

Volume

1

UNIVERSITÉ DU QUÉBEC À MONTRÉAL

Kourrich Lab

Analyse Automatisée en R Guide

KOURRICH LAB

Analyse Automatisée en R Guide

Wissem Ben Romdhane
141 Avenue Président-Kennedy, • Kourrich Lab
Montréal, QC H3C 3P8
Mail : iwissemben@gmail.com • V.1.0

Table des matières

Introduction	1
Organisation.....	2
Répertoire	3
Scripts	5
Les librairies ou packages employées.....	10
Installation	12
Utilisation	14
Lancer l'analyse	14
Créer un graphique	14
Extension à d'autres tests.....	16
Troubleshooting	17
Annexe.....	18
Bibliographie	22

Introduction

Introduction générale à la structure et aux principes du programme





Ce manuel a pour objectif de vous introduire au **ProjetR**, un programme R d'analyse automatique des résultats issus du logiciel de video-tracking SMART. Nous verrons donc comment le **ProjetR** s'organise, comment l'installer et l'utiliser. Nous présenterons également quelques informations clés qui permettront d'étendre ses fonctionnalités et résoudre des problèmes qui pourraient être rencontrés. Une bibliographie est disponible en fin de document afin de vous introduire et d'approfondir les notions du **langage R** qui ont permis d'écrire ce programme.

ProjetR désigne stricto-sensu l'ensemble du code qui réalise les opérations de manipulation de données, de calculs statistiques, de génération des graphiques et des fichiers rapports Excel et PDF pour l'analyse.

Le **ProjetR** a été réalisé avec une volonté de standardiser la manipulation de grandes quantités de données et la génération de premiers graphiques pour une analyse préliminaire et globale des résultats issus du video-tracking. **ProjetR** se place donc à cheval entre le logiciel SMART et les étapes d'analyses ultérieures plus poussées dont l'utilisateur pourrait avoir besoin.

Il a nécessité l'utilisation d'une série de libraires ou packages pour étendre les fonctions du langage de base R.

ICON KEY

	Syntaxe répertoires
	Règle de syntaxe
	Fonction personnalisée
	Commande

Préalable : L'exécution de **ProjetR** nécessite la version 4.2.1 de R ainsi que Rstudio. L'installation de ces logiciels et des librairies du programme nécessite une connexion internet. En amont la génération des fichiers Excel par le logiciel SMART doit être standardisée pour que seules les valeurs des variables soient inconstantes. Le nom et le nombre de variables doivent être constants pour un test donné, paramétrez donc en amont le **Summary Report** de SMART avant exportation des données.

Pour tirer profit au maximum du programme et étendre ses fonctionnalités, consultez la bibliographie avec en priorité les références précédées d'un astérisque (*) et en particulier la « formation en ligne R aux MTES et MCTRCT » des ministères Français. Les modules

1, 2 et 5 constituent une excellente introduction au langage R et à RStudio. Ils ont servi de base à la conception de ce programme. Le module 6 n'est pas encore déployé à ce jour, mais il sera utile pour améliorer les rapports PDF si besoin.

Organisation

Présentons tout d'abord l'organisation et la logique globale du programme, afin de savoir comment se repérer. Cette section est donc subdivisée en trois sous parties, la première présente le **Répertoire** qui contient le projet ainsi que son arborescence, la deuxième présente les « morceaux » de programme ou **Scripts R** et leurs relations d'interdépendance. La troisième et dernière sous partie présente succinctement les librairies employées pour réaliser le programme.

Cette section sera utile tant pour le simple utilisateur que pour celui qui entend étendre les fonctions du **ProjetR**.

Répertoire

Le **ProjetR** est contenu dans un répertoire principal intitulé « `Projet_Automatisation_Analyse_R V.x.y` ». Ce dossier est **portable**, il comporte tous les éléments nécessaires et suffisants à son utilisation. Il est alors possible de le déplacer sur l'ordinateur sans nuire à son bon fonctionnement.

Cette arborescence est fixe et permet au programme de gérer les fichiers **input** et **output**, renommer l'un des dossiers induira une erreur de programme qui ne saura plus retrouver les différents fichiers.

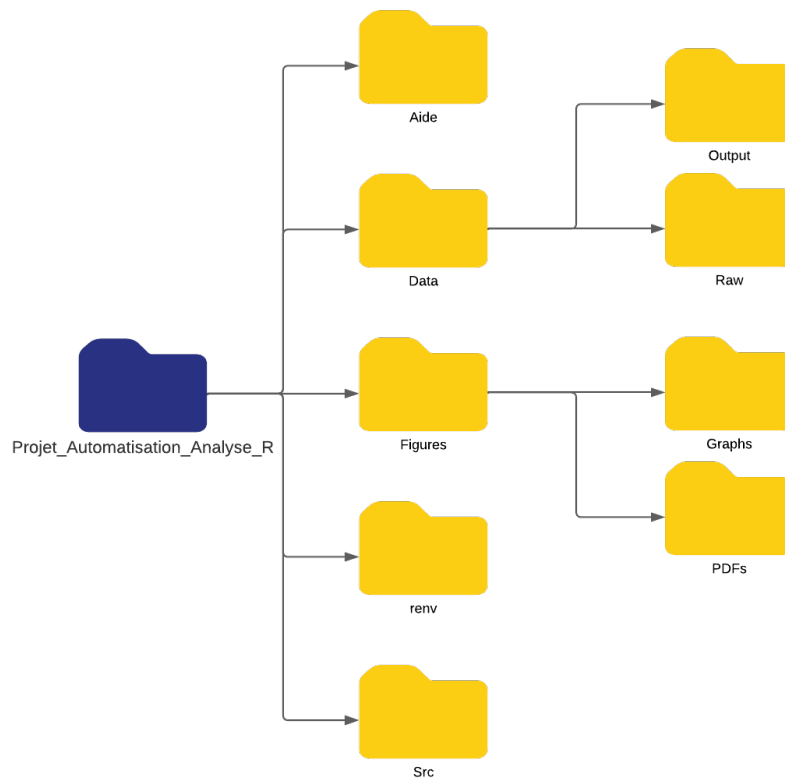






Figure 1: Arborescence du projet R

Le répertoire principal est organisé selon l'arborescence représentée en **Figure 1**. Chacun des sous dossiers représentés en Jaune dans la **Figure1** présente une fonction :

- Aide : Dossier comportant cette aide, et des images illustratives
- Data :
 - Sous dossier "Raw" : c'est dans ce dossier que l'on mettra les fichiers bruts à analyser. **Au format Excel « .xls »**
 - Des fichiers bruts test y sont présents (ex : "OF report Wissem.xls »)

- Sous dossier "Output" : c'est dans ce dossier que le programme sauvegardera un classeur Excel généré par l'analyse. Il résumera les données analysées (graphiques et tables associées). Le nom du fichier produit sera sous la forme :
 « Rapport_Nom_du_fichier_brut.xlsx »
- Figures :
 - Sous dossier "Graphs" : c'est dans ce dossier que seront sauvegardés les graphiques générés par l'analyse sous forme d'images, pour une utilisation potentielle. Le nom des fichiers images produits sera sous la forme :
 « Nom_du_fichier_brut_Graphi.png »
 - Sous dossier "PDFs" : c'est dans ce dossier qu'un rapport PDF de l'analyse sera sauvegardé. Le nom du fichier produit sera sous la forme :
 « Rapport_PDF_Analyse_Nom_du_fichier_brut.pdf »
- renv : Ce dossier contient toutes les librairies sauvegardées qui ont été importées lors de la conception. La sauvegarde fige la version des librairies du projet, afin d'empêcher leur mise à jour. Utiliser une version différente des librairies dans le projet peut provoquer des incompatibilités, rendant le programme inutilisable. Pour restaurer la sauvegarde des packages utilisez la commande  `renv::restore()`
- Src : Contient tous les scripts du **ProjetR**

Scripts

Comme nous l'avons dit plus haut, **ProjetR** est un programme écrit avec le langage de programmation R. Les instructions qui composent ce programme sont écrites sous forme de lignes de commandes dans des fichiers appelés « Scripts » avec une extension « .R ». Ces scripts sont écrits puis interprétés/exécutés par des logiciels tels que R ou RStudio.

Le **ProjetR** aurait pu être contenu dans un seul script, cette organisation aurait toutefois conduit à un programme indigeste. Il a alors été conçu en éclatant le programme et ses instructions sur plusieurs scripts **spécialisés** et **interdépendants**. Cette organisation à première vue plus complexe facilite en réalité la lecture et le repérage au sein du programme. Elle permet ainsi une meilleure lisibilité du programme, de faciliter la recherche et la résolution d'erreurs, mais avant tout facilite l'ajout de nouvelles fonctionnalités au programme.

Ainsi par extension - de la définition fournie en introduction – l'ensemble des morceaux de code ou **Scripts** contenus dans le **dossier Src** forment le programme **ProjetR**.

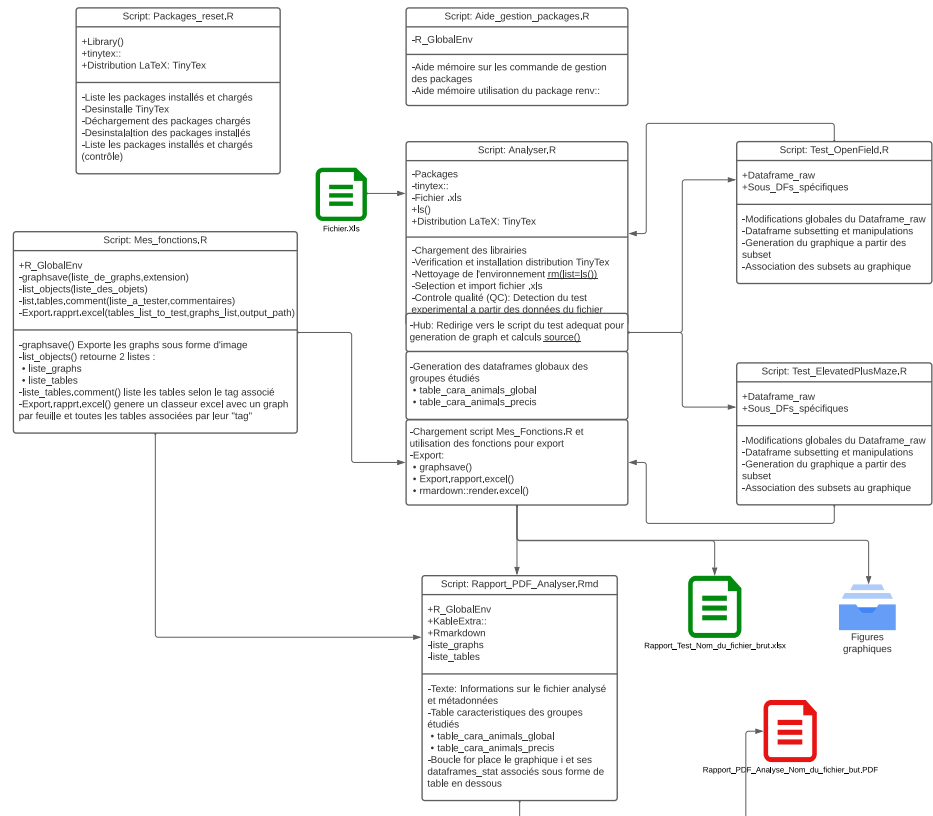



Figure 2: Digramme des relations entre les scripts du ProjetR

Le **projetR** a été conçu de sorte que tous les scripts qui le compose aient accès à l'environnement global du projet, c'est-à-dire à l'ensemble des variables créés par les différents scripts. La **Figure2** présente les relations de dépendance entre les différents scripts ainsi que les input-output du **ProjetR**.

Détail des différents scripts :

- *Mes_fonctions.R* : Regroupe des fonctions créées pour répondre à des besoins spécifiques rencontrés lors de la conception du **ProjetR**. Il peut être vu comme un script comportant des outils personnalisés, créés sur-mesure pour les employer dans des cas spécifiques. Une fonction peut être vue comme une suite d'instructions réalisant une tâche élémentaire. Les fonctions de ce script ont été conçues en faisant appel au langage R de base (*baseR*) ainsi qu'à des fonctionnalités supplémentaires apportées par des libraires externes ou packages.
 - Ce script ne s'exécute pas, il est appelé par d'autres scripts.
 - Pour appeler une fonction du script *Mes_fonctions.R*, il faut d'abord sourcer le script dans le script de travail puis appeler une de ses fonctions en y renseignant les arguments adaptés (variables de l'environnement global).
- *Aide_gestion_packages.R* : Il s'agit d'une aide-mémoire listant les commandes utiles pour la gestion des librairies (packages) du projet et en particulier le package *renv*.
- *Packages_reset.R* : Permet de réinitialiser les packages installés dans le cas où un problème toucherait ces derniers. L'exécuter permet de réaliser les opérations suivantes dans cet ordre :
 1. Liste les librairies installées, et les librairies chargées.
 2. Désinstalle la distribution **TinyTex** (nécessite la librairie *tinytex* préchargée).
 3. Décharge les librairies chargées.
 4. Désinstalle toutes les librairies installées par l'utilisateur.
 5. Liste les librairies installées, et les librairies chargées.
- *Rapport_PDF_Analysé.Rmd* : Il s'agit d'un script particulier, si vous avez l'œil vous vous êtes sans doute aperçus que son extension n'est pas en « . R » mais en « .Rmd ». Rmd est un diminutif de RMarkdown, il s'agit d'un format de fichiers permettant de produire des documents dynamiques avec R. Simplement, ce format permet de combiner au sein d'un même fichier (au moins) 3 langages : le YAML (métadonnées), du texte ainsi que des morceaux de code (code chunks). Ceci permet de produire des documents dont le contenu varie selon les variables renseignées en input du fichier

«.Rmd»¹. Notre fichier *Rmd* est conçu pour générer un fichier rapport au format PDF contenant plusieurs informations :

1. Les caractéristiques du fichier analysé, et celles de l'expérience. Ceci nécessite de combiner à la fois du texte et des lignes de code dits « inline code » (code chunks intégré au texte).
2. Deux tableaux renseignant la composition globale des groupes étudiés. Ces deux tableaux se basent sur deux *dataframes* construits dans le script *Analyser.R* (étape indépendante du test)
3. Les graphiques construits ainsi que les tables statistiques associées. Ceci est réalisé par une boucle *for* qui va répéter la suite d'instructions suivante pour chaque élément de la liste des graphiques *liste_graphs* :
 - Placer le Graphi sur la page
 - Lister les *dataframes* statistiques ayant pour « tag » : « Graphi »
 - Sauter une ligne
 - Convertir les *dataframes* listés du Graphi en tables
 - Placer ces tables sur la même page les uns après les autres
 - Saut de page
4. Une fois tous les éléments placés, le fichier « .Rmd » subit une succession de conversions pour permettre d'interpréter en **LaTeX** les différents langages du fichier et générer un document PDF :
 « Rapport_PDF_Analyse_Nom_du_fichier_brut.pdf »

• Notes

- Le script « .Rmd » s'exécute par la commande *rmarkdown::render(arguments)* à la condition que toutes les variables qu'il nécessite en entrée soient créés et initialisées dans l'environnement global du projet. Par défaut, le script *Analyser.R* l'exécute en phase d'export.
- Pour que le programme puisse être capable d'associer une table statistique à un graphique et réaliser cette suite d'instructions, il est nécessaire de lui attribuer un « tag ». Lors de la création d'un *dataframe* pour construire un

¹ Pour plus de détails voir le chap2 de RMarkdown Cookbook.

graphique, **ne pas oublier** d'attribuer au dataframe un **tag** correspondant au nom du graphique associé (Cf. Chapitre2, Utilisation, Créer un graphique).






- Cet exemple illustre l'intérêt de RMarkdown pour produire des documents dynamiques. Peu importe le nombre de graphiques ou de tables statistiques associées, un fichier PDF sera généré avec un contenu mis à jour automatiquement.
- *Analyser.R* : Ce script est le script central qui met en relation tous les autres qui l'entourent (cf Figure2). Il réalise plusieurs fonctions :
 - Phase 1 : Initialisation programme
 1. Chargement des librairies (packages)
 2. Vérification de l'installation du logiciel (distribution) *TinyTex*², si non détectée installation.
 3. Nettoyage de l'environnement global du projet : suppression des variables créées précédemment
 - Phase2 : Importation du fichier à analyser et formatage du type des données
 - Phase3 : Contrôle qualité/HUB

Cette phase est dédiée à la détection du type de test à analyser


 1. Création d'une **table de référence** *Exps_mastertable* listant les tests à prendre en charge et les paramètres qui les caractérise (nom du test, durée du test, dimension du dispositif...)
 2. Création d'une **variable** *Cara_Exps_Fichier* listant les paramètres du test renseignées dans le fichier Excel à analyser.
 3. Test vérifiant à quelle ligne de *Exps_mastertable* correspondent les données de la liste *Cara_Exps_Fichier*.
 - Retourne la ligne de *Exps_mastertable*
 4. HUB :

Selon la ligne détectée (et donc le test détecté), le hub va appeler le script d'analyse du test adéquat pour la manipulation des données, le calcul et la génération de graphiques spécifique à chaque type de test.
 - Phase4 : Export

² *TinTex* est une distribution de *LaTeX* chargée de réaliser la dernière conversion du fichier intermédiaire .md en .pdf, elle s'installe en appelant la librairie installée et chargée tinytex et sa fonction `tinytex::install_tinytex()`. Pour plus de détails voir le chap2 de RMarkdown cookbook

1. Appel du script *Mes_fonctions.R* pour exploiter les fonctions personnalisées
 2.  Lister les dataframes (tables) et graphiques
 3.  Export des graphiques sous forme d'image (.png ou .pdf)
 4.  Export Excel :
 - Arguments : liste des tables, liste des graphs, nom fichier de sortie
 5. Génération du rapport PDF : exécution du script *Rapport_PDF_Analysé.Rmd*
- *Test_X.R* :
 Un script est disponible pour chaque test à analyser. Cette organisation en scripts distincts est pratique, car elle comportementalise les traitements spécifiques à chaque type de test.
 La structure de ce script est simple et suit la même logique :
 - Pour chaque Graphique que l'on veut créer :
 1. On crée un sous dataframe du *Dataframe_raw* et on sélectionne les variables d'intérêt
 2. On réalise nos traitements sur ce sous dataframe (grouper les données selon un facteur par exemple) et on en crée un nouveau basé sur ce dernier pour les statistiques
 3. On crée le graphique couche par couche en lui renseignant comme données d'entrée en argument les dataframes adéquats. Ce graphique doit être donné à une variable nommée selon le format « GraphXX »  avec X un chiffre
 4. Enfin on attribue à tous les dataframes créés pour le Graphi un « tag » de la forme « Df.Graphi » 
 On répètera ensuite l'opération autant que de graphiques souhaités.

• Notes

- Afin de maintenir une cohérence dans la nomenclature des scripts, on nommera les scripts associés à chaque test *Test_X.R*  ou X est le nom du test en question.
- Bien que le HUB du script *Analyser.R* se charge de détecter le type de test dont est issu le fichier Excel brut, renseigner des données erronées conduira à une erreur au niveau du HUB ou en aval lors de la génération des graphiques.
- Par ailleurs les fichiers Excel bruts sont différents d'un test à l'autre et les variables également, traiter un fichier Excel d'un test Openfield avec le

script `Test_ElevatedPlusMaze.R` résulterait en une erreur à la génération des graphiques.

Les librairies ou packages employés

Le **tidyverse** est -comme son nom l'indique- un univers bien rangé. Il est un regroupement de plusieurs librairies (cf Figure3). Ces librairies lorsqu'elles sont employées de concert permettent de standardiser et faciliter la manipulation des données dans R. Sans le **tidyverse**, le travail des données serait bien fastidieux.

Une **librairie** ou **package** en programmation est un ensemble de fonctions et de classes permettant de réaliser des actions élémentaires. Les packages permettent d'étendre les fonctionnalités d'un langage de programmation et de faire gagner du temps au programmeur.

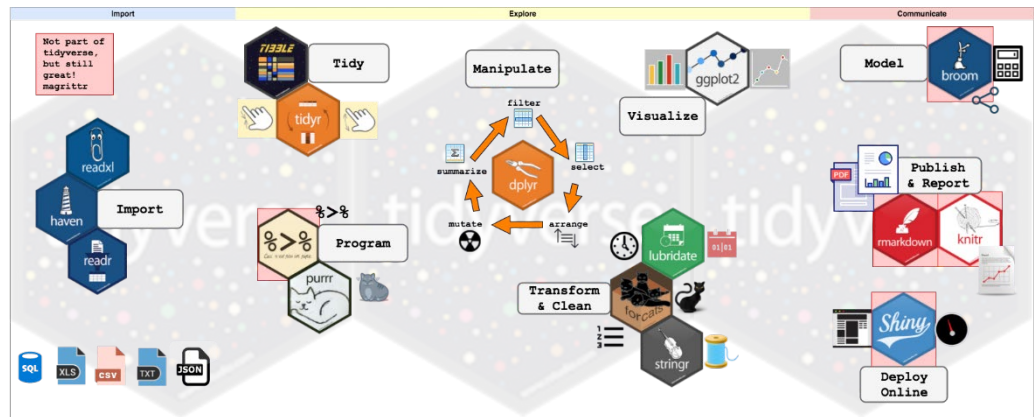


Figure 3 : Les packages composant le **Tidyverse**

Présentons succinctement les librairies du **tidyverse** utilisées dans ce programme. Chacune des librairies présente un site internet sur lequel vous pourrez approfondir vos recherches.

- **Tidy** : Permet de remettre en forme et d'organiser les dataframes de manière à les rendre conforme au standard **tidy**.
- **Pipes** : Etend la syntaxe R en apportant l'opérateur pipe « `%>%` ». L'opérateur pipe allège l'écriture et facilite la lecture des instructions pour la manipulation des données.
- **Dplyr** : Etend la syntaxe R pour permettre la manipulation des données. En apportant notamment 6 fonctions élémentaires ou **verbes** représentant des actions à appliquer à un tableau de données (dataframe).
- **Ggplot2** : Conception de graphiques à partir de dataframes





- Forcats : Gestion des facteurs et des variables qualitatives pour le calcul statistique et les graphiques
- RMarkdown et knitr : Génération du rapport PDF
- Readxl : Prise en charge des fichiers .xls pour l'import (Non inclus).
- Openxlsx : Production de fichiers Excel personnalisés (Non inclus).
- Chron : Gestion des variables de type temps (Non inclus).
- KableExtra : Conversion des dataframes en tables lisibles en LaTeX pour RMarkdown (Non inclus).

Pour plus de détails, voir la bibliographie et le module 2 chapitre 2 du MTECT-MTE / Formations à R explique en détail les fonctionnalités et la syntaxe propre aux différents packages du tidyverse.

Installation

Vous connaissez désormais la structure globale du programme, voyons maintenant comment installer le **ProjetR** sur votre ordinateur.

Prérequis : R.4.2.1, Rstudio, connexion internet.

1. Décompressez l'archive  *Projet_Automatisation_Analyse_R V.x.y.zip*
2. Vous trouvez alors un dossier portant le même nom, vous pouvez le placer où vous le souhaitez sur votre ordinateur. Sur le bureau par exemple.
3. Ouvrez le dossier, vous retrouverez l'arborescence explicitée en Figure 1.
4. Exécutez alors le fichier « *Projet_Automatisation_Analyse_R.Rproj* », ce fichier est un « fichier projet ». Il s'exécute avec le logiciel Rstudio. La console prendra la majorité de la fenêtre de Rstudio (cf. Annexe Figure 5).
5. Sélectionnez ensuite l'ensemble des scripts contenus dans le dossier *Src* et ouvrez-les à l'aide du menu clic-droit. Tous les scripts devraient alors apparaître dans la fenêtre de Rstudio au-dessus de la console (cf. Annexe Figure 6).
6. Placez-vous ensuite sur le script « *Analyser.R* » en sélectionnant l'onglet adéquat sur Rstudio.
7. Un message en jaune dans la console vous indique que le projet n'est pas synchronisé (cf. Annexe Figure 6). En effet les packages (bibliothèques) n'ont pas été trouvés dans le projet. Les bibliothèques doivent être préalablement installées avant de pouvoir être chargées.
 - Pour installer les bibliothèques demandées exécutez dans la console la commande  `renv::restore()`. Validez ensuite en entrant « y » puis « Entrée » dans la console. L'installation se conclut par l'apparition d'un chevron orange (cf. Annexe Figure 7).
8. Exécutez ensuite le script « *Analyser.R* » avec la combinaison de touches  **Ctrl+Shift+S**. Une fenêtre secondaire apparaît vous demandant de sélectionner le fichier  **.xls** brut à analyser.

9. Si R ne détecte pas de distribution *TinyTex* le programme l'installera automatiquement avec les librairies qui ont été chargées.
10. Des fichiers rapport d'analyse sont générés directement après l'installation indiquant le bon fonctionnement du programme (cf. Annexe Figure 8).


L'installation est terminée, fermez la fenêtre Rstudio et sauvegardez l'espace de travail. Par la suite, il ne sera plus nécessaire de réaliser ces étapes tant que le dossier du projet n'est pas déplacé.


• Notes

- Le fichier « .Rproj » permet entre autres de sauvegarder l'état du projet (Scripts dernièrement ouverts, lignes de commandes exécutées, figures générées..) après chaque utilisation et de le retrouver lorsqu'on réouvre le fichier.
- Une connexion internet est requise seulement pour le téléchargement et l'installation de la distribution de *TinyTex* et des librairies. L'accès à internet peut être ensuite retirée.
- Les fichiers bruts sortant de SMART à analyser doivent être placés dans le dossier Data/raw. Le nom des fichiers de sortie sera fonction de celui du fichier brut à analyser.
- La 1ere étape d'installation de *TinyTex* (7) est rapide. Elle consiste à télécharger la distribution, et se termine comme dans l'image (cf Annexe Figure 9)
- La seconde étape de l'installation de *TinyTex* est lente et se déroule au moment de générer le PDF pour la 1ere fois à partir du fichier temporaire ".md" (cf.PDF LaTeX erreur 3). Elle se conclut en générant le fichier PDF souhaité (cf Annexe Figure 10)
- Après ces 5 étapes, le projet R est bien installé sur l'ordinateur et peut être utilisé simplement en utilisant Ctrl+Shift+S sur le script "Analyser.R". Il ne sera plus nécessaire de réinstaller *TinyTex* sur le poste par la suite.

Utilisation

Exécuter un script

 Ctrl+Shift+S

Une fois l'installation du programme complétée, relancez le fichier « `Projet_Automatisation_Analyse_R.Rproj` » et placez-vous sur l'onglet « `Analyser.R` ». Pour exécuter le script, utilisez la **combinaison de touches** :  Ctrl+Shift+S. Cette combinaison de touches exécute la totalité du script. D'autres combinaisons de touches exécutent une partie du script, il n'est pas recommandé de les employer sauf si l'utilisateur appartient à un public averti.

Lancer l'analyse

• Notes



- N'oubliez pas de placer vos fichiers Excel à analyser **au format .xls** dans le dossier :

 `Data/raw`


- N'exécutez que le script « `Analyser.R` » pour lancer l'analyse.



Créer un graphique

Vous pouvez étendre l'analyse d'un test comportemental, en ajoutant de nouveaux graphiques. Pour ajouter des graphiques, vous devez suivre le schéma suivant :

1. Créez un sous dataframe (variable) basé sur le  `dataframe_raw` (données brutes du fichier Excel importé et formaté).
2. Sélectionner ensuite uniquement les paramètres pertinents (noms de colonnes) pour la génération du graphique souhaité. Nommez votre sous dataframe avec la **syntaxe**  `SousDF_Nomexplicite` afin de faciliter la relecture de votre code.

Utilisez l'opérateur pipe `%>%` afin de passer les variables dans des fonctions qui vont les manipuler et rendre votre code plus lisible.

3. Créez un autre sous dataframe basé sur le dataframe précédent. Nommez le avec la **syntaxe**  `SousDF_Nomexplicite_stat`, le terme « `_stat` » est important car il permet au programme de reconnaître les tables statistiques.
4. Réalisez dans ce dernier sous dataframe les calculs statistiques adéquats. Par exemple pour représenter la dispersion des données dans un graphique par la SEM vous devrez calculer sur `SousDF_Nomexplicite` la moyenne de la variable d'intérêt, son écart type, le nombre d'observations pour ensuite calculer la SEM.

5. Vous avez désormais deux sous dataframes basés sur les données brutes : *SousDF_Nomexplicite* et *SousDF_Nomexplicite_stat*. Le premier contient toutes les observations brutes, et le second les statistiques des groupes sur ces observations brutes.
 6. Pour construire un graphique, il faut au préalable créer une variable nommée avec la **syntaxe**  *GraphXY*. Avec X la dizaine et Y l'unité. Pour créer le graphique 1, nommez le *Graph01*, le 2e *Graph02*, le 11e *Graph11* et ainsi de suite.
 7. Ensuite appelez la fonction *ggplot* en l'affectant à la variable *GraphXY*, spécifier le sous dataframe sur lequel le graphique doit se baser *SousDF_Nomexplicite* (celui contenant les observations brutes). Puis affecter aux ordonnées et abscisses les variables adéquates.
 8. Construisez ensuite le graphique couche par couche selon vos besoins. Pour un diagramme en barres avec des barres d'erreurs SEM et des points représentant chaque observation, ajoutez l'objet *geom_bar*, puis *geom_errorbar*, puis *geom_point*. En faisant attention à ce que chaque couche ait en X et Y les données adéquates.
- Notez que chaque *geom* hérite des paramètres X et Y précédents.
9. La variable *GraphXY* sera désormais un objet de classe **ggplot**.
 10. Enfin attribuez **systématiquement** un **tag** à chaque dataframe utilisé pour former vos graphiques. Ce tag doit être de la forme  « *Df.GraphXY* », il permettra d'associer les dataframes aux graphiques pour l'export des résultats. Ne pas le spécifier résultera en une erreur.

• Notes



- Le Chaque *geom* hérite des paramètres X et Y précédent, si ils doivent employer des variables issues d'autres dataframe, spécifiez le dataframe d'intérêt et renseigner les variables X et Y.

Si vous souhaitez voir un exemple, observez le code utilisé pour générer le Graph1 du script *Test_OpenField.R*.

Le module 5 du MTECT-MTE / Formations à R explique en détail les fonctionnalités et la syntaxe propre au package ggplot2.

Extension à d'autres tests


Pour prendre en charge d'autres tests comportementaux, il est nécessaire de suivre les étapes suivantes.

1. Commencez par créer un nouveau script via Rstudio. Avant de commencer à écrire vos premières instructions, sauvegardez le dans le sous-dossier  Src du projet, et nommez le fichier selon le format suivant :  Test_Nomdutest.R . Le nom de fichier ne doit comporter aucun caractère spécial, ni d'espaces, utilisez plutôt des **underscores** pour séparer les mots.
2. Le nouveau script créé, sélectionnez le script « Analyser.R » sur Rstudio, trouvez la section « contrôle qualité du dataset » puis ajoutez via le code les caractéristiques attendues du test à prendre en charge dans la **table de référence** *Exps_mastertable* (cf Figure4).

	DExperiment	Exp.protocol.name	ACQ.time(HH:MM:SS.00)	AN.Time.interval.split	DimV(cm)	DimH(cm)
1	Open Field	OF	00:30:00	00:05:00	72	72
2	Elevated Plus Maze	EPM	00:05:00	00:00:00	65	65

Figure 4: Table de référence des tests pris en charge




Dans l'ordre les variables de la table sont : Nom du test [*VarExm1*], Acronyme du test (SMART) [*VarExm2*], Durée totale du test [*VarExm3*], Durée intervalle capture du test (SMART) [*VarExm4*], DimensionV[*VarExm5*] et DimensionH [*VarExm6*] en cm du dispositif de test.

3. La **table de référence** étendue, trouvez la section « HUB », ajoutez une condition alternative  *else if* et sourcez le script précédemment créé, en suivant la syntaxe que vous y trouverez.
4. Enfin sauvegardez les modifications du script.

Vous pouvez désormais réaliser vos traitements spécifiques au nouveau test à prendre en charge dans le script nouvellement créé. Ecrivez y donc les instructions souhaitées. Lorsque « Analyser.R » sera exécuté et le fichier du nouveau test sélectionné, le HUB exécutera automatiquement le nouveau script.

Troubleshooting

Un message d'erreur apparaît dans la console en jaune :

- Un message d'erreur [Error in library(XXX) :aucun package nommé 'XXX' n'est trouvé] (cf. Annexe Figure 11) :
 - Le programme a été exécuté sans avoir installé ou chargé les librairies requises.
 1. Entrez dans la console la commande :
 -  `renv::restore()`
 - Puis confirmez l'installation en entrant « y » dans la console
 - Si un message « The library is already synchronized with the lockfile » apparaît, voir point 3
- Un message d'erreur ["pdf latex" not found] (cf. Annexe Figure 12). apparaît après l'exécution du script principal « *Analyser.R* » :
 - La distribution ***LaTeX(TinyTex)*** n'est pas détectée ou présente un problème d'installation.
 - Faites appel au package `tinytex` pour
 1. La désinstaller : Exécutez la commande :
 -  `tinytex::uninstall_tinytex()`
 2. La réinstaller : Exécutez la commande
 -  `tinytex::install_tinytex()`.

Notez que l'installation et la désinstallation de la distribution ***TinyTex*** nécessite d'avoir préalablement installé et chargé le package `TinyTex`

- Si un problème touche les librairies., réinitialisez l'environnement R en restaurant les librairies et la distribution ***TinyTex*** peut suffire à le résoudre.
 - Ouvrez le script « *Packages_reset.R* » et sélectionner le dans R studio
 - Exécutez la totalité du script.

Ce script va d'abord lister les librairies installées et chargées, désinstaller la distribution ***TinyTex***, télécharger les librairies, les désinstaller puis lister de nouveau les librairies installées et chargées.

- Procédez alors à la réinstallation du projet comme pour une première installation
- Vous pouvez selon les cas exécuter seulement une partie du script si vous avez identifié la fonction touchée.

Annexe

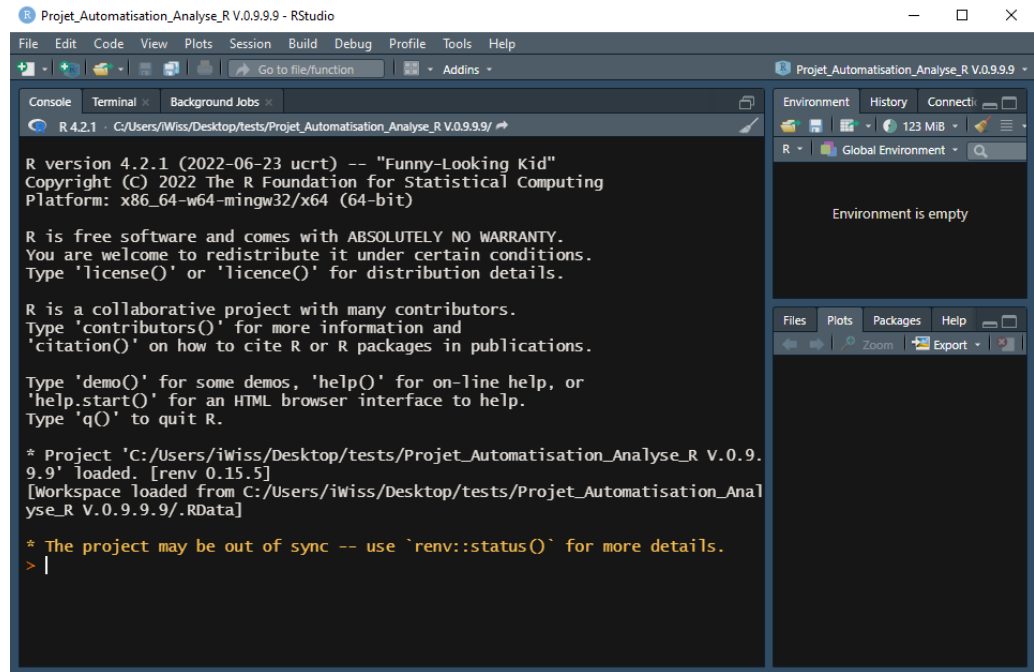


Figure 5: 1ere ouverture fichier Rproj

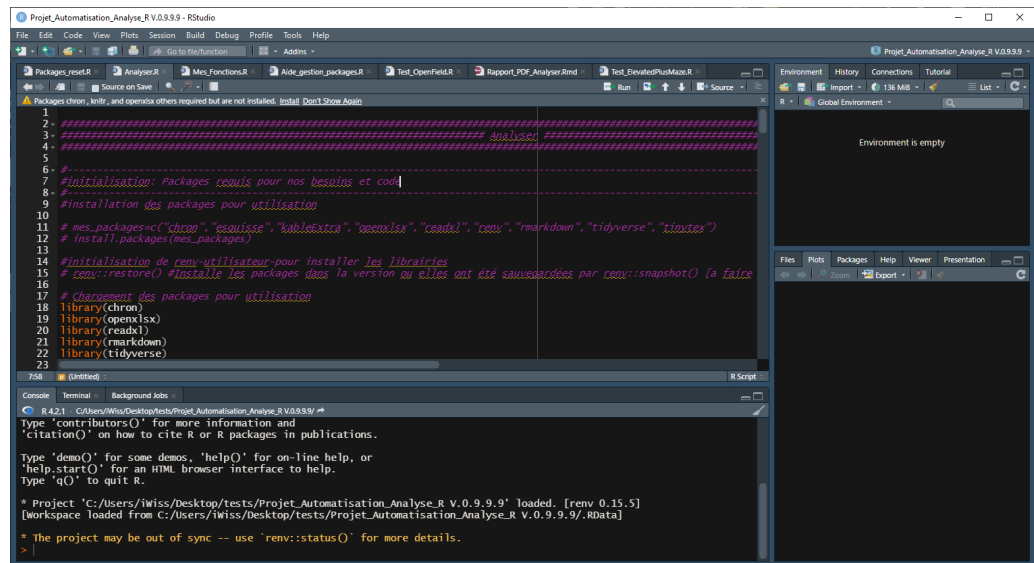


Figure 6: Rproj avec tous les scripts ouverts, Analyser.R sélectionné et librairies non synchronisées

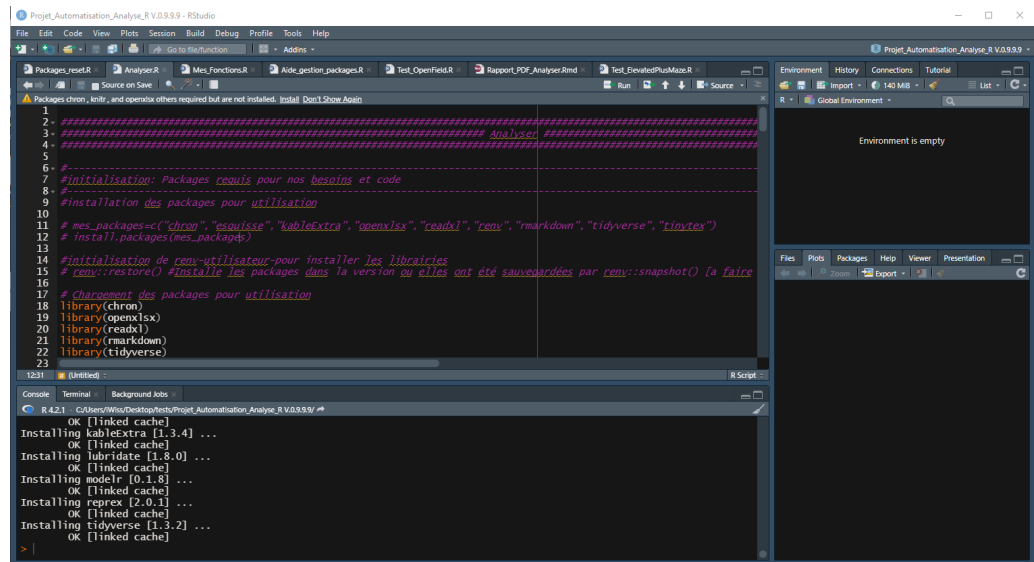


Figure 7: Installation des packages complétée, chevron orange dans la console

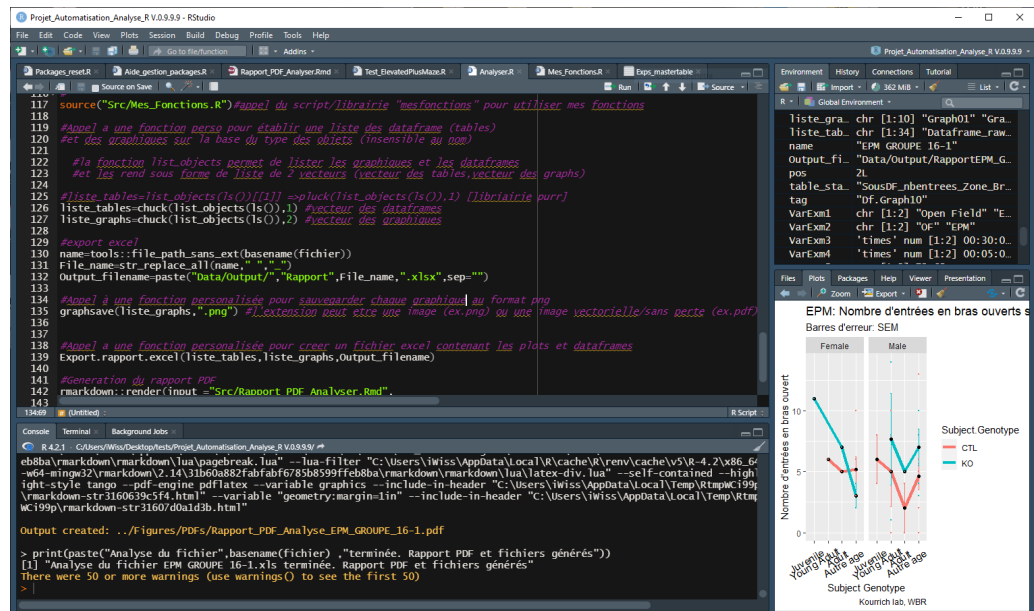


Figure 8: Fin d'installation, fichiers générés

```

R 4.2.1 C:\Users\Wiss\Desktop\tests\Projet_Automatisation_Analyse_R_V0.9.5 sans TT\Projet_Automatisation_Analyse_R_V0.9.6 /#
ngl_backslash --output ../Figures/PDFs/Rapport_PDF_Analyse_OF_report_Wissem.tex --lua-filter "C:\Users\Wiss\AppData\Local\cache\RenV\cache\v5\R-4.2\86-64-w64-mingw32\markdown\2.14\31b60a882fabfab6785b8599feb8ba\markdown\lua\pagebreak.lua" --lua-filter "C:\Users\Wiss\AppData\Local\cache\RenV\cache\v5\R-4.2\86-64-w64-mingw32\markdown\2.14\31b60a882fabfab6785b8599feb8ba\markdown\lua\latex-div.lua" --self-contained --highlight-style tango --pdf-engine pdflatex --variable graphics --include-in-header "C:\Users\Wiss\AppData\Local\Temp\RtmpOMP5jt\markdown-strcd047a51143.html" --variable "geometry:margin=1in" --include-in-header "C:\Users\Wiss\AppData\Local\Temp\RtmpOMP5jt\markdown-strcd04362320c.html"
Error: LaTeX failed to compile ../Figures/PDFs/Rapport_PDF_Analyse_OF_report_Wissem.tex. See https://yihui.org/tinytex/r/#debugging for debugging tips.
In addition: Warning message:
In system2(..., stdout = if (use_file_stdout()) fl else FALSE, stderr = f2) :
  "pdflatex" not found
> tinytex::install_tinytex()
trying URL 'https://yihui.org/tinytex/TinyTeX-1.zip'
Content type 'application/octet-stream' length 103686067 bytes (98.9 MB)
downloaded 98.9 MB
3 fichier(s) copi  (s)
1 fichier(s) copi  (s)
Running fc-cache -v -r

```

Figure 9: Fin 1ere   tape installation TinyTex

```

R 4.2.1 C:\Users\Wiss\Desktop\tests\Projet_Automatisation_Analyse_R_V0.9.5 sans TT\Projet_Automatisation_Analyse_R_V0.9.6 /#
[1/1, ???/??/??] install: ulem [7k]
running mktexlsr ...
done running mktexlsr
tlmgr.pl: package log updated: C:\Users\Wiss\AppData\Roaming\TinyTeX\texmf-var\web2c\tlmgr.log
tlmgr.pl: command log updated: C:\Users\Wiss\AppData\Roaming\TinyTeX\texmf-var\web2c\tlmgr-commands.log
tlmgr.pl: package repository https://mirror.its.dal.ca/ctan/systems/texlive/tlnet (not verified: gpg unavailable)
[1/1, ???/??/??] install: makecell [5k]
running mktexlsr ...
done running mktexlsr
tlmgr.pl: package log updated: C:\Users\Wiss\AppData\Roaming\TinyTeX\texmf-var\web2c\tlmgr.log
tlmgr.pl: command log updated: C:\Users\Wiss\AppData\Roaming\TinyTeX\texmf-var\web2c\tlmgr-commands.log
Output created: ../Figures/PDFs/Rapport_PDF_Analyse_OF_report_Wissem.pdf
> print("Analyse du fichier termin  e, Rapport g  n  r   et fichiers")
[1] "Analyse du fichier termin  e, Rapport g  n  r   et fichiers"
> #####Reste a faire#####
> ## faire
> ## fait
> ##### .... [TRUNCATED]
>

```

Figure 10: Fin 2e   tape installation TinyTex et g  n  ration du fichier PDF

```

R 4.2.1 C:\Users\Wiss\Desktop\tests\Projet_Automatisation_Analyse_R_V0.9.5 sans TT\Projet_Automatisation_Analyse_R_V0.9.6 /#
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

* Project 'C:\Users\Wiss\Desktop\tests\Projet_Automatisation_Analyse_R_V0.9.5 sans TT\Projet_Automatisation_Analyse_R_V0.9.6' loaded. [renv 0.15.5]
[workspace loaded from C:\Users\Wiss\Desktop\tests\Projet_Automatisation_Analyse_R_V0.9.5 sans TT\Projet_Automatisation_Analyse_R_V0.9.6/.RData]

* The project may be out of sync -- use 'renv::status()' for more details.
> source("C:\Users\Wiss\Desktop\tests\Projet_Automatisation_Analyse_R_V0.9.5 sans TT\Projet_Automatisation_Analyse_R_V0.9.6\Src\Analyser.R", echo=TRUE)
> #####Reste a faire#####
> ## faire
> ## fait
> ##### .... [TRUNCATED]
Error in library(chron) : aucun package nomm   'chron' n'est trouv  
>

```

Figure 11: Package non d  tect  

```

R 4.2.1 C:\Users\Wiss\Desktop\tests\Projet_Automatisation_Analyse_R_V0.9.5 sans TT\Projet_Automatisation_Analyse_R_V0.9.6 /#
$ results: chr "asis"
|.....| 100%
ordinary text without R code

output file: Rapport_PDF_Analyser.knit.md

"C:\Program Files\RStudio\bin\quarto\bin\tools\pandoc" +RTS -K512m -RTS Rapport_PDF_Analyser.knit.md --to latex --from markdown+autolink_bare_uris+tex_math_si
ngl_backslash --output ../Figures/PDFs/Rapport_PDF_Analyse_OF_report_Wissem.tex --lua-filter "C:\Users\Wiss\AppData\Local\cache\RenV\cache\v5\R-4.2\86-64-w64-mingw32\markdown\2.14\31b60a882fabfab6785b8599feb8ba\markdown\lua\pagebreak.lua" --lua-filter "C:\Users\Wiss\AppData\Local\cache\RenV\cache\v5\R-4.2\86-64-w64-mingw32\markdown\2.14\31b60a882fabfab6785b8599feb8ba\markdown\lua\latex-div.lua" --self-contained --highlight-style tango --pdf-engine pdflatex --variable graphics --include-in-header "C:\Users\Wiss\AppData\Local\Temp\RtmpOMP5jt\markdown-strcd047a51143.html" --variable "geometry:margin=1in" --include-in-header "C:\Users\Wiss\AppData\Local\Temp\RtmpOMP5jt\markdown-strcd04362320c.html"
Error: LaTeX failed to compile ../Figures/PDFs/Rapport_PDF_Analyse_OF_report_Wissem.tex. See https://yihui.org/tinytex/r/#debugging for debugging tips.
In addition: Warning message:
In system2(..., stdout = if (use_file_stdout()) fl else FALSE, stderr = f2) :
  "pdflatex" not found
> tinytex::install_tinytex()

```

Figure 12: Erreur "Pdf latex not found"

Bibliographie

- « Aesthetic Specifications ». Consulté le 23 août 2022.
<https://ggplot2.tidyverse.org/articles/ggplot2-specs.html>.
- Baek, Howard. *Welcome | Solutions to Ggplot2: Elegant Graphics for Data Analysis*. Consulté le 13 juillet 2022. <https://ggplot2-book-solutions-3ed.netlify.app/>.
- Barnier, Julien. *Introduction à R et au tidyverse*. Consulté le 23 août 2022.
<https://juba.github.io/tidyverse/index.html>.
- business-science.io. « Grafify: Make 5 Powerful Ggplot2 Graphs Quickly with R ». Business Science, 15 juin 2021. <https://www.business-science.io/r/2021/06/15/grafify.html>.
- « Clean a Project — Clean ». Consulté le 4 août 2022.
<https://rstudio.github.io/renv/reference/clean.html?q=distribut#actions>.
- Statistics Globe. « Detach All User-Installed Packages in R (Example) | Unload & Remove ». Consulté le 4 août 2022. <https://statisticsglobe.com/detach-all-user-installed-packages-in-r/>.
- DOI, Authors Affiliations Published Not published yet. « grafify: An R package for easy graphs, ANOVAs and post-hoc comparisons ». grafify. Consulté le 29 juillet 2022. <https://grafify-vignettes.netlify.app/>.
- Gesmann, Markus. « Plotting Tables Alongside Charts in R ». mages' blog, 14 avril 2015. <https://www.magesblog.com/post/2015-04-14-plotting-tables-alsongside-charts-in-r/>.
- *Grolemund, Yihui Xie, J. J. Allaire, Garrett. *R Markdown: The Definitive Guide*. Consulté le 25 juillet 2022. <https://bookdown.org/yihui/rmarkdown/>.
- Holtz, Yan. « The R Graph Gallery – Help and Inspiration for R Charts ». The R Graph Gallery. Consulté le 23 août 2022. <https://r-graph-gallery.com/index.html>.
- *« Home | Bookdown ». Consulté le 5 août 2022. <https://bookdown.org/>.
- « Installing and Using R Packages - Easy Guides - Wiki - STHDA ». Consulté le 4 août 2022. <http://www.sthda.com/english/wiki/installing-and-using-r-packages>.
- « Introduction ». Consulté le 23 août 2022. <https://rmarkdown.rstudio.com/lesson-1.html>.
- « Introduction to R Markdown ». Consulté le 4 août 2022.
https://rmarkdown.rstudio.com/articles_intro.html.
- « kableExtra ». Consulté le 23 août 2022. <https://haozhu233.github.io/kableExtra/>.
- *« MTECT-MTE / Formations à R ». Consulté le 13 juillet 2022. <https://mtes-mct.github.io/parcours-r/>.
- Pedersen, Hadley Wickham, Danielle Navarro, and Thomas Lin. *Welcome | Ggplot2*. Consulté le 5 août 2022. <https://ggplot2-book.org/>.
- « Project Environments ». Consulté le 4 août 2022. <https://rstudio.github.io/renv/>.
- publics, Une documentation sur R. à l'usage des statisticiens. *Chapitre 9 Utiliser Des Packages R | UtilitR*. Consulté le 4 août 2022.
<https://www.book.utilitr.org/utiliser-packages.html>.
- « R Packages (2e) ». Consulté le 4 août 2022. <https://r-pkgs.org/>.
- « Renv: Project Environments for R », 6 novembre 2019.
<https://www.rstudio.com/blog/renv-project-environments-for-r/>.

- *Riederer, Yihui Xie, Christophe Dervieux, Emily. *R Markdown Cookbook*. Consulté le 21 juillet 2022. <https://bookdown.org/yihui/rmarkdown-cookbook/>.
- « RStudio Cheatsheets ». Consulté le 25 juillet 2022. <https://www.rstudio.com/resources/cheatsheets/>.
- Schoonemann, Jeanine. « How to Remove All User Installed Packages in R | R-Bloggers », 21 octobre 2016. <https://www.r-bloggers.com/2016/10/how-to-remove-all-user-installed-packages-in-r/>.
- « tableGrob · baptiste/gridExtra Wiki ». Consulté le 14 juillet 2022. <https://github.com/baptiste/gridExtra/wiki/tableGrob>.
- « Tidyverse ». Consulté le 14 juillet 2022. <https://www.tidyverse.org/>.
- « TinyTeX - Yihui Xie | 谢益辉 ». Consulté le 23 août 2022. <https://yihui.org/tinytex/>.
- Tsukahara, Jason. *6 Data Manipulation using dplyr | EngleLab: useRguide*. Consulté le 6 août 2022. <https://englelab.gatech.edu/useRguide/data-manipulation-using-dplyr.html#mutate>.
- « Welcome! » Consulté le 23 août 2022. <https://r-unimelb.gitbook.io/rbook/>.
- *« Welcome | R for Data Science ». Consulté le 23 août 2022. <https://r4ds.had.co.nz/index.html>.
- « Workflow ». Consulté le 23 août 2022. <https://cran.r-project.org/web/packages/renv/vignettes/renv.html>.
- Zhu, Hao. « Create Awesome LaTeX Table with Knitr::Kable and KableExtra », s. d., 28.