Data Analysis & Visualisation

By Quentin Bramas – ProgRes 2017

Complex JBotSim Example

We want to know, how much time it takes to transmit a message between two moving nodes:

- Assuming direct transmission
- Assuming at most k intermediate nodes

Complex JBotSim Example

To have a good overview of the performance of such transmissions, we can select at regular interval a source node and a destination node, and see how much time it takes to transmit a message from the source to the destination for various k.

Complex JBotSim Example

First we need to create a new class of Node.

Each node transmit a set of messages.

When a new message is received, it is added to the set to be forwarded.

```java
public class MovingNode extends  Node {

    private HashMap<Integer, NodeMessage> currentMessages = new HashMap<>();
    static final public int MAX_RETRANSMISSION = 5;

    @Override
    public void onClock() {
        for (NodeMessage m: currentMessages.values()) {
            if(m.destination != this)
                sendAll(m);
        }
    }

    public void startTransmissionTo(NodeMessage message) {
        currentMessages.put(message.id, message);
    }
    public void deleteMessage(NodeMessage message) {
        currentMessages.remove(message.id);
    }
    @Override
    public void onMessage(Message message) {
        NodeMessage m = new NodeMessage((NodeMessage)message.getContent(), 1);
        if(NodeMessage.isDone.test(m)) return;

        if(!currentMessages.containsKey(m.id) ||
            currentMessages.get(m.id).hop > m.hop) {

            if(m.hop < MAX_RETRANSMISSION) currentMessages.put(m.id, m);
            if(m.destination == this) {
                NodeMessage.onReceived.accept(m);
            } else {
                setColor(Color.gray);
            }
        }
    }
}
```

The messages keep track of the source, the destination, the starting time, and the number of hop

```
final public class NodeMessage {
   final public MovingNode source;
   final public MovingNode destination;
   final public int hop;
   final public int id;
   static private int lastId = 0;

   int startClock;
   static public Consumer<NodeMessage> onReceived;
   static public Predicate<NodeMessage> isDone;

   public NodeMessage(NodeMessage m, int increment) {
       source = m.source;
       destination = m.destination;
       hop = m.hop + increment;
       startClock = m.startClock;
       id = m.id;
   }
   public NodeMessage(MovingNode s, MovingNode d, int c) {
       source = s;
       destination = d;
       hop = 0;
       startClock = c;
       id = ++lastId;
   }
}
```

Complex JBotSim Example

Then a ClockListener orders at regular each node to send a message to another node selected at random

```
void startAllTransmission () {
    System.out.println(tp.getTime()+": start new transmission" );
    for(MovingNode n : nodes) {
        int j = (int)Math.floor(Math.random()*(nodes.length-1));
        if(j >= n.getID()) j++ ;

        MovingNode n2 = nodes[j];

        NodeMessage data = new NodeMessage(n, n2, tp.getTime());
        n.startTransmissionTo(data) ;
    }
}

public void onClock() {
    //startRandomNewTransmission();
    if(tp.getTime() % 500 == 0) startAllTransmission() ;
}
```

Complex JBotSim Example

Then, when a message is received we save the data in a log file

```java
static MovingNode[] nodes;
static Topology tp = new Topology();
static HashMap<Integer, Integer> best = new HashMap<>();
static PrintWriter log;

public TransmissionPerf() {

    NodeMessage.onReceived = this;
    NodeMessage.isDone = (m) -> {
            return best.containsKey(m.id) && best.get(m.id) == 1;
        };
}
public void accept(NodeMessage message) {
    int duration = (tp.getTime() - message.startClock);

    if(!best.containsKey(message.id))
         best.put(message.id, MovingNode.MAX_RETRANSMISSION);
    for(int i = message.hop; i < best.get(message.id); ++i) {
        log.println(message.startClock+","+i+","+duration);
    }
    log.flush();

    if(message.hop == 1 && best.get(message.id) > 1) {
        for(MovingNode n: nodes) {
            n.deleteMessage(message) ;
            n.setColor(Color.black);
        }
    }
    best.put(message.id, message.hop);
}
```

Complex JBotSim Example

The movements of the nodes are decided by another object. For instance an object that moves all the nodes according to the random waypoint model

```java
public class RandomWayPoint implements ClockListener, StartListener{
   private Point2D.Double target;
   private Topology tp;
   private Point2D.Double[] targets;
   private java.lang.Class<? extends Node> nodeClass;
   final static double SPEED = 1;
   public RandomWayPoint(Topology tp, java.lang.Class<? extends Node> nodeClass, int nbNode) {
       //Create nbNodes nodes
   }
   @Override
   public void onStart() {
       targets = new Point2D.Double[tp.getNodes().size()];
       int i=0;
       for(Node n: tp.getNodes()) {
           targets[i] = new Point2D.Double(Math.random() * 400, Math.random() * 400);
           n.setDirection(targets[i]);
           ++i;
       }
   }
   @Override
   public void onClock() {
       int i=0;
       for(Node n: tp.getNodes()) {
           double d = SPEED;
           if (targets[i].distance(n.getLocation()) <= d) {
               n.setLocation( targets[i]);
               d -= targets[i].distance(n.getLocation()) ;
               targets[i] = new Point2D.Double(Math.random() * 400, Math.random() * 400);
               n.setDirection(targets[i]);
           }
           n.move(d) ;
           ++i;
       }
   }
}
```

Data Analysis & Presentation

Data

r 0.000125 ns3::WifiMacHeader (MGT_BEACON ToDS=0, FromDS=0, MoreFrag=0, Retry=0, MoreData=0 Duration/ID=0us, DA=ff:ff:ff:ff:ff:ff, SA=00:00:00:00:00:0b, BSSID=00:00:00:00:00:0b, FragNumber=0, SeqNumber=0) ns3::MgtProbeResponseHeader (ssid=Amr, rates=[*6mbs 9mbs *12mbs 18mbs *24mbs 36mbs 48mbs 54mbs], HT Capabilities=0|0|0|0 , VHT Capabilities= 0|0) ns3::WifiMacTrailer ()

r 0.000256 ns3::WifiMacHeader (MGT_ASSOCIATION_REQUEST ToDS=0, FromDS=0, MoreFrag=0, Retry=0, MoreData=0 Duration/ID=60us, DA=00:00:00:00:00:0b, SA=00:00:00:00:00:03, BSSID=00:00:00:00:00:0b, FragNumber=0, SeqNumber=0) ns3::MgtAssocRequestHeader (ssid=Amr, rates=[6mbs 9mbs 12mbs 18mbs 24mbs 36mbs 48mbs 54mbs], HT Capabilities=0|0|0|0 , VHT Capabilities= 0|0) ns3::WifiMacTrailer ()

Data Formatting

Syntactic Modification required by Modeling Tools:

- Reordering of the attributes or records.
- Changes related to the constraints of the modeling tools:
  - Removing comma or tabs
  - Trimming strings to maximum allowed number of characters
  - Replacing special characters with allowed set of special characters.

Data

0.001000	72.565480	0.000435	0.015116	0.018278	1.209130
0.002000	72.520960	0.000870	0.015110	0.021045	1.392828
0.003000	72.476440	0.001305	0.015103	0.023514	1.556945
0.004000	72.431920	0.001741	0.015096	0.025747	1.705559
0.005000	72.387400	0.002176	0.015090	0.027789	1.841619
0.006000	72.342880	0.002612	0.015083	0.029673	1.967315
0.007000	72.298360	0.003047	0.015076	0.031423	2.084319
0.008000	72.253840	0.003482	0.015069	0.033061	2.193931
0.009000	72.209320	0.003918	0.015063	0.034602	2.297184
0.010000	72.164800	0.004354	0.015056	0.036058	2.394906
0.011000	72.120280	0.004789	0.015049	0.037439	2.487776

Data Presentation

Technical Simulation Results

A language that non technical people can understand

"a visual format that presents information systematically, so user can draw valid conclusions and take needed action"

Data Presentation

- Three important criteria: accuracy, conciseness, and understandability
- Researchers should always present their data in ways that most accurately represent the data

- What types of data presentation formats do you know?

- How are they different?

Charts & Graphs

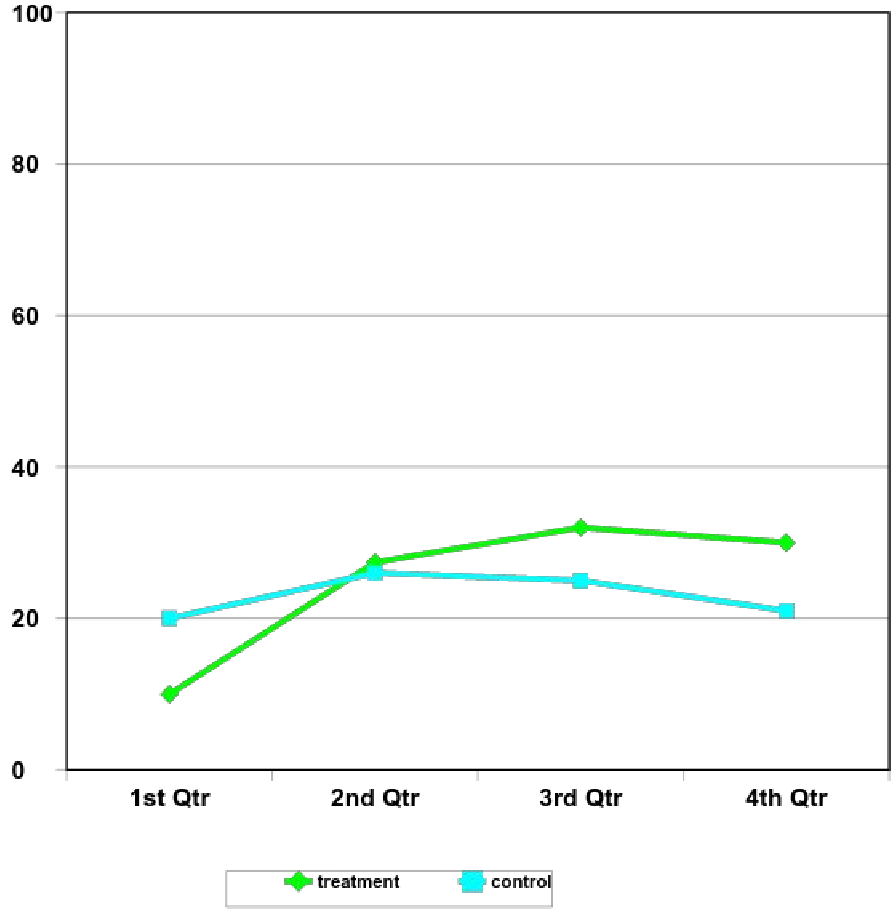
- Use a bar graph or pie chart if the variable has a limited number of discrete values
  - Nominal or ordinal measures

- Histograms and frequency curves are best for interval/ratio measures

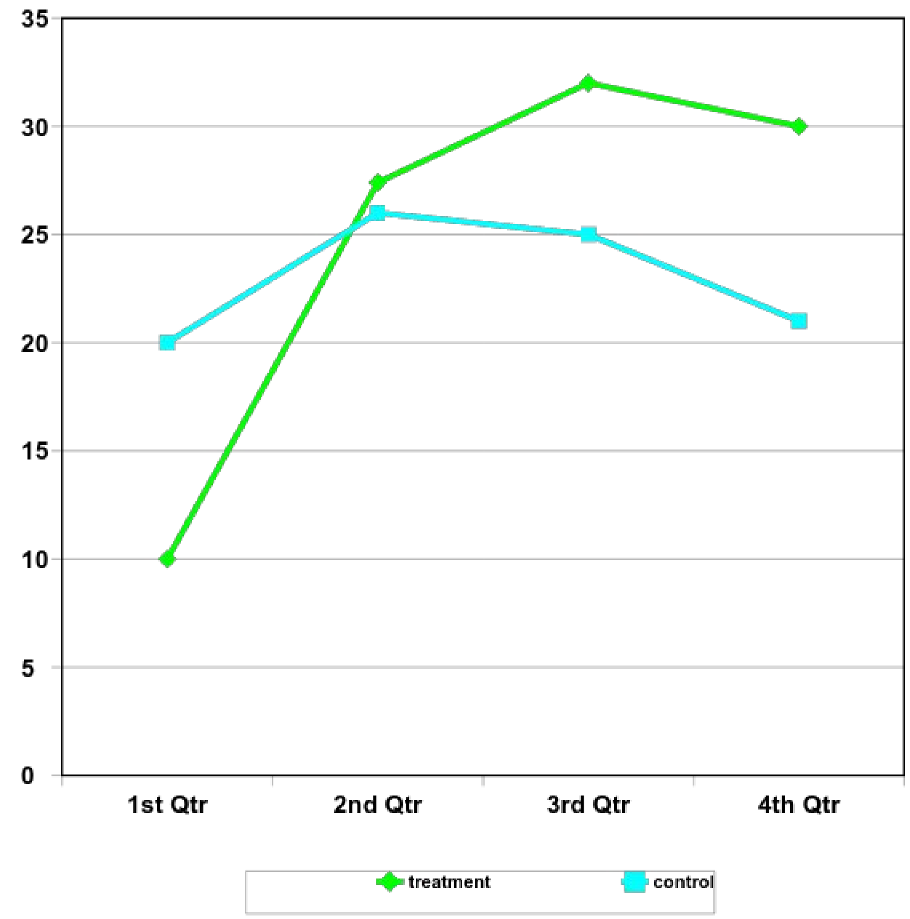
– Line graphs are useful for showing achievement gap data, as the gaps are evident

Visual Display of Same Data

Poor



Better



Scientific tools for Python

Scientific tools for Python

- Extra features required:
  - Manipulate and process data fast
  - libraries of reliable, tested scientific functions
  - Communicate results: produce figures for reports or publications, write presentations.

Numpy

Matplotlib

SciPy

Pandas

Numerical Python

- NumPy is at the core of nearly every scientific computer Python application or module.
- Numpy offers a matlab like capabilities:
  - a powerful N-dimensional array object (constructor, slicing, reshaping…)
  - LinearAlgebra Module: (Solvers, Eigenvalue, Fourier transform, inverse…)
  - sophisticated (broadcasting) functions

Numerical Python

```
>>> import numpy
```

- Arrays and Constructors

```
>>> a = zeros((3,3),Float)
>>> print a
[[0.,0.,0.],
 [0.,0.,0.],
 [0.,0.,0.]]
>>> print a.shape
(3,3)
>>> reshape(a,(9,)) # could also
use a.flat
>>> print a
[0.,0.,0.,0.,0.,0.,0.,0.,0.]
```

Numerical Python

```
>>> from LinearAlgebra import *
```

- Linear Algebra Function

```
>>> a = zeros((3,3),Float) +
2.*identity(3)
>>> print inverse(a)
[[0.5, 0., 0.],
 [0., 0.5, 0.],
 [0., 0., 0.5]]
>>> print determinant(inverse(a))
0.125
>>> print diagonal(a)
[0.5,0.5,0.5]
>>> print diagonal(a,1)
[0.,0.]
```

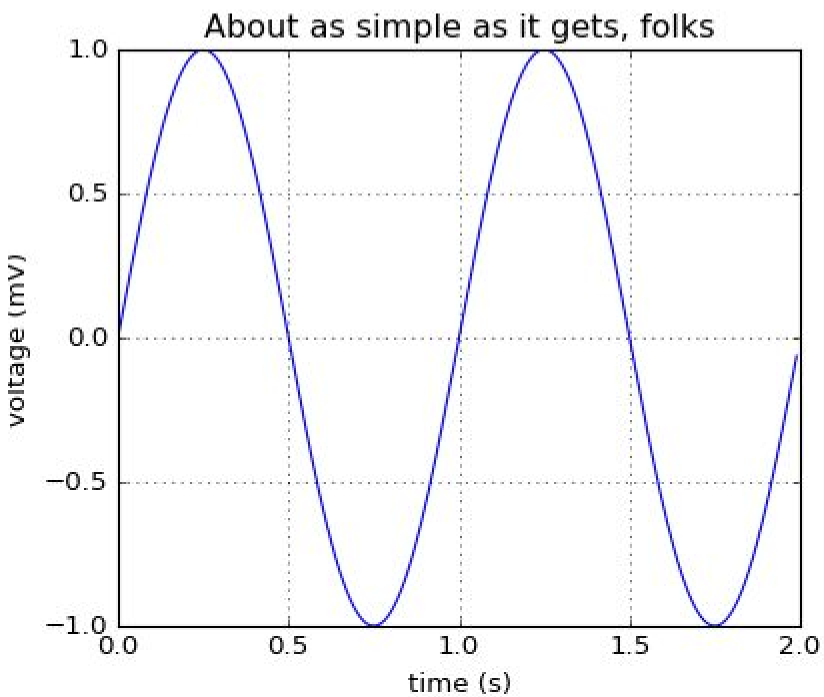
Numerical Python

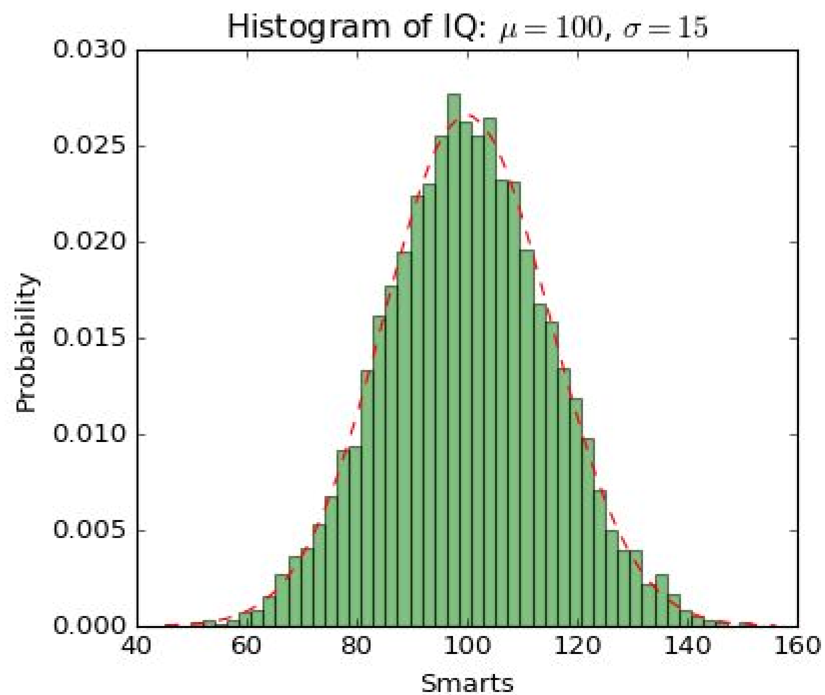
- Array operation

```
>>> a = array([[1.0, 2.0], [4.0, 3.0]])
>>> print a
[[ 1. 2.]
 [ 3. 4.]]

>>> a.transpose()
array([[ 1., 3.],
       [ 2., 4.]])

>>> inv(a)
array([[-2. , 1. ],
       [ 1.5, -0.5]])

>>> u = eye(2) # unit 2x2 matrix; "eye" represents "I"

>>> u
array([[ 1., 0.],
       [ 0., 1.]])

>>> j = array([[0.0, -1.0], [1.0, 0.0]])

>>> dot (j, j) # matrix product
array([[-1., 0.],
       [ 0., -1.]])
```

Matplotlib

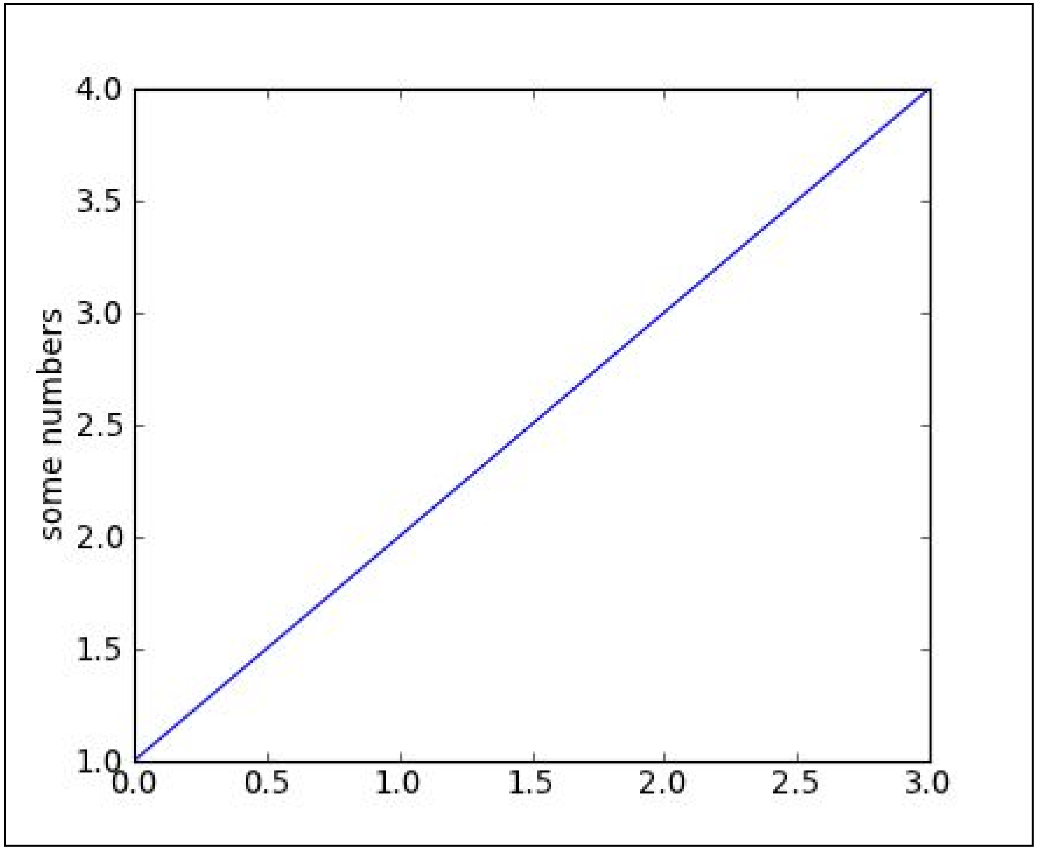
- Matplotlib is a python 2D plotting library





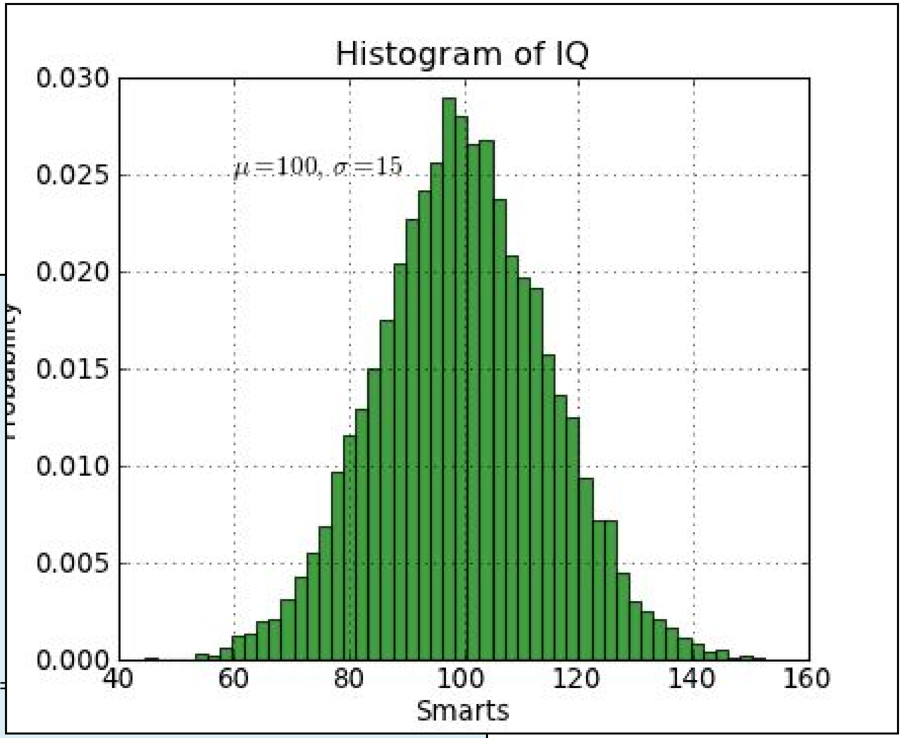
Matplotlib

```
import matplotlib.pyplot as plt
```

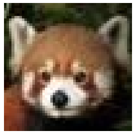



```
plt.plot([1,2,3,4])
plt.ylabel('some numbers')
plt.show()
```

Matplotlib

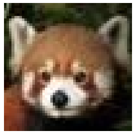



```
import numpy as np
import matplotlib.pyplot as plt

mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)

# the histogram of the data
n, bins, patches = plt.hist(x, 50, normed=
alpha=0.75)

plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title('Histogram of IQ')
plt.text(60, .025, r'$\mu=100,\ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
```



Pandas: Python Data Analysis Library

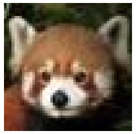
- Rich data manipulation to built on top of Numpy
- Fast and intuitive data structure
- Very flexible import/export data
- Easy visualization based on matplotlib



Pandas: **Pan**el **Da**ta **S**ystem

Series

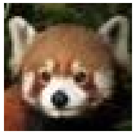
- One dimensional array-like object
- Contain an array of data (any Numpy type)
- Has an associated array of data labels



Pandas: **Pan**el **Da**ta **S**ystem

Series

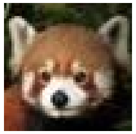
```
import pandas as pd
>>>s = pd.Series(np.random.randn(5), index=['a', 'b', 'c', 'd',
'e'])
>>>s
a -2.7828
b 0.4264
c -0.6505
d 1.1465
e -0.6631
dtype: float64
>>>s.reindex([‘e', 'b', 'c', 'd', ‘a'])
>>>s
e -2.7828
b 0.4264
c -0.6505
d 1.1465
a -0.6631
dtype: float64
```



Pandas: **Pan**el **Da**ta **S**ystem

DataFrame

- Two dimensional data structure
- Support heterogeneous columns
- The most commonly used pandas object
- Like series DataFrame accepts many kind of input



Pandas: **Pan**el **Da**ta **S**ystem

DataFrame

```
>>>d = {'one' : pd.Series([1., 2., 3.], index=['a', 'b', 'c']),
'two' : pd.Series([1., 2., 3., 4.], index=['a', 'b', 'c', 'd'])}

>>>df = pd.DataFrame(d)
>>>df
      one two
a 1 1
b 2 2
c 3 3
d NaN 4
>>>pd.DataFrame(d, index=['d', 'b', 'a'])
       one two
d NaN 4
b 2 2
a 1 1
```

Complex JBotSim Example

We want to see the duration of the transmission, for various k, depending on the time

```
import pandas
from collections import OrderedDict
import matplotlib.pyplot as plt

data = []
with open('perf.log', 'r') as logfile:
    for l in logfile:
        t, l, d = [int(x) for x in l.split(',')]
        data.append((t ,l,d))
    data = sorted(data)
```

Complex JBotSim Example

We list all the duration associated with messages transmitted at a given time, and for a given k. Then compute the mean value of those durations:

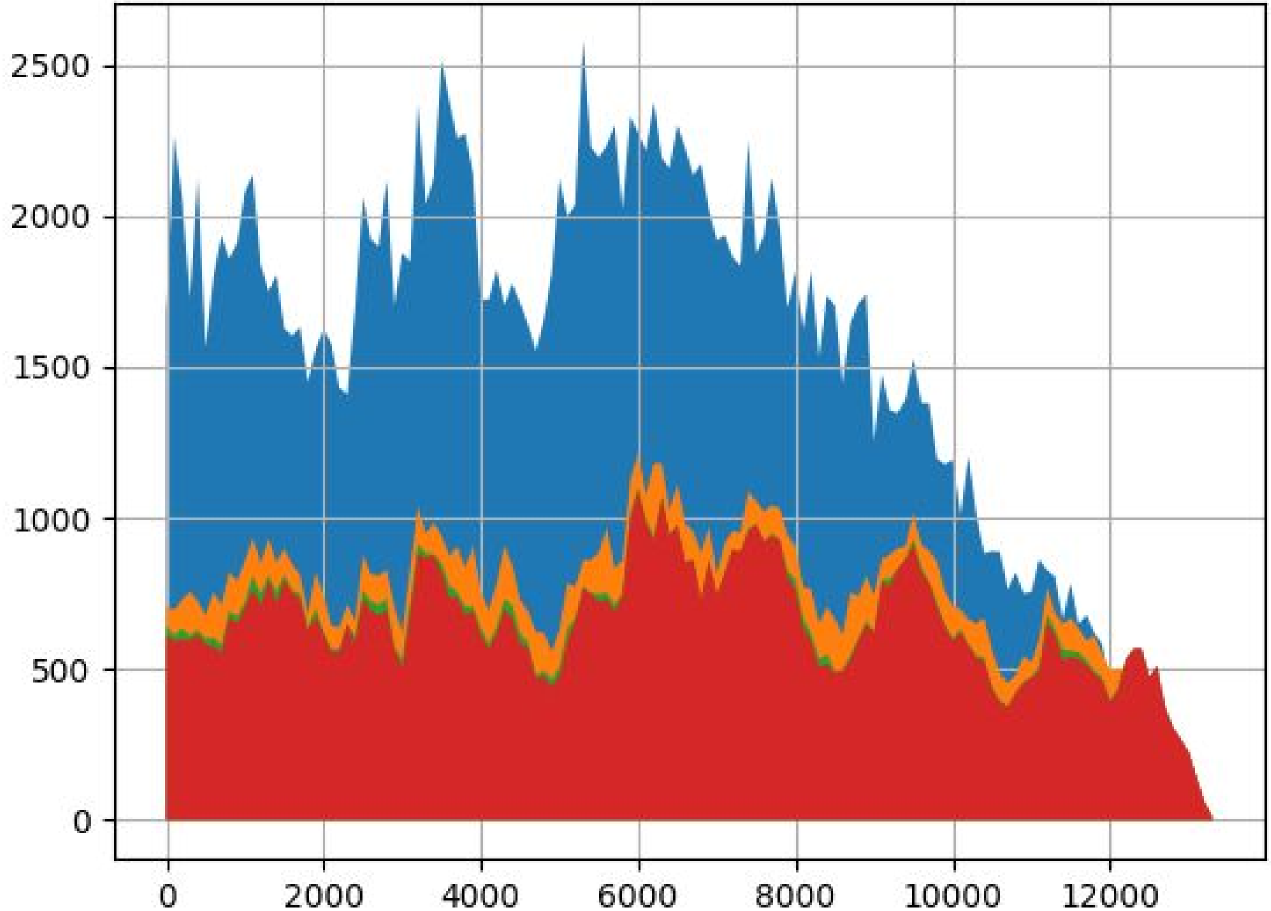
```
window = 100
delay = {}
for i in data:
   t = i[0] - (i[0]%window)
   if not i[1] in delay:
       delay[i[1]] = {}
   if not t in delay[i[1]]:
       delay[i[1]][t] = []
   delay[i[1]][t].append(i[2])


for lk, l in delay.items():
   for t in l.keys():
       l[t] = pandas.Series(l[t]).mean()
```

Complex JBotSim Example

Then for each k we draw a graph

```
fig, ax = plt.subplots()
for i in delay.keys():
   o = OrderedDict(sorted(delay[i].items() , key=lambda t: t[0]))
   x = list(o.keys())
   y = list(o.values())
   x.append(max(x))
   y.append(0)
   x.append(0)
   y.append(0)
   ax.fill(x , y, zorder=i)
ax.grid(True, zorder=10)
plt.show()
```

Complex JBotSim Example



Complex JBotSim Example

We can then replace the random waypoint by a graph of contact from a real dataset

Complex JBotSim Example