Exercise
Synthea^{TM}: Massive FHIR Data

Track: Developers
Track lead: **Jason Walonoski**
Email: jwalonoski@mitre.org

This tutorial and exercise will introduce and walk through the generation of synthetic data for your next software development, integration, connectathon, or analytics task. The session will highlight Synthea ("Synthetic Health") an open-source software package that generates synthetic patient records. The software models and simulates over 500 clinical concepts, with a growing list of community written modules.

After completing this tutorial, you will be able to:

1. Install, Configure, and Run Synthea
2. Use the Synthetic FHIR Data
3. Explore and Modify the Disease Modules
4. Localize Synthea for Alternative Geographic Locations

Have fun and remember to ask for help if you get stuck!

Please note:

- The exercises can be made in the hands-on area, where each track has its own table, indicated with a track sign. The track lead will be present for guidance and review.
- Exercises will only be discussed or reviewed during the HL7 FHIR DevDays 18 in Boston
- Any questions or remarks after the conference can be addressed in the FHIR chat on Zulip: https://chat.fhir.org

1. Install, Configure, and Generate Data

This first exercise involves installing, configuring, and running Synthea to generate synthetic data. If you only want to use synthetic data, skip to exercise #3.

Requirements

- Java Development Kit (JDK) version 1.8
- Git Version Control

To copy the repository locally and install the necessary dependencies, open a terminal window and run the following commands:

```
git clone https://github.com/synthetichealth/synthea.git
cd synthea
./gradlew build check test
```

Note: if running on Windows, use .\gradlew.bat instead of ./gradlew -- this guide uses ./gradlew for brevity

The primary entry point of Synthea is the “run” gradle task, which can be run from the terminal in the synthea folder by running the following command.

```
./gradlew run
```

Alternatively, you may run the provided `run_synthea` script which makes it easier to pass command-line arguments to the simulation:

```
./run_synthea
```

Note: if running on Windows, use .\run_synthea.bat instead of ./run_synthea -- this guide uses ./run_synthea for brevity

When you run this command, you should see output similar to the following:

```
example@hostname ~/synthea $ ./run_synthea

> Task :run
Loading C:\Users\example\synthea\build\resources\main\modules\allergic_rhinitis.json
Loading C:\Users\example\synthea\build\resources\main\modules\allergies\allergy_incidence.json
[... many more lines of Loading ...]
Loading C:\Users\example\synthea\build\resources\main\modules\wellness_encounters.json
Loaded 68 modules.
Running with options:
Population: 1
Seed: 1519063214833
Location: Massachusetts

1 -- Jerilyn993 Parker433 (10 y/o) Lawrence, Massachusetts
```

This command takes additional parameters to specify different regions or common run options. Any options not specified are left at the default value.

```
run_synthea [-s seed]
            [-p populationSize]
            [-g gender]
            [-a minAge-maxAge]
            [state [city]]
```

Some examples:

- `run_synthea Massachusetts` -- to generate a population in all cities and towns in Massachusetts
- `run_synthea Alaska Juneau` -- to generate a population in only Juneau, Alaska
- `run_synthea -s 12345` -- to generate a population using seed 12345. Populations generated with the same seed and the same version of Synthea should be identical
- `run_synthea -p 1000` -- to generate a population of 1000 patients
- `run_synthea -a 30-40` -- to generate a population of 30 to 40-year olds
- `run_synthea -g F` -- to generate only female patients
- `run_synthea -s 987 Washington Seattle` -- to generate a population in only Seattle, Washington, using seed 987
- `run_synthea -s 21 -p 100 Utah "Salt Lake City"` -- to generate a population of 100 patients in Salt Lake City, Utah, using seed 21

The tutorial and code for this exercise can be found at:
https://github.com/synthetichealth/synthea/wiki/Basic-Setup-and-Running

2. Use the Synthetic FHIR Data

The tutorial and code for this exercise can be found at:
https://github.com/synthetichealth/synthea/wiki/HL7-FHIR

Synthea generates HL7 FHIR records using the HAPI FHIR library to generate a FHIR Bundle for each Patient. Currently, only JSON is supported for these FHIR Resources:

- `Bundle`
- `Patient`
- `Encounter`
- `Condition`
- `AllergyIntolerance`
- `Observation`
- `DiagnosticReport`
- `Procedure`
- `ImagingStudy`
- `Immunization`
- `CarePlan`
- `MedicationRequest`

Configuring FHIR

The exporting of FHIR can be configured using the `src/main/resources/synthea.properties`:

```
# Abridged synthea.properties file
# default FHIR configuration.
exporter.fhir.export = true
# transaction bundle 'true' produces transaction Bundles
# while 'false' produces collection Bundles.
exporter.fhir.transaction_bundle = true
# Standard Health Record (SHR) extensions for STU3
exporter.fhir.use_shr_extensions = true
# Exporting FHIR DSTU2
exporter.fhir_dstu2.export = false
# Exporting Hospital Provider Data in STU3 or DSTU2
exporter.hospital.fhir.export = true
exporter.hospital.fhir_dstu2.export = false
```

Producing and Using FHIR

In the `synthea.properties` file, make sure that `exporter.fhir.export`, `exporter.fhir.transaction_bundle`, and `exporter.hospital.fhir.export` are all set to `true`.

Next, produce some synthetic data:

```
./run_synthea -p 100
```

By default, FHIR JSON data should now exist in the `./output/fhir` sub-directory. If you have a FHIR server and client, you can POST those bundles to the FHIR server. For example:

```
curl http://hapi.fhir.org/baseDstu3
  --data-binary "@/Users/example/synthea/output/fhir/Maryetta775_Rowe323_2cb7e4dd-9d8b-49cf-b1e4-9839be8bc754.json"
  -H "Content-Type: application/fhir+json"
```

The FHIR server should return a `transaction-response` Bundle:

```
{
  "resourceType": "Bundle",
  "id": "ca4d459f-b078-4c04-a152-c7ce76a25179",
  "type": "transaction-response",
  "link": [
    {
      "relation": "self",
      "url": "http://hapi.fhir.org/baseDstu3"
    }
  ],
  "entry": [
    {
      "response": {
        "status": "201 Created",
        "location": "Patient/4147259/_history/1",
        "etag": "1",
        "lastModified": "2018-06-06T18:26:06.038+00:00"
      }
    },
    ...abridged...
  ]
}
```

Other Examples of Using Synthea FHIR Data

- The Healthcare Services Platform Consortium (HSPC) has a developer sandbox environment that allows developers to spin-up FHIR servers and load them with Synthea data. They also host a FHIR server preloaded with Synthea data called HSPC Synthea STU3 (3.0.1) (authentication required).
  - https://www.hspconsortium.org/
  - https://sandbox.hspconsortium.org/#/login

- The SMART Health IT (https://smarthealthit.org/) team has pre-generated datasets (http://docs.smarthealthit.org/data/stu3-sandbox-data.html) available for download including Synthea data, which are also available through https://dev.smarthealthit.org. They also provide a Docker version of the HAPI FHIR Server preloaded with Synthea data here: https://github.com/smart-on-fhir/hapi, and have used Synthea data with their Bulk Data Server (https://github.com/smart-on-fhir/bulk-data-server).

- Algorex Health has used Synthea data to explore "Open Clinical Analysis"
  - https://blog.algorexhealth.com/2017/04/open-clinical-analysis-with-mitre-part-2/

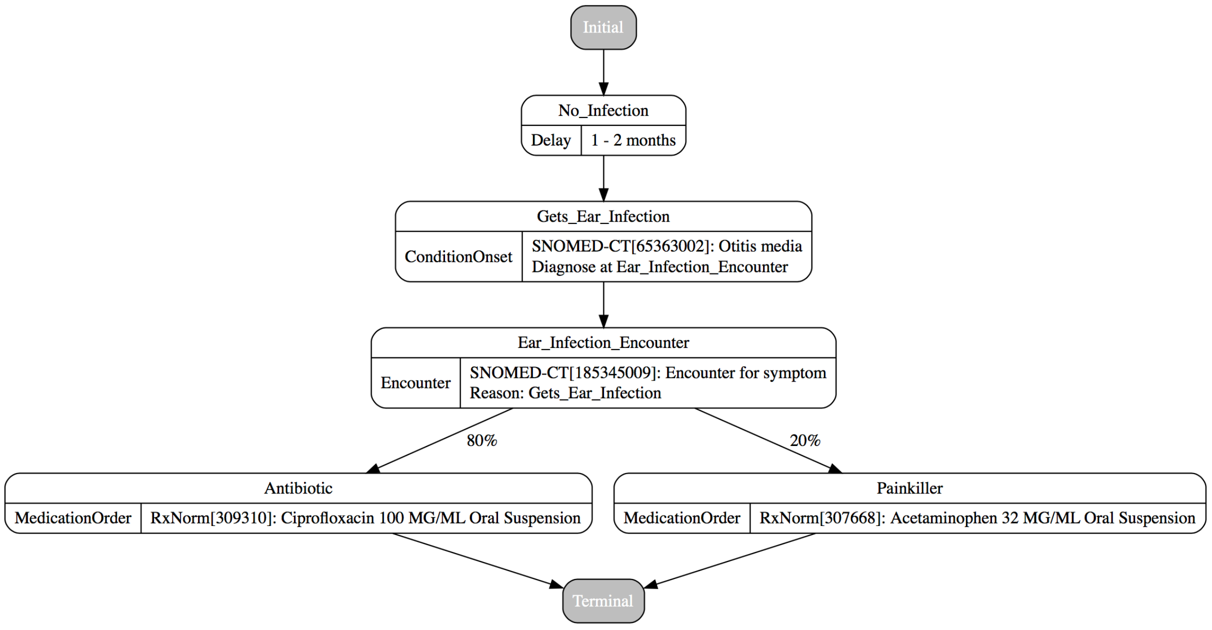
- The MITRE Corporation has used Synthea data to create SyntheticMass (https://syntheticmass.mitre.org/), a 1/7th scale simulated model of the Commonwealth of Massachusetts, including a FHIR server (FHIR v1.8).

- An MSDN blog post illustrates "Loading Synthea FHIR Data with Logic Apps and Functions in Azure Government"
  - https://blogs.msdn.microsoft.com/mihansen/2018/05/10/loading-synthea-fhir-data-with-logic-apps-and-functions-in-azure-government/

3. Explore and Modify the Disease Modules

This tutorial can be found at:
https://github.com/synthetichealth/synthea/wiki/Generic-Module-Framework

Synthea contains a framework for defining modules using JSON. These JSON modules describe a progression of states and the transitions between them. On each Synthea generation time-step, the generic framework processes states one at a time to trigger conditions, encounters, medications, and other clinical events.


Initial
No_Infection
Delay
1 - 2 months
Gets_Ear_Infection
ConditionOnset
SNOMED-CT[65363002]: Otitis media
Diagnose at Ear_Infection_Encounter
Ear_Infection_Encounter
Encounter
SNOMED-CT[185345009]: Encounter for symptom
Reason: Gets_Ear_Infection
80%
20%
Antibiotic
MedicationOrder
RxNorm[309310]: Ciprofloxacin 100 MG/ML Oral Suspension
Painkiller
MedicationOrder
RxNorm[307668]: Acetaminophen 32 MG/ML Oral Suspension
Terminal

This simplified example of childhood ear infections shows the flow of a generic module. In this instance, children get an ear infection (ConditionOnset) after a random Delay of 1 to 2 months, are then diagnosed at an Encounter, and then are prescribed (MedicationOrder) either an antibiotic or a painkiller.

```
{
  "name": "Ear Infections",
  "states": {
    "Initial": {
      "type": "Initial",
      "direct_transition": "No_Infection"
    },
    "No_Infection": {
      "type": "Delay",
      "direct_transition": "Gets_Ear_Infection",
      "range": { "low": 1, "high": 2,
                 "unit": "months" }
    },
    "Gets_Ear_Infection": {
      "type": "ConditionOnset",
      "target_encounter": "Ear_Infection_Encounter",
      "codes": [{
        "system": "SNOMED-CT", "code": "65363002",
        "display": "Otitis media"
      }],
      "direct_transition": "Ear_Infection_Encounter"
    },
    "Ear_Infection_Encounter": {
      "type": "Encounter",
      "encounter_class": "outpatient",
      "reason": "Gets_Ear_Infection",
      "codes": [{
        "system": "SNOMED-CT", "code": "185345009",
        "display": "Encounter for symptom"
      }],
      "distributed_transition": [
        { "distribution": 0.8,
          "transition": "Antibiotic" },
        { "distribution": 0.2,
          "transition": "Painkiller" }
      ]
    },
    "Antibiotic": {
      "type": "MedicationOrder",
      "codes": [{
        "system": "RxNorm", "code": 309310,
        "display": "Ciprofloxacin 100 MG/ML
                    Oral Suspension"
      }],
      "direct_transition": "Terminal"
    },
    "Painkiller": {
      "type": "MedicationOrder",
      "codes": [{
        "system": "RxNorm", "code": 307668,
        "display": "Acetaminophen 32 MG/ML
                    Oral Suspension"
      }],
      "direct_transition": "Terminal"
    },
    "Terminal": { "type": "Terminal" }}}
```

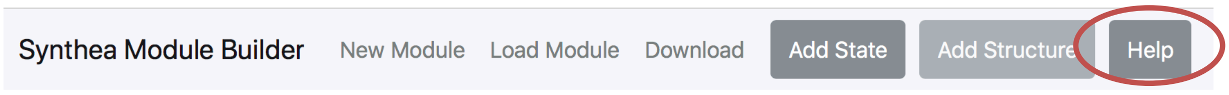
The JSON for this simplified "Ear Infections" module shows how this generic module is defined in a JSON object with a *name* and a collection of *states*. Each state has a *name*, *type,* and a *transition*. Different types of *states* and *transitions* have different properties. These can be hand-edited following the documentation at:

https://github.com/synthetichealth/synthea/wiki/Generic-Module-Framework

Alternatively, modules can be edited using the online Module Builder tool:

https://synthetichealth.github.io/module-builder/#

For this tutorial, use the Module Builder and edit or create a new module. The user interface is intended to be straightforward and natural to use, but if necessary you can click on the Help button to orient yourself.



The menu bar above shows some of the options: New Module, Load Module, and Download. Once editing a module, you can Add State. Editing or deleting states can be done by clicking on the state and using the side bar.

By default, the module builder starts with the Examplitis module. Examplitis is a fictional yet painful condition that affects only males. Most patients can be cured with Examplitol or an Examplotomy, but some never recover. You can get a complete walk-through of the Examplitis module here:

https://github.com/synthetichealth/synthea/wiki/Generic-Module-Framework%3A-Complete-Example#examplitis-walk-through

In this tutorial, edit and play with the Examplitis module. See what you can come up with. When you are ready to test your Examplitis module, click on the *Download* button. That will open a pop-up where you can copy and paste the JSON. In this case, just click the second *Download* button and save the Examplitis module.

Copy the downloaded module into Synthea, test it by generating some patients, and search for "Examplitis":

```
synthea$ cp ~/Downloads/examplitis.json src/main/resources/modules/.
synthea$ ./run_synthea -s 0 -p 100

... skipping output...

synthea$ grep "Examplitis" output/fhir/*.json
output/fhir/Art115 Steuber698_98498ae6-0ac7-4b12-bbda-67c1c3bc49d2.json:       "display": "Examplitis"
output/fhir/Art115 Steuber698_98498ae6-0ac7-4b12-bbda-67c1c3bc49d2.json:          "text": "Examplitis"
output/fhir/Otis335 Johns824_a0030b55-0848-437a-81e3-23351debf675.json:        "display": "Examplitis"
output/fhir/Otis335 Johns824_a0030b55-0848-437a-81e3-23351debf675.json:           "text": "Examplitis"
output/fhir/Rich940 Harber290_e4ddce9d-efe6-403e-a618-b3b83fb12a8c.json:       "display": "Examplitis"
output/fhir/Rich940 Harber290_e4ddce9d-efe6-403e-a618-b3b83fb12a8c.json:          "text": "Examplitis"
```

It looks like 3 of the 100 patients we generated contracted Examplitis!

Open those FHIR records and examine the FHIR JSON output associated with Examplitis. See if you can find any evidence of medications prescribed by the module or other changes you might have made using the Module Builder.

4. Localize Synthea for Alternative Geographic Locations

The tutorial and code for this exercise can be found at:
https://github.com/synthetichealth/synthea/wiki/Other-Areas

By default, Synthea contains demographics, zip codes, providers, names, and costs for the entire United States, post-processed from publicly available files.

You can modify these files or have Synthea use alternative files by altering the `src/main/resources/synthea.properties` file:

```
# Abridged synthea.properties file
# Default demographics is every city in the US
generate.demographics.default_file = geography/demographics.csv
generate.geography.zipcodes.default_file = geography/zipcodes.csv
generate.geography.country_code = US

# Default provider files (1 of 10 files)
generate.providers.hospitals.default_file = providers/hospitals.csv

# Default costs, to be used for pricing something that we don't have a specific price for
generate.costs.default_procedure_cost = 500.00
generate.costs.default_medication_cost = 255.00
generate.costs.default_encounter_cost = 125.00
generate.costs.default_immunization_cost = 136.00
```

You can find detailed instructions on each of these file formats on the Synthea Wiki:

- **Demographics** file contains city-level distributions of Gender, Race, Age, Income, and Education which all play a role in health access, outcomes, and costs. https://github.com/synthetichealth/synthea/wiki/Demographics-for-Other-Areas
- **Zip or postal code** file contains postal codes and geographic locations (latitude and longitude) for each location. https://github.com/synthetichealth/synthea/wiki/Zip-or-Postal-Codes
- **Provider** files contains information and locations (latitude and longitude) for each provider location. https://github.com/synthetichealth/synthea/wiki/Provider-Data
- **Names** file contains language-specific family names and first names by gender. It also contains names suitable for street addresses. https://github.com/synthetichealth/synthea/wiki/Name-Data
- **Cost** files contain costs for different types of Encounters, Procedures, Medications, and Immunizations. https://github.com/synthetichealth/synthea/wiki/Cost-Data

Abridged Example for Shrewsbury, Shropshire, United Kingdom

Districting and addresses in the United Kingdom differ from the United States. Let's start with the **demographics** file:

```
,COUNTY,NAME,STNAME,POPESTIMATE2015,CTYNAME,TOT_POP,TOT_MALE,TOT_FEMALE,WHITE,HISPANIC,BLAC
K,ASIAN,NATIVE,OTHER,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,00..10,10..15,15..25,25..
35,35..50,50..75,75..100,100..150,150..200,200..999,LESS_THAN_HS,HS_DEGREE,SOME_COLLEGE,BS_
DEGREE
1,1,Shrewsbury,West
Midlands,2620,Shropshire,486300,0.489,0.511,0.985,0,0.001,0.004,0,0,0.063,0.063,0.063,0.052
,0.052,0.052,0.052,0.052,0.076,0.076,0.076,0.076,0.076,0.035,0.035,0.035,0.035,0.035,0.1333
33333,0.133333333,0.133333333,0.147225,0.147225,0.147225,0.147225,0.1,0.01,0.001,0.18,0.387
,22.35,22.35
```

Next, we need to create postal codes suitable to Shrewsbury, so we edit the **zip codes** file:

```
,USPS,ST,NAME,ZCTA5,LAT,LON
0,West Midlands,WMS,Shrewsbury,SY1,52.7081,-2.7549
1,West Midlands,WMS,Shrewsbury,SY2,52.7083,-2.7546
2,West Midlands,WMS,Shrewsbury,SY3,52.7089,-2.7543
```

Finally, we need to provide at least one hospital for our Salopians, so we edit the **hospitals** file:

```
,id,name,address,city,state,zip,county,phone,type,ownership,emergency,quality,LAT,LON
0,010001,Royal Shrewsbury Hospital,Mytton Oak Rd,Shrewsbury,WMS,SY3
8XQ,Shropshire,01743261000,NHS Teaching Hospital,Government - Hospital District or
Authority,Yes,5,52.7091,-2.7931
```

For now, let's leave the **names** and **cost** files as-is. We're ready to generate data, but there is one last cosmetic change we need to make. In this example, we're generating the United Kingdom (UK), instead of the default United States (US). Let's change the `country_code` in `synthea.properties` so generated postal addresses look correct:

```
generate.geography.country_code = UK
```

Now, let's generate some data.

```
./run_synthea -s 0 -p 1 "West Midlands" Shrewsbury

...abridged...

Loaded 70 modules.
Running with options:
Population: 1
Seed: 0
Location: Shrewsbury, West Midlands
Min Age: 0
Max Age: 140
1 -- Brigitte394 Stark857 (50 y/o F) Shrewsbury, West Midlands DECEASED
1 -- Jeanine128 Gleason633 (57 y/o F) Shrewsbury, West Midlands
{alive=1, dead=1}
```

Check the FHIR output of `Jeanine128 Gleason633`:

```
...abridged...
        "address": [{
            "extension": [{
                "url": "http://hl7.org/fhir/StructureDefinition/geolocation",
                "extension": [
                  { "url": "latitude", "valueDecimal": -2.7546 },
                  { "url": "longitude", "valueDecimal": 52.7083 }
                ]
            }],
            "line": [ "560 Corkery Annex Suite 36" ],
            "city": "Shrewsbury",
            "state": "West Midlands",
            "postalCode": "SY1",
            "country": "UK"
        }],
...abridged...
```

We did it! We've generated `Jeanine128 Gleason633` from Shrewsbury in the United Kingdom.
If you look at the FHIR data carefully, you'll notice a few oddities. First, Jeanine128 has a Social Security Number (SSN) and a phone number that looks suspiciously like it is in the United States. Also, the FHIR records do not use FHIR Profiles from NIH. So, localization in Synthea is not fully solved, and in particular profile conformance is a remaining challenge.