CancerSim

Release 1.0.0

Aug 20, 2020

Contents:

Build Status Documentation Status Binder

CHAPTER 1

Documentation

Documentation for `CancerSim`, including this README and the API reference manual is hosted on readthedocs.

CHAPTER 2

Background

Cancer is a group of complex diseases characterized by excessive cell proliferation, invasion, and destruction of the surrounding tissue [*1*] . Its high division and mutation rates lead to excessive intratumour genetic heterogeneity which makes cancer highly adaptable to environmental pressures such as therapy [*2*] . This process is known as somatic evolution of cancer. Throughout most of its existence a tumour is inaccessible to direct observation and experimental evaluation. Therefore, computational modelling can be useful to study many aspects of cancer. Some examples where theoretical models can be of great use include early carcinogenesis, as lesions are clinically observable when they already contain millions of cells, seeding of metastases, and cancer cell dormancy [*3*] .

Here, we present `CancerSim`, a software that simulates somatic evolution of tumours. The software produces virtual spatial tumours with variable extent of intratumour genetic heterogeneity and realistic mutational profiles. Simulated tumours can be subjected to multi-region sampling to obtain mutation profiles that are realistic representation of the sequencing data. This makes the software useful for studying various sampling strategies in clinical cancer diagnostics. An early version of this cancer evolution model was used to simulate tumours subjected to sampling for classification of mutations based on their abundance [*4*]. Target users of `CancerSim` are scientists working in the field of mathematical oncology. Simplicity and accessibility of our model in comparison to more advanced models (see e.g. Ref. [*5*]) makes it particularly suitable for students with interest in somatic evolution of cancer.

Our model is abstract, not specific to any neoplasm type, and does not consider a variety of biological features commonly found in neoplasm such as vasculature, immune contexture, availability of nutrients, and architecture of the tumour surroundings. It most closely resembles the superficially spreading tumours like carcinoma in situ, skin cancers, or gastric cancers, but it can be used to model any tumour on this abstract level.

The tumour is simulated using a two-dimensional, on-lattice, agent-based model. The tumour lattice structure is established by a sparse matrix whose non-zero elements correspond to the individual cells. Each cell is surrounded by eight neighbouring cells (Moore neighbourhood). The value of the matrix element is an index pointing to the last mutation the cell acquired in the list of mutations which is updated in each simulation step.

The simulation advances in discrete time-steps. In each simulation step, every tumour cell in the tumour that has an unoccupied neighbour can divide with a certain probability (set by the parameter `division_probability`). The daughter cell resulting from a cell division inherits all mutations from the parent cell and acquires a new mutation with a given probability `mutation_probability`). A new mutation that changes death and birth probability of cell can be introduced at into random cell at the specific time step defined by `adv_mutation_wait_time`. By changing fitness parameters of a mutant cell

adv_mutant_division_probability and adv_mutant_death_probability one can model various evolutionary processes like emergence of a faster dividing sub-clone or selective effects of a drug treatment.

The simulation allows the acquisition of more than one mutational event per cell (number_of_mutations_per_division). In that case, variable amounts of sequencing noise [6] can be added to make the output data more biologically realistic. The key parameters number_of_generations, division_probability and death_probability determine the final size of the tumour, while the degree of intratumour heterogeneity can be varied by changing the mutation_probability parameter. For neutral tumour evolution, parameter adv_mutant_division_probability and adv_mutant_death_probability must be the same as division_probability and death_probability.

Throughout the cancer growth phase, CancerSim stores information about the parent cell and a designation of newly acquired mutations for every cell. Complete mutational profiles of cells are reconstructed a posteriori based on the stored lineage information.

The division rules which allow only cells with empty neighbouring nodes to divide, cause exclusively peripheral growth and complete absence of dynamics in the tumour centre. To allow for variable degree of growth inside the tumour, we introduced a death process. At every time step, after all cells attempt their division, a number of random cells die according to death_probability and adv_mutant_death_probability and yield their position to host a new cancer cell in a subsequent time step.

After the simulation, the tumour matrix, and the lists of lineages and frequencies of each mutation in the tumour are exported to files. Furthermore, the virtual tumour can be sampled and a histogram over the frequency of mutations will be visualised. Alternatively, a saved tumour can be loaded from file and then be subjected to the sampling process.

CHAPTER 3

Installation

CancerSim is written in Python (version >3.5). We recommend to install it directly from the source code. To download the code:

EITHER clone the repository:

```
$> git clone https://github.com/mpievolbio-scicomp/cancer_sim.git
```

OR download the source code archive:

```
$> wget https://github.com/mpievolbio-scicomp/cancer_sim/archive/master.zip
$> unzip master.zip
$> mv cancer_sim-master cancer_sim
```

Change into the source code directory

```
$> cd cancer_sim
```

We provide for two alternatives to install the software after it was downloaded:

3.1 Alternative 1: conda

3.1.1 New conda environment

We provide an `environment.yml` to be consumed by `conda`. To create a fully self-contained conda environment (named `casim`):

```
$> conda env create -n casim --file environment.yml
```

This will also install the cancer simulation code into the new environment.

To activate the new conda environment:

```
$> source activate casim
```

or

```
$> conda activate casim
```

if you have set up conda appropriately.

3.1.2 Install into existing and activated conda environment

To install the software into an already existing environment:

```
$> conda activate <name_of_existing_conda_environment>
$> conda env update --file environment.yml
```

3.2 Alternative 2: `pip`

The file `requirements.txt` is meant to be consumed by `pip`:

```
$> pip install -r requirements.txt [--user]
```

The option `--user` is needed to install without admin privileges.

3.3 Installed module

After installation, `CancerSim` is available in python as the `casim` module. E.g. in a python script, one would import the module as:

```
>>> from casim import casim
```

CHAPTER 4

Testing

Although not strictly required, we recommend to run the test suite after installation. Simply execute the `run_tests.sh` shell script:

```
$> ./run_tests.sh
```

This will generate a test log named `casim_test@<timestamp>.log` with `<timestamp>` being the date and time when the test was run. You should see an `OK` at the bottom of the log. If instead errors or failures are reported, something is wrong with the installation or the code itself. Feel free to open a github issue at https://github.com/mpievolbio-scicomp/cancer_sim/issues and attach the test log plus any information that may be useful to reproduce the error (version hash, computer hardware, operating system, python version, a dump of `conda env export` if applicable).

The test suite is automatically run after each commit to the code base. Results are published on travis-ci.org.

CHAPTER 5

High-level functionality

5.1 Setting the simulation parameters

The parameters of the cancer simulation are specified in a python module or programmatically via the `CancerSimulationParameters` class. A documented example `params.py` is included in the source code (under `casim/params.py`) and reproduced here:

```
$> cat casim/params.py
################################################################################
#                                                                              #
# Commented casim parameter input file.                                        #
# Valid settings are indicated in parentheses at the end of each comment line. #
# [0,1] stands for the closed interval from 0 to 1, including the limits; ||   #
# means "or".                                                                  #
#                                                                              #
################################################################################

# Number of mesh points in each dimension (>0)
matrix_size = 1000

# Number of generations to simulate (>0).
number_of_generations = 20

# Probability of cell division per generation ([0,1]).
division_probability = 1

# Probability of division for cells with advantageous mutation ([0,1]).
adv_mutant_division_probability = 1

# Fraction of cells that die per generation ([0,1]).
death_probability = 0.1

# Fraction of cells with advantageous mutation that die per generation ([0,1]).
adv_mutant_death_probability = 0.0
```

(continues on next page)

(continued from previous page)

```
# Probability of mutations ([0,1]).
mutation_probability = 1

# Mutation probability for the adv. cells ([0,1]).
adv_mutant_mutation_probability = 1

# Number of mutations per cell division (>=0).
number_of_mutations_per_division = 10

# Number of generations after which adv. mutation occurs (>=0).
adv_mutation_wait_time = 10

# Number of mutations present in first cancer cell (>=0).
number_of_initial_mutations = 150

# Tumour multiplicity ("single" || "double").
tumour_multiplicity = "single"

# Sequencing read depth (read length * number of reads / genome length).
read_depth = 100

# Fraction of cells to be sampled ([0,1]).
sampling_fraction = 0.1

# Sampling position (list of (x,y) coordinates). If blank, random position will
# be chosen.
# sampling_positions = None # This will randomly set a single sampling position.
sampling_positions = [(500,500),(490,490)]

# Plot the tumour growth curve (True || False).
plot_tumour_growth = True

# Export the tumour growth data to file (True || False).
export_tumour = True
```

Here, we simulate a single 2D tumour on a 1000x1000 grid (`matrix_size=1000`) for a total of 20 generations (`number_of_generations=20`). On average, both healthy and mutant cells divide once per generation (`division_probability`). The first cancer cell carries 150 mutations (`number_of_initial_mutations=150`); both healthy and mutant cells aquire 10 new mutations (`number_of_mutations_per_division=10`) in each generation with a certainty of 100% (`mutation_probability=0.1`). The advantageous mutation happens in the 10th generation (`adv_mutation_wait_time=10`). Mutant cells with advantageous mutations live on forever (`adv_mutant_death_probability=0`) while healthy cells die with a rate of 0.1 per generation (`death_probability=0.1`).

Two spatial samples are taken, one from the tumour center and one from a slightly more lateral position (`sampling_positions = [(500,500),(490,490)]`). Each sample contains 10% closely positioned tumour cells (`sampling_fraction=0.1`). The samples are subject to genetic sequencing with a read depth of 100 (`read_depth=100`). The data is written to disk (`export_tumour=True`) and plots showing the mutation histograms for the whole tumour as well as for the sampled part of the tumour are generated. Furthermore, a plot showing the tumour growth over time is saved (`plot_tumour_growth=True`).

Users should start with the template and adjust the parameters as needed for their application by setting experimentally or theoretically known values or by calibrating the simulation output against experiments or other models.

5.2 Run the example

The simulation is started from the command line. The syntax is

```
$> python -m casim.casim [-h] [-s SEED] [-p PARAMS] [-o DIR]
```

`SEED` is the random seed. Using the same seed in two simulation runs with identical parameters results in identical results. If not given, `SEED` defaults to 1. `PARAMS` should point to a python parameter file. If not given, it defaults to `params.py` in the current working directory. If that file does not exist, default parameters are assumed. `DIR` specifies the directory where to store the simulation log and output data. If not given, output will be stored in the directory `casim_out` in the current directory. For each seed, a subdirectory `cancer_SEED` will be created. If that subdirectory already exists because an earlier run used the same seed, the run will abort. This is a safety catch to avoid overwriting data from previous runs.

5.3 Output

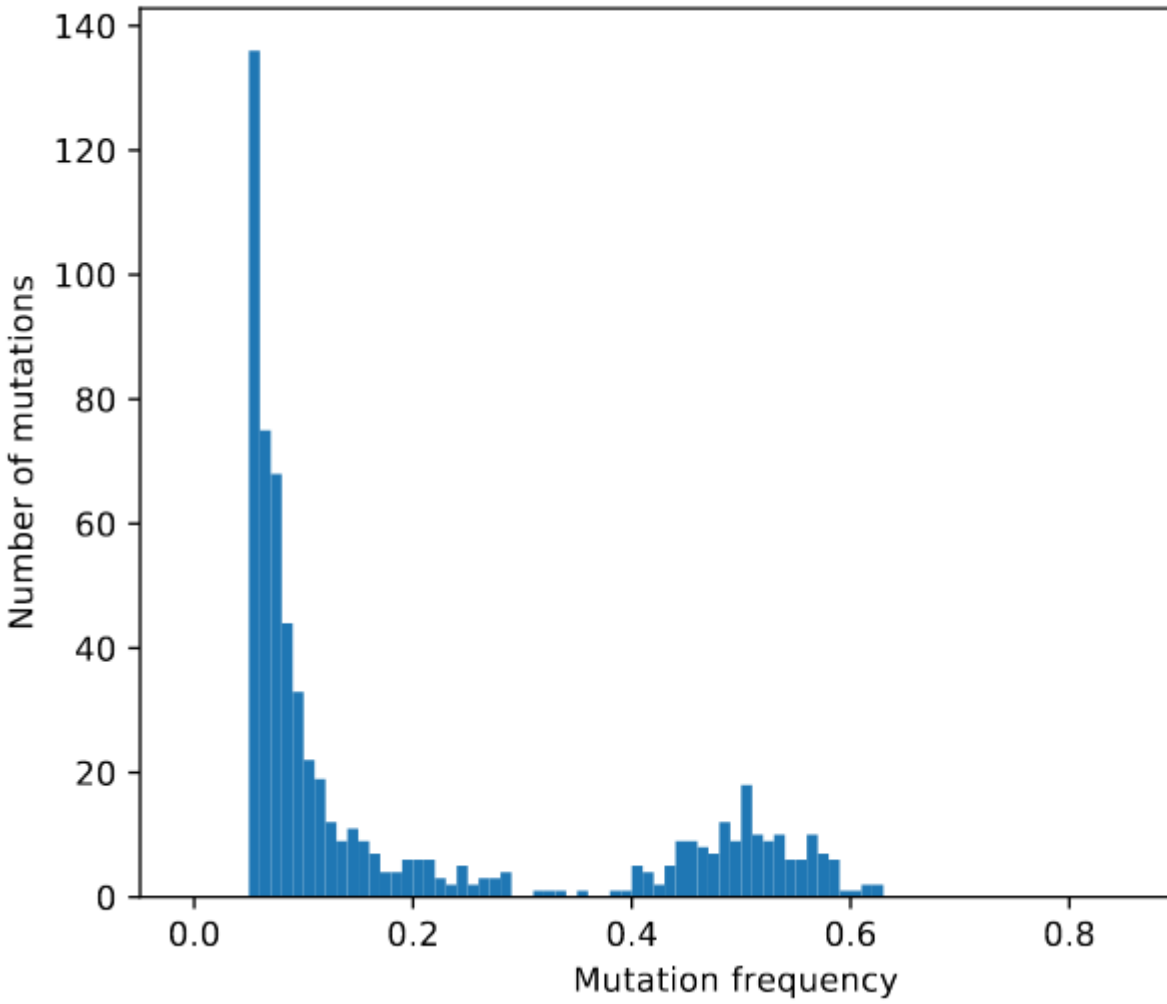
After the run has finished, you should find the results in the specified output directory:

```
$> ls out/cancer_1/simOutput
growthCurve.pdf  mut_container.p              sample_out_490_490.txt
mtx.p            sampleHistogram_490_490.pdf  sample_out_500_500.txt
mtx_VAF.txt      sampleHistogram_500_500.pdf  wholeTumourVAFHistogram.pdf
```

Let's take a look at the `.txt` files. They contain the simulation output: `mtx_VAF.txt` is a datafile with three columns: `mutation_id` lists the index of each primary mutation, `additional_mut_id` indexes the subsequent mutations that occur in a cell of a given `mutation_id`; `frequency` is the frequency which at a given mutation occurs.

Corresponding to the given sample positions, there is one `sample_out_XXX_YYY.txt` and one `sampleHistogram_XXX_YYY.pdf` for each position. The `.txt` filel lists all mutations of the artificial sample taken from the whole tumour. Columns are identical to `mtx_VAF.txt`.

The `.pdf` files are plots of sampled tumour histogram. `wholeTumourVAFHistogram.pdf` is the histogram for the complete tumour. You should see figures similar to these:



Whole tumour histogram:
tumour histogram

include/img/example_sampled_tumour_pos1.png

Central sample histogram: Sampled tumour histogram

include/img/example_sampled_tumour_pos2.png

Lateral sample histogram: Sampled tumour histogram

The remaining output files are serialized versions ("pickles") of the tumour geometry as a 2D matrix (`mtx.p`) and the mutation list (list of tuples listing the cancer cell index and the mutation ID of each tumour cell, `mut_container.p`).

5.4 Example notebooks

An example demonstrating how to parametrize the simulation through the `CancerSimulationParameters` API is provided in the accompanying jupyter notebook at quickstart_example.ipynb‘. Launch it in binder.

In `run_dump_reload_continue.ipynb`, we demonstrate how to use the restart capability to modifiy tumour growth parameters in the middle of a run. In this way, one can model different phases of tumour growth, e.g. tumour dormancy or onset of cancer therapy.

CHAPTER 6

Community Guidelines

As an Open Source project, we welcome all contributions to `CancerSim`. We recommend the usual github workflow: Fork this repository, commit your changes and additions to the fork and create a pull request back to the master branch on this repository. If uncertain about anything, please create an issue at https://github.com/mpievolbioscicomp/cancer_sim/issues.

Comments, bug reports, or other issues as well as requests for support should be submitted as a github issue. Please check the list of issues if your problem has already been addressed. We will do our best to respond in a timely manner.

CHAPTER 7

References

[1] J. C. A. Vinay Kumar Abul K. Abbas, *Robbins Basic Pathology*, 10th ed. (Elsevier, 2017). ISBN: 9780323353175.

[2] S. Turajlic, A. Sottoriva, T. Graham, and C. Swanton, Nat Rev Genet (2019). DOI: 10.1038/s41576-019-0114-6

[3] P. M. Altrock, L. L. Liu, and F. Michor, Nat Rev Cancer **15**, 730 (2015). DOI: 10.1038/nrc4029

[4] L. Opasic, D. Zhou, B. Werner, D. Dingli, and A. Traulsen, BMC Cancer **19**, 403 (2019). DOI: 10.1186/s12885-019-5597-1

[5] B. Waclaw, I. Bozic, M. E. Pittman, R. H. Hruban, B. Vogelstein, and M. A. Nowak, Nature **525**, 261 (2015). DOI: 10.1038/nature14971

[6] M. J. Williams, B. Werner, C. P. Barnes, T. A. Graham, and A. Sottoriva, Nature Genetics **48**, 238 (2016). DOI: 10.1038/ng.3489

CHAPTER 8

Quickstart example for cancer simulation

This notebook illustrates how to run a 2D cancer simulation using CancerSim. It's a rather contrieved example with no actual scientific background. It's main purpose is to give a simple example to be used as a template for more complex simulation runs. It also demonstrates how to parametrize the simulation through the `CancerSimulationParameters` API instead of the `params.py` module.

The simulation treats a 2D tumour on a 20x20 grid for 10 generations. Healthy cells and cancer cells differ by their division probabilities (0.5 vs 0.8), death probabilities (0.1 vs. 0.4), and mutation probabilities (0.2 vs. 0.8). The first cancer cell carries 2 mutations, new mutations occur with a 20% probability and 10 fold abundance. The first advantageous mutation occurs in the 3rd generation. We sample 10% of the tumour at a read depth of 100.

After the run, we plot the tumour growth curve and the mutation frequency histograms for the whole tumour and the sampled tumour.

8.1 Import modules

[6]:
```
# The cancer simulation module.
from casim import casim
import logging
```

[2]:
```
#  3rd party modules.
import os, shutil
from wand.image import Image as WImage # To render pdf images in nb.
from glob import glob
```

8.2 Setup parameters

[3]:
```
parameters=casim.CancerSimulatorParameters(
                                matrix_size=20,
                                number_of_generations=10,
```

(continues on next page)

(continued from previous page)

```
                                    division_probability=0.5,
                                    adv_mutant_division_probability=0.8,
                                    death_probability=0.1,
                                    adv_mutant_death_probability=0.4,
                                    mutation_probability=0.2,
                                    adv_mutant_mutation_probability=0.8,
                                    number_of_mutations_per_division=10,
                                    adv_mutation_wait_time=3,
                                    number_of_initial_mutations=2,
                                    sampling_fraction=0.1,
                                    plot_tumour_growth=True,
                                    export_tumour=True
                                   )
```

8.3 Setup the simulation engine.

[4]:
```
if os.path.isdir('out'):
    shutil.rmtree('out')
```

[7]:
```
cancer_sim = casim.CancerSimulator(parameters, seed=1, outdir='out/')
```

[11]:
```
# Get more verbose logging.
logger = logging.getLogger().setLevel("INFO")
```

8.4 Run the simulation

[10]:
```
cancer_sim.run()
```

```
2020-08-10 16:14:45,538 INFO: Running in single tumour mode.
2020-08-10 16:14:45,540 INFO: First cell at (10, 10).
2020-08-10 16:14:45,542 INFO: Ready to start CancerSim run with these parameters:
2020-08-10 16:14:45,543 INFO: matrix_size = 20
2020-08-10 16:14:45,544 INFO: number_of_generations = 10
2020-08-10 16:14:45,545 INFO: division_probability = 0.5
2020-08-10 16:14:45,546 INFO: adv_mutant_division_probability = 0.8
2020-08-10 16:14:45,547 INFO: death_probability = 0.1
2020-08-10 16:14:45,547 INFO: adv_mutant_death_probability = 0.4
2020-08-10 16:14:45,548 INFO: mutation_probability = 0.2
2020-08-10 16:14:45,549 INFO: adv_mutant_mutation_probability = 0.8
2020-08-10 16:14:45,550 INFO: number_of_mutations_per_division = 10
2020-08-10 16:14:45,551 INFO: adv_mutation_wait_time = 3
2020-08-10 16:14:45,551 INFO: number_of_initial_mutations = 2
2020-08-10 16:14:45,552 INFO: tumour_multiplicity = single
2020-08-10 16:14:45,553 INFO: read_depth = 100
2020-08-10 16:14:45,554 INFO: sampling_fraction = 0.1
2020-08-10 16:14:45,554 INFO: plot_tumour_growth = True
2020-08-10 16:14:45,555 INFO: export_tumour = True
2020-08-10 16:14:45,556 INFO: Tumour growth in progress.
2020-08-10 16:14:45,560 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,567 INFO: No new mutation in normal division, inheriting from␣
↪parent
```

(continues on next page)

(continued from previous page)

```
2020-08-10 16:14:45,571 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,572 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,578 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,582 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,583 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,586 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,587 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,588 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,591 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,593 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,593 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,594 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,595 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,598 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,598 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,602 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,604 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,605 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,607 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,609 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,610 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,612 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,616 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,616 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,617 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,618 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,619 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,621 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,622 INFO: No new mutation in normal division, inheriting from␣
↪parent
```

(continues on next page)

(continued from previous page)

```
2020-08-10 16:14:45,623 INFO: No new mutation in normal division, inheriting from␣
↪parent
2020-08-10 16:14:45,624 INFO: All generations finished. Starting tumour␣
↪reconstruction.
2020-08-10 16:14:45,625 INFO: Reconstruction done,  get statistics.
2020-08-10 16:14:45,630 INFO: Exporting simulation data
2020-08-10 16:14:45,901 INFO: Growth curve graph written to out/cancer_1/simOutput/
↪growthCurve.pdf.
2020-08-10 16:14:46,082 INFO: CancerSim run has finished.
2020-08-10 16:14:46,084 INFO: Simulation output written to: out/cancer_1/simOutput.
2020-08-10 16:14:46,084 INFO: Log files written to: out/cancer_1/log.
2020-08-10 16:14:46,085 INFO: Consumed Wall time of this run: 0.525742 s.
```

[10]:
```
0
```

```
<Figure size 432x288 with 0 Axes>
```

8.5 Output

After the run has finished, you should find the results in `out/cancer_1/simOutput`.

[12]:
```
!ls out/cancer_1/simOutput
```

```
death_list.p    mtx_VAF.txt               sample_out_12_11.txt
growthCurve.pdf  mut_container.p           wholeTumourVAFHistogram.pdf
mtx.p            sampleHistogram_12_11.pdf
```

Files with the extension `.p` are binary files (python pickles) needed to restart a simulation. Let's take a look at the `.txt` files. They contain the simulation output: `mtx_VAF.txt` is a datafile with three columns: `mutation_id` lists the index of each primary mutation, `additional_mut_id` indexes the subsequent mutations that occur in a cell of a given `mutation_id`; `frequency` is the frequency at which a given mutation occurs.

The file `sample_out_502_488.txt` lists all mutations of the artificial sample taken from the whole tumour. Columns are identical to `mtx_VAF.txt`.

The two `.pdf` files are plots of the whole tumour histogram and the sampled tumour histogram, respectively. You should see figures similar to these:

8.6 Display results.

The mutation frequency histogram for the whole tumour and the for sampled part of the tumour, as well as the tumour growth curve (number of tumour cells vs. time) are stored as pdf images in the simulation output directory.

[13]:
```
image_path=os.path.join(cancer_sim.outdir,'cancer_1', 'simOutput')
```

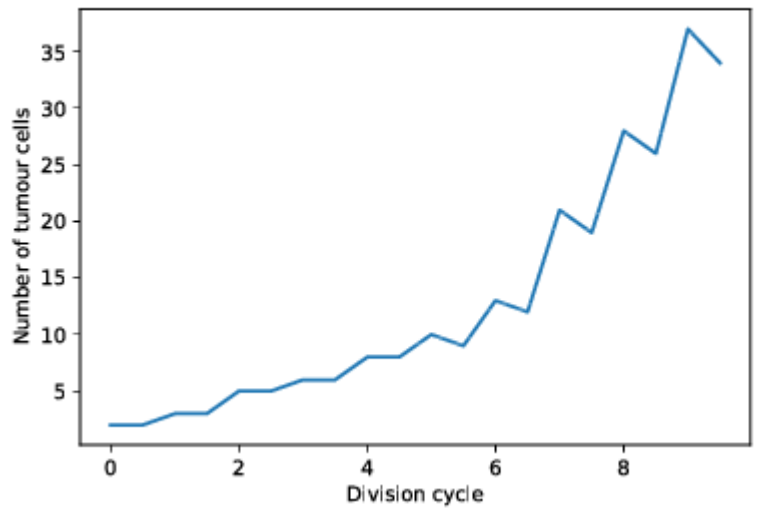
[14]:
```
pdfs = glob(os.path.join(image_path,"*.pdf"))
```

[15]:
```
pdfs
```

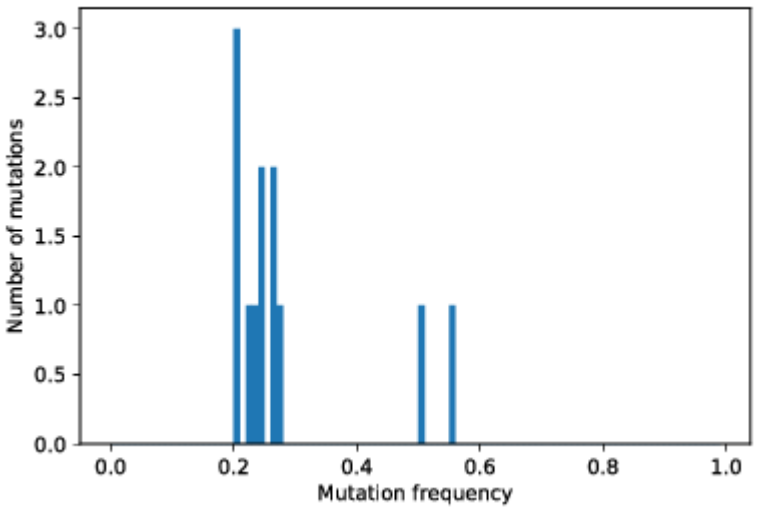
[15]:
```
['out/cancer_1/simOutput/growthCurve.pdf',
 'out/cancer_1/simOutput/sampleHistogram_12_11.pdf',
 'out/cancer_1/simOutput/wholeTumourVAFHistogram.pdf']
```

[16]:
```
for pdf in pdfs:
    print(pdf)
    display(WImage(filename=pdf))
```

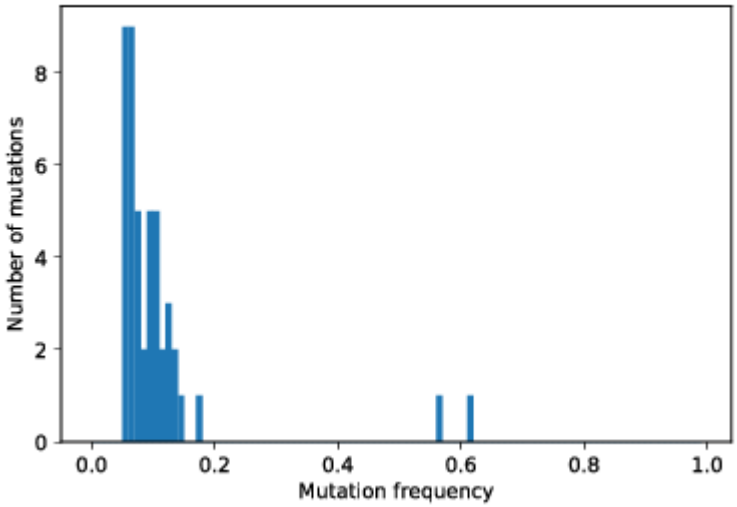
```
out/cancer_1/simOutput/growthCurve.pdf
```




```
out/cancer_1/simOutput/sampleHistogram_12_11.pdf
```




```
out/cancer_1/simOutput/wholeTumourVAFHistogram.pdf
```



8.6.1 Load binary files (pickles)

The remaining output files are serialized versions ("pickles") of the tumour geometry as a 2D matrix (`mtx.p`), the death list (`death_list.p`), and the mutation list (list of tuples listing the parent and the mutation ID of each tumour cell, `mut_container.p`).

To read the pickled data, we define a utility function

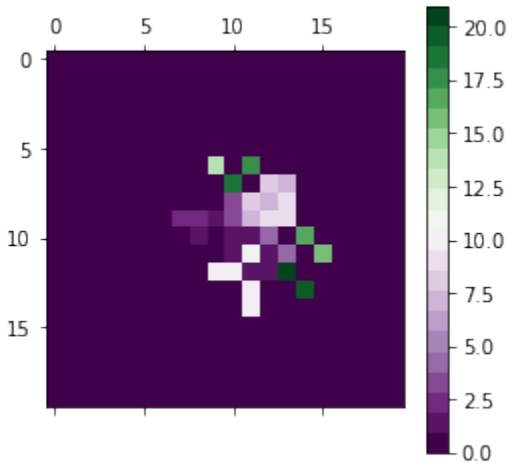
[17]:
```
import pickle
from matplotlib import pyplot
```

[18]:
```
def unpickle(file):

    with open(os.path.join(cancer_sim._CancerSimulator__simdir, file),'rb') as fp:
        return pickle.load(fp)
```

[19]:
```
# Load the mutation matrix
mtx=unpickle('mtx.p').toarray()
```

[20]:
```
# Plot the tumour as a 2D map color coding the mutation ID
cmap = pyplot.cm.get_cmap('PRGn', mtx.max()+1)

pyplot.matshow(mtx, cmap=cmap)
pyplot.colorbar()
```

[20]:
```
<matplotlib.colorbar.Colorbar at 0x7fc24be3b358>
```



[21]:
```
unpickle('mut_container.p')
```

[21]:
```
[(0, 0),
 (0, 1),
 (1, 2),
 (1, 3),
 (1, 4),
 (1, 5),
 (5, 6),
 (5, 7),
 (6, 8),
 (6, 9),
 (1, 10),
 (1, 11),
 (3, 12),
 (3, 13),
 (3, 14),
 (3, 15),
 (4, 16),
 (4, 17),
 (15, 18),
 (15, 19),
 (1, 20),
 (1, 21)]
```

CHAPTER 9

Reference Manual

class casim.casim.**CancerSimulator**(*parameters=None*, *seed=None*, *outdir=None*)

Class CancerSimulator Represents the Monte-Carlo simulation of cancer tumour growth on a 2D grid.

Construct a new CancerSimulation.

Parameters

- **parameters** (*CancerSimulationParameters*) – The cancer simulation parameters
- **seed** (*int*) – The random seed.
- **outdir** (*(str || path-like object)*) – The directory where simulation data is saved. Default: "casim_out/" in the current working directory.

count_mutations(*mutation_list*, *get_frequencies*)

Count number each time mutation is detected in the sample

Parameters **mutation_list** (*list of lists*) – mutation profiles of each cell in the sample

death_step(*step*)

Takes a group of random cells and kills them

Parameters **step** (*int*) – The time step in the simulation

division(*cell*, *beneficial*, *neighbors*, *step*, *mutation_counter*, *pool*)

Perform a cell division.

Parameters

- **cell** (*tuple*) – The mother cell coordinates.
- **beneficial** (*bool*) – Flag to indicate if the cell carries the beneficial mutation.
- **neighbors** (*list*) – The neighboring cells.
- **step** (*int*) – The time step in the simulation

- **mutation_counter** (*int*) – The counter of mutations to be updated
- **pool** (*list*) – The (temporary) pool of cells.

dump()
Serialize the object. The current simulation will be stored in a machine readable format to <OUTDIR>/cancer_<SEED>/cancer_sim.py.dill, where <OUTDIR> is the specified output directory or (if the latter was not defined) a temporary directory.

export_histogram(*sample_data*, *sample_coordinates*)
Create and export histogram of mutational frequencies (aka variant allelic frequencies)

Parameters

- **sample_data** (*list*) – List of mutations and their frequencies
- **sample_coordinates** (*tuple (i,j) of cell indices*) – coordinates of central sample cell

export_sample(*sample_data*, *sample_coordinates*)
Export (write to disk) frequencies of samples.

Parameters

- **sample_data** (*list*) – List of mutations and their frequencies
- **sample_coordinates** (*tuple (i,j) of cell indices*) – coordinates of central sample cell

export_tumour_matrix(*tumour_mut_data*)
Export (write to disk) the matrix of tumour cells.

Parameters **tumour_matrix** (*array like*) – The tumour matrix to export

extend_sample(*sample_center*, *sample_size*)
Takes a subset of cells from the tumour positioned around single input cell with specific coordinates. Output is a list of tuples of cells belonging to the sample. :param sample_center: coordinates of cell that will be center of the sample :type sample: tuple

Parameters **sample_size** (*float*) – The size of the sample (fraction of total cells.)

increase_mut_number(*original_mut_list*)
Scale up the number of mutations according to the 'number_of_initial_mutations' 'and number_of_mutations_per_division' parameter.

Parameters **solid_pre_vaf** (*list*) – The list of mutations to scale.

mutation(**args*)
Perform a mutation.

Parameters

- **cell** – At which cell the mutation occurs
- **neighbors** – The neighboring cells
- **mutation_counter** – The current number of mutations, to be incremented.
- **pool** – The pool of all cells.
- **place_to_divide** – The position at which the mutation occurs.
- **beneficial** – Flag to control whether the mutation is beneficial or not.

mutation_reconstruction(*cells_to_reconstruct*)
Reconstructs list of mutations of individual cell by going thorough its ancestors.

Parameters **cell** (*list of tuples [(i,j)] of cell indices.*) – Cell for which mutational profile will be recovered.

neighbours(*cell*)
Returns the nearest-neighbor cells around the given node.

Parameters **cell** (*tuple (i,j) of cell indices.*) – The node for which to calculate the neighbors.

place_to_divide()
Selects random unoccupied place on the matrix where cell will divide.

run()
Run the simulation.

Returns 0 if the run finishes successfully.

After a successful run, simulation output and log will be written to the output directory *<DIR>/cancer_<SEED>/simOutput* and *<DIR>/cancer_<SEED>/log*, respectively. Simulation output is split into several files:

- *mtx_VAF.txt* is a datafile with three columns: *mutation_id* lists the index of

each primary mutation, *additional_mut_id* indexes the subsequent mutations that occur in a cell of a given *mutation_id*; *frequency* is the frequency which at a given mutation occurs.

- *sample_out_XXX_YYY.txt* lists all mutations of the artificial sample

taken from the whole tumour. Columns are identical to *mtx_VAF.txt*.

- *wholeTumourVAFHistogram.pdf* contains a histogram plot of the mutation frequencies for the whole tumour
- *sampleHistogram_XXX_YYY.pdf* is the mutation frequency histogram for the sampled portion of the tumour. The two numbers XXX and YYY are the positional coordinates (grid indices) in the tumour matrix.
- *mtx.p* is the serialized (aka "pickled") 2D tumour matrix in sparse matrix format.
- *mut_container.p* is the serialized (aka "pickled") mutation list, a list of tuples [t_i]. Each tuple t_i consists of two values, t_i = (c_i, m_i). The first element c_i is the cell number in which the i'th mutation occurs. The second element, m_i, is the mutation index m_i=i.

simulate_seq_depth(*extended_vaf*)
Ads a beta binomial noise to sampled mutation frequencies

Parameters **extended_vaf** (*list*) – The list of cells to take a sample from.

terminate_cell(*cell*, *step*)
Kills cancer cell and removes it from the pool of cancer cells

Parameters

- **cell** (*tuple (i,j) of cell indices.*) – cell chosen for termination
- **step** (*int*) – The time step in the simulation

tumour_growth()
Run the tumour growth simulation.

class casim.casim.**CancerSimulatorParameters**(*matrix_size=None*, *number_of_generations=None*, *division_probability=None*, *adv_mutant_division_probability=None*, *death_probability=None*, *adv_mutant_death_probability=None*, *mutation_probability=None*, *adv_mutant_mutation_probability=None*, *number_of_mutations_per_division=None*, *adv_mutation_wait_time=None*, *number_of_initial_mutations=None*, *tumour_multiplicity=None*, *read_depth=None*, *sampling_fraction=None*, *plot_tumour_growth=None*, *export_tumour=None*, *sampling_positions=None*)

Class CancerSimulatorParameters Represents the parameters for a cancer simulation.

Construct a new CancerSimulationParameters object.

Parameters

- **matrix_size** (*int (matrix_size > 0)*) – The size of the (square) grid in each dimension.
- **number_of_generations** (*int (number_of_generations > 0)*) – The number of generations to simulate.
- **division_probability** (*float (0.0 <= division_probability <= 1.0)*) – The probability for a cell division to occur during one generation.
- **adv_mutant_division_probability** (*float (0.0 <= adv_mutant_division_probability <= 1.0)*) – The probability for the division of a cell with advantageous mutation to occur during one generation.
- **death_probability** (*float (0.0 <= death_probability <= 1.0)*) – The probability for a cell to die during one generation.
- **adv_mutant_death_probability** (*float (0.0 <= adv_mutant_death_probability <= 1.0)*) – The probability for a cell with advantageous mutation to die during one generation.
- **mutation_probability** (*float (0.0 <= mutation_probability <= 1.0)*) – The probalitiy of mutation.
- **adv_mutant_mutation_probability** (*float (0.0 <= adv_mutant_mutation_probability <= 1.0)*) – The rate for an advantageous mutation to occur during one generation.
- **number_of_mutations_per_division** (*int (0 < number_of_mutations_per_division)*) – The number of mutations per division
- **adv_mutation_wait_time** (*int (adv_mutation_wait_time > 0)*) – The number of generations into the simulation after which the advantageous mutation is inserted.
- **number_of_initial_mutations** (*int (number_of_initial_mutations >= 0)*) – Number of mutations present in first cancer cell.
- **tumour_multiplicity** – Run in single or double tumour mode (i.e.

consider growth of one single tumour or two tumours simultaneously). Possible values: “single”, “double”. :type tumour_multiplicity: str

Parameters

- **read_depth** (*int (read_depth >= 0)*) – The sequencing read depth (read length * number of reads / genome length). Default: 100.
- **sampling_fraction** (*float (0 <= sampling_fraction <= 1)*) – The fraction of cells to include in a sample. Default: 0.
- **sampling_positions** (*List (or array) of tuples of ints. E.g. ([10,20], [2,31])*) – The positions of cells to include in a sample. Default: Random position.
- **plot_tumour_growth** – Render graph of the tumour size as function

of time. Default: True. :type plot_tumour_growth: bool

Parameters **export_tumour** (*bool*) – Dump the tumour data to file. Default: True.

casim.casim.**check_set_number**(*value*, *typ*, *default=None*, *minimum=None*, *maximum=None*)
Checks if a value is instance of type and lies within permissive_range if given.

casim.casim.**load_cancer_simulation**(*dumpfile*)
Unpickle a cancer simulation from a dill generated dump. :param dumpfile: Path to the file that contains the dumped object. :type dumpfile: str

casim.casim.**main**(*arguments*)
The entry point for the command line interface.

Parameters **arguments** (*Namespace*) – The command line arguments for the cancer simulation tool.

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

Python Module Index

C

Index