TEST PLAN

June 20, 2019

Abstract

In this document we will go over all tests, including unit tests as well as scenario tests, documenting their name along with their purpose and how we intend to check this.

Benjamin Vandersmissen benjamin.vandersmissen@student.uantwerpen.be
Lars Van Roy lars.vanroy@student.uantwerpen.be
Evelien Daems evelien.daems@student.uantwerpen.be
Frank Jan Fekkes franciscus.fekkes@student.uantwerpen.be

Contents

1 Introduction

For the rest of this document we will give a short summary of every newly created test that will check if the new features work as we expected. These new additions include the two new contact types, being Daycare and PreSchool, new data formats, being JSON and HDF5 as well as global schemes that will be used by all students. There will also be a visual output for the data via QT and an improvement to the generation of the population by factoring in new parameters that will make the generation of the population more diverse, so that we can check even more parameters for their influence on the spread of diseases. Every student will write the tests related to their code that they wrote. Scenario tests may overlap and those will be written by the student that felt the need for them.

2 Daycare and PreSchool

2.1 Generators

2.1.1 DaycareGenerator.ZeroLocationTest

What will it test? This test will check whether the DaycareGenerator can function correctly with no locations given.

How will we achieve this? We will achieve this by creating a geogrid with no locations and handing it over to the Generator, then checking whether the size of the geogrid is zero.

2.1.2 DaycareGenerator.OneLocationTest

What will it test? This test will check whether the DaycareGenerator can function correctly with one location given.

How will we achieve this? We will achieve this by creating a geogrid with one location and handing it over to the Generator, then comparing the amount of contactPools for the location with the expected amount of contactPools.

2.1.3 DaycareGenerator.MultipleLocationtest

What will it test? This test will check whether the DaycareGenerator can function correctly with multiple locations given.

How will we achieve this? We will achieve this by creating a geogrid with multiple locations and handing it over to the Generator, then comparing the amount of contactPools for each location with the expected amount of contact-Pools in that location.

2.1.4 PreSchoolGenerator.ZeroLocationTest

What will it test? This test will check whether the PreSchoolGenerator can function correctly with no locations given.

How will we achieve this? We will achieve this by creating a geogrid with no locations and handing it over to the Generator, then checking whether the size of the geogrid is zero.

2.1.5 PreSchoolGenerator.OneLocationTest

What will it test? This test will check whether the PreSchoolGenerator can function correctly with one location given.

How will we achieve this? We will achieve this by creating a geogrid with one location and handing it over to the Generator, then comparing the amount of contactPools for the location with the expected amount of contactPools.

2.1.6 PreSchoolGenerator.MultipleLocationtest

What will it test? This test will check whether the PreSchoolGenerator can function correctly with multiple locations given.

How will we achieve this? We will achieve this by creating a geogrid with multiple locations and handing it over to the Generator, then comparing the amount of contactPools for each location with the expected amount of contactPools in that location.

2.2 Populators

2.2.1 DaycarePopulator.NoPopulation

What will it test? This test will check whether the DaycarePopulator doesn't crash when it gets a geogrid with an empty location.

How will we achieve this? We will achieve this by creating a geogrid with an empty location and passing it to the DaycarePopulator, then checking if no exceptions occur.

2.2.2 DaycarePopulator.OneLocationTest

What will it test? This test will check whether the DaycarePopulator is well-behaved when working with a geogrid with one location.

How will we achieve this? We will achieve this by creating a geogrid with one location and passing it to the DaycarePopulator, then comparing a number of parameters with the expected values, such as the amount of pools and the age range for each pool.

2.2.3 DaycarePopulator.TwoLocationTest

What will it test? This test will check whether the DaycarePopulator is well-behaved when working with a geogrid with two or more locations.

How will we achieve this? We will achieve this by creating a geogrid with two locations and passing it to the DaycarePopulator, then comparing a number of parameters for each location with the expected values, such as the amount of pools and the age range for each pool.

2.2.4 PreSchoolPopulator.NoPopulation

What will it test? This test will check whether the PreSchoolPopulator doesn't crash when it gets a geogrid with an empty location.

How will we achieve this? We will achieve this by creating a geogrid with an empty location and passing it to the PreSchoolPopulator, then checking if no exceptions occur.

2.2.5 PreSchoolPopulator.OneLocationTest

What will it test? This test will check whether the PreSchoolPopulator is well-behaved when working with a geogrid with one location.

How will we achieve this? We will achieve this by creating a geogrid with one location and passing it to the PreSchoolPopulator, then comparing a number of parameters with the expected values, such as the amount of pools and the age range for each pool.

2.2.6 PreSchoolPopulator.TwoLocationTest

What will it test? This test will check whether the PreSchoolPopulator is well-behaved when working with a geogrid with two or more locations.

How will we achieve this? We will achieve this by creating a geogrid with two locations and passing it to the PreSchoolPopulator, then comparing a number of parameters for each location with the expected values, such as the amount of pools and the age range for each pool.

3 Data formats

3.1 JSON

3.1.1 HouseholdJSONReader.validJSON

What will it test? This test will check if a given JSON file containing a household configuration is read properly and the expected values can be retreived.

How will we achieve this? A JSON formatted string is given to the reader. Then all households are checked if they have the same size and if each household contains the right values.

3.1.2 HouseholdJSONReader.invalidJSON

What will it test? It will test if a wrongly formatted JSON is given to the household reader.

How will we achieve this? Given a wrongly formatted input, the reader function is called. Then it will be checked if an exception is throw and if it is the expected error that occured.

3.1.3 GeoGridJSONReaderUnitTest.locationsTest

What will it test? It will test if the reader has extracted the right location information from a JSON file.

How will we achieve this? Given a JSON file we can read, the test will compare the ID's with the expected value and for each location all attributes will be tested on their contents.

3.1.4 GeoGridJSONReaderUnitTest.commutesTest

What will it test? It will test if the reader has extracted the right commuting information from a JSON file.

How will we achieve this? Given a JSON file we can read, the test will consider all locations and in specific if their commutes object contains the right location ID and percentage of commuters.

3.1.5 GeoGridJSONReaderUnitTest.contactPoolTest

What will it test? This test will check if each pool has the right type according to the internal coding of the different types of centers.
How will we achieve this? Given a JSON file we extract all the contactPools and their information and then we will check for each if the numeric class is available in the code. All types were specified in the JSON file and thus we can check if all types are compatible with their numerical values.

3.1.6 GeoGridJSONReaderUnitTest.peopleTest

What will it test? This test will check if a person within a contactpool can be linked to a defined person with the same ID.
How will we achieve this? From a JSON formatted file we will check if

the person within a center has a pointer to the correct object with the right information by checking all its attributes againts the expected values.

3.1.7 GeoGridJSONReaderUnitTest.intTest

What will it test? It will test if there occurs an error if a double is given and not an integer in the JSON file.
How will we achieve this? The test simply follows the same steps as the peopleTest and sees if the error occurs or everything can be simply casted to the right types.

3.1.8 GeoGridJSONReaderUnitTest.emptyStreamTest

What will it test? This test will examine if an idle input can be properly handled by the reader.
How will we achieve this? We will simply pass along an empty stream to the reader and observe its response in the form of an exception.

3.1.9 GeoGridJSONReaderUnitTest.invalidTypeTest

What will it test? This test will examine if a correct response is produced when a wrong type is used for a contactPool in the JSON format.
How will we achieve this? A JSON file is given to the reader with a wrong type for a contactPool. Then it is checked if an exception has been throw to indicate something went wrong.

3.1.10 GeoGridJSONReaderUnitTest.invalidPersonTest

What will it test? This test sees if it is possible to reference a non-existing person within a contactPool without defining this person with its attributes.
How will we achieve this? A JSON file with a wrong person in a contactPool is given to the reader to test its response. If it returns an exception indicating something has gone wrong, the test will succeed.

3.1.11 GeoGridJSONReaderUnitTest.invalidJSONTest

What will it test? A test to see what will happen if a random word is given within a JSON formatted file.
How will we achieve this? By passing a random string within a JSON file the test will determine if a proper exception is thrown and the execution is aborted.

3.1.12 GeoGridJSONWriterUnitTest.locationTest

What will it test? A test to determine if the location information retrieved from a GeoGrid is correctly formatted in a JSON file.

How will we achieve this? The test will create a GeoGrid and passes this information to the writer which will create a JSON formatted output. The test will convert this JSON and a comparison file to an XML format and check if they are the same.

3.1.13 GeoGridJSONWriterUnitTest.contactPoolTest

What will it test? A test to determine if the contactPool information retrieved from a GeoGrid is correctly formatted in a JSON file.

How will we achieve this? The test will create a GeoGrid and passes this information to the writer which will create a JSON formatted output. The test will convert this JSON and a comparison file to an XML format and check if they are the same.

3.1.14 GeoGridJSONWriterUnitTest.peopleTest

What will it test? This test will look in specific if the people are correctly represented in the resulting JSON file.

How will we achieve this? The test will create a population and passes this information to the writer which will create a JSON formatted output. The test will convert this JSON and a comparison file to an XML format and check if they are the same.

3.1.15 GeoGridJSONWriterUnitTest.commutesTest

What will it test? This test will look in specific if the commuting information is correctly represented in the written JSON file.

How will we achieve this? The test will create a population and passes this information to the writer which will create a JSON formatted output. The test will convert this JSON and a comparison file to an XML format and check if they are the same.

3.1.16 GeoGridJSONReaderWriterScenarioTest.mainTest

What will it test? This is a test that will examine if the JSON writes outputs a JSON formatted file and reads the excact information in JSON formats.

How will we achieve this? The test creates a population and writes it to a JSON file wich will be converted to a string to chech wit the expected output. Next the same file is used to be read by the JSON reader ans all values are inspected.

3.2 HDF5

3.2.1 HDF5Reader.locationsTest

What will it test? This test will check whether the locations are read correctly from the HDF5 file.

How will we achieve this? We will achieve this by comparing the locations read from the HDF5 input file with the correct locations.

3.2.2 HDF5Reader.commutesTest

What will it test? This test will check whether the commutes are read correctly from the HDF5 file.

How will we achieve this? We will achieve this by comparing the commutes read from the HDF5 input file with the correct commutes.

3.2.3 HDF5Reader.contactCentersTest

What will it test? This test will check whether the contactCenters are read correctly from the HDF5 file.

How will we achieve this? We will achieve this by comparing the contactCenters read from the HDF5 input file with the correct contactCenters.

3.2.4 HDF5Reader.peopleTest

What will it test? This test will check whether the people are read correctly from the HDF5 file.

How will we achieve this? We will achieve this by comparing the people read from the HDF5 input file with the correct people.

3.2.5 HDF5Reader.invalidHDF5Test

What will it test? This test will check whether the HDF5Reader accepts invalid formed files or not.

How will we achieve this? We will achieve this by feeding the HDF5Reader a number of invalid files and check whether or not an exception is thrown by the HDF5Reader.

3.2.6 HDF5Writer.locationsTest

What will it test? This test will check whether the locations are written correctly to the HDF5 file.

How will we achieve this? We will achieve this by comparing the resulting HDF5 file with the correct HDF5 file.

3.2.7 HDF5Writer.commutesTest

What will it test? This test will check whether the commutes are written correctly to the HDF5 file.

How will we achieve this? We will achieve this by comparing the resulting HDF5 file with the correct HDF5 file.

3.2.8 HDF5Writer.contactCentersTest

What will it test? This test will check whether the contactCenters are written correctly to the HDF5 file.

How will we achieve this? We will achieve this by comparing the resulting HDF5 file with the correct HDF5 file.

3.2.9 HDF5Writer.peopleTest

What will it test? This test will check whether the people are written correctly to the HDF5 file.

How will we achieve this? We will achieve this by comparing the resulting HDF5 file with the correct HDF5 file.

4 QT

4.1 visualiser

No tests will be made for the visualizer, as this is hard to test via code. The testing of QT will happen via simulations done by ourselves and outsiders checking if our interface is obvious and well made

4.2 QT Proto

4.2.1 introduction

In order to make the visualizer as efficient as could be, there was need for a secondary input format that specified the data specifically needed for the visualizer. ProtoBuf is one of the supported types for the visualizer data.

4.2.2 EpiGridProtoReaderTest.locationTest

What will it test? This test will check whether ProtoBuf reader is capable of reading locations properly.

How will we achieve this? We will achieve this by calling the EpiProtoReader::Read() function using a made EpiGrid. We can check if this was according to plan, by comparing the resulting EpiGrid of the reader with the original epiGrid.

4.2.3 EpiGridProtoReaderTest.historyTest

What will it test? This test will check whether ProtoBuf reader is capable of reading the history properly.

How will we achieve this? We will achieve this by calling the EpiProtoReader::Read() function using a made EpiGrid. We can check if this was according to plan, by comparing the resulting EpiGrid of the reader with the original epiGrid.

4.2.4 EpiGridProtoReaderTest.poolsTest

What will it test? This test will check whether ProtoBuf reader is capable of reading the pools inside history properly.

How will we achieve this? We will achieve this by calling the EpiProtoReader::Read() function using a made EpiGrid. We can check if this was according to plan, by comparing the resulting EpiGrid of the reader with the original epiGrid.

4.2.5 EpiGridProtoWriterTest.locationTest

What will it test? This test will check whether ProtoBuf writer is capable of writing locations properly.

How will we achieve this? We will achieve this by using the EpiGridProtoWriter with a made GeoGrid. We can check if this was according to plan, by comparing the resulting ProtoBuf object with the original GeoGrid.

4.2.6 EpiGridProtoWriterTest.contanctPoolsTest

What will it test? This test will check whether ProtoBuf writer is capable of writing the pools inside history properly.

How will we achieve this? We will achieve this by using the EpiGridProtoWriter with a made GeoGrid. We can check if this was according to plan, by comparing the resulting ProtoBuf object with the original GeoGrid.

4.2.7 EpiGridProtoReaderWriterScenartioTest.locationTest

What will it test? This test will check whether ProtoBuf reader is capable of reading the result of protoBuf writer, that wrote locations, along with a check whether or not the result is correct.

How will we achieve this? We will achieve this by using the EpiGridProtoWriter with a made GeoGrid followed by the EpiGridProtoReader that will read the resulting file. We can check if this was according to plan, by comparing the resulting EpiGrid object with the original GeoGrid.

4.2.8 EpiGridProtoReaderWriterScenartioTest.contanctPoolsTest

What will it test? This test will check whether ProtoBuf reader is capable of reading the result of protoBuf writer, that wrote a history with relevant data, along with a check whether or not the result is correct.

How will we achieve this? We will achieve this by using the EpiGridProtoWriter with a made GeoGrid followed by the EpiGridProtoReader that will read the resulting file. We can check if this was according to plan, by comparing the resulting EpiGrid object with the original GeoGrid.

4.3 QT HDF5

4.4 QT JSON

4.4.1 EpiGridJSONReaderUnitTest.locationsTest

What will it test? This test will check if the reader can properly read the locations in the json file.

How will we achieve this? We will prepare a json file with a number of locations, read it and compare the results with predefined results.

4.4.2 EpiGridJSONReaderUnitTest.locationTest

What will it test? This test will check if the reader can properly read the locations and the agebrackets in the json file.

How will we achieve this? We will prepare a json file with one location and one timestep in the history, read it and compare the results with predefined results.

4.4.3 EpiGridJSONReaderUnitTest.historyTest

What will it test? This test will check if the reader can properly read multiple timesteps in the history object of the json file.

How will we achieve this? We will prepare a json file with one location and mulitple timesteps, read it and compare the results with predefined results.

4.4.4 EpiGridJSONWriterUnitTest.locationsTest

What will it test? This test will check if the writer can write the location information, like name and id, to json.

How will we achieve this? We will create a Geogrid, fill it with locations, write it to a string and compare the string with a predefined json output string.

4.4.5 EpiGridJSONWriterUnitTest.historyTest

What will it test? This test will check if the writer can write the percentage of people per agebracket and healthstatus to json.

How will we achieve this? We will create a Geogrid, fill it with a location and write it to a string and compare the string with a predefined json output string.

4.4.6 EpiGridJSONReaderWriterScenarioTest.mainTest

What will it test? This test will check if the reader can read what the writer has written.

How will we achieve this? We will create a Geogrid, fill it with a location and write two timesteps. We will then read the output from file and compare the results with predifined values.

5 Improved population generation

5.1 Introduction

All tests related to this section, unit or scenario, will be written by Lars Van Roy, the same student that wrote the related code.

5.2 Demographic profile

5.2.1 UnitDemographicProfileTest.onlyDefaultConfiguration

What will it test? This test will check whether the original stride implementation is still functional (regarding reading the csv).

How will we achieve this? We will achieve this by calling the SetData function, part of GeoGridConfig, with a csv file. We can then check if the data was correctly read into the GeoGridConfig.

5.2.2 UnitDemographicProfileTest.oneAdditionalProvince

What will it test? This test will check whether everything goes fine when we give a second csv, linked to a province.

How will we achieve this? We will achieve this by calling the SetData function twice, once for the original and once for the province csv. We can then compare the data saved in the GeoGridConfig to the actually expected data.

5.2.3 UnitDemographicProfileTest.additionalMajorCities

What will it test? This test will check whether we are able to read major city distribution data.

How will we achieve this? We will achieve this by calling the SetData function twice, once for the original and once for the major city csv. We can then compare the data saved in the GeoGridConfig to the actually expected data.

5.2.4 ScenarioDemographicProfileTest.CorrectUseOfProvinces

What will it test? This test will check whether we correctly read province distribution data, and whether we prefer these over the default data.

How will we achieve this? We will achieve this by calling the SetData function for all possible demographic profile input tags, followed by generating all possible pools. We can then compare the data saved in the GeoGridConfig to the actually expected data, along with counting the number of pools that got created.

5.2.5 ScenarioDemographicProfileTest.CorrectUseInGenerator

What will it test? This test will check whether we correctly read province distribution data, and whether we correctly apply the included percentages within these configurations.

How will we achieve this? We will achieve this by calling the SetData function for all possible demographic profile input tags, followed by generating all possible pools. We can then compare the data saved in the GeoGridConfig to the actually expected data, along with counting the number of pools that got created.

5.3 Workplace contactpools

5.3.1 WorkplaceGeneratorTest.ZeroLocationTest

What will it test? This test will check whether the generator can handle empty GeoGrid.

How will we achieve this? We will achieve this by passing a GeoGridConfig object that has no workers. After this we can check if the workplace generator added any pools by looking at the size function.

5.3.2 WorkplaceGeneratorTest.NoCommutingWithDistribution

What will it test? This test will check whether the generator can handle a configuration that has no commuting but does have a workplace distribution.

How will we achieve this? We will achieve this by passing a GeoGridConfig object that has no commuters and a workplace distribution configuration. After this we can check whether each poolsize is within the expected boundaries given by the distribution.

5.3.3 WorkplaceGeneratorTest.NullCommutingWithDistribution

What will it test? This test will check whether the generator can handle a configuration where there are as many commuters from A to B as from B to A.

How will we achieve this? We will achieve this by passing a GeoGridConfig where such a situation occurs and one that has a workplace distribution. We can easily verify whether everything went as planned by checking if the conditions on the GeoGridConfig were in fact true followed by checking whether the poolsizes are withing the expected boundaries.

5.3.4 WorkplaceGeneratorTest.TenCommutingWithDistribution

What will it test? This test will check whether the generator can handle a configuration where there are multiple commuters from different places along with a workplace distribution configuration.

How will we achieve this? We will achieve this by passing a GeoGridConfig where such a situation occurs and one that has a workplace distribution. We can easily verify whether everything went as planned by checking if the conditions on the GeoGridConfig were in fact true followed by checking whether the poolsizes are withing the expected boundaries.

5.3.5 WorkplaceGeneratorTest.NoCommutingWithoutDistribution

What will it test? This test will check whether the generator can handle a configuration that has no commuting and no workplace distribution configuration.

How will we achieve this? We will achieve this by passing a GeoGridConfig object that has no commuters and no workplace distribution configuration.

After this we can check each whether each poolsize is equal to the expected value.

5.3.6 WorkplaceGeneratorTest.NullCommutingWithoutDistribution

What will it test? This test will check whether the generator can handle a configuration where there as many commuters from A to B as from B to A and no workplace distribution configuration.

How will we achieve this? We will achieve this by passing a GeoGridConfig where such a situation occurs and where there is no workplace distribution configuration. We can easily verify whether everything went as planned by checking if the conditions on the GeoGridConfig where in fact true followed by checking whether the poolsizes are equal to the expected values.

5.3.7 WorkplaceGeneratorTest.TenCommutingWithoutDistribution

What will it test? This test will check whether the generator can handle a configuration where there are multiple commuters from different places and no workplace distribution configuration.

How will we achieve this? We will achieve this by passing a GeoGridConfig where such a situation occurs and where there is no workplace distribution configuration. We can easily verify whether everything went as planned by checking if the conditions on the GeoGridConfig where in fact true followed by checking whether the poolsizes are equal to their expected values.

5.3.8 WorkplaceCSVReader.TestValidInput

What will it test? This test will check whether we are able to read Workplace distribution CSV's.

How will we achieve this? We will achieve this by passing a possible CSV configuration to the reader. The returned format can then be checked in order to ensure that the configuration is read correctly.

5.3.9 WorkplacePopulatorTest

For populating the original tests where kept, as it did not change by adding the workplace distribution configurations, they were however slightly altered to support the new generator.

5.3.10 WorkplaceScenarioTest.WorkplaceScenarioWithDistribution

What will it test? This test will check if everything works out when we create a scenario, using the workplace reader, workplace generator and the workplace populator.

How will we achieve this? We will achieve this by passing a csv to the reader, followed by calling the generator and populator using the result of the prev steps. We will not verify if the populator worked as expected (as this is not changed within the context, it is just tested to ensure that it does not crash when using the data from the previous step), we will only check if the generated pools are within the expected bounds.