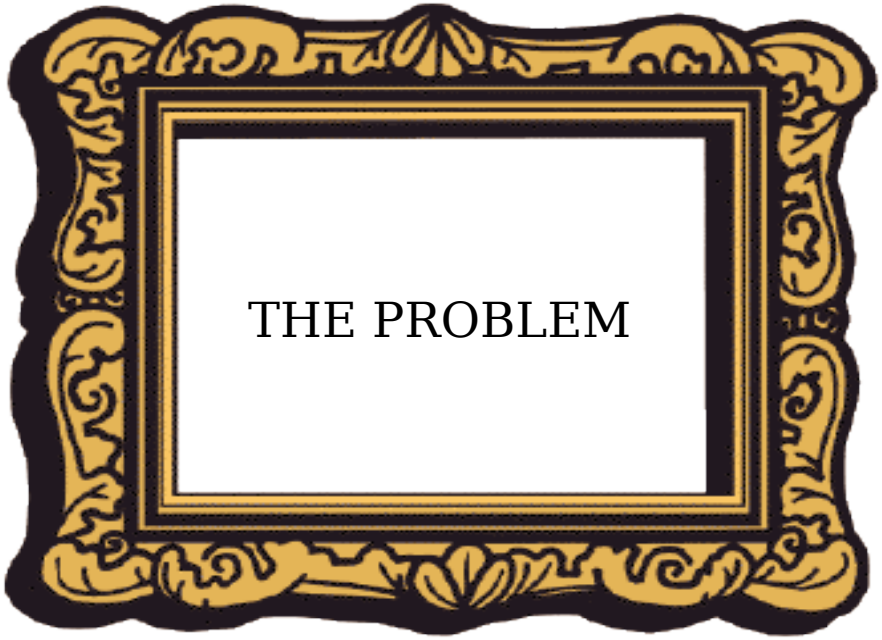
Introducing Broadwick

"A Computational Framework for Mathematical Modeling"

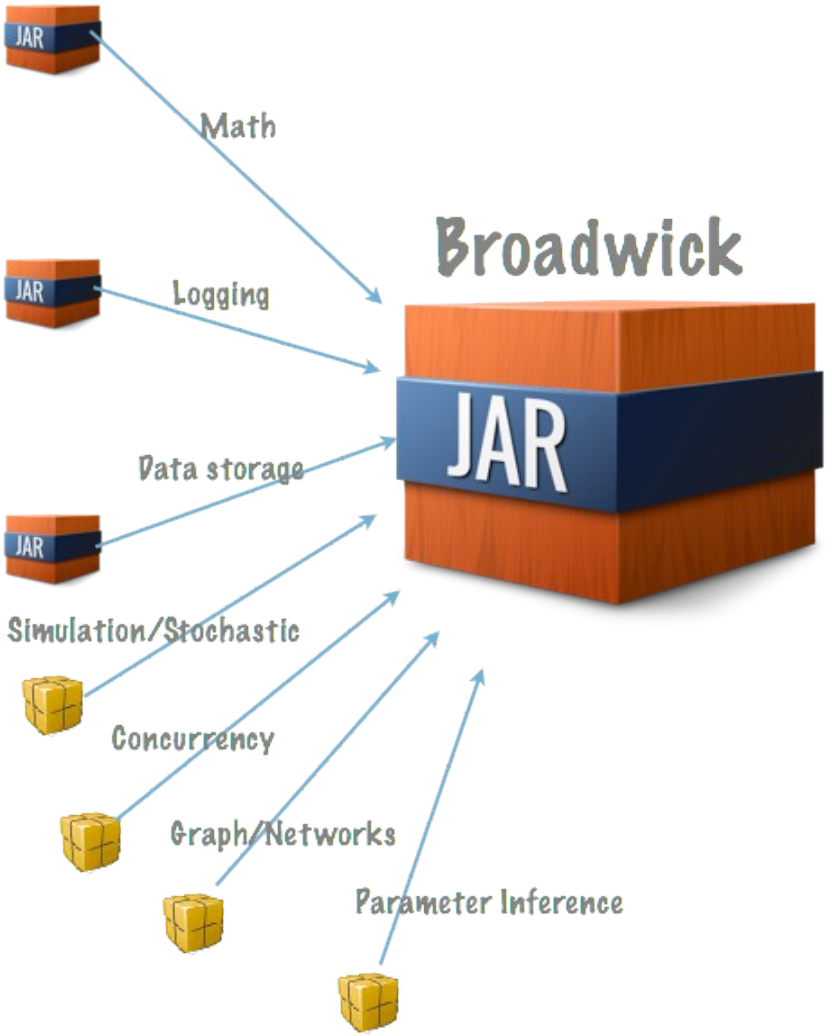


… want model continuity and not always start from scratch

… before [and especially during an outbreak] we want scientists thinking about the science NOT whether their code works.

But we still need talented people to do the research!

Broadwick is



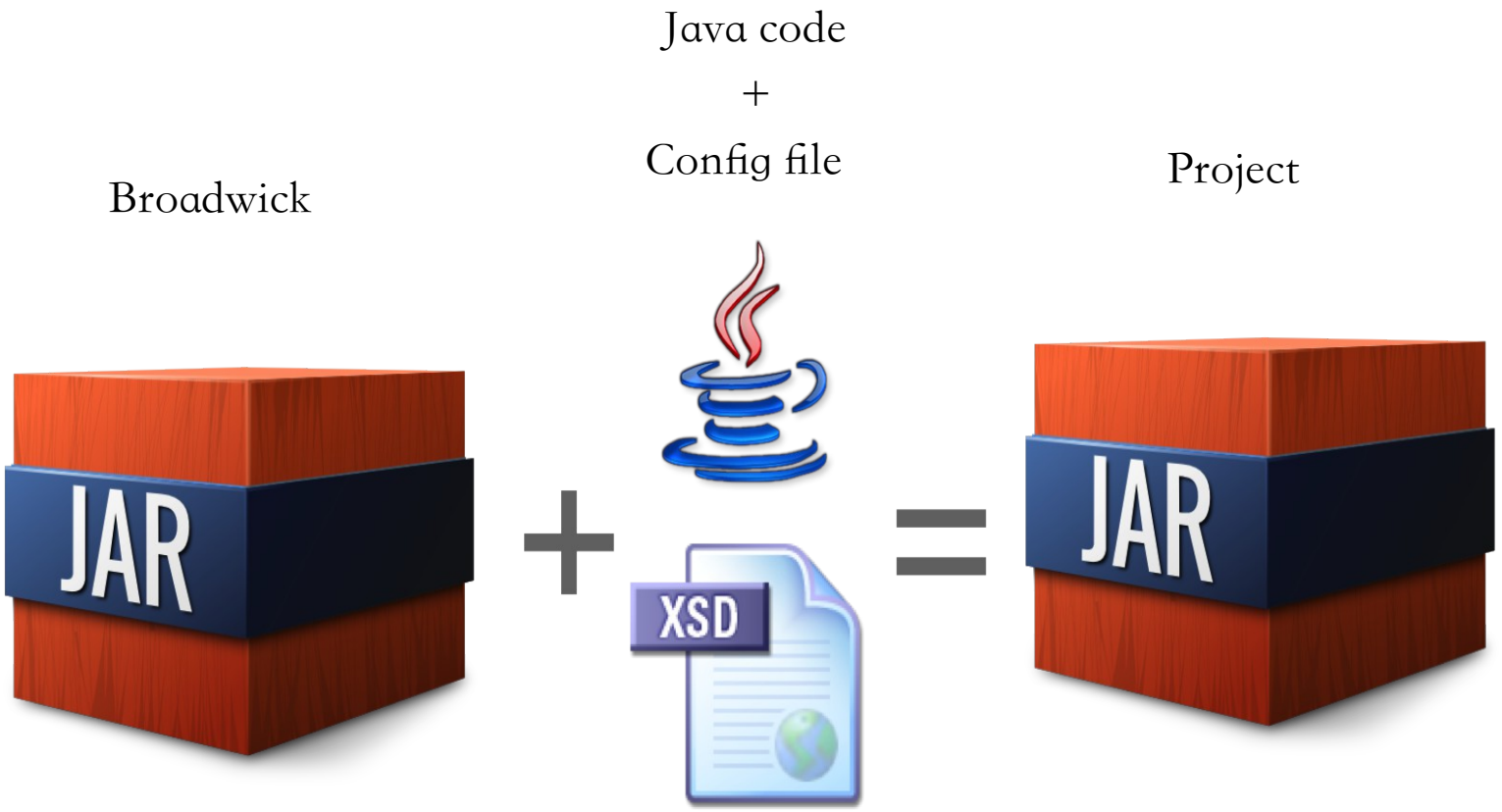
… is a framework for developing sophisticated epidemiological based mathematical models.

… essentially is a library consisting of several libraries and bespoke packages.

… contains many hooks that allow Broadwick to be used to develop models with the minimum of effort.

… is written in Java 7. Nothing else is imposed on the user (e.g. build system, IDE etc).

Using Broadwick



xsd file defines the configurable parameters in your model

… and is injected magically into the configuration file.

```
<xsd:schema xmlns:xjc="http://java.sun.com/xml/ns/jaxb/xjc"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:annotation>
  <xsd:documentation xml:lang="en"/>
</xsd:annotation>

<xsd:element name="model">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:elementname="numAgents" minOccurs="1"
                                   maxOccurs="1"
                                   type="xsd:int"/>
      <xsd:elementname="numSimulations" minOccurs="1"
                                        maxOccurs="1"
                                        type="xsd:int"/>

    </ xsd:sequence>
  </ xsd:complexType>
</ xsd:element>

</ xsd:schema>
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<project>
  <logs>
    <console>
      <level>info</level>
      <pattern>[%thread] %-5level %msg %n</pattern>
    </console>
    <file>
      <name>myModel.log</ name>
      <level>trace</level>
      <pattern>[%thread] %logger %-5level %msg %n</pattern>
      <overwrite>true</ overwrite>
    </file>
  </logs>
  <data>
    <datafiles>
      <PopulationFile>
        <name>data.csv</ name>
        <separator>,</ separator>
        <dateFormat>yyyy-MM-dd</ dateFormat>
      </PopulationFile>
    </datafiles>
  </data>
  <models>
    <model id="The name of my model">
      <classname>org.mymodel.MyModel</ classname>
      <numAgents>100</numAgents>
      <numSimulations>50</numSimulations>
    </model>
  </models>
</ project>
```

Little work is required from the modeller

- But a lot of functionality is gained

Creating a new model is easy …

```
package custom;

public class myModel extends broadwick.model.Model {

    @Override
    public void init() {
        throw new UnsupportedOperationException("Not supported yet.");
    }

    @Override
    public void run() {
        throw new UnsupportedOperationException("Not supported yet.");
    }

    @Override
    public void finalise() {
        throw new UnsupportedOperationException("Not supported yet.");
    }
}
```

Extend Broadwick's Model class

… and override the abstract methods.

… and much easier than writing ALL the code from scratch [again!]

Broadwick Contains …

Useful classes/packages you'd expect

math(factorial, sets operations, RNG), file i/o

A concurrency package

Can take advantage of multithreaded/distributed/grid computation with very little work/thought from the modeller.

Calculation packages

Stochastic Simulations, MCMC, Approx. Bayesian Computation

Very sophisticated models can be built relatively simply

Each calculation class contains intelligent defaults

```
simulation = new McModel () {

        @Override
        public McResults compute(Step step) {
            throw new UnsupportedOperationException("Not supported yet.");
        }
    };

final MonteCarlo mc = new MonteCarlo(simulation);
```

For example, the MonteCarlo class uses a Metropolis-Hastings algorithm for accepting proposals

But this can easily be customised....

```
mc.setAcceptor(new Acceptor() {
        @Override
        public boolean accept(final McResults oldResult,
                              final McResults newResult) {
            return newResult.getScore() > oldResult.getScore();
        }
    });
```

```
mc.setWriter(newFileOutput("myMcmc.dat"));
mc.run();
```

··· customising these defaults are [and should be] easy to do.

One size does not fit all after all !

What can Broadwick do for Epic?

off the shelf simulations would be readily adaptable for new diseases (e.g. what if Schmallenberg had become a more serious problem?)

implementation of parameter estimation routines directly allows for better prediction

data imputation - given the existing data (e.g. we know about impact on farms in SE England) what does that tell us about risk in Scotland?

Bang >> Buck !

Why Broadwick?

Models can be written quickly

- Code reuse
- verifiable

Off-the-shelf models can be written in advance of an outbreak.

- Plug'n'play
- Modified easily if needed

```
for u in `attendees`
do
   echo “Thank ”$u
done
```


Broadwick 1.0 is available to developers today
One more thing...