Longitudinal Analysis of Collaboration Graphs of Forked Open Source Software Development Projects

Amirhosein (Emerson) Azarbakht
School of Electrical Engineering and Computer Science
Oregon State University
azarbaka@oregonstate.edu

Work Under Supervision of Prof. Carlos Jensen
Ph.D. Committee Members:
Prof. Alex Groce
Prof. Chris Scaffidi
Prof. Drew Gerkey
Prof. A. Morrie Craig

What is this about?

Big Picture

- Developing complex software systems is complex
- Software Developers (SD) interact
  - They can have same/different goals, communication styles, values
- Interactions can be healthy or troubled
- Troubled interactions cause troubled communities -> failure
  - Some of these failures manifest as forks
  - Failure affects many people; developers and users

Can we save troubled projects?

What is this about?

First attempt: problem context

- Interactions can be modeled as graphs
- Graphs change through time
- Can these changing graphs reflect the climate happening during the run-up to fork

What is forking?

- “when a part of a development community (or a third party not related to the project) starts a completely independent line of development based on the source code basis of the project.”

- It can be Undesirable:
  - Dilution of workforce
  - Flaming & unhealthy dynamics
  - Redundant work

} people suffer

Undesirable forks = ?

- **Undesirable forks:** Perceived as bad forks, that affects the developer community and users negatively, and would've been avoided if possible.

- **Socially-related forks**: Could have left traces in the developers' interactions data.

Why Projects Fork?

Socially-related { H.F. / U.F. }

Reason for forking	Example forks
Technical (Addition of functionality)	Amarok & Clementine Player
More community-driven development	Asterisk & Callweaver
Differences among developer team	Kamailio & OpenSIPS
Discontinuation of the original project	Apache web server
Commercial strategy forks	LibreOffice & OpenOffice.org
Experimental	GCC & EGCS
Legal issues	X.Org & XFree

Related Work

What we can know about FOSS	What we do *not know*
Identifying knowledge brokers	Static point of view; People are important, but how do these roles change, how they move around around
Visual exploration of collaboration	Interesting, but not quantitatively conclusive. Many hidden processes that remain unexplored
Post-forking porting of new features	
Social structures and dynamics	How these structures change through time
Identifying “key” events	Compared only two snapshots; not longitudinal
Tension between diversity & homogeneity	
Age of developers and survival rate	How can this be used to understand forking
Communication patterns	Change through time?
Sustainability	How to maintain a sustainable project without forking

Gap: **Run-up to forks** seldom studied

Research Goals

Quantitative

- Any traces left?
- Do different forks leave different traces?
- Key indicators = ?

Qualitative

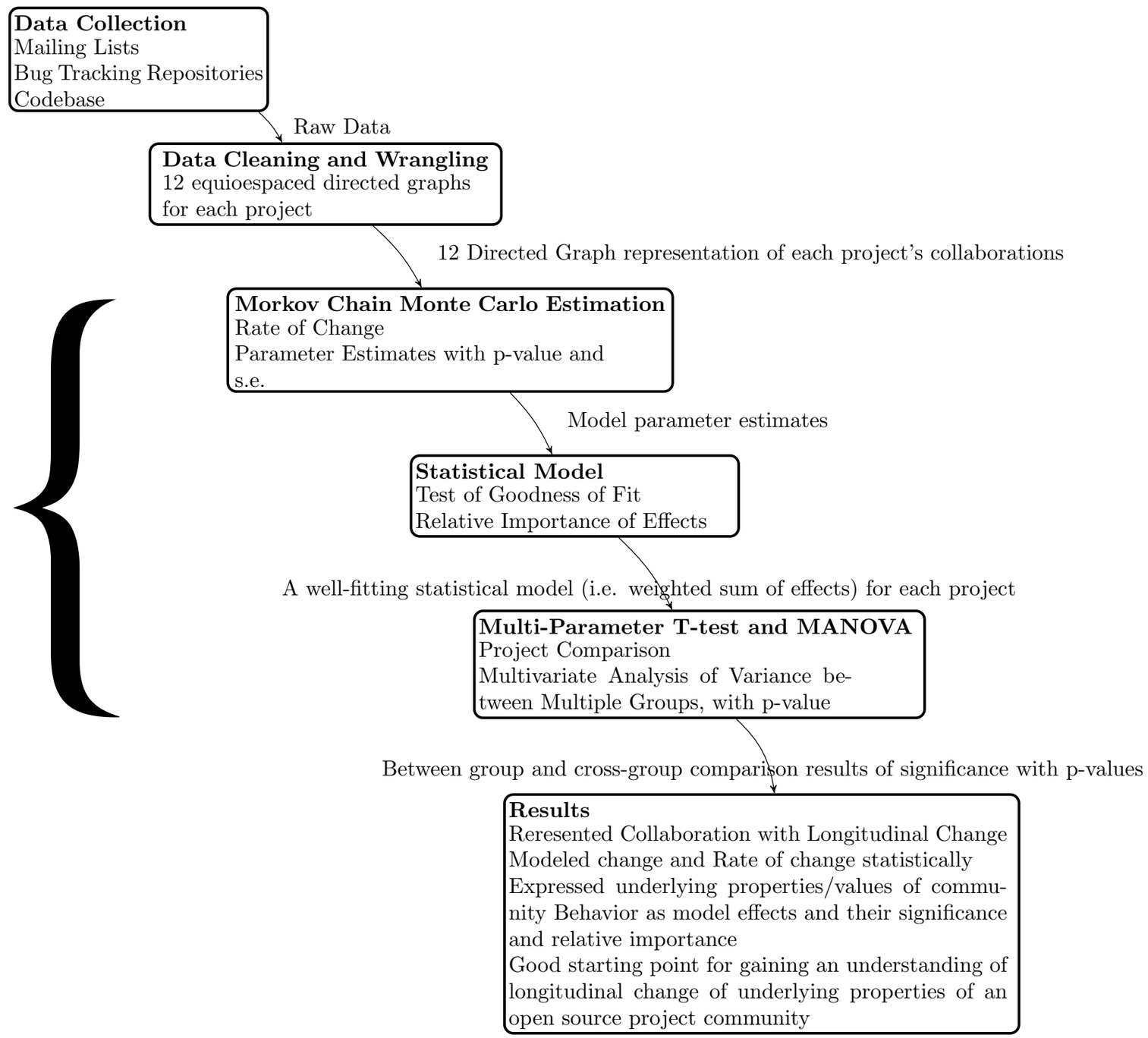
- Validate
  - Interviews
  - Surveys

- Why these are important:
  Because if we know these, we could identify unhealthy dynamics and fix them before it's irreversible.

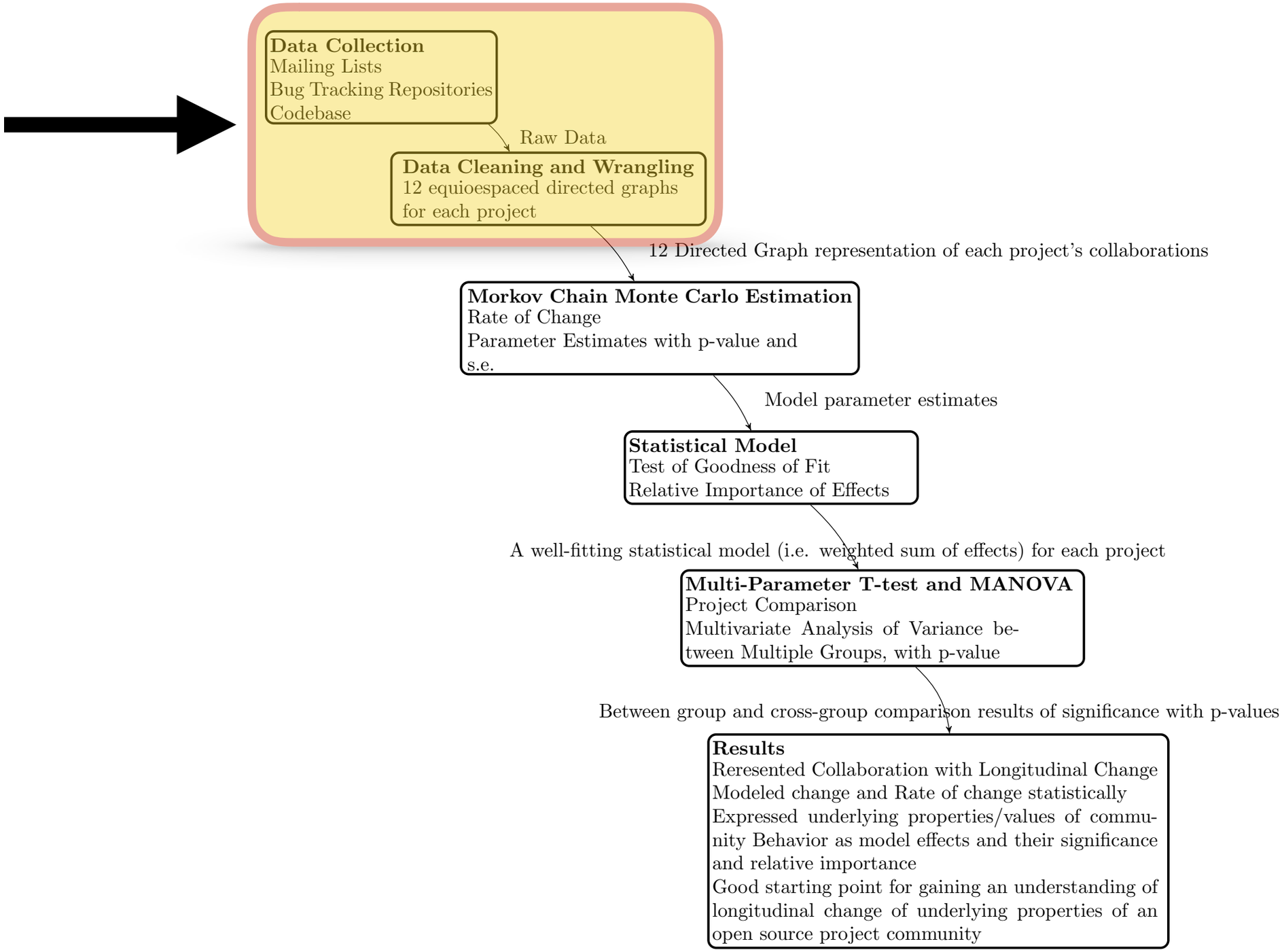
Methodology

Data Collection

Statistical Modeling



Methodology: Data Collection



Data Collection Categories

Table 6: The three types of projects for which data is collected in this study

Type of forking	**Abbreviation**
Undesirable & socially-related forking	U.F.
Other socially-related forking	H.F.
No forking at all (as the control group)	No.F.

Data Collection Projects

Table 7: List of projects in U.F. H.F., and No.F. categories selected based on the criteria described in section 5.1 for our study

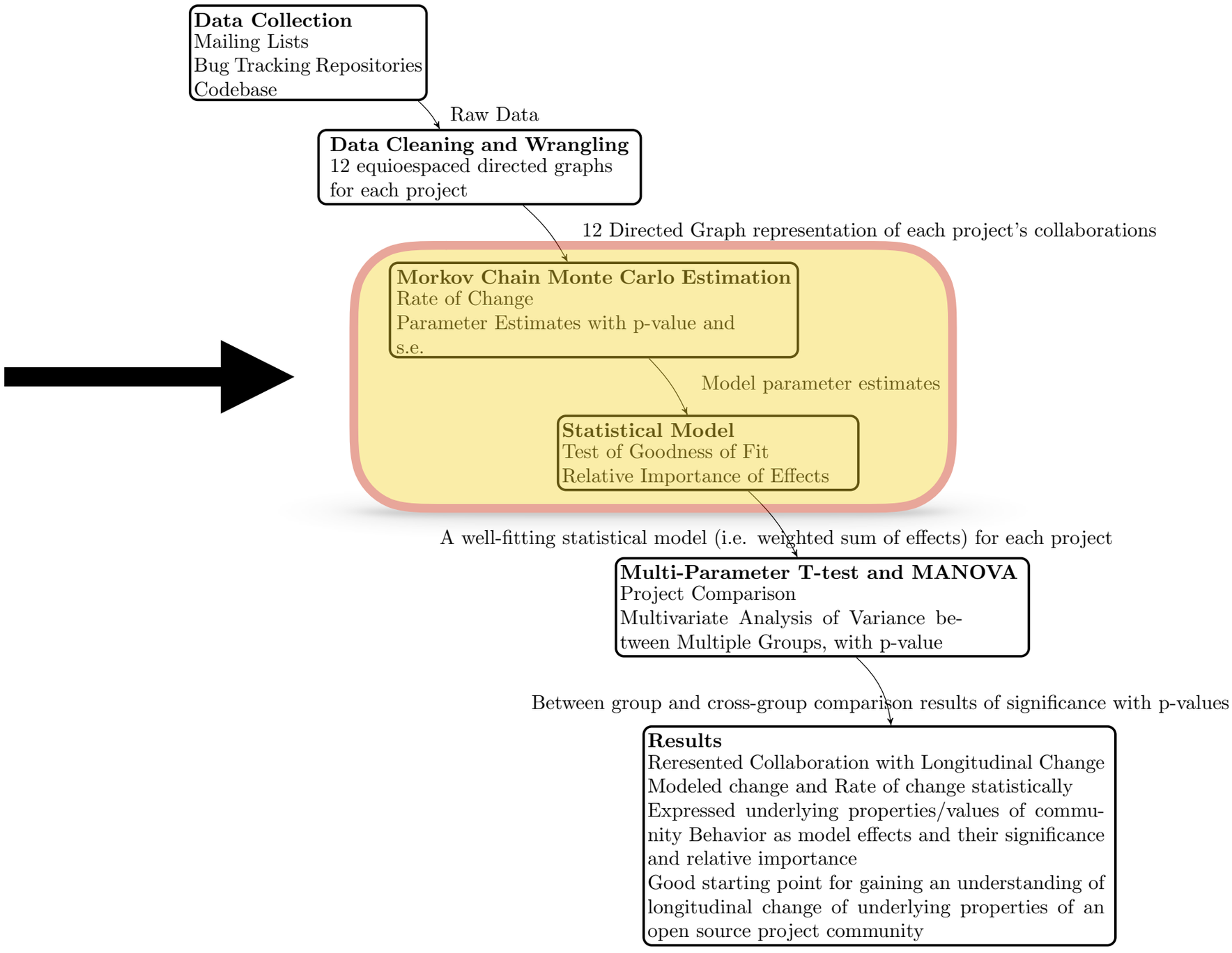
Projects	Reason for forking	Year forked	Type
Kamailio & OpenSIPS	Differences among developer team	2008	U.F.
ffmpeg & libav	Differences among developer team	2011	U.F.
Asterisk & Callweaver	More community-driven development	2007	U.F.
rdesktop & FreeRDP	More community-driven development	2010	U.F.
freeglut & OpenGLUT	More community-driven development	2004	U.F.
Amarok & Clementine Player	Technical (Addition of functionality)	2010	H.F.
Apache CouchDB & BigCouch	Technical (Addition of functionality)	2010	H.F.
Pidgin & Carrier	Technical (Addition of functionality)	2008	H.F.
MPlayer & MPlayerXP	Technical (Addition of functionality)	2005	H.F.
Ceph	Not forked	-	No.F.
Python	Not forked	-	No.F.
OpenStack Neutron	Not forked	-	No.F.
GlusterFS	Not forked	-	No.F.

U.F.
- U.F. personal
- U.F. community-driven

H.F.

No.F.

Methodology: Statistical Modeling



Statistical Model **Covariates**

What can make an impact to result in forking?

- **Reciprocity**
- Closure (**balance**)
    - Transitive triplets
- Three-cycles
- in-in degree **assortativity**
- in-out degree assortativity
- Developer's l**evel of contribution**/activity (e.g. code commits per month, or mailing list posts per month)
- Developer's **level of privilege/prestige** (e.g. admin privilege-holder vs. core contributor vs. marginal developer/user)
- Developer's **level of negative experience** (e.g. number of rejected pull requests: especially in forks for more community- driven development)
- Developer's **age/seniority** (either birth age, (young 20-year-old developer vs. older 50-year-old developer) or their seniority as a development community member (i.e. how long they have been in the community)
- Developer's **propensity for short-responses** vs. long-responses. (under-communicator vs. over-communicator)
- Developer's **communication sentiment** (generally bitter in communication known as "a jerk", or has a positive communication prose)
- Developer's **engagement style** (i.e. dive-bomber- style contributor who jumps in, overcommits and leave, vs. a steady and increasingly engaged contributor who starts off slow and grows his/her commitment gradually)

What do we expect to have pre-fork impact in each category?

Table 8: Expectations for significance of the statistical model covariates based on intuition. Note that * indicates significantly-related; +/- indicates positively/negatively related.

Model parameter	**U.F. Pers. Dif.**	**U.F. More Comm.**	**H.F. Tech. Dif.**
Reciprocity	*		*
Transitive triplets	*		*
Number of developers at distance two		*-	
Three-cycles	*		*
out-out degree assortativity		*+	
Covariate-V1 [Developer's contribution level]-related popularity	*		*
Covariate-V2 [Developer's privilege level]-related dissimilarity		*-	*
Covariate-V3 [Developer's negative experience level]-related popularity		*	
Covariate-V3 [Developer's negative experience level]-related dissimilarity	*		
Covariate-V4 [Developer's age-seniority level]-related popularity	*		
Covariate-V4 [Developer's age-seniority level]-related dissimilarity	*	*	
Covariate-V5 and V6[Developer's under/over communicator & sentiment history]-related activity and dissimilarity	*		
Covariate-V7 [Developer's graduality level]-related activity	*	*-	
Covariate-V7 [Developer's graduality level]-related dissimilarity	*	*-	

Statistical Model

- Longitudinal
- Developer-oriented:
  - Developers choose whom to talk to
    - Likelihood of forming ties, maintaining ties, …
- Stochastic
  - Developers behave to optimize an objective function; included our covariates
  - Assume observed graphs as outcomes of a continuous-time Markov process
- Estimated

Statistical Model
Mathematical notation

- Data: M repeated observations on a graph with g developers

- Representation: Directed graphs with adjacency matrices $X(t_m) = (X_{ij}(t_m))$ for $m = 1, \ldots, M$
  - where i and j range from $1, ..., g$

- X_{ij} shows whether at time t, there exists a tie from i to j (value 1) or not (value 0).

Statistical Model

Longitudinal change = a series of mini-steps

To model graph evolution from $X(t_1)$ to $X(t_2)$,
and so on, we treat the dynamics as the results of
a series of small atomic changes (mini-steps)

Statistical Model
Mini-steps

- Developer whose tie is changing, has control over the change

- The graph changes one tie at a time (**1 mini-step**)

- The moment of change & the kind of change **depends on observed covariates** and the **graph structure**

- The **moment of change** is stochastically determined by the *rate function*

- The particular **kind of change** is determined by the objective function

Statistical Model
Rate Function

The **rate function** $\lambda_i(x)$ for developer *i* is the rate at which developer *i*'s outgoing connection changes occur.

$$\lambda_i(x) = \lim_{dt \to 0} \frac{1}{dt} \mathrm{P}\big(X_{ij}(t+dt) \neq X_{ij}(t) \quad for \quad some \quad j \in \{i, ..., g\} | X(t) = x)\big)$$

It models *how frequently* the developers make mini-steps.

Statistical Model
Objective function

The **objective function** $f_i(s)$ for developer i is the value attached to the network configuration x.

$$f_i(\beta, x) = \sum_{k=1}^{L} \beta_k s_{ik}(x)$$

- Functions $s_{ik}(x)$ are the effects we define. Parameters $\beta = (\beta_1, \ldots, \beta_L)$ is to be estimated.

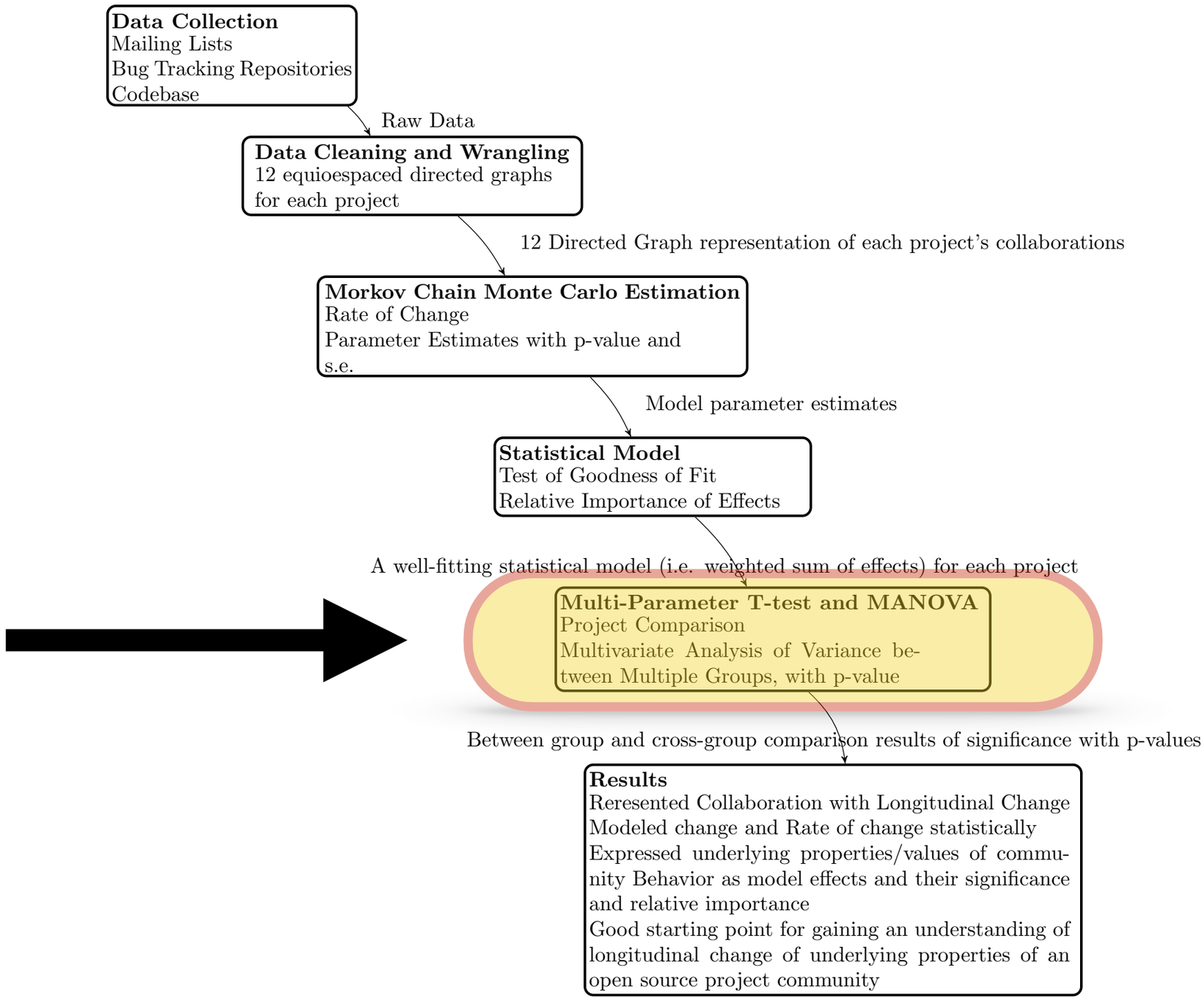
- Idea: Given the opportunity to make a change in his out-going tie variables $(X_{i1},\ldots,X_{ig})$, developer i selects the change that gives the greatest increase in the objective function.

Statistical Model Simulation and Estimation

- Using Method of Moments (MoM)
  - Equate the observed sample statistics to the theoretical population statistics and solve the equation to find the coefficients
- Markov Chain Monte Carlo (MCMC) estimation
  - Simulate the network evolution, and estimate the model based on the simulations

We fitted/have a model

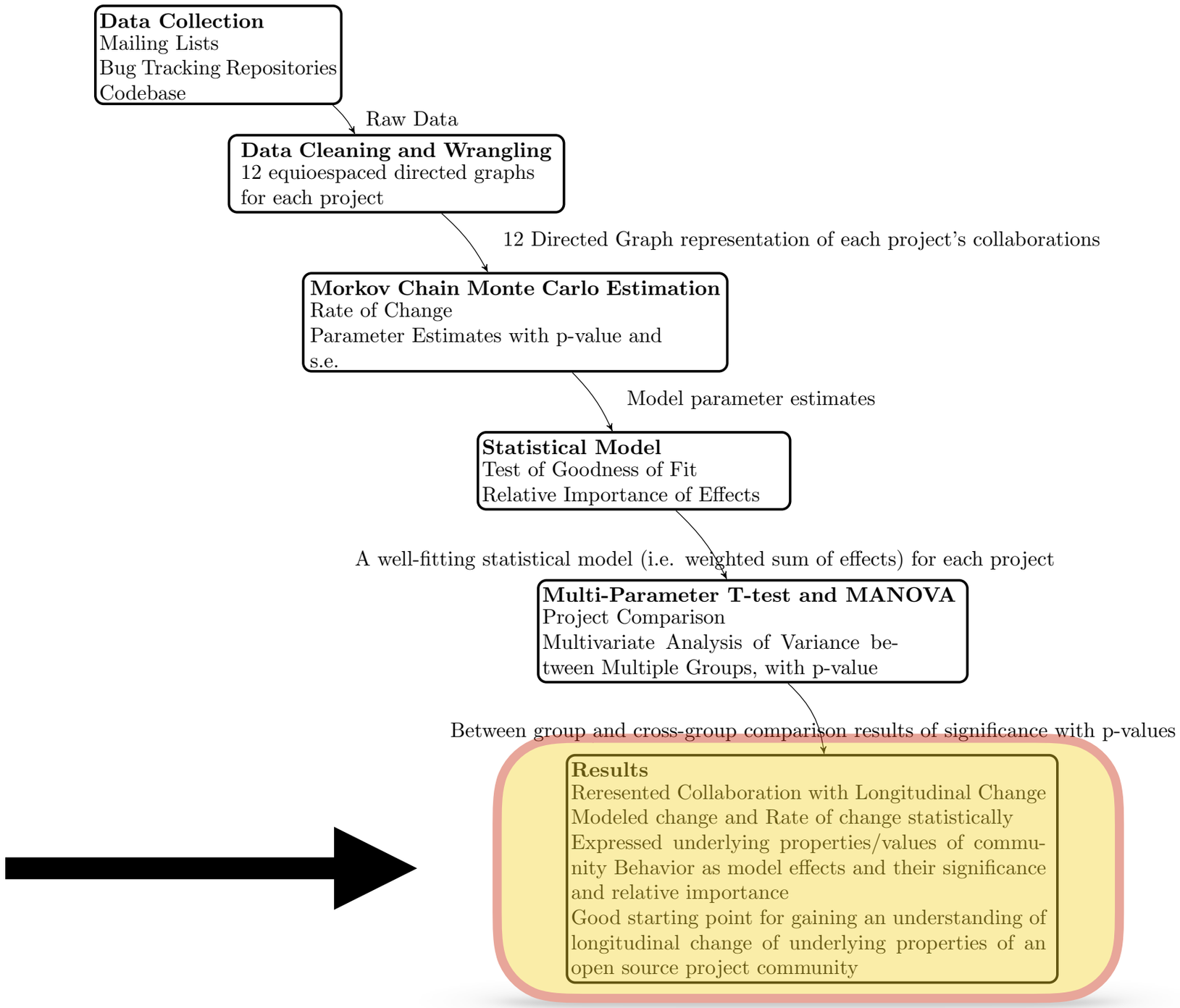
Methodology: Hypothesis Testing



Hypothesis Testing

- Single parameter test
  - To determine the significance of covariates in the model; longitudinal changes can be [partially] explained by that covariate
- Multi-parameter differences between groups
  - To compare two project models
- Multivariate ANOVA
  - To compare a group of projects and a project

Methodology: Results



Qualitative Validation

- Interviews/surveys
  - Semi-structured interviews with contributors
  - Not ~~open-coding~~. Using pre-existing categories in coding that maps to the covariates in the model
  - Multiple coders
- Sentiment analysis
  - Contents of messages sent/received by top contributors
  - A time series of sentiments, to analyze

Conclusions

- Represented Collaboration with Longitudinal Change
- Modeled change and Rate of change statistically
- Expressed underlying properties/values of community behavior as model effects and their significance and relative importance
- Validated using qualitative methods
- Good starting point for gaining an understanding of longitudinal change of underlying properties of an open source project community

What has been done, What remains to be done

- Data collection for mailing list archives is completed.
- Issue tracking system & source code interactions data is in the progress.
- Once all data is collected an cleaned, will do the statistical modeling.
- Next, we will do the qualitative validation.

Thank you!

Questions? Suggestions? Help?