Technology Adoption in a Hierarchical Network

Xintong Han[1] Lei Xu[2]

[1]Concordia University

[2]Toulouse School of Economics

Motivation

- Modern economy
  - Disaggregation, specialization
  - Hierarchical network (or vertical, input-output network)
  - Propogation of decisions and shocks
- Technology adoption
  - Network effects/externalities: $u = f(n)$
  - e.g. fax machine
  - Network structure matters

Overview

- Research Question: How technology adoption is affected by the network structure? What policies can be implemented to promote technology adoption?
- Setting: the Python programming language
- Model
  - Dynamic model of technology adoption
  - Incorporate the network structure
  - Estimation under various assumptions
- Findings
  - Package developers benefit from more user downloads
  - Decisions to adopt Python 3 propogate through the links between packages: lower adoption cost for others
- Counterfactuals
  - Targeted promotion

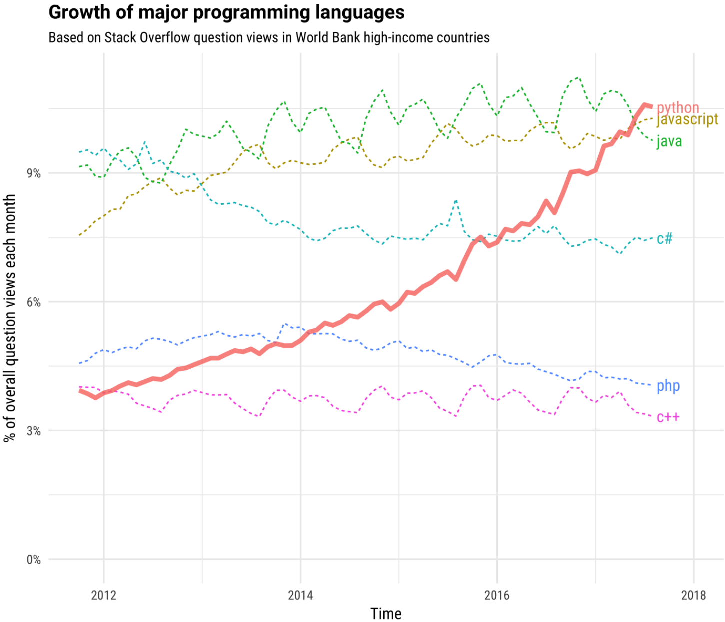
Literature

- Technology Adoption
  - Katz and Shapiro (1985) and many
  - Gowrisankaran and Stavins (2004): ACH financial transaction
  - Ryan and Tucker (2012): videocalling technology
- Social Networks: Matthew Jackson (2010)
- Hierchical or Input-Output Networks
  - Acemoglu et al. (2012):
  - Brancaccio, Kalouptsidi, Papageorgiou (2018): global trade network

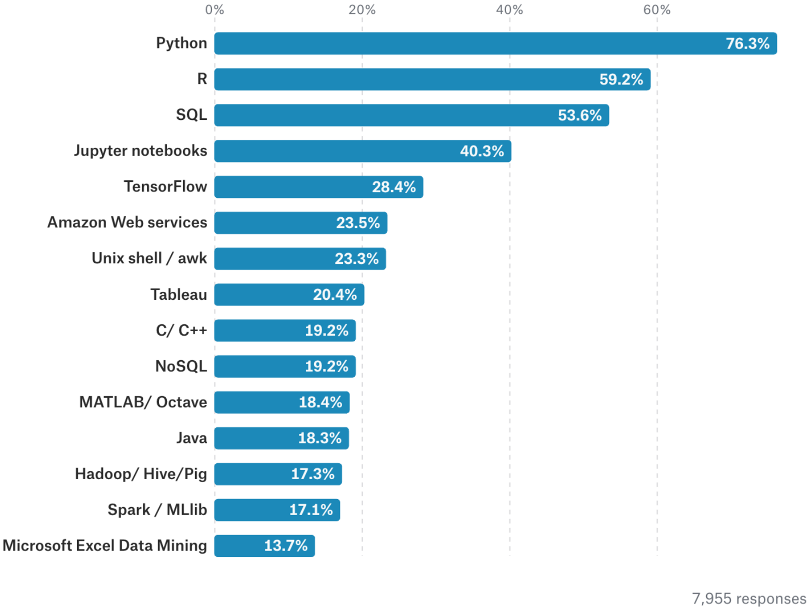
What is Python?

- A general-purpose programming language
  - like C or Java
  - vs. domain-specific languages: Stata, Matlab
  - Third-party packages: >140,000 available
  - Almost all are open source software (OSS)
- One of the most popular programming languges in the world
  - Good choice as a first programming language

Python Popularity



Python Popularity



Terminology - Packages

- Packages: A collection of tools that enables users to do advanced tasks.
- other names: libraries, modules, (sub)routine

e.g. Matrix Multiplication

$$A = [1, 2, 3], \qquad B = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

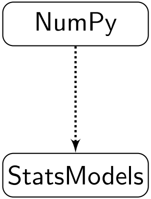
- Pure Python: loop: $1 \times 1 + 2 \times 1 + 3 \times 0 = 3$
- with package NumPy: $A \times B = 3$

Technology Adoption: Python 2 to Python 3

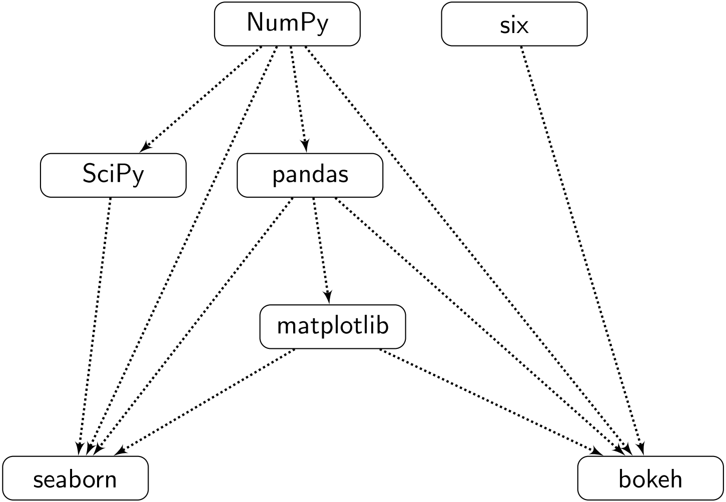
- Python 2: 2000
- Python 3: 2008
- New features that require foundamental changes more details
- Not backward compatible $\longrightarrow$ adoption/switching cost
- Painful & conscious decision
- Python 3 adoption decision for each package: add support to Python 3

Hierarchical Network - Dependencies

- Division of Labor: packages usually have specialties
- e.g. NumPy: linear algebra, random number generators, etc
    - $\hat{\beta} = (X'X)^{-1}X'y$
- StatsModels: OLS, GLM, MLE, GMM, etc
    - it requires matrix inversion & multiplications from NumPy
    - i.e. NumPy is StatsModels' dependency
    - i.e. in order to use StatsModels, one has to install NumPy first
    - installation system: automatic checks and installs dependencies



Dependency Network Example



Data - PyPI - Package

name	pandas
license	BSD
summary	Powerful data structures for data analysis
home_page	http://pandas.pydata.org
author	The PyData Development Team
author_email	pydata@googlegroups.com
version	0.10.1
requires_dist	numpy (≥1.9.0)
	pytz (≥2011k)
	python-dateutil
classifiers	Intended Audience :: Science/Research
	Programming Language :: Python :: 2
	Programming Language :: Python :: 3
	Topic :: Scientific/Engineering

Data - PyPI - Package

name	pandas
license	BSD
summary	Powerful data structures for data analysis
home_page	http://pandas.pydata.org
author	The PyData Development Team
author_email	pydata@googlegroups.com
version	0.10.1
requires_dist	numpy ($\geq$1.9.0)
	pytz ($\geq$2011k)
	python-dateutil
classifiers	Intended Audience :: Science/Research
	Programming Language :: Python :: 2
	Programming Language :: Python :: 3
	Topic :: Scientific/Engineering

Data - PyPI - Package

name	pandas
license	BSD
summary	Powerful data structures for data analysis
home_page	http://pandas.pydata.org
author	The PyData Development Team
author_email	pydata@googlegroups.com
version	0.10.1
requires_dist	numpy ($\geq$1.9.0)
	pytz ($\geq$2011k)
	python-dateutil
classifiers	Intended Audience :: Science/Research
	Programming Language :: Python :: 2
	Programming Language :: Python :: 3
	Topic :: Scientific/Engineering

Data - PyPI - Downloads

Table: Downloads Statistics (Before 2015)

File 1 (for Python 2)	
upload_time	2013-01-22T05:42:07
python_version	2.7
downloads	41564
filename	pandas-0.10.1.win-py2.exe
size	2041220

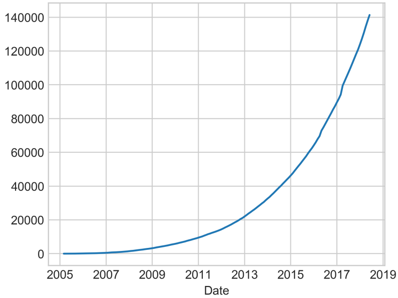
File 2 (for Python 3)	
upload_time	2013-01-22T05:54:10
python_version	3.2
downloads	50892
filename	pandas-0.10.1.win-py3.exe
size	1866691

Data - PyPI - BigQuery

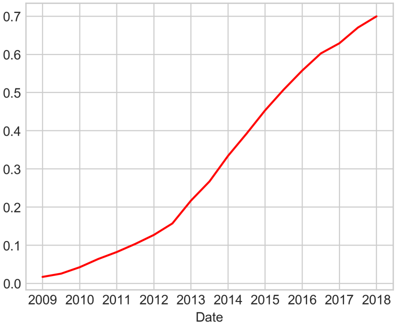
Table: Downloads Statistics (After 2015)

timestamp	2018-09-26 08:15:13.000 UTC
country_code	FR
filename	pandas-0.10.1-cp32-macosx.whl
project	pandas
version	0.10.0
python	3.2
system	Mac OS X

Python Packages with Python 3 Support

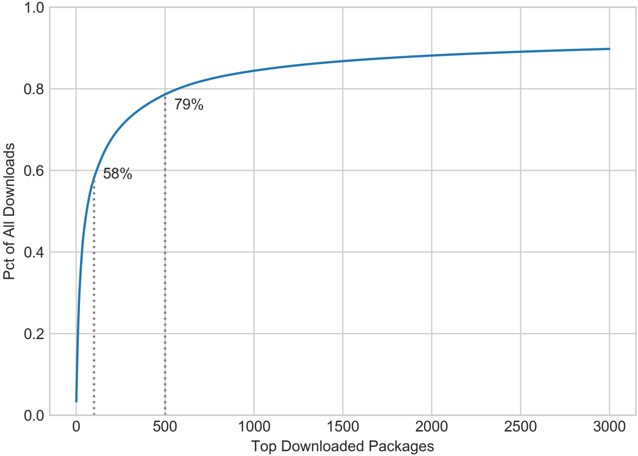


(a) Total Number of Packages



(b) % of Packages with Py3 Support

Total Downloads of Top Packages as Percentage of All Downloads in 2017



Data Selection

- Total: 140k packages
  - Time Duration (Last Release - First Release Date) $\geq$ 1 Year: 12.9%
  - Downloads Per year $\geq$ 2000: 30.8%
  - Total Number of Releases $\geq$ 5: 38.9%
  - Total Releases / Time Duration $\geq$ 1: 92.4%
  - Some Python 2/3 Support Info Available: 59.9%
  - Initial Support is Python 2 Only: 50.7%
- 4005 packages (3%) and 23267 observations

Utility Function - Downloads

- Motivations of OSS Contribution: altruism, ego gratification, career concerns
- Maximize the number of downloads per period
- Assume package developers make Python 3 adoption decisions collectively
- Denote $x_{i,t} = log(Downloads_{i,t})$
- AR1 Process:

$$x_{i,t} = \rho_0 + \rho_{ar} \cdot d_{i,t} \cdot AR_t + \rho_1 \cdot x_{i,t-1} + \epsilon_{i,t}$$

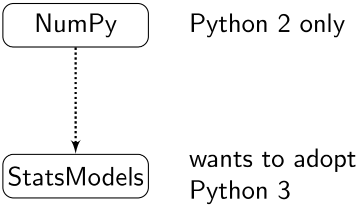
AR_t: Python 3 adoption rates by packages

$d_{i,t} \in \{0, 1\}$: Package i's adoption decision

Python 3 Adoption Decision

- $d_{i,t} \in \{0, 1\}$: package i's decision to add Python 3 support.
  i.e. support both Python 2 and Python 3
- Irreversible decision
- Note: this is package i's adoption decision

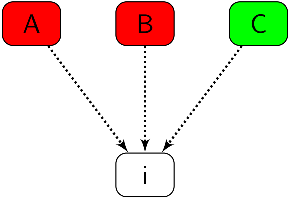
Adoption Cost - Dependencies



Adoption Costs

1. update one's own codebase
2. dependencies without Python 3 support
    - find an alternative depedency that support Python 3
    - change the required code by oneself

Adoption Cost - Dependencies



- $U_i = \{A, B, C\}$
- $d_{A,t} = 0$

  $d_{B,t} = 0$

  $d_{C,t} = 1$

- $\mu_{i,t} = \sum_{j \in \mathcal{U}_i} \mathbf{1}\{d_{j,t} = 0\}$
  number of dependencies without Python 3 support.

  in this example, $\mu_{i,t} = 2$
- $\mu_{i,t} = \sum_{j \in \mathcal{U}_i} \mathbf{1}\{d_{j,t} = 0\} \cdot ln(Size_{j,t})$

- $C_{i,t} = AC_0 + \alpha^{\mu} \cdot 2$

Assumption 1: At time t, a package i observes Python 3 adoption decisions made by its dependencies, namely, $d_{j,t}$ for all $j \in U_{i,t}$.

Reduced-Form Evidence

$$d_{i,t} = \alpha^x \, x_{i,t-1} + AC_0 + \alpha^\mu \, \mu_{i,t} + \alpha^{size} \, Size_{i,t} + \epsilon$$

	(1) OLS	(2) Logit
$x_{i,t-1}$	0.018***	0.167***
	(0.00)	(0.01)
AC_0	-0.011	-3.190***
	(0.01)	(0.13)
$\mu_{i,t}$	-0.008***	-0.230***
	(0.00)	(0.02)
$Size_{i,t}$	-0.005***	-0.041***
	(0.00)	(0.01)
R2	0.02	
Log likelihood		-8193.78
Number of Packages	4005	4005
Number of observations	23267	23267

Model - Flow Utility

$$u_{i,t}(d) = \underline{\alpha^x} x_{i,t}(d) - C_{i,t}(d) + \nu^d_{i,t}$$

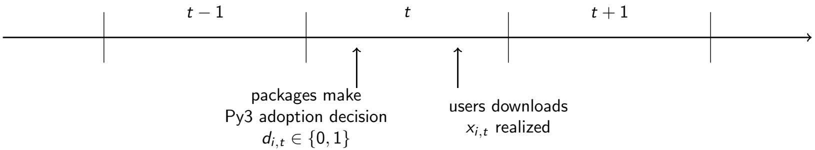
where

$$C_{i,t}(d) = \begin{cases} \underline{AC_0} + \underline{\alpha^\mu} \mu_{i,t} & \text{if package } i \text{ adopts Py3: } d = 1 \\ 0 & \text{otherwise} \end{cases}$$

Assumption 2: $d_{j,t}$ affects $u_{i,t}$ for $j \in U_{i,t}$; not vice versa.

empirical evidence

Model - Timeline



Model

State variables: $S_{i,t} \equiv (x_{i,t-1}, d_{i,t-1}, \nu_{i,t}, \{d_{j,t}, S_{j,t}\}_{j \in U_{i,t}})$

Value function & Bellman equation:

$$
\begin{aligned}
& V(\mathcal{S}_{i,t}, d_{i,t-1} = 0, v_{i,t}; \theta) \\
= & \max_{\{d_{i,t+\tau}\}_{\tau=0}^{\infty}} \mathbf{E}_t \{ \sum_{\tau=0}^{\infty} \beta^{\tau} u_{i,t+\tau}(S_{i,t+\tau}, d_{i,t+\tau}) | \mathcal{S}_{i,t}, d_{i,t}; \theta \} \\
= & \max_{d_{i,t} \in \{0,1\}} u_{i,t}(S_{i,t}, d_{i,t}; \theta) + v_{i,t}^{d_{i,t}} + \beta \mathbf{E}_t V(\mathcal{S}_{i,t+1}, v_{i,t+1} | \mathcal{S}_{i,t}, v_{i,t}, d_{i,t}; \theta)
\end{aligned}
$$

Model

Assuming $v_{i,t}^{d_{i,t}}$ are iid logit errors,

$$EV(\mathcal{S}, d = 0; \theta)$$
$$= \int_{S'} \log\{ \sum_{d' \in \{0,1\}} \exp(u(S', d'; \theta) + \beta EV(\mathcal{S}', d'; \theta))\} d\mathbf{P}_{\mathcal{S}'|\mathcal{S}}$$

$$\begin{aligned}\widehat{p}_{i,t}^{1} \equiv & P(d_{i,t} = 1 | \mathcal{S}_{i,t}, d_{i,t-1} = 0, v_{i,t}; \theta) \\ = & \frac{exp\{v(\mathcal{S}_{i,t}, v_{i,t}, d_{i,t} = 1; \theta)\}}{\sum_{d' \in \{0,1\}} exp\{v(\mathcal{S}_{i,t}, v_{i,t}, d'; \theta)\}}\end{aligned}$$

MLE: $\theta^* = \arg\max_\theta l(\theta) = \prod_{i=1}^{N} \prod_{t=1}^{T} \widehat{p}_{i,t}^{0}{}^{\mathbf{1}\{d_{i,t}=0\}} \widehat{p}_{i,t}^{1}{}^{\mathbf{1}\{d_{i,t}=1\}}$

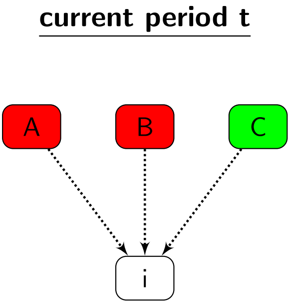
Model - Transition Matrix

$$EV(\mathcal{S}, d = 0; \theta)$$
$$= \int_{S'} \log\{ \sum_{d' \in \{0,1\}} \exp(u(S', d'; \theta) + \beta EV(\mathcal{S}', d'; \theta))\} d\underline{\mathbf{P}_{\mathcal{S}'|\mathcal{S}}}$$

State Variables: $S_{i,t} \equiv (x_{i,t-1}, d_{i,t-1}, \nu_{i,t}, \{d_{j,t}, S_{j,t}\}_{j \in U_{i,t}})$
LOM of two important elements: $x_{i,t}$, $\mu_{i,t}$

- $x_{i,t}$: AR1 process
  - AR_t: perfect foresight
- $\mu_{i,t} \equiv \sum_{j \in U_{i,t}} \mathbb{1}(d_{j,t} = 0) \cdot ln(Size_{j,t})$: the number of Python 3 incompatible dependencies (weighted by size)

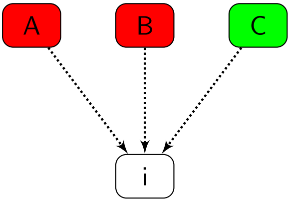
Intuition - Intertemporal Tradeoffs



- $\mu_{i,t} = 2$
- $C_{i,t}(d=1) = AC_0 + \alpha^{\mu} \cdot 2$
- e.g. package i's belief: $\widehat{p}^{1}_{A,t+1} = 0$, $\widehat{p}^{1}_{B,t+1} = 1$

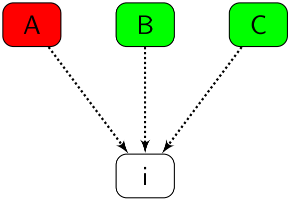
Intuition - Intertemporal Tradeoffs

current period t



- $\mu_{i,t} = 2$
- $C_{i,t}(d=1) = AC_0 + \alpha^{\mu} \cdot 2$
- e.g. package i's belief: $\widehat{p}^1_{A,t+1} = 0$, $\widehat{p}^1_{B,t+1} = 1$

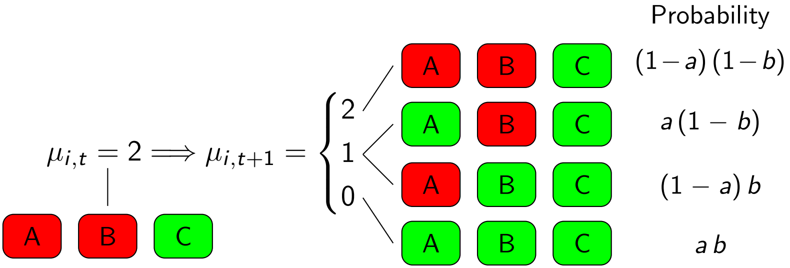
forecast t+1



- $\mu_{i,t+1} = 1$
- $C_{i,t}(d=1) = AC_0 + \alpha^{\mu}$

Model - Transition Matrix - Example

Let $\widehat{p}^1_{A,t+1} = a, \quad \widehat{p}^1_{B,t+1} = b$



Model - Transition Matrix

Assumption 3: Package i holds myopic expectations regarding the future Python 3 adoption probabilities by its dependencies, i.e. $\widehat{p}^1_{j,t+\tau} = \widehat{p}^1_{j,t}$ for all $\tau \in \mathbb{N}$.

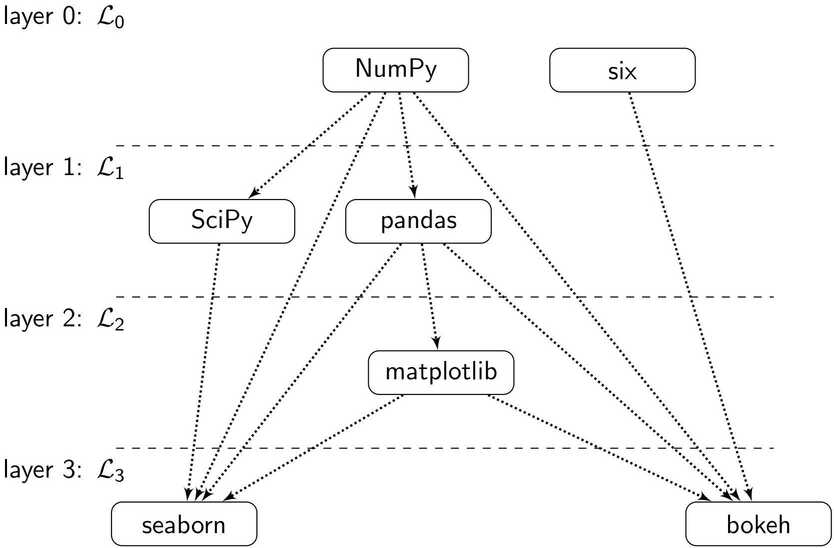
- package i calculates $\widehat{p}^1_{j,t}$ given current state $S_{i,t}$
- huge simplification
- Future version: AR1? like inclusive value function

Identification & Estimation

$$\theta = \{\underbrace{\rho_0, \rho_1, \rho_{ar}}_{\theta_D};\ \underbrace{\alpha^x, AC_0, \alpha^\mu, \alpha^{size}, \beta}_{\theta_S}\}.$$

- Step 0: Model Primitive
  - Time Period: Half Year
  - Initial estimate of θ_D^0 from AR1 process:
    $x_{i,t} = \rho_0 + \rho_{ar} \cdot \underline{d_{i,t}} \cdot AR_t + \rho_1 \cdot x_{i,t-1} + \epsilon_{i,t}$
- Step 1: Model Estimation using MLE:
  $$\theta_S^1 = \arg\max_{\theta_S} l(\theta_S, \theta_D^0) = \prod_{i=1}^{N} \prod_{t=1}^{T} \widehat{p}_{i,t}^{0}{}^{\mathbf{1}\{d_{i,t}=0\}} \widehat{p}_{i,t}^{1}{}^{\mathbf{1}\{d_{i,t}=1\}}$$
- Step 2: Re-estimate the AR1 process using $\widehat{p}_{i,t}^1(\theta_S^1, \theta_D^0)$ as IV for $d_{i,t}$

Estimation - Layered Hierarchical Network



Parameter Estimates of User Downloads

	(1) OLS	(2) IV
(ρ_{ar}) $d_{i,t} \times AR_t$	0.165***	0.074***
	(0.01)	(0.01)
(ρ_1) $x_{i,t-1}$	0.898***	0.902***
	(0.00)	(0.00)
(ρ_0) Constant	1.069***	1.061***
	(0.02)	(0.02)
N	54230	54230
R^2	0.804	0.803

Parameter Estimates of Adoption Model

Nonlinear	β	0.705***
Parameters (θ_S)		(0.074)
	α^x	3.461***
		(1.21)
	AC_0	-2.484***
		(0.099)
	α^μ	-0.145***
		(0.021)
	α^{size}	-0.045**
		(0.015)
Log Likelihood		-8143
Number of Packages		4005
Number of Observations		23267

Parameter Estimates of Adoption Model

Nonlinear	β	0.705***
Parameters (θ_S)		(0.074)
	α^x	3.461***
		(1.21)
	AC_0	-2.484***
		(0.099)
	α^μ	-0.145***
		(0.021)
	α^{size}	-0.045**
		(0.015)
Log Likelihood		-8143
Number of Packages		4005
Number of Observations		23267

- Six-month discount factor $\beta = 0.705$
- Equivalent to monthly $\beta_m = 0.943$
- Comparison to literature

  Lee (2013): 0.934

  De Groote and Verboven (2018): 0.988

Parameter Estimates of Adoption Model

Nonlinear	β	0.705***
Parameters (θ_S)		(0.074)
	α^x	3.461***
		(1.21)
	AC_0	-2.484***
		(0.099)
	α^μ	-0.145***
		(0.021)
	α^{size}	-0.045**
		(0.015)
Log Likelihood		-8143
Number of Packages		4005
Number of Observations		23267

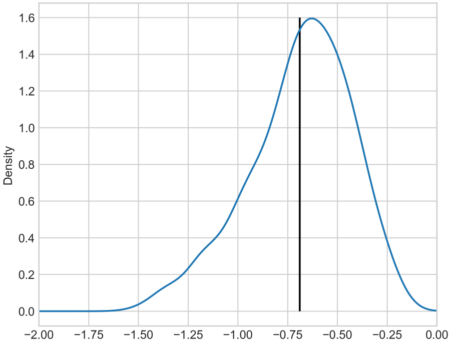
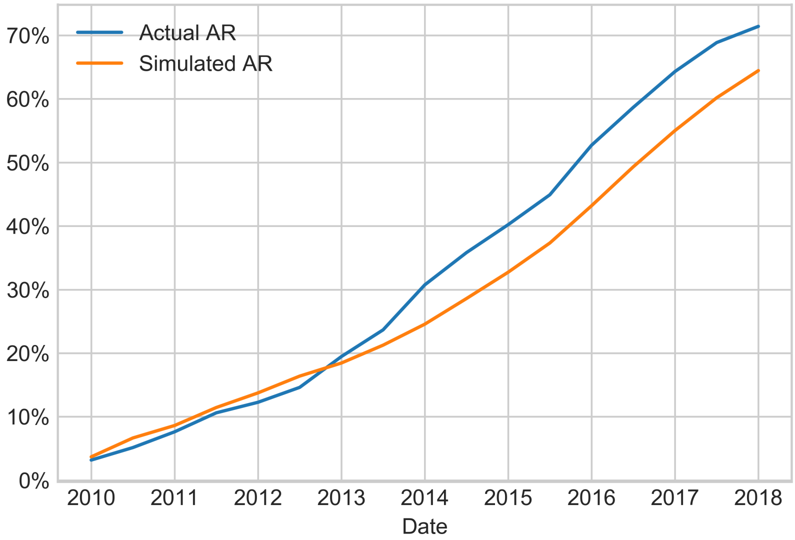


Figure: Adoption cost due to one incompatible dependency (convert α_μ to the same scale as $AC0$)

Actual vs. Simulated Adoption Rates

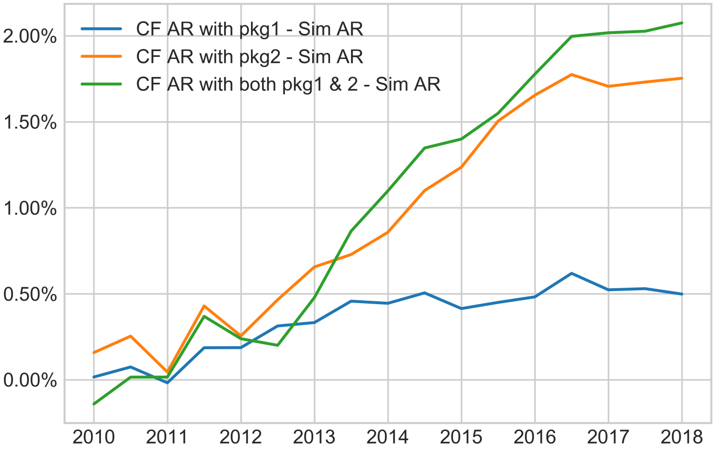


Counterfactual

Sponsorship

- Katz and Shapiro (1985) prediction: a new technology is more likely to succeed and spread faster with sponsorship (i.e. someone willing to promote it)
- Python Software Foundation
- Model costly promotion effort. Q: given limited resources, which packages should be targeted?
- Ideas of good targets:
  - low promotion cost
  - promotion is effective
  - key players in the network

Differences Between Simulated & Counterfactual AR



	Name	Number of Downstream Pkg
package 1	django	71
package 2	requests	23

Conclusion

Conclusion

- Dynamic model of technology adoption with a hierarchical network
- Estimation using a unique dataset of Python packages
- How the adoption inertia can be caused through incompatible "neighbors"

Future Work

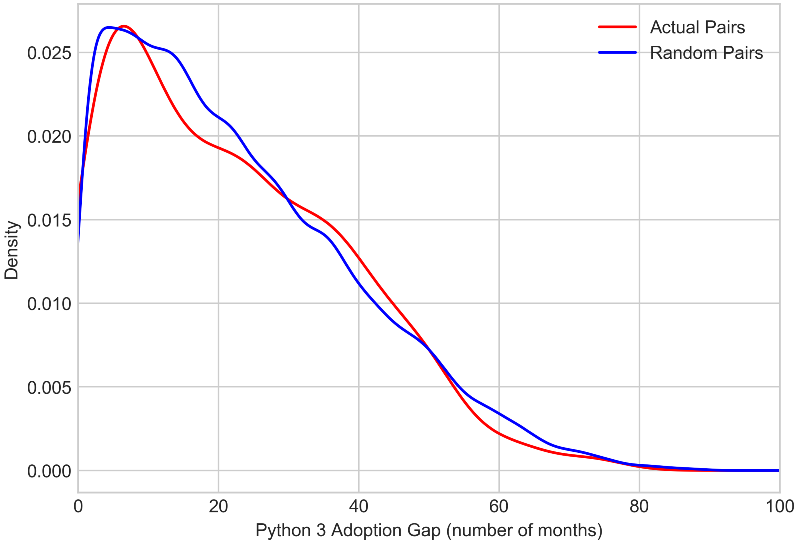
- Data cleaning
- Relax the myopic assumption of future adoption probability
- Adoption rate in the AR1 download process
- More counterfactual policies

Thank you!

$$|Py3AdoptionDate_i - Py3AdoptionDate_j|$$
$$= \alpha + \beta \frac{DL_i}{DL_j} + \gamma \mathbb{1}(j \in U_i) \frac{DL_i}{DL_j}$$

	(1) Adoption Date Gap
DL_i/DL_j	-4.801***
	(0.04)
$1(j \in U_i) \times \frac{DL_i}{DL_j}$	-1.214
	(1.97)
Constant	24.907***
	(0.02)
Number of observations	2603384
R2	0.006

Back to model.



Back to model.

Python 3 New Features

- Default Encoding System
  - Python 2: ASCII
  - e.g. "café" $\longrightarrow$ UnicodeEncodeError
  - Python 3: Unicode
  - e.g. "café" $\longrightarrow$ café
- Division
  - Python 2: $5/2 = 2$
  - Python 3: $5/2 = 2.5$

Back to Overview.