**Création du package mbtv2020**

# Base du package

Etape 1 : Vérification du nom

Pour savoir sile nom du package n’est pas emprunté, dans un script ou la console : available::available(«mbtv2020»)

Etape 2 : Création du package

Dans un script ou dans la console :

usethis::create\_package(“file/mbtv2020”)

Etape 3 : Modification du fichier DESCRIPTION

Modifiez à la main les paramètres Title, Authors@R et Description (attention la description doit se terminer par un point), lorsqu’il y a plusieurs auteurs, on les mets dans un vecteur.

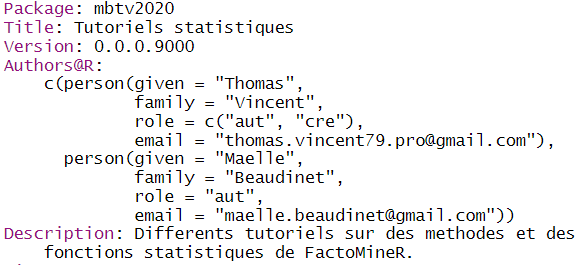


Illustration  : Fichier description après modification des champs de l’étape 3

Etape 4 : Création du script porteur du code

Ouverture d’un script vierge, qu’on sauvegarde à la racine du package : dev\_history.r

Ajoutez ce fichier a Rbuildignore, dans ce même fichier : usethis::use\_build\_ignore(«dev\_history.r»)

Etape 5 : Ajout d’une licence

Dans dev\_history.r, ajoutez une licence : usethis ::use\_gpl3\_license(« Nom Prénom »)

Etape 6 : Gestion des importations

Le package va utiliser différentes fonctions de packages, pour cela il faut les importer dans un fichier utilisé pour créer les fonction.

.1 Créez un fichier R dans lequel on va mettre toutes les importations : usethis::use\_r(«importtuto»)

.2 : Créez une forme de fonction test<-function(){} dans le fichier importtuto

.3 : Placez le curseur entre les accolades et ouvrir le skeleton ROxygen : shift+alt+ctrl+R

.4 : Rajoutez les imports sur le même schéma que le reste : @import(Nom\_package), un import par ligne. Nous avons besoin dans ce package d’importer la liste de package suivante : learnr, FactoMineR, SensoMineR, sqldf, factoextra, shiny, ggplot2, ade4, missMDA, dplyr, magritrr.

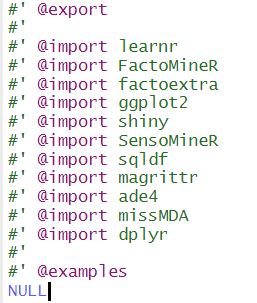
.5 : Remplacez la structure de fonction par NULL

Illustration  : Fonction des importations

Etape **ACTUALISATION** :

Lancez ces 4 fonctions dans le dev\_history.r :

* devtools::load\_all(),
* devtools::document(),
* usethis::use\_tidy\_description()
* attachment::att\_to\_description()

*Remarque : Contrôlez que tout s’est bien passé, vérification que les imports/exports ont bien été incorporés dans les fichiers NAMESPACE et DESCRIPTION.*

# Tutoriels et données

Etape 7: Création d’un tutoriel

Création des tutoriels learnr.

Dans dev\_history.r l’un après l’autre :

usethis::use\_tutorial(name="Tuto\_ACM",title="ACM avec FactoMineR")

usethis::use\_tutorial(name="Tuto\_ACP",title="ACP avec FactoMineR")

usethis::use\_tutorial(name="Tuto\_panellipse",title="Tuto\_panellipse")

usethis::use\_tutorial(name="Tuto\_decat",title="Tuto\_decat")

usethis::use\_tutorial(name="Tuto\_AFM",title="ACM avec FactoMineR")

usethis::use\_tutorial(name="MCA",title="MCA with FactoMineR")

usethis::use\_tutorial(name="MFA",title="MFA with FactoMineR")

Etape 8: Remplissage des tutoriels

Copiez-collez dans les tutoriels correspondants l’ensemble du .Rmd correspondant

Etape 9: Importation de données

Dans dev\_history.r, l’un après l’autre :

use\_data\_raw('sensochoc')

use\_data\_raw('perfumes\_qda\_experts')

use\_data\_raw('tea')

use\_data\_raw('decathlon')

use\_data\_raw('wines')

use\_data\_raw('lipstick')

9.1 : Dans le répertoire inst, création d’un répertoire « datasets ».

9.2 : Copie de l’ensemble des fichiers .csv dans ce répertoire « datasets ».

9.3 : Dataset sensochoc, copiez-collez puis exécutez dans « sensochoc.r » :

sensochoc<-read.csv("inst/datasets/sensochoc.csv",header=TRUE,sep=";",dec=",")

sensochoc$Product<-as.factor(sensochoc$Product)

sensochoc$Panelist<-as.factor(sensochoc$Panelist)

sensochoc$Rank<-as.factor(sensochoc$Rank)

sensochoc$Session<-as.factor(sensochoc$Session)

sensochoc<-sensochoc[,c(1,2,4,3,5,6,7,8,9,10,11,12,13,14,15,16,17,18)]

usethis::use\_data(sensochoc, overwrite = TRUE)

9.4 : Dataset perfumes\_qda\_experts, copiez-collez puis exécutez :

perfumes\_qda\_experts<-read.csv("inst/datasets/perfumes\_qda\_experts.csv",sep = ",",header=TRUE)

usethis::use\_data(perfumes\_qda\_experts, overwrite = TRUE)

9.5 : Dataset tea, copiez-collez puis exécutez dans « tea.r » :

tea<-read.csv("inst/datasets/tea.csv",sep = ";",header=TRUE

usethis::use\_data(tea, overwrite = TRUE)

9.6 : Dataset decathlon, copiez-collez puis exécutez dans « decathlon.r » :

decathlon<-read.csv("inst/datasets/decathlon.csv",sep=";",header=TRUE)

usethis::use\_data(decathlon, overwrite = TRUE)

9.7 : Dataset wine, copiez-collez puis exécutez dans « wine.r » :

wines<-read.csv("inst/datasets/wines.csv",sep = ";",header=TRUE, dec =",")

rownames(wines)<-wines[,1]

wines<-wines[,-c(1)])

usethis::use\_data(wines, overwrite = TRUE)

9.8 : Dataset lipstick, copiez-collez puis exécutez dans « lipstick.r » :

lipstick<-read.csv("inst/datasets/lipsticks.csv",sep = ",",header=TRUE)

usethis::use\_data(lipstick, overwrite = TRUE)

Les données sont maintenant utilisables dans les tutoriels du package.

Etape 10: Documentation des données

Dans dev\_history.r usethis::use\_r(« nom\_dataset »). La niveau de détail de la documentation vous appartient, vous pouvez résumer le dataset en une phrase ou bien déclarer pour chaque variables son type et à quoi elle correspond. Ici on fait le minimum, pour simplement éviter les warnings lors du check().

use\_r("data")

use\_r("tea")

use\_r("perfumes")

use\_r("decathlon")

use\_r(“wines”)

use\_r('lipstick')

10.1 : Pour les datasets sensochoc, lipstick, perfumes\_qda\_experts, decathlon, tea, recopiez la structure suivante en remplaçant lipstick par le nom du dataframe associé au .r :

#' lipstick

#'

#' @'format Data frame:

#' \describe{

#'}

"lipstick"

Etape **ACTUALISATION** : Lancement de 4 fonctions dans le dev\_history.r :

* devtools ::load\_all(),
* devtools ::document(),
* devtools ::use\_tidy\_description()
* attachment ::att\_to\_description()

Etape 12: Activation des tutoriels

Par défaut, vos tutoriels sont rajoutés dans le fichier .Rbuildignore. Supprimez ces lignes « ^inst/tutorials/MFA/MFA\.Rmd$ » pour le tutoriel MFA.

# GitHub

Etape 12: Commit initial

Dans dev\_history.r usethis::use\_git(), acceptez les demandes.

Etape 12: Création du repository GitHub

Pour cela aller sur la page : <https://github.com/new>, remplissez seulement la case spécifiant le nom « mbtv2020 ». Puis cliquer sur « Create repository »

Etape 13: Branch master et premier push

Copiez-collez ou recopiez ces liens dans le terminal R (changez le nom du package et de l’utilisateur)

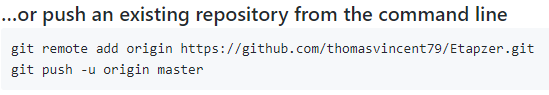


Illustration 2 : Lien pour push votre package

*Remarque : Pour les futurs push, il vous faudra d’abord commit vos fichiers en les sélectionnant puis il suffira de cliquer sur la flèche verte pointant vers le haut dans l’onglet Git en haut à droite. Pour ne pas push à chaque modifications mineures mais que vous souhaitais sauvegarder l’avancement, vous pouvez simplement commit.*

Etape 14: Ajoute d’un README

Le README est la façade de votre package sur GitHub. Dans le dev\_history.r : usethis::use\_readme\_rmd()

Copiez-collez le contenu du fichier README.rmd dans celui-ci.

Après avoir modifié votre README.rmd comme vous le souhaitez, knitez-le, afin de générer le README.md.

*Remarque : Les modifications du README se font toujours dans le .rmd, le .md se génère lors du knit de ce dernier.*

Etape 15: Ajout de l’intégration continue.

Dans le dev\_history.r : usethis ::use\_github\_action\_check\_release()

Etape 16: Gestion des liens GitHub.

Dans le dev\_history.r : usethis ::use\_github\_links()

Etape 17: Gestion des contributeurs GitHub.

Pour ajouter des collaborateurs à votre repository : allez dans l’onglet settings puis Manage access puis cliquez sur « invite collaborator »

Etape **ACTUALISATION** : Lancement de 4 fonctions dans le dev\_history.r :

* devtools ::load\_all(),
* devtools ::document(),
* devtools ::use\_tidy\_description()
* attachment ::att\_to\_description()

**Le package mbtv2020 est prêt.**