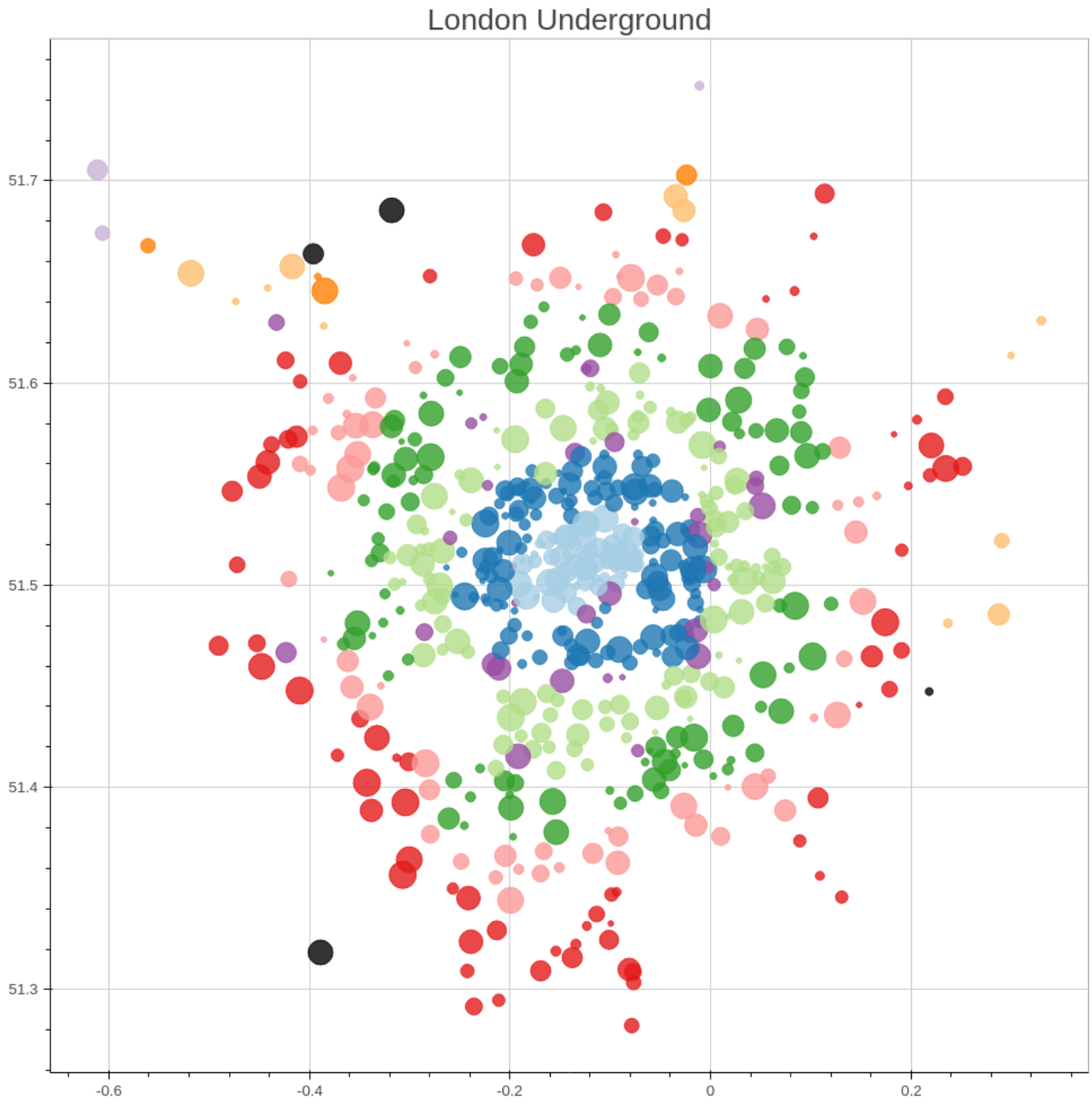
Rush Hour Dynamics: Using Python to Study the London Underground

Camilla Montonen

PyData Paris 2015



Introduction

Roadmap

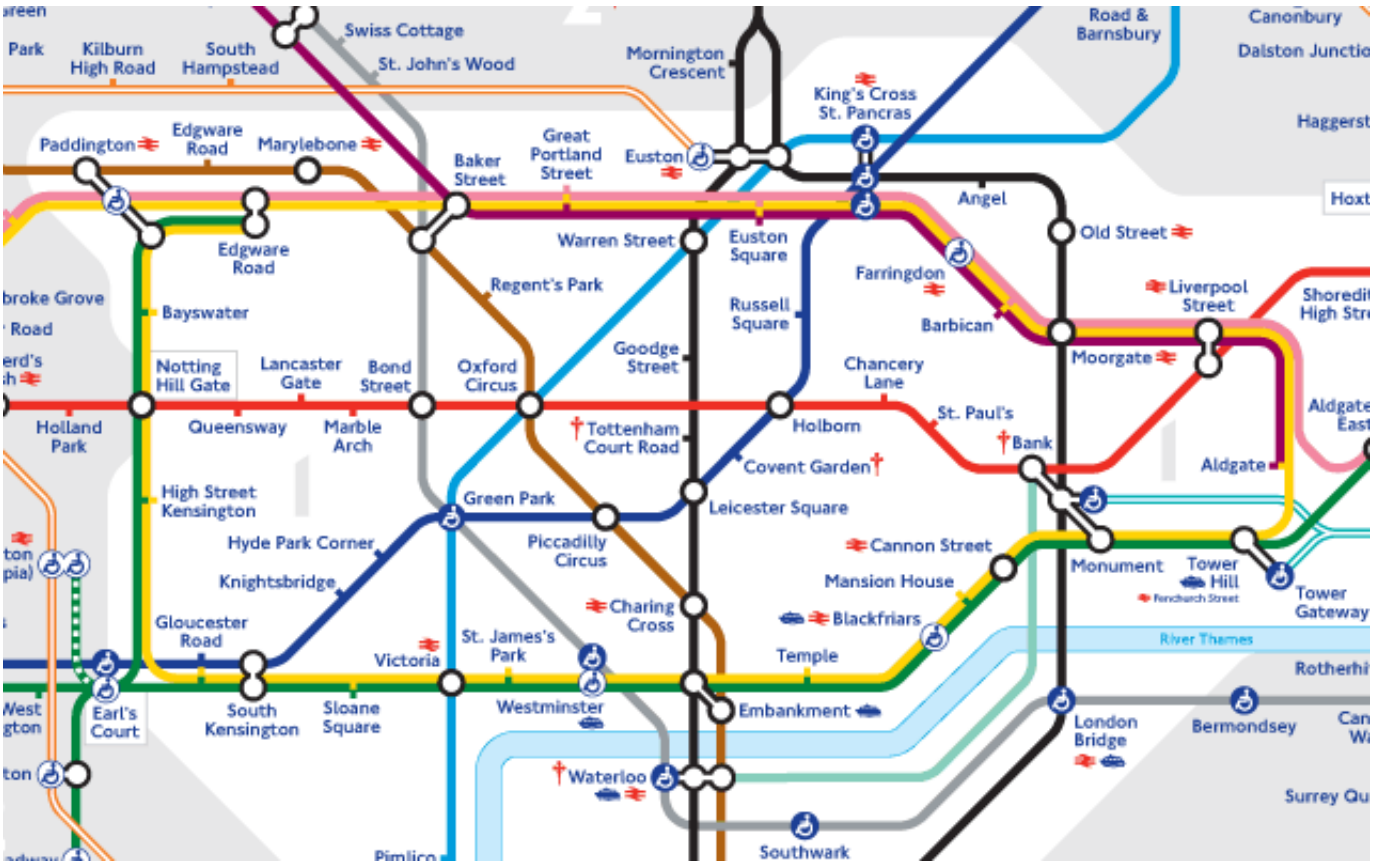
1. Motivation: Why would you want to analyse the London Underground?! Commuting on it is bad enough.
2. Data collection: The Challenge of Collecting Data Stored in a Map
3. Data analysis: Leveraging graph-tool to analyse the London Underground
4. Simulations: Creating simulations using Bokeh

The Takeaway Message

There are interesting data problems everywhere in our environment and Python provides a set of amazing tools to start asking questions and generating answers. Don't be afraid to investigate (and write some amazing Python code as you go)!

Back in August 2014...

Which Tube line should I take to work?



Motivations

- Delays and suspension on remote stations and Tube lines seem to congest even remote stations

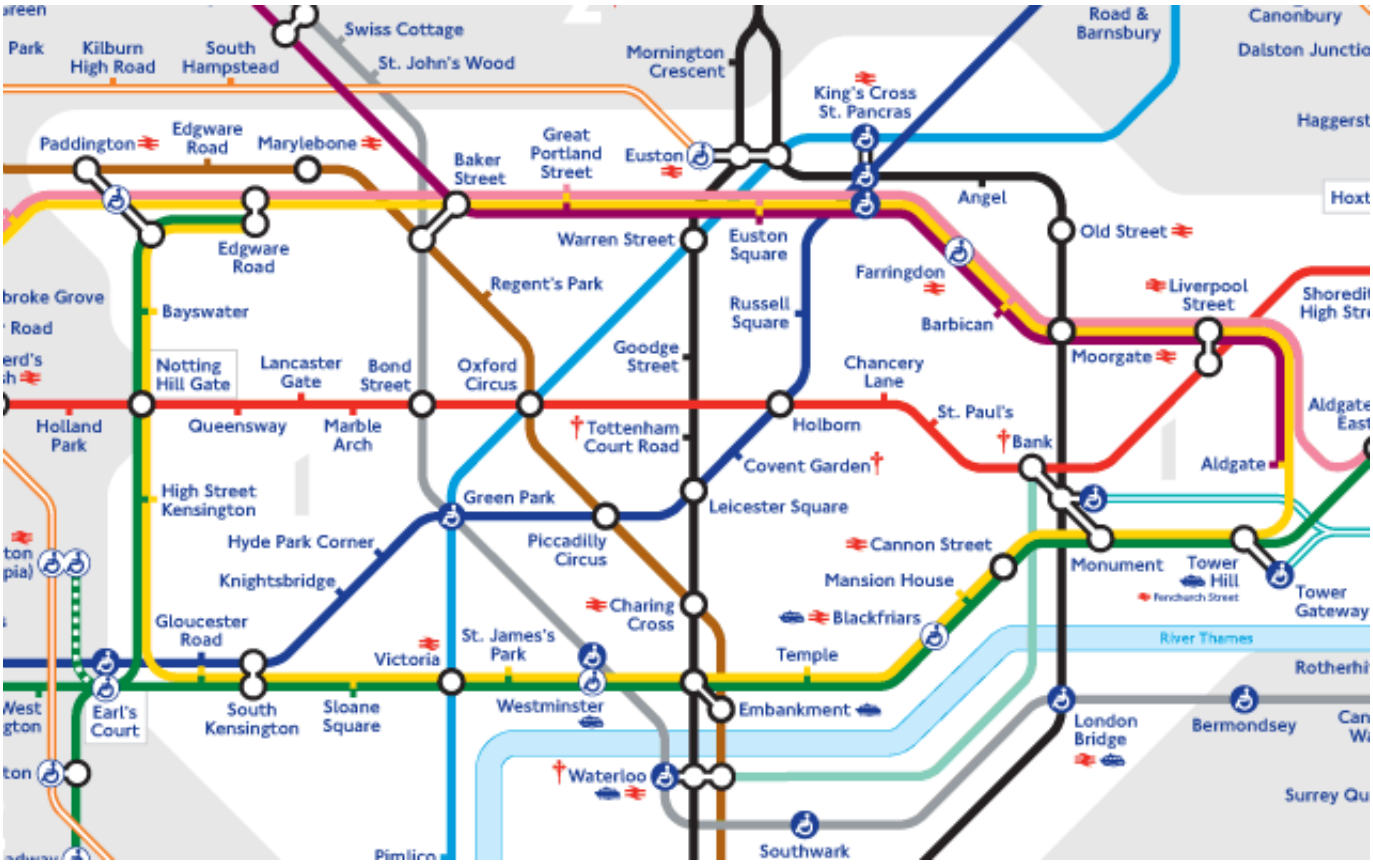
Studying the London Underground and performing simulations can help to understand:

- the overall characteristics of the London Underground Network
- The Tube stations that are critical for the proper functioning of the network

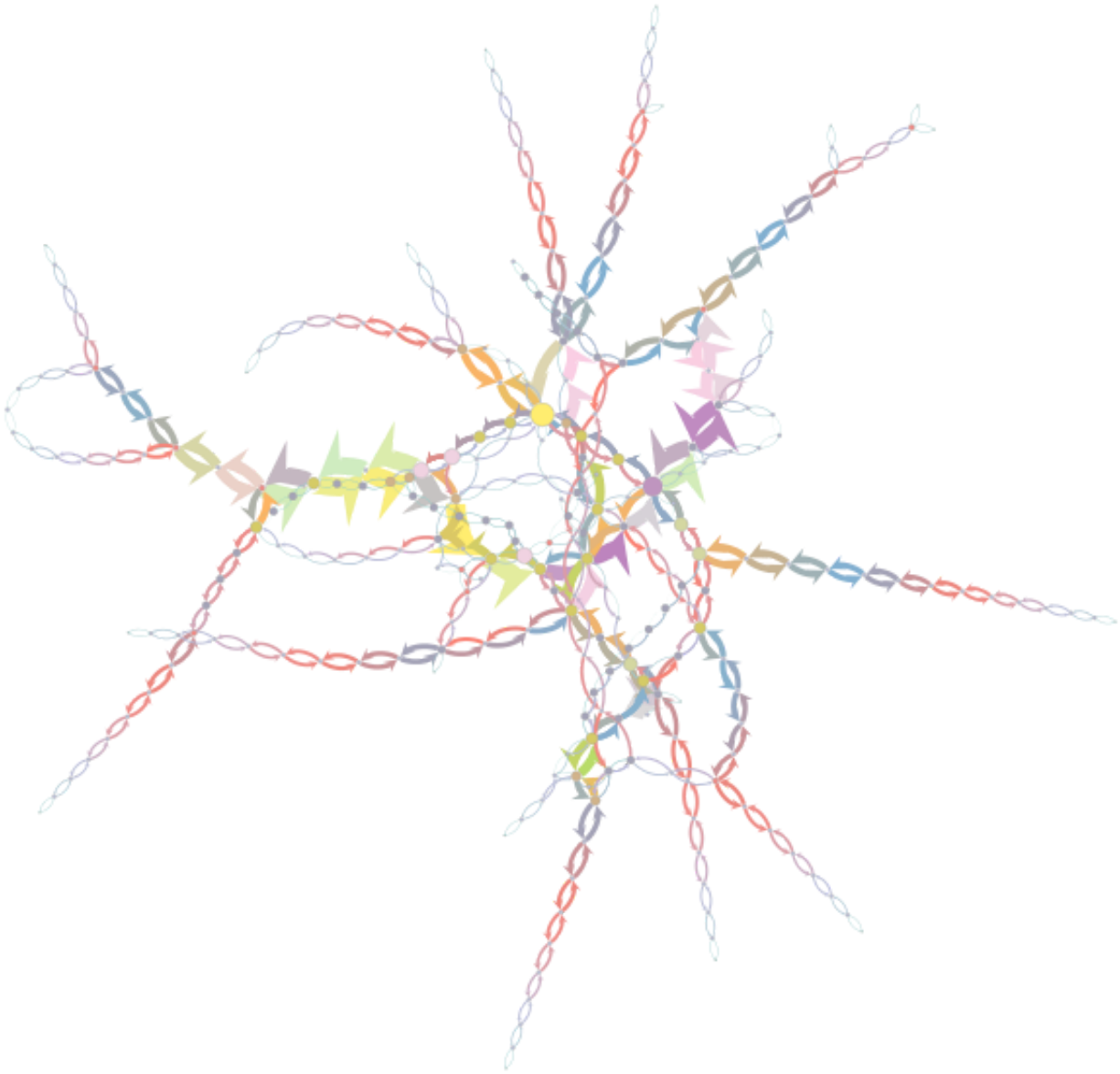
Data Collection

The challenge in data collection was to find a way to translate the standard TfL London Underground Map into a graph representation.

Start:


Swiss Cottage
St. John's Wood
Mornington Crescent
King's Cross St. Pancras
Road & Barnsbury
Canonbury
Dalston Junctio
Haggerst
Park
Kilburn High Road
South Hampstead
Paddington
Edgware Road
Marylebone
Baker Street
Great Portland Street
Euston
Angel
Old Street
Hoxt
Edgware Road
Warren Street
Euston Square
Farringdon
Liverpool Street
Shoredit High Str
broke Grove
Road
Bayswater
Regent's Park
Russell Square
Barbican
Moorgate
erd's
Notting Hill Gate
Lancaster Gate
Bond Street
Oxford Circus
Goodge Street
Chancery Lane
Aldgate East
Holland Park
Queensway
Marble Arch
Tottenham Court Road
Holborn
St. Paul's
Bank
Aldgate
High Street Kensington
Green Park
Covent Garden
Leicester Square
Hyde Park Corner
Piccadilly Circus
Cannon Street
Monument
Tower Hill
Tower Gateway
Knightsbridge
Mansion House
Charing Cross
Blackfriars
Gloucester Road
St. James's Park
Victoria
Temple
River Thames
Rotherhi
West gton
Earl's Court
South Kensington
Sloane Square
Westminster
Embankment
London Bridge
Bermondsey
Waterloo
Surrey Qu
Southwark
Pimlico

Goal:



Data collection:

It would be cool to program some kind of OCR to automatically read the data from the map and produce a data file! But alas, I had to resort to manually creating a data file:

```
#Station #Neighbour(line)
Acton Town            Chiswick Park (District), South Ealing (Picadilly),
Turnham Green (Picadilly)
Aldgate                Tower Hill (Circle; District), Liverpool Street (Me
tropolitan; Circle; District)
Aldgate East         Tower Hill (District), Liverpool Street (HammersmithCi
ty; Metropolitan)
Alperton             Sudbury Town (Picadilly), Park Royal (Picadilly)
```

Data analysis

Now that we have collected out data, it's time start interesting questions about it.

Some of the burning questions that I had:

1. What are the most "important" stations in the London Underground network?
2. What is the average shortest path between any two stations?
3. Which stations are the most critical for the proper functioning of the network?

Data analysis using graph-tool

- `graph-tool` is a Python library written by Tiago Peixoto that provides a number of tools for analyzing and plotting graphs.
- it provides a number of useful tools and methods

    1. A `Graph` object for defining graphs
    2. Property maps : helpful for associating values with vertices and edges
    3. Various methods for analyzing graph topology
    4. Built-in graph visualization

Another Python tool that you may wish to explore for graph analysis is `NetworkX`.

Data analysis using graph-tool

Let's see what the graph looks like!

Data analysis using graph-tool

`graph-tool` allows us to compute several interesting metrics, which are often used to characterize graphs:

1. Degree distribution
2. Average shortest path

In [2]:
```
# import necessary packages

# define data files
geographical_data="/home/winterflower/programming_projects/python-londontube/src/data/london_stations.csv"
network_data="/home/winterflower/programming_projects/python-londontube/src/data/londontubes.txt"
```

Data analysis using graph-tool

1. What are the most "important" stations in the London Underground network?

Of course, there are many ways to measure the importance of a vertex in a graph. One such measure is called *betweenness centrality* . Simply stated, it measures the fraction of shortest paths out of all shortest paths that pass through the vertex.

`graph-tool` provides a module `graph-tool.centrality` which allows you to compute various centrality measures out of the box.

Betweenness Centrality

What fraction of all shortest paths passes through this vertex?

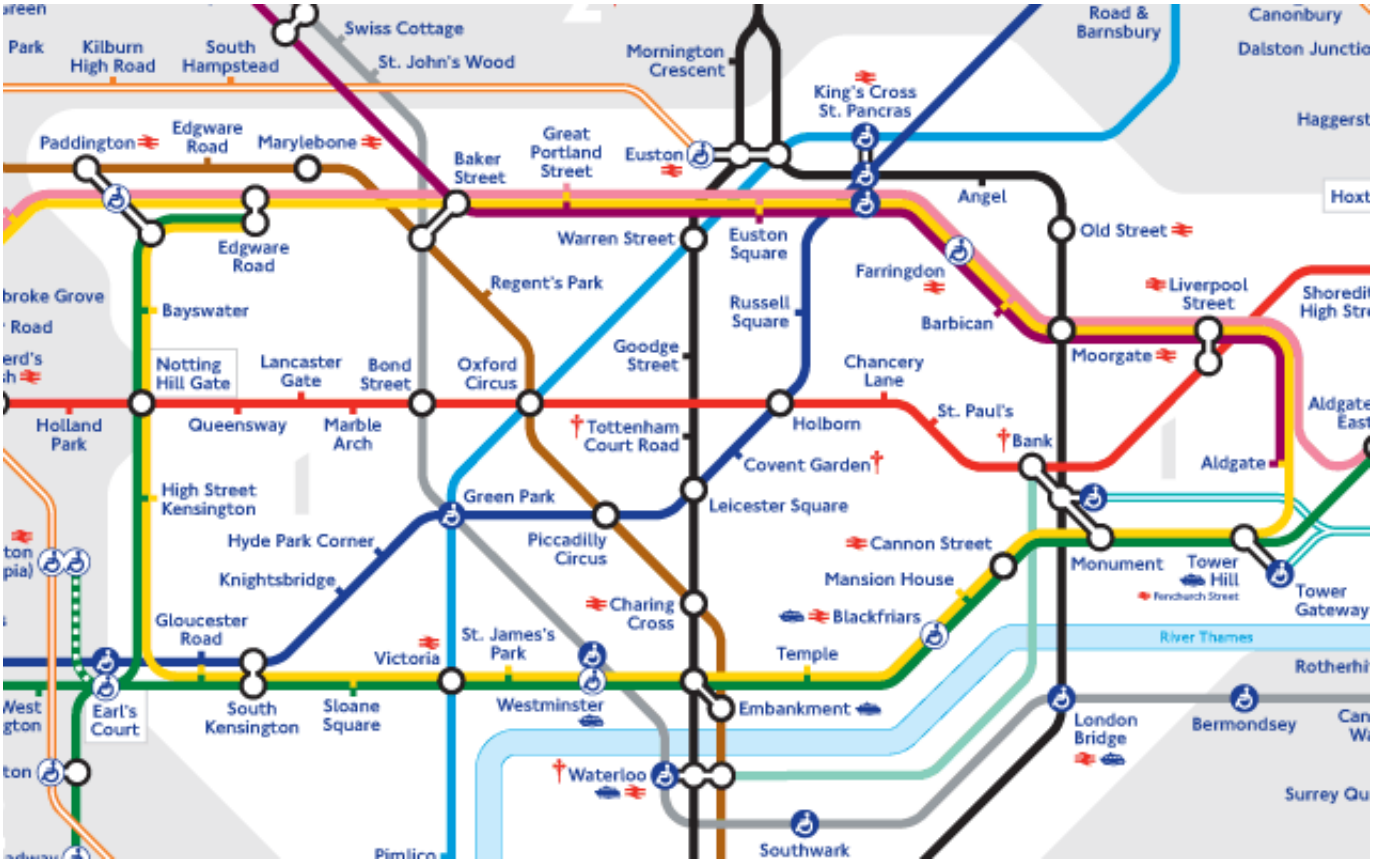
In [2]:
```
#define some useful preliminaries
geographical_data="/home/winterflower/programming_projects/python-lo
ndontube/src/data/london_stations.csv"
network_data="/home/winterflower/programming_projects/python-londont
ube/src/data/londontubes.txt"

#calculate the betweenness centrality
#create the map_object

from src import simulation_utils
from src.graph_analytics import graph_analysis
import pandas as pd
betweenness_centrality_series_object=graph_analysis.calculate_betwee
nness(network_data)
betweenness_centrality_series_object.sort(ascending=False)
print betweenness_centrality_series_object[:10]
```

```
Baker Street              0.344084
King's Cross St.Pancras   0.303868
Liverpool Street          0.267392
Green Park                0.263264
Mile End                  0.229449
Bethnal Green             0.227822
Victoria                  0.222771
Stratford                 0.220119
Finchley Road             0.211660
Waterloo                  0.207129
dtype: float64
```

Betweenness Centrality


BAKER STREET
Way out

Betweenness Centrality



Shortest paths

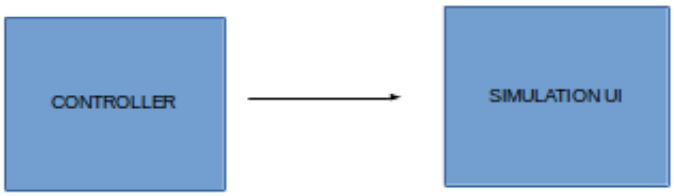
Which station has the smallest average shortest path to any other station in the graph?

In [5]:
```
## calculate the length of the shortest path from any two stations

from src.graph_analytics import graph_analysis
shortest_paths=graph_analysis.calculate_all_shortest_paths(network_data)
#calculate the mean shortest path
mean_shortest_path=shortest_paths.mean(axis=0)
#find out stations with smallest mean shortest paths
mean_shortest_path.order(ascending=True, inplace=True)
#find out the top 5 stations
mean_shortest_path[:5]
```

Out[5]:
```
Green Park       8.901887
Oxford Circus    9.007547
Bond Street      9.090566
Baker Street     9.211321
Westminster      9.339623
dtype: float64
```

Simulating commuter flow between stations

Designing a simple two component simulation:



Simulating commuter flow between stations

- Bokeh allows you to create graphs that update in "real-time"

Summary

- Python provides excellent libraries for studying real-world problems where the natural representation of the data is a graph
- In addition to calculating metrics, you can easily make amazing animations by integrating graph-tool with bokeh
- Find interesting problems, ask hard questions and start exploring!

Thank you ! (and please ask questions!)

In []: