Importation de la base de données de l’inventaire forestier du USDA Forest Service

Pour exécuter le script et faire l’importation, lire jusqu’à « Interprétation du log » et consulter la section « Problèmes actuels »

Pour plus de détails sur la stratégie et le rôle de chaque fichier, consulter le reste du document

Table des matières

[Introduction et prérequis 3](#_Toc522085593)

[Contexte : 3](#_Toc522085594)

[Objectifs : 3](#_Toc522085595)

[Prérequis : 3](#_Toc522085596)

[Guide d’utilisation 4](#_Toc522085597)

[Marche à suivre (étape par étape) : 4](#_Toc522085598)

[Après l’importation : 4](#_Toc522085599)

[Interprétation du log 6](#_Toc522085600)

[Vérification de la base de données : 6](#_Toc522085601)

[Stderr des scripts d’importation : 6](#_Toc522085602)

[Avertissements concernant les fichiers : 6](#_Toc522085603)

[Validation des données dans les fichiers : 6](#_Toc522085604)

[Exemples de résultats souhaitables : 6](#_Toc522085605)

[Interprétation des autres résultats : 6](#_Toc522085606)

[Exceptions : 7](#_Toc522085607)

[Problèmes actuels 7](#_Toc522085608)

[Détail de la stratégie 8](#_Toc522085609)

[Rôle des fichiers : 8](#_Toc522085610)

[Rôle des répertoires : 9](#_Toc522085611)

[ExportTables : 9](#_Toc522085612)

[FIADB\_PG\_X\_X\_X\_XX (FIADB\_PG\_1\_7\_2\_00 actuellement) : 9](#_Toc522085613)

[RObjects : 10](#_Toc522085614)

[Détail et fonctionnement des sections du script principal: 10](#_Toc522085615)

[Problèmes et solutions 12](#_Toc522085616)

[Fichiers batch et ligne de commande : 12](#_Toc522085617)

[Formatage de date américain 12](#_Toc522085618)

[Sortie d’erreur standard (stderr output) non-redirigé 12](#_Toc522085619)

[Pas d’attente avant la fin de l’exécution pour les scripts batch 12](#_Toc522085620)

[Encodage non permissif pour les caractères latin 12](#_Toc522085621)

[Permissions et accès utilisateur 12](#_Toc522085622)

[Modification de la variable « PATH » de manière permanente impossible 12](#_Toc522085623)

[Modification des clés de registre impossible 12](#_Toc522085624)

[Accès aux dossiers refusé lors de l’opération COPY pour copier vers ou à partir des tables 13](#_Toc522085625)

[Exécution de scripts en Windows PowerShell refusée 13](#_Toc522085626)

[R, PostgreSQL et validation : 13](#_Toc522085627)

[Nombre de connections limitées avec le package RPosgreSQL 13](#_Toc522085628)

[Types de données Postgres non reconnus par R 13](#_Toc522085629)

[La lecture des fichiers de données dans R prend trop de mémoire vive 13](#_Toc522085630)

[Les fonctions « compare » et « identical » retournent des valeurs FALSE lorsque les formats sont différents 14](#_Toc522085631)

[Messages d’erreur 15](#_Toc522085632)

[Création des tables : 15](#_Toc522085633)

[Chargement des données de référence : 15](#_Toc522085634)

[Chargement des données : 15](#_Toc522085635)

# Introduction et prérequis

## Contexte :

Le service des forets du Département d’Agriculture des États-Unis (*United States Department of Agriculture Forest Service*) met à disposition en source libre les données de tout l’inventaire forestier aux États-Unis sous forme de fichiers de données CSV en fournissant également les scripts pour réaliser l’importation des données dans une base de données locale. Pour faire l’importation des données un utilisateur doit récupérer les données manuellement et suivre les instructions fournies par le USDA pour générer la base de données en une série d’étapes. Le projet suivant propose une alternative automatisée plus simple d’utilisation qui inclut également une validation de l’importation des données.

## Objectifs :

Avec l’exécution d’un script unique en R:

* Connexion au serveur PostgreSQL
* Création de la base de données et sélection de l’emplacement physique des données
* Téléchargement des scripts d’importation et des fichiers de données
* Création de l’arborescence du projet
* Importation des données
* Validation de l’importation
* Production d’un fichier « Log » qui montre le résultat de l’importation

## Prérequis :

Pour procéder à l’importation l’utilisateur doit :

* Posséder au moins 140 Go d’espace disque sur son ordinateur (82 Go pour le projet incluant validation et 58 Go pour la base de données PostgreSQL)
* Posséder plus de 100 Go de mémoire vive (RAM) pour la portion validation (l’importation sans validation peut être effectuée avec 8 Go sans problème)
* Le logiciel PostgreSQL version 9.1 et plus (version 10 recommandée)
* Être connecté à internet pour la phase d’importation (téléchargement des données)
* Exécuter le script sur une plateforme Windows (Seul système d’exploitation supporté)

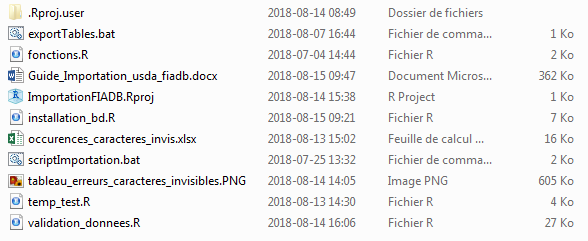


Figure 1: Contenu du projet avant exécution

# Guide d’utilisation

## Marche à suivre (étape par étape) :

1. Récupérer le projet R intitulé « ImportationFIADB » disponible dans le répertoire suivant : P:\F1272\ProjetCC\USDA-FIA
2. Copier de dossier dans un répertoire local de votre choix qui possède au moins 82 Go d’espace disque disponible (prévoir plus d’espace serait préférable à octroyer la quantité minimale requise pour minimiser le risque d’erreurs causées par manque d’espace). Le contenu du dossier avant l’importation devrait ressembler à la figure 1.

\*\*\*Il est essentiel de choisir un répertoire dans lequel l’utilisateur possède les droits de modifier les préférences des dossiers sinon cela posera un problème lors de l’importation des données dans la base de données.

1. Créer un dossier qui contiendra les données physiques de la base de données (Tablespace) (au moins 60 Go d’espace disponible à cet endroit).
2. Ouvrir le projet R (fichier « ImportationFIADB.Rproj »).
3. Dans R ouvrir le script « validation\_donnees.R » (c’est le script principal qui appellera les autres scripts au besoin.
4. Modifier la valeur des variables suivantes de la section « Variables utilisateur » :

* pw : mot de passe pour se connecter au serveur PostgreSQL
* emplacementTablespace : chaine de caractères de l’emplacement du dossier créé au point 3 (incluant le nom du dossier).
* repFIADB : Nom du répertoire de scripts du USDA (dépends du numéro de la version en ligne lors de l’importation)
* nouvNomBD : Nom que l’utilisateur souhaite donner à la base de données après sa validation pour la conserver
* nouvNomSchema : Nom que l’utilisateur souhaite donner au schéma pour le conserver après validation de l’importation
* repPG : Chemin d’accès absolu vers le répertoire « bin » de Postgres

\*\*\* Très important de spécifier le chemin d’accès avec les double barres obliques inversées « \\ » (ex : C:\\Program Files\\PostgreSQL\\9.6\\bin)

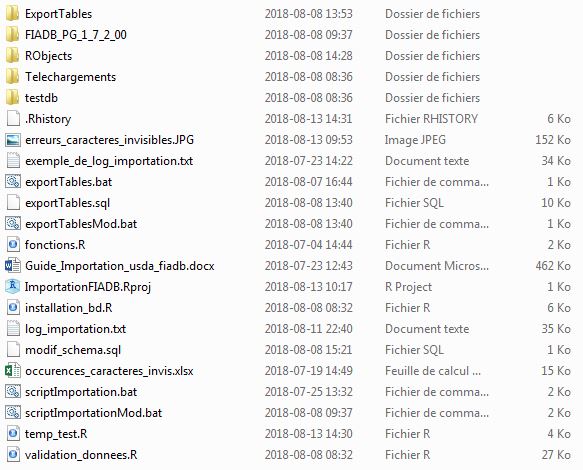
1. Sauvegarder le script (CTRL + S).
2. Exécuter le script « validation\_donnees.R » en entier (mettre le curseur à la fin du script et exécuter avec CTRL + ALT + B).

## Après l’importation :

* Consulter le log d’importation pour valider le succès ou détecter les erreurs potentielles.
* Exécuter le script « modif\_schema.sql» lorsque la base de données est importée sans erreurs.

Le contenu du projet après exécution devrait ressembler à la figure 2.

Figure 2: Contenu du projet après exécution



# Interprétation du log

Le fichier texte « log\_importation.txt » est créé pour permettre à l’utilisateur de vérifier si les différentes étapes de l’importation ont été réalisées sans erreurs et permet de valider l’intégrité des données dans les tables après importation.

## Vérification de la base de données :

Message confirmant si la base de données existe ou n’existe pas après la phase de création

## Stderr des scripts d’importation :

Sorties d’erreur standard (stderr) de l’exécution des trois scripts d’importation du USDA : «createTables.bat », « loadeReferenceTables.bat » et « loadDataTablesEntire.bat »

## Avertissements concernant les fichiers :

Section présentant le nombre de scripts SQL de création des tables, le nombre de tables dans le schéma et le nombre de fichiers CSV de données. Basé sur ces données, présente le nombre de scripts pour lesquels aucune table n’a été créé, le nombre de tables sans fichier CSV et le nombre de tables et fichiers CSV vides.

## Validation des données dans les fichiers :

Cette portion du log retourne le nom du fichier, la valeur du « compare » et la valeur du « identical » pour chaque fichier de données. Lorsque compare retourne FALSE, la liste des champs est imprimée avec la valeur de comparaison de chaque champ.

### Exemples de résultats souhaitables :

* Identical = TRUE et Compare = TRUE : Résultat optimal

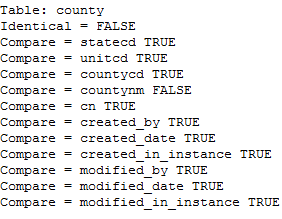


* Identical = FALSE et Compare = TRUE : Résultat acceptable car la fonction identical est très sévère et peux accrocher sur des détails très pointus qui n’auront pas d’impact sur les valeurs dans la base de données. La valeur de la fonction compare valide l’équivalence entre CSV et base de données.



### Interprétation des autres résultats :

Lorsqu’un champ retourne une valeur FALSE dans la comparaison, la sortie dans le log ressemblera un peu à ceci :



Avant de déclarer qu’il y a eu une erreur dans l’importation et que les valeurs des tables sont fausses, il est essentiel de vérifier si le champ concerné ne fait pas partie de l’une des exceptions connues.

### Exceptions :

* Tables avec le mot « evalidator » dans le nom : ces tables ne sont pas importantes pour la validation des données et peuvent être ignorées
* Le champ contient une ou plusieurs entrées avec des chaines de caractères invisibles qui faussent la comparaison mais ne causent pas d’erreurs à l’importation (les chaines de caractères sont effacées à l’importation) : consulter l’image « tableau\_erreurs\_caracteres\_invisibles.PNG » pour la liste complète.

# Situation actuelle

Au moment de la création de la stratégie d’importation il reste certains problèmes avec la donnée du USDA qui ne sont pas résolus.

Ces problèmes sont contournés ou temporairement réglés en attendant une résolution plus permanente une fois qu’ils seront signalés au USDA.

* Format de date non standard dans les fichiers « DATAMART\_TABLES.CSV » et « EVALIDATOR\_POP\_EST\_1\_6\_1.csv »
  + Actuellement, une version des fichiers dans laquelle les dates sont standardisées est copiée pour remplacer ceux qui présentent des erreurs.
  + Ces fichiers doivent se trouver à l’extérieur du répertoire source et le segment de code qui en fait les copies devra être retiré après la correction (installation\_bd.R, section 5)
* Présence de caractères invisibles dans plusieurs fichiers CSV (la liste des fichiers concernés et les détails peuvent être consultés dans l’image « tableau\_erreurs\_caracteres\_invisibles.PNG » (le fichier « occurrences\_caracteres\_invisibles.xlsx » contient le tableau modifiable)
  + Ces erreurs sont ignorées pour l’instant car elles ne semblent pas avoir d’impact sur la donnée dans la base de données. Elles seront signalées à l’USDA pour permettre une validation plus propre et contribuer à la qualité de la source de données.
* Présence d’un « NA » dans le fichier « COND.csv » au lieu d’une chaine de caractère vide. La table correspondante contient la chaine de caractère « NA » au lieu d’une case vide qui est le résultat attendu.
* La connexion au serveur du USDA à partir duquel les données sont téléchargées est instable et parfois inaccessible.
  + Le code a été modifié de sorte à faire du « error handling » dans les sections de téléchargement et réessayer jusqu’à 5 fois le téléchargement si la connexion est impossible
  + Une recommandation sera faite au USDA de fournir un MD5 pour valider l’intégrité de la donnée téléchargée à partir du serveur.Détail de la stratégie

## Rôle des fichiers :

|  |  |  |
| --- | --- | --- |
| **Nom du fichier** | **Rôle principal** | **Détail des tâches accomplies** |
| **exportTables.bat** | Fichier batch utilisé pour exporter le contenu des tables en CSV | Ajout du programme psql.exe dans le chemin d’accès Windows |
| Connection avec la base de données |
| Exécution du script SQL « exportTables.sql » pour exporter les tables en CSV |
| **fonctions.R** | Script R contenant les fonctions utiles pour le script principal | fonction getSQL : formate une requête sql pour éliminer les caractères d’arrêt et la rendre lisible dans R |
| fonction killDbConnections : termine les connections avec la base de données |
| **ImportationFIADB.proj** | Fichier de projet contenant les préférences de projet | Établir les préférences (ex : code de page) |
| **installation\_bd.R** | Script R appelé par le script principal lors de l’étape d’installation des fichiers nécessaires à l’importation des données | Création des répertoires de téléchargement des données |
| Téléchargement des scripts de l’USDA et des fichiers de données |
| Décompression des archives zip vers les emplacements prévus dans l’arborescence |
| Remplacement de la chaine de caractères associée à la variable « DATETIME » dans les batch files par une version adaptée au format « yyyy-mm-dd » |
| Exécution du fichier batch « scriptImportation.bat » |
| Remplace à nouveau la chaine de caractère dans les fichiers batch pour retirer les modifications apportées |
| **scriptImportation.bat** | Fichier batch exécuté pour lancer à tour de rôle les scripts d’importations du USDA | Change temporairement le répertoire de travail (requis par les scripts du USDA) |
| Accorde les permissions à « Tout le monde » pour chaque dossier ou des fichiers de données seront copiés |
| Apelle en ordre les trois scripts batch du USDA : « createTables.bat », « loadReferenceTables.bat » et « loadDataTablesEntire.bat » |
| Écrit un court message lorsque l’importation est terminée (permet d’assurer que la suite du script R ne sera pas exécutée avant la fin du chargement des données dans les tables) |
| **validation\_donnees.R** | Script général responsable de coordonner l’exécution des autres scripts et d’effectuer la manipulation et validation des données. | Le détail du script se trouve dans la section « Détail et fonctionnement du script principal » |

## Rôle des répertoires :

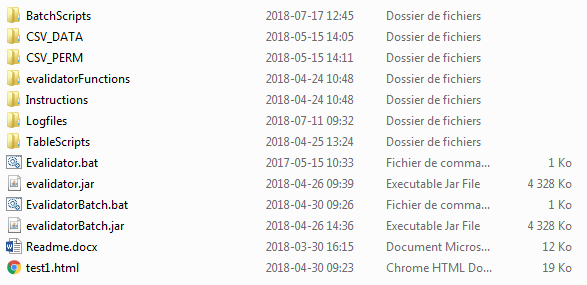
### ExportTables :

Répertoire créé pour contenir les fichiers CSV des données qui proviennent de la base de données après importation.

### FIADB\_PG\_X\_X\_X\_XX (FIADB\_PG\_1\_7\_2\_00 actuellement) :

Dossier contenant les scripts d’importation et les dossiers dans lesquels les fichiers de données sont décompressés avant de procéder à l’importation. Ce dossier est modifié à chaque mise à jour majeure de la base de données mais sa structure reste similaire[[1]](#footnote-1) (voir figure 3).

Figure 3: Contenu du dossier d'importation du FIA fourni par le USDA



Dossiers importants :

|  |  |
| --- | --- |
| **BatchScripts** | Répertoire qui contient les scripts batch d’importation « createTables.bat », « loadReferenceTables.bat » et « loadDataTablesEntire » en plus du script batch pour importer un seul état « loadDataTablesSingleState.bat » |
| **CSV\_DATA** | Répertoire (initialement vide) dans lequel seront extrait les fichiers de données CSV du FIA |
| **CSV\_PERM** | Répertoire du FIA contenant les fichiers CSV de type « DATAMART » et « EVALIDATOR », la validation en fait usage en récupérant les fichiers CSV pour en comparer le contenu avec celui des tables mais sans plus. L’outil de validation « Evalidator » n’est pas utilisé dans la validation. |
| **Instructions** | Dossier contenant des instructions d’utilisation regroupées en quatre sous-dossiers : « EVALIDator\_Desktop&Batch », « EVALIDator\_WebApp&API », « FIADB\_ExampleScripts » et « Postgres ». Le dossier « Postgres » est le plus important puisqu’il contient les instructions d’installation de Postgres et d’importation de la base de donnée sur lesquelles cette stratégie est basée. |
| **Logfiles** | Dossier où sont redirigés les sorties standard de scripts d’importation du FIADB (cela ne comprends pas les sorties d’erreur. |
| **ScriptOutputs** | Dossier créé durant la stratégie d’importation pour recevoir les sorties standard des scripts d’importation ainsi que les sorties d’erreur standard |
| **TableScripts** | Dossier contenant les scripts SQL utilisés par le USDA pour créer les tables dans la base de données |

### RObjects :

Dossier créé durant la validation pour recevoir les fichiers CSV originaux et fichiers CSV provenant de la base de données lorsqu’ils sont sauvegardés en objets R (fichiers.RData) dans le but d’être comparés entre eux.

## Détail et fonctionnement des sections du script principal:

Le script principal est divisé en 16 sections, voici le détail des tâches accomplies dans chacune des sections :

1. Initialisation:
   * Nettoyage l'environnement de travail en supprimant les variables
   * Appel de la « Garbage Collection »
2. Connection aux fichiers:
   * Exécution du script R « fonctions.R » contenant les fonctions utiles pour qu'elles soient accessibles durant le script
3. Chargement et vérification des packages:
   * Vérification de la présence des packages nécessaires
   * Installation des packages absent
   * Appel de la méthode « require » sur chaque package pour les charger
4. Variables utilisateur :

Modification des variables qui varient d’un utilisateur à l’autre et sont nécessaires pour l’installation :

* + pw : mot de passe de l’utilisateur « Postgres » (administrateur) nécessaire pour se connecter à la base de données
  + emplacementTablespace : chemin complet vers le dossier dans lequel sera créé un « Tablespace » contenant les données physiques de la base de données
  + repFIADB : Nom du dossier du FIA contenant les scripts d’importation, le nom exact de ce répertoire est appelé à changer avec les mises à jour de la base de données même si sa position relative dans l’arborescence demeure la même
  + nouvNomBD : Chaine de caractères qui servira à renommer la base de données conformément aux besoins de l’utilisateur une fois l’importation terminée
  + nouvNomSchema : Chaine de caractères qui servira à renommer le schéma une fois l’importation terminée
  + repPG : Chemin d’accès du répertoire « bin » contenant l’exécutable psql dont les commandes seront utilisées pour copier le contenu des fichiers de données dans les tables. La version de Postgres utilisée par l’utilisateur sera extraite à partir de cette variable et stocké dans la variable « versionPG »

1. Création du fichier log :
   * Création du fichier texte qui va servir de log
   * Récupération du nom d’utilisateur
   * Écriture de l'entête du log d'importation qui sera complété au cours du script
2. Connection avec la base de données PostgreSQL :
   * Connection à la base de données « postgres »
   * Création de la base de donnée « testdb »
   * Connection avec la base de données « testdb »
   * Création des extensions
   * Validation de l’existence de la base de données
3. Importation des données :
   * Exécution du script R « installation\_bd.R » qui fait :
     1. Téléchargement des données et des scripts du FIADB
     2. Création de l'arborescence des dossiers
     3. Modification des scripts d’importation
     4. Importation des données dans les tables
4. Transfert des sorties de script vers le log :
   * Redirection du contenu du Standard error (stderr) output issu de l’importation vers le log d’importation
5. Validation du nombre de tables crées :
   * Requête pour obtenir la liste des tables dans le schéma avec RPosgreSQL
   * Liste des scripts SQL de création de tables dans les dossiers
   * Comparaison entre les fichiers SQL de création des tables et les tables réellement créées
6. Liste des fichiers de données CSV :
   * Liste des fichiers de données dans les répertoires et comparaison avec les tables de la base de données
   * Retrait des tables qui n’ont pas de fichiers de données associés
7. Retrait des tables et fichiers CSV vides :
   * Lecture du nombre de lignes des fichiers CSV et des tables
   * Retrait des fichiers et tables qui n’ont aucune donnée
8. Envoi du warning vers le log :
   * Écrit le nombre de scripts SQL, le nombre de tables créées et le nombre de fichiers de données CSV
   * Indique les scripts qui n’ont pas créé de tables
   * Indique les tables qui n’ont pas de fichiers de données associés
   * Indique les tables et fichiers CSV qui n’ont aucun enregistrement
9. Requête noms de colonnes et types de colonnes tables
   * Récupère la liste des noms et des types des champs pour chaque table et converti les types de données PostgreSQL en types reconnu par R (ex: smallint ---> int)
10. Importation des dataframes CSV
    * Importe (dans une boucle) les csv en dataframes avec le package « data.frame » et les sauvegarde en Robjects
11. Exportation des tables en CSV
    * Exécute un script batch qui exporte le contenu des tables en fichiers CSV dans le but de les lire pour comparer les valeurs avec celles des CSV originaux
12. Importation des dataframes PostgreSQL à partir des CSV
    * Importe (dans une boucle) les csv créés à la section 13 en dataframes avec le package « data.frame » et les sauvegarde en Robjects
13. Comparaison par valeur et identitaire des dataframes
    * Charge le dataframe (Robject) des données originales
    * Charge le dataframe des données provenant de l'importation
    * Parcourt des conditions pour manipuler les formats de données (dates)
    * Compare les dataframes avec compare (package compare)
    * Teste l'identité avec « identical » (package base)
    * Écrit le résultat des tests dans le log d'importation
14. Création du script sql pour renommer le schéma et la BD
    * Créé « modif\_schema.sql » qui :
      1. Renomme la base de données
      2. Renomme le schéma
15. Déconnection
    * Terminaison de la connexion au serveur PostgreSQL

# Problèmes et solutions

Cette section liste des problèmes rencontrés lors de l’élaboration et l’exécution de la stratégie d’importation et de validation ainsi que les solutions employées pour ceux-ci.

## Fichiers batch et ligne de commande :

### Formatage de date américain

Problème : Les scripts d’importation de la FIADB produisent un affichage de la date et heure du lancement du script qui est envoyé au log automatiquement, mais puisqu’ils utilisent un formatage de la date différent du nôtre, la date ne s’affiche pas correctement.

Solution : Ajout d’une section dans le code R où les lignes concernées des scripts sont modifiées pour retourner la date correctement dans un format « yyyy-MM-dd ».

### Sortie d’erreur standard (stderr output) non-redirigé

Problème : Les scripts d’importation de la FIADB produisent un log automatiquement, mais il ne redirige pas les messages d’erreur vers des fichiers afin de les sauvegarder.

Solution : Dans R, création du répertoire « Script Outputs » pour recevoir les logs d’importation et dans le fichier batch d’importation, redirection du stderr vers des fichiers texte.

### Pas d’attente avant la fin de l’exécution pour les scripts batch

Problème : Lors de l’exécution du dernier script batch d’importation de la FIADB (loadDataTablesEntire.bat), lui-même exécuté par un fichier batch (scriptImportation.bat), puisque la simple exécution du script batch marque la fin du fichier, R passe immédiatement à l’étape suivante alors que celle-ci requiert la fin de l’importation.

Solution : Option « Wait = True » requise lors de l’exécution d’un script avec la fonction « shell » dans R et ajout d’une commande à la fin du fichier batch comme mesure supplémentaire pour s’assurer que R ne poursuive que si le script a terminé son exécution.

### Encodage non permissif pour les caractères latin

Problème : Lors de la création des fichiers texte de sorties de script ainsi que du fichier log de l’importation, les caractères latin ne s’affichent pas correctement.

Solution : Pour les scripts, le code de page est changé de 850 à 1252 pour autoriser les caractères latins et pour l’écriture du log avec R directement l’encodage est changé à « UTF-8 » dans la section 5 du fichier « validation\_donnees.R »

## Permissions et accès utilisateur

### Modification de la variable « PATH » de manière permanente impossible

Problème : La connexion à la base de données avec fichier batch se fait grâce aux commandes de psql.exe qui ne sont pas reconnues comme commandes système par défaut et doivent être ajoutées dans la variable PATH ce qui ne peux être fait que temporairement.

Solution : Le script « scriptImportation.bat » comprends une ligne de code qui fait l’ajour de l’exécutable au PATH avant l’importation.

### Modification des clés de registre impossible

Problème : L’utilisateur qui n’est pas administrateur ne peut pas modifier les clés de registre comme par exemple la valeur de « sShortDate », ce qui pose problème pour adapter l’environnement à celui des États-Unis (pour lequel l’importation de base est adaptée)

Solution : Les variables tel « DATETIME » sont adaptées à l’environnement actuel (voir solution pour « Formatage de date américain »

### Accès aux dossiers refusé lors de l’opération COPY pour copier vers ou à partir des tables

Problème : Lors de la copie de données des fichiers vers les tables ou des données des tables dans des fichiers par les scripts batch, l’accès aux répertoires est refusé.

Solution : Dans le script batch qui fait les copies, l’accès aux répertoires est donné à l’utilisateur « Tout le monde ». (Ce problème est connu du USDA et la solution est donnée dans leur guide d’importation.

### Exécution de scripts en Windows PowerShell refusée

Problème : La politique d’exécution par défaut du PowerShell est fixée à « Restricted » ce qui veut dire qu’il n’est pas possible d’exécuter des scripts de commandes PowerShell.

Solution : Les opérations sur le système doivent être écrites en Windows cmd par l’entremise de scripts batch.

Remarque : L’utilisateur qui souhaite reprendre et améliorer la stratégie et qui possède les droits d’administrateurs est encouragé à faire usage du PowerShell puisque c’est un outil puissant qui offre plus de possibilités que la ligne de commande cmd.

## R, PostgreSQL et validation :

### Nombre de connections limitées avec le package RPosgreSQL

Problème : Même si cela est très improbable lors de l’importation puisqu’en théorie le script principal en R ne devrait être exécuté qu’une seule fois, il est possible qu’un utilisateur qui souhaite tester le code pour l’améliorer atteigne le nombre maximal de connections avec la base de données soit 16.

Solution : Exécuter la fonction « killDbConnections » dans R qui terminera toutes les connexions établies avec la base de données. (Voir la documentation de la fonction dans le fichier « fonctions.R »)

### Types de données Postgres non reconnus par R

Problème : Lors de la requête dans la base de données pour obtenir les types de chaque champ dans le but de forcer l’importation des données dans R avec ces mêmes types, on obtient les types de données Postgres qui ne sont pas compatibles avec R.

Solution : Faire la substitution des types dans la liste pour leur équivalent dans R (ex : « smallint » devient « int », « character varying » devient « character »)

### La lecture des fichiers de données dans R prend trop de mémoire vive

Problème : Dans R pour manipuler les dataframes il faut les charger en mémoire vive ce qui occupe rapidement beaucoup d’espace vu la taille des jeux de données à analyser.

Solution : Plusieurs moyens ont été employés pour réduire la consommation de mémoire vive ou permettre au programme d’exécuter. Le problème existe toujours car certaines solutions ne font que contourner le problème et non le régler, ainsi la mémoire vive représente une des principales limitations de l’utilisateur.

1. Utilisation du package « Data.Table » (similaire à bigdata et ff) qui charge les données plus efficacement en conservant une partie de la donnée sur le disque dur.
2. Séparation du plus gros jeu de données en deux morceaux avec la fonction « split »
3. Appel de la « Garbage Collection » entre chaque chargement de dataframe pour libérer la mémoire vive
4. Exécuter le code (section de validation) sur un ordinateur ayant une grande quantité de mémoire vive

La stratégie a été développée dans le but d’être exécutée sur un ordinateur possédant 128 Go de mémoire vive. Si un utilisateur futur souhaite rendre le code exécutable sur un ordinateur aux capacités plus restreintes, la première étape serait de répéter la solution 2 sur tous les jeux de données problématiques en scindant en plusieurs morceaux.

### Les fonctions « compare » et « identical » retournent des valeurs FALSE lorsque les formats sont différents

Problème : Surtout observé pour les dates, certaines données peuvent être les mêmes (valeur) mais passent d’un format à un autre lors de l’importation vers la base de données. Ces données retournent un résultat FAUX à la comparaison.

Solution : Il suffit de faire une conversion de type (un cast) des données provenant des deux sources vers un même type pour que la comparaison indique s’il y a une différence dans les valeurs.

# Messages d’erreur

## Création des tables :

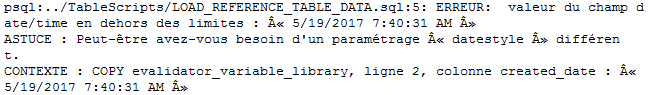


Explication : Le script « createTables.bat » appelle une série de scripts SQL (du répertoire TablesScripts), mais l’un des scripts appelés n’existe pas.

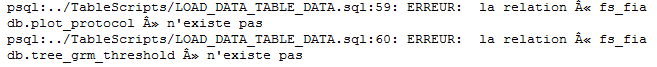
## Chargement des données de référence :



Explication : Cette erreur survient après l’exécution du script « loadReferenceTables.bat ». Une des lignes devait commenter le code avec « REM --», cependant il manque un espace entre le « M » et le tiret, ainsi le commentaire s’affiche comme du code.



Explication : Lors de l’exécution du script « loadReferenceData.bat » dans l’instruction copy de la table « evalidator\_variable\_library » le script ne peut pas copier certaines dates. Ces dates sont dans un format différent et doivent être standardisées pour que la donnée soit importée correctement.



Explication : Cette erreur est connue du USDA, une de leurs fonctions Evalidator ne se crée pas correctement dû à certains caractères spéciaux. Puisque nous n’utilisons pas Evalidator cela a peu d’impact sur la stratégie mais il suffit de copier manuellement le contenu du fichier « congcdlabel.sql » and le « Query tool » de pgAdmin pour corriger le problème.

## Chargement des données :



Explication : Lors de l’exécution du script « loadDataTablesEntire.bat », le script SQL appelé exécute une opération pour copier le contenu des fichiers « plot\_protocol » et « tree\_grm\_threshold » dans les tables du même nom, cependant les tables et les fichiers n’existent pas. Les scripts SQL pour créer les tables existent mais ne sont jamais appelés et de toute façon il n’y a pas de données associées à ces tables.

1. Le dossier « ScriptsOutputs » est créé lors de la validation et ne fait pas parti du contenu original du dossier. [↑](#footnote-ref-1)