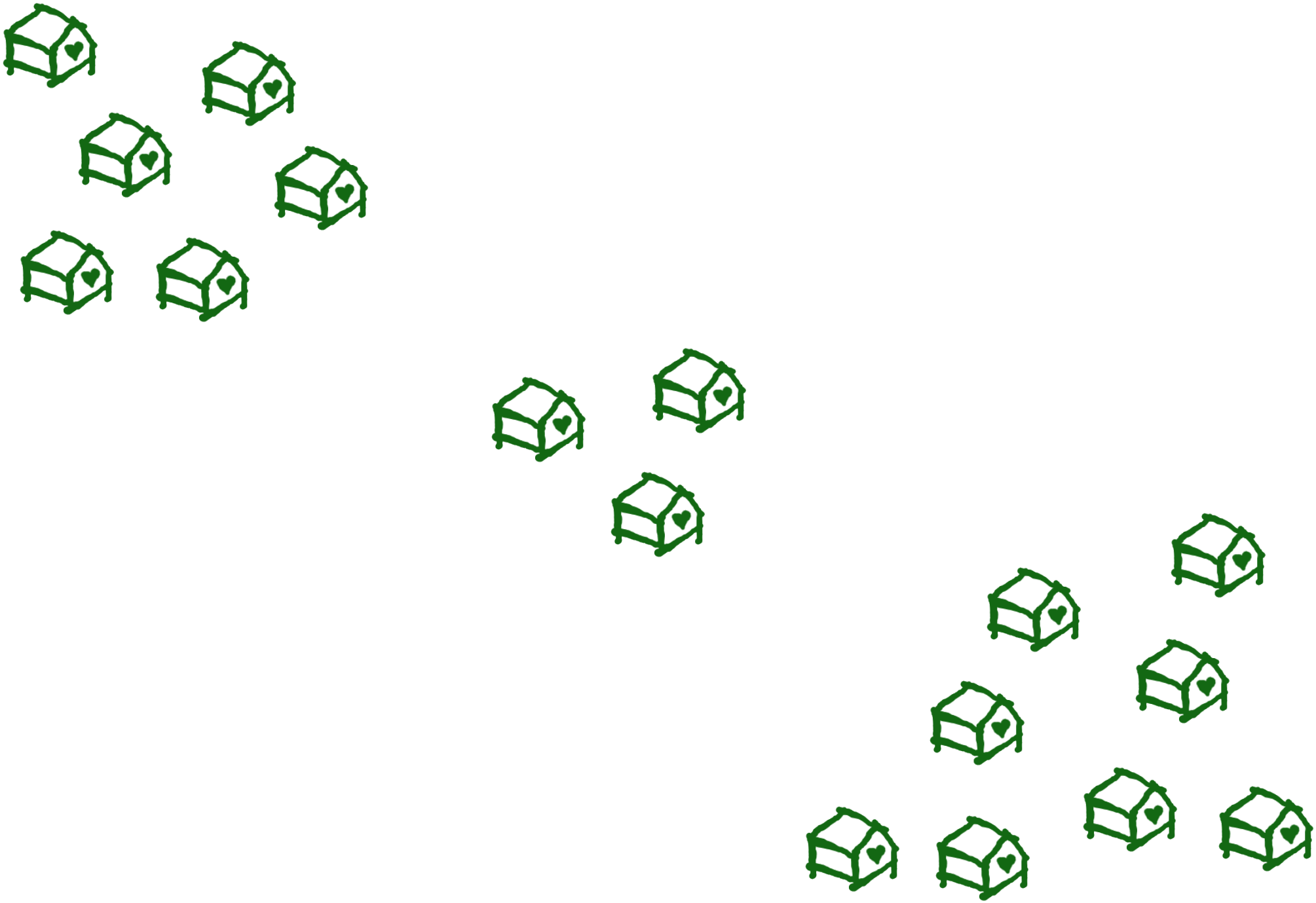
# How python™ is guiding infrastructure construction in Africa

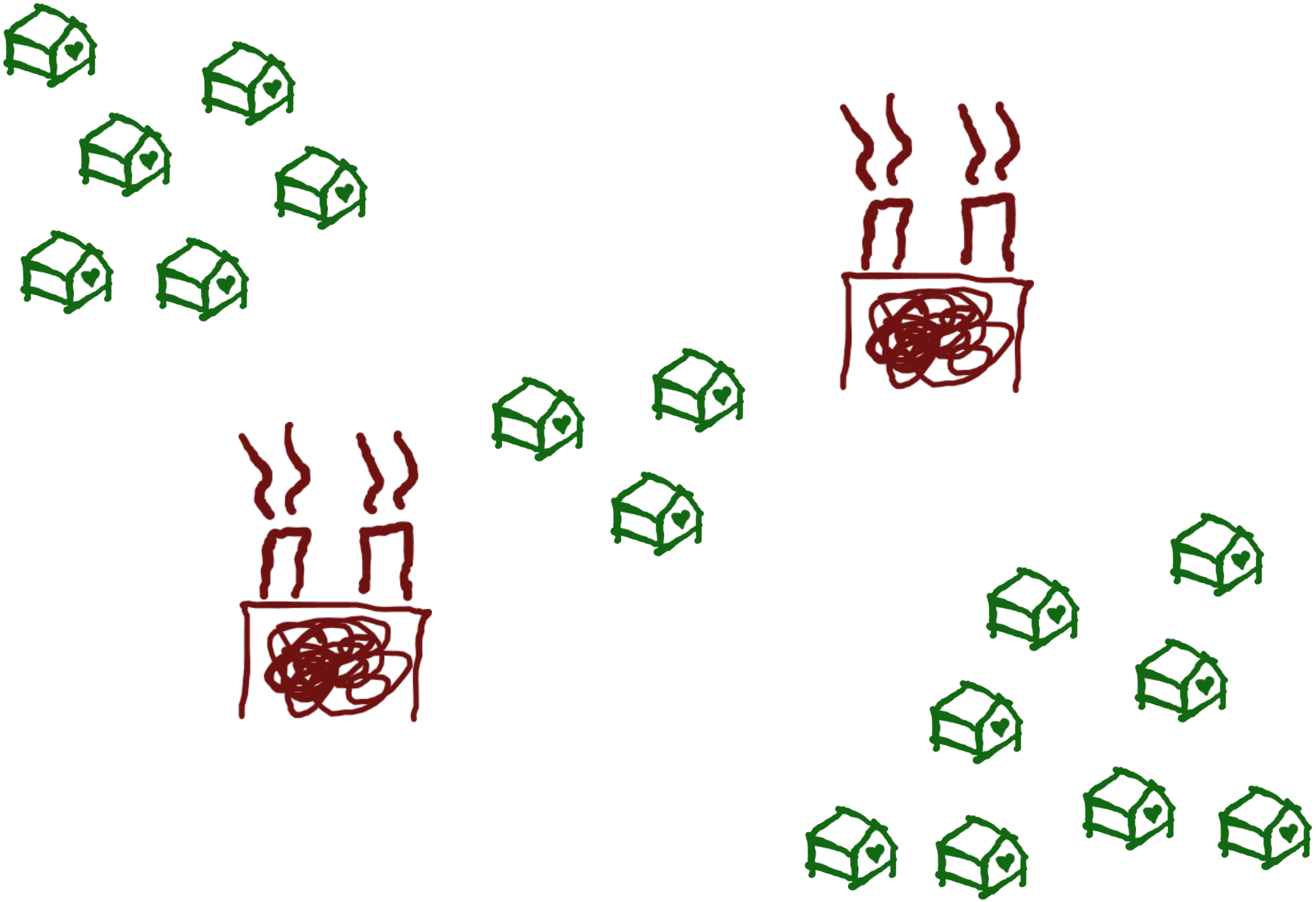
Roy Hyunjin Han
Lead Software Engineer
Modi Research Group
Earth Institute at Columbia University

2010 PyCon Atlanta

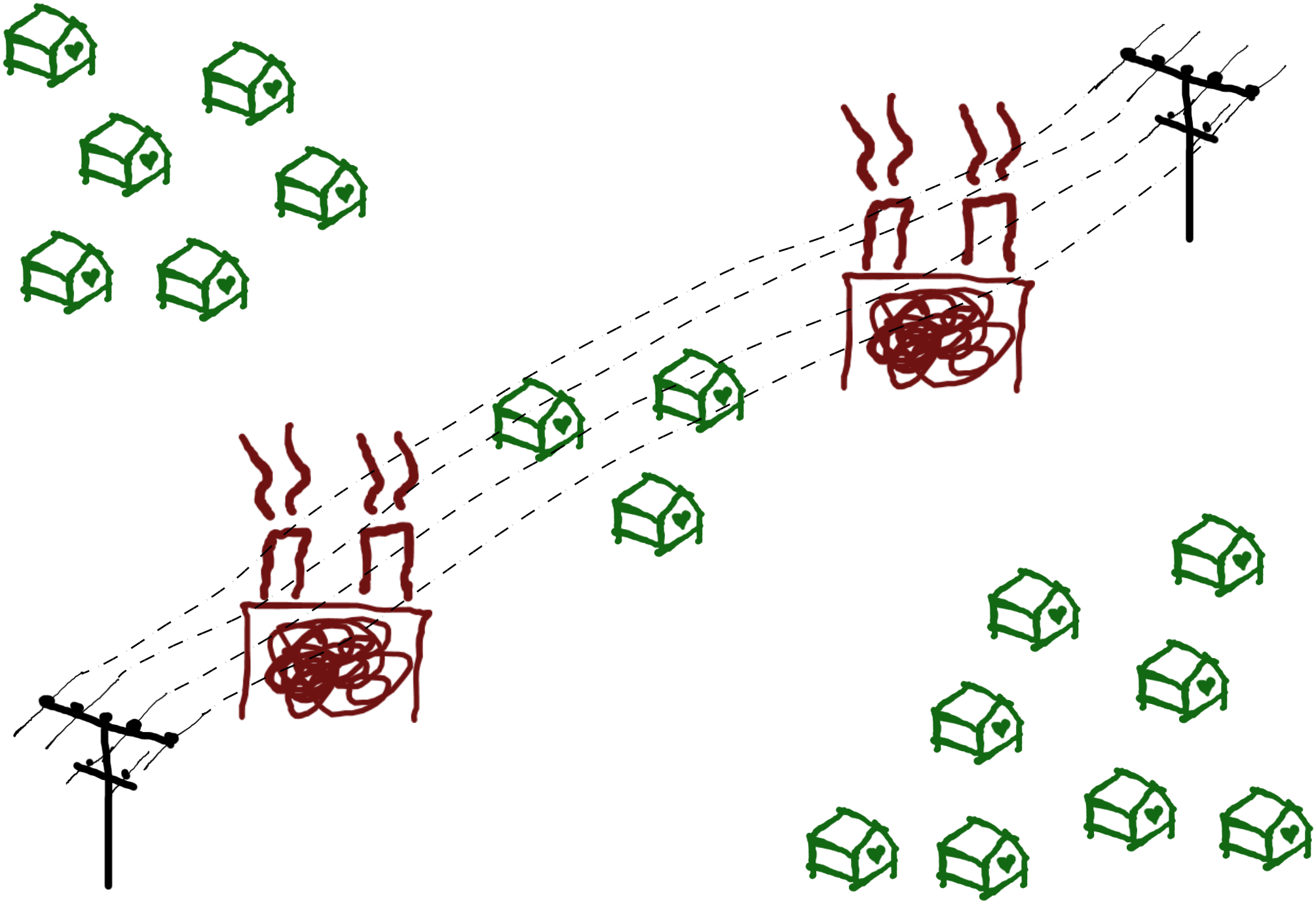
# The cost of poor planning



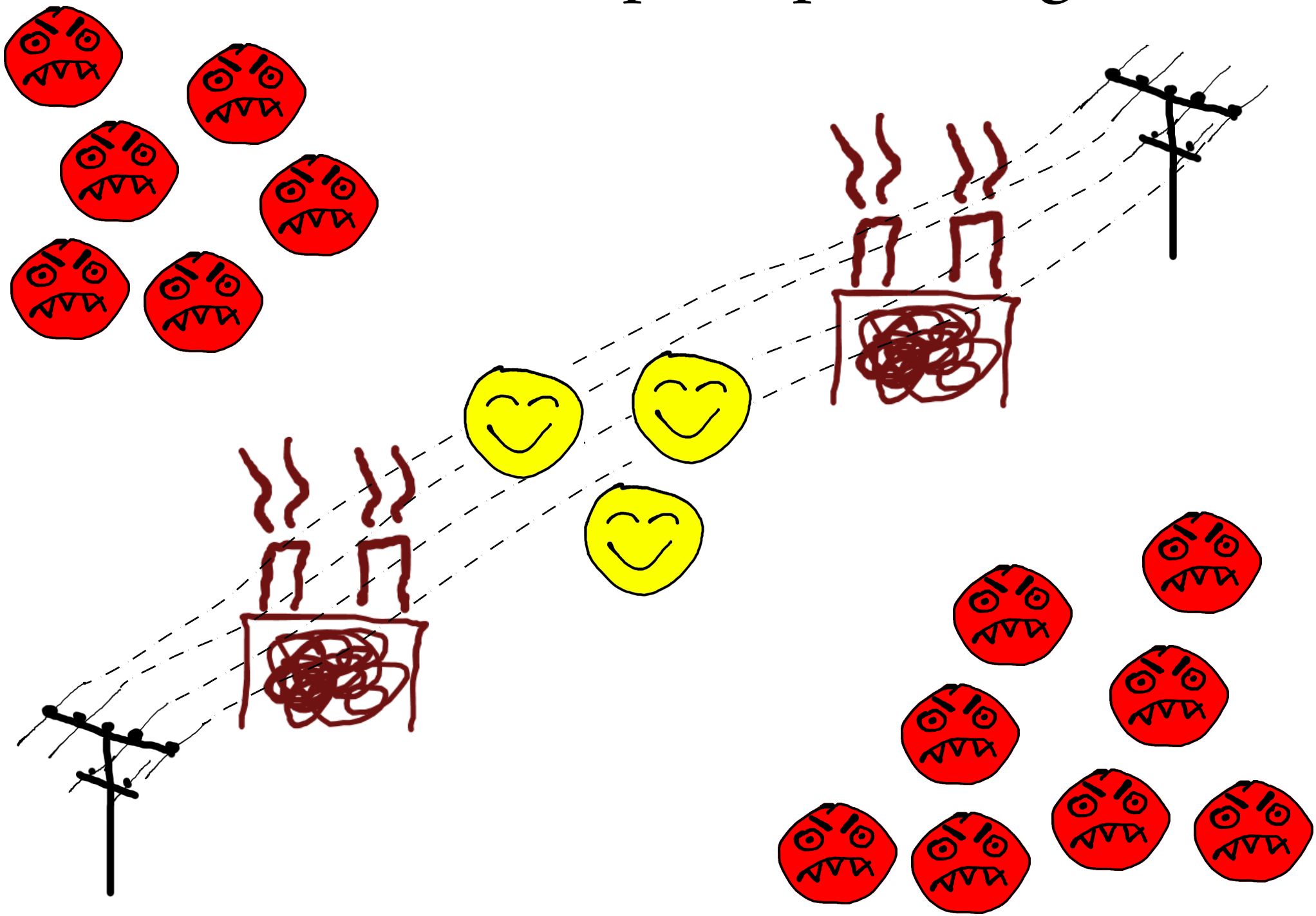
# The cost of poor planning



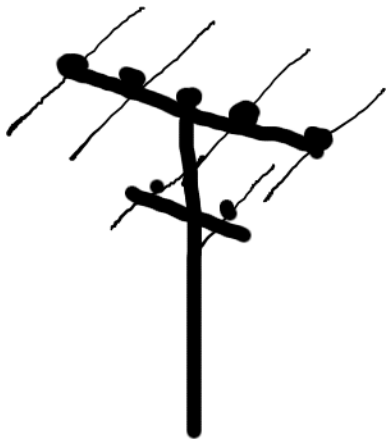
# The cost of poor planning



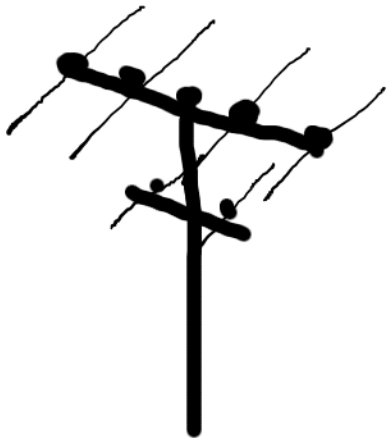
# The cost of poor planning



# Give people electricity access

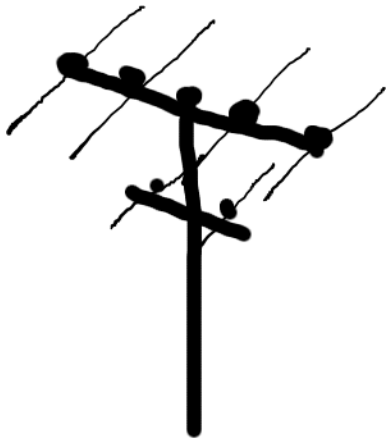


# Give people electricity access



# Give people electricity access

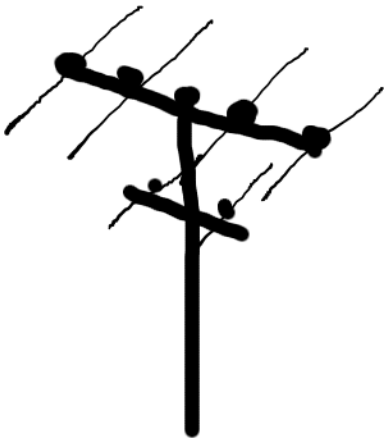
## Country goes to bank



# Give people electricity access

## Country goes to bank

Bank recruits consultants, who talk to ministers, utility owners, local leaders

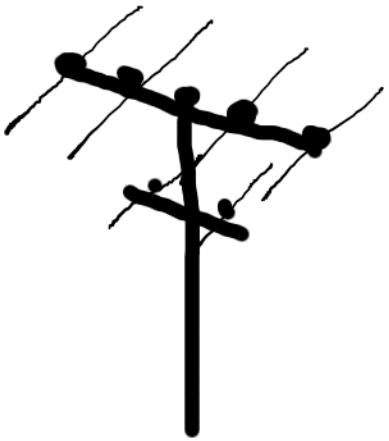


# Give people electricity access

Country goes to bank

Bank recruits consultants, who talk to ministers, utility owners, local leaders

Consultants draft plan



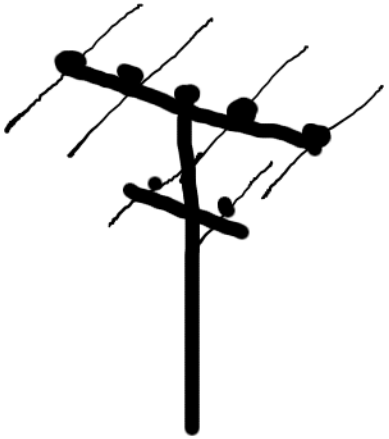
# Give people electricity access

Country goes to bank

Bank recruits consultants, who talk to ministers, utility owners, local leaders

Consultants draft plan

Bank sends plan to financiers



# Give people electricity access

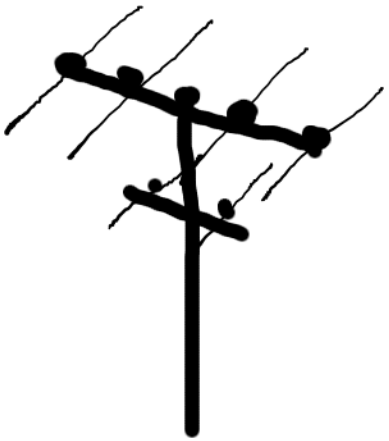
Country goes to bank

Bank recruits consultants, who talk to ministers, utility owners, local leaders

Consultants draft plan

Bank sends plan to financiers

Financiers fund construction



# Give people electricity access

Country goes to bank

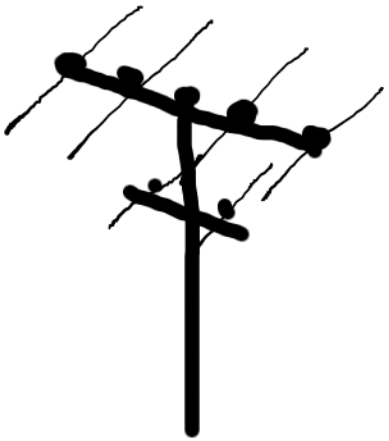
Bank recruits consultants, who talk to ministers, utility owners, local leaders

Consultants draft plan

Bank sends plan to financiers

Financiers fund construction

People get electricity

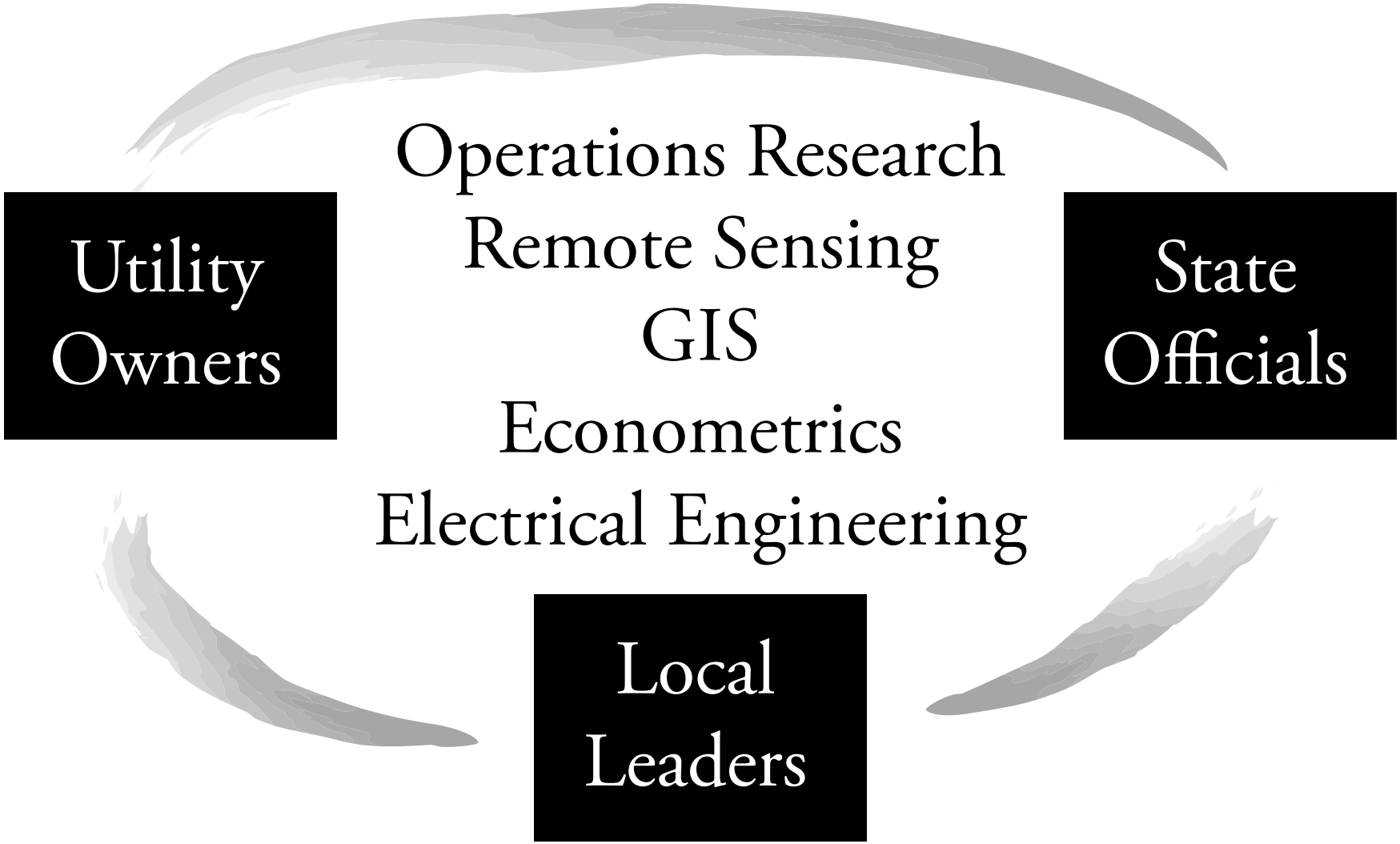


A credit-worthy infrastructure plan

IS GROUNDED IN
geospatial and economic **analysis**

IS THE RESULT OF
**negotiation** between stakeholders

# By making expert technical analysis accessible, we enable policymakers to focus on negotiation

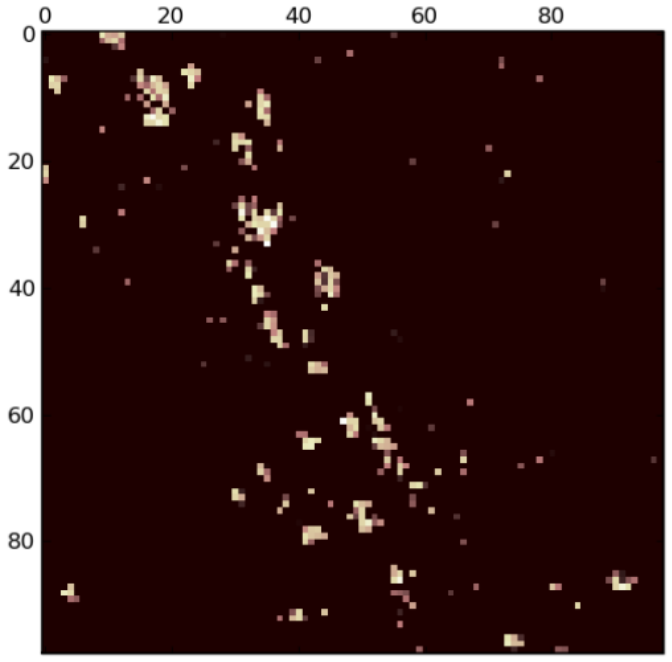


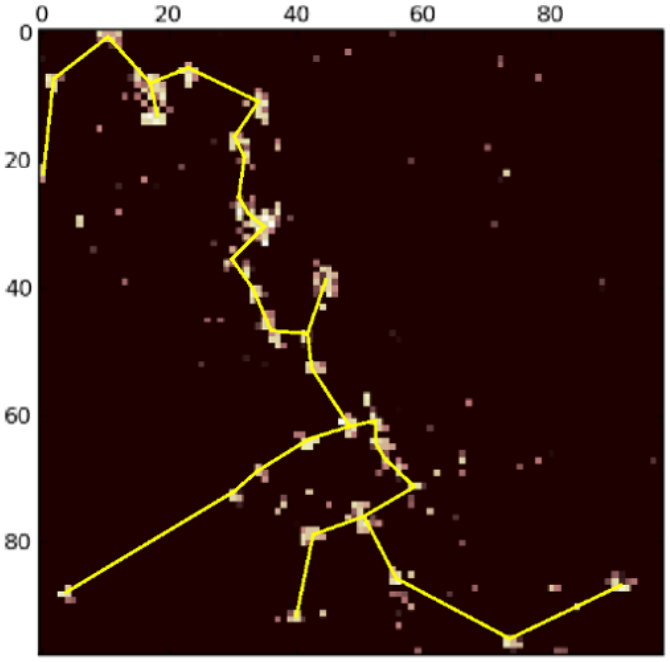
# A decision support tool for planning infrastructure

Where do people live?
What is the expected demand at a node?
How much will it cost to connect a node?
Using which technology?
In what order?

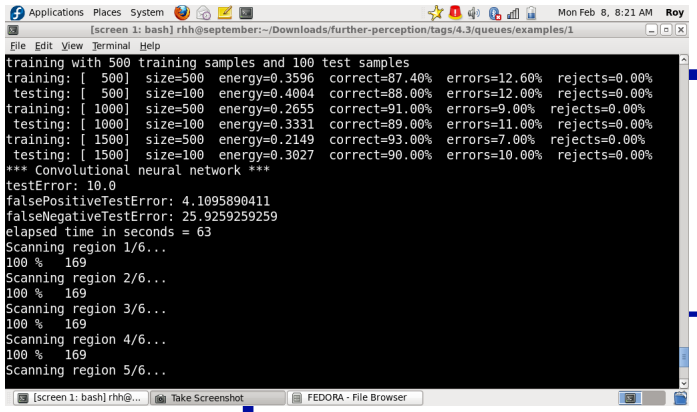
# Architecture

**Computational core**

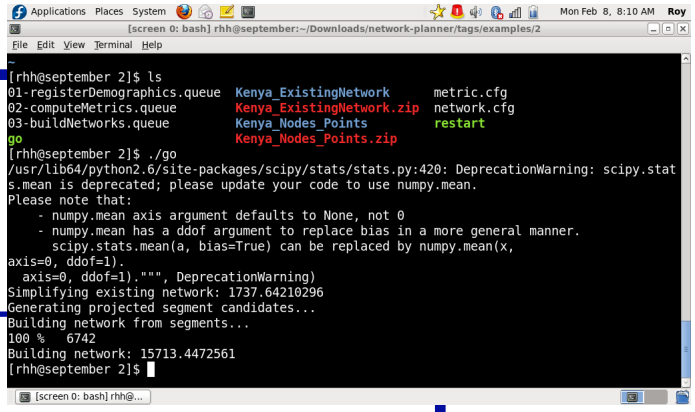


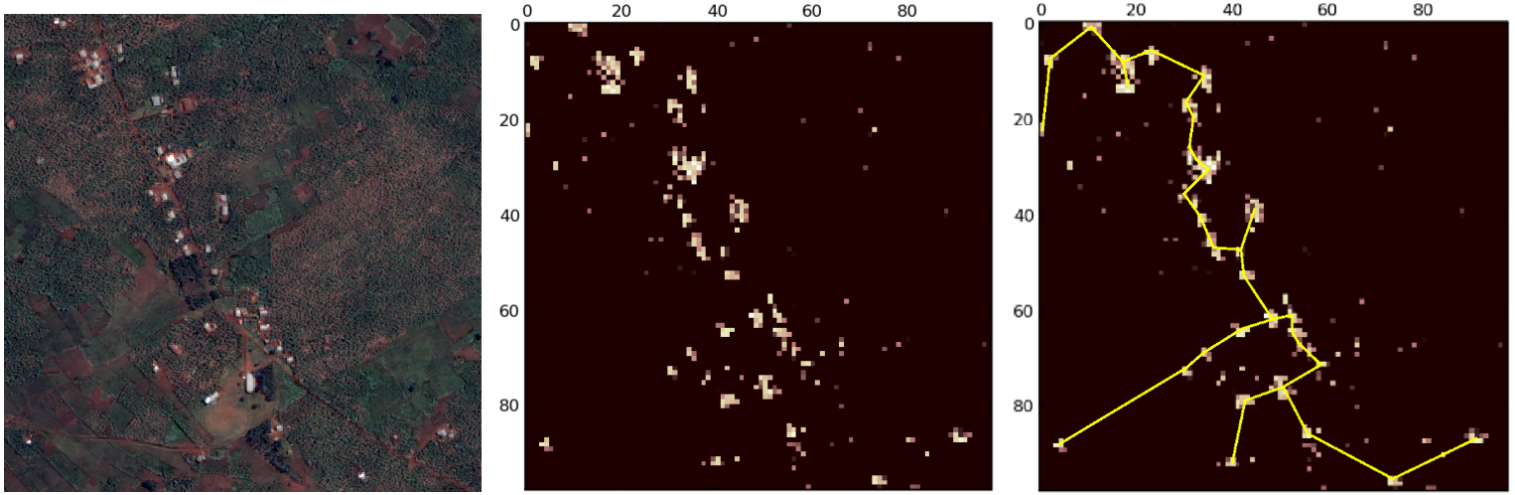
# Architecture




## Command-line utilities

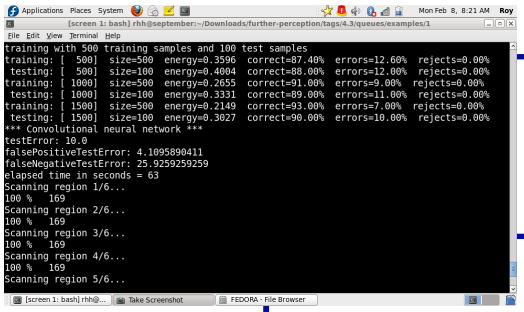



## Computational core

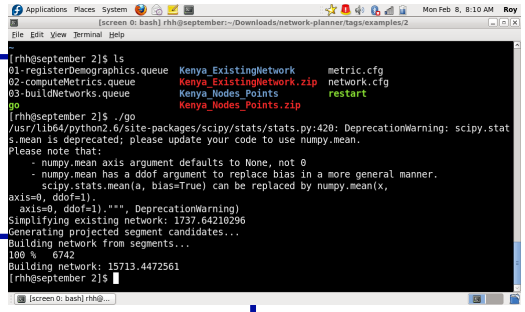


# Architecture

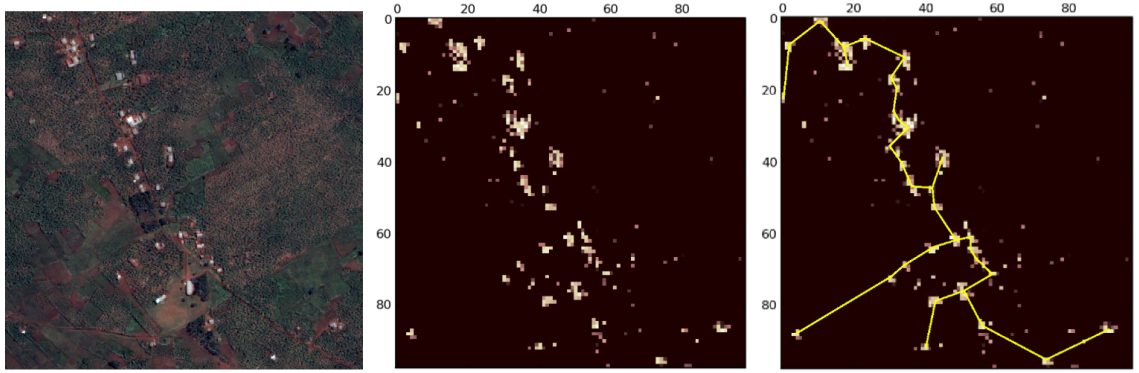
## Web service



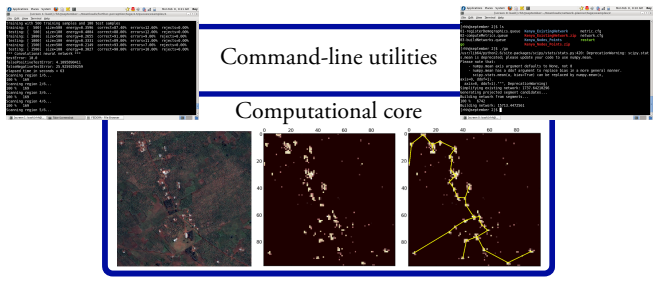

Command-line utilities

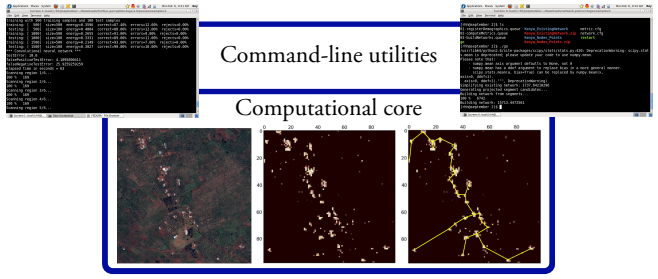


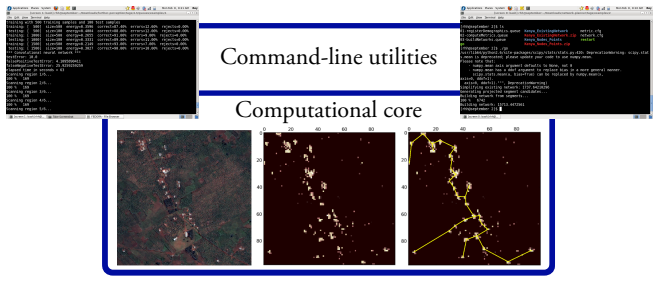

Computational core



# Architecture



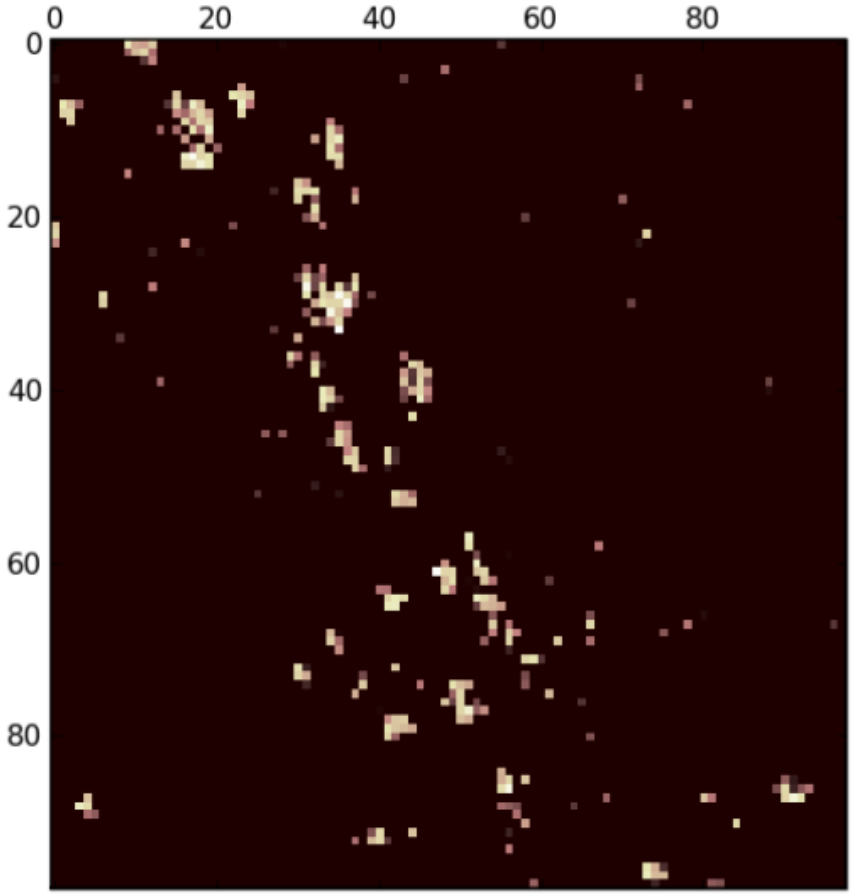




**Scalable web service**

# Success story #1: Remote sensing

| | | |
|---|---|---|
| Question | Where do people live? | |
| Manual | Send people with GPS devices<br>Click on houses in satellite images | |
| Method | Machine learning | Image recognition |
| Command line | subprocess \| Lush | osgeo \| GDAL Generators |
| Scalable web | Pylons \| SQLAlchemy | ampqlib \|RabbitMQ |

# Computational core: Remote sensing



# Where do people live?

# Where do people live?

# How do we mark houses automatically?

1. Train local classifier to recognize houses

# How do we mark houses automatically?

1. Train local classifier to recognize houses

House? Yes
Confidence 88%

House? No
Confidence 95%

# How do we mark houses automatically?

1. Train local classifier to recognize houses

Convolutional Neural Networks (LeNet5)

Yann LeCun, Courant Institute, NYU

- Recognizes houses correctly at least 98% of the time when trained with grayscale images

# How to scan an image with a classifier

Increasing scanRatio results in more overlapping windows. The scanGeoInterval is the number of meters the scanning window moves horizontally and vertically and is a fraction of windowGeoLength. Thus if windowGeoLength=20m and scanRatio=2 then scanGeoInterval=10m.

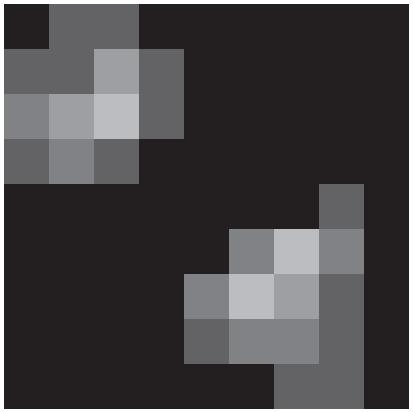
# How to scan an image with a classifier

Increasing scanRatio results in more overlapping windows. The scanGeoInterval is the number of meters the scanning window moves horizontally and vertically and is a fraction of windowGeoLength. Thus if windowGeoLength=20m and scanRatio=2 then scanGeoInterval=10m.

# How to scan an image with a classifier

Increasing scanRatio results in more overlapping windows. The scanGeoInterval is the number of meters the scanning window moves horizontally and vertically and is a fraction of windowGeoLength. Thus if windowGeoLength=20m and scanRatio=2 then scanGeoInterval=10m.

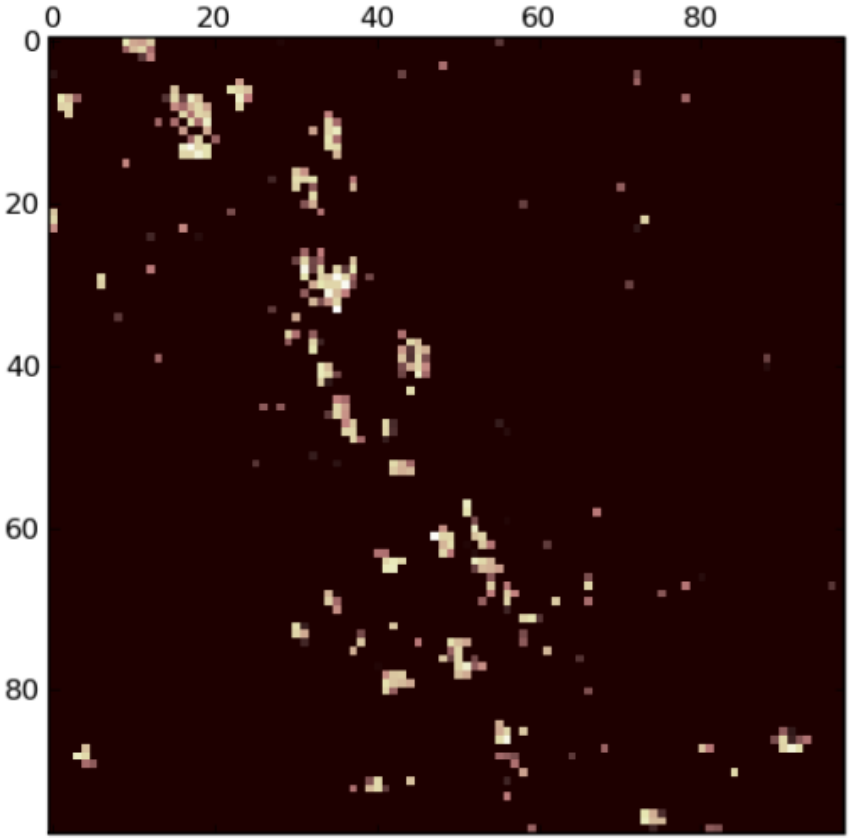
# How to scan an image with a classifier



Increasing scanRatio results in more overlapping windows. The scanGeoInterval is the number of meters the scanning window moves horizontally and vertically and is a fraction of windowGeoLength. Thus if windowGeoLength=20m and scanRatio=2 then scanGeoInterval=10m.

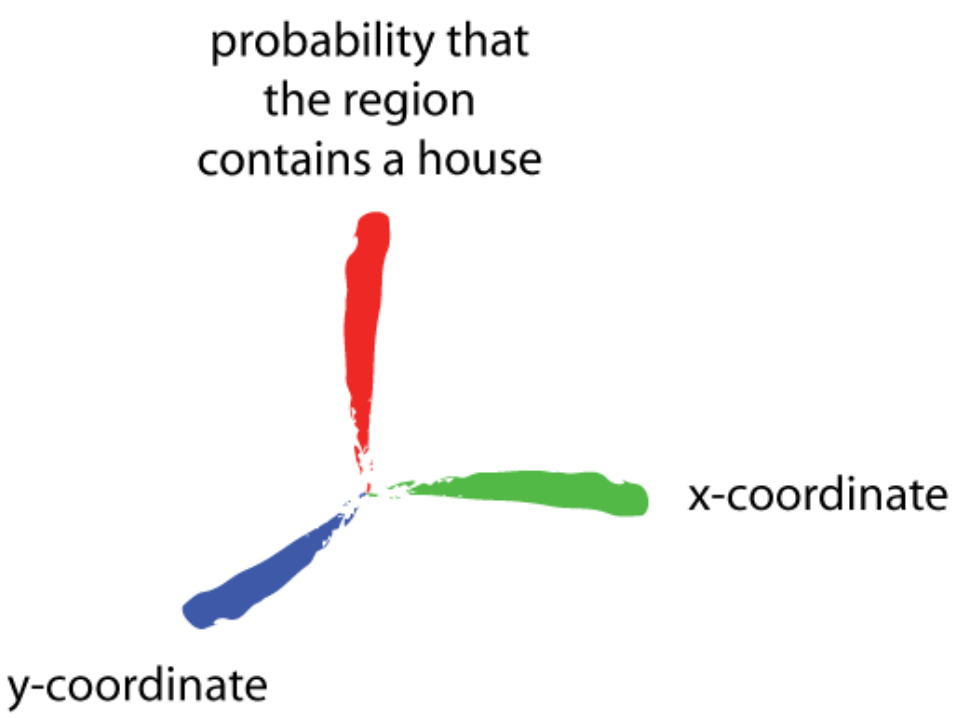
# How do we mark houses automatically?

2. Scan local classifier over image to generate a matrix of probabilities



# How do we mark houses automatically?

3. Cluster in three dimensions to turn local probabilities into points



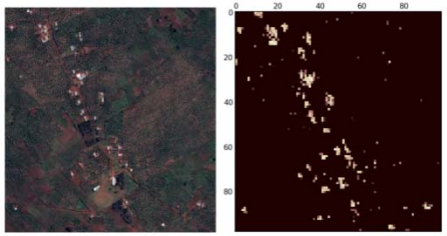

# How do we mark houses automatically?

3. Cluster in three dimensions to turn local probabilities into points

Use K-means

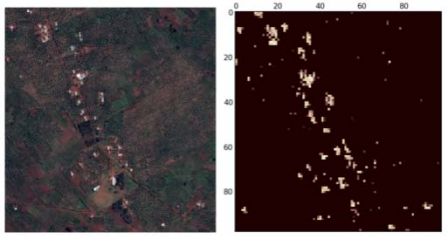
Pop large clusters

Remove small clusters



# Computational core: Remote sensing

| | |
|---|---|
| Load raster data from images<br>Save vector data to shapefiles | `import osgeo.gdal`<br>`import osgeo.ogr` |
| Run Lisp machine learning code | `import subprocess` |
| Cluster points<br>Save matrices | `import scipy.cluster.vq`<br>`import scipy.io` |
| Scan image with little memory | `yield` |
| Render 16-bit satellite image | `import matplotlib` |



# Computational core: Remote sensing

Run Lisp machine learning code

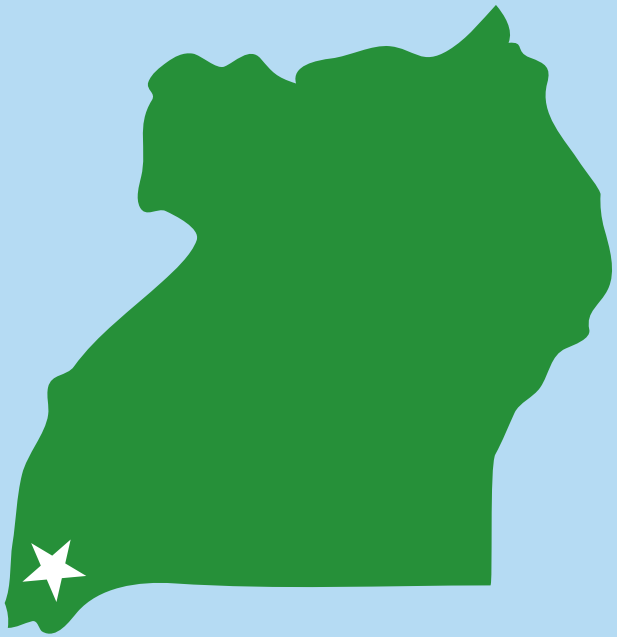
```
import subprocess
```

```
p = subprocess.Popen(programArguments,
    stdin=subprocess.PIPE,
    stdout=subprocess.PIPE)

for imageData in imageDataGenerator:
     # Send imageData to Lisp program
     p.stdin.write(imageData + '\n')
     # Receive result from Lisp program
     result = p.stdout.readline().rstrip()
```

# Where is Uganda?

# Where is Ruhiira?



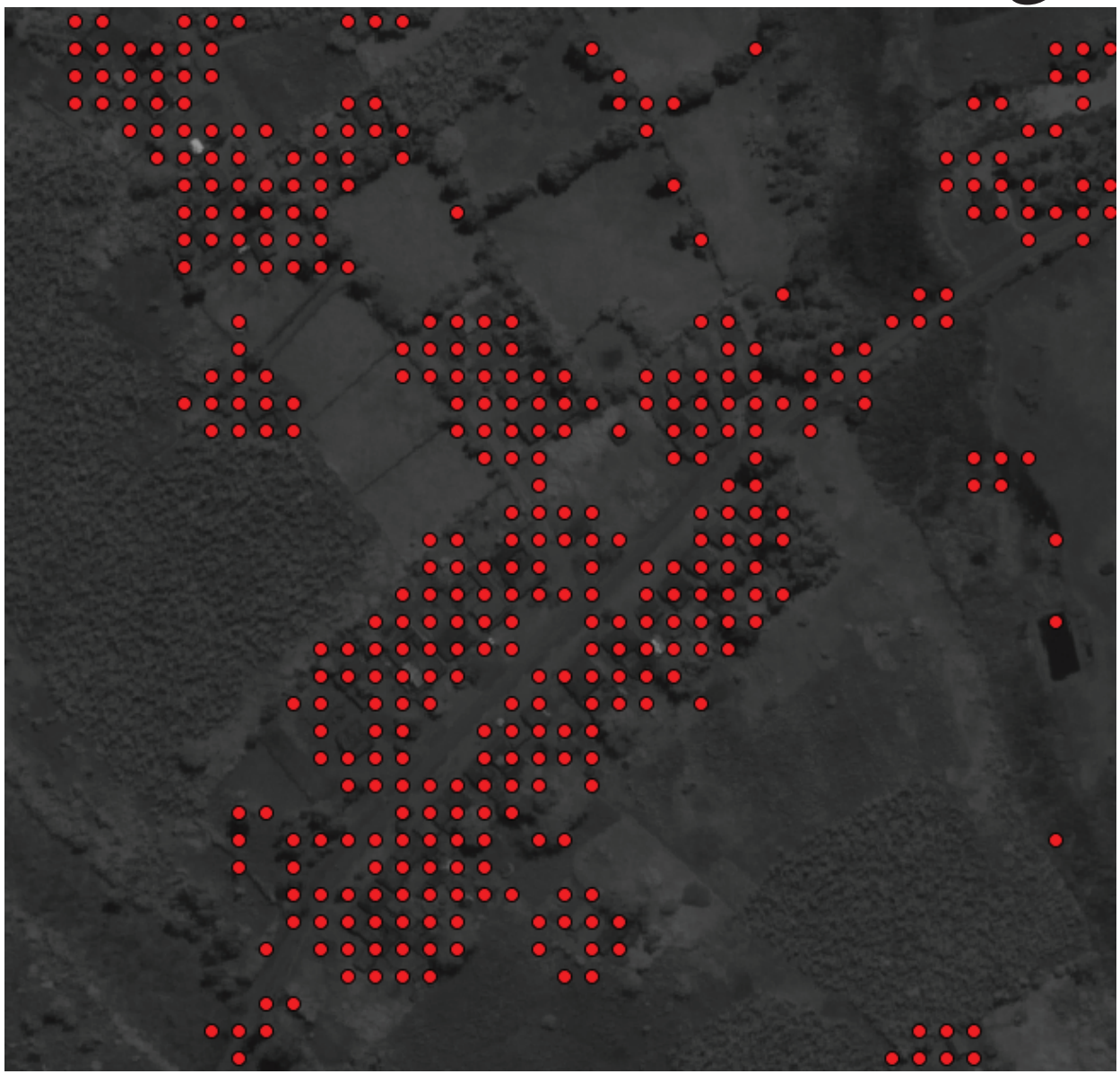
# Results from Ruhiira in Uganda

Farm 2 - Multispectral

# Results from Ruhiira in Uganda

Farm 2 - Panchromatic

# Results from Ruhiira in Uganda

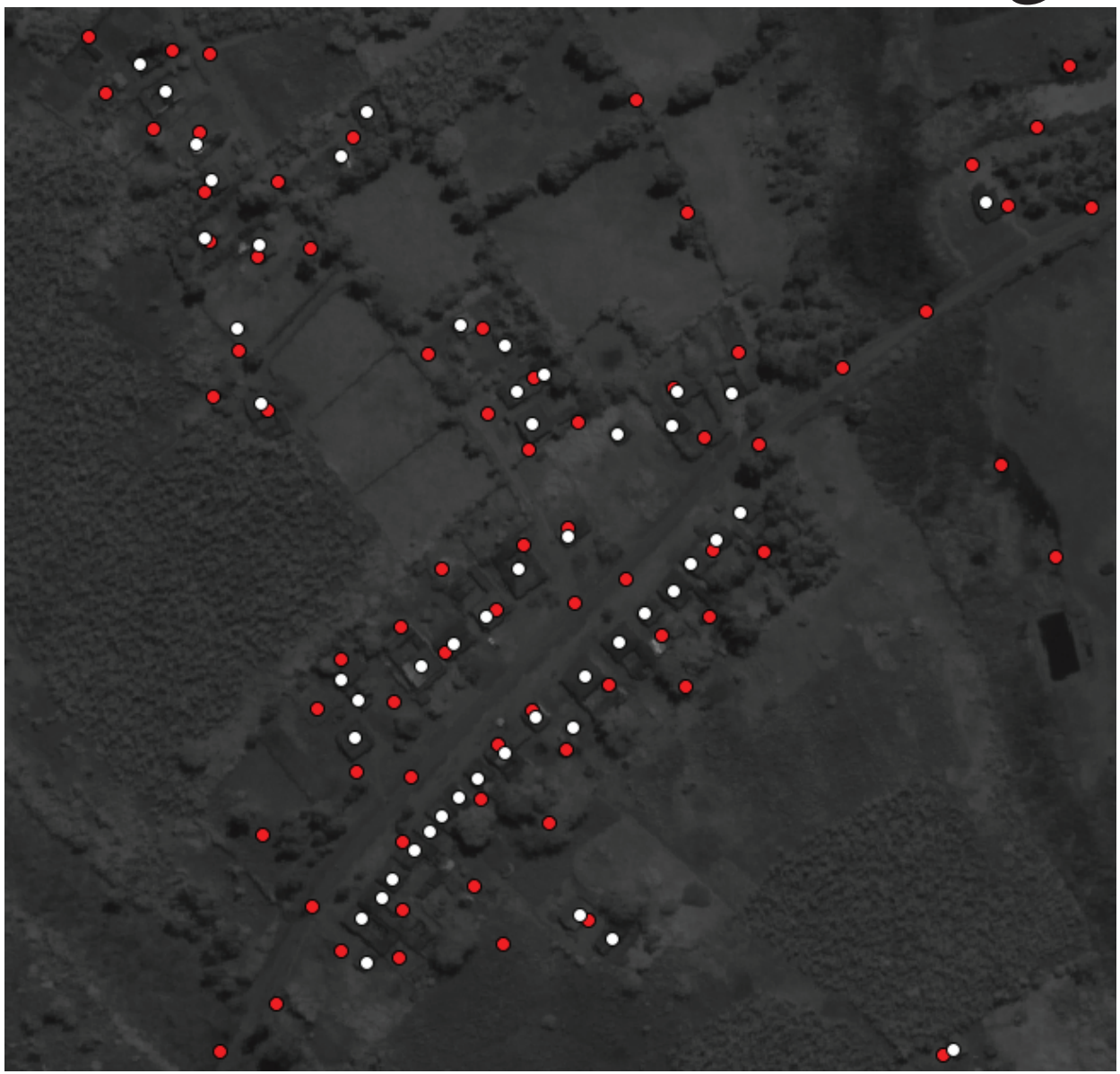


Farm 2 - Computer probabilities

# Results from Ruhiira in Uganda

Farm 2 - Computer

# Results from Ruhiira in Uganda



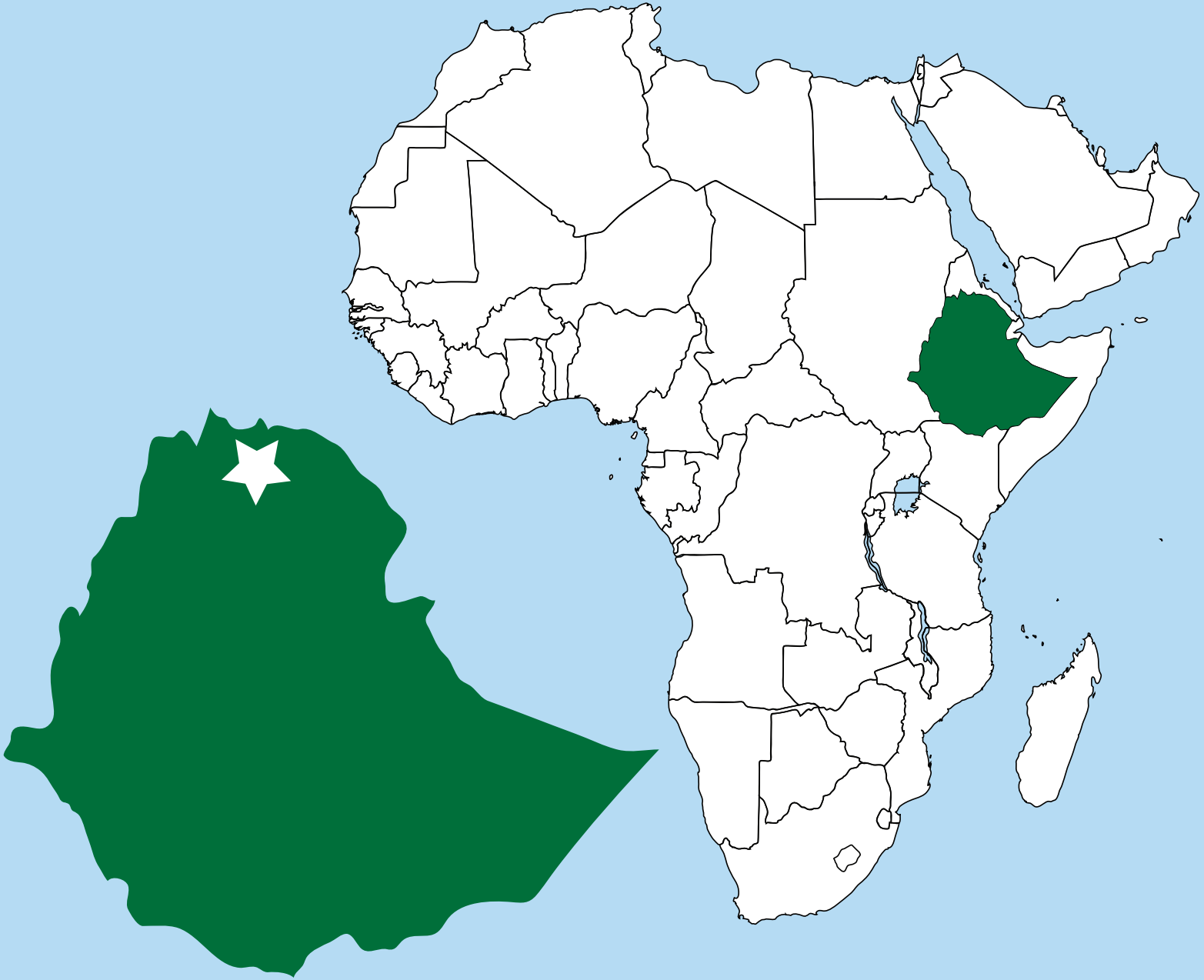
Farm 2 - Computer & Human

# Results from Ruhiira in Uganda

Farm 2 - Human

# Where is Ethiopia?

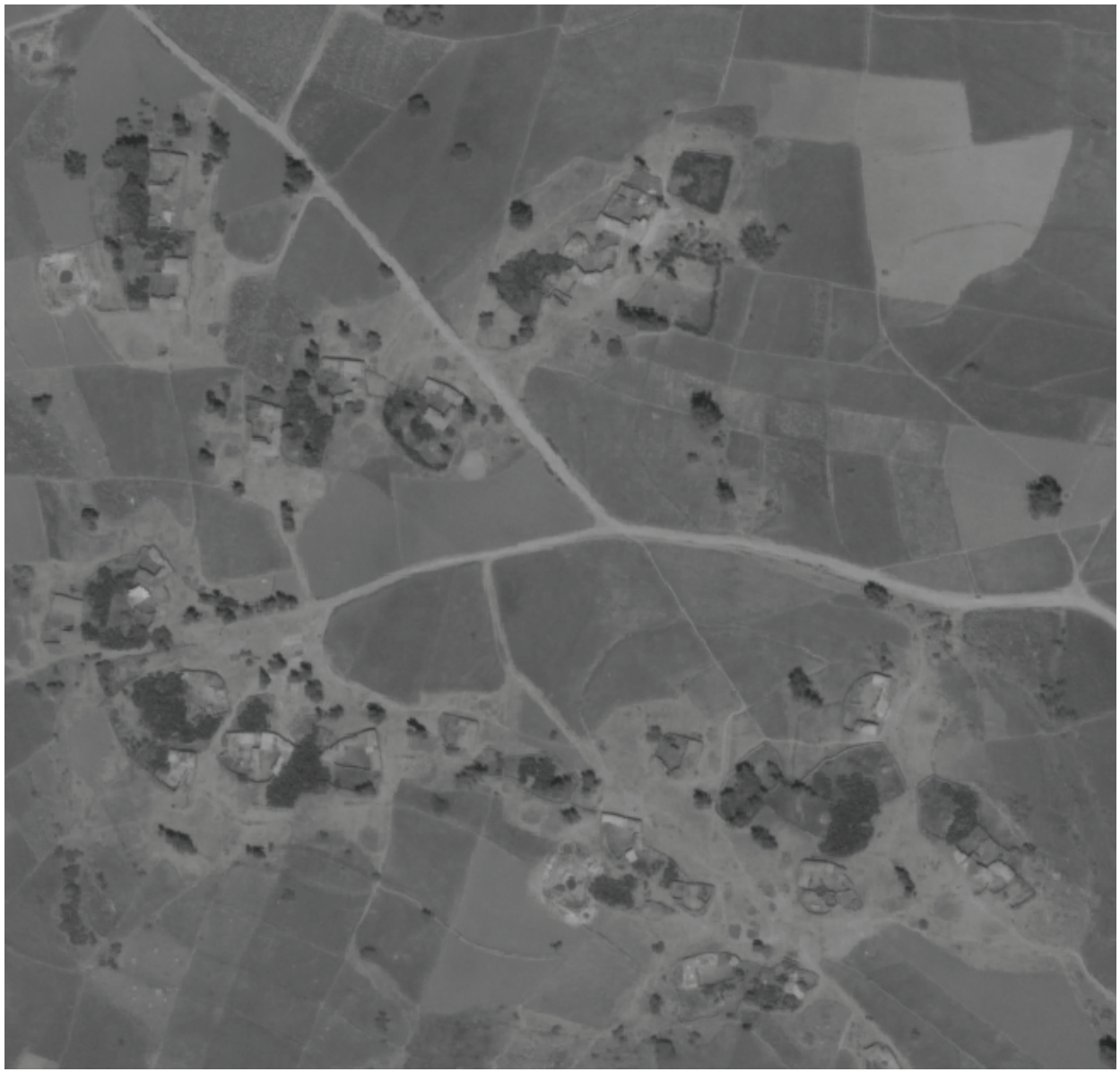
# Where is Koraro?



# Results from Koraro in Ethiopia

Farm 1 - Multispectral

# Results from Koraro in Ethiopia

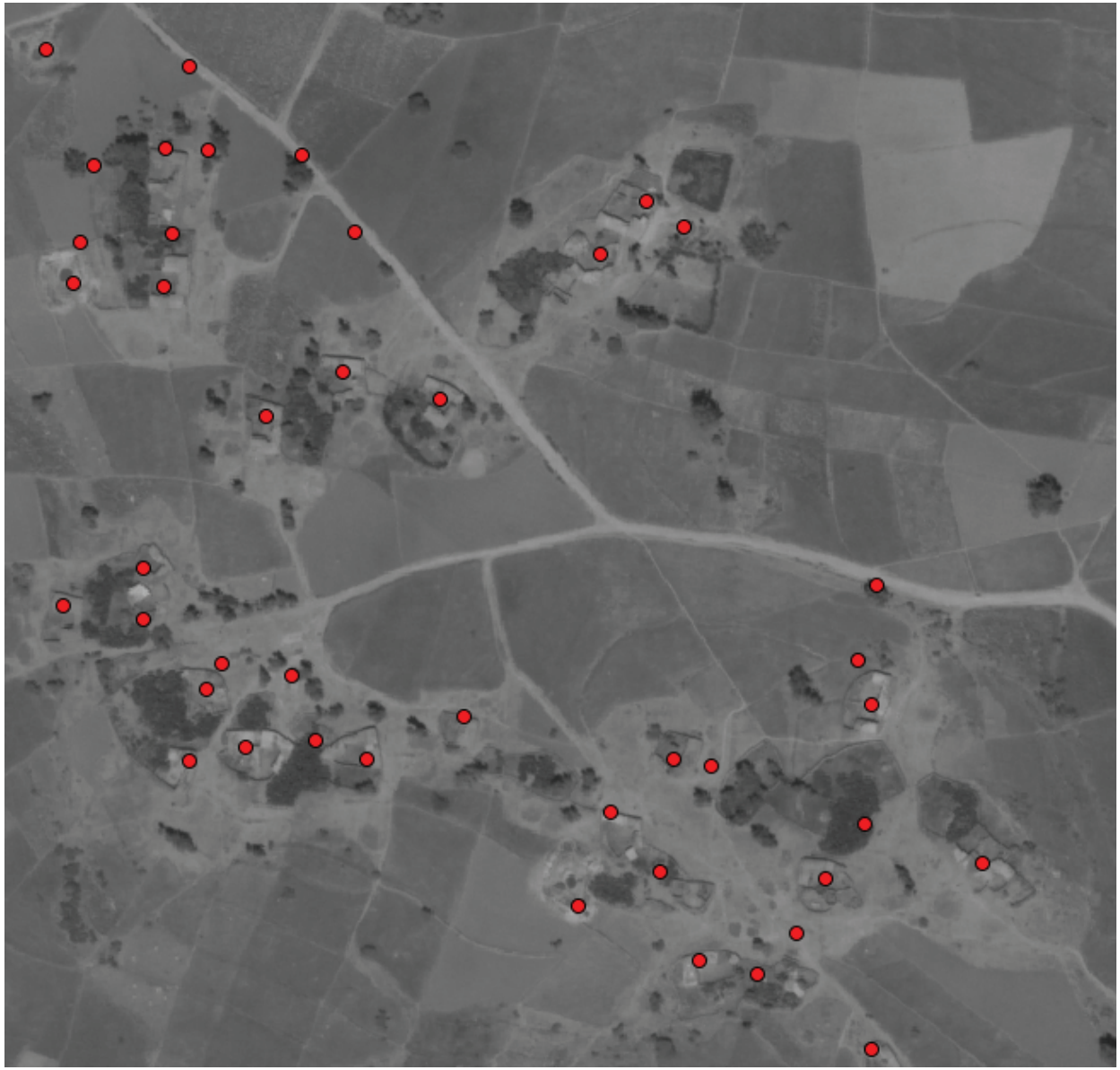


Farm 1 - Panchromatic

# Results from Koraro in Ethiopia

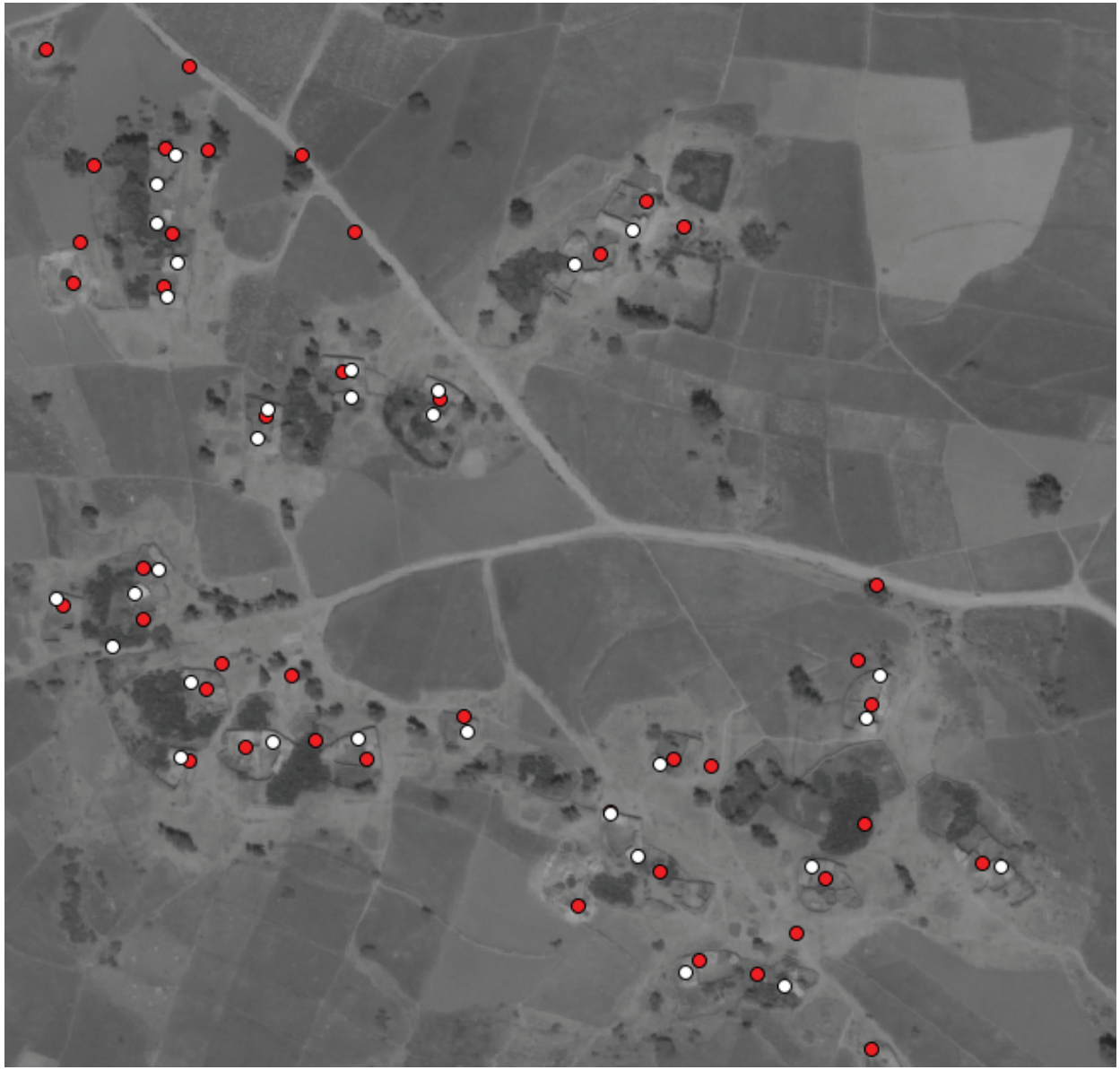
Farm 1 - Computer probabilities

# Results from Koraro in Ethiopia



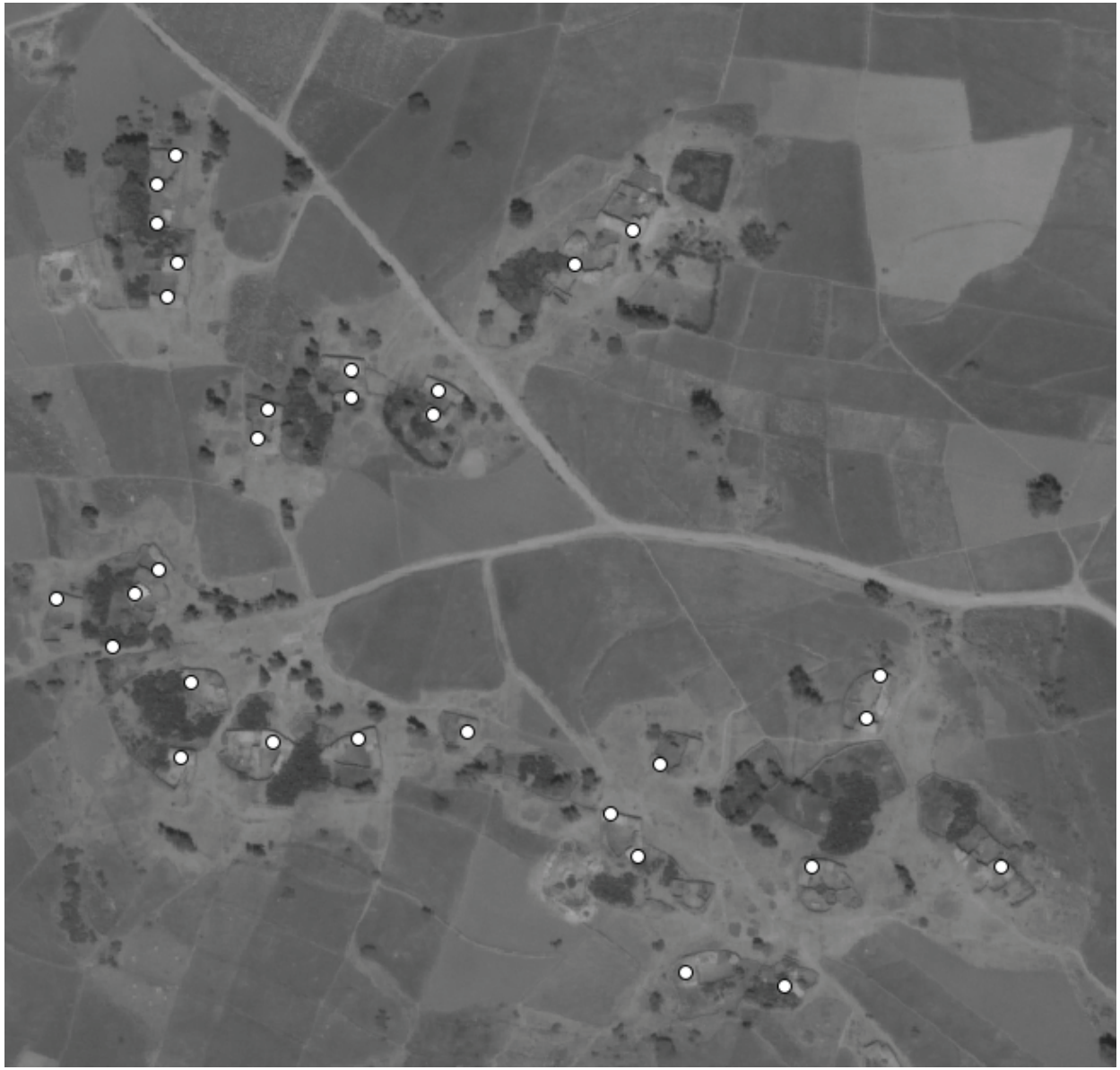
Farm 1 - Computer

# Results from Koraro in Ethiopia



Farm 1 - Computer & Human

# Results from Koraro in Ethiopia

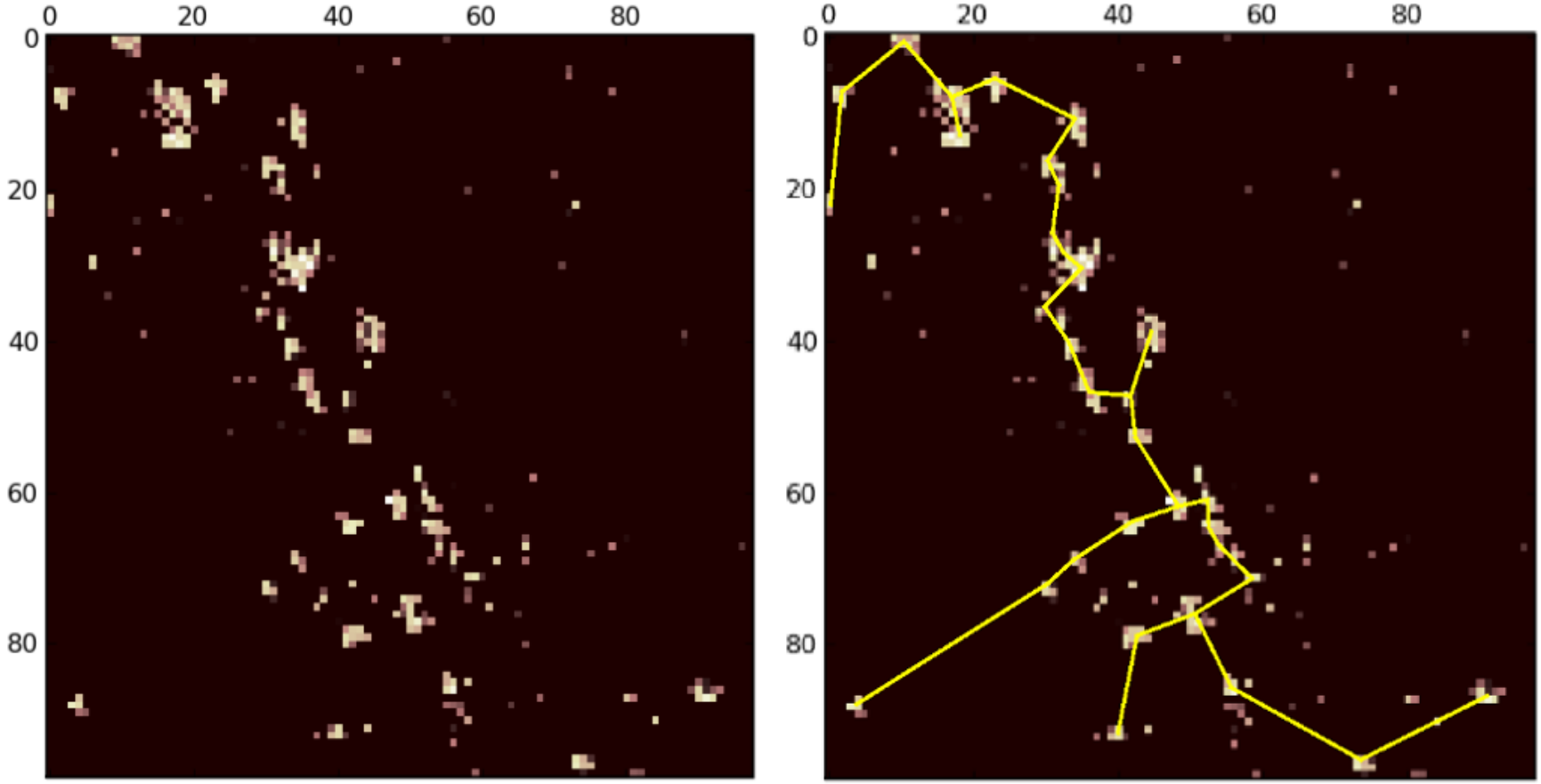


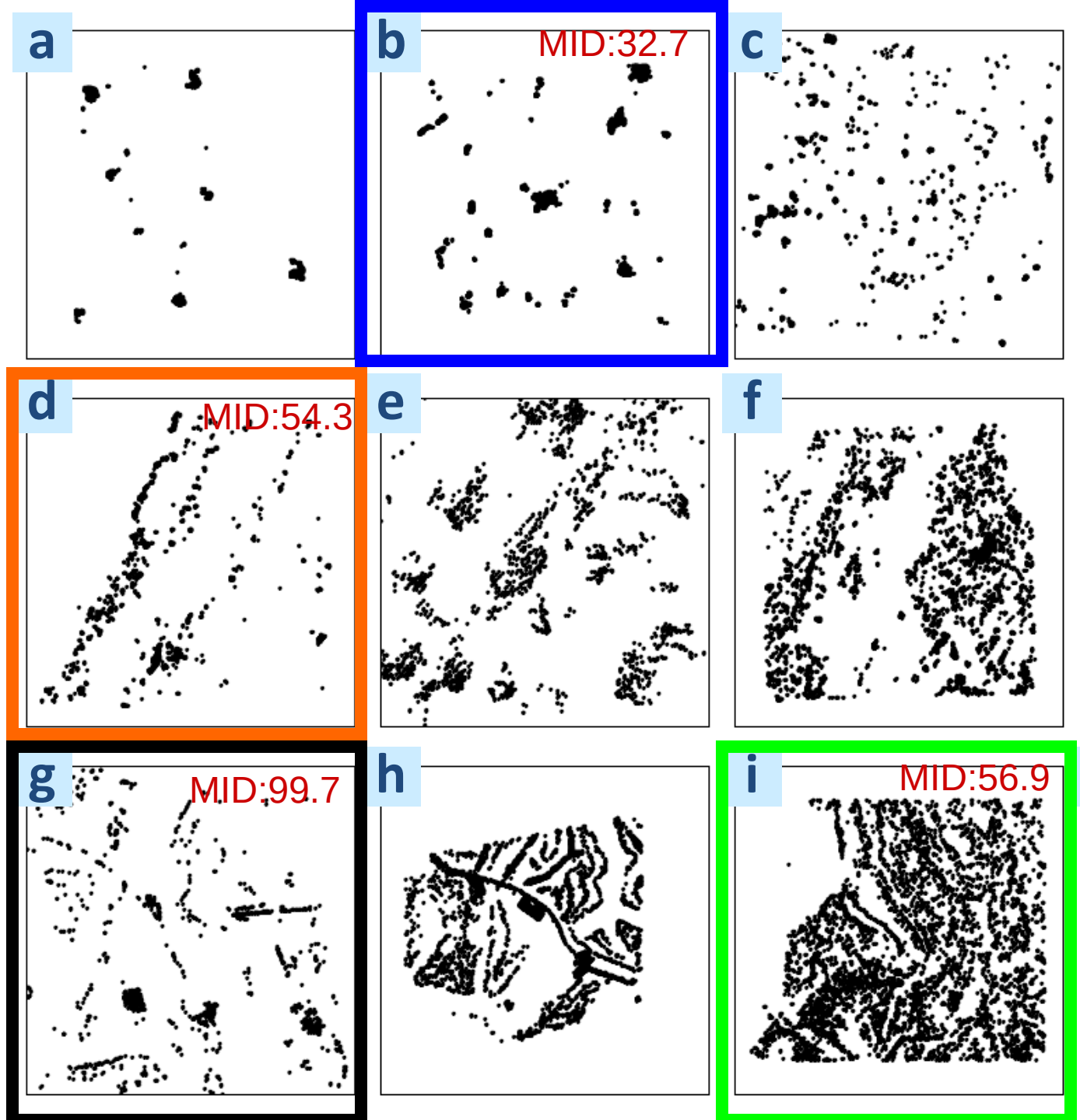
Farm 1 - Human

# Success Story #2: Network Modeling

| Question | Where should we build infrastructure? | | |
|---|---|---|---|
| Manual | Spreadsheets | Java | Desktop GIS |
| Method | Mathematical modeling | Geospatial optimization | Visualization |
| Command line | numpy<br>scipy | shapely \| geos | geojson \| openlayers<br>osgeo \| proj4 |
| Scalable web | Pylons \| SQLAlchemy | | ampqlib \|RabbitMQ |

# Computational core: Network modeling





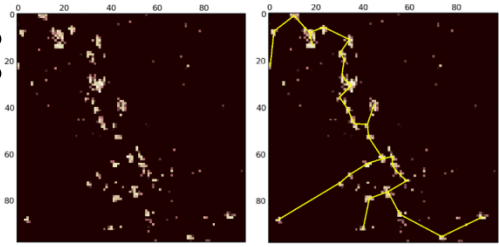

a) Bonsaaso, **GHANA**
b) Tiby, **MALI**
c) Pampaida, **NIGERIA**
d) Potou, **SENEGAL**
e) Koraro, **ETHIOPIA**
f) Mwandama, **MALAWI**
g) Mbola, **TANZANIA**
h) Mayange, **RWANDA**
i) Ruhiira, **UGANDA**
j) Sauri, **KENYA**

Zvoleff, Kocaman, Huh, Modi (2009)

0 2 4 6 8 10 Km

1 cm = 2 km

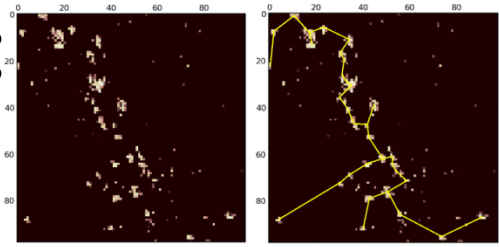
# Computational core: Network modeling



For each node,

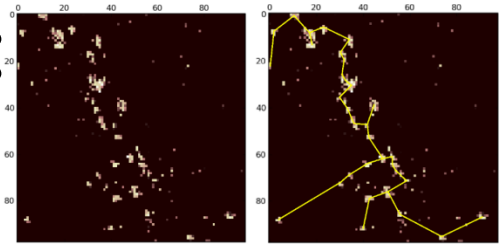
- Project demographic growth
- Project electricity demand
- Estimate construction/maintenance cost
- Select electricity system via geospatial info


# Computational core: Network modeling



| | |
|---|---|
| Estimate metrics for each node | `import scipy.stats` |
| Find distance between lines<br>Project points onto lines<br>Find intersections between lines<br>Merge and simplify lines | `import shapely` |
| Store nodes and segments efficiently | `import sqlalchemy` |

# Computational core: Network modeling



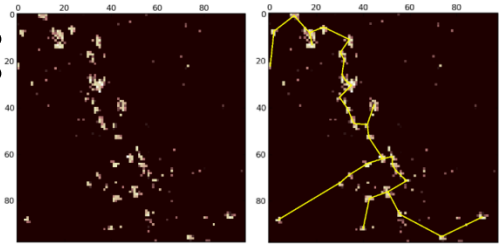
**Find distance between lines**
Project points onto lines
Find intersections between lines
Merge and simplify lines

```
import shapely
```

```
import shapely.geometry as g
line1 = g.LineString([(0,0), (1,0)])
line2 = g.LineString([(0,1), (1,1)])

line1.distance(line2)
```

# Computational core: Network modeling




Find distance between lines

**Project points onto lines**

Find intersections between lines

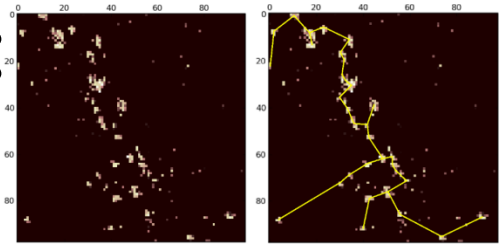
Merge and simplify lines

```
import shapely
```

```
import shapely.geometry as g
line = g.LineString([(0,0), (1,0)])

line.interpolate(line.project(g.Point(0.5, 2)))
```

# Computational core: Network modeling



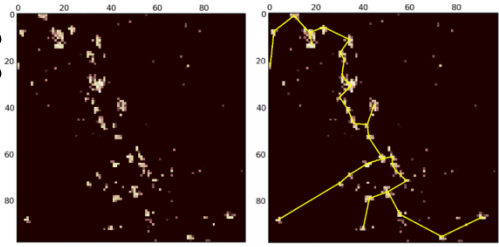
Find distance between lines
Project points onto lines
**Find intersections between lines**
Merge and simplify lines

```
import shapely
```

```
import shapely.geometry as g
line1 = g.LineString([(0,0), (1,0)])
line3 = g.LineString([(1,0), (1,1)])

line1.intersection(line3)
```

# Computational core: Network modeling

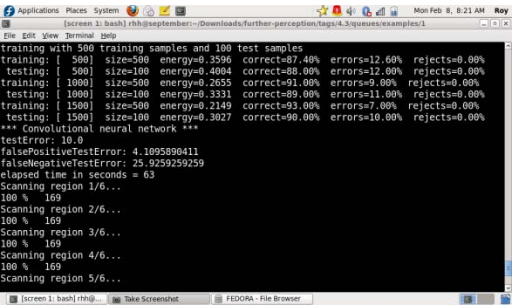


Find distance between lines
Project points onto lines
Find intersections between lines
**Merge and simplify lines**

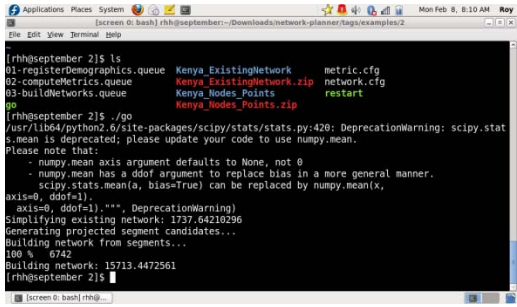
```
import shapely
```

```
import shapely.geometry as g
import shapely.ops
line1 = g.LineString([(0,0), (1,0)])
line4 = g.LineString([(1,0), (2,0)])

shapely.ops.linemerge(line1.union(line4)).simplify(0)
```



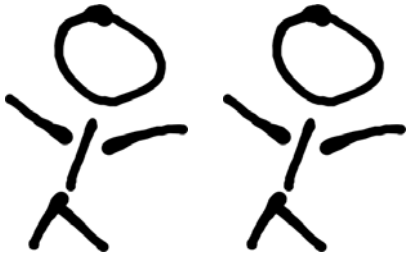

# Command-line



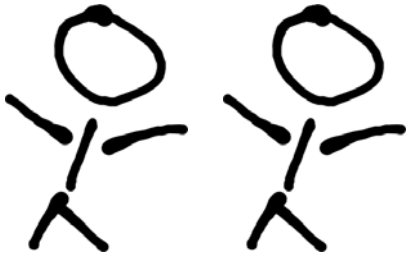

| | |
|---|---|
| Parse arguments and options | `import optparse` |
| Load and save configuration files | `import ConfigParser` |
| Compress and uncompress data | `import ZipFile` |
| Get script path | `os.path.abspath(__file__)` |

# Web service

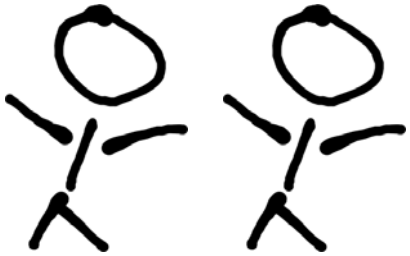
| | |
|---|---|
| Build web service | `import pylons` |
| Generate RESTful interface | `$ paster restcontroller` |
| Render map with OpenLayers | `import osgeo.osr (PROJ.4)`<br>`import geojson` |
| Serialize job in database | `import cPickle` |
| Process jobs | `$ crontab` |



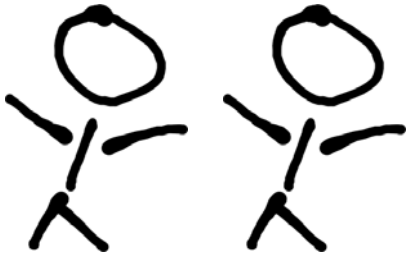
# Scalable web service

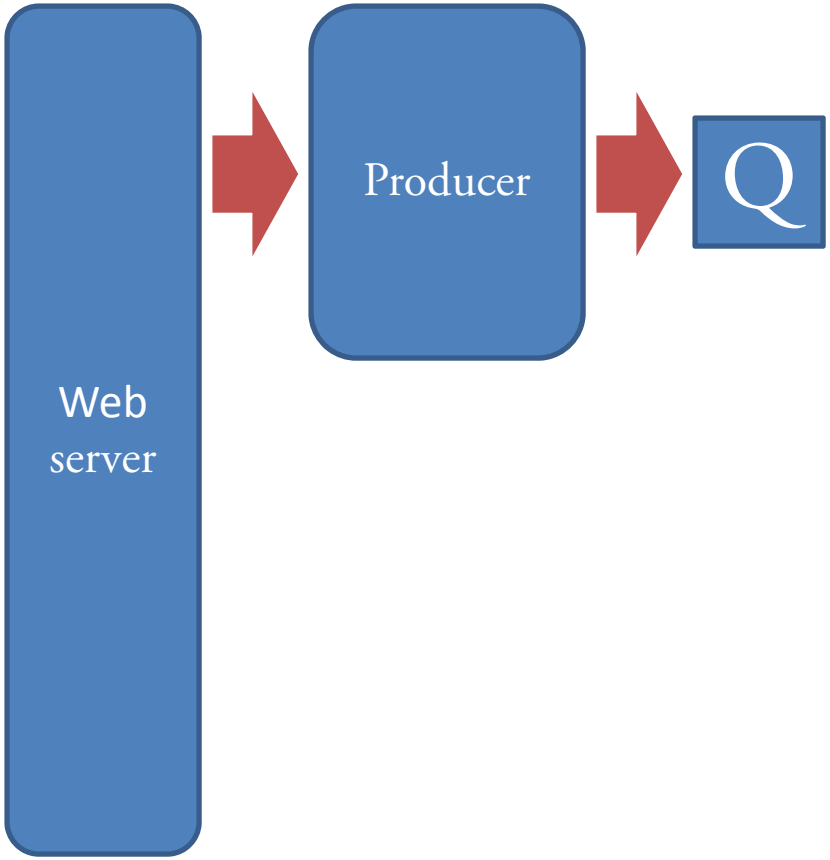


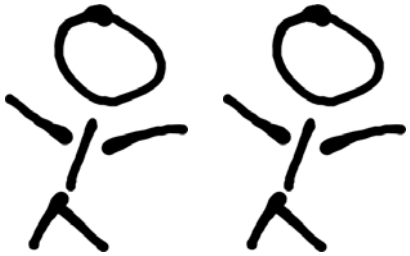
| | |
|---|---|
| Serve queues | `$ RabbitMQ` |
| Put message on queue<br>Get message from queue | `import amqplib` |



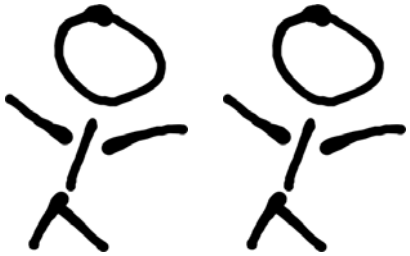
# Scalable web service

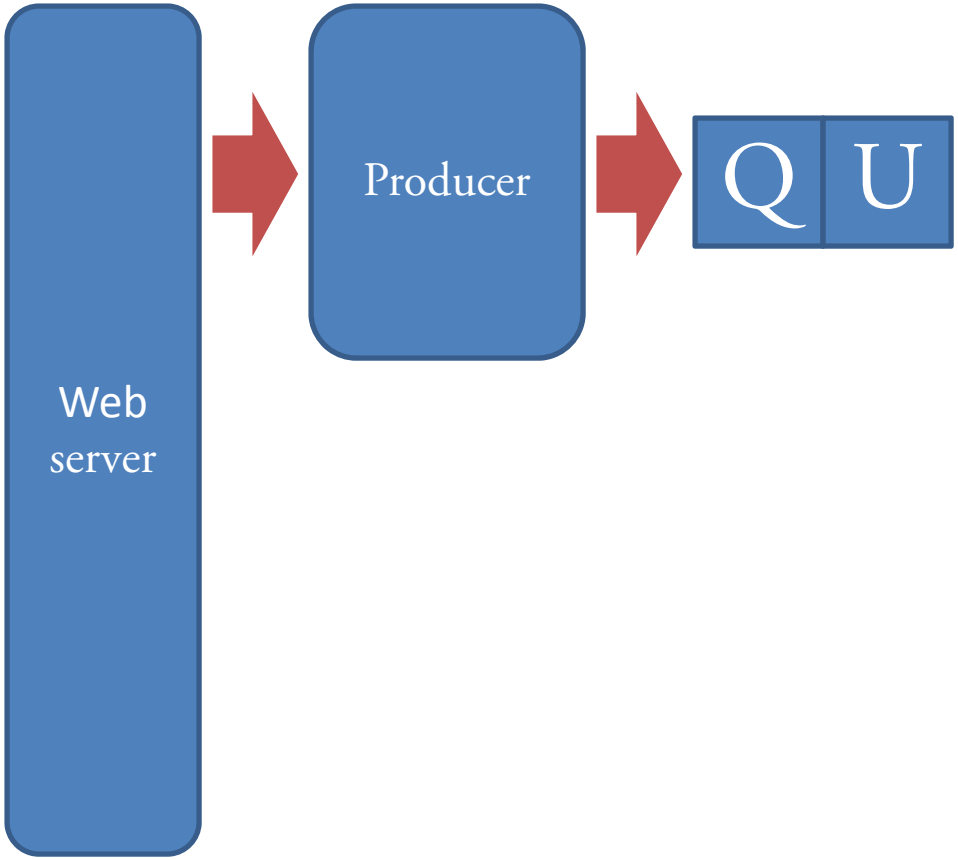


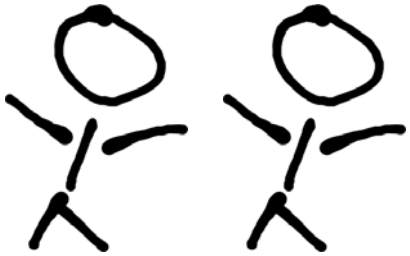




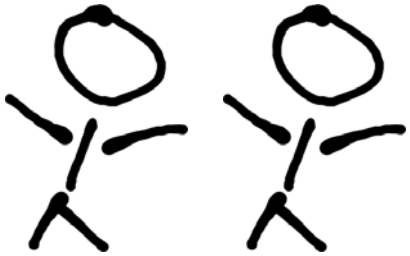
# Scalable web service

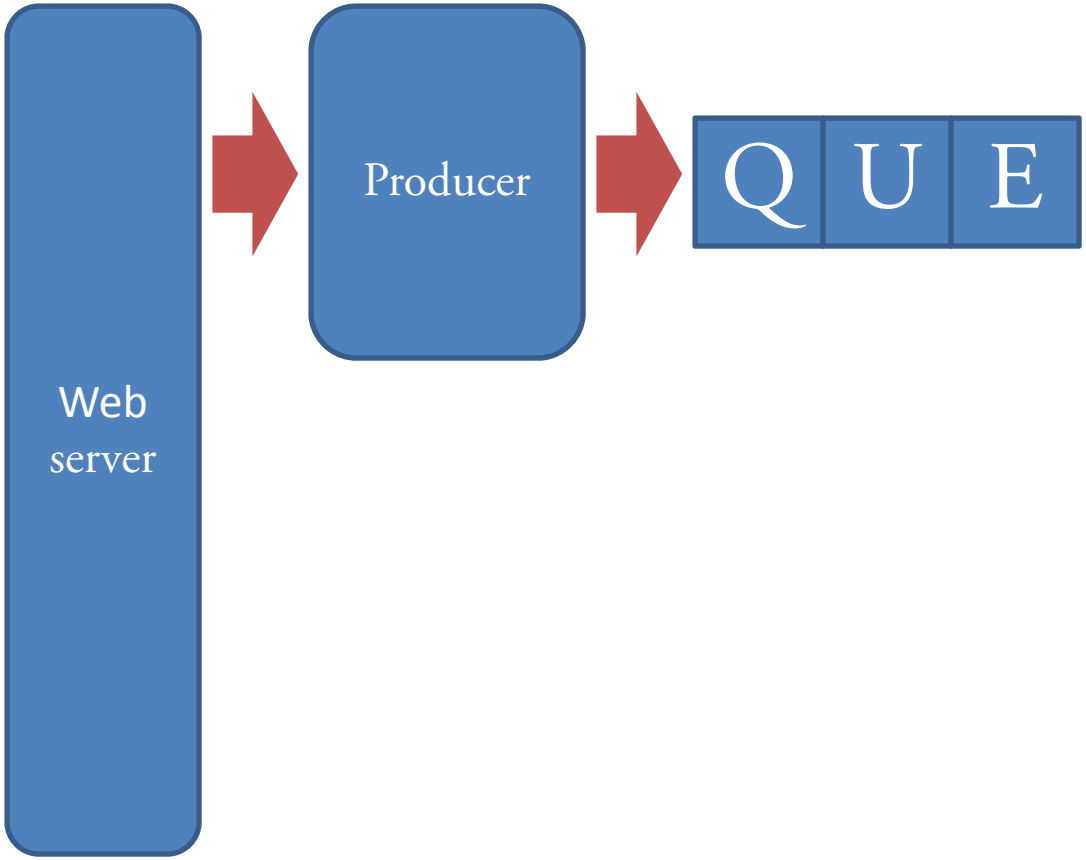


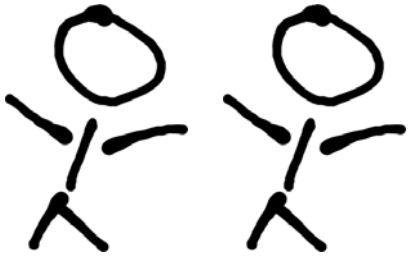




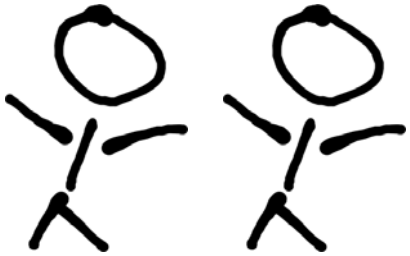
# Scalable web service

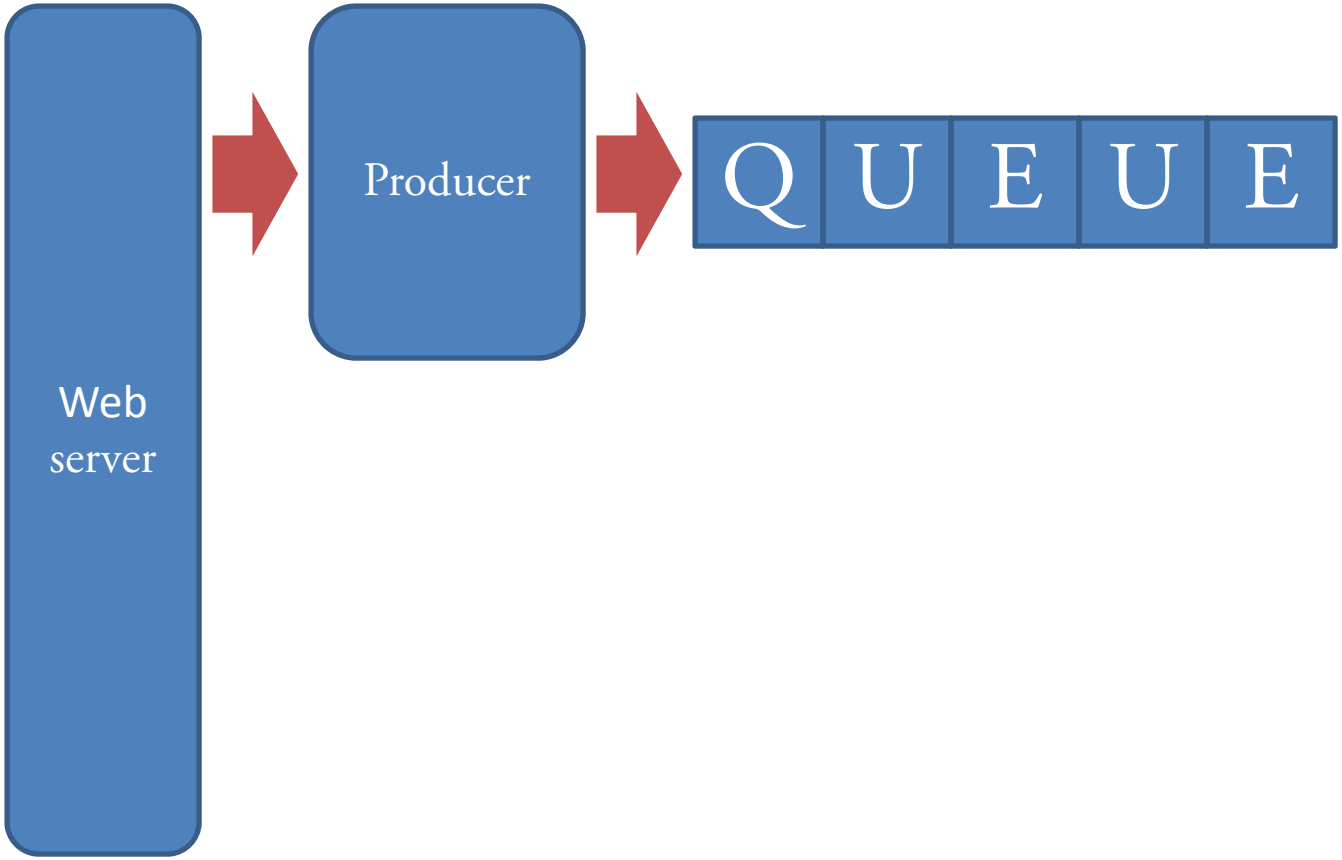


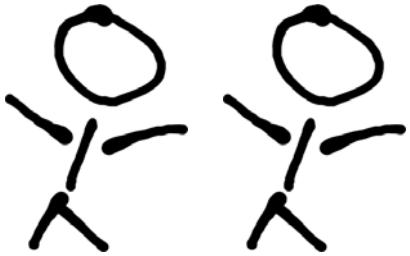




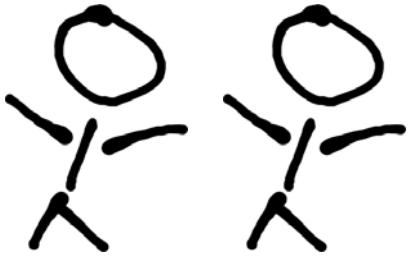
# Scalable web service

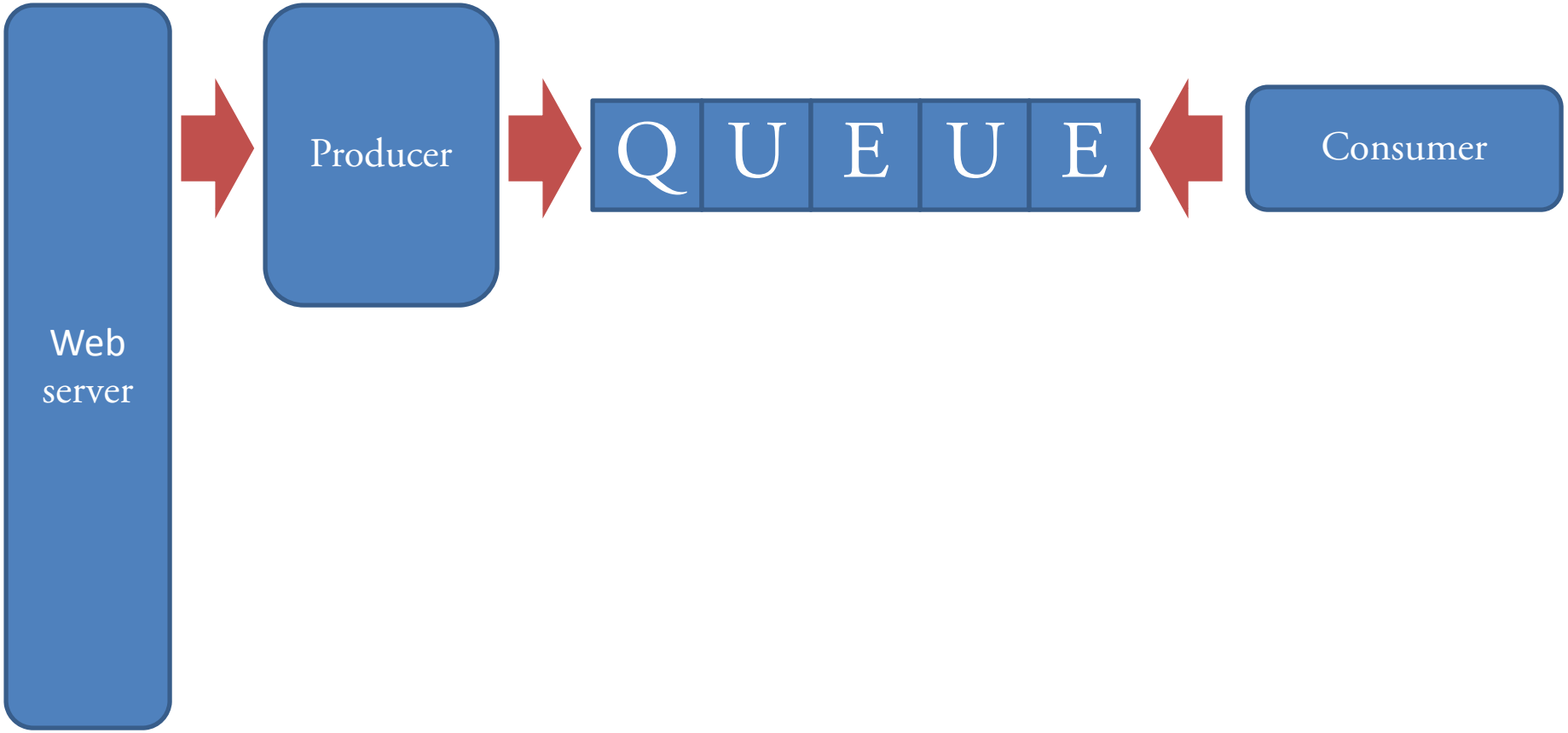


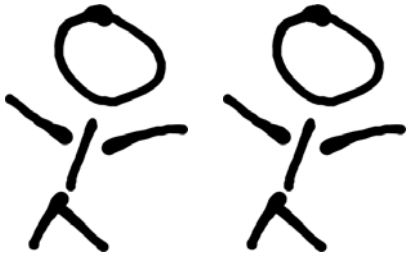




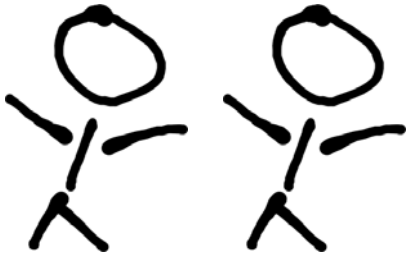
# Scalable web service

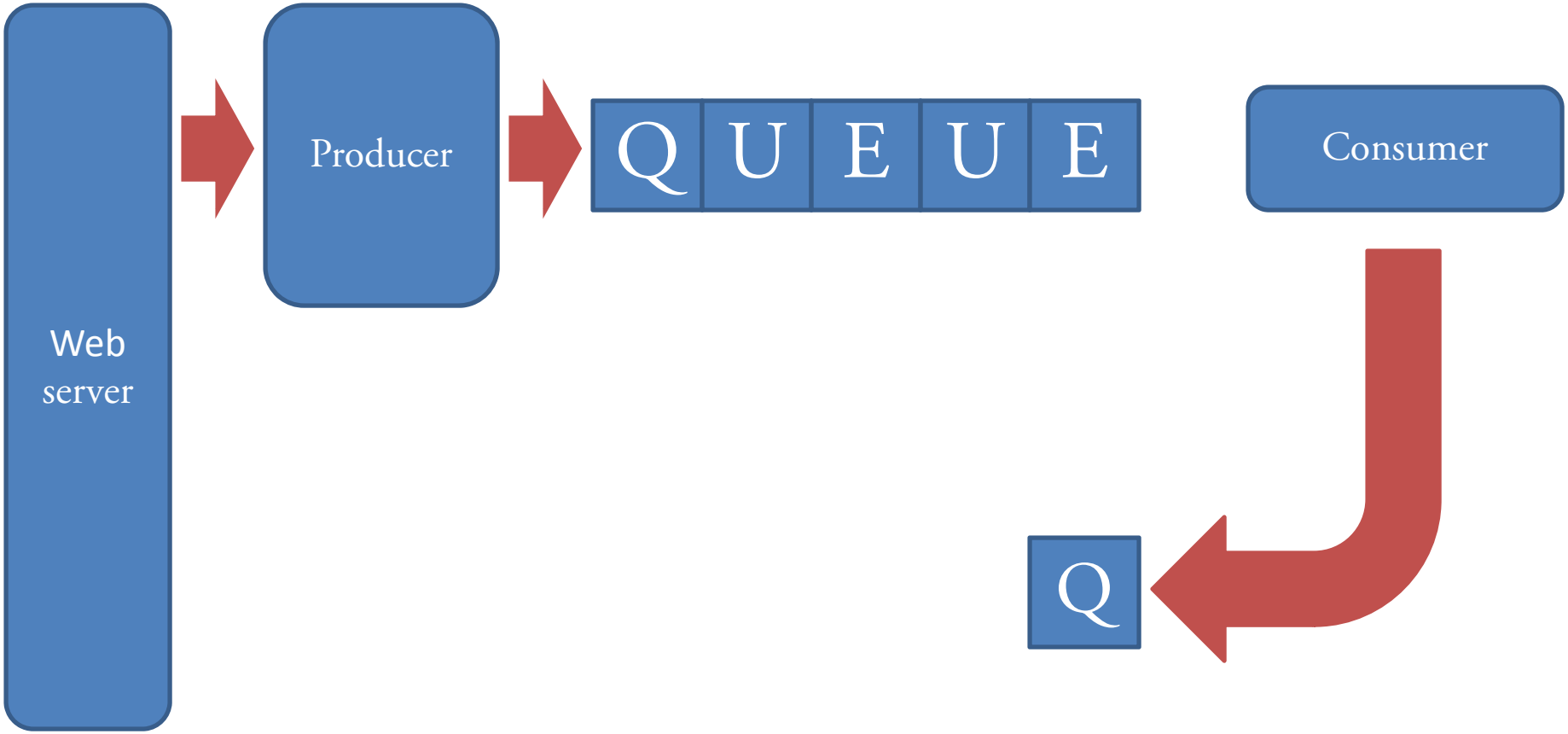


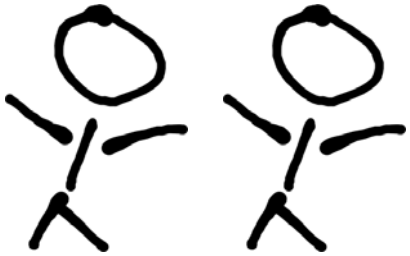




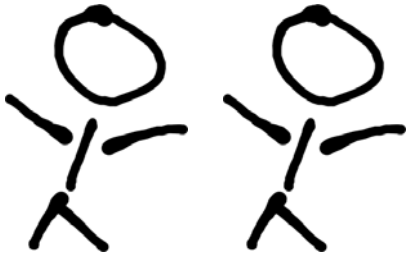
# Scalable web service

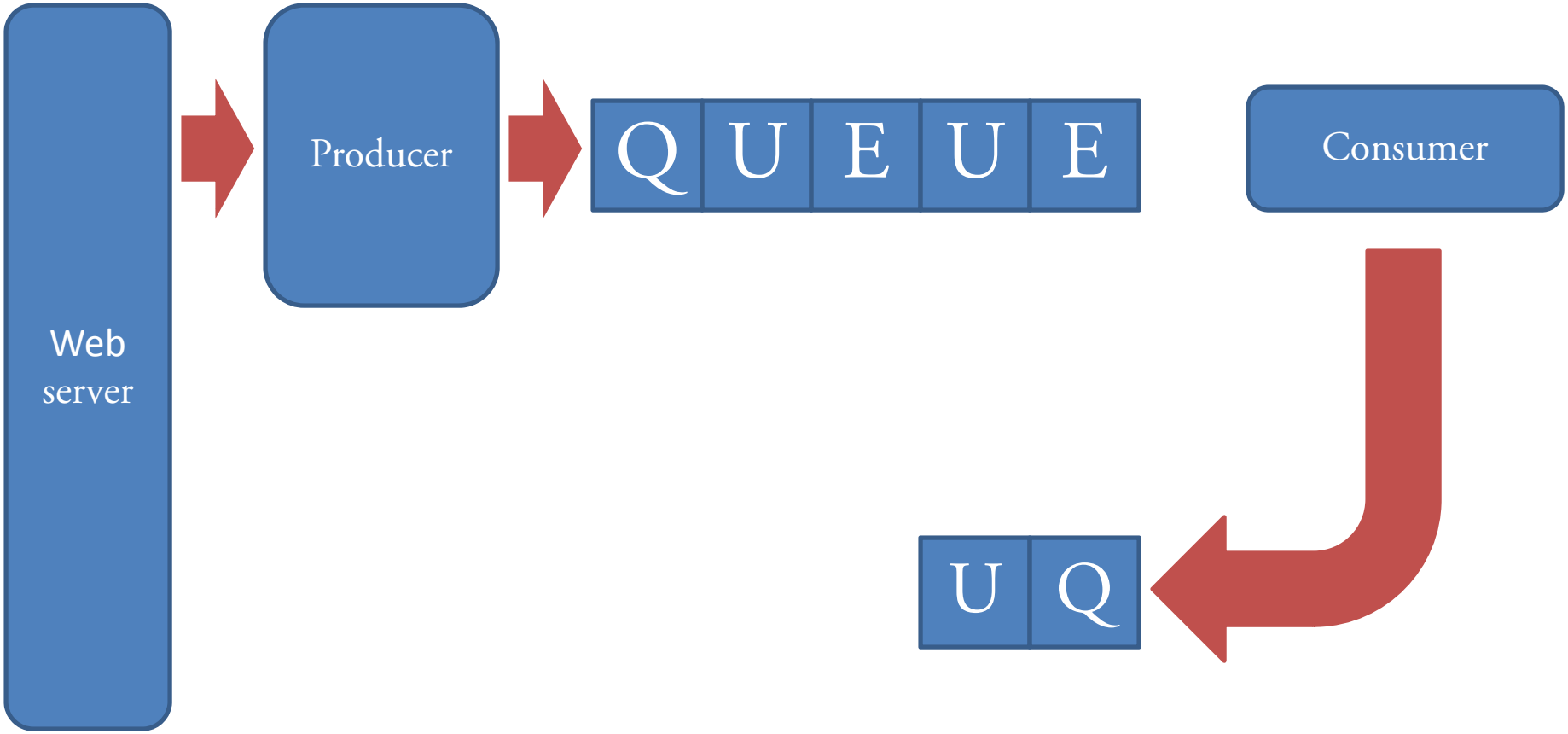


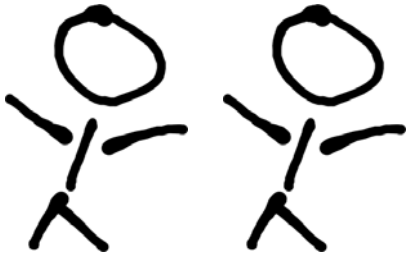




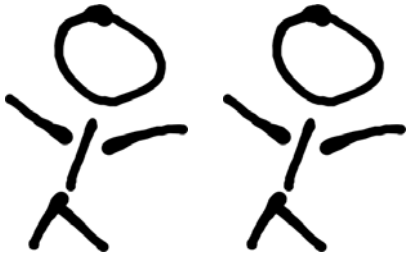
# Scalable web service

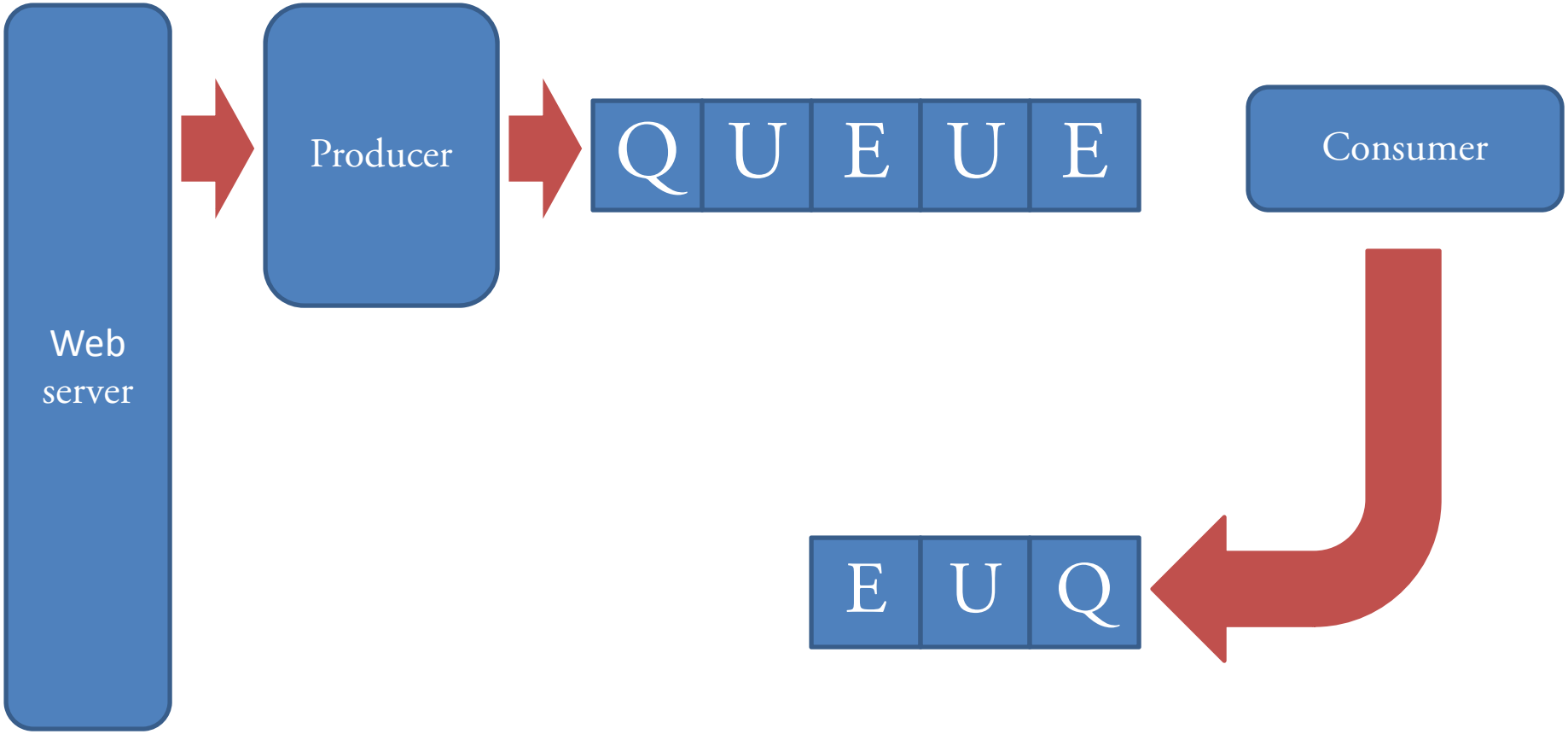


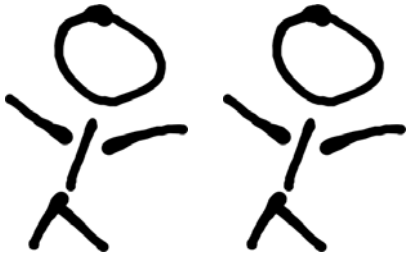




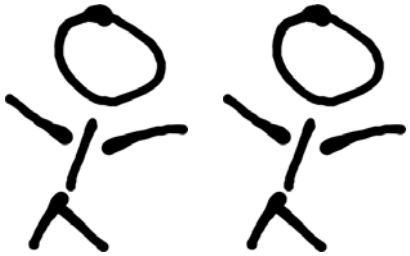
# Scalable web service

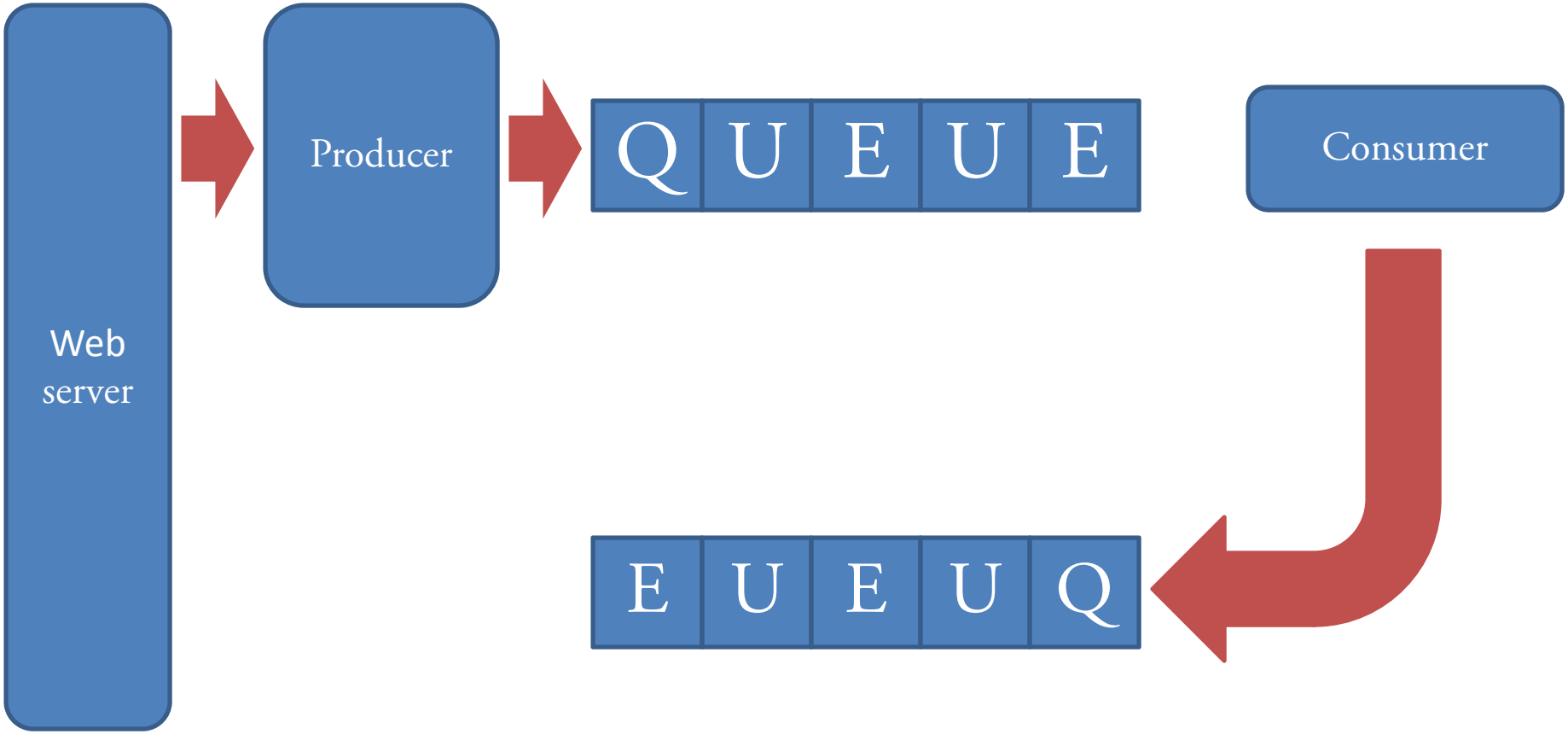


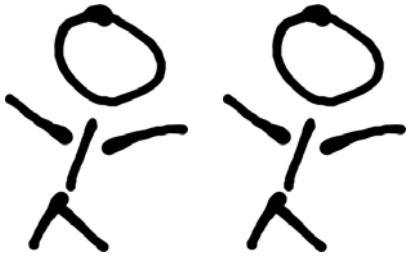




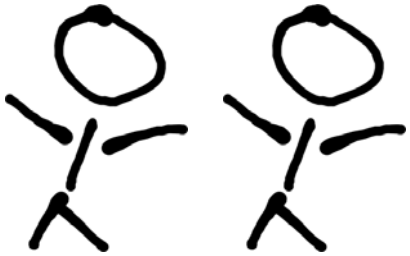
# Scalable web service

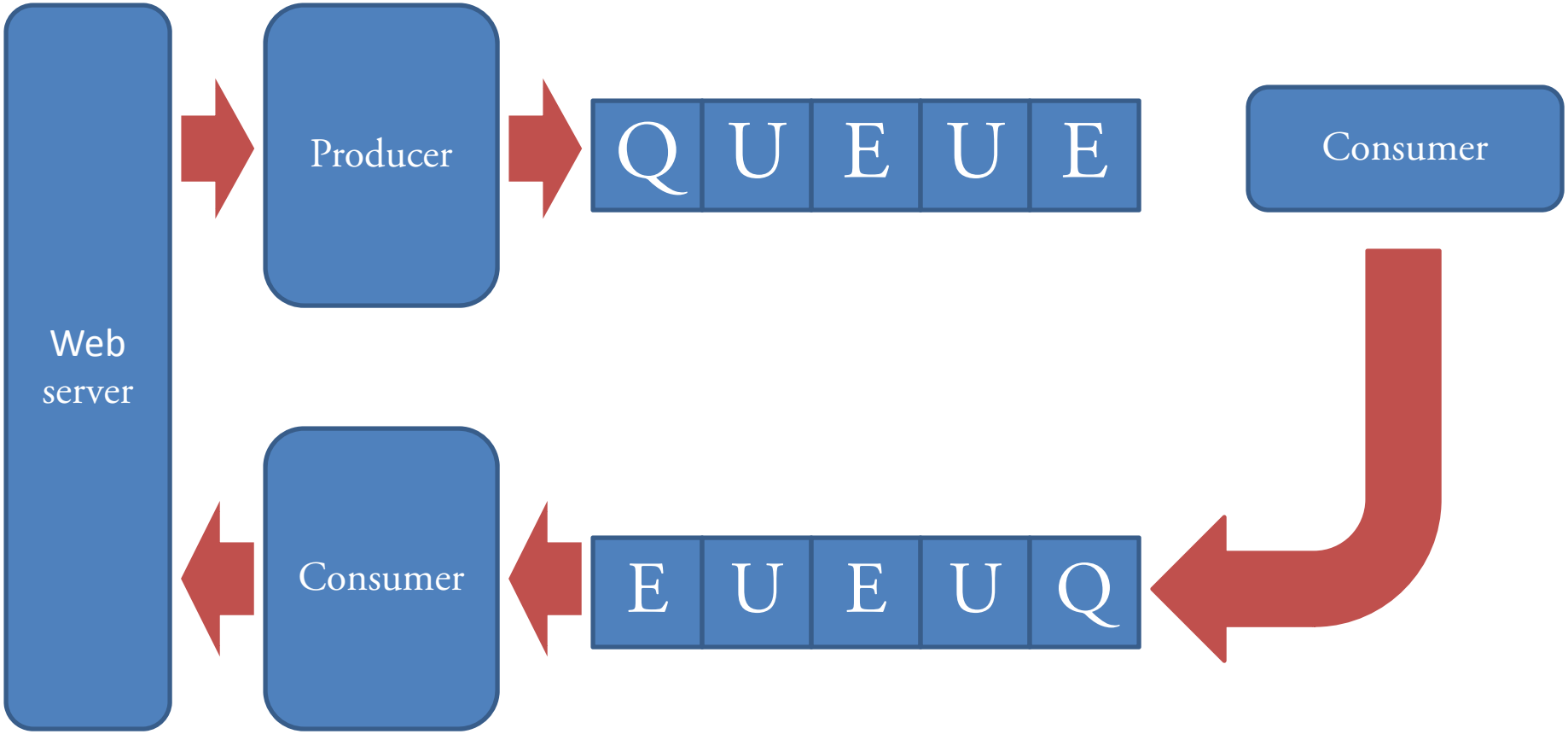


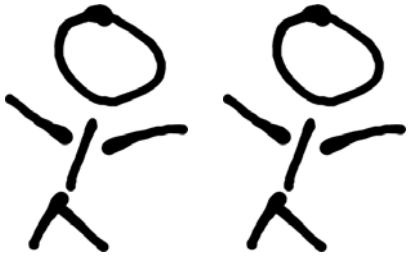




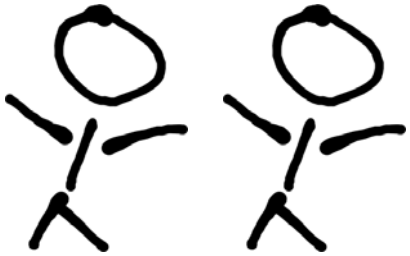
# Scalable web service

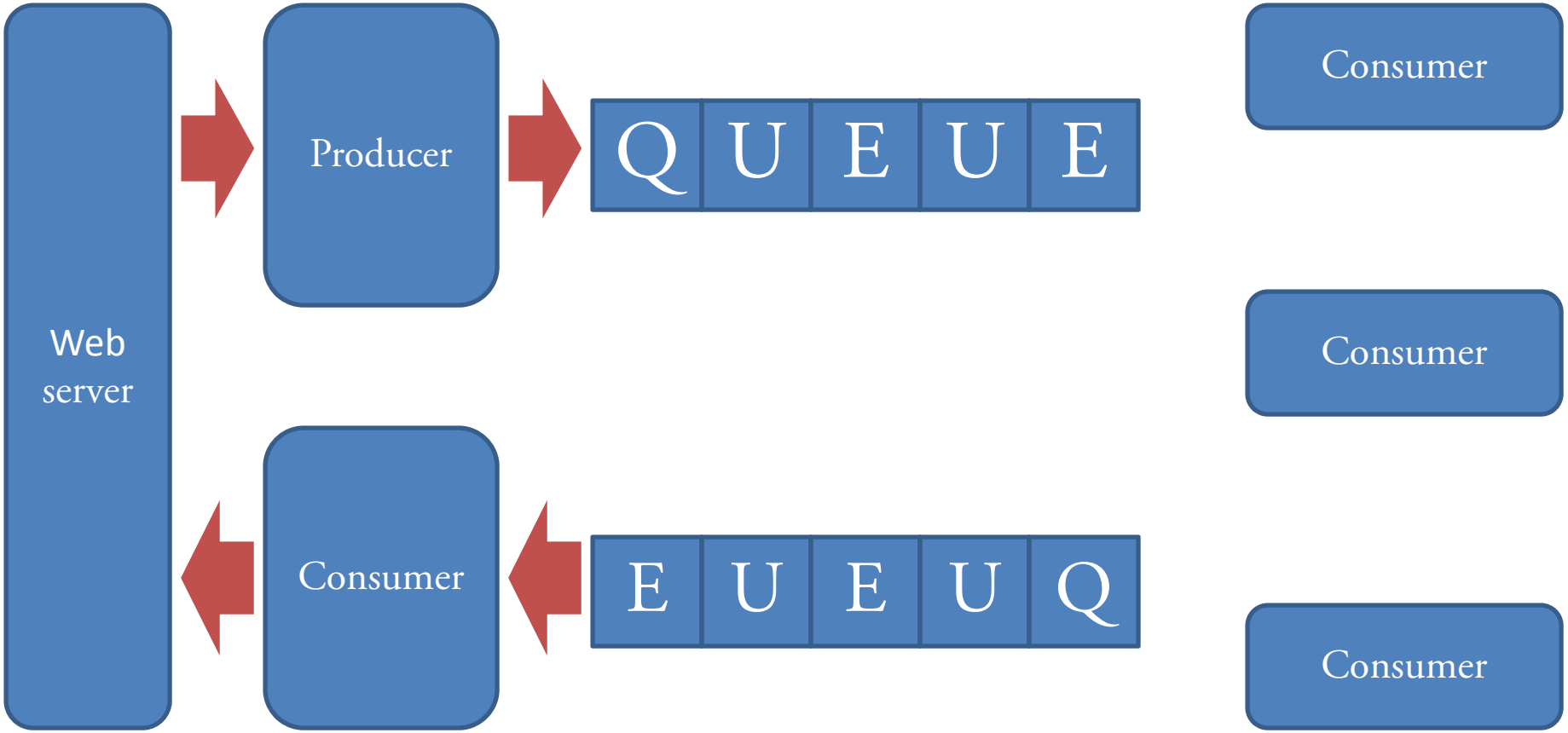


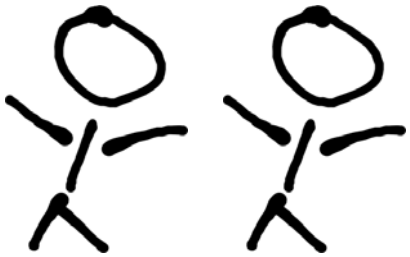




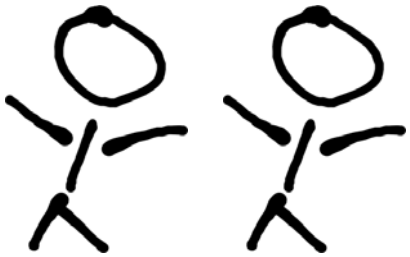
# Scalable web service







Scalable web service



Put message on queue

Get message from queue

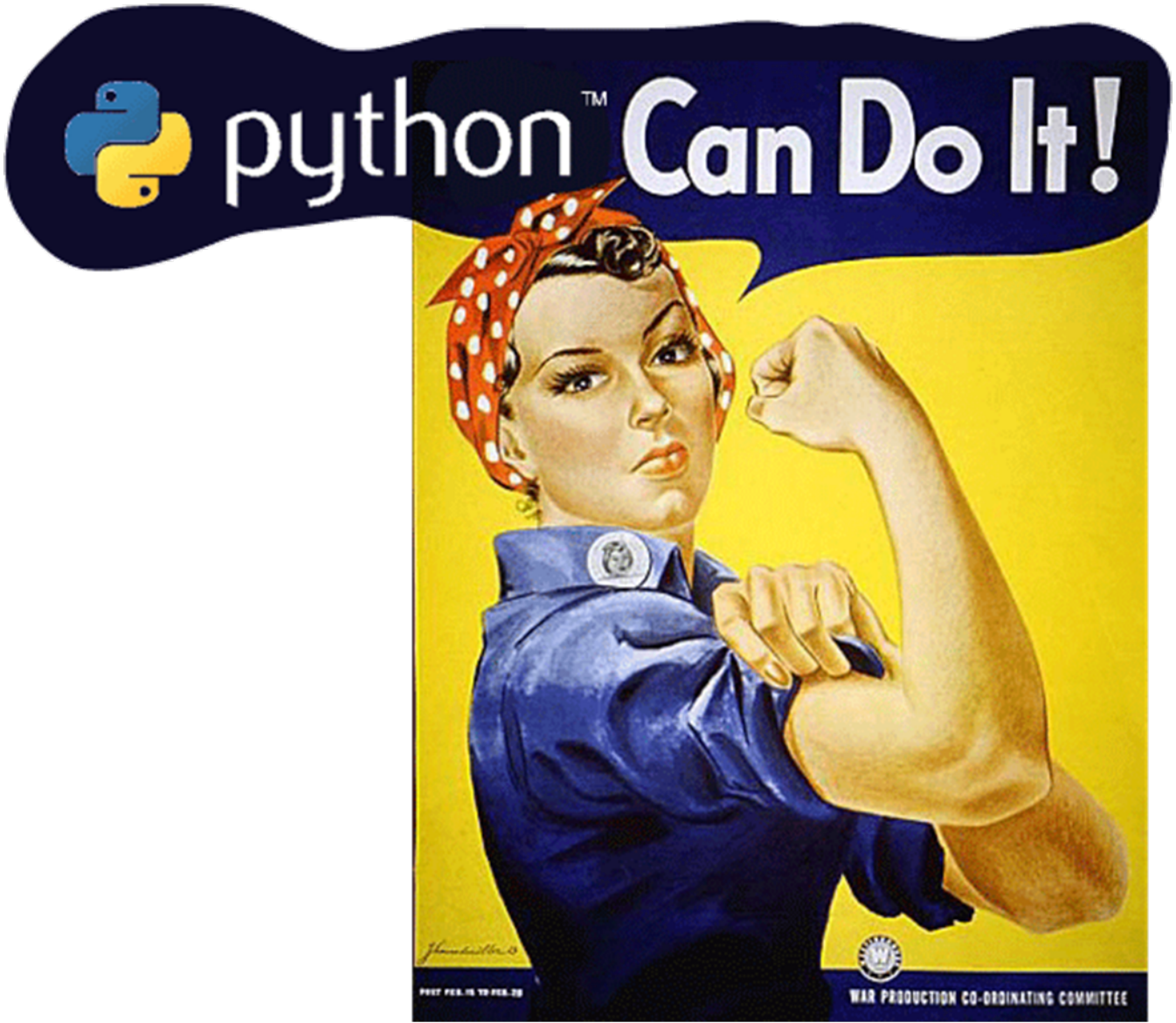
Get data from storage

# Tip from AMQP

Don't tell people what to do

**Train people how to do it**

# Take-away message



# Take-away message




| | | |
|---|---|---|
| Question | Where do people live? | |
| Manual | Send people with GPS devices<br>Click on houses in satellite images | |
| Method | Machine learning | Image recognition |
| Command line | subprocess \| Lush | osgeo \| GDAL Generators |
| Scalable web | Pylons \| SQLAlchemy | ampqlib \|RabbitMQ |

# Take-away message


| | | | |
|---|---|---|---|
| Question | Where should we build infrastructure? | | |
| Manual | Spreadsheets | Java | Desktop GIS |
| Method | Mathematical modeling | Geospatial optimization | Visualization |
| Command line | numpy<br>scipy | shapely \| geos | geojson \| openlayers<br>osgeo \| proj4 |
| Scalable web | Pylons \| SQLAlchemy | ampqlib \|RabbitMQ | |

# Credits

Susan Kum
Ayse Selin Kocaman
Po-Han “Freeza” Huang
Andy Doro
Alex Hofmann
Sahil Shah
Alex Zvoleff
Anders Pearson
Matt Berg

Vijay Modi
Edwin Adkins
Dana Pillai
Aly Sanoh
Lily Parshall
Yann LeCun
Marc’Aurelio Ranzato
Jiehua Chen
Brett Gleitsmann

# Tutorials

| | |
|---|---|
| Face-to-face | PyCon OpenSpaces |
| Read tutorials<br>Request a topic | invisibleroads.com |

# Links

| | |
|---|---|
| Modi Research Group<br>Earth Insitute<br>Columbia University | modi.buildafrica.org |
| Computational and Biological Learning Lab<br>Courant Insitute of Mathematical Sciences<br>New York University | cs.nyu.edu/~yann |
| Tutorials & Workshops | invisibleroads.com |