Comments on the Phylogeography Project

A Note on Using the Cluster

My working directory in the Cluster was **/nfs/research1/goldman/kalkauskas/** and the files of the project are in **/nfs/research1/goldman/kalkauskas/Phylogeography/**.

I also worked initially to some extent in **/hps/research1/goldman/kalkauskas/** and in **/homes/kalkauskas/** but all of the important work was moved to **/nfs/research1/goldman/kalkauskas/**.

Any computationally intensive tasks on the Cluster (such as running simulations or installing Anaconda) should be submitted as jobs. To read about it please refer to the link bellow: https://tsc.ebi.ac.uk/article/systems-infrastructure/technical-wiki

Git Repository

The git repository is here: https://github.com/AntanasKal/Phylogeography

```
git clone https://github.com/AntanasKal/Phylogeography.git
```

Installing Anaconda

Anaconda has to be installed. I do not remember how exactly I did that, but here are notes by Claudia on installing Anaconda:

```
wget https://repo.continuum.io/archive/Anaconda3-5.1.0-Linux-x86_64.sh
#see https://www.anaconda.com/download/#linux
bash Anaconda3-5.1.0-Linux-x86_64.sh
#add to .bashrc, then log out and back in
```

Python libraries

Python libraries not included in Anaconda package that had to be installed:

- dendropy

- ercs

- discsim

- pymc3 (only needed to analyse output data, namely find HPD interval)

https://dendropy.org/

https://github.com/jeromekelleher/discsim

https://pypi.org/project/ercs/

Installing BEAST

In the cluster I installed BEAST so that I could run it from command line (by adding PATH variable in **./bashrc**) following this tutorial: https://beast.community/install_on_unix.

Then BEAST could be run with Python by:

```
beast -overwrite -seed 123456795 "beast_xml_file.xml"
```

But it might be easier to run BEAST from jar file. Just copy the **beast.jar** to a convenient directory and run:

```
java -jar beast.jar -overwrite -seed 123456795 "beast_xml_file.xml"
```

If needed, BEAGLE can also be installed, but I did not use it.

Note that Python code might need to be changed in order to run

Building PHYREX

To build PHYREX binary from source use the following code:

```
git clone https://github.com/stephaneguindon/phyml.git
cd phyml
sh ./autogen.sh
./configure --enable-phyrex
make clean
make
```

Afterwards in **src/** folder there will be **phyrex** binary file which can be placed in different different directory and it can be run by:

```
./phyrex --xml=file_name.xml
```

How to Use the Code

simulation.py

Script **simulation.py** is the main script of the code. It simulates the phylogeny of a tree and Brownian motion along it, subsamples the tree in 4 scenarios, generates the needed files for BEAST and launches BEAST on these 4 subsampling scenarios.

```
python simulation.py
```

Some of the parameters of the program:

- **-N**: number of simulations in sequence in one run.

- **-jobi**: index of a job. It is needed if we want to do a number of simulations in parallel and keep the track of files. So if **jobi** is j and **N** is n, the simulations will be done and files will be generated for indices

$$nj, \ nj + 1, \ ..., \ n(j + 1) - 1$$

  Usually I set **N** to 1 and iterate **jobi** from 0 to 99, so that 100 simulations would be done.

- **-dims**: number of dimensions (1 or 2) for which the random walk is generated (default: 2). **Note:** for 1 dimension some parts of code might not work so they need to be commented out. These are some parts of the code that generate some less important output.

- **-treetype**: type of tree generated.

  nuc - nonultrametric coalescent

  uc - ultrametric coalescent

  bd - birth-death tree

  yule - Yule tree

  (default is "yule")

- **-mcmc**: MCMC chain length for BEAST (default is 5000)

- **--linux**: whether the program is run on Cluster (different console commands are used the for executing BEAST).

- **-ntips**: number of tips for ultrametric coalecent, birth-death and Yule trees (default 100)

- **-ntipspp**: number of tips per period for nonultrametric coalescent trees (default 20).

- **-nps**: number of periods for nonultrametric coalescent trees (default 25).

- **--c_beast**: whether the program should generate input files (not output) for corrected BEAST.

Other parameters could be seen by:

```
python simulation.py -h
```

Shell script **run_beast.sh** runs 100 simulations on Cluster in parallel.

```
for i in $(seq 0 99)
do bsub -o console_output/out_"$i".txt -e console_output/err_"$i".txt\
python simulation.py -jobi "$i" -N 1 -dims 2\
-treetype yule -ntips 2000 --linux -mcmc 10000 --c_beast
done
```

These simulation should take not much longer than 10 minutes to run. To run corrected BEAST the script **launch_corrected_beast.py** is needed. It takes the files previously generated by **simulation.py** and runs BEAST on them. These runs could take about 10 hours. The script has these parameters:

- **-sample_index**: index of the sampling scenario (usually from 1 to 4)

- **-i**: index of the simulation (usually from 0 to 99)

The shell script **run_c_beast.sh** executes **launch_corrected_beast.py** on all the files generated.

```
for i in $(seq 0 99)
do bgadd -L 10 /c_beast/sim"$i"
bsub -g /c_beast/sim"$i" -o console_output/c_beast_out1_"$i".txt\
-e console_output/c_beast_err1_"$i".txt\
python launch_corrected_beast.py -index $i -sample_index 1
bsub -g /c_beast/sim"$i" -o console_output/c_beast_out2_"$i".txt\
-e console_output/c_beast_err2_"$i".txt\
python launch_corrected_beast.py -index $i -sample_index 2
bsub -g /c_beast/sim"$i" -o console_output/c_beast_out3_"$i".txt\
-e console_output/c_beast_err3_"$i".txt\
python launch_corrected_beast.py -index $i -sample_index 3
bsub -g /c_beast/sim"$i" -o console_output/c_beast_out4_"$i".txt\
-e console_output/c_beast_err4_"$i".txt\
python launch_corrected_beast.py -index $i -sample_index 4
done
```

treegenerator.py

Script **treegenerator.py** generates the trees in various scenarios.

- **treegenerator.generate_ultrametric_coalescent_tree(num_tips, lamb)**: returns ultrametric coalescent tree with **num_tips** of leaves and coalescent rate **lamb**.

- **treegenerator.generate_yule_tree(num_tips, br)**: returns a Yule tree with **num_tips** leaves and birthrate **br**.

- **treegenerator.generate_nonultrametric_coalescent_tree(num_tips_per_period, num_periods, period_length, lamb)**: returns nonultrametric coalescent tree with **num_periods** of periods, **num_tips_per_period** of tips per period, **period_length** of time between periods and coalescent rate **lamb**

- **treegenerator.generate_birthdeath_tree(num_extinct, br, dr)**: returns a subtree of a birth-death with leaves being the first **num_extinct** extinct nodes. Birthrate is **br** and deathrate is **dr**.

Other functions in the script that are useful:

- **treegenerator.simulate_brownian(t, sigma, dimension)**: simulates Brownian motion along the tree **t** for $\sigma =$ **sigma** and returns the tree with every node having attributes **X** (and **Y** if **dimension** is 2).

- **treegenerator.calculate_times(t)**: calculates times of each node (seed node has time 0) and returns the tree with each node having **time** attribute.

sampling.py

Script **sampling.py** is used to sample the tree according to different scenarios. The 4 ones that were mainly used are here:

- **sampling.sample_unbiased(tree, dimension=2, sample_ratio=0.1)**: returns subtree of **tree** with **sample_ratio** of tips taken uniformly at random.

- **sampling.sample_biased_most_central(t, dimension=2, sample_ratio=0.1)**: returns the subtree with leaves that are closest to the centre.

- **sampling.sample_biased_diagonal(tree, dimension=2, sample_ratio=0.1)**: returns the subtree with leaves that are closest to the diagonal.

- **sampling.sample_biased_extreme(tree, dimension=2, sample_ratio=0.1)**: returns the tree with leaves that have the largest x coordinate.

Running PHYREX

Script **run_phyrex.sh** runs PHYREX on the output files generated.

```
for i in $(seq 0 99)
do bgadd -L 10 /yule/phyrex"$i"
bsub -g /yule/phyrex"$i" -o console_output/phyrex_out1_"$i".txt\
-e console_output/phyrex_err1_"$i".txt\
./phyrex --xml=output/phyrex/sampled1/phyrex_input/phyrex"$i".xml
bsub -g /yule/phyrex"$i" -o console_output/phyrex_out2_"$i".txt\
-e console_output/phyrex_err2_"$i".txt\
./phyrex --xml=output/phyrex/sampled2/phyrex_input/phyrex"$i".xml
bsub -g /yule/phyrex"$i" -o console_output/phyrex_out3_"$i".txt\
-e console_output/phyrex_err3_"$i".txt\
./phyrex --xml=output/phyrex/sampled3/phyrex_input/phyrex"$i".xml
bsub -g /yule/phyrex"$i" -o console_output/phyrex_out4_"$i".txt\
-e console_output/phyrex_err4_"$i".txt\
./phyrex --xml=output/phyrex/sampled4/phyrex_input/phyrex"$i".xml
done
```

Other Scripts

Scripts **beastxmlwriter.py** and **phyrexxmlwriter.py** are used for writing BEAST and PHYREX input files respectively.

Output Folders

Output is organised in this way: **output** folder has subfolders **beast**, **c_beast**, **phyrex** which contain input and output files of BEAST, corrected BEAST and PHYREX. Each of these folders contains 4 folder for the results of 4 sampling scenarios. There are a few other folders, but they are not necessary. The **.trees.txt** files take lots of space and only the root locations were important for us. Processed root locations are put into **root_data/** folders and if I am downloading the data to my computer I usually delete the **.tree.txt** files by going to the folders containing them and executing this command:

```
find . -name "*.trees.txt" -type f -delete
```

Here is a rough scheme of the **output/** folder:

- **beast**
  - **sampled1**
    * **beast_input**
    * **beast_output**
    * **root_data**
  - **sampled2**
    * ...
  - **sampled3**
    * ...
  - **sampled4**
    * ...
- **phyrex**
  - **sampled1**
    * **phyrex_input**
    * **phyrex_output**
  - ...
- **c_beast**
  - ...

Discsim Scripts

If λ is the rate of events is ΛV model, μ is the death probability, L is the diameter of torus and r is the radius of sampling circle, then the rate at which sampling event occurs at given point is

$$\frac{\lambda \pi r^2}{L^2}$$

Given that a point is hit, the probability that it is displaced is μ. Also the mean square distance of one displacement is the mean square distance between two points chosen uniformly at random in the disc and it is r^2 (can be computed by integrating

$$\int_D \left(\int_D (x_1 - x_2)^2 + (y_1 - y_2)^2 \frac{1}{\pi r^2} dx_1 dy_1 \right) \frac{1}{\pi r^2} dx_2 dy_2$$

$$= \int_D \left(\int_D (x_1^2 + x_2^2 - 2x_1 x_2 + y_1^2 - 2y_1 y_2 + y_2^2) \frac{1}{\pi r^2} dx_1 dy_1 \right) \frac{1}{\pi r^2} dx_2 dy_2$$

$$= \int_D \left(\int_D (x_1^2 + x_2^2 + y_1^2 + y_2^2) \frac{1}{\pi r^2} dx_1 dy_1 \right) \frac{1}{\pi r^2} dx_2 dy_2$$

$$= \int_0^{2\pi} \int_0^r \int_0^{2\pi} \int_0^r (\rho_1^2 + \rho_2^2) \frac{1}{\pi^2 r^4} \rho_1 \rho_2 d\theta_1 d\rho_1 d\theta_2 d\rho_2$$

$$= \frac{4}{r^4} \int_0^r \int_0^r (\rho_1^2 + \rho_2^2) \rho_1 \rho_2 d\rho_1 d\rho_2 = \frac{4}{r^4} \int_0^r \int_0^r (\rho_1^2 + \rho_2^2) \rho_1 \rho_2 d\rho_1 d\rho_2$$

$$= \frac{4}{r^4} \int_0^r \left(\frac{r^4}{4} \rho_2 + \rho_2^3 \frac{r^2}{2} \right) d\rho_2 = \frac{4}{r^4} \left(\frac{r^6}{8} + \frac{r^6}{8} \right) = r^2$$

where D is a disc of radius r centred at 0 and observing that terms such as $2x_1 x_2$ integrate to 0).

So over a unit time, mean square displacement of an individual is

$$\frac{\lambda \mu \pi r^2}{L^2} r^2 = \frac{\lambda \mu \pi r^4}{L^2}$$

For Lemey model with diffusion matrix being identity, mean square displacement over a unit time is 2.

So having set λ, μ, r, in order to get ΛV process to be similar to Brownian motion as with diffusion rate being identity matrix we need to set

$$\lambda = \frac{2L^2}{\mu \pi r^4}$$

I have added the code to check this in Jupyter notebook "Phylogeography Output Analysis".

discs.py

This script runs a discsim simulation, generates BEAST and PHYREX files for that and runs BEAST. Currently it is set up in such a way that simulations indexed 0-99 are sampled from a square having opposite corners at (25, 25) and (75, 75), and simulations indexed 100-199 are sampled from a square having opposite corner at (45, 45) and (55, 55). It outputs the obtained root in folder **output/root_data/**.

Shell script **run_discs.sh** runs the 200 simulations:

```
bgadd -L 200 /discs
for i in $(seq 0 200)
do bsub -g /discs -o console_output/out_"$i".txt -e\
console_output/err_"$i".txt python discs.py -jobi "$i" -N 1
done
```

Shell script **run_phyrex.sh** runs PHYREX on all the simulations:

```
bgadd -L 200 /LV_phyrex
for i in $(seq 0 199)
do bsub -g /LV_phyrex -o console_output/phyrex_out_"$i".txt\
-e console_output/phyrex_err_"$i".txt\
./phyrex --xml=output/phyrex/LV/phyrex_input/phyrex"$i".xml
done
```

Other scripts

Script **newick.py** for converting directed trees outputted by discsim to newick trees. I edited this script after taking it from here:

https://github.com/tyjo/newick.py/blob/master/newick.py

Scripts **treegenerator.py** and **beastxmlwriter.py** in the folder are just copies of these scripts from Phylogeography folder.

Script **disc_phyrexxmlwriter.py** is edited script **phyrexxmlwriter.py** to write PHYREX xml scripts in this case.