

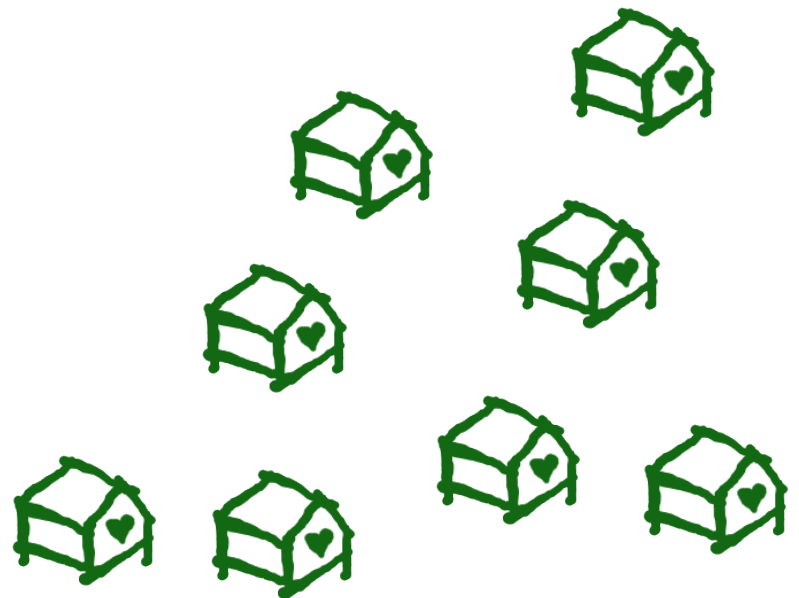
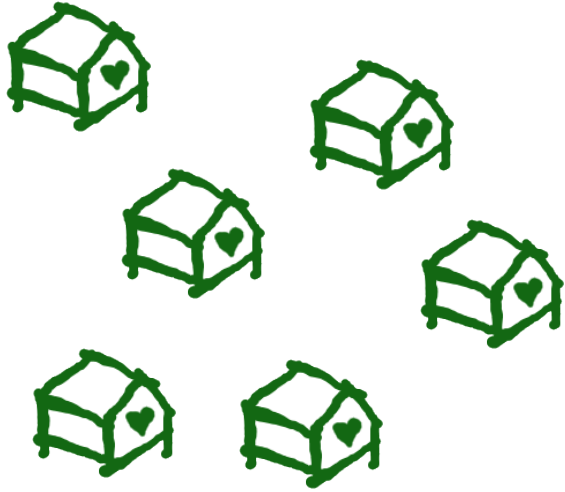
How python™ is guiding infrastructure construction in Africa

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

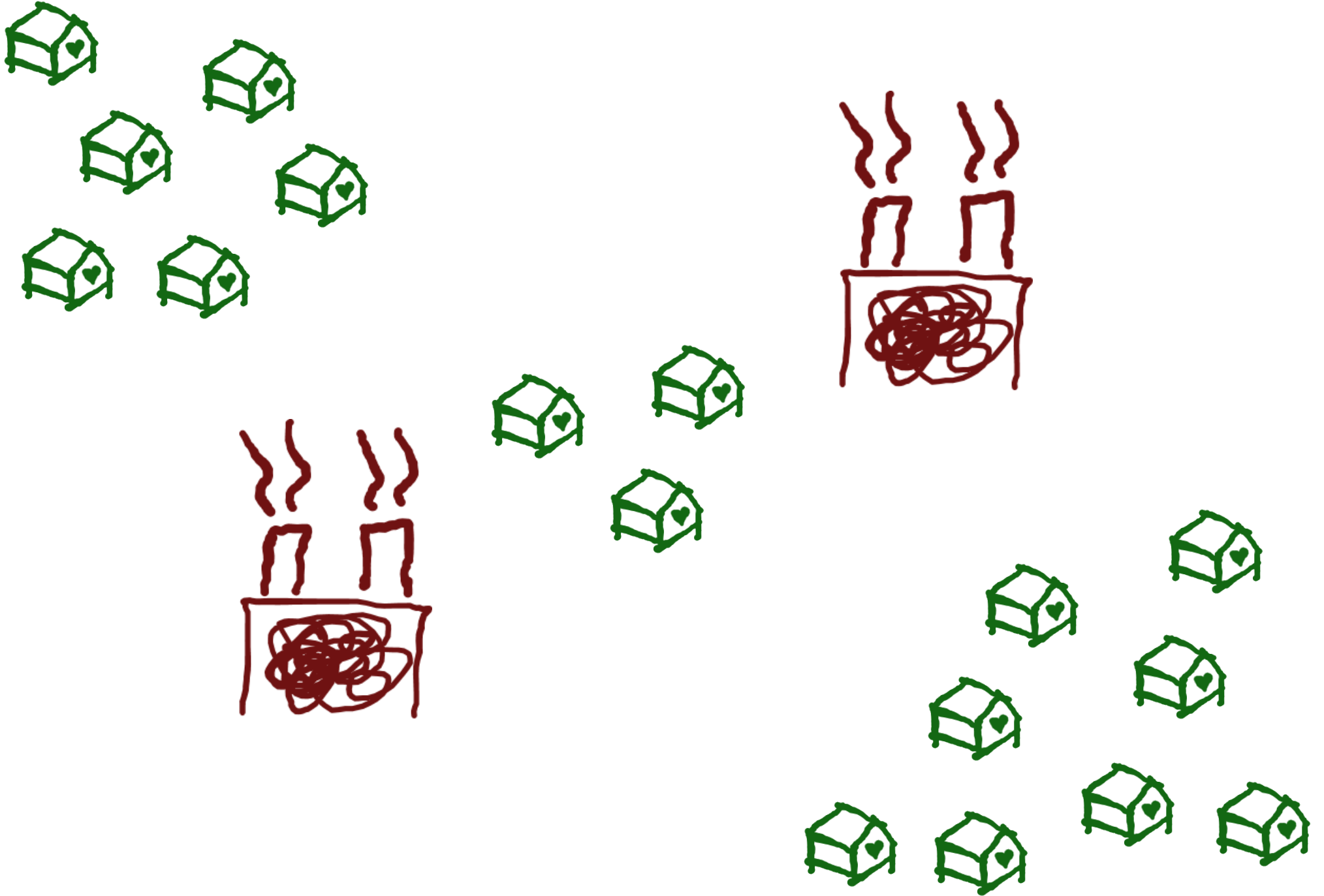


2010 PyCon Atlanta

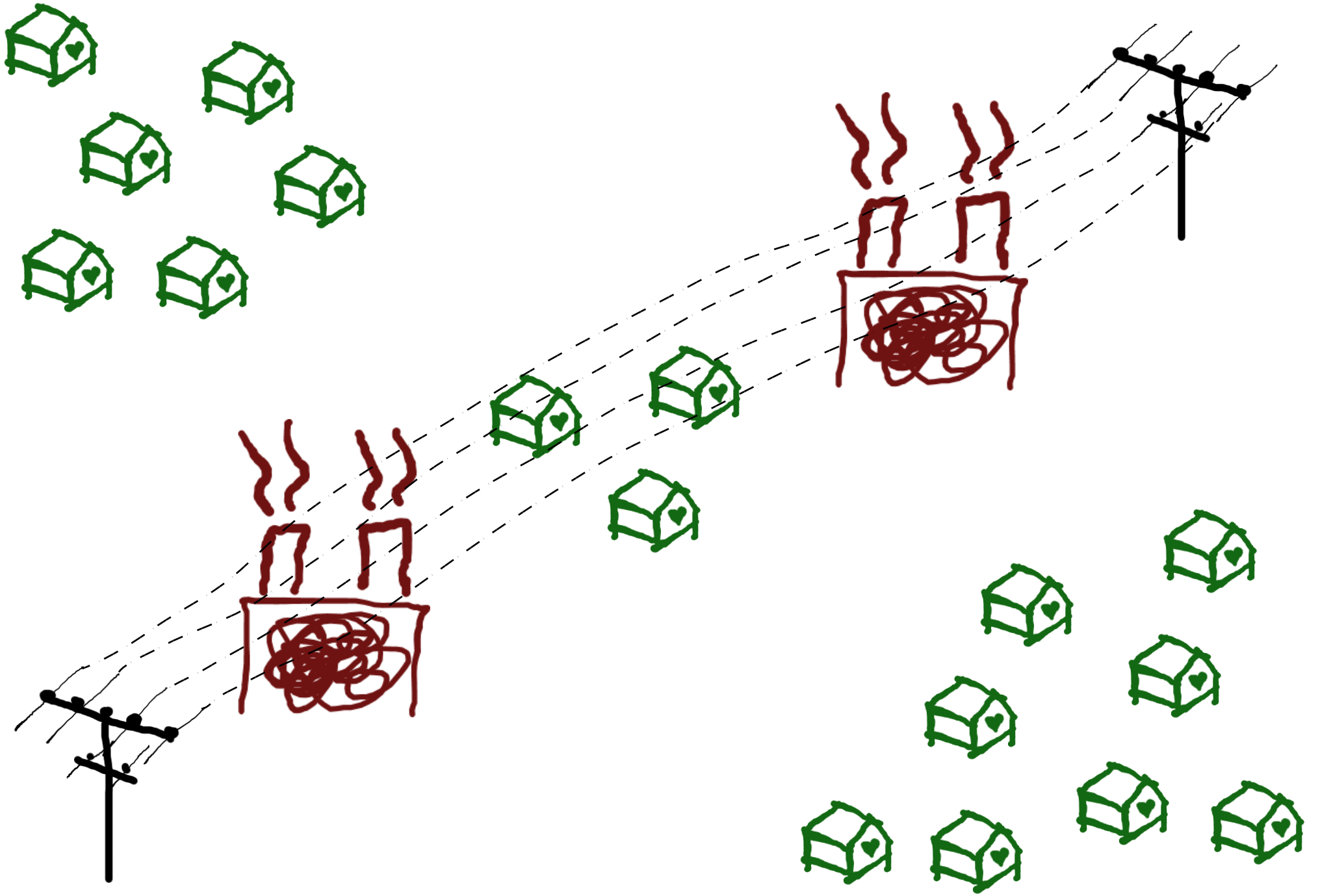
The cost of poor planning



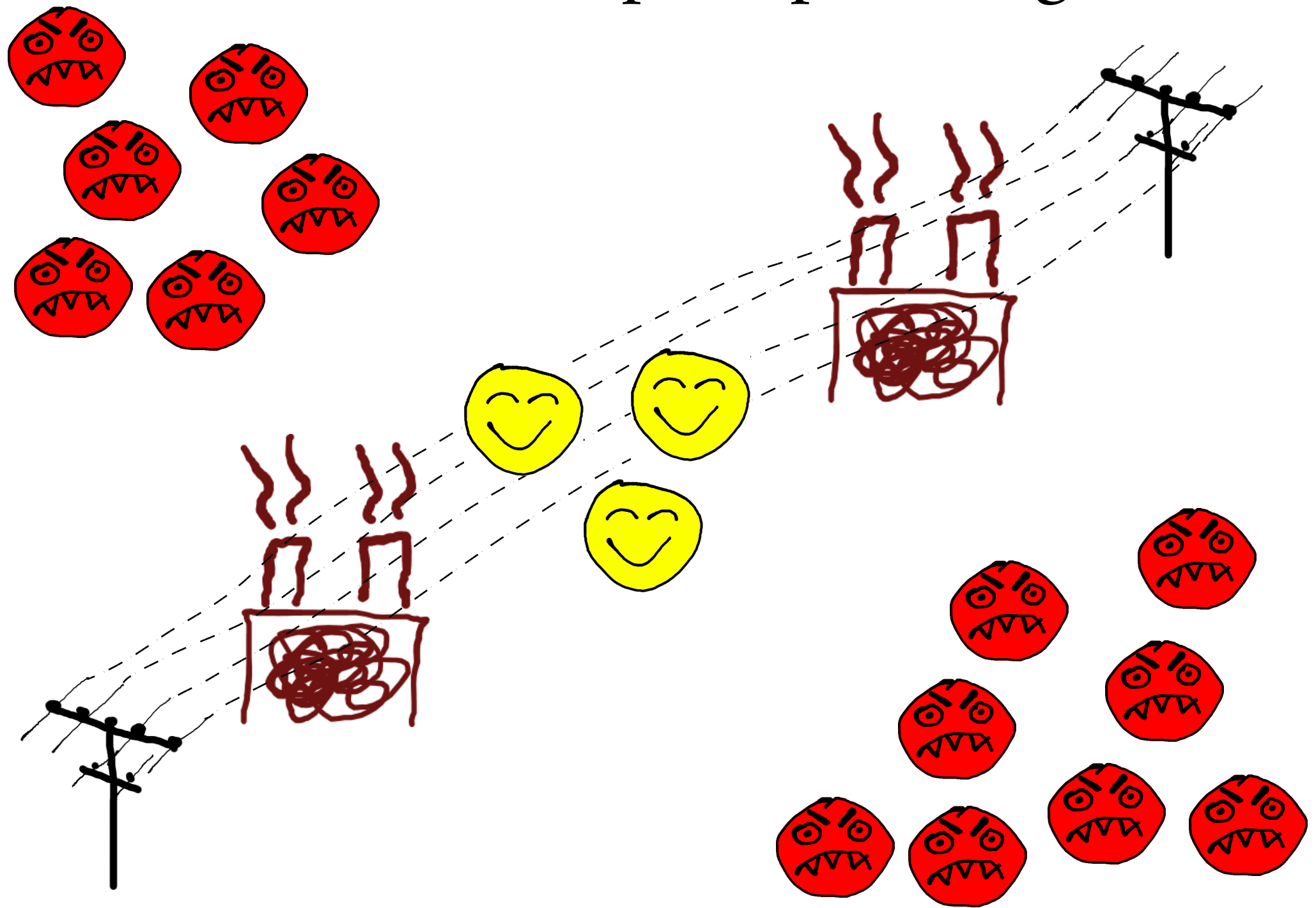
The cost of poor planning



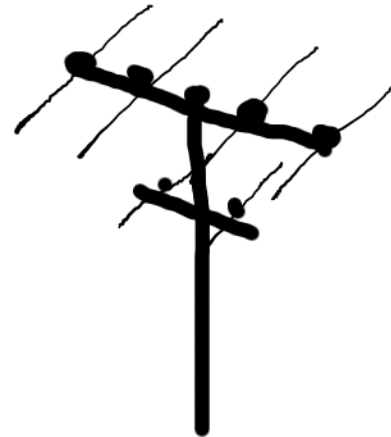
The cost of poor planning



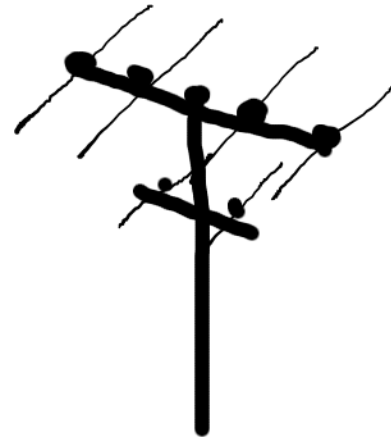
The cost of poor planning



Give people electricity access



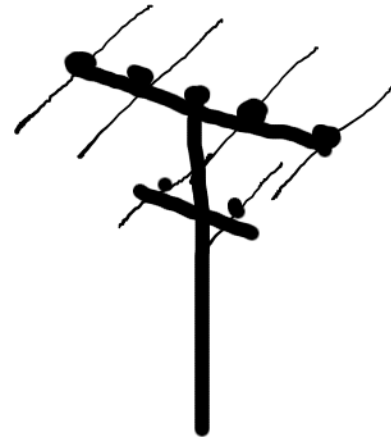
Give people electricity access





Give people electricity access

Country goes to bank

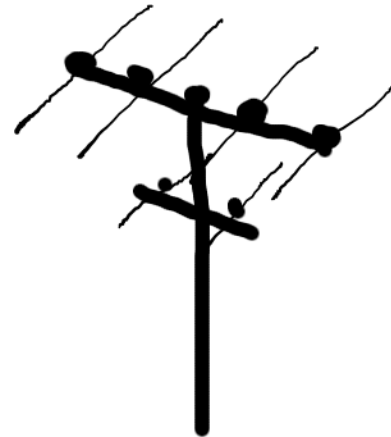




Give people electricity access

Country goes to bank

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



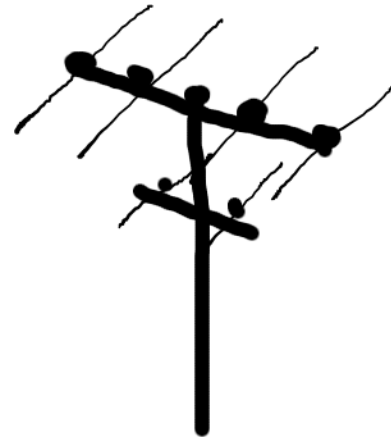


Give people electricity access

Country goes to bank

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

Consultants draft plan





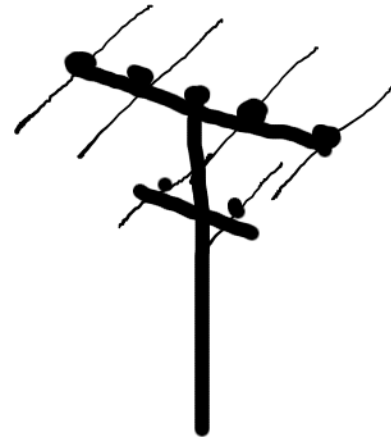
Give people electricity access

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

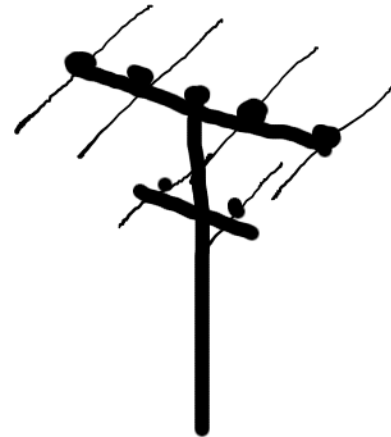
Country goes to bank

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

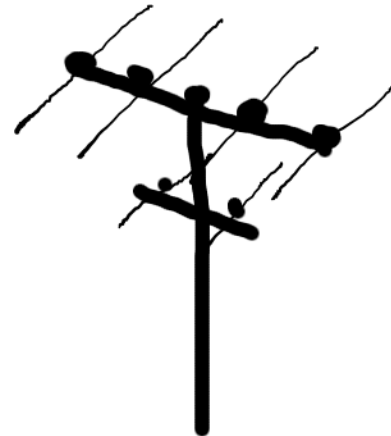
Bank recruits consultants, who talk to ministers, utility owners, local leaders

Consultants draft plan

Bank sends plan to financiers

Financiers fund construction

People get electricity



A credit-worthy infrastructure plan

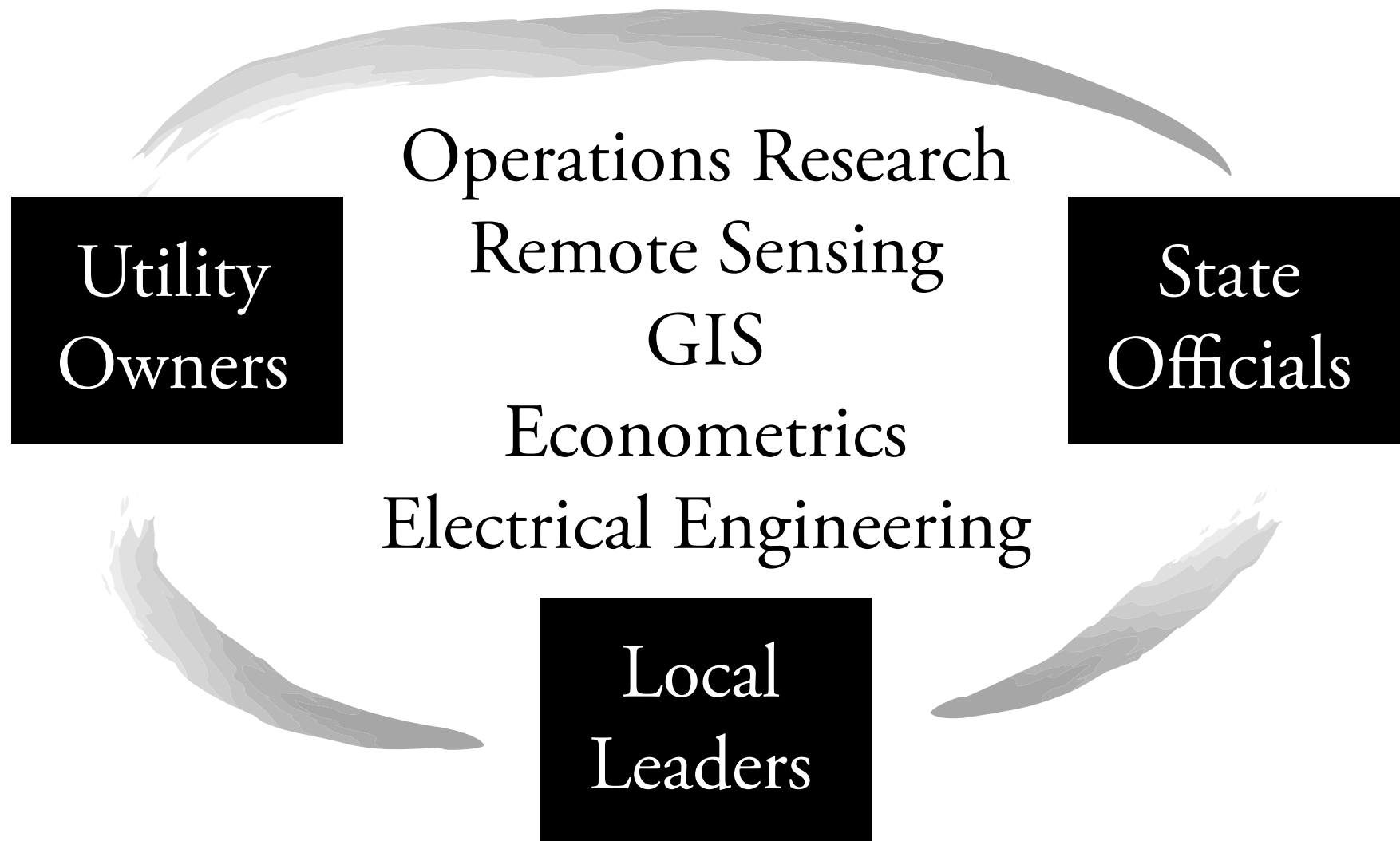
IS GROUNDED IN

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?

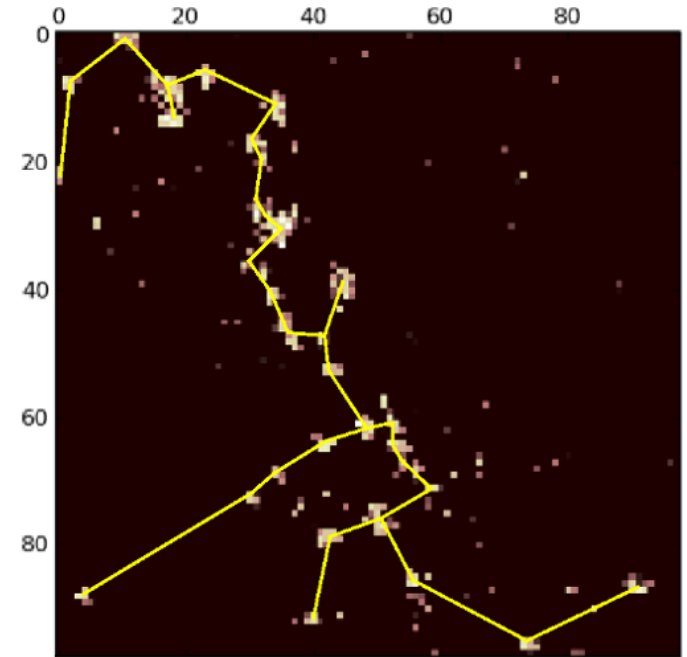
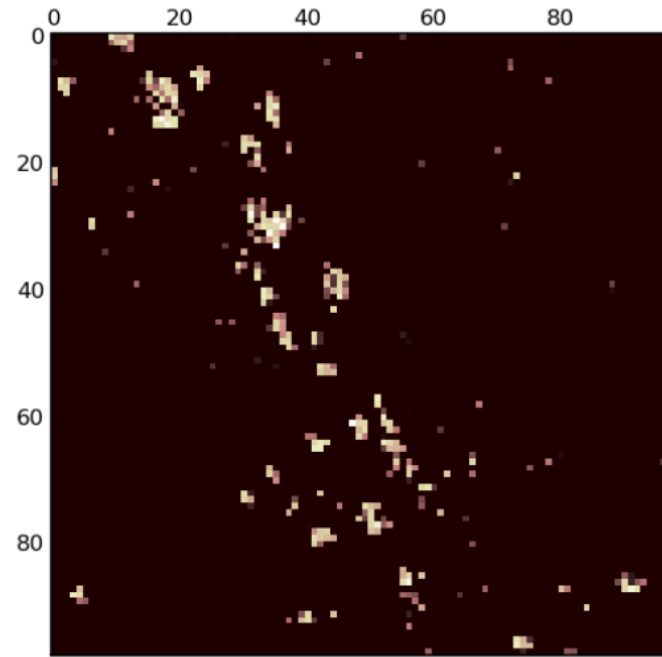
How much will it cost to connect a node?

Using which technology?

In what order?

Architecture

Computational core



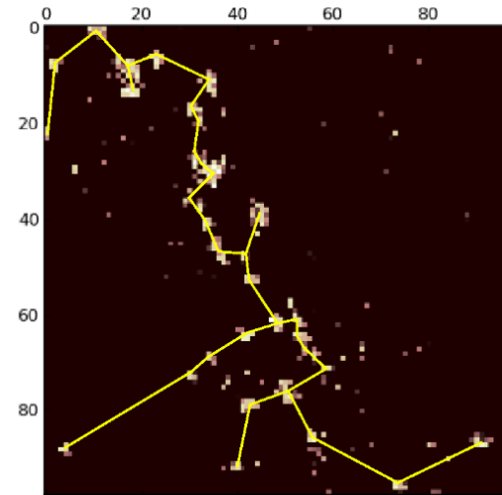
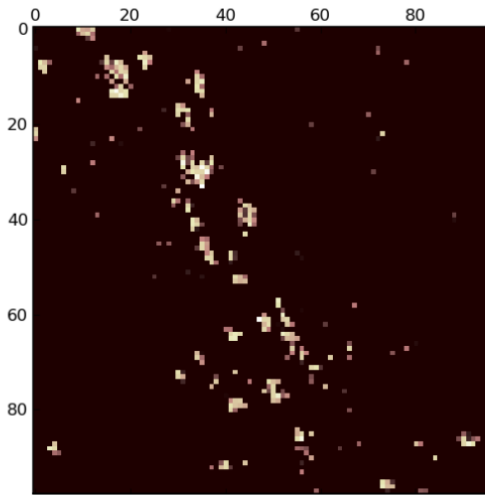
Architecture

Command-line utilities

Computational core

```
Applications Places System Mon Feb 8, 8:21 AM Roy
[screen 1: bash] rhh@september:~/Downloads/further-perception/tags/4.3/queues/examples/1
File Edit View Terminal Help
training with 500 training samples and 100 test samples
training: [ 500] size=500 energy=0.3596 correct=87.40% errors=12.60% rejects=0.00%
testing: [ 500] size=100 energy=0.4004 correct=88.00% errors=12.00% rejects=0.00%
training: [ 1000] size=500 energy=0.2655 correct=91.00% errors=9.00% rejects=0.00%
testing: [ 1000] size=100 energy=0.3331 correct=89.00% errors=11.00% rejects=0.00%
training: [ 1500] size=500 energy=0.2149 correct=93.00% errors=7.00% rejects=0.00%
testing: [ 1500] size=100 energy=0.3027 correct=90.00% errors=10.00% rejects=0.00%
*** Convolutional neural network ***
testError: 10.0
falsePositiveTestError: 4.1095890411
falseNegativeTestError: 25.9259259259
elapsed time in seconds = 63
Scanning region 1/6...
100 % 169
Scanning region 2/6...
100 % 169
Scanning region 3/6...
100 % 169
Scanning region 4/6...
100 % 169
Scanning region 5/6...
```

```
Applications Places System Mon Feb 8, 8:10 AM Roy
[screen 0: bash] rhh@september:~/Downloads/network-planner/tags/examples/2
File Edit View Terminal Help
[rrh@september 2]$ ls
01-registerDemographics.queue Kenya_ExistingNetwork metric.cfg
02-computeMetrics.queue Kenya_ExistingNetwork.zip network.cfg
03-buildNetworks.queue Kenya_Nodes_Points restart
go Kenya_Nodes_Points.zip
[rrh@september 2]$ ./go
/usr/lib64/python2.6/site-packages/scipy/stats/stats.py:420: DeprecationWarning: scipy.stat
s.mean is deprecated; please update your code to use numpy.mean.
Please note that:
- numpy.mean axis argument defaults to None, not 0
- numpy.mean has a ddof argument to replace bias in a more general manner.
scipy.stats.mean(a, bias=True) can be replaced by numpy.mean(x,
axis=0, ddof=1).
axis=0, ddof=1)"""', DeprecationWarning)
Simplifying existing network: 1737.64210296
Generating projected segment candidates...
Building network from segments...
100 % 6742
Building network: 15713.4472561
[rrh@september 2]$
```



Architecture



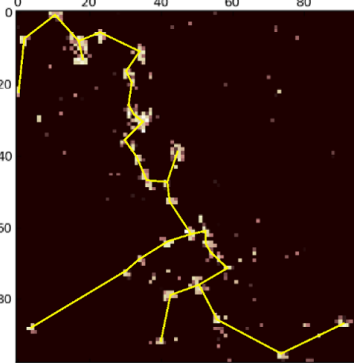
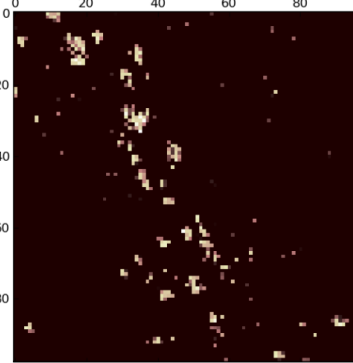
Web service

Command-line utilities

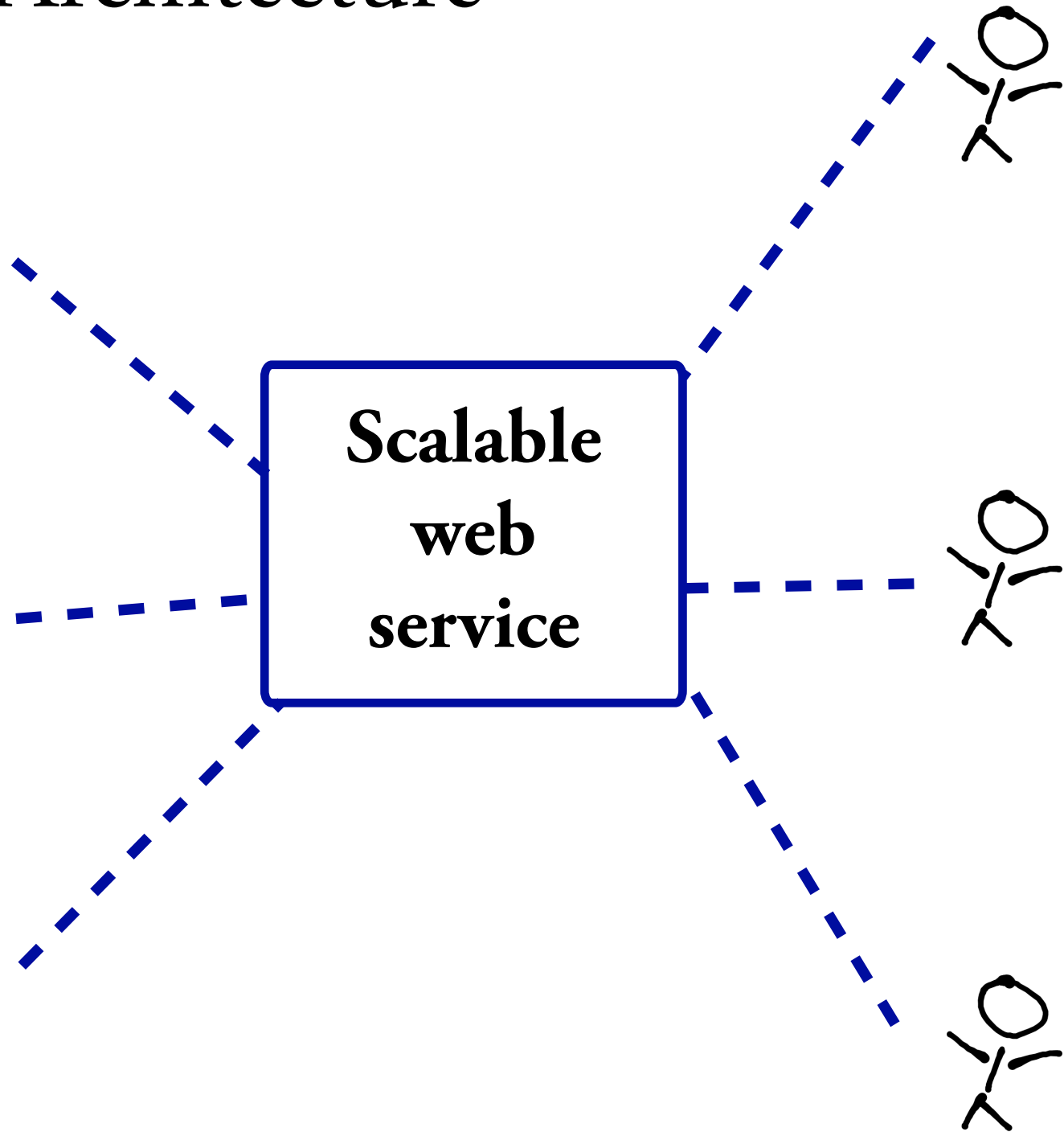
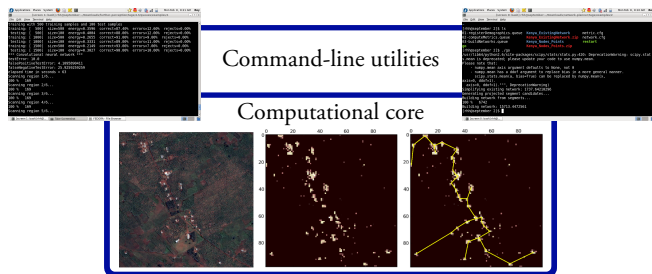
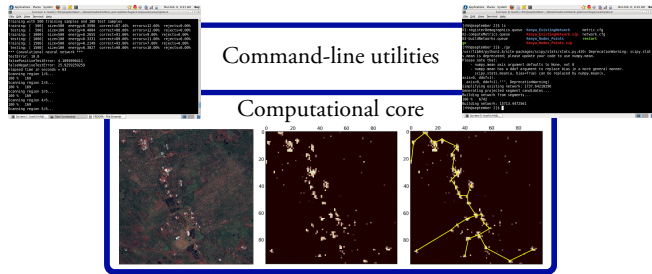
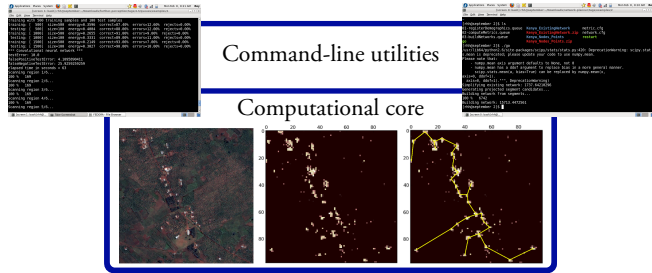
Computational core

```
Applications Places System Mon Feb 8, 8:21 AM Roy
[screen 1: bash] rhh@september:~/Downloads/further-perception/tags/4.3/queues/examples1
File Edit View Terminal Help
training with 500 training samples and 100 test samples
training: [ 500] size=500 energy=0.3596 correct=87.40% errors=12.60% rejects=0.00%
testing: [ 500] size=100 energy=0.4604 correct=88.00% errors=12.00% rejects=0.00%
training: [ 1000] size=500 energy=0.2655 correct=91.00% errors=9.00% rejects=0.00%
testing: [ 1000] size=100 energy=0.3331 correct=89.00% errors=11.00% rejects=0.00%
training: [ 1500] size=500 energy=0.2149 correct=93.00% errors=7.00% rejects=0.00%
testing: [ 1500] size=100 energy=0.3627 correct=90.00% errors=10.00% rejects=0.00%
*** Convolutional neural network ***
testError: 10.0
falsePositiveTestError: 4.1095890411
falseNegativeTestError: 25.9259259259
elapsed time in seconds = 63
Scanning region 1/6...
100 % 169
Scanning region 2/6...
100 % 169
Scanning region 3/6...
100 % 169
Scanning region 4/6...
100 % 169
Scanning region 5/6...
100 % 169
[screen 1: bash] rhh@... Take Screenshot FEDORA - File Browser
```

```
Applications Places System Mon Feb 8, 8:10 AM Roy
[screen 0: bash] rhh@september:~/Downloads/network-planner/tags/examples2
File Edit View Terminal Help
[rhh@september 2] $ ls
01-registerDemographics.queue Kenya_ExistingNetwork metric.cfg
02-computeMetrics.queue Kenya_ExistingNetwork.zip network.cfg
03-buildNetworks.queue Kenya_Nodes_Points restart
go Kenya_Nodes_Points.zip
[rhh@september 2] $ /go
/usr/lib64/python2.6/site-packages/scipy/stats/stats.py:420: DeprecationWarning: scipy.stat
s.mean is deprecated; please update your code to use numpy.mean.
Please note that:
- numpy.mean axis argument defaults to None, not 0
- numpy.mean has a ddof argument to replace bias in a more general manner.
scipy.stats.mean(a, bias=True) can be replaced by numpy.mean(x,
axis=0, ddof=1).
axis=0, ddof=1) *** DeprecationWarning)
Simplifying existing network: 1737.64210296
Generating projected segment candidates...
Building network from segments...
100 % 0/42
Building network: 15713.4472561
[rhh@september 2] $
```



Architecture

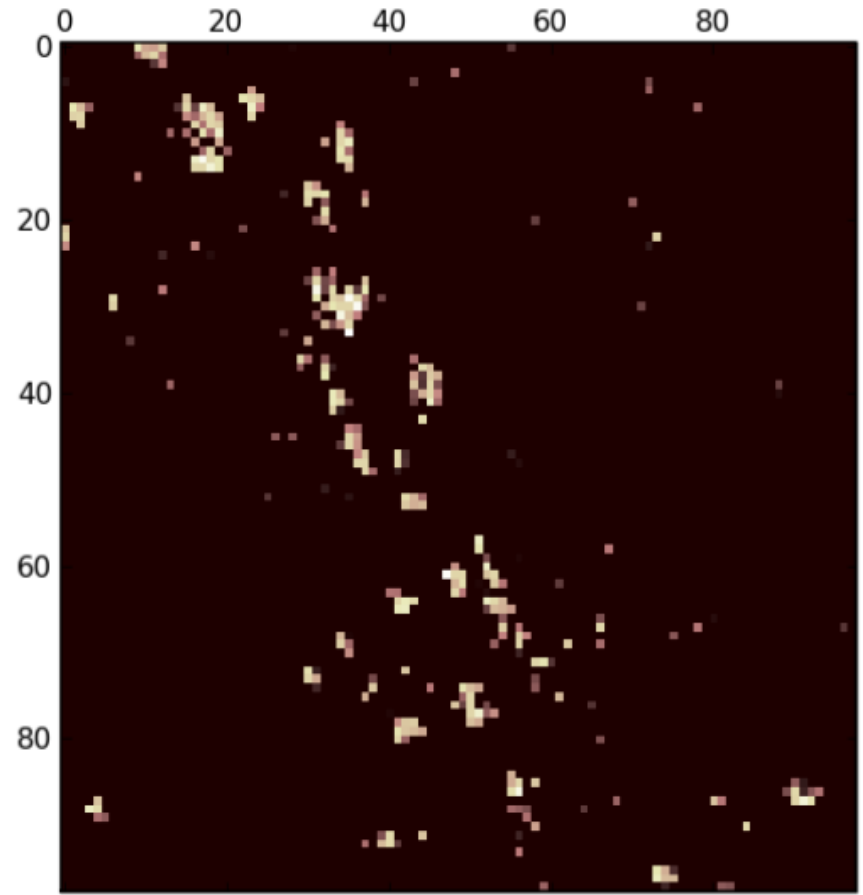


Success story #1: Remote sensing



Question	Where do people live?	
Manual	Send people with GPS devices Click on houses in satellite images	
Method	Machine learning	Image recognition
Command line	subprocess Lush	osgeo GDAL Generators
Scalable web	Pylons SQLAlchemy amqp 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

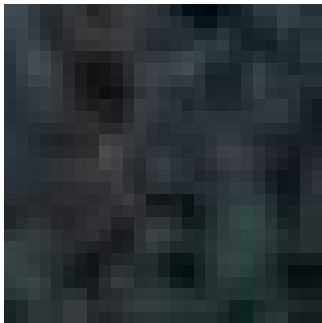


How do we mark houses automatically?

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.

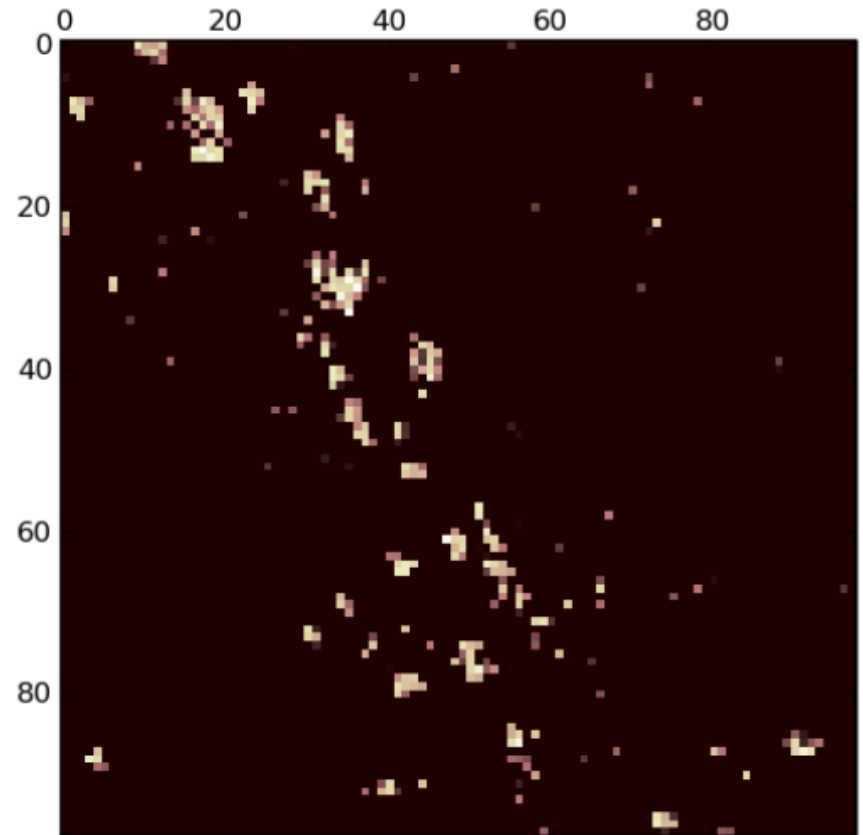
How to scan an image with a classifier



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?

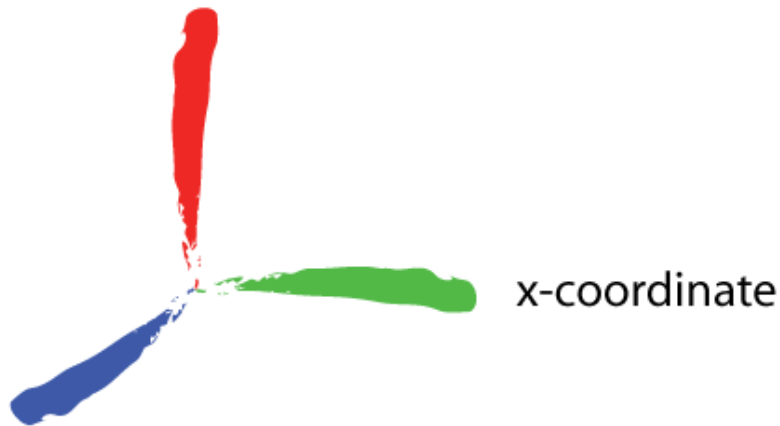
2. Scan local classifier over image to generate a matrix of probabilities



How do we mark houses automatically?

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

probability that
the region
contains a house



y-coordinate



How do we mark houses automatically?

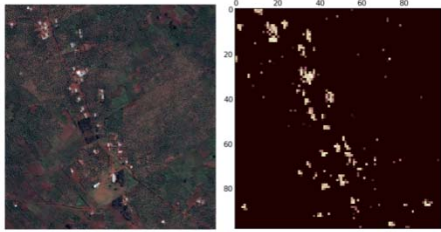
3. Cluster in three dimensions to turn local probabilities into points

Use K-means

Pop large clusters

Remove small clusters





Computational core: Remote sensing

Load raster data from images
Save vector data to shapefiles

```
import osgeo.gdal  
import osgeo.ogr
```

Run Lisp machine learning code

```
import subprocess
```

Cluster points
Save matrices

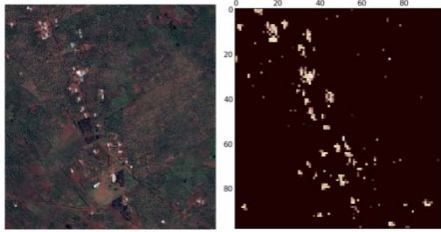
```
import scipy.cluster.vq  
import scipy.io
```

Scan image with little memory

```
yield
```

Render 16-bit satellite image

```
import matplotlib
```



Computational core: Remote sensing

Run Lisp machine learning code

```
import subprocess
```

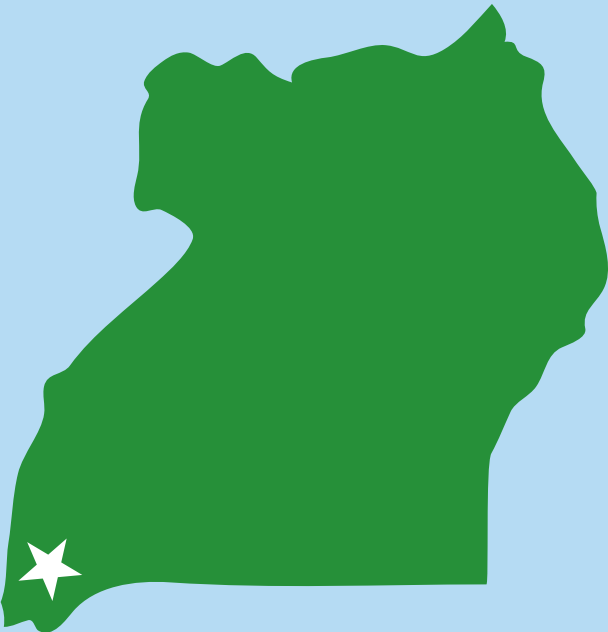
```
p = subprocess.Popen(programArguments,  
                      stdin=subprocess.PIPE,  
                      stdout=subprocess.PIPE)
```

```
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



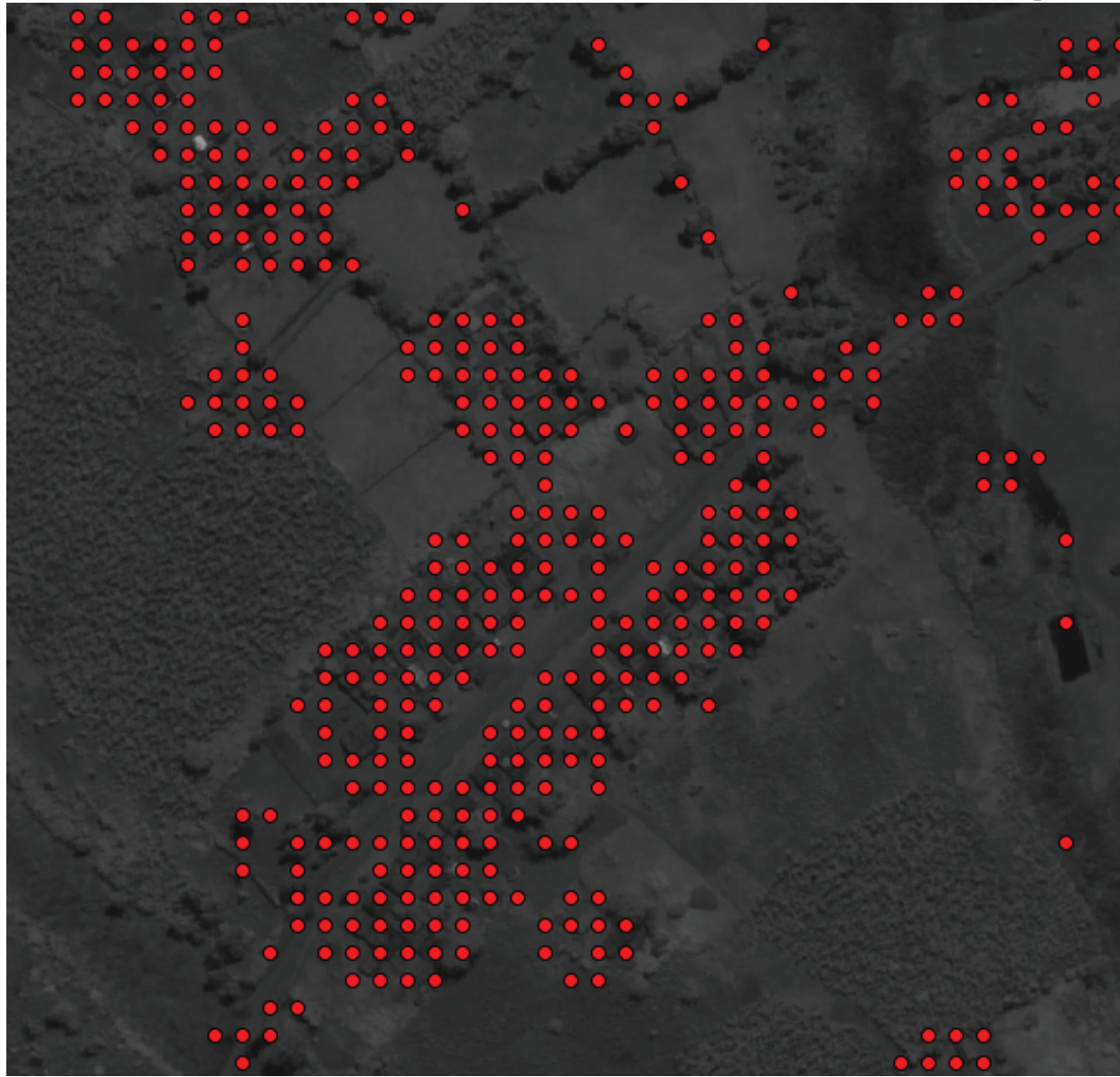
Farm 2 - Multispectral

Results from Ruhiira in Uganda



Farm 2 - Panchromatic

Results from Ruhiira in Uganda



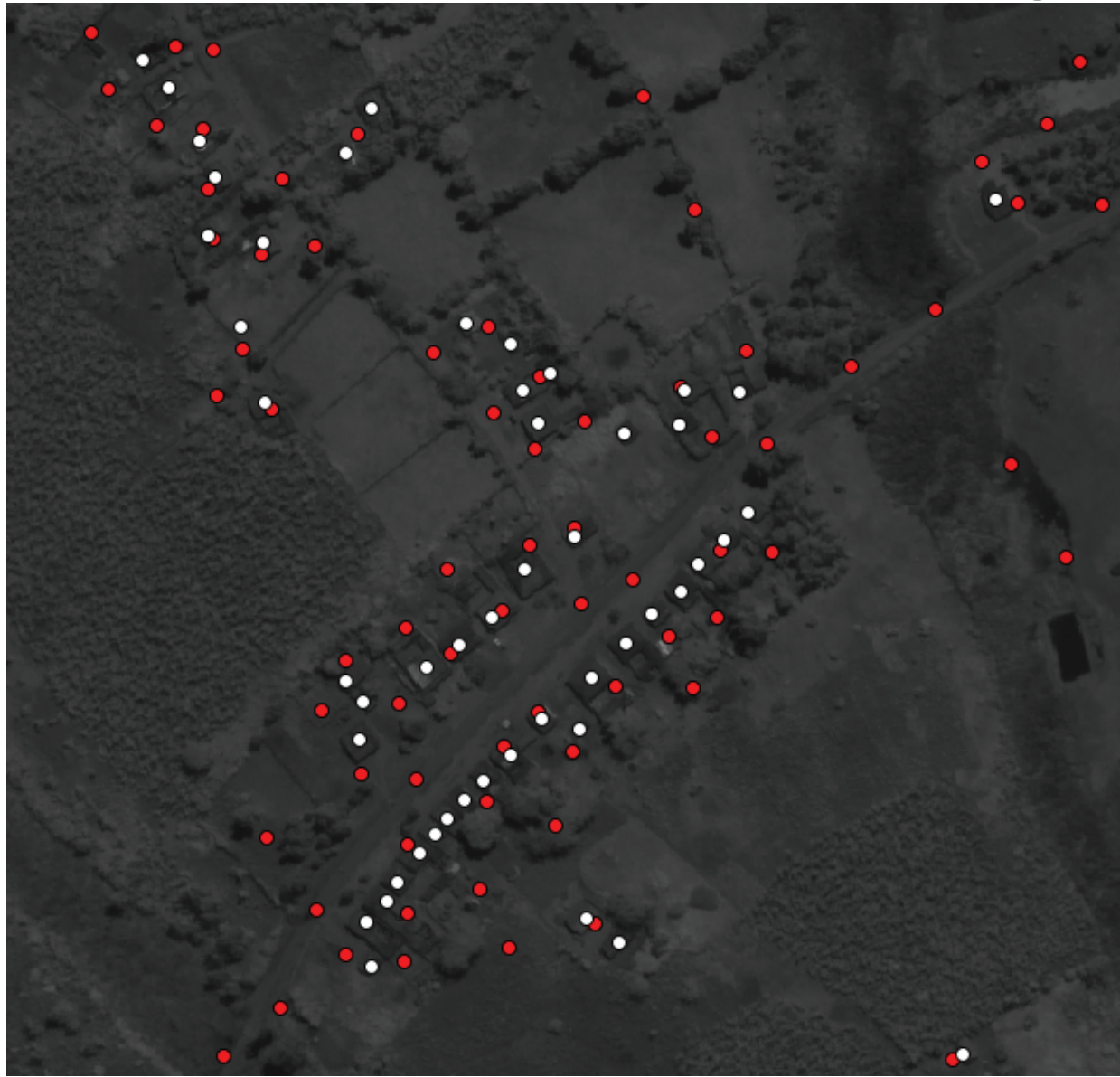
Farm 2 - Computer probabilities

Results from Ruhiira in Uganda



Farm 2 - Computer

Results from Ruhiira in Uganda



Farm 2 - Computer & Human

Results from Ruhiira in Uganda

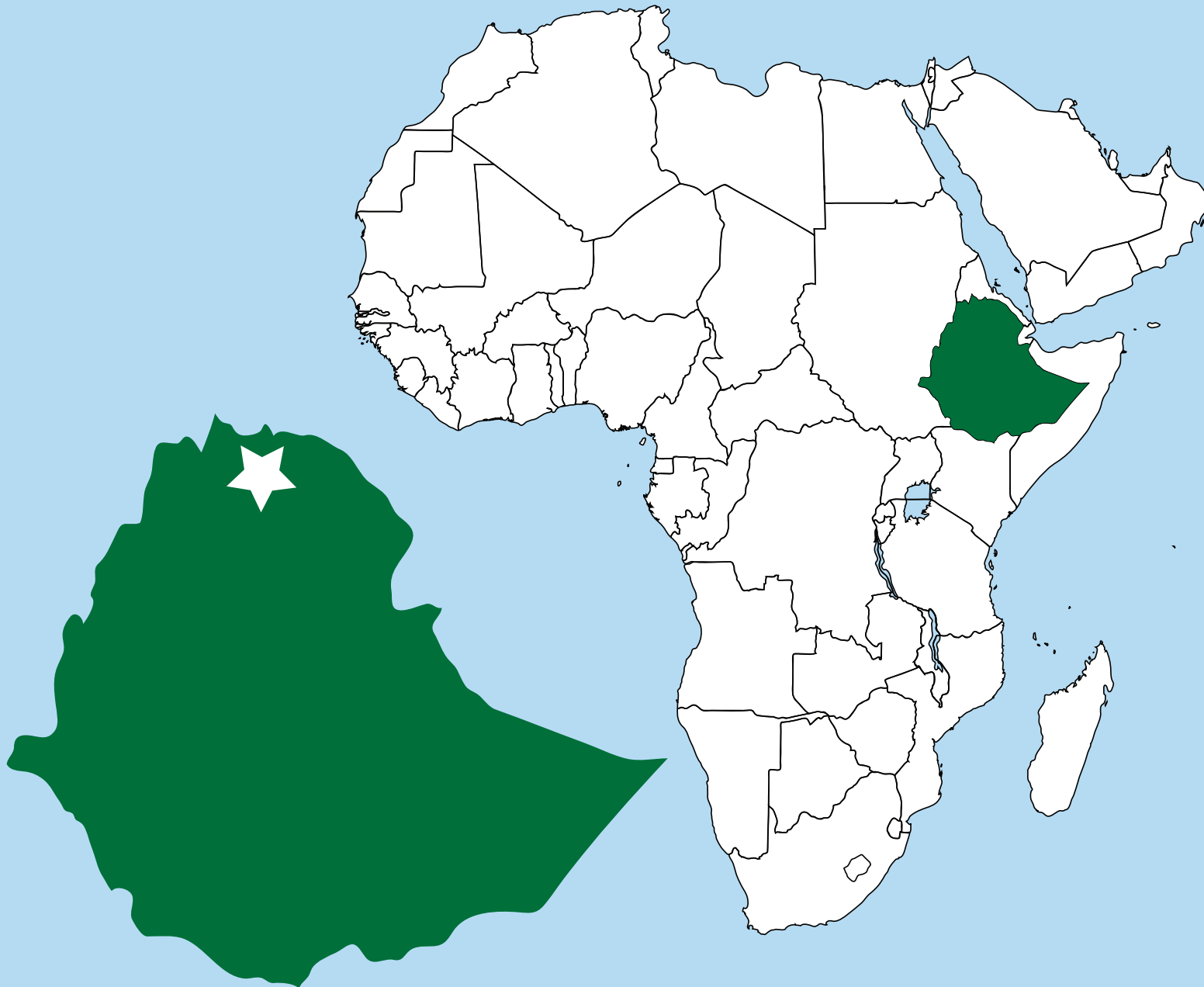


Farm 2 - Human

Where is Ethiopia?



Where is Koraro?



Results from Koraro in Ethiopia



Farm 1 - Multispectral

Results from Koraro in Ethiopia



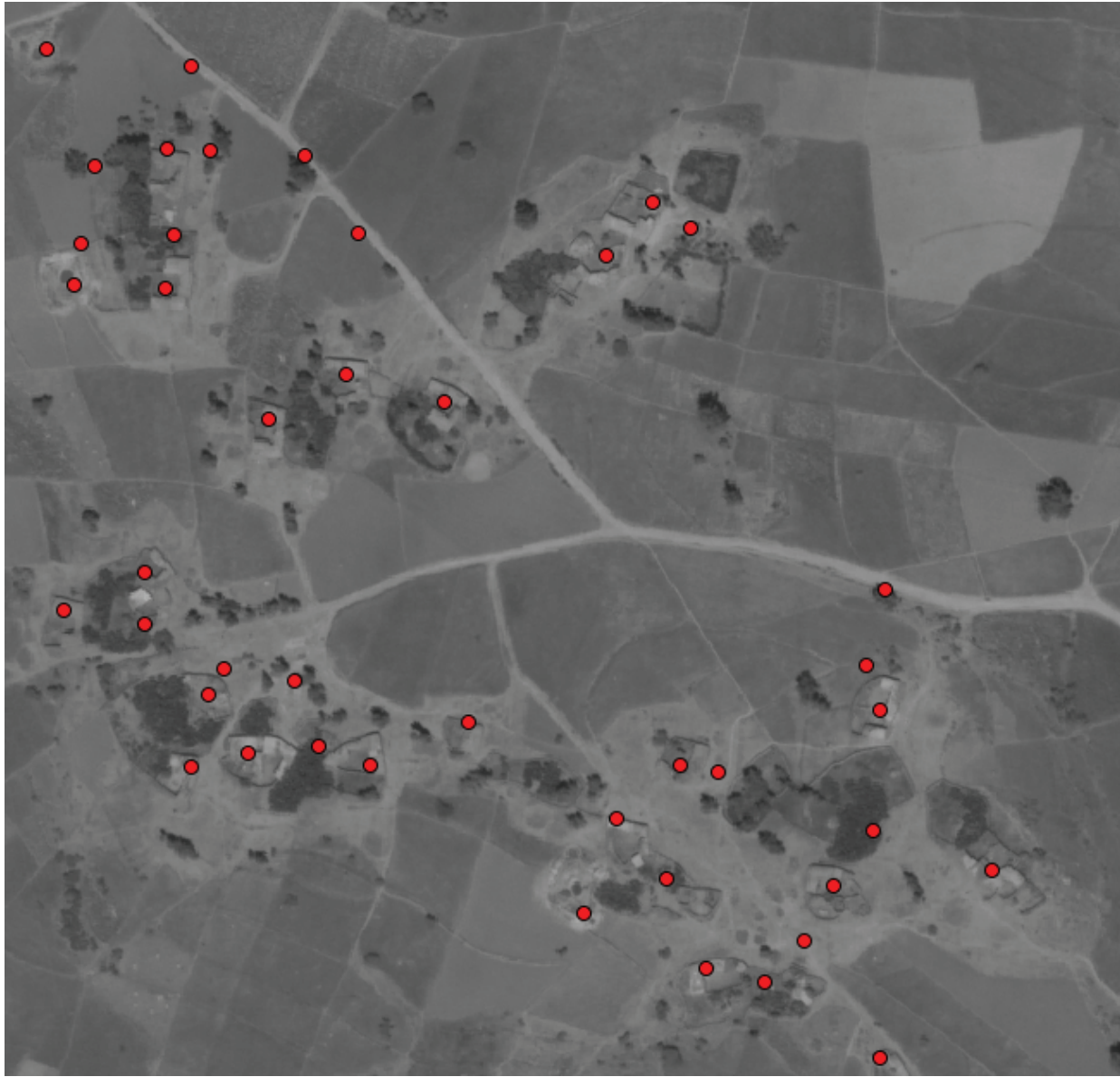
Farm 1 - Panchromatic

Results from Koraro in Ethiopia



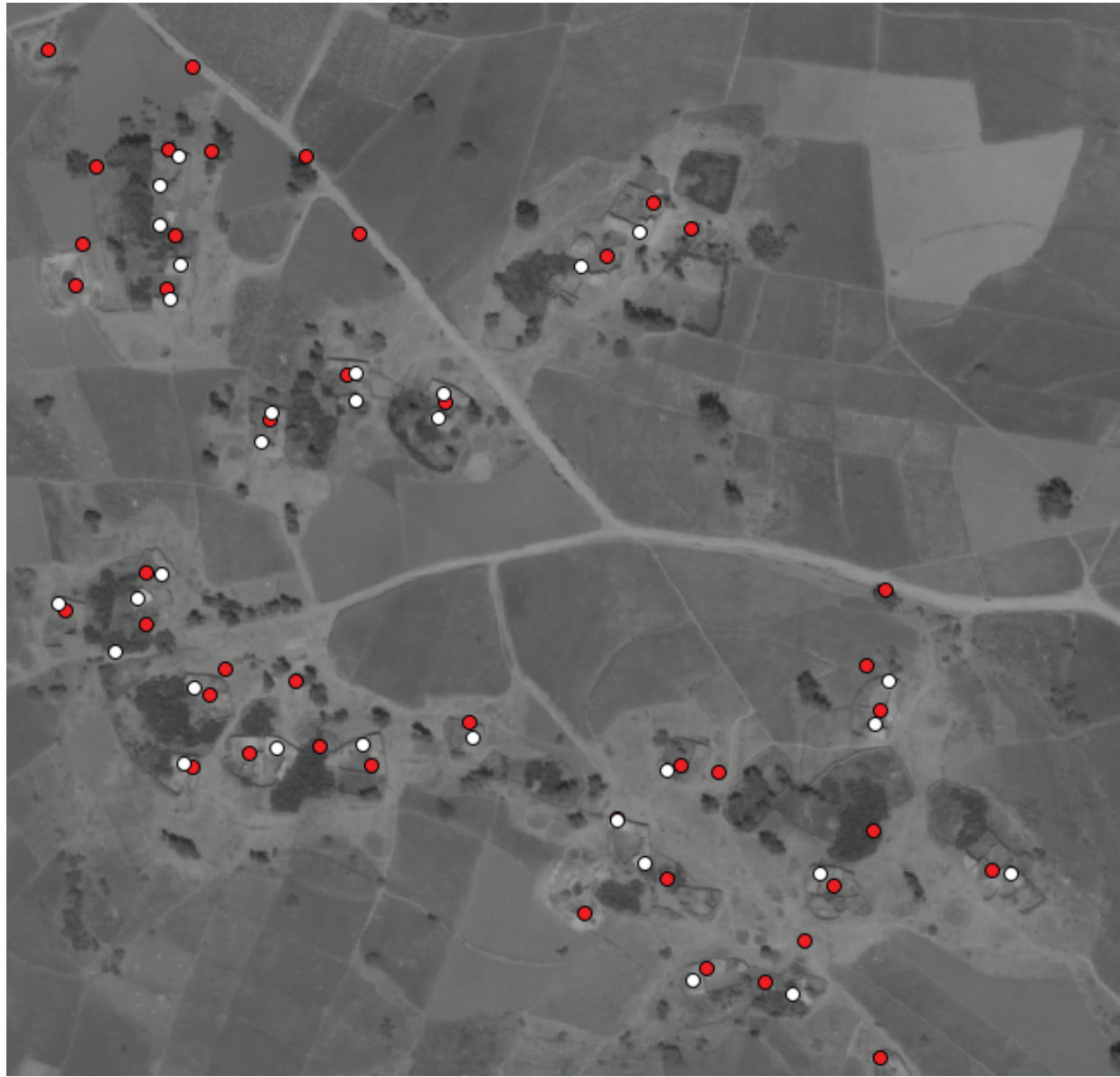
Farm 1 - Computer probabilities

Results from Koraro in Ethiopia



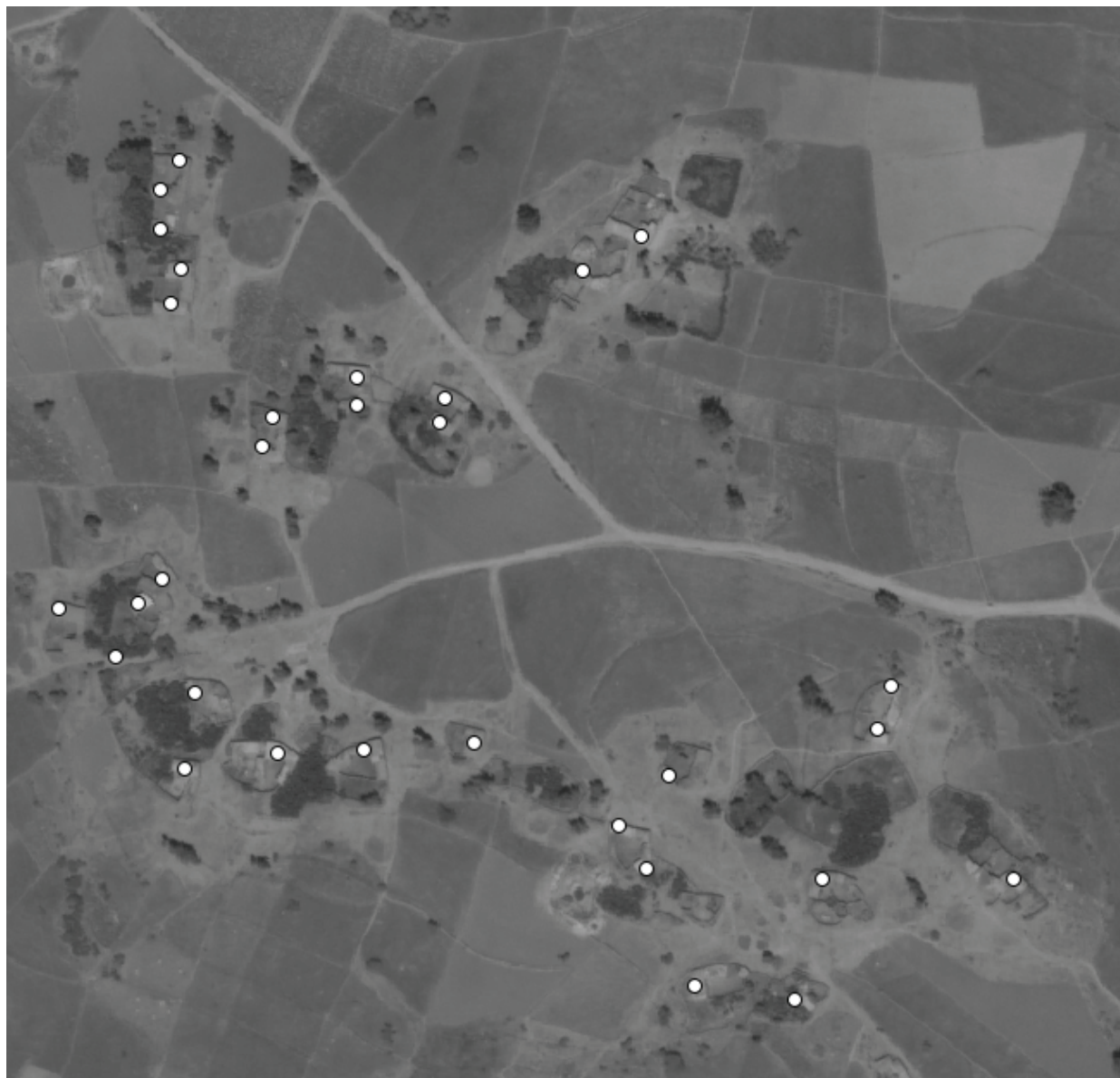
Farm 1 - Computer

Results from Koraro in Ethiopia



Farm 1 - Computer & Human

Results from Koraro in Ethiopia

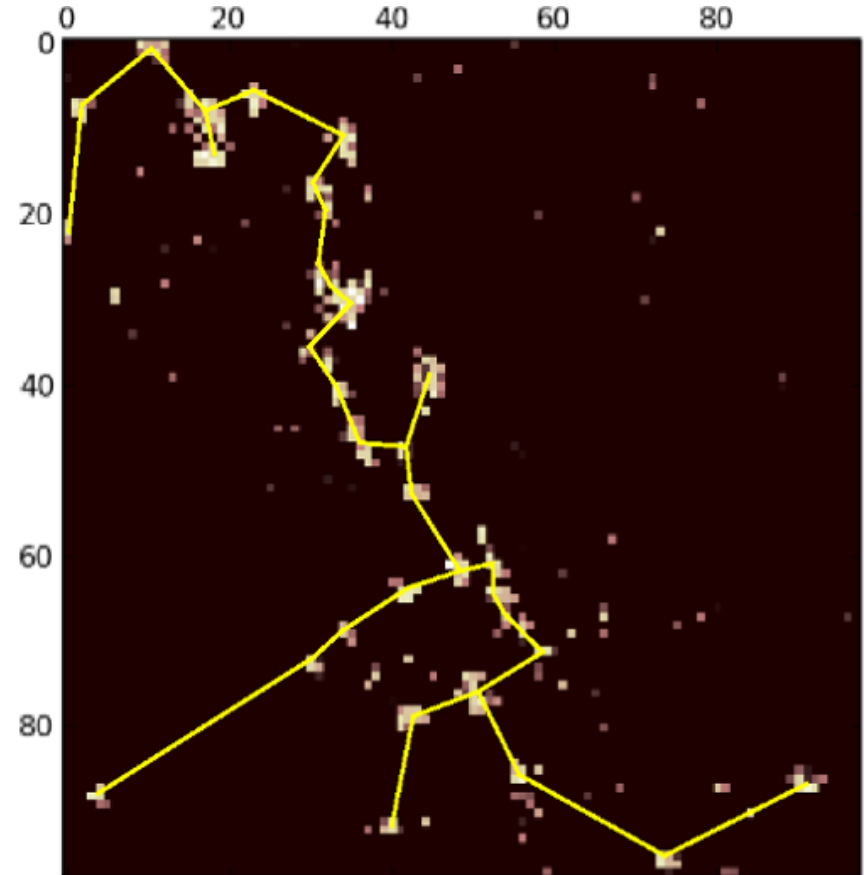
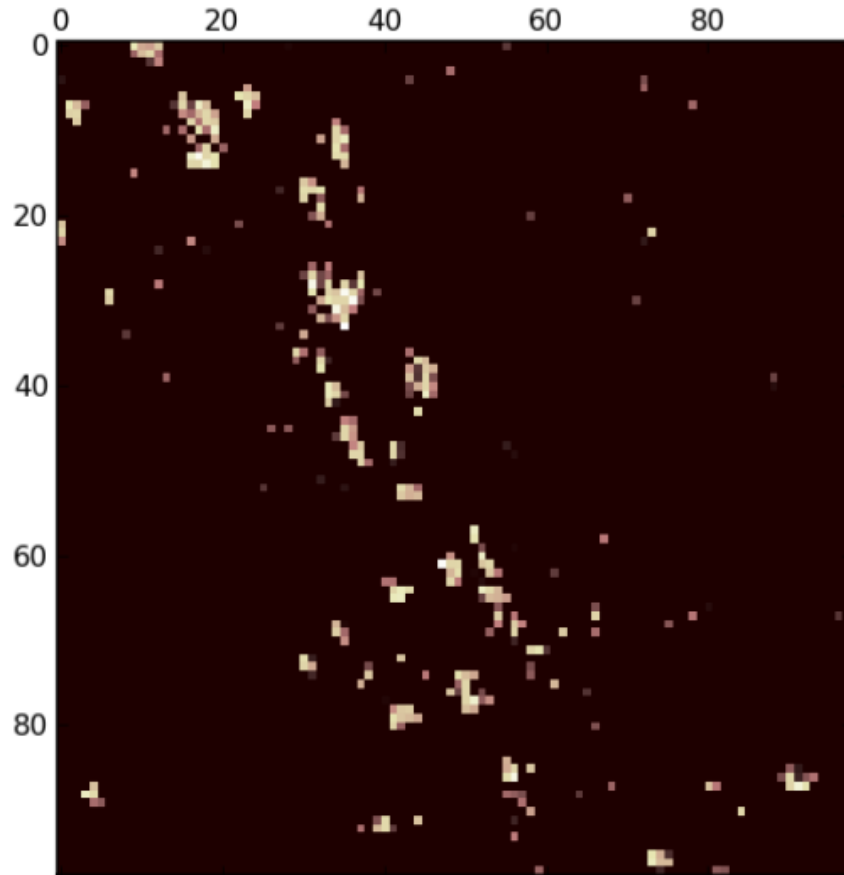


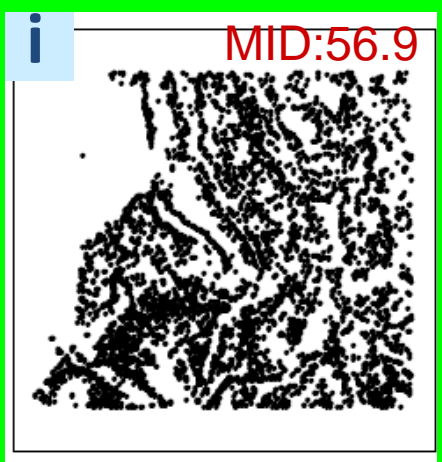
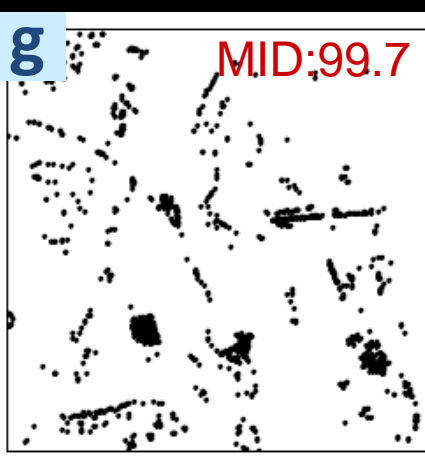
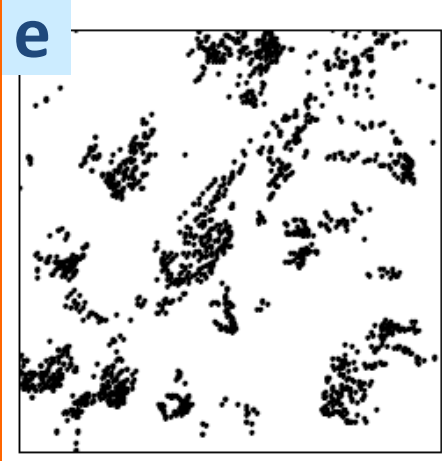
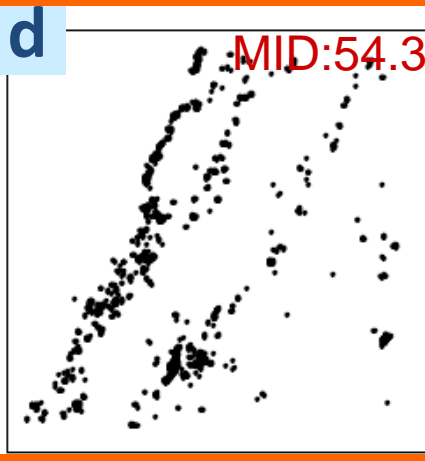
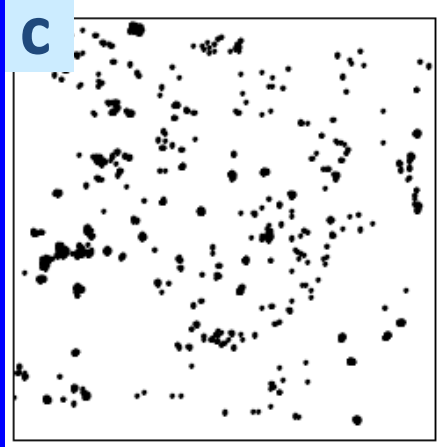
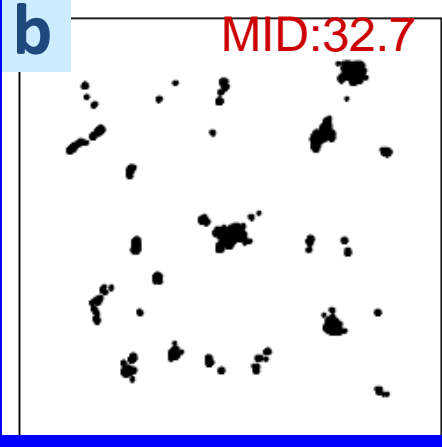
Farm 1 - Human

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 scipy	shapely geos	geojson openlayers osgeo proj4
Scalable web	Pylons SQLAlchemy amqp RabbitMQ		

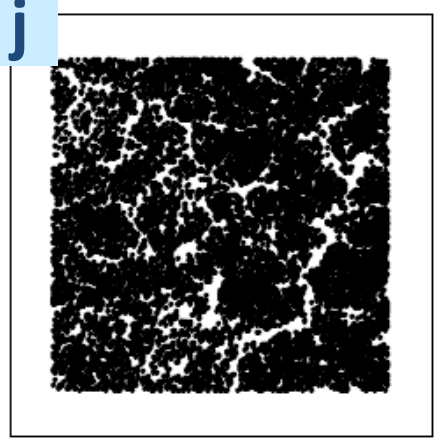
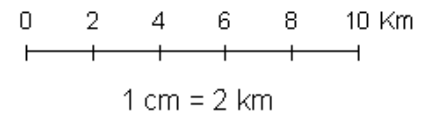
Computational core: Network modeling



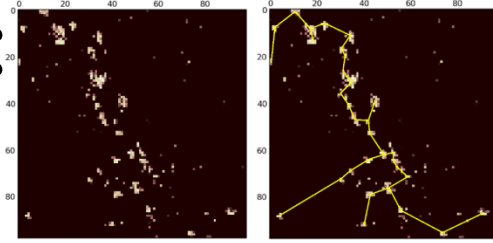


- a) Bonsaaso, GHANA
- b) Tiby, MALI
- c) Pampaida, NIGERIA
- d) Potou, SENEGAL
- e) Koraro, ETHIOPIA
- f) Mwandama, MALAWI
- g) Mbola, TANZANIA
- h) Mayange, RWANDA
- i) Ruhiira, UGANDA
- j) Sauri, KENYA

Zvoleff, Kocaman,
Huh, Modi (2009)



Computational core: Network modeling



For each node,

Project demographic growth

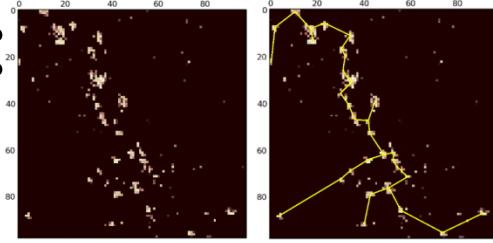
Project electricity demand

Estimate construction/maintenance cost

Select electricity system via geospatial info



Computational core: Network modeling



Estimate metrics for each node

```
import scipy.stats
```

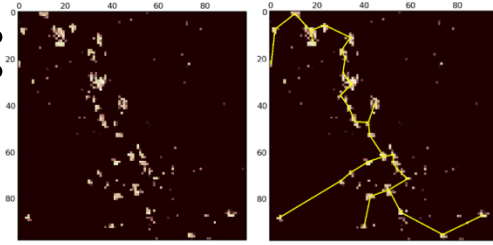
Find distance between lines
Project points onto lines
Find intersections between lines
Merge and simplify lines

```
import shapely
```

Store nodes and segments efficiently

```
import sqlalchemy
```

Computational core: Network modeling

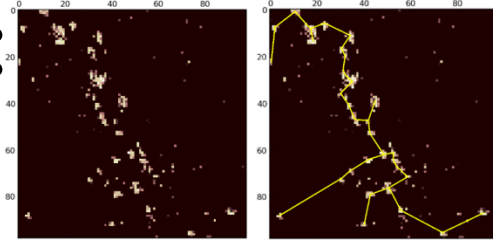


Find distance between lines
Project points onto lines
Find intersections between lines
Merge and simplify lines

```
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



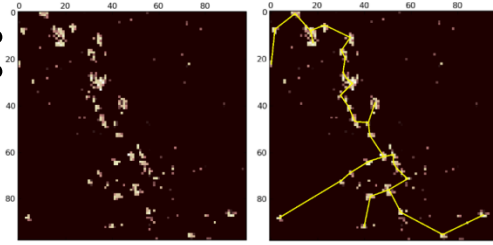
Find distance between lines
Project points onto lines
Find intersections between lines
Merge and simplify lines

```
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

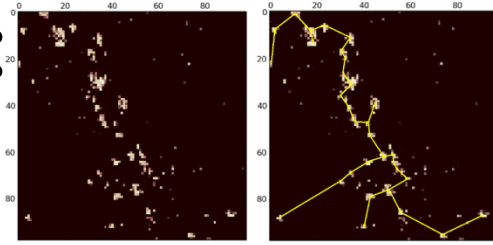


Find distance between lines
Project points onto lines
Find intersections between lines
Merge and simplify lines

```
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)
```

```
Applications Places System [screen 1: bash] rhh@september:~/Downloads/further-perception/tags/4.3/queues/examples/1
training with 500 training samples and 100 test samples
training: [ 500] size=500 energy=0.3590 correct=87.40% errors=12.60% rejects=0.00%
testing: [ 500] size=100 energy=0.4004 correct=88.00% errors=12.00% rejects=0.00%
training: [ 1000] size=500 energy=0.2655 correct=91.00% errors=9.00% rejects=0.00%
testing: [ 1000] size=100 energy=0.3331 correct=89.00% errors=11.00% rejects=0.00%
training: [ 1500] size=500 energy=0.2149 correct=93.00% errors=7.00% rejects=0.00%
testing: [ 1500] size=100 energy=0.3027 correct=90.00% errors=10.00% rejects=0.00%
*** Convolutional neural network ***
testError: 10.0
falsePositiveTestError: 4.1095890411
falseNegativeTestError: 25.9259259259
elapsed time in seconds = 63
Scanning region 1/6...
100 % 169
Scanning region 2/6...
100 % 169
Scanning region 3/6...
100 % 169
Scanning region 4/6...
100 % 169
Scanning region 5/6...
```

Command-line

```
Applications Places System [screen 0: bash] rhh@september:~/Downloads/network-planner/tags/examples/2
[rhh@september 2] $ ls
01-registerDemographics.queue Kenya_ExistingNetwork metric.cfg
02-computeMetrics.queue Kenya_ExistingNetwork.zip network.cfg
03-buildNetworks.queue Kenya_Nodes_Points restart
go Kenya_Nodes_Points.zip
[rhh@september 2] $ ./go
/usr/lib64/python2.6/site-packages/scipy/stats/stats.py:420: DeprecationWarning: scipy.stat
s.mean is deprecated; please update your code to use numpy.mean.
Please note that:
- numpy.mean has a ddof argument to replace bias in a more general manner.
  scipy.stats.mean(a, bias=True) can be replaced by numpy.mean(x,
axis=0, ddof=1).
Simplifying existing network: 1737.64210296
Generating projected segment candidates...
Building network from segments...
100 % 6742
Building network: 15713.4472561
[rhh@september 2] $
```

Parse arguments and options

`import optparse`

Load and save configuration files

`import ConfigParser`

Compress and uncompress data

`import zipfile`

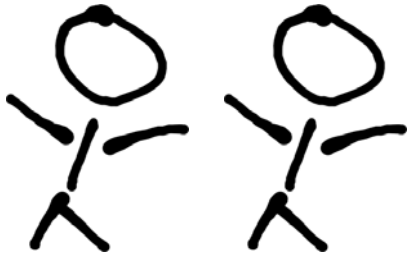
Get script path

`os.path.abspath(__file__)`

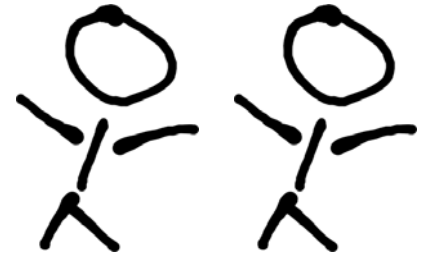


Web service

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



Scalable web service

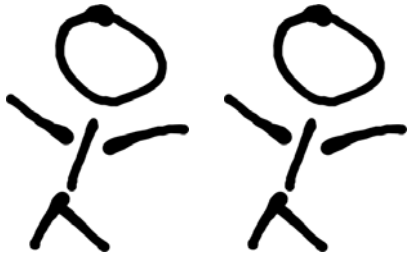


Serve queues

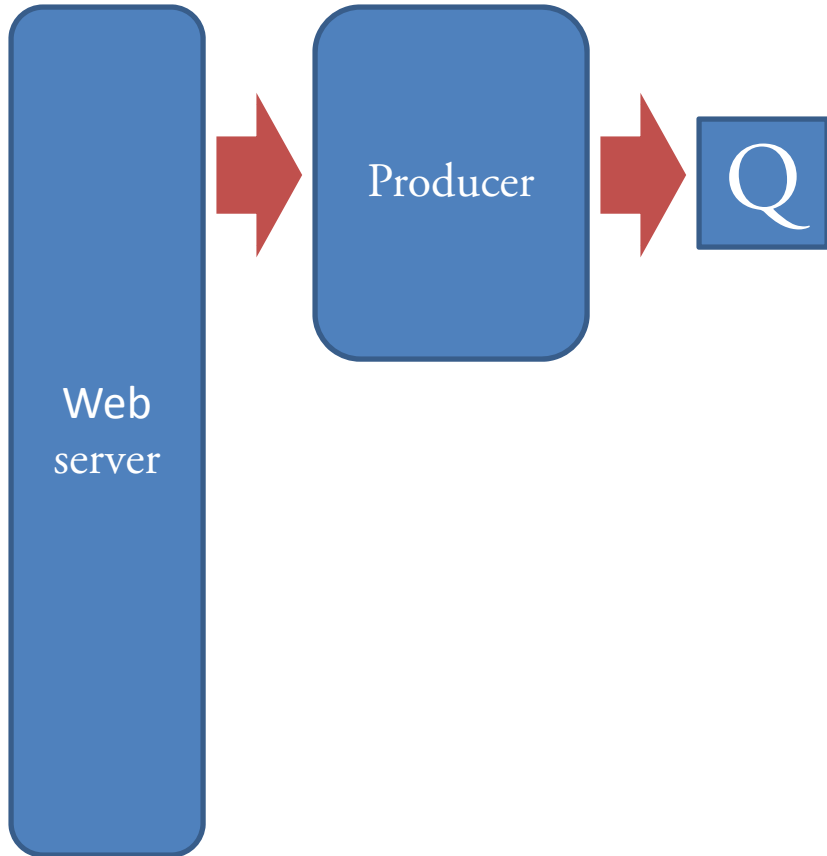
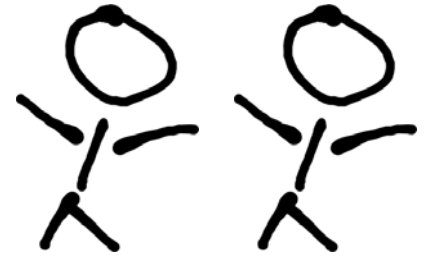
```
$ rabbitmq
```

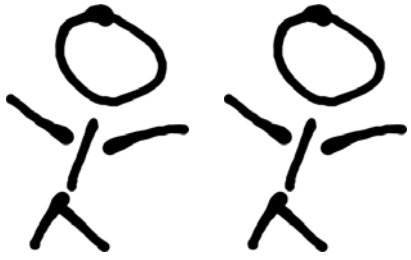
Put message on queue
Get message from queue

```
import amqp.lib
```

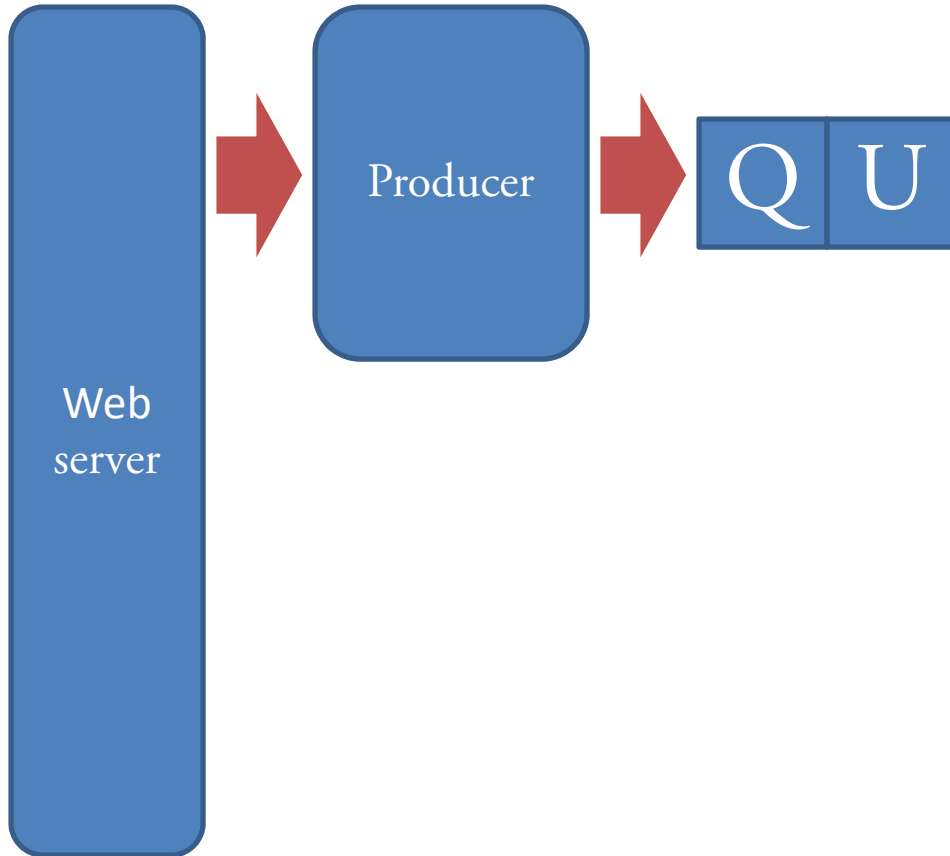
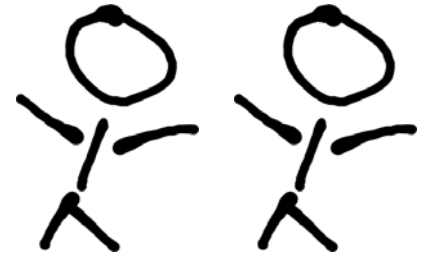



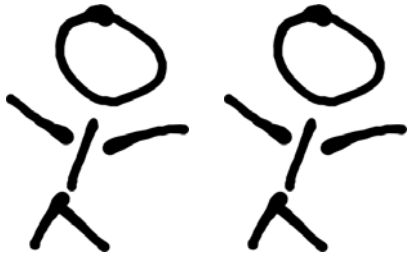
Scalable web service



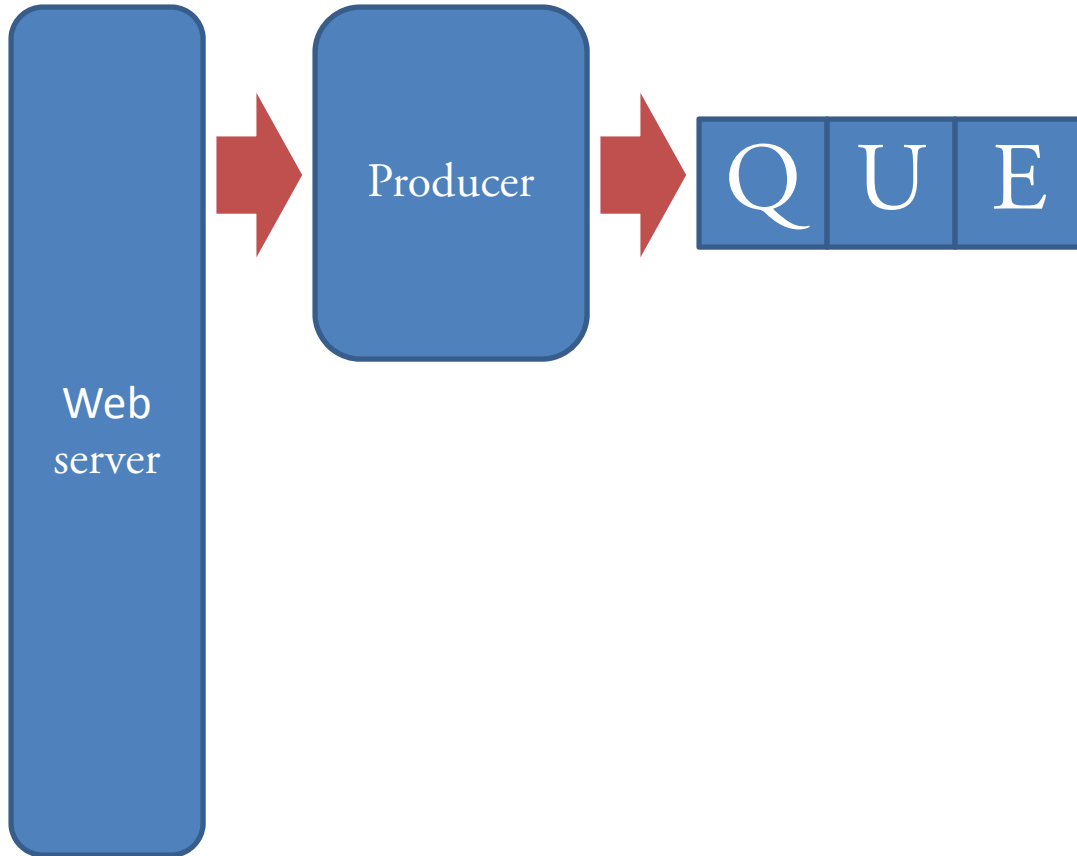
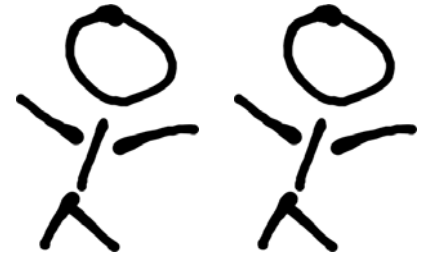


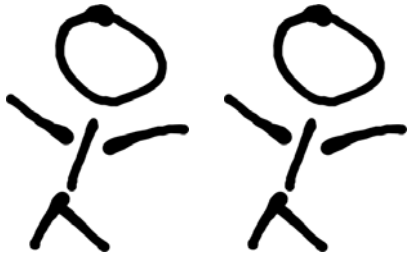
Scalable web service



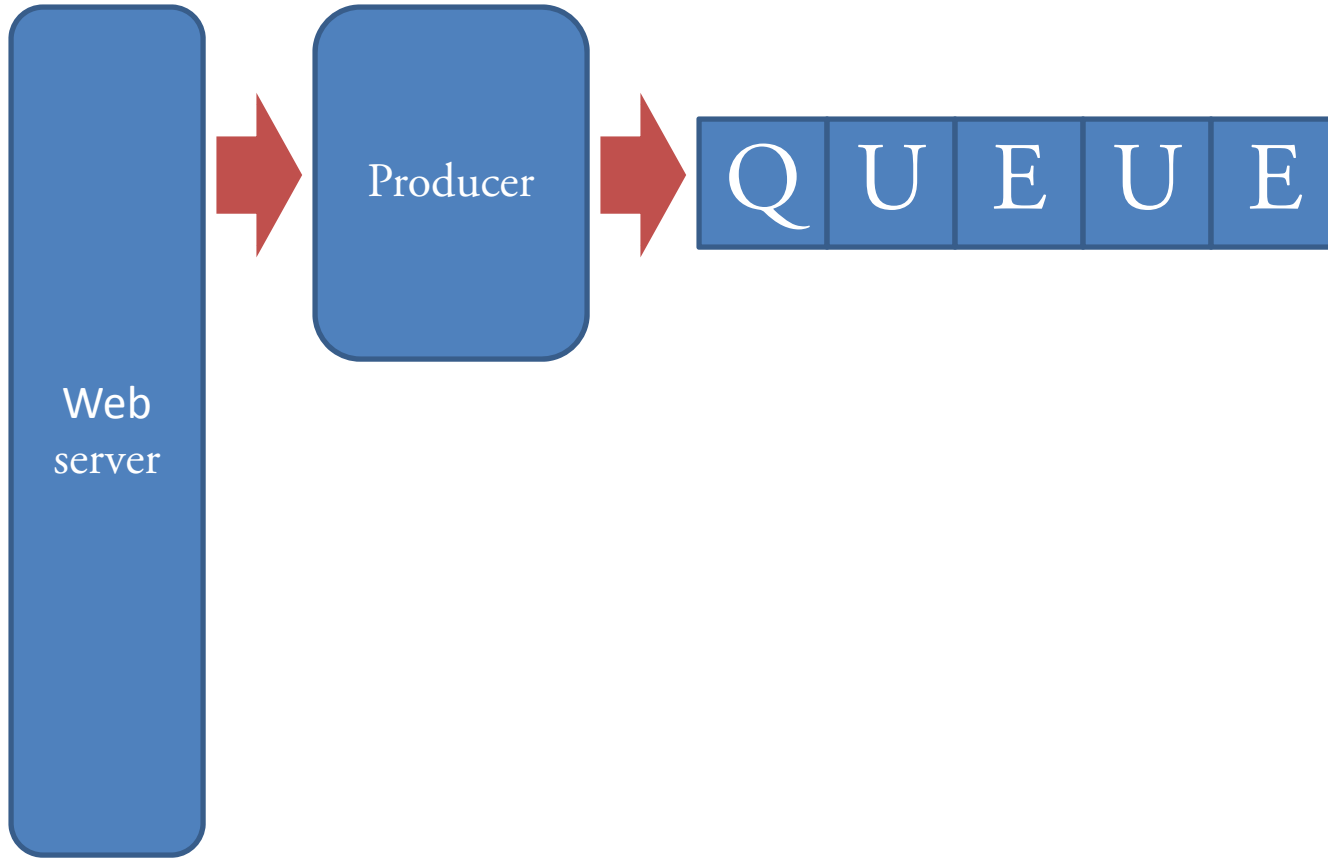
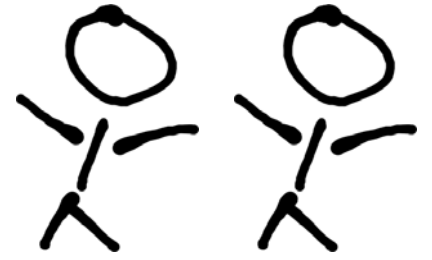


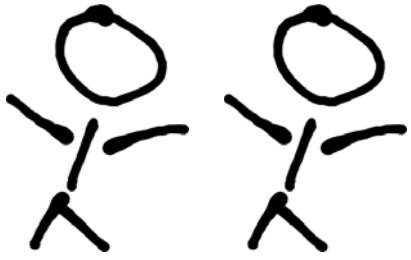
Scalable web service



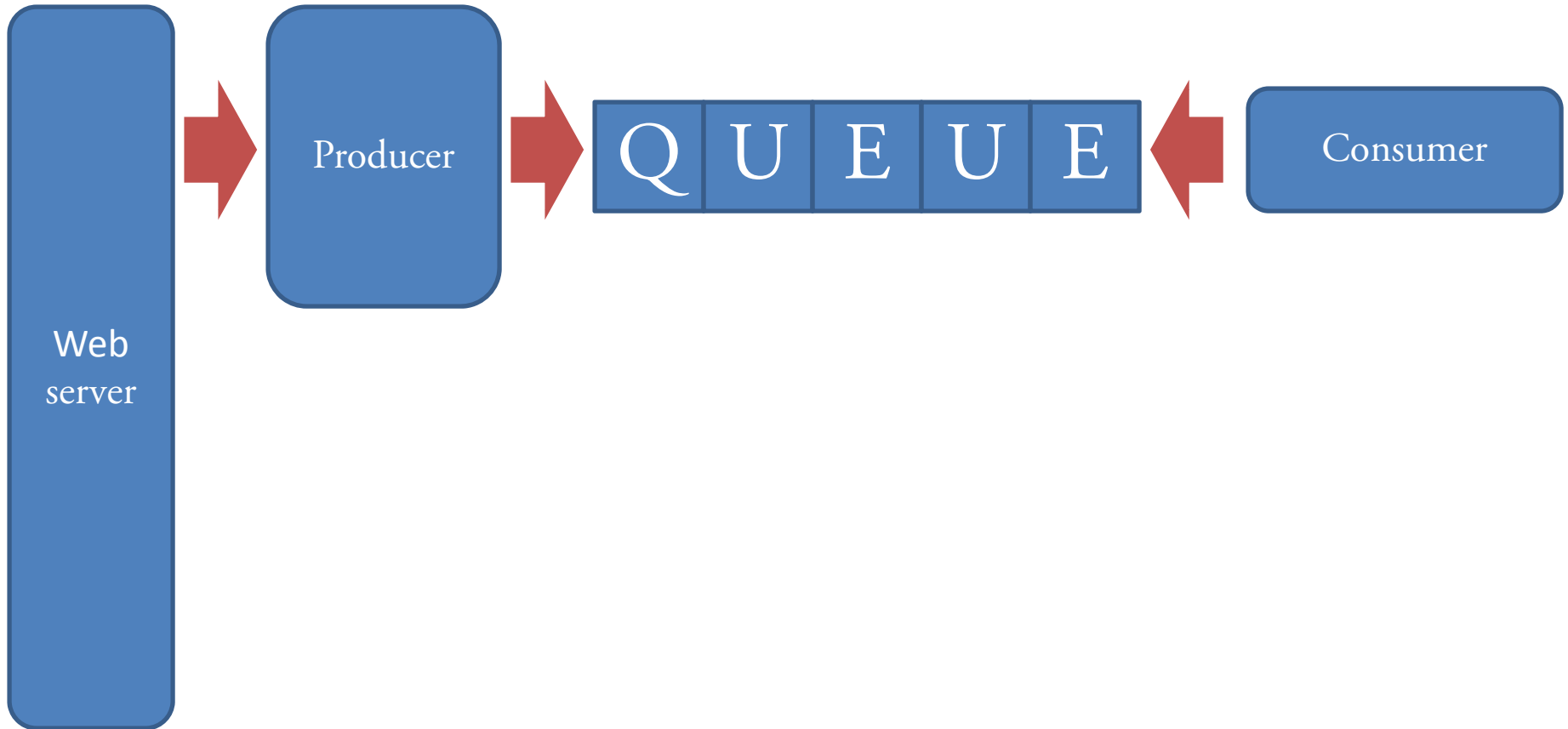
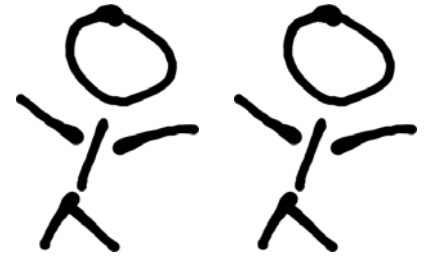


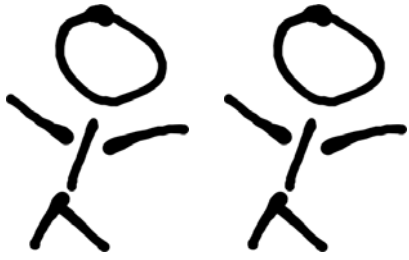
Scalable web service



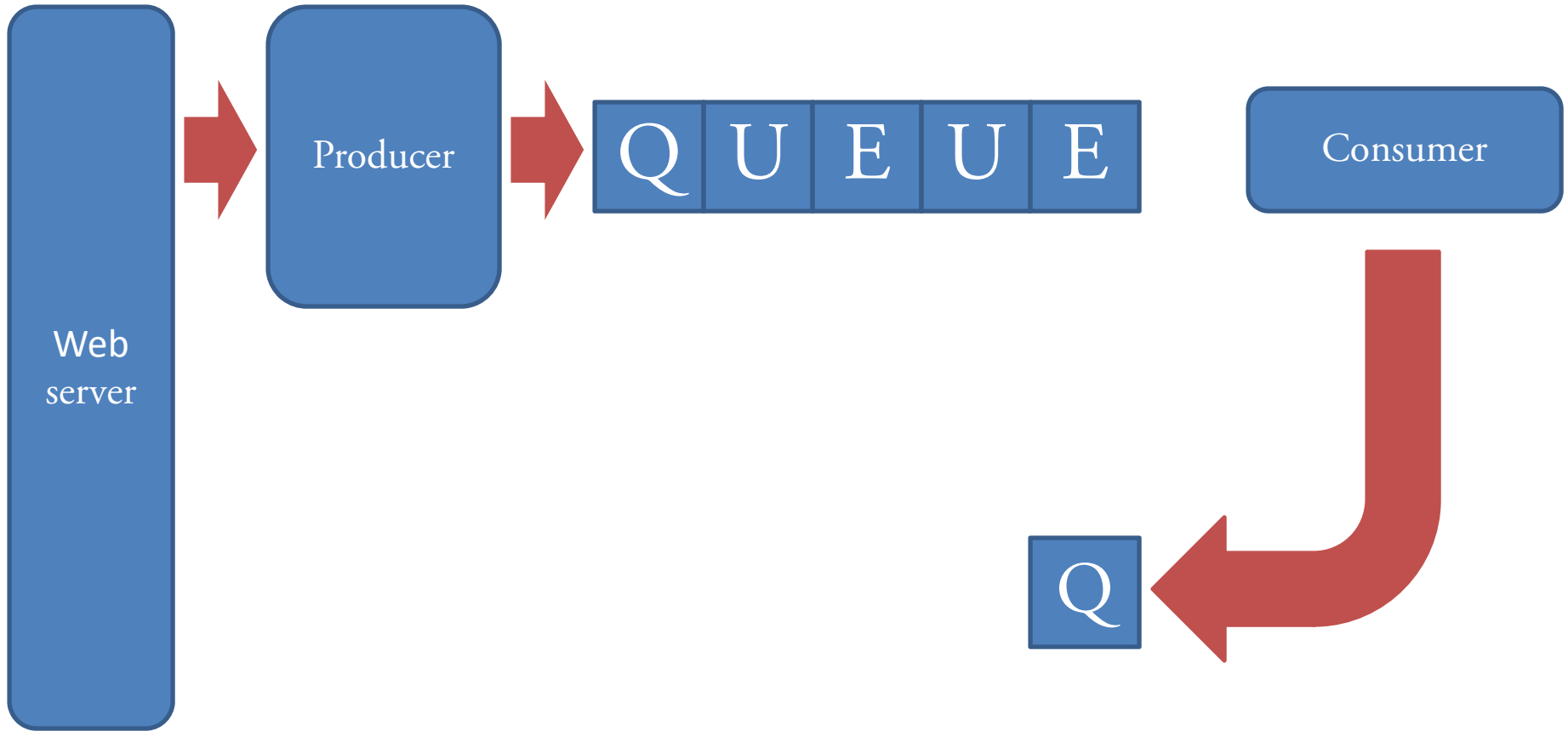
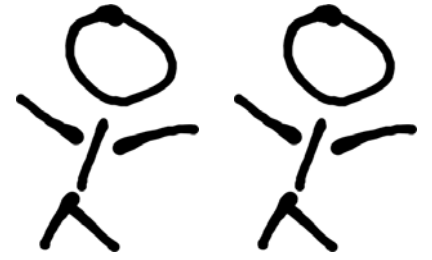


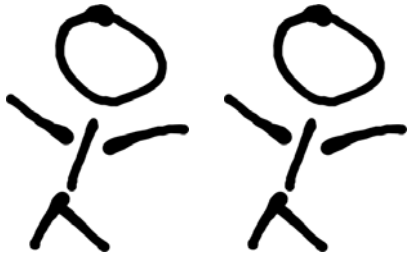
Scalable web service



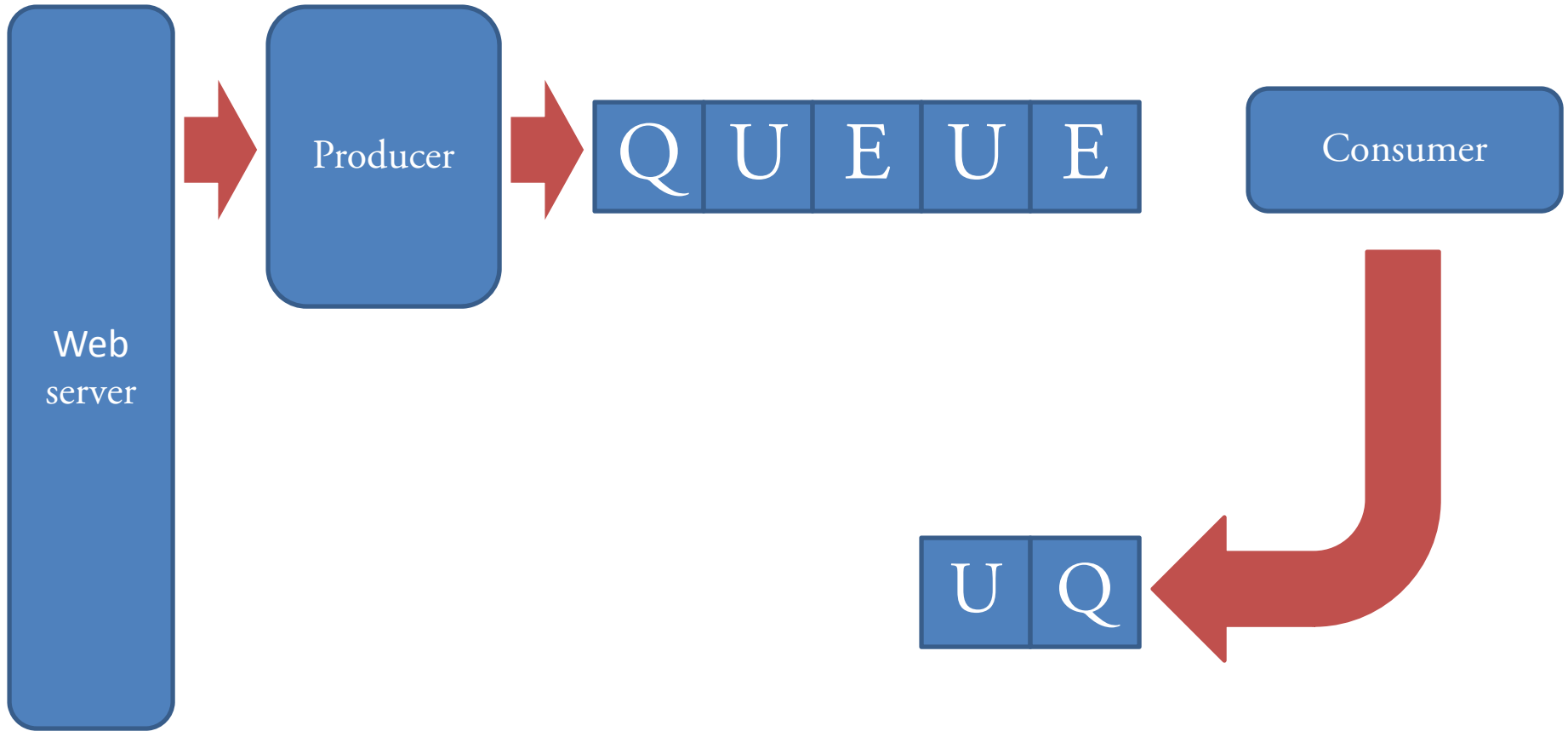
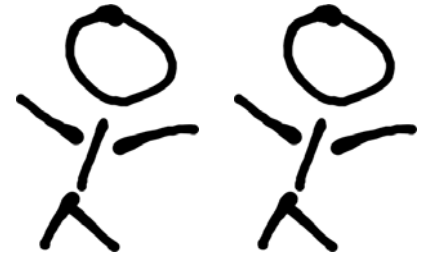


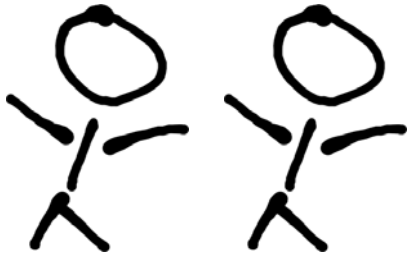
Scalable web service



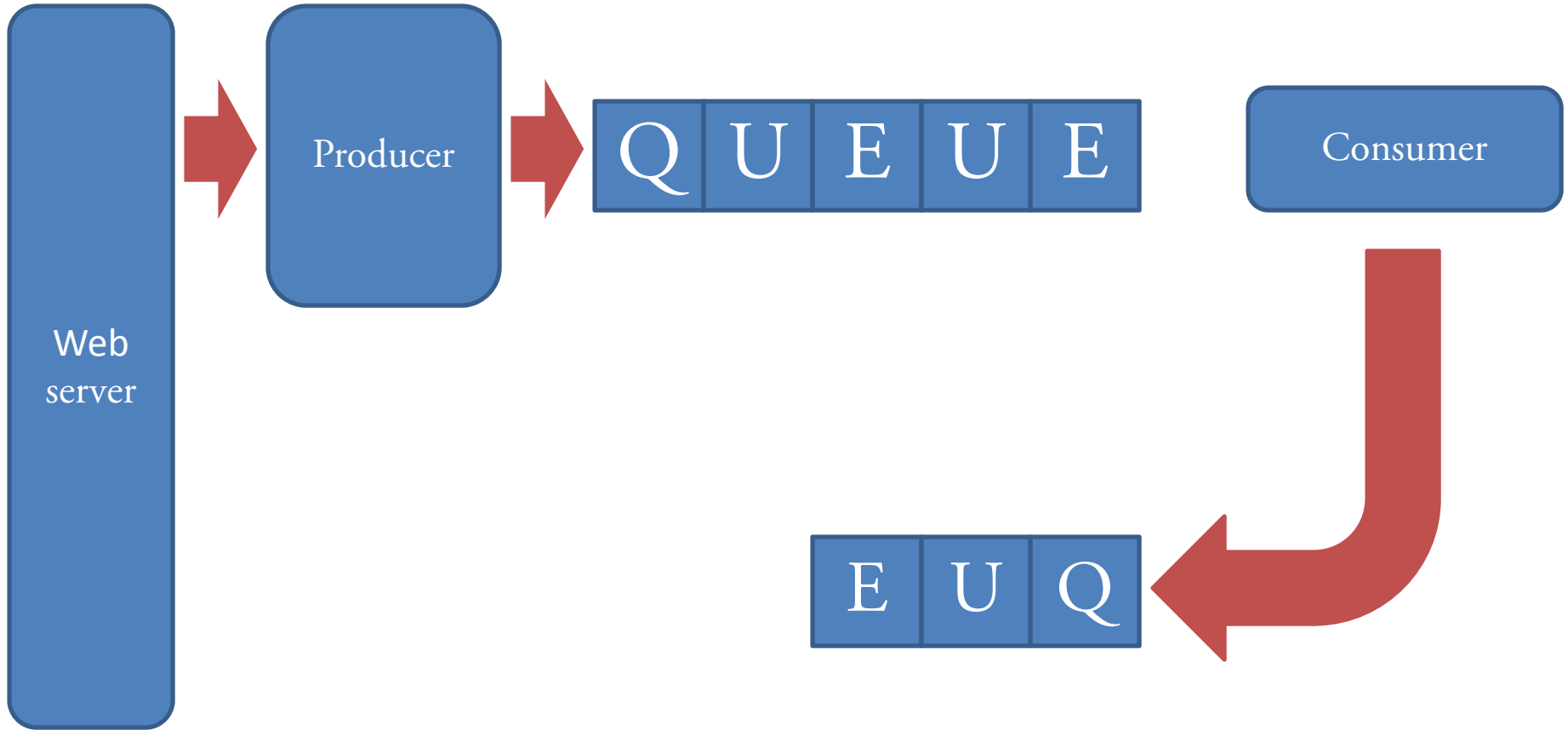
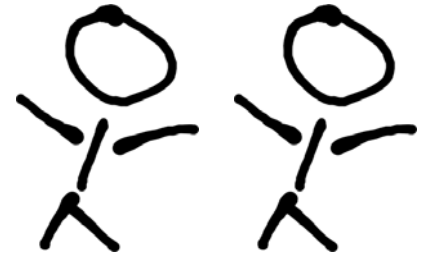


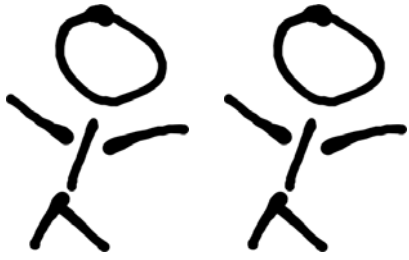
Scalable web service



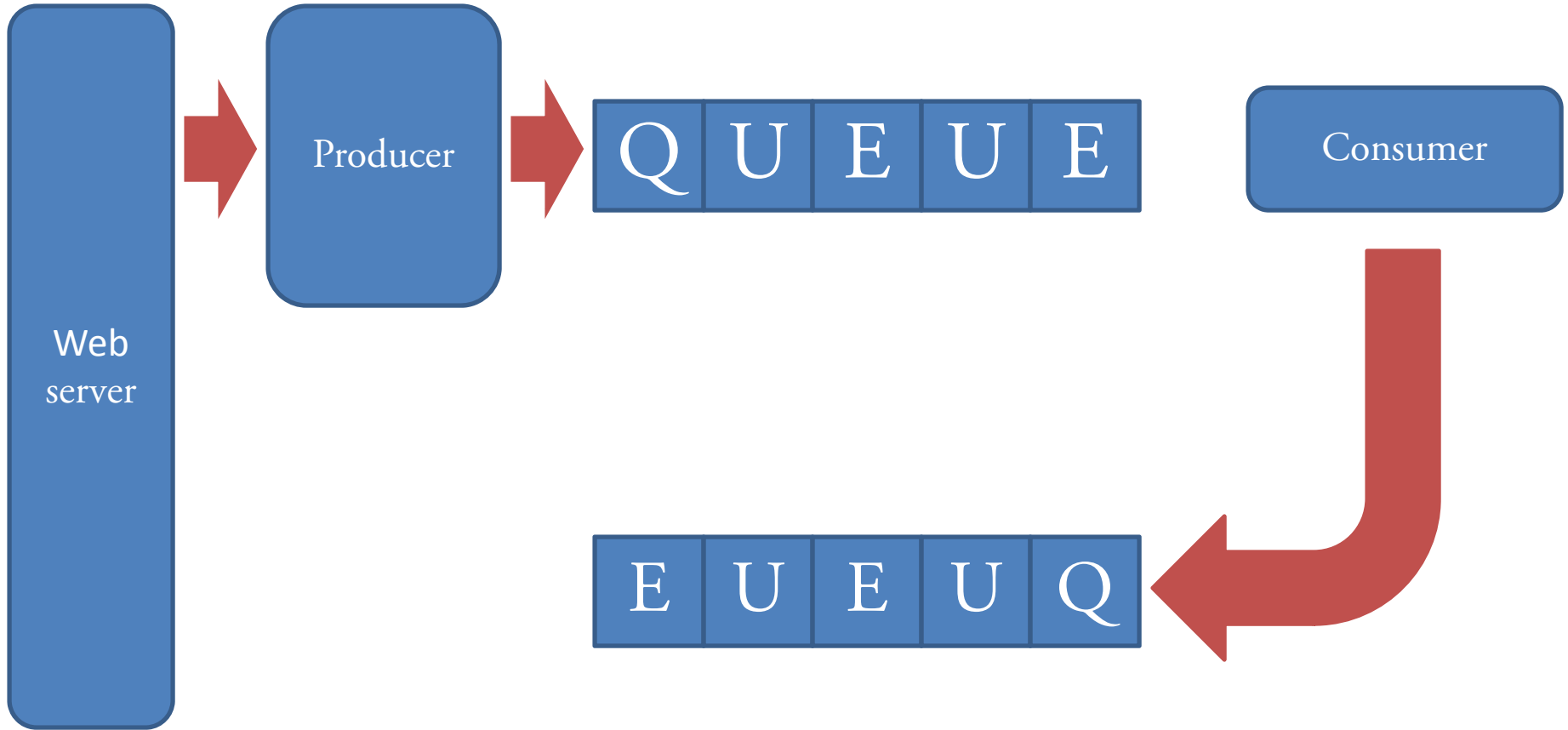
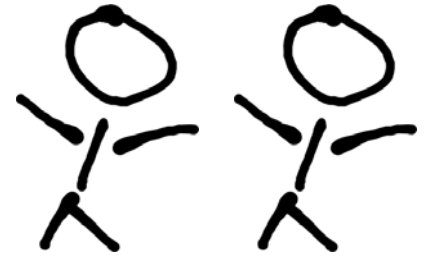


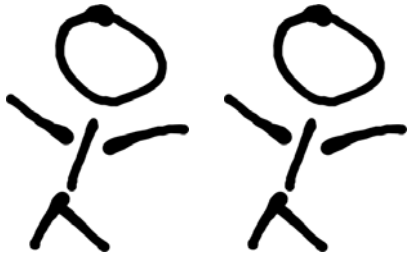
Scalable web service



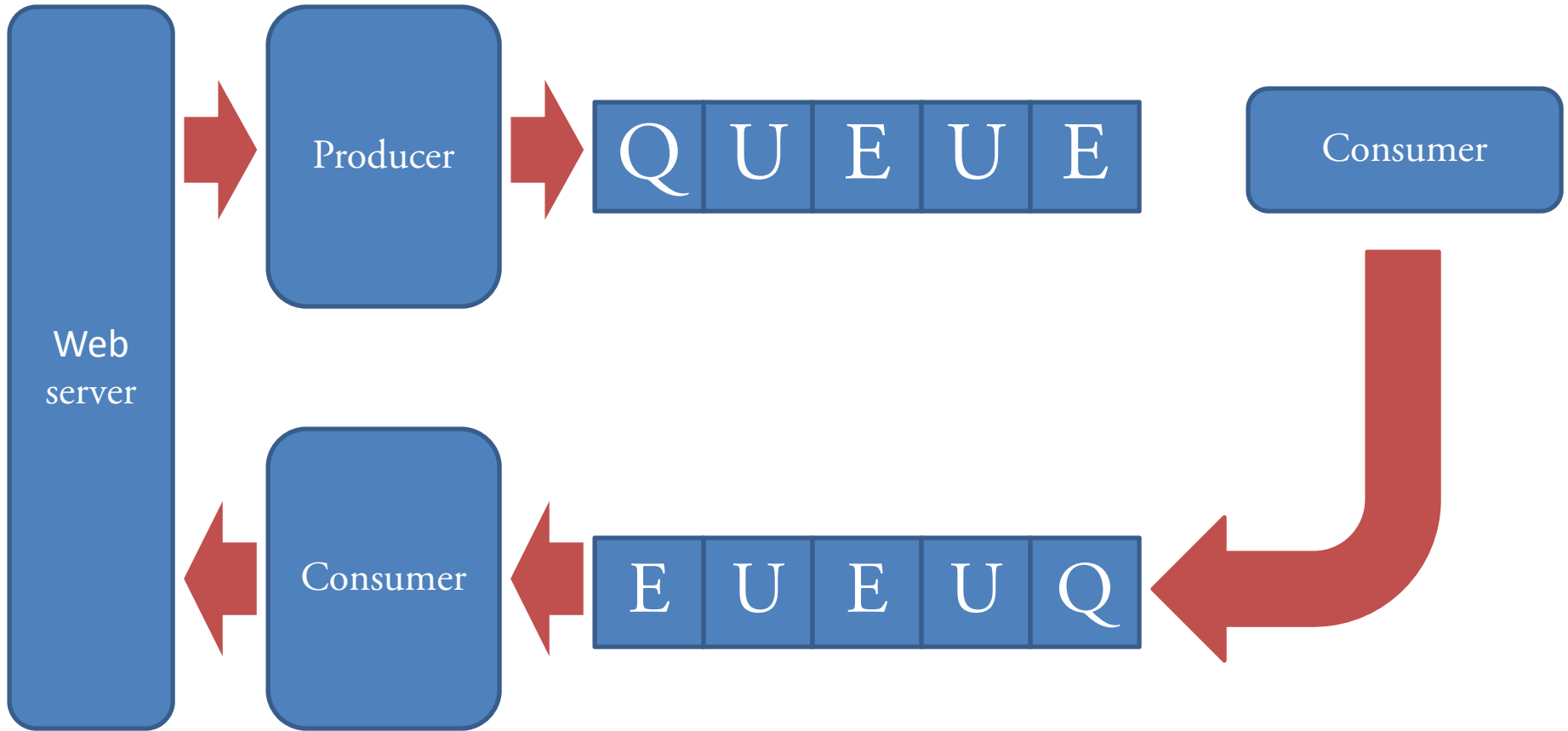
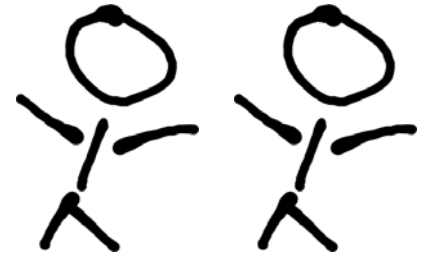


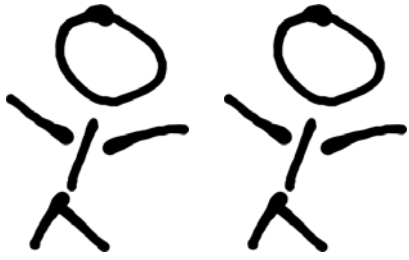
Scalable web service



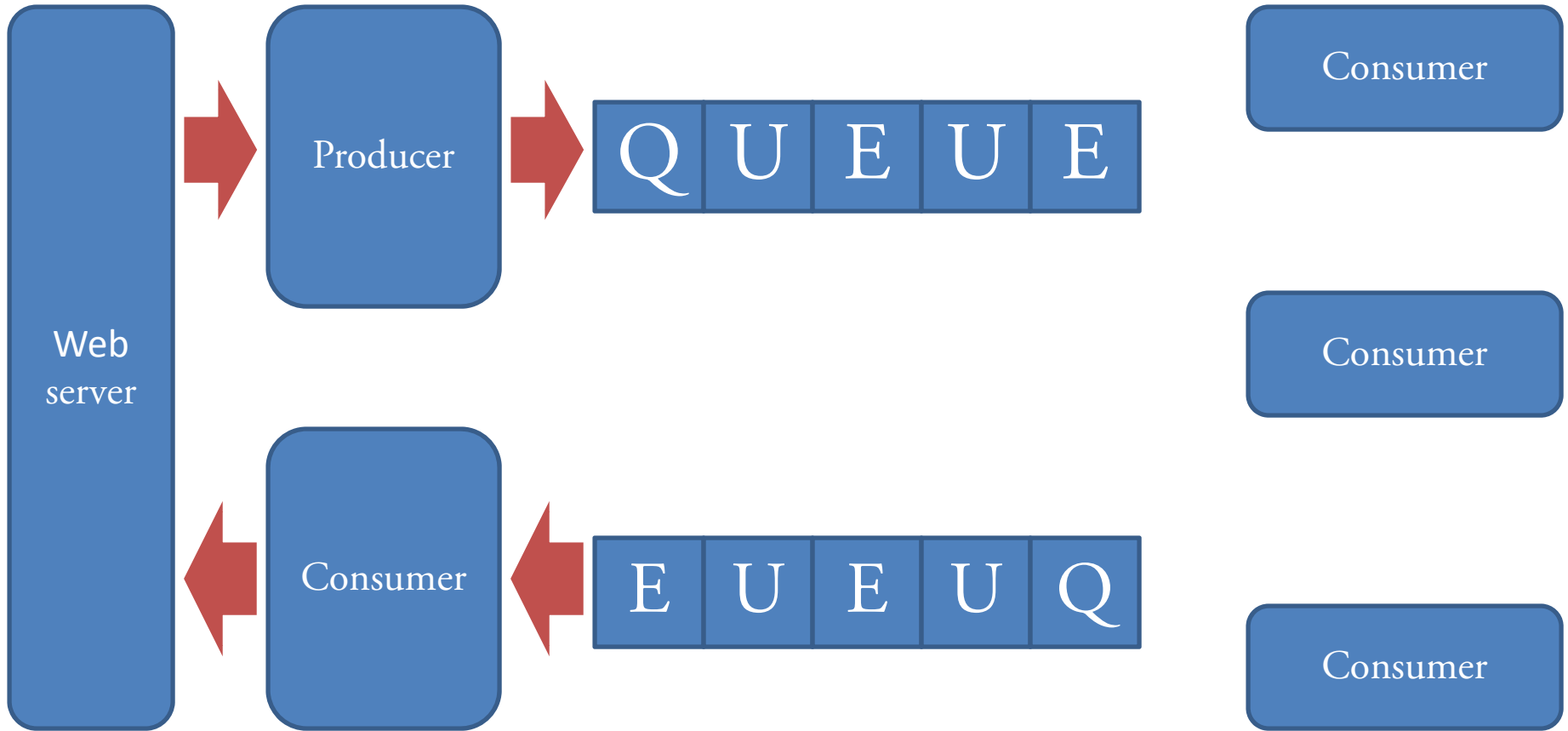
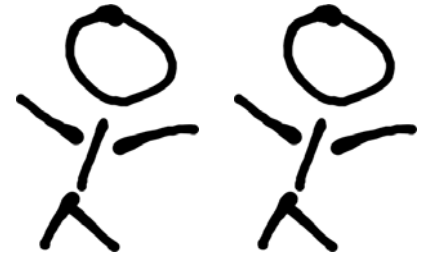


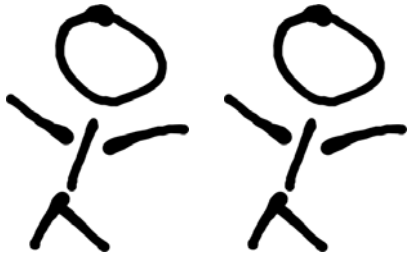
Scalable web service



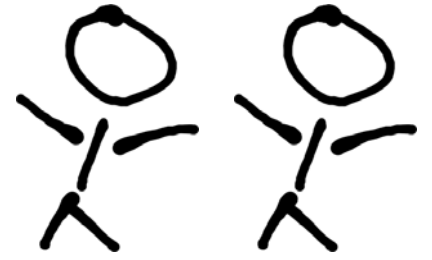


Scalable web service





Scalable web service



Put message on queue

Get message from queue

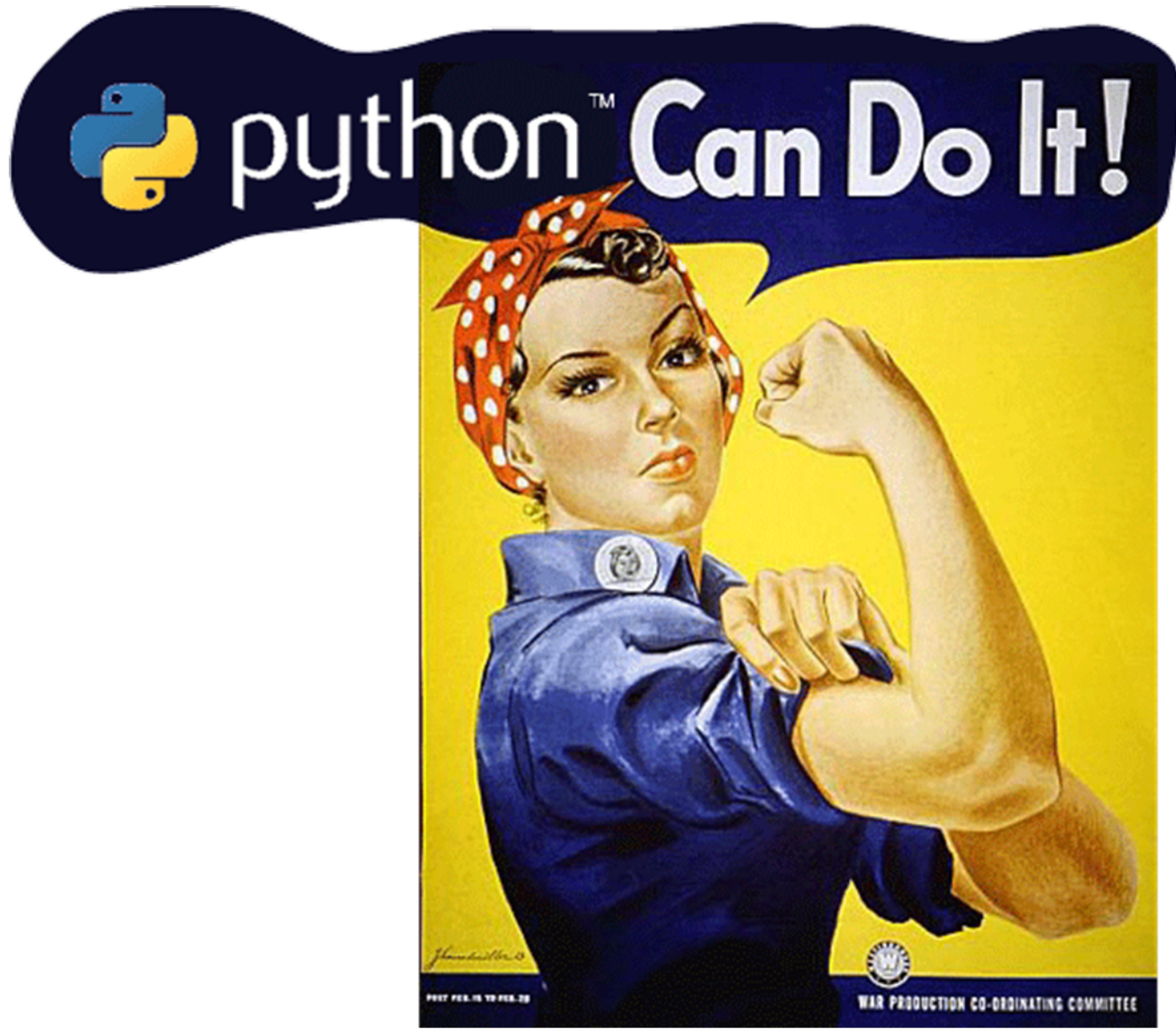
Get data from storage

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 Click on houses in satellite images	
Method	Machine learning	Image recognition
Command line	subprocess Lush	osgeo GDAL Generators
Scalable web	Pylons SQLAlchemy amqp RabbitMQ	

Take-away message



Question	Where should we build infrastructure?		
Manual	Spreadsheets	Java	Desktop GIS
Method	Mathematical modeling	Geospatial optimization	Visualization
Command line	numpy scipy	shapely geos	geojson openlayers osgeo proj4
Scalable web	Pylons SQLAlchemy amqp 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
Request a topic

invisibleroads.com

Links

Modi Research Group
Earth Insitute
Columbia University

modi.buildafrica.org

Computational and Biological Learning Lab
Courant Insitute of Mathematical Sciences
New York University

cs.nyu.edu/~yann

Tutorials & Workshops

invisibleroads.com