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 σ = **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.

Discsim Scripts

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**
  - ...

This is an example of a short document typeset using LaTeX to show some features such as the inclusion of

- tables,
- figures, and
- single and multi-line equations.

This document is in no way a substitute for the many helpful resources which may be found online, of which a few suggestions are given at the end of the main part of this document.

Note that although LaTeX is well suited for mathematical typesetting, it is absolutely acceptable to write reports using other word-processing software (such as Microsoft Word or LibreOffice). See § 2.3 of the *Introduction* to the *Computational Projects Manual* for general guidance.

0.1 Finding a LaTeX program

If you decide to use a LaTeX, or you wish to try it out, LaTeX is available on the University's *Managed Cluster Service*. If you are going to use it extensively, then you will probably want to install LaTeX on your own personal computer. This can be done for free, e.g. for recommendations and packages see

- `http://www.tug.org/begin.html#install` and
- `http://www.tug.org/interest.html#free`

As a 'front-end' (i.e. 'clever editor'), Mac users will probably want to use TeXShop, while for Windows and Linux users (and Mac users) there is, *inter alia*, TeXworks; see the *Introduction* to the Part IB or Part II *Computational Project Manuals* for more details.

0.2 Setting up the document

At the start of your document you will need commands to set up the style, font-size, and other personal preferences. Many of these are *optional*, but the main body of text to be typeset must start with

```
\begin{document}
```

and end with

```
\end{document}
```

As indicated in the *Computational Projects Manual*, you are requested to leave a space 5cm by 11cm for a label at the top right hand corner of the front page of your project. The box at the start of this page was produced by the lines:

```
\hfill\framebox{\parbox[t][5 true cm][c]{11 true cm}
{\hfil Space for project label}}
```

0.3 Headings, sections and subsections

It is sensible to include headings and to organise your write-ups into sections, subsections and subsubsections, e.g. so that your answer to each question has a heading. For instance the document title was produced by

```
\begin{center}\LARGE\bf
  A Brief \LaTeX\ Guide for CATAM
\end{center}
```

where the `\` after the `\LaTeX` command is necessary to generate a space (you might like to experiment to see the effect of omitting the `\`).

LaTeX has specific commands to generate sections, subsections, subsubsections, etc. Moreover, it is possible to get LaTeX to number these automatically (the default). For instance,

- the first section heading in this document was produced by a line

  ```
  \section{This document}
  ```

- and, similarly, this subsection heading was produced by

  ```
  \subsection{Headings, sections and subsections}
  ```

In order to omit the numbering, append `'*'` to the command, e.g. `\section*{This document)`.

0.4 Numbering and labels

Objects such as figures, tables, equations, sections, and page numbers can be referred to using *labels*; the labels will be replaced with the correct reference when typeset by LaTeX. For example, we might wish to refer to a (sub)section by number, or to a program listing by page number (e.g. page **??** in this document), before knowing either what the section numbers is, or on which page the program will occur in the final typeset document.

Numbered [sub-]sections are *labelled* by adding a `\label` command after the [sub-]section name. For example, we can refer to § 0.9 below, because the subsection heading is followed by a label thus:

```
\subsection{Program listings and page number references}\label{proglist}
```

so allowing us to refer to the label `proglist` from elsewhere in the text by using, say, the reference '`\S\,\ref{proglist}`'.

0.5 Mathematical expressions

Mathematical expressions such as $2x + 5 - 3\sin x = 0$ can be included in-line by putting $ signs around the expression to tell LaTeX to treat it as mathematics (where the upright 'sin' is produced within the $ signs by using the command `\sin`; similarly for 'cos', etc.).

It is also possible to produce 'displayed' equations. The equation

$$|f'(x_*)| \equiv \left| \frac{3\cos x_* + k}{2+k} \right| > 1\,. \tag{1}$$

is produced using

```
\begin{equation}
  |f^{\prime}(x_*)| \equiv \left|\frac{3\cos{x_*}+k}{2+k}\right| > 1 \,.
  \label{picard3}
\end{equation}
```

The equation is numbered automatically, and has again been associated with a label in order to refer to it by number elsewhere in the text; thus the LaTeX

```
 'see equation (\ref{picard3})'
```

produces 'see equation (1)'.

Moreover,

- if the equations are subsequently re-ordered then, after a couple of invocations, LaTeX will successfully automatically correct the references;
- this use of labels also works for figures, tables, etc. (see below);
- if you do not wish to number your equation[s],
  - either append a '`*`' to `equation` in `\begin{equation} ... \end{equation}`,
  - or replace `\begin{equation} ... \end{equation}` with `\[ ... \]`.

Sometimes an equation will not fit on a single line, or there are multiple lines in an expression. LaTeX almost always has a solution to any typesetting problem. For instance. here is a more complicated example of a single numbered equation split over multiple lines:

$$\begin{aligned} x_N &= x_{N-1} - \frac{F(x_{N-1})}{F'(x_{N-1})} \\ &= x_* + \frac{F''\left(x_*\right)}{2F'\left(x_*\right)}\left(x_{N-1}-x_*\right)^2 + \dots\,. \qquad (2) \end{aligned}$$

This was produced by

```
\begin{eqnarray}
x_N &=& x_{N-1} - \frac{F(x_{N-1})}{F'(x_{N-1})} \nonumber\\
    &=& x_* + \frac{F^{\prime\prime}\left(x_*\right)}
                  {2 F^{\prime}\left(x_*\right)}
              \left(x_{N-1}-x_*\right)^2 + \dots \,.
\end{eqnarray}
```

Here, the `&` signs are used to align the equations – in this case at the equality signs. We also note that

- each line apart from the last must end with a 'new line' symbol `\\`;
- the `\nonumber` command can be used to tell LaTeX not to number that line.

0.6 Itemized lists

This document has a number of itemized lists. For instance

```
\begin{itemize}
\item first item;
\item second item;
  \begin{itemize}
  \item[$\circ$] second item, first sub-item;
  \item[$\circ$] second item, second sub-item.
  \end{itemize}
\end{itemize}
```

produces

- first item;
- second item;
  - ∘ second item, first sub-item;
  - ∘ second item, second sub-item.

It is also possible to generate numbered lists; e.g.

```
\begin{enumerate}
   \item This is the first line.
   \item This is the second.
   \item Here's the third.
\end{enumerate}
```

produces

1. This is the first line.
2. This is the second.
3. Here's the third.

0.7 Example of a figure

LaTeX has many ways of inserting graphs. In what follows we assume that the source file for the graph is in a sub-directory `Plots`, and that the source file is called

- `Figure-1.pdf` if the graph is in PDF form;
- `Figure-1.eps` (or `Figure-1.ps`) if the graph is in PostScript form.

If you are working with the `pdflatex` command on Linux (or with its equivalents on Windows/Mac), then a graph in PDF form may be preferable, although some modern versions of the `pdflatex`, or its equivalents, command also accept graphs in PostScript form. If you are working with the `latex` command, or its equivalents, then the graph should not be in PDF form but in, say, PostScript form instead.

The following LaTeX source *should* work with both the `pdflatex` and `latex` commands on Linux, or with its equivalents on Windows/Mac, and should insert the graph somewhere *near* here!

```
\begin{figure}[ht]\centering\label{q3fig1}
  \includegraphics[height=78mm,angle=0]{Plots/Figure-1}
  \caption{A caption can go here. Figures can be numbered automatically.}
\end{figure}
```

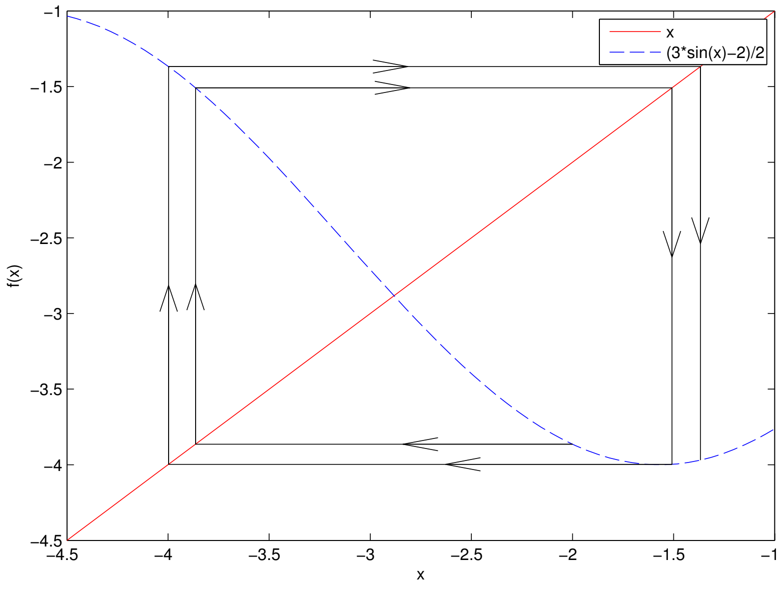

Figure 1: A caption can go here. Figures are numbered automatically.

Remarks

- Figures 'float', i.e. they are included when and where there is space; the `[ht]` specification means include the figure '**h**ere' or at the '**t**op of a page'.
- Most modern versions of LaTeX will append `.pdf` or `.eps` (or `.ps`) as appropriate to `Figure-1`; if your version does not do this then you will need to add the suffix explicitly by editing the `\includegraphics` line to refer to, say, `Plots/Figure-1.pdf`.
- Many `latex` commands that produce DVI output will accept many different graphics formats in addition to PostScript (again you may have to edit the `\includegraphics` line).

0.8 Example of a table

You may have the need to present the output of calculations in a table. The following commands produce the table which follows:

```
\begin{table}[ht]\centering\label{k=16:x=-2}
\caption{The table caption can go here.}\medskip\small
\begin{tabular}{ccccc}
 $N$ & $x_N$ & $\epsilon_N$ & $\epsilon_N/\epsilon_{N-1}$ &
 $f'(x_N)$\\[1mm]
   0 & -2.0000000 &  8.832369e-01 \\
   1 & -2.2071051 &  6.761317e-01 &  0.7655158 &  0.7898504 \\
   2 & -2.3736981 &  5.095388e-01 &  0.7536087 &  0.7689931 \\
&&&&\\[-4mm]
```

```
\vdots & \vdots & \vdots & \vdots & \vdots \\[ 2mm]
  32 & -2.8831971 &  3.981016e-05 &  0.7277548 &  0.7277554 \\
  33 & -2.8832079 &  2.897202e-05 &  0.7277545 &  0.7277549 \\
  34 & -2.8832158 &  2.108451e-05 &  0.7277543 &  0.7277546 \\
\end{tabular}
\end{table}
```

Table 1: The table caption can go here.

N	x_N	ϵ_N	$\epsilon_N/\epsilon_{N-1}$	$f'(x_N)$
0	-2.0000000	8.832369e-01		
1	-2.2071051	6.761317e-01	0.7655158	0.7898504
2	-2.3736981	5.095388e-01	0.7536087	0.7689931
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
32	-2.8831971	3.981016e-05	0.7277548	0.7277554
33	-2.8832079	2.897202e-05	0.7277545	0.7277549
34	-2.8832158	2.108451e-05	0.7277543	0.7277546

- As in the case of a `figure`, by using the `[ht]` specification, LaTeX will try to place the table near to where it occurs in the source file, or failing that at the top of page.
- The command `\begin{tabular}{ccccc}` tells LaTeX that there are 5 columns and the entries are to be <u>c</u>entred.
- Column entries are separated by ampersand `&`, and lines are terminated using `\\`.

0.9 Program listings and page number references

Program listings must be included at the end of your CATAM reports. These *either* can be included separately, *or* can be included via your LaTeX code. As an example we have included a program from a file called `question2.m` (located in the sub-directory `MATLAB`) at the end of this brief guide, on page **??**. The LaTeX code to do this is as follows:

```
\begin{center}\label{question:2:program}
   Program \texttt{program-2.m} for Question 2
\end{center}
{\small \verbatiminput{MATLAB/program-2.m}}
```

- The line `\label{question:2:program}` labels the page so that it can be referred to elsewhere in the text, say by ‘`see the code on page \pageref{question:2:program}`’ in which the label will be replaced automatically with the correct number, e.g. ‘see the code on page **??**’.

0.10 Some LaTeX references

Beginners, as well as more advanced users, may find some of the following references helpful (all working at time of writing):

- `http://physics.nyu.edu/%7Ephyslab/Lab%5FMain/Latexguide.pdf`
- `http://www.cs.princeton.edu/courses/archive/spr10/cos433/Latex/latex-guide.pdf`

- http://www.maths.tcd.ie/%7Edwilkins/LaTeXPrimer/
- http://tobi.oetiker.ch/lshort/lshort.pdf
- http://www.maths.adelaide.edu.au/anthony.roberts/LaTeX/index.php
- http://en.wikibooks.org/wiki/LaTeX/