Notes for Undergraduate Research Work

Hollis Bui

June 9, 2016

Contents

General

1 R Notes

Format of If/Else:

```
if {

}else{

}
```

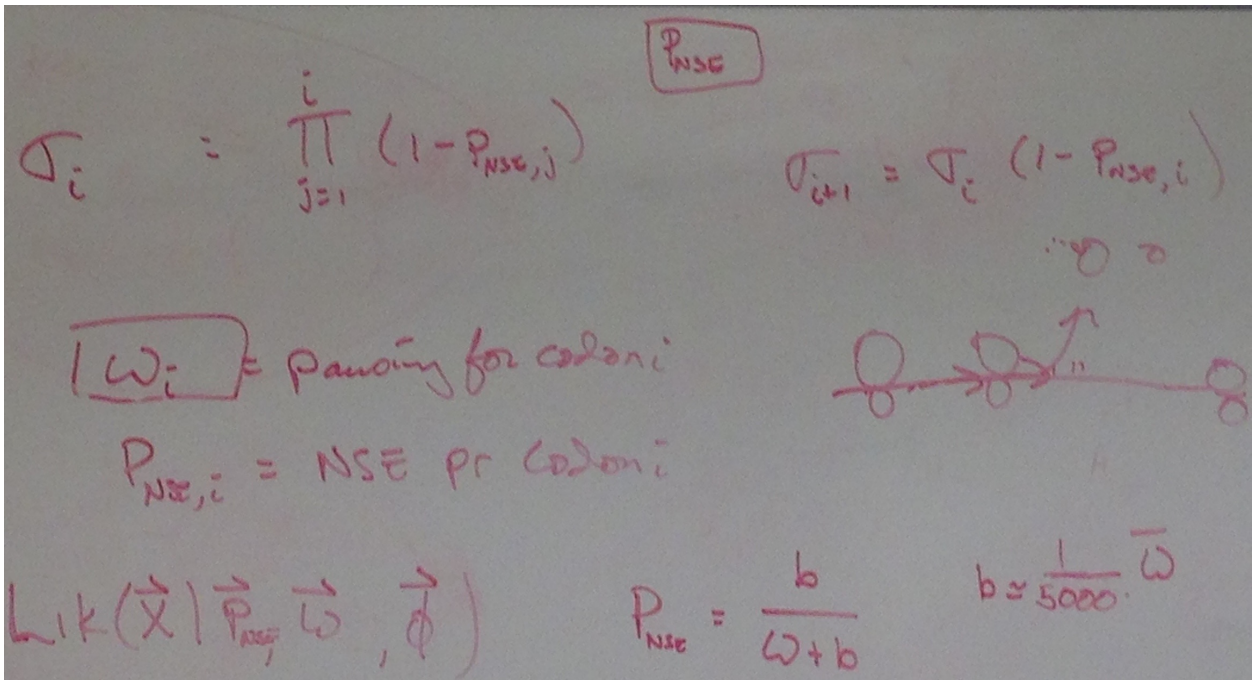
2 TODOs

1. PANSE model implentation:
   a) PANSEParameter.cpp
   b) PANSEModel.cpp
   c) PANSEParameter.h
   d) PANSEModel.h
   e) Ask about sigma term – Done
   f) Ask about lambda prime term (is it lambda prime?) — check RFP section for how to actually calculate — DONE
2. Expand Unit Testing:
   a) Test Cov Matrixes — STALLED: Still need final two
   b) Test MCMC - STALLED: Need run, varyInitialConditions, calculateGewekeScore, getLogLikelihoodPosteriorMean, and setRestartFileSettings as well as two test that only functions.
      - Implement other unit testing first
   c) Parameter – In progress
   d) Test RFP Parameter
   e) Test Trace
   f) ...Per class basis
   g) Eventually, some R scripts to do a short run for each model: Talk to Cedric
3. r

4. When working with gene-specific parameters, the openmp statements aren't working (memory is such a mess in the area) — break down parallelization, try to find where the issue is. Perhaps start with dynamic arrays, change to vectors. Gabriel thinks the slowdown from vectors in general is made up by better parallelization in avoiding dynamic arrays.
    - —STALLED. Literally can't test speeds of various optimizations and cores right now.
5. Documentation

May 13, 2016 Notes

1 PANSE Concepts



$$\sigma_i = \prod_{j=1}^{i} (1 - P_{NSE,j}) \tag{0.1}$$

ω_i = pausing for codon i
p_nse, i = NSE Pr (probability) for codon i
This is codon-based.
Likelihood of the data given the parameters: $\mathcal{L}(\vec{x}|\vec{P_{NSE}}, \vec{\omega}, \vec{\phi})$
Will be a much smaller data set, and with hundreds of calculations rather than thousands.
Randomly select ∼600 genes instead of 5400
Sigma vector of: $\sigma_{i+1} = \sigma_i(1 - P_{NSE,i})$
Function is of probability of getting there vs waiting time once there

2 TODOs

- Getting pausing values with simpler models (ROC)
- First analysis could be just estimating these terms

- This would mean creating a simulated data set.
- For simulation: $P_{NSE} = \frac{b}{\omega + b}$, where b is on the order of 1/5000 times average omega. ($b \simeq \frac{1}{5000}\overline{\omega}$) Talk to Jeremy about this, he may have finished this by now.

See the 2015 paper, 2011 paper with primal

May 19, 2016 Notes

1 PANSE Concepts

rfp.model.pdf: Reasoning [for lambda] is that for the sampling the Boltsman coefficient. See the explanation around equation (4) and the Z's and Y's.

Lambda Prime = Lambda.c * Z / Y, or call it K.

$\lambda' = \lambda_c * \frac{Z}{Y}$

Z is the overall state space

Y is what is sampled

$\lambda_c = \lambda' * \frac{C}{K}$. Let K be a new independent parameter, and keep track of Lambda Prime.

May 25, 2016 Notes

1 PANSE Concepts

Codon-Specific Elongation Rate: $P_{NSE} = \frac{b}{b+c}$ where b is where it flies off and c is where it continues.

Omega is the odds ratio of $\frac{P_{NSE}}{1-P_{NSE}}$. Therefore $\omega = \frac{b}{c}$

Look at 2006, 2007 papers.

LOOK AT UPDATED PDF: IT'S IN FRAMEWORK

Psi (the symbol which I *thought* was Omega) is the ribosome initiation rate: Rate at which ribosomes are jumping onto the mRA. Phi is the rate that they are jumping off at the very end.

If you have 50% chance to get to the end, then Psi is twice as long as Phi

Phi = Psi * Sigma.

Don't redo calculations from scratch, but rather in series.

2 Parallelization

- Only 20 AA's — Only 20 cores to spread load unto
- AA's with 6 codons of course take more time than those with 2

Gilchrist thinks what is meant by Gene-Specific Parameters is to parallelize at the highest level, i.e. at the gene or amino acid level.

I should check the code; find where the OpenMP statements are etc

Mostly something to ask other people about if I want to tackle the problem.

May 26, 2016 Notes

1 Parallelization

Cedric's input:

- phi calculation, with mcmc accept/reject
- dynamic arrays
- big loop around everything
- code doesn't work
- couldn't figure out why
- didn't spend that much time

we ended up parallelizing in the model class:
calculateLogLikelihoodRatioPerGene, apparently doesn't do much. Perhaps better to parallelize outside, with the big loop
Run a ROC model, then RFP
I'm running a fasta file that is simulated, so I know that it is true
I kinda need the R side
Get to the point where we suspect memory is the problem
Dynamic Arrays ->Vectors

May 31, 2016 Notes

- Start 1:21
- break 3:19
- back 3:24
- break 4:55
- return 5:02
- end 7:02

$2 + 1.5 + 2$

1 Parallelization

Go ahead and replace dynamic arrays with vectors, first

And then do this barebones calculation of runs to see if it makes it faster, without regards to parallelization.

June 1, 2016 Notes

- Start 1:30
- Break 3:30
- Return 3:35
- End 7:00

2 + 3.5

1 Parallelization

From yesterday:

```
0.00621732 - 10
0.00687881 - 100
0.00947537 - 1000
0.00713974 - 10000
0.00785908 - 10000
0.00750889 - 10
```

For today:

```
0.0572747 - 10
0.0698414 - 100
```

. . . Odd, 10x as long on average
The above was in DEBUG mode. Release mode redos:

A or V	Runs	Modifiers	Avg Time
V	100		0.0141421
A	100		0.0047742
V	10000		0.00850093
A	10000		0.00479609
V	10000	No Deletion	0.00871843
A	10000	No Deletion	0.00491614
V	10000	std::sort	0.00841396
A	10000		0.00598796
A	10000		0.00520682
A	100000		0.00455916
A	100000	std::sort	0.00776886
V	100000		0.00795495
V	100000	std::sort	0.00785736
A	100000	std::sort	0.00383634
A	100000		0.00385638
A	100000	std::sort	0.00392021

Note: Vectors are 2x as long on average now

2 PANSE Implementation

Next step: Make a list of everything PANSE touches and unit test these things (first and foremost before actually writing PANSE)

ALSO: Estimate and track how long, in reality, it takes to do each unit testing

PARFP, PTRFP? Just calling it RFP might be misleading.

June 2, 2016 Notes

- Start 1:01
- Break 3:35
- Return 3:50
- End 6:58

Spent till 4 (3 hours) compiling notes and creating a git directory.

1 PANSE Implementation

Expecting to spend 1 hour deciding on what PANSE will need (or, rather, what RFP will need).

Talk with Gilchrist:

So data position feeds into:

- a) data on gene
  - ab) to feed into ROC-RFP
- or b) PANSE-RFP

2 Lareau Data

Which file type should I be reading in? RFP or Fasta?

For sample data for PANSE:

Lareau Paper ->GSE ->The untreated replicates 1,2,3. Take one, and even then only a subset of one of them as sample data.

The Lareau material may have undergone more processing that the new Weinberg GSE published Feb 10 2016.

"Start with Lareau paper data" – Gilchrist, 5:33

June 3, 2016 Notes

- Start 1:35
- Break 4:09
- Return 4:14

1 Lareau Data

Decided to start reading the Lareau material. Began by looking directly at definition of data set (I chose untreated replicate 1) and then parse the data to get a smaller subset (file size otherwise is too large at 35MB)

Took longer than expected... When files finally parsed, 5:45.

Now have a data set of size 400 KB: those genes with 11 to 100 (inclusive) codons.

June 6, 2016 Notes

1 TODOs

Immediate future goals:

1. Generate new Lareau material following specifications of Gilchrist talk, below.
2. Just work, from now on, with the labbook class. Don't have to reformat old content.
3. Formally write up a list of things TODO with Unit Testing for Parameter
4. Unit Test up-to-date with Parameter
5. Write up pseudo-code with PANSE itself to prepare for it
6. Create and test a function for reading in Lareau material (low priority)
7. Parallelization is after the initial PANSE stuff is implemented, very low priority

2 Lareau Data

Talk with Gilchrist:
Let's get a randomly distributed set of data rather straight up isolation.
See below for how to randomly distribute; want only 100 genes.
61 Parameters Pausing Time
Lots of gene-specific parameters that scale with each gene.
Let's say average of each gene is 300 AAs.
So 7 observations per gene.
Try to get 2 parameters for a fair amount of information. Calculating at sigma is going increase at gene length.
And of course longer gene sequences take longer to parse.
So probably want a data base for playing around with of 100 genes, between 200 and 400 AAs long
Do we need to test with all 61 parameters? 2-codon AA's are the quickest thing to work with.
So may want to start with 100 genes of 200-400 AAs Estimate these parameters with a small subset of the codons, starting with the 2-codon ones. If they are behaving properly, scale up to 3/4/etc.

Long is de-facto standard Lareau argues that Short is also relevant despite usually being thrown out
Long and short: tell how elongation is at each position. Our model is based on pausing.
So how do long and short factor in? Well, we don't know yet.

We could base it on just one or the other or combine the two.
For now let's just base it on Long.

After about thirty minutes following the talk with Gilchrist – new subset of data produced via modifying old Perl scripts. Now have the specified data set in the final "finalData.txt" – 516 KB.

Interestingly small size – seems like old data set had that many genes of smaller AA length.

Spent an hour afterward reading over labbook documentation and reformatting notes where needed.

June 7, 2016 Notes

In the course of running an RFP Model, the following functions are called (and have yet to be unit tested).

1 TODOs

- initParameterSet (actually already done... mostly) – general parameter
  - test std_csp changes – Done
  - test numAcceptForCodonSpecificParameters changes – Done
  - Possibly setNumMutationSelectionValues – Ignore for now
  - Possibly initCategoryDefinitions –Ignore for now
  - For the two above – Find how to check delM and delEta of category (a vector of Mixture Definitions) – Done
  - Check many final changes at the end of this function – Three remaining
- initRFPParameterSet – RFP exclusive
- getSelectionCategory – general parameter – Done
- InitializeSynthesisRate – general parameter
  - calculateSCUO
  - quickSortPair
  - quickSort
  - Parameter::randLogNorm
- setParameter – RFP model exclusive
- mcmc.run – MCMC function on RFP(TODO later?)

2 Unit Testing

Going to take it one step at a time, finish up initParameterSet testing...

Finished most of initParameterSet completely. Need to ask Cedric about a duplicate function before finishing the final two functions.

May need to write a function to unit test with the categories variable itself, but all that happens otherwise is it pushes unto the vector of vector of vectors.

Encountering a strange printing bug right before the end. While if statement works correctly, the final confirmation of initParameterSet isn't being printed.

Example output:

```
Parameter getMixtureAssignment --- Pass
Parameter setMixtureAssignment --- Pass
Parameter getMutationSelectionState --- Pass
Parameter getNumParam --- Pass
Parameter getNumMixtureElements --- Pass
Parameter getStdDevSynthesisRate --- Pass
Parameter setStdDevSynthesisRate --- Pass
Parameter getCurrentStdDevSynthesisRateProposalWidth --- Pass
Parameter getNumAcceptForStdDevSynthesisRate --- Pass
Parameter getStdCspForIndex --- Pass
Parameter getNumAcceptForCspForIndex --- Pass
Parameter getNumMutationCategories --- Pass
Parameter getNumSelectionCategories --- Pass
Parameter getMutationCategory --- Pass
Parameter getSelectionCategory --- Pass
Parameter getMixtureElementsOfMutationCategory --- Pass
Parameter getMixtureElementsOfSelectionCategory --- Pass
Parameter getCategoryProbability --- Pass
Parameter setCategoryProbability --- Pass
Parameter getSynthesisRate --- Pass
Parameter setSynthesisRate --- Pass
Parameter getSynthesisRateProposalWidth --- Pass
0
Parameter initParameterSet --- Pass

Process finished with exit code 0
```

vs

```
Parameter getMixtureAssignment --- Pass
Parameter setMixtureAssignment --- Pass
Parameter getMutationSelectionState --- Pass
Parameter getNumParam --- Pass
Parameter getNumMixtureElements --- Pass
Parameter getStdDevSynthesisRate --- Pass
Parameter setStdDevSynthesisRate --- Pass
Parameter getCurrentStdDevSynthesisRateProposalWidth --- Pass
Parameter getNumAcceptForStdDevSynthesisRate --- Pass
Parameter getStdCspForIndex --- Pass
Parameter getNumAcceptForCspForIndex --- Pass
Parameter getNumMutationCategories --- Pass
Parameter getNumSelectionCategories --- Pass
Parameter getMutationCategory --- Pass
Parameter getSelectionCategory --- Pass
Parameter getMixtureElementsOfMutationCategory --- Pass
Parameter getMixtureElementsOfSelectionCategory --- Pass
Parameter getCategoryProbability --- Pass
Parameter setCategoryProbability --- Pass
Parameter getSynthesisRate --- Pass
Parameter setSynthesisRate --- Pass
Parameter getSynthesisRateProposalWidth --- Pass


Process finished with exit code 0
```

June 8, 2016 Notes

1 Unit Testing

Started by writing and testing a function for numAcceptForSynthesisRate.

Printing this statement seems to have fixed the odd print bug mentioned yesterday.

After doing that, I decided to write up on the documentation of Unit Testing I have done so far.

Asked Cedric about getSelectionCategory and getSynthesisRateCategory (paraphrased):

> It's related to how we may have delta and Phi values...
>
> We are trying to find out how efficient your codons have to be to reach a production rate assuming cost is constant.
>
> Cost = Benefit * production rate (Phi)
>
> Switching to a different selection environment: Different cost, different benefit, and therefore different Phi.
>
> If you have 2 selection categories, you have to have two synthesis categories. They are the same even if we don't know them, but it saves renaming it to something more general.

Finally finished initParameterSet besides checking categories matrix itself (minor, to do later).

Continued TODO:

2 TODOs

- initRFPParameterSet – RFP exclusive
- InitializeSynthesisRate – general parameter
  - calculateSCUO
  - quickSortPair
  - quickSort
  - Parameter::randLogNorm
- setParameter – RFP model exclusive
- mcmc.run – MCMC function on RFP(TODO later?)

Next goals: Do the internal functions for InitializeSynthesisRate related to quickSort.
Finish up InitializeSynthesisRate.

June 9, 2016 Notes

Began by checking github; fixed semicolon error and confirmed package worked correctly.
To truly finish testing InitializeSynthesisRate, we would need:

1 TODOs

- calculateSCUO
- Parameter::randLogNorm
- quickSortPair
  - pivotPair
    - swap (doubles)
- quickSort
  - pivot
    - swap (ints)

Ask Cedric if it'd be a good idea to just use std::sort instead of quickSort.
This is very math-intensive, so Unit Testing it may be lower priority.
For now, I am switching to unit testing RFP-exclusive functions.

- initRFPParameterSet (in RFPParameter.cpp)
  - check currentCodonSpecificParameter
  - check proposedCodonSpecificParameter
  - check lambdaValues (optional – currently unused variable)
  - check numParam – Done
  - check bias_csp – always set to 0, can ignore for now
  - check std_csp – Done
  - check groupList – Done
- setParameter (in RFPModel.cpp) – may be untestable, ignore for now

In the course of working on checking RFPParameter, wrote up Unit Testing for all Group List functions in Parameter.cpp

May need a wrapper function for currentCodonSpecificParameter and proposedCodonSpecificParameter extraction. Talk to Cedric.

Refocusing overall goals:

1. Unit Test up-to-date with Parameter

- initRFPParameterSet
  - check currentCodonSpecificParameter – stalled, ask Cedric
  - check proposedCodonSpecificParameter – stalled, ask Cedric
- InitializeSynthesisRate
  - calculateSCUO
  - Parameter::randLogNorm
  - quickSortPair
    - pivotPair
      - swap (doubles)
  - quickSort
    - pivot
      - swap (ints)
- mcmc.run – MCMC function on RFP

2. Write up pseudo-code with PANSE itself to prepare for it
3. Create and test a function for reading in Lareau material (low priority)
4. Parallelization is after the initial PANSE stuff is implemented, very low priority

TODO tomorrow:

Ask about current/proposed CSP Parameter, quicksort vs std::sort, and how (if) to unit test the more computation-intensive functions.

Finally replace cout statements in my own section of main.

Begin testing with MCMC run while waiting for others to arrive.