QuCAD User Manual

Version 2023

Yee Lam Elim Thompson

March 12, 2023

Contents

List of Code Listings

1 Introduction

The primary goal of QuCAD is to assist investigators with evaluating the wait-time-saving performance of Computed-Aided Triage and Notification (CADt) devices. This type of Software as a Medical Device (SaMD) is intended to triage patients' radiological medical images based on the binary outputs of the underlying Artificial Intelligence and Machine Learning (AI/ML) algorithms. Their main benefit is to speed up radiological review of images with time-critical findings, such as Large Vessel Occlusion (LVO), increasing the likelihood of a timely diagnosis and potentially a life-saving treatment.

To quantify how much time is saved due to the use of a CADt device, this python software simulates patient image flow to determine the waiting time distributions for different subgroups of patient images in both the presence and absence of the CADt device, from which the time-saving distributions can be generated. The software also includes a theoretical calculations of average waiting times based on queueing theory to verify simulation results.

Please note that the first version of this software is command-line-heavy. It is our intention to include more interactive, user-friendly Graphical User Interface (GUI) that also allows real-time processing in the future.

1.1 Credits

- Frank Samuelson - Present
  - Project Leader
- Yee Lam Elim Thompson - Present
  - Original creator
  - Consolidated on the original user manual
- Jixin (Audrey) Zheng - Present
  - Optimized runtime performance

2 Getting Started

2.1 Python environment

This software is written entirely in python, specifically version 3.9.4. The following additional packages are required to use this software

- numpy
- pandas
- scipy
- matplotlib
- statsmodels

A `requirements.txt` file is available with a list of all packages and their version numbers.

2.1.1 Cloning QuCAD from GitHub

In a terminal (Unix/macOS) or a PowerShell (Windows) console, run the following command line. Replace */path/to/your/working/directory/* to a work space you would like to work at. This will download the entire repository to your working directory.

```
$ cd /path/to/your/working/directory/
$ git clone https://github.com/DIDSR/QuCAD.git QuCAD
```

2.1.2 Setting up virtual environment

Users can follow the Python documentation on how to create virtual environments. In the following example, we assume that a virtual environment under the QuCAD repository with a path */path/to/your/working/directory/QuCAD/env/*.

Once a virtual environment is created, users can activate it in a terminal (Unix/macOS) or a PowerShell (Windows) via the following command line.

```
  $ source env/bin/activate ## Unix/macOS
OR
  $ .\env\Scripts\activate  ## Windows
```

To set up the environment for this software, one can use the `requirements.txt`.

```
  $ python3 -m pip install -r requirements.txt ## Unix/macOS
OR
  $ py -m pip install -r requirements.txt      ## Windows
```

Installing packages may take awhile. Once the installation is completed, user can follow the instructions in Section 2.2 to run simulations. Next time, when the user wants to run the simulation again, they can follow Section 2.1.3 to reactivate the virtual environment without re-installing all the required packages.

For Anaconda users, installation instructions will be provided in the next version. In the future, this software will also be packaged into a python module that can be installed via `pip install`.

2.1.3 Activating virtual environment

To re-activate the virtual environment that was created, users can open a terminal (Unix/macOS) or a PowerShell (Windows) and proceed to where the virtual environment was created.

```
  $ cd /path/to/your/working/directory/QuCAD/
  $ source env/bin/activate ## Unix/macOS
OR
  $  cd /path/to/your/working/directory/QuCAD/
  $ .\env\Scripts\activate  ## Windows
```

2.2 Running `run_sim.py`

`run_sim.py` is the main script to run the software. Users can specify all parameters via an input `config.dat` file or argument flags. For explanations of each parameters, see Section 3.1.

Listing 1: Example printouts when running `run_sim.py`

```
Reading user inputs:
+-----------------------------------------
| traffic: 0.8
| TPFThresh: 0.9
| FPFThresh: None
| prevalence: 0.1
| nRadiologists: 1
| fractionED: 0.0
| meanServiceTimes diseased: 10.0
| meanServiceTimes non-diseased: 10.0
| meanServiceTimes interrupting: 5.0
| rocFile: ../inputs/exampleROC.dat
| statsFile: ../outputs/stats/stats.p
| runtimeFile: ../outputs/stats/runTime.txt
| plotPath: ../outputs/plots/
| verbose: True
| doTrialOnly: False
| configFile: ../inputs/config.dat
| nTrials: 10
| nPatientsTarget: 1000
| runTimeFile: ../outputs/stats/runTime.txt
| doPlots: True
| doRunTime: True
+-----------------------------------------

+-----------------------------------------------
| prevalence setting : 0.100
| prevalence from sim: 0.101
|
| PPV from theory: 0.3431252
| PPV from sim   : 0.3389831
| NPV from theory: 0.9864444
| NPV from sim   : 0.9920635
+-----------------------------------------------
10 trials took 0.75 minutes
```

2.2.1 Via input `config.dat` file

To run the script using `inputs/config.dat`, use the following command line:

```
$ cd /path/to/your/working/directory/QuCAD/scripts/
$ python run_sim.py --configFile ../inputs/config.dat
```

By default, `config.dat` asks the software to print out information as it runs, and one may see the printouts as shown in Listing 1. Once the software tells you the runtime to simulate all the trials, the run is successful.

2.2.2 Via argument flags

An alternative way to run `run_sim.py` is to specify the parameters via argument flags. Flags in parenthesis are optional. To use these flags, please remove the parenthesis in the above example.

```
$ python run_sim.py --traffic 0.8 --TPFThresh 1.0 --prevalence 0.1
                    --nRadiologists 1 --fractionED 0.0
                    --meanServiceTimeDiseasedMin 10
                    --meanServiceTimeNonDiseasedMin 10
                    --meanServiceTimeInterruptingMin 5
                    --statsFile /path/to/outputs/stats/stats.p
                    --nTrials 10 --nPatientsTarget 1000
```

```
(--FPFThresh 0.1)
(--rocFile /path/to/inputs/exampleROC.dat)
(--runtimeFile /path/to/outputs/runTime.txt)
(--plotPath /path/to/outputs/plots/)
(--verbose)
```

If the run is successful, the same printouts in Listing 1 will be displayed.

2.3 Verbose messages

Besides Listing 1, users may see warning and error messages if unexpected input parameters are provided.

2.3.1 Warnings

When a user encounters the following warning messages, the simulation run will likely to continue. However, the warning messages may help users to spot any unintentional values for the input parameters.

2.3.1.1 Limitations on theoretical calculation

Our current theoretical calculation is limited to a setting with more than 2 radiologists (`nRadiologists` $>$ 2) ***and*** with the presence of interrupting cases (`fractionED` $>$ 0). When such a setting is encountered, a warning message is shown, but the simulation will go on.

```
1  WARN: There are more than 2 radiologits with presence of interrupting images.
2         Theoretical values for AI negative and diseased/non-diseased subgroups will not be available.
3  WARN: Cannot calculate theoretical values for AI negative subgroup when more than 2 radiologists
4        with presence of interrupting patient cases.
```

2.3.1.2 Inputting both `rocFile` and `FPFThresh`

When both input parameters are provided, the software will use `FPFThresh` to determine the CADt operating threshold for the simulation run. That is, the `rocFile` input value will be ignored, and a user will see the following warning message.

```
1  WARN: Received both FPFThresh and rocFile. Taking FPFThresh and ignoring rocFile.
```

2.3.2 Errors

Unlike warning messages, error messages will immediately kill the execution.

2.3.2.1 Invalid `traffic`

By design, the traffic intensity must be between 0 and 0.95. Values larger than 0.95 represents a unrealistically congested queueing system in which patient images are stuck because images are coming in much faster than the radiologists can handled. If a value larger than 0.95 is provided, one will get the following Error message.

```
ERROR: Input traffic 1.000 is too high.
OSError: Please limit traffic below 0.95.
```

2.3.2.2 Invalid values for parameters ranging between 0 and 1

By definition, `TPFThresh`, `FPFThresh` (if provided), `prevalence`, and `fractionED` must be between 0 and 1. If a value outside of that range is provided, one will get the following Error message.

```
ERROR: Input prevalence 2.000 is too high.
OSError: Please provide a prevalence between 0 and 1.
```

2.3.2.3 Invalid paths and input files

If the user provides a value for `rocFile` and `configFile`, and the file does not exist, an error will be thrown. Similarly, for `statsFile`, `runtimeFile`, and `plotPath`, the existence of these paths are checked before running any simulations. If the paths do not exist, the software will try to create one. If failed, the user will see an error message.

```
ERROR: Path does not exist: /does/not/exist/
ERROR: Trying to create the folder ...
OSError: Cannot create the folder.
          Please provide a valid {0} path.
```

2.3.2.4 Neither `rocFile` nor `FPFThresh`

To simulate image patient flow with the use of CADt device, either one of the two input parameters is required to simulate CADt diagnostic performance. If neither of them are provided, the following error message will be displayed.

```
ERROR: Neither FPFThresh nor rocFile is provided.
OSError: Please provide either FPFThresh or rocFile.
```

2.4 Expected output files

Depending on the user input parameters, different output files will be generated. If run as default, one can expect the following output files. For descriptions of each files, see Section 3.2.

- `outputs/stats/stats.p`
- `outputs/stats/runTime.txt`

Listing 2: Parameters for clinical settings in `inputs/config.dat`

```
##############################################
## Clinical setting
##############################################
traffic                         0.8 # Hospital busyness:
                                    # Between 0.0 (very quiet) and 0.95 (very congested)
fractionED                      0.0 # Fraction of interrupting patients:
                                    # Between 0.0 (no interrupting images) and
                                    # 1.0 (all images are interrupting)
nRadiologists                   1   # Number of radiologists reading during the time window:
                                    # At least 1
meanServiceTimeDiseasedMin      10  # Average radiologist's service time [in minutes]
                                    # for cases s/he claims diseased (based on diagnosis)
meanServiceTimeNonDiseasedMin   10  # Average radiologist's service time [in minutes] for
                                    # cases s/he claims non-diseased (based on diagnosis)
meanServiceTimeInterruptingMin  5   # Average radiologist's service time [in minutes]
                                    # for interrupting cases
```

- `outputs/plots/n_patients_system_distribution_with_CADt.pdf`
- `outputs/plots/n_patients_system_distribution_without_CADt.pdf`
- `outputs/plots/patient_timings_withCADt.pdf`
- `outputs/plots/patient_timings_withoutCADt.pdf`
- `outputs/plots/ROC_anAI.pdf` (only if `rocFile` is provided)
- `outputs/plots/waiting_time_distribution_AIcall.pdf`
- `outputs/plots/waiting_time_distribution_radDiagnosis.pdf`

3 Inputs and output

3.1 Input parameters

As mentioned in Section 2.2, users have two ways to run this software: 1. via argument flags and 2. via an `inputs/config.dat` file with all the parameters. Either way, the users specify a set of input parameters. In the following sections, input parameters are categorized into clinical, disease condition specific, CADt AI diagnostic performance, and simulation parameters.

3.1.1 Clinical settings

Clinical parameters are listed in Listing 2, and a brief summary is presented in Table 3.1.1.

Parameter	Argument Flag	Description
traffic	--traffic 0.8	Hospital busyness Default: 0.8 Range: [0 - 0.95]
fractionED	--fractionED 0.0	Fraction of interrupting patients in system Default: 0.0 Range: [0 - 1]
nRadiologists	--nRadiologists 1	Number of radiologists to review images Default: 1 Range: Integer $\geq$ 1
meanServiceTimeDiseasedMin	--meanServiceTimeDiseasedMin 10	Mean service time [min] for diseased images Default: 10.0 Range: Positive float
meanServiceTimeNonDiseasedMin	--meanServiceTimeNonDiseasedMin 10	Mean service time [min] for non-diseased images Default: 10.0 Range: Positive float
meanServiceTimeInterruptingMin	--meanServiceTimeInterruptingMin 5	Mean service time [min] for interrupting images Default: 5.0 Range: Positive float

3.1.1.1 traffic

Traffic describe how busy a radiology department is. In queueing theory, a traffic intensity is defined by the ratio between patient image arrival rate and radiologist's service rate. For a quiet clinic with no images coming in, `traffic` is 0. On the other hand, `traffic` is 1 for a very congested clinic where how fast images enters into the system is the same as how fast a radiologist services an image. Therefore, by definition, `traffic` ranges from 0 to 1; `traffic` greater than 1 implies an infinitely growing queue because images arrive faster than they can leave. For simulation purposes, `traffic` must be smaller than 0.95.

3.1.1.2 fractionED

This parameter refers to the fraction of interrupting patient images with respect to all patient images that enter into the system. Typically, a radiologist has a list of images to read, and if a CADt device is used, the device will read all images in the list. Interrupting images represent images that require a radiologist's immediate attention. These images are outside the reading list and are not reviewed by the CADt device. By definition, `fractionED` must be between 0 and 1. To simulate the scenario where simulated radiologists will never be interrupted to review an image outside of their reading list, set `fractionED` to 0. One could also set `fractionED` to 1 for an unrealistic situation where all images reviewed by the radiologists are not from the reading list.

3.1.1.3 nRadiologists

`nRadiologists` is the number of radiologists reviewing images in the same reading list. Typically, a clinic has at least one radiologist at all times. For a larger hospital, multiple radiologists may be available during the day. The number of radiologists may also depend on the time of the day and day of the week. For simulation, one could set a very large number of radiologists. However, the theoretical computation of average wait-time-savings for certain subgroups of patient images may not be available if `nRadiologists` is equal to or greater than three.

Listing 3: Parameters for disease-related settings in `inputs/config.dat`

```
##############################################
## Disease condition specific parameters
##############################################
prevalence 0.1 # Disease prevalence among cases in the radiologist reading list
               # Between 0.0 (radiologist calls all cases non-diseased) and 1.0
```

3.1.1.4 meanServiceTimeDiseasedMin

This parameter is the average service time in minutes by a radiologist for cases that s/he claims to be diseased. In queueing theory, the service time distribution is expected to follow an exponential distribution characterized by a service rate. When a radiologist calls an image diseased, it may take them longer (on average) to process the case e.g. highlighting regions, calling other staff, etc. This would imply a lower service rate for the diseased image subgroup compared to that for the non-diseased image subgroup.

3.1.1.5 meanServiceTimeNonDiseasedMin

`meanServiceTimeNonDiseasedMin` is the average service time in minutes by a radiologist for cases that s/he claims to be non-diseased. Similar to the mean service time for diseased subgroup, when a radiologist calls an image non-diseased, it may take them faster (on average) to close the case. This would imply a higher service rate for the diseased image subgroup.

3.1.1.6 meanServiceTimeInterruptingMin

This is the average service time in minutes by a radiologist for an interrupting case. An interrupting case is assumed to be very urgent and time-critical that a radiologist should review it relatively quickly. Therefore, these cases may have an even higher service rate.

3.1.2 Disease related parameters

While QuCAD software is not tied to a specific disease condition, the amount of time-savings due to the CADt device does depend on the disease prevalence.

Parameter	Argument Flag	Description
`prevalence`	`--prevalence 0.1`	Disease prevalence in reading list Default: 0.1 Range: [0 - 1]

3.1.2.1 prevalence

Disease prevalence is defined to be the ratio of the number of diseased patient images (called by the radiologist) in the reading queue to the number of all patient images in the reading queue. It should be noted that both the numerator and denominator are with respect to images in the reading queue. Since interrupting patient images are not in the reading queue, the `prevalence` defined here does not consider the radiologist's diagnoses of the interrupting subgroup. This definition is intentional because

Listing 4: Parameters related to the CADt device in `inputs/config.dat`

```
#############################################
## CADt AI diagnostic performance
#############################################
TPFThresh 0.9                       # CADt AI sensitivity
FPFThresh None                      # CADt AI (1-specificity)
                                    # If using an `rocFile`, set this to "None"
rocFile   ../inputs/exampleROC.dat  # CADt ROC curve:
                                    #  First column = False-positive fraction
                                    #  Second column = True-positive fraction
                                    # Note: a Bi-normal distribution is assumed for parameterization
```

our goal is to determine the time-saving for images processed by the CADt device. Interrupting images that are outside of the reading queue and (thus) are not reviewed by the CADt device will not experience any time-saving. By definition, `prevalence` is between 0 and 1. A `prevalence` of 0 means that the radiologist calls all images diseased, whereas 1 implies that all images are labelled as non-diseased by the radiologist.

3.1.3 CADt AI related parameters

Parameters related to the diagnostic performance of a CADt device is listed in Listing 4 and Table 3.1.3.

Parameter	Argument Flag	Description
`TPFThresh`	`--TPFThresh 0.9`	True-positive fraction (TPF) i.e. Sensitivity of the operating point Default: 0.9 Range: [0 - 1]
`FPFThresh`	`--TPFThresh 0.1`	False-positive fraction (FPF) i.e. 1 - Specificity of the operating point If None, `rocFile` must be provided. Default: None Range: [0 - 1]
`rocFile`	`--rocFile ../inputs/exampleROC.dat`	A text file with two columns (first: TPF, second: FPF) If None, `FPFThresh` must be provided Default: ../inputs/exampleROC.dat

3.1.3.1 `TPFThresh`

The true-positive fraction (TPF) threshold operating point, also known as sensitivity, is defined to be the ratio of the number of AI-positive patient images to the number of diseased patient images called by the radiologist. Similar to `prevalence`, this definition of sensitivity only consider images in the reading queue (that would be read by CADt in the with-CADt scenario). Interrupting patient images are not included in the calculation of sensitivity. By definition, `TPFThresh` is between 0 and 1. A `TPFThresh` of 0 means that the CADt calls all diseased images negative, whereas 1 implies that all diseased images are correctly labelled as positive by the CADt device.

3.1.3.2 `FPFThresh`

The false-positive fraction (FPF) threshold operating point, also known as 1 - specificity, is defined to be the ratio of the number of AI-positive patient images to the number of non-diseased patient images called by the radiologist. Again, this definition of sensitivity excludes interrupting patient images. If

Listing 5: The first 10 lines of `inputs/exampleROC.dat`

```
0.0000,0.0000
0.0002,0.2500
0.0043,0.5000
0.0441,0.7500
0.1912,0.9000
0.1955,0.9020
0.2000,0.9040
0.2047,0.9061
0.2094,0.9081
0.2143,0.9101
```

`FPFThresh` is provided, its value must be between 0 and 1. A `FPFThresh` of 0 means that the CADt correctly calls all non-diseased images negative, whereas 1 implies that all non-diseased images are mistakenly labelled as positive by the CADt device. If `FPFThresh` is not provided, the software will expects a valid `rocFile`.

3.1.3.3 `rocFile`

`rocFile` is required if `FPFThresh` is set to None. This parameter states the path and filename of an existing text file that has two columns (with no headers) as shown in Listing 5. The first column is the false-positive fraction, and the second is the true-positive fraction. As described in Section 4.1.1, these values are used to reconstruct the diseased and non-diseased distributions assuming a bi-normal distribution. It should be noted that, if a user has an empirical Receiver-Operating Characteristic curve (ROC curve), it may be better to run the software at different {TPF, FPF} using `TPFThresh` instead of using the assumption of an underlying bi-normal distribution.

3.1.4 Simulation parameters

The parameters in Listing 6 and Table 3.1.4.7 are related to simulation runs.

3.1.4.1 `nTrials`

`nTrials` specifies the number of trials to be generated.

3.1.4.2 `nPatientsTarget`

This parameter specifies the targeted number of patient images to be generated per trial. This value, along with the input `traffic`, is used to estimate the number of days required to reach the target number. As a result, the actual number of simulated patient images may be different.

3.1.4.3 `verbose`

When testing and debugging, it is recommended to turn on `verbose`. As for now, printout messages include repeating user inputs and checking whether simulated runs have similar values for disease prevalence, positive predictive value, and negative predictive values. The time it took to run all trials are also printed.

Listing 6: Parameters related to a simulation run in `inputs/config.dat`

```
##############################################
## Simulation setting
##############################################
nTrials         10                          # Number of trials to perform
nPatientsTarget 1000                        # Number of targeted patient images
verbose         True                        # False to turn off all printouts
doTrialOnly     False                       # If False, also run 1 simulation run for debugging
                                            # Otherwise (esp for job submission), put it to False
statsFile       ../outputs/stats/stats.p    # Pickled file with the parameters and simulation results
runTimeFile     ../outputs/stats/runTime.txt # Text file showing runtime performance
                                            # If None, do not output the runtime performance
plotPath        ../outputs/plots/           # Folder where all output plots are stored
                                            # If None, no plots are generated
```

3.1.4.4 `doTrialOnly`

By default, this parameter is set to False, and `run_sim.py` runs a simulation run before running trials. This is good for debugging. However, when submitting jobs, it is recommended to turn it on.

3.1.4.5 `statsFile`

`statsFile` contains both the file path and name that stores the output results from the simulation runs as well as other theoretical values. Please visit Section 3.2.1 for more information on what information is stored and how to access them.

3.1.4.6 `runTimeFile`

If provided a file path and name, a text file summarizing the runtime performance is dumped. This contains how many times each function is called, how long it takes, etc.

3.1.4.7 `plotPath`

If provided a file path, plots associated to the simulation run are generated and stored in the provided path. See Section 3.2.2 for the descriptions of output plots.

Parameter	Argument Flag	Description
`nTrials`	`--nTrials 10`	Number of trials to be simulated Default: 10 Range: ≥ 1
`nPatientsTarget`	`--nPatientsTarget 1000`	Number of targeted patient images Default: 1000 Range: ≥ 1
`verbose`	`--verbose`	If True, print out information as the simulation runs. Default: False
`doTrialOnly`	`--doTrialOnly`	If True, do not run a local simulation runs (the output of which will not be stored). This may be useful when submitting jobs. Default: False
`statsFile`	`--statsFile ../outputs/stats/stats.p`	If provided a filename, simulation and theoretical prediction will be stored in a pickled file. See Section 3.2.1 for more information. When doing job submission, it is recommended to have a unique output filename to avoid replacement. Default: ../outputs/stats/stats.p
`runTimeFile`	`--runTimeFile ../outputs/stats/runTime.txt`	If provided, dump out a runtime performance summary. Default: ../outputs/stats/runTime.txt
`plotPath`	`--plotPath ../outputs/plots/`	If provided, generate plots. Default: ../outputs/plots/

3.2 Output

This section explains all the output files that will be generated if the corresponding input parameters are turned on and/or provided. The exact outputs depend on the input user settings. However, to provide examples, the snapshots in the Sections 3.2.1 through 3.2.3 are generated using the command line in Listing 7.

3.2.1 Pickled python dictionary

The most important file is probably the pickled dictionary with a default name of `stats.p`. Listing 8 shows a code snippet that opens the pickled file and displays some information.

3.2.1.1 `params`

The `params` key contains both user input parameters and theoretical predictions which includes probabilities of a patient being in a specific subgroup and the waiting time and time-savings. Specifically, *adict['params']['theory']* contains the theoretical values (in minutes) of average waiting times for different subgroups in both with and without CADt scenarios as well as the average time difference between the two scenarios. As shown in Listing 9 (Lines 1-28), diseased patient images are benefited from the CADt device with a predicted mean time-saving of 15.72 minutes, whereas non-diseased images are on averaged delayed by 1.7 minutes. Lines 30-73 shows the example codes to access the corresponding simulation results.

3.2.1.2 `wpatients`

If one would like to access the individual simulated patient, one can use the `wpatients` key, as shown in Listing 10. The dataframe where each row is a simulated patient. Each simulated patient has a unique ID which is based on the ordering of its arrival time. The first patient image arrived in the system has a *patient_id* of '0000000', and the next one is '0000001', etc. The column *trial_id* indicates

Listing 7: Running via argument flags for the outputs in the following sub-sections

```
> python run_sim.py --traffic 0.8 --TPFThresh 0.9 --prevalence 0.1 --FPFThresh 0.1
                    --nRadiologists 2 --fractionED 0.1 --statsFile ../outputs/stats/stats.p
                    --nTrials 1 --nPatientsTarget 20000  --runtimeFile ../outputs/stats/runTime.txt
                    --plotPath ../outputs/plots/ --verbose

Reading user inputs:
+----------------------------------------
| traffic: 0.8
| TPFThresh: 0.9
| FPFThresh: 0.1
| prevalence: 0.1
| nRadiologists: 2
| fractionED: 0.1
| meanServiceTimes diseased: 10
| meanServiceTimes non-diseased: 10
| meanServiceTimes interrupting: 5
| rocFile: None
| statsFile: ../outputs/stats/stats.p
| runtimeFile: ../outputs/stats/runTime.txt
| plotPath: ../outputs/plots/
| verbose: True
| doTrialOnly: False
| configFile: None
| nTrials: 1
| nPatientsTarget: 20000
| doPlots: True
| doRunTime: True
+----------------------------------------

+-----------------------------------------------
| prevalence setting : 0.100
| prevalence from sim: 0.101
|
| PPV from theory: 0.5000000
| PPV from sim   : 0.4957983
| NPV from theory: 0.9878049
| NPV from sim   : 0.9878123
+-----------------------------------------------
1 trials took 2.12 minutes
```

Listing 8: Ipython example code to open the pickled python dictionary

```
In [1]: import pickle

In [2]: with open ('../outputs/stats/stats.p', 'rb') as f:
   ...:     adict = pickle.load (f)
   ...: f.close()

In [3]: adict.keys()
Out[3]: dict_keys(['params', 'lpatients', 'wpatients', 'lstats', 'wtstats'])
```

Listing 9: Ipython example code to retrieve theoretical and simulated related to waiting time

```
In [4]: for subgroup, predictions in adict['params']['theory'].items():
   ...:     if subgroup in ['diseased', 'non-diseased', 'positive', 'negative']:
   ...:         print ('+-------------------------------------')
   ...:         print ('| {0}'.format (subgroup))
   ...:         for key, value in predictions.items():
   ...:             print ('|   - {0}: {1:.2f}'.format (key, value))
   ...: print ('+-------------------------------------')
+-------------------------------------
| diseased
|   - waitTimeWithoutCADt: 18.29
|   - waitTimeWithCADt: 2.57
|   - waitTimeSaving: -15.72
+-------------------------------------
| non-diseased
|   - waitTimeWithoutCADt: 18.29
|   - waitTimeWithCADt: 19.98
|   - waitTimeSaving: 1.70
+-------------------------------------
| positive
|   - waitTimeWithoutCADt: 18.29
|   - waitTimeWithCADt: 0.39
|   - waitTimeSaving: -17.90
+-------------------------------------
| negative
|   - waitTimeWithoutCADt: 18.29
|   - waitTimeWithCADt: 22.16
|   - waitTimeSaving: 3.87
+-------------------------------------

In [5]: for subgroup in ['diseased', 'non-diseased', 'positive', 'negative']:
   ...:     print ('+-------------------------------------')
   ...:     print ('| {0}'.format (subgroup))
   ...:
   ...:     ## waiting time without CADt from simulation
   ...:     simStatsWaitTimeWithCADt =  adict['wtstats']['preresume']['waitTime'][subgroup]
   ...:     mean = simStatsWaitTimeWithCADt['mean']
   ...:     cl95 = [simStatsWaitTimeWithCADt['lower_95cl'], simStatsWaitTimeWithCADt['upper_95cl']]
   ...:     print ('|   - waitTimeWithoutCADt: {0:.2f} ({1:.2f} - {2:.2f})'.format (mean, cl95[0], cl95[1]))
   ...:
   ...:     ## waiting time with CADt from simulation
   ...:     simStatsWaitTimeWithoutCADt =  adict['wtstats']['fifo']['waitTime'][subgroup]
   ...:     mean = simStatsWaitTimeWithoutCADt['mean']
   ...:     cl95 = [simStatsWaitTimeWithoutCADt['lower_95cl'], simStatsWaitTimeWithoutCADt['upper_95cl']]
   ...:     print ('|   - waitTimeWithoutCADt: {0:.2f} ({1:.2f} - {2:.2f})'.format (mean, cl95[0], cl95[1]))
   ...:
   ...:     ## time difference between with and without CADt from simulation
   ...:     simStatsDeltaTime =  adict['wtstats']['preresume']['diff'][subgroup]
   ...:     mean = simStatsDeltaTime['mean']
   ...:     cl95 = [simStatsDeltaTime['lower_95cl'], simStatsDeltaTime['upper_95cl']]
   ...:     print ('|   - waitTimeSaving: {0:.2f} ({1:.2f} - {2:.2f})'.format (mean, cl95[0], cl95[1]))
   ...: print ('+-------------------------------------')
+-------------------------------------
| diseased
|   - waitTimeWithoutCADt: 2.42 (0.00 - 31.14)
|   - waitTimeWithoutCADt: 17.77 (0.00 - 81.91)
|   - waitTimeSaving: -15.36 (-79.80 - 3.64)
+-------------------------------------
| non-diseased
|   - waitTimeWithoutCADt: 18.70 (0.00 - 100.12)
|   - waitTimeWithoutCADt: 16.94 (0.00 - 81.68)
|   - waitTimeSaving: 1.76 (-24.93 - 30.03)
+-------------------------------------
| positive
|   - waitTimeWithoutCADt: 0.34 (0.00 - 4.67)
|   - waitTimeWithoutCADt: 17.50 (0.00 - 81.33)
|   - waitTimeSaving: -17.16 (-81.28 - 0.00)
+-------------------------------------
| negative
|   - waitTimeWithoutCADt: 20.79 (0.00 - 102.55)
|   - waitTimeWithoutCADt: 16.92 (0.00 - 81.81)
|   - waitTimeSaving: 3.87 (-4.76 - 31.67)
+-------------------------------------
```

Listing 10: Ipython example code to retrieve individual simulated patient

```
In [6]: adict['wpatients']
Out[6]:
            fifo  is_interrupting is_diseased  is_positive  preresume   trial_id patient_id     delta
0       0.000000            False       False        False   0.000000  trial_000    0000000  0.000000
1       0.000000             True        None         None   0.000000  trial_000    0000001  0.000000
2       0.000000             True        None         None   0.000000  trial_000    0000002  0.000000
3       0.000000             True        None         None   0.000000  trial_000    0000003  0.000000
4       0.790644            False       False        False   4.200786  trial_000    0000004  3.410142
...          ...              ...         ...          ...        ...        ...        ...       ...
19921   0.000000            False       False        False   0.000000  trial_000    0019921  0.000000
19922   0.000000            False       False        False   0.000000  trial_000    0019922  0.000000
19923   0.000000            False       False        False   0.000000  trial_000    0019923  0.000000
19924   2.693036            False       False        False   2.693036  trial_000    0019924  0.000000
19925   0.000000            False       False        False   0.000000  trial_000    0019925  0.000000

[19926 rows x 8 columns]
```

which trial this patient is in. For this example, since only one trial was requested (`nTrials` = 1), all simulated patients has the same *trial_id*. The patient's status is also recorded, including whether it is an interrupting case, whether the radiologist calls it diseased (if it is a non-interrupting case), and whether the AI calls it positive (also for non-interrupting case only). It should be noted that, since CADt devices are only beneficial to non-interrupting cases, we do not assign diseased/non-diseased and positive/negative status to interrupting cases. Last, the absolute waiting time (in minutes) when the patient is placed in a with-CADt scenario is stored in the *fifo* column. The *preresume* column has the waiting time in a without-CADt scenario, and the per-patient time difference between the two scenarios are given in the *delta* column.

3.2.2 Plots

No plots will be generated if the user does not specify a valid `plotPath`. Depending on `doTrialOnly` and `rocFile`, some of following plots may not be generated.

3.2.2.1 Patient Timings

If `doTrialOnly` is turned off, two plots related to patient image flows are generated. By default, timestamps of the first 200 patients are displayed in these two plots, and Figure 1 shows the timing information of a small subset of patients in without-CADt (left) and with-CADt (right) scenario. These patients are identical in terms of patient status and arrival times. However, the number and total length of the wait-time period (blue sections) may be different between the two scenarios, demonstrating the per-patient effects due to the use of a CADt device.

3.2.2.2 ROC of the AI

If `rocFile` is not provided (which is the case in our example), the *ROC_anAI.pdf* will not be generated. However, for demonstrating purposes, Figure 2 shows the output plot if `inputs/exampleROC.dat` were used. As mentioned in Section 3.1.3.3, a bi-normal assumption is used during the parameterization. Therefore, whenever `rocFile` is provided, it is important to check the goodness of fit (R^2 in the middle plot of Figure 2) and the overall agreement between the input data points and the fitted ROC curve.

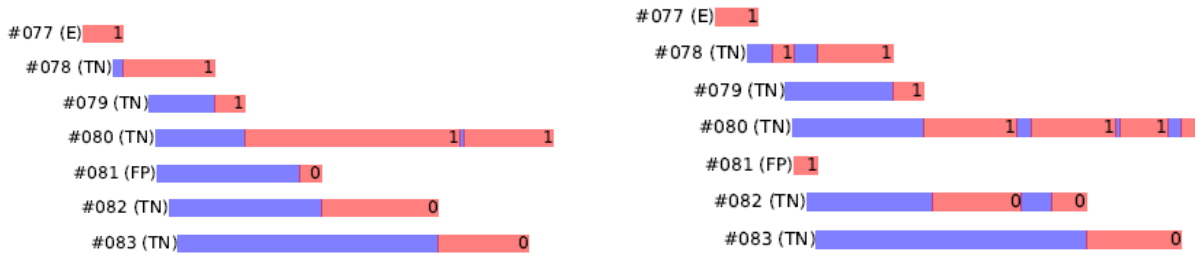

Figure 1: Patient image flows in without-CADt scenario (left) and in with-CADt scenario (right). Each bar represents the timestamps of a patient with an ID (e.g. "#077") followed by its status ("E" for interrupting, "TN" for non-diseased and AI negative, "FP" for non-diseased and AI positive, "TP" for diseased and AI positive, "FN" for diseased and AI negative). The length of each blue section represents the wait-time period experienced by the patient, whereas the length of each red section is the time period in which the patient image is being reviewed by a radiologist). Every red section has a number (0 or 1) that represents the radiologist ID that is handling that patient image. For this example, the user specifies two radiologists. Hence, the first radiologist is named 0, and the second is 1.

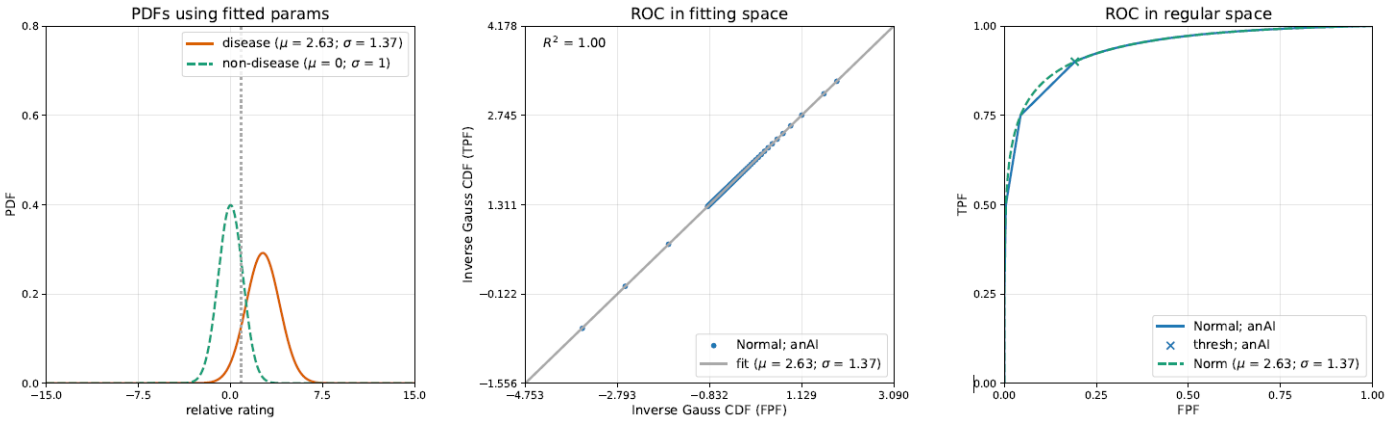

Figure 2: (Left) The relative rating probability distribution function for diseased (orange, solid) and non-diseased (green, dashed). Non-diseased histogram always has a mean μ of 0 and standard deviation σ of 1. Vertical gray dotted line indicates the rating threshold of the corresponding user-input `TPFThresh`. (Middle) The fitted ROC curve in an inverse normal cumulative distribution function space. Blue dots are the converted values from user's input ROC file. Best fit mean μ and standard deviation σ for the diseased rating histogram are stated in the legend, along with the R^2 value as a rough estimate of the goodness of fit. (Right) The same as the middle plot except that it is in a typical ROC space. The green dashed line represent the fitted ROC curve assuming a bi-normal distribution, and the gray cross is the operating point to be used in simulation.

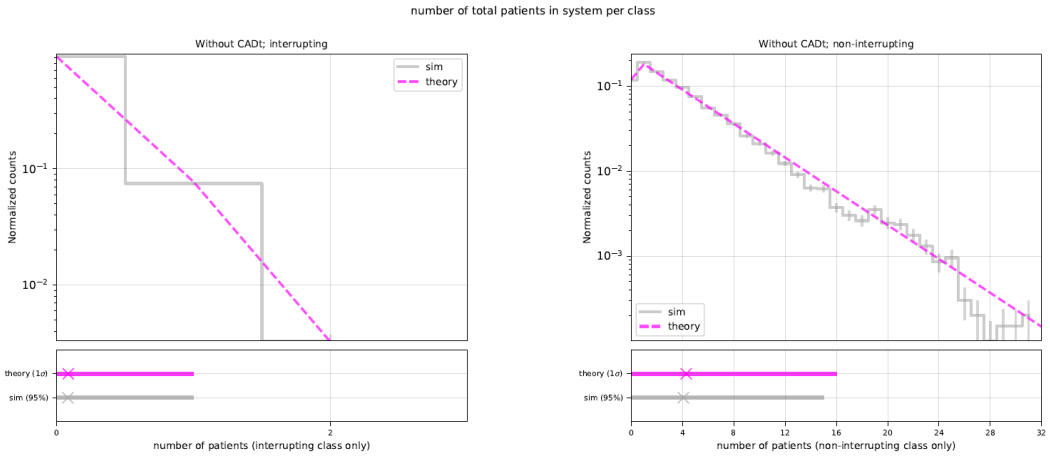

Figure 3: (Left) The distribution of the number of interrupting patients (higher priority) in system right before a new arrival normalized to unit area in a without-CADt scenario. (Right) The distribution of the number of non-interrupting (lower priority) patients in system right before a new arrival normalized to unit area in a without-CADt scenario. This distribution is the same as the state probability curve in the theoretical approach. The gray lines and bars are obtained from simulation, and the magenta lines and bars are theoretical predictions. Top plots show the distributions themselves, and the bottom plots show the 95% confidence limits from both theory and simulation.

3.2.2.3 Number of patients Distributions

If a valid `plotPath` is provided, one is expected to see two PDF files with normalized distributions (to unit area) of the number of patient images in the system for each subgroup observed by every in-coming patient image. It is important to note that these distributions are different than the total number of patient images. Rather, right before a new simulated patient enters the system, we record how many interrupting and non-interrupting patients are currently in the system in the without-CADt scenario (see Figure 3). Similarly, Figure 4) shows the normalized distributions of the number of interrupting, AI-positive, and AI-negative images right before a new patient image arrives in the with-CADt scenario. These normalized distributions are, by definition, the state probabilities of the queueing system that are needed to theoretically calculate the average waiting time for each priority class. Therefore, agreement between theory (magenta lines and bars) and simulation (gray lines and bars) are expected.

3.2.2.4 Distributions related to waiting time

In addition to the distributions of number of patient images, one is also expected to see two PDF files with distributions of waiting time for different subgroups as well as their time difference between with and without CADt scenarios. Figure 5 shows the waiting time and per-patient time difference distributions for interrupting, non-interrupting (in without-CADt scenario), AI-positive (in with-CADt scenario), and AI-negative (in with-CADt scenario). Figure 5 shows the distributions by interrupting, non-interrupting, diseased, and non-diseased subgroups. It is noted that, while one can obtain the waiting time distributions via Monte Carlo, theoretical prediction of distributions are difficult due to the complexity of these queueing models. However, their mean waiting times and per-patient time differences can be calculated, and one should see agreement between theoretical means (magenta pluses) and simulation means (gray crosses).

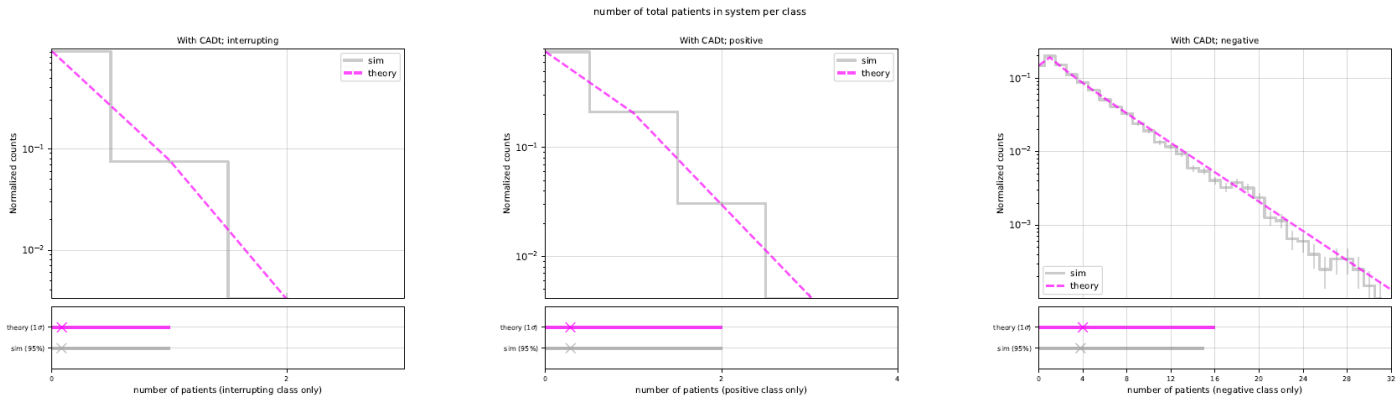

Figure 4: (Left) The distribution of the number of interrupting patients (highest priority) in system right before a new arrival normalized to unit area in a with-CADt scenario.(Middle) The distribution of the number of AI-positive patients (middle priority) in system right before a new arrival normalized to unit area in a with-CADt scenario. (Right) The distribution of the number of AI-negative (lowest priority) patients in system right before a new arrival normalized to unit area in a with-CADt scenario. This distribution is the same as the state probability curve in the theoretical approach. The gray lines and bars are obtained from simulation, and the magenta lines and bars are theoretical predictions. Top plots show the distributions themselves, and the bottom plots show the 95% confidence limits from both theory and simulation.

Listing 11: Run time performance text file

```
         452587860 function calls (447840218 primitive calls) in 286.312 seconds

   Ordered by: cumulative time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
   100858   25.446    0.000  134.865    0.001 {method 'remove' of 'list' objects}
        1    0.000    0.000  128.398  128.398 .\tools\trialGenerator.py:499(simulate_trials)
        1    0.688    0.688  127.864  127.864 .\tools\simulator.py:1493(simulate_queue)
101787264   57.295    0.000  109.466    0.000 .\tools\patient.py:94(__eq__)
    39858    0.371    0.000   65.663    0.002 .\tools\simulator.py:1148(_read_newest_urgent_patient)
204549942   52.485    0.000   52.485    0.000 .\tools\patient.py:142(caseID)
```

3.2.3 Runtime statistics

Last but not least, if a valid `runTimeFile` is provided, a summary text file *runTime.txt* is stored stating the number of times each function is called, as well as their runtimes (see Listing 11). This is especially useful for larger simulation runs. However, if running the script in a cluster, it is recommended to turn this functionality off.

4 Methods

4.1 Simulations

A majority of scripts in this tool are related to simulations. A simulation run is performed in the `simulator` class. However, other instances play their roles throughout the simulated patient flow.

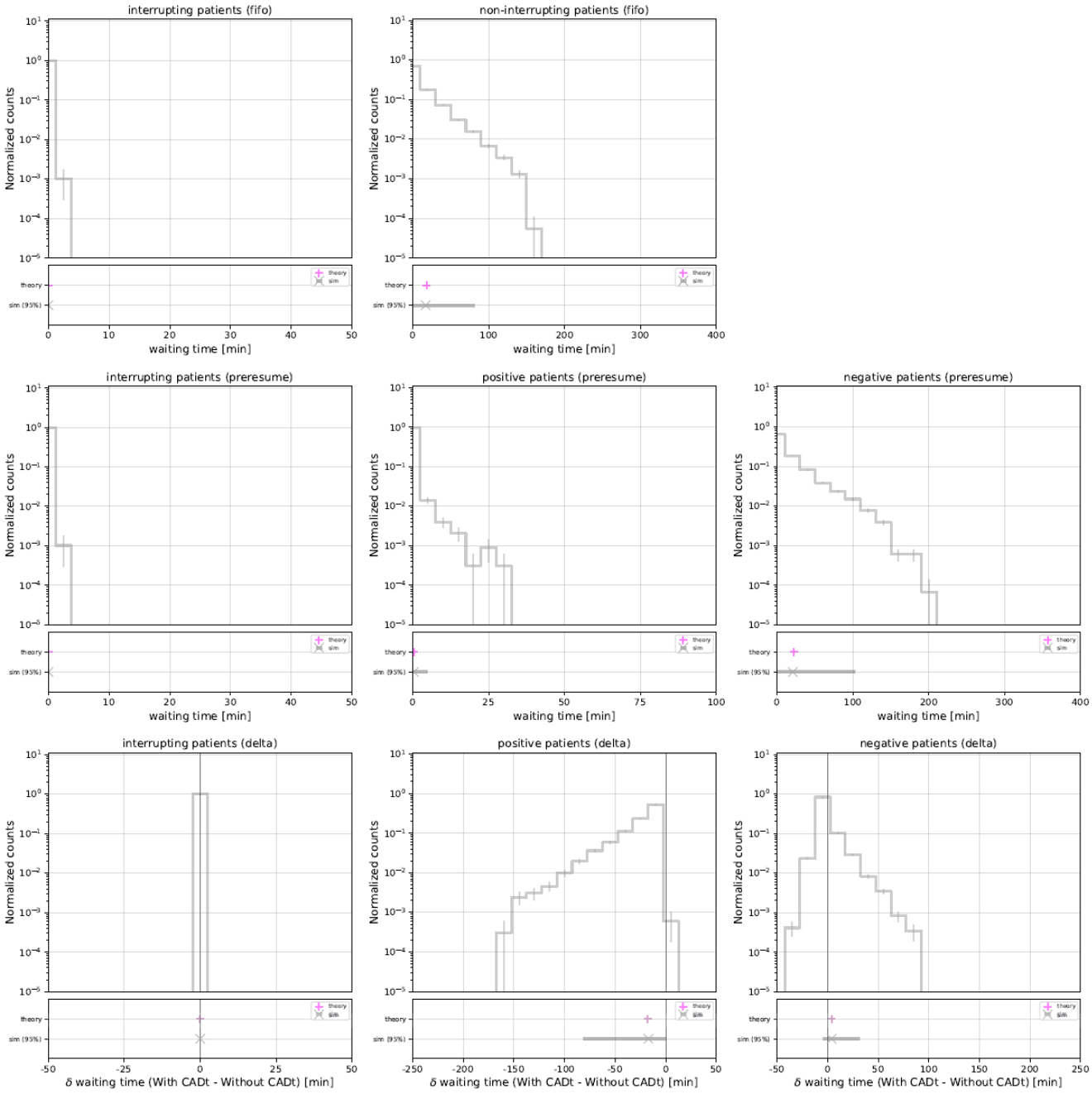

Figure 5: *Left columns*: The distributions of interrupting patient images' waiting time in a without-CADt scenario (top), waiting time in a with-CADt scenario (middle), and per-patient time difference between the two scenarios (bottom). *Middle columns*: The distributions of waiting time of non-interrupting patient images in a without-CADt scenario (top), waiting time of AI-positive patient images in a with-CADt scenario (middle), and per-patient time difference between the two scenarios for AI-positive patient images (bottom). *Right columns*: The distributions of waiting time of AI-negative patient images in a with-CADt scenario (middle), and per-patient time difference between the two scenarios for AI-negative patient images. The gray lines and bars are obtained from simulation, and the magenta lines and bars are theoretical predictions. For each subplot, top plot show the distribution, and the bottom plot shows the theoretical mean value and the simulation results with a 95% confidence limit.

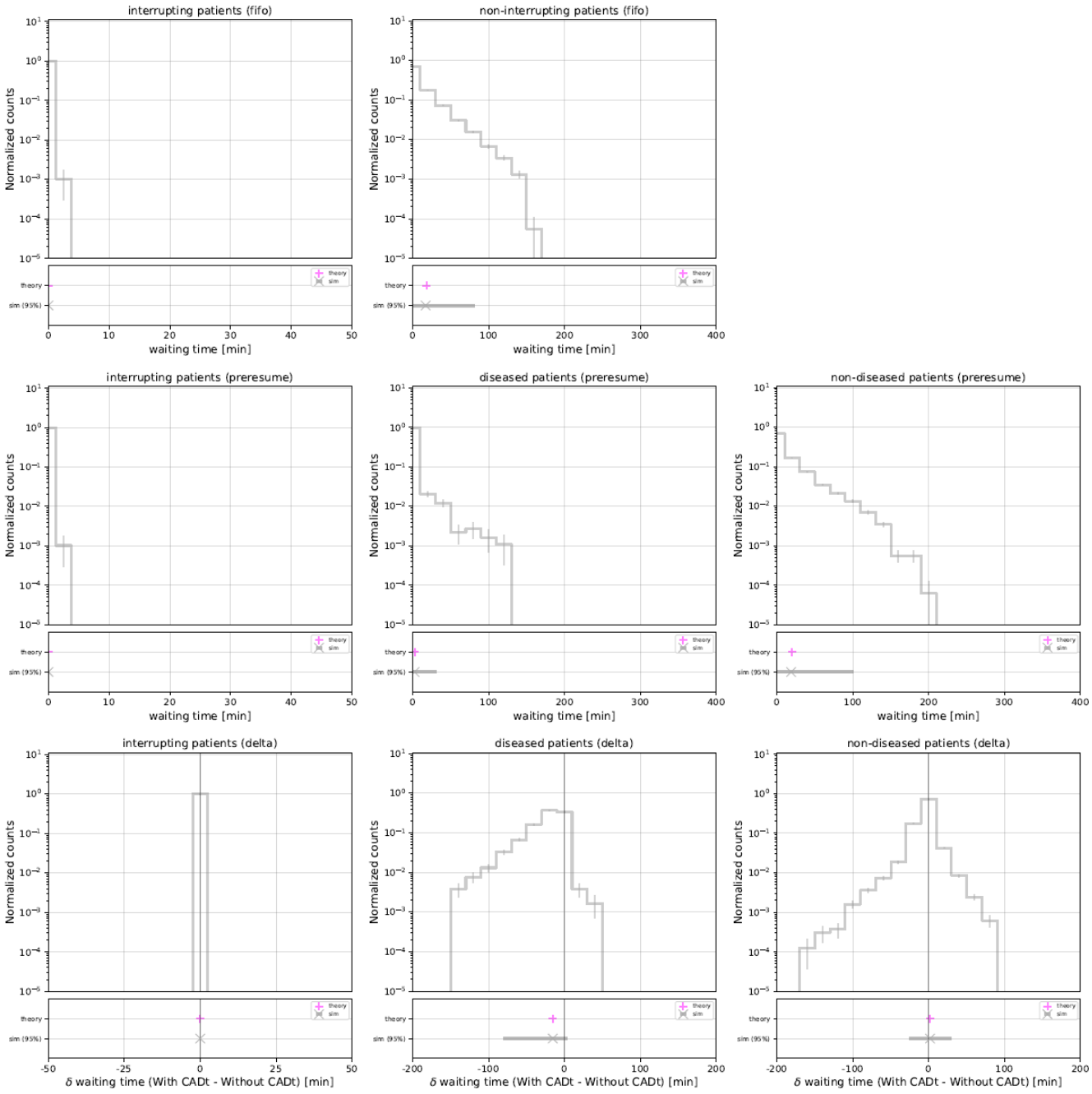

Figure 6: *Left columns*: The distributions of interrupting patient images' waiting time in a without-CADt scenario (top), waiting time in a with-CADt scenario (middle), and per-patient time difference between the two scenarios (bottom). *Middle columns*: The distributions of waiting time of non-interrupting patient images in a without-CADt scenario (top), waiting time of diseased patient images in a with-CADt scenario (middle), and per-patient time difference between the two scenarios for diseased patient images (bottom). *Right columns*: The distributions of waiting time of non-diseased patient images in a with-CADt scenario (middle), and per-patient time difference between the two scenarios for non-diseased patient images. The gray lines and bars are obtained from simulation, and the magenta lines and bars are theoretical predictions. For each subplot, top plot show the distribution, and the bottom plot shows the theoretical mean value and the simulation results with a 95% confidence limit.

The goal of this section is to provide a big-picture description for each class involved. For the step-by-step details, please visit the comments and documentations in the individual scripts.

4.1.1 CADt AI diagnostic performance

`tools/AI.py` is the class to simulate CADt's diagnostic performance. It can be initialized by an input operating point or by an input sensitivity with an ROC curve via a text file. The key job of an AI instance is to label whether a non-interrupting image is AI-positive or AI-negative, given the radiologist diagnosis (diseased vs non-diseased). If an ROC curve is provided, the AI call is a random rating from the corresponding (diseased or non-diseased) normal distribution. If the random rating is above the rating threshold, the image is labeled as AI-positive. If an operating point (both True-Positive Fraction (TPF) and False-Positive Fraction (FPF)) are provided, a random score is generated using a uniform distribution between 0 and 1. For a diseased image, if the random score is below the sensitivity threshold, the AI instance calls it positive. On the other hand, the AI instance calls a non-diseased image positive if the random score is larger than the specificity threshold.

4.1.2 Patient

`tools/patient.py` encapsulates information of a simulated patient image, including its interrupting status, disease (radiologist diagnosis) status, and AI call. Whether a patient image is an interrupting case is determined by a binomial distribution with a success probability equal to input parameter `fractionED`. If the patient image is indeed an interrupting case, it will have no disease status or AI call status. Otherwise, the disease status is obtained by a random number from another binomial distribution with a success probability equal to input parameter `prevalence`. Once the disease status is determined, an AI instance can generate the AI call label to determine if this patient is AI-positive or AI-negative.

Besides status, a patient instance also keeps a record of important timestamps such as trigger (arrival) time, radiologist's reading time, waiting time, total time in system, etc. It is important to note that for every patient is put into both with-CADt and without-CADt scenarios, implying that the arrival and radiologist's reading time of this very same patient are identical in the two scenarios. To determined the arrival time, an interarrival time is first randomly generated from an exponential distribution defined by the arrival rate that was calculated from the input parameters `traffic` and the effective radiologist's service time. We then take the arrival timestamp of the previous patient image and add the randomly determined interarrival time to obtain the arrival timestamp of the current patient. The radiologist's reading time for this patient is determined by a radiologist instance (see Section 4.1.3).

4.1.3 Radiologist

Similar to the patient class, `tools/radiologist.py` encapsulates a server (radiologist) class. Each radiologist has a name and a patient instance that the radiologist is currently handling. Throughout the simulation, the current patient instance is updated as the radiologist gets interrupted, reads a new case, and/or closes the current case.

The radiologist class also contains a function that determines how much time it takes to review this patient. This time is randomly generated based on an exponential distribution characterized by a rate,

which can either be the service rate for interrupting subgroup, diseased subgroup, or non-diseased subgroup.

4.1.4 Simulator

This `tools/simulator.py` is probably the most important class of the simulation software. It simulates a workflow of patient images in a radiology department. When an in-coming patient is initiated, an ID based on the arrival order is assigned, as well as other randomly generated properties such as arrival timestamp, radiologist's service time, and interrupting/diseased/AI call status. This simulated patient is simultaneously put into two queues: one in the with-CADt scenario and one in the without-CADt scenario. In both scenarios, a preemptive resume scheduling is assumed, in which a radiologist would be stopped reading the lower priority image in hand when an image of higher priority arrives. In the without-CADt scenario, the higher priority images are the interrupting cases that are not reviewed by the CADt device and require the immediate attention. The lower priority cases are non-interrupting cases in the reading queue and can be grouped by the radiologist's diagnosis (diseased vs non-diseased). It should be noted that, when a CADt device is in used, these non-interrupting cases will be analysed by the CADt device upon their arrival to the system. Therefore, in addition to the diseased vs non-diseased subgroups, these non-interrupting cases can also be categorized into AI-positive and AI-negative subgroups in the with-CADt scenario, and hence the true-positive, false-positive, true-negative, and false-negative subgroups.

At the beginning of the simulation run, the first patient image arrives, seeing an empty system with no images. This very first patient image is then immediately reviewed by a radiologist, and the timestamp at which the radiologist starts reviewing the case is recorded. The waiting time is therefore the time difference between the radiologist-start-reading timestamp and the arrival timestamp (i.e. zero minutes). When the radiologist is reviewing the first patient, another patient image may arrive, seeing that the system has already had one case with a specific priority class (either interrupting/non-interrupting in without-CADt scenario or interrupting/AI-positive/AI-negative in with-CADt scenario). Note that, upon each new arrival, the number of cases in each priority class is recorded to check the agreement of state probabilities between simulation and theory. If only one radiologist is available, and the second patient has a higher priority class than the first patient, the radiologist will stop reading the first patient image (putting it back to the queue) and immediately review the second patient. For the first patient, the timestamp at which it is put back into the queue is recorded, and its waiting period starts. Otherwise, the second patient will be put in the reading queue. When the one radiologist finishes reviewing the first patient image, it will start reviewing the second patient image. If more than one radiologist is available, the second radiologist will start reviewing the second patient image regardless of the priority of the second patient image.

As more patients are simulated, the queue changes whenever action is needed. For example, when a radiologist reads a new case, the patient image is removed from the queue, and when the radiologist closes a case, the image is removed from the system. When a radiologist is interrupted by a higher priority image, the lower priority image is put back into the queue. It is worth noting that a simulated patient is not immediately put into the queue when first initialized. When a new patient is initialized, its arrival time may be after the radiologist closes several cases. Therefore, before reaching the new arrival timestamp, several patient cases should be removed from the system.

Each with- and without-CADt scenario has its own reading queue. For with-CADt scenario, the queue has two priority classes: interrupting (with a higher priority) and non-interrupting (lower priority). For without-CADt scenario, images have three priority classes: interrupting (highest priority), AI-positive

Listing 12: Ipython example code to calculate the theoretical mean time-savings for diseased subgroup

```
In [7]: diseased_waittime_withoutCADt = calculator.get_theory_waitTime ('diseased', 'fifo', adicts['params'])

In [8]: diseased_waittime_withCADt = calculator.get_theory_waitTime ('diseased', 'preresume', adicts['params'])

In [9]: diseased_waittime_difference = calculator.get_theory_waitTime ('diseased', 'delta', adicts['params'])

In [10]: print ('For diseased subgroup:')
    ...: print ('  * average wait time [min] if without-CADt: {0:.2f}'.format (diseased_waittime_withoutCADt))
    ...: print ('  * average wait time [min] if with-CADt   : {0:.2f}'.format (diseased_waittime_withCADt))
    ...: print ('  * average time difference [min]          : {0:.2f}'.format (diseased_waittime_difference))
For diseased subgroup:
  * average wait time [min] if without-CADt: 18.29
  * average wait time [min] if with-CADt   : 2.57
  * average time difference [min]          : -15.72
```

(middle priority), and AI-negative (lowest priority). Within each priority class, patient images are ordered by the arrival timestamps.

While simulating the patient image flow, this `simulator` class keeps track of information from all patients in the system, including the disease/interrupting status, the AI call, the reading duration (i.e. service time), the various timestamps (arrival, opened and closed, interrupted, etc.), waiting time, radiologist service time, etc. When a patient arrives, the number of patients in the system per priority class right before the patient's arrival are recorded. Besides the simulations itself, this class also has functions for debugging purposes, including the option to print a log file of all patients and the functions to extract patients of specific subgroups. Besides functions that simulate patient image flows, this class also contains functions that extract all patients of a specified subgroup in either with-CADt or without-CADt scenario.

4.1.5 Trial generator

Last, `tools/trialGenerator.py` defines a class that generate trials. This class passes user input parameters and calls the simulator class to simulate reading flow. At the end of running all trials, this script contains function that calculates the statistics (e.g. means and 95% confidence limits) of waiting time and number of patients in system observed by an in-coming patient for each subgroup.

4.2 Theoretical calculation

Compared to the simulation software, the code for the theoretical calculation is much more straight forward. The `calculator.py` script contains functions and classes to compute the state probabilities, average waiting time, and mean time difference between with- and without-CADt scenarios. For detailed information on the calculation, please visit Thompson et al. (2023) **?**.

4.2.1 Get theoretical waiting time and time difference

One can use the `get_theory_waitTime()` function to calculate the average waiting time and mean time difference for a specific subgroup of patient images. An example code snippet is shown in Listing 12, using the example in Section 3.

It should be noted that theoretical predictions of mean waiting time and mean time difference for AI-negative, diseased, and non-diseased subgroups are not available when there are more than 2 radiologists `nRadiologists` $>$ 2 ***and*** interrupting patients are present `fractionED` $>$ 0. If such a request is given to the function, a `None` will be returned. This is a limitation that will be improved in the future.

5 Future directions

Here is a list of improvement and extension we are working on expand our software in the near future.

- design a more user-friendly, interactive GUI interface
- include an installation instruction for Anaconda users
- package the software into a python package to be installed via `pip install`
- provide job submission and more plotting tools to demonstrate safety and effectiveness of CADt devices
- consider a scenario with multiple CADt devices arranging images in the same queue that have multiple disease conditions
- consider other AI/ML-based medical devices that are intended to improve radiologist's workflow

If you are interested in providing feedback and in collaborating with us on the software, please contact Elim Thompson.

6 Related publications

- arXiv pre-print (2023) - To Be Published
- SPIE Medical Imaging (2022)
- FDA Science Forum (2021)

7 List of Acronyms

SaMD Software as a Medical Device 3

AI/ML Artificial Intelligence and Machine Learning 3

CADt Computed-Aided Triage and Notification 3