

Mode opératoire technique

|  |  |
| --- | --- |
|  | Intégration et chargement des publications/mesh relatives aux NCT en bases postgre |
| V1.0 |

***Paris, le 21/02/2022***

***A destination de :*** *Frédéric LEGRAND et Stéphane DE GRAEVE*

Contenu

[1- CONTEXTE 4](#_Toc96703749)

[1.1- Structure projet 4](#_Toc96703750)

[1.2- Organisation des répertoires 4](#_Toc96703751)

[2- LES TRAITEMENTS 6](#_Toc96703752)

[2.1- Bloc INIT 6](#_Toc96703753)

[2.2- Bloc PUBMED 7](#_Toc96703754)

[2.2.1- Le cœur du traitement 7](#_Toc96703755)

[2.2.2- Le fichier de suivi 8](#_Toc96703756)

[2.2.3- Mécanique de reprise 9](#_Toc96703757)

[2.2.4- Etapes de vérification 9](#_Toc96703758)

[2.2.5- Comptage des balises 10](#_Toc96703759)

[2.2.6- Chargement postgre 12](#_Toc96703760)

[2.3- Blocs Référentiels Mesh 14](#_Toc96703761)

[2.3.1- Traitements XML 14](#_Toc96703762)

[2.3.2- Etapes de vérification 16](#_Toc96703763)

[2.3.3- Comptage des balises 16](#_Toc96703764)

[2.3.4- Chargement postgre 16](#_Toc96703765)

[2.4- Bloc tree codes 16](#_Toc96703766)

[2.5- Log et temps de traitement 17](#_Toc96703767)

[3- Gestion des schémas 18](#_Toc96703768)

[4- MCD 19](#_Toc96703769)

# 1- CONTEXTE

Ce document vient en complément du DAT et ne présentera pas la partie déploiement du projet. Le *modop* suit le déroulement du programme principal ***01-MAIN\_Traitements.R*** (*. /Prod/*).

**La première étape** est d’ouvrir le projet sur R en se plaçant à la racine du projet et en double cliquant sur « L2-PUBMED » (.Rproj). Cela est nécessaire pour ouvrir le projet correctement et cela initie la valeur du chemin du répertoire de travail (tous les chemins relatifs démarrent à la racine du projet).

## 1.1- Structure projet

L’ensemble du projet est construit autour d’un programme principal. Nous allons voir une vision globale de son mécanisme.

*Les parties 3 à 7 sont des référentiels*

CHARMENT DES TABLES POSTGRES + GESTION DES SCHEMAS

LOG ET FICHIERS TEMPS DE TRAITEMENT

FICHIERS OUTPUT

SCRIPT DE PARAMETRAGE

FICHIERS INPUT

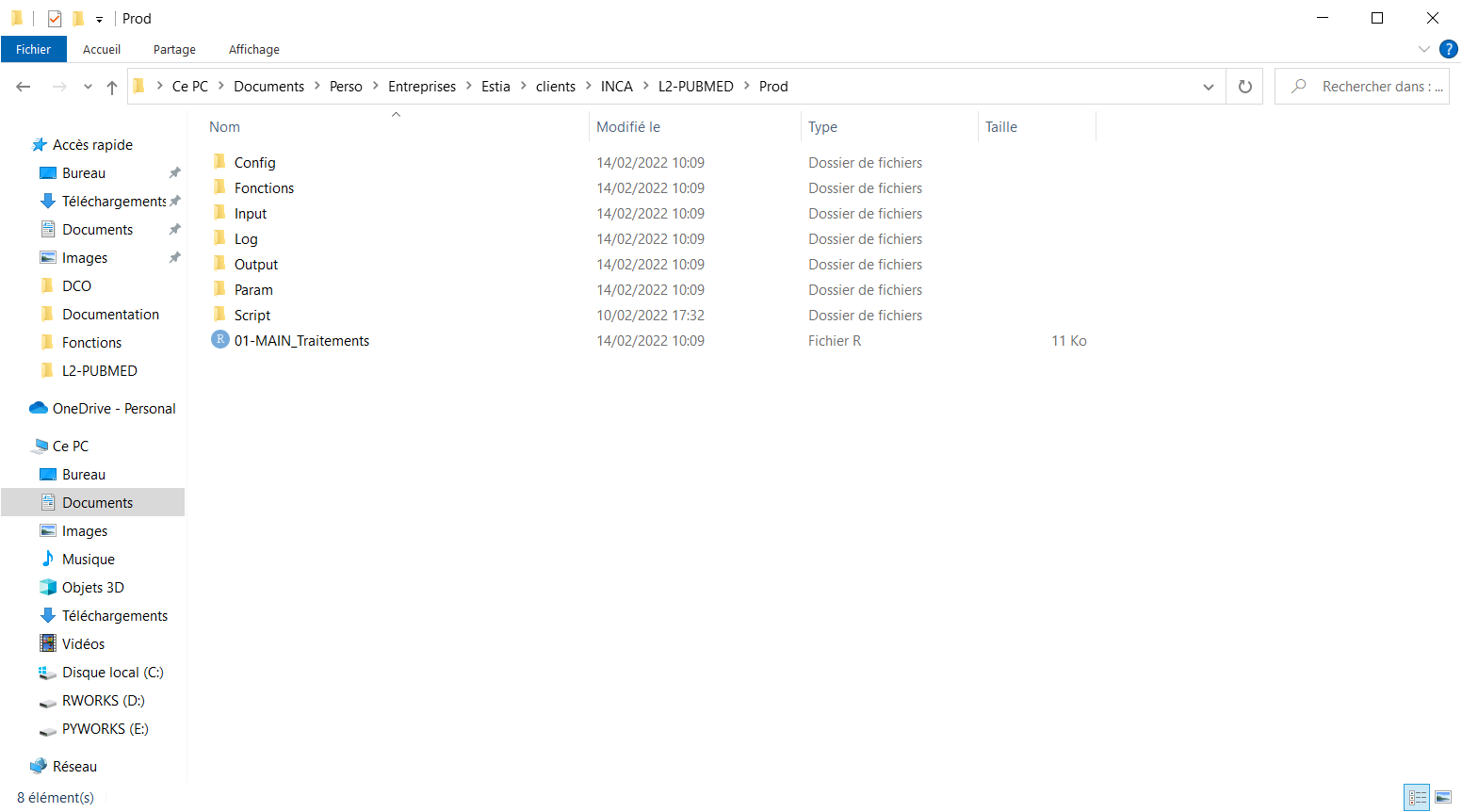
Programme principal – « ***MAIN*** »

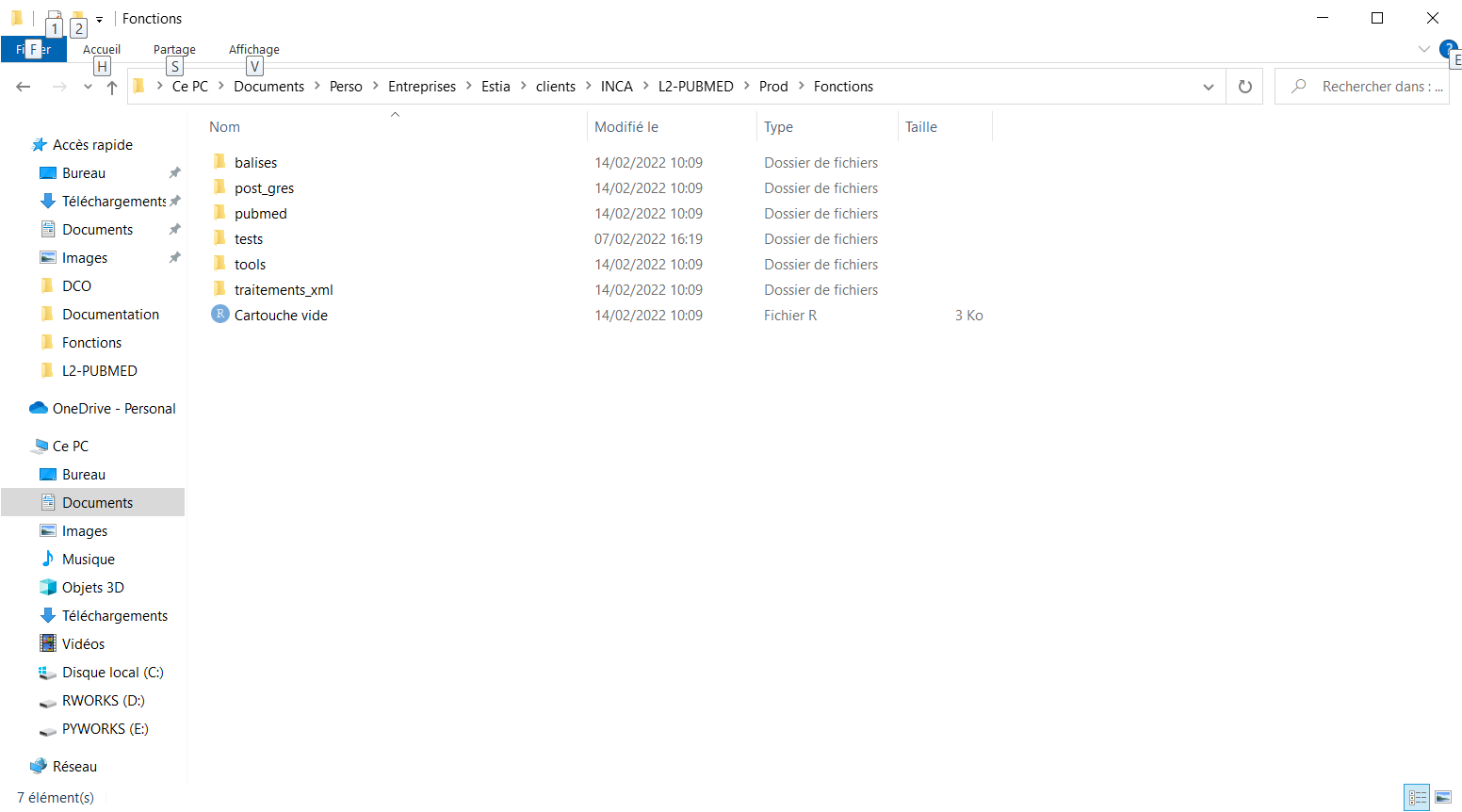
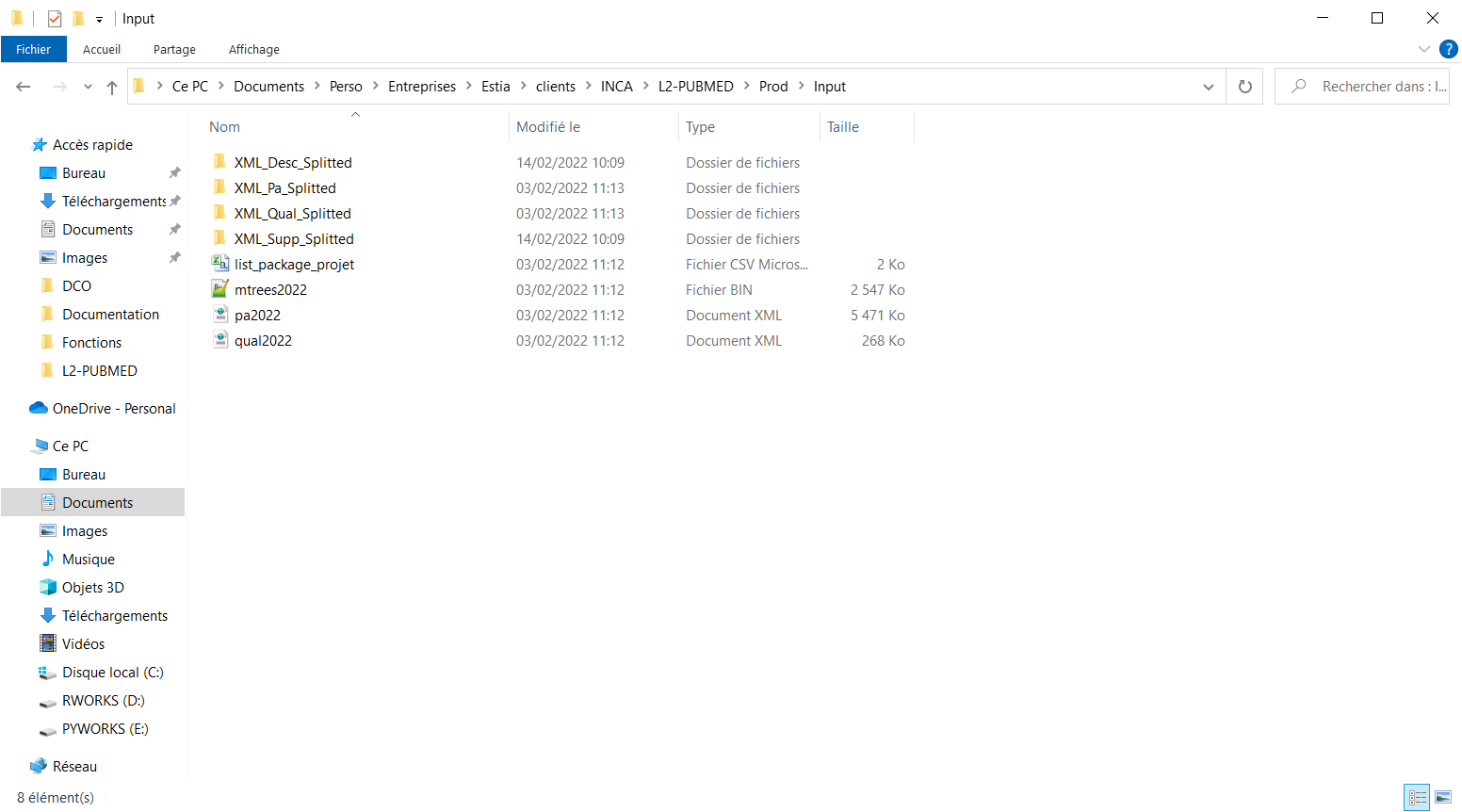
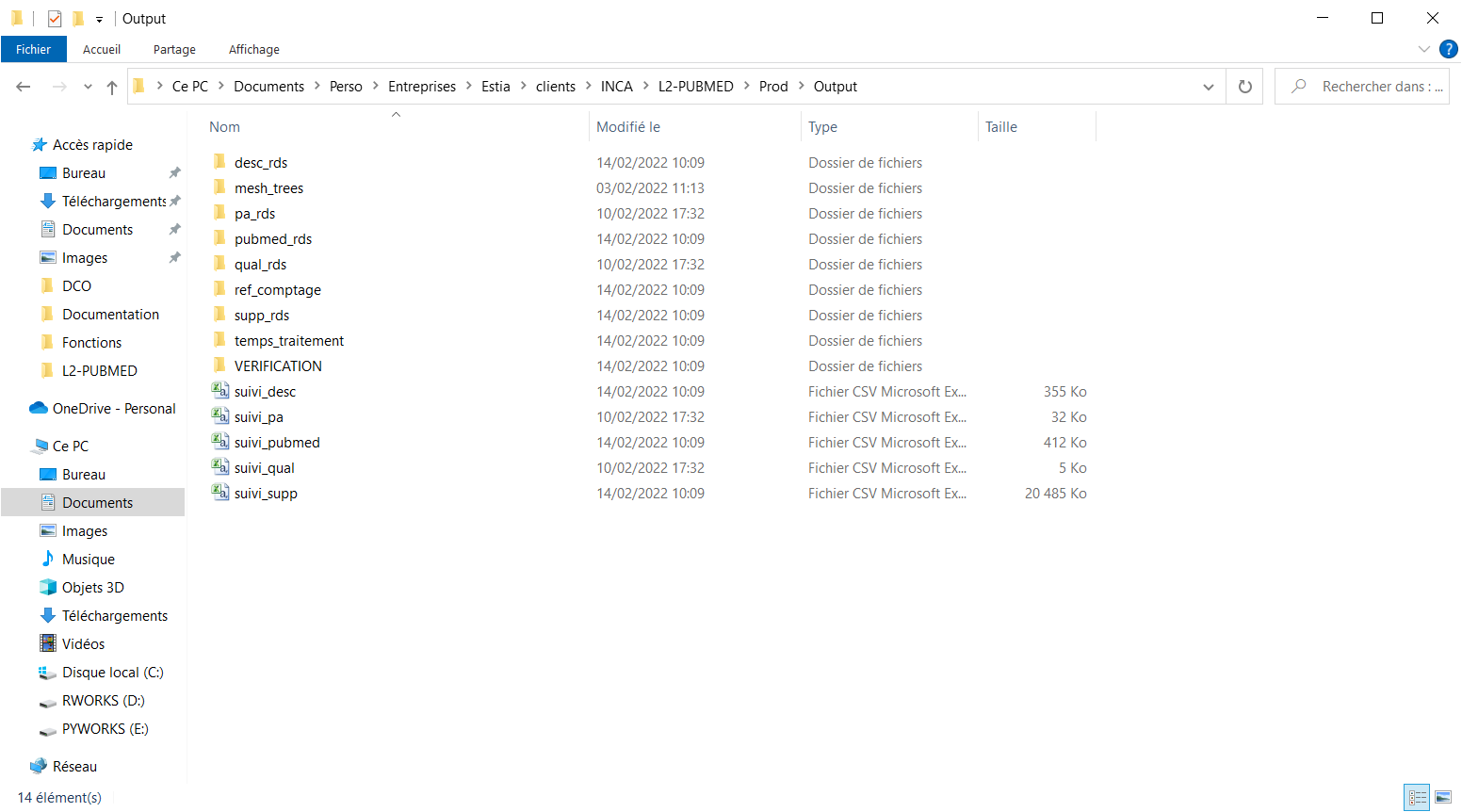
1. BLOC – INIT
2. BLOC – PUBMED
3. BLOC – DESC
4. BLOC – SUPP
5. BLOC – PA
6. BLOC – QUAL
7. BLOC – TREE CODES
8. BLOC – GESTION SCHEMAS POSTGRE

## 1.2- Organisation des répertoires

A la racine du projet nous avons le dossier Prod/ qui est le point d’entrée du programme. Tous les chemins relatifs commencent par Prod/… ou ./Prod/….

**Architecture des répertoires de travail :**



* Config/ : Contient les informations de connexion aux bases postgre
* Fonctions/ : Contient toutes les fonctions nécessaires au projet
  + - * + 
        + Les fonctions sont placées dans des répertoires spécifiques pour faciliter la navigation.
* Input/ : Contient tous les éléments nécessaires au projet
  + - * + Les référentiels MESH (desc, supp, pa, qual)
        + [Lien référentiel MESH](https://nlmpubs.nlm.nih.gov/projects/mesh/2021/xmlmesh/)
        + Une arborescence de fichiers dédiée au traitement des référentiels au format XML
        + 
      * Log/ : Contient l’ensemble des fichiers de log nommés par programme et datés
      * Output/ : Contient à la racine des fichiers de suivi des traitements (pubmed + référentiels) et une arborescence de sous répertoires de stockage
        + 
        + Les sous répertoires sont principalement des fichiers RDS contenant de la DATA (pubmed ou référentiels)
        + Nous pouvons trouver d’autres répertoires pour les temps de traitement des programmes, les référentiels des balises ou un répertoire de vérification de fichiers RDS.
      * Param/ : Contient le script de paramétrage (variables globales + chargement des librairies)
      * Script/ : Contient un script d’installation automatique des packages

# 2- LES TRAITEMENTS

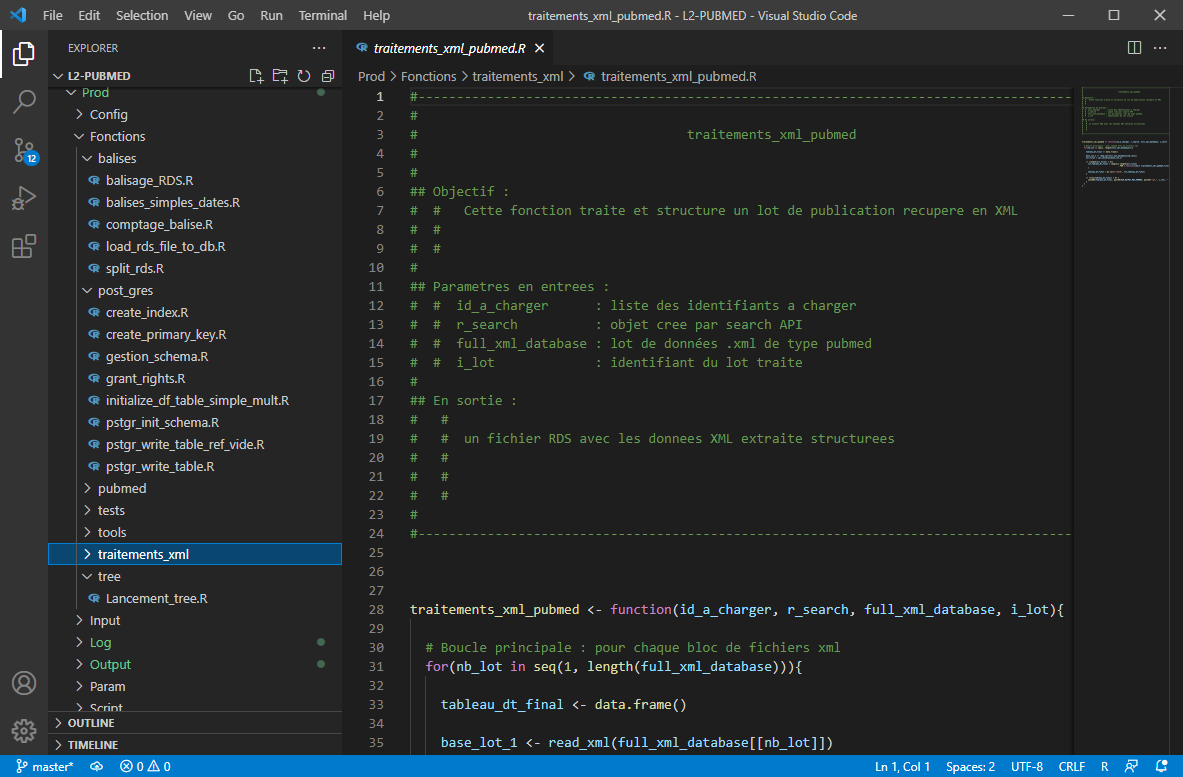
Les traitements sont lancés à l’aide du programme principal Prod/ 01-MAIN\_Traitements.R. L’organisation du script de lancement est faite par BLOC :

1. BLOC – INIT
2. BLOC – PUBMED
3. BLOC – DESC
4. BLOC – SUPP
5. BLOC – PA
6. BLOC – QUAL
7. BLOC – TREE CODES
8. BLOC – GESTION SCHEMAS POSTGRE

## 2.1- Bloc INIT

Le projet permet de lancer chaque BLOC de façon indépendante mais implique d’exécuter le premier BLOC dédié au chargement des paramètres (déclaration des variables globales et chemins) et au chargement de l’ensemble des fonctions en mémoire.

1. **source**("Prod/Param/01-00-Definitions\_parametres\_globaux.R") : Exécution du script de paramétrage
2. **sapply**(FILE\_SOURCES,source) : Exécution des fonctions situées dans le répertoire (Prod/Fonctions/)



*L’ensemble des scripts contenant les fonctions sont exécutés de façon dynamique*

## 2.2- Bloc PUBMED

Cette partie va se charger du téléchargement des publications, des traitements DATA, déterminer les champs des tables simples et multiples puis effectuer le chargement des tables en bases postgre.

### 2.2.1- Le cœur du traitement

Le bloc démarre sur l’appel de la fonction Lancement\_PUBMED(), la description technique est présente dans le cartouche de la fonction (Prod/Fonctions/pubmed/Lancement\_PUBMED.R).

Requêtes API

Fonction : Lancement\_PUBMED()

Suivi des téléchargements

Paramètres en entrée :

* *stop\_index\_id*
* *reprise*

Traitements des fichiers XML (fichiers .RDS en output)

**API pubmed** :

1. Recherche des publications pubmed avec le critère "(ClinicalTrials.gov[Secondary Source ID])"
   1. Cela permet de cibler les publications pubmed qui font références à une publication NCT sur le site <https://www.clinicaltrials.gov/> (voir la fonction, Prod/Fonctions/pubmed/searchAPI.R)
2. A partir de la liste des id, télécharger les publications au format XML
   1. Des options sont nécessaires pour optimiser les téléchargements (voir la fonction, Prod/Fonctions/pubmed/fetchAPI.R)

**La fonction Lancement\_PUBMED() doit être lancée deux fois** :

1. Chargement avec une limite (permet d’éviter un « time out » autour de 77K id sur l’API)
   1. Lancement\_PUBMED(stop\_index\_id = 60000, reprise = FALSE)
2. Chargement du reste des publications avec l’option de reprise
   1. Lancement\_PUBMED(stop\_index\_id = 999999,reprise = TRUE)

Le paramètre stop\_index\_id = 999999 signifie un lancement sur la totalité des id récupérés depuis l’API.

Chaque publication au format XML est traitée, convertie en data frame puis exportée.

**En sortie de la fonction**:

1. Nous avons des fichiers .RDS en OUTPUT (499 publications par fichier .RDS). Ces fichiers .RDS sont stockés dans le répertoire /Prod/Output/pubmed\_rds/ (Le nombre de publications est fixé en paramètre STEP\_BY\_ID = 499).
2. Un fichier de suivi est créé en sortie (/Prod/Output/suivi\_pubmed.csv)

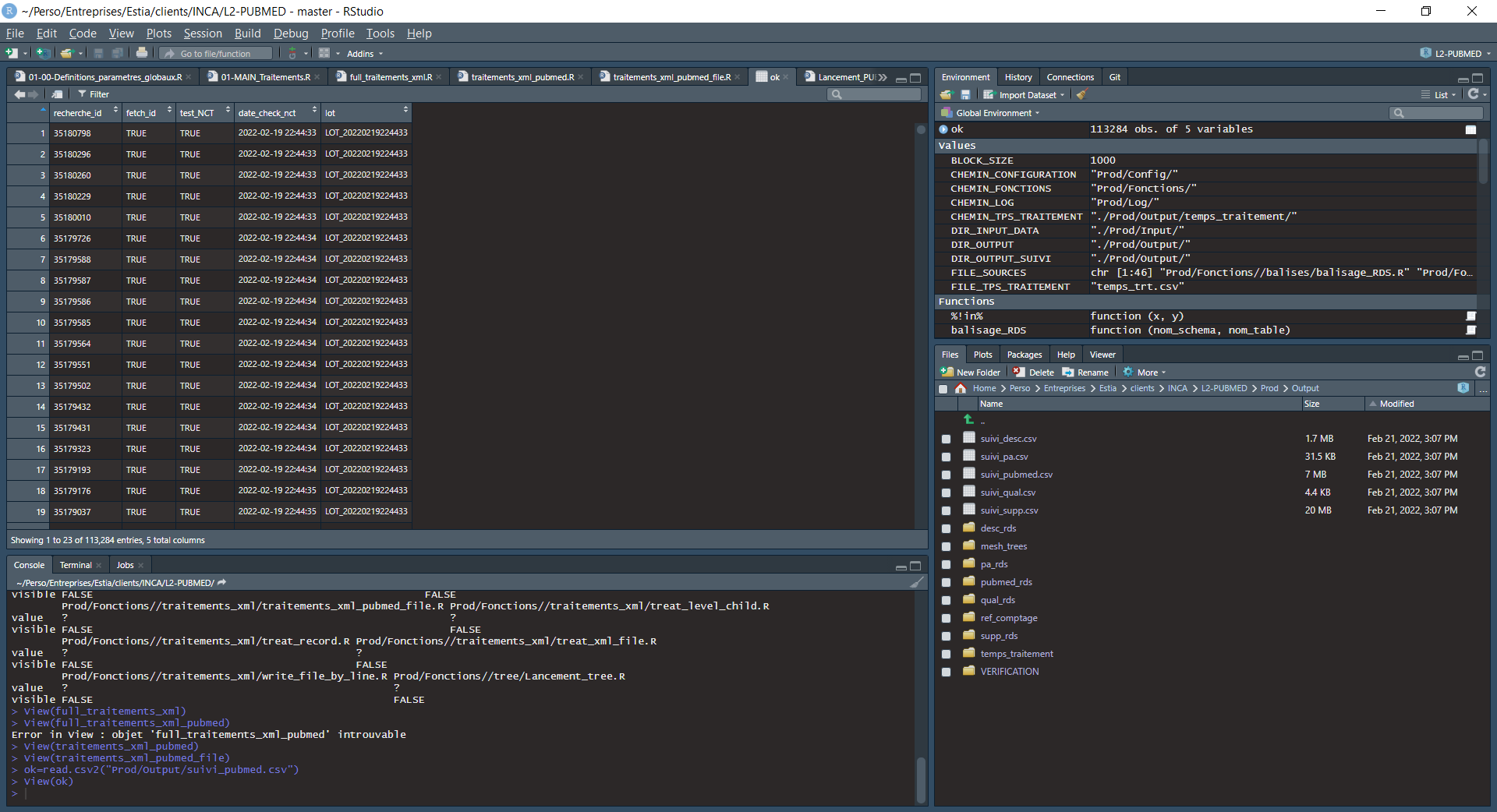
### 2.2.2- Le fichier de suivi

Un fichier de suivi au format *csv* (/Prod/Output/suivi\_pubmed.csv) est créé lors de l’appel de la fonction Lancement\_PUBMED() :

* Si le paramètre reprise = FALSE alors un nouveau fichier de suivi est créé (suppression d’un fichier de suivi déjà existant)
* Si le paramètre reprise = TRUE alors le fichier de suivi est utilisé pour effectuer une reprise des traitements

Le fichier de suivi permet plusieurs contrôles :

* Tracer l’id de la publication pubmed fournit par l’API
* Vérifier pour chaque id listé si il est bien présent dans les téléchargements
* Vérifier la présence d’un NCT dans la publication (Prod/Fonctions/pubmed/check\_xml.R)
* Tracer dans quel ficher .RDS est présent l’id (la publication)
* Tracer la date du traitement



Ce fichier permettra le suivi du déroulement total des traitements des fichiers XML et aussi d’assurer la partie reprise.

### 2.2.3- Mécanique de reprise

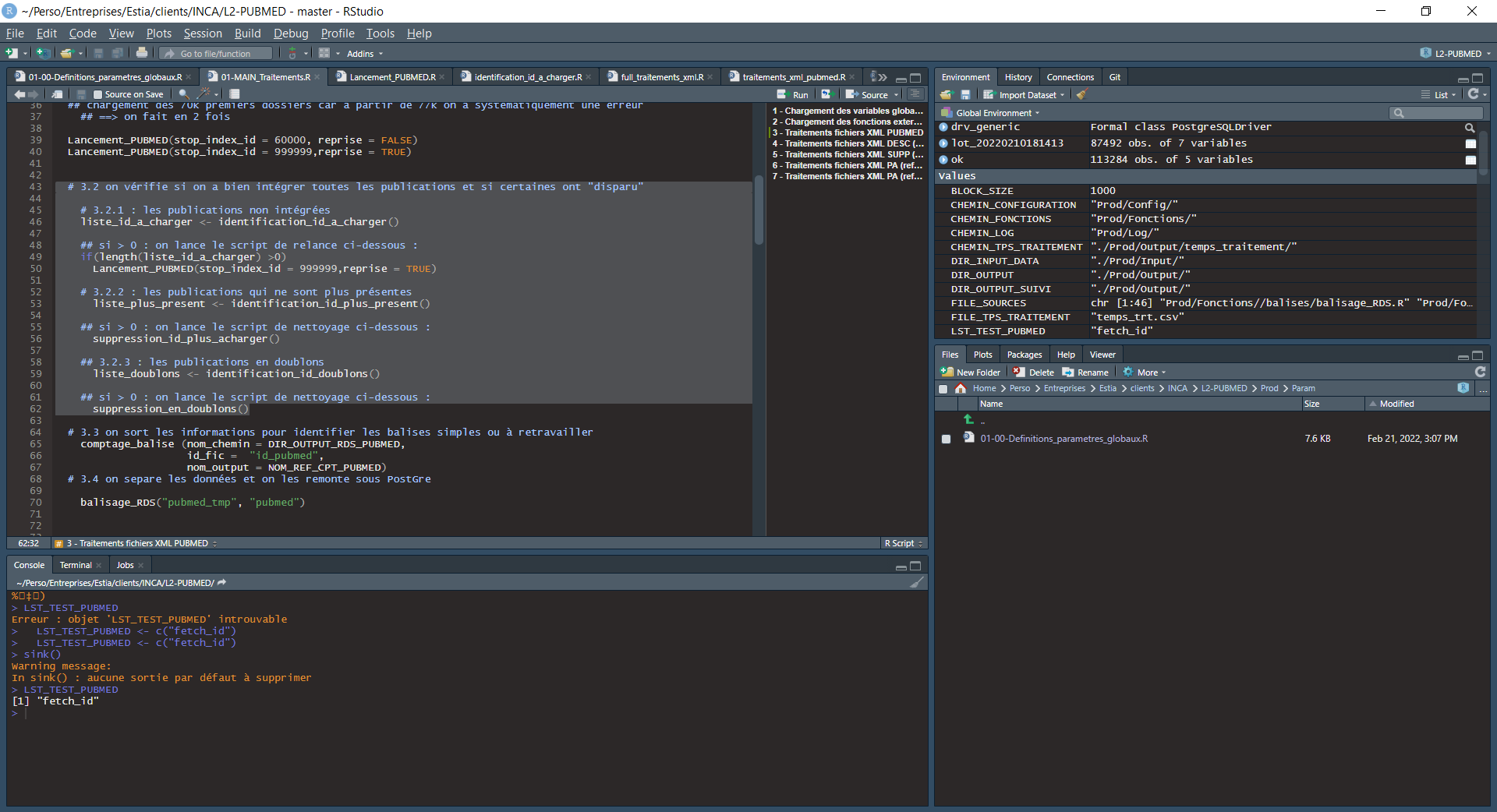
Si pour des raisons diverses, le téléchargement s’interrompt ou un problème survient, il suffit de lancer la fonction Lancement\_PUBMED(stop\_index\_id = 999999,reprise = TRUE) avec l’option *reprise = TRUE*.

Suite à cette action, le fichier de suivi (/Prod/Output/suivi\_pubmed.csv) sera utilisé.

1. Nous allons lister les id qui ont bien été traités et stockés au format .RDS. Nous avons cette information si dans le fichier de suivi, la colonne *fetch\_id* est égale à *TRUE*.
2. Nous requêtons à nouveau l’API pour récupérer la liste des id (publications pubmed comportant un NCT)
3. On lance à nouveau le traitement sur les id non présents dans le fichier de suivi

### 2.2.4- Etapes de vérification

Une fois que le téléchargement et le traitement des publications pubmed sont terminés, des fonctions sont chargées de faire les contrôles.



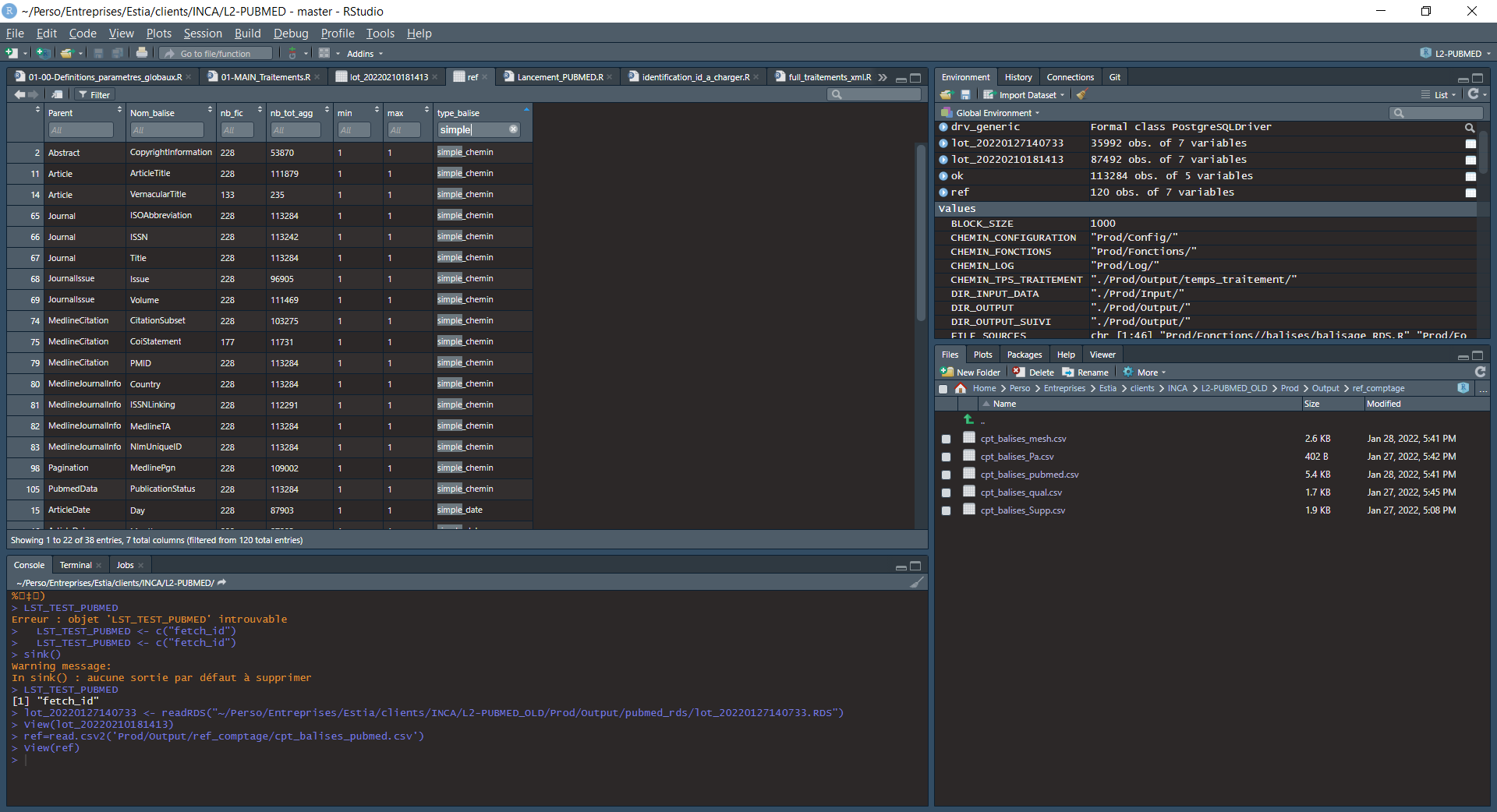
1. **Vérifier les id téléchargés** : la fonction identification\_id\_a\_charger() ouvre le fichier de suivi puis effectue une requête API pour lister les id à télécharger (la fonction retourne un vecteur pour les id)
2. **Lancer une reprise si besoin** : à partir du vecteur retourné par la fonction identification\_id\_a\_charger(), si le vecteur comporte des id non présents une reprise est lancée
3. **Identifier si une publication existe encore**: la fonction identification\_id\_plus\_present() effectue une mise à jour de la liste des id en requêtant l’API. Si un id n’est plus dans cette liste il est identifié et doit être supprimé
4. **Nettoyer et supprimer les publications** : la fonction suppression\_id\_plus\_acharger() ne garde que les id à jour
5. **Identifier et supprimer les doublons**:
   1. la fonction identification\_id\_doublons() recherche et identifie des doublons dans le fichier de suivi
   2. la fonction suppression\_en\_doublons() à partir de la liste de doublons fournit en sortie par la fonction identification\_id\_doublons() procède à la suppression des id dans les fichiers .RDS

### 2.2.5- Comptage des balises

Cette étape va permettre d’identifier des balises dites « simples » ou « multiples ». Pour cela, nous allons créer un fichier de comptage à partir de toutes les publications (tous les fichiers .RDS).

Ce fichier de comptage permettra d’identifier la fréquence d’apparition des balises. Plus précisément, nous nous intéresserons au couple (« **Parent** », « **Nom\_balise** »).

Le traitement des balises se fait dans la fonction Prod/Fonctions/balises/comptage\_balise.R.



A partir de la table de comptage, nous allons créer le référentiel de comptage des balises. Il existe un référentiel de comptage pour chaque source de données (pubmed + les référentiels Mesh). Le référentiel de comptage est présent en sortie dans le répertoire Prod/Output/ref\_comptage/cpt\_balises\_pubmed.csv.

Ce fichier pourra être challengé sur la partie encadrée en rouge correspondant au type de la balise.

**Variables de comptage** :

A partir d’une vision agrégée des données, on compte pour chaque publication l’apparition des couples **Parent/Nom\_balise** et on créé des variables de comptage :

* nb\_fic : le nombre total d’apparitions dans les fichiers .RDS (un fichier .RDS contient plusieurs publications)
* nb\_tot\_agg : le nombre total d’apparitions (plusieurs fois par fichier .RDS)
* min : le nombre minimal d’apparitions par publication
* max : le nombre maximal d’apparitions par publication

**Les deux familles principales de balise** :

1. **Les balises simples** : le couple **Parent/Nom\_balise** n’apparait qu’une seule fois (**min** et **max** sont égaux à 1)
2. **Les balises multiples**: le couple **Parent/Nom\_balise** apparait avec un **max** différent de 1

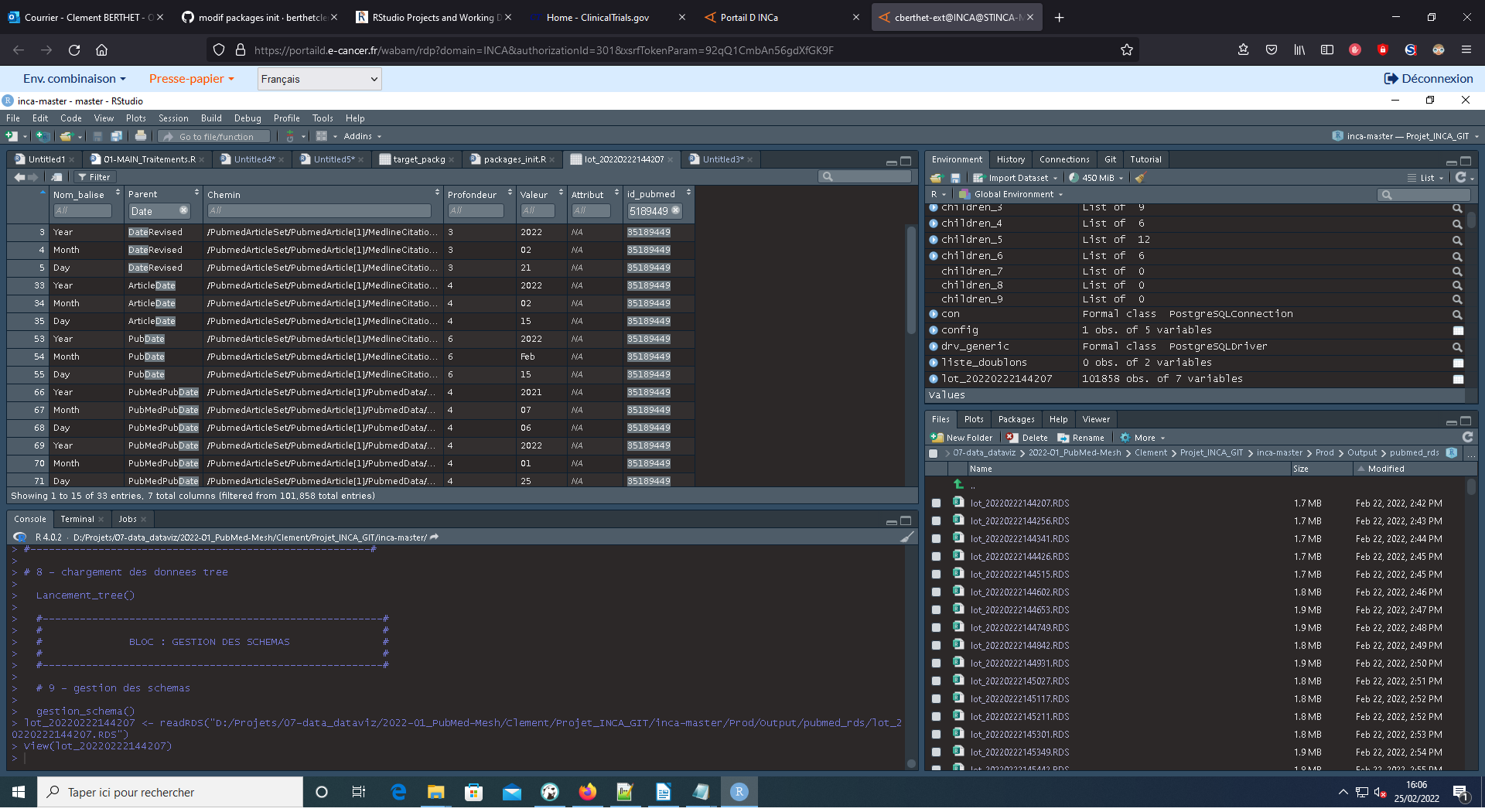
**Définition des types de balises** :

Nous avons identifié 120 couples (Parent, Nom\_balise) et 7 types de balise qui permettront de découper les fichiers .RDS en tables simples et multiples.

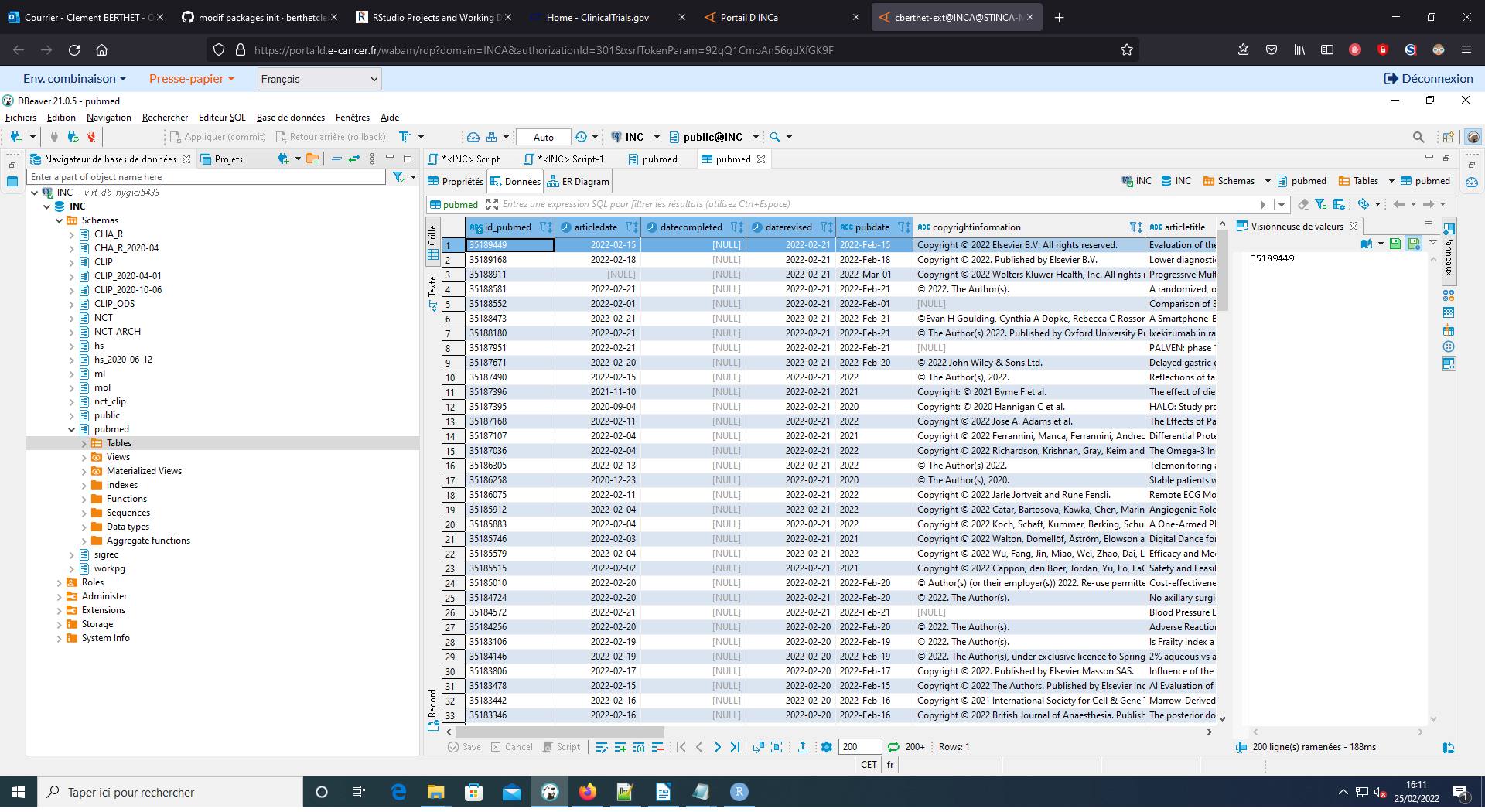
|  |  |  |  |
| --- | --- | --- | --- |
| **NOM** | **MIN** | **MAX** | **REGLES DE GESTION** |
| simple\_chemin | 1 | 1 |  |
| simple\_date | 1 | 1 | Parent contient « Date » |
| simple\_PubDate | 1 | 1 | Parent contient « Pubdate » |
| simple\_non\_significative | 1 | 1 | nb\_tot\_agg < 0.1% \* nb balises simples |
| simple\_erronnee | 1 | 1 | Parent ou Nom\_balise contient ("b" "i", …)[[1]](#footnote-1) |
| multiple | 1 | >= 1 |  |
| multiple\_erronnee | 1 | >=1 | Parent ou Nom\_balise contient ("b" "i", …) |

**Cas particuliers** :

Quelques précisions sur les tables simples concernant les dates. Les dates sont découpées par année/mois/jour et sur 3 lignes dans les fichiers .RDS.



Sur ces dates, dans les tables simples, n’apparaitra qu’une seule ligne date avec les données année/mois/jour.



Nous avons aussi identifié des balises interprétées à tort dans les fichiers XML récupérés depuis l’API. L’anomalie se produit dans des paragraphes de texte (abstract principalement) ou des formules mathématiques sont écrites et mises en forme avec Latex.

Si une publication comporte des balises taguées « simple\_erronnee » ou « multiple\_erronnee » nous taguerons cette information dans les tables simples ou multiples (champ « balises\_error »).

### 2.2.6- Chargement postgre

Cette partie est la dernière partie du BLOC pubmed. Une fois que le référentiel de comptage a été validé, il servira au découpage des fichiers .RDS contenant les publications XML en tables simples/multiples.

L’ensemble des étapes est traitée par la fonction balisage\_RDS() (Prod/Fonctions/balises/balisage\_RDS.R). Cette fonction **générique** qui sera réutilisée dans les blocs dédiés aux référentiels prend deux paramètres en entrée :

1. nom\_schema = «pubmed\_tmp»
2. nom\_table = «pubmed»

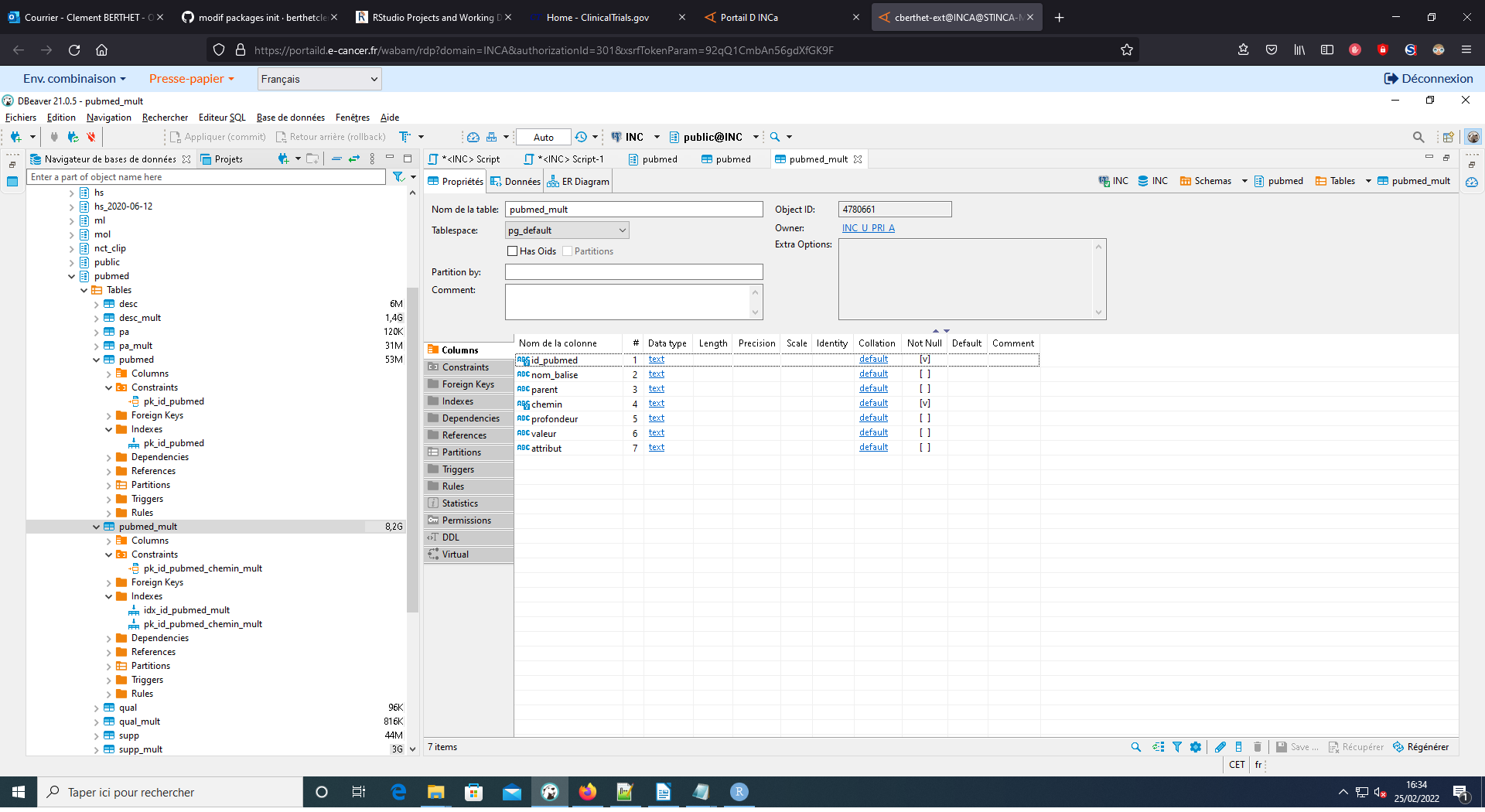
**Schéma fonctionnel de la fonction** :

1. Création du schéma si non existant
2. Création des tables vides simples/multiples à partir du référentiel de comptage + typage des colonnes
3. Création des clés primaires/index
4. Chargement des tables simples/multiples dans l’environnement postgre INCA

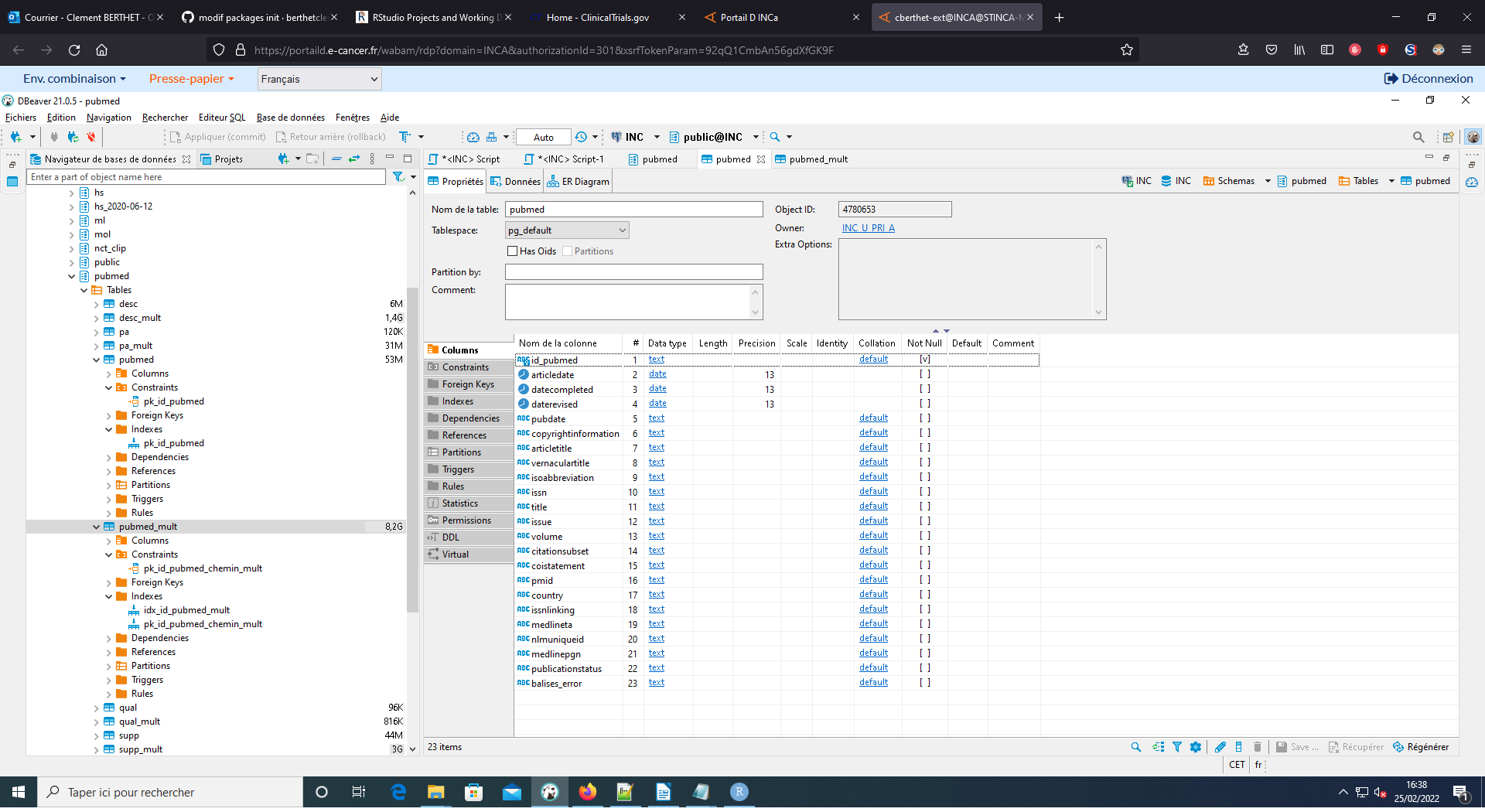
Fonction : balisage\_RDS()

**Structure de la table vide** :

Les champs des **tables multiples** sont fixés :



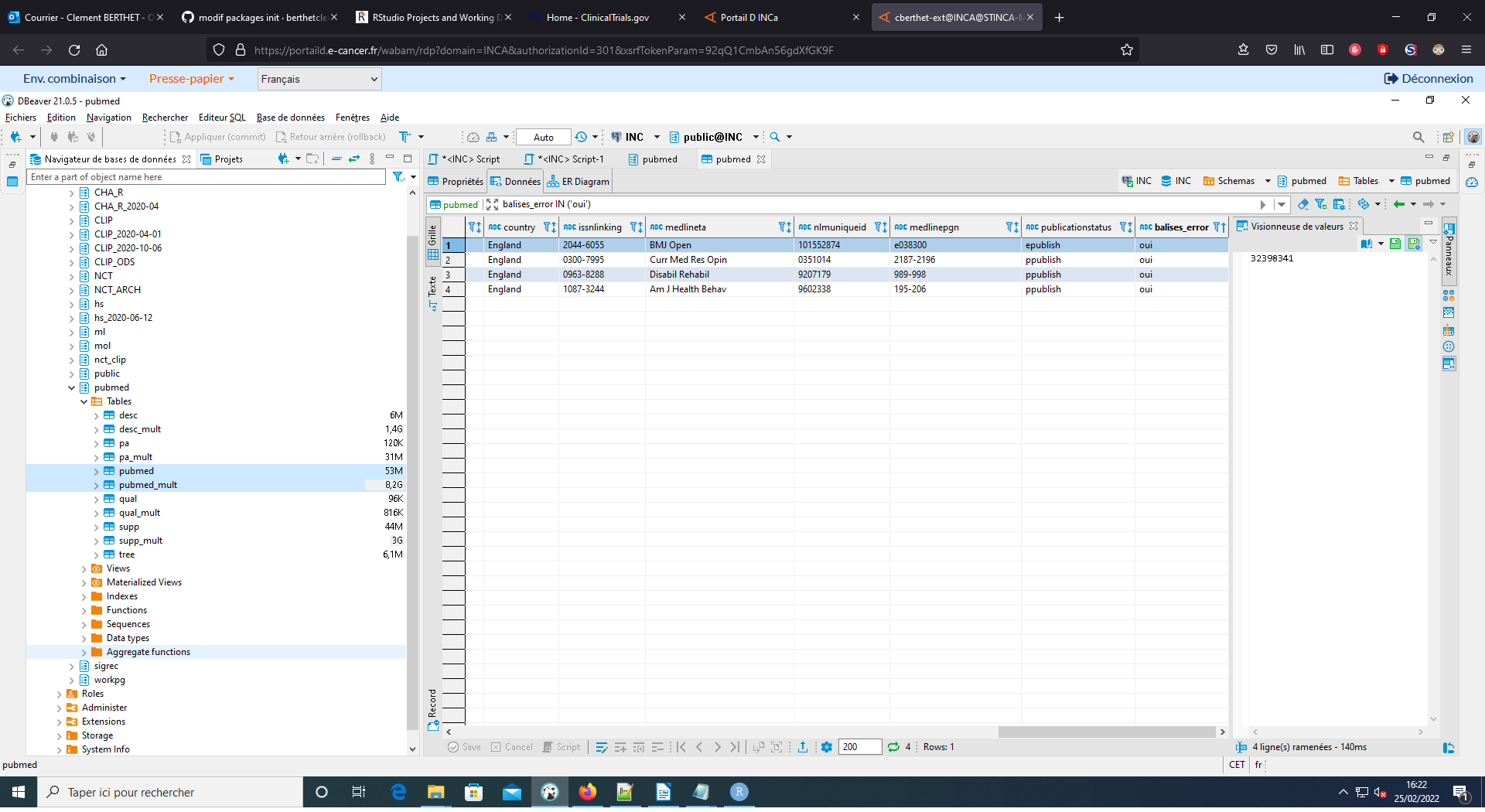
Les champs des **tables simples** peuvent évoluer selon les publications pubmed. Les champs sont définis à partir du référentiel de comptage.



**En sortie** :

Nous avons les tables simples/multiples en bases postgre :

* Le nom de la table simple est le nom passé en paramètre de la fonction (nom\_table=« pubmed »)
* Le nom de la table multiple est fabriqué à partir du nom de la table simple en ajoutant « \_mult »



## 2.3- Blocs Référentiels Mesh

Cette partie résume l’ensemble des BLOCS Mesh. Chaque BLOC a la même structure et se base sur des fonctions génériques.

Les référentiels sont stockés dans le répertoire Prod/Input et sont disponibles via le lien [xmlmesh](https://nlmpubs.nlm.nih.gov/projects/mesh/2021/xmlmesh/).

**L’ensemble des BLOCS** :

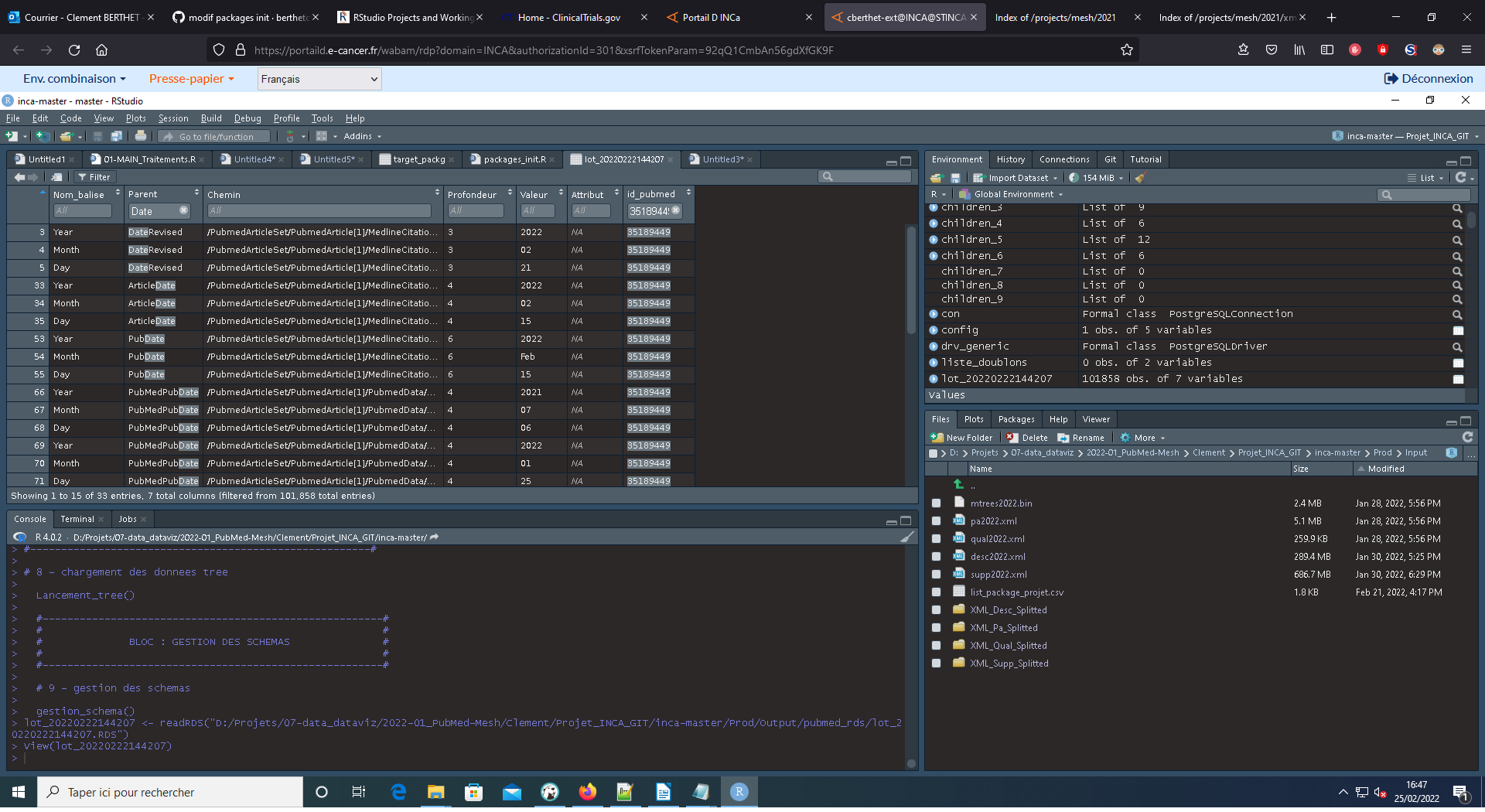
1. BLOC – DESC
2. BLOC – SUPP
3. BLOC – PA
4. BLOC – QUAL

Pour chaque BLOC, nous avons 4 étapes :

1. Le traitement du référentiel en INPUT
2. Vérification des codes Mesh avant et après traitements
3. Une étape pour les balises (création et validation d’un référentiel de comptage)
4. Chargement des tables simples/multiples dans les bases postgre

### 2.3.1- Traitements XML

La fonction traitements\_xml\_by\_type() se charge de lire les fichiers de référentiel en INPUT (Prod/Input).



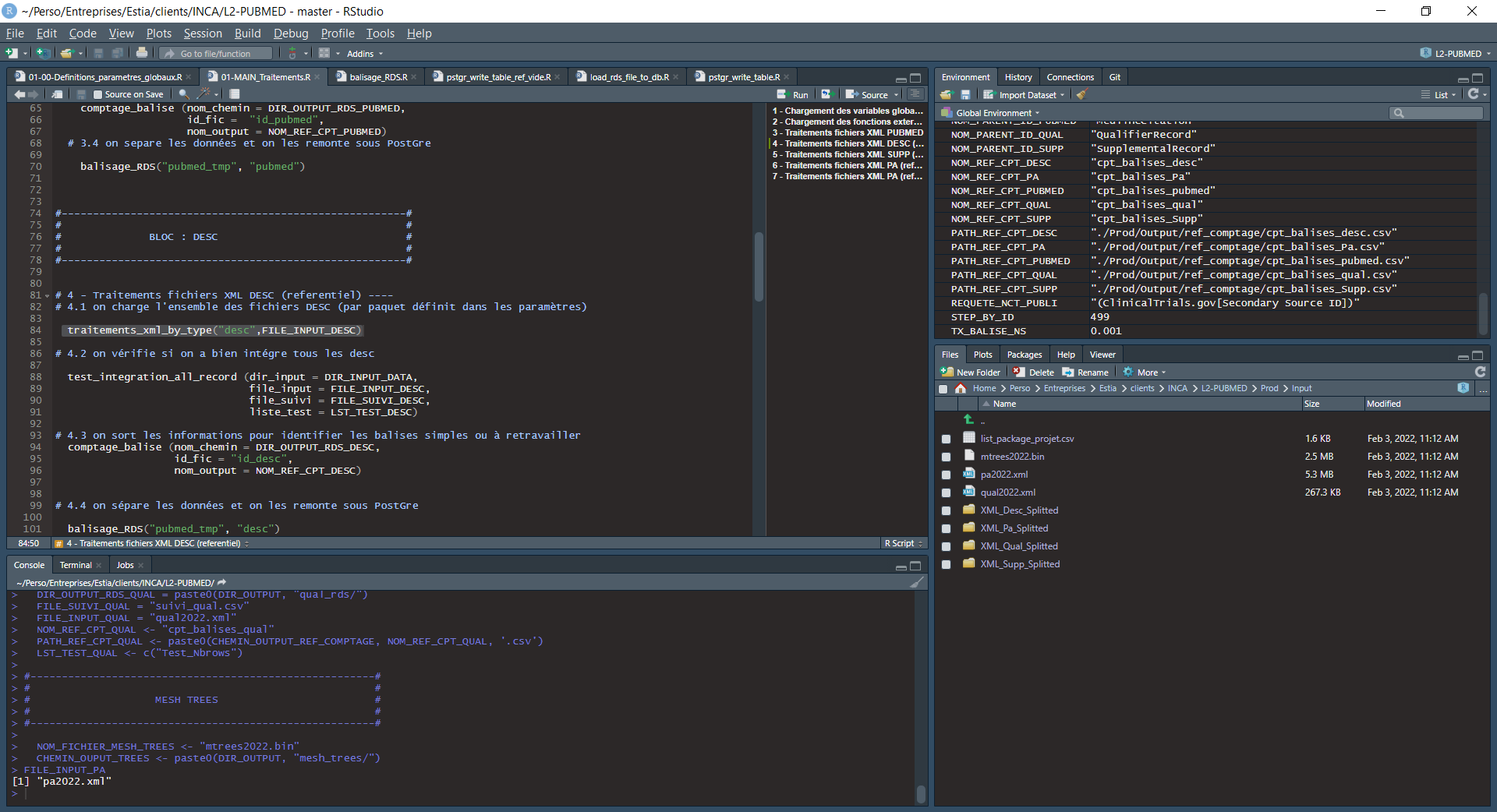
Cette fonction est générique et possède 2 paramètres :

1. Le paramètre type = « desc » ou « supp » ou « pa » ou « qual » selon le BLOC
2. Le paramètre file\_input = FILE\_INPUT\_DESC (nom du référentiel)

Les noms des référentiels sont stockés dans des objets et définis dans le script de paramétrage :

|  |  |  |
| --- | --- | --- |
| **Nom de l’objet** | **Valeur** | **BLOC** |
| FILE\_INPUT\_DESC | "desc2022.xml" | DESC |
| FILE\_INPUT\_SUPP | "supp2022.xml" | SUPP |
| FILE\_INPUT\_PA | "pa2022.xml" | PA |
| FILE\_INPUT\_QUAL | "qual2022.xml" | QUAL |

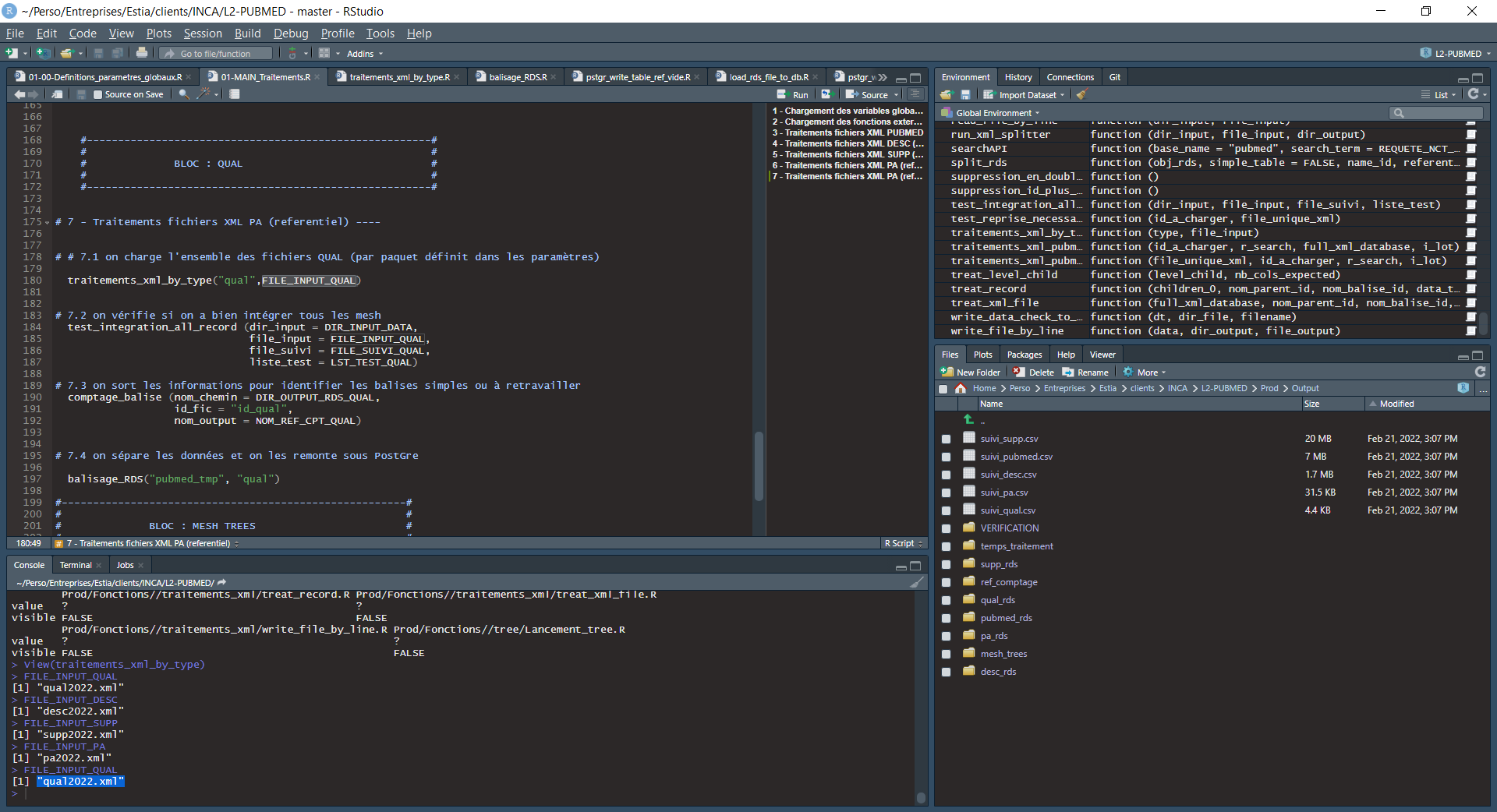
Cette fonction pour gagner du temps machine va splitter les référentiels XML avant de les traiter.



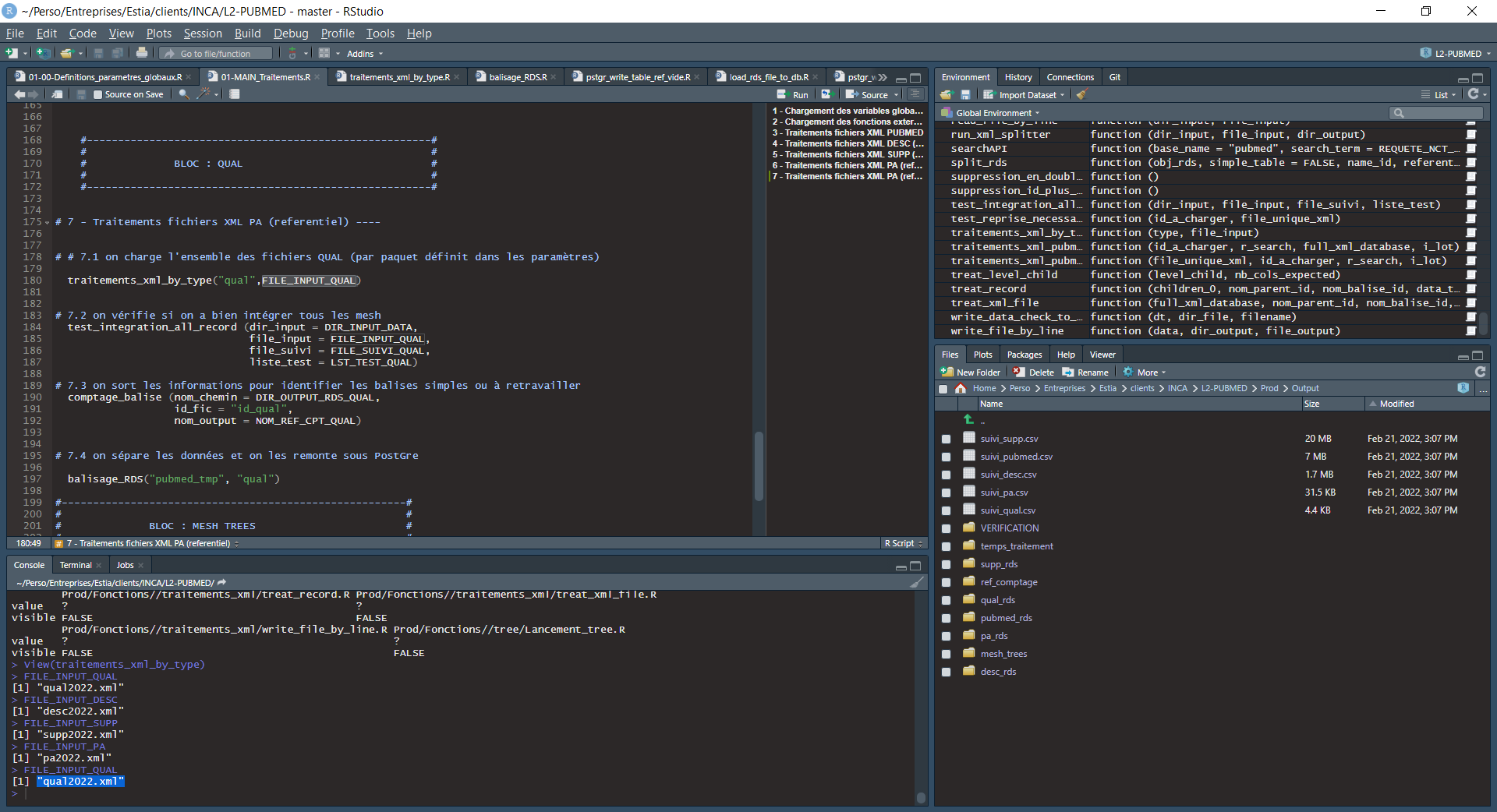
Ces fichiers XML splittés sont stockés en INPUT dans des répertoires spécifiques et identifiés.

**En sortie de la fonction** :

1. Tout comme dans la partie pubmed, nous stockons le contenu des fichiers splittés dans des fichiers .RDS. Ces fichiers .RDS sont placés dans le répertoire Prod/Output/desc\_rds (chaque référentiel a son répertoire : supp\_rds/pa\_rds/qual\_rds).



1. Un fichier de suivi des id mesh traités est créé pour chaque référentiel



### 2.3.2- Etapes de vérification

La fonction test\_integration\_all\_record() vérifie la bonne intégrité des id Mesh entre les fichiers XML et les id Mesh traités en sortie dans fichiers .RDS.

La fonction prend en entrée 4 paramètres qui seront interchangeables selon les référentiels :

1. dir\_input = DIR\_INPUT\_DATA (toujours le même chemin en Input)
2. file\_input = FILE\_INPUT\_QUAL (le nom du référentiel à changer selon les blocs)
3. file\_suivi = FILE\_SUIVI\_QUAL (le nom du fichier de suivi du référentiel à changer selon les blocs)
4. liste\_test = LST\_TEST\_QUAL (à changer pour chaque bloc : nom d’une variable pour test)

L’ensemble des valeurs données en paramètre sont définis dans des objets dans le script de paramétrage.

### 2.3.3- Comptage des balises

La fonction comptage\_balise() (Prod/Fonctions/balises/comptage\_balise.R) effectue le même traitement que sur la partie pubmed :

1. Comptage à partir des fichiers .RDS des référentiels (Prod/Output/)
2. Création des référentiels de comptage en sortie (Prod/Output/ref\_comptage/)

### 2.3.4- Chargement postgre

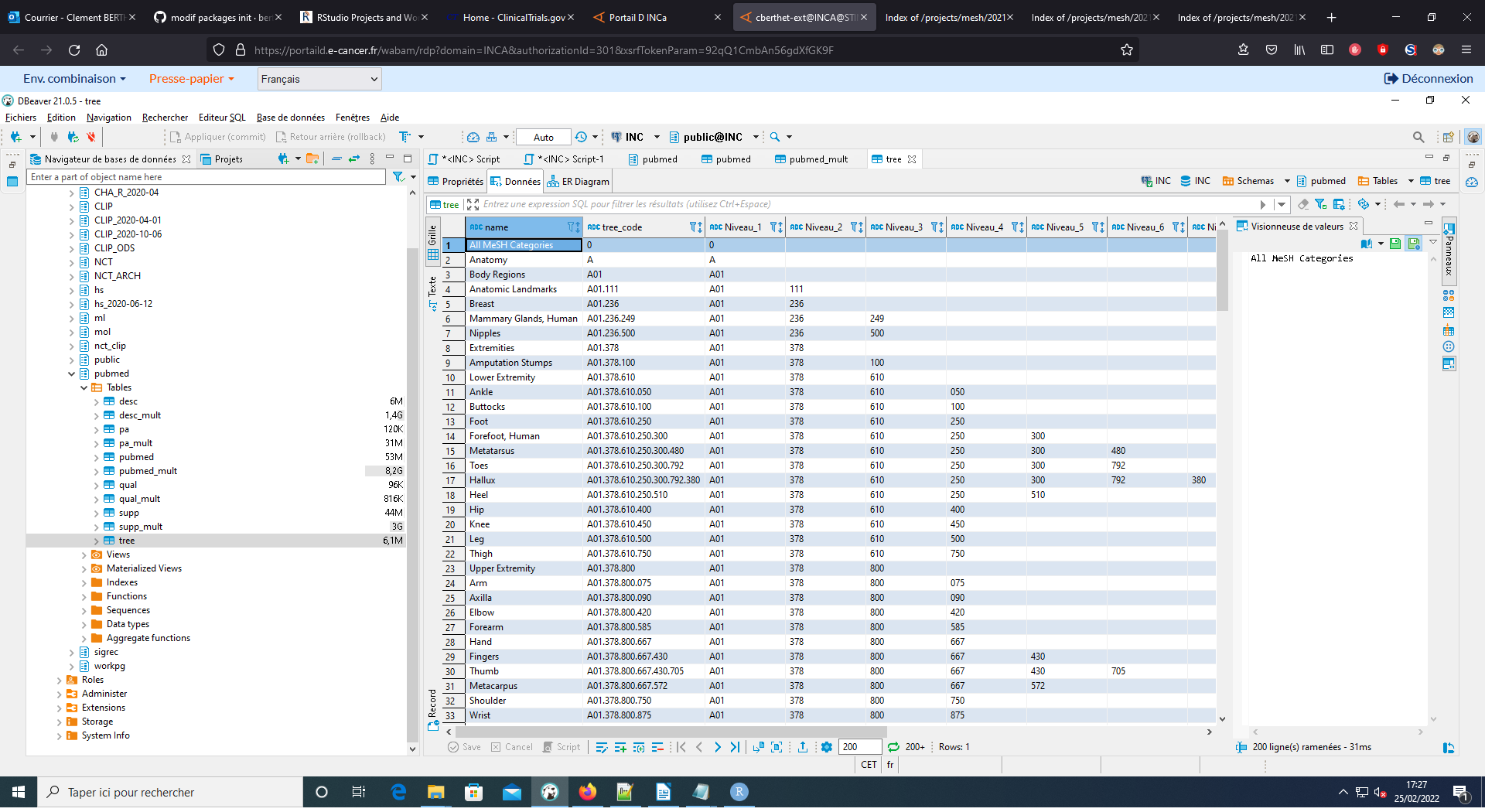
La fonction balisage\_RDS() (Prod/Fonctions/balises/balisage\_RDS.R) effectue le même traitement que sur la partie pubmed :

1. Création du schéma si non existant
2. Création des tables vides simples/multiples à partir du référentiel de comptage + typage des colonnes
3. Création des clés primaires/index
4. Chargement des tables simples/multiples dans l’environnement postgre INCA

## 2.4- Bloc tree codes

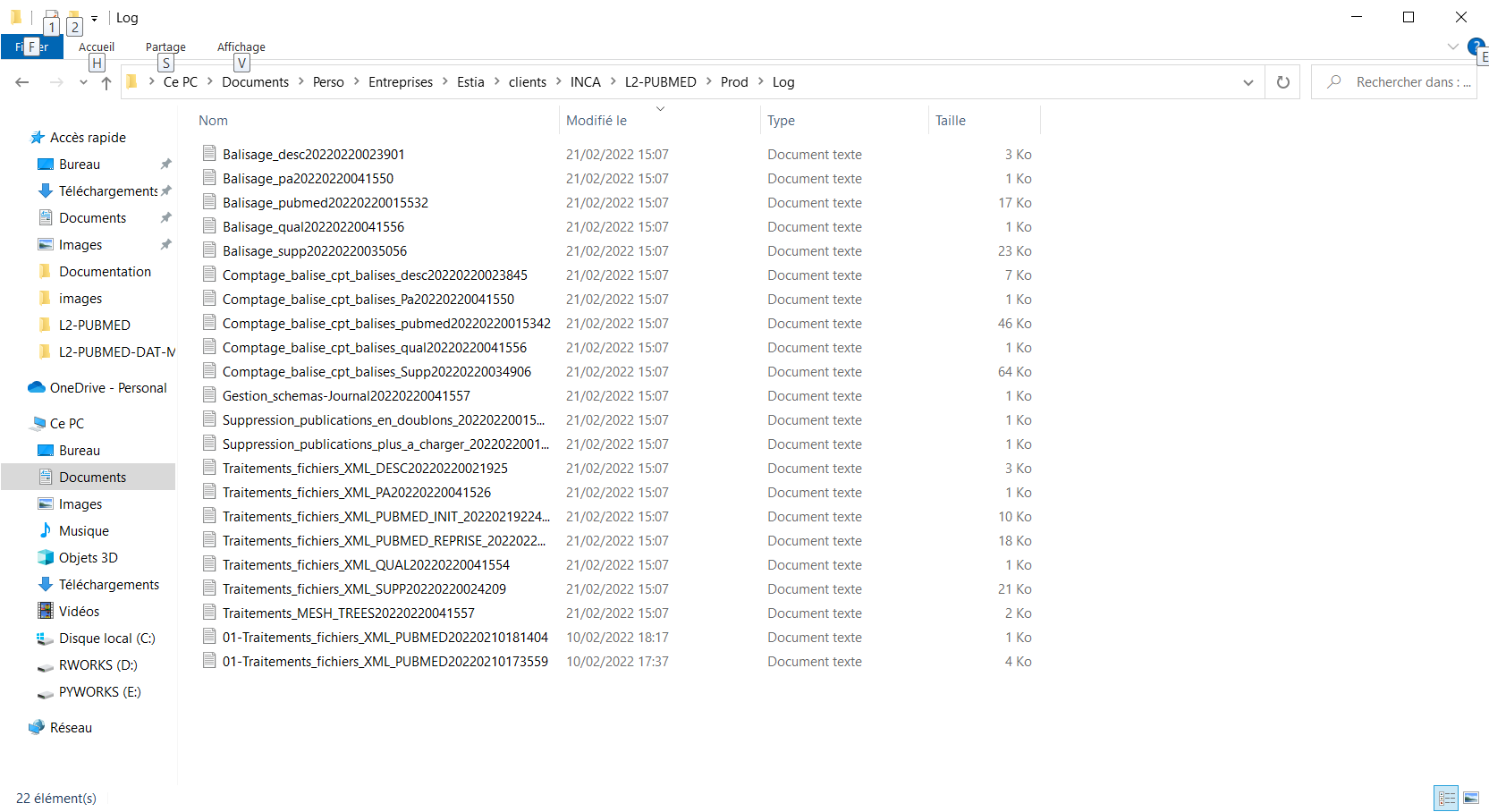
Chaque code Mesh correspond à un code et ces codes forment une arborescence. Le référentiel est stocké en INPUT (Prod/Input/ mtrees2022.bin) et il est disponible via le lien [meshtrees](https://nlmpubs.nlm.nih.gov/projects/mesh/2021/meshtrees/).

La fonction Lancement\_tree() (Prod/Fonction/tree/Lancement\_tree.R) traite le référentiel pour créer un data frame puis lance le chargement de la table en base postgre.

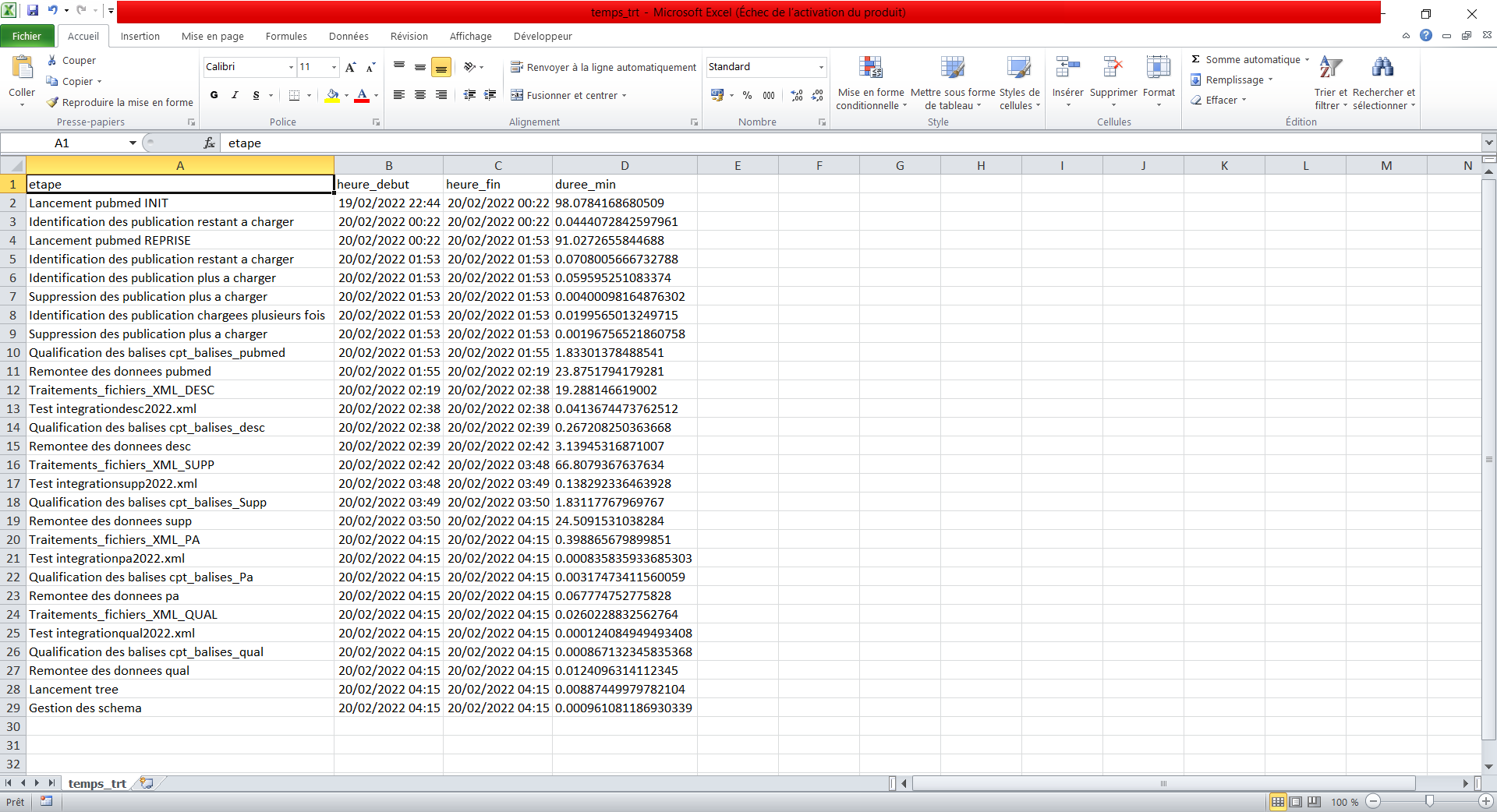


## 2.5- Log et temps de traitement

Les fonctions appelées dans le programme principal sont tracées et suivies à l’aide de fichiers de **LOG** (Prod/Log/).



Les temps de traitement sont également tracés dans un fichier *csv* (Prod/Output/temps\_traitement/**temps\_trt.csv**).



Tous ces fichiers sont créés au moment de l’exécution des fonctions dans le programme principal.

# 3- Gestion des schémas

La fonction gestion\_schema() (Prod/Fonctions/post\_gres/gestion\_schema.R) est à exécuter une fois que le chargement des bases pubmed et des référentiels est validé.

Cette fonction va permettre la gestion des schémas et opérer une bascule entre les différents niveaux.

**Nous avons trois niveaux de schéma** :

1. **Le schéma temporaire**
2. **Le schéma de prod**
3. **Le schéma archivé**

La fonction de gestion des schémas opère une bascule du niveau du schéma vers un autre.

Temporaire

Archivé

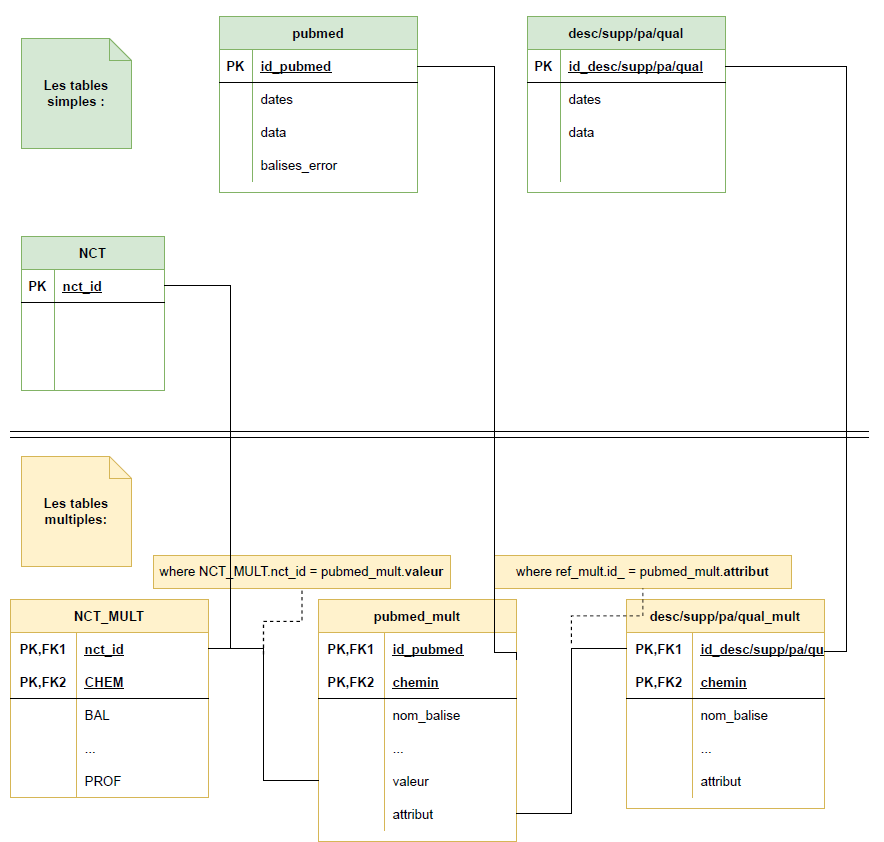
Prod

**Les étapes** :

1. Test de l'existence du schéma temporaire (*pubmed\_tmp*)
2. Test de l'existence du schéma archivé (*pubmed\_arch*)
3. Test de l'existence du schéma de prod (*pubmed*)
4. Gestion de la bascule des schémas de *pubmed\_tmp* vers *pubmed* et de *pubmed* vers *pubmed\_arch*

# 4- MCD

Ce schéma peut-être challengé et montre les jointures/liaisons entre les tables simples et multiples, NCT et pubmed, pubmed et les tables référentiels Mesh.



*Les cardinalités ne sont pas précisées dans ce schéma, seules les jointures sont illustrées*

1. Voir le paramètre LISTE\_BALISE\_ERRONNEES dans le script de paramétrage [↑](#footnote-ref-1)