Homework 5

Ben Tward

2025-04-09

Question 1

a.

According to the Spectral Theorem, any symmetric matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ can be factored into the product of matrices $\mathbf{VDV}^T$ where $\mathbf{V}$ is orthonormal and $\mathbf{D}$ is a diagonal matrix of the eigenvalues of $\mathbf{A}$. We can also express this as a summation $\sum_{i=1}^{n} \lambda_i \mathbf{v}_i \mathbf{v}_i^T$.

We can also say that for any matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{A}^T\mathbf{A}$ is symmetric and positive semi-definite (the eigenvalues $\{\lambda_i\}_{i=1}^{n} > 0$. Define $\sigma_i^2 = \lambda_i$, and these values must be real by definition. From this, we get that $\mathbf{A}^T\mathbf{A} = \mathbf{VDV}^T$. We also know $\mathbf{A}^T\mathbf{A}\mathbf{v}_i = \lambda_i \mathbf{v}_i = \sigma_i^2 \mathbf{v}_i$. We can kind of rearrange this and define a new eigenvector of $\mathbf{AA}^T$ which is $\mathbf{u}_i = \frac{\mathbf{A}\mathbf{v}_i}{\sigma_i}$.

If we make a matrix Σ which consists of the singular values σ on the diagonal, we can express $\mathbf{U} = \mathbf{AV\Sigma}^{-1}$. Rearranging this, we get $\mathbf{A} = \mathbf{U\Sigma V}^T$ which is precisely our SVD. We can also see through this process that all these calculations are concrete and therefore the SVD solution is unique.

b.

We know that a rank-k approximation of $\mathbf{A}$ is $\mathbf{A}_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$. We want show that $\mathbf{A}_k$ minimizes the Frobenius norm (or equivalently the squared norm):

$||\mathbf{A} - \mathbf{A}_k||_F^2 = ||\sum_{i=1}^{n} \sigma_i \mathbf{u}_i \mathbf{v}_i^T - \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^T||_F^2 = ||\sum_{i=k+1}^{n} \sigma_i \mathbf{u}_i \mathbf{v}_i^T||_F^2 = \sum_{i=k+1}^{n} \sigma_i^2$

Because $\mathbf{u}_i$ and $\mathbf{v}_i^T$ are orthogonal so the σ_i^2 terms add.

If we compare this to any arbitrary matrix $\mathbf{B}$ with the constraint $rank(\mathbf{B}) = k$, we must show that $||\mathbf{A} - \mathbf{A}_k||_F^2 \leq ||\mathbf{A} - \mathbf{B}||_F^2$

If we do a k-rank approximation for $\mathbf{B}$, we get terms that cannot cancel and we are left with the result that $\mathbf{A}_k$ is the best approximation to minimize the Frobenius norm.

Question 2

a)

```
data = read.csv('CityDistances.csv')
print(data)
```

```
##      City...City Salt.Lake.City Ann.Arbor  Tokyo Addis.Ababa Cape.Town
## 1 Salt Lake City            0.0    1452.9 5473.4      8520.8    9702.6
## 2      Ann Arbor         1452.9       0.0 6389.5      7368.4    8312.9
## 3          Tokyo         5473.4    6389.5    0.0      6465.3    9158.2
## 4    Addis Ababa         8520.8    7368.4 6465.3         0.0    3252.1
```

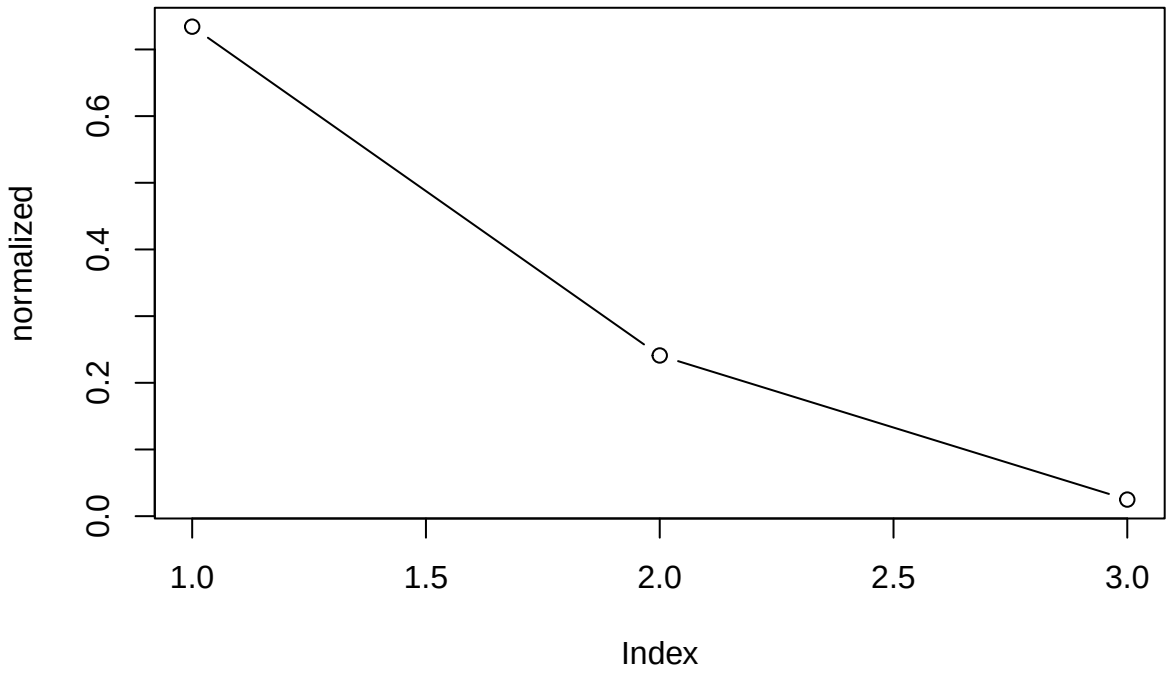
```
## 5      Cape Town         9702.6    8312.9 9158.2      3252.1       0.0
## 6    Los Angeles          580.5    1945.5 5472.2      9099.9    9975.2
## 7  New York City         1968.0     515.7 6737.0      6959.3    7806.8
##   Los.Angeles New.York.City
## 1       580.5        1968.0
## 2      1945.5         515.7
## 3      5472.2        6737.0
## 4      9099.9        6959.3
## 5      9975.2        7806.8
## 6         0.0        2448.8
## 7      2448.8           0.0
```

b)

```
mds = function(D, k) {
  D = as.matrix(D)
  n = dim(D)[1]
  e = as.matrix(rep(1, n), n, 1)
  I = diag(nrow=n)
  H = I - ((1/n) * (e %*% t(e)))
  B = -.5 * (H %*% D %*% H)
  eigenB = eigen(B)
  Uk = eigenB$vectors[,1:k]
  Lambdak = eigenB$values[1:k]
  Xtilde = Uk %*% diag(Lambdak)
  return(list(Xtilde = Xtilde, eigs = Lambdak))
}
```

c)

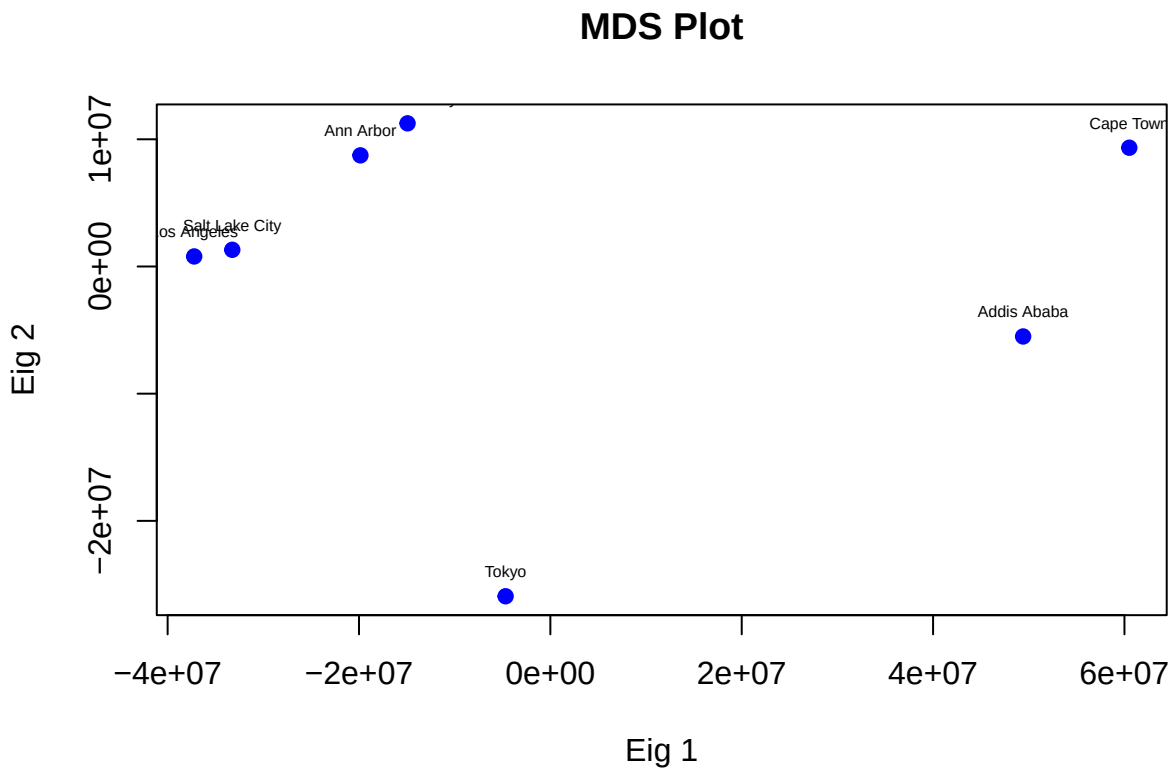
```
D = (data[,-1])^2
mdsD = mds(D, 3)
eigs = mdsD$eigs
normalized = eigs / sum(eigs)
plot(normalized, type='b')
```



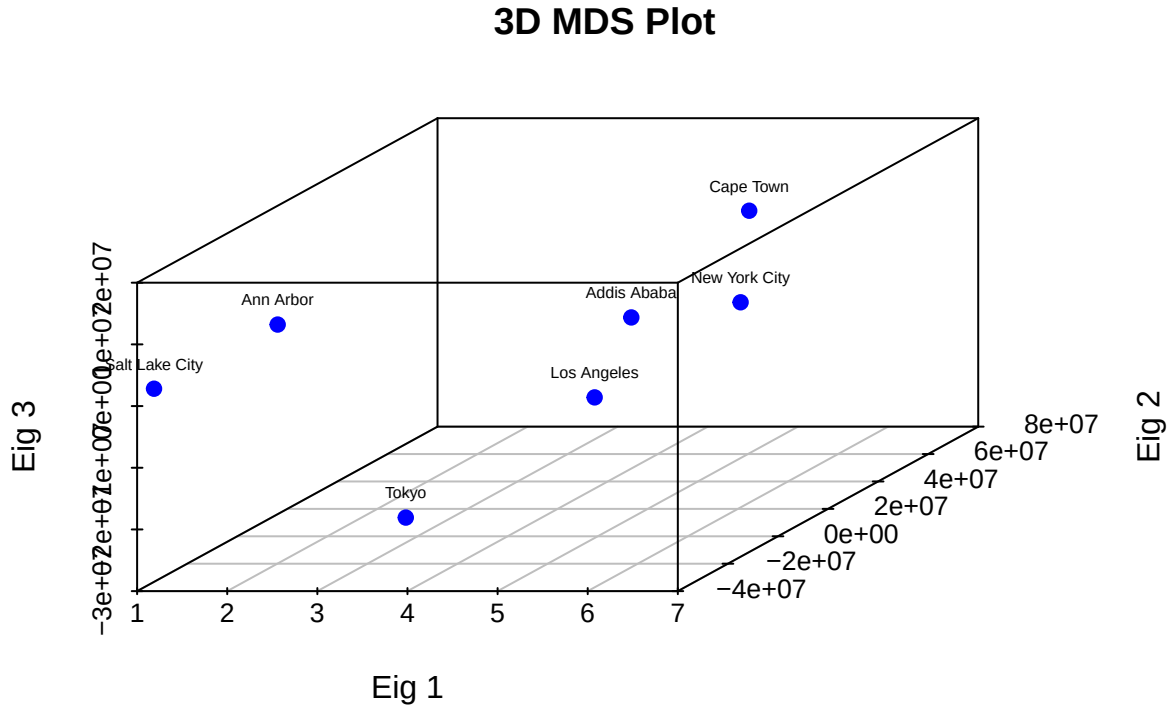
I do not see any negative eigenvalues in my data. However, I read online that it is possible for there to be negative eigenvalues, which is usually a sign that MDS is inappropriate on that data. If our distance matrix $\mathbf{D}^{\mathbf{X}}$ is computed using Euclidian distance, then $\mathbf{B}^{\mathbf{X}}$ is guaranteed to be positive semi-definite.

d)

```
names = data[,1]
mdsD = mds(D, 2)
X = mdsD$Xtilde
plot(X[,1], X[,2], xlab = "Eig 1", ylab = "Eig 2",
     main = "MDS Plot", pch = 19, col = "blue")
text(X[,1], X[,2], labels = names, pos = 3, cex = 0.5)
```




```
s3d = scatterplot3d(X, xlab = "Eig 1", ylab = "Eig 2", zlab = "Eig 3",
                    pch = 19, color = "blue", main = "3D MDS Plot")

coords = s3d$xyz.convert(X)
text(coords$x, coords$y, labels = names, pos = 3, cex = 0.5)
```



I notice that when looking at the 2-dimensional representation, we can see a distinct separation of cities in the USA versus Asia versus Africa. So that representation seems good. However, when I look at the 3-dimensional representation, the clusterings are more difficult to perceive. It might be the scaling of the plot or the challenge to put 3d coordinates on a 2d screen.

In [63]:
```
import scanpy as sc
import pandas as pd
import numpy as np
from pathlib import Path
import matplotlib.pyplot as plt
from matplotlib.image import imread
import seaborn as sns
import os
import json


import matplotlib.colorbar as mplcb
import matplotlib.cm as mplcm

#sc.logging.print_header()
#sc.settings.verbosity = 3
```

In [64]:
```
def read_visium(path, library_id='stx'):
    r"""Read 10x-Genomics-formatted visum dataset.
    """
    path = Path(path)
    matrix_path = path / "filtered_feature_bc_matrix/"
    adata = sc.read_10x_mtx(matrix_path)

    adata.uns["spatial"] = dict()
    adata.uns["spatial"][library_id] = dict()

    tissue_positions_file = (
        path / "spatial/tissue_positions.csv"
        if (path / "spatial/tissue_positions.csv").exists()
        else path / "spatial/tissue_positions_list.csv"
    )
    files = dict(
        tissue_positions_file=tissue_positions_file,
        scalefactors_json_file=path / "spatial/scalefactors_json.json",
        hires_image=path / "spatial/tissue_hires_image.png",
        lowres_image=path / "spatial/tissue_lowres_image.png",
    )

    # check if files exists, continue if images are missing
    for f in files.values():
        if not f.exists():
            if any(x in str(f) for x in ["tissue_hires_image", "tissue_lowres_image
                logg.warning(
                    f"You seem to be missing an image file.\nCould not find {f}."
                )
            else:
                msg = f"Could not find {f}"
                raise OSError(msg)

    adata.uns["spatial"][library_id]["images"] = dict()

    for res in ["hires", "lowres"]:
        image_path = str(files[f"{res}_image"])
```

```
        adata.uns["spatial"][library_id]["images"][res] = imread(image_path)

        # read json scalefactors
        adata.uns["spatial"][library_id]["scalefactors"] = json.loads(
            files["scalefactors_json_file"].read_bytes()
        )

        # read coordinates
        positions = pd.read_csv(
            files["tissue_positions_file"],
            header=0 if tissue_positions_file.name == "tissue_positions.csv" else N
            index_col=0,
        )
        positions.columns = [
            "in_tissue",
            "array_row",
            "array_col",
            "pxl_col_in_fullres",
            "pxl_row_in_fullres",
        ]

        adata.obs = pd.merge(
            adata.obs,
            positions,
            how='left',
            left_index=True,
            right_index=True,
        )

        adata.obsm["spatial"] = adata.obs[
            ["pxl_row_in_fullres", "pxl_col_in_fullres"]
        ].to_numpy()
        adata.obs.drop(
            columns=["pxl_row_in_fullres", "pxl_col_in_fullres"],
            inplace=True,
        )

    return adata

fpath = "/HFD14/ST/"
current_directory = os.getcwd()+fpath
print(current_directory)
adata = read_visium(current_directory)
sc.logging.print_memory_usage()
adata
```

```
c:\Users\Ben\OneDrive\Documents\GitHub\STATS-547\ProblemSet5\Starter Code and Data/H
FD14/ST/
Memory usage: current 0.52 GB, difference +0.26 GB
```

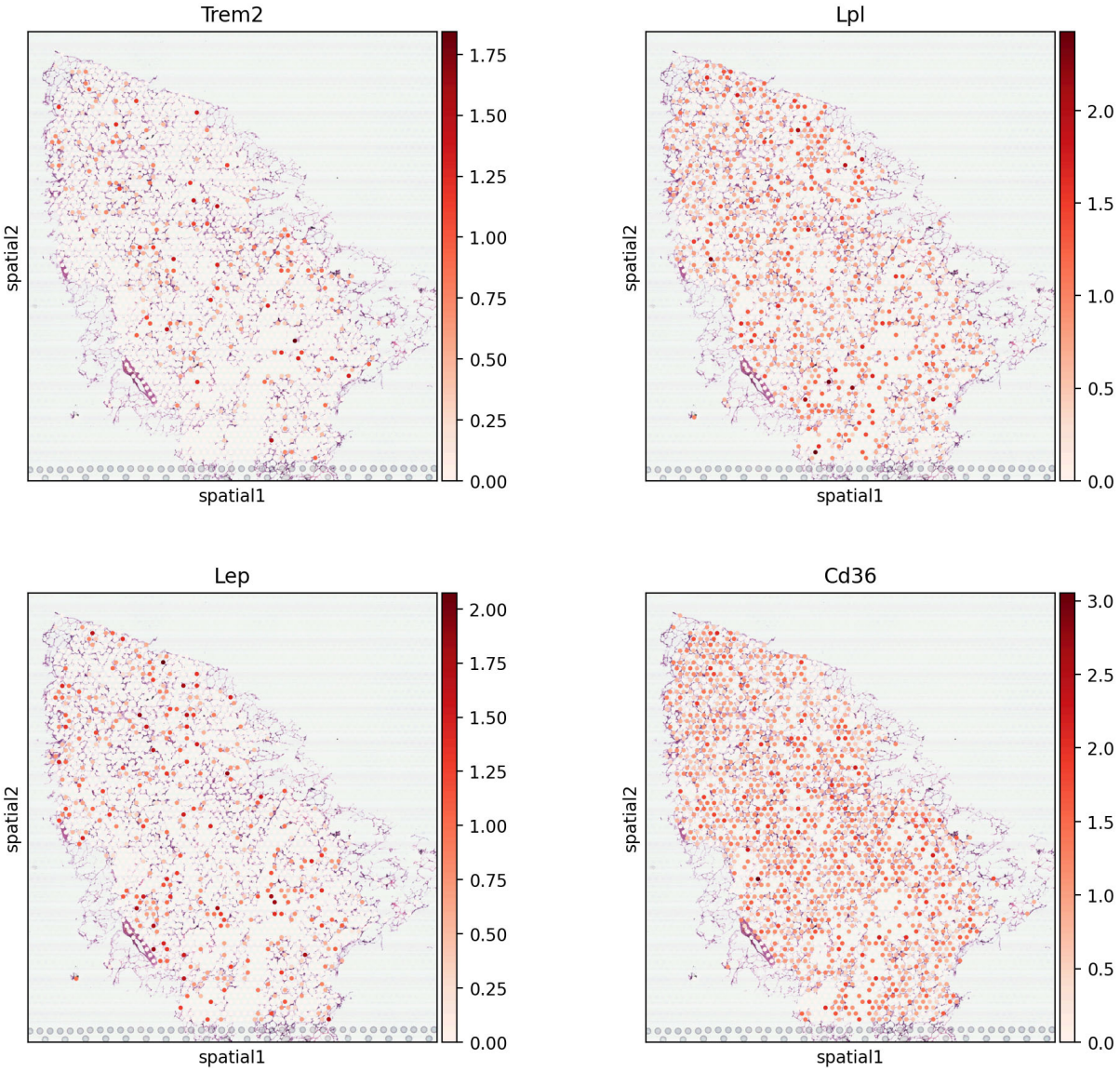
Out[64]:
```
AnnData object with n_obs × n_vars = 1994 × 31053
    obs: 'in_tissue_x', 'array_row_x', 'array_col_x', 'in_tissue_y', 'array_row_
y', 'array_col_y'
    var: 'gene_ids', 'feature_types'
    uns: 'spatial'
    obsm: 'spatial'
```

In [65]:
```
# simple preprocessing
sc.pp.normalize_total(adata)
sc.pp.log1p(adata)
sc.pp.highly_variable_genes(adata, flavor="seurat", n_top_genes=2000)
adata
```

Out[65]:
```
AnnData object with n_obs × n_vars = 1994 × 31053
    obs: 'in_tissue_x', 'array_row_x', 'array_col_x', 'in_tissue_y', 'array_row_
y', 'array_col_y'
    var: 'gene_ids', 'feature_types', 'highly_variable', 'means', 'dispersions',
'dispersions_norm'
    uns: 'spatial', 'log1p', 'hvg'
    obsm: 'spatial'
```

In [66]:
```
plt.rcParams['figure.dpi'] = 200
plt.rcParams['figure.figsize'] = 5, 5
sc.pl.spatial(
    adata, img_key="hires",
    color=["Trem2", 'Lpl', 'Lep', 'Cd36'],
    color_map='Reds',
    ncols=2,
)
```

```
C:\Users\Ben\AppData\Local\Temp\ipykernel_19748\3498164135.py:3: FutureWarning: Use
`squidpy.pl.spatial_scatter` instead.
  sc.pl.spatial(
```



Load the points for a single gene

In [67]:
```
gene = 'Trem2'

df = adata[:, gene].to_df()
print(f"{df.shape=}")

coords = pd.DataFrame(adata.obsm['spatial'], index=df.index, columns=['x', 'y'])
print(f"{coords.shape=}")

# merge in the coordinates
df = pd.merge(
    df,
    coords,
    how='left',
    left_index=True,
    right_index=True,
)
```

```
df.head()
```

```
df.shape=(1994, 1)
coords.shape=(1994, 2)
```

Out[67]:

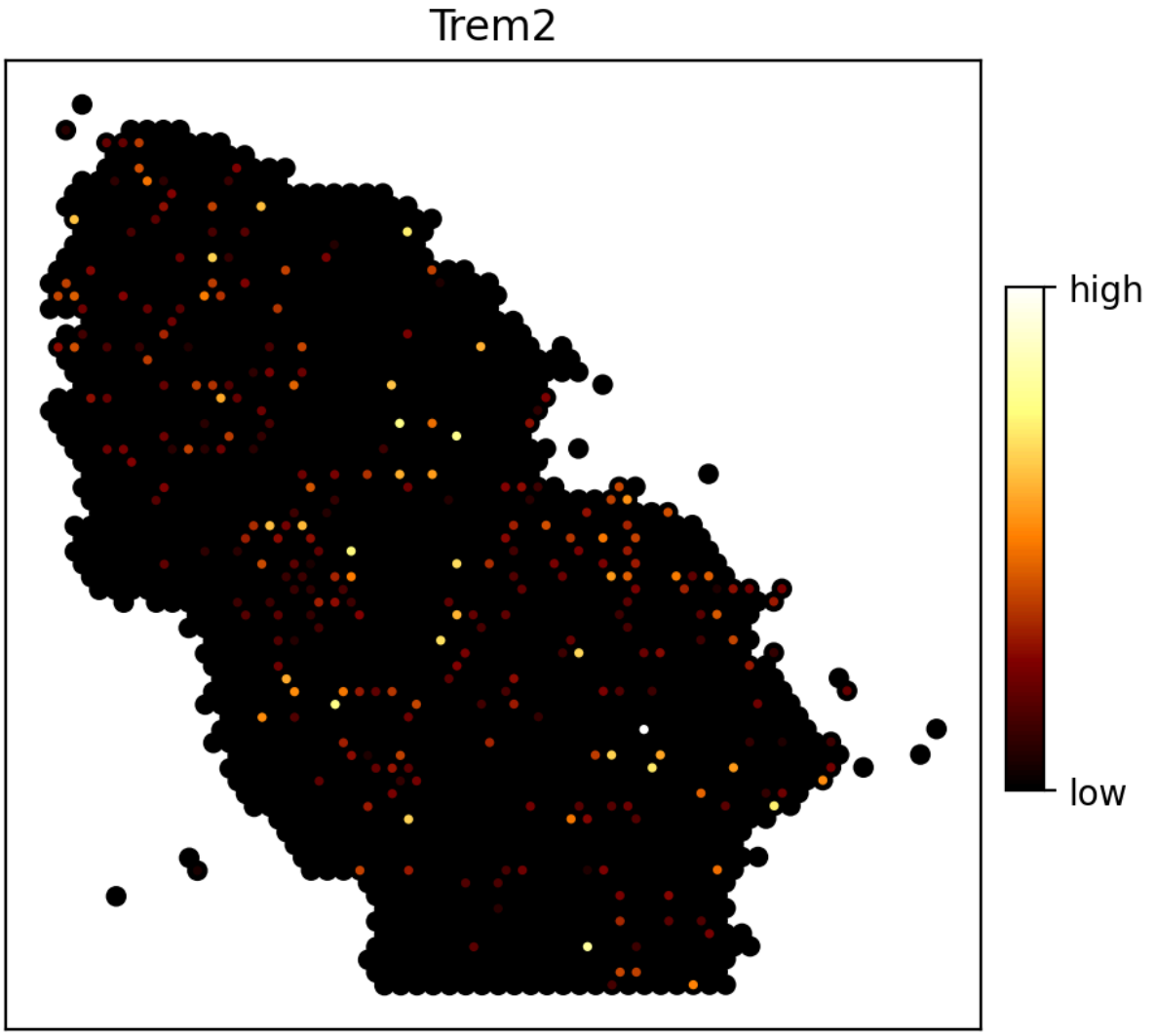
	Trem2	x	y
AAACATTTCCCGGATT-1	0.0	23202	27803
AAACCGGGTAGGTACC-1	0.0	9528	21281
AAACCGTTCGTCCAGG-1	0.0	12306	24722
AAACCTAAGCAGCCGG-1	0.0	20432	29186
AAACCTCATGAAGTTG-1	0.0	7743	19562

In [68]:

```
plt.rcParams['figure.dpi'] = 200
plt.rcParams['figure.figsize'] = 5, 5

cmap = 'afmhot'

sns.scatterplot(
    data=df,
    x='x',
    y='y',
    c='k',
    palette='Reds',
    s=35,
    ec='none'
)

scatter_plot = sns.scatterplot(
    data=df,
    x='x',
    y='y',
    hue=gene,
    ec='none',
    palette=cmap,
    s=7,
    legend=False,
)

plt.gca().invert_yaxis()
plt.yticks([])
plt.xticks([])
plt.ylabel("")
plt.xlabel("")
plt.title(gene)

norm = plt.Normalize(df[gene].min(), df[gene].max())
sm = mplcm.ScalarMappable(norm=norm, cmap=cmap)
sm.set_array([]) # need to set array for colorbar to work

cbar_ax = plt.gcf().add_axes([0.92, 0.3, 0.03, 0.4])  # [left, bottom, width, heigh
cbar = plt.colorbar(sm, cax=cbar_ax, orientation='vertical')
```

```
cbar.set_ticks([df[gene].min(), df[gene].max()])
cbar.set_ticklabels(['low', 'high'])

plt.show()
```

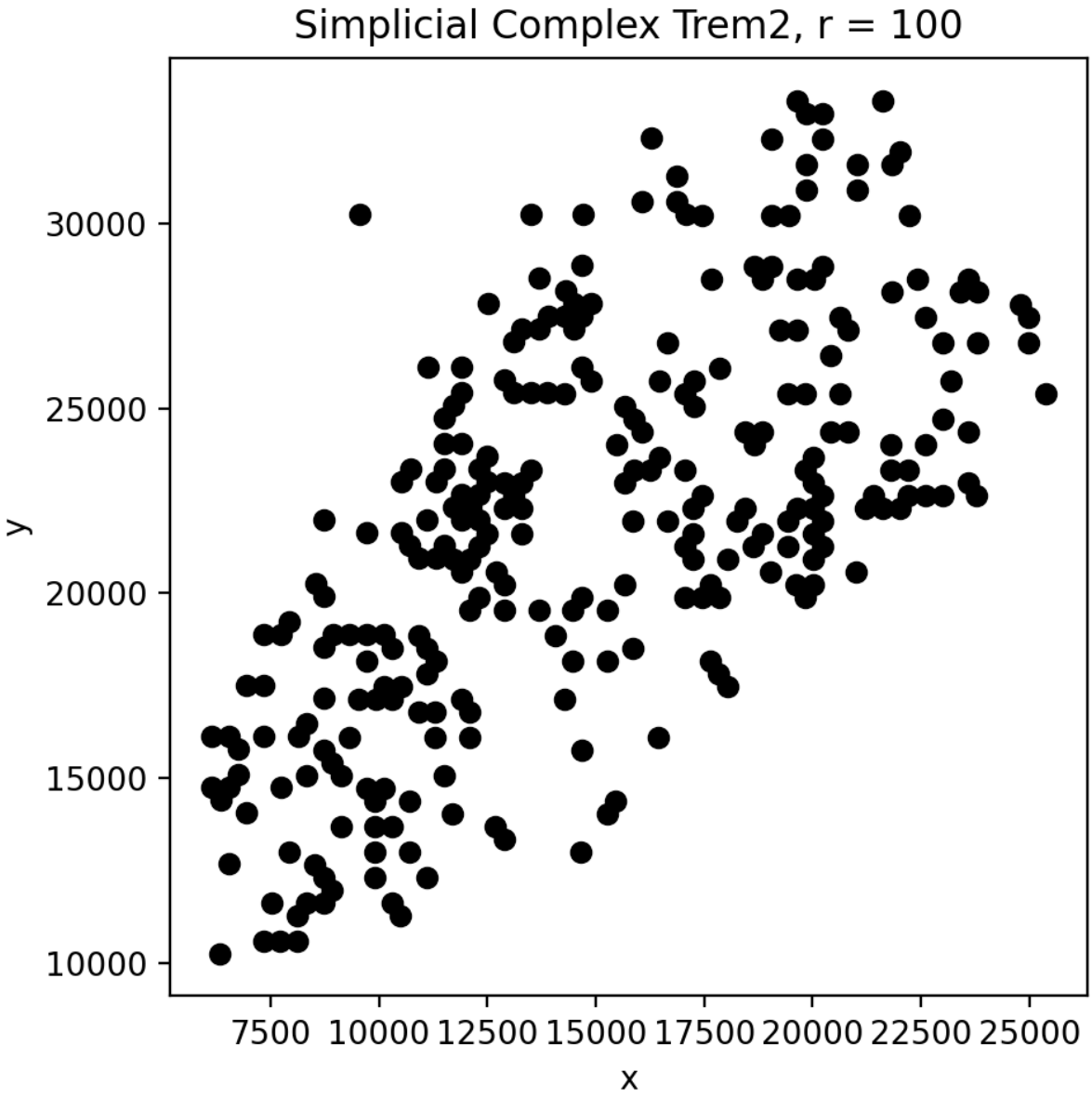
```
C:\Users\Ben\AppData\Local\Temp\ipykernel_19748\3173582000.py:6: UserWarning: Ignori
ng `palette` because no `hue` variable has been assigned.
  sns.scatterplot(
```

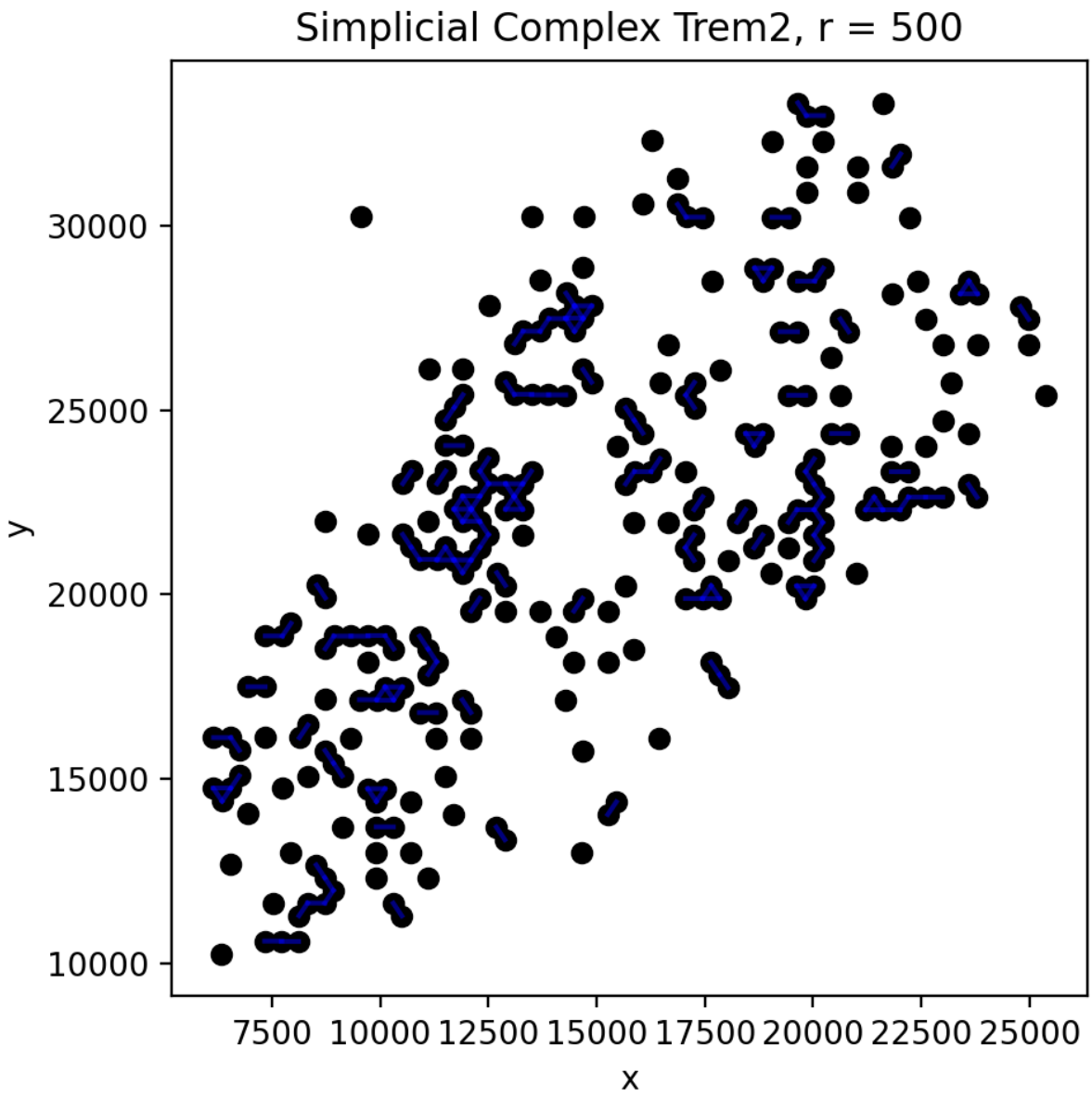


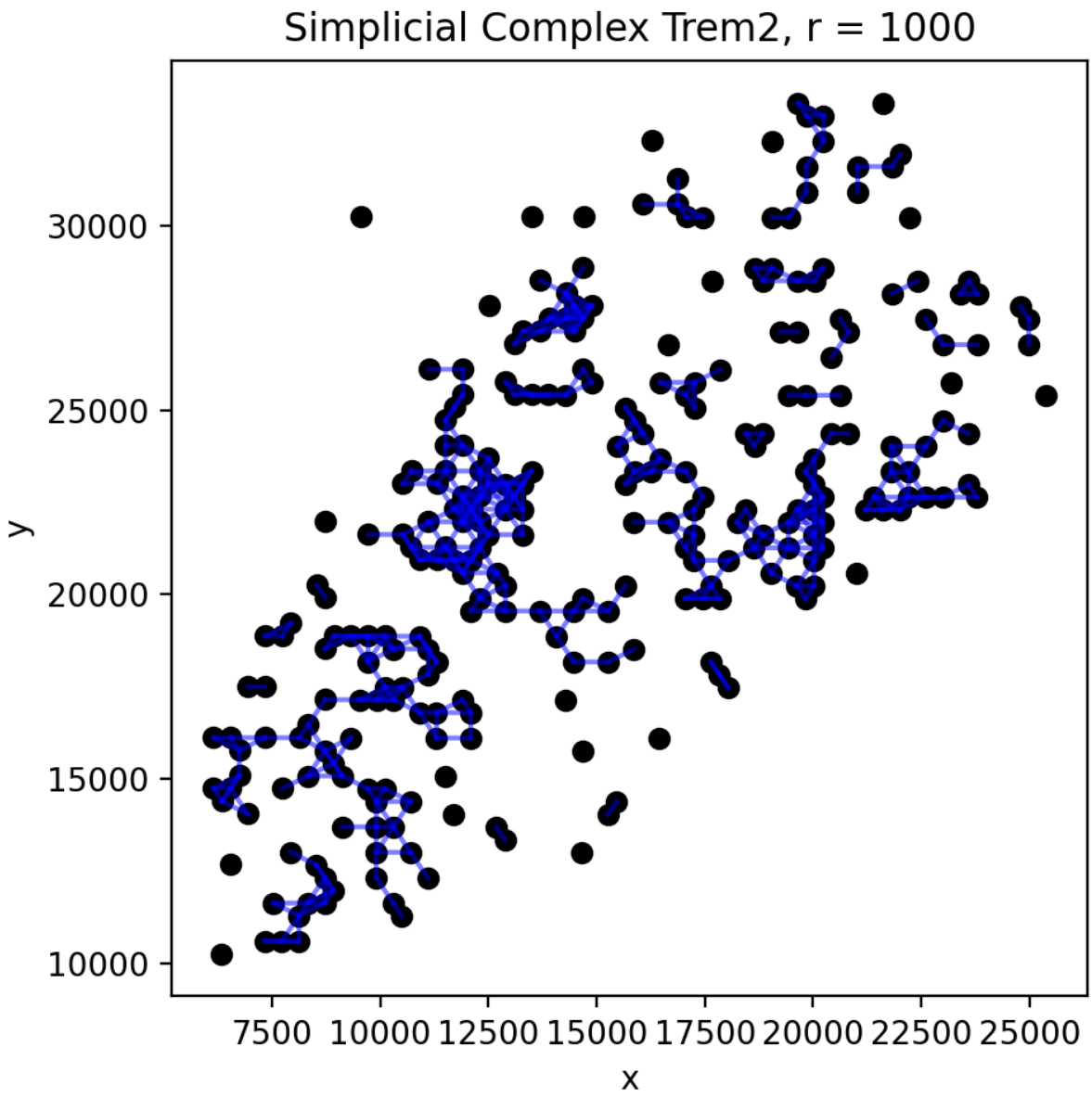
In [69]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial.distance import cdist

def create_simplicial(data, radius, gene):
    points = np.array(data[['x', 'y']])
    dist_matrix = cdist(points, points)

    plt.figure(figsize=(5,5))
    plt.scatter(points[:, 0], points[:, 1], color='black')

    n_points = len(points)
    for i in range(n_points):
        for j in range(i + 1, n_points):
```

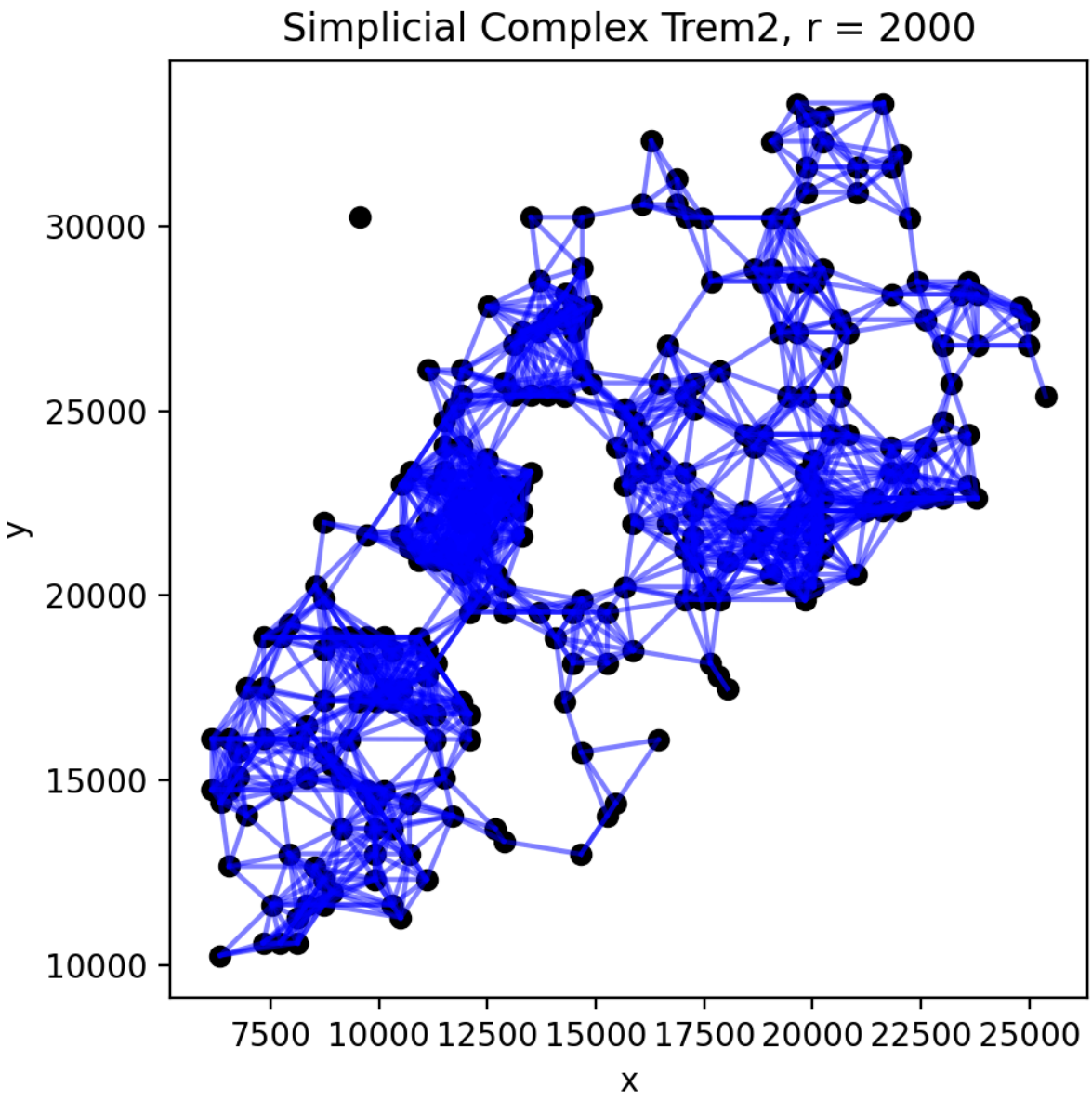
```
            if dist_matrix[i, j] <= radius:
                x_vals = [points[i, 0], points[j, 0]]
                y_vals = [points[i, 1], points[j, 1]]
                plt.plot(x_vals, y_vals, color='blue', alpha=0.5)

    plt.title(f"Simplicial Complex {gene}, r = {radius}")
    plt.xlabel("x")
    plt.ylabel("y")
    plt.tight_layout()

    plt.show()
```

In [70]:
```
#Threshold nonzero
filtered_df = df[df['Trem2'] > 0]
for i in [100, 500, 1000, 2000]:
    create_simplicial(filtered_df, i, "Trem2")
```

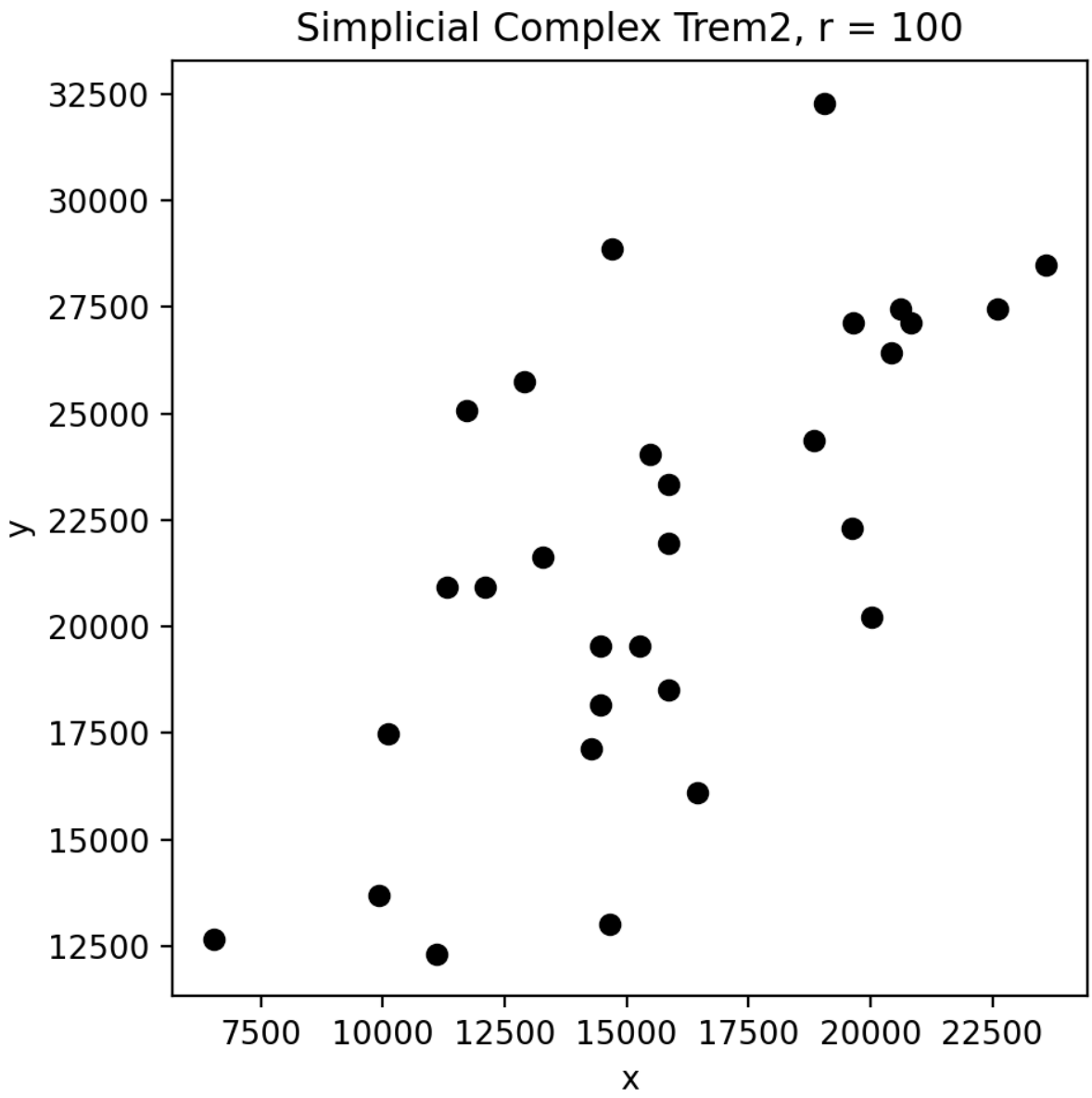


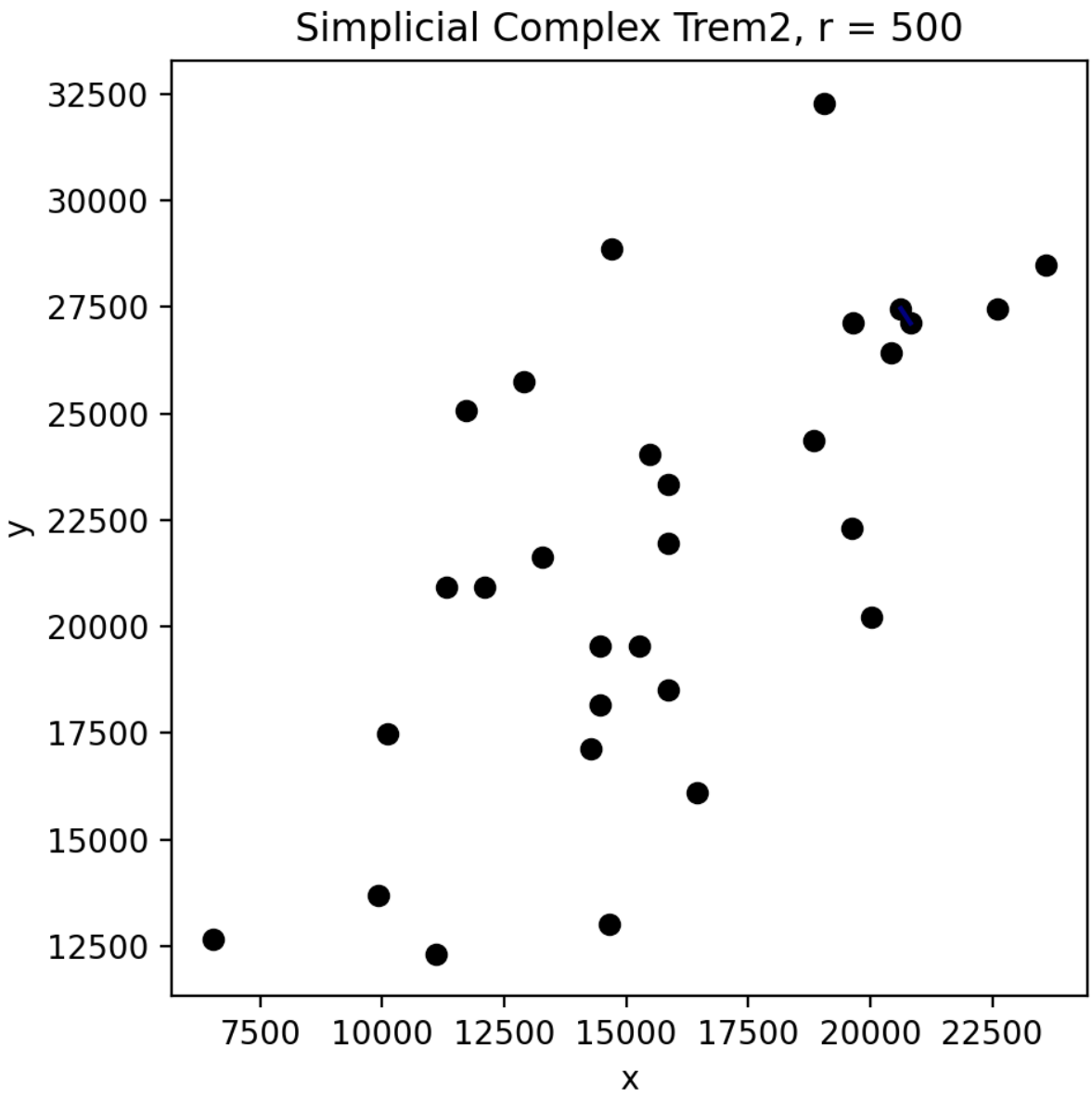

Simplicial Complex Trem2, r = 500
30000
25000
20000
15000
10000
y
7500
10000
12500
15000
17500
20000
22500
25000
x

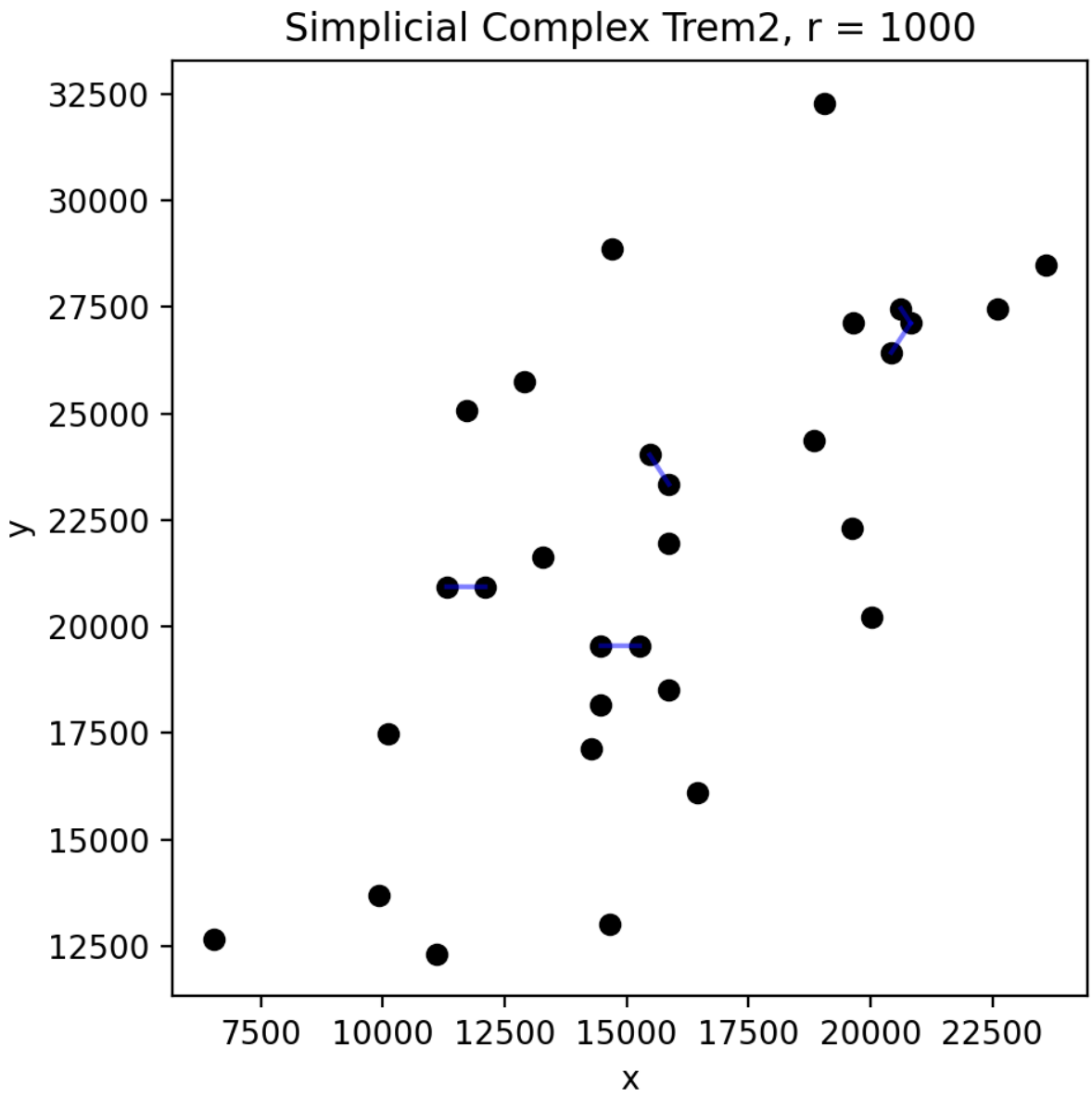

Simplicial Complex Trem2, r = 1000
30000
25000
20000
15000
10000
y
7500
10000
12500
15000
17500
20000
22500
25000
x

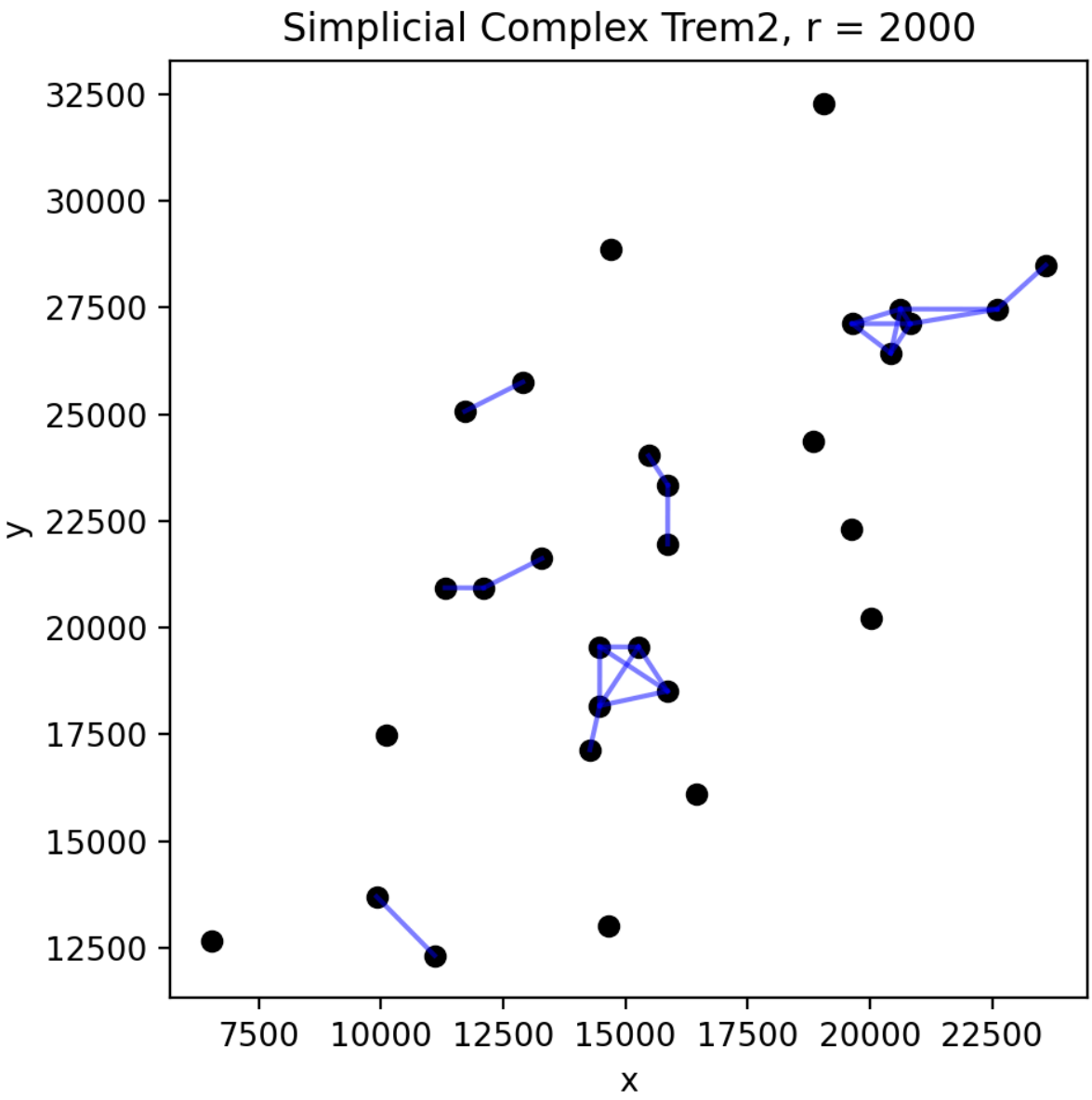


In [71]:
```
#Threshold greater than 1
filtered_df = df[df['Trem2'] > 1]
for i in [100, 500, 1000, 2000]:
    create_simplicial(filtered_df, i, "Trem2")
```


Simplicial Complex Trem2, r = 100
32500
30000
27500
25000
22500
20000
17500
15000
12500
y
7500
10000
12500
15000
17500
20000
22500
x


Simplicial Complex Trem2, r = 500
32500
30000
27500
25000
22500
20000
17500
15000
12500
y
7500
10000
12500
15000
17500
20000
22500
x

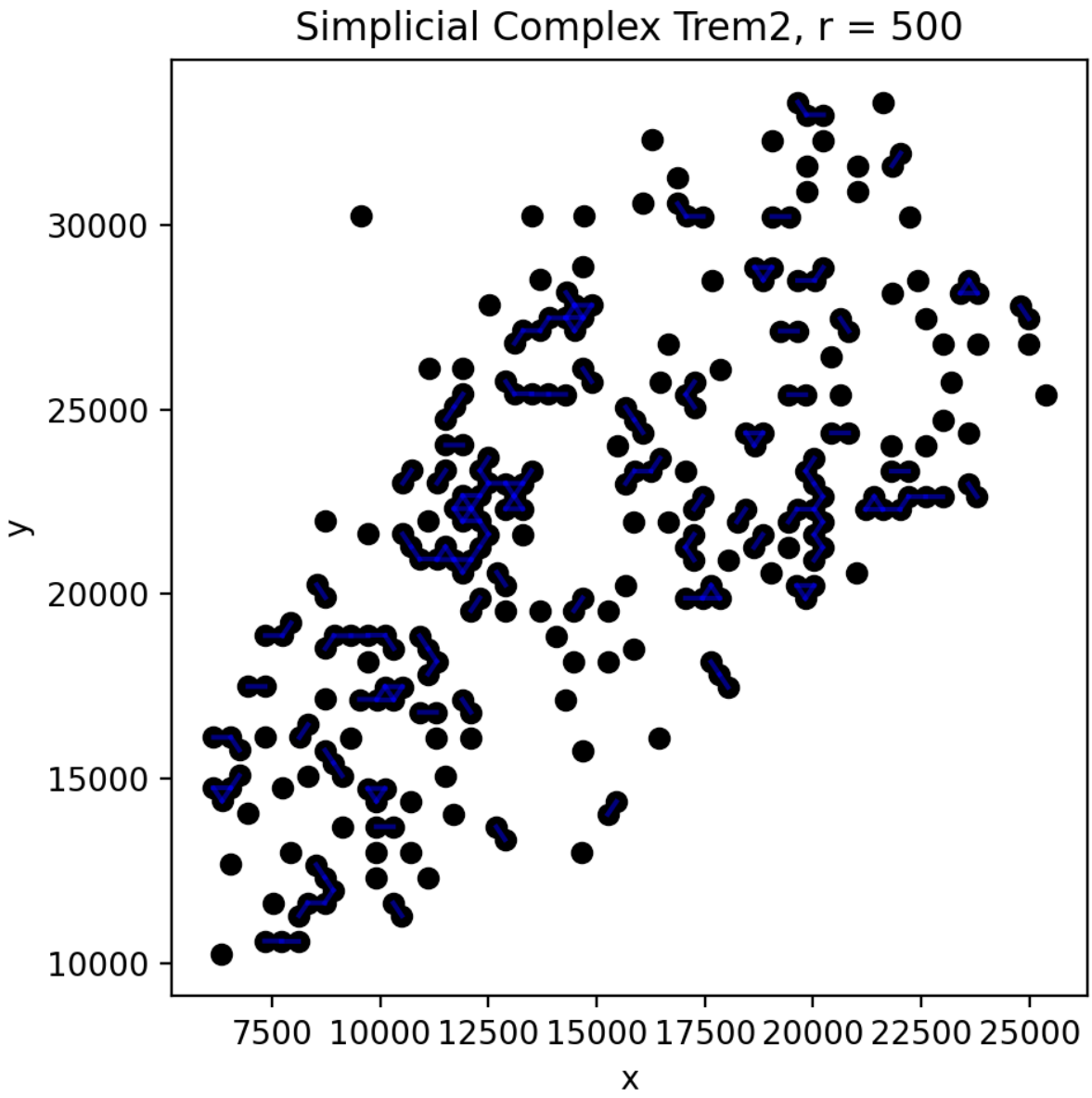

Simplicial Complex Trem2, r = 1000
32500
30000
27500
25000
22500
20000
17500
15000
12500
y
7500
10000
12500
15000
17500
20000
22500
x

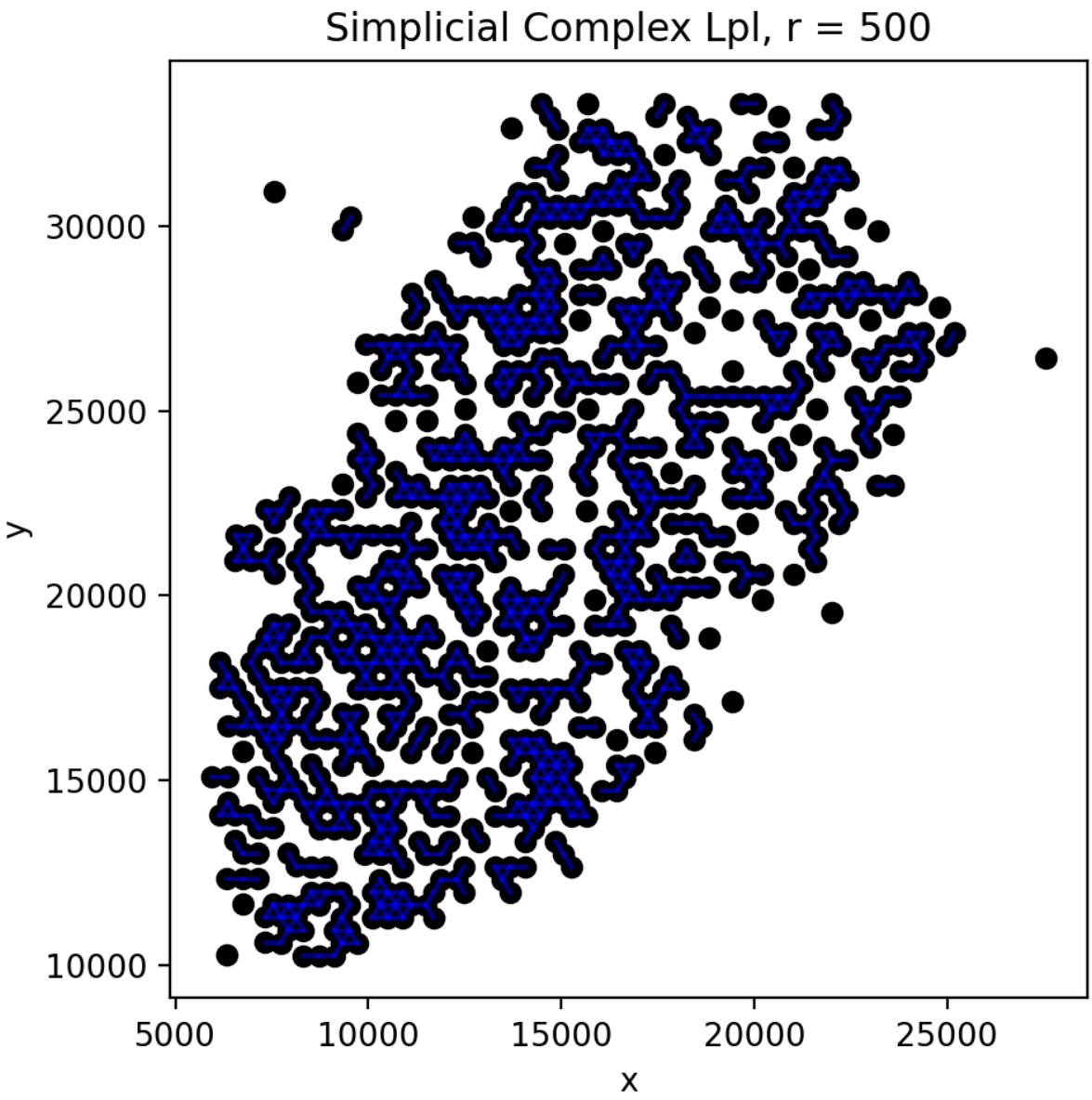


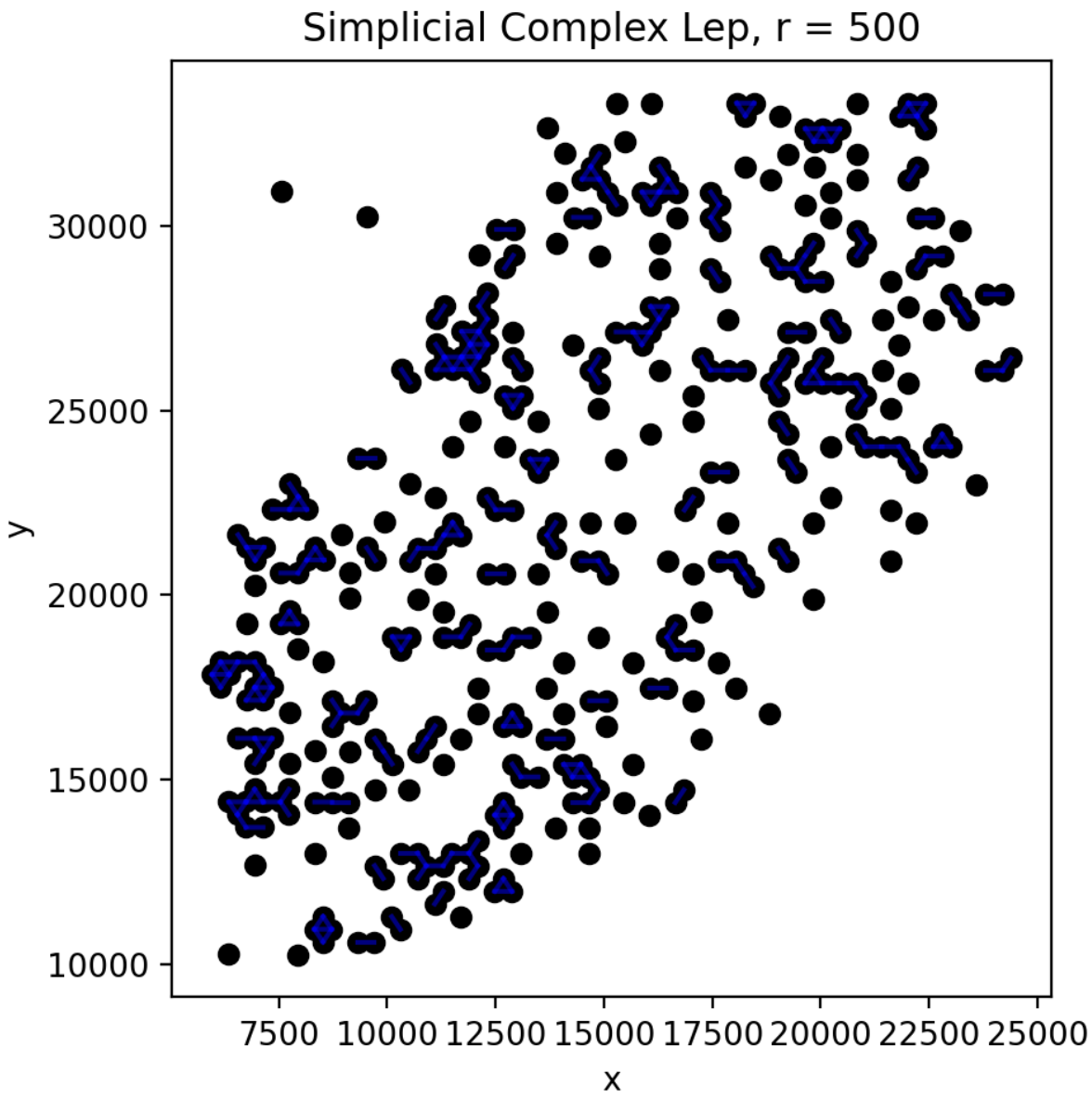
When we set the threshold higher, there are less points in the cloud overall. This means that even though we maintained the same radii as the previous question, there are significantly less connections. The structure is more sparse and captures the more meaningful groupings.

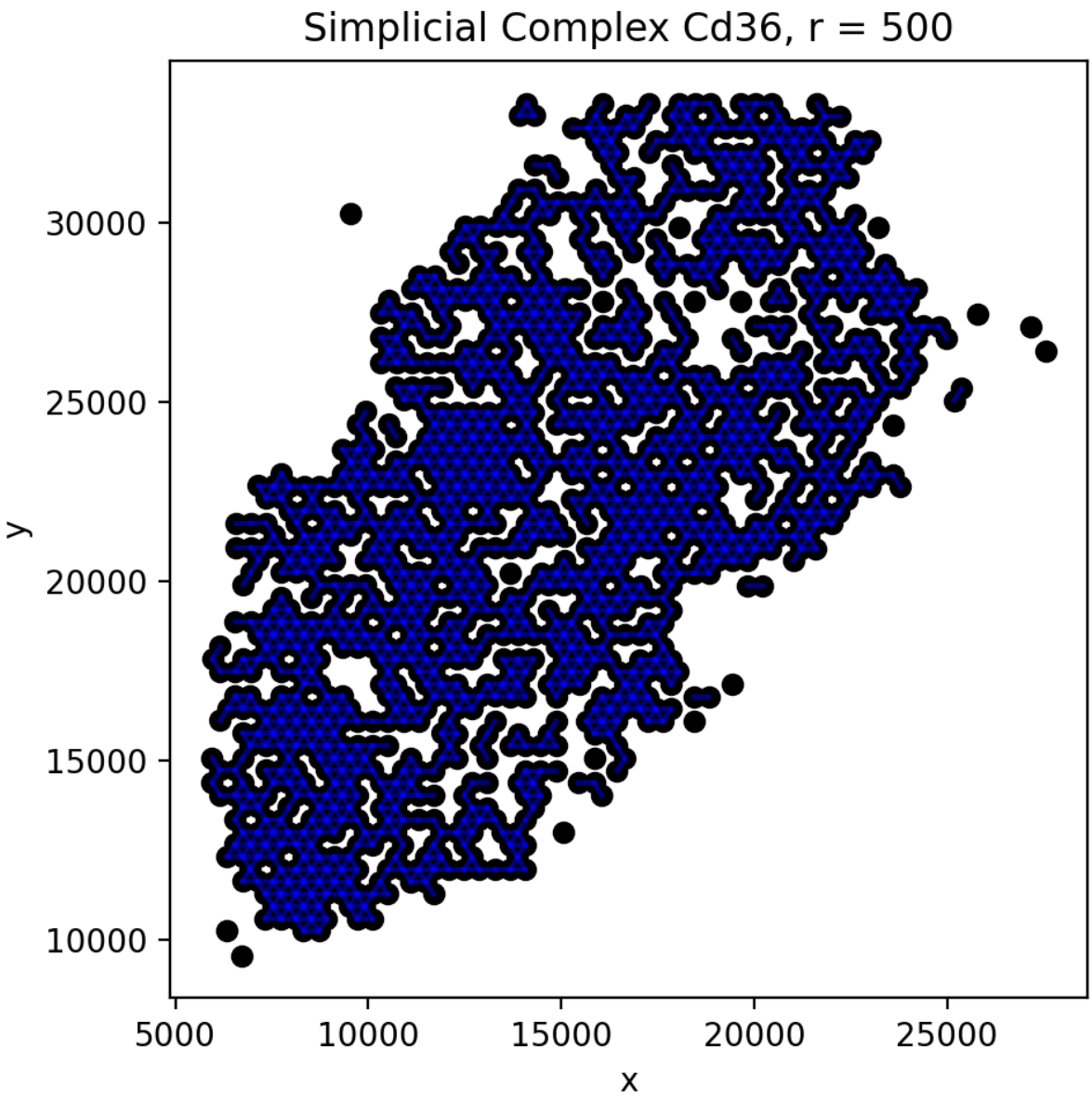
In [72]:
```
#Set threshold > 0 and radius = 500
for gene in ['Trem2', "Lpl", "Lep", "Cd36"]:
    df = adata[:, gene].to_df()

    coords = pd.DataFrame(adata.obsm['spatial'], index=df.index, columns=['x', 'y']

    # merge in the coordinates
    df = pd.merge(
        df,
        coords,
        how='left',
        left_index=True,
        right_index=True,
    )
```

```
filtered_df = df[df[gene]>0]
create_simplicial(filtered_df, 500, gene)
```




Simplicial Complex Lpl, r = 500
30000
25000
20000
15000
10000
y
5000
10000
15000
20000
25000
x


Simplicial Complex Lep, r = 500
30000
25000
20000
15000
10000
y
7500
10000
12500
15000
17500
20000
22500
25000
x



Cd30 is much more connected in its structure, because there are many data points that are greater than my threshold. The other genes have more sparse graphs, though Lpl has a topological structure somewhere in between being sparse and fully connected.