Background

The Supreme Court of the United States has long been the most enigmatic but powerful parts of the federal government. During key decisions, the 9 people who sit on the court hold huge portions of the country's fate in their hands. As such, predicting how the court will vote has become something of a national media obsession. But out of the nearly 100 cases every year, the media only publicizes a couple, and we hear mostly about 5-4 splits on important cases. What about all of the other cases? What were the key inflection points in the court's history? Which justices are actually most central? Is it even possible to know by looking at only important decisions? We decided to approach the court from a new, more quantitative angle that is befitting of the age we live in: as a social network. We hope to provide a more comprehensive, quantitative view of the court that might help people supplement their analysis of the court. We present our data-sourcing, our approach of the problem, and our results below.

Data Collection

Although all of the supreme court data is publically available there is no resource which collectively displays all cases and gives a realistic view of the judicial system. What we attempted to do was just this. Collate all the data from wikipedia for the separate years and make one aggregate data set for all the years from 2000-2014. From here we can calculate some relevant metrics like which justices actually have the most centrist voting records and how the justices voted together.

Measuring Centrality

In this section of the article, we will discuss the theoretical aspects of taking our raw data and turning it into insights.

Data Structure

In order to measure justice voting centrality, we decided to compose our data into a graph. Each justice was counted as a vertex and the weighting of the edges between any two justices was the number of times that they voted the same way when they both chose to vote on the same case (we counted two justices voting the same way for different reasons as a half a vote the same way). By using raw numbers of times justices voted together, we aimed to lower scores of justices who chose to be less active on the court.

Algorithm

Determining centrality in networks is a key issue in network analysis and has significant uses in many fields. Possibly the most salient example of this is Google's PageRank algorithm, which helps the search engine identify the most relevant pages to return to users. Centrality algorithms use many different heuristics to assign scores to vertexes, including degree centrality, betweenness centrality, eigenvector centrality, and subgraph centrality.

We examined many different ways of calculating centrality and ultimately chose Laplacian centrality. It balanced the global positioning description of betweenness centrality with the ability of degree centrality to describe localities, and fit the best with our graph: a very densely connected graph with many weights close in size (almost all of our correlation percentages were between 65-85%).

The algorithm is as follows:
1) Define matrix W using the weights of the graphs as follows:

$$W_{ij} = \begin{cases} w_{ij} & : i \neq j \\ 0 & : i = j \end{cases}$$

2) Define matrix X using the sum of weights associated with each vertex:

$$X_{ij} = \begin{cases} 0 & : i \neq j \\ \sum_{i=0}^{n} W_i & : i = j \end{cases}$$

3) The Laplacian is defined as X-W, and the Laplacian Energy of the graph is the sum of the eigenvalues squared:

$$E = \sum_{i=0}^{n} \lambda_i^2$$

4) Energy can also be calculated without eigensolving the matrix(proof in reference paper):

$$E = \sum_{i=0}^{n} x_i^2 + 2 \sum_{i<j} w_{ij}^2$$

5) To define the centrality of any vertex, we remove it from the network and recalculate the remaining energy(E_l) and then divide the difference by the original energy((E-E_l)/E).

Graph Theory Description of the Algorithm

Laplacian centrality uses graph 2-walks[(1)] to measure the centrality of an algorithm. To count the number of possible 2-walks in a weighted graph we multiply consecutive edge weights. To understand this intuitively, think of an edge weight of value m as being m different unweighted edges between two vertices. In order to count the total number of 2-walks that go through a vertex v in graph G we can partition the set of such 2-walks into 3 types.

Let G be a graph and let be H be G with vertex v removed. Using the above simplified method for calculating laplacian energy without eigen-solving, we can algebraically show the difference in energy is a linear combination of the numbers of the 3 different types of two walks[2].

Type 1. *Closed 2-walks containing the vertex v: the number of such 2-walks is*

$$NW_2^C(v) = \sum_{y_i \in N(v)} w_{v,y_i}^2.$$

Type 2. *Non-closed 2-walks containing the vertex v as one of the end-vertices: the number of such 2-walks is*

$$NW_2^E(v) = \sum_{y_i \in N(v)} \left(\sum_{z_j \in \{N(y_i) - v\}} w_{v,y_i} w_{y_i,z_j} \right).$$

Type 3. *Non-closed 2-walks containing the vertex v as the middle point, the number of such 2-walks is*

$$NW_2^M(v) = \sum_{y_i, y_j \in N(v), y_i \neq y_j} w_{y_i,v} w_{v,y_j}.$$

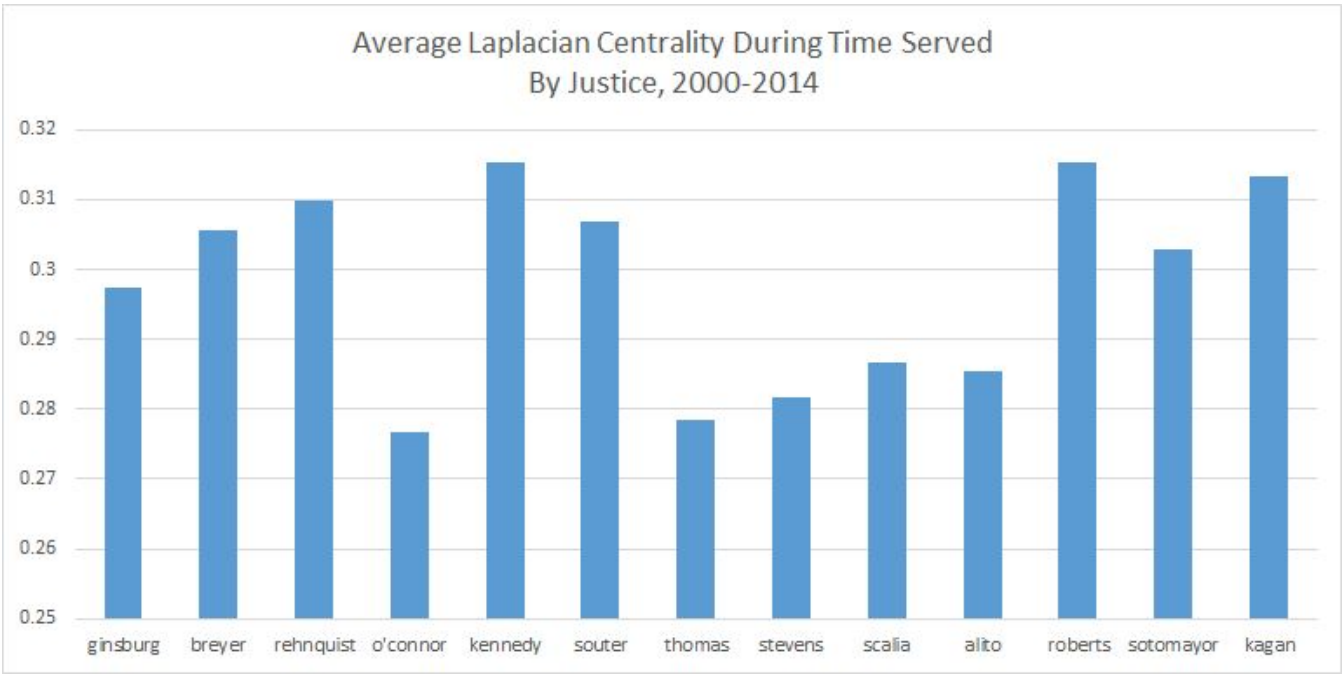
By measuring the reach of 2-walks from a vertex, Laplacian centrality strikes a good balance between the advantages of degree and betweenness centrality. The algorithm also factors in a larger environment than just a vertex's neighbors by counting 2-walks. Yet, by weighting closed 2-walks more than the other two types of 2-walks, the algorithm emphasizes the most local environment of v, which should intuitively be more important to its centrality score.

Implementation Details and Time Complexity

We implemented Laplacian centrality in Java using our own Graph objects but with a method that can parse a CSV with an adjacency matrix into a graph, making the algorithm very easy to run test. Our implementation uses the first method we introduced to calculate Laplacian energy and runs in $O(n^2)$ time. However, Laplacian energy can theoretically be calculated in $O(n*d^2))$, where d is the maximum degree of a vertex in the graph.

Results:

Below is the results we got, along with links to all of our data as well as to code for our scraping, cleaning, and algorithms. We will leave interpreting the results up to you!



Centrality Score by Year and Justice

	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014
Justice Ginsburg	0.297	0.303	0.313	0.315	0.301	0.297	0.298	0.280	0.290	0.313	0.277	0.265	0.300	0.291	0.320
Justice Breyer	0.292	0.304	0.300	0.311	0.318	0.301	0.303	0.310	0.307	0.305	0.296	0.289	0.310	0.308	0.332
Justice Rehnquist	0.308	0.309	0.317	0.313	0.303										
Justice O'connor	0.315	0.300	0.311	0.321	0.323	0.090									
Justice Kennedy	0.317	0.320	0.308	0.303	0.316	0.304	0.325	0.312	0.326	0.318	0.321	0.326	0.310	0.308	0.317
Justice Souter	0.316	0.311	0.317	0.302	0.300	0.308	0.303	0.308	0.298						
Justice Thomas	0.288	0.278	0.273	0.273	0.274	0.288	0.275	0.276	0.291	0.279	0.272	0.301	0.276	0.298	0.235
Justice Stevens	0.273	0.298	0.272	0.288	0.279	0.295	0.265	0.287	0.295	0.264					
Justice Scalia	0.287	0.270	0.288	0.268	0.280	0.310	0.287	0.281	0.302	0.280	0.292	0.292	0.289	0.298	0.276
Justice Alito						0.201	0.320	0.314	0.287	0.296	0.300	0.302	0.282	0.283	0.269
Justice Roberts						0.320	0.318	0.329	0.298	0.322	0.325	0.324	0.313	0.306	0.298
Justice Sotomayor										0.318	0.299	0.290	0.302	0.290	0.318
Justice Kagan											0.311	0.305	0.310	0.315	0.326

Percentage Over or Under Average Centrality Score on Court Each Year

	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014
ginsburg	-1%	1%	4%	5%	1%	9%	0%	-7%	-3%	4%	-7%	-12%	0%	-3%	7%
breyer	-2%	2%	0%	4%	6%	11%	1%	3%	3%	2%	-1%	-4%	4%	3%	11%
rehnquist	3%	3%	6%	4%	1%										
o'connor	5%	0%	4%	7%	8%	-67%									
kennedy	6%	7%	3%	1%	5%	12%	9%	4%	9%	6%	7%	9%	4%	3%	6%
souter	6%	4%	6%	1%	0%	13%	1%	3%	-1%						
thomas	-4%	-7%	-9%	-9%	-9%	6%	-8%	-8%	-3%	-7%	-9%	1%	-8%	0%	-21%
stevens	-9%	0%	-9%	-4%	-7%	9%	-11%	-4%	-1%	-12%					
scalia		-10%	-4%	-10%	-6%	14%	-4%	-6%	1%	-6%	-2%	-3%	-3%	-1%	-8%
alito						-26%	7%	5%	-4%	-1%	0%	1%	-6%	-5%	-10%
roberts						18%	6%	10%	-1%	8%	9%	8%	5%	2%	0%
sotomayor										6%	0%	-3%	1%	-3%	6%
kagan											4%	2%	4%	5%	9%

A new, much more interesting problem:

Over the course of this project, we uncovered an incredibly interesting data science problem that we'll be building on in the blog. One of the reasons we broke out the centrality of justices by years instead of trying to do a comprehensive set was because we had no way of estimating how two justices who never sat on the court together would have voted together. In other words, we were having trouble predicting weights in a graph where weights represented correlations. We hope to really break this problem down and come up with machine learning

algorithms for best estimating such missing correlations in a graph and seeing if we can apply those techniques to real problems in a future post.

Footnotes:
(1) A 2-walk is defined as any path through 2 edges between connected vertices in a graph
(2) w_ij is the edge weight between i and j, and N(v) is the set of all of the neighbors of v

Data Set:
https://github.com/kbharathala/SCJusticeGraphs/

References:
http://www.math.wvu.edu/~cqzhang/Publication-files/my-paper/INS-2012-Laplacian-W.pdf