GUSTAVE ROUSSY
CANCER CAMPUS
GRAND PARIS

Paralleliser purrr... (voire le tydiverse !) : le package **furrr**

Matthieu FARON

Département de chirurgie viscérale oncologique

Département de biostatistiques et epidémiologie

INSERM 1018 (CESP) ONCOSTAT méthodologie et épidémiologie clinique en oncologie moléculaire

Essais thérapeutiques, méta analyses, méta analyses en réseau…

Essai thérapeutique

Type d'étude médicale ayant pour but **d'évaluer l'effet d'un traitement**

L'efficacité est toujours **comparative**

- Vs placebo
- VS traitement de référence

Randomisé

Double aveugle

Nombre de patient calculé à l'avance

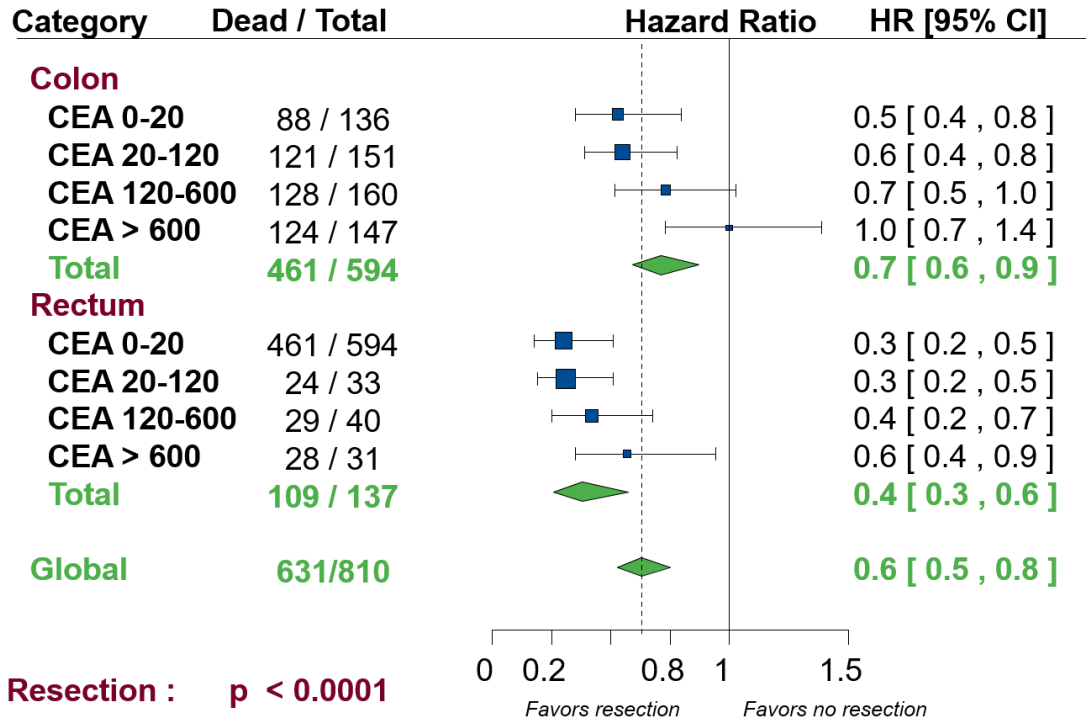
Variation et interaction

L'effet traitement est une variable **aléatoire**

- ➔ Susceptible de varier d'un essai à l'autre
- ➔ Résultats qui semblent contradictoires

Interaction

- Effet traitement varie selon une caractéristique du patient
- Mais essai dimensionné pour montrer l'efficacité dans la population totale pas dans un sous groupe



Faron, ASCO, 2012
Faron, Eur J Cancer, 2015

Méta analyse (MA)

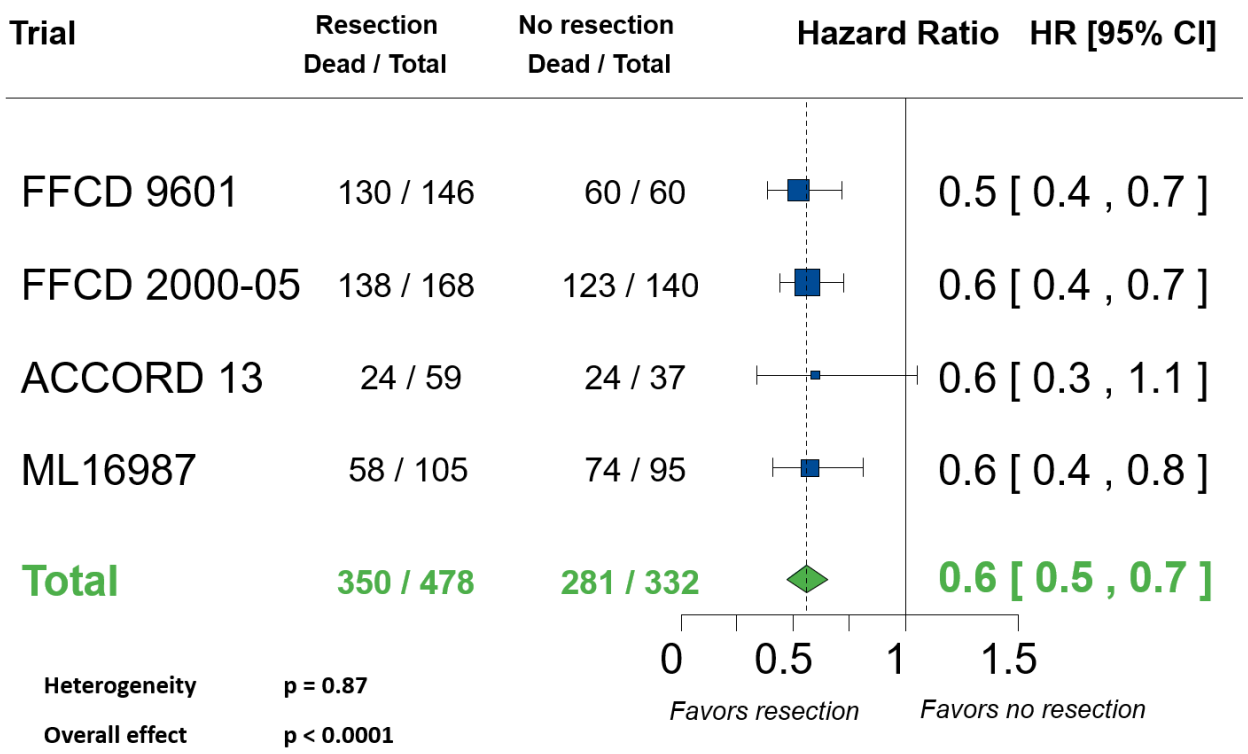
Etude qui met en commun tous les essais thérapeutiques ayant comparé 2 traitements

- Retrouver tous les essais : limiter le biais de publication

Permet de **gagner de la puissance**

- Obtenir une estimation « poolée »

Estimer les interactions grâce au gain de puissance



Faron, ASCO, 2012
Faron, Eur J Cancer, 2015

Méta-analyse en réseau (NMA)

Extension de la méta-analyse au cas où il y a

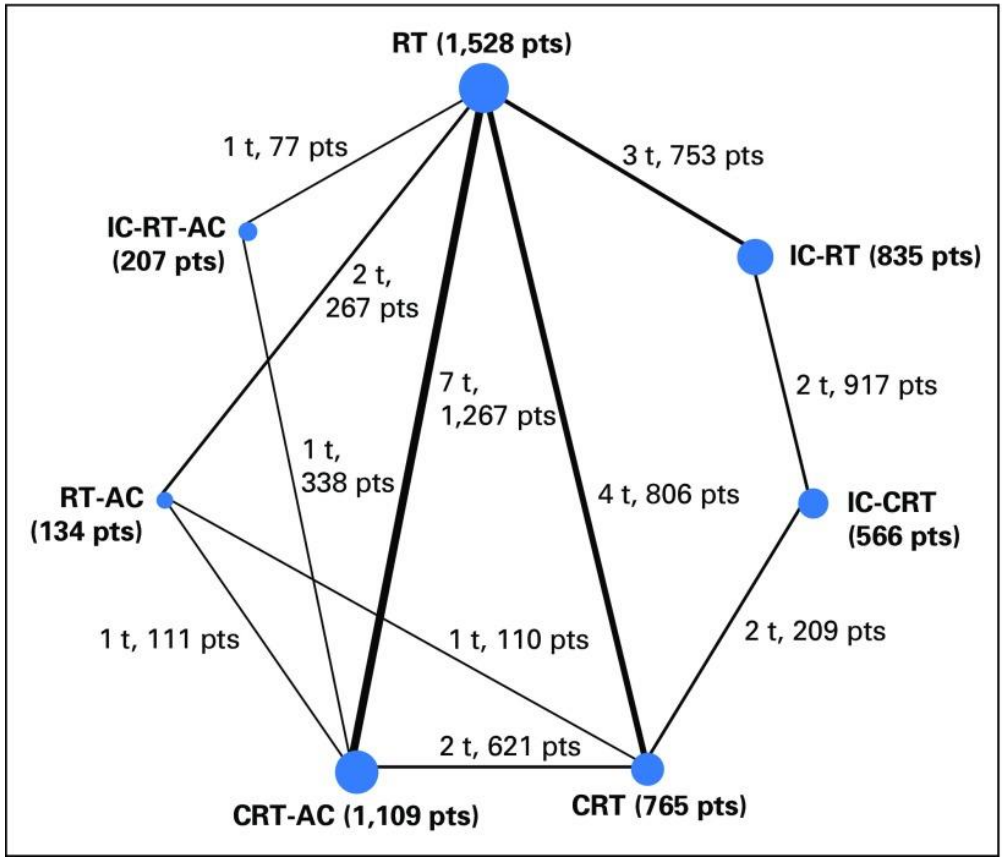
plus de deux traitements

Certaines comparaisons plus ou moins

étudiées … voire pas du tout

- Estimation directe
- Estimation indirecte

Hypothèse de **consistency**



Deux types de méta analyse

DONNÉES AGRÉGÉES (AD)

Données récupérées de la publication

- L'effet traitement (1 par essai)
- son écart type

Pas d 'équipe à contacter

Etude de l'interaction impossible

DONNÉES INDIVIDUELLES (IPD)

On récupère les données des patients

- On recalcul tout

Ajustement / Interaction possible

Contacter les équipes

En l'absence d'interaction les deux approches donnent les mêmes résultats

La plus grande difficulté est d'obtenir les données !

Notre problématique

NMA en IPD

La plupart des Méta Analyses sont faites sur données agrégées

➔ Encore plus vrai pour les NMA (plus de données à récupérer)

Notre équipe utilise traditionnellement les données invidiuelles (vrai méta-analyse ;-))

- ➔ Et donc aussi de l'IPD-NMA

Mais l'IPD-NMA est peu utilisée ou alors

- En Bayésien (équipe du MRC de Londres)
- Ou en 2 étapes

Donc l'interaction est peu étudiée

La question de recherche

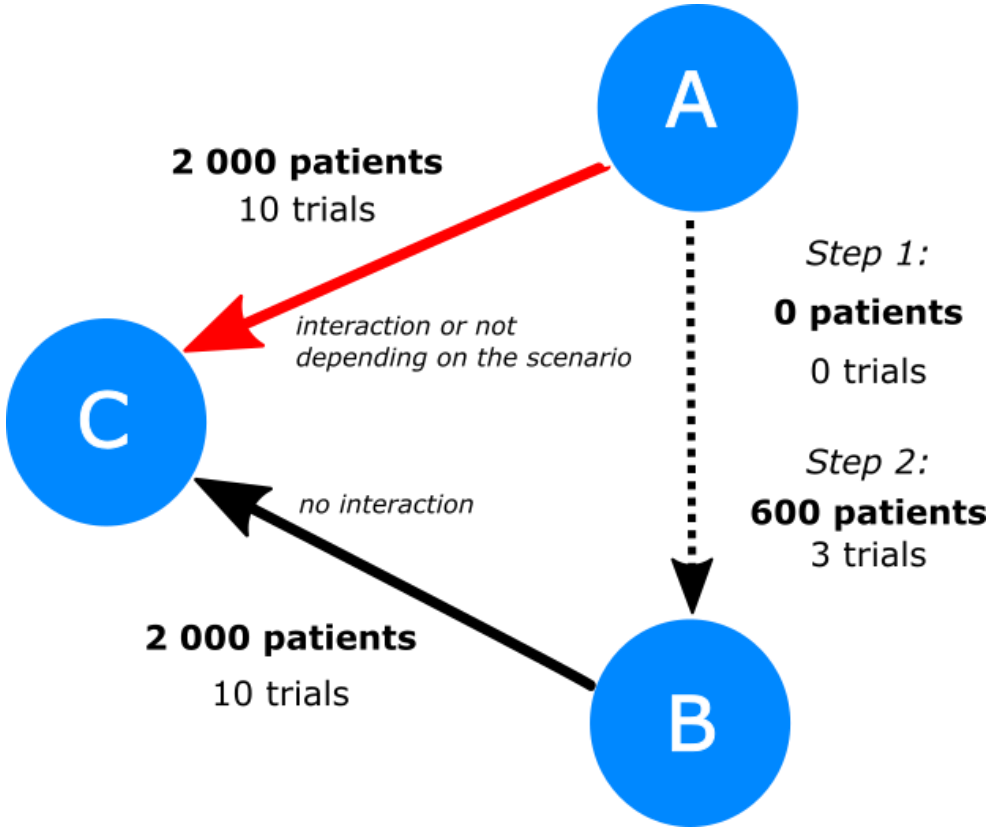
Que se passe-t-il si dans une **MA en réseau** si :

- L'**interaction** n'est pas la même
- La **distribution** du modificateur n'est pas la même
- dans **toutes les arêtes** du réseau

L'approche par AD-NMA donnent-elles encore des résultats correctes ?

Décision d'approcher le problème par simulation…

Le design



On veut estimer AB

Réseau simple 3 traitement (NMA)

- 1ère étape : que de l'information indirecte
- 2nd étape : un peu d'information

J'ai écrit une fonction qui simule des données de survie

- Proche de celles de la vraie-vie (Weibull)
- Intercept aléatoires
- Effet aléatoires des traitements
- Interactions
- Fonction de censures (Weibull)

Protocole de simulations

Variable parameters	Possible value
Scenario	• None (no interaction, no distribution differences) • Interaction (interaction in AC, no distribution differences) • Both (interaction in AC, mean age 52 in AC and 68 in B C)
Treatment effect	-0.2 (HR=0.82) ou -0.5(HR= 0.61)
Random variation of baseline hazard (σ)	0.1 ou 0.01
Random variation of treatment effect (τ)	0.1 ou 0.01
Total number of possibility	4 * 2 * 2 * 2 = 32
Number of replications for each possibility:	1 000
Steps	Step 1 : No trials in the direct comparison A vs. B Step 2 : 3 trials of 200 patients in the direct comparison A vs. B
Overall number of simulation :	32 * 1 000 * 2 = 64 000

Différents modèles à tester

Analyse séparées des essais sans interaction

AD-NMA

coxph / glm

netmeta

Analyse séparées mais **avec interaction** en 4 classes

4 AD-NMA (une pour chaque classe) : **IPD-2step**

coxph / glm

netmeta

NMA-IPD 1 step (1 seul gros modèle)

coxme / glmer

Packages : survival, coxme, lme4, netmeta

IPD en 1 temps :écriture du model

$$log(status) =$$ *Evènements*

$$1 + ns(periode, 3) +$$ *Risque de base*

$$TTT_1 + TTT_2 + age +$$ *Effets fixes*

$$age * TTT_1 + age * TTT_2 +$$ *Interactions*

$$offest(log(exposition)) +$$ *Offset*

$$(1|essai) + (TTT_1|essai) + (TTT_2|essai)$$ *Effets aléatoires*

family = « poisson »

On travail sur des « pseudo-patients », chaque patient pouvant être présent dans un ou plusieurs intervalles

Comment garder rangé ?

Fichiers séparés ?

- Par quel paramètre séparer ?

Nested lists ?

- Difficile à générer

➔ Pas pratique à exploiter par la suite

```
for(i in c("none", "interaction", "both"))
{
  for(j in c(-2L, -5L)) {
    for(k in c(10L, 100L)) {
      for(l in c(10L, 100L)) {
        …
      }
    }
  }
}
```

Tidy simulations

```r
parameters <- list(
  scenario = c("none", "interaction", "both"),
  ttt = c(-2L, -5L),
  sig = c(10L, 100L),
  tau = c(10L, 100L),
  n_sim = 1:1000,
  comp = c("AC", "BC", "AB"),
  essai = 1:10
) %>% cross_df()
```

```
## # A tibble: 720,000 x 7
##    scenario      ttt   sig   tau n_sim comp  essai
##    <chr>       <int> <int> <int> <int> <chr> <int>
##  1 none           -2    10    10     1 AC        1
##  2 interaction    -2    10    10     1 AC        1
##  3 both           -2    10    10     1 AC        1
##  4 none           -5    10    10     1 AC        1
##  5 interaction    -5    10    10     1 AC        1
##  6 both           -5    10    10     1 AC        1
##  7 none           -2   100    10     1 AC        1
##  8 interaction    -2   100    10     1 AC        1
##  9 both           -2   100    10     1 AC        1
## 10 none           -5   100    10     1 AC        1
## # ... with 719,990 more rows
```

On génère toutes les possibilités

Tidy simulations

```
sim_df <- parameters %>% mutate(
  poisson_df = pmap(list(scenario = scenario,
                         ttt = ttt,
                         sig = sig,
                         tau = tau,
                         comp = comp),
                    survsim)
)
```

On garde « rangées » les data avec les paramètres qui les ont générées

```
## # A tibble: 720,000 x 8
##    scenario      ttt   sig   tau n_sim comp  essai poisson_df
##    <chr>       <int> <int> <int> <int> <chr> <int> <list>
##  1 none           -2    10    10     1 AC        1 <df[,8] [966 x 8]>
##  2 interaction    -2    10    10     1 AC        1 <df[,8] [1,019 x 8]>
##  3 both           -2    10    10     1 AC        1 <df[,8] [1,043 x 8]>
##  4 none           -5    10    10     1 AC        1 <df[,8] [928 x 8]>
##  5 interaction    -5    10    10     1 AC        1 <df[,8] [1,074 x 8]>
##  6 both           -5    10    10     1 AC        1 <df[,8] [1,195 x 8]>
##  7 none           -2   100    10     1 AC        1 <df[,8] [935 x 8]>
##  8 interaction    -2   100    10     1 AC        1 <df[,8] [989 x 8]>
##  9 both           -2   100    10     1 AC        1 <df[,8] [949 x 8]>
## 10 none           -5   100    10     1 AC        1 <df[,8] [1,011 x 8]>
## # ... with 719,990 more rows
```

La dernière colonne contient maintenant les données pour chaque possibilités

```r
do_sim <- function(sim_df) {
results <- list(
  ad_models = glm(…),
  ad_nets = netmeta(…),
  ipd_2steps_models = glm(…),
  ipd_nets = netmeta(…),
  ipd_1step_models = glmer(…)
  )
}
```

On crée une fonction qui fait tous les modèles

```r
## On call la fonction
final <- sim_df %>%
 group_by(n_sim) %>%
 nest() %>%
 transmute(res = map(data,
 do_sim))


## On unnest les résultats
final <- final %>%
 unnest_wider(res)
```

On voudrait une colonne supplémentaire pour chaque modèle, rangées avec les paramètres correspondants

Le « problème »

Faisons la 1ère simulations (des 32 variations)

```
library(tictoc)

tic()

res <- sim_df %>% filter(n_sim < 2) %>%
  group_by(n_sim) %>% nest() %>%
  transmute(res = map(data, do_sim))

toc()
```

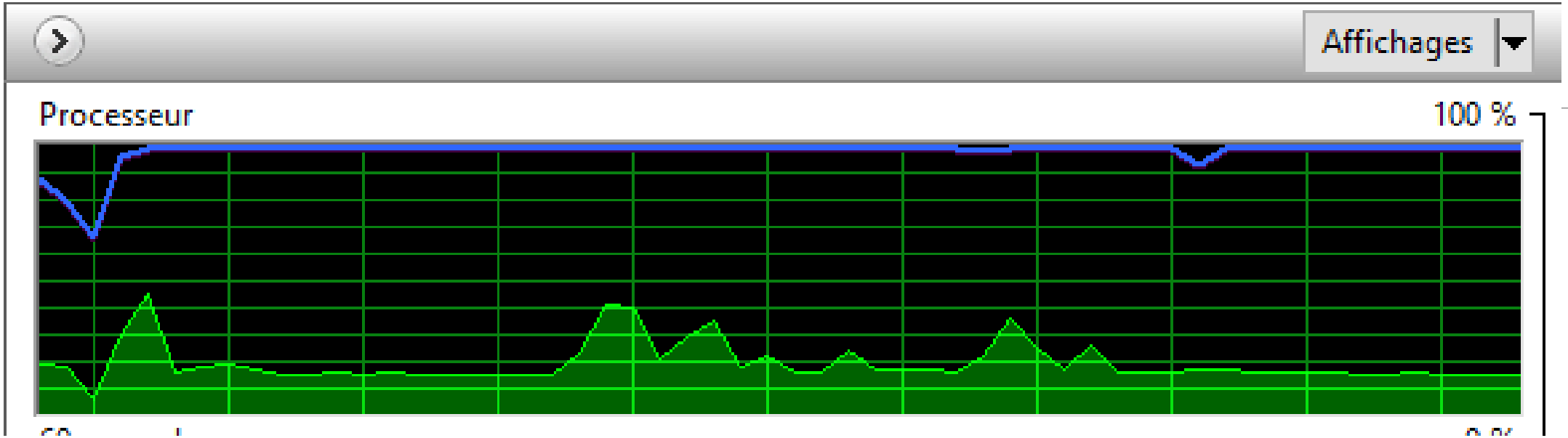
```
## 478.76 sec elapsed
```

1 000 simulations ➔ 5,5 JOURS

Avec les 2 steps : 11 jours …

Pourquoi est-ce si lent ?

Ouvrons le gestionnaire de ressource…



Le processeur semble ne tourner que à 1/8 de sa puissance max…

Processeur à plusieurs cœurs

La plupart des ordinateur ont des processeurs

à **plusieurs cœurs**

- Unité fonctionnelle indépendante

Permet de faire tourner **plusieurs programmes**

en même temps

Mais il faut que le programme sache utiliser

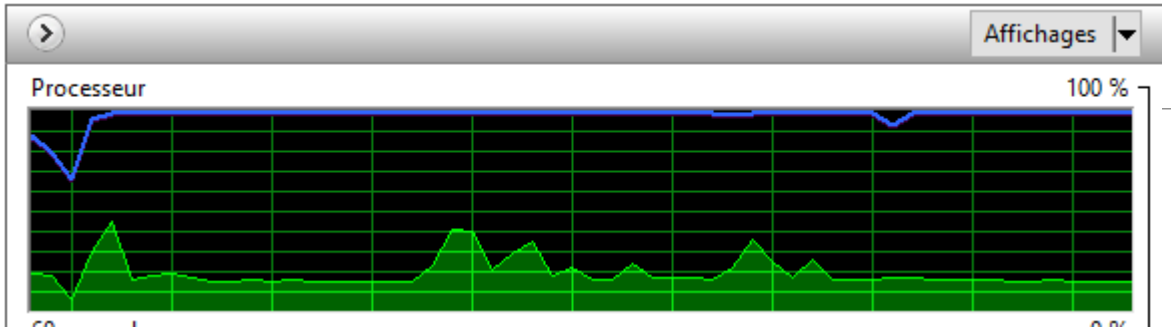
tous les cœurs

Processeur Intel® Core™ i3-9100

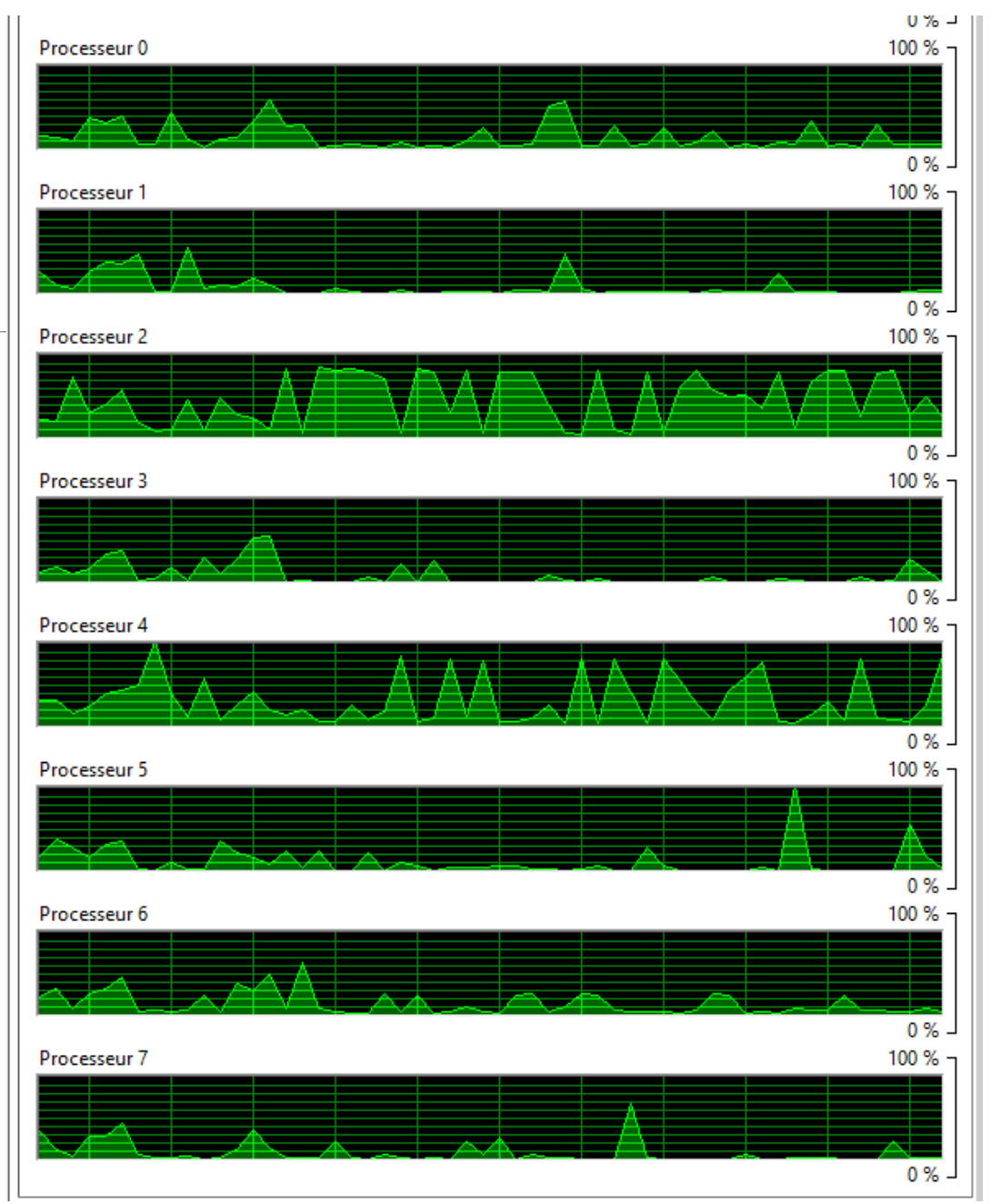
- 6 MB Intel® Smart Cache Mémoire cache
- 4 Cœurs
- 4 Fils
- 4.20 GHz Fréquence Turbo maxi
- 9th Generation

Processeur Intel® Xeon® Platinum 8280L
GHz)

- 38.5 MB Mémoire cache
- 28 Cœurs
- 56 Fils
- 4.00 GHz Fréquence Turbo maxi



Manifestement dans « base R », tous les cœurs ne sont pas utilisés…



R en parallel processing

Parallel processing

R n’exécute pas spontanément le code en parallel

Package(parallel)

- Base-R
- Différence windows / mac
- Exporter manuellement les données
- Exporter manuellement les packages

`lapply()` ➔ `parLapply()`

Package(Foreach)

- Plus proche d’une boucle

Package(Multidplyr) :

- N’est plus développé
- Scinder manuellement le jeu de données
- Appliquer le code avec do()

multidplyr

Le package future

Exporte automatiquement (essaye)

- Données
- Package
- ➔ Plus simple

Changer en une ligne le type d'exécution

- Plan(multisession)

README.md

future: Unified Parallel and Distributed Processing in R for Everyone

Introduction

Henrik Bengtsson

The purpose of the future package is to provide a very simple and uniform way of evaluating R expressions asynchronously using various resources available to the user.

In programming, a *future* is an abstraction for a *value* that may be available at some point in the future. The state of a future can either be *unresolved* or *resolved*. As soon as it is resolved, the value is available instantaneously. If the value is queried while the future is still unresolved, the current process is *blocked* until the future is resolved. It is possible to check whether a future is resolved or not without blocking. Exactly how and when futures are resolved depends on what strategy is used to evaluate them. For instance, a future can be resolved using a sequential strategy, which means it is resolved in the current R session. Other strategies may be to resolve futures asynchronously, for instance, by evaluating expressions in parallel on the current machine or concurrently on a compute cluster.

Controlling How Futures are Resolved

The future package implements the following types of futures:

Name	OSes	Description
synchronous:		*non-parallel:*
`sequential`	all	sequentially and in the current R process
`transparent`	all	as sequential w/ early signaling and w/out local (for debugging)
asynchronous:		*parallel:*
`multiprocess`	all	multicore, if supported, otherwise multisession
`multisession`	all	background R sessions (on current machine)
`multicore`	not Windows/not RStudio	forked R processes (on current machine)
`cluster`	all	external R sessions on current, local, and/or remote machines
`remote`	all	Simple access to remote R sessions

https://github.com/HenrikBengtsson/future

Le package furrr

Combine **purrr / future**

Modifie les fonctions de purrr pour les paralleliser

Naming cohérent :

- **map() ➔ future_map()**
- **pmap() ➔ future_pmap()**
- **map_dbl() ➔ future_map_dbl()**
- **map_at() ➔ future_map_at()**

build passing CRAN 0.1.0 build passing

furrr

The goal of furrr is to simplify the combination of `purrr`'s family of mapping functions and `future`'s parallel processing capabilities. A new set of `future_map_*()` functions have been defined, and can be used as (hopefully) drop in replacements for the corresponding `map_*()` function.

The code draws *heavily* from the implementations of `purrr` and `future.apply` and this package would not be possible without either of them.

What has been implemented?

The full range of `map()`, `map2()`, `pmap()`, `walk()`, `imap()`, `modify()`, and `invoke_map()` functions have been implemented.

This includes strict versions like `map_dbl()` through `future_map_dbl()` and predicate versions like `map_at()` through `future_map_at()`.

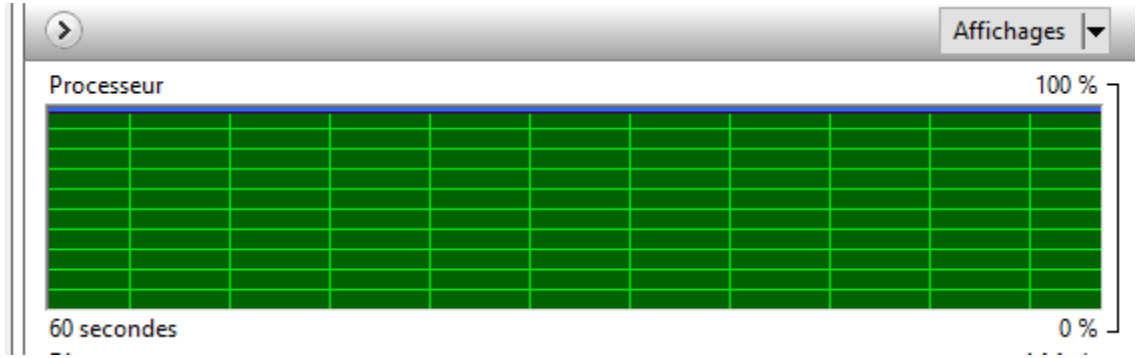
Générer toutes les données (8 cœurs)

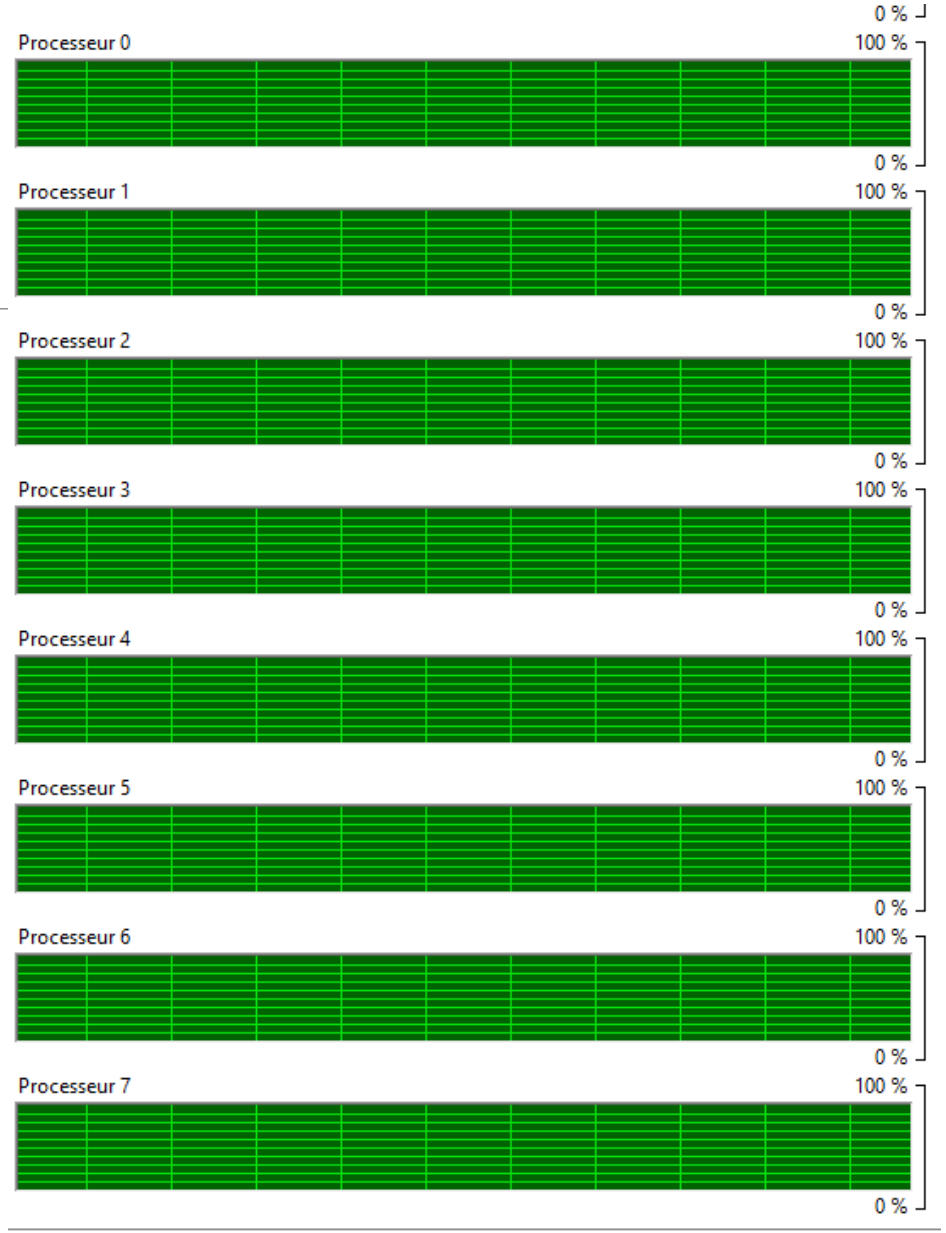
```
tic()
plan(sequential)
sim_df <- parameters  %>% mutate(
  poisson_df = future_pmap(list(
     scenario = scenario,
     ttt = ttt / 10,
     sig = 1 / sig,
     tau = 1 / tau,
     comp = comp),
  survsim2))
toc()
```

```
## 2861 sec elapsed
```

```
tic()
plan(multisession)
sim_df <- parameters  %>% mutate(
  poisson_df = future_pmap(list(
     scenario = scenario,
     ttt = ttt / 10,
     sig = 1 / sig,
     tau = 1 / tau,
     comp = comp),
  survsim2))
toc()
```

```
## 727 sec elapsed
```





Le processeur tourne à fond !

Tous les cœurs fonctionnent

Quelques limitations

Ce n’est pas 8 fois plus rapide

Démarrer le « cluster » prends du temps

- Exporter les données
- Exporter les packages

Le transfert de données entre les « workers » prend du temps

- Aller / retour
- Fusion des données

Nécessite parfois de petites optimisations

- Limiter transfert de données
- Equilibrer la charge de travail

1ère approche

```
ad_models      <- future_map(sim_df, glm)
ad_nets        <- future_map(sim_df, netmeta)
ipd_2s_models  <- future_map(sim_df, netmeta)
ipd_nets       <- future_map(sim_df, glm2)
ipd_ma_models  <- future_map(sim_df, netmeta)
ipd_1s_models  <- future_map(sim_df, glmer)
```

Le jeu de données est renvoyé à chaque fois

Les résultats collectées à chaque fois

```
## 2434 sec elapsed
```

2nd approche

```
do_sim <- function(sim_df) {
   ad_models      <- map(sim_df, glm)
   ad_nets        <- map(sim_df, netmeta)
   ipd_2s_models  <- map(sim_df, netmeta)
   ipd_nets       <- map(sim_df, glm2)
   ipd_1s_models  <- map(sim_df, glmer)
}
final <- sim_df %>%
  group_by(n_sim) %>% nest() %>%
  transmute(res = future_map(data, do_sim))
```

On crée une fonction qui fait tous les modèles

On sépare par un paramètre, j'ai choisis n_sim car n_sim >> availableCores()

On applique la fonction par future_map()
Les données ne sont envoyées qu'une seule fois

```
## 516 sec elapsed
```

Le résultat

Le résultat

Simulations **lancées à distance**

- Rstudio server

Sur le **serveur du CESP**

- 80 cœurs
- 256 GO de mémoire vive

J'avais aussi envisagé Amazon Web service

- Location pour quelques dollars par heure
- Instances toutes prêtes

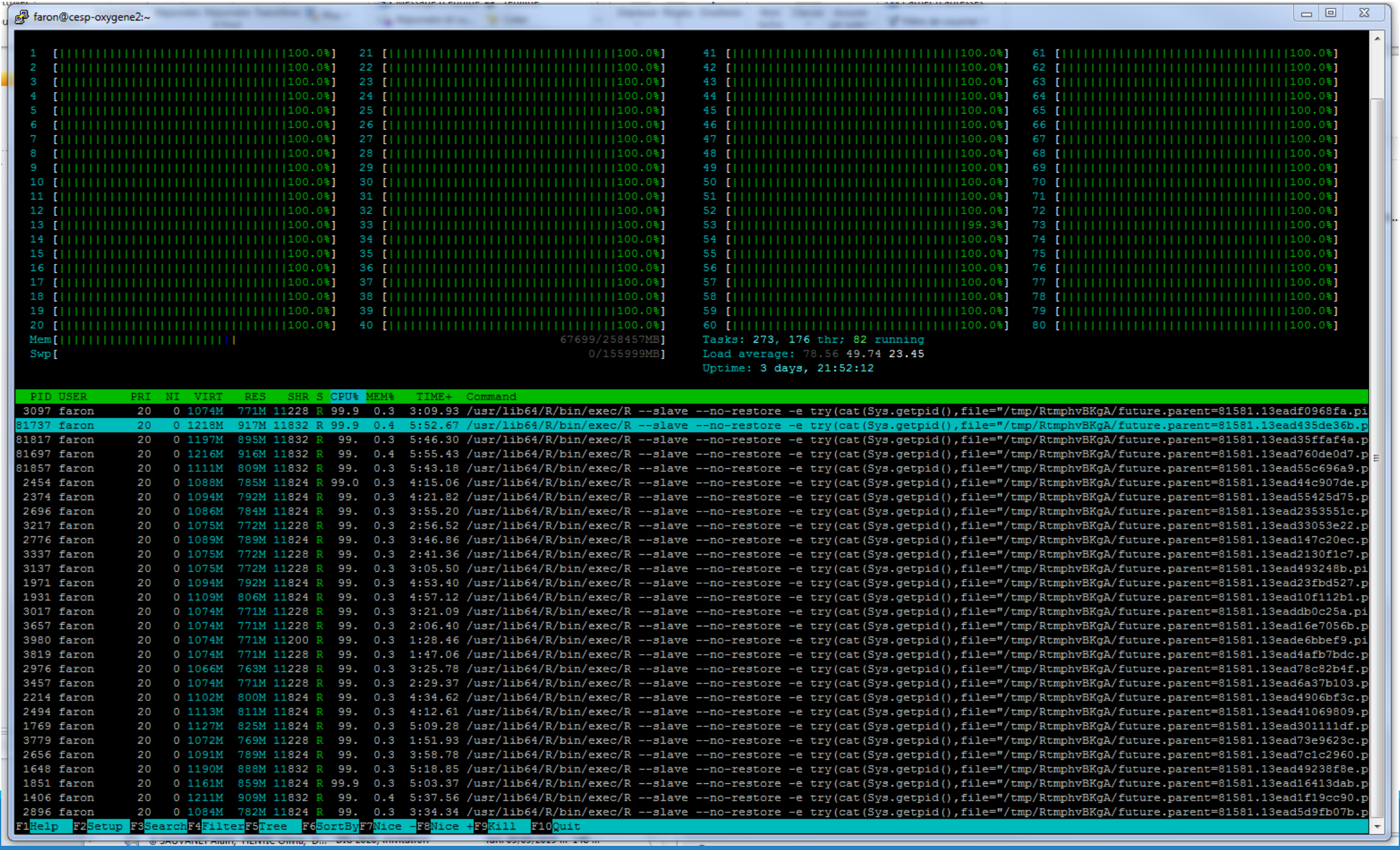
Mais j'ai eu des problèmes de **mémoire vive**…

- Dépassement des 256 GO

Donc simulations faites en plusieurs « chunk »

En quelques heures

- < 12 heures


faron@cesp-oxygene2:~
Mem[67699/258457MB]
Swp[0/155999MB]
Tasks: 273, 176 thr; 82 running
Load average: 78.56 49.74 23.45
Uptime: 3 days, 21:52:12
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
3097 faron 20 0 1074M 771M 11228 R 99.9 0.3 3:09.93 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13eadf0968fa.pi
81737 faron 20 0 1218M 917M 11832 R 99.9 0.4 5:52.67 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13ead435de36b.p
81817 faron 20 0 1197M 895M 11832 R 99. 0.3 5:46.30 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13ead35ffaf4a.p
81697 faron 20 0 1216M 916M 11832 R 99. 0.4 5:55.43 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13ead760de0d7.p
81857 faron 20 0 1111M 809M 11832 R 99. 0.3 5:43.18 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13ead55c696a9.p
2454 faron 20 0 1088M 785M 11824 R 99.0 0.3 4:15.06 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13ead44c907de.p
2374 faron 20 0 1094M 792M 11824 R 99. 0.3 4:21.82 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13ead55425d75.p
2696 faron 20 0 1086M 784M 11824 R 99. 0.3 3:55.20 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13ead2353551c.p
3217 faron 20 0 1075M 772M 11228 R 99. 0.3 2:56.52 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13ead33053e22.p
2776 faron 20 0 1089M 789M 11824 R 99. 0.3 3:46.86 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13ead147c20ec.p
3337 faron 20 0 1075M 772M 11228 R 99. 0.3 2:41.36 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13ead2130f1c7.p
3137 faron 20 0 1075M 772M 11228 R 99. 0.3 3:05.50 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13ead493248b.pi
1971 faron 20 0 1094M 792M 11824 R 99. 0.3 4:53.40 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13ead23fbd527.p
1931 faron 20 0 1109M 806M 11824 R 99. 0.3 4:57.12 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13ead10f112b1.p
3017 faron 20 0 1074M 771M 11228 R 99. 0.3 3:21.09 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13eaddb0c25a.pi
3657 faron 20 0 1074M 771M 11228 R 99. 0.3 2:06.40 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13ead16e7056b.p
3980 faron 20 0 1074M 771M 11200 R 99. 0.3 1:28.46 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13eade6bbef9.pi
3819 faron 20 0 1074M 771M 11228 R 99. 0.3 1:47.06 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13ead4afb7bdc.p
2976 faron 20 0 1066M 763M 11228 R 99. 0.3 3:25.78 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13ead78c82b4f.p
3457 faron 20 0 1074M 771M 11228 R 99. 0.3 2:29.37 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13ead6a37b103.p
2214 faron 20 0 1102M 800M 11824 R 99. 0.3 4:34.62 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13ead4906bf3c.p
2494 faron 20 0 1113M 811M 11824 R 99. 0.3 4:12.61 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13ead41069809.p
1769 faron 20 0 1127M 825M 11824 R 99. 0.3 5:09.28 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13ead301111df.p
3779 faron 20 0 1072M 769M 11228 R 99. 0.3 1:51.93 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13ead73e9623c.p
2656 faron 20 0 1091M 789M 11824 R 99. 0.3 3:58.78 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13ead7c1c2960.p
1648 faron 20 0 1190M 888M 11832 R 99. 0.3 5:18.85 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13ead49238f8e.p
1851 faron 20 0 1161M 859M 11824 R 99.9 0.3 5:03.37 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13ead16413dab.p
1406 faron 20 0 1211M 909M 11832 R 99. 0.4 5:37.56 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13ead1f19cc90.p
2896 faron 20 0 1084M 782M 11824 R 99. 0.3 3:34.34 /usr/lib64/R/bin/exec/R --slave --no-restore -e try(cat(Sys.getpid(),file="/tmp/RtmphvBKgA/future.parent=81581.13ead5d9fb07b.p
F1Help F2Setup F3SearchF4FilterF5Tree F6SortByF7Nice -F8Nice +F9Kill F10Quit

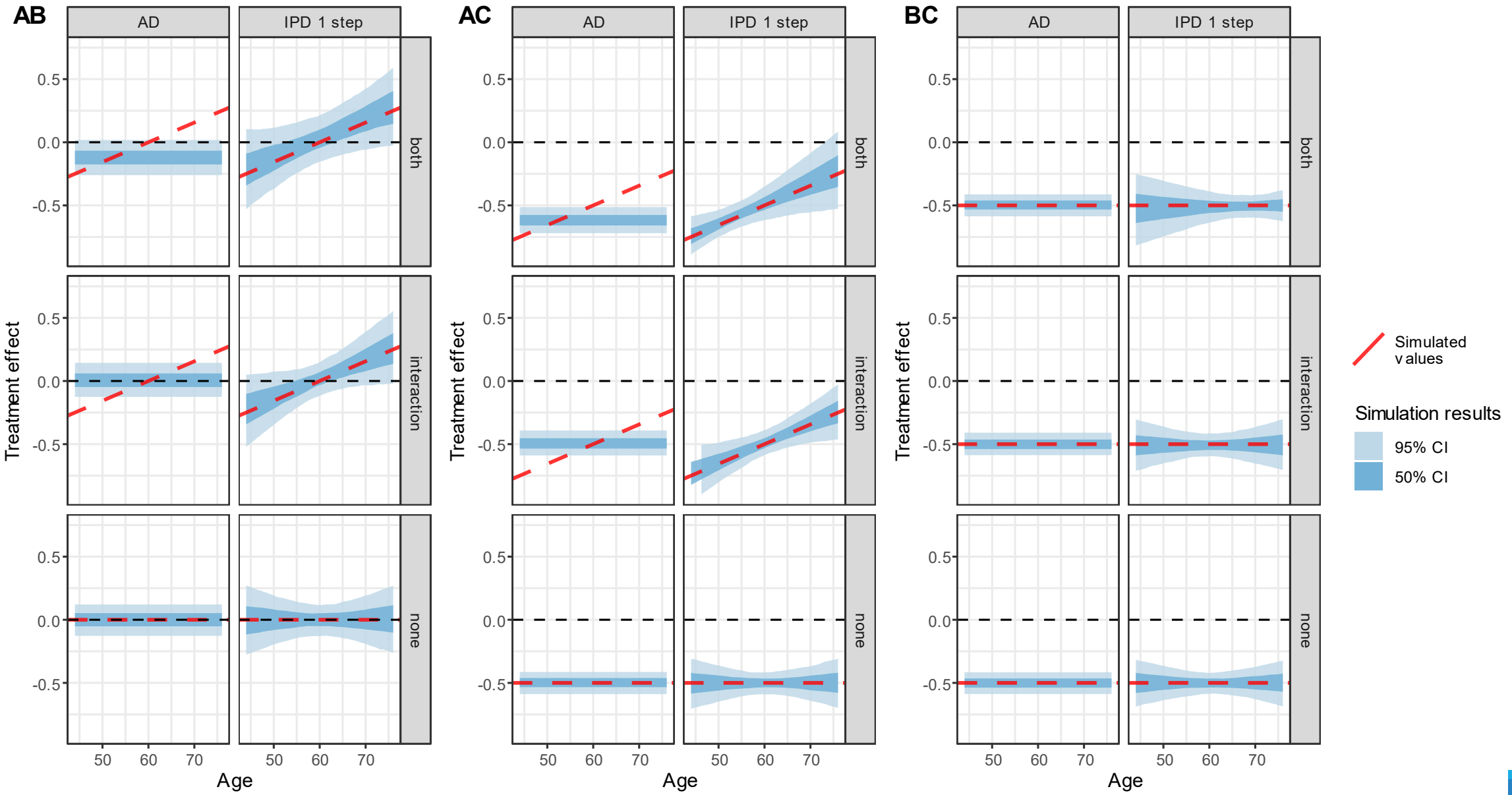
On obtient ce qu'on voulait !

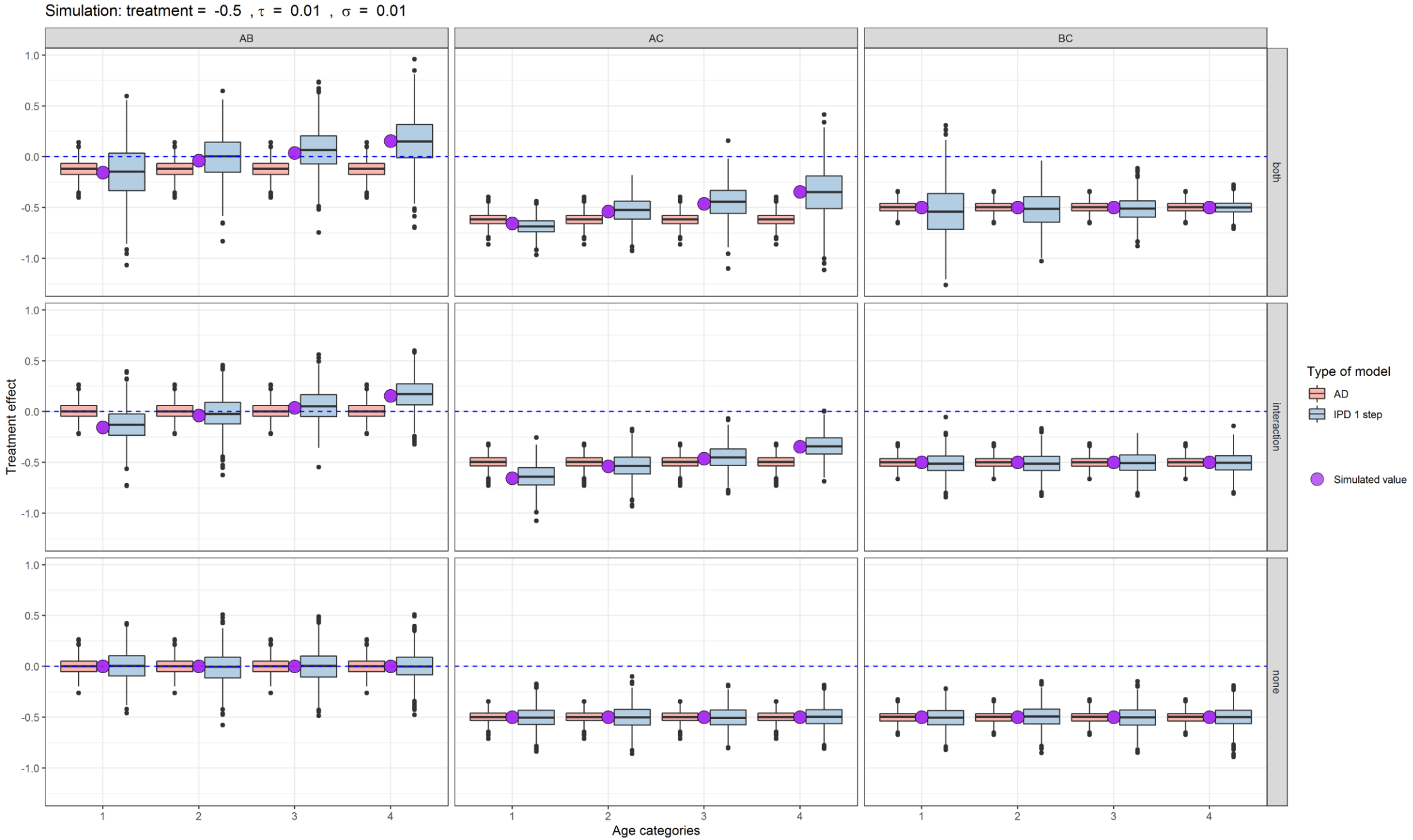
```
final %>% unnest_wider(res)
```

```
# A tibble: 1000 x 7
   n_sim ad_models          ad_nets          ipd_2steps_models  ipd_nets         ipd_ma_models     ipd_1step_models
   <int> <list>             <list>           <list>             <list>           <list>            <list>
 1     1 <tibble [480 x 7]> <tibble [24 x 6]> <tibble [480 x 8]> <tibble [24 x 9]> <tibble [48 x 7]> <tibble [24 x 6]>
 2     2 <tibble [480 x 7]> <tibble [24 x 6]> <tibble [480 x 8]> <tibble [24 x 9]> <tibble [48 x 7]> <tibble [24 x 6]>
 3     3 <tibble [480 x 7]> <tibble [24 x 6]> <tibble [480 x 8]> <tibble [24 x 9]> <tibble [48 x 7]> <tibble [24 x 6]>
 4     4 <tibble [480 x 7]> <tibble [24 x 6]> <tibble [480 x 8]> <tibble [24 x 9]> <tibble [48 x 7]> <tibble [24 x 6]>
 5     5 <tibble [480 x 7]> <tibble [24 x 6]> <tibble [480 x 8]> <tibble [24 x 9]> <tibble [48 x 7]> <tibble [24 x 6]>
 6     6 <tibble [480 x 7]> <tibble [24 x 6]> <tibble [480 x 8]> <tibble [24 x 9]> <tibble [48 x 7]> <tibble [24 x 6]>
 7     7 <tibble [480 x 7]> <tibble [24 x 6]> <tibble [480 x 8]> <tibble [24 x 9]> <tibble [48 x 7]> <tibble [24 x 6]>
 8     8 <tibble [480 x 7]> <tibble [24 x 6]> <tibble [480 x 8]> <tibble [24 x 9]> <tibble [48 x 7]> <tibble [24 x 6]>
 9     9 <tibble [480 x 7]> <tibble [24 x 6]> <tibble [480 x 8]> <tibble [24 x 9]> <tibble [48 x 7]> <tibble [24 x 6]>
10    10 <tibble [480 x 7]> <tibble [24 x 6]> <tibble [480 x 8]> <tibble [24 x 9]> <tibble [48 x 7]> <tibble [24 x 6]>
# ... with 990 more rows
```

Simulation: treatment = -0.5 , τ = 0.01 , σ = 0.01

Comme les données sont rangées il est facile de les exploiter




Simulation: treatment = -0.5 , τ = 0.01 , σ = 0.01
AB
AC
BC
both
interaction
none
Treatment effect
Age categories
Type of model
AD
IPD 1 step
Simulated value

Conclusions

furrr est intéressant si…

On aime le concept de purrr

On aime « nester » des data.frame

Le temps de calcul est long :

- On a vraiment beaucoup de ligne (> 10^6)
- Chaque ligne prend longtemps (plusieurs secondes)

Vraiment facile à initier :

- **plan(multisession)** et c'est tout

Naming facile à retenir

- **future_***

Changer de « mode » en un mot

Merci de votre attention !

Matthieu FARON

matthieu.faron@gustaveroussy.fr

Département de chirurgie viscérale oncologique

Département de biostatistiques et epidémiologie

INSERM 1018 (CESP) ONCOSTAT

GUSTAVE ROUSSY

CANCER CAMPUS GRAND PARIS