Tecnológico Nacional de México

Instituto Tecnológico de Tijuana

Doctorado en Ciencias de la Ingeniería

Interactive system for ambulance dispatching in the city of Tijuana: Simulator

By:

M. C. Noelia Araceli Torres Cortés

Thesis Director:

Dra. Yazmin Maldonado Robles

Dr. Leonardo Trujillo Reyes

February 24, 2022

Content

1. Introduction

Emergency medical system

In recent decades, a major problem has been to reduce the response times, a critical factor because when the service is not provided quickly it can put people's lives at risk.

Figure 1: Emergency medical service system by the Tijuana Red Cross.

2. Problem statement

Problem statement

This work aims to address the problem of dispatching prehospital emergency units (ambulances) in the city of Tijuana, considering the travel time (duration of the EMS) and relocation of the units.

According to the American Heart Association an EMS must be attended between 4 and 6 minutes [1]. However, given the geographical conditions of the city and the few resources that the Tijuana Red Cross has it is difficult to attend this expectation (in Tijuana an EMS is attended in approximately 14 minutes).

Other important factors are:

- Traffic conditions (peak hours, weekdays, etc.)
- The priority of the call
- Streets and avenues in poor condition
- Hard to find the geographical location of the call

[1] American Heart Association AHA.http://circ.ahajournals.org/content/84/2/960.citation.2018

Problem statement

This work aims to address the problem of dispatching prehospital emergency units (ambulances) in the city of Tijuana, considering the travel time (duration of the EMS) and relocation of the units.

According to the American Heart Association an EMS must be attended between 4 and 6 minutes [1]. However, given the geographical conditions of the city and the few resources that the Tijuana Red Cross has it is difficult to attend this expectation (in Tijuana an EMS is attended in approximately 14 minutes).

Other important factors are:

- Traffic conditions (peak hours, weekdays, etc.)
- The priority of the call
- Streets and avenues in poor condition
- Hard to find the geographical location of the call

[1] American Heart Association AHA.http://circ.ahajournals.org/content/84/2/960.citation.2018

Problem statement

Tijuana features:

- Approximately 1.6 million inhabitants.

RCT features:

- Covers approximately 98% of all requests for EMS
- With approximately between 12 to 16 ambulances.
- 6 static bases.

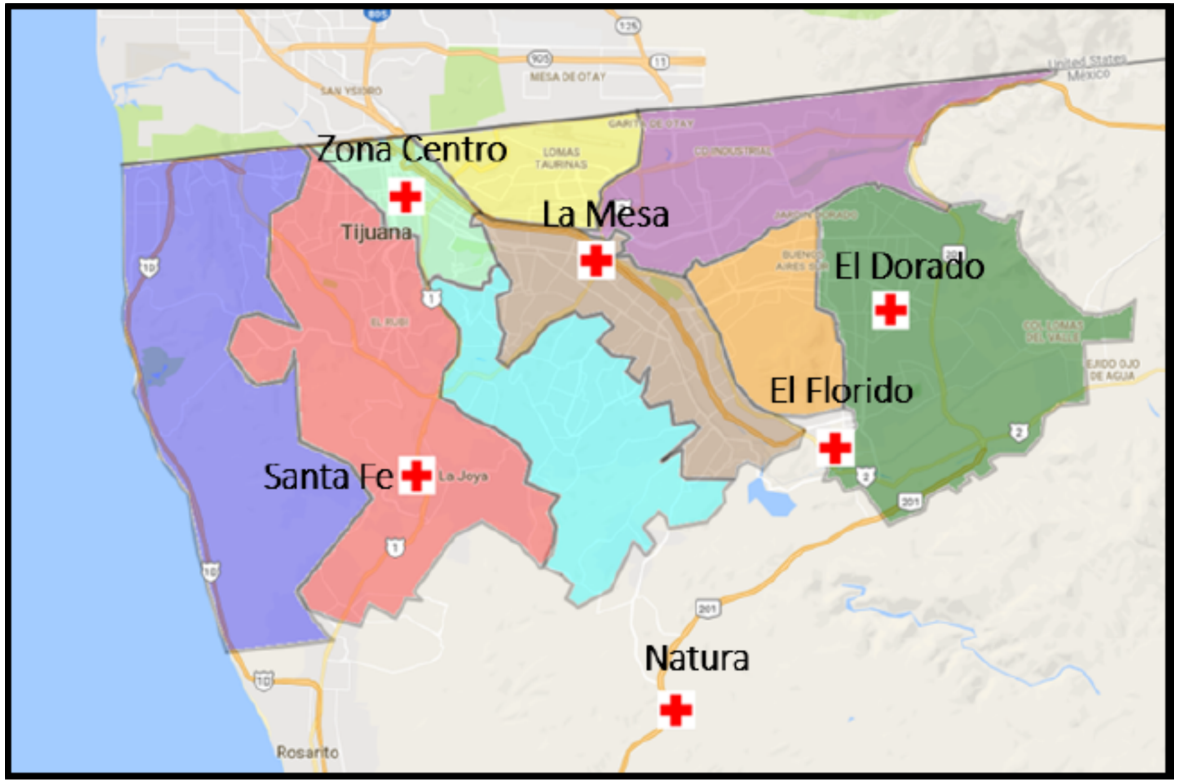


Figure 2: Tijuana, Baja California.

3. Objective

Objective

The objective is to generate an optimal dispatch rule of the ambulances for the city of Tijuana, that minimizes the average response time of the ambulances during a given period of time, but at the same time, maximizes the average coverage provided during that period of time.

4. Simulator Module Overview

Simulator module

The simulator is based on state machines that allow to evaluate different events, cases (emergency calls), ambulance location, ambulance dispatch.

- The original version (UCSD) of the simulator is available at: https://github.com/EMSTrack/Algorithms
- The modified version of the simulator is available at: https://onx.la/75e83

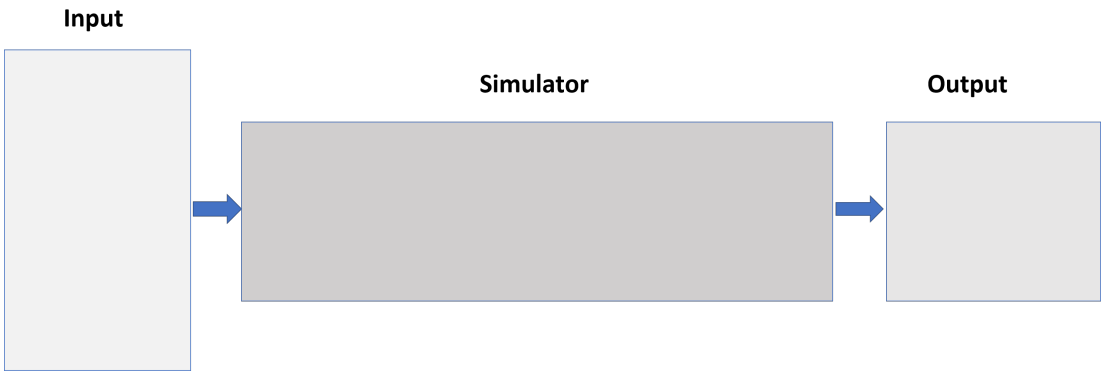


Figure 3: Simulator module.

5.1 Requirements to run the simulator module

Software requirements

Software recommendations	
System Operative	UNIX-based operating system such as Mac OS or Linux.
Integrated development environment	PyCharm
Python	Version 3.7

Table 1: Software recommendations

Setting up a Python interpreter in PyCharm

Open PyCharm and find the **Algorithm** folder that contains the project.

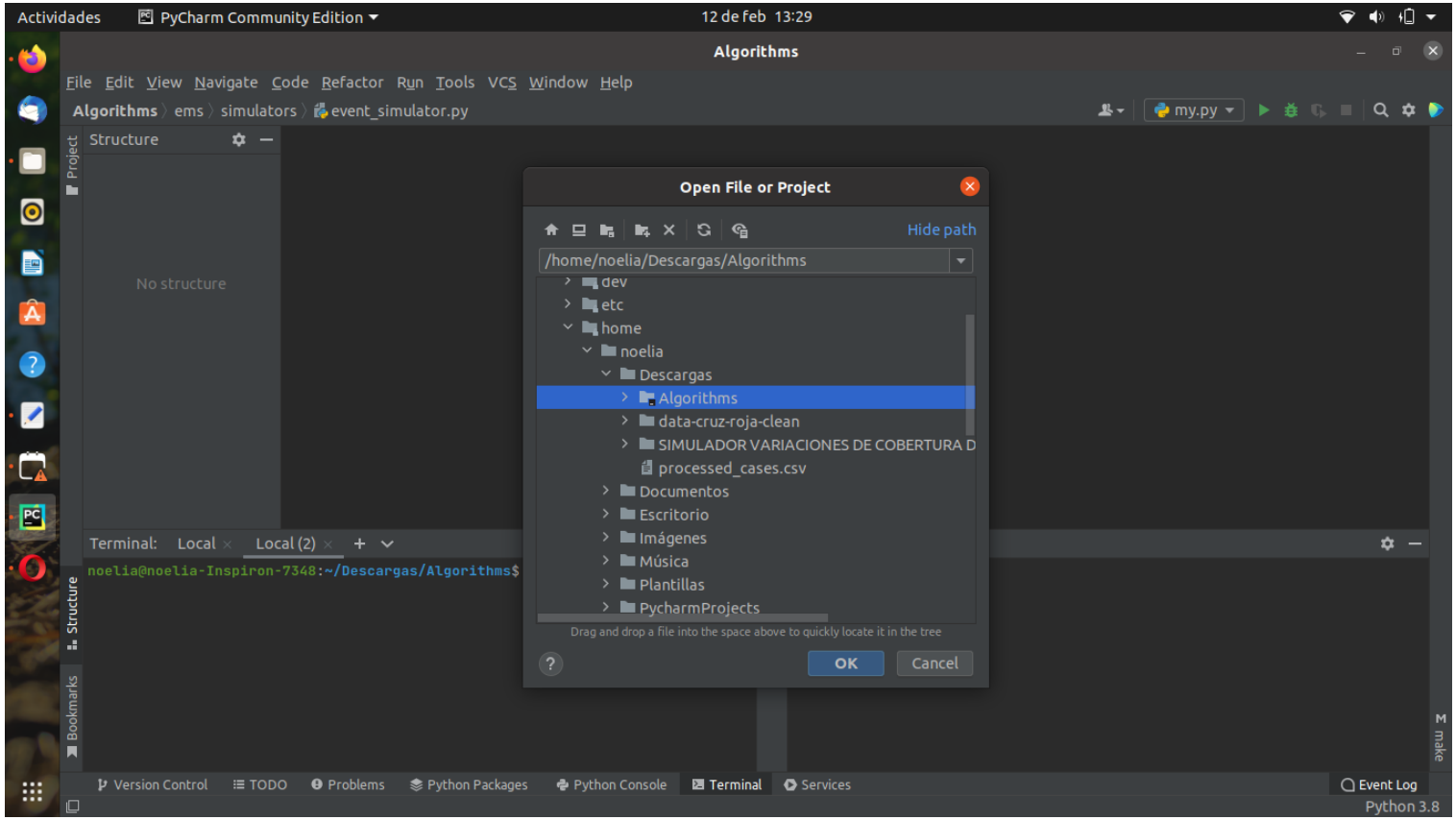


Figure 4: Initial PyCharm setup.

Setting up a Python interpreter in PyCharm

In PyCharm select **File –** $>$ **Settings**, in this window we will have to include the interpreter with which we want to work in the project.

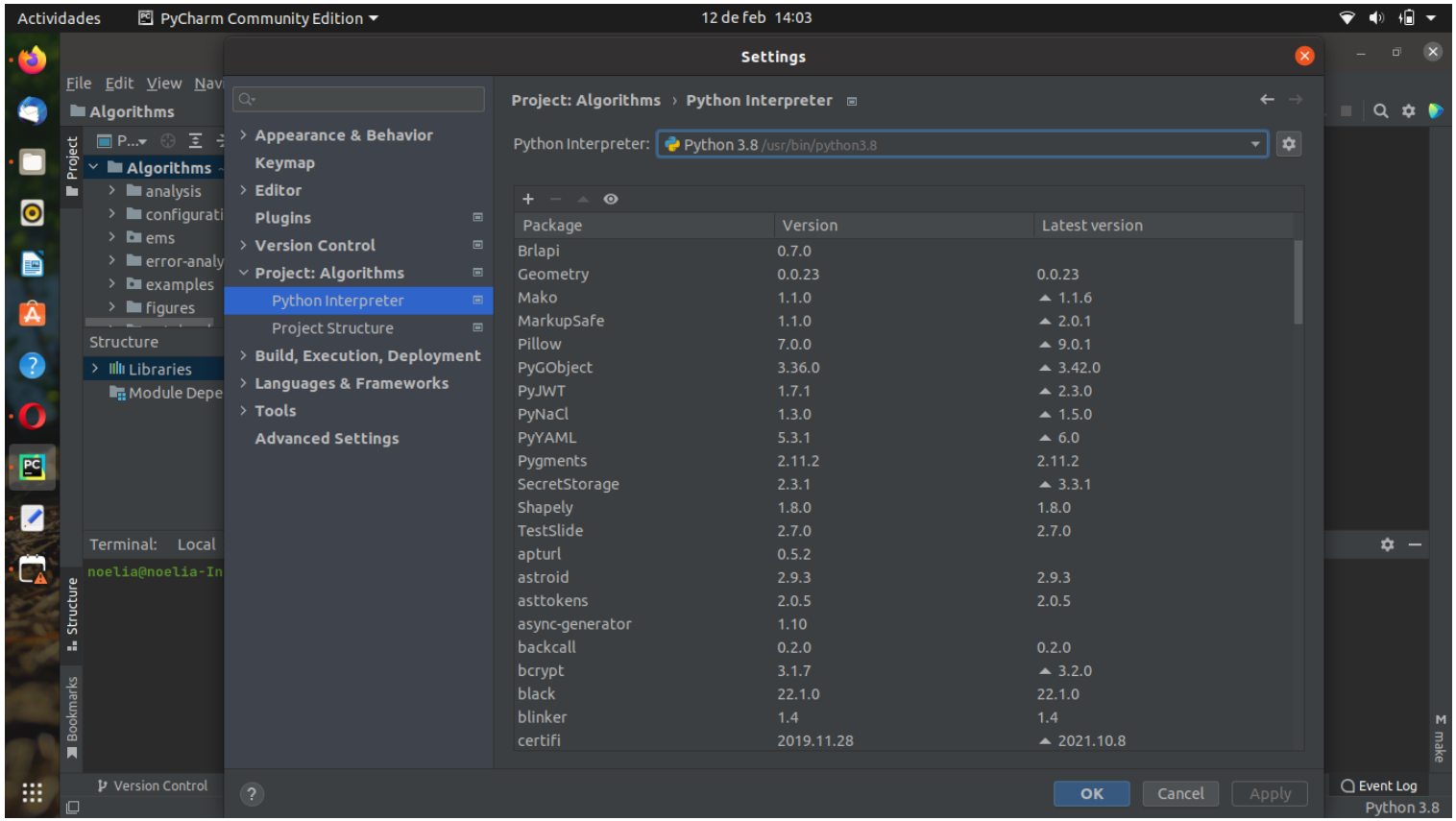


Figure 4: Initial PyCharm setup.

Installing Python packages

To download and install the packages required for the Python project. A text file (called requirements.txt) is used where the packages are located for automatic installation.

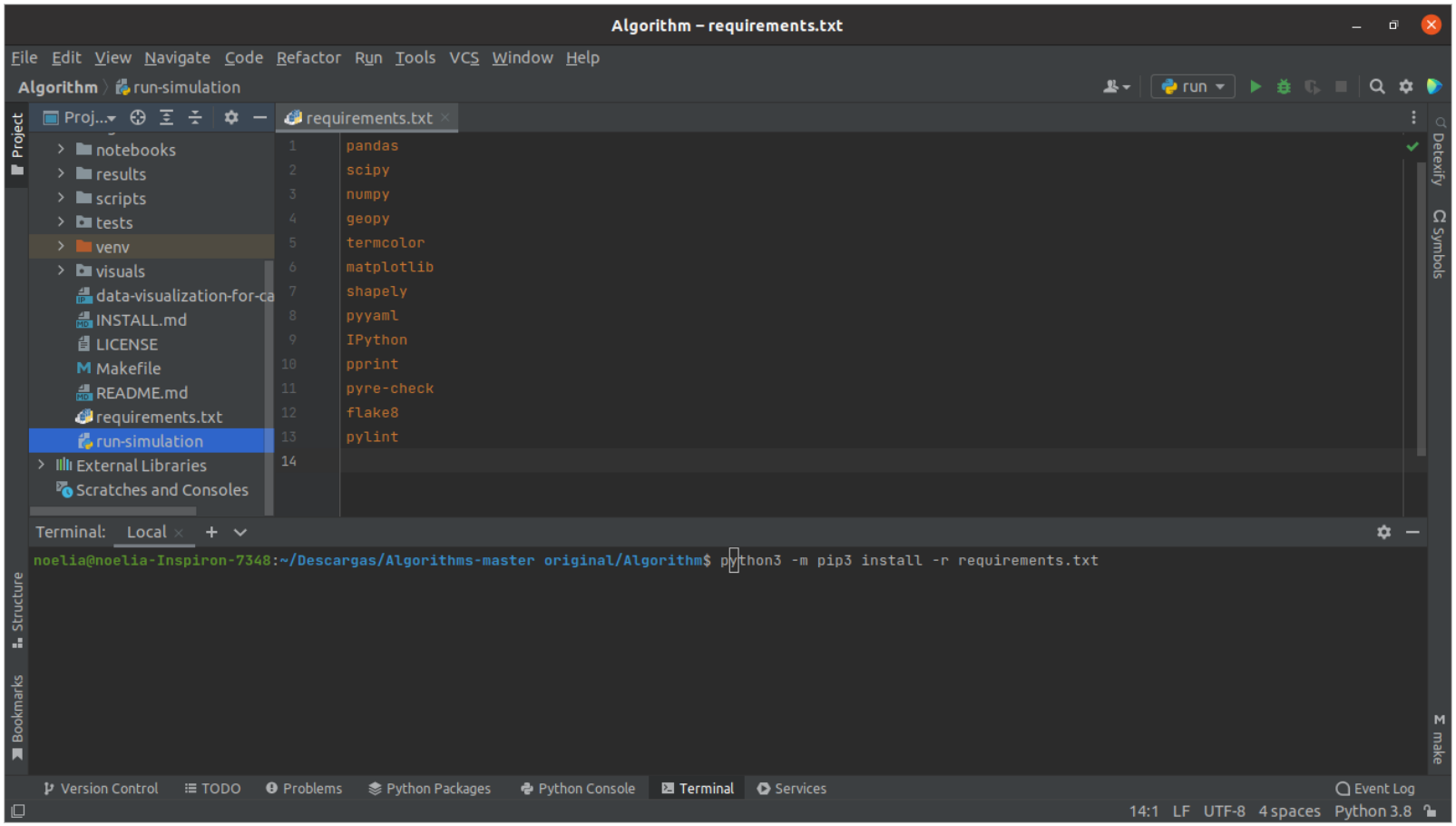


Figure 5: Installing Python packages.

Installing Python packages

Simply run the following command in the terminal:

python3 -m pip3 install -r requirements.txt

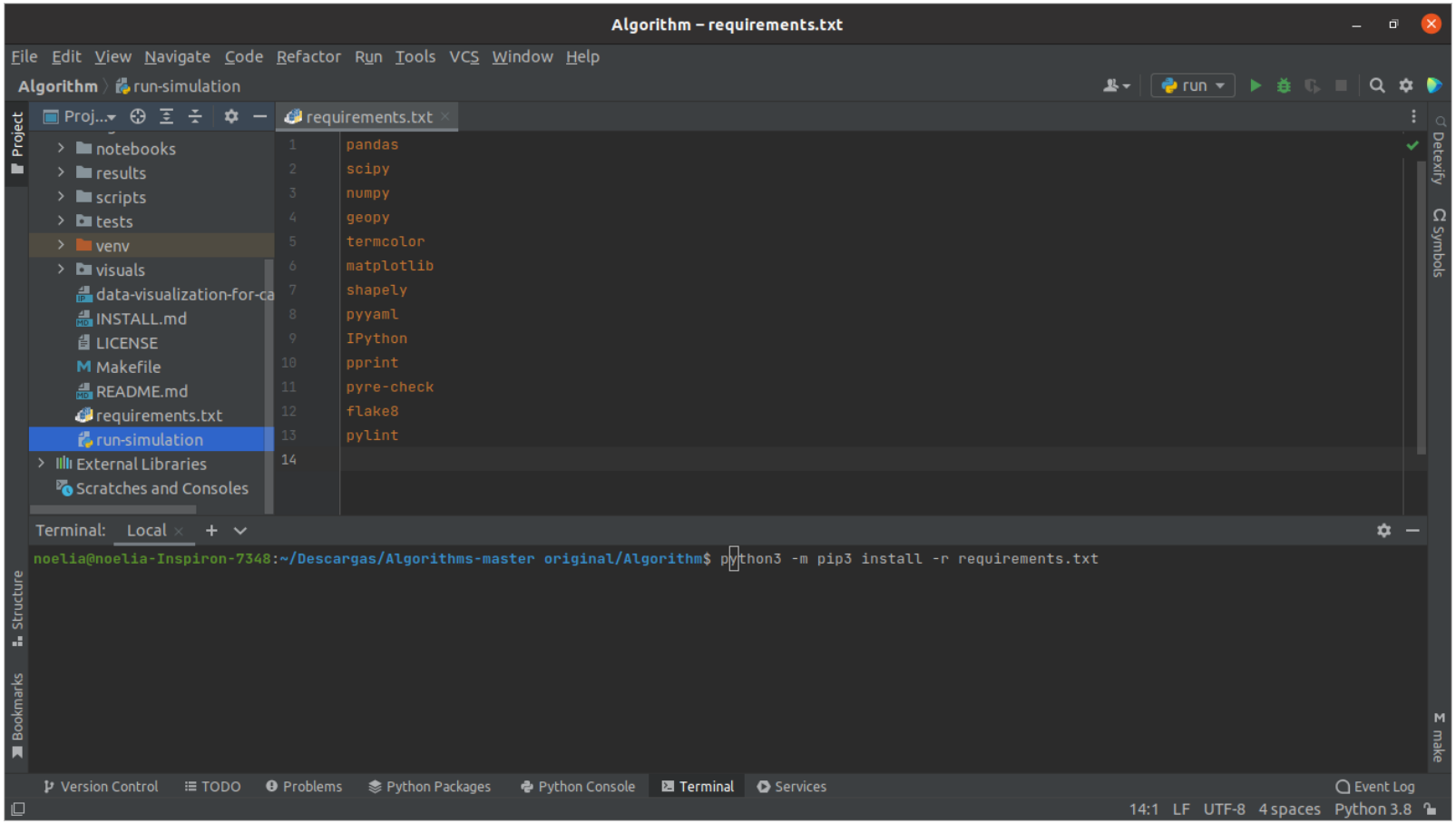


Figure 5: Installing Python packages

5. Simulator module (original version from UCSD)

Simulator module input

The simulator has an input where it requires the following information:

These are:

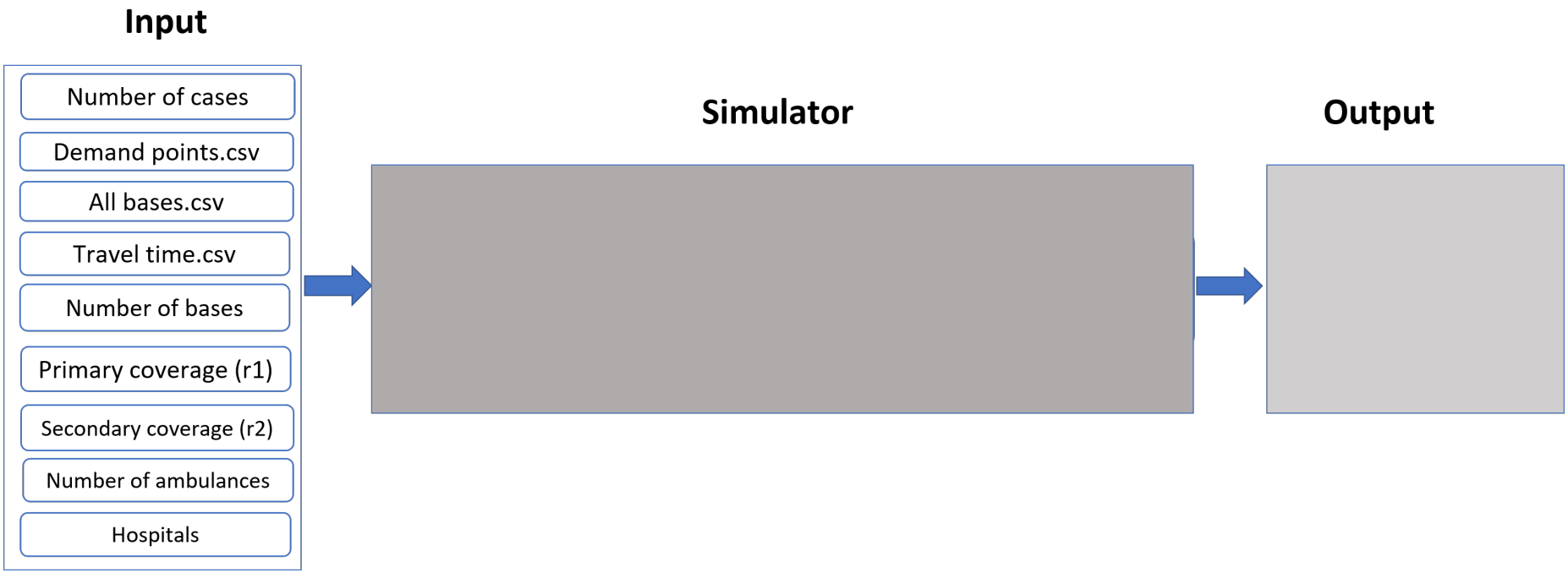


Figure 6: Simulator module.

Simulator module input

Number of cases : the number of cases (emergency calls) to be simulated.

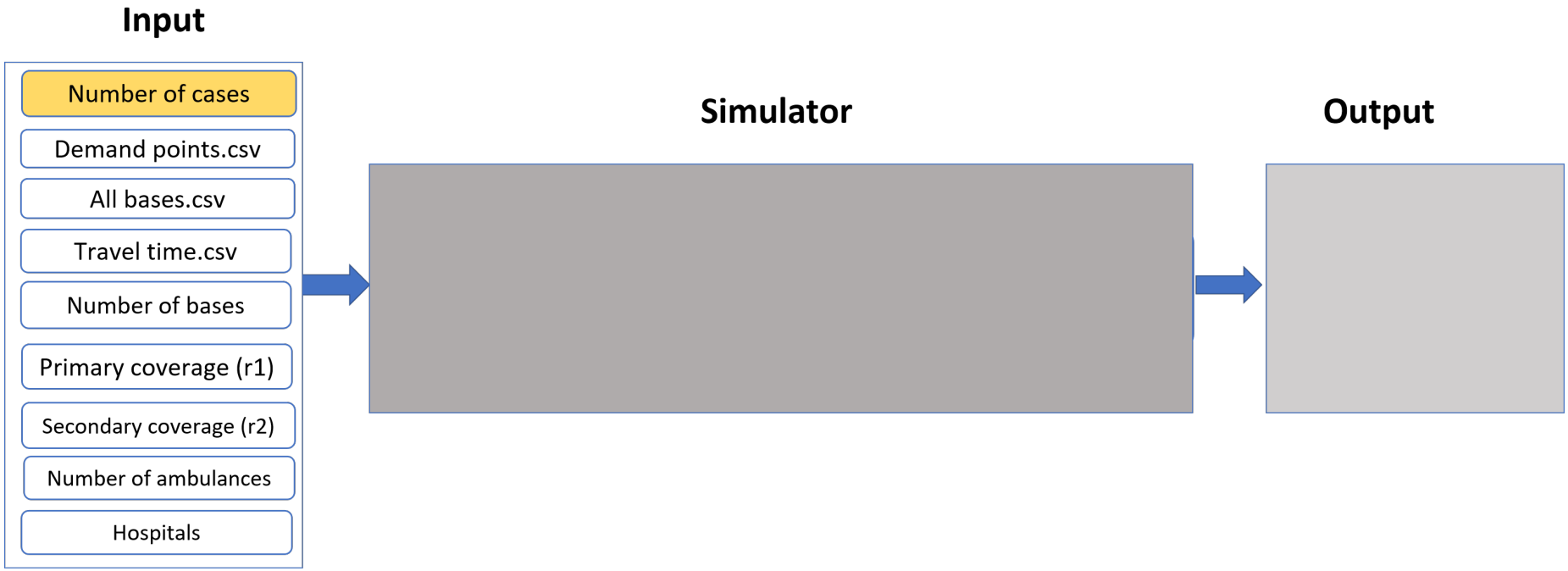


Figure 6: Simulator module.

Simulator module input

These files are available at:

https://onx.la/7936c

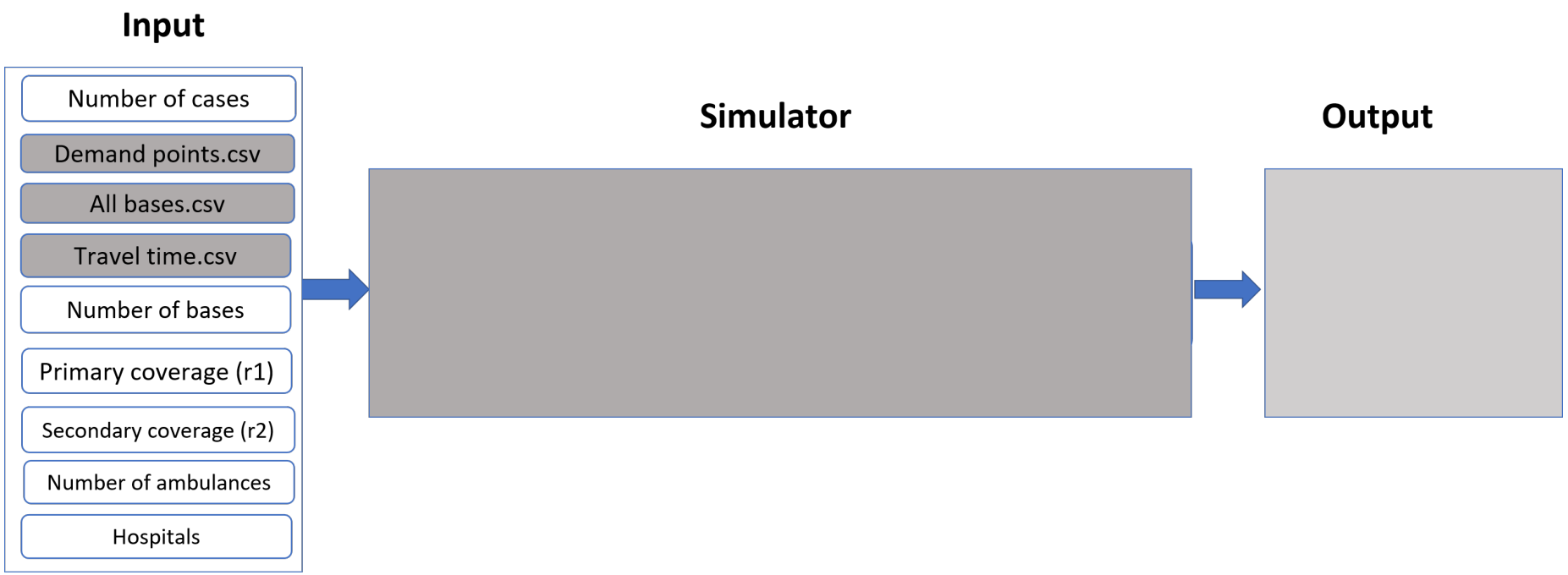


Figure 6: Simulator module.

Simulator module input

Demand points: are the group of calls determined by a clustering algorithm that generates the results in a *.csv file.

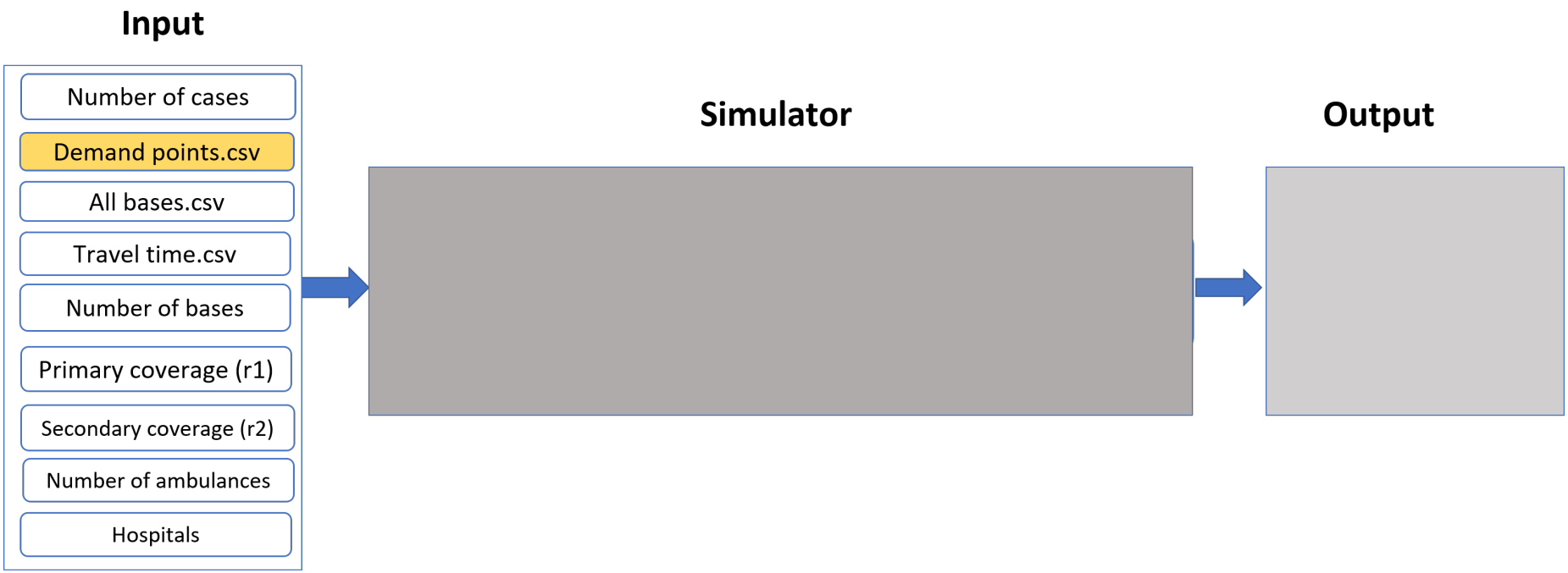
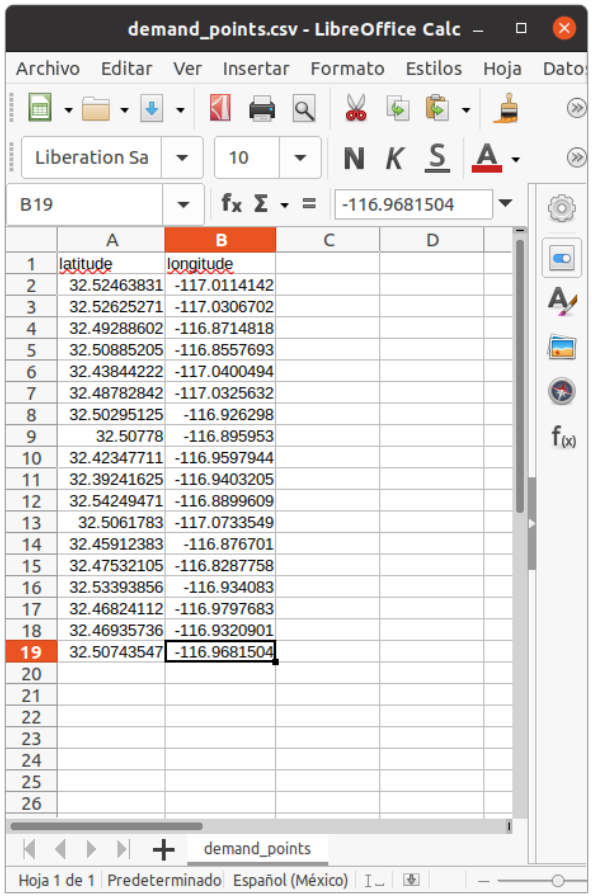


Figure 6: Simulator module.

Simulator module input

The file contains the geographic coordinate (latitude, longitude) of the location of the demand points.



	A	B	C	D
1	latitude	longitude		
2	32.52463831	-117.0114142		
3	32.52625271	-117.0306702		
4	32.49288602	-116.8714818		
5	32.50885205	-116.8557693		
6	32.43844222	-117.0400494		
7	32.48782842	-117.0325632		
8	32.50295125	-116.926298		
9	32.50778	-116.895953		
10	32.42347711	-116.9597944		
11	32.39241625	-116.9403205		
12	32.54249471	-116.8899609		
13	32.5061783	-117.0733549		
14	32.45912383	-116.876701		
15	32.47532105	-116.8287758		
16	32.53393856	-116.934083		
17	32.46824112	-116.9797683		
18	32.46935736	-116.9320901		
19	32.50743547	-116.9681504		
20				
21				
22				
23				
24				
25				
26				

Figure 7: Demand points file.

Simulator module input

All bases: 1691 locations (schools, fire stations, etc.) are considered candidate base locations and are available in a csv file.

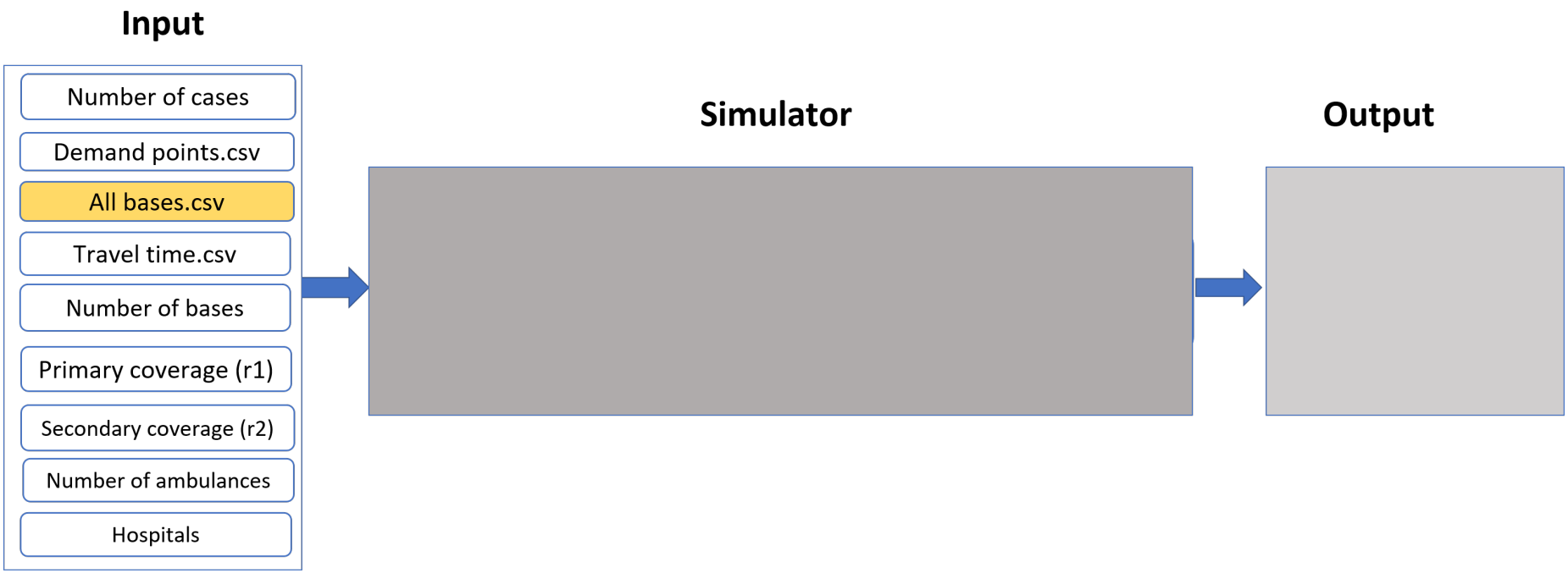
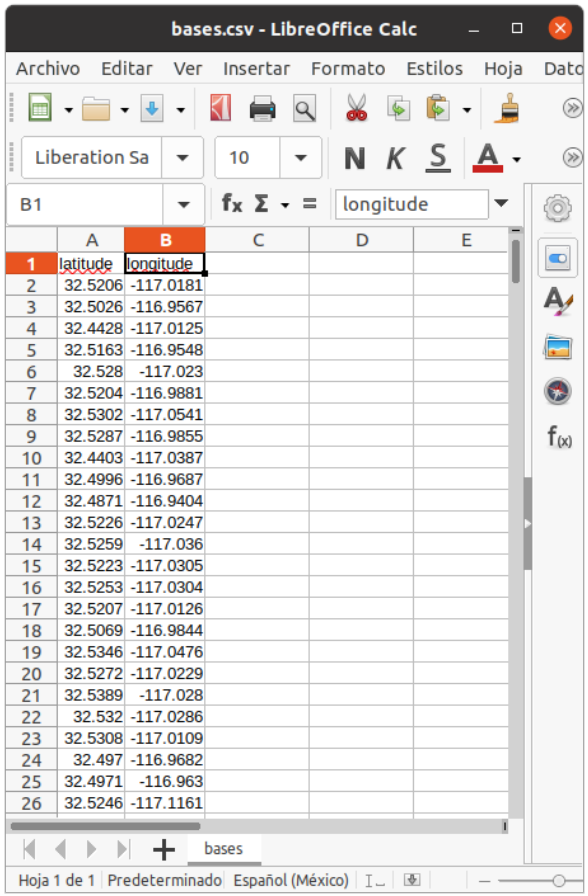


Figure 8: Simulator module.

Simulator module input

The file contains the geographic coordinate (latitude, longitude) of the location of the 1691 places, as possible bases.



	A	B	C	D	E
1	latitude	longitude			
2	32.5206	-117.0181			
3	32.5026	-116.9567			
4	32.4428	-117.0125			
5	32.5163	-116.9548			
6	32.528	-117.023			
7	32.5204	-116.9881			
8	32.5302	-117.0541			
9	32.5287	-116.9855			
10	32.4403	-117.0387			
11	32.4996	-116.9687			
12	32.4871	-116.9404			
13	32.5226	-117.0247			
14	32.5259	-117.036			
15	32.5223	-117.0305			
16	32.5253	-117.0304			
17	32.5207	-117.0126			
18	32.5069	-116.9844			
19	32.5346	-117.0476			
20	32.5272	-117.0229			
21	32.5389	-117.028			
22	32.532	-117.0286			
23	32.5308	-117.0109			
24	32.497	-116.9682			
25	32.4971	-116.963			
26	32.5246	-117.1161			

Figure 9: 1691 places as candidate bases.

Simulator module input

Travel time: A matrix of travel times (seconds) calculated with Google Maps, of the 1691 potential bases to each demand point, available in csv file.

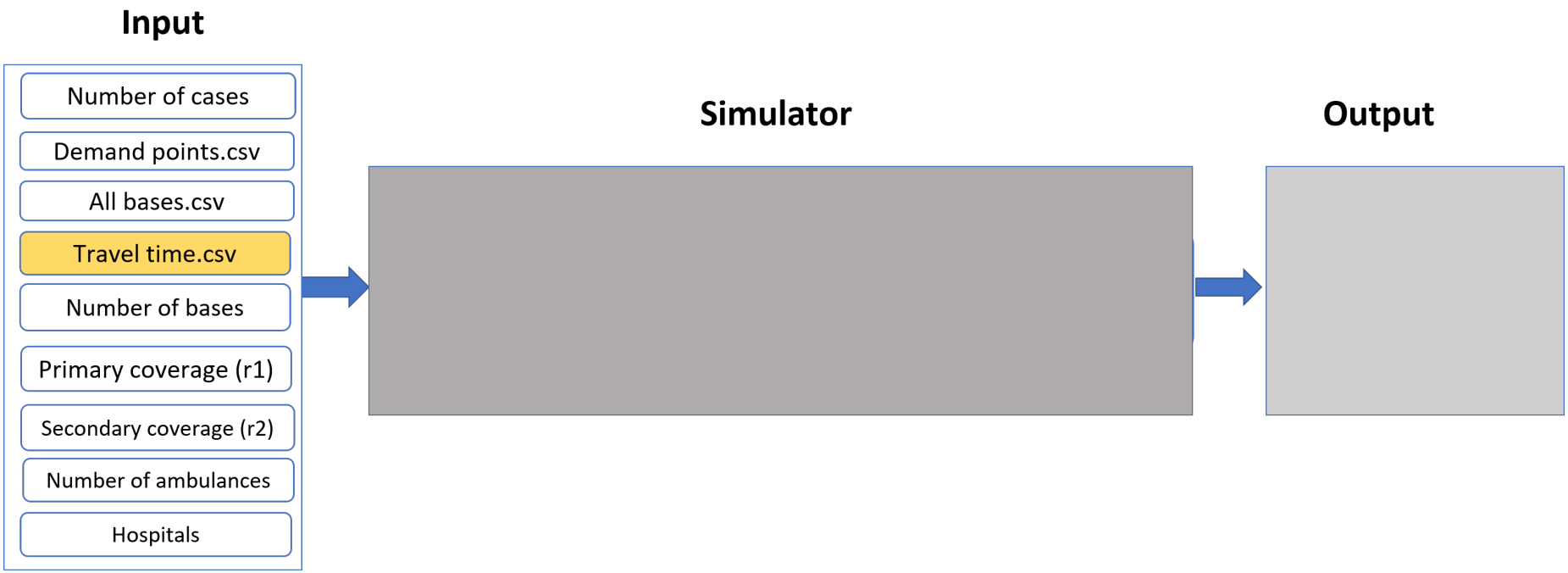
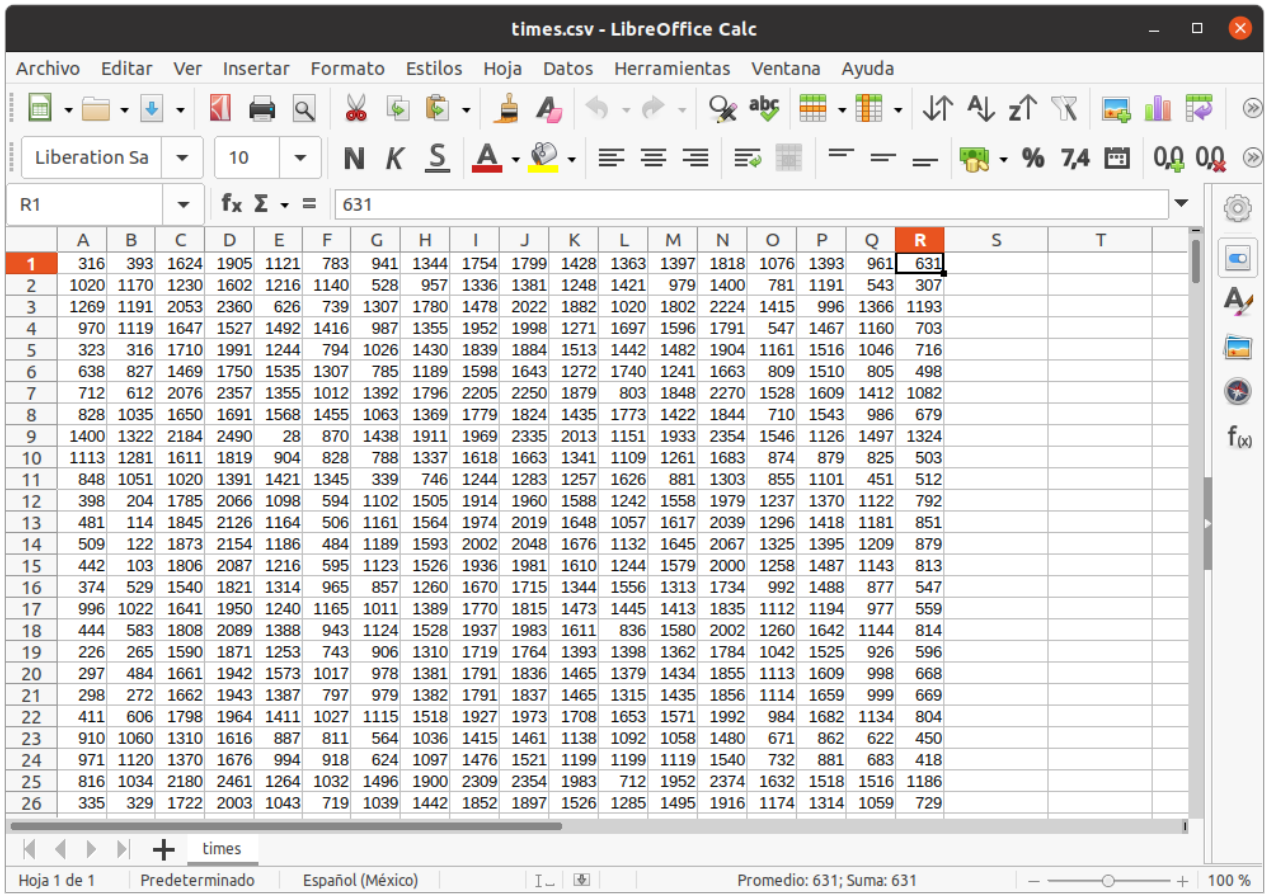


Figure 10: Simulator module.

Simulator module input

The file contains the travel times in seconds of the 1691 candidate bases to each demand point.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	316	393	1624	1905	1121	783	941	1344	1754	1799	1428	1363	1397	1818	1076	1393	961	631		
2	1020	1170	1230	1602	1216	1140	528	957	1336	1381	1248	1421	979	1400	781	1191	543	307		
3	1269	1191	2053	2360	626	739	1307	1780	1478	2022	1882	1020	1802	2224	1415	996	1366	1193		
4	970	1119	1647	1527	1492	1416	987	1355	1952	1998	1271	1697	1596	1791	547	1467	1160	703		
5	323	316	1710	1991	1244	794	1026	1430	1839	1884	1513	1442	1482	1904	1161	1516	1046	716		
6	638	827	1469	1750	1535	1307	785	1189	1598	1643	1272	1740	1241	1663	809	1510	805	498		
7	712	612	2076	2357	1355	1012	1392	1796	2205	2250	1879	803	1848	2270	1528	1609	1412	1082		
8	828	1035	1650	1691	1568	1455	1063	1369	1779	1824	1435	1773	1422	1844	710	1543	986	679		
9	1400	1322	2184	2490	28	870	1438	1911	1969	2335	2013	1151	1933	2354	1546	1126	1497	1324		
10	1113	1281	1611	1819	904	828	788	1337	1618	1663	1341	1109	1261	1683	874	879	825	503		
11	848	1051	1020	1391	1421	1345	339	746	1244	1283	1257	1626	881	1303	855	1101	451	512		
12	398	204	1785	2066	1098	594	1102	1505	1914	1960	1588	1242	1558	1979	1237	1370	1122	792		
13	481	114	1845	2126	1164	506	1161	1564	1974	2019	1648	1057	1617	2039	1296	1418	1181	851		
14	509	122	1873	2154	1186	484	1189	1593	2002	2048	1676	1132	1645	2067	1325	1395	1209	879		
15	442	103	1806	2087	1216	595	1123	1526	1936	1981	1610	1244	1579	2000	1258	1487	1143	813		
16	374	529	1540	1821	1314	965	857	1260	1670	1715	1344	1556	1313	1734	992	1488	877	547		
17	996	1022	1641	1950	1240	1165	1011	1389	1770	1815	1473	1445	1413	1835	1112	1194	977	559		
18	444	583	1808	2089	1388	943	1124	1528	1937	1983	1611	836	1580	2002	1260	1642	1144	814		
19	226	265	1590	1871	1253	743	906	1310	1719	1764	1393	1398	1362	1784	1042	1525	926	596		
20	297	484	1661	1942	1573	1017	978	1381	1791	1836	1465	1379	1434	1855	1113	1609	998	668		
21	298	272	1662	1943	1387	797	979	1382	1791	1837	1465	1315	1435	1856	1114	1659	999	669		
22	411	606	1798	1964	1411	1027	1115	1518	1927	1973	1708	1653	1571	1992	984	1682	1134	804		
23	910	1060	1310	1616	887	811	564	1036	1415	1461	1138	1092	1058	1480	671	862	622	450		
24	971	1120	1370	1676	994	918	624	1097	1476	1521	1199	1199	1119	1540	732	881	683	418		
25	816	1034	2180	2461	1264	1032	1496	1900	2309	2354	1983	712	1952	2374	1632	1518	1516	1186		
26	335	329	1722	2003	1043	719	1039	1442	1852	1897	1526	1285	1495	1916	1174	1314	1059	729		

Figure 11: Travel time file.

Simulator module input

Number of bases: It is the number of the best bases to be determined from the set of bases.

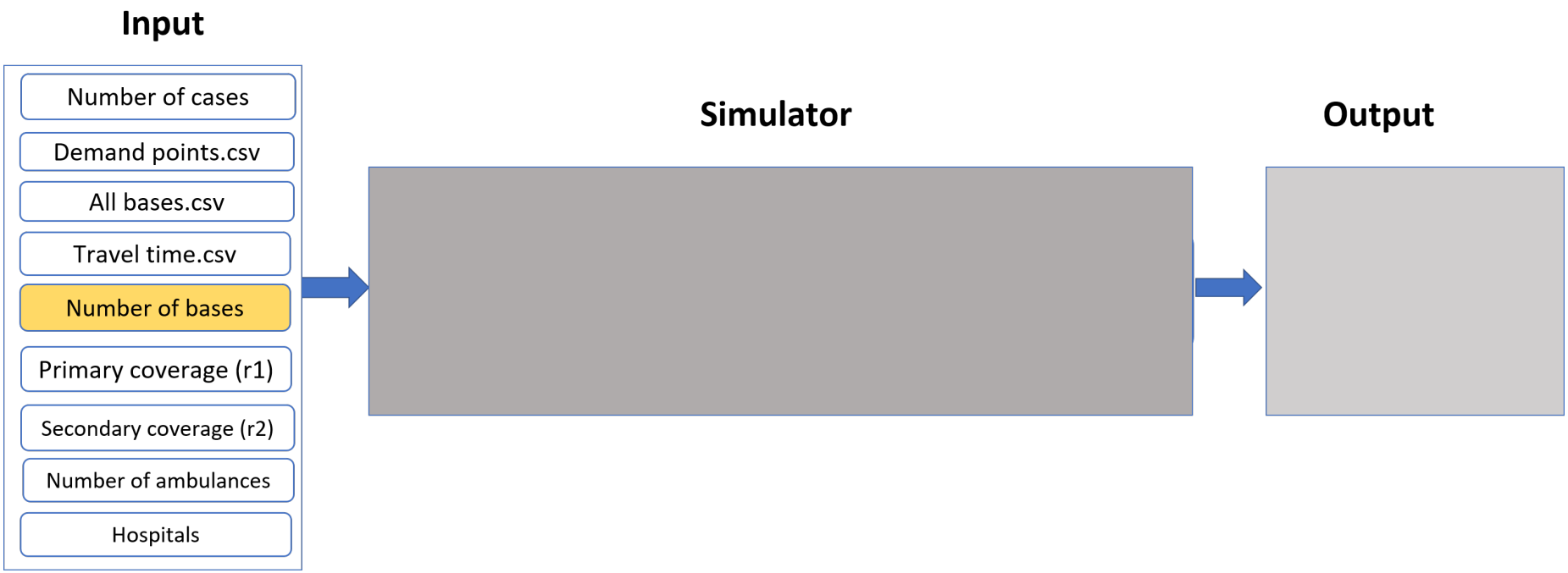


Figure 12: Simulator module.

Simulator module input

Radius 1 (r1): The minimum time, in seconds, that the ambulance can take to cover a point of demand.

Radius 2 (r2): The maximum time, in seconds, that the ambulance can take to cover a point of demand.

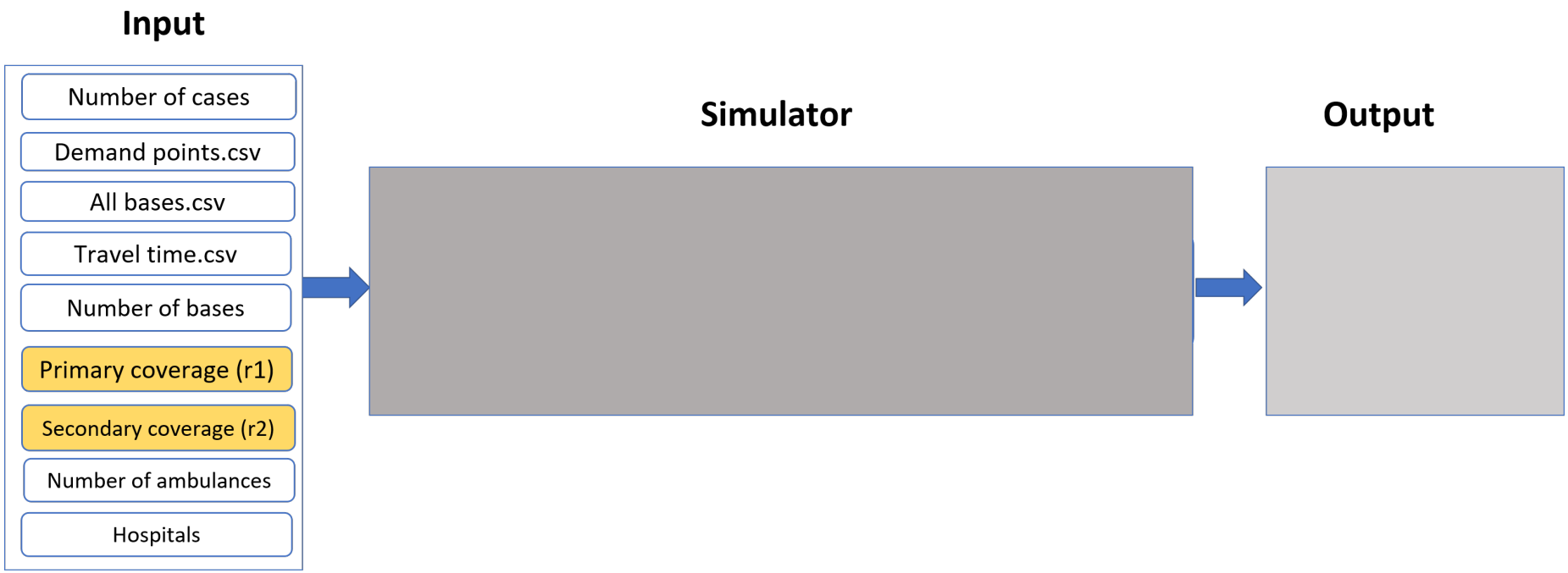


Figure 13: Simulator module.

Simulator module input

Radius 1 (r1): The minimum time, in seconds, that the ambulance can take to cover a point of demand.

Radius 2 (r2): The maximum time, in seconds, that the ambulance can take to cover a point of demand.

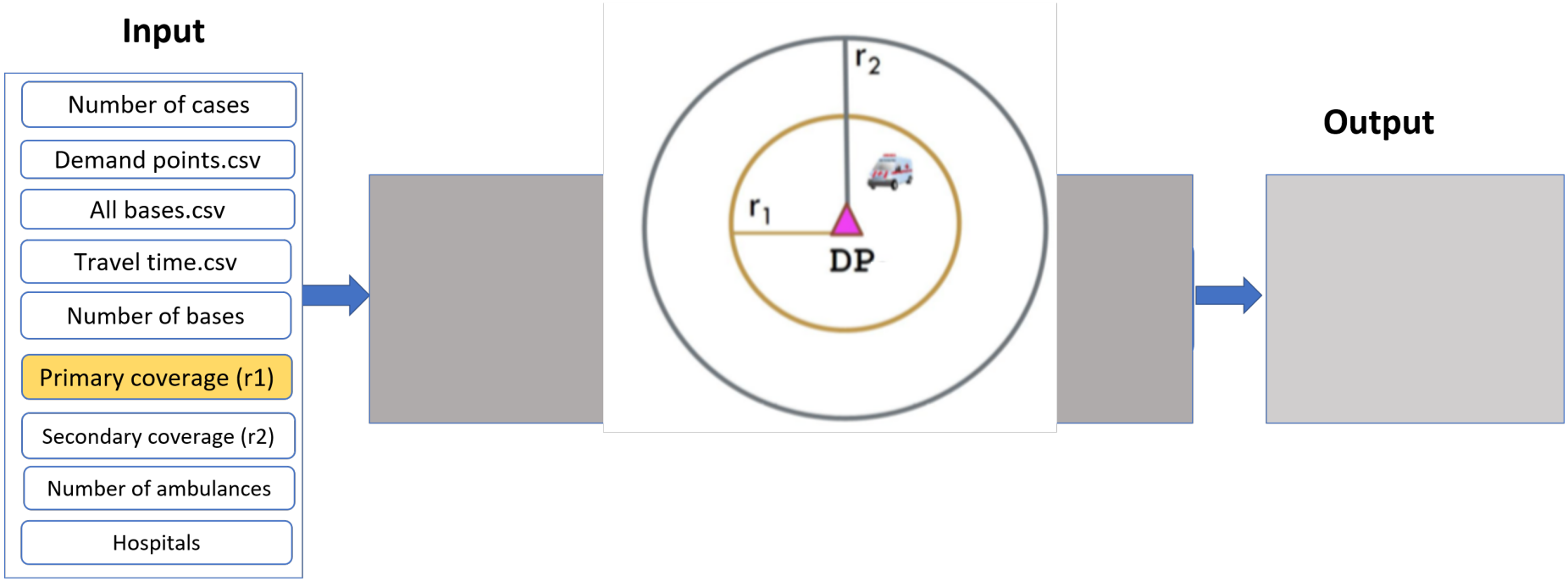


Figure 13: Simulator module.

Simulator module input

Radius 1 (r1): The minimum time, in seconds, that the ambulance can take to cover a point of demand.

Radius 2 (r2): The maximum time, in seconds, that the ambulance can take to cover a point of demand.

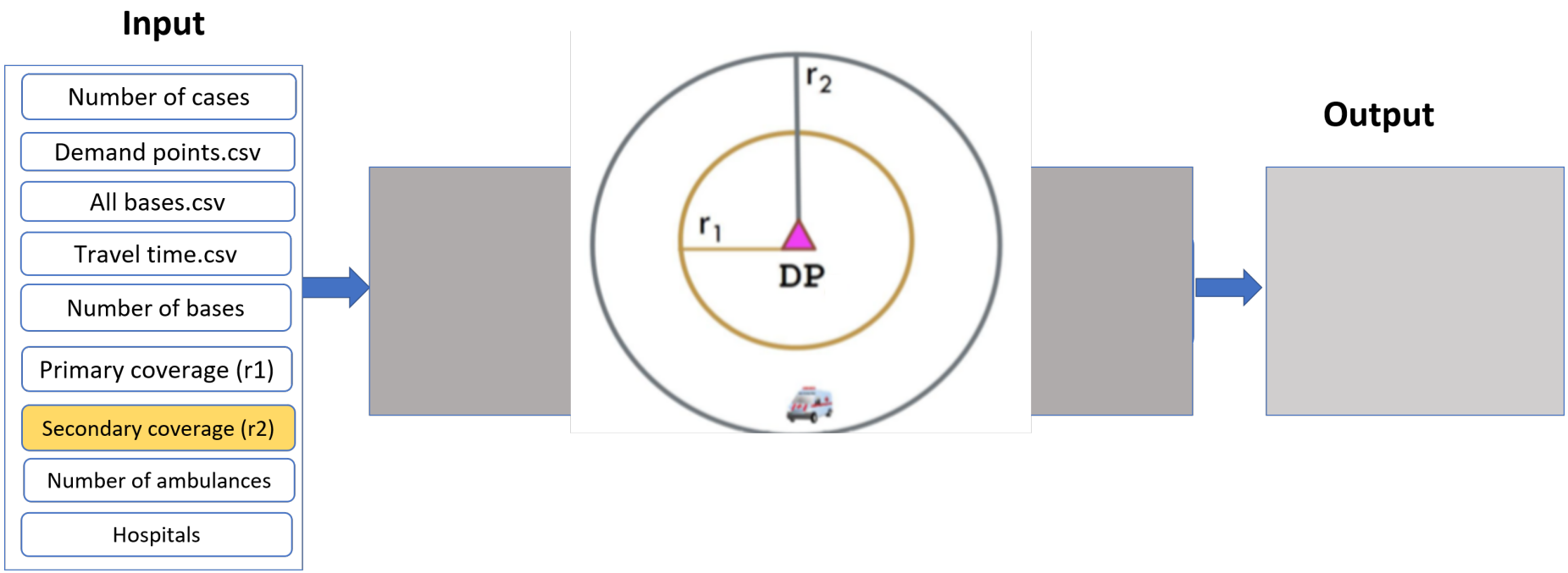


Figure 13: Simulator module.

Simulator module input

Number of ambulance: The number of ambulances available, this data must coincide with the number of bases established, so that each base has an ambulance.

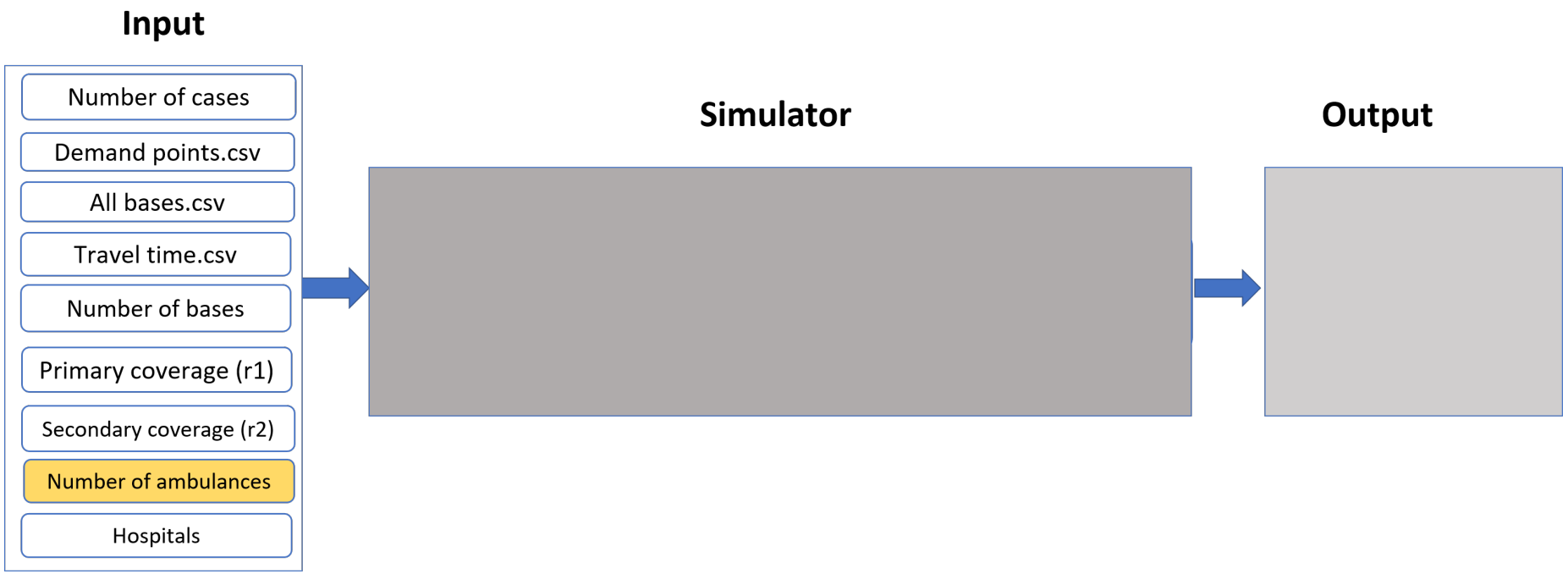


Figure 13: Simulator module.

Simulator module input

Hospitals: hospital locations (coordinates)

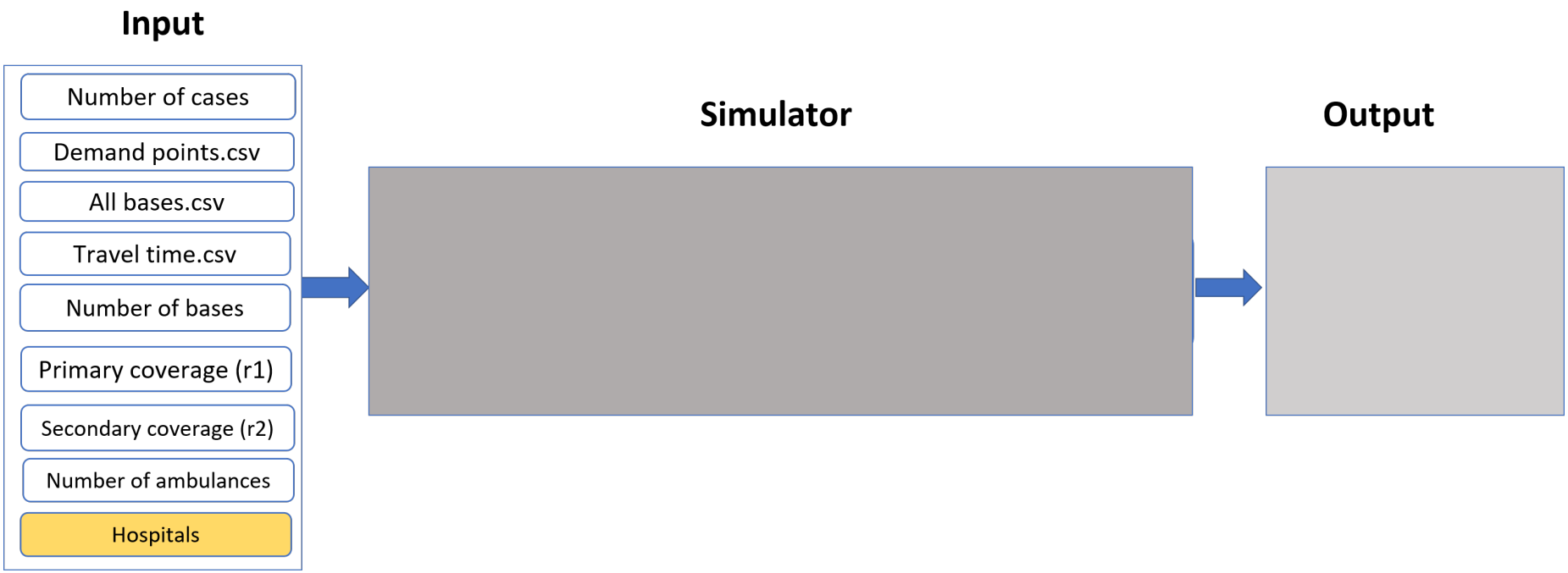
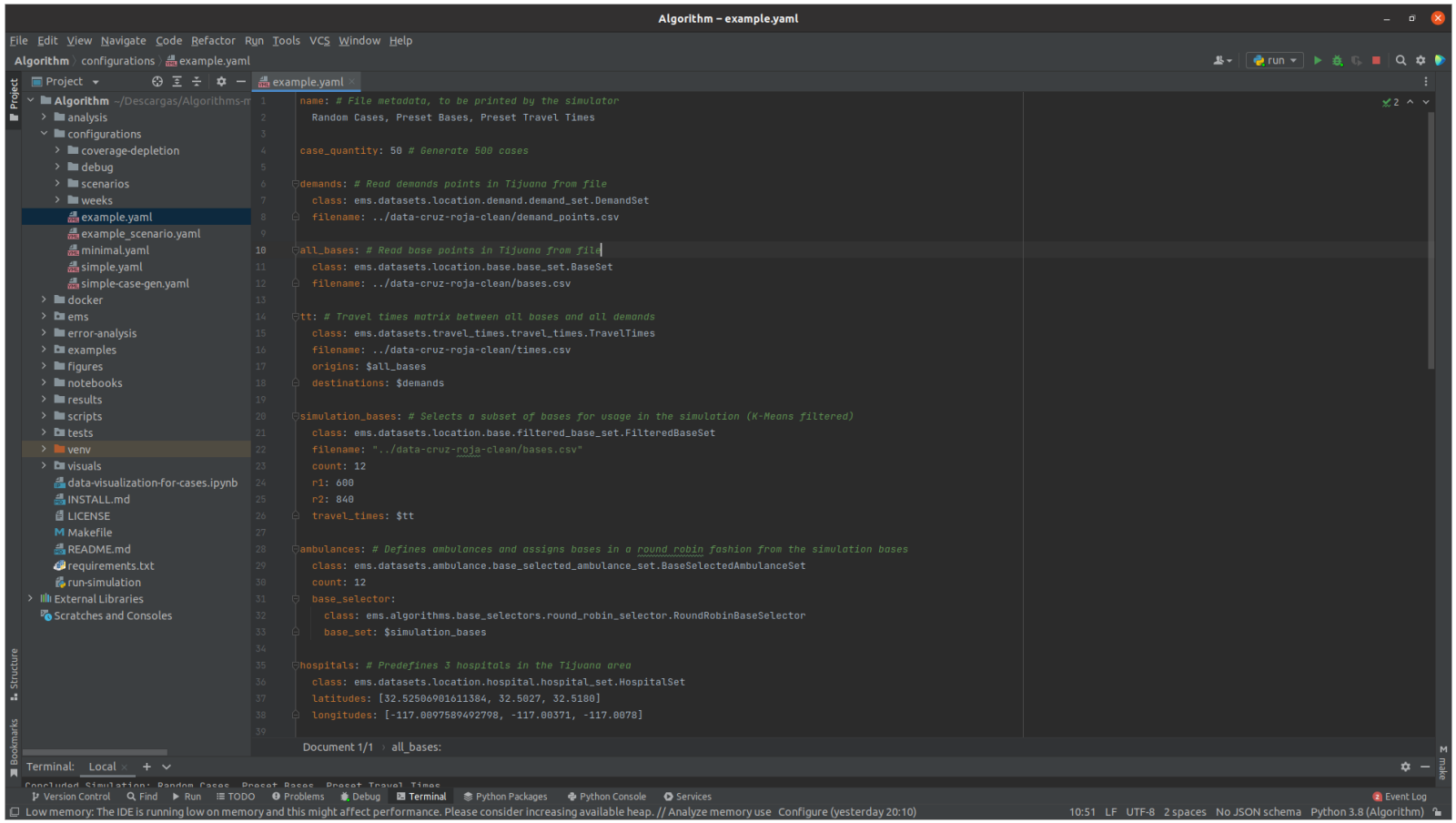


Figure 13: Simulator module.

Simulator input

The data required as input to the simulator is located in the **configurations – > example.yaml**

Watch video



Simulator process and output

In this block, the number of bases is determined from a set of bases. The number of **bases** must be equal to the **number of ambulances**, so that all bases are occupied. Finally, it is determined if there is primary coverage (r1) and secondary coverage (r2).

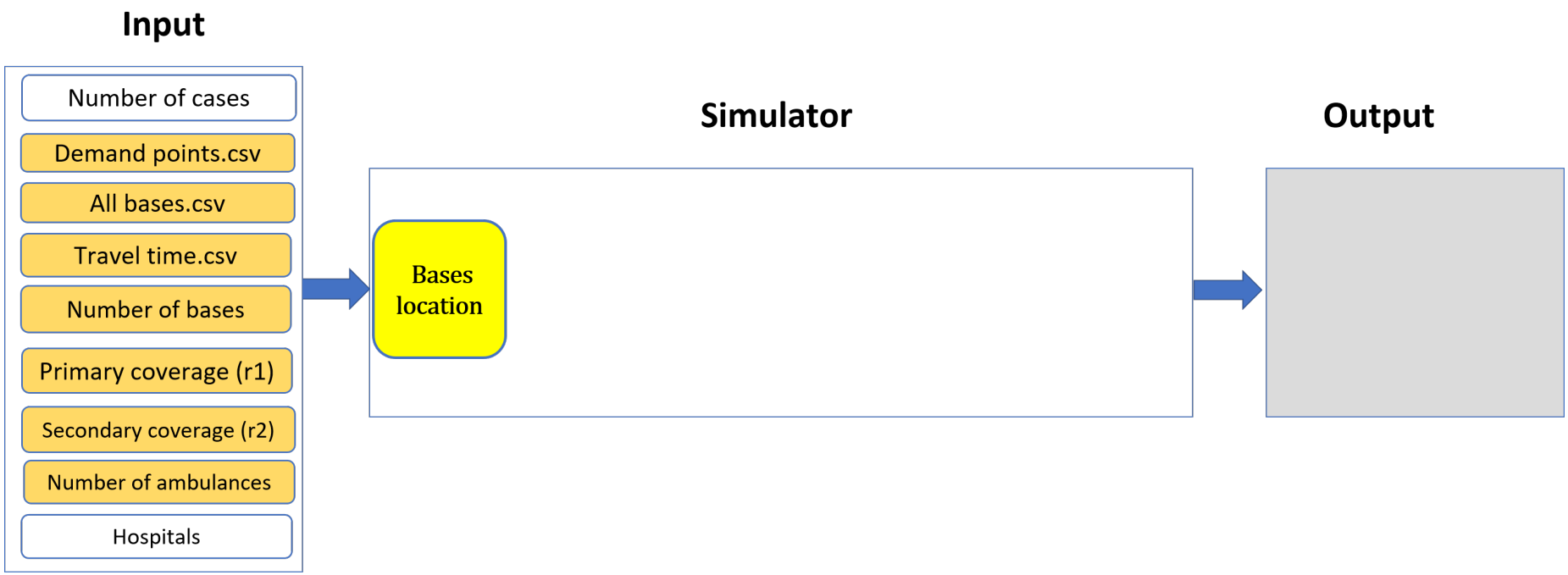


Figure 14: Simulator module.

Simulator process and output

With this block you will obtain the chosen_ambulances.csv file, which contains the coordinates of the selected bases.

The file is saved in **results - > Random Cases, Preset Bases, Preset Travel Times - > chosen_bases.csv**.

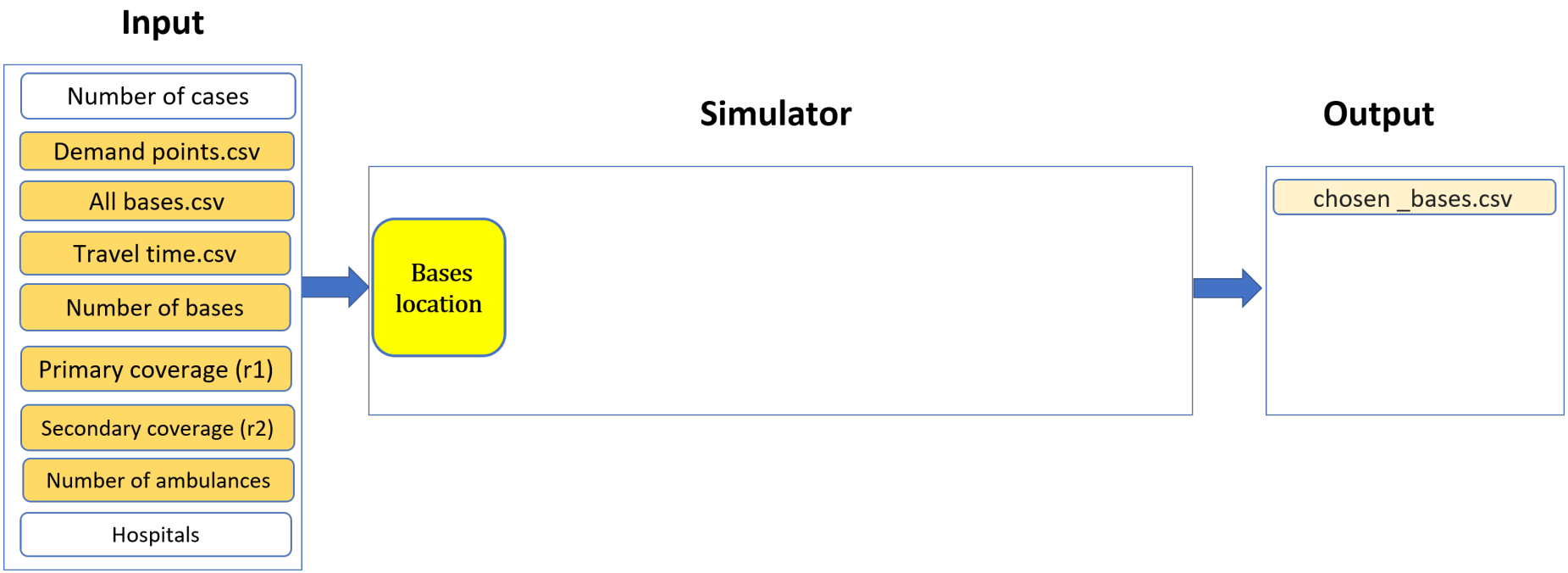


Figure 14: Simulator module.

Simulator process and output

The run-simulation.py file is responsible for calling the blocks in the example.yaml file following a sequence. In this case only the **simulation_bases** block.

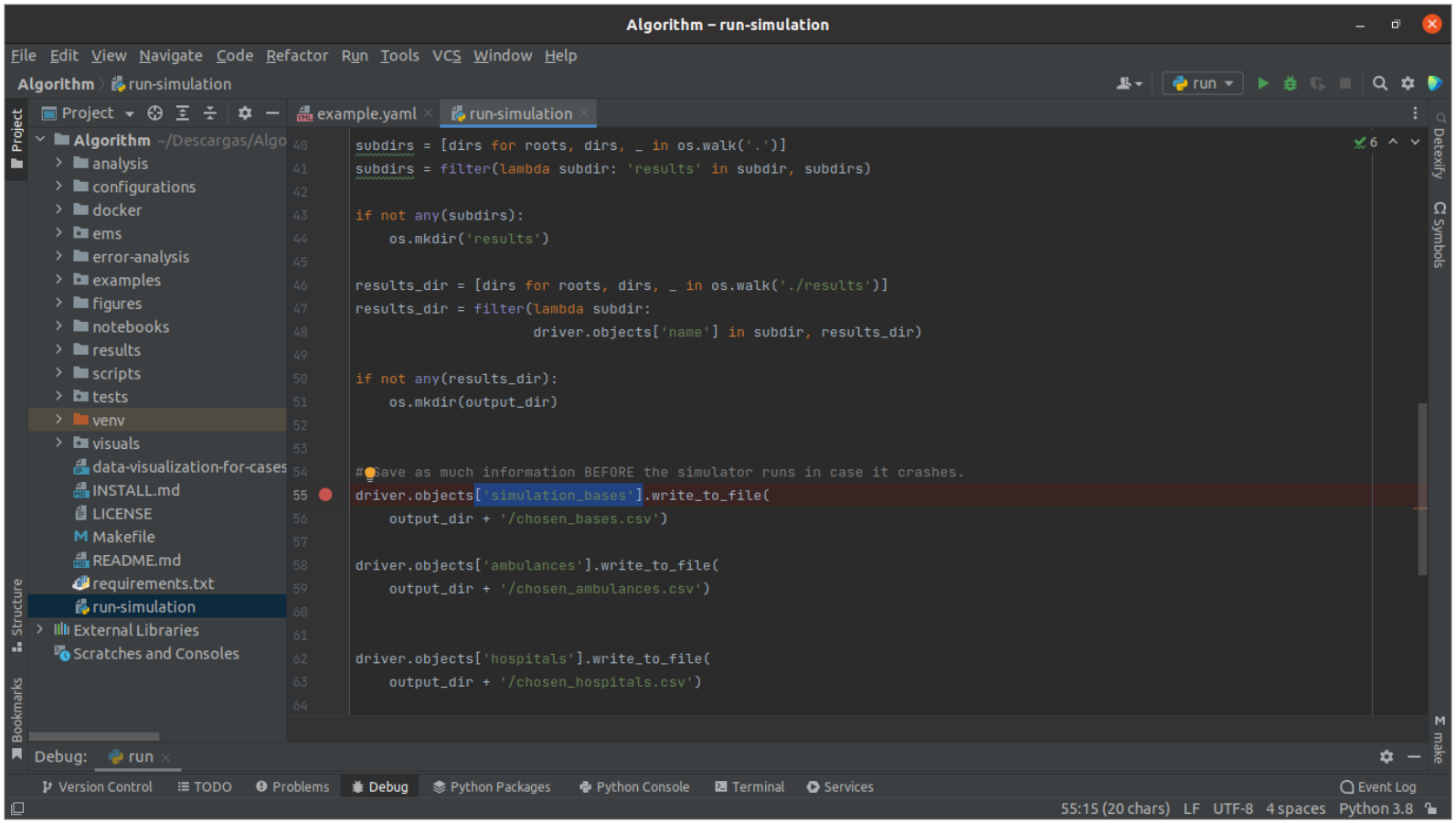


Figure 15: run-simulation.py source code

Simulator process and output

The run-simulation.py file is responsible for calling the blocks in the example.yaml file following a sequence. In this case only the **simulation_bases** block.

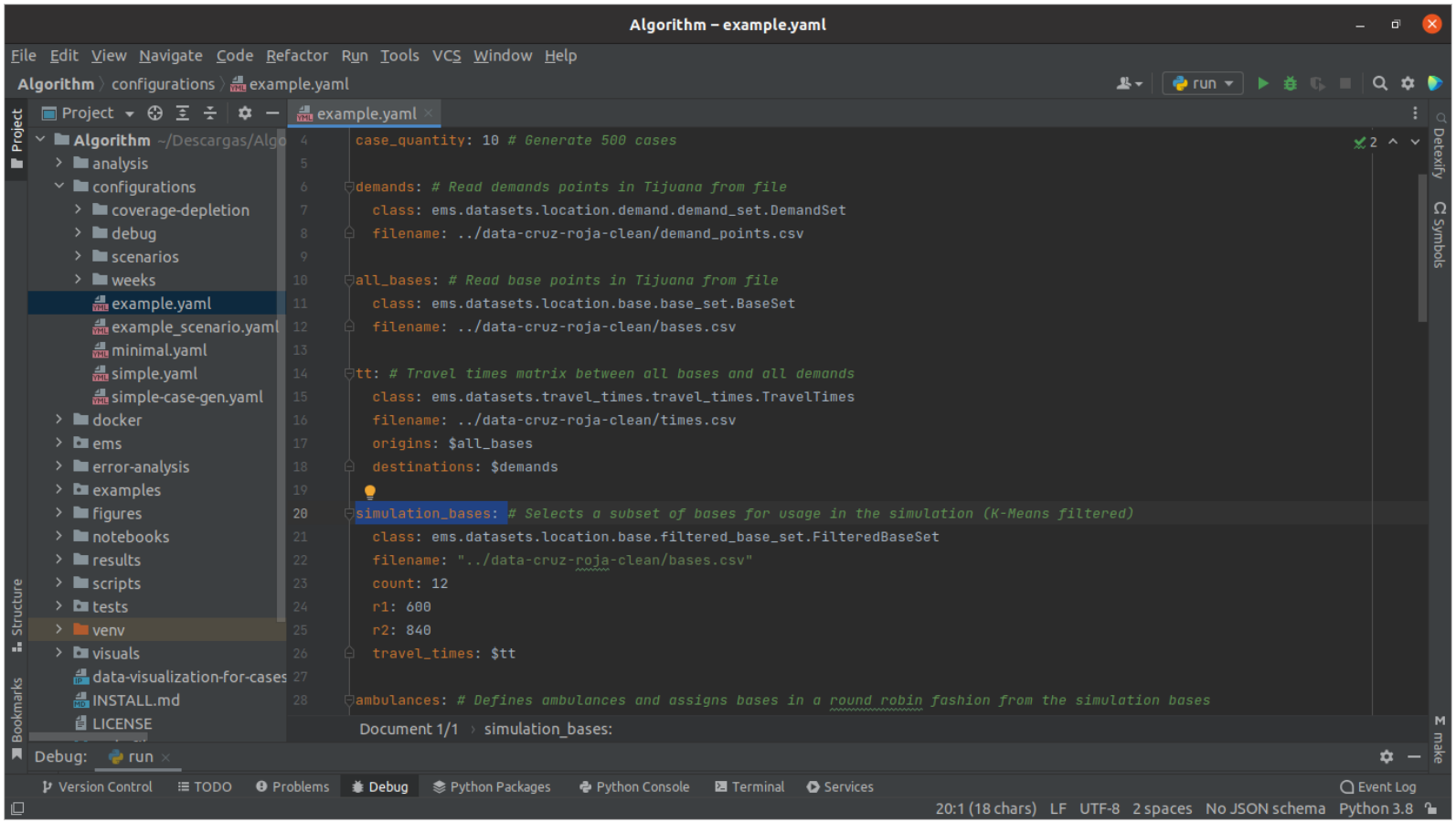
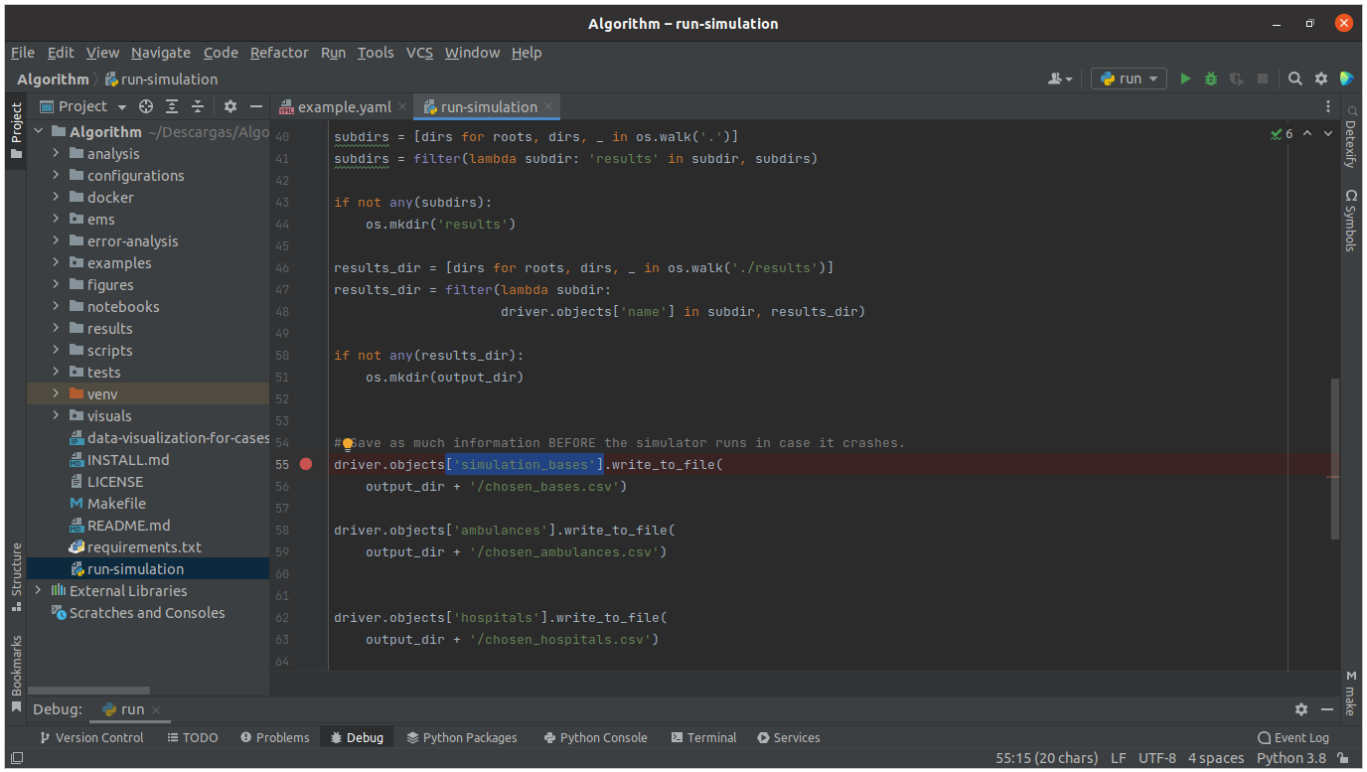


Figure 15: run-simulation.py source code

Simulator process and output

The video shows the files needed to run this first block, in this point, you can stop and get these first results.

Watch video



Simulator process and output

With this block the ambulances are defined by assigning an ambulance to each base, saving the location coordinates of the ambulances in chosen_ambulances.csv, the file is saved in **results - > Random Cases, Preset Bases, Preset Travel Times- > chosen_ambulances.csv**.

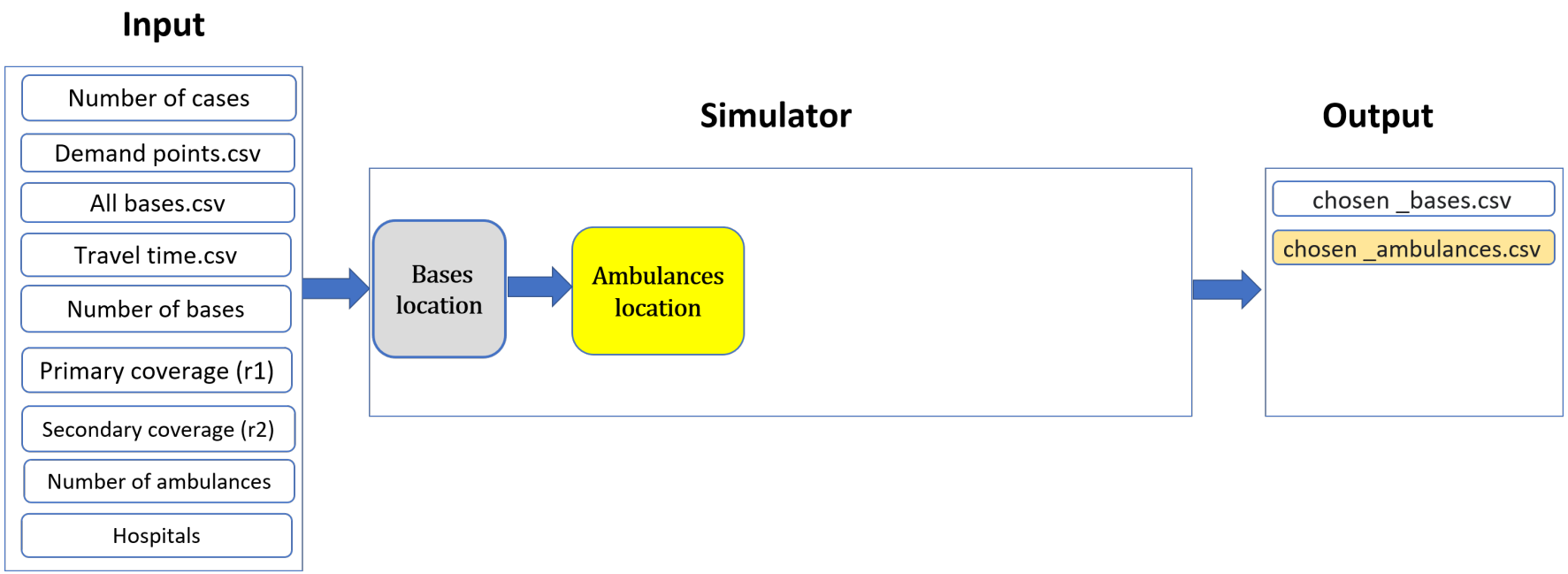


Figure 16: Simulator module.

Simulator process and output

In this case, with the information already generated from the base location block, the following ambulances block is executed.

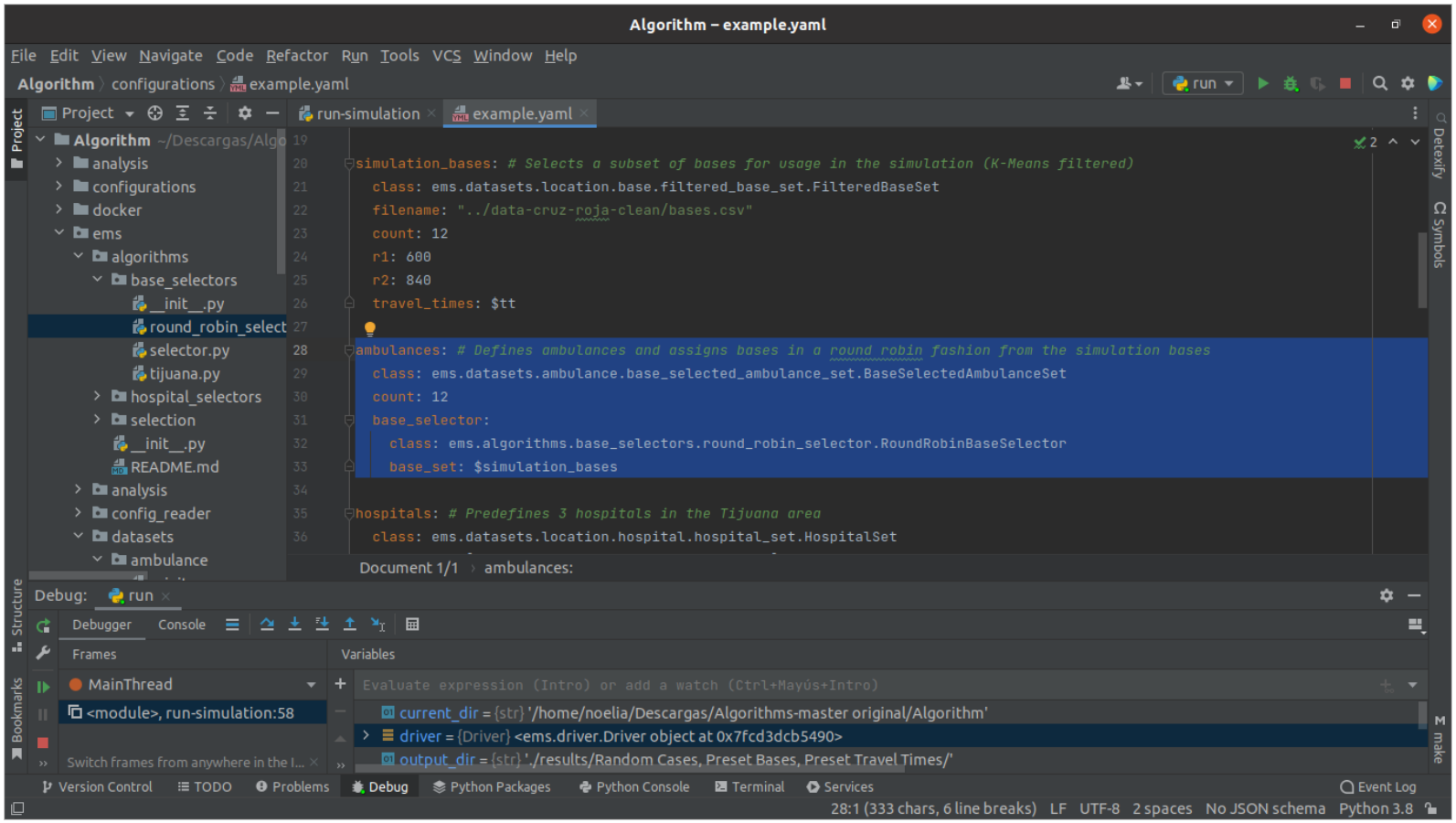
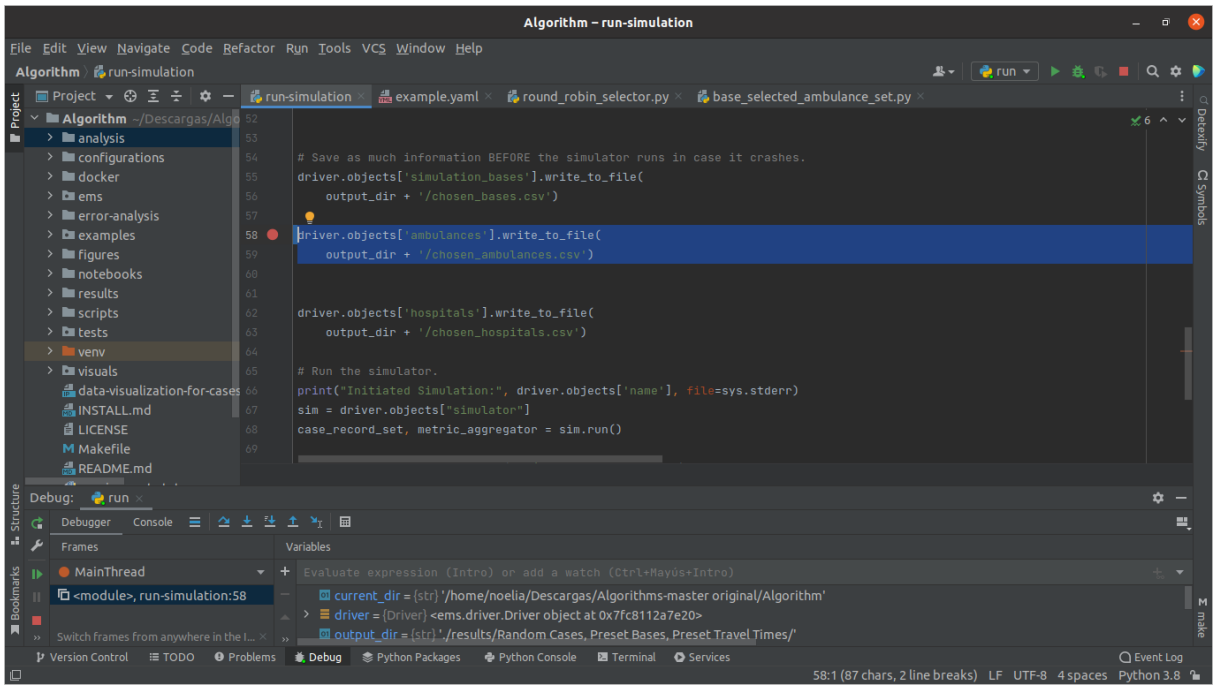


Figure 17: example.yaml source code

Simulator process and output

The video shows the files needed to run this second block, in this point, you can stop and get the location of the ambulances which are the same as the location of the bases. Finally, when the file is saved, an additional field (capacity) is added, but this field will not be used in the experiments.

Watch video



Simulator process and output

In the hospital location block, the data are prepared to be the possible destinations of the patients to be transferred. At the same time it is managed in the chosen_hospitals.csv file.

The file is saved in **results - > Random Cases, Preset Bases, Preset Travel Times - > chosen_hospitals.csv**.

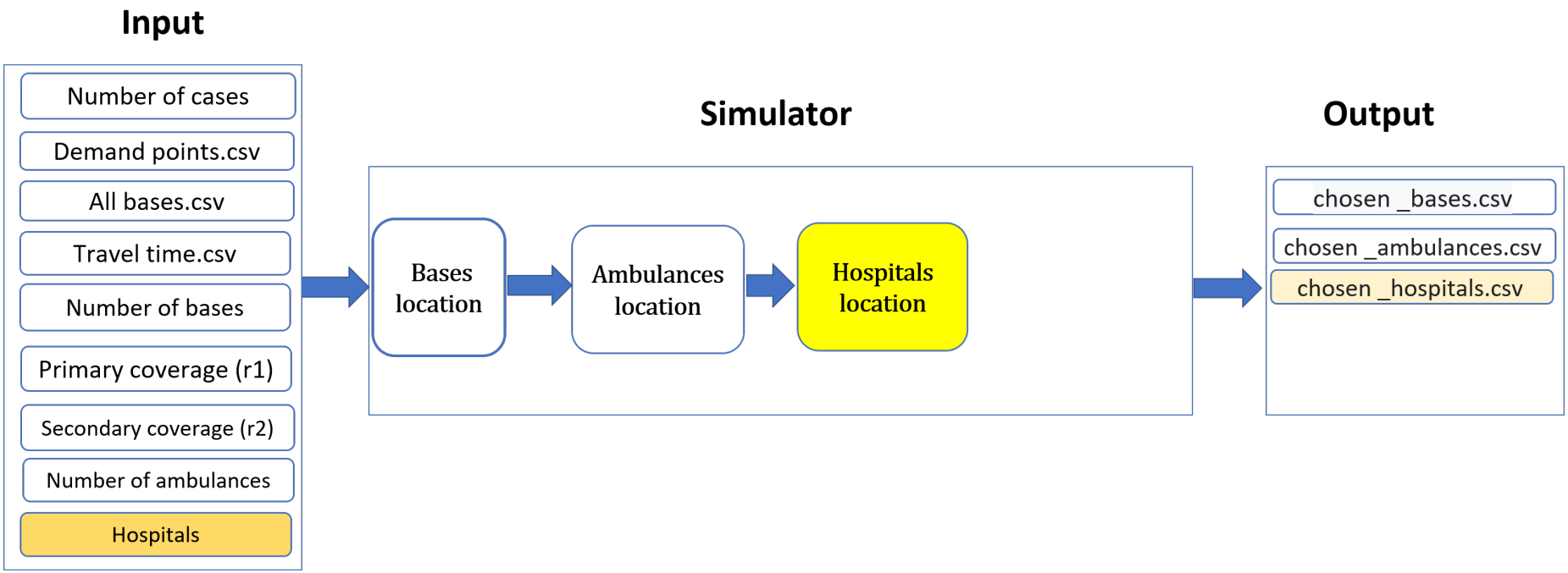


Figure 18: Simulator module.

Simulator process and output

This block indicates the coordinates of the hospitals.

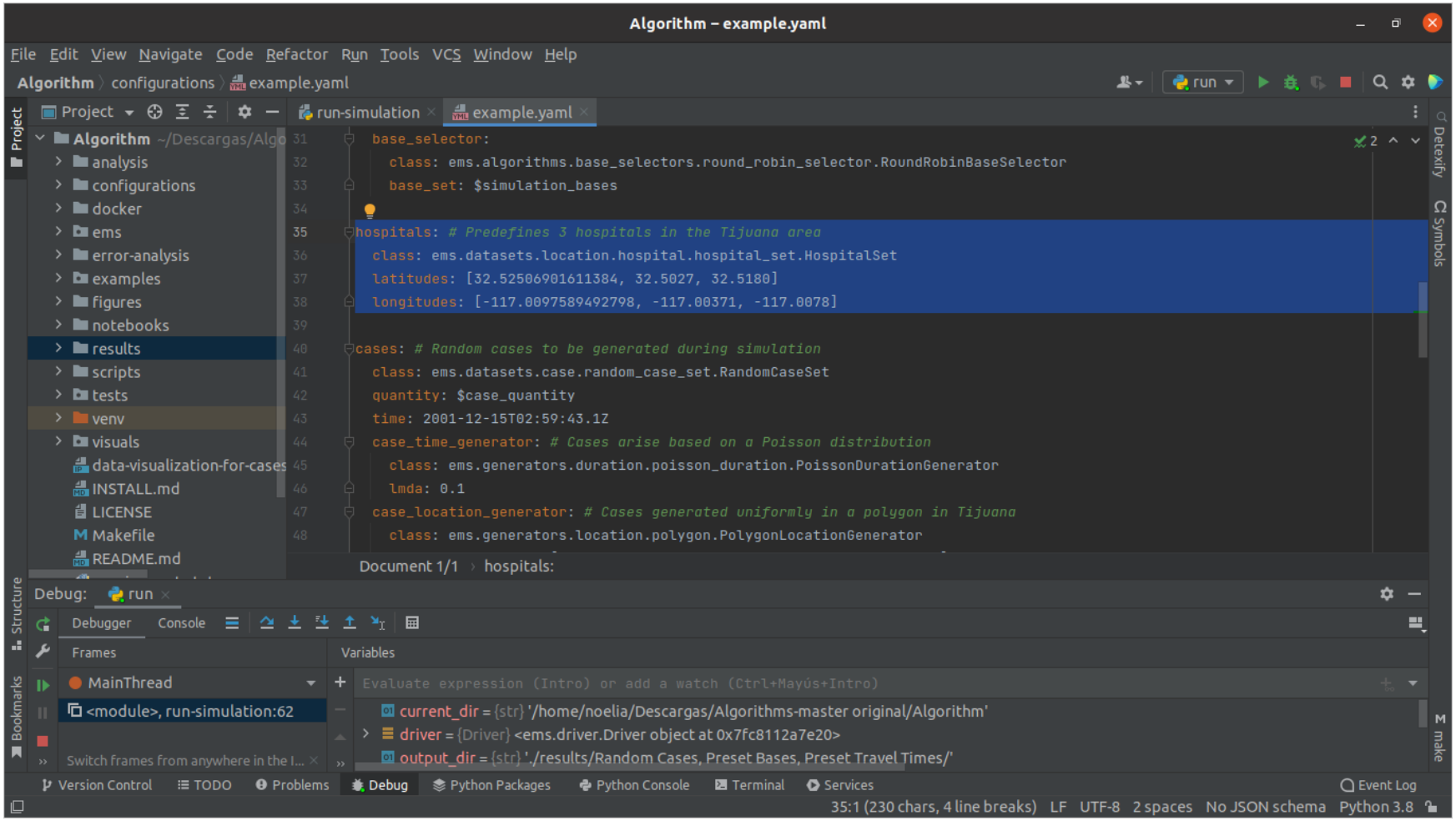
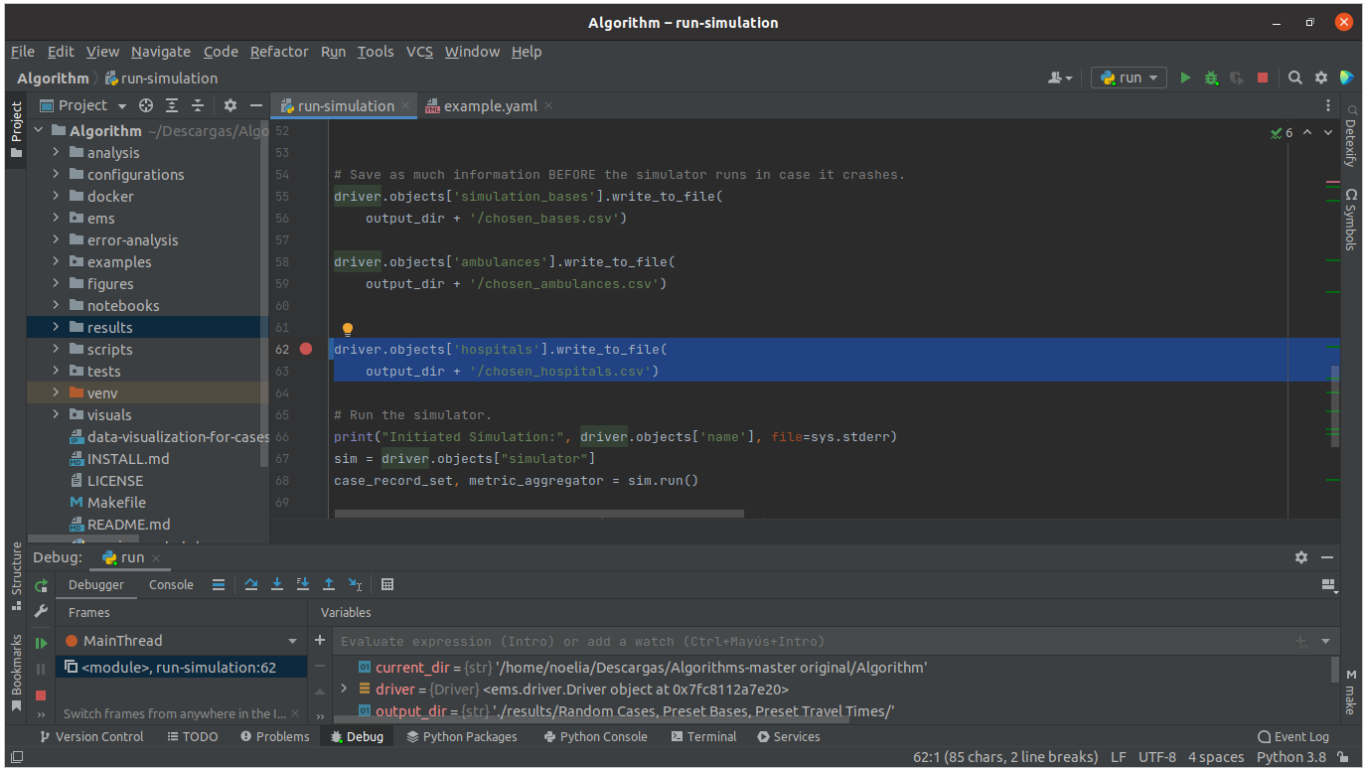


Figure 19: example.yaml source code

Simulator process and output

The video shows the point that you can stop and see how hospital data is prepared for later use.

Watch video



Simulator process and output

In this block several tasks are performed at the same time.

- Cases are generated.
- Ambulances are sent.
- Different events are generated.
- The percentage of coverage is calculated.

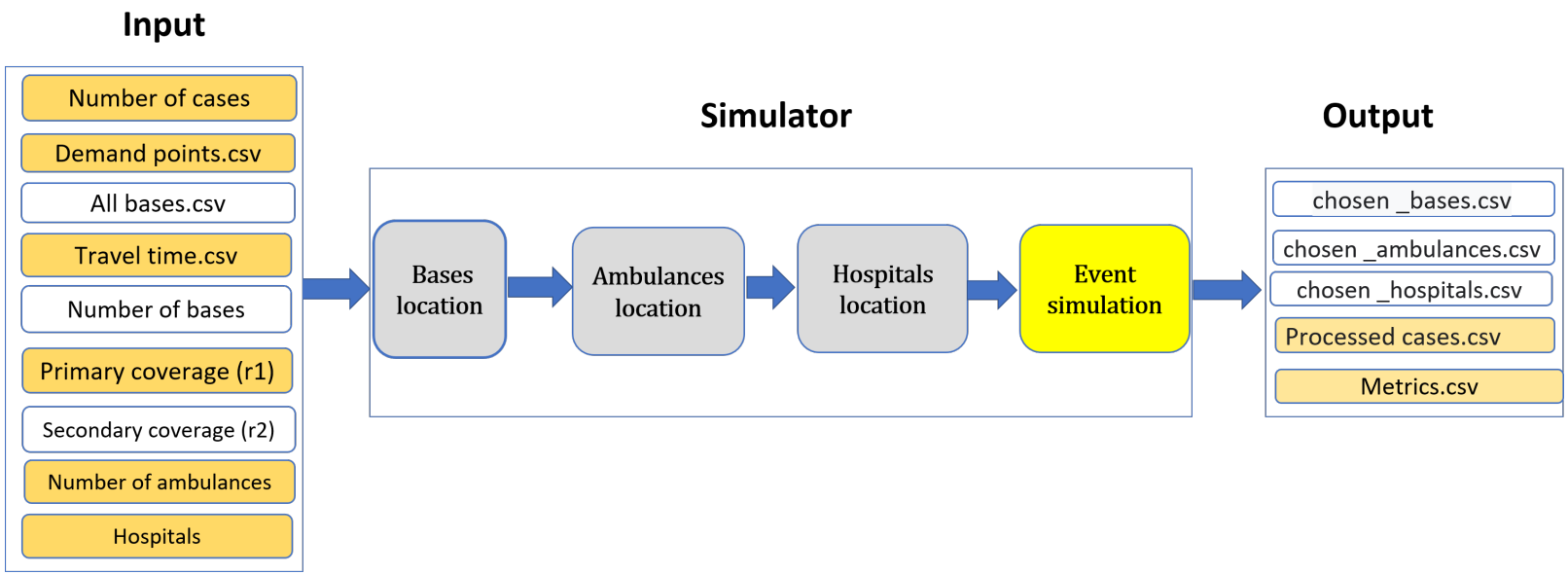


Figure 20: Simulator module.

Simulator process and output

This is the main block of the event simulation.

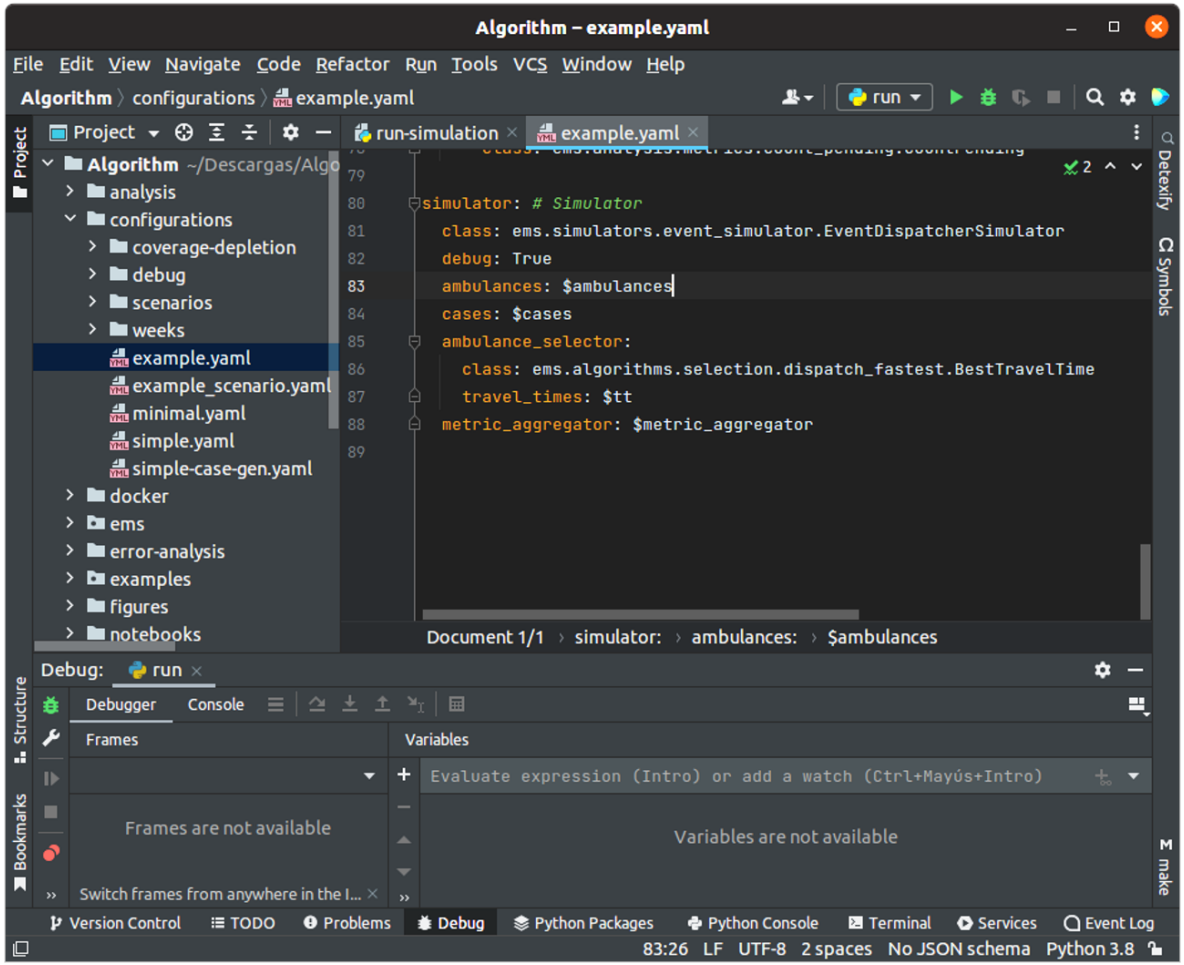


Figure 21: example.yaml source code

Simulator process and output

The class of this block indicates the path to the file that performs all the processing.

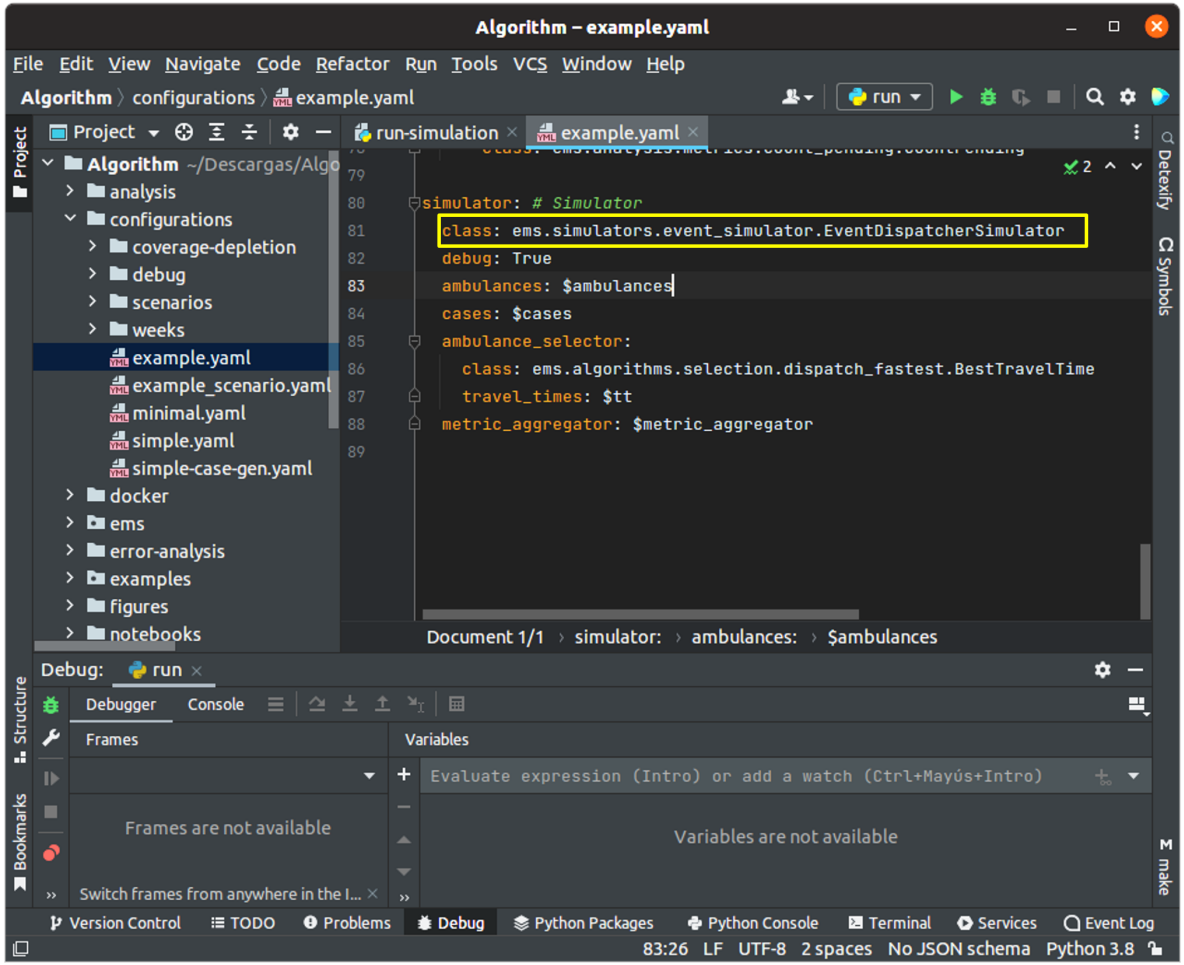


Figure 21: example.yaml source code

Simulator process and output

Once inside the file, the first thing to do is to generate a case.

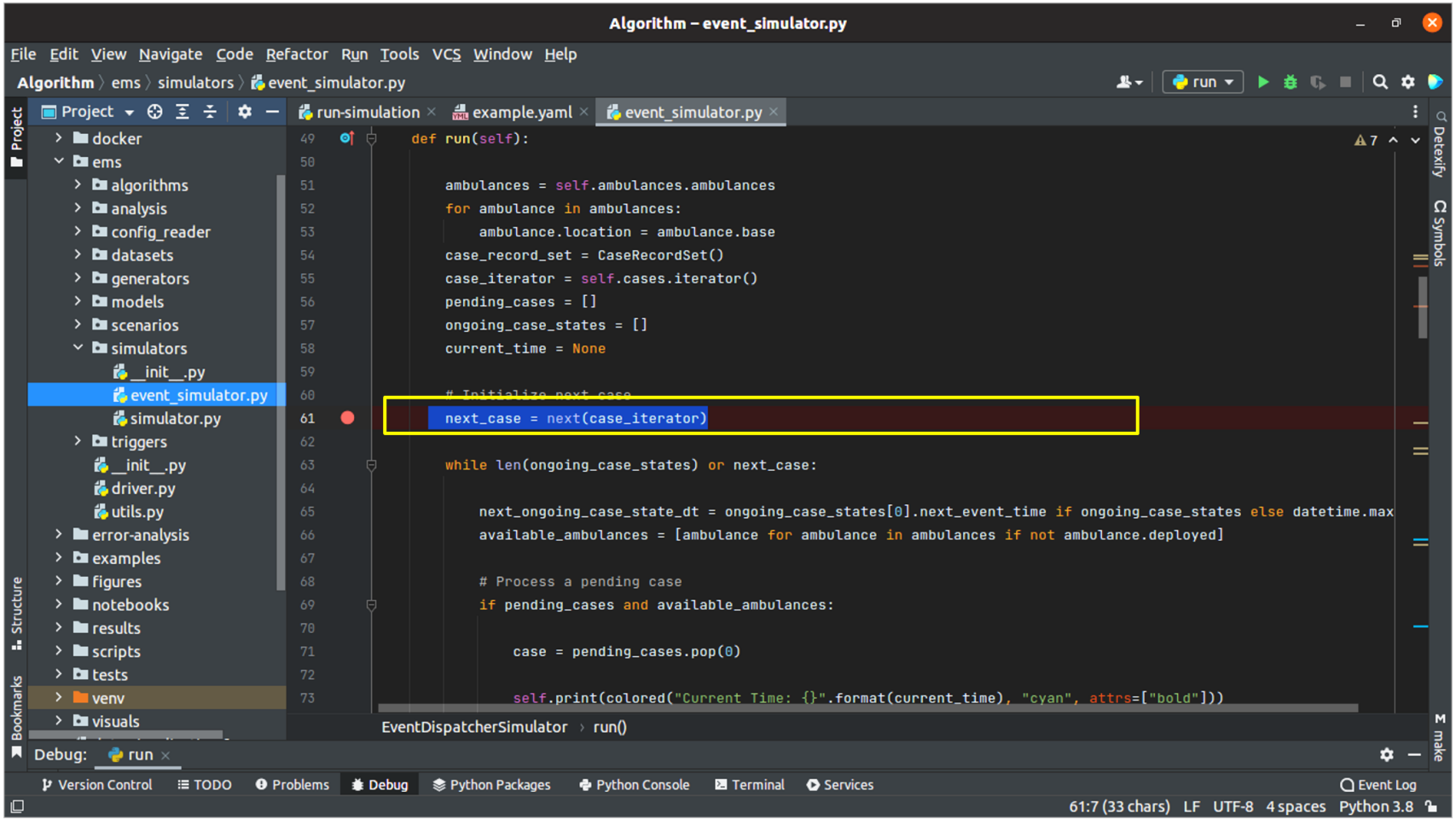


Figure 22: event_simulator.py source code

Simulator process and output

This is the block in charge of generating this new case, the class indicates the file path where this process is carried out.

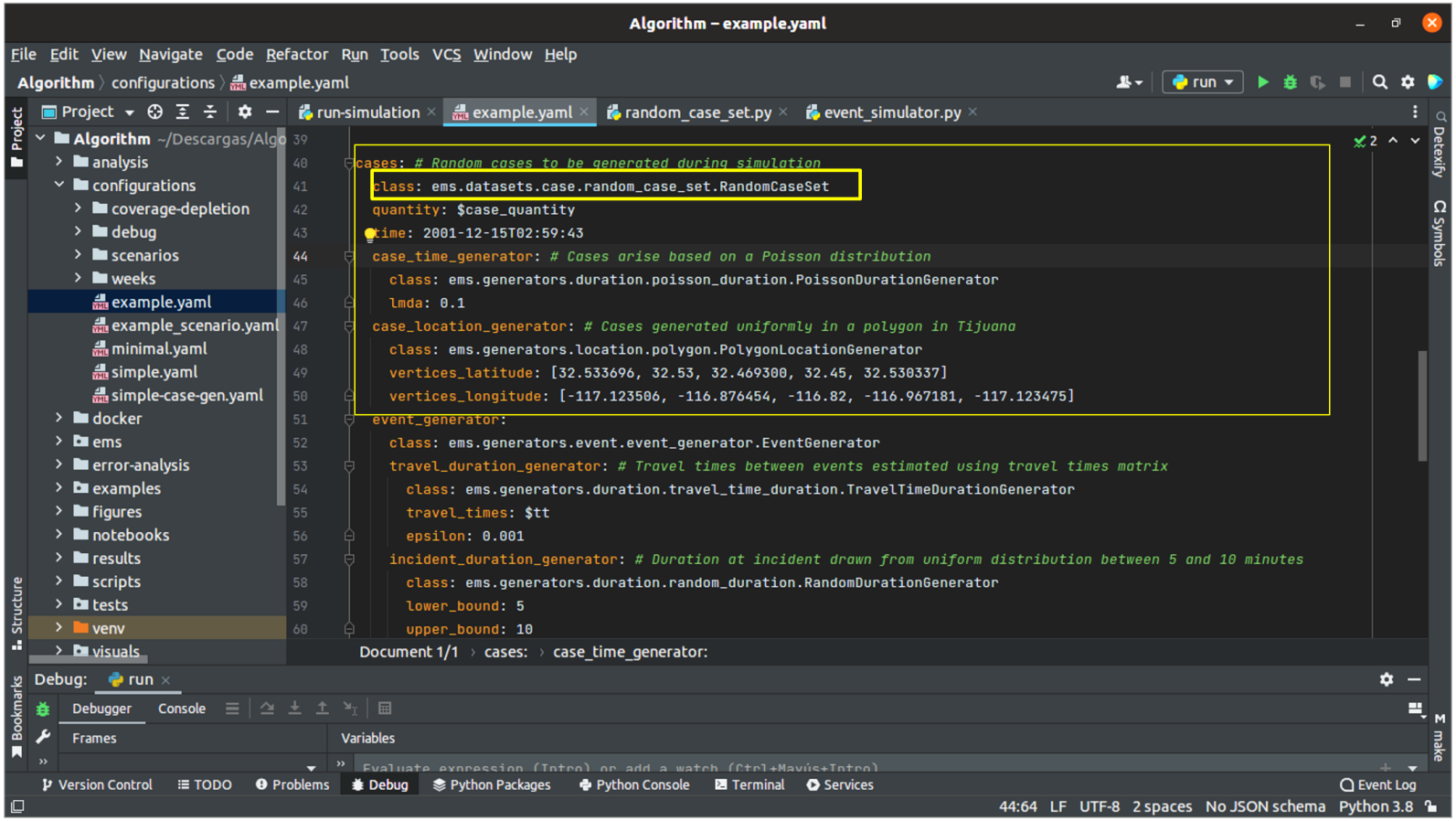


Figure 23: example.yaml source code

Simulator process and output

Within this file, a new case is generated with:

- A random time for the call.
- A random location of the call within a previously set area.

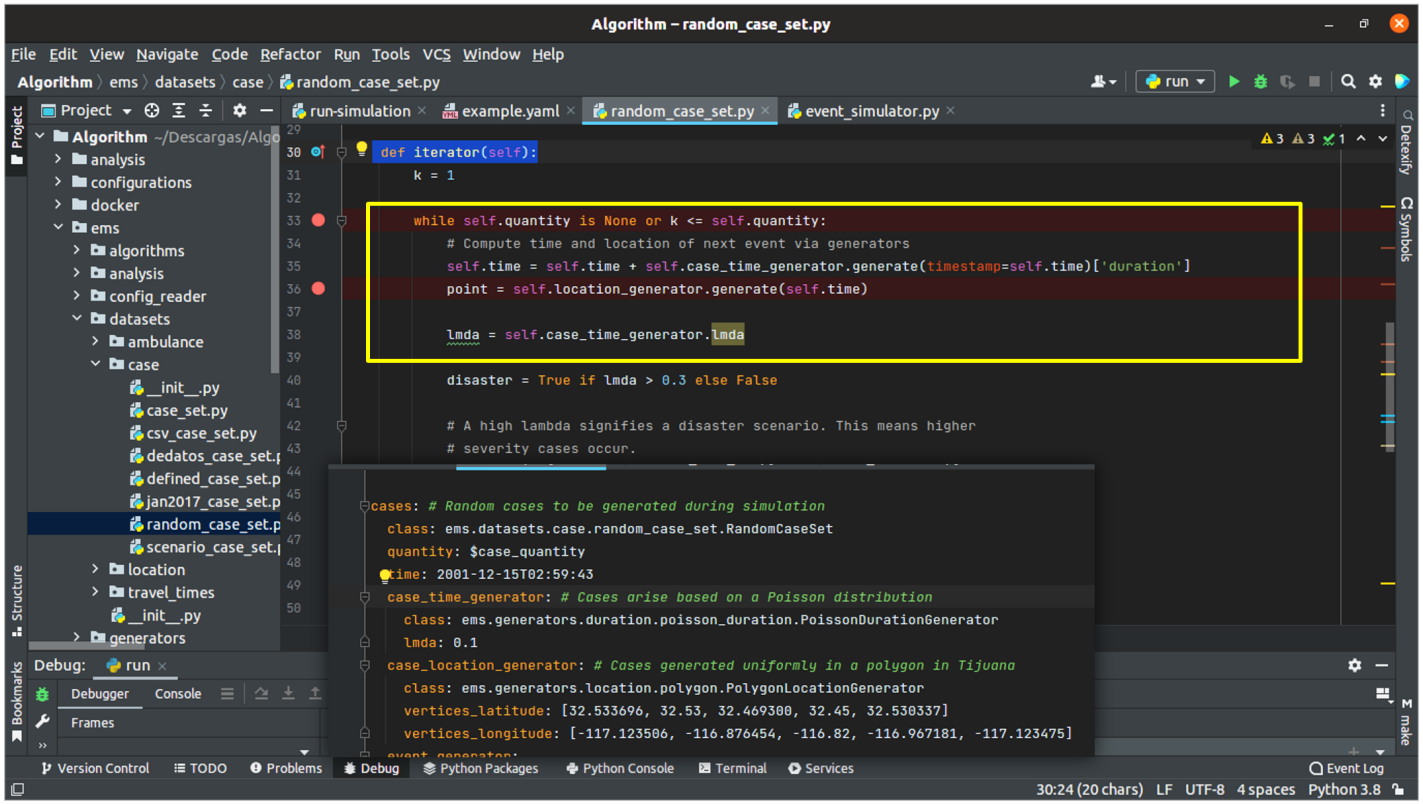


Figure 24: random_case_set.py source code.

Simulator process and output

The events that the case may have are generated.

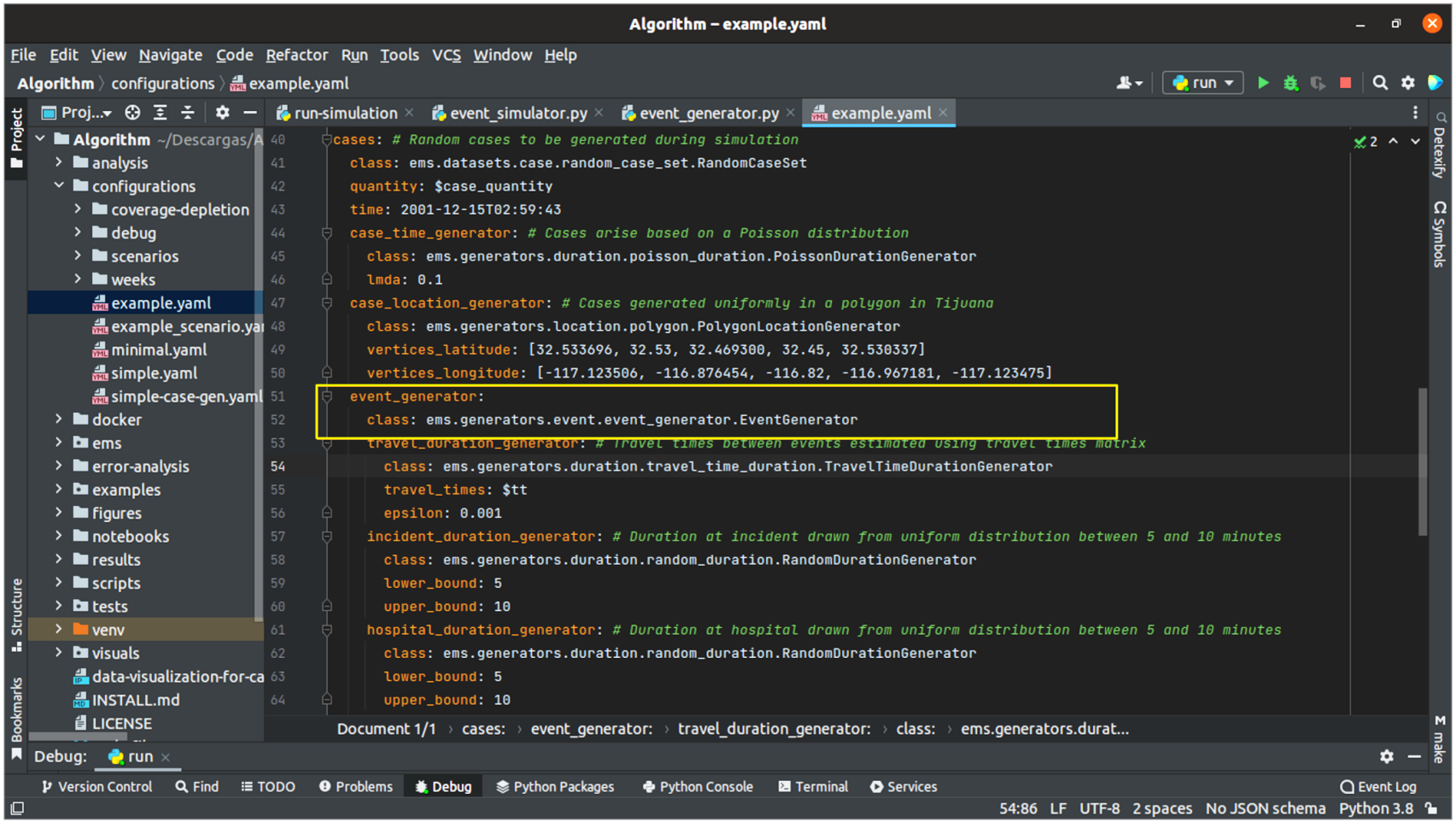


Figure 25: example.yaml source code.

Simulator process and output

Events are defined in this file.

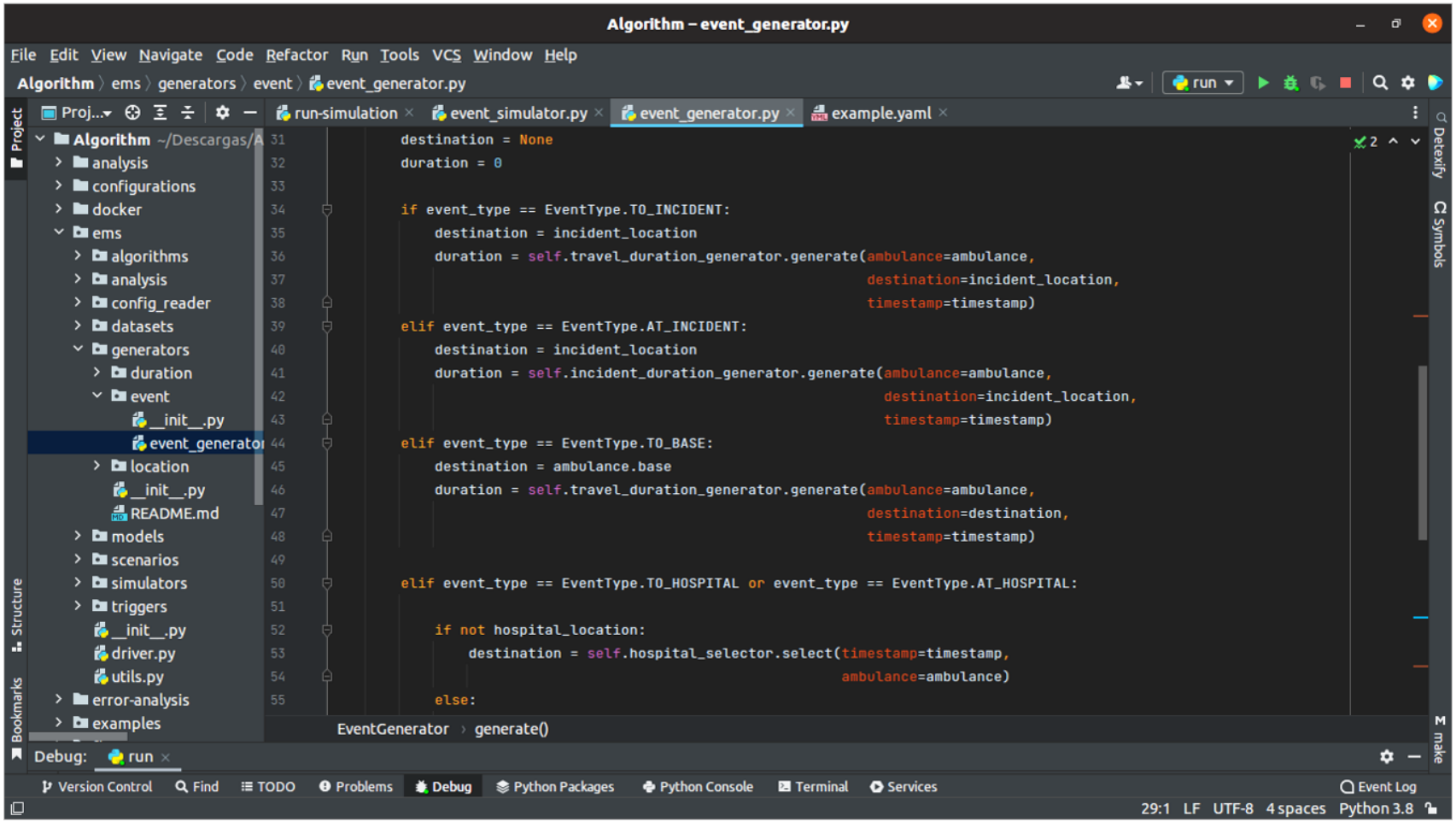


Figure 26: event_generator.py source code.

Simulator process and output

A case can have different events and depending on this, the status is established.

Cases	Event	Status
1	Arrive to incident	Ongoing
2	Attending to incident	Ongoing
3	Heading to hospital	Ongoing
4	Leave patient	Ongoing
5	Returning to base	Done

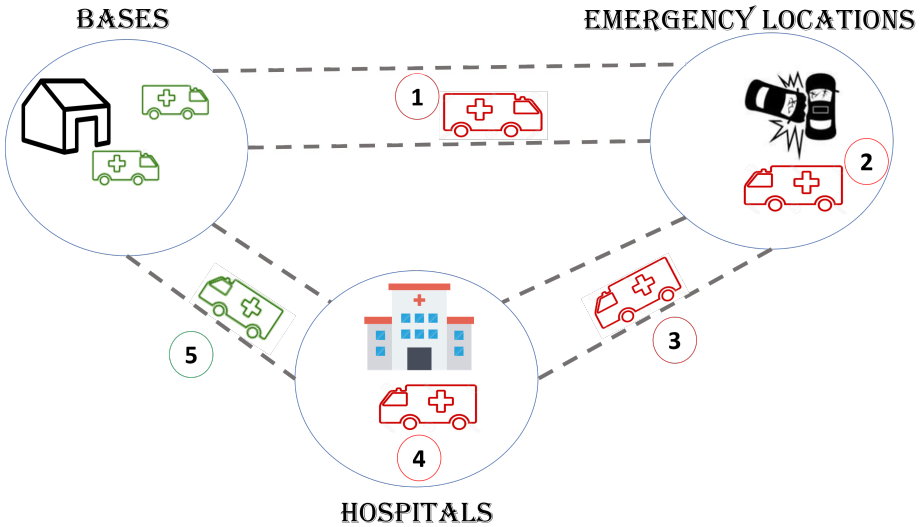


Figure 27: Case events.

Simulator process and output

Continuing with the main file, the cases start running as long as the first condition is met:

if there are cases in ongoing or a new case has been generated.

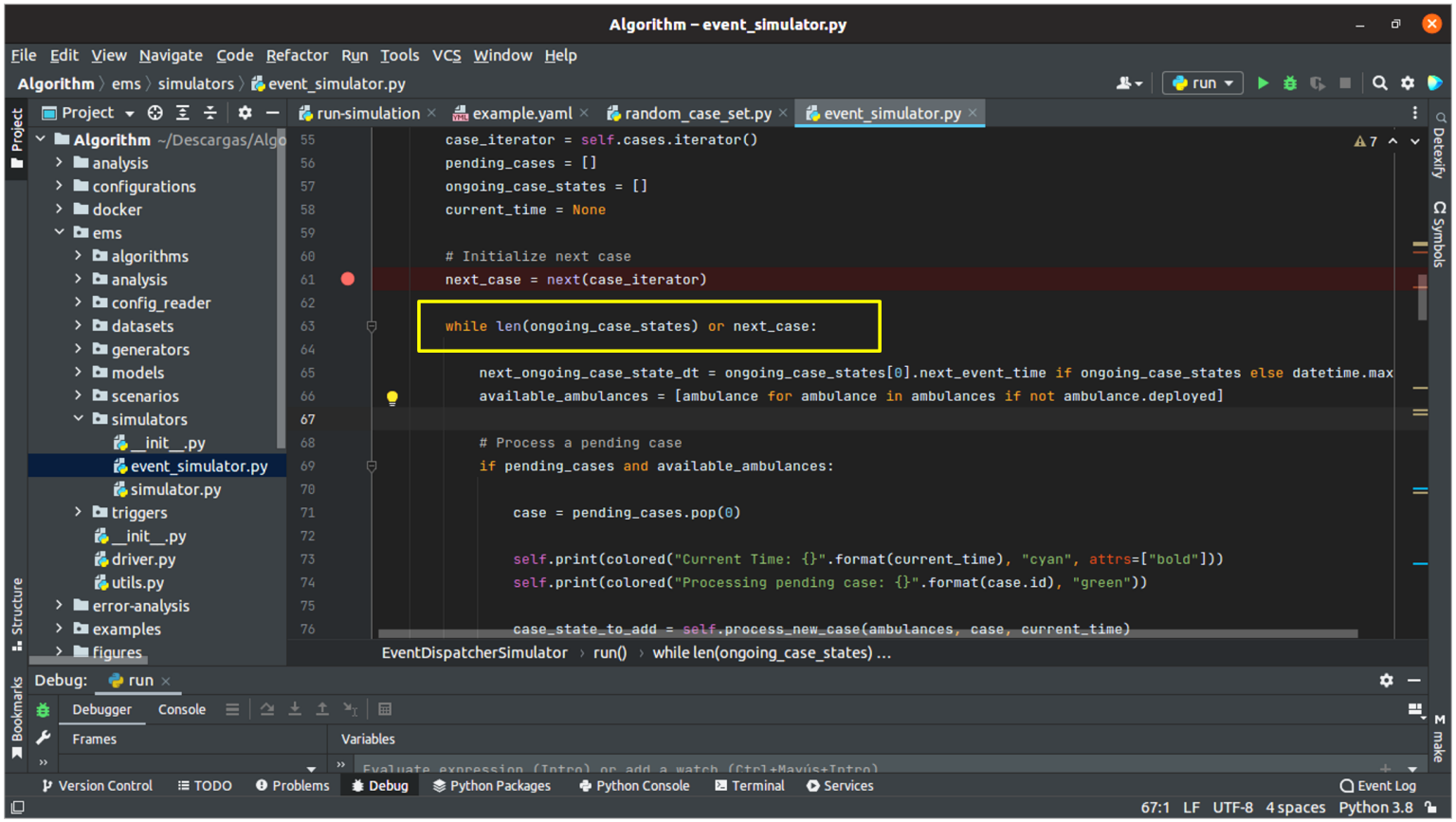


Figure 28: event_simulator.py source code..

Simulator process and output

The available ambulances are verified.

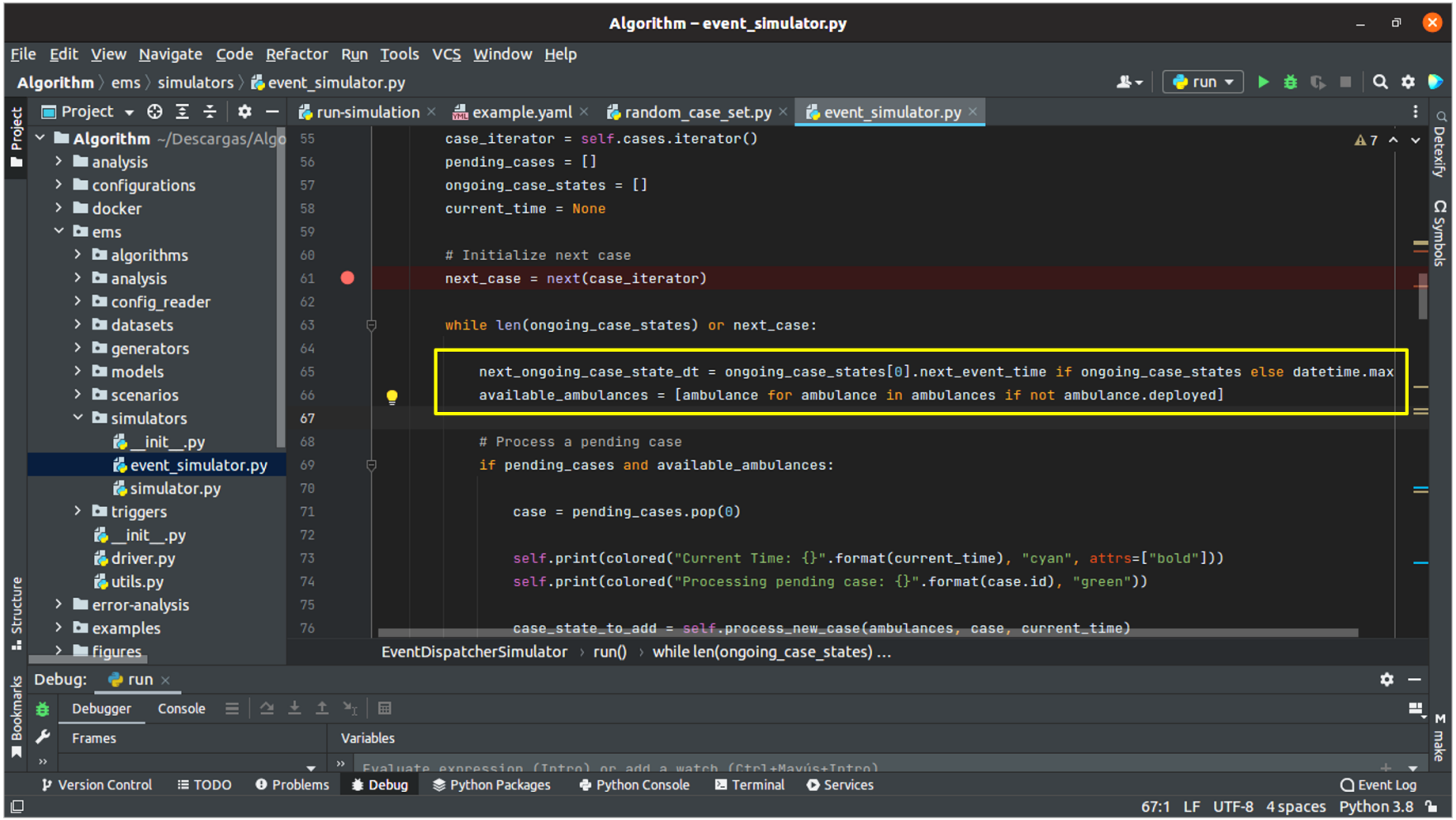


Figure 28: event_simulator.py source code..

Simulator process and output

Moving on to the next condition:

If there are pending cases to attend and ambulances available.

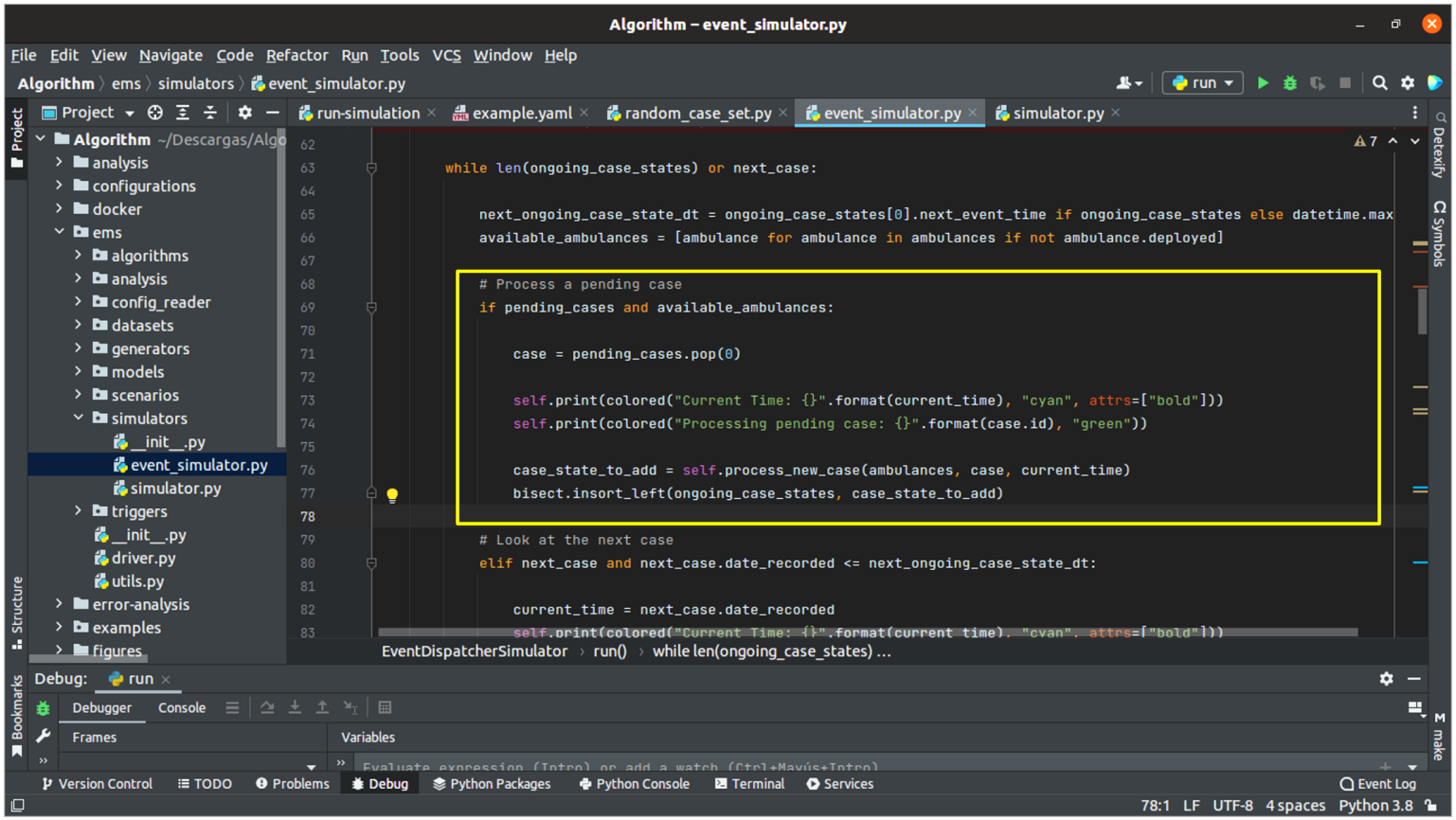


Figure 28: event_simulator.py source code..

Simulator process and output

Moving on to the next:

Start a new case.

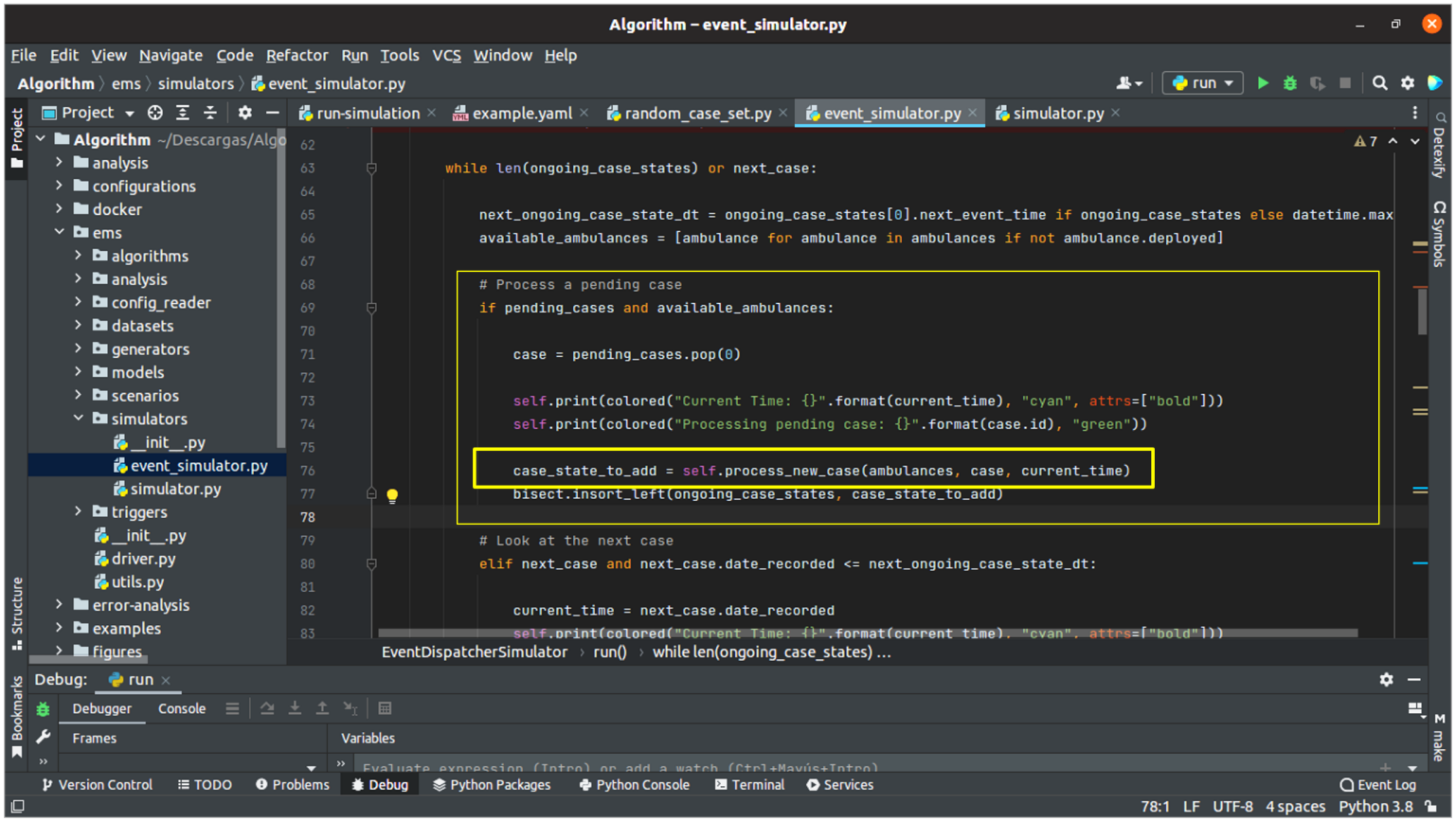


Figure 28: event_simulator.py source code..

Simulator process and output

When a new case is started, this section of the code is executed.

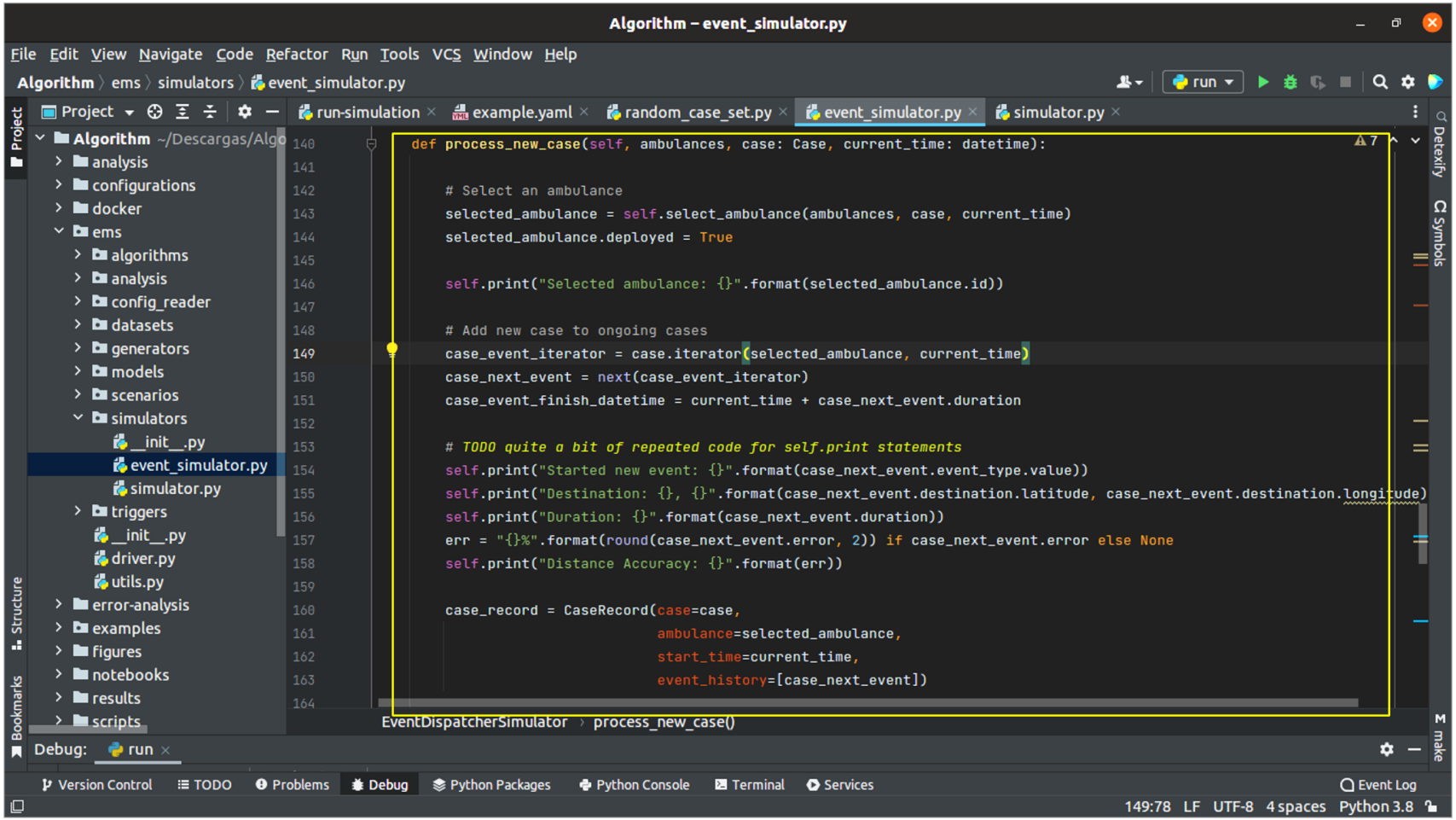


Figure 29: event_simulator.py source code.

Simulator process and output

Starting with the selection of the ambulance that will attend the new case.

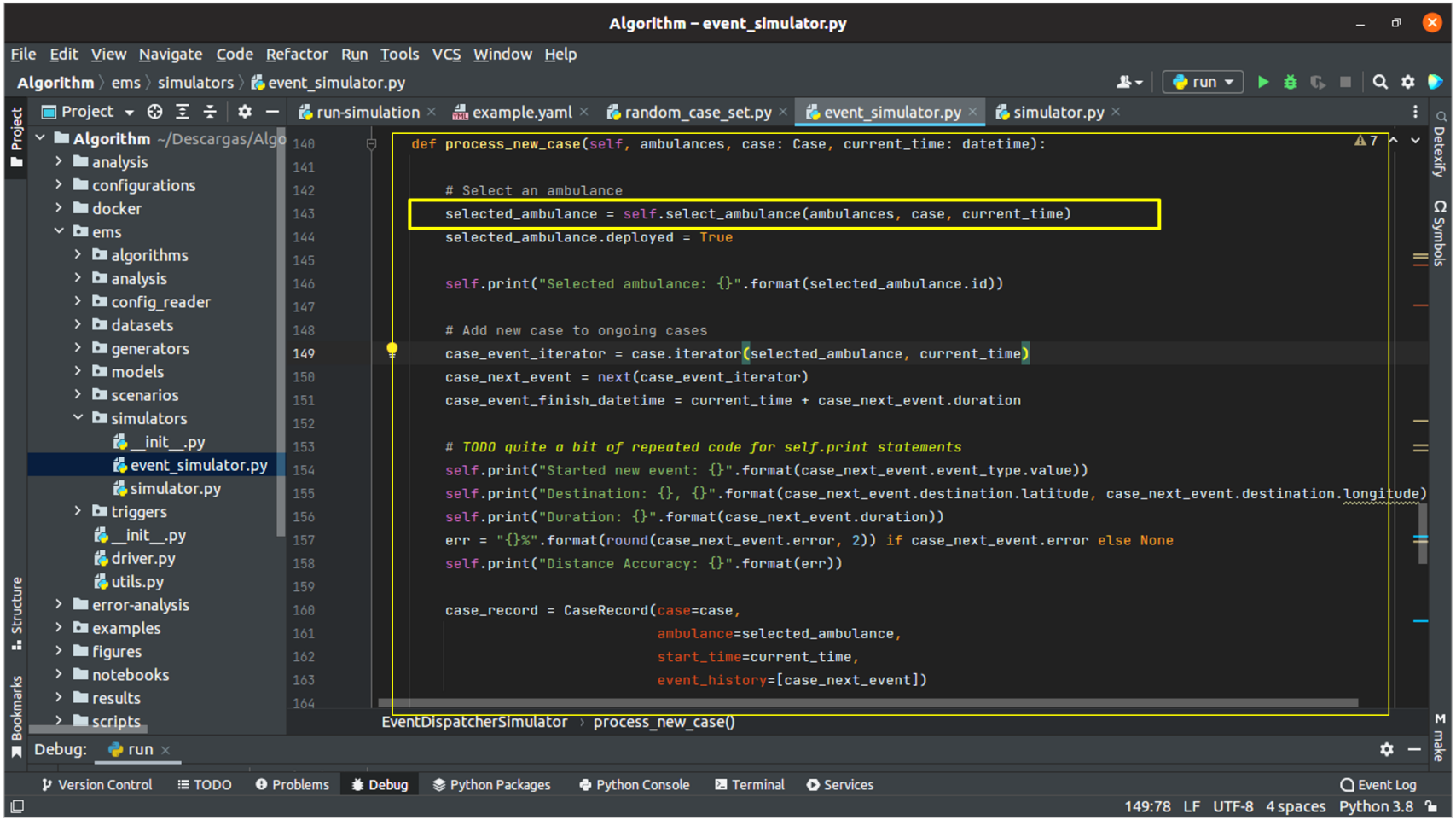


Figure 29: event_simulator.py source code.

Simulator process and output

The selec_ambulance function is invoked, which is responsible for initiating this process.

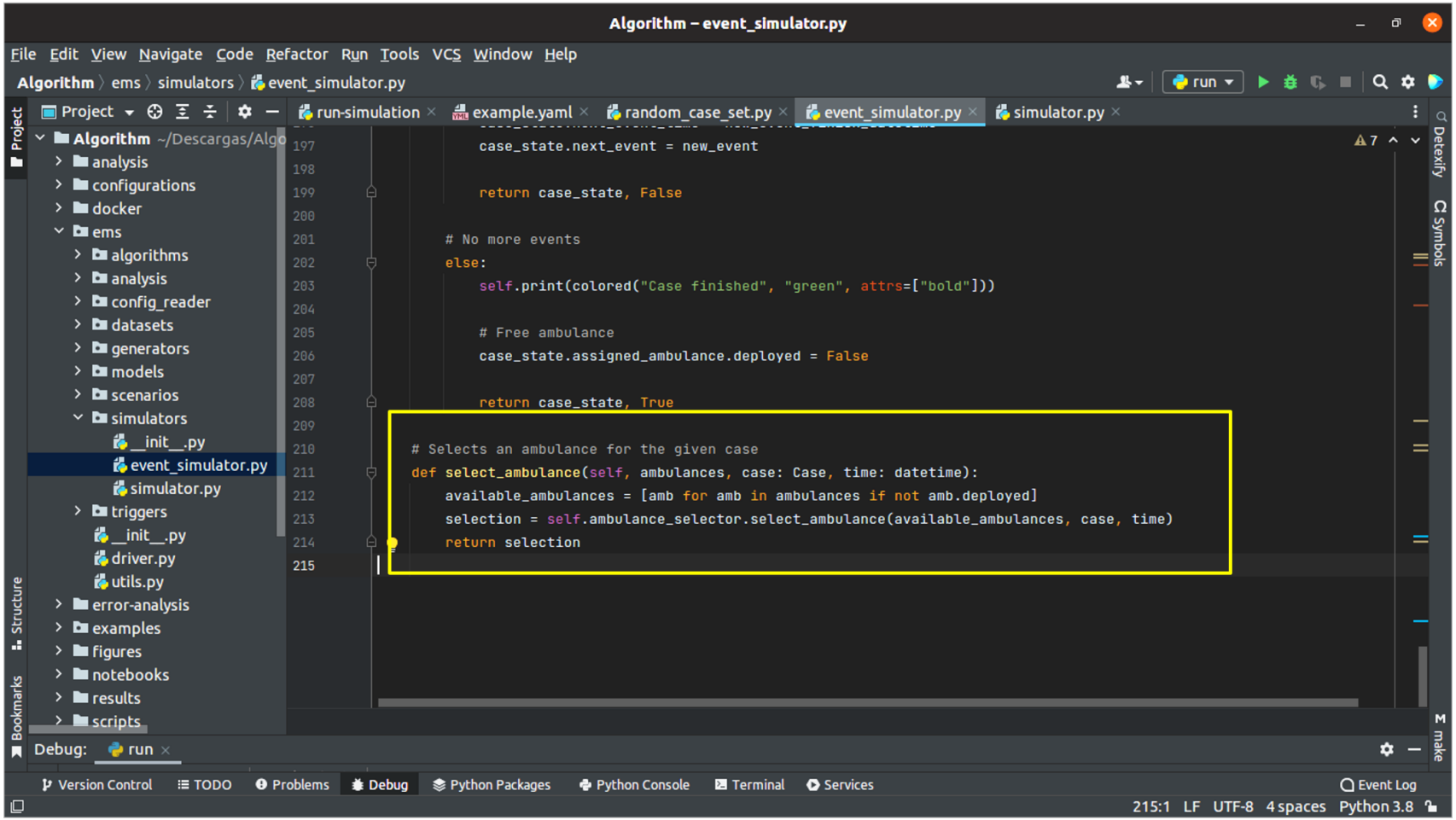


Figure 29: event_simulator.py source code.

Simulator process and output

In turn is called the block that is responsible for carry out the process.

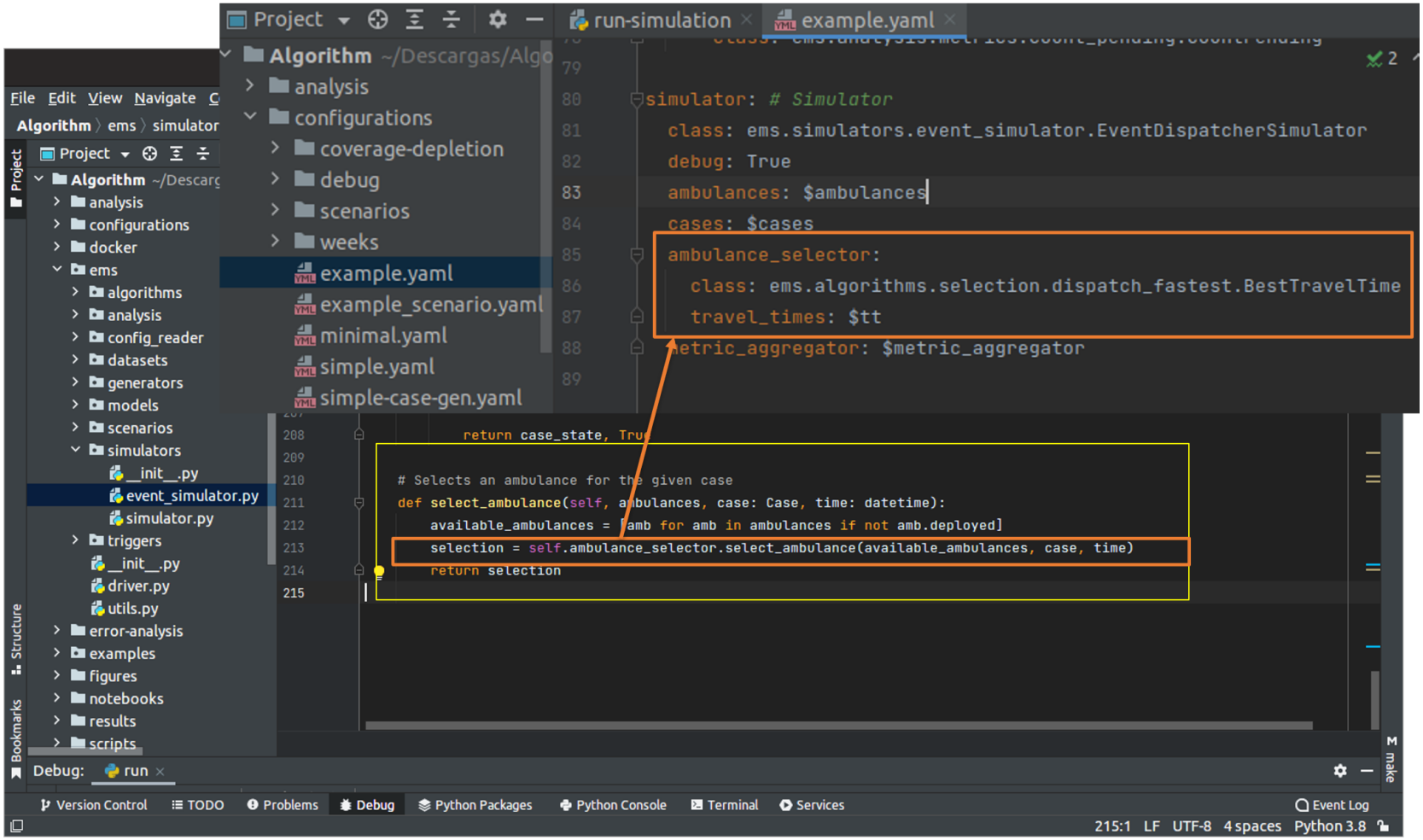


Figure 29: event_simulator.py source code.

Simulator process and output

Once inside the file.

1. The new case is located in one of the demand points.

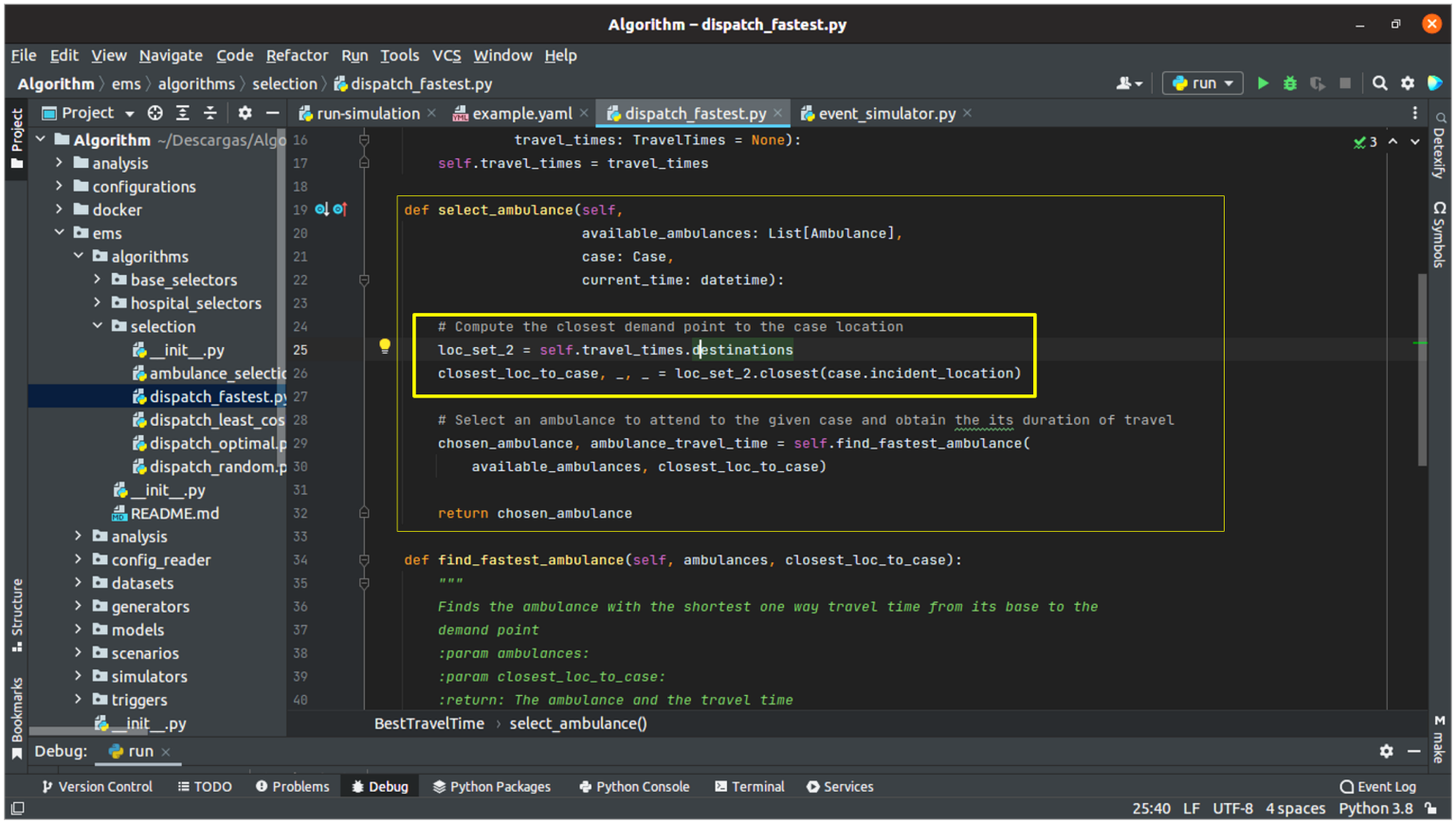


Figure 30: dispatch_fastest.py source code.

Simulator process and output

2. Check the distance in the travel time matrix of each ambulance to the demand point, until you find the ambulance that will arrive in the shortest possible time.

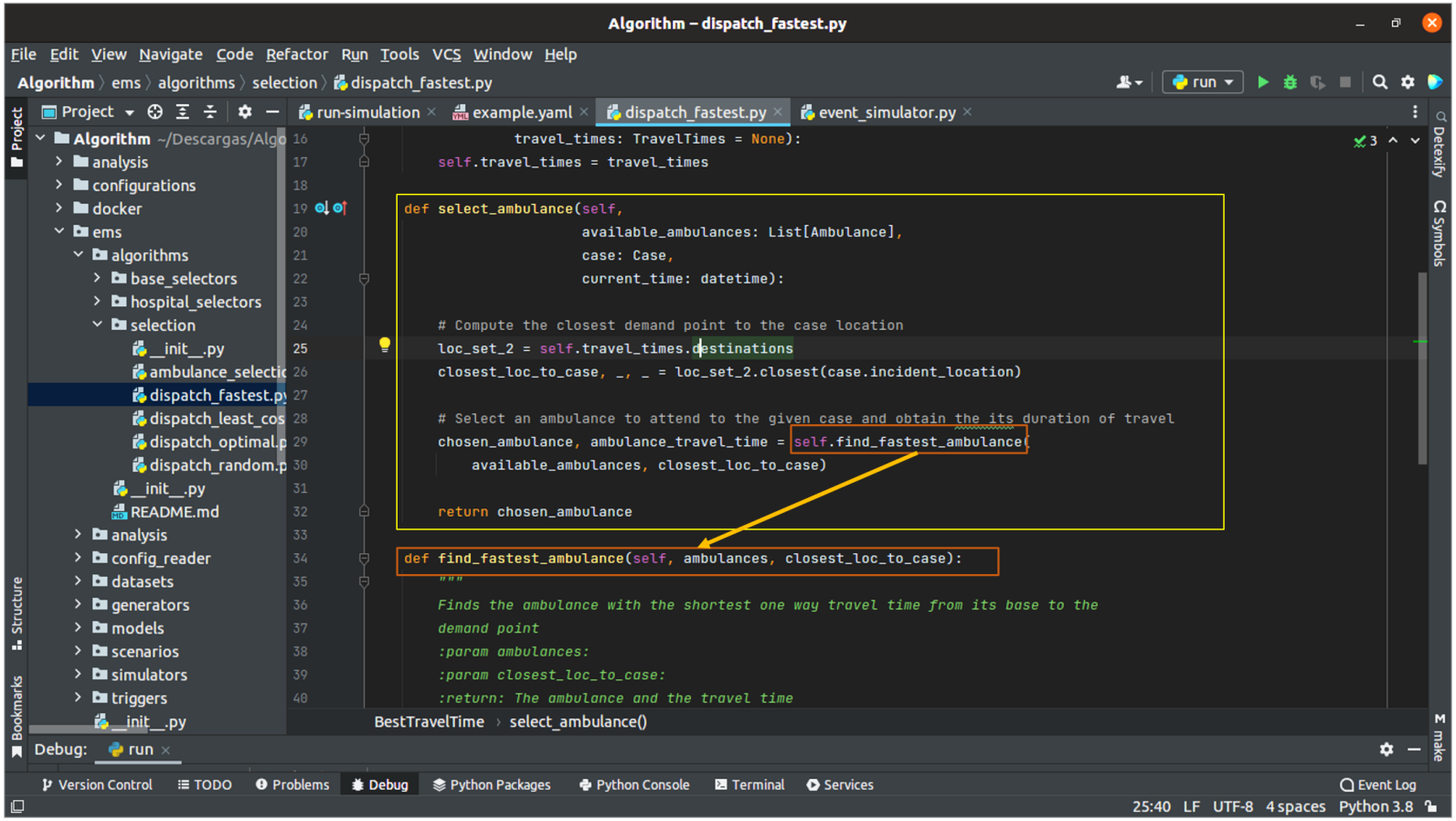


Figure 30: dispatch_fastest.py source code.

Simulator process and output

2. Check the distance in the travel time matrix of each ambulance to the demand point, until you find the ambulance that will arrive in the shortest possible time.

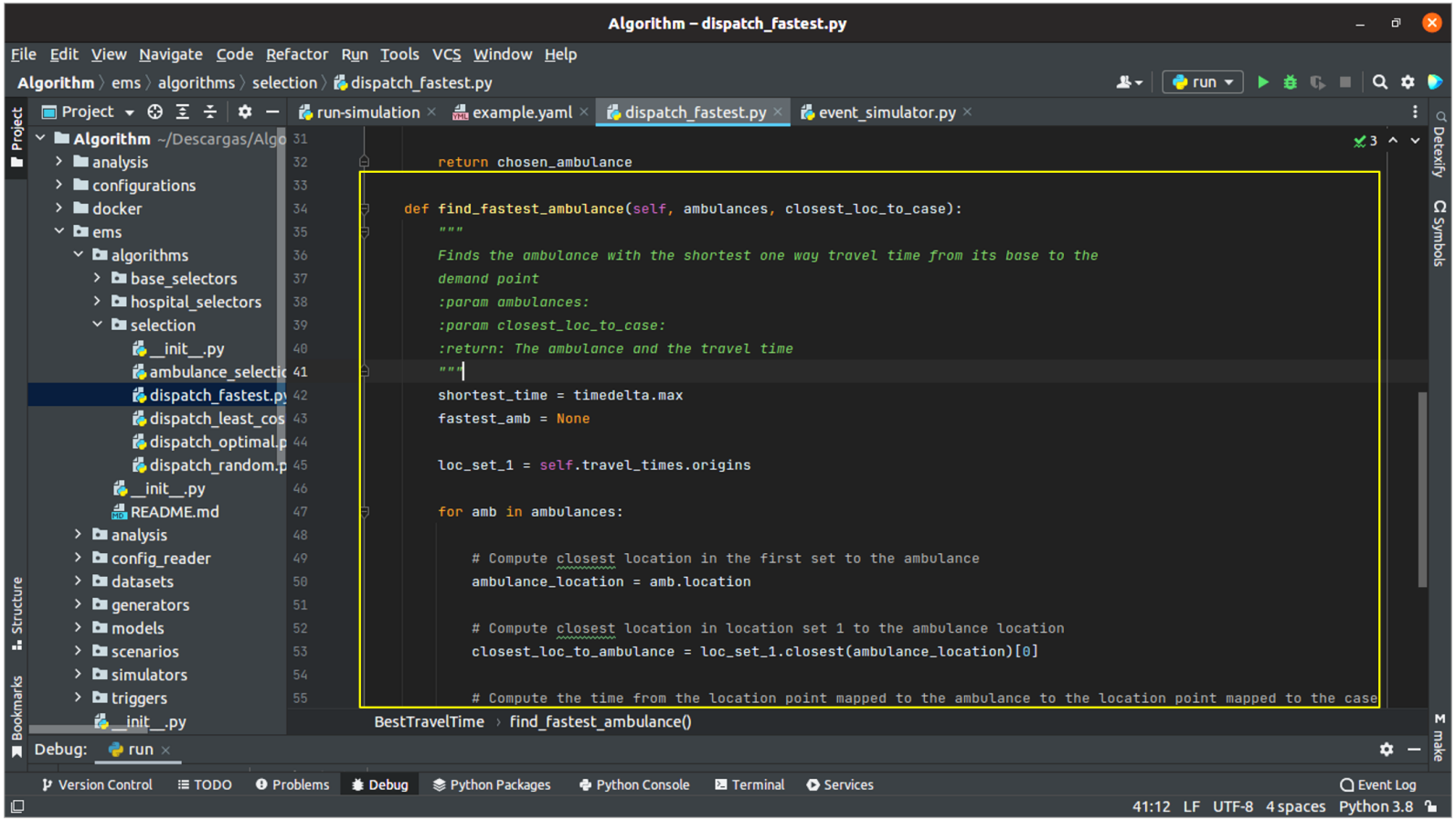


Figure 30: dispatch_fastest.py source code.

Simulator process and output

Continuing with the new case, once the ambulance is selected, it will be set as occupied.

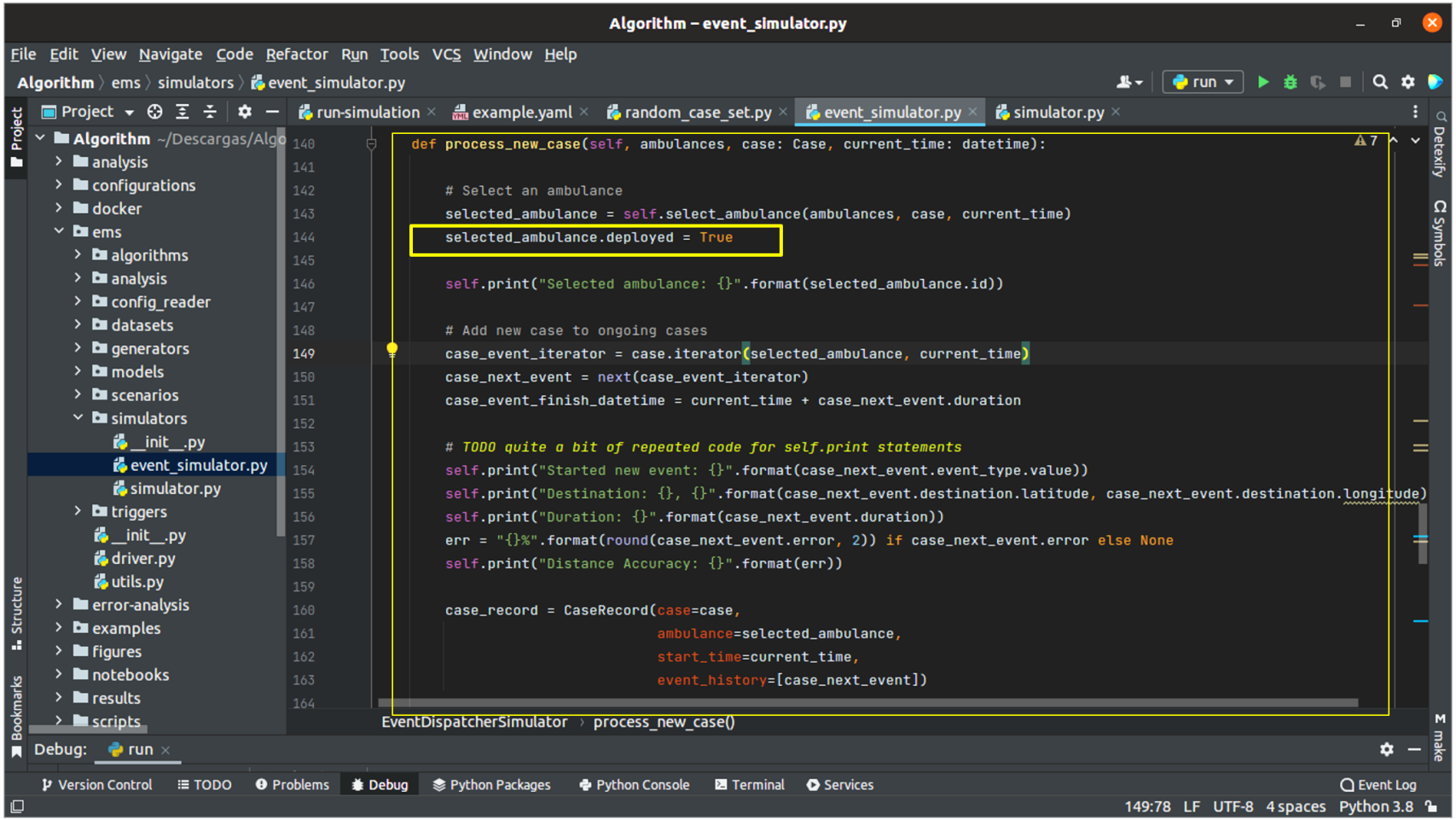


Figure 31: event_simulator.py source code

Simulator process and output

The first event generated is: Arrive to incident

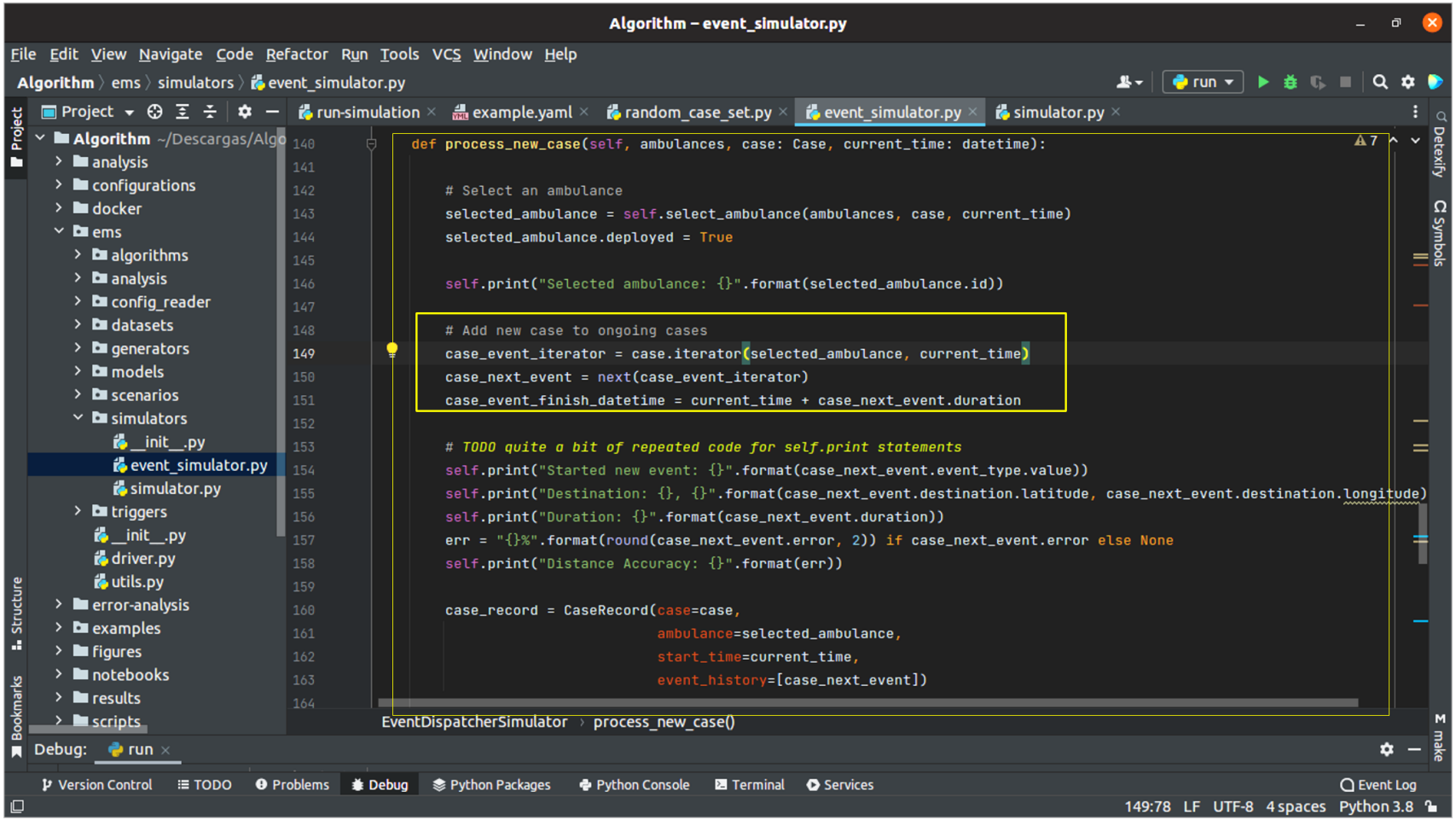


Figure 31: event_simulator.py source code

Simulator process and output

The travel time from the base to the demand point is obtained from the previously saved matrix.

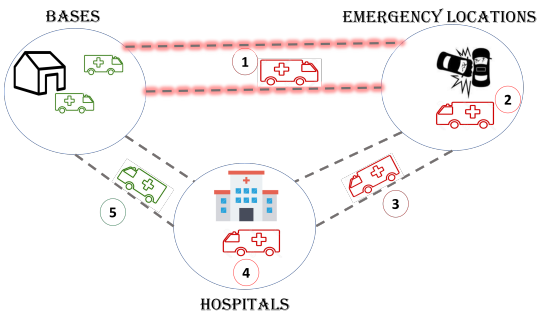


Figure 32: Case events

```
51  event_generator:
52    class: ems.generators.event.event_generator.EventGenerator
53    travel_duration_generator: # Travel times between events estimated using travel ti
54      class: ems.generators.duration.travel_time_duration.TravelTimeDurationGenerator
55      travel_times: $tt
56      epsilon: 0.001
57    incident_duration_generator: # Duration at incident drawn from uniform distributio
58      class: ems.generators.duration.random_duration.RandomDurationGenerator
59      lower_bound: 5
60      upper_bound: 10
61    hospital_duration_generator: # Duration at hospital drawn from uniform distributio
62      class: ems.generators.duration.random_duration.RandomDurationGenerator
63      lower_bound: 5
64      upper_bound: 10
65    hospital_selector: # Fastest hospital chosen
66      class: ems.algorithms.hospital_selectors.select_fastest.FastestHospitalSelector
67      hospital_set: $hospitals
68      travel_times: $tt
69
```

Figure 33: example.yaml source code

Simulator process and output

The duration at the place of the new case is randomly generated between 5 and 10 minutes.

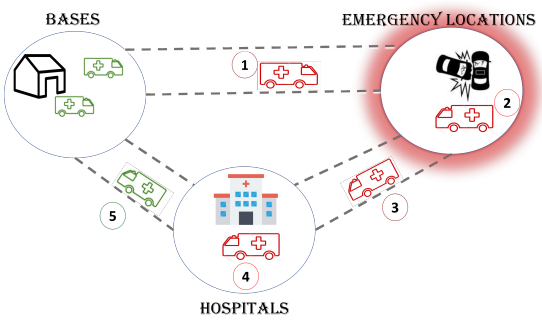


Figure 34: Case events

```
51  event_generator:
52    class: ems.generators.event.event_generator.EventGenerator
53    travel_duration_generator: # Travel times between events estimated using travel ti
54      class: ems.generators.duration.travel_time_duration.TravelTimeDurationGenerator
55      travel_times: $tt
56      epsilon: 0.001
57    incident_duration_generator: # Duration at incident drawn from uniform distributio
58      class: ems.generators.duration.random_duration.RandomDurationGenerator
59      lower_bound: 5
60      upper_bound: 10
61    hospital_duration_generator: # Duration at hospital drawn from uniform distributio
62      class: ems.generators.duration.random_duration.RandomDurationGenerator
63      lower_bound: 5
64      upper_bound: 10
65    hospital_selector: # Fastest hospital chosen
66      class: ems.algorithms.hospital_selectors.select_fastest.FastestHospitalSelector
67      hospital_set: $hospitals
68      travel_times: $tt
69
```

Figure 35: example.yaml source code

Simulator process and output

The duration in the hospital is randomly generated between 5 and 10 minutes.

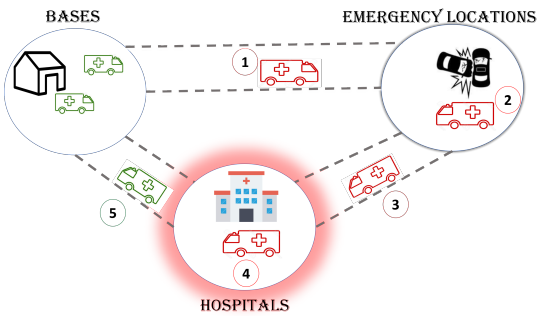


Figure 36: Case events

```
51  event_generator:
52    class: ems.generators.event.event_generator.EventGenerator
53    travel_duration_generator: # Travel times between events estimated using travel ti
54      class: ems.generators.duration.travel_time_duration.TravelTimeDurationGenerator
55      travel_times: $tt
56      epsilon: 0.001
57    incident_duration_generator: # Duration at incident drawn from uniform distributio
58      class: ems.generators.duration.random_duration.RandomDurationGenerator
59      lower_bound: 5
60      upper_bound: 10
61    hospital_duration_generator: # Duration at hospital drawn from uniform distributio
62      class: ems.generators.duration.random_duration.RandomDurationGenerator
63      lower_bound: 5
64      upper_bound: 10
65    hospital_selector: # Fastest hospital chosen
66      class: ems.algorithms.hospital_selectors.select_fastest.FastestHospitalSelector
67      hospital_set: $hospitals
68      travel_times: $tt
69
```

Figure 37: example.yaml source code

Simulator process and output

The selection of the hospital is according to the one located closest to the point of demand.

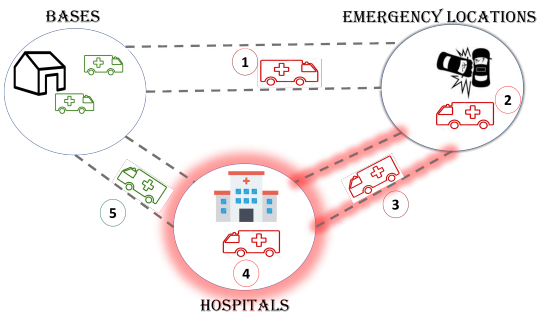


Figure 38: Case events

```
51  event_generator:
52    class: ems.generators.event.event_generator.EventGenerator
53    travel_duration_generator: # Travel times between events estimated using travel ti
54      class: ems.generators.duration.travel_time_duration.TravelTimeDurationGenerator
55      travel_times: $tt
56      epsilon: 0.001
57    incident_duration_generator: # Duration at incident drawn from uniform distributio
58      class: ems.generators.duration.random_duration.RandomDurationGenerator
59      lower_bound: 5
60      upper_bound: 10
61    hospital_duration_generator: # Duration at hospital drawn from uniform distributio
62      class: ems.generators.duration.random_duration.RandomDurationGenerator
63      lower_bound: 5
64      upper_bound: 10
65    hospital_selector: # Fastest hospital chosen
66      class: ems.algorithms.hospital_selectors.select_fastest.FastestHospitalSelector
67      hospital_set: $hospitals
68      travel_times: $tt
69
```

Figure 39: example.yaml source code

Simulator process and output

Once the first event has occurred, the information is saved.

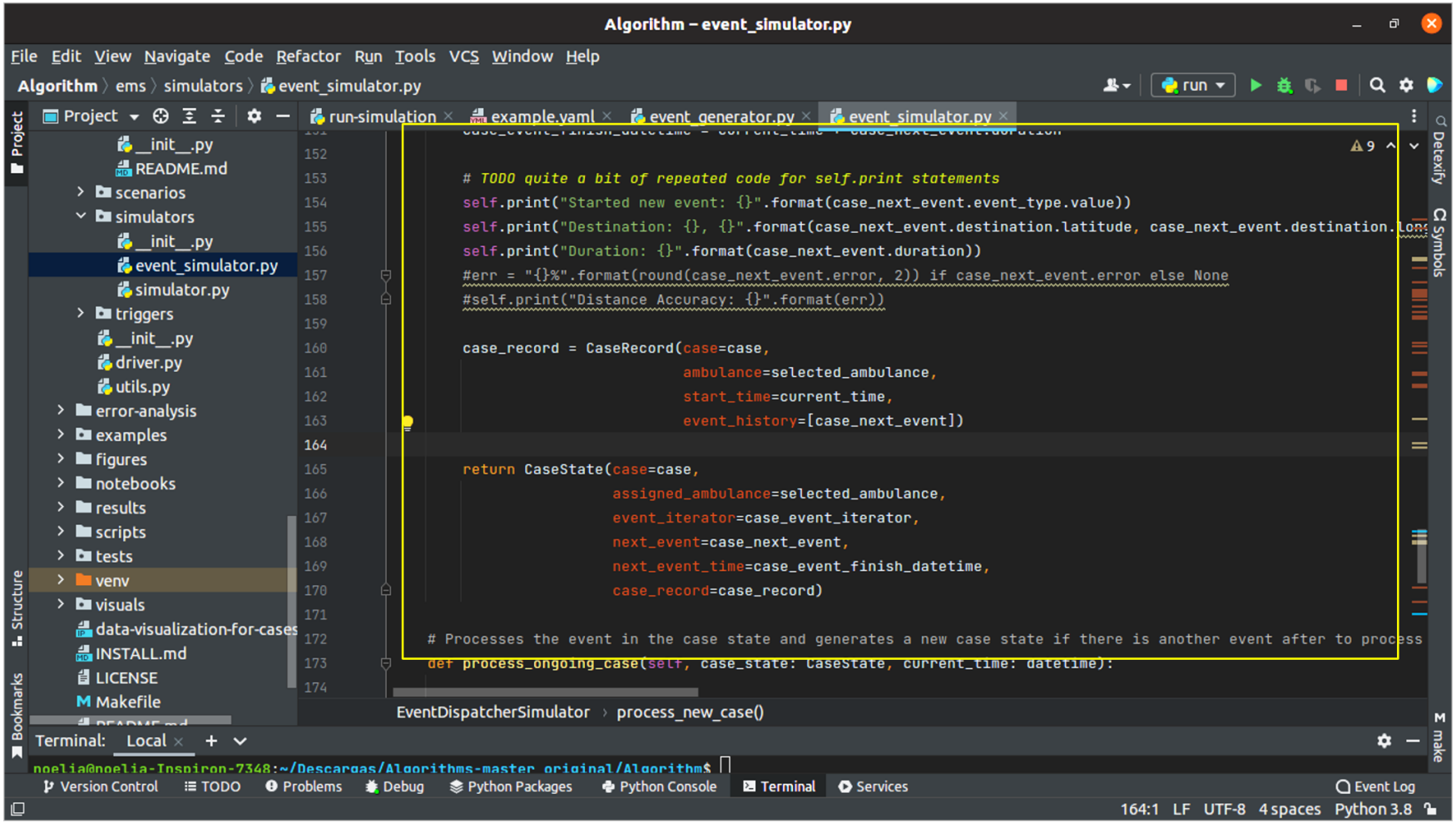


Figure 40: event_simulator.py source code

Simulator process and output

Continuing this case, if it is the last event, terminates the case.

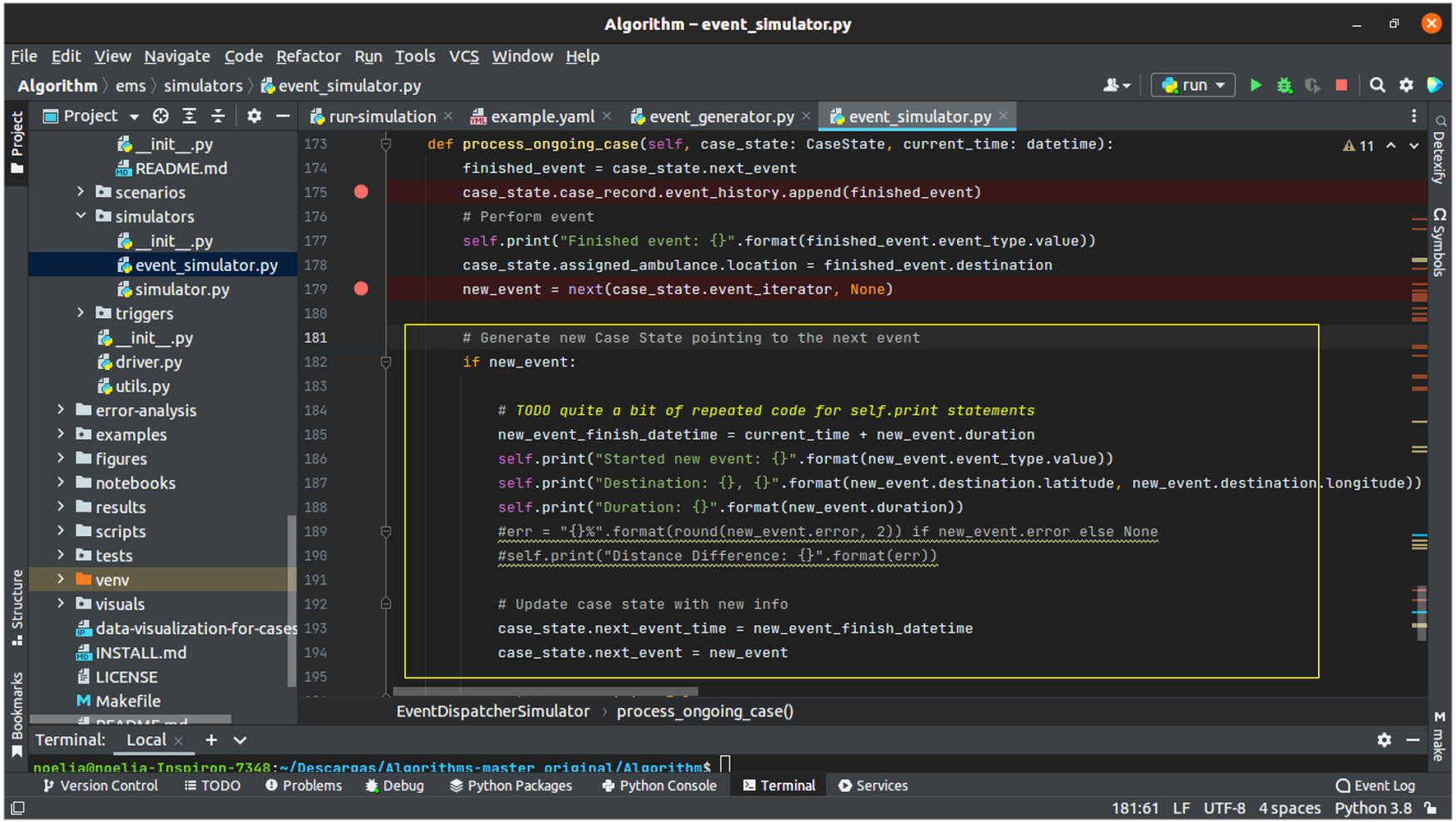


Figure 40: event_simulator.py source code

Simulator process and output

Continuing this case, if it is the last event, terminates the case.

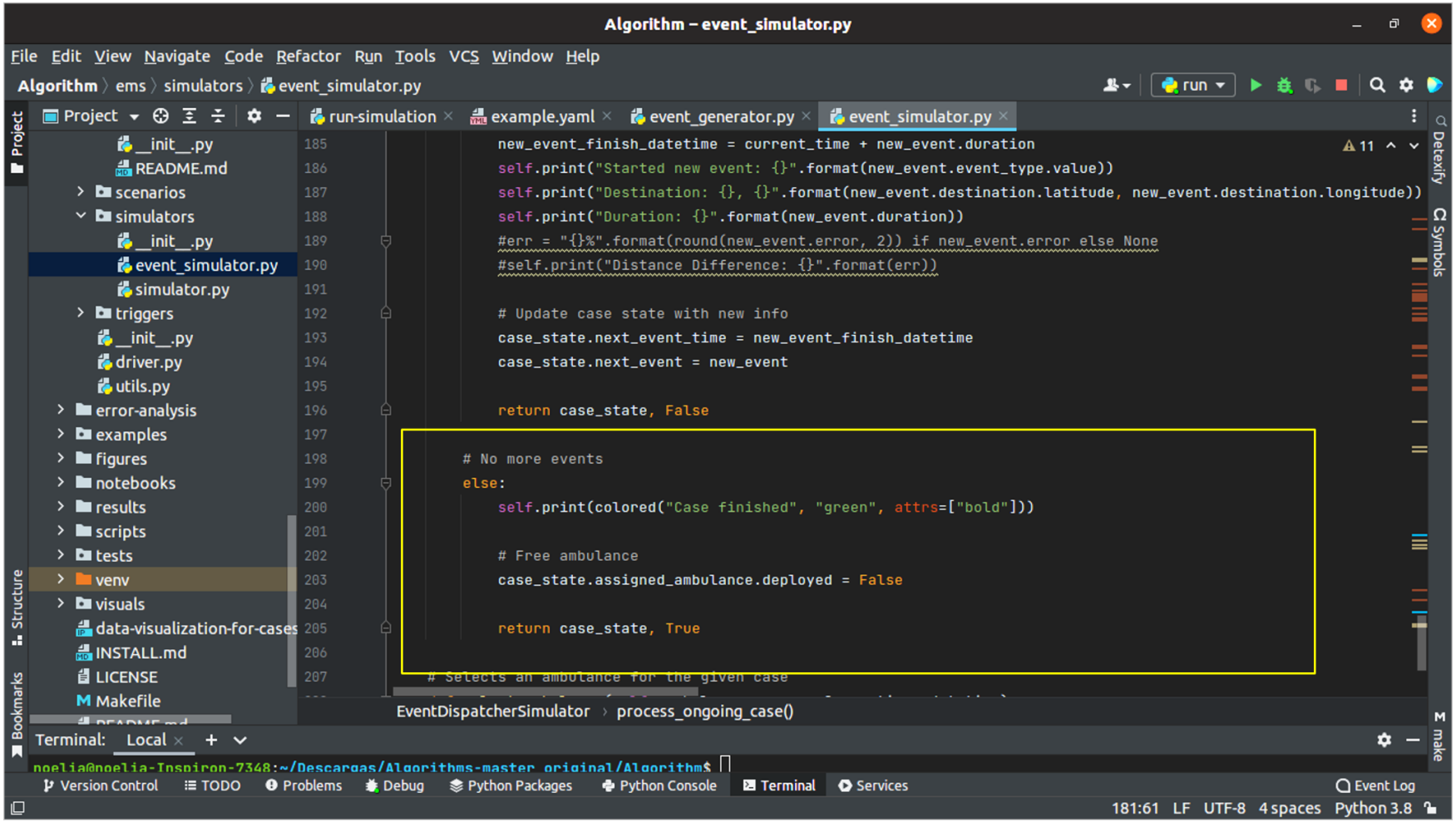


Figure 40: event_simulator.py source code

Simulator process and output

Another important point to highlight is that there is a possibility that a case may not be attended because ambulances are not available, therefore, the case remains pending until it can be addressed.

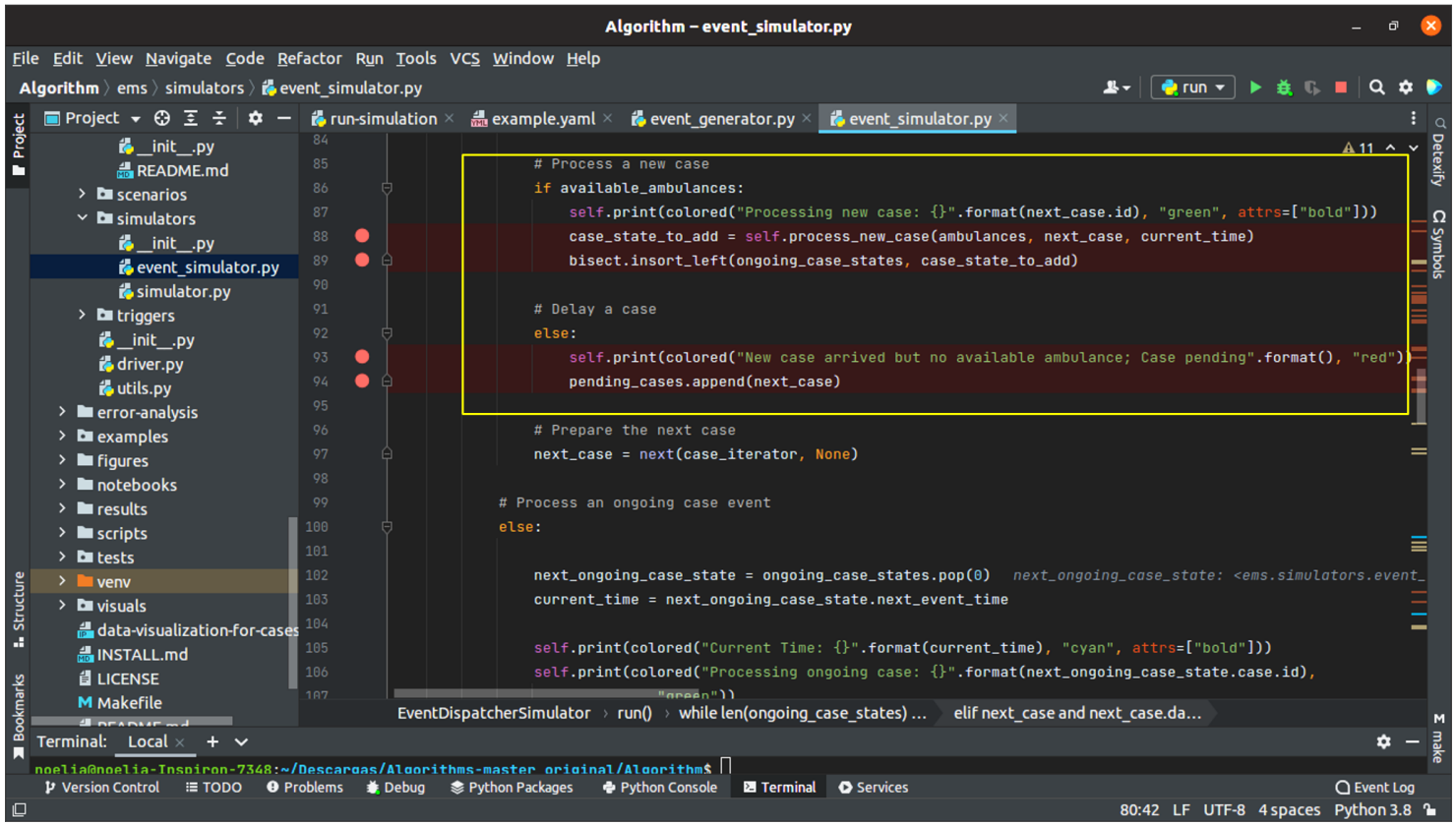


Figure 40: event_simulator.py source code

Simulator process and output

From this block you enter the module to calculate the percentage of coverage, within the minimum established time.

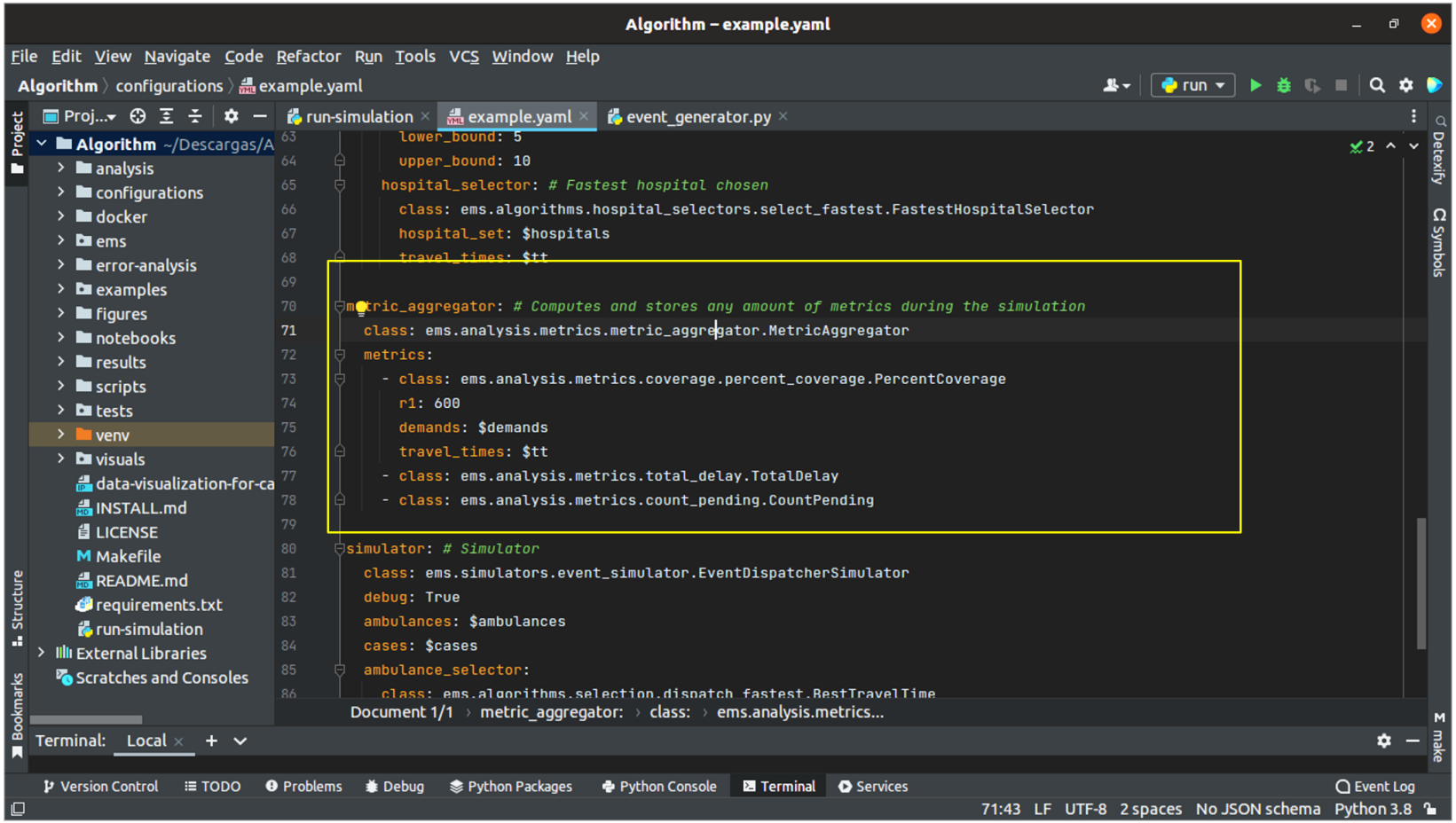


Figure 41: example.yaml source code

Simulator process and output

It is calculated for each event, this depends on how many ambulances are available and how many demand points are covered.

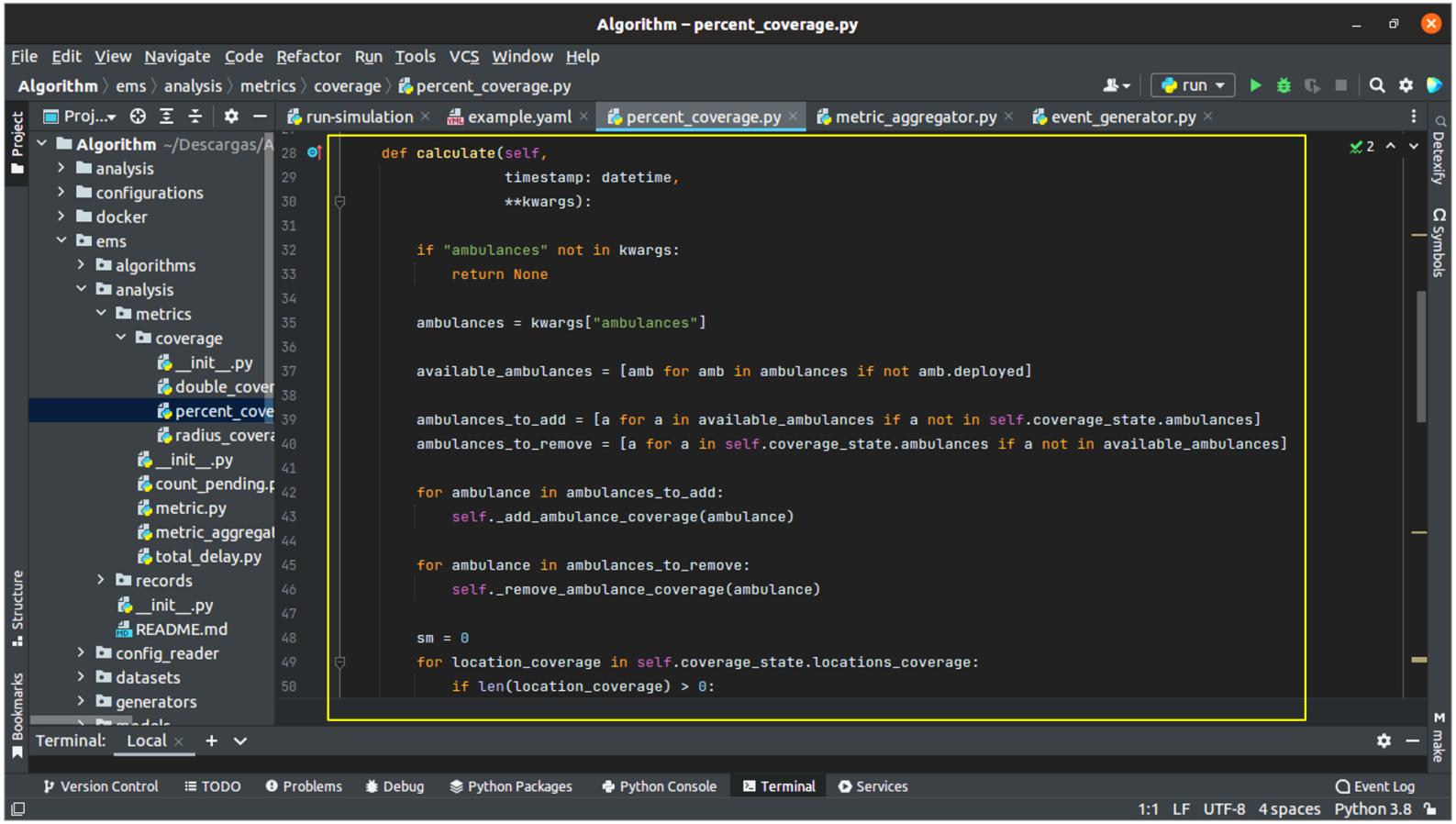


Figure 42: percent_coverage.py source code

Simulator process and output

Once the execution is completed, the results are obtained, which are the cases generated with their respective events and duration, as well as the percentage of coverage.

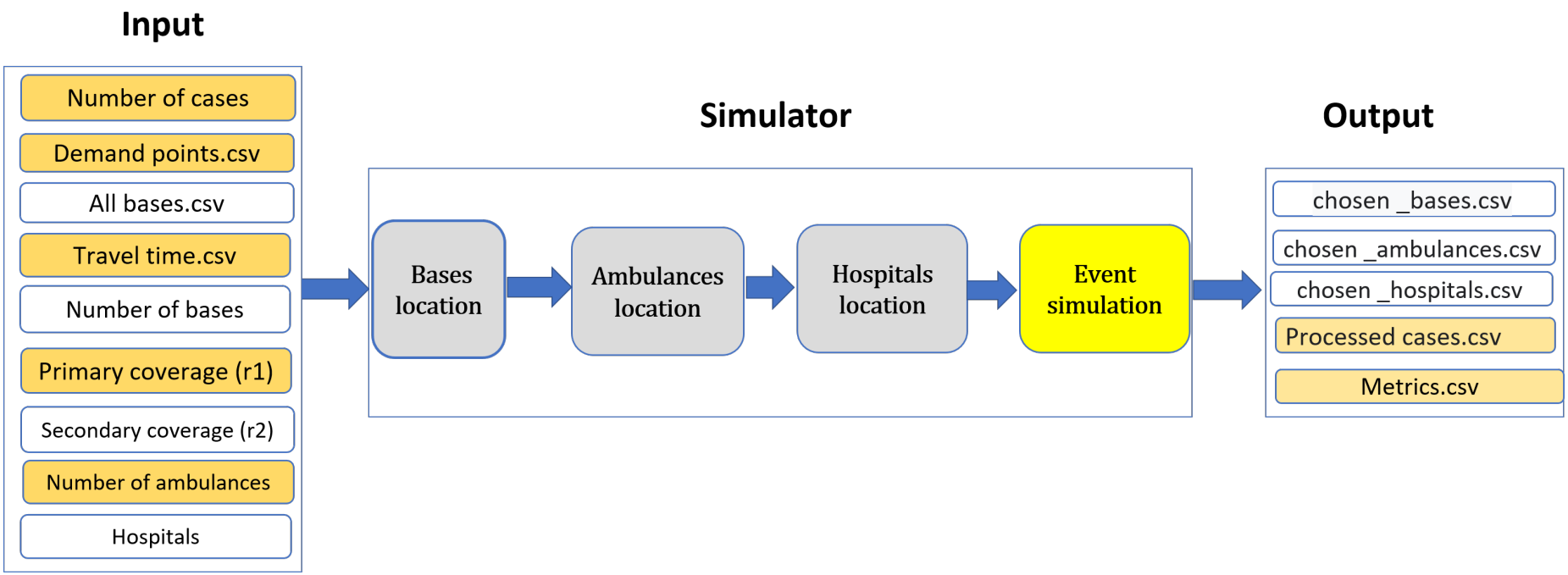
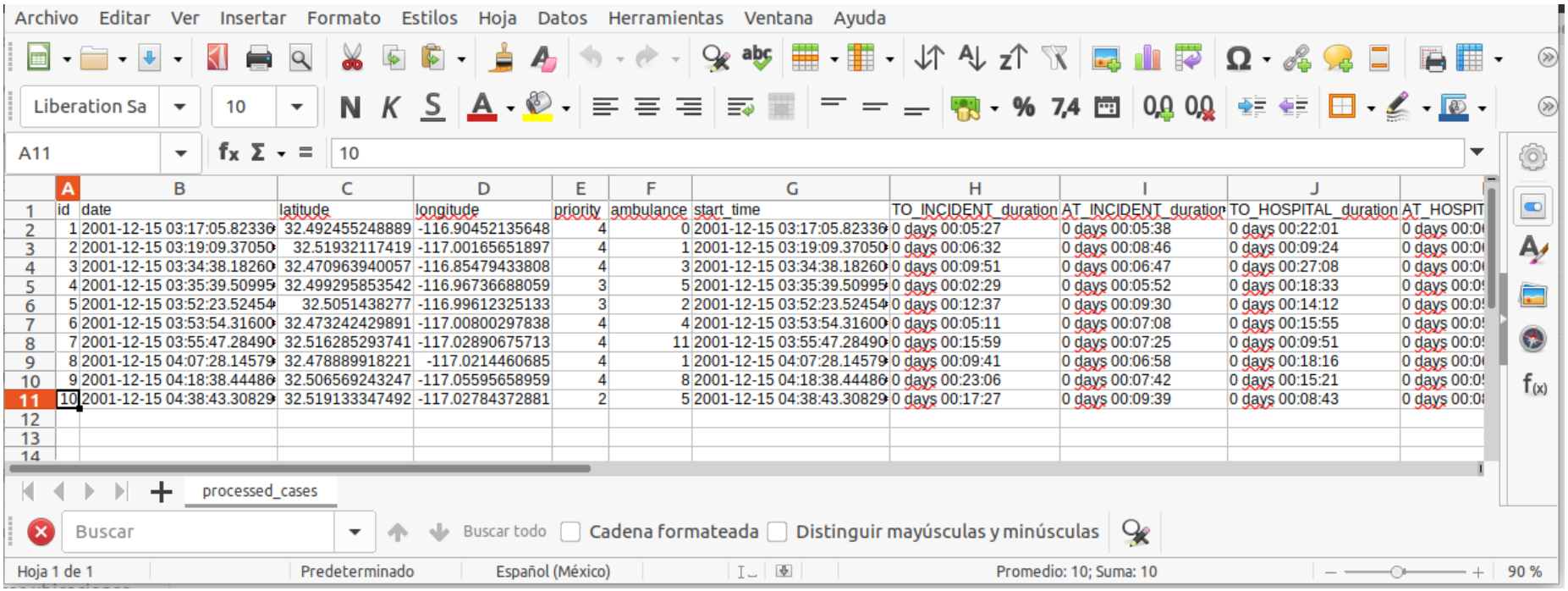


Figure 43: Simulator module.

Simulator process and output

The file contains the history of each case, the time taken for each event and the ambulance that attended the service.

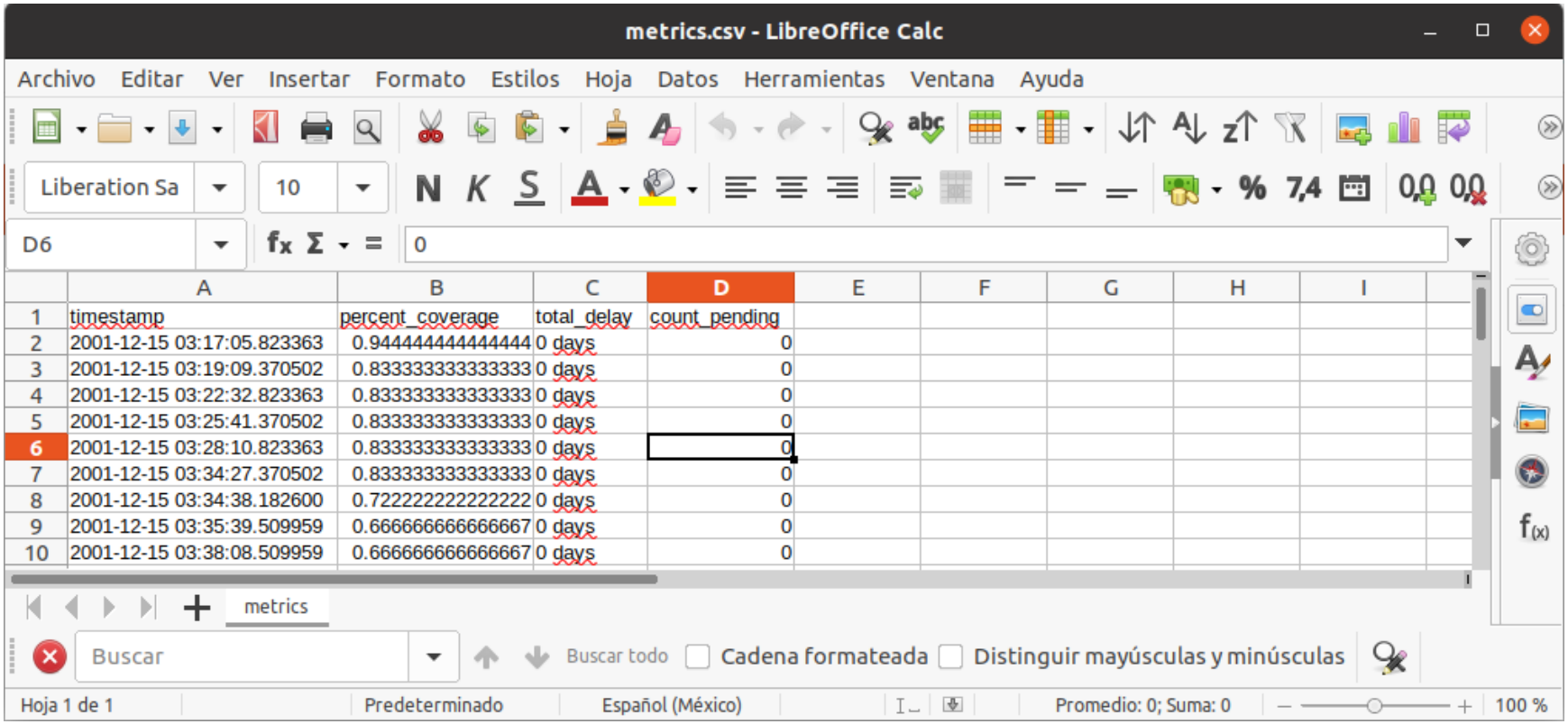


id	date	latitude	longitude	priority	ambulance	start_time	TO_INCIDENT_duration	AT_INCIDENT_duration	TO_HOSPITAL_duration	AT_HOSPIT
1	2001-12-15 03:17:05.82336	32.492455248889	-116.90452135648	4	0	2001-12-15 03:17:05.82336	0 days 00:05:27	0 days 00:05:38	0 days 00:22:01	0 days 00:0
2	2001-12-15 03:19:09.37050	32.51932117419	-117.00165651897	4	1	2001-12-15 03:19:09.37050	0 days 00:06:32	0 days 00:08:46	0 days 00:09:24	0 days 00:0
3	2001-12-15 03:34:38.18260	32.470963940057	-116.85479433808	4	3	2001-12-15 03:34:38.18260	0 days 00:09:51	0 days 00:06:47	0 days 00:27:08	0 days 00:0
4	2001-12-15 03:35:39.50995	32.499295853542	-116.96736688059	3	5	2001-12-15 03:35:39.50995	0 days 00:02:29	0 days 00:05:52	0 days 00:18:33	0 days 00:0
5	2001-12-15 03:52:23.52454	32.5051438277	-116.99612325133	3	2	2001-12-15 03:52:23.52454	0 days 00:12:37	0 days 00:09:30	0 days 00:14:12	0 days 00:0
6	2001-12-15 03:53:54.31600	32.473242429891	-117.00800297838	4	4	2001-12-15 03:53:54.31600	0 days 00:05:11	0 days 00:07:08	0 days 00:15:55	0 days 00:0
7	2001-12-15 03:55:47.28490	32.516285293741	-117.02890675713	4	11	2001-12-15 03:55:47.28490	0 days 00:15:59	0 days 00:07:25	0 days 00:09:51	0 days 00:0
8	2001-12-15 04:07:28.14579	32.478889918221	-117.0214460685	4	1	2001-12-15 04:07:28.14579	0 days 00:09:41	0 days 00:06:58	0 days 00:18:16	0 days 00:0
9	2001-12-15 04:18:38.44486	32.506569243247	-117.05595658959	4	8	2001-12-15 04:18:38.44486	0 days 00:23:06	0 days 00:07:42	0 days 00:15:21	0 days 00:0
10	2001-12-15 04:38:43.30829	32.519133347492	-117.02784372881	2	5	2001-12-15 04:38:43.30829	0 days 00:17:27	0 days 00:09:39	0 days 00:08:43	0 days 00:0

Figure 44: Processed file.

Simulator process and output

The file contains the current time of the event, the percentage of coverage of the demand points, whether it has taken more than one day to complete, and pending cases if any.



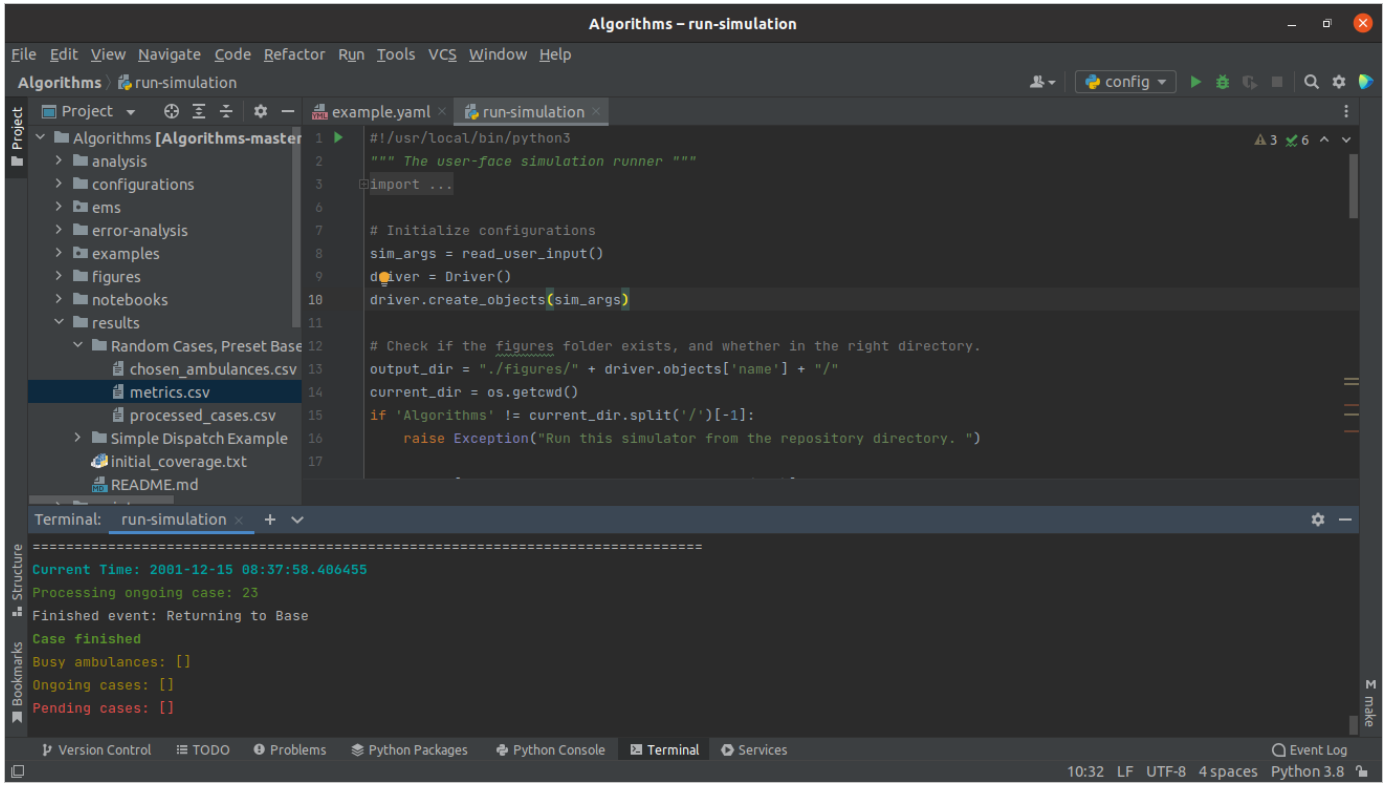
	A	B	C	D
1	timestamp	percent_coverage	total_delay	count_pending
2	2001-12-15 03:17:05.823363	0.944444444444444	0 days	0
3	2001-12-15 03:19:09.370502	0.833333333333333	0 days	0
4	2001-12-15 03:22:32.823363	0.833333333333333	0 days	0
5	2001-12-15 03:25:41.370502	0.833333333333333	0 days	0
6	2001-12-15 03:28:10.823363	0.833333333333333	0 days	0
7	2001-12-15 03:34:27.370502	0.833333333333333	0 days	0
8	2001-12-15 03:34:38.182600	0.722222222222222	0 days	0
9	2001-12-15 03:35:39.509959	0.666666666666667	0 days	0
10	2001-12-15 03:38:08.509959	0.666666666666667	0 days	0

Figure 44: Metrics file.

Simulator input

The video shows the process of the simulator (original version).

Watch video



6. Integration of the simulator module to the interactive system for ambulance dispatching in the city of Tijuana

Integration of the Simulator module

The system proposed to be developed presents significant advances in the problem addressed.

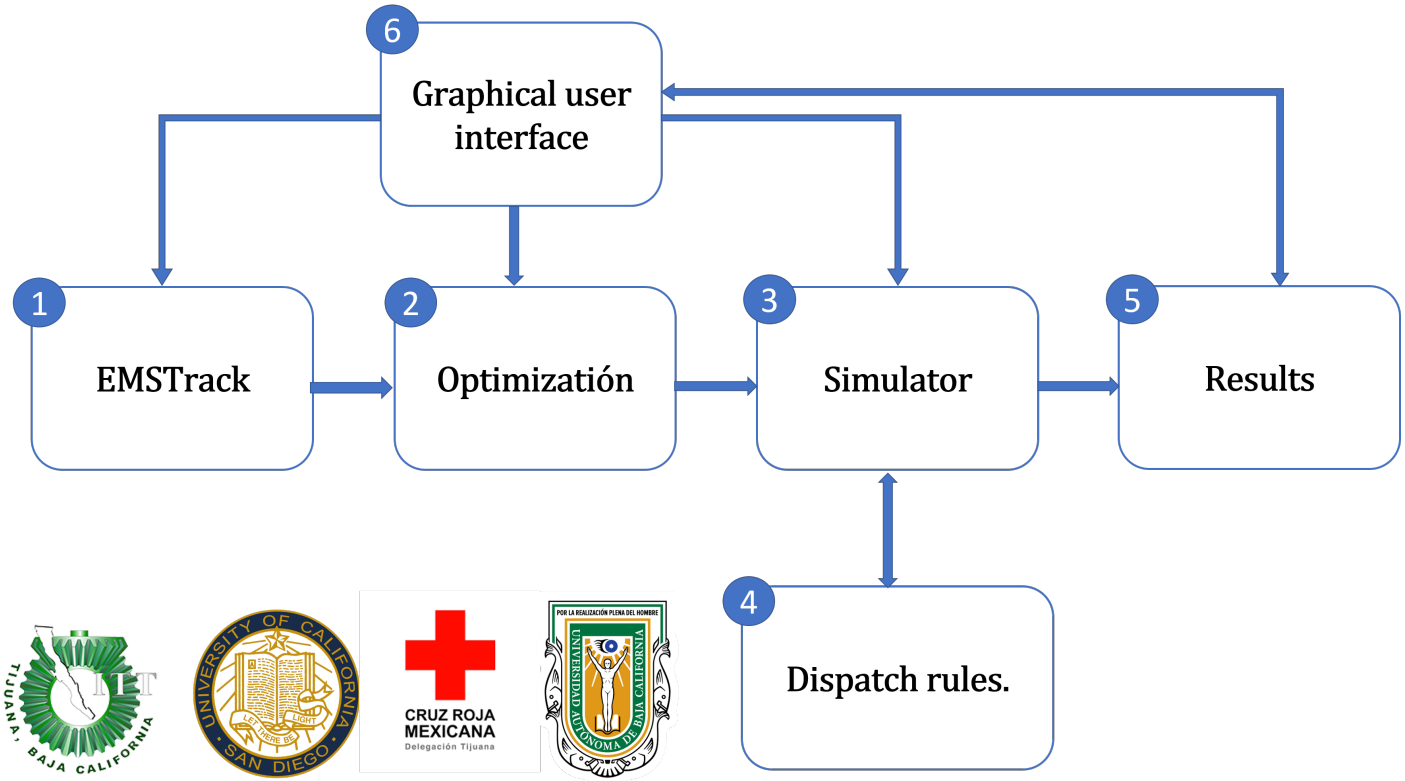


Figure 45: Tijuana Red Cross interactive System Diagram.

Integration of the Simulator module

We have EMSTrack, a software module developed by UCSD, for ambulance management and tracking, which allows real-time geolocation of the units and allows detailed general call histories.

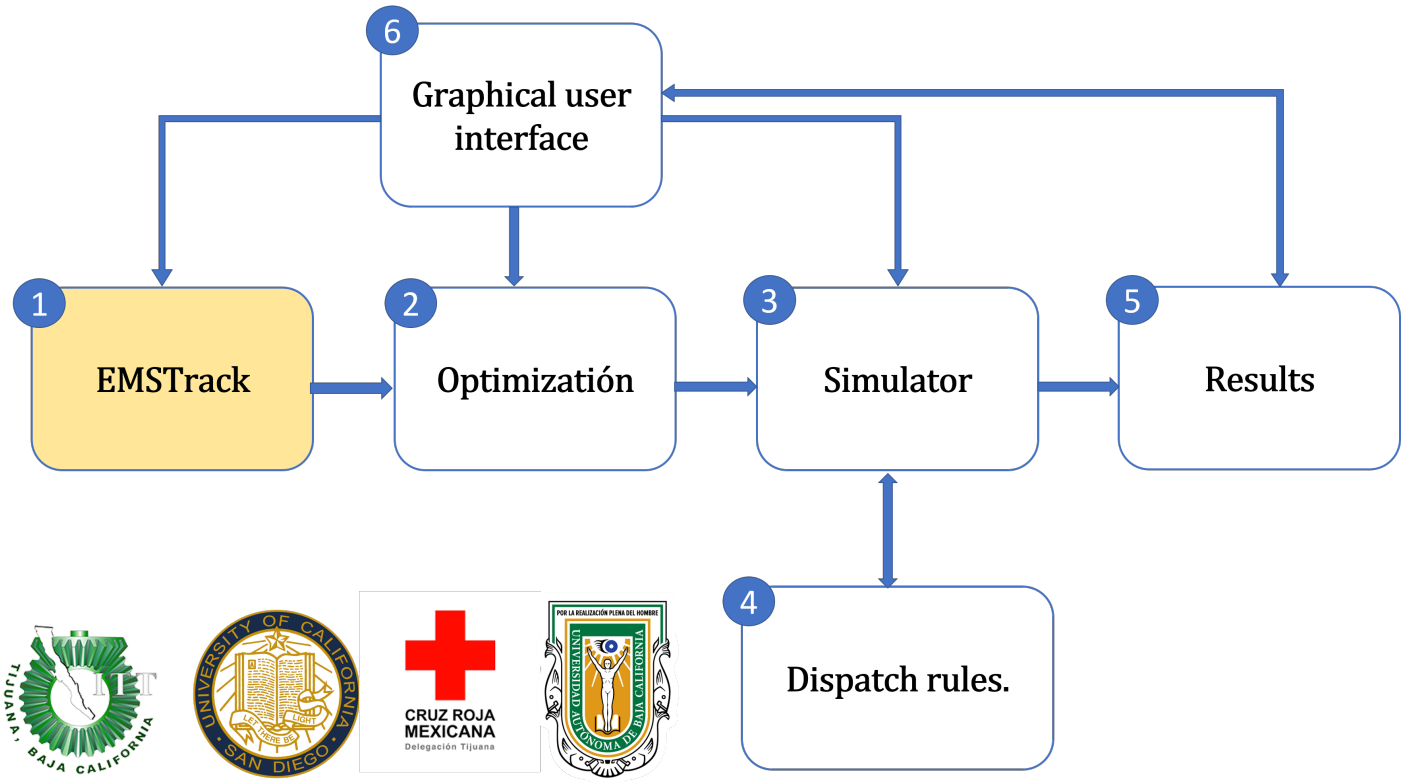


Figure 45: Tijuana Red Cross interactive System Diagram.

Integration of the Simulator module

Continuing with the next optimization module, which includes methods for analyzing and clustering the historical call data generated in module 1, as well as for modeling and optimizing problems and relocation problems based on the DSM model.

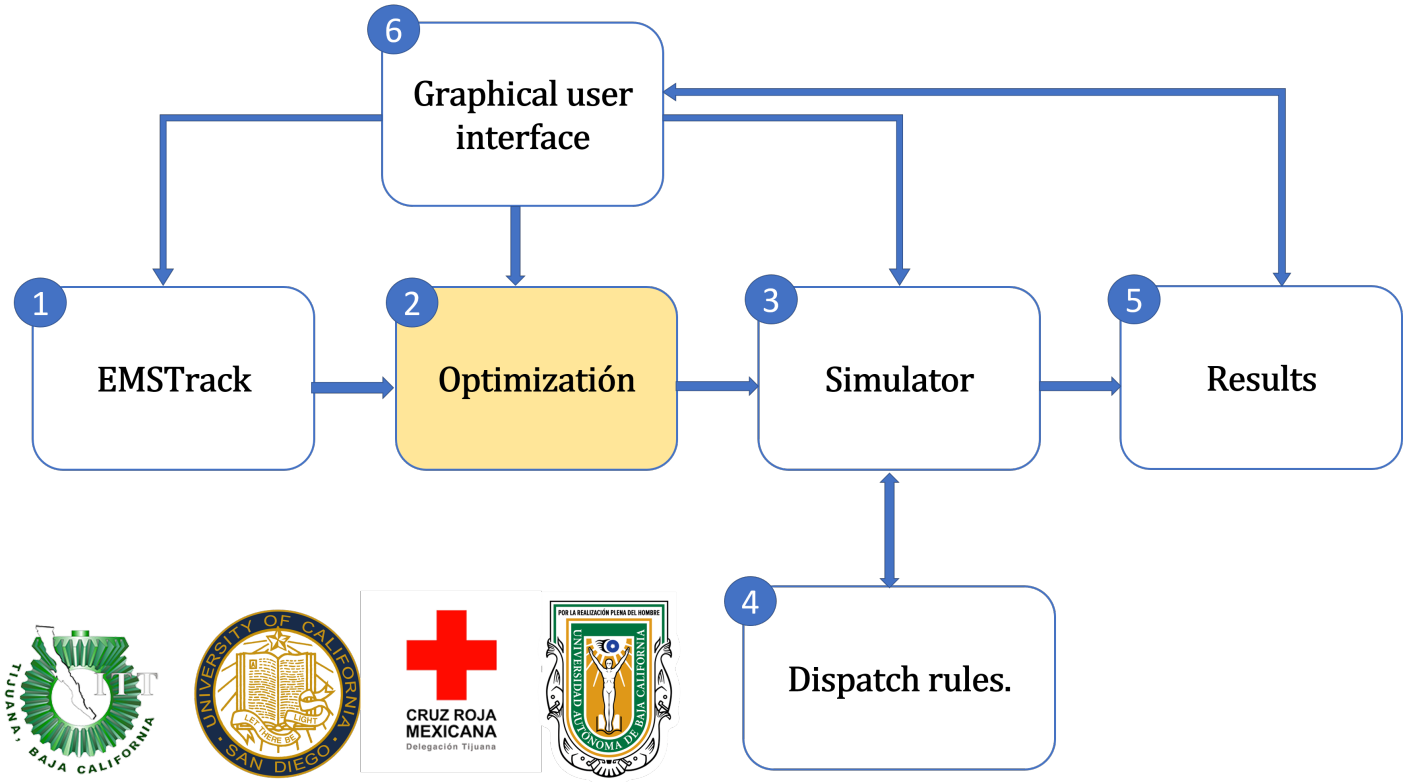


Figure 45: Tijuana Red Cross interactive System Diagram.

Integration of the Simulator module

In module 3 we have the simulator that is based on state machines that allow us to evaluate different events, call frequency, ambulance location and response times.

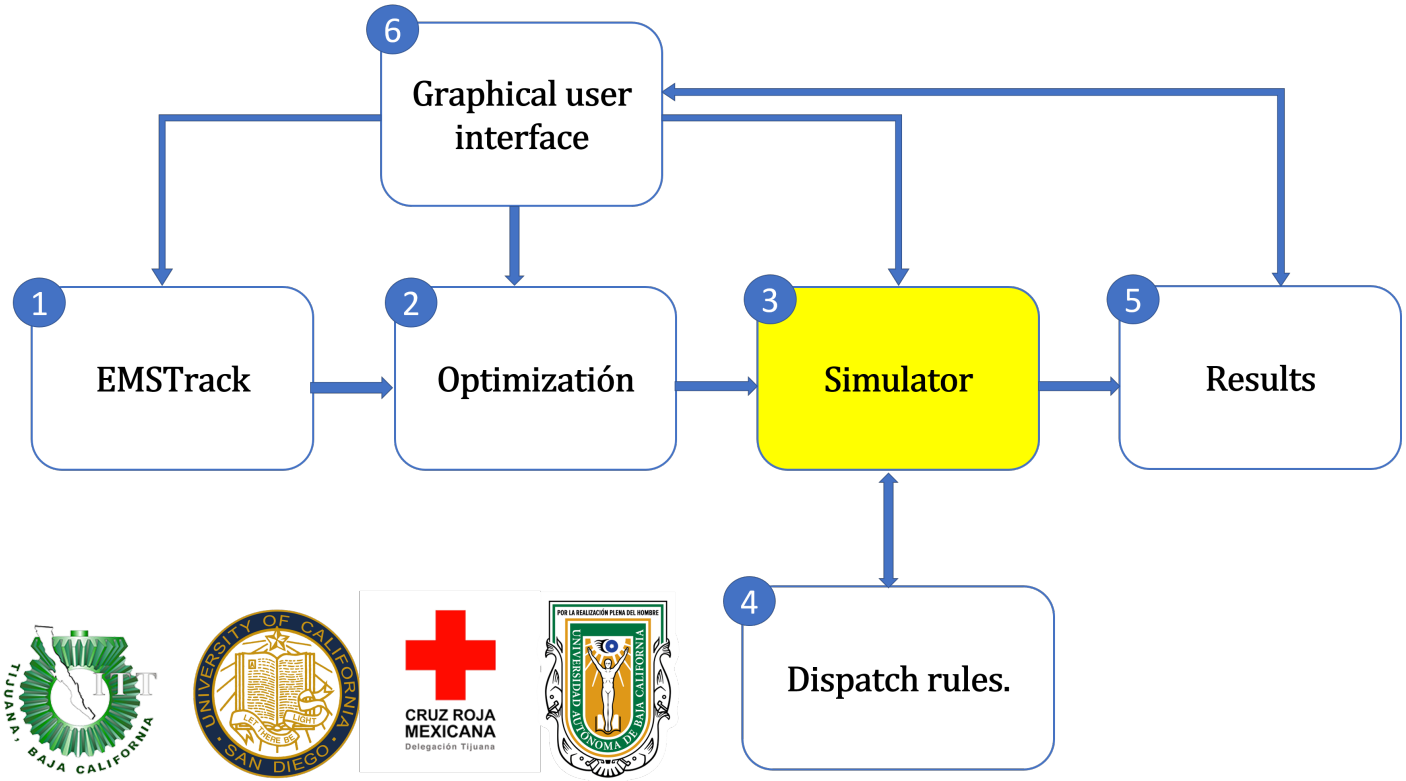


Figure 45: Tijuana Red Cross interactive System Diagram.

Integration of the Simulator module

Module 4 corresponds to the ambulance dispatch rules and is pending to be added.

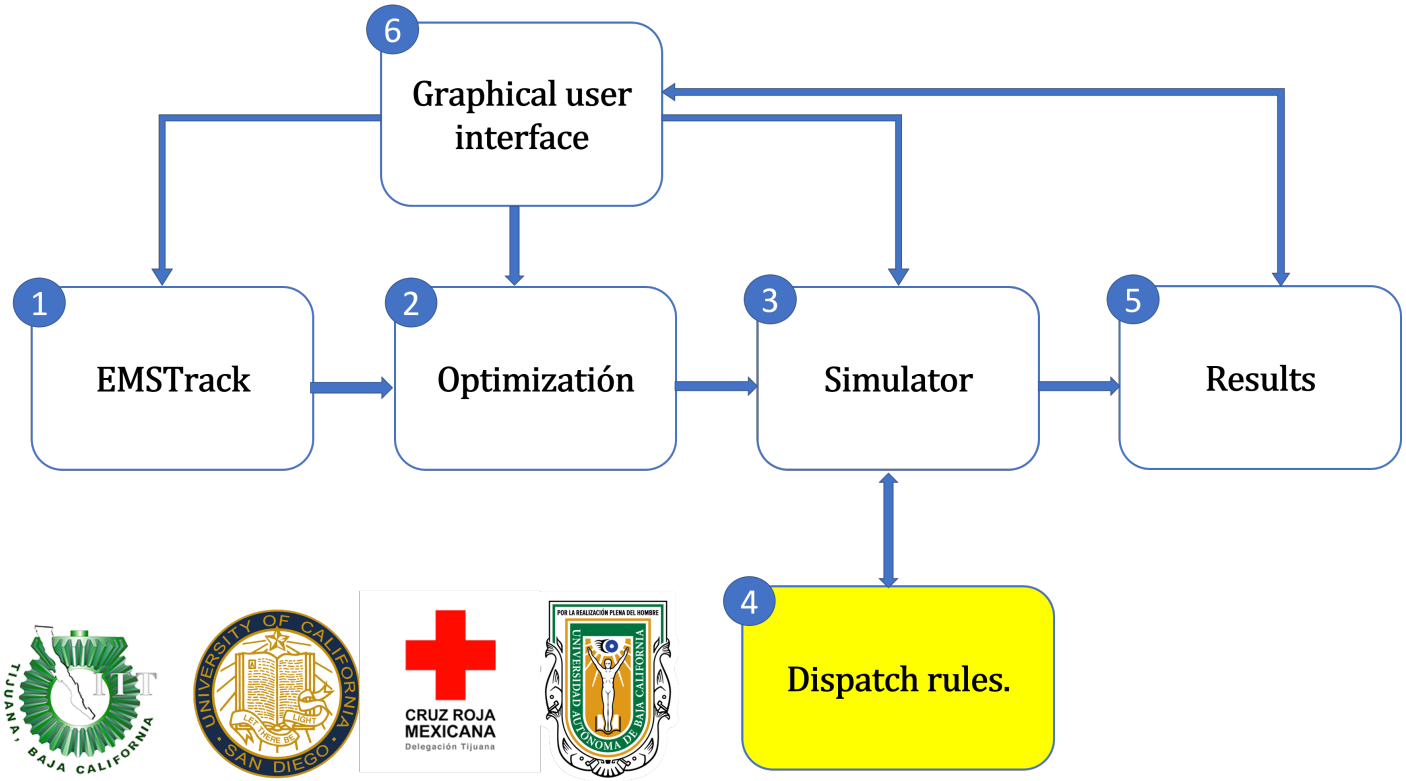


Figure 45: Tijuana Red Cross interactive System Diagram.

Integration of the Simulator module

Module 5 generates and organizes the results in a structured way for the administrative personnel in charge of decision making. And finally, Module 6 is the graphical user interface that will allow interacting with the system.

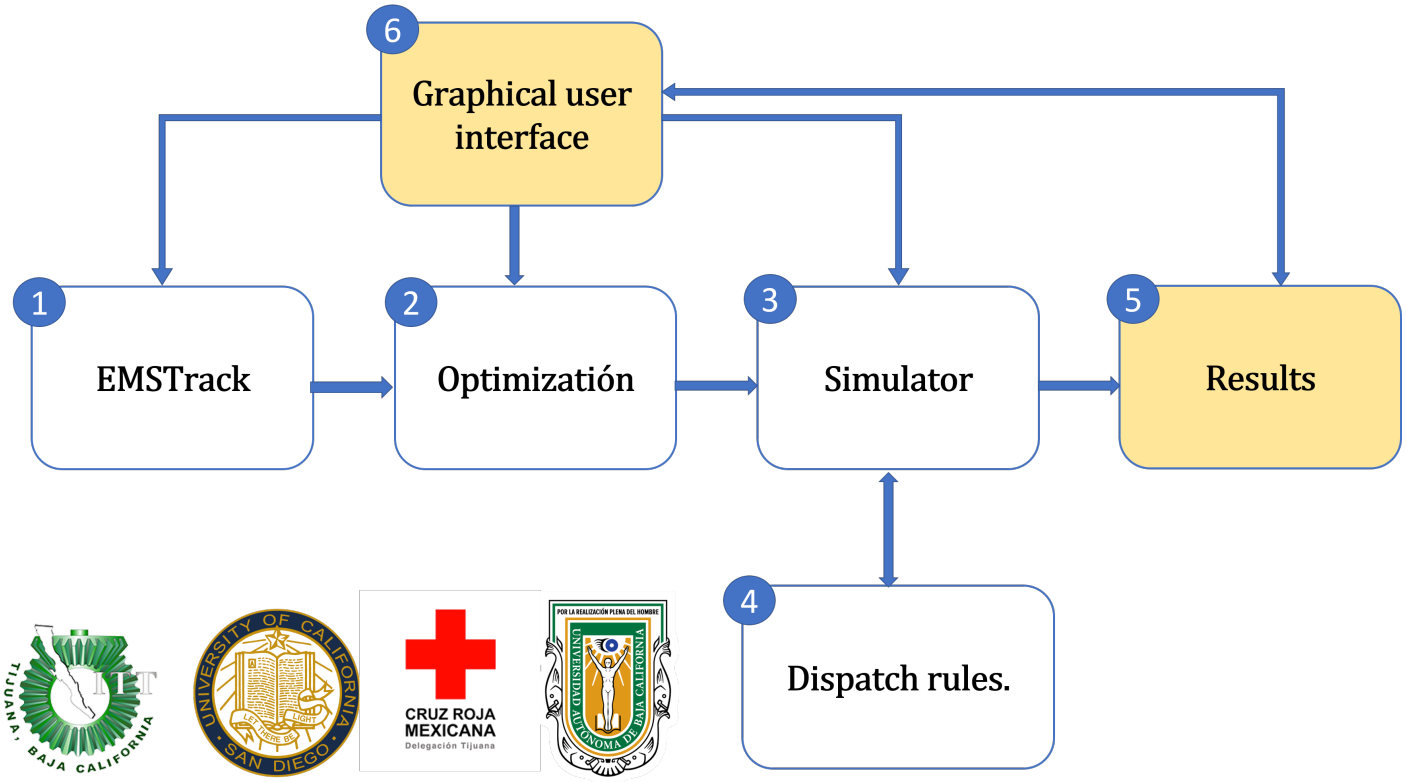


Figure 45: Tijuana Red Cross interactive System Diagram.

6.1 Updated simulator module

Updated simulator module

Returning to the simulator module in its original version.

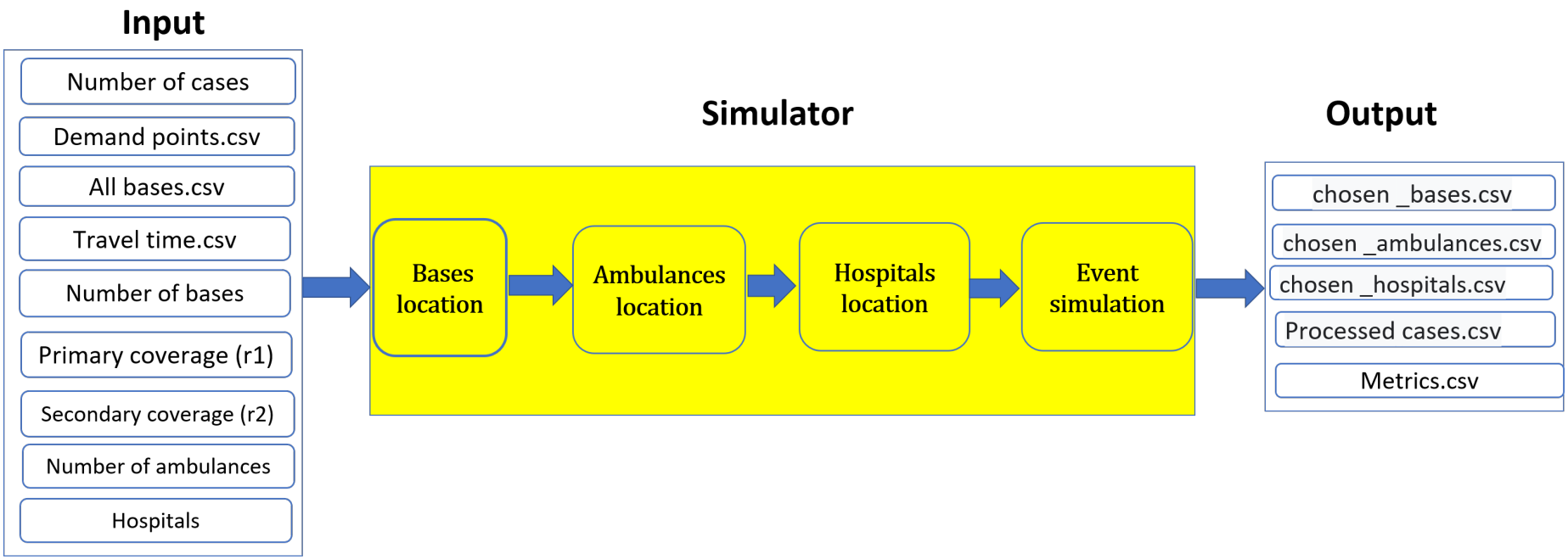


Figure 46: Simulator module.

Updated simulator module

The changes that have been made are taking into account that in the future it will be integrated into the system.

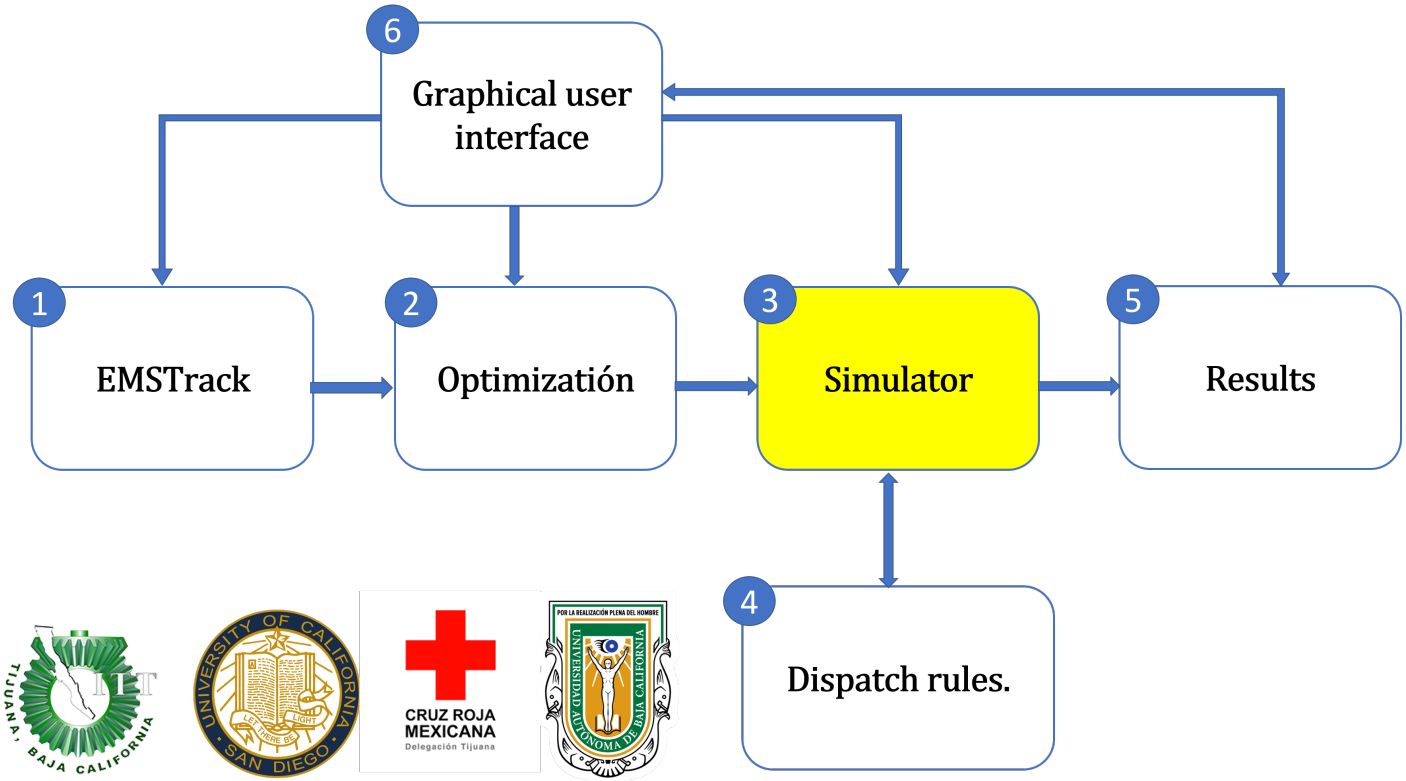


Figure 47: Tijuana Red Cross interactive System Diagram.

Updated simulator module

The changes made are:

1. The way of establishing the number of calls or emergency cases to be simulated, which were randomly generated from time to time, was modified. Now they are loaded from a file in .csv format, which contains the calls, information provided by the Red Cross of Tijuana.

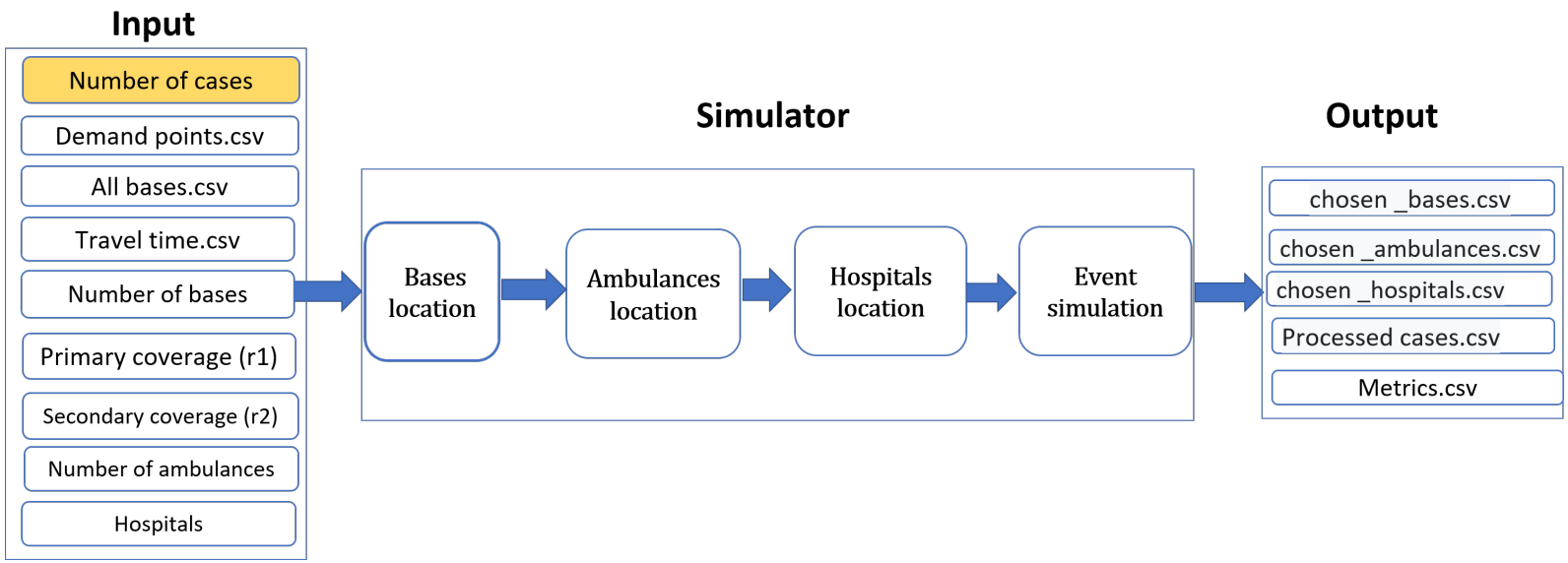


Figure 48: Tijuana Red Cross interactive System Diagram.

Updated simulator module

The changes made are:

1. The way of establishing the number of calls or emergency cases to be simulated, which were randomly generated from time to time, was modified. Now they are loaded from a file in .csv format, which contains the calls, information provided by the Red Cross of Tijuana.

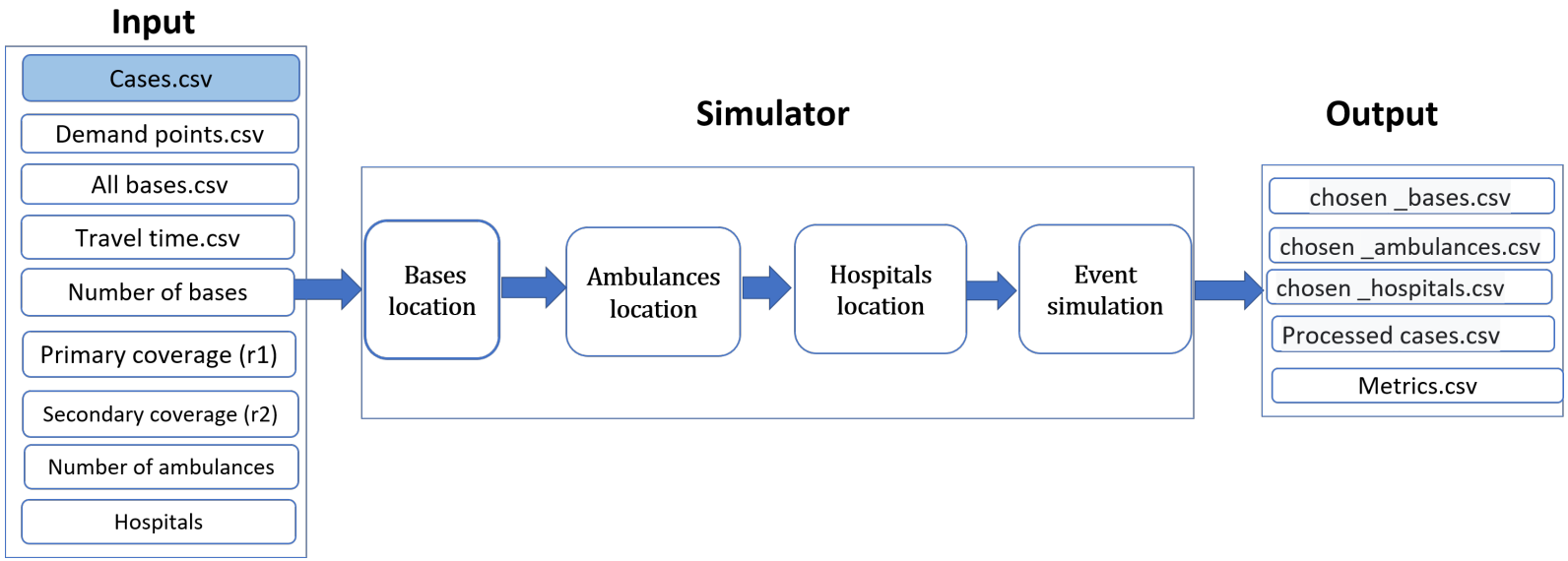
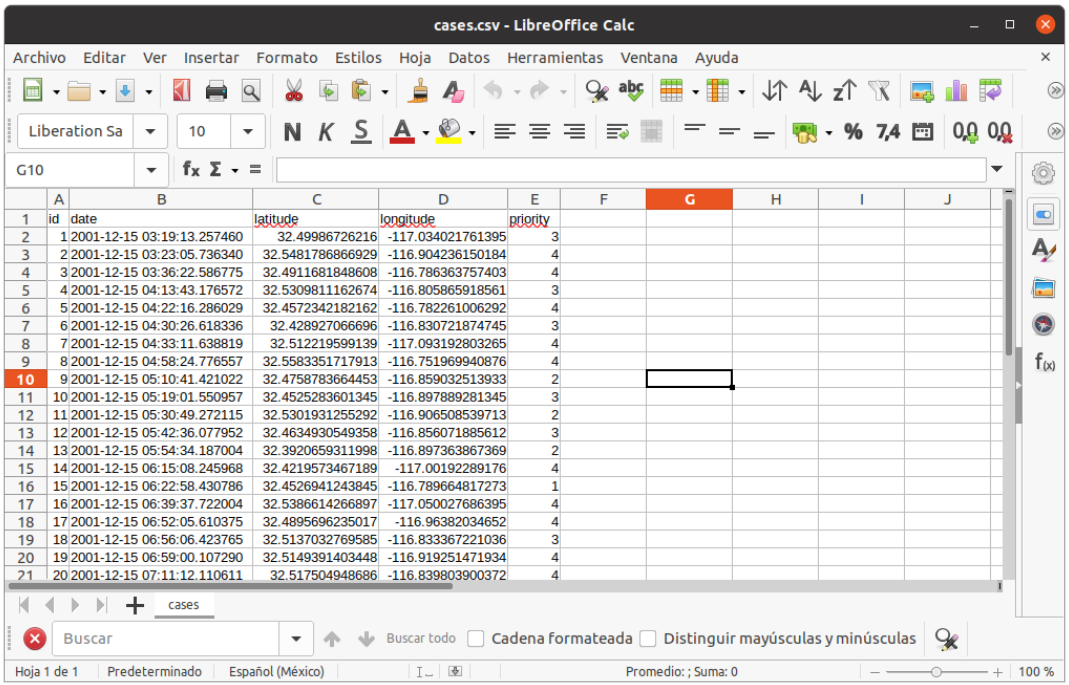


Figure 48: Tijuana Red Cross interactive System Diagram.

Updated simulator module

The file contains:

- The start time of the cases
- The coordinates of the cases
- The priority of the call, which is not yet taken into account for sending the ambulance.



	A	B	C	D	E
1	id	date	latitude	longitude	priority
2	1	2001-12-15 03:19:13.257460	32.49986726216	-117.034021761395	3
3	2	2001-12-15 03:23:05.736340	32.5481786866929	-116.904236150184	4
4	3	2001-12-15 03:36:22.586775	32.4911681848608	-116.786363757403	4
5	4	2001-12-15 04:13:43.176572	32.5309811162674	-116.805865918561	3
6	5	2001-12-15 04:22:16.286029	32.4572342182162	-116.782261006292	4
7	6	2001-12-15 04:30:26.618336	32.428927066696	-116.830721874745	3
8	7	2001-12-15 04:33:11.638819	32.512219599139	-117.093192803265	4
9	8	2001-12-15 04:58:24.776557	32.5583351717913	-116.751969940876	4
10	9	2001-12-15 05:10:41.421022	32.4758783664453	-116.859032513933	2
11	10	2001-12-15 05:19:01.550957	32.4525283601345	-116.897889281345	3
12	11	2001-12-15 05:30:49.272115	32.5301931255292	-116.906508539713	2
13	12	2001-12-15 05:42:36.077952	32.4634930549358	-116.856071885612	3
14	13	2001-12-15 05:54:34.187004	32.3920659311998	-116.897363867369	2
15	14	2001-12-15 06:15:08.245968	32.4219573467189	-117.00192289176	4
16	15	2001-12-15 06:22:58.430786	32.4526941243845	-116.789664817273	1
17	16	2001-12-15 06:39:37.722004	32.5386614266897	-117.050027686395	4
18	17	2001-12-15 06:52:05.610375	32.4895696235017	-116.96382034652	4
19	18	2001-12-15 06:56:06.423765	32.5137032769585	-116.833367221036	3
20	19	2001-12-15 06:59:00.107290	32.5149391403448	-116.919251471934	4
21	20	2001-12-15 07:11:12.110611	32.517504948686	-116.839803900372	4

Figure 49: Cases file.

Updated simulator module

To omit this process in the original version, we just delete the **case_quantity** that had the value of the cases to be generated and the **case** module, which generates the coordinates and the time of the case, processing now the cases from a csv file.

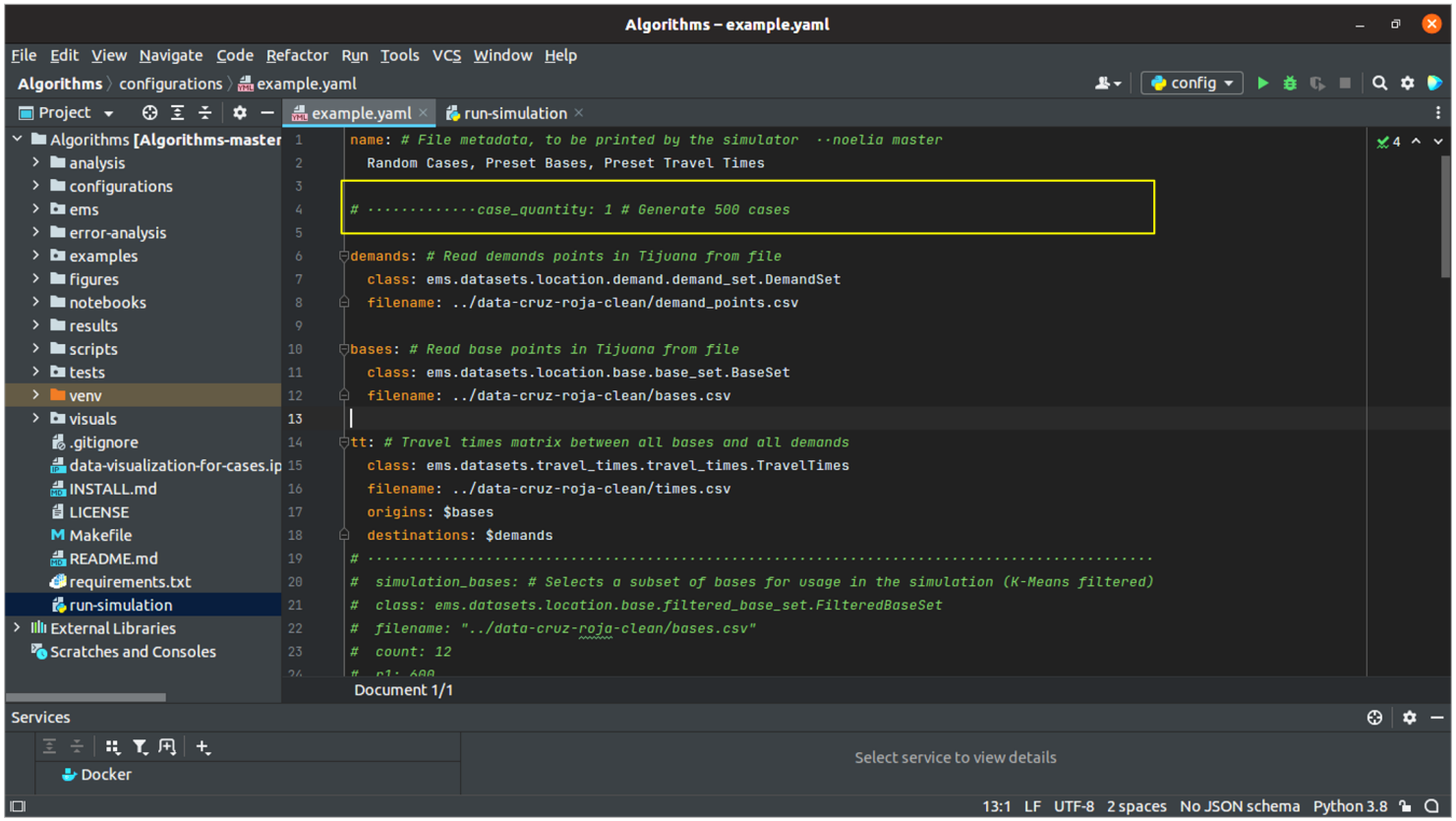


Figure 50: example.yaml source code

Updated simulator module

To omit this process in the original version, we just delete the **case_quantity** that had the value of the cases to be generated and the **case** module, which generates the coordinates and the time of the case, processing now the cases from a csv file.

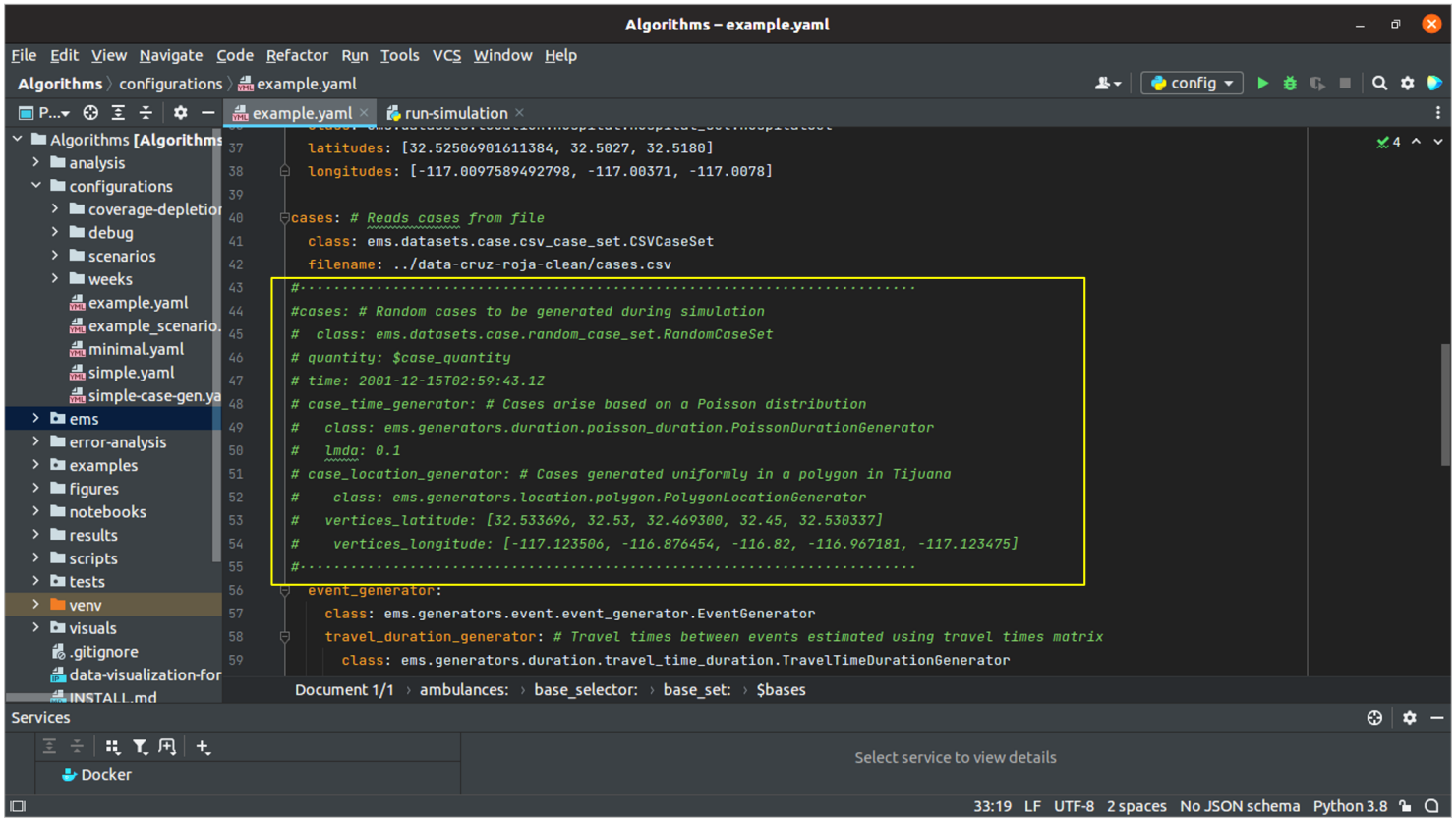


Figure 50: example.yaml source code

Updated simulator module

2. In the optimization module, there is a sub-module with a clustering algorithm, which allows to determine from a set of calls, the optimal number of clusters, called demand points, saving the results in the file demand_points.csv. This is the file currently in use.

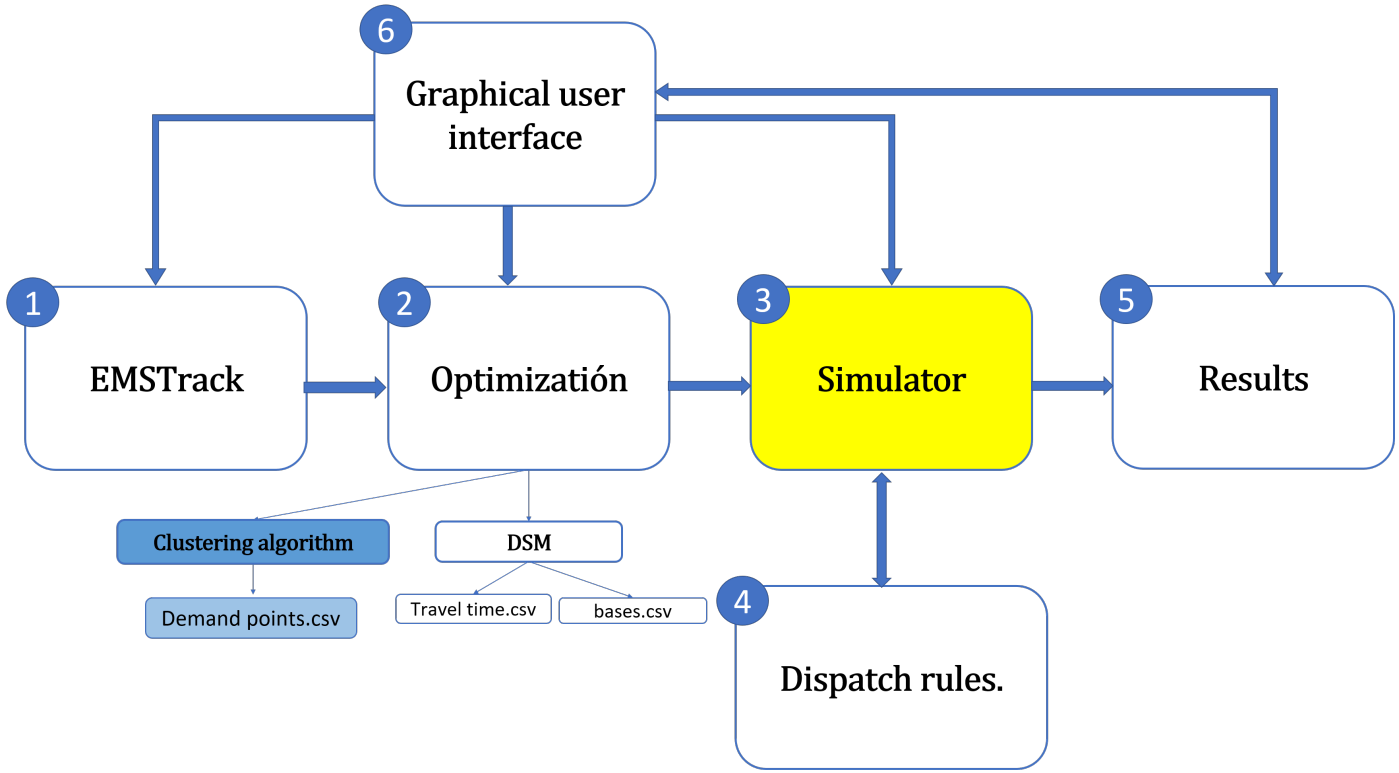


Figure 51: Tijuana Red Cross interactive System Diagram.

Updated simulator module

2. In the optimization module, there is a sub-module with a clustering algorithm, which allows to determine from a set of calls, the optimal number of clusters, called demand points, saving the results in the file demand_points.csv. This is the file currently in use.

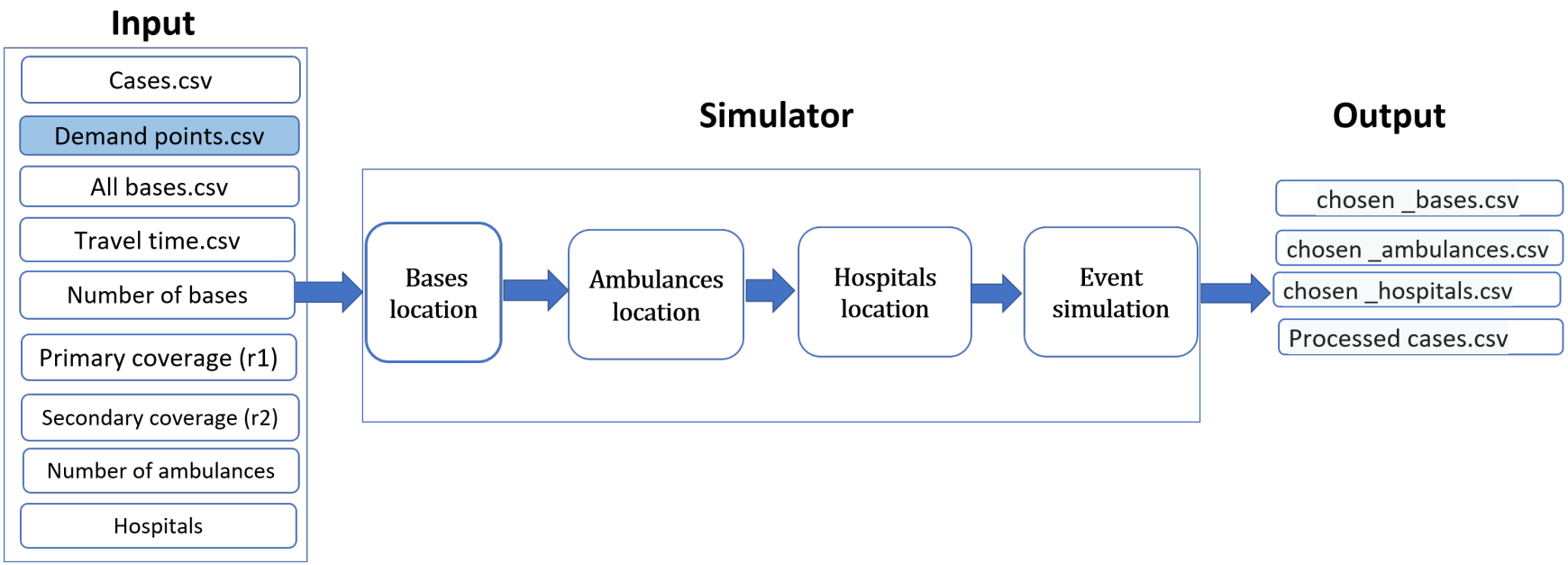


Figure 51: Simulator module.

Updated simulator module

3. Another optimization sub-module is the Double Standard Model (DSM), which maximizes the demand covered by at least two ambulances in a small radius and allows determining from a set of bases the ones that offer the best coverage, assigning a minimum of one ambulance and a maximum of two ambulances, while calculating the travel times from each of the bases to each demand point, using the Open Source Routing Machine (OSRM).

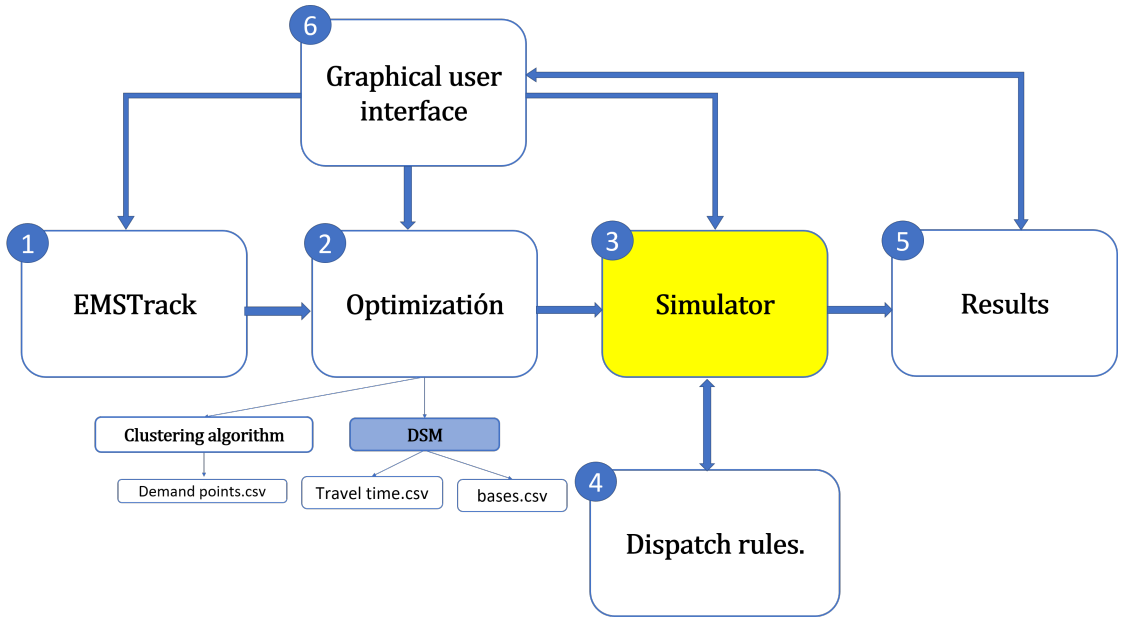


Figure 52: Tijuana Red Cross interactive System Diagram.

Updated simulator module

Saving the information in csv file.

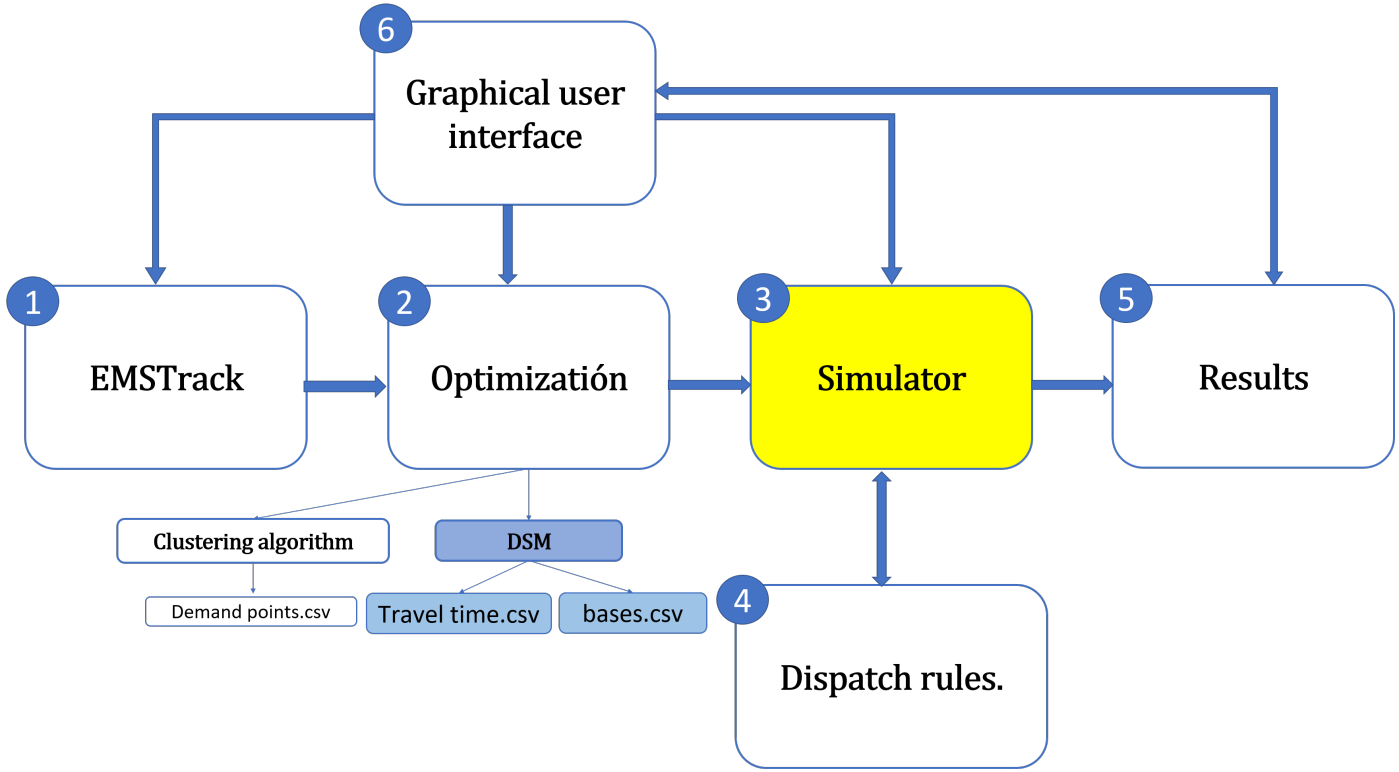


Figure 53: Tijuana Red Cross interactive System Diagram.

Updated simulator module

4.Thus, in the original version of the simulator, the block that generated the bases for dispatch was eliminated and the DSM results are used.

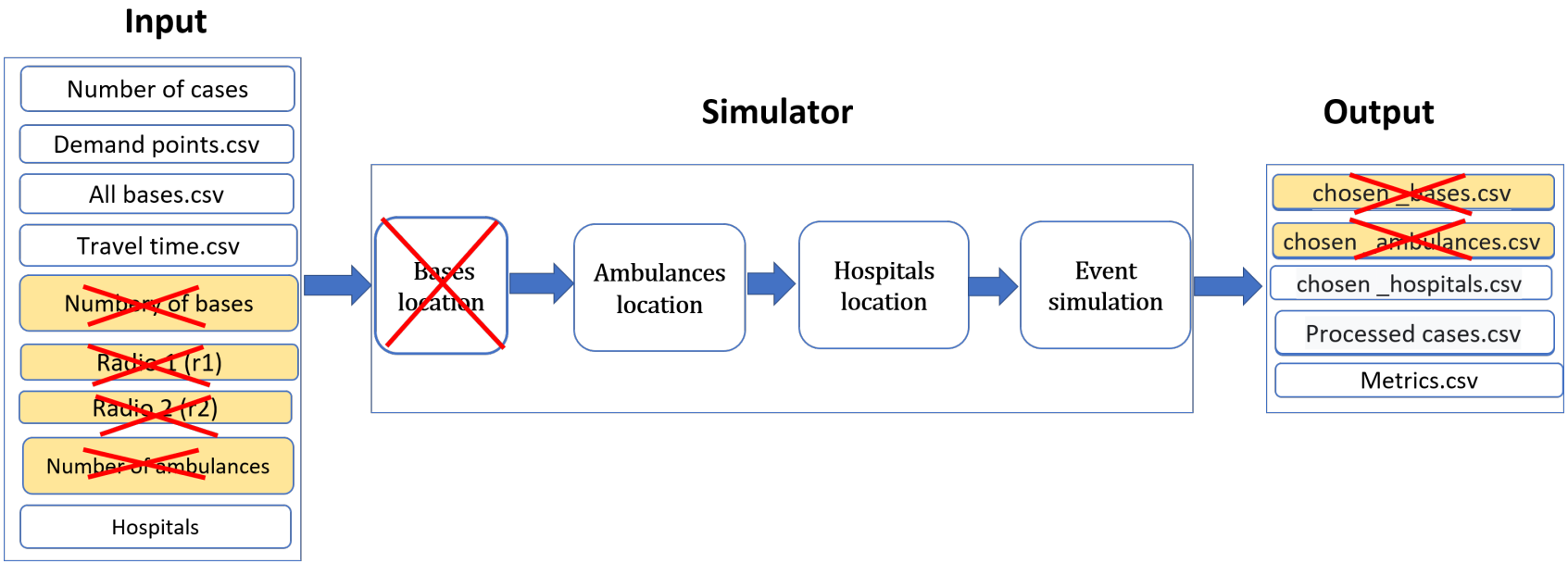


Figure 54: Simulator module.

Updated simulator module

To skip this process in the original version just remove the **simulation_bases** block.

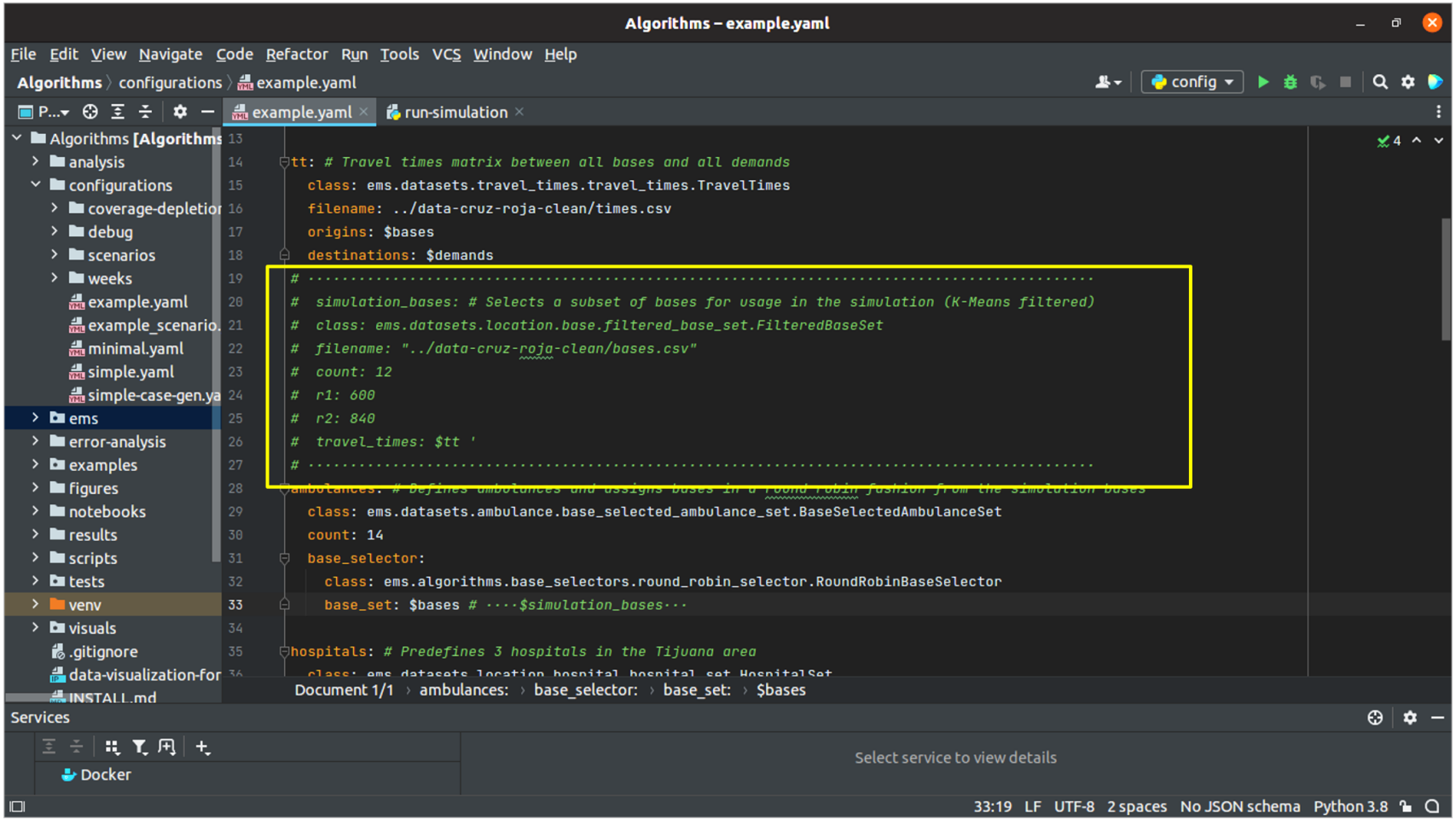


Figure 55: Tijuana Red Cross interactive System Diagram.

Updated simulator module

This is how the files currently in use are obtained.

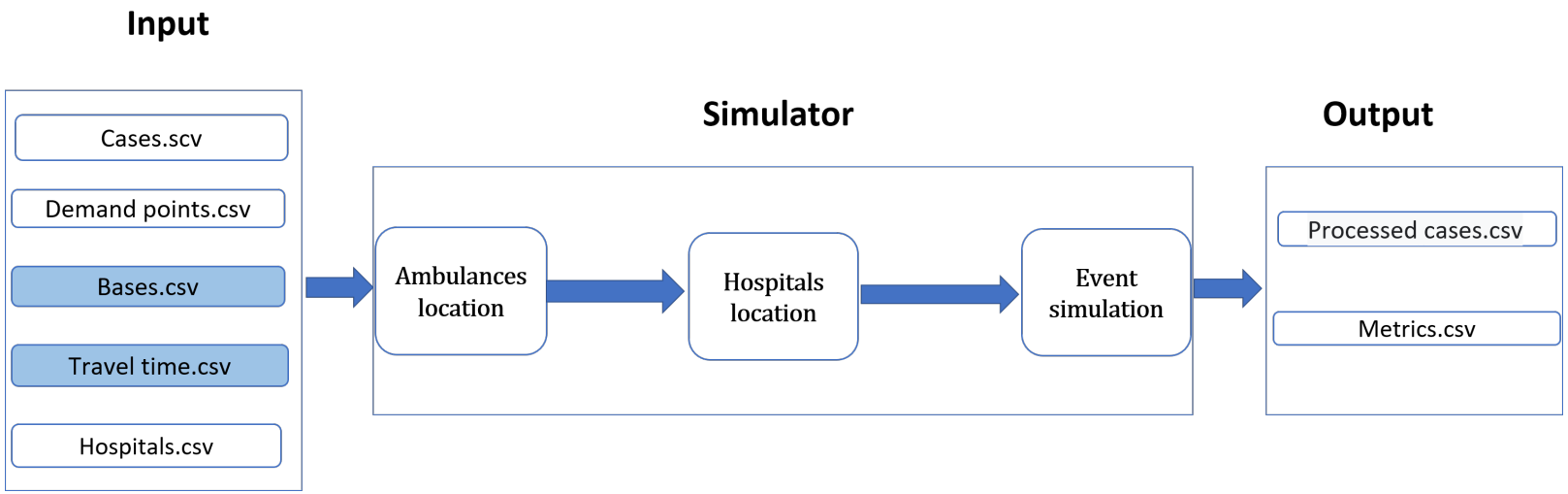


Figure 56: Simulator module.

Updated simulator module

5. Hospital locations are entered using a file with a .csv extension, because there can be many hospitals.

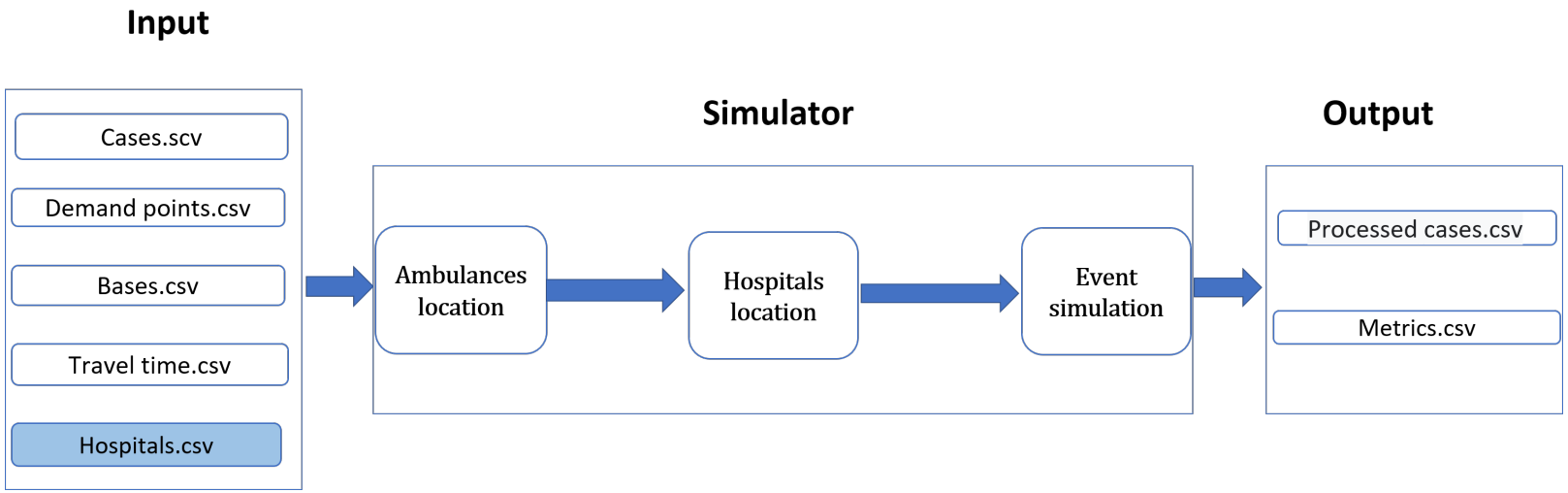
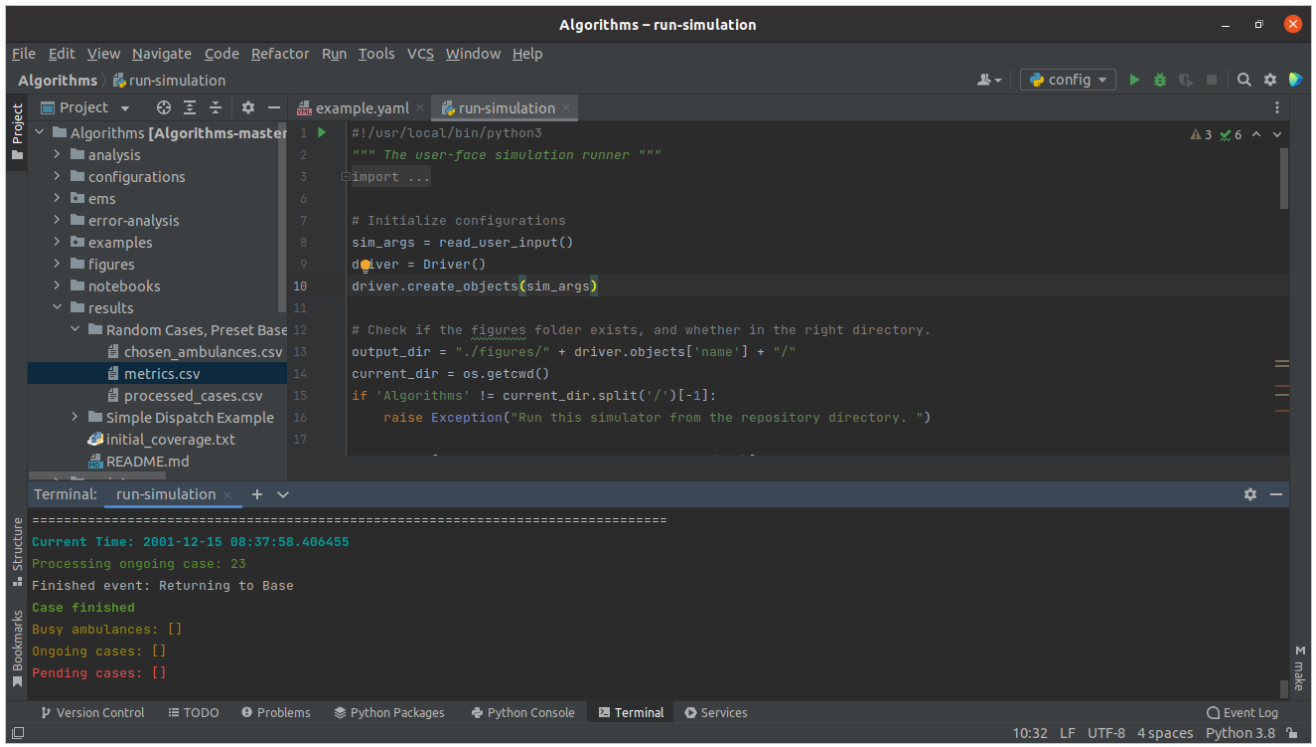


Figure 56: Simulator module.

Updated simulator module

Generating all processes and events as in the original simulator. The video shows the process of the simulator, in its current version.

Watch video



5. Conclusion

Conclusion

The Simulator module is functional, but lacks the basic functions required by the Tijuana Red Cross. There is already a database with information from the emergency medical services to carry out the simulations.

6. Outlook

Outlook

As future work it is proposed:

- Implement an algorithm that evolves dispatch rules, which given the current status (coverage, ambulance location, incoming call priority, etc.) determine which ambulance should be dispatched.
- For travel times, the OSRM routing engine could be implemented since it does not vary much from Google Maps according to previous experiments, in addition to having the advantage of being free.
- Change the policies for the selection of the hospital to which the ambulance must be transferred.

Thanks for your attention!!

noelia.torres@tectijuana.edu.mx

yaz.maldonado@tectijuana.edu.mx

leonardo.trujillo@tectijuana.edu.mx

TECNOLÓGICO NACIONAL DE MÉXICO

TIJUANA, BAJA CALIFORNIA

CONACYT