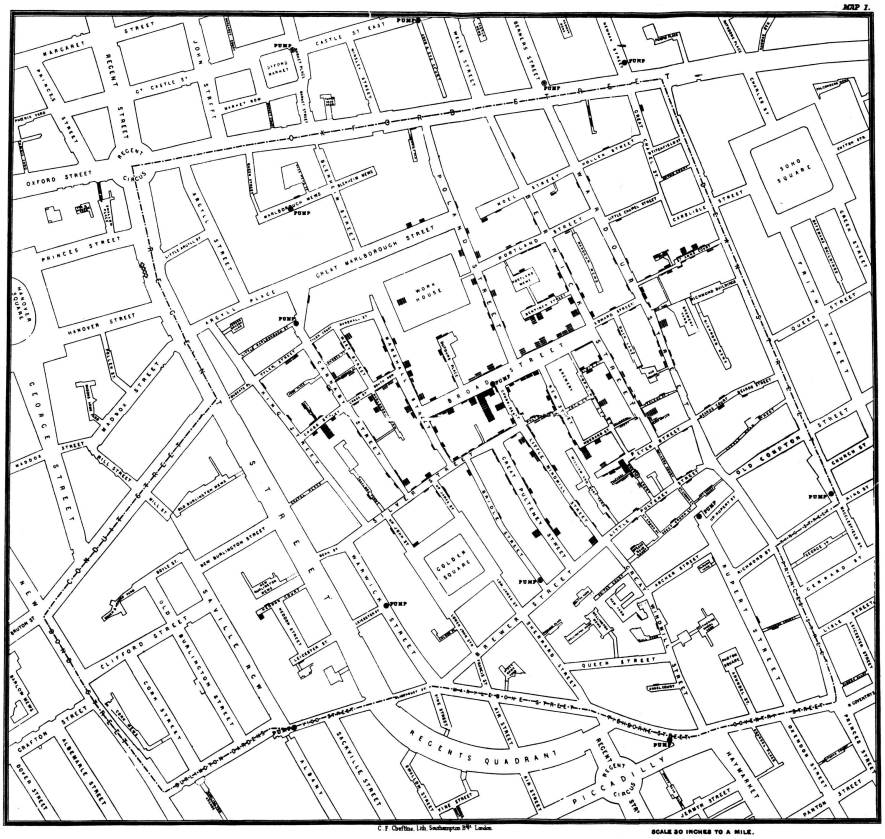
Broadwick Manual



Published by C.F. Cheffins, Lith, Southhampton Buildings, London, England, 1854 in Snow, John. On the Mode of Communication of Cholera, 2nd Ed, John Churchill, New Burlington Street, London, England, 1855.

(This image was originally from http://en.wikipedia.org/wiki/File:Snow-cholera-map-1.jpg)

Cover Picture.

A variant of the original map drawn by Dr. John Snow (1813-1858), a British physician who is one of the founders of medical epidemiology, showing cases of cholera in the London epidemics of 1854, clustered around the locations of water pumps.

Index

1 Introduction

Broadwick is a computational framework written in Java for epidemiological computation. It removes the boilerplate code so that users can be more productive in developing models.

1.1 License

Broadwick is released under the Apache 2 license.

2 Using Broadwick

Broadwick contains a set of packages that can be used as required. The framework is designed to be flexible and does not place any requirement on the user on how to use the framework e.g. the user is free to create a main() method and choose any method to incorporate project data.

It is possible to use the classes and packages of Broadwick without using the powerful framework, creating your own main() method and taking responsibility for reading data files and configuration items though this is not the recommended way of using Broadwick.

Broadwick contains an entry point that will process an xml configuration file and perform any preprocessing of data files before creating the user project (called a model in the framework) defined in the configuration file. The remainder of this chapter will describe how to extend Broadwick to incorporate a simple model and how Broadwick handles data files for movements, location and population data.

Using Broadwick is as easy as extending Broadwick's 'Model' class and packaging the compiled code as an executable jar containing Broadwick as a dependency. Any build tool can be used to do this and Broadwick does not impose any specific tool on the user. Extending the Model class allows Broadwick to manage the running of the project. As well as extending the Model class, an xsd file that defines the configuration items for a project can be 'injected' into the Broadwick configuration file that will map the xsd elements to java classes for easy reading of configuration items.

The Broadwick source contains a number of examples that extend the Model class and contain xsd definitions for inputs. The rest of this chapter will describe how run Broadwick in this manner.

To run a project using Broadwick on the command line a single command line argument ("-c") that specifies the name of the configuration file to use is required by Broadwick.

```
java -jar myProject.jar -c myProject.xml
```

A simplified outline of how Broadwick initialises itself is shown in Illustration 1.

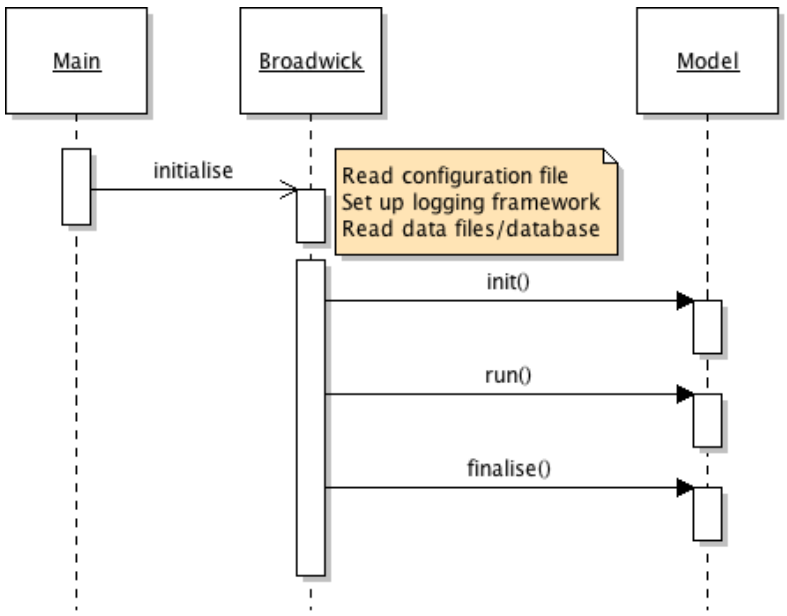


Illustration 1: Summary of actions performed by Broadwick on start-up.

2.1 Creating a Model

Creating a model in Broadwick is as simple as extending the broadwick.model.Model class and overriding the abstract methods for initiailising, running and tidying up the model as shown below.

```
package markovchain;
public class MarkovChainExample 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.");
    }
 }
```

Running this model requires adding a <model> section in the configuration file that specifies the full name of the class (including the package), e.g. “markovchain.MarkovChainExample” in the example above. If configuration items are required by the model, these should be specified in an xsd file which is added to this model section.

A description of the configuration file is outlined in the next section.

2.2 Configuration Files

```
<?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>debug</level>
         <pattern>[%thread] %-5level %msg %n</pattern>
         <overwrite>true</overwrite>
       </file>
   </logs>

   <models>
      <model id="Example Markov Chain model">
```

```
            <classname>markovchain.MarkovChainExample</classname>
        </model>
    </models>
</project>
```

2.3 Running Broadwick

3 Packages

4 Algorithms

4.1 Approximate Bayesian Computation (ABC)

4.2 Markov Chain Monte Carlo (MCMC) methods

```
simulation = new MySimulation();

final MonteCarlo mc = new MonteCarlo(simulation);
mc.setWriter(newFileOutput("myMcmc.dat"));
mc.run();
```

4.3 Stochastic Models

5 Appendix

5.1 Using Maven as a build tool

There is no requirement to use maven as a build tool but as Broadwick and it's examples are built using maven this section will give a brief outline of how maven is used to create a simple model.

Maven uses an xml file to describe the classes to be built as well as the dependencies, dynamically downloading required libraries as needed. It uses the 'convention over configuration' paradigm imposing the directory structure given in the table below.

Directory	Purpose
Project home	Contains the pom and all the subdirectories.
src/main/java	Contains the java source code for the project.
src/main/resources	Contains the xsd file for configuring the project.
src/test/java	Contains any [Junit or TestNG] test cases for the project.
src/test/resources	Contains resources necessary for testing.

Maven's equivalent to a makefile or Ant's build.xml is a 'project object model' which is stored in a pom.xml file. The pom file for the example used throughout this manual and which can be modified for custom models is given below. The <properties> section of the pom specifies the versions of the plugins and dependencies used in the remainder of the pom. Sonar is used by Broadwick to maintain code quality and the java files that are excluded from this analysis is also specifed in the properties section.

The onejar-maven-plugin plugin creates an executable jar for the project including the dependent jars.

The maven-compiler-plugin plugin specifies the version of Java required to build the project.

The maven-jaxb2-plugin plugin generates Java class files from the xsd definition in the configuration subsection.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                             http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>BroadwickExamples</groupId>
  <artifactId>BroadwickExamples</artifactId>
  <version>1.0</version>
  <packaging>jar</packaging>

  <name>BroadwickExamples</name>
  <url>http://maven.apache.org</url>
```

```
<prerequisites>
   <maven>3.0.0</maven>
</prerequisites>

<properties>
   <!-- Sonar exclusions. [CSV] List of files for sonar to ignore (this will
        be exclusively 3rd party files. -->
   <sonar.exclusions>markovchain/config/generated/*.java</sonar.exclusions>

   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
   <broadwick.version>1.0</broadwick.version>
   <exec-maven-plugin.version>1.2.1</exec-maven-plugin.version>
   <onejar-plugin.version>1.4.5</onejar-plugin.version>
   <maven-compiler-plugin.version>3.1</maven-compiler-plugin.version>
   <maven-jaxb2-plugin.version>0.8.3</maven-jaxb2-plugin.version>
</properties>

<pluginRepositories>
   <pluginRepository>
      <id>onejar-maven-plugin.googlecode.com</id>
      <url>http://onejar-maven-plugin.googlecode.com/svn/mavenrepo</url>
   </pluginRepository>
</pluginRepositories>
<dependencies>

   <dependency>
      <groupId>broadwick</groupId>
      <artifactId>broadwick</artifactId>
      <version>${broadwick.version}</version>
   </dependency>

</dependencies>

<build>
   <plugins>
      <!-- Package all the jars into a single executable jar -->
      <plugin>
         <groupId>org.dstovall</groupId>
         <artifactId>onejar-maven-plugin</artifactId>
         <version>${onejar-plugin.version}</version>
         <executions>
            <execution>
               <configuration>
                  <mainClass>broadwick.Broadwick</mainClass>
                  <attachToBuild>false</attachToBuild>
                  <classifier>onejar</classifier>
               </configuration>
               <goals>
                  <goal>one-jar</goal>
               </goals>
            </execution>
         </executions>
      </plugin>

      <plugin>
         <groupId>org.codehaus.mojo</groupId>
         <artifactId>exec-maven-plugin</artifactId>
         <version>${exec-maven-plugin.version}</version>
         <executions>
```

```
            <execution>
               <phase>install</phase>
               <goals>
                  <goal>java</goal>
               </goals>
               <configuration>
                  <executable>java</executable>
                  <mainClass>broadwick.Broadwick</mainClass>
                  <classpathScope>test</classpathScope>
                  <includePluginDependencies>true</includePluginDependencies>
                  <includeProjectDependencies>true</includeProjectDependencies>
                  <!--<commandlineArgs> - - port 9876</commandlineArgs>-->
               </configuration>
            </execution>
         </executions>
         <dependencies>
            <dependency>
               <groupId>broadwick</groupId>
               <artifactId>broadwick</artifactId>
               <version>0.0</version>
            </dependency>
         </dependencies>
      </plugin>

      <plugin>
         <groupId>org.apache.maven.plugins</groupId>
         <artifactId>maven-compiler-plugin</artifactId>
         <version>${maven-compiler-plugin.version}</version>
         <configuration>
            <source>1.7</source>
            <target>1.7</target>
         </configuration>
      </plugin>

      <!-- generate the java classes from the xsd definition -->
      <plugin>
         <groupId>org.jvnet.jaxb2.maven2</groupId>
         <artifactId>maven-jaxb2-plugin</artifactId>
         <version>${maven-jaxb2-plugin.version}</version>
         <configuration>
            <quiet>true</quiet>
            <verbose>false</verbose>
            <readOnly>true</readOnly>
            <arguments>-mark-generated</arguments>
            <removeOldOutput>false</removeOldOutput>
            <clearOutputDir>false</clearOutputDir>
            <forceRegenerate>true</forceRegenerate>
         </configuration>
         <executions>
            <execution>
               <id>markovchain-generate</id>
               <goals>
                  <goal>generate</goal>
               </goals>
               <configuration>
                  <schemaDirectory>src/main/resources/markovchain</schemaDirectory>
                  <schemaIncludes>
                     <include>markovchain.xsd</include>
                  </schemaIncludes>
```

```
                        <generatePackage>markovchain.config.generated</generatePackage>
                        <generateDirectory>${project.build.directory}/generated-
sources/xjc</generateDirectory>
                    </configuration>
                </execution>

            </executions>
        </plugin>
    </plugins>
  </build>
</project>
```