murdasp: a tool to simulate the pandemic using activity-based transport model

Moez Kilani
CNRS UMR 9221, Lille Economie Management – LEM
University of Littoral Opal Coast, Dunkerque
`moez.kilani@univ-littoral.fr`
2022-12-5 — Version 1

Abstract

This document describes `murdasp`, a tool developed to simulate a wave of the pandemic. It is a set of `bash` and `awk` scripts that collect the output of transport simulation model, produces social interactions and applies an epidemic model to assess the infections that occurs during the simulation period. This is a preliminary version of the documentation that should be updated in the near future. A basic information is provided.

Contents

1 Installation and quick start

1.1 Installation

To install the required scripts, clone the corresponding `git` directory:

```
git gogs --clone
```

1.2 Quick start

An event file with structure described below is required in the `data/` directory. Then, process the following commands (being at the root directory):

1. Generate the social interactions,

   ```
   $ sh/activities.sh
   ```

   This command produces, for each activity and transport mode, a distinct file tracing the interactions between the agents.

2. Edit the file `simulate.sh` to fill the needed values

3. Run the simulation

   ```
   $ simulate.sh
   ```

   This command will generate an initial cluster (found in directory `data`) run the simulations for the first day, and then run the simulation along the simulation horizon.

At the end of the simulation, the output reporting the infection that can be found in `output/days-xx/`, where `xx` corresponds to the label provided in `simulate.sh` script.

2 Overview and file structure

2.1 File structure

The file tree for `murdasp` has the following structure. The event file in the `data` directory is the input data used to identify the location of the agents during the day. except `simulate.sh`, which runs the main simulation, the `bash` scripts are in directory `sh`.

```
.
├── simuls.sh
├── data
│   ├── output_events.xml.gz
│   ├── xy_facilities.csv
│   ├── cluster-1.csv
│   ├── cluster-2.csv
│   ..
├── sh
│   ├── activities.sh
│   ├── day_update.sh
│   ├── xy_pandemic.sh
│   ├── init_cluster.sh
│   │
```

```
│    ├── infections.sh
│    └── simulate.sh
├── awk
│    ├── act_infect.awk
│    ├── day_end.awk
│    ├── interact.awk
│    ├── process.awk
│    ├── process_trsp.awk
│    ├── cluster.awk
│    ├── init_infect_cluster.awk
│    ├── infection.awk
│    └── stat.awk
└── output
     ├── population
     │    └── ids.csv
     ├── interactions
     │    ├── leisure.txt
     │    ├── rail.txt
     │    ├── ...
     │   ..
     ├── days-xx
     │    ├── ...
    ..   ..
```

2.2 The events' file

This is the file that reports all the events that have occurred during the simulation. A typical file is a large collection of `xml` tags.

The structure of the ''events.xml'' file

```xml
<event time="26018.0" type="actend"
  person="082016114007087_0903"
  facility="082016114007087_09"
  link="203352" actType="home" />
<event time="26018.0" type="departure"
  person="082016114007087_0903" link="203352"
  legMode="walk" />
<event time="26018.0" type="actend"
  person="082016101010042_0101"
  facility="082016101010042_01"
  link="274002"
  actType="home"  />
<event time="26018.0" type="PersonEntersVehicle"
   person="062006029008083_0101"
   vehicle="06200602900808" />
```

The event file produced for the simulations in the North of France has a total of 106 325 621 `xml` tags.

2.3 Collecting social interactions

In a preliminary step, the script `activities.sh` is used to collect interactions that it stores in files located in directory `output/interactions`. It uses `process.awk` and `process-trsp.awk` to process interactions in the activities (lines 16 to 27) and in the transport modes (lines 29 to 39), respectively.

The ''activities.sh'' script

```
1  #!/bin/zsh
2
3  activities=("leisure" "shop" "home" "work" "education" "primary");
4  transport=("bus" "subway" "tram" "rail");
5
6  file_inter="output/interactions";
7
8  if [[ -d "$file_inter" ]]
9  then
10   echo "$file_inter directory exists; exiting"
11   exit 0
12 fi
13
14 mkdir $file_inter
15
16 for act in $activities; do
17   echo "Processing activity $act";
18   file="output/interactions/"$act".txt";
19   zcat data/output_events.xml.gz | grep "actstart\|actend" | \
20     grep $act | \
21     mawk -F'"' '{for(i=1;i<=NF;++i) {if($i ~ "actType=") {I=i+1;}} ; \
22     print $2 ";" $4 ";" $6 ";" $8 ";" $I}' | \
23     sort -t ';' -k3,3 -k1,1 -k2,2r | mawk -f awk/process.awk | \
24     sort -t ";" -k4,4 -k1,1  | \
25     mawk -F";" 'BEGIN{id="";} $4 != id {print "===";} \
26     {print $1";"$2";"$5";"$4";"$3; id=$4;}' > $file
27 done
28
29 for trsp in $transport; do
30   echo "Processing transport mode $trsp";
31   file="$file_inter/$trsp.txt";
32   zcat data/output_events.xml.gz | grep "PersonEnters\|PersonLeaves" | \
33     grep $trsp | \
34     mawk -F'"' -v mode=$trsp '{print $2 ";" $4 ";" $6 ";" $8";" mode}'| \
35     sort -t ';' -k3,3 -k1,1 -k2,2r | mawk -f awk/process_trsp.awk | \
36     sort -t ";" -k4,4 -k1,1  | \
37     mawk -F";" 'BEGIN{id="";} $4 != id {print "===";} \
38     {print $1";"$2";"$5";"$4";"$3; id=$4;}'  > $file
39 done
```

This script should be run before any simulation. If it exits without errors, it adds a file for each activity and for each transport mode in the `output/interactions/` directory (the the tree structure above). Activities and transport modes in lines 3 and 4 should be consistent with the `event.xml` file. To diregard an activity or a transport mode, it can be remved from these lists.

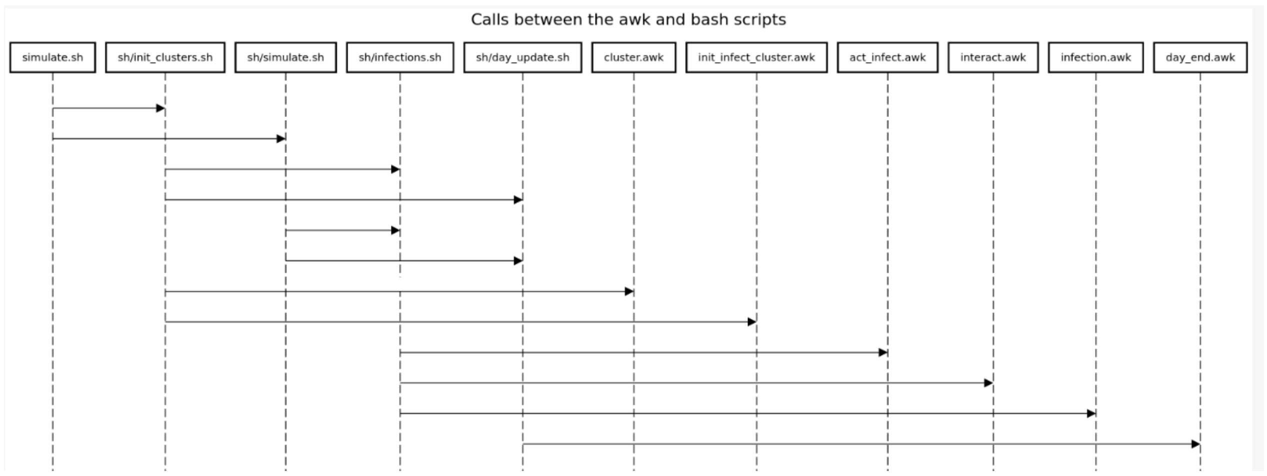

Figure 1: The main scripts and how they communicate. All the **awk** scripts are in the awk directory.

2.4 Running a simulation

The following commented file runs a complete simulation. Required data is first first provided (lines 2 to 6), then an initial cluster is created (line 8). The simulation for `n_iter` days is then run by the command that extends from line 12 to line 14.

The ''simulate.sh'' script

```
1   #!/bin/bash
2
3   n_max=20;   # number of infected agents in initial cluster
4   n_iter=150; # number of simulated days iterations)
5   n_dir=80;   # label of the output directory
6   x0=703744; y0=7059180; # coordinates where clusters appear
7
8   sh/init_cluster.sh $n_dir $x0 $y0 $n_max # create the initial cluster
9
10  # Run the simulation by providing parametres theta:
11  # work leisure shop education primary home bus tram rail subway
12  sh/simulate.sh $n_dir $n_iter \
13    .1 .3 1 0.08 0.009 0.15 2.1 1.2 0.9 1.75 \    # theta's
14     $x0 $y0 $n_max
```

Of course, several simulations can be inserted in the same script. The script `sh/init_cluster.sh` create an initial cluster for agent locations around x_0 and y_0 by considering a radius of 2 km. This value can be changed in the `awk` script `awk/init_cluster.awk`. A large value can be set to generate infections uniformly on the study region. The script `sh/simulate.sh` then simulates infections from day 1 to day `n_iter` (set to 120 or above to simulate the whole wave).

2.5 Organisation of scripts

The main simulation uses five `bash` scripts and six `awk` scripts. The sequence diagram in Figure 1 shows how they are called. The creation of the interaction uses a `bash` script (the file `sh/activities`) and two further `scripts` (respectively `awk/process.awk` and `awk/process_trsp`).

3 Computation time

Depending on the transport model, some steps are time consuming. It mainly depends on the number of agents and the number of facilities (the location where interaction occur). When a facility is patronized by n agent the number of interaction is of n^2 order. So, computation time can increase when a facility is used by many agents and this occurs when the number of facilities is small. Education, including primary schools is particularly impacted since, in general the number of schools is limited within an agglomeration.

An easy way to reduce computation time is to assume that not all agents interact in the facility by duplicating it.

For the case Study of the North of France, the simulation of a whole wave of the pandemic takes about ten hours. This is not excessive since several sceanrios can be runned in parallel (each simulation uses one core), but the parallelizations (and optimization) of some parts of the code could improve efficiency. This was not the main objective in the early development and could be considered in the future versions.