CSE 564
Visualization & Visual Analytics

Dimension reduction

Klaus Mueller

Computer Science Department
Stony Brook University
<table>
<thead>
<tr>
<th>Lecture</th>
<th>Topic</th>
<th>Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Intro, schedule, and logistics</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Applications of visual analytics, basic tasks, data types</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Introduction to D3, basic vis techniques for non-spatial data</td>
<td>Project #1 out</td>
</tr>
<tr>
<td>4</td>
<td>Data assimilation and preparation</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Data reduction and notion of similarity and distance</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Visual perception and cognition</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Visual design and aesthetics</td>
<td>Project #1 due</td>
</tr>
<tr>
<td>8</td>
<td><strong>Dimension reduction</strong></td>
<td>Project #2 out</td>
</tr>
<tr>
<td>9</td>
<td>Data mining techniques: clusters, text, patterns, classifiers</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Data mining techniques: clusters, text, patterns, classifiers</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Computer graphics and volume rendering</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Techniques to visualize spatial (3D) data</td>
<td>Project #2 due</td>
</tr>
<tr>
<td>13</td>
<td>Scientific and medical visualization</td>
<td>Project #3 out</td>
</tr>
<tr>
<td>14</td>
<td>Scientific and medical visualization</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Midterm #1</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>High-dimensional data, dimensionality reduction</td>
<td>Project #3 due</td>
</tr>
<tr>
<td>17</td>
<td>Big data: data reduction, summarization</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Correlation and causal modeling</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Principles of interaction</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Visual analytics and the visual sense making process</td>
<td>Final project proposal due</td>
</tr>
<tr>
<td>21</td>
<td>Evaluation and user studies</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Visualization of time-varying and time-series data</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Visualization of streaming data</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Visualization of graph data</td>
<td>Final Project preliminary report due</td>
</tr>
<tr>
<td>25</td>
<td>Visualization of text data</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Midterm #2</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Data journalism</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Final project presentations</td>
<td>Final Project slides and final report due</td>
</tr>
</tbody>
</table>
Data Reduction
Dimension Reduction
Are there attributes that “go together”? Can you name a few?
FEATURE VECTOR (1)

Physical attributes

- color
- number of doors
- number of wheels
- retractable roof
- height
- length
- frames around side windows

Which attributes are useful to distinguish SUVs from convertibles?

- number of doors (4 vs. 2) --> numerical, two levels
- retractable roof (no vs. yes) --> categorical, two levels
- frames around side windows (yes vs. no) --> categorical, two levels
- height (higher vs. lower) --> numerical, many levels
Which attributes are not so useful?

- number of wheels (constant 4) --> no discriminative power
- length (short and long SUVs, convertibles) --> confounding
- color (colors are seemingly random, or are they?)

Is color useful?

- the convertibles seem to have more vibrant colors (red, yellow, ...)
- so maybe we made a discovery
Need to consider more than two attributes

- *height* attribute would have distinguished the Range Rover from the convertibles and caused it to be an outlier.
New classes are constantly evolving over time

- this is known as *cluster evolution*
- measuring more features will increase the chance of discovery

why can empty feature spaces be interesting or useful?

new class: the convertible SUV
The more data (examples) the better

- increases the chances to discover the rare specimen

- but some attributes are useless
- we can cull them away
- perform attribute reduction or *dimension reduction*
**Dimensionality Reduction**

By axis rotation

- determine a more efficient basis
- Principal Component Analysis (PCA)
- Singular value decomposition (SVD)
- Latent semantic analysis (LSA)

By type transformation

- determine a more efficient data type
- Fourier analysis and Wavelets for grids
- Multidimensional scaling (MSD) for graphs
- Locally Linear Embedding
- Isomap
- Self Organizing Maps (SOM)
- Linear Discriminant Analysis (LDA)
PRINCIPAL COMPONENT ANALYSIS (PCA)
Covariance

- measures how much two random variables change together

For N variables we have $N^2$ variable pairs

- we can write them in a matrix of size $N^2$ → the covariance matrix
- for two variables $X_1$ and $X_2$

$$\text{Var}[X] = \begin{bmatrix} \text{Var}[X_1] & \text{Cov}[X_1,X_2] \\ \text{Cov}[X_2,X_1] & \text{Var}[X_2] \end{bmatrix}$$
Covariance \( \text{cov}(X,Y) \)

\[
\text{COV}(X,Y) = \frac{\sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})}{n-1}
\]

Pearson’s correlation \( r \)

- is covariance normalized by the individual variances for attributes X and Y

\[
r_{xy} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2 \sum_{i=1}^{n} (y_i - \bar{y})^2}}
\]

mean of all data item values \( x_i \) and \( y_i \) for attributes X and Y, resp.
Correlation rates between -1 and 1:

- 1.0
- 0.8
- 0.4
- 0.0
- -0.4
- -0.8
- -1.0

Important to note:
- correlation is defined for linear relationships
- visualization can help
- none of these point distributions have correlations:
Covariance Matrix

Analytical: \( \text{Cov}(X, Y) = E[(X - \mu_x)(Y - \mu_y)] \)

Samples: \( \sigma_{xy} = \text{cov}_{xy} = \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) \)

An n-D dataset has \( n \) variables \( x_1, x_2, \ldots, x_n \)

- define pairwise covariance among all of these variables
- construct a covariance matrix

\[
\Sigma = \text{Cov}(X) = \begin{bmatrix}
\sigma_{11} & \sigma_{12} & \cdots & \sigma_{1n} \\
\sigma_{21} & \sigma_{22} & \cdots & \sigma_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
\sigma_{n1} & \sigma_{n2} & \cdots & \sigma_{nn}
\end{bmatrix}
\]

- a correlation matrix would just list the correlations instead
## Correlation Matrix

<table>
<thead>
<tr>
<th></th>
<th>MO</th>
<th>FP</th>
<th>MP</th>
<th>IM</th>
<th>IC</th>
<th>FM</th>
<th>FE</th>
<th>FI</th>
<th>SPC</th>
<th>DSC</th>
<th>DST</th>
</tr>
</thead>
<tbody>
<tr>
<td>MO</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FP</td>
<td>0.31&lt;sup&gt;a&lt;/sup&gt;</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MP</td>
<td>0.32&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0.71&lt;sup&gt;a&lt;/sup&gt;</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IM</td>
<td>0.36&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0.12&lt;sup&gt;c&lt;/sup&gt;</td>
<td>0.14&lt;sup&gt;c&lt;/sup&gt;</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IC</td>
<td>0.39&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0.18&lt;sup&gt;b&lt;/sup&gt;</td>
<td>0.21&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0.62&lt;sup&gt;a&lt;/sup&gt;</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FM</td>
<td>0.26&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0.21&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0.14&lt;sup&gt;c&lt;/sup&gt;</td>
<td>0.30&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0.27&lt;sup&gt;a&lt;/sup&gt;</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FE</td>
<td>0.47&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0.21&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0.18&lt;sup&gt;b&lt;/sup&gt;</td>
<td>0.38&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0.28&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0.24&lt;sup&gt;a&lt;/sup&gt;</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FI</td>
<td>0.53&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0.26&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0.22&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0.36&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0.37&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0.29&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0.47&lt;sup&gt;a&lt;/sup&gt;</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPC</td>
<td>0.32&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0.22&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0.31&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0.51&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0.47&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0.32&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0.37&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0.35&lt;sup&gt;a&lt;/sup&gt;</td>
<td>1.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSC</td>
<td>-0.12&lt;sup&gt;c&lt;/sup&gt;</td>
<td>0.03&lt;sup&gt;c&lt;/sup&gt;</td>
<td>0.05&lt;sup&gt;c&lt;/sup&gt;</td>
<td>0.17&lt;sup&gt;b&lt;/sup&gt;</td>
<td>0.08&lt;sup&gt;c&lt;/sup&gt;</td>
<td>0.18&lt;sup&gt;b&lt;/sup&gt;</td>
<td>-0.05&lt;sup&gt;c&lt;/sup&gt;</td>
<td>0.06&lt;sup&gt;c&lt;/sup&gt;</td>
<td>0.01&lt;sup&gt;c&lt;/sup&gt;</td>
<td>1.00</td>
<td></td>
</tr>
<tr>
<td>DST</td>
<td>-0.02&lt;sup&gt;c&lt;/sup&gt;</td>
<td>-0.01&lt;sup&gt;c&lt;/sup&gt;</td>
<td>0.05&lt;sup&gt;c&lt;/sup&gt;</td>
<td>0.24&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0.14&lt;sup&gt;c&lt;/sup&gt;</td>
<td>0.05&lt;sup&gt;c&lt;/sup&gt;</td>
<td>-0.05&lt;sup&gt;c&lt;/sup&gt;</td>
<td>0.05&lt;sup&gt;c&lt;/sup&gt;</td>
<td>0.05&lt;sup&gt;c&lt;/sup&gt;</td>
<td>0.56&lt;sup&gt;a&lt;/sup&gt;</td>
<td>1.00</td>
</tr>
<tr>
<td>DM</td>
<td>0.05&lt;sup&gt;c&lt;/sup&gt;</td>
<td>0.144</td>
<td>0.136&lt;sup&gt;c&lt;/sup&gt;</td>
<td>0.199&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0.169&lt;sup&gt;b&lt;/sup&gt;</td>
<td>0.247&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0.08&lt;sup&gt;c&lt;/sup&gt;</td>
<td>0.11&lt;sup&gt;c&lt;/sup&gt;</td>
<td>0.14&lt;sup&gt;c&lt;/sup&gt;</td>
<td>0.46&lt;sup&gt;a&lt;/sup&gt;</td>
<td>0.71&lt;sup&gt;a&lt;/sup&gt;</td>
</tr>
</tbody>
</table>

**Climatic predictors**

- WetDays
- TempJuly
- TempJan
- TempAnn
- RHJuly

**just value**

**distribution (scatterplot matrix)**
Ultimate goal:

- find a coordinate system that can represent the variance in the data with as few axes as possible

- rank these axes by the amount of variance (blue, red)
- drop the axes that have the least variance (red)
Find the principal components (factors) of a distribution

First characterize the distribution by
- covariance matrix Cov
- correlation matrix Corr
- let's call it C
- perform QR factorization or LU decomposition to get
  \[ C = QQ^{-1} \]
  
  Q: matrix with Eigenvectors
  \( \Lambda \): diagonal matrix with Eigenvalues \( \lambda \)
- now order the Eigenvectors in terms of their Eigenvalues \( \lambda \)
EIGENVECTORS AND VALUES
When to use what?

- use the covariance matrix when the variable scales are similar
- use the correlation matrix when the variables are on different scales
- the correlation matrix *standardizes* the data
- in general they give different results, especially when the scales are different
Before PCA
After PCA

- $\lambda_1 = 9.8783$  $\lambda_2 = 3.0308$  Trace = 12.9091
- PC 1 displays (“explains”) $\frac{9.8783}{12.9091} = 76.5\%$ of total variance
See other slide sets posted on the course website

Principal Component Analysis (PCA)
- Theory, Practice, and Examples
- PCA loadings, and what they mean for analysis
Some familiar faces...
We can reconstruct each face as a linear combination of “basis” faces, or Eigenfaces [M. Turk and A. Pentland (1991)]
90% variance is captured by the first 50 eigenvectors.

Reconstruct existing faces using only 50 basis images.

We can also generate new faces by combining eigenvectors with different weights.
Transformations
**Multidimensional Scaling (MDS)**

MDS is for irregular structures
- scattered points in high-dimensions (N-D)
- adjacency matrices

Maps the distances between observations from N-D into low-D (say 2D)
- attempts to ensure that differences between pairs of points in this reduced space match as closely as possible
MDS turns a distance matrix into a network or point cloud
- correlation, cosine, Euclidian, and so on

Suppose you know a matrix of distances among cities

<table>
<thead>
<tr>
<th></th>
<th>Chicago</th>
<th>Raleigh</th>
<th>Boston</th>
<th>Seattle</th>
<th>S.F.</th>
<th>Austin</th>
<th>Orlando</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chicago</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Raleigh</td>
<td>641</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boston</td>
<td>851</td>
<td>608</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Seattle</td>
<td>1733</td>
<td>2363</td>
<td>2488</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S.F.</td>
<td>1855</td>
<td>2406</td>
<td>2696</td>
<td>684</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Austin</td>
<td>972</td>
<td>1167</td>
<td>1691</td>
<td>1764</td>
<td>1495</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Orlando</td>
<td>994</td>
<td>520</td>
<td>1105</td>
<td>2565</td>
<td>2458</td>
<td>1015</td>
<td>0</td>
</tr>
</tbody>
</table>
RESULT OF MDS
Compare with real Map
MDS Algorithm

- Task:
  - Find that configuration of image points whose pairwise distances are most similar to the original inter-point distances !!!

- Formally:
  - Define: $D_{ij} = ||x_i - x_j||_D$, $d_{ij} = ||y_i - y_j||_D$
  - Claim: $D_{ij} \equiv d_{ij}$ $\forall i, j \in [1, n]$

- In general: an exact solution is not possible !!!

- Inter Point distances $\rightarrow$ invariance features
MDS Algorithm

Strategy (of metric MDS):

- iterative procedure to find a good configuration of image points
  - 1) Initialization
     - Begin with some (arbitrary) initial configuration
  - 2) Alter the image points and try to find a configuration of points that minimizes the following sum-of-squares error function:
MDS Algorithm

Strategy (of metric MDS):

- iterative procedure to find a good configuration of image points

  1) Initialization
     \[ \rightarrow \text{Begin with some (arbitrary) initial configuration} \]

  2) Alter the image points and try to find a configuration of points that minimizes the following sum-of-squares error function:

\[
E = \sum_{i<j}^{N} (D_{ij} - d_{ij})^2
\]
Spring-like system

- insert springs within each node
- the length of the spring encodes the desired node distance
- start at an initial configuration
- iteratively move nodes until an energy minimum is reached

**Force-Directed Algorithm**
Spring-like system

- insert springs within each node
- the length of the spring encodes the desired node distance
- start at an initial configuration
- iteratively move nodes until an energy minimum is reached
USES OF MDS

Data layout

Attribute layout
Distance (similarity) metric

- Euclidian distance (best for data)
- Cosine distance (best for data)
- |1-corr| distance (best for attributes)
- use 1-corr to move correlated attribute points closer
- use || if you do not care about positive or negative correlations
MDS Examples
Combine Data and Attribute Layouts
Achieved by Joint Matrix Optimization
no dream school here: good athletics, low tuition, high academic score
Data Context Map:
Choose a Good University
MDS EXAMPLES
MANIFOLD LEARNING: ISOMAP


Tries to unwrap a high-dimensional surface (A) \(\rightarrow\) manifold
- noisy points could be averaged first and projected onto the manifold

Algorithm
- construct neighborhood graph G \(\rightarrow\) (B)
- for each pair of points in G compute the shortest path distances by adding small Euclidian hops (Floyd’s, Dijkstra’s algorithm) \(\rightarrow\) geodesic distances
- fill similarity matrix with these geodesic distances
- embed (layout) in low-D (2D) with MDS \(\rightarrow\) (C)
MANIFOLD LEARNING: 
LOCALLY LINEAR EMBEDDING (LLE)


Based on simple geometric intuitions.

- suppose the data consist of \( N \) real-valued vectors \( X_i \), each of dimensionality \( D \)
- each data point and its neighbors are expected to lie on or close to a locally linear patch of the manifold

High dimensional Manifold      Low dimensional Manifold
LLE Overview

1. Select neighbors
2. Reconstruct with linear weights
3. Map to embedded coordinates
Steps:

- assign K neighbors to each data point $\tilde{X}_i$
- compute the weights $W_{ij}$ that best linearly reconstruct the data point from its K neighbors, solving the constrained least-squares problem

$$\hat{e}(W) = \sum_i |\tilde{X}_i - \sum_j W_{ij} \tilde{X}_j|^2 \quad \sum_j W_{ij} = 1$$

- compute the low-dimensional embedding vectors $\tilde{Y}_i$ best reconstructed by $W_{ij}$

$$\Phi(Y) = \sum_i |\tilde{Y}_i - \sum_j W_{ij} \tilde{Y}_j|^2$$
Self-Organizing Maps (SOM)

Introduced by Teuvo Kohonen

- unsupervised learning and clustering algorithm
- has advantages compared to hierarchical clustering
- often realized as an artificial neural network

SOMs group the data

- perform a nonlinear projection from N-dimensional input space onto two-dimensional visualization space
- provide a useful topological arrangement of information objects in order to display clusters of similar objects in information space
Map a dataset of 3D color vectors into a 2D plane

- assume you have an image with 5 colors
- want to see how many there are of each
- compute an SOM of the color vectors
Create array and connect all elements to the N input vector dimensions
- shown here: 2D vector with $4 \times 4$ elements
- initialize weights

For each input vector chosen at random
- find node with weights most like the input vector
- call that node the Best Matching Unit (BMU)
- find nodes within neighborhood radius $r$ of BMU
  - initially $r$ is chosen as the radius of the lattice
  - diminishes at each time step
- adjust the weights of the neighboring nodes to make them more like the input vector
  - the closer a node is to the BMU, the more its weights get altered
SOM Example: Poverty Map

SOM – Result Example

World Poverty Map

A SOM has been used to classify statistical data describing various quality-of-life factors such as state of health, nutrition, educational services etc. Countries with similar quality-of-life factors end up clustered together. The countries with better quality-of-life are situated toward the upper left and the most poverty stricken countries are toward the lower right.

‘Poverty map’ based on 39 indicators from World Bank statistics (1992)
SOM Example: ThemeScape

Height represents density or number of documents in the region
Invented at Pacific Northwest National Lab (PNNL)
SOM / MDS Example: VxInsight (Sandia)

Figure 4: Multi-resolution exploration with detail on demand.
PRACTICAL ASPECTS
See [this excellent page](#) for more detail

- uses MongoDB as a NoSQL database (non-relational SQL)

**Step 1: Build a python server, say app.py**

- use Flask as the web framework

```python
from flask import Flask
from flask import render_template

app = Flask(__name__)

@app.route("/")
def index():
    return render_template("index.html")

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5000, debug=True)
```

**Example 1:**

Make an index.html file containing

```
<h1>Hello World!</h1>
```

Run the below from a terminal window

```
$ python app.py
```

Open a browser and go to http://localhost:5000/, you will see the message Hello World!.
Step 2: Add all your processing code to app.py

- in this case it mainly involves storing data into the database

```python
@app.route("/"")
def index():
    return render_template("index.html")

@app.route("/donorschoose/projects")
def donorschoose_projects():
    connection = MongoClient(MONGODB_HOST, MONGODB_PORT)
    collection = connection[DBS_NAME][COLLECTION_NAME]
    projects = collection.find(projection=FIELDS)
    json_projects = []
    for project in projects:
        json_projects.append(project)
    json_projects = json.dumps(json_projects, default=json_util.default)
    connection.close()
    return json_projects

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5000, debug=True)
```

Example 2:

Start the server by running python app.py
Go to (in this example) http://localhost:5000/donorschoose/projects
You will see all the projects data printed in the browser.
Step 3: Build the charts

- create a JavaScript file, say, charts.js
- gets the data from the python URL and other provided JSON files
- calls function, here makeGraphs(), to do the d3 rendering

```javascript
queue()
  .defer(d3.json, "/donorschoose/projects")
  .defer(d3.json, "static/geojson/us-states.json")
  .await(makeGraphs);
```

```javascript
function makeGraphs(error, projectsJson, statesJson) {
  ...
}
```

- check the webpage for more detail on how to build the charts
Step 3: Build the charts

- ....
- call the renderAll() function for rendering all the charts within index.html need to reference all the charts we defined in charts.js
- for example, if you want to show the US map chart, you will have to add the following line below to the index.html file.
COUPLING D3 WITH PYTHON

Start app.py web server
Add query to the index.html file
Call http://localhost:5000/ in the browser to see the dashboard
Folder Structure

All files are available in a dedicated [github repository](https://github.com).

One more thing:

**JSON IS THE GLUE BETWEEN PYTHON AND JS**
  - csv data gets stored in MongoDB (4th most popular database)


There are other pages ... use google