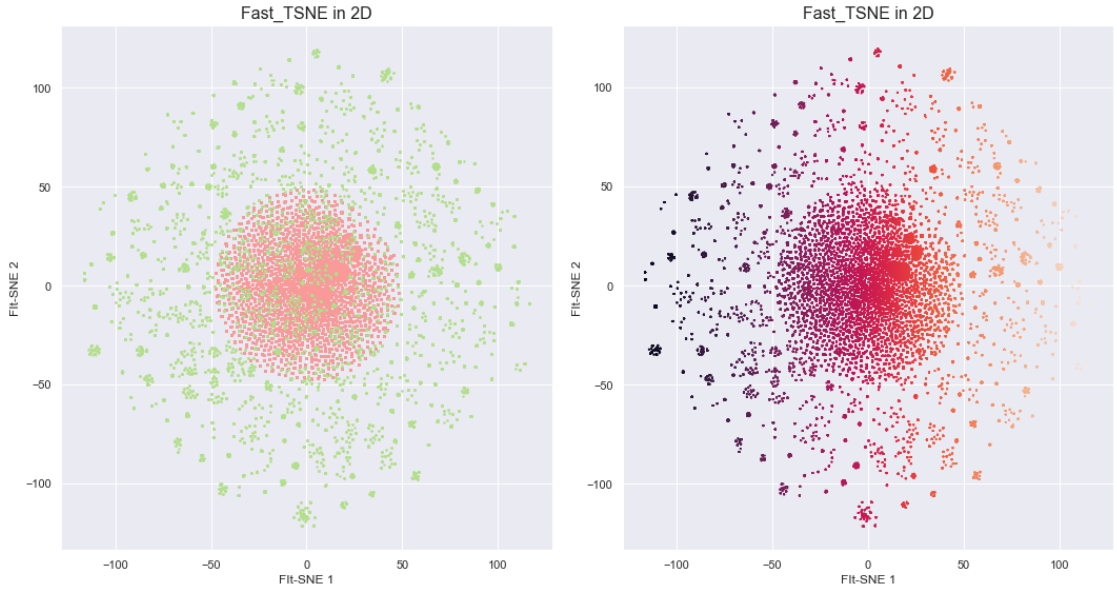


[264]:
```
catFITSNE = np.concatenate((Z1, Z2), axis=0)
catFITSNE
```

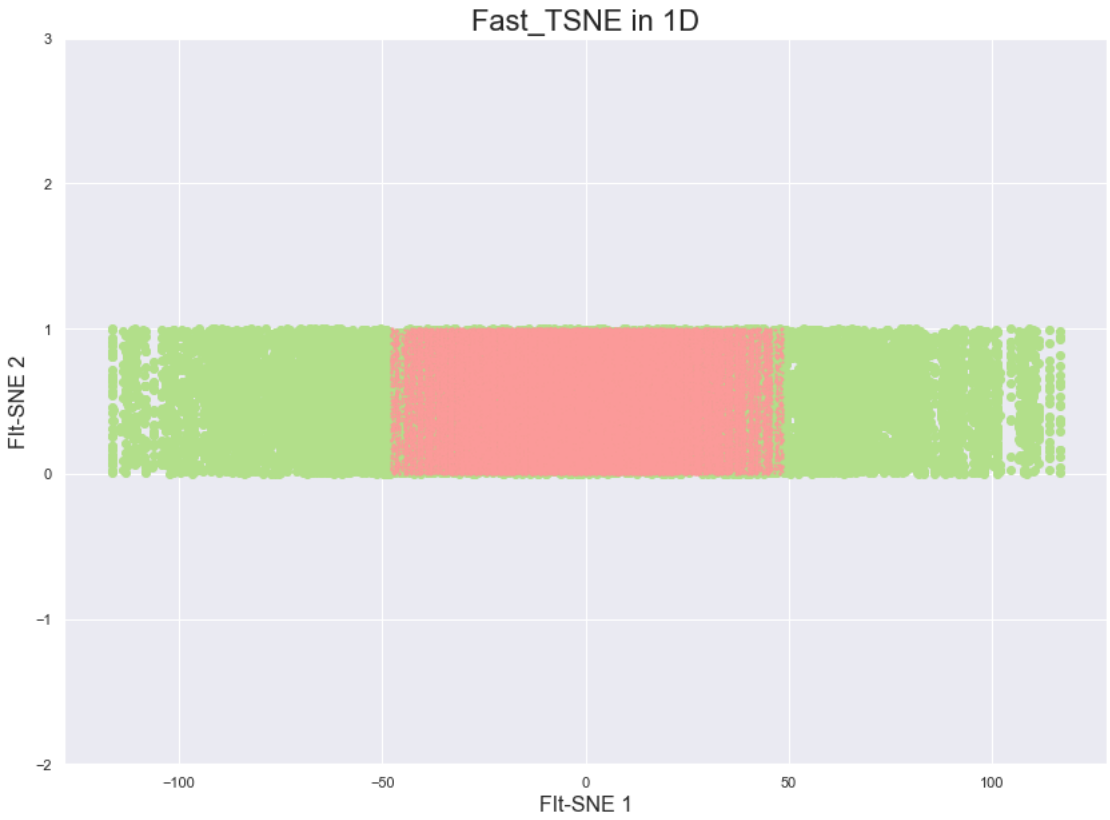
[264]:
```
array([[ 23.25843808,   6.15583781],
       [-32.08442197,  24.02644805],
       [ 12.17818374,  10.03877508],
       ...,
       [ 88.16826289,  11.23938027],
       [ 24.46635059,  53.8562224 ],
       [-49.84789625, -54.63553186]])
```

[262]:
```
plt.figure(figsize=(15,8))
plt.subplot(121)
plt.scatter(Z1[:,0], Z1[:,1],c='#fb9a99',label='positive', s=1)
plt.scatter(Z2[:,0], Z2[:,1], c='#b2df8a',label='negitive', s=1)
plt.title('Fast_TSNE in 2D', fontsize=16)
plt.xlabel('FIt-SNE 1')
```

```
plt.ylabel('FIt-SNE 2')
plt.subplot(122)
plt.scatter(catFITSNE[:,0], catFITSNE[:,1],c=catFITSNE[:,0], s=1)
plt.title('Fast_TSNE in 2D', fontsize=16)
plt.xlabel('FIt-SNE 1')
plt.ylabel('FIt-SNE 2')
plt.tight_layout()
```



[266]:
```
plt.figure(figsize=(14,10))
plt.scatter(Z2[:,0], np.random.uniform(size=Z2.shape[0]), c='#b2df8a',label =␣
↪'-ve')
plt.scatter(Z1[:,0], np.random.uniform(size=Z1.shape[0]), c='#fb9a99', label =␣
↪'+ve', s=1)#positive data in pink
plt.ylim([-2,3])
plt.title('Fast_TSNE in 1D', fontsize=22)
plt.xlabel('FIt-SNE 1', fontsize=16)
plt.ylabel('FIt-SNE 2', fontsize=16)
plt.show()
```

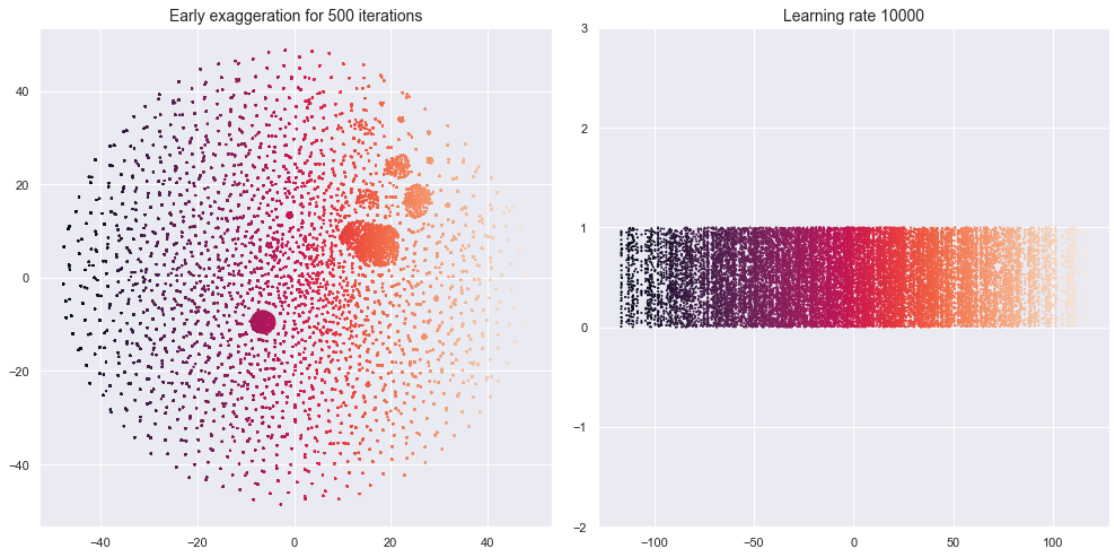


[180]:
```
# Two classes are separated into two parts on the above figure.
# This can be fixed if one uses stronger/longer early exaggeration (e.g.␣
 ↪stop_early_exag_iter=500)
# or higher learning rate (e.g. learning rate=1000)

Z1 = fast_tsne(X_pembedded, stop_early_exag_iter=500, initialization=pca_result)

Z2 = fast_tsne(X_nembedded, learning_rate=10000, initialization=pca_result)

plt.figure(figsize=(14,7))
plt.subplot(121)
plt.scatter(Z1[:,0], Z1[:,1], c=Z1[:,0], s=1)
plt.title('Early exaggeration for 500 iterations', fontsize=14)
plt.subplot(122)
plt.scatter(Z2[:,0], np.random.uniform(size=Z2.shape[0]), c=Z2[:,0], s=1)
plt.ylim([-2,3])
plt.title('Learning rate 10000', fontsize=14)
plt.tight_layout()
```



[185]:
```
plt.figure(figsize=(14,7))
plt.subplot(121)
plt.scatter(Z1[:,0], Z1[:,1], s=1, c='#b2df8a')
plt.scatter(Z2[:,0], Z2[:,1], s=1)
plt.title('Early exaggeration for 500 iterations', fontsize=14)
plt.subplot(122)
plt.scatter(Z2[:,0], np.random.uniform(size=Z2.shape[0]), c=Z2[:,0], s=1)
plt.ylim([-2,3])
plt.title('Learning rate 10000', fontsize=14)
plt.tight_layout()
```

[176]:
```
#Also takes a bit (even though they claim it is faster than TSNE ~ <10m mins)
Posumapped = umap.UMAP().fit_transform(Peuc_ohe)
Posumapped
```

```
/Users/kalyanidhusia/anaconda3/lib/python3.6/site-packages/umap/spectral.py:229:
UserWarning: Embedding a total of 55 separate connected components using meta-
embedding (experimental)
  n_components
```

[176]:
```
array([[ 19.277739 ,  -5.5177336],
       [  4.8800936,  -0.6964975],
       [ 19.265244 ,  -5.323438 ],
       ...,
       [  9.944127 , -10.583306 ],
       [ 20.865965 ,   7.9038887],
       [-16.938892 ,  -1.6059988]], dtype=float32)
```

[177]:
```
#Also takes a bit (even though they claim it is faster than TSNE ~ <10m mins)
Negumapped = umap.UMAP().fit_transform(Neuc_ohe)
Negumapped
```

[177]:
```
array([[-2.6379023, 18.61463  ],
       [ 1.7651438, -3.989851 ],
       [ 1.791181 , -4.0574775],
       ...,
       [-4.190588 , 13.3563175],
       [ 3.6450846, -0.8570059],
       [ 0.8062493, -1.4134121]], dtype=float32)
```

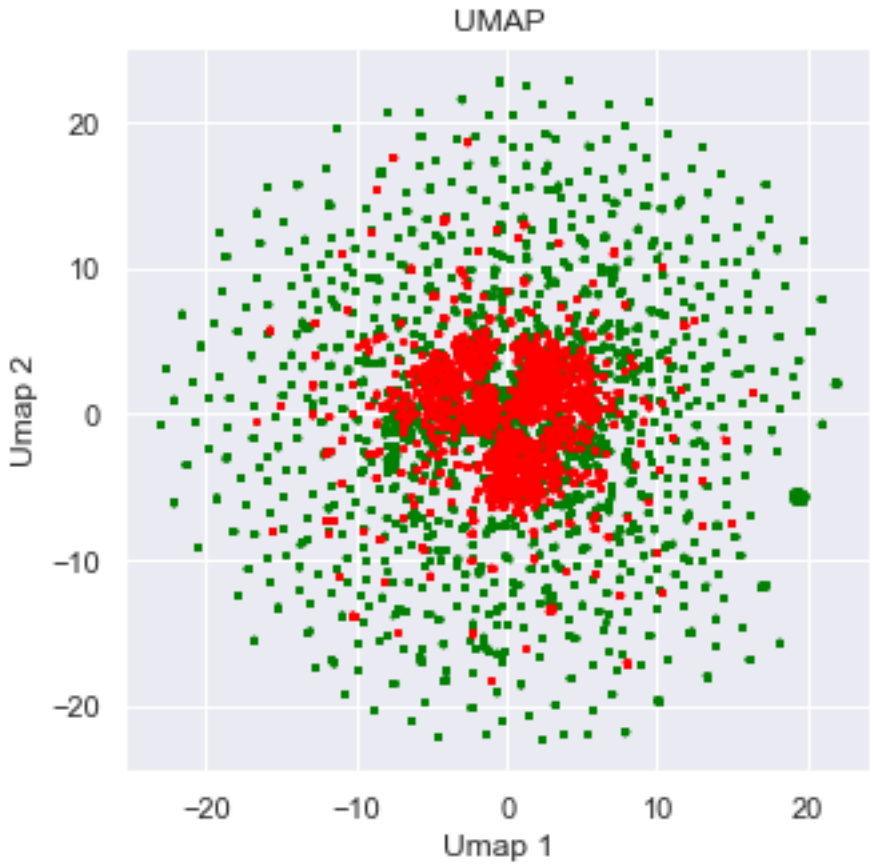
[268]:
```
catumap = np.concatenate((Posumapped, Negumapped), axis=0)
catumap
```

[268]:
```
array([[19.277739 , -5.5177336],
       [ 4.8800936, -0.6964975],
       [19.265244 , -5.323438 ],
       ...,
       [-4.190588 , 13.3563175],
       [ 3.6450846, -0.8570059],
       [ 0.8062493, -1.4134121]], dtype=float32)
```

[278]:
```
#Graphing example
umap1 = umapped[:, 0]
umap2 = umapped[:, 1]

plt.figure(figsize=(5,5))
plt.scatter(Posumapped[:, 0],Posumapped[:, 1], c='green' , label='+ve', s=1)
plt.scatter(Negumapped[:, 0],Negumapped[:, 1], c='red' , label='+ve', s=1)
plt.title("UMAP")
plt.xlabel('Umap 1')
plt.ylabel('Umap 2')
```

```
plt.show()
#plt.savefig('UMAP_NRF2_Positive')
```

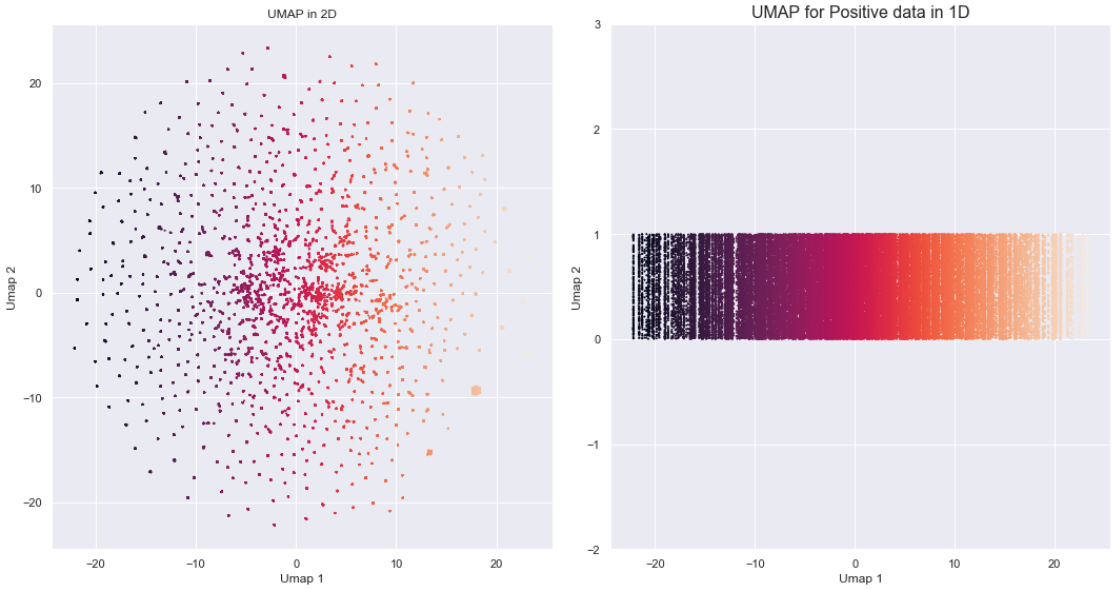


[45]:
```
#Making the kernel more heavy-tailed
Z1 = fast_tsne(umapped, stop_early_exag_iter=500, initialization=pca_result)

Z2 = fast_tsne(umapped, learning_rate=10000, initialization=pca_result, df=.5)
#df: double
#Controls the degree of freedom of t-distribution. Must be positive. The actual␣
 ↪degree of
#freedom is 2*df-1. The standard t-SNE choice of 1 degree of freedom␣
 ↪corresponds to df=1.
#Large df approximates Gaussian kernel. df<1 corresponds to heavier tails,␣
 ↪which can often
#resolve substructure in the embedding. See Kobak et al. (2019) for details.␣
 ↪Default is 1.0.

plt.figure(figsize=(15,8))
plt.subplot(121)
plt.scatter(umap1, umap2, c=umap1 , s=1)
```

```
plt.title('UMAP in 2D')
plt.xlabel('Umap 1')
plt.ylabel('Umap 2')
plt.subplot(122)
plt.scatter(umap2, np.random.uniform(size=umap2.shape[0]), c=umap2, s=1)
plt.ylim([-2,3])
plt.title('UMAP for Positive data in 1D', fontsize=16)
plt.xlabel('Umap 1')
plt.ylabel('Umap 2')
plt.tight_layout()
#plt.savefig('UMAP_NRF2_Positive_kernel')
```



[277]:
```
#Graphing example
umap1 = catumap
umap2 = catumap
plt.figure(figsize=(5,5))
plt.scatter(umap1[:,0], umap2[:,1], c=umap1 , s=1)
plt.title("UMAP for collective data")
plt.xlabel('Umap 1')
plt.ylabel('Umap 2')
plt.show()
#plt.savefig('Umap_NRF2_collective')
```

```
␣
↪------------------------------------------------------------------------
```

```
        IndexError                                Traceback (most recent call␣
↪last)

        <ipython-input-277-98977d4bee97> in <module>
          3 umap2 = catumap
          4 plt.figure(figsize=(5,5))
    ----> 5 plt.scatter(umap1[:,0], umap2[:,1], c=umap1 , s=1)
          6 plt.title("UMAP for collective data")
          7 plt.xlabel('Umap 1')


        IndexError: too many indices for array


<Figure size 360x360 with 0 Axes>
```

[ ]: