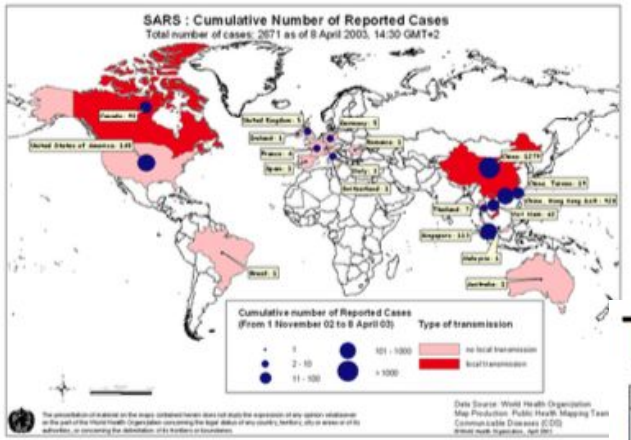
Studying Network Diffusion Dynamics

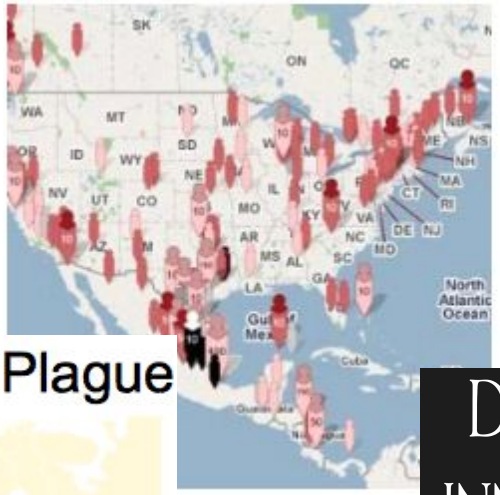
Giulio Rossetti, Letizia Milli, Salvatore Rinzivillo,
Alina Sirbu, Dino Pedreschi and Fosca Giannotti

University of Pisa, Italy
ISTI-CNR, Italy

Diffusive Phenomena are everywhere

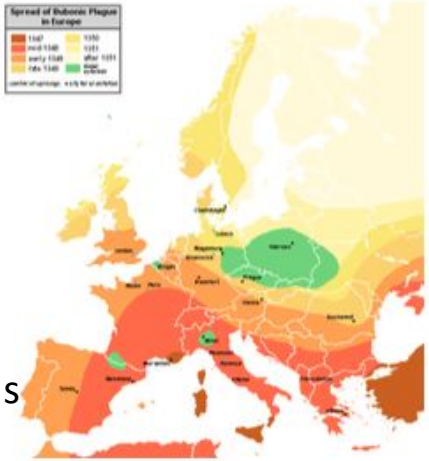
SARS

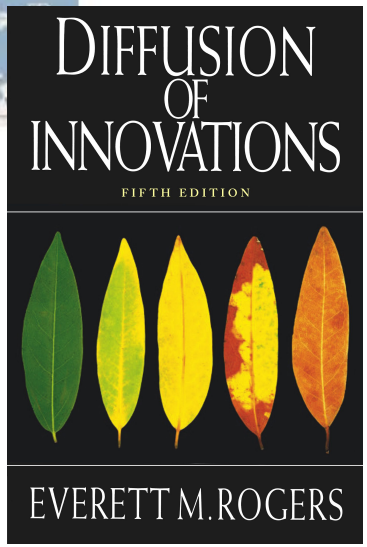




H1N1

The Great Plague





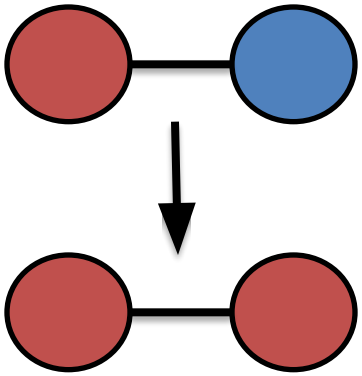
Epidemic Spreading,
Diffusion of Innovations and Ideas,
Gossip…

All those **phenomena** can be modeled as diffusive processes

Diffusive Processes and Networks

Diffusion implies network structure!

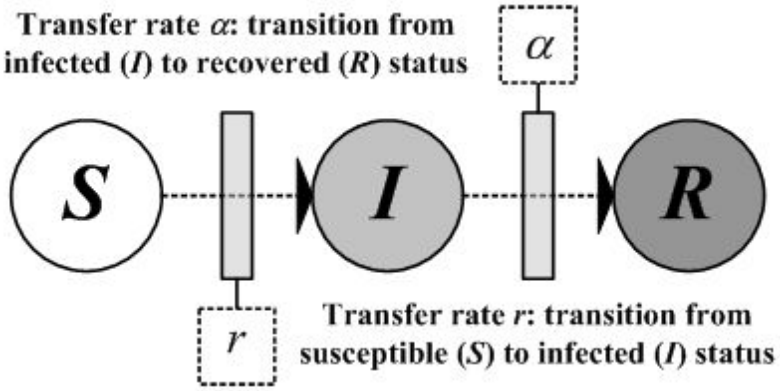
- It happens only when the carries of the diseases/virus/idea are **connected to each other**.



Diffusive phenomena can modeled by describing *"node statuses"* and *"transition rules"*

Example SIR model:

- Three node status:
  (**S**)usceptible, (**I**)nfected, (**R**)ecovered
- Two transition available:
  S->I; **I->R**

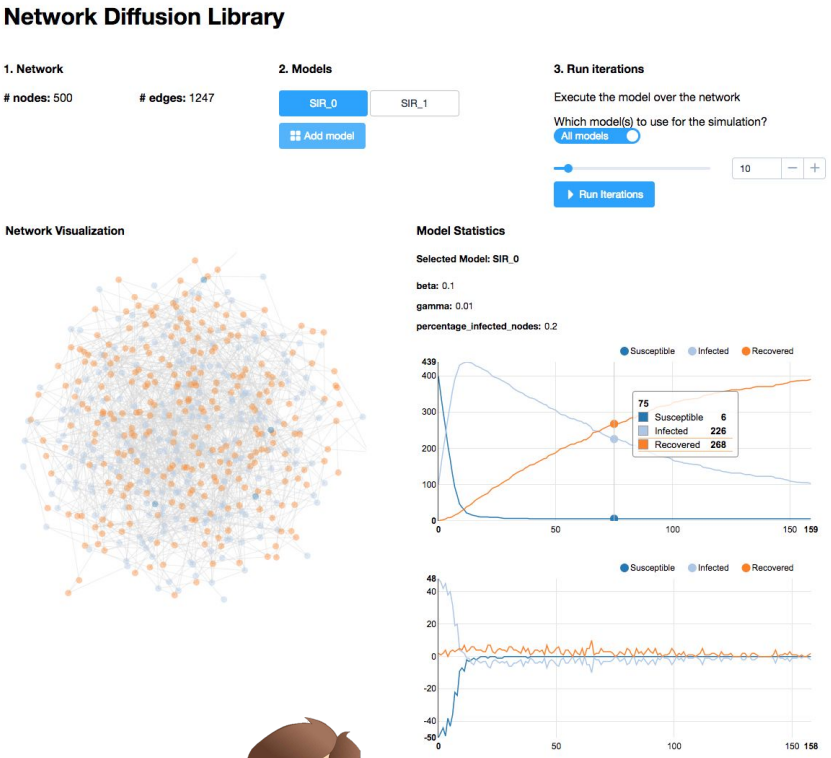


NDlib
Network Diffusion Library

A Network Diffusion Framework!

Simulate
Epidemics and
Opinion Dynamics
processes

Unfolding on top of
complex network structures



Programmatically
and **Visually**!

NDlib
Network Diffusion Library

Available Models

Epidemics

(10 Models)

- SI / SIS / SIR
- SEIS / SEIR /SWIR
- Threshold / Profile / Profile-Threshold / Threshold-Blocked
- Independent Cascades

Opinion Dynamics

(5 Models)

- Majority Rule
- Voter / Q-Voter
- Sznajd
- Cognitive Opinion Dynamics
- Algorithmic Bias

A unique simulation workflow!

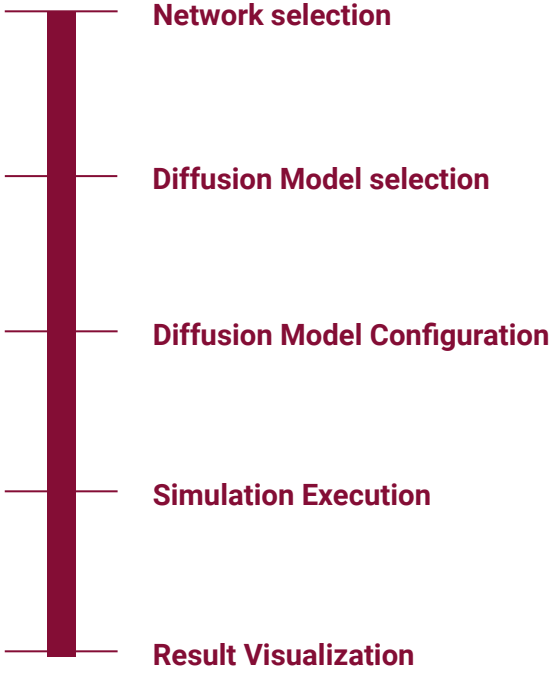
NDlib provide a common workflow to both programmers and analysts:

Programmers:

- Unified interface for several diffusion model
- Results Visualization facilities
- I/O standardization
- Extensibility

Analysts:

- Visual (web-based) platform
- Experiment configuration/execution
- Analytics as-a-service



Programmer:
SIR Code Example

A simple, unified, interface:

- Load the Graph
- Select and configure the model
- Run the simulation

All models follow the same programmatic pattern and produce standardized results

```
import networkx as nx
import ndlib.models.ModelConfig as mc
import ndlib.models.epidemics.SIRModel as sir

# Network topology
g = nx.erdos_renyi_graph(1000, 0.1)

# Model selection
model = sir.SIRModel(g)

# Model Configuration
cfg = mc.Configuration()
cfg.add_model_parameter('beta', 0.01)
cfg.add_model_parameter('gamma', 0.005)
cfg.add_model_parameter("percentage_infected", 0.05)
model.set_initial_status(cfg)

# Simulation execution
iterations = model.iteration_bunch(200)
```

Programmer:
Visual Analysis

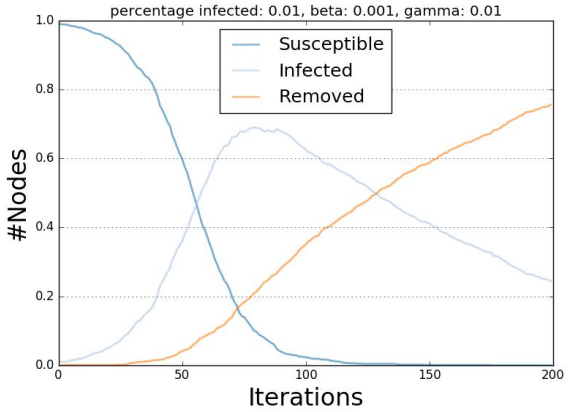
ndlib.viz implements visualization facilities

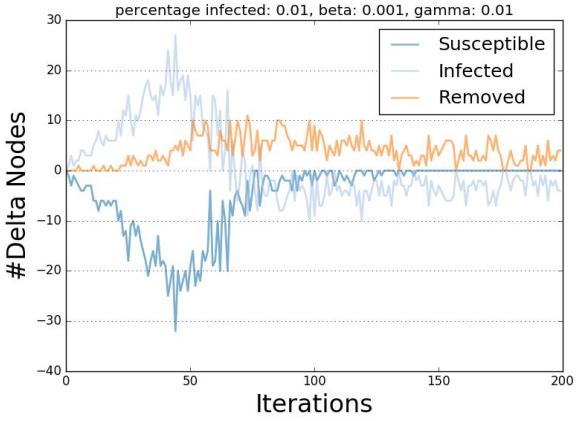
Base Viz

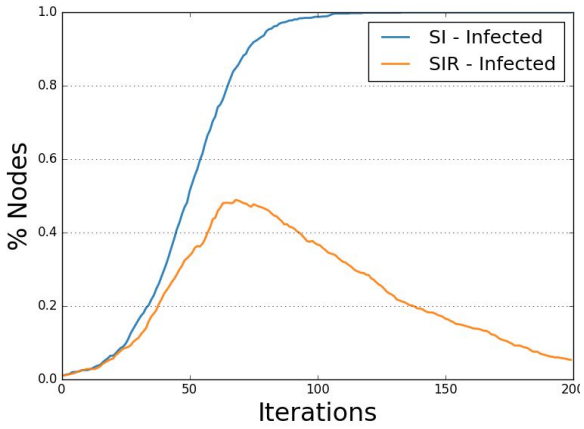
- Diffusion Trends
- Prevelence

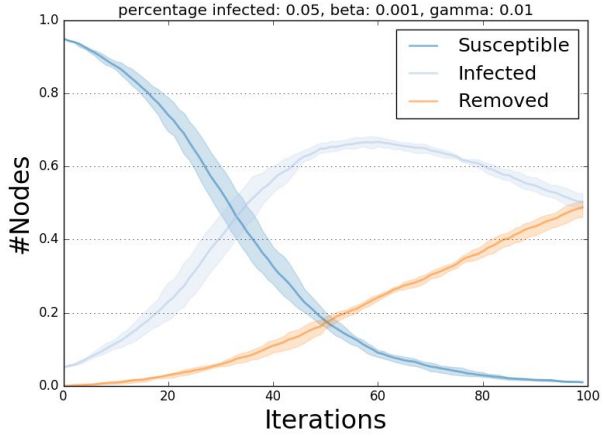
Advanced Viz

- Compare Models
- Multiple Run





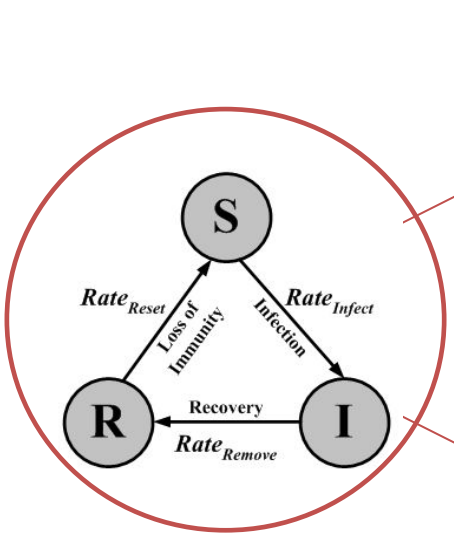




Programmer:
Describe New Models

New models can be added to NDlib easily:

1. Extend the base class ndlib.models.DiffusionModel
2. Specify model **metadata** *(i.e., available status, required parameters)*
3. Describe the **transition rules** *(i.e., under which circumstances a node becomes infected?)*



Metadata
- [S, I, R]
- [alpha, gamma]

Transition Rules
- S->I
- I->R

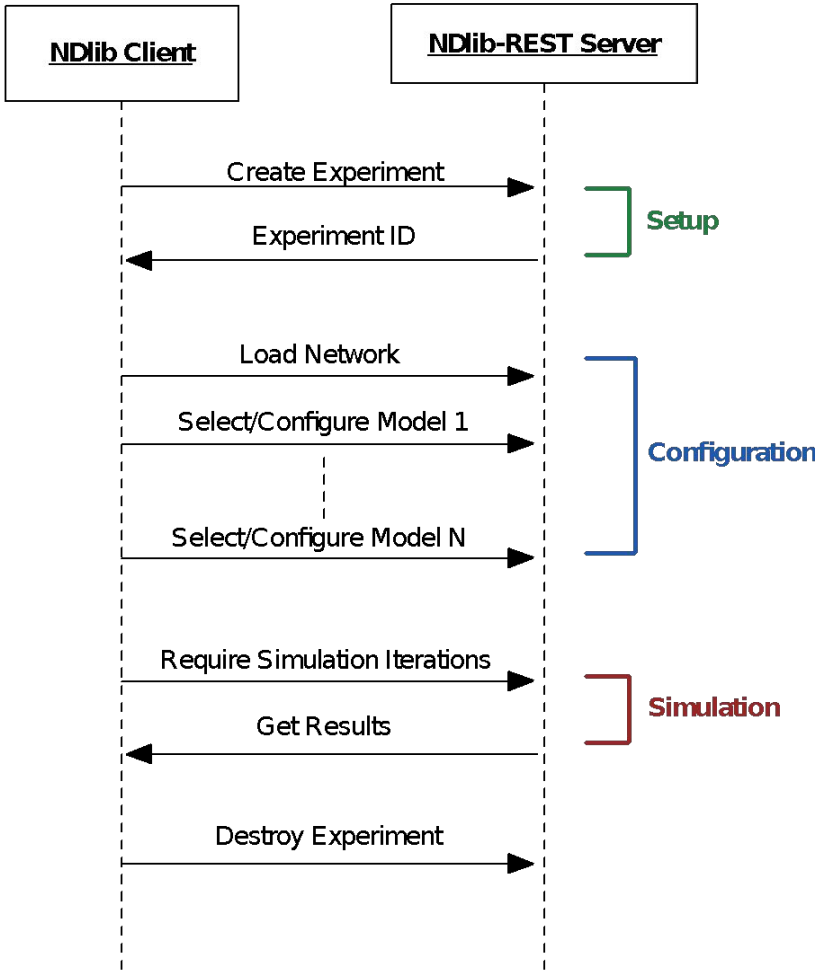
Programmer:
Remote Experiments

NDlib offers a **remote experiment server** that, using a REST-full API, allows to:

- Create Ndlib experiments
- Configure them
- Execute them remotely

NDlib-REST aims to:

1. Decouple experiment definition/execution
2. Increase scalability



Analyst:
Visual Simulation

Network Diffusion Library

1. Network

 # nodes: 500 **# edges:** 1247

2. Models

SIR_0 | SIR_1

Add model

3. Run iterations

Execute the model over the network

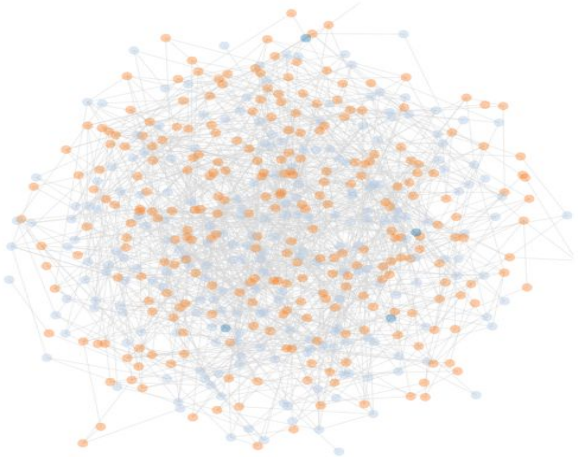
Which model(s) to use for the simulation?

All models

10

Run Iterations

Network Visualization



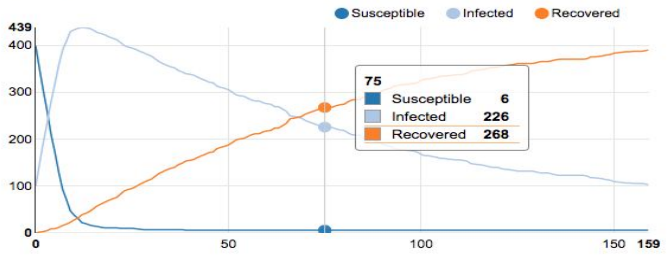
Model Statistics

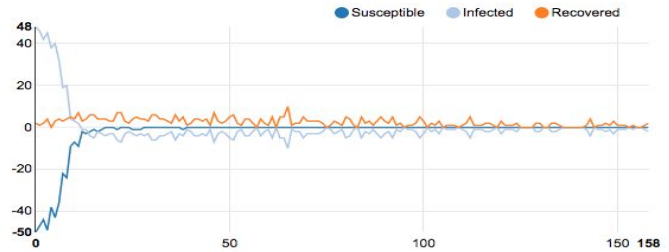
Selected Model: SIR_0

beta: 0.1

gamma: 0.01

percentage_infected_nodes: 0.2





Ndlib 4.0:
Advanced Features

Composite model definition

- Design diffusive models defining their transition rules as *trees* of atomic actions (compartments)

Support for Dynamic Network models

- Integration with DyNetX (ad-hoc library by CNR-UNIPI)

NDQL: Network Diffusion Query Language

- High-level query language for defining diffusion processes

```
CREATE_NETWORK g1
TYPE erdos_renyi_graph
PARAM n 300
PARAM p 0.1

MODEL SI

STATUS Susceptible
STATUS Infected

# Compartment definitions

COMPARTMENT c1
TYPE NodeStochastic
PARAM rate 0.1
TRIGGER Infected

# Rule definitions

RULE
FROM Susceptible
TO Infected
USING c1

# Model configuration

INITIALIZE
SET Infected 0.1

EXECUTE SI ON g1 FOR 100
```

When:

Rigth now, NDlib v4.0.1 is out!

Where:

- Pypi: https://pypi.python.org/pypi/ndlib
- GitHub NDlib: https://github.com/GiulioRossetti/ndlib
- GitHub NDlib-REST: https://github.com/GiulioRossetti/ndlib-rest
- Documentation: http://ndlib.readthedocs.io/
- SoBigData: http://www.sobigdata.eu