BacArena - An Agent-Based Modeling Framework for Cellular Communities

Eugen Bauer and Johannes Zimmermann

2020-05-20

Introduction to BacArena

Microbial communities are essential for global ecosystems and human health. Computational modeling of microbial consortia is thus a major goal in systems biology and microbial ecology. BacArena is a project to simulate bacterial behavior in communities. A lot of progress is done in the last years to create genome wide metabolic reconstructions of certain organisms, which open a wide field of mathematical analysis. One of this new methods is flux balanced analysis (FBA) to estimate optimal metabolic fluxes under certain constraints. Some of these models are available in an exchangeable format (SBML) and can be found in http://systemsbiology.ucsd.edu/InSilicoOrganisms/OtherOrganisms. The idea of this project is to use this existing reconstructions and put them in a spatial and temporal environment to study their possible interactions. This is achieved by the combination of agent based modeling with FBA. Each bacterium is considered as an agent with individual states, own properties and rules to act. Agents are located on a grid where they can move and interact via metabolic exchanges computed by FBA. The starting point for our project is curiosity of what could be done with these huge models. We just throw those models into an arena to see what kind of actions will evolve.

Getting started

Simple simulations with the *Escherichia coli* core metabolic model

First we have to load the installed package in the workspace with

```
library("BacArena")
```

Next we set-up the *Escherichia coli* core metabolic model

```
data("Ec_core")
```

The model is already integrated in the sybil R package by default. Alternatively you can also load your own model of interest with commands of the sybil and libSBML. After we loaded the model, we convert it into an object of class Bac by calling its constructor

```
bac <- Bac(Ec_core, limit_growth=FALSE)
```

```
## [1] "Median lower bound for non-zero and non-Inf exchanges is:-106"
```

```
## Loading required package: glpkAPI
```

```
## using GLPK version 4.65
```

(The parameter limit_growth=FALSE is set to disable growth limitation. This is important when space is limitting because bacteria can either to stop growing or accumulate a lot of biomass.) Now we have to set up an environment in which the organisms can interact

```
arena <- Arena(n=20, m=20)
```

Here, by default we chose 20 times 20 as the grid size of the environment. If not specified, the physical area of the environment is set automatically, in this case it is 0.005cm times 0.005cm. Details could be checked by

```
arena
```

Next we want to put our created organism in its environment by

```
arena <- addOrg(arena,bac,amount=5)
```

With this command we added 20 individuals of our bacterium. Now we can add the substances to the environment

```
arena <- addSubs(arena, smax=0.5, mediac="EX_glc(e)", unit="mM")
arena <- addSubs(arena, smax=1, mediac=c("EX_pi(e)", "EX_h2o(e)", "EX_o2(e)", "EX_nh4(e)"), unit="mM")
```

Here we added a minimal medium with 0.5 mM glucose. It is possible to check which substances were added at which amount by calling the *arena* object again

```
arena
```

```
## [1] "substances with 312.5 fmol per gridcell:"
## [1] "EX_glc(e)"
##
## [1] "substances with 625 fmol per gridcell:"
## [1] "EX_h2o(e)" "EX_nh4(e)" "EX_o2(e)"  "EX_pi(e)"
##
## [1] "arena grid cells: 20 x 20"
## [1] "arena grid size [cm]: 0.005 x 0.005"
## [1] "dimension of one grid cell [cm]: 0.00025 x 0.00025"
## [1] "area of one grid cell [cm^2]: 6.25e-08"
## [1] "flux unit: mmol/(h*g_dw)"
## [1] "1 mM in arena correspons to mmol/grid_cell: 6.25e-10"
## [1] "Arena of size 20 x 20 with 5 organisms of 1 species."
```

Finally, we can start *in silico* experiment simulating 12 hours of *E. coli* growth with

```
eval <- simEnv(arena,time=12)
```

During simulation, some basic statistics about growth will be printed. The object eval stores all 12 simulation steps. After we retrieve the eval object we can start analyzing the data. To get a simple overview about what happened we can look at substances having high variation

```
getVarSubs(eval)
```

```
##    EX_h2o(e)    EX_co2(e)     EX_o2(e)      EX_h(e)    EX_glc(e)    EX_nh4(e)
## 24662857867 14291804781 12904374161 10857594594   3220185174    787804655
##     EX_pi(e)     EX_ac(e)
##    358565207      1692507
```

To pick one substance and check its time series

```
getSubHist(eval, "EX_glc(e)")
```
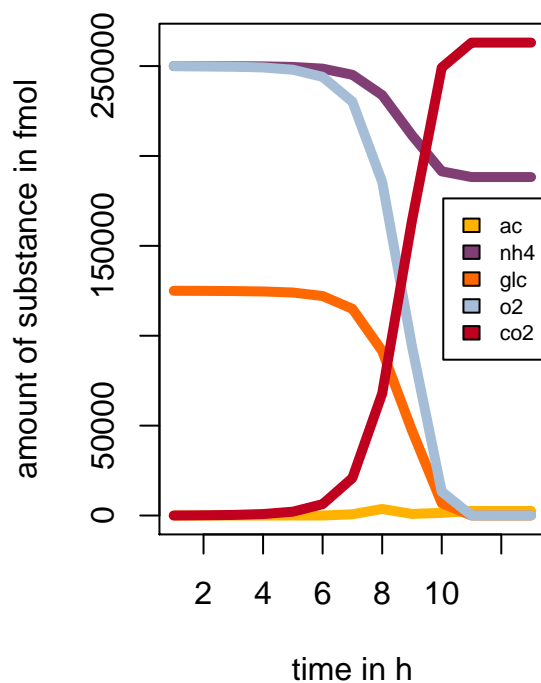
```
##              [,1]
## 1  125000.00000
## 2  124951.46756
## 3  124859.24151
## 4  124634.49124
## 5  124031.17212
```

```
## 6   122113.80271
## 7   114994.46568
## 8    91491.63331
## 9    47363.39769
## 10    6833.77746
## 11      17.08444
## 12      17.08444
## 13      17.08444
```
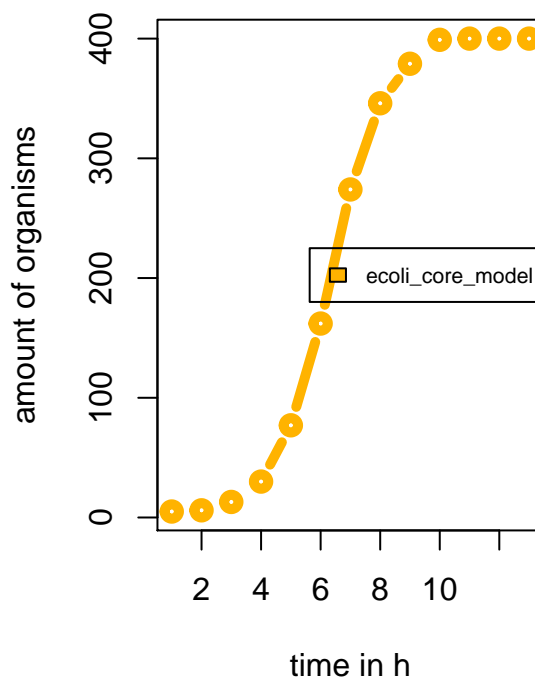
To start with graphical analysis let's investigate the growth curve in combination with the substance variations

```
par(mfrow=c(1,2))
plotCurves2(eval, legendpos = "right")
```
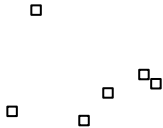

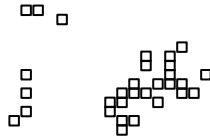Spatial and temporal changes can be shown via

```
par(mfrow=c(2,3))
evalArena(eval, show_legend = FALSE, time=seq(1,12,2))
```

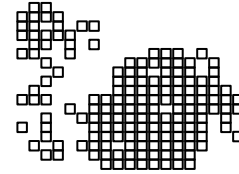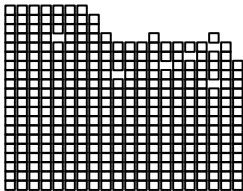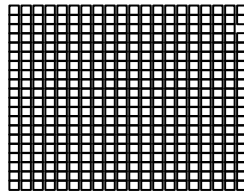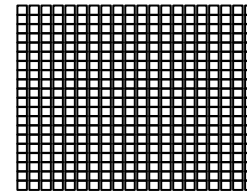**Population : # 2**   **Population : # 4**   **Population : # 6**

Population : # 8   **Population : # 10**   **Population : # 12**

This will produce multiple plots for each second simulation step with the spatial structure of the population (black dots represent individuals).

Simulation of multiple organisms

Now we want to multiple organisms or organism types in the environment. For this we create two different types of the *Escherichia coli* core metabolic model: A wild type *E. coli* and an auxotrophic mutant which is unable to use aerobic respiration.

```
bac1 <- Bac(Ec_core,type="ecoli_wt")
```

```
## [1] "Median lower bound for non-zero and non-Inf exchanges is:-106"
```

Now we create the auxotrophic mutant by using basic commands of the sybil package.

```
ecore_aux <- changeBounds(Ec_core, "EX_o2(e)",lb=0)
bac2 <- Bac(ecore_aux,type="ecoli_aux", setExInf=FALSE)
```

```
## [1] "Median lower bound for non-zero and non-Inf exchanges is:-106"
```

Again we set up an environment and insert organisms and substances

```
arena <- Arena(n=20, m=20)
arena <- addOrg(arena,bac1,amount=5)
arena <- addOrg(arena,bac2,amount=5)
arena <- addSubs(arena, smax=0.5, mediac="EX_glc(e)", unit="mM")
arena <- addSubs(arena, smax=1, mediac=c("EX_pi(e)", "EX_h2o(e)", "EX_o2(e)", "EX_nh4(e)"), unit="mM")
eval <- simEnv(arena,time=10)
```

Here we put the both organism types we created next to each other (given by their x position) in the environment and then started the simulation for 10 time steps. Next we perform again all evaluation steps

```
par(mfrow=c(1,2))
plotCurves2(eval)
```

Strongly changing substances
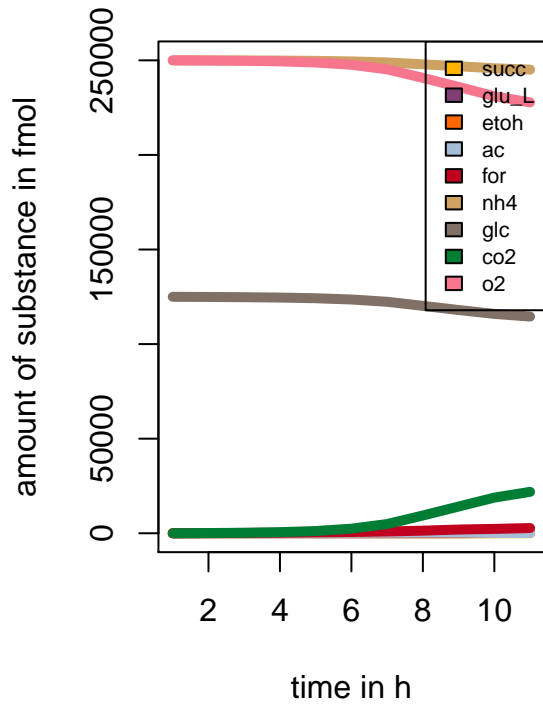
Growth curve
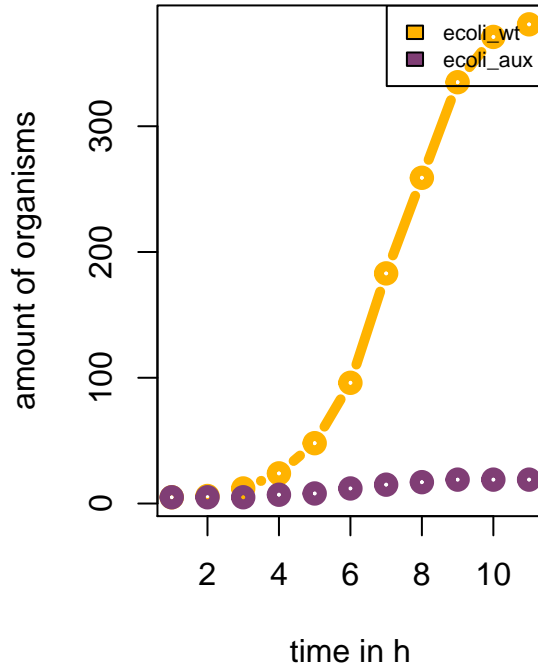
And the spatial pattern of the community with the substances glucose, oxygen, and ethanol plotted by different ways:

```
par(mfrow=c(1,4))
evalArena(eval,c("Population","EX_glc(e)","EX_o2(e)","EX_etoh(e)"), time=10)
```

EX_etoh(e) : # 11

EX_glc(e) : # 11

EX_o2(e) : # 11

Population : # 11
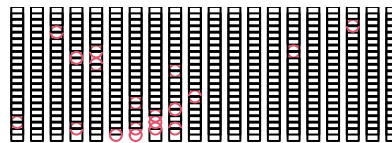
```
plotSubDist2(eval, sub = c("EX_etoh(e)"), times = c(1,5,10))
```


EX_etoh(e)
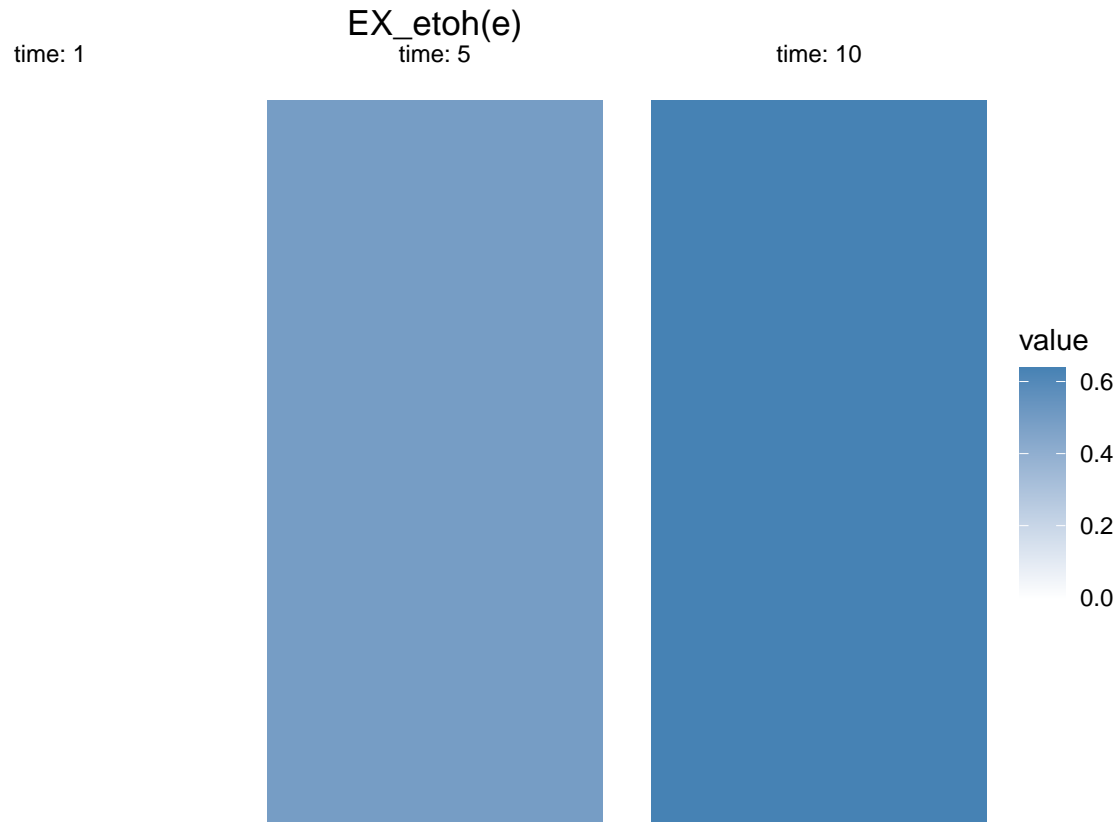
Here different point colors in this first figure indicate the two different organism types. Finally we can have a look on phenotypes. The command will create a PCA plot with the similarity of the different phenotypes. If we are interested in the definition of the phenotypes we can also retrieve the original phenotype matrix

```
minePheno(eval)
```
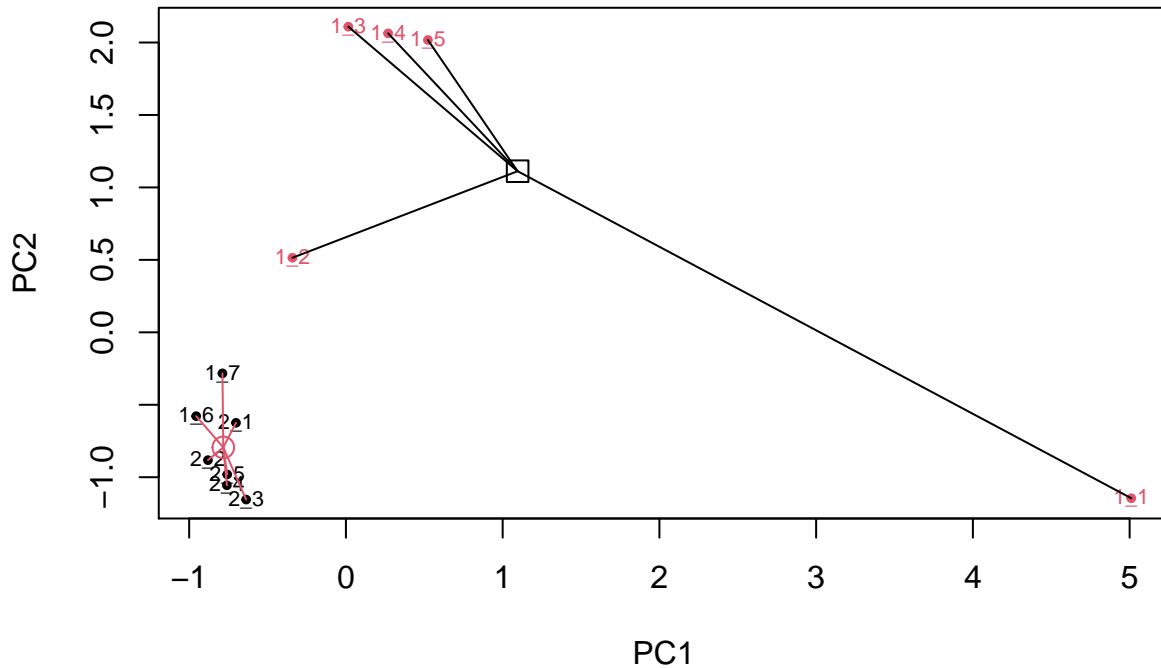
```
pmat <- getPhenoMat(eval)
pmat[,which(colSums(pmat)>0)]
```

```
##              EX_ac(e) EX_acald(e) EX_akg(e) EX_co2(e) EX_etoh(e) EX_for(e)
## ecoli_wt.1         1           2         1         1          1         1
## ecoli_wt.2         0           0         0         1          0         0
## ecoli_wt.3         2           0         0         1          2         0
## ecoli_wt.4         2           0         0         1          2         0
## ecoli_wt.5         2           0         0         1          2         0
## ecoli_aux.1        1           0         0         2          1         1
## ecoli_aux.2        1           0         0         0          1         1
## ecoli_aux.3        1           0         0         2          1         1
## ecoli_aux.4        0           0         0         2          1         1
## ecoli_aux.5        0           0         0         1          1         1
## ecoli_aux.6        0           0         0         2          1         1
## ecoli_aux.7        0           0         0         1          1         1
##              EX_fru(e) EX_glc(e) EX_gln_L(e) EX_glu_L(e) EX_h(e) EX_h2o(e)
## ecoli_wt.1           2         0           2           2       1         2
## ecoli_wt.2           0         2           0           0       1         1
## ecoli_wt.3           0         2           0           0       1         1
## ecoli_wt.4           0         2           0           1       1         1
## ecoli_wt.5           0         2           0           2       1         1
## ecoli_aux.1          0         2           0           0       1         2
## ecoli_aux.2          0         2           0           0       1         2
## ecoli_aux.3          0         2           0           1       1         2
## ecoli_aux.4          0         2           0           1       1         1
## ecoli_aux.5          0         2           0           1       1         2
## ecoli_aux.6          0         2           0           1       1         1
## ecoli_aux.7          0         2           0           1       1         2
##              EX_lac_D(e) EX_mal_L(e) EX_nh4(e) EX_o2(e) EX_pi(e) EX_pyr(e)
## ecoli_wt.1             2           2         1        2        2         1
## ecoli_wt.2             0           0         2        2        2         0
## ecoli_wt.3             0           0         2        2        2         0
```

```
## ecoli_wt.4              0          0          2       2       2       0
## ecoli_wt.5              0          0          2       2       2       0
## ecoli_aux.1             0          0          2       0       2       0
## ecoli_aux.2             0          0          2       0       2       0
## ecoli_aux.3             0          0          2       0       2       0
## ecoli_aux.4             0          0          2       0       2       0
## ecoli_aux.5             0          0          2       0       2       0
## ecoli_aux.6             0          0          2       0       2       0
## ecoli_aux.7             0          0          2       0       2       0
##              EX_succ(e)
## ecoli_wt.1            1
## ecoli_wt.2            0
## ecoli_wt.3            0
## ecoli_wt.4            0
## ecoli_wt.5            0
## ecoli_aux.1           0
## ecoli_aux.2           0
## ecoli_aux.3           0
## ecoli_aux.4           0
## ecoli_aux.5           1
## ecoli_aux.6           1
## ecoli_aux.7           0
```

Based on these results we can see, that the auxotrophic organism type grows slower in general and uses just fermentation of glucose, whereas the the wild type can respire glucose with the aid of oxygen. We can also create customized microbial communities or multicellular systems by importing external SBML models using the readSBMLmod function in the sybilSBML package.

Advanved

Simulation with replicates

BacArena supports parallel computing, e.g., there is a parallelized *simEnv_par()*. Additionally, several simulations could be run in parallel to get replicates for better validation of the results. This could be done via the parallel package which supports parallelism in Linux, MacOS and Windows. Therefore, the basic BacArena methods, introduced in the last section, are used within a parallelized lapply

```
library(parallel)
replicates <- 2
cores <- ifelse(detectCores()>=2, 2, 1)
cl <- makeCluster(cores, type="PSOCK")
clusterExport(cl, "Ec_core")
simlist <- parLapply(cl, 1:replicates, function(i){
  bac   <- BacArena::Bac(model=Ec_core)
  arena <- BacArena::Arena(n=20, m=20)
  arena <- BacArena::addOrg(arena, bac, amount=10)
  arena <- BacArena::addSubs(arena, smax=0.5, mediac="EX_glc(e)", unit="mM")
  arena <- BacArena::addSubs(arena, smax=1, mediac=c("EX_pi(e)", "EX_h2o(e)", "EX_o2(e)", "EX_nh4(e)"),
  sim   <- BacArena::simEnv(arena, time=5)
})
```

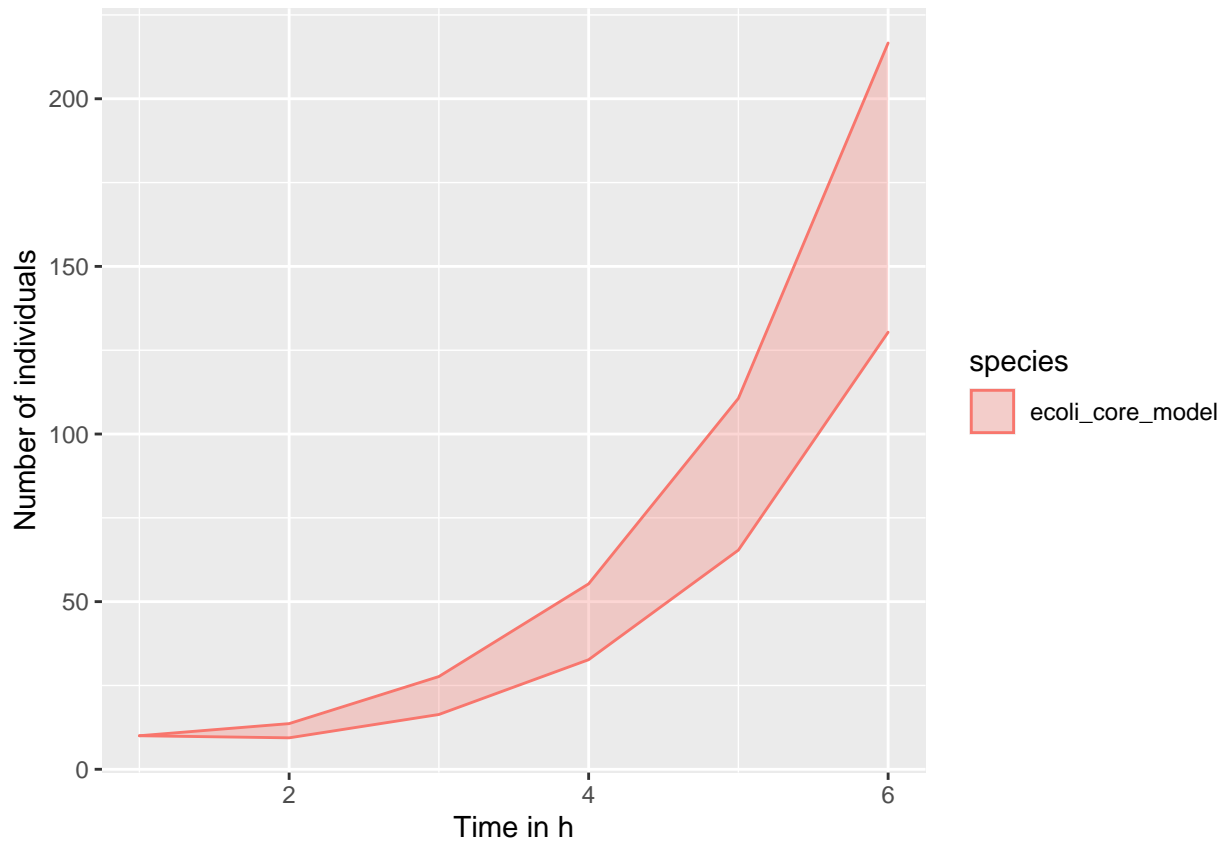We can obtain a growth curve with standard deviation

```
p <- plotGrowthCurve(simlist)
```

```
## Warning: `fun.ymin` is deprecated. Use `fun.min` instead.
```

```
## Warning: `fun.ymax` is deprecated. Use `fun.max` instead.

## Warning: `fun.y` is deprecated. Use `fun` instead.

## Warning: `fun.y` is deprecated. Use `fun` instead.
p[[2]]
```
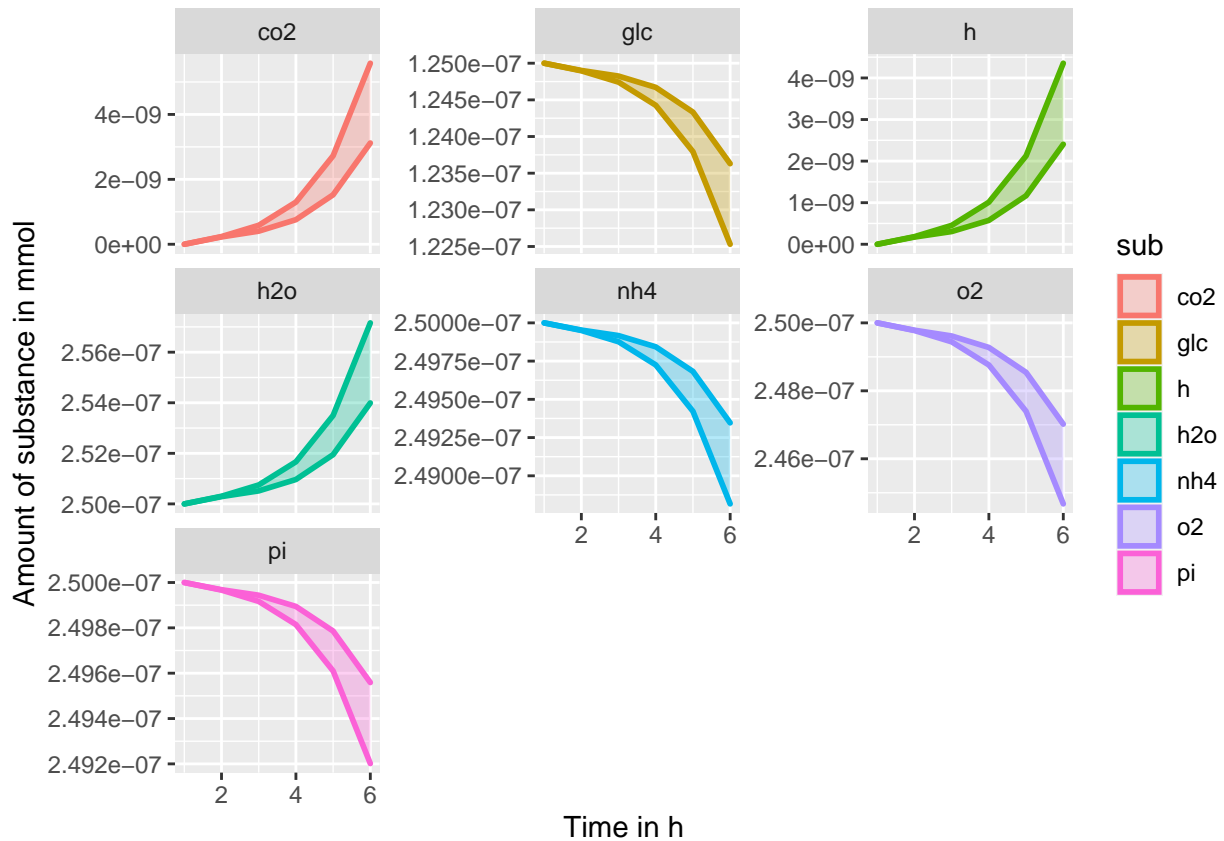

Additionally, there is substance time curve with standard deviation, too

```
p <- plotSubCurve(simlist)
```

```
## Warning: `fun.ymin` is deprecated. Use `fun.min` instead.

## Warning: `fun.ymax` is deprecated. Use `fun.max` instead.

## Warning: `fun.ymin` is deprecated. Use `fun.min` instead.

## Warning: `fun.ymax` is deprecated. Use `fun.max` instead.

## Warning: `fun.y` is deprecated. Use `fun` instead.
p[[3]]
```

Community analysis

BacArena was developed with focus on microbial communities. Simulation results are stored in object of class *eval*. Therefore, simulation could be separated from analysis and simulation files easily exchanged. To test the available methods for community analysis, BacArena provides a sample simulation file containing a 10 hour simulation of 8 gut microbes (SIHUMI community) fed with a standard diet (Altromin 1310). The test data is called *sihumi_test*

```
data("sihumi_test")
```

To get a summary of the growth, we start analysis by looking on the abundances of each species over time

```
p <- plotAbundance(sihumi_test)
p + ggplot2::scale_fill_manual(values=colpal2) + ggplot2::scale_color_manual(values=colpal2)
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```
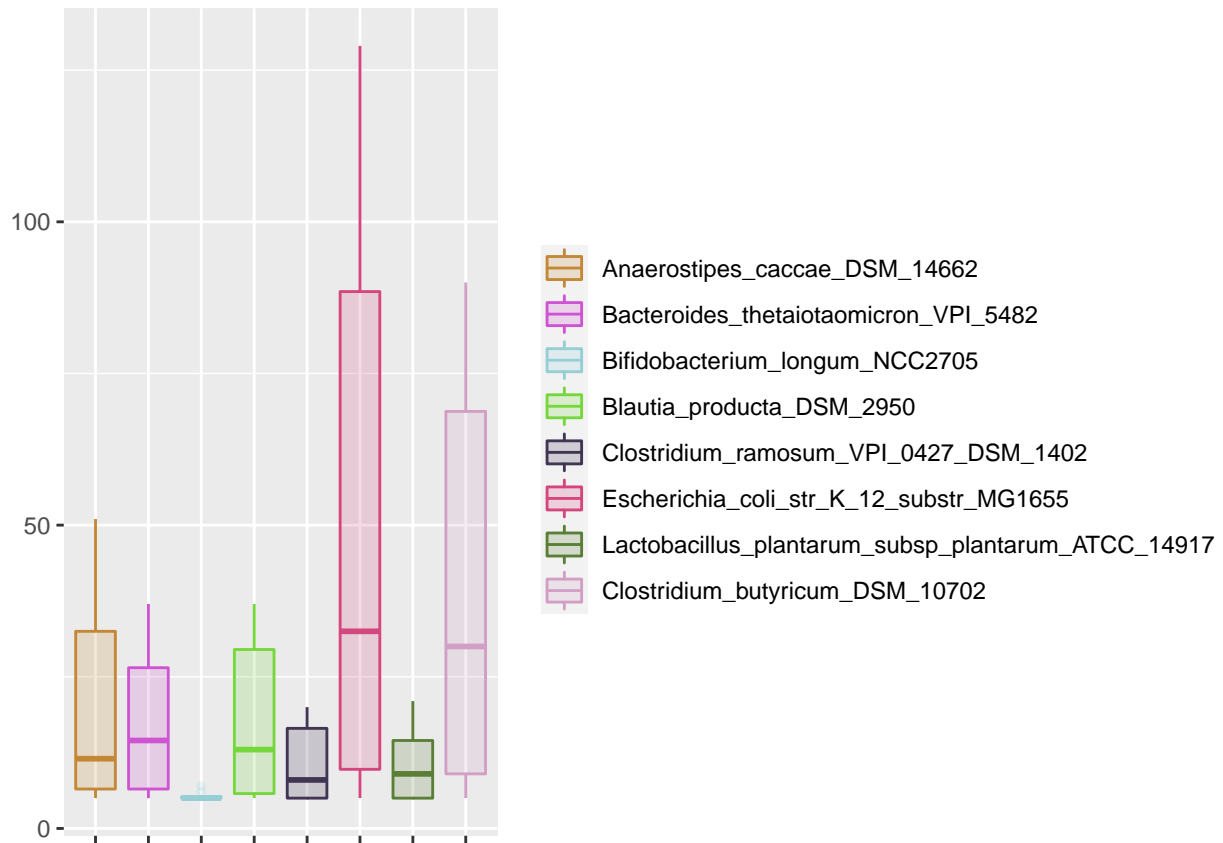
The methods *plotAbundances()* returns a ggplot object which can be modified, e.g., to use different colors.

It is quite to investigate the metabolic activity of all organisms. We obtain a figure with uptake (negative values) and production (positive values) patterns of substances which changed most during simulation by

```
plotSpecActivity(sihumi_test)[[2]]
```
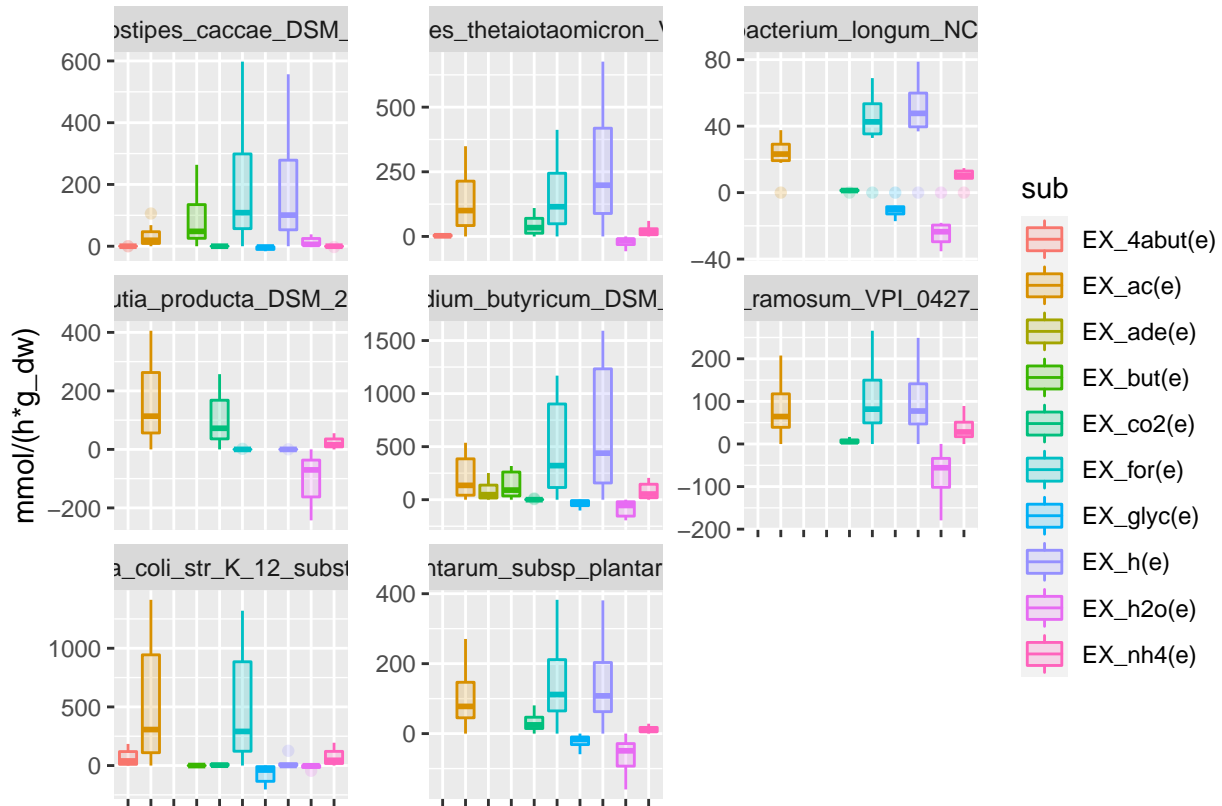
We see for example that succinate is produced by *B. producta*, *C. ramosum*, and *B. thetaiotaomicron* and is used by all other species except *B. longum*. Furthermore, the concentration of lactate is increased by for example *Lactobacillus plantarum* and again lactate supports growth of other species such as *B. thetaiotaomicon*. Moreover, we can identify *A. caccae* and *C. butyricum* as butyrate producers which is a short chain fatty acid (SCFA).

We can also try to find cross feeding relationship automatically

```
g <- findFeeding3(sihumi_test, time = 5, mets = c("EX_lac_D(e)", "EX_etoh(e)") )
```
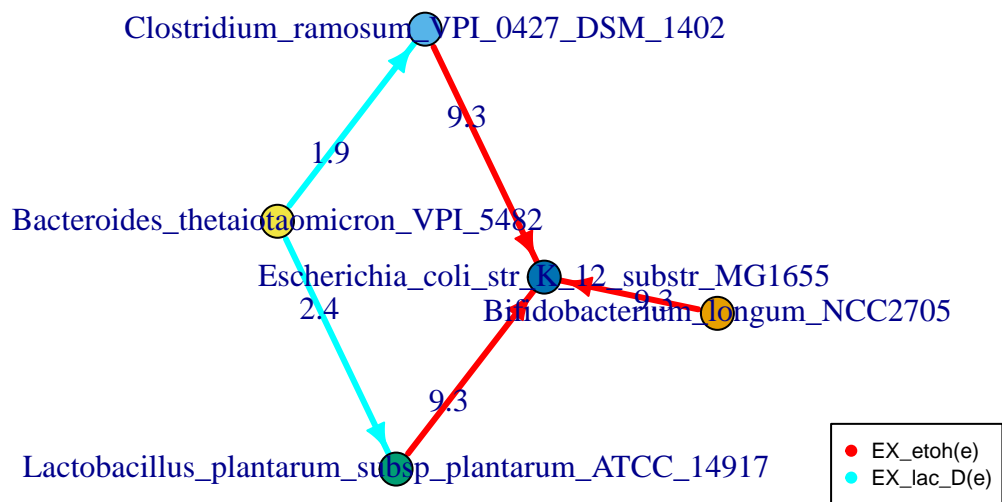

This graph can be interpreted as an ecological food web.

The main components of the diet are protein, starch, sucrose, oil, cellulose. Afterwards, we study the consumption of diet components (without amino acids) per species

```
plotSubUsage(sihumi_test, subs = c("EX_sucr(e)", "EX_cellb(e)", "EX_ocdca(e)"))[[2]]
```
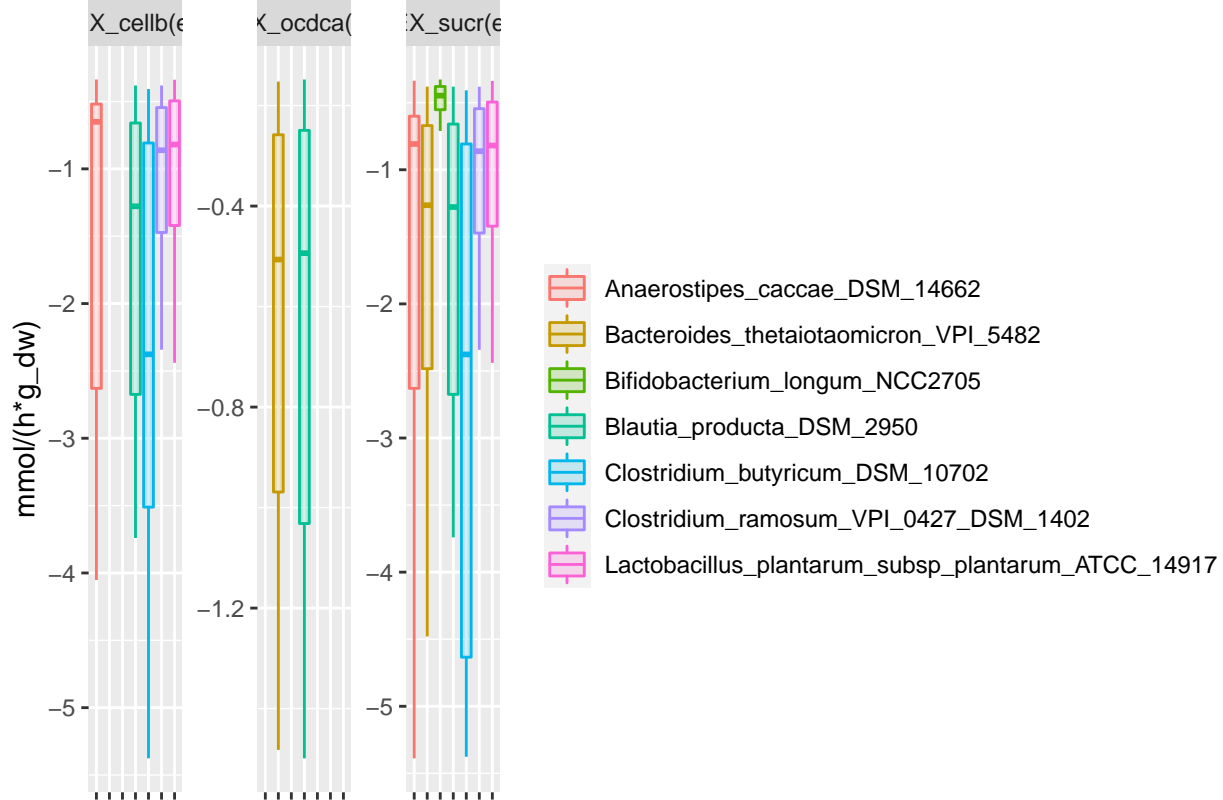

Whereby sucrose is used by all organisms but the the stearic acid only by *B. thetaiotaomicron* and *B. longum*.

We can also have a look at components which are missing in the diet and which could contribute to growth when added in higher amounts:

```
plotShadowCost(sihumi_test, spec_nr=7)[[2]]
```

```
## Warning: Use of `df$sub` is discouraged. Use `sub` instead.

## Warning: Use of `df$sub` is discouraged. Use `sub` instead.

## Warning: Use of `df$sub` is discouraged. Use `sub` instead.

## Warning: Use of `df$shadow` is discouraged. Use `shadow` instead.
```

Lactobacillus_plantarum_subsp_plantarum_ATCC_14917