Explanation for CORPSE scripts

CORPSE\_array.py

* **expected\_params**: holds parameters (constants) to run the model. These can be varied to account for different microbial communities and soil properties
* **chem\_types**: defines the types of C used in the model
* **expected\_pools**: defines C pools to be used in the model assuming each of the chem\_types has a protected and unprotected pool
* **prot\_clay**: function that updates the protected C formation rate for different soils textures
  + Informed by constants from Table 3 in Mayes et al. 2012 (Relation between Soil Order and Sorption of Dissolved Organic Carbon in Temperate Subsoils)

Protection = Qmax (mg C kg-1 soil), converted to g m-3 using bulk density

Qmax = maximum DOC sorption capacity

Mayes et al. 2012: Figure 5 is observed and modeled relationship between Qmax and “the most significant independent variables for all pedons”

Resulting equation is log(Qmax) = 0.483 x log(%clay) + 0.009(Fe) + 2.328

Iron content is a relatively minor contribution so cut it.

Log(Qmax) = 0.483 x log(%clay) + 2.328

Raise each side to the 10 to the power of…

Qmax = 10^(0.483 x log(%clay) + intercept); Units: mg kg-1

Convert to g m-3 by multiplying by bulk density (g cm-3) and 10-6?

* check\_**params**: takes the parameters (params) entered in the Whitman\_sims.py script and compares them against expected\_params. If anything is missing, it raised an error.
* **CORPSE**\_deriv: Main model function
  + Inputs = SOM dictionary, temperature, theat, params, model for clay-driven protection

1. Convert theta and T into arrays with one value each.
   1. What is theta? Min = 0.4 and max = 0.9
   2. Theta controls soil moisture. Why is it based on a cosine function?
2. Constrain theta to between 0 and 1
3. Calculates decompRate (\*see below)
4. Calculates microbial turnover (kg yr-1)
5. Calculate maintenance respiration:

Where *et* is the fraction of microbial biomass turnover (death) that goes to necromass instead of being immediately mineralized to CO2

1. Calculate necromass production:
2. Calculate CO2 production:

Where *decompt* is the maximum decomposition rate for each C pool and *CUEt* is the carbon use efficiency for each C pool

1. Calculate microbial growth:
2. Update protected C turnover by dividing protected C by protection turnover rate
3. Update protected C production by multiplying unprotected C by protected C formation rate \* clay modifier.
4. Update total MBC pool by adding microbial growth and subtracting microbial turnover
5. Update protected and unprotected C pools.
   1. Unprotected = unprotected + protected turnover – protected formation
   2. Protected = protected – protected turnover + protected formation
6. Update necromass pool as necromass + necromassnew

* **decompRate**: Calculate maximum potential C decomposition rate
  + corrects Vmax for moisture
  + inputs = SOM, temperature, theta, params

1. Corrects units of vmaxref to be normalized for other factors so they are actually in year-1 units
2. Calculates Vmax for each C pool (function = Vmax)
3. “dodecomp” = check that there is >0 unprotected C, soil is not completely dry, and living MBC is present and create an array with True/False for each pool
4. Decomposition rate (drate) = Where C pools > 0, set decomposition rate using the following equation:

Where i is each pool in chem\_types (‘Fast’,’Slow’,’Necro’,’Py’]

“Decomposition rate is determined by a temperature-dependent maximum enzymatic conversation rate, the size of the unprotected C pool, and the ratio of microbial biomass to unprotected C. The dependence on microbial biomass is expressed in the form of a reverse Michaelis-Menten saturation. With this model form, the decomposition rate scales linearly with total C, as long as the ratio of microbial biomass to unprotected C remains constant.” (Sulman, Phillips, Oishi, Shevliakova, & Pacala, 2014)

* **Vmax**: calculates Vmax for each C pool at each time step. Vmax varies over the course of a simulation because Vmaxref and activation energy (Ea) are different for each pool and temperature varies over the course of a simulation
  + Vmax = vmaxreft \* exp(-Eat \* 1/(Rugas\*T) – 1/(Rugas\*Tref))
  + Rugas = molar gas constant, 8.312 J mol-1 K-1
  + Temperature in K
  + Tref = 293.15 K
  + Example:
    - Vmaxref = {Fast: 19, Slow:0.25, Necro: 10, PyC:1}
    - Ea = {Fast: 5e3, Slow:30e3, Necro:5e3, PyC:5e3}
* **sumCtypes**: Calculates sum of all C pools excluding MBC and CO2
  + unprotected and protected
  + fast, slow, necro, pyC

CORPSE\_solvers.py: Functions for running the CORPSE model.

Includes two approaches for running the model. One uses the python ordinary differential equation (ODE) solver. The other explicitly iterates the model using a fixed time step.

* run\_models\_ODE:
  + This is the function that actually runs the simulation using the ODE (ordinary differential equations) solver
  + Input:
    - Tmin & Tmax – allow sinusoidal variation in temperature
      * (set Tmin = Tmax for constant state)
    - Thetamin & Thetamax – allow sinusoidal variation in moisture
      * (set Thetamin = Thetamax for constant state)
    - Times – array of all the time steps (t) for the simulation
    - Clay – clay content (%) used to adjust protected C formation rates
    - Initvals – dictionary of the initial values for all pools (livingMBC, uFastC, etc.)
    - Params = dictionary of all parameter values (vmaxref, Ea, kC, gas\_diffusion\_exp, etc)
    - Inputs – dictionary of the C input rates of all pools
      * Assumes zero rate for pools not in the inputs data structure
      * In Whitman\_sims, this is left empty (i.e. no C inputs to system)
* Run\_models\_iterator:
  + Runs a simulation using the explicit iterator instead of the ODE solver
  + Can edit this function to allow more complex temperature and moisture patterns
    - i.e. input is an array of T and theta?
* totalCarbon:
  + Function for adding together all C pools

Whitman\_sims.py:

* SOM\_init – creates a dictionary with the initial pool values for SOM
  + Pools = fast, slow, necromass, PyC + protected/unprotected
  + Protected C = no microbial decomposition
* Params – input for parameters controlling the model
  + Vmaxref – relative maximum enzymatic decomposition rates for each C type
    - Units = year-1
  + Ea – activation energy for each C type (i.e. controls T dependence)
  + kC – Michaelis-Menton half saturation parameter
    - Units: g Microbial biomass / g substrate
  + gas\_diffusion\_exp – Rate at which gas diffuses through soil; controls suppression of decomposition at high soil moisture
  + substrate\_diffusion\_exp – Controls suppression of decomposition at low soil moisture?
  + minMicrobeC – minimum microbial biomass, prevents microbes from disappearing from model
    - Units: fraction of total C
  + Tmic – Mean microbial biomass lifetime
    - Units: years
  + Et – fraction of microbial biomass turnover that becomes necromass (vs. being mineralized to CO2)
  + Eup – microbial CUE for each substrate type (i.e. fraction of C from decomposition that becomes biomass; remainder is immediately respired as CO2)
  + tProtected – Protected C turnover time; this is the time scale for which protected C is released back to unprotected state
    - Units: year-1
  + protection\_rate – protected C formation rate; higher numbers mean that more unprotected C will become protected. This can be modified as a function of soil texture/mineralogy to represent different sorption potentisl
    - Units: years-1
  + new\_resp\_units – Changes units of vmaxref to aid in interpretation (leave as ‘‘True’’)