Analyzing code contributions to the CPython project using NetworkX and Matplotlib

PyData Barcelona 2017

Jordi Torrents

Department of Sociology
University of Barcelona

May 17, 2017

Contents

Theoretical Approaches to Cooperation

Macro level approach

Cooperation as a macro level phenomenon in which the center of analysis is the collective or group (Marx, 1990; Adler and Heckscher, 2006; Adler, 2015).

- Focus on large organizations and groups: Collaborative Communities
  - Shared values and goals
  - Generalized trust
  - Authority forms

Theoretical Approaches to Cooperation

Macro level approach

Cooperation as a macro level phenomenon in which the center of analysis is the collective or group (Marx, 1990; Adler and Heckscher, 2006; Adler, 2015).

- Focus on large organizations and groups: Collaborative Communities
  - Shared values and goals
  - Generalized trust
  - Authority forms

Micro level approach

Cooperation as a micro level phenomenon in which the center of analysis is the dyad (Axelrod and Hamilton, 1981; Watts, 1999; Eguíluz et al., 2005).

- Reductionist approach: Cooperation as an atomic process.
  - Strategic dyadic interactions.
  - Agent-based models.
  - Payoffs of different strategies.

Theoretical Approaches to Cooperation

Macro level approach

Cooperation as a macro level phenomenon in which the center of analysis is the collective or group (Marx, 1990; Adler and Heckscher, 2006; Adler, 2015).

- Focus on large organizations and groups: Collaborative Communities
  - Shared values and goals
  - Generalized trust
  - Authority forms

Micro level approach

Cooperation as a micro level phenomenon in which the center of analysis is the dyad (Axelrod and Hamilton, 1981; Watts, 1999; Eguíluz et al., 2005).

- Reductionist approach: Cooperation as an atomic process.
  - Strategic dyadic interactions.
  - Agent-based models.
  - Payoffs of different strategies.

A Meso level approach to Cooperation

Focus on Cooperation networks: patterns of relations that direct producers establish in the production process.

- Structural approach, that is, a network approach.
  - Sub-groups that are more connected internally than with the rest of the network.
  - Longitudinal analysis of the formation and dissolution of these groups.
  - Key mechanisms to explain and understand large scale cooperation.

Group Cohesion in the sociological literature

Central concept that has a long and illustrious history in sociology. Its use in most sociological research has been ambiguous at best (Moody and White, 2003):

1. sloppy definitions of cohesion with lack of generality.
2. grounded mostly in intuition and common sense.

Group cohesion (Doreian and Fararo, 1998) can be divided analytically into:

Ideational component based on the members' identification with a collectivity.

Relational component based on the patterns of connections among members.

Group Cohesion in the sociological literature

Central concept that has a long and illustrious history in sociology. Its use in most sociological research has been ambiguous at best (Moody and White, 2003):

1. sloppy definitions of cohesion with lack of generality.
2. grounded mostly in intuition and common sense.

Group cohesion (Doreian and Fararo, 1998) can be divided analytically into:

Ideational component based on the members' identification with a collectivity.

Relational component based on the patterns of connections among members.

The relational component of group cohesion has been the focus of Social Network Analysis.

Network theory measures used to define group cohesion

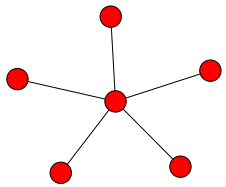
Classical measures cliques, clans, clubs, k-cores, lambda sets, ...

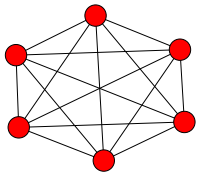
Community algorithms detect groups of nodes more densely connected among them than with the rest of the network.

Neither the classical approaches nor new developments in community analysis work well in empirical analysis of group cohesion.

Key properties that a cohesion measures should have

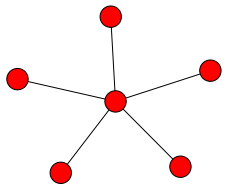
Robustness its qualification as a group should not be dependent on the actions of a single individual.

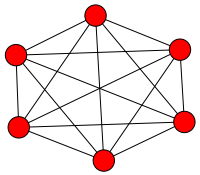




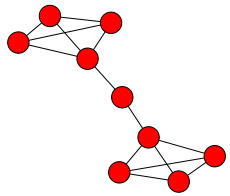
Key properties that a cohesion measures should have

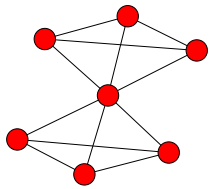
Robustness its qualification as a group should not be dependent on the actions of a single individual.





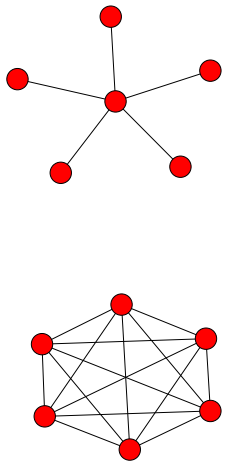
Overlap a cohesion measure should allow some actors to be part of more than one cohesive subgroup.



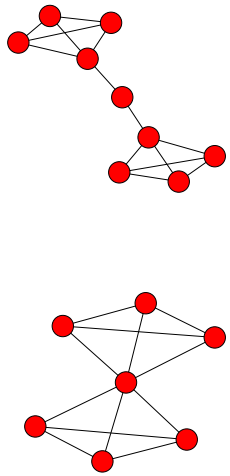


Key properties that a cohesion measures should have

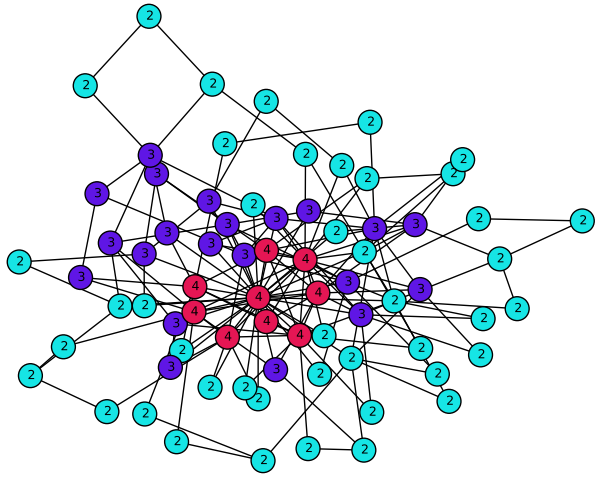
Robustness its qualification as a group should not be dependent on the actions of a single individual.



Overlap a cohesion measure should allow some actors to be part of more than one cohesive subgroup.



Hierarchical highly cohesive subgroups are nested inside less cohesive ones.



The structural cohesion model

The structural cohesion model (White and Harary, 2001; Moody and White, 2003) is based on two mathematically equivalent definitions of cohesion.

Precise definition of group cohesion based on node connectivity

- a group's structural cohesion is equal to the minimum number of actors who, if removed from the group, would disconnect the group.
- a group's structural cohesion is equal to the minimum number of node independent paths linking each pair of actors in the group.

This equivalence relation has a deep sociological meaning because it allows to define structural cohesion in terms of:

- the difficulty to pull a group apart by removing actors.
- multiple relations between actors that keep a group together.

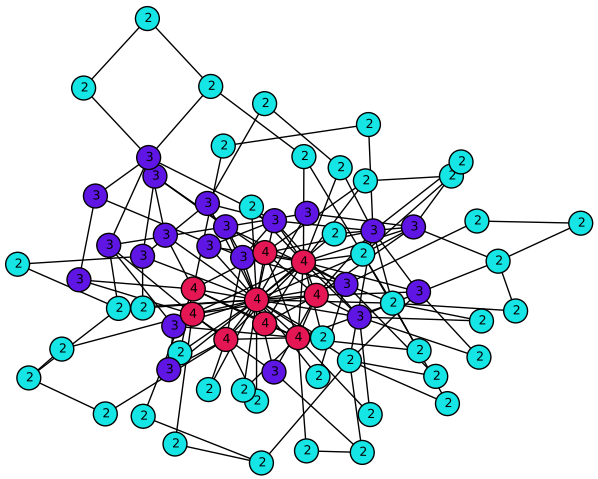
Components and *k*-components

Component

A *component* of a graph *G* is a maximal connected subgraph, which means that there is at least one path between any two nodes in that subgraph.

k-components

A *k*-component is a maximal subgraph that has, at least, node connectivity *k*: we need to remove at least *k* nodes to break it into more components.



- **Red Nodes** form a 4-component: we need to remove 4 nodes to disconnect it.
- **Purple Nodes** form a 3-component along with red nodes: we need to remove 3 nodes to disconnect it.
- **Blue Nodes** form a 2-component along with red and purple nodes: we need to remove 2 nodes to disconnect it.

Free and Open Source Projects: Debian and Python

Free Software, broadly defined, is computer software that allows users to run, copy, distribute, study, change and improve it.

The Debian Project: 1999-2012

- A free Operating System
- 392 developers in 1999, 1435 in 2012
- 2876 programs in 1999, 10469 in 2012
- Widely used in servers (google), desktops (Ubuntu) and embedded devices (raspberry pi)

The Python project: 1999-2014

- A free Programming Language
- 9 developers in 1999, 62 in 2014
- 1137 files in 1999, 2134 in 2014
- Widely used in web development (reddit, youtube) and scientific computing

Cooperation Networks and Null Models

My modeling strategy to capture the patterns of relations among developers in these two projects is to focus on the actual contributions of each developer to the project.

Building Cooperation Networks

- Focus on the patterns of relations among developers in the productive process.
- Cooperation networks are bipartite or two mode; node sets are developers and programs/files and edges only link nodes from opposite sets.
- A developer is linked to the package (in Debian) or source code file (in Python) that she works on.
- We have complete electronic records of all package uplads to Debain and all sorce code file edits in Python.

Cooperation Networks and Null Models

My modeling strategy to capture the patterns of relations among developers in these two projects is to focus on the actual contributions of each developer to the project.

Building Cooperation Networks

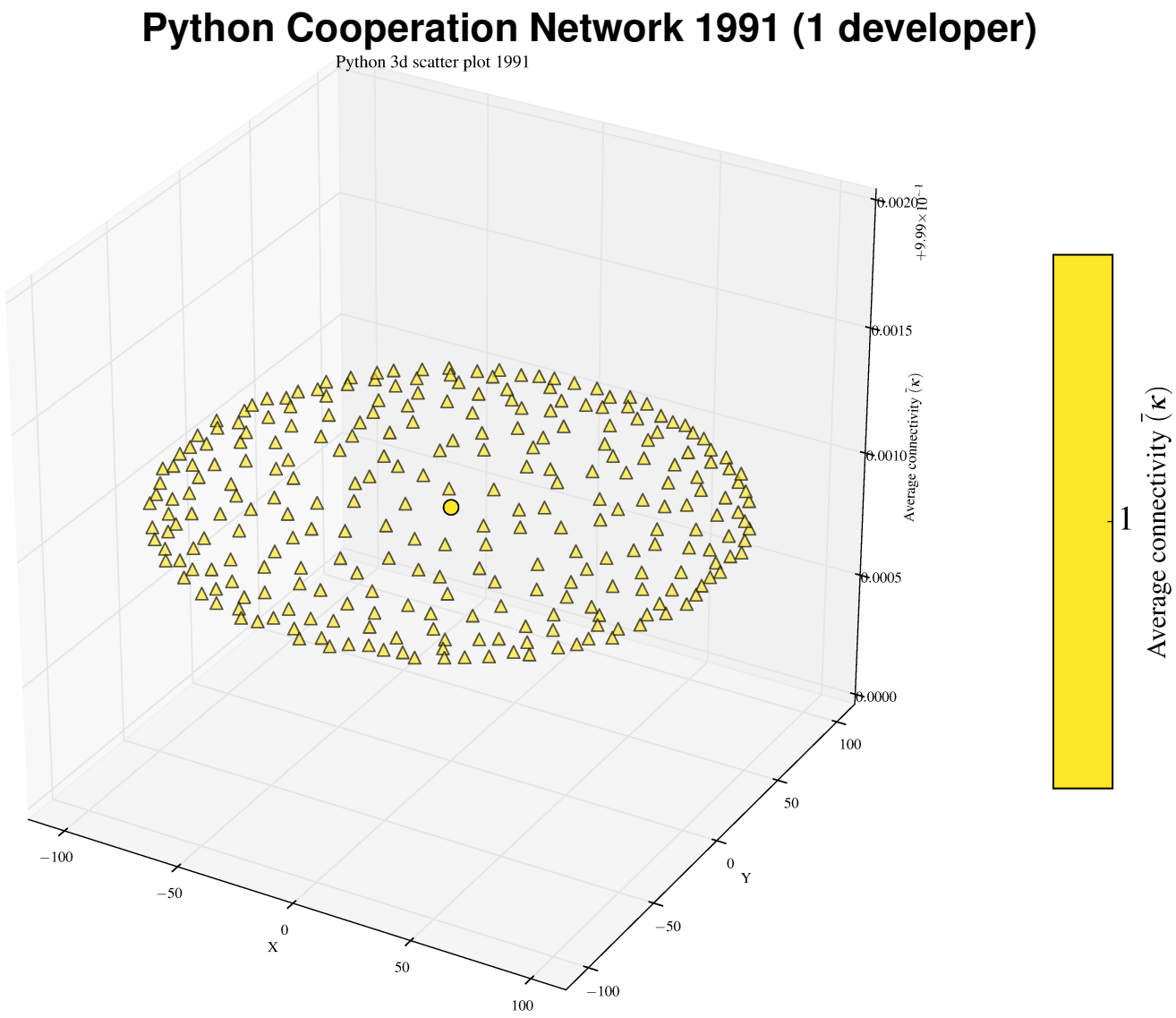
- Focus on the patterns of relations among developers in the productive process.
- Cooperation networks are bipartite or two mode; node sets are developers and programs/files and edges only link nodes from opposite sets.
- A developer is linked to the package (in Debian) or source code file (in Python) that she works on.
- We have complete electronic records of all package uplads to Debain and all sorce code file edits in Python.

Building Suitable Null models

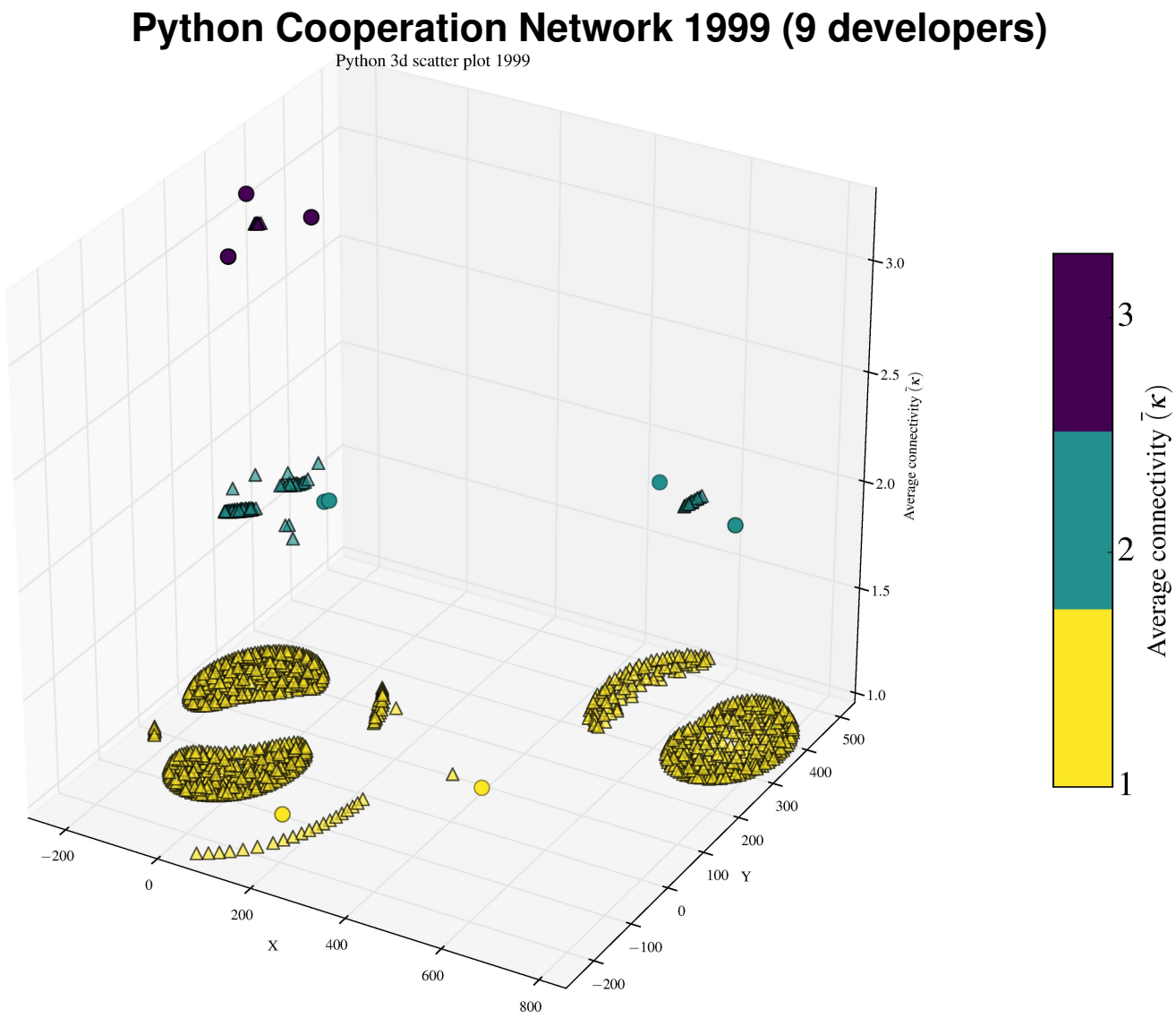
- We need to compare the statistical measures obtained from the actual networks with a suitable null model in order to assert that what we observe is not the result of pure chance.

Configuration Model assigns at random developers to packages, or developers to source code files, maintaining the concrete skewed distribution of packages by developer and files by developer observed in the actual networks. That is, the degree distribution.

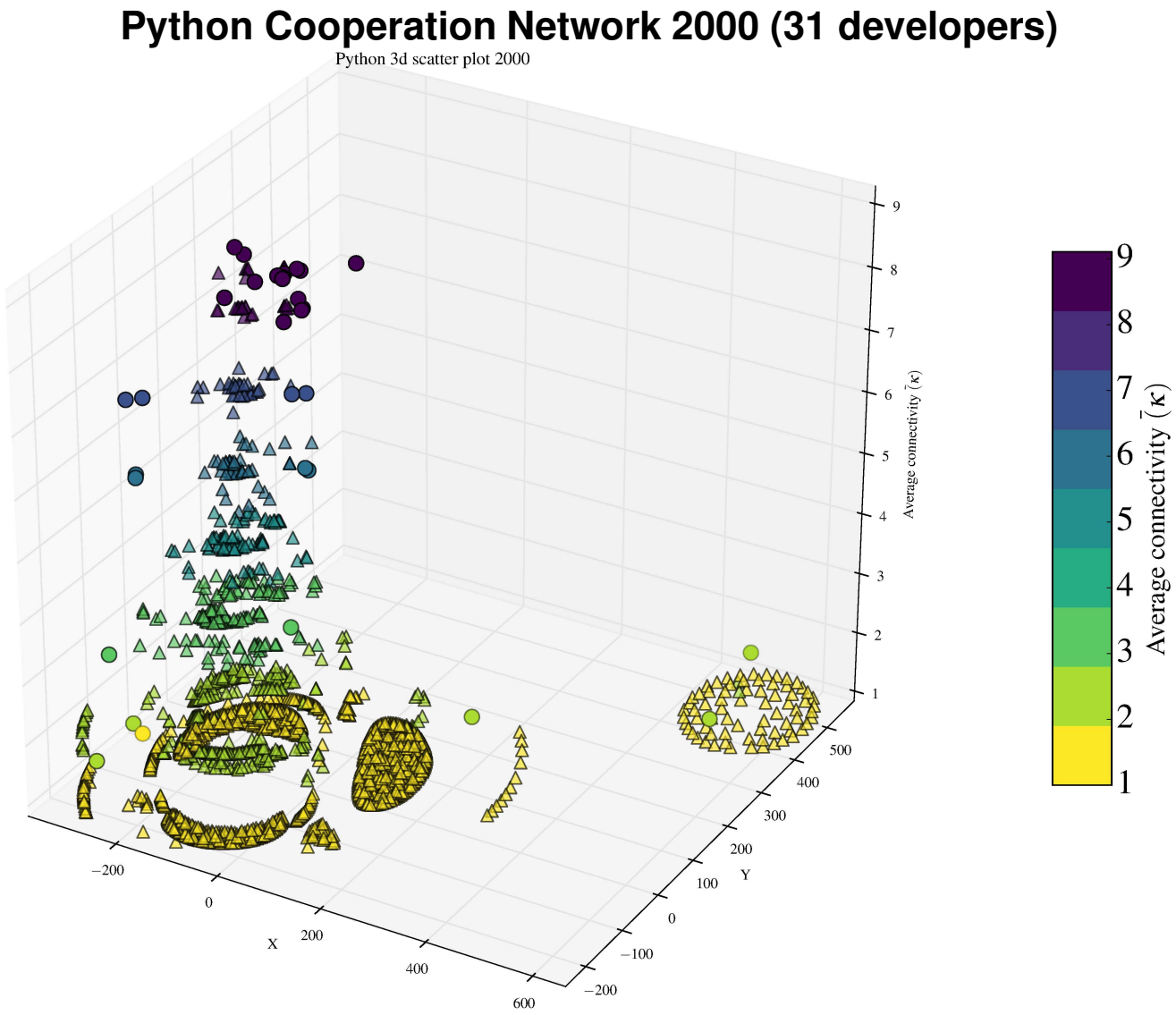
Graphical Representation of the Structural Cohesion Analysis



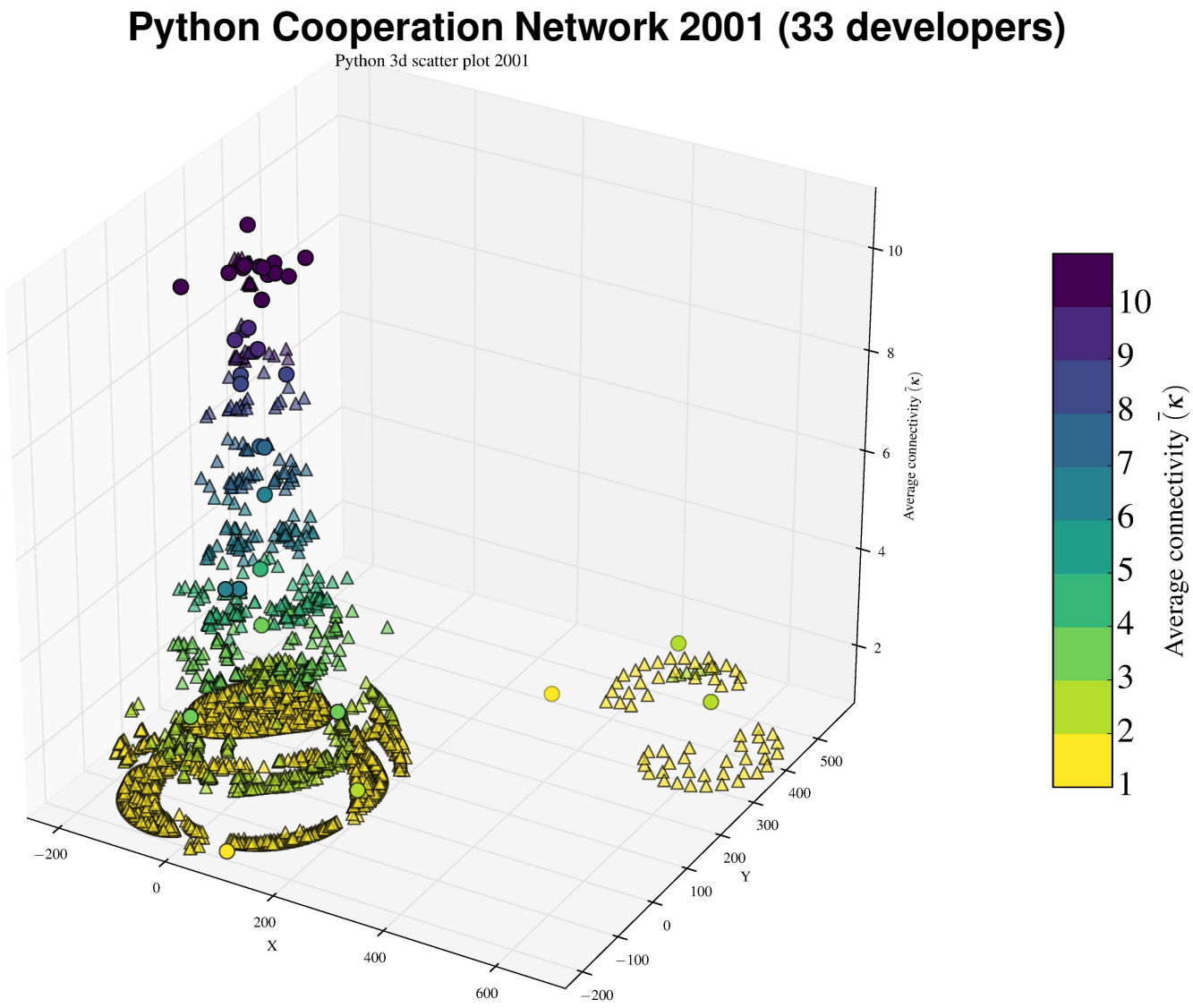
Graphical Representation of the Structural Cohesion Analysis



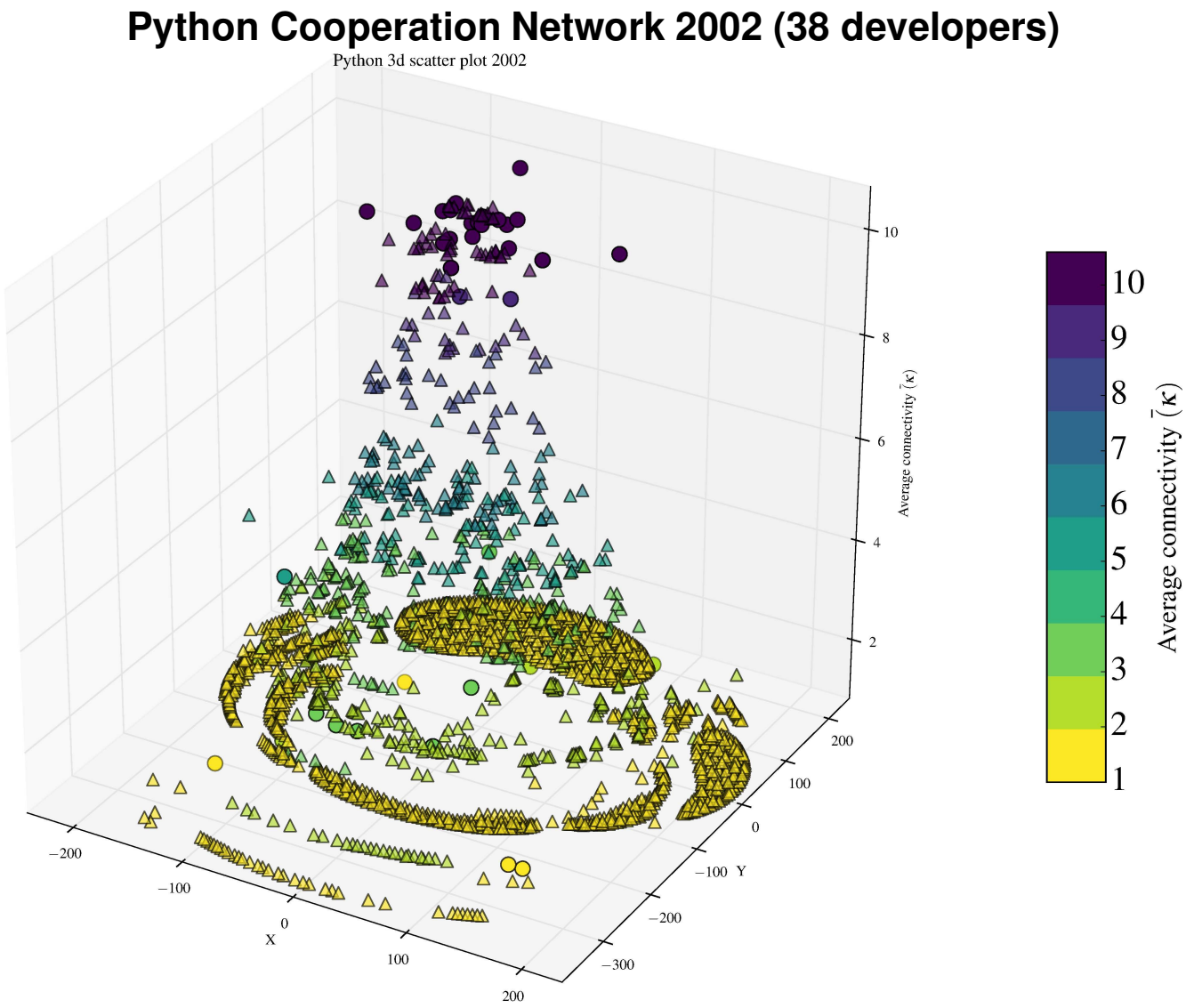
Graphical Representation of the Structural Cohesion Analysis



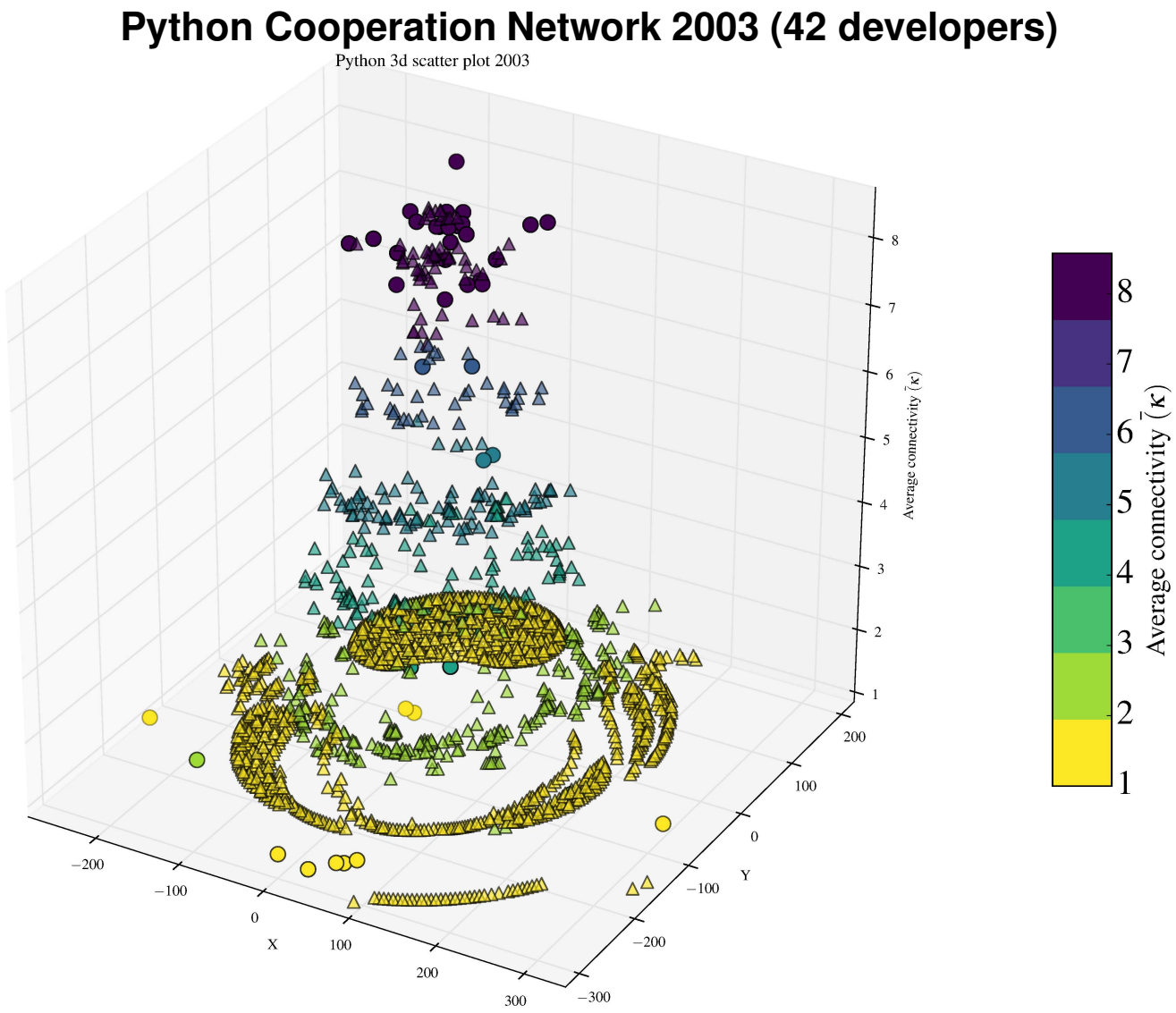
Graphical Representation of the Structural Cohesion Analysis



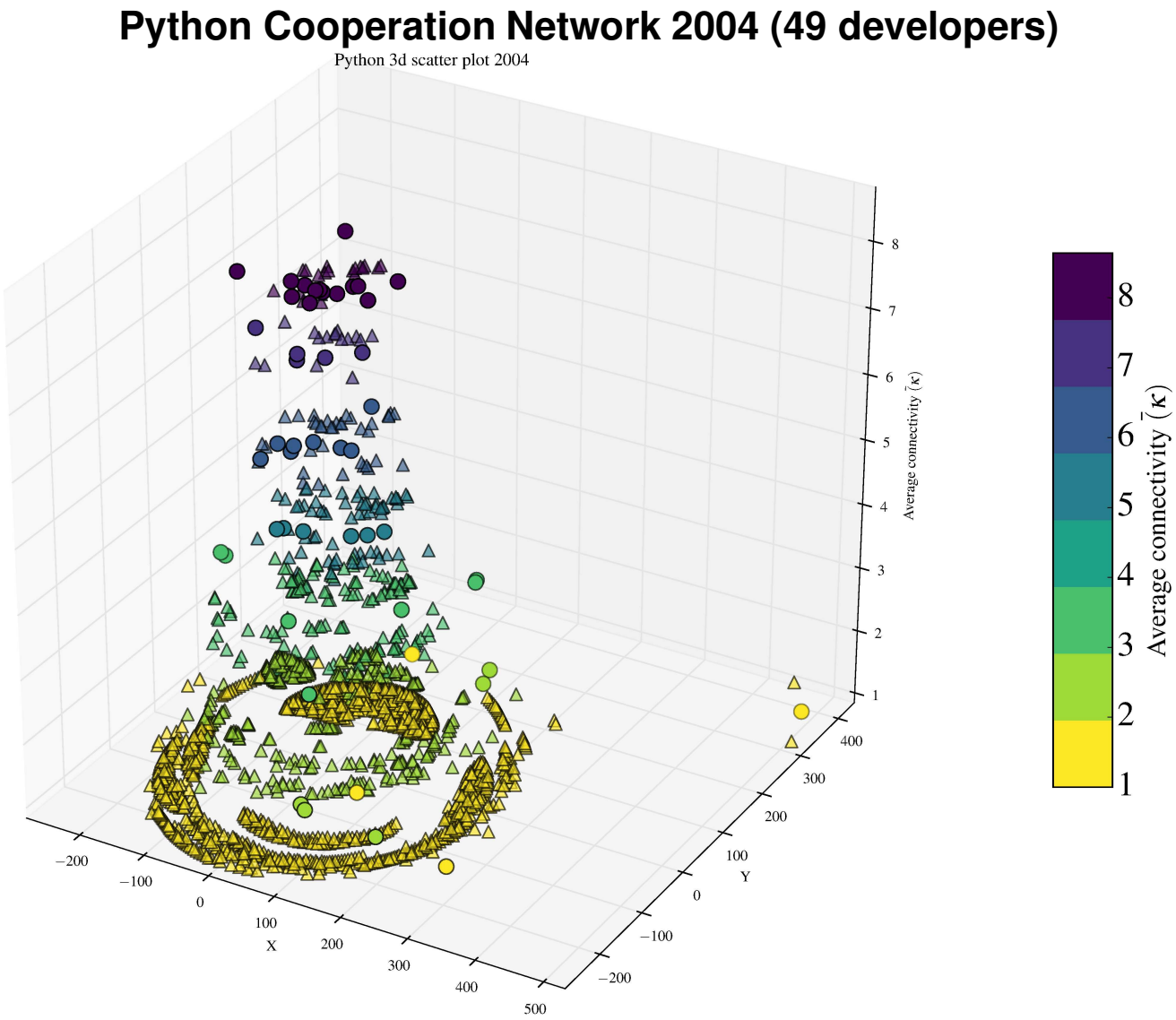
Graphical Representation of the Structural Cohesion Analysis



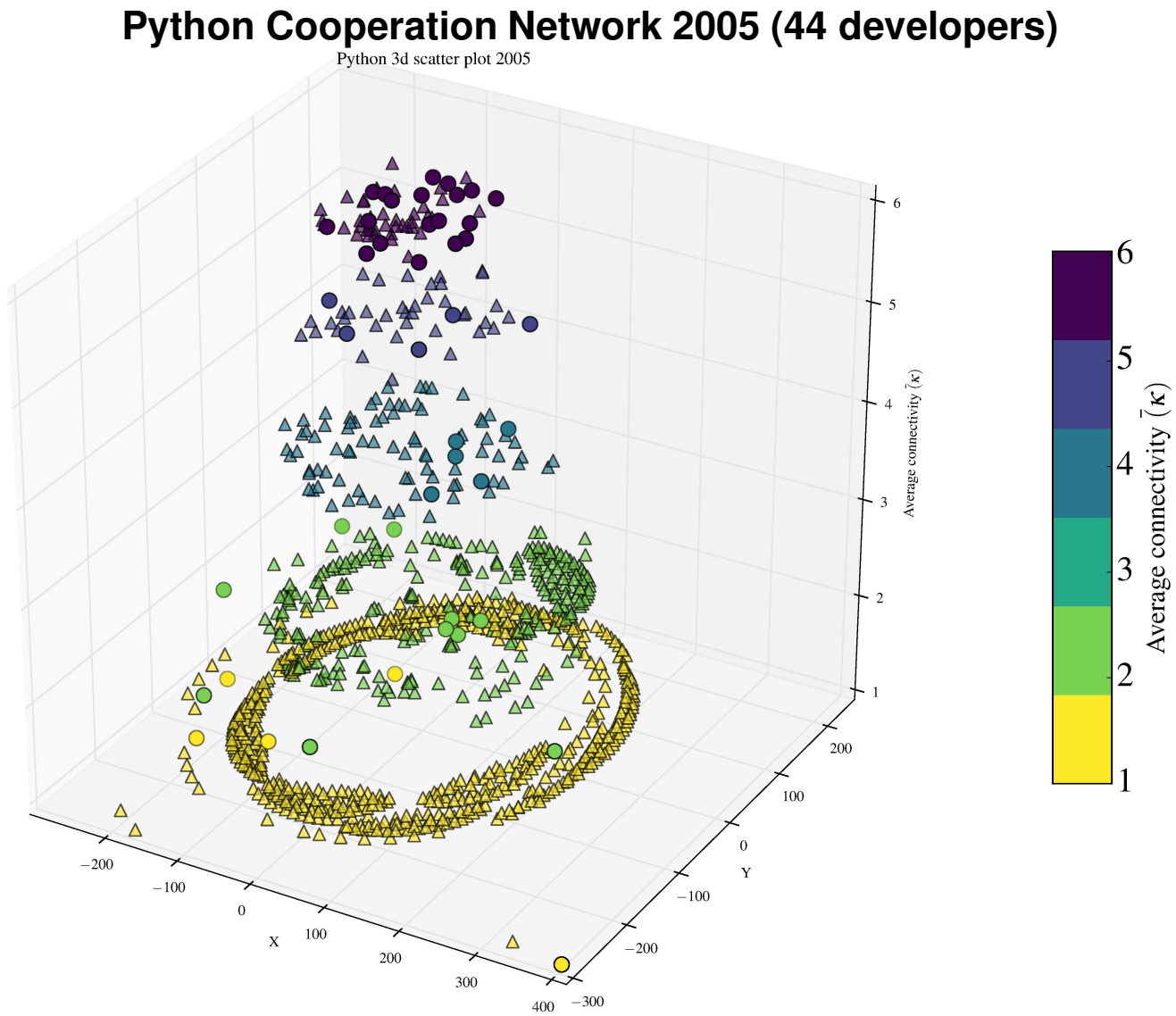
Graphical Representation of the Structural Cohesion Analysis



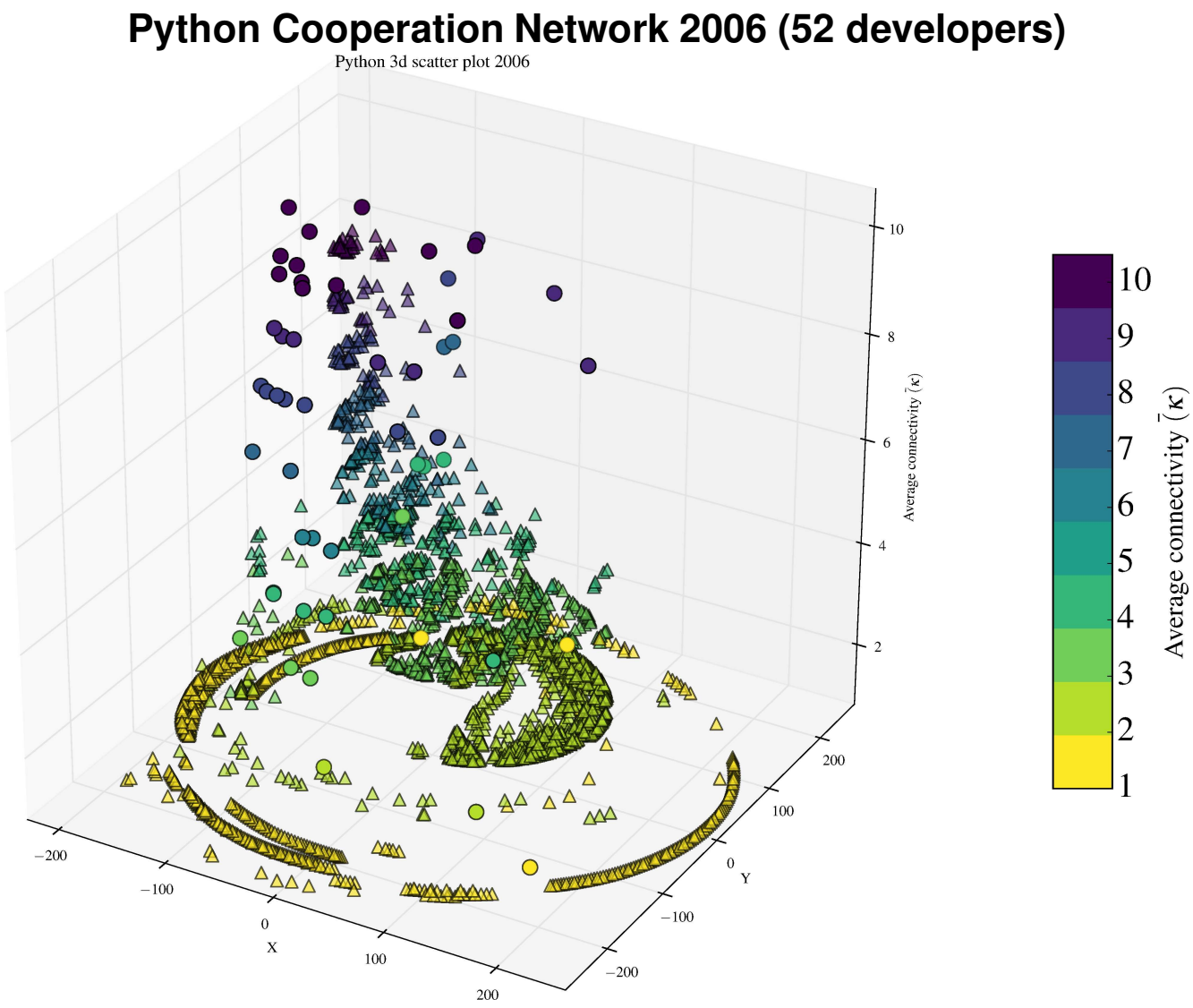
Graphical Representation of the Structural Cohesion Analysis



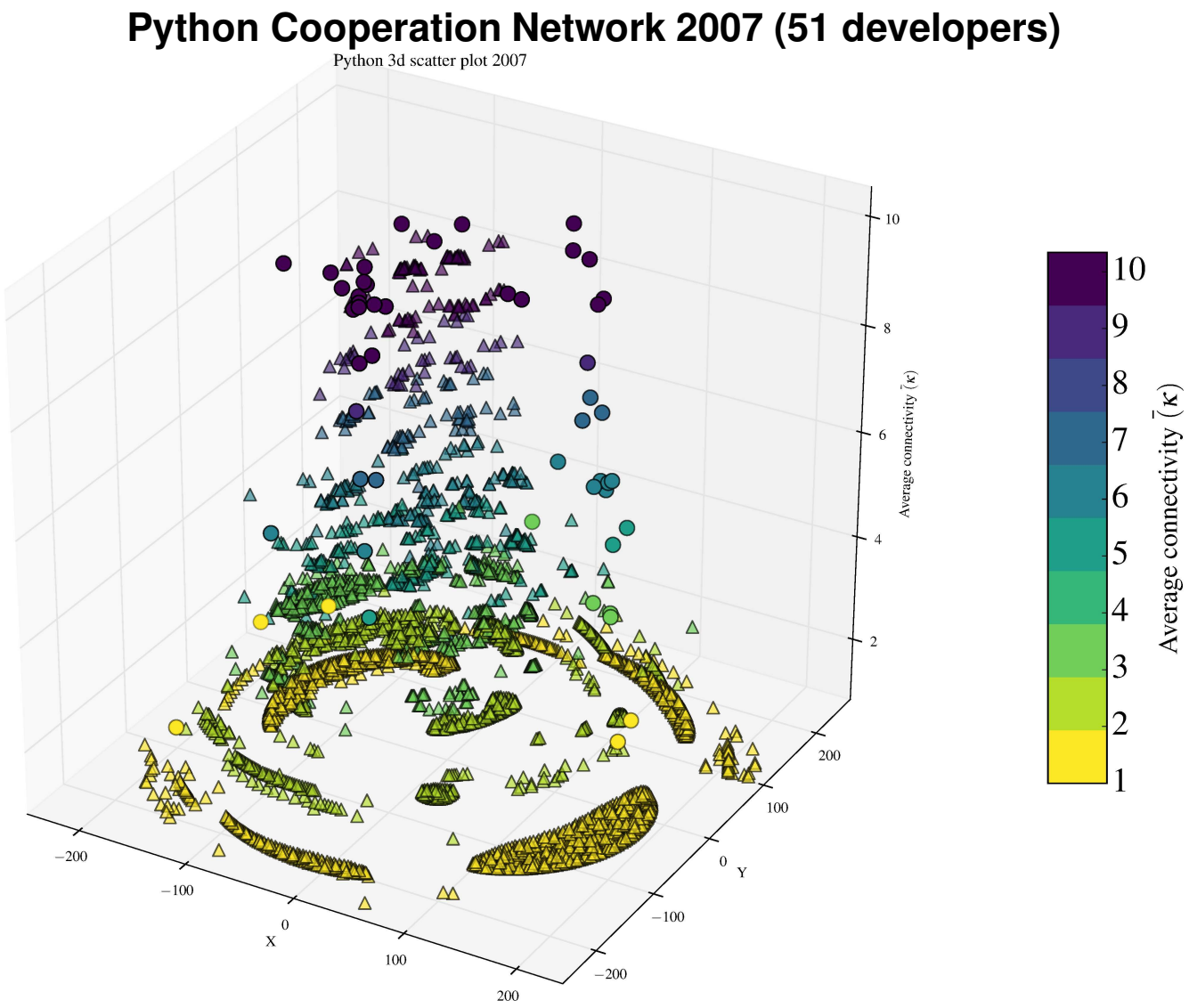
Graphical Representation of the Structural Cohesion Analysis



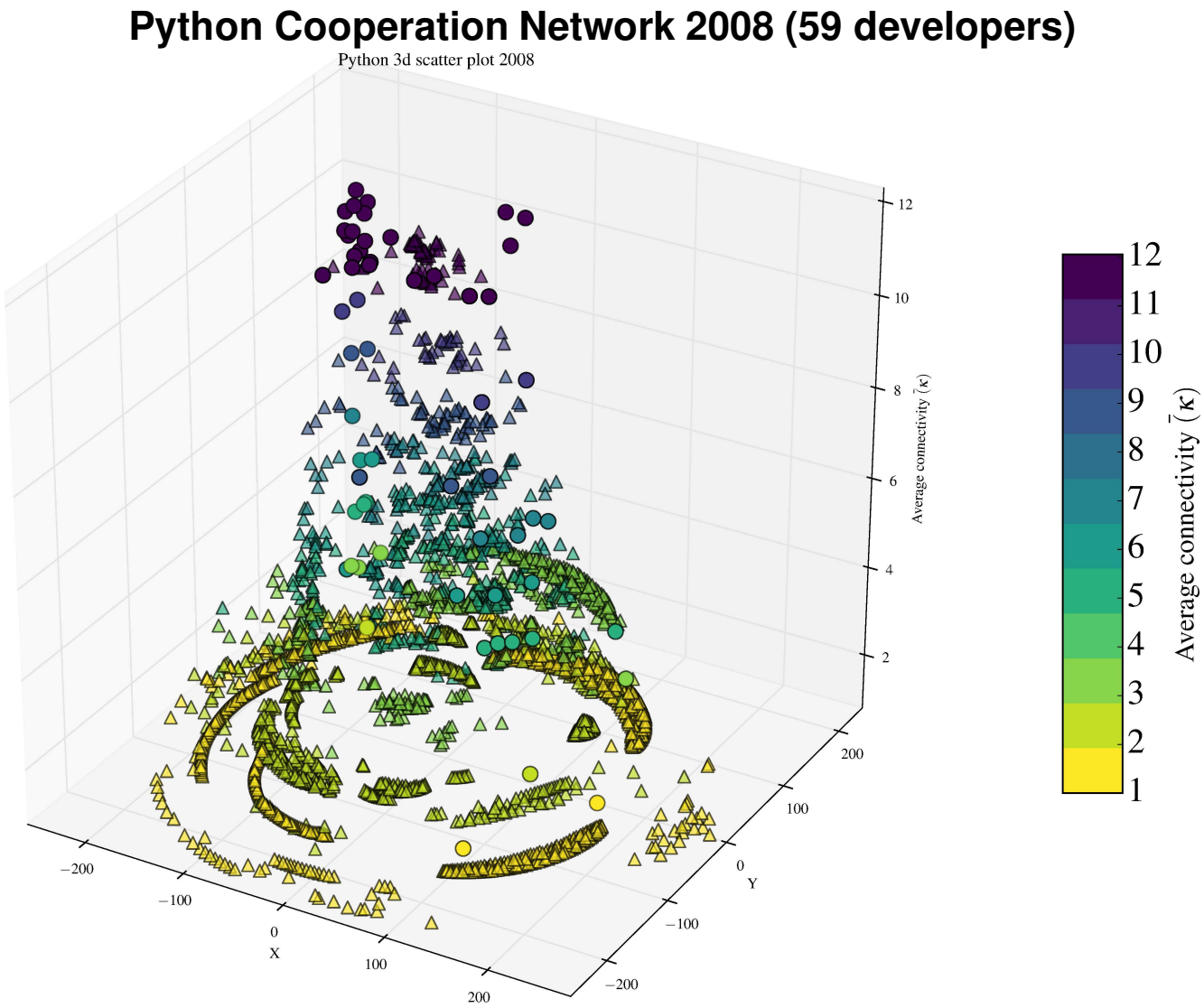
Graphical Representation of the Structural Cohesion Analysis



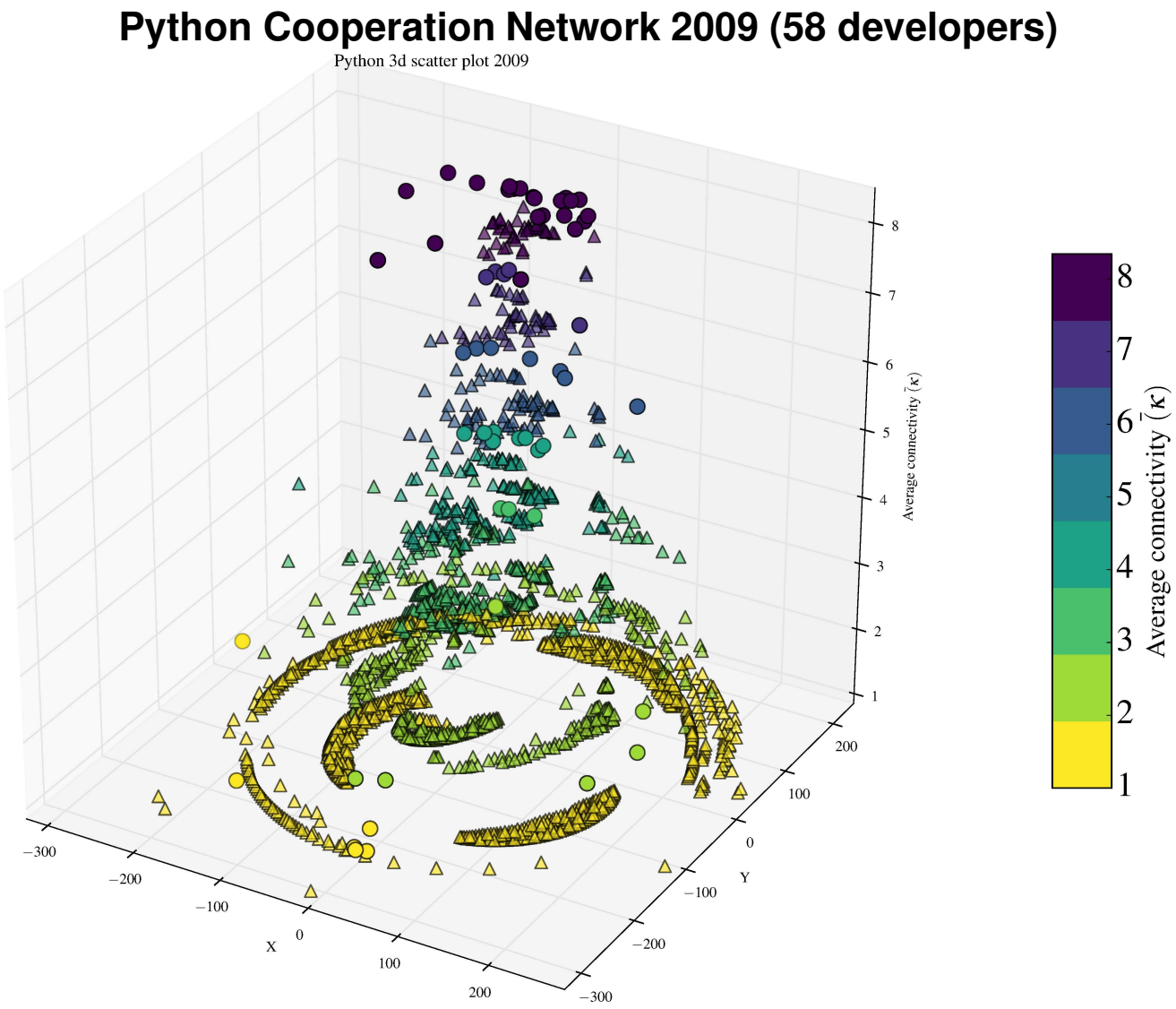
Graphical Representation of the Structural Cohesion Analysis



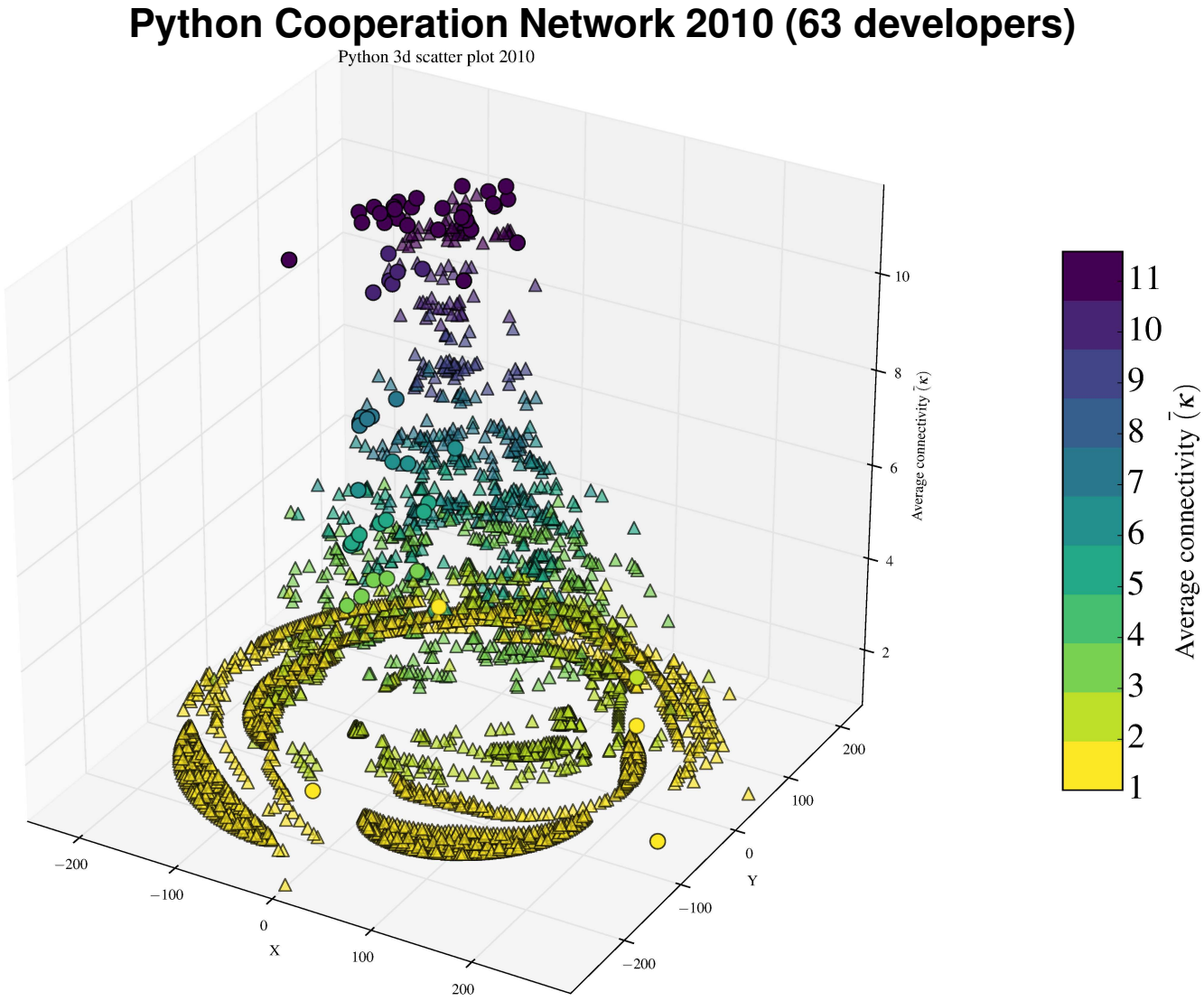
Graphical Representation of the Structural Cohesion Analysis



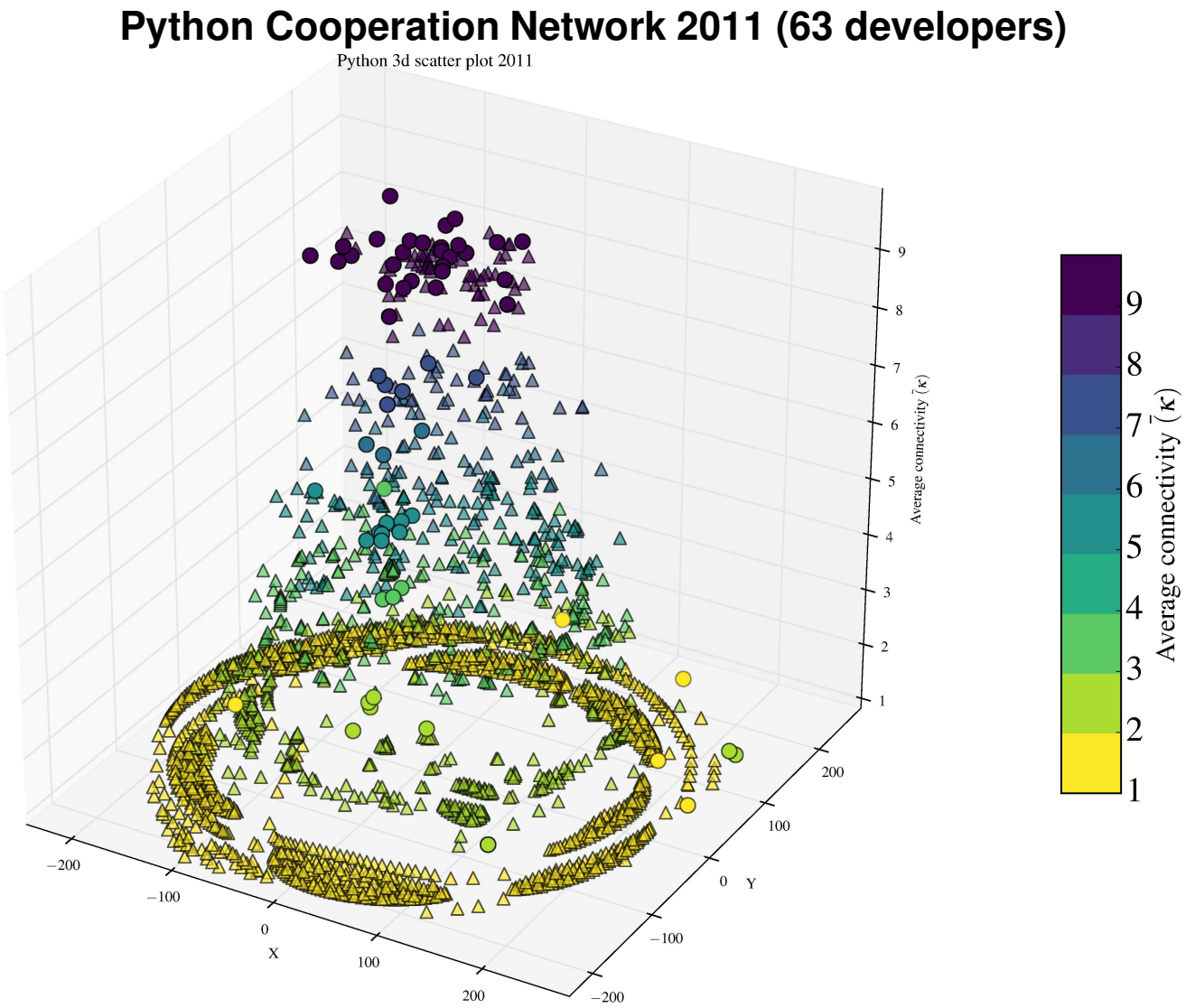
Graphical Representation of the Structural Cohesion Analysis



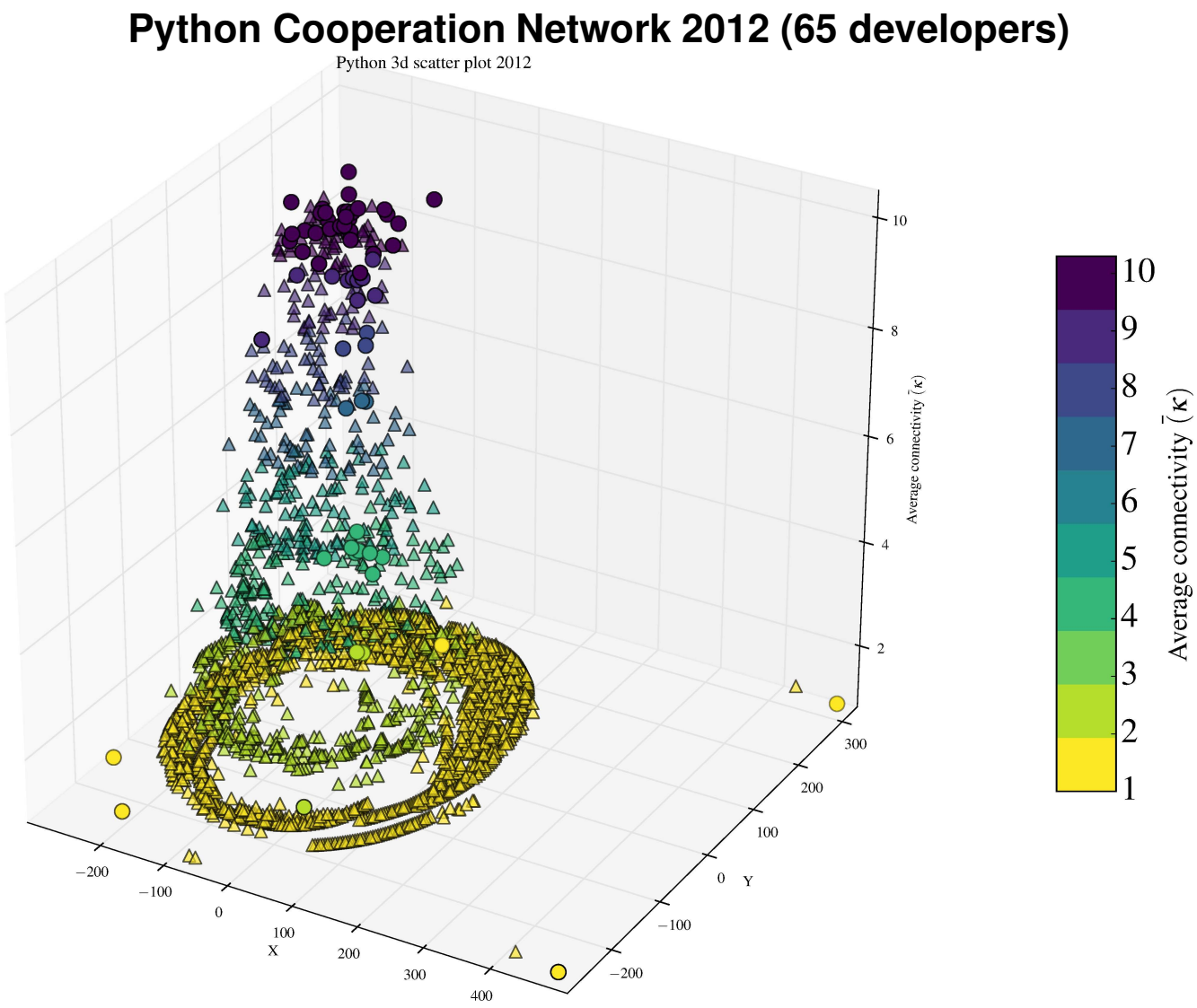
Graphical Representation of the Structural Cohesion Analysis



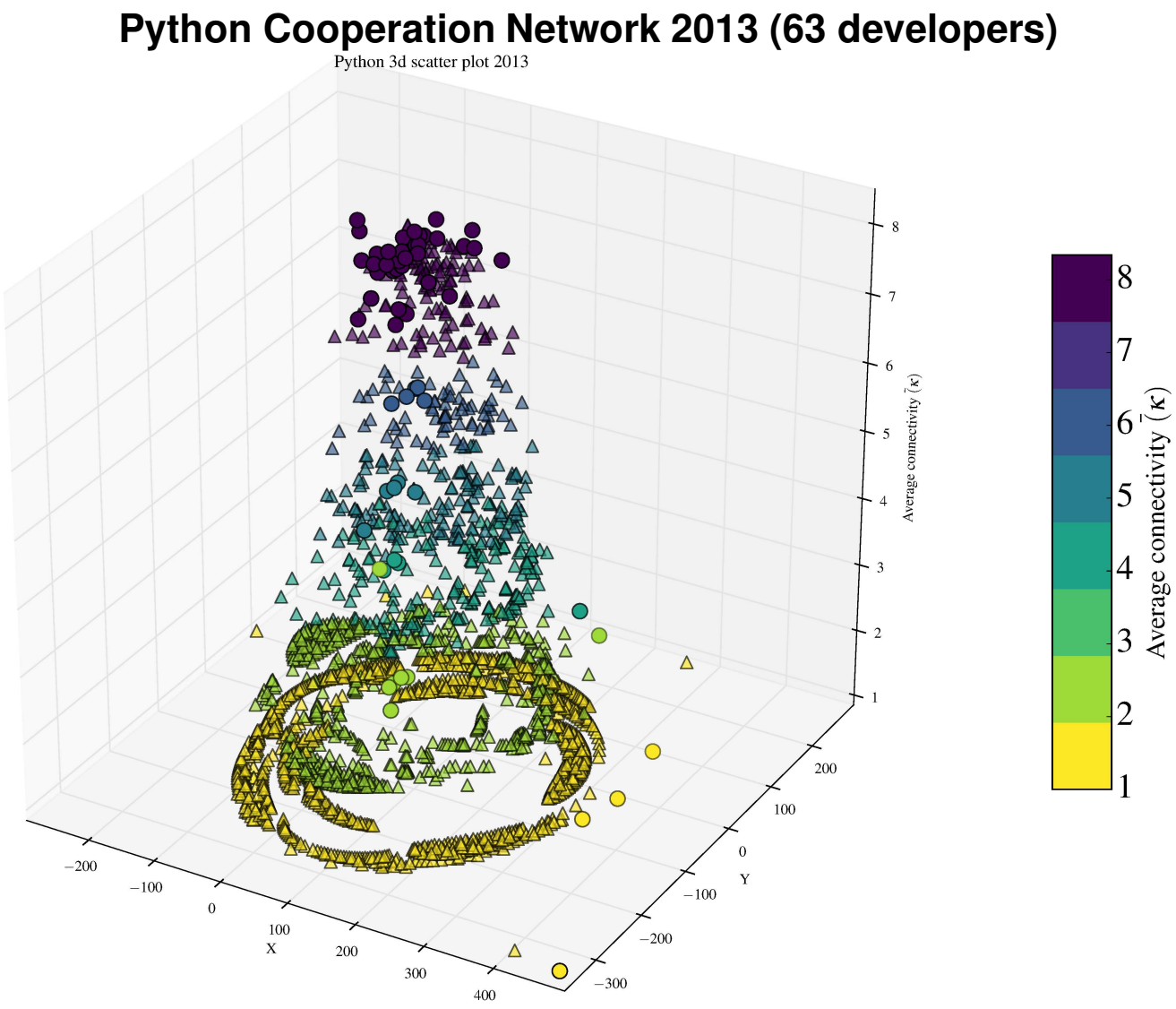
Graphical Representation of the Structural Cohesion Analysis



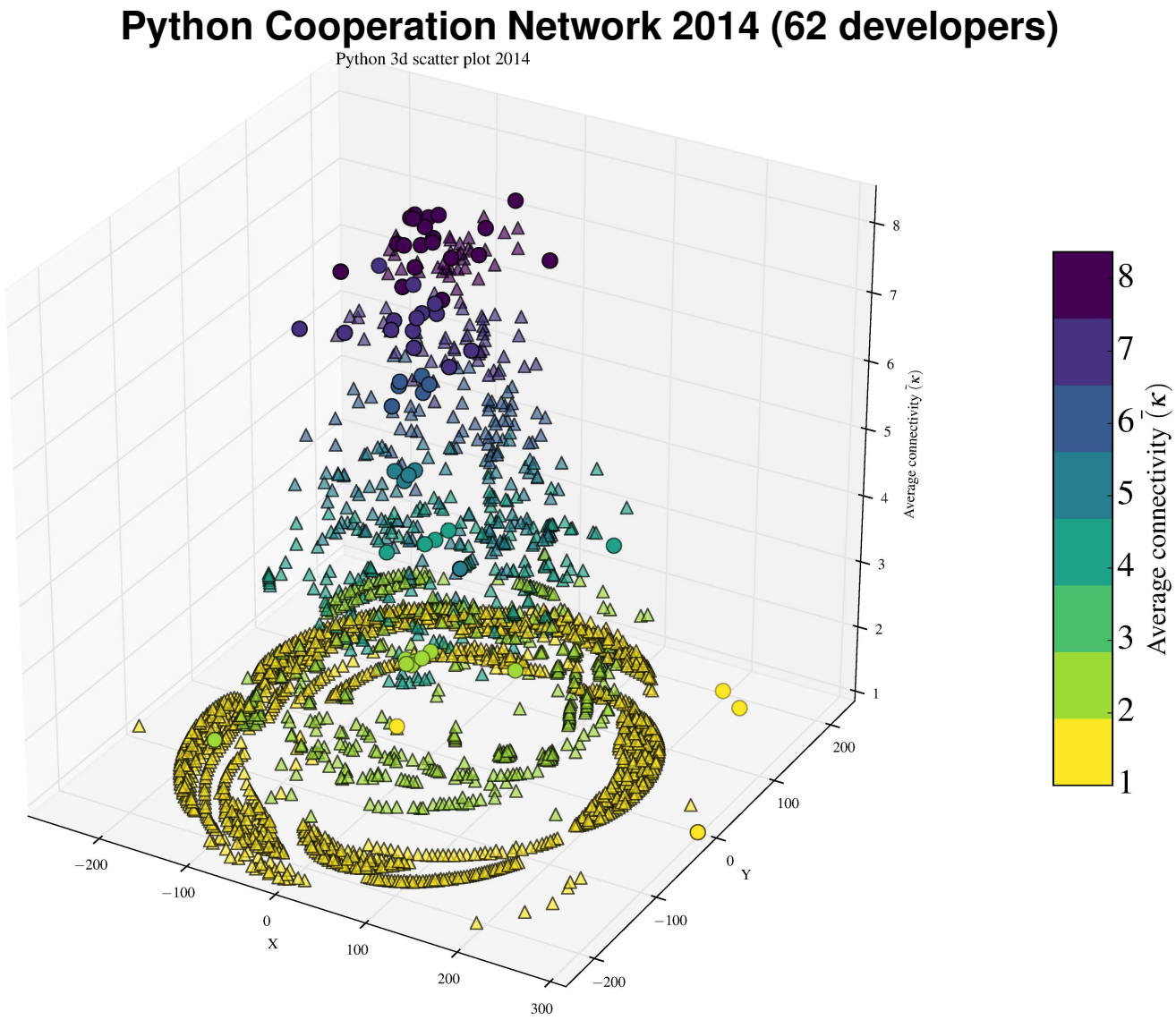
Graphical Representation of the Structural Cohesion Analysis



Graphical Representation of the Structural Cohesion Analysis

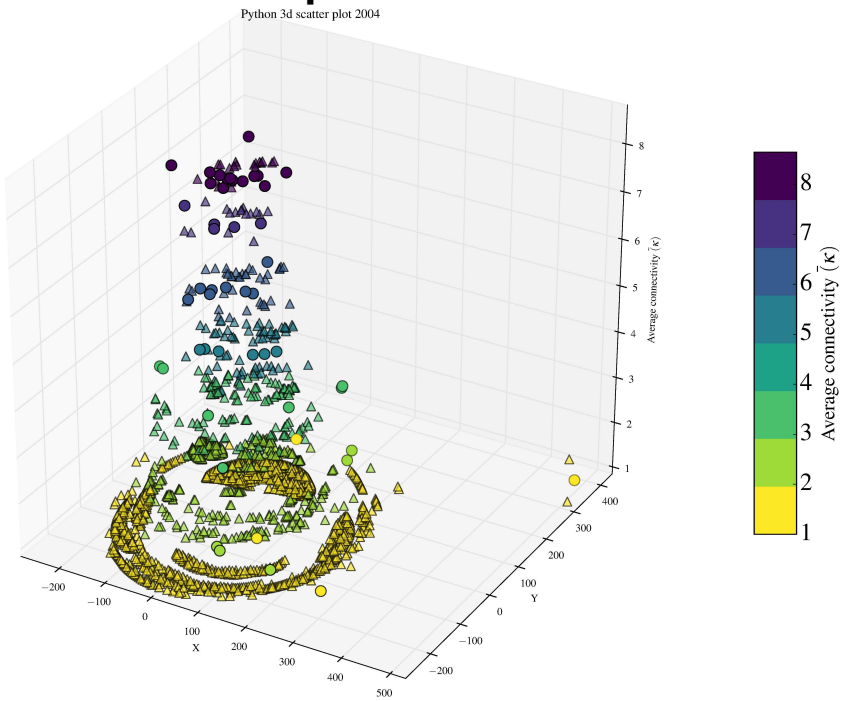


Graphical Representation of the Structural Cohesion Analysis

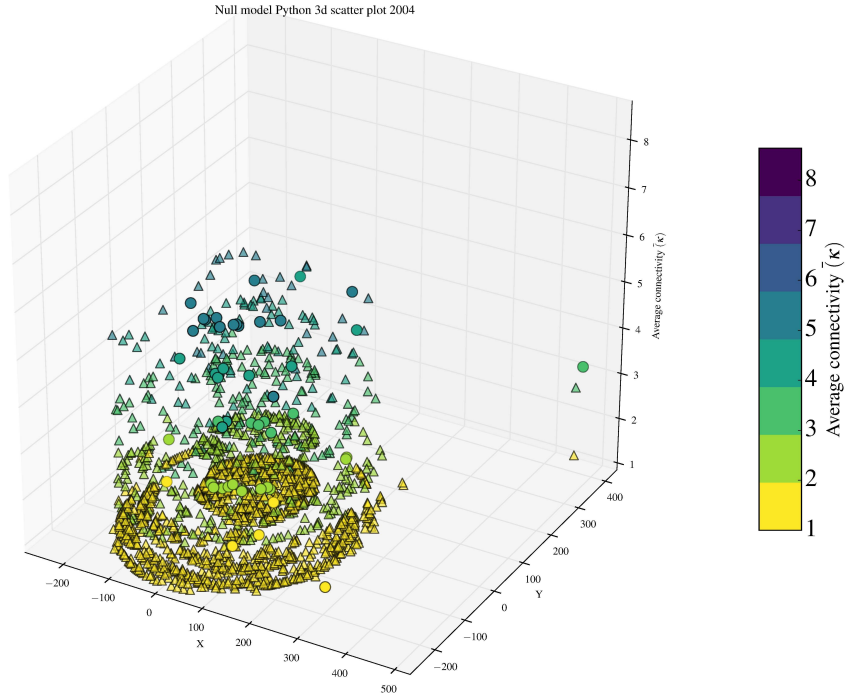


Comparison with null models

Actual Cooperation Network 2004



Null model



The Hierarchical Structure of FOSS projects

Free and Open Source Software (FOSS) communities have attracted a lot of attention from researchers. Academic efforts took mainly two directions:

Reconcile with neoclassical economy

- Mainly focused on the individual motivations of the participants. What is usually referred as “microfundaments”.
- Trying to explain motivations in terms of of rational self-interested individuals to match the dominant economic accounts.
- Proposed that individuals

The Hierarchical Structure of FOSS projects

Free and Open Source Software (FOSS) communities have attracted a lot of attention from researchers. Academic efforts took mainly two directions:

Reconcile with neoclassical economy

- Mainly focused on the individual motivations of the participants. What is usually referred as “microfundaments”.
- Trying to explain motivations in terms of of rational self-interested individuals to match the dominant economic accounts.
- Proposed that individuals

Uncritically celebratory and a bit naive

- Very influenced by practitioners account of the phenomenon.
- Supported the ethical stand that valued more cooperation and reciprocity than competition and self-interest.
- Assumed that FOSS communities are composed by loosely organized individuals with a very flat or nonexistent hierarchy among them.

The Hierarchical Structure of FOSS projects

Free and Open Source Software (FOSS) communities have attracted a lot of attention from researchers. Academic efforts took mainly two directions:

Reconcile with neoclassical economy

- Mainly focused on the individual motivations of the participants. What is usually referred as "microfundaments".
- Trying to explain motivations in terms of of rational self-interested individuals to match the dominant economic accounts.
- Proposed that individuals

Uncritically celebratory and a bit naive

- Very influenced by practitioners account of the phenomenon.
- Supported the ethical stand that valued more cooperation and reciprocity than competition and self-interest.
- Assumed that FOSS communities are composed by loosely organized individuals with a very flat or nonexistent hierarchy among them.

Well established empirical fact about FOSS projects

- Only few of the participants account for the lion's share of the work done.
- The deep contribution inequalities may mean that these projects follow the "iron law of the oligarchy" (Shaw and Hill, 2014).
- I propose that one of the central characteristics of FOSS projects is their high ratio of turnover in key hierarchical positions.

Empirical Analysis of Individual Contributions

Well established empirical fact about FOSS projects

Only few of the participants account for the lion's share of the work done.

Longitudinal Analysis of the Social Structure of FOSS projects

The social structure of a community are the patterns of relations established among individual participants in the production process. But I do not limit the analysis to one point in time; I analyze its evolution.

Empirical Analysis of Individual Contributions

Well established empirical fact about FOSS projects

Only few of the participants account for the lion's share of the work done.

Longitudinal Analysis of the Social Structure of FOSS projects

The social structure of a community are the patterns of relations established among individual participants in the production process. But I do not limit the analysis to one point in time; I analyze its evolution.

Puzzle: FOSS projects as Oligarchies?

Michels (1915) "iron law of oligarchy" states that organizations tend towards oligarchy as they grow, even if democracy and participation are part of their core goals.

FOSS projects as Open Elites

The continuous renewal of the people that does most of the work is a key mechanism to explain how FOSS projects can thrive and success through time.

Empirical Analysis of Individual Contributions

Well established empirical fact about FOSS projects

Only few of the participants account for the lion's share of the work done.

Longitudinal Analysis of the Social Structure of FOSS projects

The social structure of a community are the patterns of relations established among individual participants in the production process. But I do not limit the analysis to one point in time; I analyze its evolution.

Puzzle: FOSS projects as Oligarchies?

Michels (1915) "iron law of oligarchy" states that organizations tend towards oligarchy as they grow, even if democracy and participation are part of their core goals.

FOSS projects as Open Elites

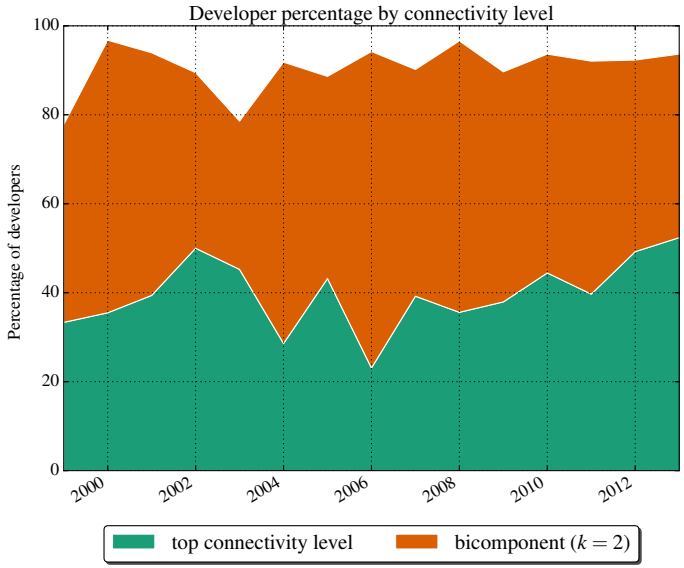
The continuous renewal of the people that does most of the work is a key mechanism to explain how FOSS projects can thrive and success through time.

Results of the empirical analysis:

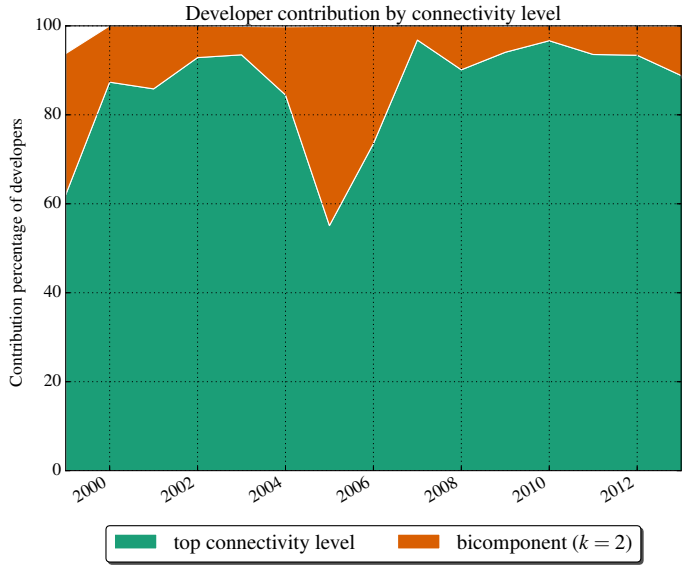
- My analysis shows that the ratio of renewal of individuals at these structural positions is quite fast, which characterizes FOSS communities as dynamic hierarchies and open elites.
- I found that the position of an individual in the connectivity structure of the cooperation network also impacts significantly in the median active life of a developer in the project.

Individual contributions by connectivity level

- The developers that contribute the most are in the higher levels of the connectivity structure of the project's cooperation networks.
- The hierarchical connectivity structure shapes the volume of contribution of individual developers.



(a) Python: developer % by connectivity level



(b) Python: contributions by connectivity level

Figure: Evolution of the percentage of developers by connectivity level (left) and evolution of the percentage of contributions by developers by connectivity levels (right) in the Python project. The green surface represents the developers in the top connectivity level, that is developers that are part of a k-component with maximum k. The orange surface represents developers in bicomponents, that is k-components with $k = 2$.

Analyzing Source Code Contributions and beyond

To further the analysis I used more sophisticated statistical modeling strategies to assess the impact of the connectivity structure in the individual contributions to the project.

- **Debian package uploads:** Modeled with a negative binomial regression. Over-dispersed and discrete dependent variable: # of uploads. The *k*-component number is the second most important independent variable only after Degree Centrality.
- **Added source code lines to Python:** Modeled with a panel regression with individual and year fixed effect. Dependent variable treated as a continuous variable. The *k*-component number is also the second most important independent variable only after Degree Centrality.

Analyzing Source Code Contributions and beyond

To further the analysis I used more sophisticated statistical modeling strategies to assess the impact of the connectivity structure in the individual contributions to the project.

- **Debian package uploads:** Modeled with a negative binomial regression. Over-dispersed and discrete dependent variable: # of uploads. The *k*-component number is the second most important independent variable only after Degree Centrality.
- **Added source code lines to Python:** Modeled with a panel regression with individual and year fixed effect. Dependent variable treated as a continuous variable. The *k*-component number is also the second most important independent variable only after Degree Centrality.

Potential endogenity problems of the previous models

There is a relation between the dependent variable and the network metrics (independent variables) because the cooperation networks are build precisely based on contributions to the projects.

- **Number of accepted *Python Enhancement Proposals* (PEP):** Modeled with a zero inflated negative binomial. PEPs are design documents that define the language; are thus contributions not directly related to the cooperation networks. Being part of the top connectivity level is also the second most important independent variable, this time only after developer's tenure (in years).

Cooperation Networks' Connectivity Hierarchies as Open Elites

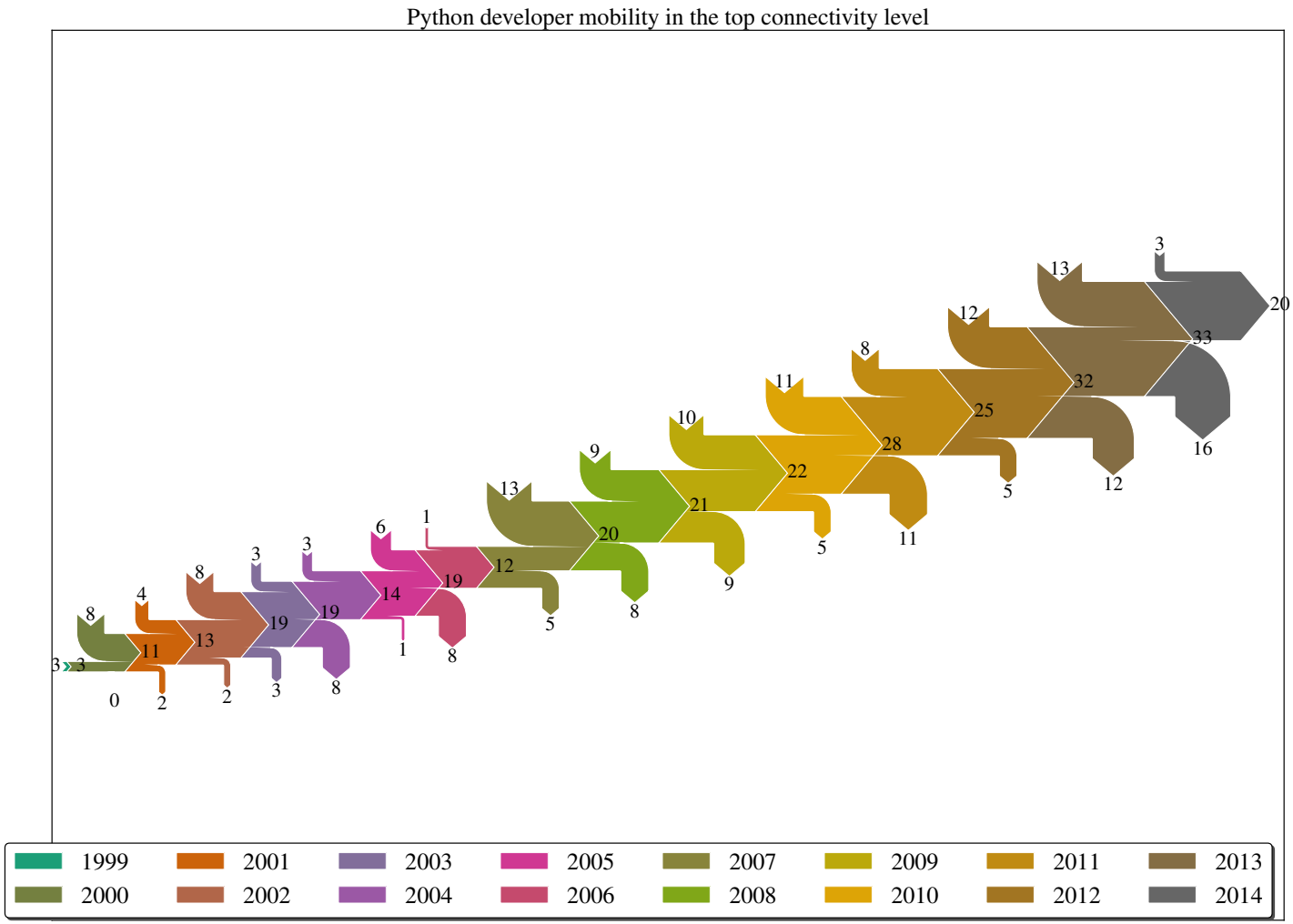


Figure: Sankey diagram of Python developer mobility at the top connectivity level.

Table: Developer mobility in the top connectivity level for the Python project.

Years	Top Developers	New Developers	Developers Out	Developers back
1999	3 (33.3%)	3 (100.0%)	0 (0.0%)	0 (0.0%)
2000	11 (35.5%)	8 (72.7%)	1 (9.1%)	0 (0.0%)
2001	13 (39.4%)	4 (30.8%)	1 (7.7%)	0 (0.0%)
2002	19 (50.0%)	8 (42.1%)	1 (5.3%)	0 (0.0%)
2003	19 (45.2%)	3 (15.8%)	5 (26.3%)	0 (0.0%)
2004	14 (28.6%)	3 (21.4%)	1 (7.1%)	0 (0.0%)
2005	19 (43.2%)	1 (5.3%)	4 (21.1%)	5 (26.3%)
2006	12 (23.1%)	0 (0.0%)	4 (33.3%)	1 (8.3%)
2007	20 (39.2%)	10 (50.0%)	7 (35.0%)	3 (15.0%)
2008	21 (35.6%)	6 (28.6%)	6 (28.6%)	3 (14.3%)
2009	22 (37.9%)	9 (40.9%)	3 (13.6%)	1 (4.5%)
2010	28 (44.4%)	9 (32.1%)	7 (25.0%)	2 (7.1%)
2011	25 (39.7%)	7 (28.0%)	2 (8.0%)	1 (4.0%)
2012	32 (49.2%)	8 (25.0%)	12 (37.5%)	4 (12.5%)
2013	33 (52.4%)	8 (24.2%)	16 (48.5%)	5 (15.2%)
2014	20 (32.3%)	3 (15.0%)	0 (0.0%)	0 (0.0%)

Modeling robustness of Cooperation Networks

Usual measures of robustness

- Failures: remove nodes at random and see how this affects the size of the giant component.
- Attacks: remove nodes incrementally starting for the ones with higher degree.

Modeling robustness of Cooperation Networks

Usual measures of robustness

- Failures: remove nodes at random and see how this affects the size of the giant component.
- Attacks: remove nodes incrementally starting for the ones with higher degree.

My approach: Survival analysis

- I model active life of a developer as the period that this developer is contributing to the project.
- I consider a developer “dead” when she no longer contributes to the project.

Modeling robustness of Cooperation Networks

Usual measures of robustness

- Failures: remove nodes at random and see how this affects the size of the giant component.
- Attacks: remove nodes incrementally starting for the ones with higher degree.

My approach: Survival analysis

- I model active life of a developer as the period that this developer is contributing to the project.
- I consider a developer “dead” when she no longer contributes to the project.

Cox proportional hazards model

- Time-dependent variables and right-censoring.
- Being part of the top connectivity level decreases the yearly hazard of leaving the project by 77%.
- Also an increment of one connectivity level decreases the yearly hazard of leaving the project by 26%.

Modeling robustness of Cooperation Networks

Usual measures of robustness

- Failures: remove nodes at random and see how this affects the size of the giant component.
- Attacks: remove nodes incrementally starting for the ones with higher degree.

My approach: Survival analysis

- I model active life of a developer as the period that this developer is contributing to the project.
- I consider a developer "dead" when she no longer contributes to the project.

Cox proportional hazards model

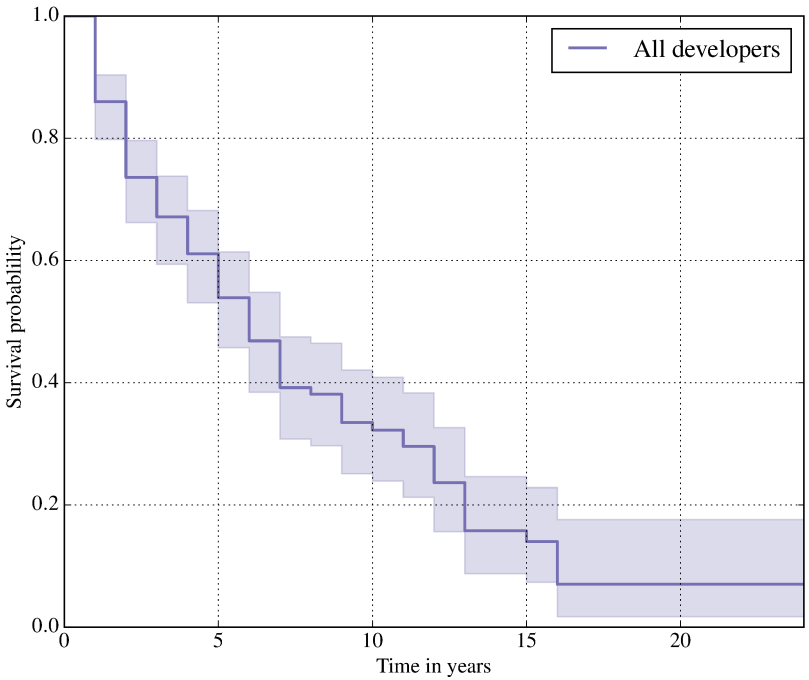
- Time-dependent variables and right-censoring.
- Being part of the top connectivity level decreases the yearly hazard of leaving the project by 77%.
- Also an increment of one connectivity level decreases the yearly hazard of leaving the project by 26%.

Estimating the survival function with the Kaplan-Meier estimator

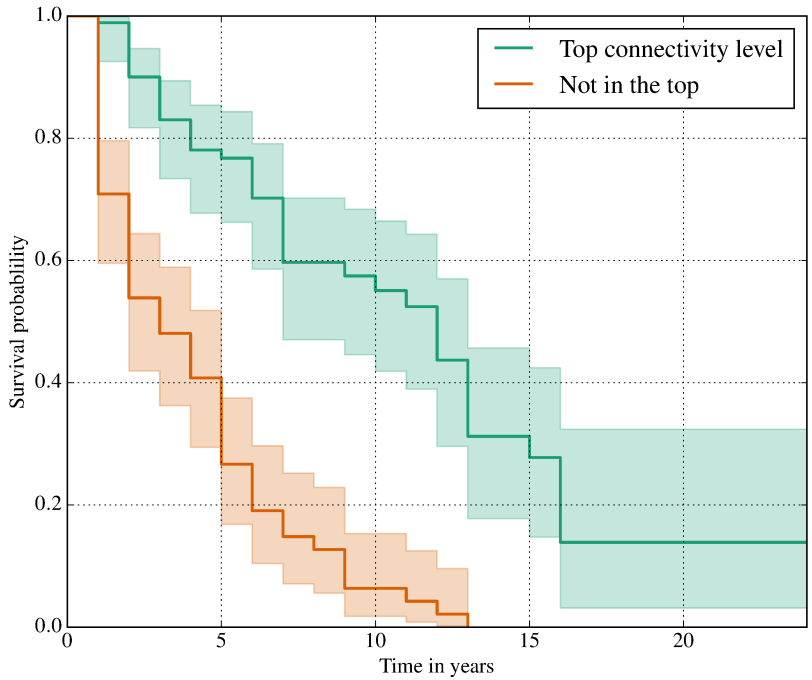
$$\hat{S}(t) = \prod_{t_i < t} \frac{n_i - d_i}{n_i}$$

where d_i are the number of "death events" at time t and n_i is the number of subjects at risk of death at time t.

Modeling robustness as median active live of individuals in the project



(a) Survival Function for all developers



(b) Survival Function for developers in the top connectivity level

Figure: Estimation of the survival function using the Kaplan-Meier estimate. The median survival time of a developer in the community, defined as the point in time where on average half of the population has abandoned the community, is 6 years if I consider all developers (left). But if I consider separately the developers in the top level of the connectivity hierarchy (right), their median survival time is 12 years; but only 3 years for the developers that are not on the top of the connectivity hierarchy.

References

Adler, P. and C. Heckscher (2006). *The Firm as a Collaborative Community: Reconstructing Trust in the Knowledge Economy*. Oxford University Press, Oxford, UK.

Adler, P. S. (2015). Community and innovation: from tönnies to marx. *Organization Studies 36*(4), 445–471.

Axelrod, R. and W. Hamilton (1981). The evolution of cooperation. *Science 211*(4489), 1390–1396.

Doreian, P. and T. Fararo (1998). *The problem of solidarity: theories and models*. Routledge.

Eguíluz, V., M. Zimmermann, C. Cela-Conde, and M. Miguel (2005). Cooperation and the Emergence of Role Differentiation in the Dynamics of Social Networks 1. *American journal of sociology 110*(4), 977–1008.

Marx, K. (1990). *Capital: a critique of political economy. Volume 1*. Penguin Books in association with New Left Review.

Michels, R. (1915). *Political parties: A sociological study of the oligarchical tendencies of modern democracy*. Hearst's International Library Company.

Moody, J. and D. White (2003). Social cohesion and embeddedness: A hierarchical conception of social groups. *American Sociological Review 68*(1), 103–28.

Shaw, A. and B. M. Hill (2014). Laboratories of oligarchy? how the iron law extends to peer production. *Journal of Communication 64*(2), 215–238.

Watts, D. (1999). *Small Worlds: The Dynamics of Networks Between Order and Randomness*. Princeton University Press.

White, D. and F. Harary (2001). The cohesiveness of blocks in social networks: Node connectivity and conditional density. *Sociological Methodology*, 305–359.