ThyrosimIM Workflow:

1. **Import patient data to Julia notebook**. I would recommend using the DataFrames.jl library
   1. Patient data should all be in terms of *days*. However the model is in terms of ­*hours* so make sure you adjust the timescale of the data either in Julia or in the original python notebook that pulls the data.
   2. Plotting the data will help you know if you have adjusted correctly. Make sure to check both lab and medications
2. **Parameters and initial conditions**. Ben initialized them in a very complicated way to allow parameter variation between different sex/BMI. For now, would recommend initializing them without patient specific variation and adding them back in later - makes fitting easier
3. **ODEs**. System to be solved. Takes in derivatives, state variables, parameters, and time.
4. **Loss function**. In order to fit with an optimizer, we try to *minimize loss* (model predicted – patient data). To do so, we have to come up with a loss function. I use log least-squares loss because we have state variables on different magnitude scales. If you normalize the model, you can use a non-log loss function.
   1. You **must** convert model output to FT3, FT4 when calculating loss (the data is for free hormones) Here are the correct functions for conversion. Do not try to modify them!

"""

Converts TT4 (μmol) to FT4 (μg/dL) using Thyrosim's internal 4th order polynomial

"""

function FT4(TT4::Float64)

    return (0.000289 + (0.000214 \* TT4) + (0.000128 \* TT4^2) - (8.83e-6 \* TT4^3)) \* TT4 \* 24281.25 \* .45  *# Convert from TT4 to FT4 then from μmol FT4 to μg/dL FT4 via scaling factor*

end

"""

Converts TT3 (μmol) to FT3 (μmol) using Thyrosim's internal 4th order polynomial

"""

function FT3(TT3::Float64)

    return (0.00395 + (0.00185 \* TT3) + (0.000610 \* TT3^2) - (-0.000505 \* TT3^3)) \* TT3 \* 28304000.35 \* 0.5

end

1. **Fitting Routine**. This, along with proper import of patient data, is most difficult. You have to pass *free parameters (*along with an initial guess for these free parameters)*, fixed parameters*, initial conditions, and data to the objective function (loss function) ***within*** the scope of the optimizer. To do so, you must pass the free parameters each time the optimizer runs (as they will be updated by the fitting function) and the rest only once.
   1. You also will probably need something to bound the parameters. You can do this manually via making loss infinity when parameters go out of bound or use Optim.jl’s bounded models.
   2. One of the tricky aspects is that you have to deal with patient medications using something called a *callback* function. These meds must be passed through the gut compartment by adding them to the 10th state variable u[10]. To keep it simple, I’d reccomend using a periodic callback with a period of 24 hours. You will have to adjust the meds data to fit this, but due to the half life of T4 and long time scale of the patient data it is ok to combine 2x daily doses into one larger dose or split up less frequent doses.
      1. Most patients change dose over the course of the data so we cannot pass dose to the function like Ben did via parameters. Instead, make sure the medication data is a global variable and then do a time-match on the date of the dose to find out how much needs to be added to u[10]
   3. As far as fitting algorithm, Nelder-Mead is the way to go initially. For parameter ID we will use Newton’s (or Newton family i.e. BFGS) to get an approximate inverse hessian which itself approximates covariance matrix. However to do this we need a different loss function (see 7 below)
2. **Plot loss/results.** To determine if things are working, easiest thing is to graph the fit over the data. Would recommend doing this throughout the fitting process to see how many iterations are actually needed for fitting. Plotting loss is good sanity check as well, but you will have to initialize loss as a global variable and save it at each step if you want to keep track of it during fitting as it is thrown out at each step normally.
3. **Parameter identifiability**. Once fitting looks ok (does not have to be perfect by any means, just tracks with general trends in T3, T4, TSH and fairly close to datapoints) we need to do another round of fitting using these good parameters. With this round, we need to use maximum likelihood estimation and add variance parameters with a new loss function. Ben has a good example in his code, and there may be an old version in mine. This allows us to track variance and thus get covariance estimate by fitting the new loss function with a Newton or Newton family method.

|  |  |
| --- | --- |
| Component | Issues / Todo |
| Patient Data | Convert everything to hours, either in OG python notebook before you import to julia or in julia itself. Make sure med dosing frequency is standardized at 24 hours and data corresponds accordingly. |
| Parameters and IC | I think my solution for free-fixed parameters is much easier to work with than Ben’s initialization process, but feel free to experiment. |
| ODEs | None as far as I know. Be careful if you modify as it can be hard to check ODEs are correct without re-writing everything and comparing parameter numbers. Current model is not perfect so consider alternatives (i.e. Meha’s) though they also have issues. |
| Loss functions | No problem with loss function itself, but you have to make sure everything is on the same timescale and units (i.e. everything in hours and FT4/FT3 not total hormones). The maximum likelihood function needs work. |
| Fitting routine | Callback needs to be validated. Feed everything into u[10] and do not modify the medication data frame when you do so.  Boundary conditions should be implemented in one way or another. Ben and I both bypass loss function and return INF loss if any parameters go out of bounds, but Optim.jl has functions to handle this more gracefully.  Be sure to write the fitting function *modularly*, do not try to make everything one function or it is impossible to see where the problem is. Also be sure to make it easy to change free and fixed parameters, do not have hard-coded parameters in any of the fitting functions. |
| Plotting | The plots look pretty good. You might have to modify them based on your other functions, though. |
| Parameter ID | Maximum likelihood loss function needs to be revamped. Note that you need to run several iteration of BFGS or Newton’s method even if your parameters are good from Nelder-Mead already – you aren’t trying to refit the model, but it takes time to adjust the hessian and get valid variability estimates. Some way to visualize this (i.e. heatmap) is pretty nice for sanity check. |