TP 1

Sylvain Prigent

sylvain.prigent@inra.fr
UMR 1332 BFP - Équipe métabolisme

Novembre 2018

Des slides

À télécharger ici :
`http://176.31.120.7/TP_final_MOCELL.html`

CobraPy : LE package python pour réaliser de la modélisation de réseaux métaboliques

Package de modélisation de réseaux métaboliques
Initialement développé pour Matlab (COBRA toolbox), c'est aujourd'hui totalement fonctionnel pour python

Installation :

```
sudo pip install cobra pytest pytest-benchmark
```

Ou si vous n'avez pas les droits...

```
pip install --user cobra pytest pytest-benchmark
```

Fonctionne avec Python 2.7 ou Python 3, je vous laisse choisir

CobraPy : LE package python pour réaliser de la modélisation de réseaux métaboliques

Tester l'installation

```
from cobra.test import test_all
test_all()
```

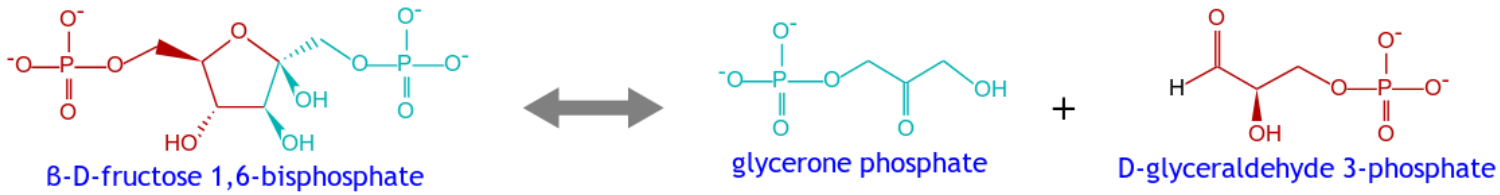
Probablement quelques erreurs, notamment lié à la lecture/écriture de fichiers sbml, on y reviendra plus tard

Chargeons cobra et... c'est parti !

```
from cobra import *
```

Création de la première réaction

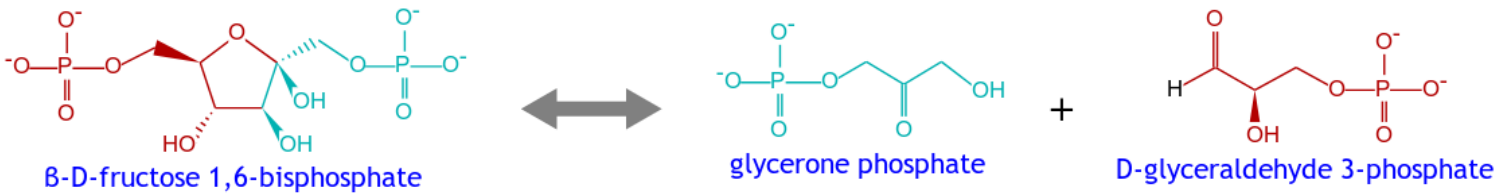
La fructose-bisphosphate aldolase



Une réaction sera composée de métabolites, qu'il faudra donc créer

- Soit à la main
  - Un id
  - Un nom
  - Une formule chimique
  - Un compartiment
- Soit en utilisant des métabolites présents dans des modèles existants

Création de la première réaction : les métabolites

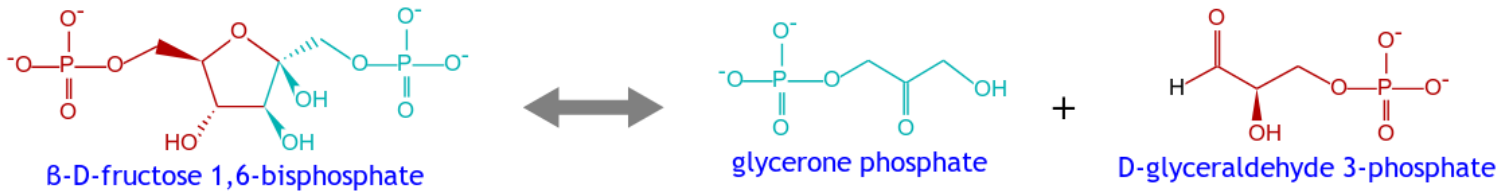



```
from cobra import *

bDfruct16P_c = Metabolite(
    id = 'bDfruct16P_c',
    formula='C6H10O12P2',
    name='beta-D-fructose 1,6-bisphosphate',
    compartment='c')
```

Exercice : Créer le glycérone phosphate (id = glyceroneP_c) et le D-glyceraldehyde 3-phosphate (id = glyceraldehyde3P_c)

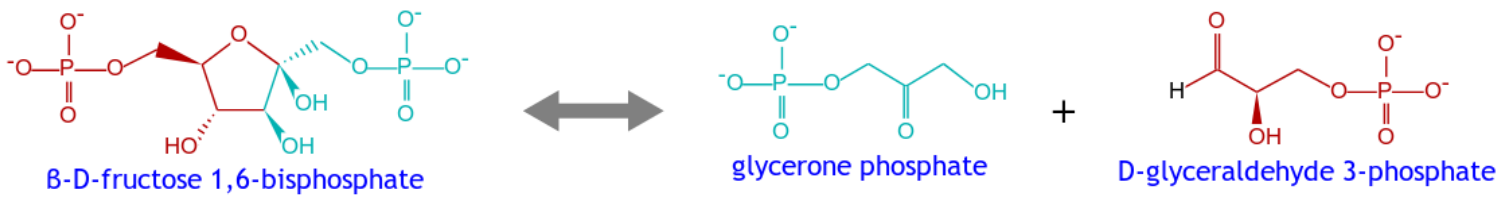
Création de la première réaction : les métabolites




```
glyceroneP_c = Metabolite(
    id = 'glyceroneP_c',
    formula='C3H5O6P',
    name='glycerone phosphate',
    compartment='c')

glyceraldehyde3P_c = Metabolite(
    id = 'glyceraldehyde3P_c',
    formula='C3H5O6P',
    name='D-glyceraldehyde 3-phosphate',
    compartment='c')
```

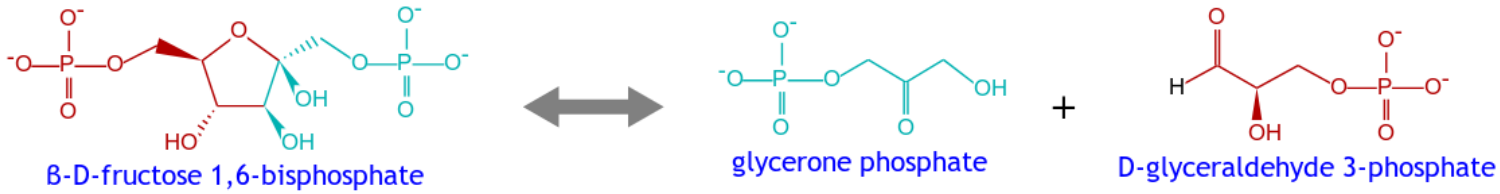
Création de la première réaction : la réaction



Une réaction sera composée de

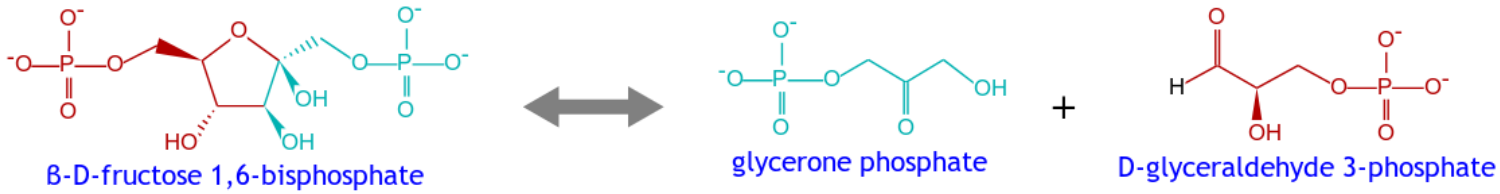
- un identifiant
- un nom
- un "subsystem" (correspond aux voies métaboliques)
- une lower et upper bound
  - La réaction est réversible : LB = -1000, UB = 1000
  - La réaction est irréversible : LB = 0, UB = 1000
  - On a d'autres informations sur les flux ? On les met ici
- de métabolites avec une stœchiométrie

Création de la première réaction : la réaction




```
reaction1 = Reaction('fructoseBiPaldolase')
reaction1.name = 'fructose-bisphosphate aldolase'
reaction1.subsystem = 'glycolysis I'
reaction1.lower_bound = -1000
reaction1.upper_bound = 1000
```

Création de la première réaction : la réaction




```
reaction1.add_metabolites({
    bDfruct16P_c: -1.0,
    glyceroneP_c: 1.0,
    glyceraldehyde3P_c: 1.0
})

print(reaction1.reaction)

#on ajoute un gène

reaction1.gene_reaction_rule  = '(gene1 or gene2)'

#utiliser '(gene1 and gene2)' si complexe protéique
```

Récupérer toutes les réactions impliquant un métabolite

Par exemple : dans quelles réactions est impliqué le glycérone phosphate ?

```
for reac in glyceroneP_c.reactions:
    print(reac)
    #ou print(reac.id) ou ce que vous voulez
```

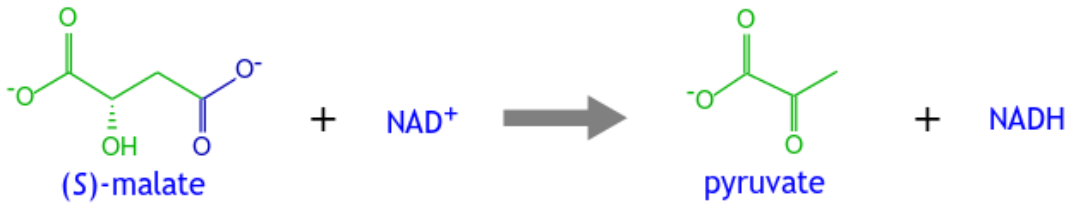
Une réaction équilibrée ?

La formule chimique des métabolites a été spécifiée, on peut vérifier si la réaction est équilibrée

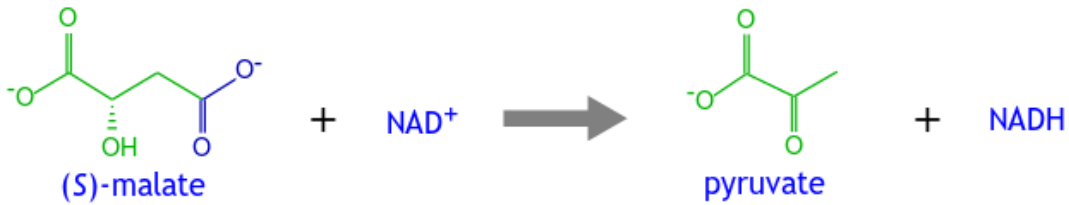
```
print(reaction1.check_mass_balance())
```

Exercice : une réaction non équilibrée

En utilisant le même principe, créer une seconde réaction (malateDehydrogenase) et vérifier si elle est équilibrée



Création de la deuxième réaction



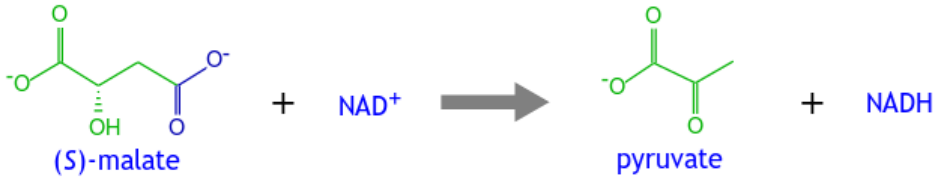

```
smalate_c = Metabolite(
    id = 'smalate_c',
    formula='C4H4O5',
    name='(S)-malate',
    compartment='c')

nad_c = Metabolite(
    id = 'nad_c',
    formula='C21H26N7O14P2',
    name='NAD+',
    compartment='c')
```

```
pyruvate_c = Metabolite(
    id = 'pyruvate_c',
    formula='C3H3O3',
    name='pyruvate',
    compartment='c')

nadh_c = Metabolite(
    id = 'nadh_c',
    formula='C21H27N7O14P2',
    name='NADH',
    compartment='c')
```

Création de la deuxième réaction

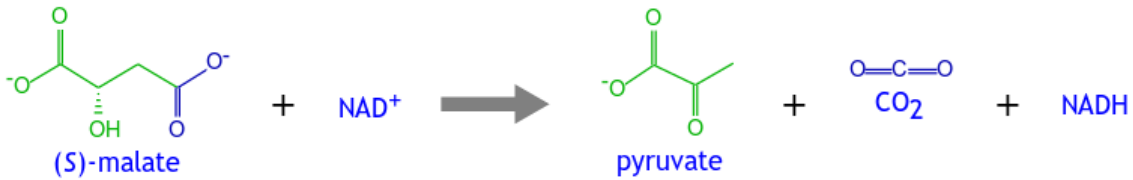



```
reaction2 = Reaction(
        id = "maladeDehydrogenase",
        name = "malade dehygrogenase
                (oxaloacetate-decarboxylating)",
        subsystem = {"chitin degradation to ethanol",
                "L-carnitine degradation III"},
        lower_bound = 0,
        upper_bound = 1000
)
reaction2.gene_reaction_rule = 'gene2'

reaction2.add_metabolites({
        smalate_c: -1, nad_c: -1, pyruvate_c: 1, nadh_c: 1 })

print(reaction2.check_mass_balance())
```

Corriger l'erreur de mass balance




```
co2_c = Metabolite(
    id = 'co2_c',
    formula = 'CO2',
    name = 'CO2',
    compartment = 'c')

reaction2.add_metabolites({co2_c: 1})

print(reaction2.check_mass_balance())
```

Un modèle pour les encapsuler toutes

Le modèle contiendra toutes les réactions

```
model = Model('premierModel')
print('Ce modèle contient %i réactions' %
    len(model.reactions))
print('Ce modèle contient %i métabolites' %
    len(model.metabolites))
```

C'est normal, on n'a pas rajouté la (les) réaction(s) au modèle

```
model.add_reactions([reaction1])
# ou model.add_reaction(reaction1) si une seule réaction

print('Ce modèle contient %i réactions' %
    len(model.reactions))
print('Ce modèle contient %i métabolites' %
    len(model.metabolites))
```

Jouons avec les objets

Faites en sorte d'afficher toutes les réactions, métabolites et gènes du modèle (contenant seulement reaction1)

```
Réactions :
fructoseBiPaldolase :
        bDfruct16P_c <=> glyceroneP_c + glyceraldehyde3P_c

Métabolites :
bDfruct16P_c : C6H10O12P2
glyceroneP_c : C3H5O6P
glyceraldehyde3P_c : C3H5O6P

Gènes :
gene1 catalyse fructoseBiPaldolase
gene2 catalyse fructoseBiPaldolase
```

Jouons avec les objets

Correction

```
print("Réactions :")
for reac in model.reactions:
    print("%s : %s" % (reac.id, reac.reaction))

print("")
print("Métabolites : ")
for species in model.metabolites:
    print('%s : %s' % (species.id, species.formula))

print("")
print("Gènes :")
print("------")
for gene in model.genes:
    print("%s catalyse :" % gene.id, end=" ")
    for reacId in gene.reactions:
        print(reacId.id + " ")
```

Jouons avec les objets

rajouter reaction2 au même modèle, et afficher les mêmes informations, mises à jour

```
Réactions :
fructoseBiPaldolase :
        bDfruct16_c <=> glyceraldehyde3P_c + glyceroneP_c
maladeDehydrogenase :
        nad_c + smalate_c --> co2_c + nadh_c + pyruvate_c

Métabolites :
bDfruct16_c : C6H10O12P2
glyceroneP_c : C3H5O6P
glyceraldehyde3P_c : C3H5O6P
smalate_c : C4H4O5
nad_c : C21H26N7O14P2
pyruvate_c : C3H3O3
nadh_c : C21H27N7O14P2
co2_c : CO2

Gènes :
gene1 catalyse : fructoseBiPaldolase
gene2 catalyse : maladeDehydrogenase fructoseBiPaldolase
```

Retrouver une réaction ou un métabolite

get_by_id() : retrouver des réactions ou des métabolites en connaissant leur id
Exemple : prendre la seconde réaction et retirer le CO_2

```
reaction3 = model.reactions.get_by_id('maladeDehydrogenase')
metabolite = model.metabolite.get_by_id('co2_c')
for reac in metabolite.reaction:
    print('CO2 intervient dans %s' % (reac.id))
reaction3.subtract_metabolites({
    model.metabolites.get_by_name('CO2'): 1})
print(reaction3)
```

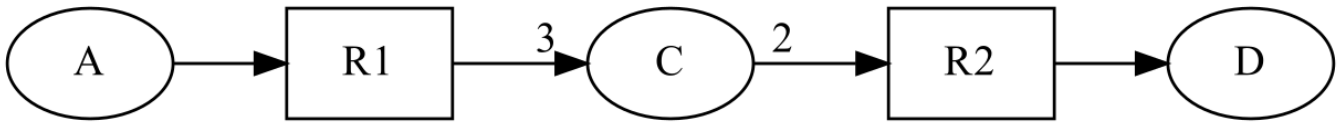
Et en se basant sur le nom ?

```
metaboCo2 = model.metabolites.query('^CO2$',
        attribute = 'name')
print(metaboCo2[0].formula)
```

Une première analyse de flux

Construire un nouveau modèle contenant ces réactions

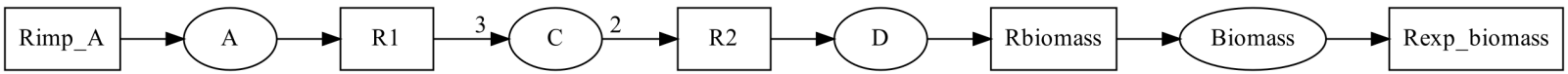
- R1 : A $\rightarrow$ 3 C
- R2 : 2 C $\rightarrow$ D



Une première analyse de flux

Créer deux réactions d'échange (boundary reactions) et une réaction de biomasse :

- Rimp_A : $\rightarrow$ A
  - avec UB = 20
- Rbiomass : D $\rightarrow$ Biomass
- Rexp_biomass : Biomass $\rightarrow$



L'ajout de biomasse fonctionnera toujours comme ça :
X + Y $\rightarrow$ Biomass $\rightarrow$
On maximisera les flux passant dans la réaction "Rexp_biomass"

Une première analyse de flux

On identifie les boundary reactions d'un modèle avec "model.boundary"

```
for reac in model.boundary:
    print(reac)
```

On veut maximiser les flux passant par une réaction donnée : cette réaction devient "l'objectif" du modèle.

```
model.objective = "Rexp_biomass"

#et on optimise

solution = model.optimize()
fluxMax  = solution.objective_value
print(fluxMax)
```

Quelle est la valeur obtenue ? Comment l'expliquez-vous ?

Le rapport à rendre

M'envoyer ce rapport à : sylvain.prigent@inra.fr

Le rapport contiendra les réponses aux questions rouges des slides suivantes :

- 30
- 31/32
- 35
- 37
- 38 (facultatif)

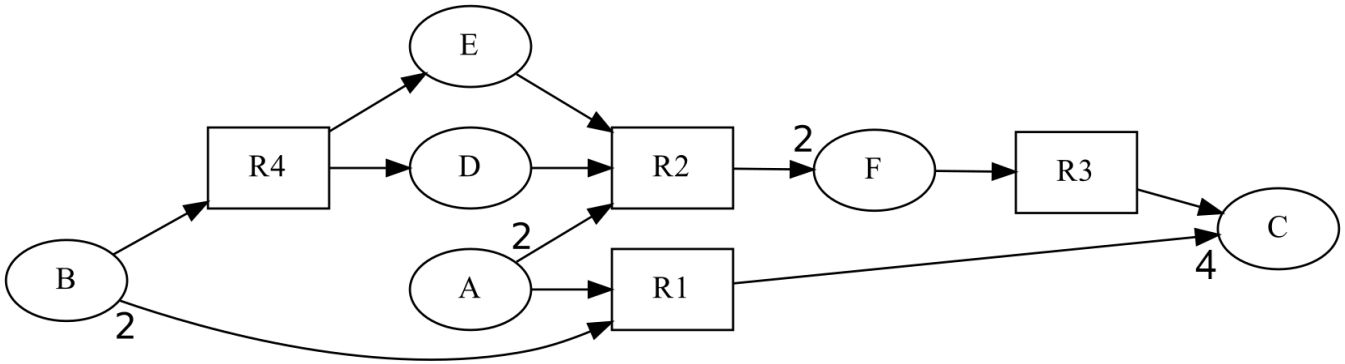
Les réponses ne sont pas forcées d'être très longues, montrez juste que vous avez compris ce que vous faites.

Joindre également les scripts vous ayant permis de répondre aux questions.

Une seconde analyse de flux (1/2)

Réaliser le même travail sur un nouveau modèle, avec ce réseau :

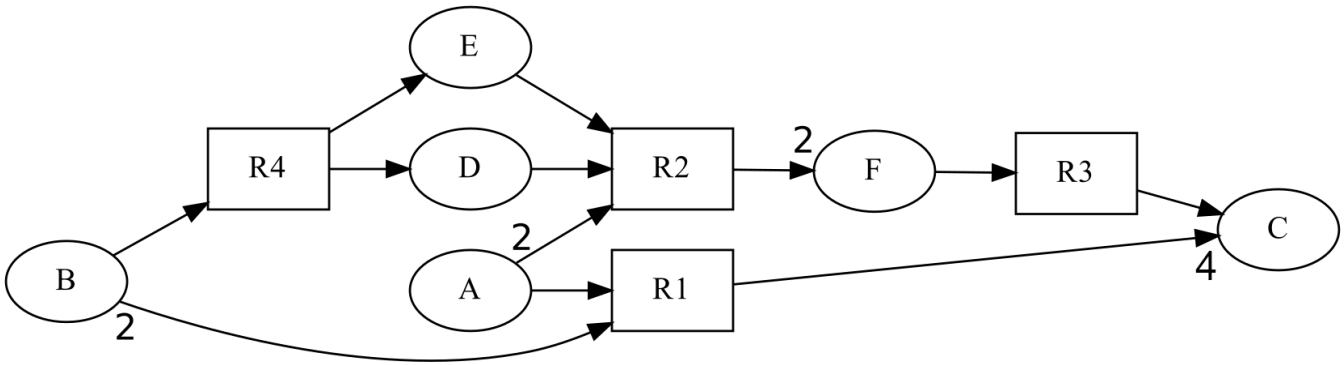
- R1 : A + 2 B $\rightarrow$ 4 C
- R2 : 2 A + D + E $\rightarrow$ 2 F
- R3 : F $\rightarrow$ C
- R4 : B $\rightarrow$ D + E



Une seconde analyse de flux (1/2)

Réaliser le même travail sur un nouveau modèle, avec ce réseau :

- R1 : A + 2 B $\rightarrow$ 4 C
- R2 : 2 A + D + E $\rightarrow$ 2 F
- R3 : F $\rightarrow$ C
- R4 : B $\rightarrow$ D + E



Métabolites et réactions définis dans le fichier "reseauFlux.py"

Une seconde analyse de flux (2/2)

Réaliser le même travail sur ce modèle, avec ces contraintes :

- Import de molécules B (UB = 10)
- La biomasse est composée de 2 molécules C

- Quelle est la valeur obtenue ?
- Comment l'expliquez-vous ?
- Comment le corrigeriez-vous en se basant sur le graphe ?
- Quelle valeur obtenez-vous alors ?
- Que renvoie "model.summary()" ?

Analyse de la distribution des flux

On peut également étudier par où passent les flux pour obtenir cet optimum

```
solution = model.optimize()

#flux contenus dans solution.fluxes
#par exemple :
for reac in model.reactions:
  print(str(reac.id) + " : " + str(solution.fluxes[reac.id]))

#ou juste print(solution.fluxes)
```

Étudiez cette distribution de flux. Était-ce attendu ?
Y a-t-il une autre distribution de flux possible selon vous ?

Analyse de la variabilité des flux

La FVA pour analyser tous les flux possibles

```
from cobra.flux_analysis import flux_variability_analysis

solution = model.optimize()

FVA_result = flux_analysis.variability.
        flux_variability_analysis(
        model, fraction_of_optimum=1.0)

print(FVA_result)
```

Dans ce petit modèle, quelles sont les réactions essentielles, bloquées et alternatives ?

Analyse de productibilité de molécules

Pour les 6 exemples suivants, vérifier si les molécules T sont productibles à partir de S en prenant en compte la stœchiométrie (FBA) et sans la prendre en compte (analyse de graphe). L'expliquer.

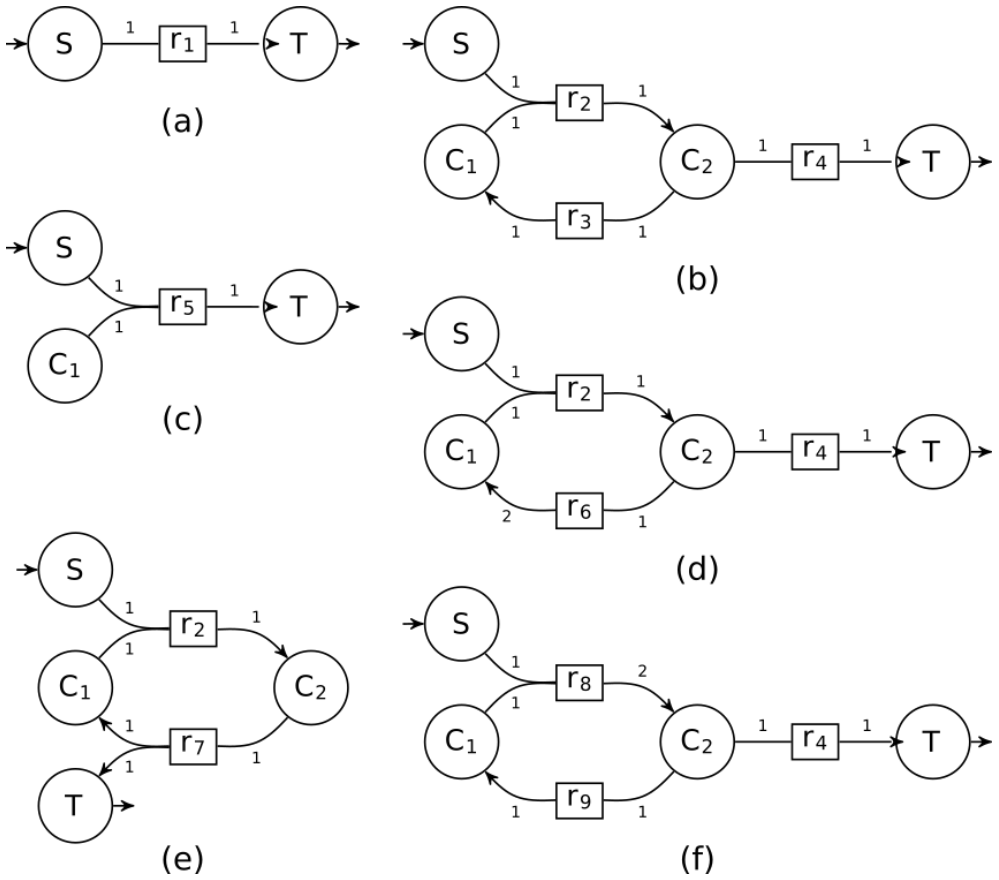
- r1 : S $\rightarrow$ T
- r2 : S + C1 $\rightarrow$ C2
- r3 : C2 $\rightarrow$ C1
- r4 : C2 $\rightarrow$ T
- r5 : S + C1 $\rightarrow$ T
- r6 : C2 $\rightarrow$ 2 C1
- r7 : C2 $\rightarrow$ C1 + T
- r8 : S + C1 $\rightarrow$ 2 C2
- r9 : C2 $\rightarrow$ C1

- model a : r1
- model b : r2, r3, r4
- model c : r5
- model d : r2, r6, r6
- model e : r2, r7
- model f : r8, r9, r4

- Version "graphe" dans la slide suivante

Analyse de productibilité de molécules

Pour les 6 exemples suivants, vérifier si les molécules T sont productibles à partir de S en prenant en compte la stœchiométrie (FBA) et sans la prendre en compte (analyse de graphe). L'expliquer.



réactions et métabolites : fichier "reacsProd.py"

Croissance sur différents substrats

Il peut être intéressant de regarder comment se comporte la production de biomasse ou de molécules précises lors de la consommation de différents substrats

Par exemple : un organisme consomme de l'oxygène et du glucose pour croitre.

- Y a-t-il un ratio optimal de consommation de chaque métabolite pour maximiser la production d'un métabolite ?
- Calcul de "l'enveloppe de production" (production enveloppe)

Croissance sur différents substrats

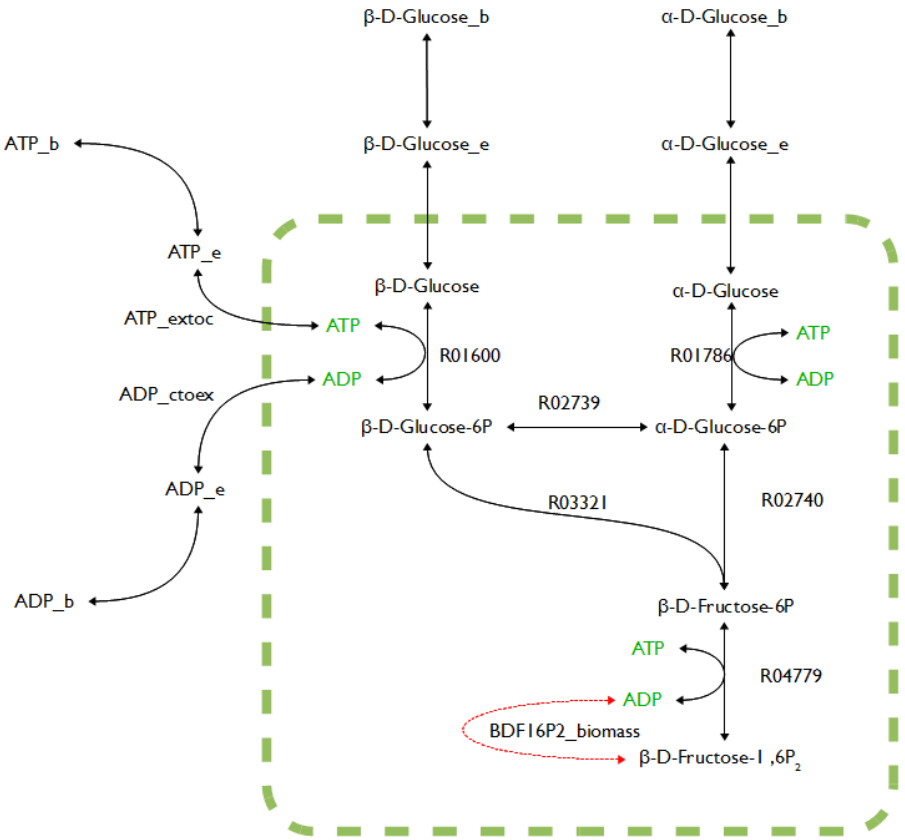
```
import cobra.test
from cobra.flux_analysis import *
modelTextbook = cobra.test.create_test_model("textbook")
#importe un modèle de test "inclut" dans cobra
prod_env = production_envelope(modelTextbook,
        reactions = ["EX_o2_e"], objective = "EX_ac_e",
        carbon_sources = "EX_glc__D_e")
#production de "ac" en fonction de la consommation d'O2

prod_env2 = production_envelope(modelTextbook,
        ["EX_glc__D_e", "EX_o2_e"])
#production de biomasse en fonction de la consommation
#d'O2 et de glucose
```

- Décrire rapidement le modèle importé
- Qu'est-ce que produit la réaction étudiée dans le premier cas ?
- Affichez ces enveloppes de production sous forme de graphe
  - Flux max à travers EX_ac_e en fonction de consommation d'O_2
  - Prod. max de biomasse en fonction de la consommation d'O_2 et de glucose

Un dernier exercice autour des flux

Fichier "fromWikipedia.py"



- Boundaries : import/export de :
  - ATP_b et ADP_b
  - α-D-Glucose_b et β-D-Glucose_b
- Contraintes : import max de glucose total = 20
- Biomasse : β-D-Fructose-1,6P_2

Faire tourner le modèle, et trouver une manière de représenter graphiquement le résultat obtenu

Et si on retire la réaction R01786 du modèle ?

Import / export de modèles

CobraPy gère différents formats de fichiers

- JSON
  - cobra.io.load_json_model("fichierInput.json")
  - cobra.io.save_json_model(model, "fichierOutput.json")
- SBML
  - Nécessite l'installation de libsbml (ceci est un lien)
    - sudo pip install python-libsbml
  - cobra.io.read_sbml_model("fichierInput.xml")
  - cobra.io.write_sbml_model(model, "fichierOutput.xml")
- MATLAB
- etc

Des modèles plus gros

CobraPy "contient" deux modèles métaboliques à l'échelle du génome (GEMs) : E. coli et Salmonella

```
import cobra.test

modelEcoli = cobra.test.create_test_model("ecoli")
modelSalmonella = cobra.test.create_test_model("salmonella")
```

Comparez ces deux modèles :

- Nombre de réactions, de métabolites, de gènes
- Nombre de réactions et de métabolites partagés
- Différences dans la composition de biomasse
- Étude de flux dans ces deux modèles
  - Soyez inventifs !

Un peu de bioengeenering

Imaginons que vous travaillez dans une entreprise pharmaceutique. Vous souhaitez produire une molécule d'intérêt en utilisant de la transgénique sur *E. coli* et en s'aidant de la modélisation

- En s'aidant des bases de données de réactions, essayez de produire une des molécules suivantes :
  - isopenicillin N
  - actinomycin D
  - griseofulvin

- Combien de moles peut-on en produire par mole de glucose ?
- Et en concervant une production de biomasse égale à 20% de la production maximale ?
- Quelles réactions ne peuvent être utilisées que pour produire la molécule d'intérêt et pas pour produire de la biomasse ?

Le rapport à rendre (rappel)

M'envoyer ce rapport à : sylvain.prigent@inra.fr

Le rapport contiendra les réponses aux questions rouges des slides suivantes :

- 30
- 31/32
- 35
- 37
- 38 (facultatif)

Les réponses ne sont pas forcées d'être très longues, montrez juste que vous avez compris ce que vous faites.

Joindre également les scripts vous ayant permis de répondre aux questions.