Tutorial 3 Creation human-microbiome whole-body models (mWBMs)

In this section we combine the newly created microbiome community models with the previously developed WBMs [4]. We will show the code required to create one mWBM. Later on we will explain which functions to use to generate multiple mWBMs with just one command.

First we need to load a microbiome model and a WBM model. Here we will load the microbiome model corresponding to sample ID CSM5MCXD and the female WBM.

```
% If you have not just run tutorial 1 or 2, or you have cleared your
% workspace, you will need to run the code below to define your directories
% again:
% resultPath = '';
% paths.General.metadataPath = '';
% paths.Mars.readsTablePath = '';
%
% paths.seqC.outputPathSeqC = [resultPath, filesep, 'resultSeqC'];
% paths.Mars.outputPathMars = [resultPath, filesep, 'resultMars'];
% paths.mgPipe.outputPathMgPipe = [resultPath, filesep, 'resultMgPipe'];
% paths.persWBM.outputPathPersonalisation = [resultPath, filesep, 'personalisedWBMs'];
% paths.mWBM.outputPathMWBM = [resultPath, filesep, 'mWBMmodels'];
% paths.fba.outputPathFluxResult = [resultPath, filesep, 'resultFlux'];
% paths.fba.outputPathFluxAnalysis = [paths.fba.outputPathFluxResult, filesep, 'fluxAnalysis'];
% paths.stats.outputPathStatistics = [resultPath, filesep, 'resultStatistics'];
```

```
% Load microbiome model. As we load it into a variable, we also extract the
% actual model
modelM = load([paths.mgPipe.outputPathMgPipe, filesep, 'Diet', filesep, 'microbiota_model_diet_

% Load the female WBM model
modelH = loadPSCMfile('Harvetta');
```

When combining the WBM with the microbiome model, the [u] section in the microbiome model is change to [luM] which stands for lumen microbiome. Instead of [e] the microbiome now exchanges metabolites with [LuLi] which is the lumen of the large intestine. Reactions are added to ensure metabolites unique to the microbiome model can be taken up from the diet and be excreted in the feces. The function used to combine a microbiome model with a WBM model is called combineHarveyMicrotiota and takes the following inputs

- modelH Whole-body metabolic model structure
- modelM Microbiome model
- couplingConstraint Coupling constraint for microbiome model

A coupling constraint in this context connects the metabolic activities of the WBM and microbiome model, ensuring that the exchange of metabolites is balanced and physiologically realistic. The value 400 is often used arbitrarily as a scaling factor to adjust the relative metabolic fluxes between the two systems, accounting for their differences in metabolic capacities. This ensures stability and consistency in the model's simulations.

Some warning might pop up:

- Reaction EX_biomass[c] is not in model
- The inserted Model contains an old style coupling matrix (A). The Matrix will be converted into a Coupling Matrix (C) and fields will be adapted.

These are expected and can be safely ignored

Now we can combine the models

```
% Set the coupling constraint to 400
couplingConstraint = 400;

% Combine the microbiome mdoel with harvey
mWBM = combineHarveyMicrotiota(modelH, modelM, couplingConstraint);
```

```
% Add additional fields to the HM model for more information
mWBM.sex = "female";
mWBM.version = modelH.version;
mWBM.name = modelM.name;
```

After the mWBM is created, we set the excretion of miciobiome biomass in the feces to 1 feacal exchange/ day [1,1]. This is done to ensure that production of microbiome biomass is the same for all models and that the amount of produced microbiome biomass does not influence the simulation results. We change the constraints of the reaction with the changeRxnBounds function which takes the following inputs:

- model - A metabolic model, in our case mWBM
- rxnNameList - A cell array of reactions that have to have their bounds changed.A cell array of reaction IDs or a string for a single reaction ID that species which reaction(s) need their bounds altered.
- value - A numeric value that say to which value the bound has to be changed to. Can be one value for all reactions, or one value for each reaction to be changed.
- boundType - A string, indicating which bound has to be changed, u= upper, l= lower, b = both.

We will set the inputs and change the bounds of the faecal microbiome biomass exchange.

```
% The name of the exchange of faecal microbiome biomass
rxnNameList = 'Excretion_EX_microbiota_LI_biomass[fe]';

% The value of the updated bounds
value = 1;

% Which bounds need to be adjusted, b for both
boundType = 'b';

% Change the bounds of the microbiome biomass fecal excretion to [1,1]
mWBM = changeRxnBounds(mWBM,rxnNameList, value, boundType);
```

Here we also define the diet that we want to use to simulate our human-microbiome models with. The default option is EUAverageDiet but other diet options are HighFiberDiet, HighProteinDiet, UnhealthyDiet and VegetarianDiet. If you want to change the diet use one of the afore-mentioned diet names and set that as the Diet variable istead of 'EUAverageDiet'. Using setDietConstraints we put the diet on the mWBM. The function also adds additional metabolites in small amounts (0.1-1 mmol/day). These extra metabolites are added as they are usually not present in the diet files, either pre-made or designed on vmh.life. The metabolites range from ions to small molecules required to ensure microbiome biomass production. The function takes the following inputs

- model - model structure, in our case mWBM
- diet - Diet option: 'EUAverageDiet' (default)
- factor - value between 0 and 1; default is 1, i.e, 100% of the provided diet

We will define our inputs and set the diet

```
% Set the chosen diet
diet = 'EUAverageDiet'
factor = 1;
mWBM = setDietConstraints(mWBM, diet, factor);
```

Now we have succesfully created a mWBM. The next step is testing if the mWBM is feasible.

Fixing infeasible mWBM

Now we will check if our create mWBM is feasible. That means can it produce both microbiome biomass and statisfy the body maintanance reaction. First we will set the bound of faecal exchange of microbiome biomass to 1 faecal exchange/day. This was done in line 123, but sometimes you would already have mWBMs created previously or they are obtained via another person. It is thus good practice to always reset the bounds on the microbiome biomass fecal exchange, even if it seems redudant.

```
% The name of the exchange of faecal microbiome biomass
rxnNameList = 'Excretion_EX_microbiota_LI_biomass[fe]';

% The value of the updated bounds
value = 1;

% Which bounds need to be adjusted, b for both
boundType = 'b';

% Change the bounds of the microbiome biomass fecal excretion to [1,1]
mWBM = changeRxnBounds(mWBM,rxnNameList, value, boundType);
```

The same goes for the human body maintanance reaction. We will reset it to one to ensure all models are comparable. More information on the 'Whole_body_objective_rxn' can be found in tutorial 2.

```
mWBM = changeRxnBounds(mWBM, 'Whole_body_objective_rxn', 1, 'b');
```

Now that the bounds have been reset we will set the objective of the model to optimise the faecal microbiome biomass exchange. Using changeObjective we can set the reaction we want to have optimised.

```
% Set the objective for excretion of microbiome biomass in the feces
mWBM = changeObjective(mWBM,'Excretion_EX_microbiota_LI_biomass[fe]');
```

Then we optimise the model with optimizeWBModel. More information on setting objectives and solving models can be found in tutorial 2.

```
% Solve the HM WBM
solution = optimizeWBModel(mWBM);
% Print the value of the solution
solution.f
```

If the sol.f value is 1, great! That means that the model can produce both enough microbiome biomass and can maitain the body maintanance reaction. To ensure we can use the model again we will also save it. The path to where mWBMs are saved was created in section 1 and saved in the paths variable.

```
% Save the model
save([paths.mWBM.outputPathMWBM,filesep,'mWBM_CSM5MCXD_female.mat'],'-struct','mWBM')
```

If the output is not 1, it means that the model is unable to create 1 microbiome biomass per day. We will check if the model is able to do it if we open up all dietary reactions. To find all dietary reactions we use the find and contain function to obtain all the reaction indexes that have the prefix Diet_EX_, which indicates dietary exchange reactions. We then use changeRxnBounds to set the lower bound of each dietary exchange reaction to -100000 the unconstrained value used in the WBMs. Then we solve the model again to see if the diet was the problem. The upper bound does not have to be set to 0, as any unused metabolite can pass from the diet to the fecal exchange via various compartments.

```
% Find all dietary reactions
idx = contains(mWBM.rxns,'Diet_EX');

% Open all dietary exchange reactions
modelHMOpen = changeRxnBounds(mWBM, mWBM.rxns(idx),-1000000,'l');
modelHMOpen.osenseStr = 'max';

% Solve the model
solOpen = optimizeWBModel(modelHMOpen);

% Print the solution value
solOpen.f
```

If the solOpen.f value gives 1 that means that means that we can solve the infeasibility by adapting the diet. which is coverd directly below. However if solOpen.f is not 1, the chosen diet is not the problem. You can either a) try another diet or b) check the microbiome explained in the next paragraph.

More often than not the cause of infeasibility is the absence of micronurtients and can be solved by adding the missing compound to the diet with a value of 0.1 mmol/day. To find out which compound needs to be added

we use the function *getMissingDietModelHM.m*. The function opens up the bounds on all dietary reactions to see if the model can be made feasible through dietary adaptations. If it is solvable with all dietary reactions open, the function will then close random dietary reactions in batches to try and narrow down which reaction is required for feasibility. It will continue to do this untill it has narrowed down a set a essential reactions. As the process is random, running the code twice on the same model might give differing results. If no dietary solution is possible it is good practice to double check that the germ-free (WBM withouth the microbiome) models and the microbiome models are individually feasible with the chosen diet. If the microbiome models are infeasible it might be good to check a) a different diet, b) any irregularities in the present_species file generated from MARS. If a solution cannot be found it is possible to open a issue on the googlegroups for support: https://groups.google.com/g/cobra-toolbox

The function getMissingDietModelHM takes the following inputs:

- WBM - A WBM model, in our case mWBM
- missingDietComponents - List of diet exchange reactions that are known to be missing or that have been identified in a previous run of this function. We set this to an empty cell array as we have no previous known identified reactions.
- testInitialFeasibility - Boolean to see if an initial feasibility test should be run. We set this to 0 as we already did this beforehand.

```
% Set inputs
dietExtensions = {};
testInitialFeasibility = 0;

% Find missing diet component and add to the model
missingDietComponents = getMissingDietModelHM(mWBM,dietExtensions,testInitialFeasibility);
```

```
% Add the missing components to the diet
mWBM.lb(matches(mWBM.rxns,string(missingDietComponents)))=-0.1;

% Save the updated mWBM
save([paths.mWBM.outputPathMWBM,filesep,'mWBM_CSM5MCXD_female.mat'],'-struct','mWBM');
```

Creating multiple mWBMs

In order to easily generate multiple mWBM models we need a file that matches the sex with the sample IDs. This will determine if the microbiome model created from that sample will be combined with a female (Harvetta) or male (Harvey) WBM. The diet we will use here is the EUAverageDiet. The metadata path, the path to the microbiome models and the path where the mWBMs should be stored were all defined at the beginning of this tutorial.

The function *createBatchMWBM.m* will create for all the microbiome models in the microbiome directory a mWBM. Given that the sample IDs matching to the microbiome models have a corresponding sex in the metadata. The code also checks if the any of the mWBMs are infeasible and then runs the *getMissingDietModelHM.m* function in a loop to ensure all the mWBMs that are created are feasible and all have the same dietary constraints. This allows for proper comparisons of the flux predictions later on. The function *createBatchMWBM.m* takes the following inputs:

- **microbiomeDir** - A string with the path to the directory where the microbiome models are stored, created via MgPipe. Created in section 1
- **saveDir** - A string with the path to the directory where the mWBMs should be stored. Created in section 1.
- **metadataPath** - A string with the path to the metadata file. Defined in section 1.
- **diet** - A string defining which diet should be used. Defaults to EUAverageDiet
- **numWorkersCreation** - Number of parrelel instances create mWBMs. Same value as used for MgPipe in section 3 is recommended. Defaults to 4.
- **numWorkersOptimisation** - Number of parrelel instances used to solve the mWBMs, defaults to 2. If the number of workers is too high, it can cause time-out errors. Reducing the number of workers should resolve that issue.
- **checkFeasibility** - A boolean to indicate if the function checks if the created mWBMs can grow on the the specified diet if set to true. Defaults to true.

Now we can set the inputs and run the function.

```
% Set the diet
diet = 'EUAverageDiet';

% The microbiome path was created already in the beginning of the tutorial
% and stored in the paths variable.
microbiomeDir = paths.mgPipe.outputPathMgPipe;

% The path where the HM models should be stored was already created in the
% beginning of the tutorial and stored in the paths variable.
mWBMdir = paths.mWBM.outputPathMWBM;

% Set numWorkers if not done so before by removing the % you can alter the
% value
% numWorkersCreation = 1;

% Set checkFeasibility to true
checkFeasibility = true;

% Set the number of workers for optimisation
numWorkersOptimisation = 2;

% Generate multiple mWBM models
createBatchMWBM(microbiomeDir, mWBMdir, paths.General.metadataPath, "diet",diet, 'numWorkersCre
    numWorkersCreation, 'numWorkersOptimisation', numWorkersOptimisation, "checkFeasibility", c
```

This concludes this tutorial on how to generate mWBMs. Please see the other tutorials on personalising the WBM models, solving mWBMs models and analysing flux results.

References (APA style)

[4] Thiele, I., Sahoo, S., Heinken, A., Hertel, J., Heirendt, L., Aurich, M. K., & Fleming, R. M. (2020). Personalized whole-body models integrate metabolism, physiology, and the gut microbiome. *Molecular systems biology*, *16*(5), e8982.