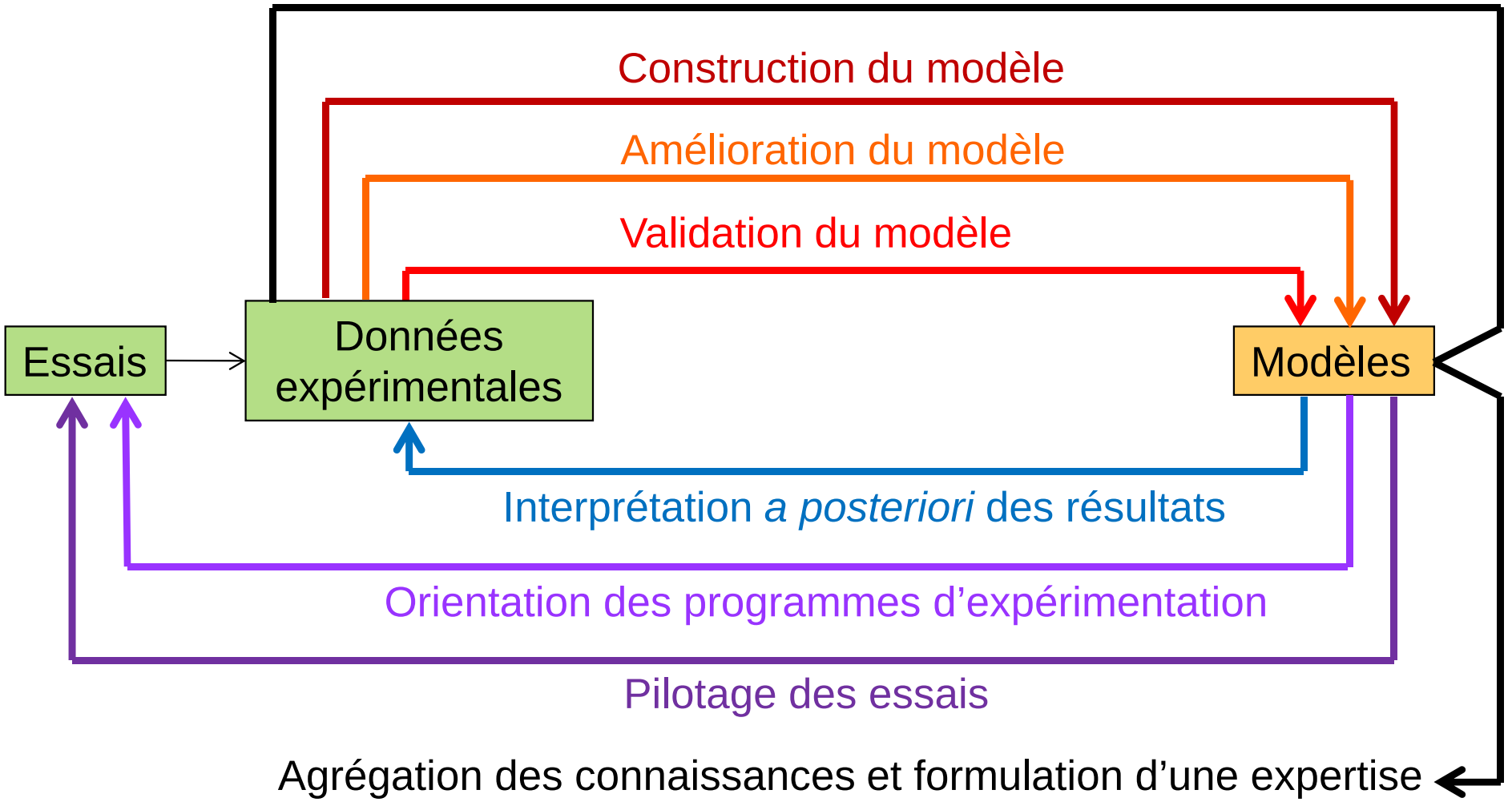
Quelques compléments
1) organisation de la relation données-simulation
2) aspects informatiques

François Brun (ACTA)

1) organisation de la relation données-simulation

Interactions entre expérimentation et modélisation



Besoins de données

- Pour mettre en œuvre:
  - estimation des paramètres
  - évaluation du modèle
  - (assimilation de données en temps réels)

Organisation

- Objectif: pour chaque donnée disponible, réussir à mettre en face la donnée simulée correspondante.

Ysim = f(X=var. entrée)

Yobs : des situations, définissant les variables d’entrées.

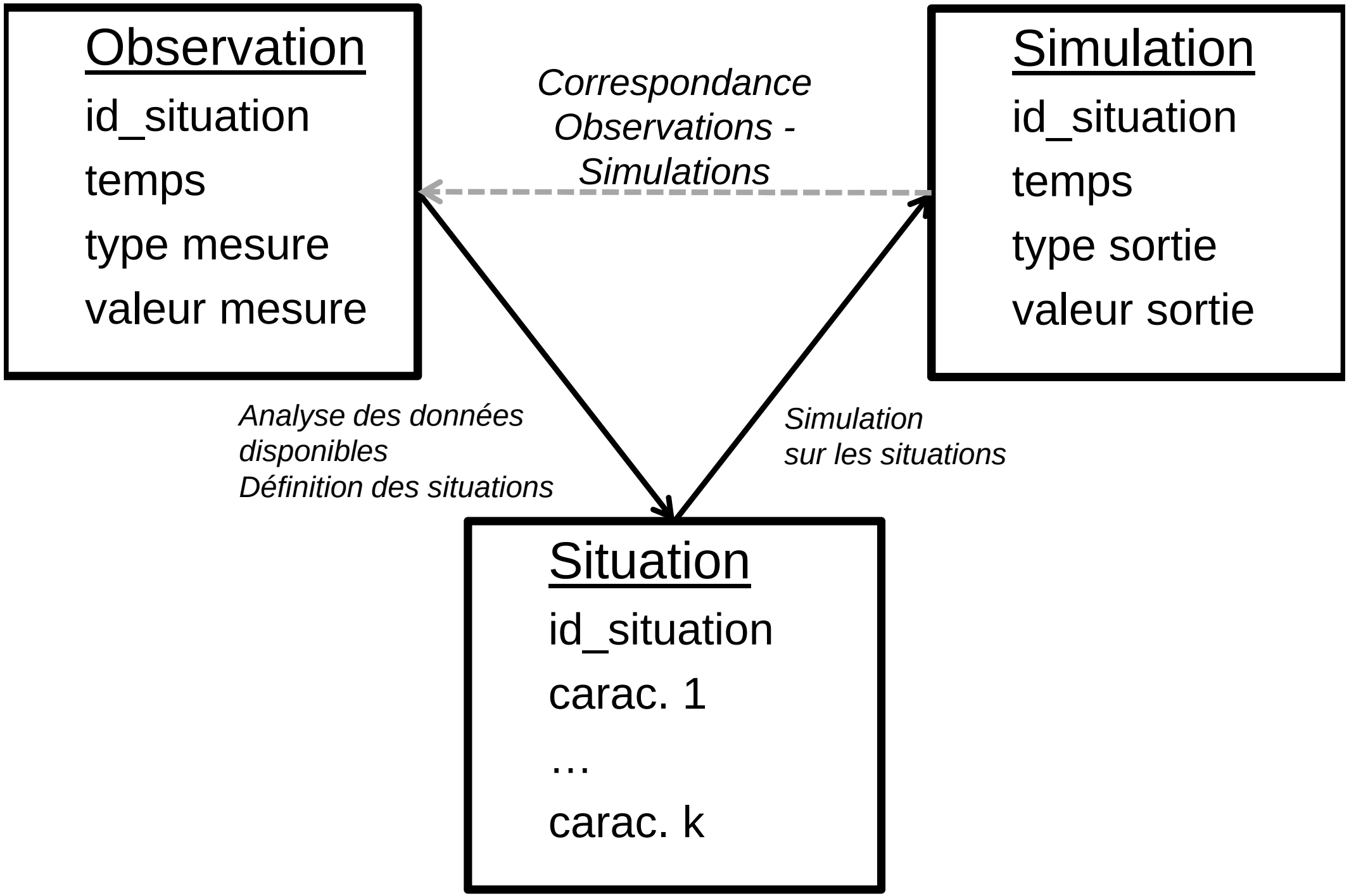
- Observations:
  - situations
  - temps
- Simulations
  - Situations => variables d'entrée
  - temps

⇒1) Définir une table caractérisant les situations

⇒Par exemple :

sol s	météo m	variété v
sol s	météo m'	variété v
sol s	météo m''	variété v
sol s'	météo m	variété v
sol s'	météo m'	variété v
sol s'	météo m''	variété v

3 tables


Observation
id_situation
temps
type mesure
valeur mesure
Correspondance Observations - Simulations
Simulation
id_situation
temps
type sortie
valeur sortie
Analyse des données disponibles
Définition des situations
Simulation sur les situations
Situation
id_situation
carac. 1
...
carac. k

Organisation des programmes

- Définir liste de toutes les situations disponibles
  - Ou Sous-liste des situations pour la méthode
- Application d'une méthode :
  - Début de l'algorithme
  - 1 à n fois
    - réaliser les simulations sur l'ensemble des situations, en y associant l'identifiant des situations.
    - Combiner simulations et observations (jointure sur situation ET temps)
    - Définition de l'étape suivante
  - Fin de l'algorithme

exemple

- Démo : ZeBook
  - Ch10maize2evaluation
  - Case study, Maize model. Evaluation.

```
# liste situations
list_n_sy=unique(maize.data_EuropeEU$sy)

# simulations
maize.multisy(maize.define.param()["nominal",], list_n_sy,
100,250, weather_all=weather_EuropeEU)

# jointure simulations-observations
simobs <- merge(sim,maize.data_EuropeEU, by=c("sy", "day"),
all.x=TRUE)

# cœur de la méthode
goodness.of.fit(simobs$B,simobs$Bobs,draw.plot=FALSE)
```

maize.multisy

```
function(param, list_site_year, sdate,ldate,
weather_all=weather_EuropeEU){
  sim <- data.frame()
  for(sy in list_site_year){
    weather = maize.weather(working.year=strsplit(sy,"-
")[[1]][2], working.site=strsplit(sy,"-
")[[1]][1],weather_all=weather_all)
    result <- maize.model2(param, weather, sdate, ldate)
    result <- cbind(sy,result)
    sim <- rbind(sim, result)
    }
  return(sim)
}
```

2) aspects informatiques

Les étapes d'une simulation

On peut décrire l'algorithme de simulation d'un modèle dynamique ainsi :

- **Initialiser toutes les variables d'état, les paramètres** et autres variables nécessaires pour effectuer les calculs.
- **Propager, à chaque pas de temps** (par exemple, jour). Boucle sur les 2 étapes suivantes jusqu'à la fin de la simulation.
  - **Calculer les taux de changement des variables d'état** ou les changements dans ces variables d'état qui ont lieu à chaque pas de temps.
  - **Mettre à jour les variables d'état** pour le nouveau pas de temps en ajoutant le taux de changement, multiplié par le pas de temps à la valeur de la variable d'état à partir du pas de temps précédent. C'est ce qu'on appelle l'intégration d'Euler.

    (par exemple : $TT(j+1) = TT(j) + \Delta TT(j)$).
- **Enregistrer les résultats** des simulations dans une table pour une analyse ultérieure, en conservant la dynamique des variables d'état.

Implémentation dans le langage R

```r
#Function
lactation.model.calf <- function(cu,kdiv,kdl,kdh,km,ksl,kr,ks,ksm,mh,mm,p,mum,rc,fin,temps)
{
 # Initialize variables
H=rep(NA,(fin-1)/temps)
CS=rep(NA,(fin-1)/temps)
M=rep(NA,(fin-1)/temps)
Mmoy=rep(NA,(fin-1)/temps)
RM=rep(NA,(fin-1)/temps)

 # Initialization of state variables
H[1]=1.0
CS[1]=520.0
M[1]=0.0
Mmoy[1]=0.0

i=1
 # Simulation loop
for (t in seq(0, fin, by = temps))
  {
 # Calculate rates of change of state variables (dH,dCS,dM,dMmoy)
    dH = - kdh * H[i] * temps
    dCS = (mum * (H[i]/(kdiv+H[i]))*cu - (ks + ksm*((Mmoy[i]/mh)^p/(1+(Mmoy[i]/mh)^p)))*CS[i] ) * temps
    dM = (km * CS[i] * ((mm-M[i])/(mm-M[i]+ksl))-(M[i]/(kdl+M[i]))*rc ) * temps
    dMmoy = kr*(M[i]-Mmoy[i]) * temps

 # Uptade state variables
    H[i+1]= H[i] +dH
    CS[i+1]= CS[i] + dCS
    M[i+1]= M[i] + dM
    Mmoy[i+1]= Mmoy[i] + dMmoy

 # removal of milk
    RM[i]=(M[i]/(kdl+M[i]))*rc

  i=i+1
  }
  # End simulation loop

results=data.frame(CS=CS[1:(fin/temps)],M=M[1:(fin/temps)],Mmoy=Mmoy[1:(fin/temps)],RM=RM[1:(fin/temps)],
day=seq(0.1,fin,by=temps),week=seq(0.1/7,fin/7,by=temps/7))
return(results)
}
```

Fonction

Initialisation des variables d'état

Calcul des taux de variation

Mise à jour des variables d'état

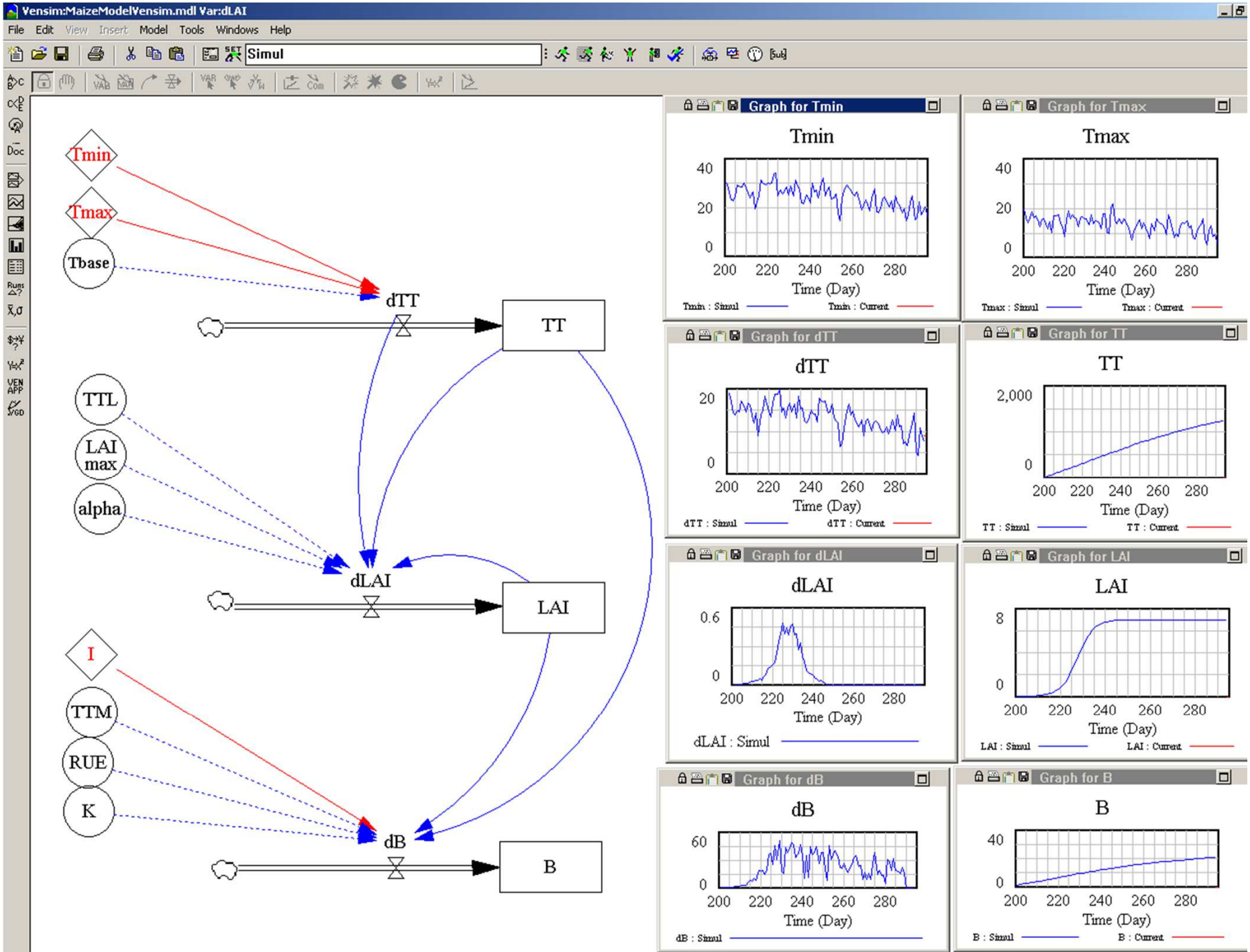
Boucle d'intégration

Renvoi des résultats

Le simulateur n’est pas écrit en R

- Cas le plus fréquent
  - En général, simulateur n’est pas écrit en R
  - Mais : exécutable ou sous autre plateforme
    - Codé dans un autre langage : Fortran (Stics), Java (CAPSIS), C++, C#, python (open aléa)
    - Inclus dans une plateforme (RECORD-VLE; Stella, Vensim,…)

Vensim (formalisme Forester)



Vensim (2) - Mais derrière l'interface…

```
alpha=    0.00243
Tbase= 6
TTL=700
TTM= 1200
RUE= 1.85
LAImax= 7
K= 0.7

Tmax:= GET XLS DATA('Y1.xls','Y1','1','B7')
Tmin:= GET XLS DATA('Y1.xls','Y1','1','B6')
I:= GET XLS DATA('Y1.xls','Y1','1','B5')

dB= IF THEN ELSE( (TT<=TTM) , RUE*(1-exp(-K*LAI))*I , 0 )
dLAI= IF THEN ELSE( TT<=TTL, alpha*dTT*LAI*max(LAImax-LAI,0),0)
dTT= max((Tmin+Tmax)/2-Tbase, 0)

TT= INTEG (dTT,0)
LAI= INTEG (dLAI, 0.01)
B= INTEG (dB,1)
********************************************************
        .Control
********************************************************~
          Simulation Control Parameters
FINAL TIME  = 294
        ~    Day
        ~    The final time for the simulation.
INITIAL TIME  = 200
        ~    Day
        ~    The initial time for the simulation.
SAVEPER  =
          TIME STEP
        ~    Day [0,?]
        ~    The frequency with which output is stored.
TIME STEP  = 1
        ~    Day [0,?]
        ~    The time step for the simulation.
```

…on retrouve bien la définition des paramètres, des variables d'entrées, des équations, de la procédure d'intégration

2 possibilités

- Réécrire le simulateur ?
  - Non, trop couteux…
  - Sauf, si on n'a pas la main… sur les entrées et les sortie
  - Alors R n'est pas forcément une bonne solution (lent pour écire un modèle dynamique : exemple modèle de lactation…)

- Privilégie l'interfaçage R - simulateur

Interfaçage R - simulateur

- Condition nécessaire: pouvoir interagir avec le simulateur.
- Pour chaque simulation
  - Fournir les données d’entrée automatiquement à partir de R
  - Lancer la simulation
  - Récupérer la sortie

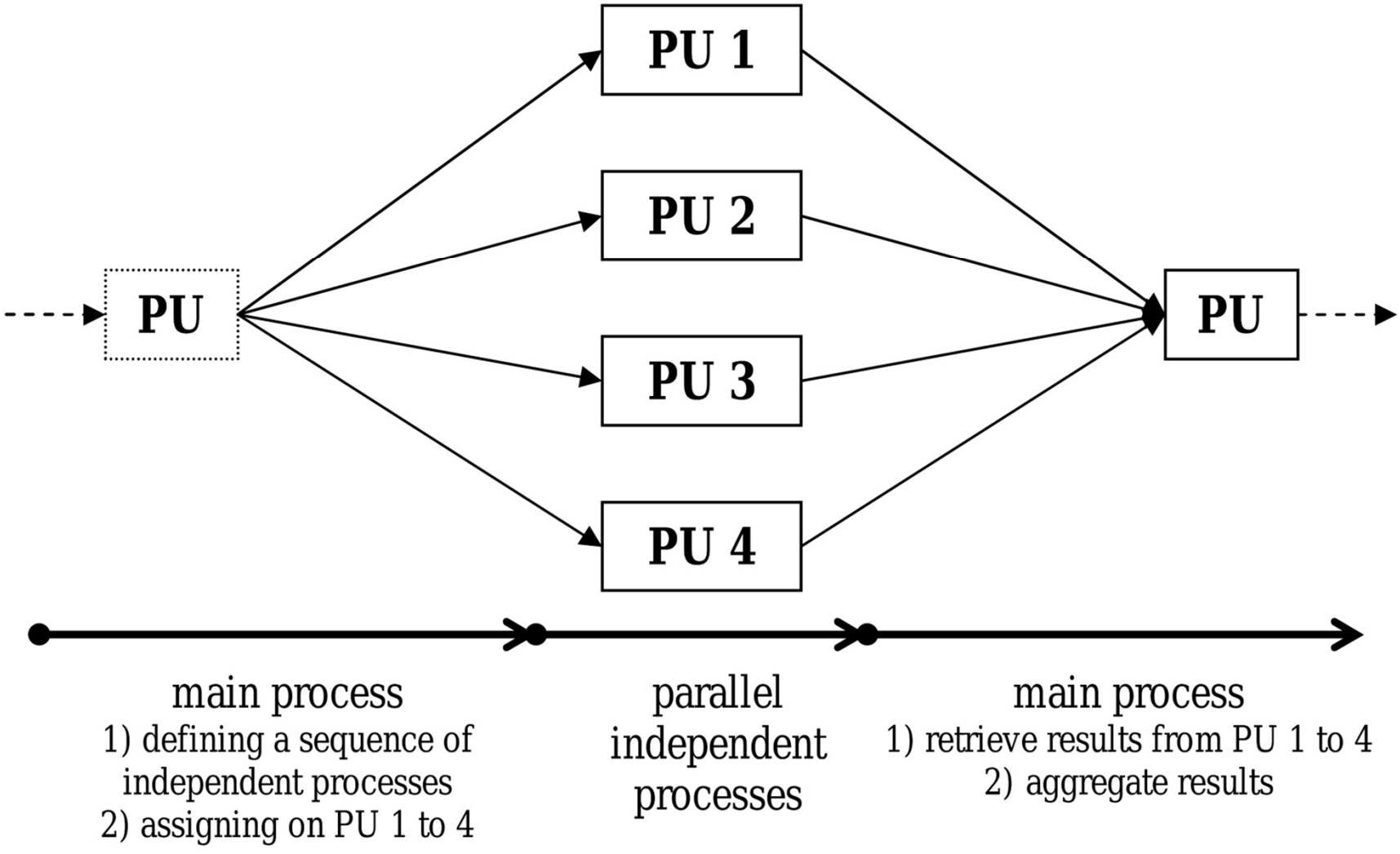
Exemple R - Exécutable

```
library(ZeBook)
# define the working directory where maize.exe resides
setwd("c:/ZeBook/R_intro")
# write one year of weather data into the file weather.dat
write.table(maize.weather(working.year=2010,
working.site=1),file="weather.dat",sep=";",col.names=T,row.names=F,quote=
F,dec="." )
# prepare the parameter values, and write into file "param.ini"
param <- maize.define.param()["nominal",]
param <-
rbind(data.frame(param=names(param),value=param,stringsAsFactors=F),
c("sdate",100),c("ldate",250))
write.table(param,file="param.ini",sep=";",col.names=T,row.names=F,quote=
F,dec="." )
# launch the execution of the maize model
shell("maize.exe")
# retrieve the results which are in the file result.dat
result <- read.table(file="result.dat",sep=";",header=T,dec="." )
```

Performances : paralléliser sous R

- Demo : ch03R_maize_parallel

(Improving performance with parallel code for maize.model)



Package ZeBook

- Version 0.5

  install.packages("ZeBook")

- Puis utilisation :

  En début de script :

  library(ZeBook)

  Aide:

  help(ZeBook)

  Ou sur une fonction particulière

  help(maize.model)

  Voir le contenu d'une fonction :

  maize.model sans ()

  Voir le début d'un jeu de données

  head(weather_SouthAsia)

Contenu du package ZeBook

- ZeBook-package — ZeBook : Working with Dynamic Crop Models.
- epirice.define.param — Define parameter values of the epirice model
- epirice.model — The EPIRICE model function
- epirice.multi.simul — Wrapping fonction to run an virtual experimental design for EPIRICE model
- epirice.weather — Reading Weather data function for EPIRICE (South Asia Weather)
- evaluation.criteria — Computation of classical evaluation criteria function
- fRcA — Correction coefficient for infection - age (epirice)
- fRcT — Correction coefficient for infection - temperature (epirice)
- fRcW — Correction coefficient for infection - Relative humidity and rain (epirice)
- grain.filling — Wheat grain weight measurements after anthesis
- magarey.define.param — Define parameter values of the model function
- magarey.model — The Magarey model
- magarey.model2 — The Magarey model, taking in argument a vector of param
- magarey.simule — Wrapping fonction to run an virtual experimental design for Magarey model
- maize.define.param — Define parameter values of the maize model
- maize.model — The basic Maize model function
- maize.model2 — The basic Maize model function with a litle change to use with maize.simul
- maize.RUEtemp — Effect of the temperature on RUE for Maize
- maize.simule — Wrapping fonction to run an virtual experimental design for maize model
- maize.weather — Reading Weather data function for MAIZE model (West of France Weather)
- maize_cir.model — the Maize model function with additionnal state varaiable CumInt
- maize_cir_rue.model — The Maize model function with rue (and CumInt)
- maize_cir_rue_ear.model — The Maize model function with CumInt, RUE and ear growth

Contenu du package ZeBook (suite)

• param.runif	Building a random plan with uniform distribution
• q.arg.fast.runif	Building the q.arg argument for FAST function
• verhulst.model	the Verhulst model function - integration loop
• verhulst.update	the Verhulst model function - Update Y
• watbal.define.param	Define parameter values of the Water Balance model
• watbal.model	Water Balance model function - Integration loop
• watbal.simobsdata	Soil water content measurements and
• watbal.update	Water Balance model function - Update
• watbal.weather	Reading Weather data function for Water Balance model (West of France Weather)
• weather_EuropeEU	Weather serie for EuropeEU from NASA POWER agroclimatology
• weather_FranceWest	Weather serie for West of France from NASA POWER agroclimatology
• weather_SouthAsia	Weather serie for South of Asia from NASA POWER agroclimatology
• weed.define.param	Define parameter values of the weed model function
• weed.model	The weed model function
• weed.simule model	Wrapping fonction to run an virtual experimental design for weed
• weed.update	The weed update function
• Wheat_GPC	Grain Protein Contents in Wheat Grains
• ZeBook	ZeBook : Working with Dynamic Crop Models.