UNIVERSITE DE LA REUNION

UFR SCIENCES ET TECHNOLOGIES

RAPPORT DE STAGE DE MASTER M2 INFORMATIQUE

Centre de Coopération Internationale en Recherche Agronomique pour le Dévelopement(CIRAD)

Thème :
Intégration d'outils de visualisation des données issues d'agroécosystème à La Réunion

---

Auteur :
DAGE Aimée Blanchette
n° étudiant : 39008020

Encadrant :
AUZOUX Sandrine
Enseignant Référent :
COURDIER Remy

Période de stage : Du 23 Janvier au 21 juillet 2023

Je dédie ce rapport à ma fille Aina Tsako DIANZINGA.

Remerciements

Je profite de cette occasion pour exprimer ma gratitude à toutes les personnes qui m'ont soutenu et guidé tout au long de mon stage. Leur contribution et leur soutien ont été inestimables dans le succès de mon expérience professionnelle.

Je souhaite remercier chaleureusement Mme Sandrine AUZOUX Informaticienne et Responsable du stage pour m'avoir offert l'opportunité d'effectuer mon stage au sein du CIRAd. Sa confiance en moi et son encadrement tout au long du stage m'ont permis d'acquérir de précieuses compétences professionnelles.

J'adresse également mes remerciements à l'ensemble de l'équipe de AIDA (département dans lequel j'ai travaillé) pour leur accueil chaleureux, Particulierement à Mathias CHRISTINA(agro-modélisateurs) pour sa disponibilité à répondre aux questions et à faire des tests sur mes scripts et Monsieur Jean Christophe SOULIER Informaticien pour ses idées dans l'organisation de mon projet. Leurs conseils, leurs commentaires constructifs et leur collaboration ont grandement enrichi mon expérience.

Je tiens à exprimer ma reconnaissance envers mes collègues de travail spécialement à Fabre Ferber Frédérick, doctorant, qui de par son exprérience dans le domaine informatique m'a guidé dans mes recherches et Léa CHEVALIER pour la disponibilité à répondre aux questions et l'ambiance de travail conviviale . Ils ont rendu mon stage d'autant plus agréable.

Je souhaite exprimer ma gratitude envers ma famille du Cameroun pour l'accompagnement et les conseils ainsi que Madame et Monsieur WAMBO mes parents de la Réunion pour leur amour et disponibilité depuis mon arrivée sur l'île.

Je tiens à remercier Niry DIANZINGA pour tes encouragements ton soutien malgré toutes les difficultés.

Enfin j'exprime ma gratitude à mes amis Zavier pour ta diponibilité et ton aide malgré la distance et soeurs mervedi et danielle et marie-claude qui gardé ma fille pour que je puisse avancer dans le stage et la rédaction.

Je tiens à remercier tous ceux que j'ai oublié et qui ont contribué de près ou de loin à la réussite de mon stage. Leur soutien, leurs conseils et leur bienveillance ont été des facteurs déterminants dans mon développement professionnel. Je suis reconnaissante pour cette expérience enrichissante et les souvenirs précieux que j'emporte avec moi.

Résumé

Conçu pour faciliter l'acquisition, l'analyse, l'exploitation et le partage des données issues de l'agro-écologie, le système d'information Aegis développé par le Cirad présente des limites dans l'exploitation de ces données. Dans le but de valider le diplôme de Master 2 Informatique, une mission m'a été assignée dans le cadre de mon stage réalisé au sein de l'unité de recherche AIDA au Cirad (Agroécologie et Intensification Durable des cultures Annuelles), sur le site de La Bretagne à Sainte-Clotilde. Cette mission consiste à créer un outil d'aide ou d'appui à Aegis à travers des scripts R. L'intérêt de ces scripts axés sur les données est de permettre aux chercheurs agronomes leur importation, leur sauvegarde et leur extraction de la base de données en fonction des essais, de deux manières possibles : soit par le package R, soit par l'interface Rshiny. Cette dernière offre également une visualisation des données manipulées

Mots-clés

: CIRAD, AEGIS, PostgreSQL, RStudio, Packages, RShiny, Visualisation, système d'information

Abstract

Designed to facilitate the acquisition, analysis, use and sharing of agro-ecological data, the Aegis information system developed by CIRAD has its limitations. In order to validate my Master 2 diploma in Informatics, I was assigned a project as part of my internship with the AIDA research unit at CIRAD (Agroécologie et Intensification Durable des cultures Annuelles), on the La Bretagne site in Sainte-Clotilde. The aim of these scripts is to enable agricultural researchers to import and save their data, as well as to extract data from the database according to the trials, in two possible ways : either via the R package, or via the Rshiny interface. The latter also offers a visualization of the data manipulated.

[3]

Keywords

:CIRAD, AEGIS, PostgreSQL, RStudio, Packages, RShiny, Visualisation, système d'information.

Table des matières

Table des figures

Liste des tableaux

1 INTRODUCTION

1.1 contexte général du projet

AEGIS (Agro-Ecological Global Information System) est un système d'information qui permet d'acquérir, de gérer, d'exploiter et de partager des données provenant d'études en agroécologie. L'agroecologie, cherche à favoriser la biodiversité, à maintenir la fertilité des sols et à promouvoir des pratiques agricoles respectueuses de l'environnement. C'est dans cette quête que les chercheurs agronomes vont réaliser des essais sur différentes parcelles et générer un ensemble de données hétérogènes. AEGIS est capable d'utiliser ces données et de produire des données homogènes à différentes échelles d'observations en répondant à des standards de qualité et, de répondre aux attentes des acteurs de la recherche, de l'enseignement et des filières agricoles, par la mise à disposition d'outils génériques d'analyse statistique et d'outils de visualisation.

Des travaux ont déjà été effectués par des étudiants et des ingénieurs en informatique sur la visualisation et la cartographie des données sur AEGIS avec l'outil D3.js. Pour faciliter l'exploitation des données par les agronomes dans AEGIS, il a été décidé d'opter pour la création d'un Package R et de développer une interface Rshiny.

Ce stage a pour objectif de **créer un ensemble de traitements statistiques permettant une exploration et une visualisation en première approche des données contenues dans AEGIS**. Une étude des besoins a été réalisée en interaction avec un panel de chercheurs agronomes du CIRAD[1], dont les données des projets sont déjà dans AEGIS, afin d'établir une liste de traitements statistiques permettant de satisfaire la plupart des besoins exprimés.

1.2 Problématique

Ma première activité a été d'échanger avec différents chercheurs (agronomes, malherbologues, biogéochimistes, agropédologues) pour les questionner sur leurs différents essais, connaître quels étaient leurs réels besoins en termes de gestion et de valorisation de leurs données expérimentales. Il en est ressorti que les chercheurs ont du mal à enregistrer les données, fichier par fichier, dans la base de données de la plateforme AEGIS. Cela prend beaucoup de temps et nécessite un effort conséquent pour nettoyer et harmoniser les données. Au bout du compte, ils préfèrent garder les données collectées sur leur ordinateur et ne les partagent pas. Pour pallier à cette problématique, nous allons tenter de simplifier au maximum la procédure d'importation des données dans AEGIS, c'est-à-dire automatiser la structuration des jeux de données au format AEGIS et la normalisation des variables à partir du dictionnaire d'AEGIS. Comme les chercheurs sont très habiles avec la manipulation de R, nous avons choisi de créer des scripts R qu'ils pourraient réutiliser. En deuxième approche, les chercheurs souhaitent pouvoir réaliser des visualisations et des analyses plus ou moins complexes des données contenues dans AEGIS. Ma deuxième activité concernera la création de visualisations analytiques des données. Ainsi, nous avons structuré notre travail en 2 grandes parties :

— Collecte et traitement des données

— Création scripts R pour Visualisation et sauvegarde des données

2 ENVIRENNEMENT DE TRAVAIL : AEGIS

Le système d'information Aegis est constitué de deux parties principales : Une application WEB et Un Serveur de base de données PostgreSQL.

2.1 Une Application WEB

L'application web Aegis est développée avec CodeIgniter[2], un framework basé sur le modèle MVC (Modèle-Vue-Contrôleur)1 qui permet de créer des applications web puissantes et évolutives. Cette structure facilite la maintenance, la réutilisation et la cohérence du code.

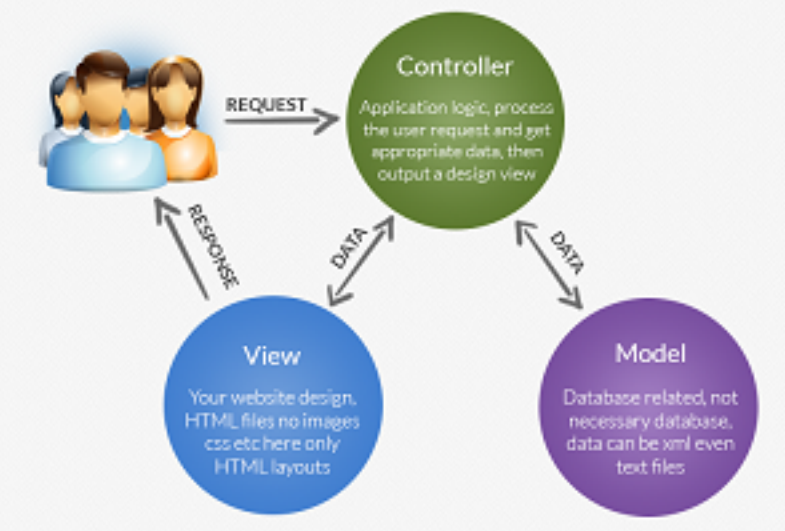

Figure 1: Modèle MVC

2.2 Un Serveur de base de données

Le serveur de base de données est utilisé pour sauvegarder et gérer les données générées par Aegis. CodeIgniter offre une couche d'abstraction de base de données qui facilite l'interaction avec différents types de bases de données telles que MySQL, PostgreSQL, Oracle, etc. Il permet d'effectuer des opérations de création, de lecture, de mise à jour et de suppression (CRUD) de manière sécurisée et efficace. Nous avons choisi PostgreSQL [3] car il s'agit d'un système de gestion de base de données relationnelle (SGBDR) open source et robuste. Il est réputé pour sa fiabilité, ses performances élevées, sa conformité aux normes SQL et son extensibilité.

Du fait de ces caratéritiques, le schéma de la base de données PostgreSQL utilisé dans Aegis permet aux utilisateurs d'accéder à la base de données, d'effectuer des requêtes 2 complexes et de manipuler les données de manière puissante sans interférence les uns avec les autres. Ce schéma relationnel est utilisé pour stocker les différentes expérimentations agronomiques. Son organisation est présentée dans l'annexe 17.

2.3 Description du modèle de données

Le modèle de données AEGIS est divisé en 7 groupes de données (voir Annexe 18) :

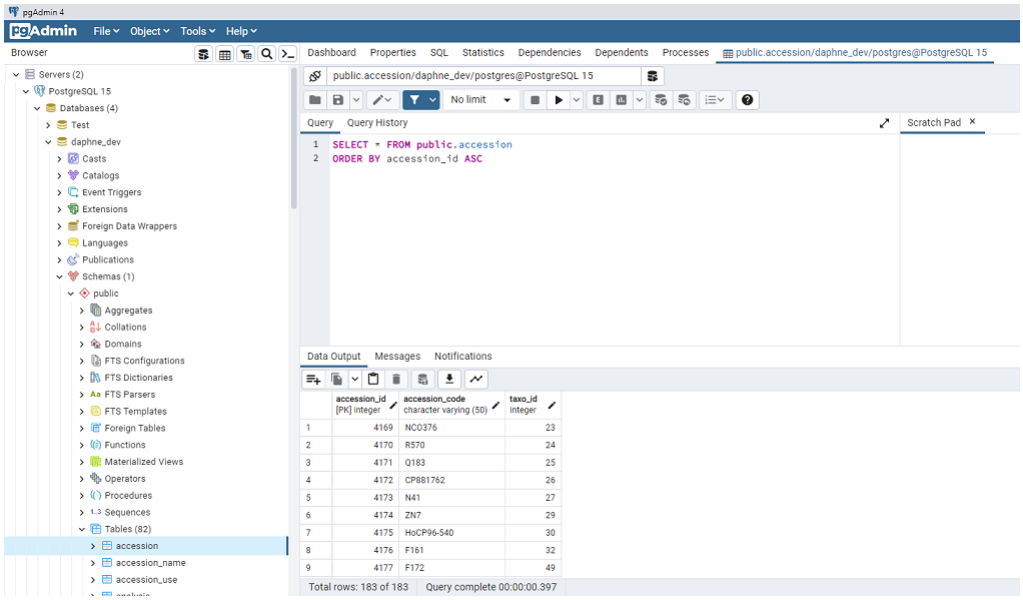

Figure 2: Interface Postgres

1. Le groupe de conception expérimentale stocke les données de la conception statistique de l'expérience.
2. Le matériel végétal et entomologique est défini dans le groupe taxonomie.
3. Les conditions environnementales de l'expérience concernent les informations relatives à l'eau, au climat et au sol.
4. Le groupe des pratiques culturales se réfère à toutes les pratiques culturales possibles, telles que le choix des variétés, la préparation du sol, l'irrigation, le déchiquetage, la récolte, la gestion des ravageurs, des maladies et des mauvaises herbes.
5. Le groupe des observations intègre toutes les informations relatives aux mesures effectuées en environnement contrôlé ou sur le terrain à différentes échelles temporelles et spatiales.
6. Les données relatives aux échantillons analysés en laboratoire sont stockées dans le groupe de suivi des échantillons.
7. Le dernier groupe contient les métadonnées relatives à l'ontologie et au dictionnaire des variables.

Notre travail ne s'est étalé que sur quatre groupes : les conditions environnementales, les observations, la conception exprérimentale, la taxionomie. C'est sur la base de ces données que nous avons eu à effectuer nos missions.

3 MISSIONS EFFECTUEES

Dans le but de simplifier le travail des chercheurs qui collectent des données sur le terrain, nous avons prévu de créer un ensemble de scripts R, ainsi qu'une interface de visualisation conviviale appelée Rshiny pour visualiser les données contenues dans les tables de la base de données. Notre travail principal était concentré sur la gestion des données du système Aegis, en incluant l'importation, le traitement, la sauvegarde et l'exportation des données.

Avant de commencer à développer l'outil de traitement des données, une première étape de mon travail consistait à collecter toutes les données des différents projets déposés par Sandrine Auzoux (Ingénieur en charge de la base de données Aegis) dans le référentiel centralisé de gestion, de stockage et de partage de données appelé Dataverse [4]. Cela correspondait à un total de 8 projets menés par le Cirad à La Réunion , répartis en 28 dossiers. Comme les chercheurs ne connaissaient pas le dictionnaire de données d'AEGIS et les dossiers contenaient un grand volume de données hétérogènes, nous avons dû :

1. Faire une analyse minutieuse des champs de chaque table, pour mieux comprendre les données.De cette analyse il ressort que :
   (a) Certaines tables contenaient des données dupliquées
   (b) Pour un même champs on peut avoir des données de différents types. Par exemple : int et char
   (c) Pour les champs de type date, le format (jj/mm/AAAA) ne correspondait pas à celui de la base de données(AAAA-mm-jj)
   (d) Certaines données sont enregistrées sous forme de colonnes au lieu des lignes dans les fichiers excels
2. Transformer manuellement les données (labellisation et structuration) afin qu'elles correspondent au mieux à l'arborescence de notre base des données ;
3. Développer des scripts R pour l'import et réaliser la sauvegarde vers la base de données ;
4. Créer un package R à partir des scripts transformés en fonctions

3.1 Création d'un package R

3.1.1 Définition

Rstudio[5] : RStudio est un environnement de développement intégré (IDE) spécialement conçu pour le langage de programmation R.
R[6] : est un langage de programmation open source utilisé pour organiser un ensemble de données, faire des tests statistiques et faire des representations sous forme de graphe.R est hautement extensible grace à des packages et est puissant grace à sa capacité à créer des packages.

Un package R : est un logiciel qui regroupe des fonctions, des données et de documentation , permettant d'étendre les fonctionnalités de base du langage R. Les packages R sont généralements

utilisés pour organiser et distribuer du code réutilisable.
La création d'un package R, est bénéfique tant pour une utilisation personnelle que pour une caummunauté plus large. Un package R est composé des éléments suivant :

- Fonctions : qui implémentent des algorithmes, des analyses statistiques, des visualisations
- Données : Utilisées dans le package pour illustrer les fonctionnalités ou permettre aux utilisateurs d'expérimentrer et comprendre les fonctions du package avec des exemples concrets.
- Documentation : Permets de décrire les fonctions disponibles, leurs paramètres et leurs exmples.La documentation peut être fournie sous forme de fichier PDF ou html.
- Dépendances : Les packages peuvent dépendre d'autres package pour fonctionner correctement. Ces dépendances sont généralement spécifiées dans les métadonnées du package, afin que les utilisateurs puissent facilement les installer.

3.1.2 Création d'un projet

Il existe des packages facilitant la conception d'autres packages. Avant la création d'un nouveau projet, assurons nous d'avoir installé les packages tels que :

- `devtools` offre des fonction avec plusieurs fonctionnalités. On a entre autre des fonctions qui :facilitent la création d'un nouveau projet de package(devtools : :create()), pertmet de faire des tests unitaires sur les fonctions du package afin afin de décéler les erreurs les warnings , les notes et de les corriger avec la fonction (devtools : :check()),facilite l'installation du package avec la fonction (devtools : :install).
- `usethis` : étant les fonctionnalité de devtools en fournissant les fonctions supplementaires pour faciliter le développement de packages.
- `roxygen2` : qui permet de générer automatiquement la documentation des fonctions encourage les bonne pratiques et facilite la comprehension du package par les utilisateurs.

Leur installation est simple. Une fois Rstudio ouvert, il suffit de saisir :
install.packages(c`("devtools", "usethis", "roxygen2"))` dans la console de R et valider.
Pour créer notre projet, nous allons sur `File->New Project -> New Directory`. (voir figure3)

Ensuite, nous avons choisi pour notre projet **R Package using Devtools** (voir figure 4)qui comparé à d'autres approches à l'instard Rcpp , Rcpp Armadillo ou Rcpp Eigen (qui font intervenir d'autres langanges de programmations comme le C++), Devtools ne fait intervenir que du langage R.

Une Boite de dialogue s'ouvre, nous invite à renseigner le nom de notre package et de choisir un emplacement pour sauvegarder notre projet(voir figure 5). Nous cliquons enfin sur create pour create project et RStudio s'ouvre sur un nouveau projet. Une fois le projet ouvert, nous avons une arborescence de fichiers et dossiers (voir figure 6). Nous n'allons nous interesser qu'au fichier Description et au Dossier R. Le .gitignore permet de mentionner les fichiers que l'on ne souhaite pas voir sur le git tandis que le .Rbuidignore mentionne les fichiers dont le package ne prendra pas en compte.

Tout est en place mais, avant de sdémarrer il nous a fallu régler quelques paramètres. Dans le

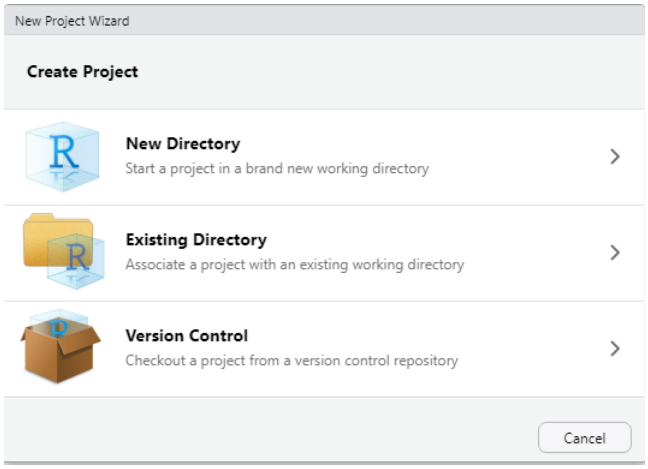

Figure 3: Création d'un nouveau projet

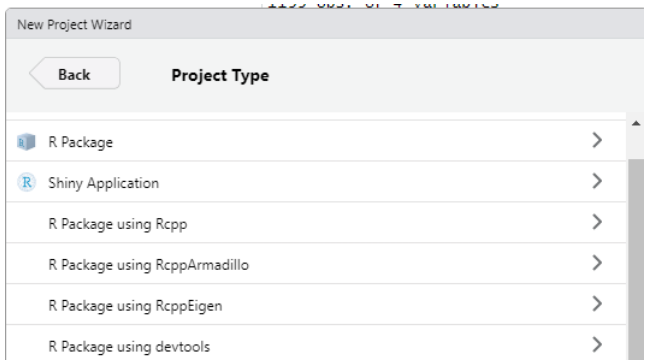

Figure 4: Choix du type de projet

menu BUILD-> cliquez sur configure build tools->cochez la case install and restart.(voir figure 7).

Il est à noter que toutes modifications fichiers nécessitent un check à la fin. Ceci nous permettra de corriger directement toutes les erreurs , les alertes et notes potentielles et d'en avoir zéro avant de continuer. Pour le faire, il faut aller dans l'onglet Build-> check.

3.1.3 Fichier Description

Ce fichier (voir figure8) est comme la pièce d'identité de notre package. Il renseigne sur

- le Titre du package
- la version qui change en fonction des améliorations sur les scripts
- le ou les Auteurs et leurs roles
- la Description qui nous donne un aperçu de ce que fait le package
- La Licence qui est GPL-3 pour notre cas
- l'encodage utilisé
- Les Imports qui prennent en charge toutes les versions à venir des packages à partir de celles utilisées.

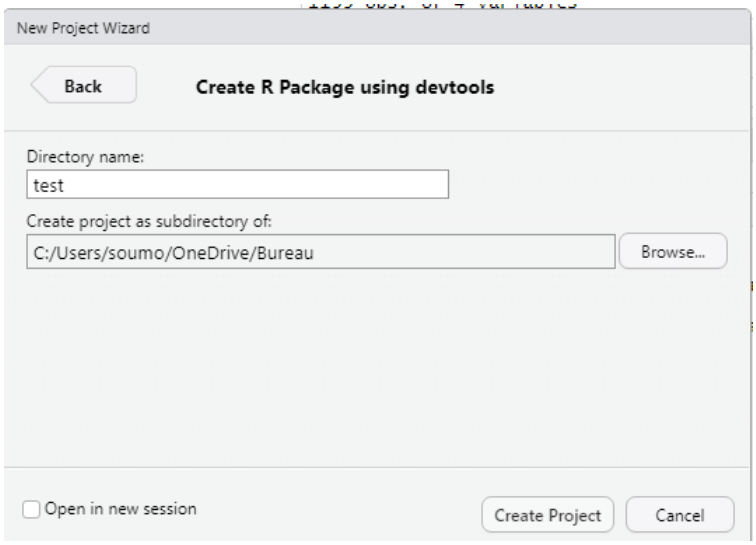

Figure 5: Attribution du nom au projet

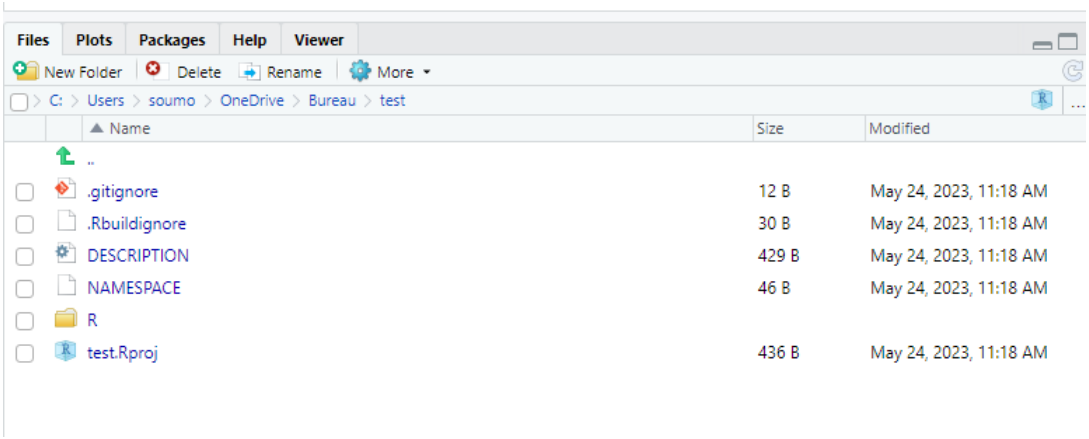

Figure 6: Arborescence du projet

3.1.4 Les Fonctions

Une fonction est un ensemble d'instructions qui permet de réaliser une tâche spécifique. Un package est utile seulement si chacune de ses fonctions est bien définie. Il est donc préférable, notamment lorsque plusieurs auteurs sont impliqués, de créer chaque fonction dans un fichier distinct. La structure d'un fichier de fonction est établie à l'aide de Roxygen. Chacun de ces fichiers est donc placé dans le dossier R de l'arborescence.

Afin de faciliter l'utilisation d'une fonction, il est nécessaire de respecter une certaine structure, qui est générée par Roxygen. Pour ce faire, il faut se positionner au début de la fonction à l'aide du curseur, puis cliquer sur le menu **" Code > Insérer le squelette Roxygen"**. Juste au-dessus de la fonction, la structure suivante s'affichera : nom, description, paramètres de la fonction avec

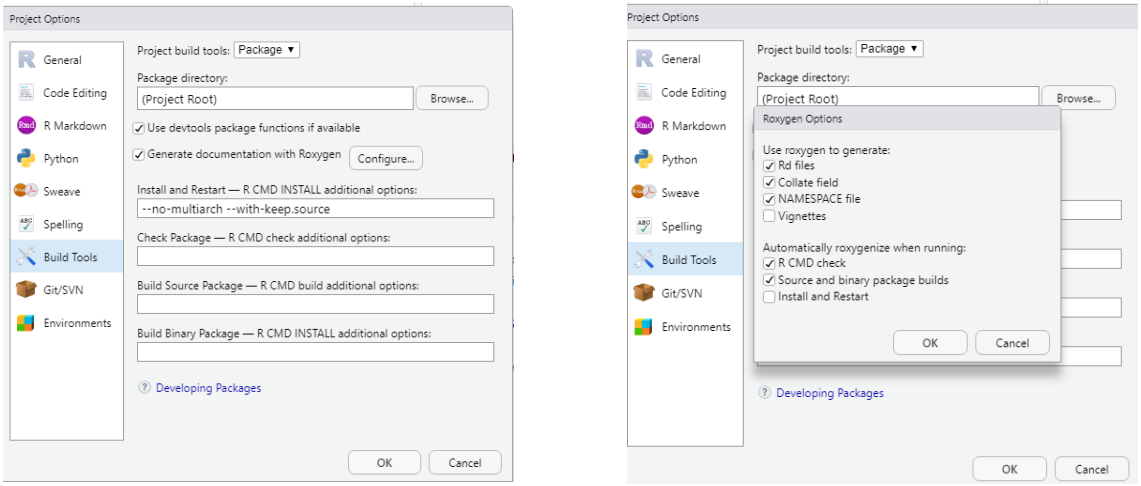

Figure 7: Parametrage

```
Package: save
Title: Import_Export
Version: 0.0.0.9000
Authors@R:
    person("DAGE", "AIMEE", , "ab.dage@yahoo.com", role = c("aut", "cre"),
           comment = c(ORCID = "YOUR-ORCID-ID"))
Description: Test pour importation automatique et exportation des données.
License: GPL-3
Encoding: UTF-8
Roxygen: list(markdown = TRUE)
RoxygenNote: 7.2.3
Suggests:
    knitr,
    rmarkdown
VignetteBuilder: knitr
Imports:
    DBI(>= 1.1.3),
    readxl(>= 1.4.1)
```

Figure 8: Fichier Description

@param, résultat de la fonction avec @return, **@export** (que nous n'utiliserons pas, mais qui permet de déclarer quelles fonctions doivent être rendues publiques et disponibles pour une utilisation externe) et enfin **@example**, qui permet de fournir un exemple d'application de la fonction. Ces élémentsC.1.1 sur la création d'une fonction, doivent être remplacés par les informations propres aux différentes fonctions de notre package.

Pour notre projet, nous avons répartis nos fonctions en quatre.

— La fonction **import_Aegis**C.1.2 : qui permet d'importer les feuilles des fichiers excels depuis un emplacement de notre choix sur la machine. Cette fonction utilise principalement les fonctions **read_excel** (pour lire le fichier excel) et **excel_sheet** (pour lire une feuille du fichier excel) du package **readxl**.La fonction prend en paramètre le chemin vers lequel se

trouve notre fichier excel.

— La fonction **import_database**C.1.3 : qui permet à partir de la fonction **dbGetQuery** du package **DBI** et de la requête formulée, d'envoyer la requête, récupérer le résultat et afficher sur R pour une utilisation ultérieure. Cette fonction prend en paramètre la connexion à la base de données, le nom de la base de données et les champs pour lesquels nous voulons des informations.

— La fonction **trait_sauve** qui permet de faire la sauvegarde des données importées vers la base de données en utilisant la fonction **dbWriteTable** et les arguments **Append = TRUE**(qui permettent d'ajouter des lignes dans une table existante) et **Overwrite=FLASE**(pour éviter que la table soit écrasée pour etre remplacée pas celle qu'un veut enregistrer) du package **DBI**.Cette fonction prend en paramètre la connexion à la base de données, le nom de la table dans la base de données et le nom du fichier à sauvegarder. Il est à noter que la fonction dbGetQuery est aussi utile si vous souhaitez créer et charger une table en même temps au cas ou elle n'existe pas ou d'écraser la table existante.Dans ces cas de ficgures, pas la peine de mettre les arguments.

```
    trait_sauve <- function(con, table_db, fichier){
  DBI::dbWriteTable( con, table_db, fichier, append = TRUE, overwrite = FALSE )
}
```

L'exemple de cette fonction est délicat car chaque fichier a nécessité des traitements particuliers avant la sauvegarde. Ces traitements sétalent sur plusieurs ordres.

— Sauvegarde directe : Pour ces tables, il n'y a aucun traitement à faire. Il faut juste utiliser la fonction pour sauvegarder.

```
#' #Scale
#' trait_sauve( con, "scale",  Scale )
```

— Vérification et Sauvegarde : Certaines tables n'ont pas besoin de données dupliquées. Etant donné que la clé primaire est auto incrément et que les numéros sont uniques, il est impératif de filtrer l'entrée du deuxième champs afin de garantir l'unicité des données avant de sauvegarder.

```
#' #----------------------Factor ------------------------###
#' for(i in seq_len(nrow(Factor))){
#'
#'   #vérifier si la valeur numero i du champ factor existe déja dans la bd
#'   existe<-dbGetQuery(con, paste0("SELECT * FROM factor WHERE
#'    factor='", Factor$factor[i], "'"))
```

```
#'
#'   if((existe)>0{
#'     message("la valeur",Factor$factor[i],"Existe déjà dans la table")
#'   }
#'
#' # si cette valeur là n'existe pas,l'ajouter à la suite
#'   if (nrow(existe)==0){
#'     trait_sauve(con, "factor", Factor[i,])
#'
#'   }
#' }
```

— La jointure , suppression des colonnes et sauvegarde Pour d'autres tables, il faut utiliser les données exportées avec la fonction **import_database**, faire la jointure avec les tables importées, supprimer les colonnes inutiles, faire un filtrage avant de sauvegarder

```
#'  factor_level0<- merge(factor_database, Factor_Level, by="factor")
#' factor_database
#' # supprimer la colonne factor
#' Factor_Level<- select(factor_level0, -factor)
#'
#' for(i in seq_len(nrow(Factor_Level))){
#'
#'   #vérifier si la valeur numero i du champ factor existe déja dans la bd
#'   existe<-dbGetQuery(con, paste0("SELECT * FROM factor_level
#'                       WHERE factor_level='", Factor_Level$factor_level[i], "'"))
#'
#'   if(nrow(existe)>0){
#'     message("la valeur",Factor_Level$factor_level[i],"Existe déjà dans la table")
#'   }
#'
#'   # si cette valeur là n'existe pas,l'ajouter à la suite
#'   if (nrow(existe)==0){
#'     trait_sauve(con, "factor_level", Factor_Level[i,])
#'   }
#' }
```

— La fonction **export_database**C.1.4 :qui permettra à partir de la fonction **dbGetQuery** du package **DBI**, de la connexion à la base de données et de la requête formulée, d'extraire les données depuis la base de données pour un code précis.Par la suite la fonction

write.csv2(fonction de base de R), qui prend en paramètre le résultat de la requête, l'emplacement de sauvegarde avec le nom du fichier y compris, permet d'exporter les données sous forme de fichier csv à l'emplacement qu'on lui précisera. Notre fonction d'exportation prend en paramètre le code pour lequel nous voulons des informations et l'emplacement.

Le package offre un espace appelé **vignette** pour rédiger mode d'emploi du package. Etant donné que notre package est disponible sur Aegis, nous avons rendu un guide d'utilisation accessible directement sur le site.
Pour l'installer il faut saisir :

```
usethis::use\_vignette("comment-utiliser-mon-package")
```

il va installer un dossier contenant un fichier **.Rmd** à remplir par le mode d'emploi du package. Une Fois le package terminé et pour faciliter le partage, nous l'avons transformé en **tar.gz** via

```
menu Build > Build Source Package
```

. Il a généré un fichier compressé que nous avons déposé sur Aegis afin de le rendre accéssible à tous les chercheurs qui travaillent sur le projet Aegis.Le package installé ressemble à cette (figure 9)

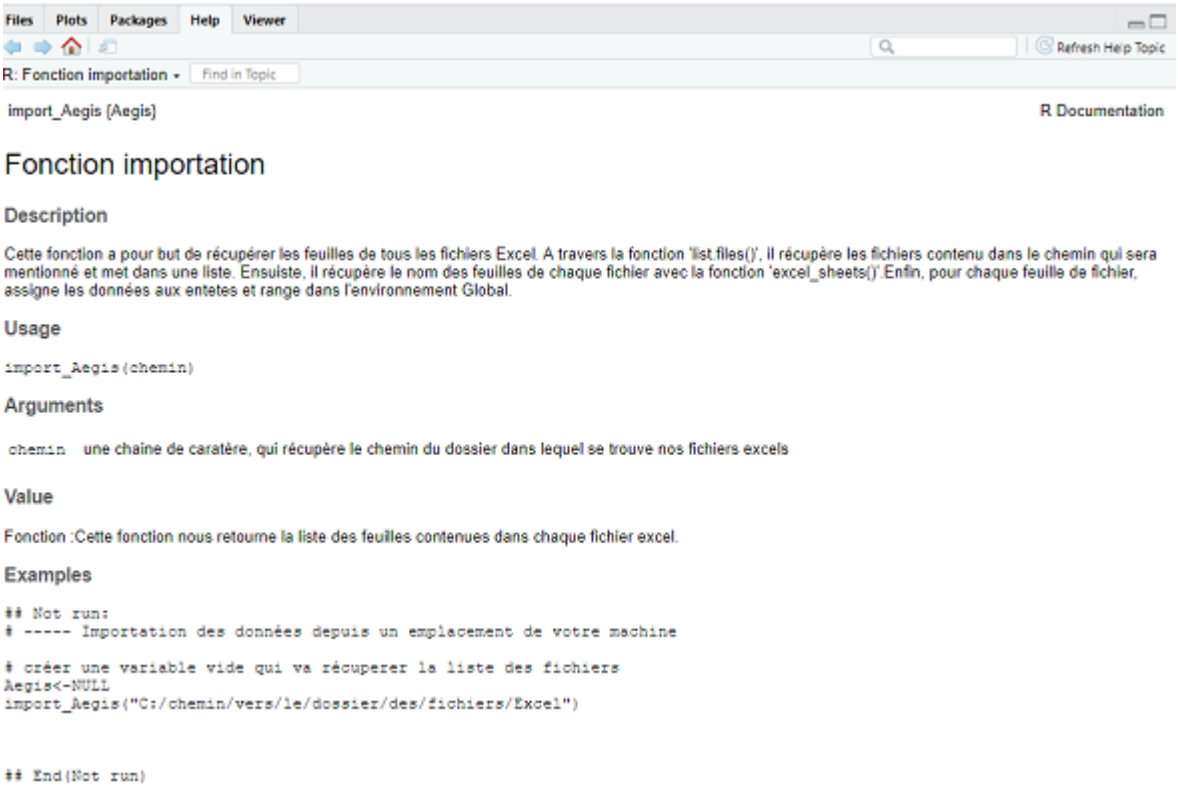

Figure 9: Package installé

3.2 RShiny

Dans la continuité du processus de visualisation des données et pour des personnes moins habiles sur le code R, notre soucis était de proposer en plus du package, une interface polyvalente pour l'importation, la visualisation et la sauvegarde des données. Dans nos recherches, nous avons constaté que R propose des outils de développements pour interface graphique : tcltk et RShiny. Du fait de ses avantages comparé à tcltk (voir table 1), j'ai adopté RShiny. Shiny[7] est un package R open source qui fournit un cadre Web élégant et puissant pour la création d'applications Web à l'aide de R. Shiny vous aide à transformer vos analyses en applications Web interactives sans nécessiter de connaissances en HTML, CSS ou JavaScript. Pour développer une interface R Shiny, vous avons d'abord installé le package Shiny avec **install.packages("shiny")** dans votre environnement R.

3.2.1 Structure d'une application shiny

Une application Shiny a deux composants principaux à définir : Linterface Utilisateur et le serveur.

1. **l'interface utilisateur (UI)** : contrôle la mise en page et l'apparence de l'application. L'interface utilisateur se scinde aussi en deux : une partie pour les entrées(input) et une partie MainPanel pour choisir la façon dont les données seront affichées (output).
   Pour notre interface, nous avons défini des widgets d'entrées tels que .
   — **fileinput** pour récupérer les fichiers depuis un emplacement de notre machine ;
   — **selectinput** pour récupérer le nom des fichiers pour mettre sous forme de liste déroulante et un autre pour le choix de nos tracés ;
   — **actionButton** qui nous allons utiliser pour sauvegarder les données dans la base de données ;
   — **helpText** pour laisser une note ou une information pour les utilisateurs ;
   — **dowloadButton** qui permet de télécharger les fichiers à remplir.
   La partie **mainPanel** : panneau principal se compose de trois onglets : un pour les données brutes, un autre pour les données traitées, le dernier pour les tracés. Ces onglets permettront de visualiser les données en fonction du type de sortie choisi. Le code ci-dessous nous présente deux fonctions de sortie(Output) : La fonction DTOuput() qui nous permettra de visualiser les données sous forme de table et la fonction plotOutput qui permettra de visualiser les données sous forme de graphe.

```
        mainPanel(
    tabsetPanel(type = "tabs",
                tabPanel('TABLES', DTOutput("tables")),
                tabPanel("TABLE DATABASE", DTOutput("table_data")),
                tabPanel('VISUALISATION',plotOutput("graphe"))
    )
)
```

2. La fonction **serveur** : contient les instructions qui gèrent les calculs et les mises à jour basées sur les actions de l'utilisateur. Afin de rendre chaque widget actif et faciliter l'interaction avec l'utilisateur, notre serveur est divisé en plusieurs fonctions parmi lesquelles :
une fonction **import_file** crée pour récupérer les données depuis un emplacement de votre machine et charge dans le fileInput.

   Une fonction réactive **observe()** qui surveille tous les changements effectués sur le fileInput. S'il n'est pas vide, notre fonction **basename()** va récupérer les noms de chaque fichier et mettre à jour le selectInput() comme le décrit le code ci-dessous.

```
 observe({
#files récupère le chemin complet de tous les fichier
files <- input$file
# basename()permet d'extraire le nom des fichiers contenus dans ces chemins
if (!is.null(files)) {
  choices <- basename(files$name)
  #print(files$name)
  updateSelectInput(session, "select", choices = choices)
}
})
```

   une fonction de rendu **renderDT()** comme décrit le code ci-dessous, est affectée à la sortie DTOutput de l'interface utilisateur afin de rendre les données accéssibles sous forme de table en fonction du fichier sélectionné dans le selectInput.

```
    output$tables <- renderDT({
files <- input$file
if (is.null(files)) {
  return(NULL)
}
#vérifier si le nom de fichier selectionné sur le select
#input est présent dans la liste des noms de fichiers(file$name)

selected_files <- files$name %in% input$select

# on appelle la fonction a qui on donne le chemin et affiche nos données
data1 <- import_csv_files(files[selected_files, ])
```

```
  if (length(data1) > 0) {
    datatable(as.data.frame(data1))
  } else {
    NULL
  }
})
```

La sauvegarde du fichier vers la base de données utilise une fonction réactive **observeEvent** qui a pour rôle d'effectuer la sauvegarde lorsqu'on actionne le bouton **Sauvegarder**.

Pour démarrer l'application on utilise la fonction **runApp** qui prend en paramètre l'interface utilisateur et le serveur ou directement le fichier .R. Nous avons donc notre interface avec les widgets, les onglets et les données.

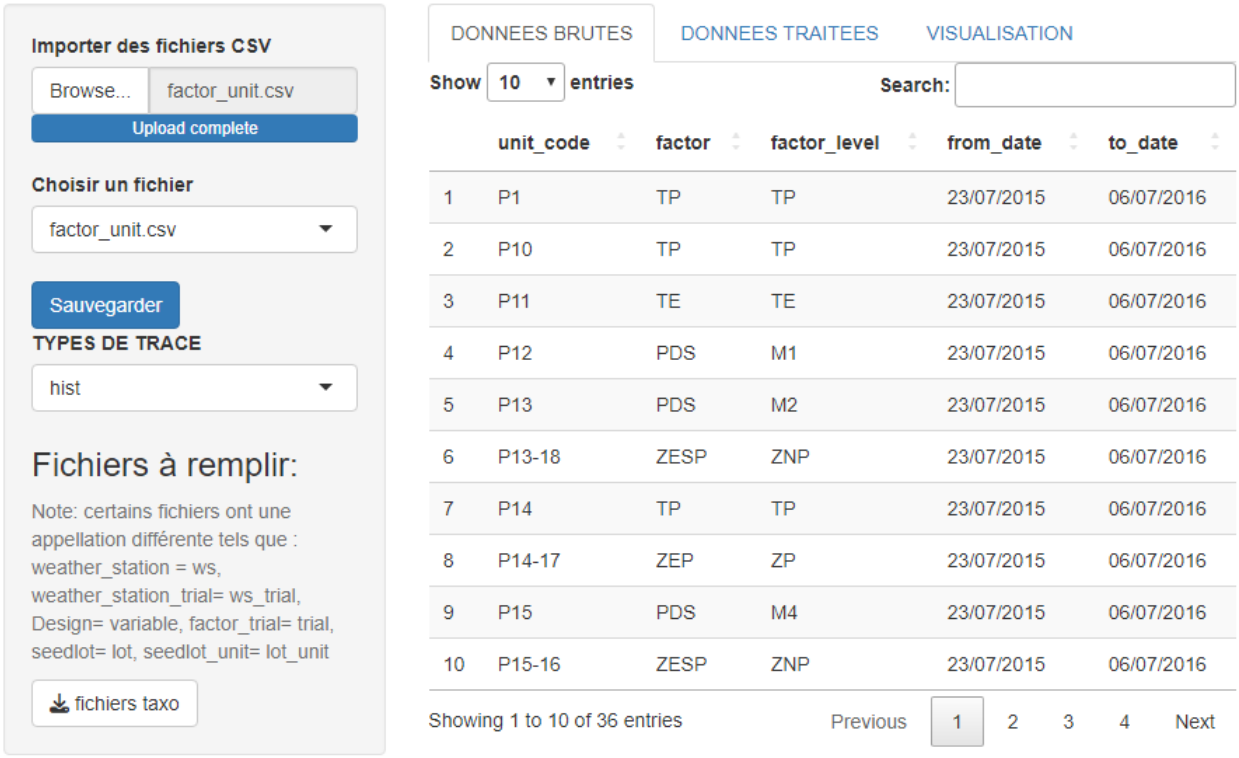

Figure 10: acceuil

4 AUTRES MISSIONS

Lors de mon stage, j'ai été affectée à diverses tâches et responsabilités qui ont été définies au début de la période de stage. Cependant, au fur et à mesure de l'avancement de mon travail, d'autres tâches m'ont été attribuées, qui ont nécessité un investissement de temps et d'efforts significatif.

4.1 AEGIS

Comme nous l'avons vu précédemment dans la section référencée 2.1, Aegis utilise principalement la version 3 de Bootstrap pour son design. Bootstrap est un framework qui propose une série de styles CSS prédéfinis, de composants JavaScript et de dispositions de grille réactives. Cela permet aux développeurs de créer facilement des interfaces Web cohérentes et visuellement attrayantes.

C'est sur cette base qu'un travail a été effectué par les précédents ingénieurs informatiques sur la maintenance de l'application web de Aegis.Dû aux différentes mises à jour des versions des langages et infrastrutures logicielles utilisées, le systeme présentait des dysfonctionnements.

Pour tester et corriger AEGIS localement, avant de le déployer sur un serveur en ligne, nous avons utilisé **WampServer** qui est un environnement de développement local populaire pour les applications Web.

Les missions à effectuer consistaient à :

1. renommer le sous menu Import Et Saisie : nous avons dans un premier temps changé le nom de ce sous menu qui était **"Importation et Saisie"** en **"Importation/Exportation"**
2. Réorganiser le sous menu Import Et Saisie Actuellement nous avons deux façons d'importer et sauvegarder les données réparties en deux onglets :
   — Importation(voir figure 11) : Onglet permettant d'importer et sauvegarder les données via différents menus (figure **??**).
   — Package (voir figure 12) : Onglet permettant de rendre notre package accessible depuis le site.
3. Création de l'espace pour le dépot du package Pour créer cet espace, nous avons créé dans le controleur, un fichier package avec une fonction **download_form**

```
 public function download_form()
{
    $this->load->helper('download');
    $file_path = 'download_forms/DagePackages.tar.gz';
    force_download($file_path, NULL);
}
```

   Nous chargeons la bibliothèque **'download'** à l'aide de la fonction **'load->helper'** . Ensuite, nous créons une variable à laquelle nous attribuons le chemin vers lequel se trouve le fichier à télécharger **'download_forms/DagePackages.tar.gz'**. Enfin nous utilisons la fonction **force_download** de la librairie **download** pour déclencher le téléchargement du fichier

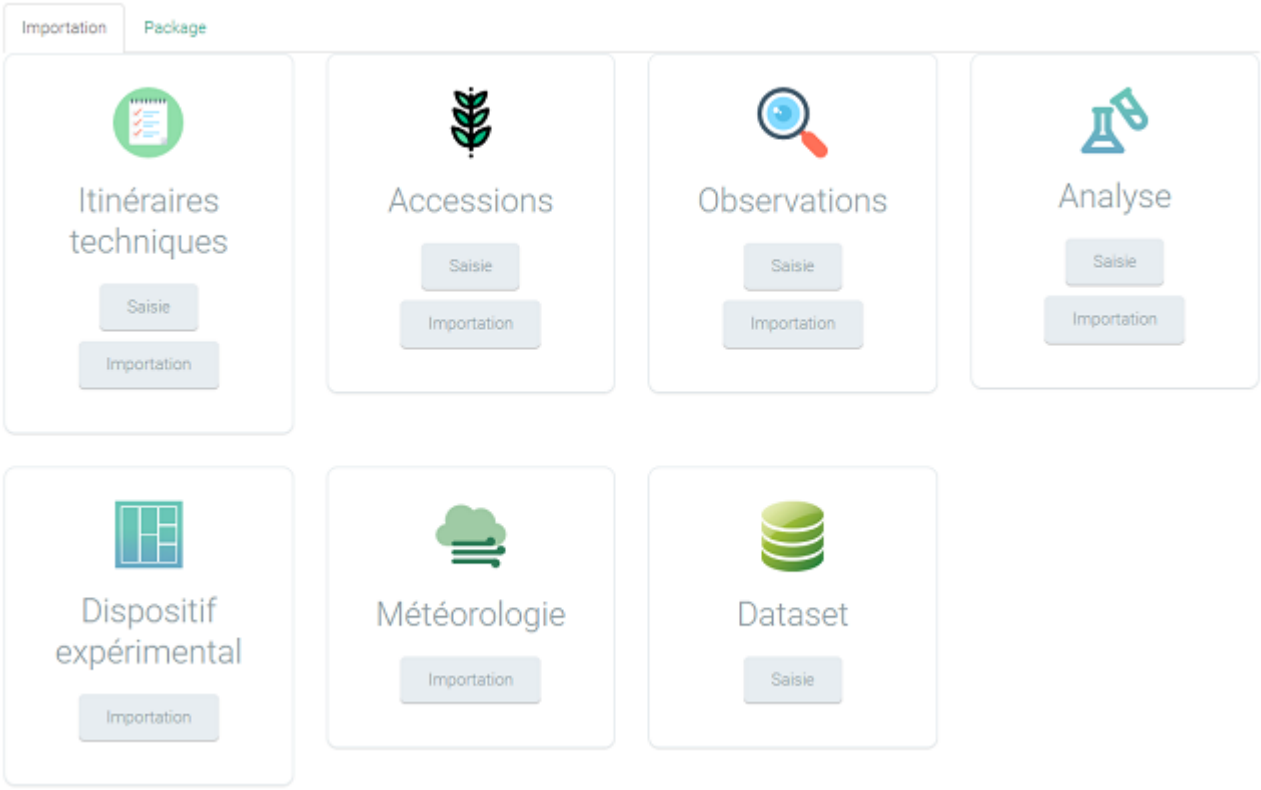

Figure 11: onglet Importation

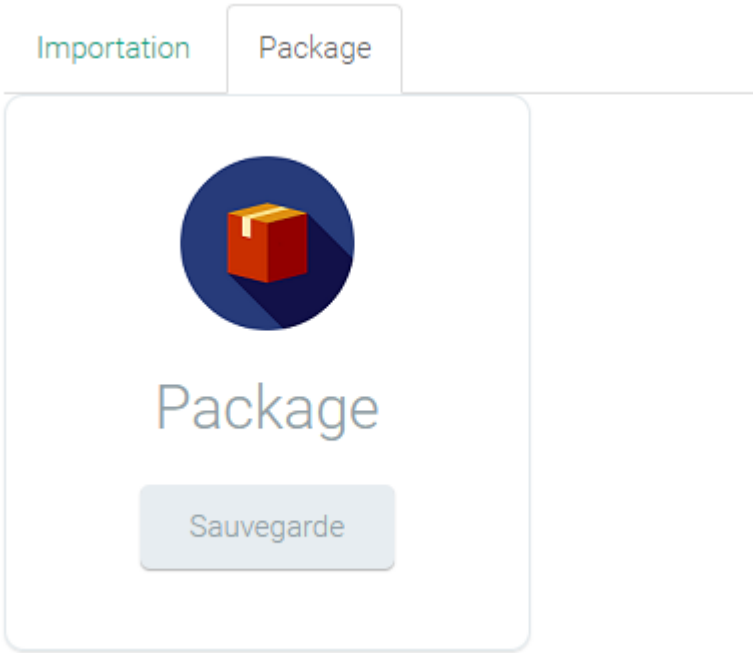

Figure 12: onglet Package

spécifié dans le chemin.
Nous avons ensuite créé une autre fonction **import** qui definit un titre et sous titre pour notre vue.

```
$this->view('Package/telechargement', $page['title'],
$page['subtitle'], $data, $scripts);
```

Dans la vue, nous avons créé un dossier package et un fichier téléchargement. Dans le fichier nous avons structuré à l'aide du code html notre page lui permettant de récuprer les informations sur l'entete de la page, qui viennent de la fonction import de notre controller

```
<?= form_open_multipart('Package/import', 'class="contact-form"');?>
```

, Ensuite nous avons une grille de deux colonnes de dimensions différentes, l'une présentant le guide d'utilisation du package 13 que les utilisateurs attentivement avant l'utilisation du package, et l'autre définissant le bouton de téléchargement du package14. Pour Fonctionner,

Figure 13: Guide d'utilisation du package

notre bouton fait appel à la fonction download_form du controller.

4. Correction des erreurs sur les Accessions Dans le sous menu Toutes les ressources, il était impossible d'afficher la liste des accessions de la base de données. Il renvoyait le message suivant :

5 Etat d'avancement du projet

Au début du stage, nous avons listé les tâches à effectuer afin d'avoir un meilleur suivi et anticiper les éléments qui pourraient être des freins au bon déroulement du projet. Nous avons,

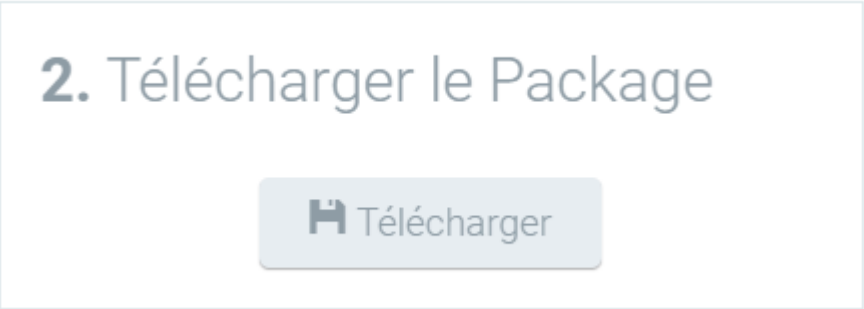

Figure 14: Bouton téléchargement du package

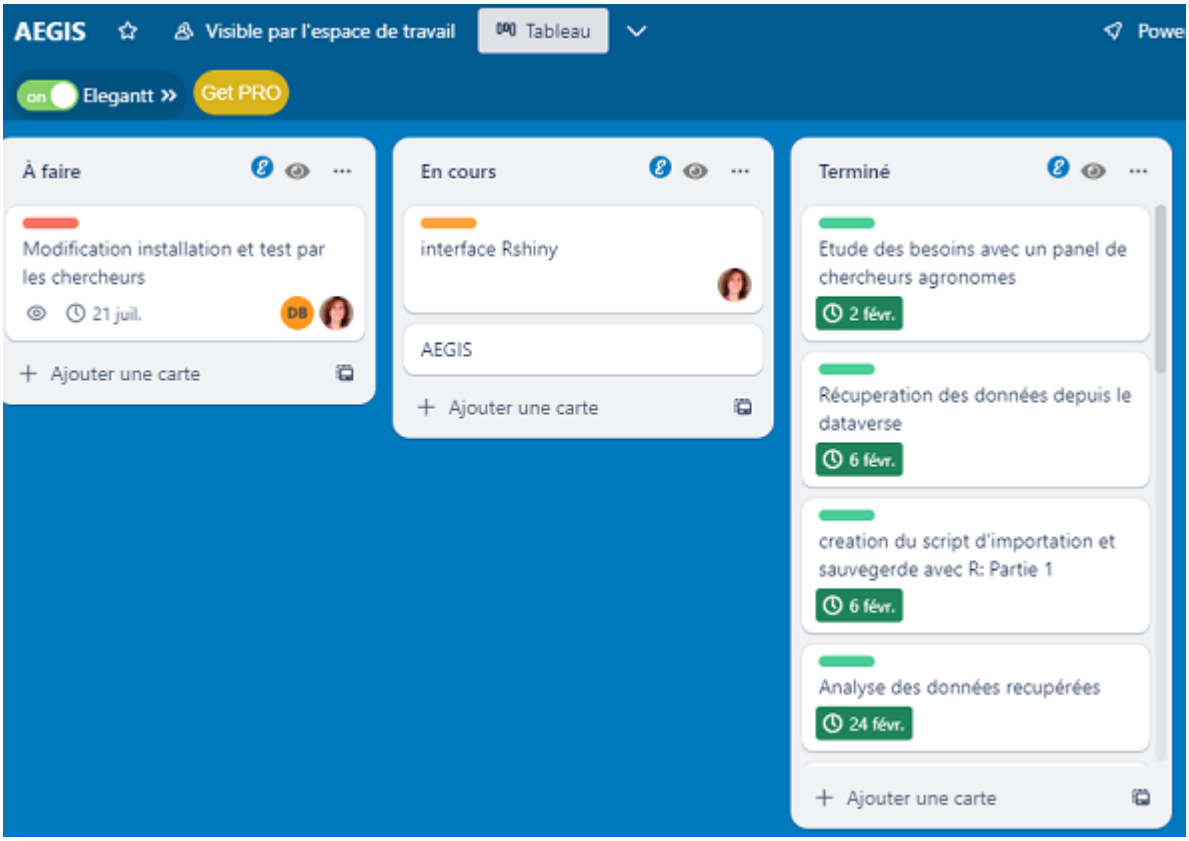

Figure 15: Liste des tâches

crée un dépot git pour partager le code source et utilisé l'outil de gestion de projet Trello (figure 15) pour répartir nos tâches en trois : travaux à faire, travaux en cours et travaux terminés afin partager l'état d'avancement du projet aux différents membres.Il est à noter que durant l'évolution du stage, des tâches se sont rajoutées. A ce jour, les taches les plus importantes sont terminées.

6 Difficultés rencontrées

6.1 Données

La principale difficulté résidait dans la compréhension générale des données.Il y a deux points à relever :

1. La correspondance entre chaque fichier et une table de la base de données. La plupart des noms de fichiers étaient conçus pour faciliter le travail des chercheurs, mais ne correspondaient pas aux noms des tables.
2. reconstruire un modèle relationnel en fonction des fichiers disponibles, afin de déterminer quels fichiers devaient être traités en premier.

6.2 R

Lors de la création du package, plusieurs difficultés se sont présentées parmi lesquelles :

1. Trouver des ouvrages qui traitent de R et la base de données. On trouvait majoritairement R sur l'analyse de données.
2. Difficultés à structurer le package particulièrement pour les fonctions et les exemples.
3. Nous avons été confrontés à plusieurs reprises à des conflits de noms de fonctions, car d'autres packages présentaient des fonctions portant le même nom. Cela a posé des difficultés dans la gestion des dépendances.
4. Structurer la fonction de sauvegarde vers la base de données en respectant les exigeances de chaque fichier.
5. comprendre le fonctionnement de Rshiny.

7 Conclusion

Le projet de stage réalisé pour valider mon Master avait pour objectif principal de créer un ensemble de traitements statistiques permettant une exploration et une visualisation des données contenues dans AEGIS à travers des scripts sous un package R et une interface RShiny. Ces scripts avaient pour but d'aider les chercheurs dans le traitement et la sauvegarde des données expérimentales, facilitant ainsi la prise des décisions et un gain de temps.

La création de ces scripts, pour en faire un package R et l'interface utilisateur sur RShiny a été une expérience passionnante et enrichissante. J'ai pu exploiter de nombreuses fonctions d'autres packages de R et de RShiny pour créer notre propre package et aboutir à une interface interactive et conviviale. De plus grace à l'integration de cette interface j'ai pu offrir une visualisation dynamique permettant aux utilisateurs d'obtenir des informations précieuses.

Durant ce stage, j'ai pu appliquer mes connaissances et compétences en base de données et en visualisation des données pour créer un outil utile et pratique. Ce projet m'a permis de développer des compétences en programmation R, en particulier dans le développement de packages et d'interfaces utilisateur sur RShiny qui étaient tous deux nouveaux pour moi.Il m'a également permis de travailler de manière autonome, et de découverir le travail collaboratif avec des personnes qui sont d'un autre domaine que le mien.

Pour des impératifs à régler sur le site de Aegis, le développement de cette page web est en cours. Cela pourrait donner l'occasion d'apporter une graphique sur les données, de les exporter et de créer une interface pour limiter les accès des utilisateurs par authentification. Par ailleurs il serait judicieux de créer un accès à l'interface depuis le site de AEGIS.

Annexes

A CIRAD

A.A Présentation de l'organisme d'Acceuil

Fondé en 1984, le Centre de coopération internationale en recherche agronomique pour le développement (Cirad)[1] est un organisme français dédié à la recherche agronomique et à la coopération internationale pour le développement durable des régions tropicales et méditerranéennes. Depuis 60 ans, le Cirad joue un rôle actif à La Réunion. Il est composé de 33 unités de recherche et de trois départements scientifiques :

— département Systèmes biologiques (BIOS) ;
— département Performances des systèmes de production et de transformation tropicaux (PERSYST) ;
— département Environnements et sociétés (ES).

Les missions du Cirad consistent à mener des activités de recherche et de développement qui ont bénéficié à la croissance des secteurs agricoles et alimentaires de La Réunion. Aujourd'hui, il accompagne ces secteurs dans leur transition écologique et leur adaptation au changement climatique. AIDA se positionne sur l'intensification durable des systèmes de production agricole, en particulier à base de cultures annuelles en milieu tropical. Ses recherches visent la pleine valorisation des ressources disponibles (eau, sols et biodiversité) au service de la production, notamment par la mobilisation des processus écologiques qui régissent leur dynamique au sein des agroécosystèmes

A.A Organigramme

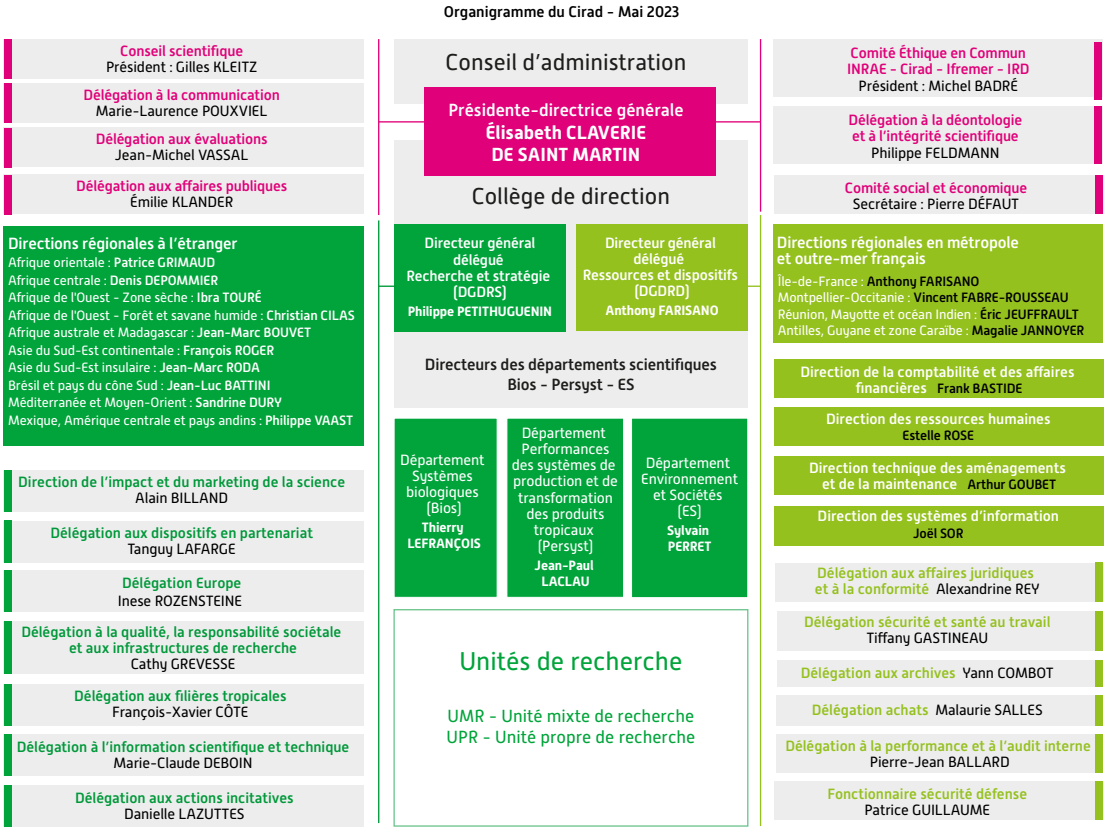

Figure 16: Organigramme

B Environnement de Travail

B.1 Base de données réduite

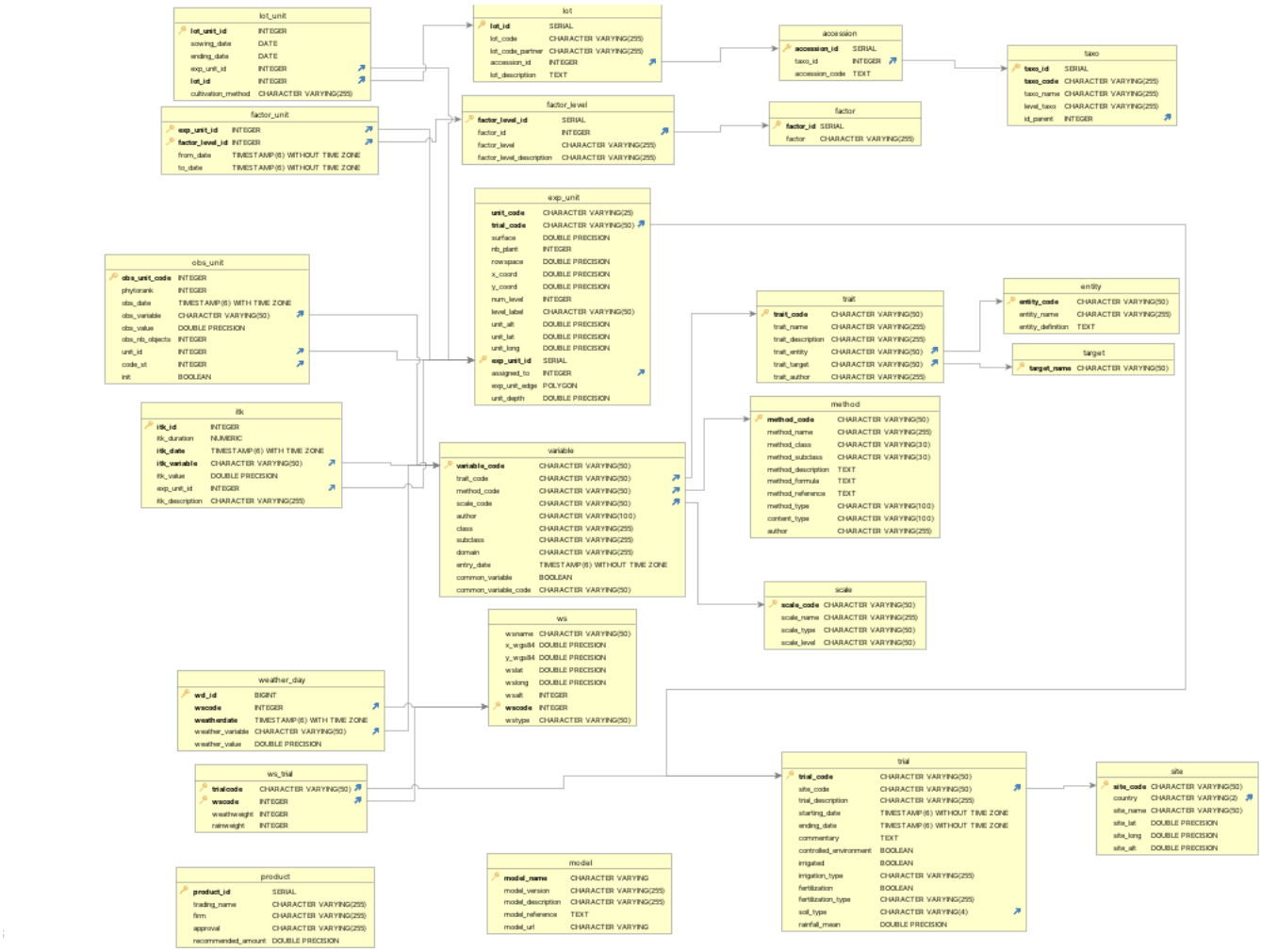

Figure 17: Modèle Relationnel des tables utilisées

B.2 Base de données AEGIS

C R

C.1 package

C.1.1 structure d'une fonction

```
    #' Importer des colonnes depuis la base de données
#'
#' Cette fonction nous permettra d'appeler la connexion et
#'récupérer certains champs dans des tables de la base de données.
#'
#' @param con  appelle la connexion qui permet d'accéder à la base de données
#' @param table_db récupère le nom de la table pour laquelle nous voulons des informations
#' @param champs récupère le nom des champs que nous voulons utiliser.
#'
#' @return fonction: Cette fonction, retourne les données des différentes
#'colonnes pris depuis la base de données
#' @export
#'
#' @examples
#'
#'
#'# -----Importation des données depuis la base de données
#'
#'library(DBI)
#'library(RPostgres)
#'
#'User='postgres'
#'mot_passe='dage'
#'host_db='localhost'
#'port_db=5432
#'dbname='daphne_dev'
#'con <- DBI::dbConnect(RPostgres::Postgres(), user=User, password=mot_passe,
#'                      host=host_db, port=5432, dbnam = dbname)
#'
#' # taxo
#'taxo_database<-import_database(con, "public.taxo", c("taxo_id", "taxo_code"))
#'
#' #wday
#'ws_database<-import_database(con, "public.ws", c("wscode", "wsname"))
#'
#'
```

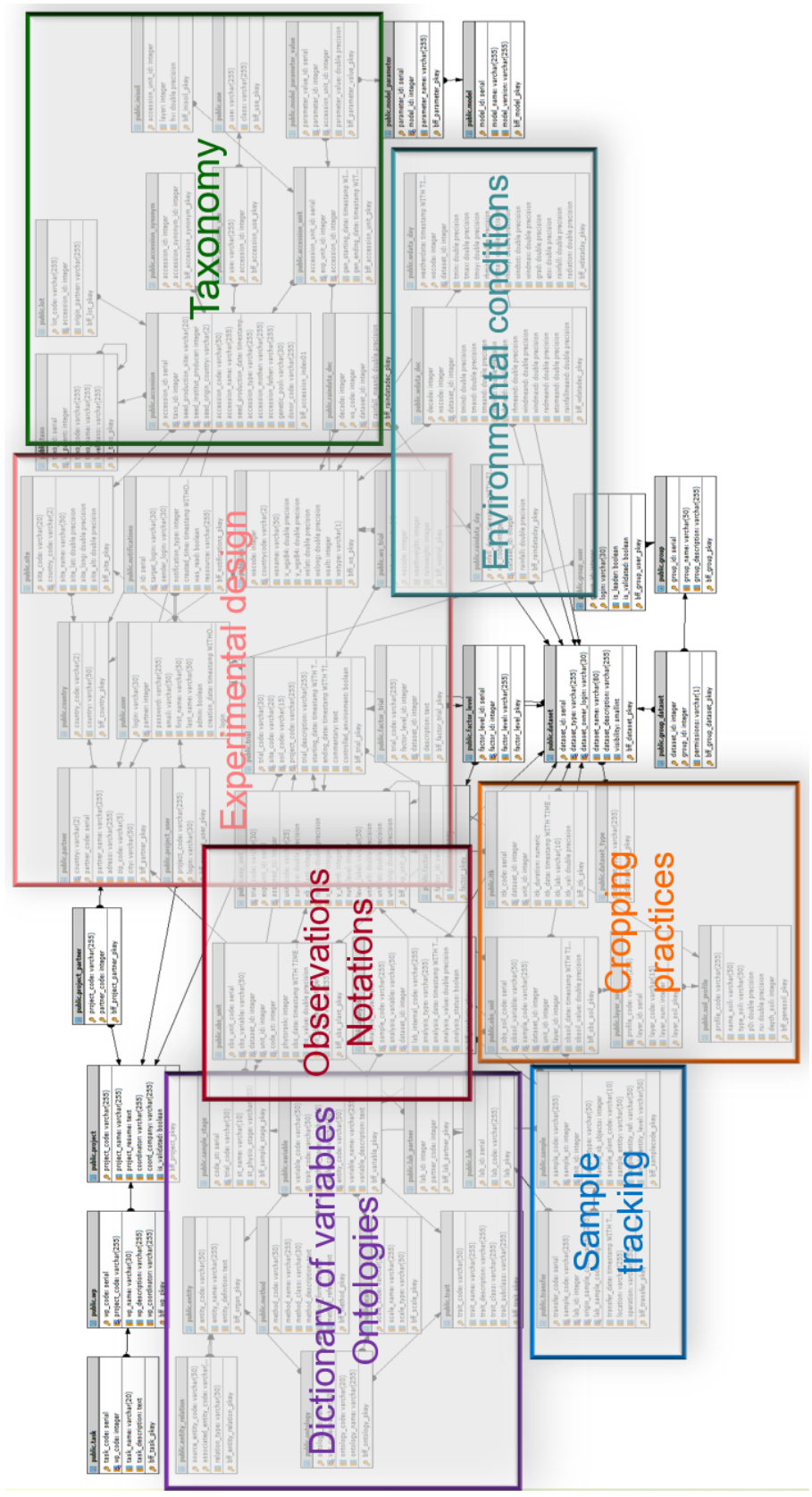

Figure 18: Modèle Relationnel AEGIS

```
#'

import_database<- function(con, table_db, champs ){
  query<- paste("SELECT", paste(champs,collapse=","), "FROM", table_db)
  result<-DBI::dbGetQuery(con, query )

  print(result)
}
```

C.1.2 Fonction d'importation des données depuis AEGIS

```
#' Fonction importation depuis votre espace de travail
#'
#'Cette fonction a pour but de récupérer les feuilles de tous les fichiers Excel.
#'A travers la fonction 'list.files()', il récupère les fichiers contenu dans
#'le chemin qui sera mentionné et met dans une liste. Ensuiste, il récupère le
#'nom des feuilles de chaque fichier avec la fonction 'excel_sheets()'.Enfin,
#'pour chaque feuille de fichier, assigne les données aux entetes et range
#'  dans l'environnement Global.
#'
#' @param chemin une chaine de caratère, qui récupère le chemin du dossier dans lequel se trouve n
#'
#' @return Fonction :Cette fonction nous retourne la liste des feuilles contenues dans chaque fich
#' @export
#'
#' @examples
#' \dontrun{
#' # ----- Importation des données depuis un emplacement de votre machine
#'
#' # créer une variable vide qui va récuperer la liste des fichiers
#' Aegis<-NULL
#'import_Aegis("C:/chemin/vers/le/dossier/des/fichiers/Excel")
#'
#'
#'}

import_Aegis<-function(chemin){

  files<- list.files(chemin, pattern = "*.xlsx", full.names = TRUE)
```

```
  files
  #compte le nombre de fichiers présents dans le document
  nb_files <- length(files)
  nb_files

  #creation d'un vecteur vide pour récupérer les entetes
  data_names <- vector("list",length=nb_files)

  #pour récupérer les entetes des feuilles
  for (i in 1 : nb_files){
    #data_names[i] <- strsplit(files[i], split=".csv")
    data_names [[i]]<- readxl::excel_sheets(files[i])

  }

  # creation d'une liste vide pour recuperer les fichiers
  data<-list()
  # importe les données et les associer aux feuilles
  for (i in 1:nb_files) { #pour les fichiers
    for (j in 1:length(data_names[[i]])){ # pour chaque feuilles de  fichiers
      data[[paste0(data_names[[i]][j])]]<-readxl::read_excel( files[i], sheet= data_names[[i]][j])
    }
  }
  Aegis<<-data
}
```

C.1.3 importation depuis la base de données

```
    #' Importer des colonnes depuis la base de données
#'
#' Cette fonction nous permettra d'appeler la connexion et
#'récupérer certains champs dans des tables de la base de données.
#'
#' @param con  appelle la connexion qui permet d'accéder à la base de données
#' @param table_db récupère le nom de la table pour laquelle nous voulons des informations
#' @param champs récupère le nom des champs que nous voulons utiliser.
#'
#' @return fonction: Cette fonction, retourne les données des différentes
#'colonnes pris depuis la base de données
```

```
#' @export
#'
#' @examples
#'# -----Importation des données depuis la base de données
#'
#'library(DBI)
#'library(RPostgres)
#'
#'User='postgres'
#'mot_passe='dage'
#'host_db='localhost'
#'port_db=5432
#'dbname='daphne_dev'
#'con <- DBI::dbConnect(RPostgres::Postgres(), user=User, password=mot_passe,
#'                      host=host_db, port=5432, dbnam = dbname)
#' # taxo
#'taxo_database<-import_database(con, "public.taxo", c("taxo_id", "taxo_code"))
#'
#' #wday
#'ws_database<-import_database(con, "public.ws", c("wscode", "wsname"))
#'

import_database<- function(con, table_db, champs ){
  query<- paste("SELECT", paste(champs,collapse=","), "FROM", table_db)
  result<-DBI::dbGetQuery(con, query )

  print(result)
}
```

C.1.4 exportation de la base de données

```
#' # Exportation des données dpuis la base de données
    Export_database <- function(trial_code, output_file) {

  User='postgres'
   mot_passe='dage'
   host_db='localhost'
   port_db=5432
   dbname='daphne_dev'
```

```
 con <- DBI::dbConnect(RPostgres::Postgres(), user=User, password=mot_passe,
 host=host_db, port=5432, dbnam = dbname)

Query <- paste0("select trial.*,
                rainweight,
                unit_code,
                site_name,
                cultivation_method,
                itk_description,
                from_date,
                lot_code,
                variable_code,
                trait_name, trait_target,
                method_name,
                scale_name,
                weatherdate,
                factor_level,
                factor,
                wsname,
                accession_code,
                entity_code,
                obs_value

      from public.trial
      left join  ws_trial  ON ws_trial.trialcode=trial.trial_code
      left join  exp_unit  ON exp_unit.trial_code= trial.trial_code
      left join  site  ON site.site_code = trial.site_code
      left join  itk ON itk.exp_unit_id= exp_unit.exp_unit_id
      left join  factor_unit  ON factor_unit.exp_unit_id= exp_unit.exp_unit_id
      left join  lot_unit ON lot_unit.exp_unit_id= exp_unit.exp_unit_id
      left join  lot ON lot.lot_id= lot_unit.lot_id
      left join  variable ON variable.variable_code= itk.itk_variable
      left join  trait ON trait.trait_code= variable.trait_code
      left join  method ON method.method_code = variable.method_code
      left join  scale ON scale.scale_code= variable.scale_code
      left join  factor_level ON factor_level.factor_level_id=factor_unit.factor_level_id
      left join  factor ON factor.factor_id= factor_level.factor_id
      left join  weather_day ON weather_day.weather_variable= variable.variable_code
      left join  ws ON ws.wscode = ws_trial.wscode
      left join  accession ON accession.accession_id = lot.accession_id
```

```
        left join  taxo ON taxo.taxo_id = accession.taxo_id
        left join  entity ON  entity.entity_code= trait.trait_entity
        left join  obs_unit ON obs_unit.unit_id = exp_unit.exp_unit_id

        where trial.trial_code='", trial_code, "'")

  essai <- DBI::dbGetQuery(con, Query)
  utils::write.csv2(essai, output_file, row.names = FALSE)

  DBI::dbDisconnect(con)
}
```

C.2 RShiny

Packages	Avantages	Limites
RShiny	— Shiny fournit une approche plus avancée pour développer des interfaces graphiques, en permettant aux utilisateurs de créer des applications web interactives sans avoir besoin de connaissances en programmation web — Shiny offre également de nombreuses fonctionnalités avancées, telles que la réactivité des données, les mises à jour en temps réel, la gestion des sessions utilisateur, l'intégration de widgets interactifs et la personnalisation graphique — Shiny permet de créer des tableaux de bord interactifs, des formulaires interactifs, des graphiques dynamiques et bien plus encore. Shiny utilise une syntaxe simple et intuitive basée sur R, ce qui facilite la création d'applications web interactives même pour les utilisateurs novices.	— Étant donné que RShiny est étroitement lié à R, les applications RShiny ne peuvent être exécutées que sur des serveurs compatibles avec R .Cela peut limiter la portabilité des applications RShiny dans certains environnements ou plates-formes spécifiques. — Bien que RShiny offre une syntaxe simple et intuitive, la création d'applications web complexes avec des fonctionnalités avancées peut nécessiter une connaissance approfondie de R, de HTML, de CSS et de JavaScript. Cela peut rendre le développement plus complexe et nécessiter des compétences supplémentaires.
Tcltk	Tcl/Tk : Tcl (Tool Command Language) est un langage de script utilisé pour créer des interfaces graphiques, tandis que Tk est une boîte à outils graphique qui permet de créer des fenêtres, des boutons, des menus, etc. Tcl/Tk est intégré à R grâce au package "tcltk". — Il permet aux utilisateurs de créer des interfaces graphiques simples en utilisant des fonctionnalités de base de Tcl/Tk.L'avantage de Tcl/Tk est sa simplicité d'utilisation et sa portabilité, car il est disponible sur différentes plateformes.	— peut être limité en termes de fonctionnalités avancées et de personnalisation graphique. — Apparence graphique limitée : Les capacités graphiques de base de Tcl/Tk peuvent sembler plus rudimentaires et moins esthétiques par rapport à d'autres outils de développement d'interfaces graphiques plus modernes. L'aspect visuel des interfaces créées avec Tcl/Tk peut sembler moins sophistiqué et attrayant. — Les applications utilisant Tcl/Tk peuvent être moins réactives ou moins performantes, en particulier lors du traitement de grandes quantités de données.

Table 1: Taleau comparatif de RShiny et tcltk

Références

[1] Ripoche Aude Soulie Jean-Christophe Auzoux Sandrine, Christina Mathias. *Coupling of cropping system models with the AEGIS platform*, 2020. available at `https://agritrop.cirad.fr/595121/`.

[2] Inc. CodeIgniter, EllisLab. *CodeIgniter est un framework libre écrit en PHP. Il suit le motif de conception MVC et s'inspire du fonctionnement de Ruby on Rails*, 4.2.11 edition, 2006. available at `https://codeigniter.com/`.

[3] Michael Stonebraker postgres. *PostgreSQL is a powerful, open source object-relational database system*, 15.2 edition, 1996. available at `https://www.postgresql.org`.

[4] *Dataverse est un entrepôt institutionnel des donnéesermet aux scientifiques du Cirad et à leurs partenaires de déposer, partager et/ou rendre publiques les données produites ou coproduites dans le cadre des travaux de recherche.*, 2019. available at `https ://dataverse.cirad.fr/`.

[5] Joseph J. Allaire rstudio. *RStudio est un environnement de développement gratuit, libre et multiplateforme pour R*, 12.0 edition, 2011. available at `https://www.rstudio.com`.

[6] Robert Gentleman R, Ross Ihaka. *R is a free software environment for statistical computing and graphics*, 4.2.3 edition, 1993. available at `https://www.r-project.org/`.

[7] Josué AFOUDA rshiny. *Shiny is an R package that makes it easy to build interactive web apps straight from R*, 1.7.4 edition, 2021. available at `https://shiny.rstudio.com`.

[8] Hervé Leclerc Wampserver, Romain Bourdon. *WampServer est une plateforme de développement Web de type WAMP, permettant de faire fonctionner localement des scripts PHP*, 3.3.0 edition, 2011. available at `https://www.wampserver.com/`.

[9] Rasmus Lerdorf. *PHP est un langage de script utilisé le plus souvent côté serveur. Il a été conçu pour permettre la création d'applications dynamiques*, 8.2.1 edition, 1994. available at `https://www.php.net/ChangeLog-8.php8.0.7`.

[10] Refnes Data. *W3Schools est un site web destiné à l'apprentissage en ligne des technologies web*, 1998. available at `https://www.w3schools.com/r/default.asp`.

[11] Jacob Thornton Mark Otto. *Bootstrap est un framework frontal populaire pour la création de sites Web et d'applications Web réactifs et mobiles*, 5.2 edition, 2011. available at `https://getbootstrap.com/`.