

Traitement des données PMSI avec R

Guillaume Pressiat // SIMAP / DOMU / Assistance Publique - Hôpitaux de Paris

2017-03-24

Tables des matières

1	Introduction	5
2	Contexte et motivations	7
2.1	Avantages de R	7
2.2	Contenu du package	8
2.3	Installation du package	8
3	Les archives PMSI	9
3.1	Arborescence des archives	9
3.2	Informations sur les archives	10
3.3	Dézippage	11
3.4	Suppression	11
4	Import des données	15
4.1	MCO	15
4.2	HAD	17
4.3	SSR	19
4.4	PSY	19
4.5	RSF	19
4.6	Dictionnaire de variables	20
4.7	Labels	20
5	Requêtes sur des pathologies / actes	21
5.1	Transposition des codes diagnostics	21
5.2	Recherche de codes diagnostics	21
5.3	Recherche de codes actes	22
6	Étude des files actives	23
6.1	Import des données Anohosp	23
6.2	File active d'une pathologie	23
6.3	File active d'une chirurgie	23
7	Fichier TRA	25
7.1	Ajout du TRA en MCO	25
7.2	Ajout du TRA en HAD	26
7.3	Ajout du TRA en SSR	26
7.4	Ajout du TRA en PSY	26
8	Statistiques du PMSI	29
8.1	Âge et durée de séjour	29
8.2	Nombre de séjours par catégorie majeure de diagnostics	29
8.3	Case-mix MCO, DMS par GHM / GHS	29

1

Introduction

Ce livret numérique présente des exemples de traitements de données PMSI avec R. L'objectif est de concentrer ici :

- une documentation permettant de débiter avec l'import de données via le package *pmeasyr*
- des exemples d'analyses PMSI :
 - requêtes sur les diagnostics et les actes
 - analyse des files actives pour une pathologie
 - statistiques élémentaires sur des variables du PMSI
 - analyse du case-mix et de la dms par ghm

2

Contexte et motivations

Les données du Programme de Médicalisation des Systèmes d'Information (PMSI) sont souvent traitées via des logiciels spécifiques au PMSI (ou des outils statistiques / bases de données du marché) ne permettant pas de réaliser des traitements statistiques et des infographies satisfaisantes. Les départements d'information médicale sont donc souvent amenés à retraiter ces données avec R.

L'évolution récente de R intègre la manipulation de bases de données de taille importante. Le package *pmeasyr* s'inscrit dans cette veine et permet de réaliser de façon autonome l'ensemble des traitements (de l'import des données à leur analyse) avec R.

2.1 Avantages de R

2.1.1 Un flux de travail unique

En travaillant uniquement avec R, on peut mettre en place un flux de travail épuré : un seul projet, un seul programme, un seul logiciel. La traçabilité, la reproductibilité et la mise à jour des opérations sont ainsi facilitées.

Le travail avec de multiples logiciels oblige à l'export / import de fichiers entre les différents logiciels, et chaque modification du début du flux de travail génère des fichiers exportés v1, v2, ...

Avec un flux complet dans R, toute nouvelle modification est intégrée au processus de travail global. La localisation de toutes les étapes d'une analyse en un seul point évite les erreurs et la confusion lorsque l'on reprend l'analyse ultérieurement.

2.1.2 R et le PMSI

L'utilisation de R confère aux données du PMSI la liberté proposée par le logiciel :

- les requêtes sur les diagnostics et les actes peuvent s'écrire de multiples façons et c'est l'utilisateur qui crée ses propres programmes
- les données sont dans R : prêtes pour des modèles linéaires, logistiques, des classifications...
- la confrontation des données in* (reflet du codage des établissements) aux données out* (reflet de la valorisation accordée à l'établissement) est facilitée par l'import du fichier tra, cela peut permettre aux équipes DIM d'améliorer leur recueil
- le reporting de l'activité en excel, pdf, word, html, ou en créant des applications ([shiny](#))
- l'utilisation des graphiques pour représenter des volumes d'activités et des cartographies interactives pour visualiser la localisation d'activités, de patientèles, et les flux de patients
- le partage de projets RStudio, qui facilite et encourage les travaux en équipe.

NB: Données In / Out : données en entrée / sortie des logiciels de l'ATIH

2.1.3 Des outils performants

L'engouement autour de R est lié au développement de packages intuitifs et performants : [readr](#), [dplyr](#), [tidyr](#), [magrittr](#), pour n'en citer que quelques-uns. [pmeasyr](#) s'appuie sur ces packages pour proposer des imports de données rapides sur des fichiers de taille importante (l'entité juridique de l'AP-HP est prise en charge sans problème avec un ordinateur récent).

Dans le cas de [pmeasyr](#), l'import de 100 000 rsa (partie fixe, parsing des passages unités médicales, des diagnostics associés et des actes) nécessite en moyenne 5 secondes avec un processeur i7 – 16Go de ram.

En dernier ressort, R travaillant en mémoire vive, les exécutions de requêtes sont très rapides.

2.2 Contenu du package

Le package contient des fonctions pour la gestion des archives PMSI en entrée / sortie des logiciels de l'ATIH : dézippage, suppression des archives, et des fonctions pour l'import des fichiers des champs PMSI MCO, SSR, HAD, PSY et RSF.

Il est utilisé depuis un an à l'AP-HP pour des analyses d'activité et la description des prises en charge.

2.3 Installation du package

```
devtools::install_github('IM-APHP/pmeasyr')
```

Cette commande lance l'installation du package et de ses dépendances.

3

Les archives PMSI

Cette partie aborde le point de départ des études PMSI : les archives PMSI. Ces archives sont les fichiers en entrées / sorties des logiciels de l'ATIH.

Les manuels techniques de ces logiciels, relatifs aux champs MCO, SSR, HAD, PSY et RSF, respectivement [Genrsa](#), [Genrha](#), [Paprica](#), [Pivoine](#) et [Preface](#) sont disponibles dans l'[espace de téléchargement](#) sur le site de l'ATIH.

3.1 Arborescence des archives

Le package *pmeasyr* prend en charge les données des quatre champs PMSI MCO, SSR, HAD, PSY ainsi que les RSF.

Placer les archives dans un répertoire, par exemple ici dans `~/Documents/data/mco` :

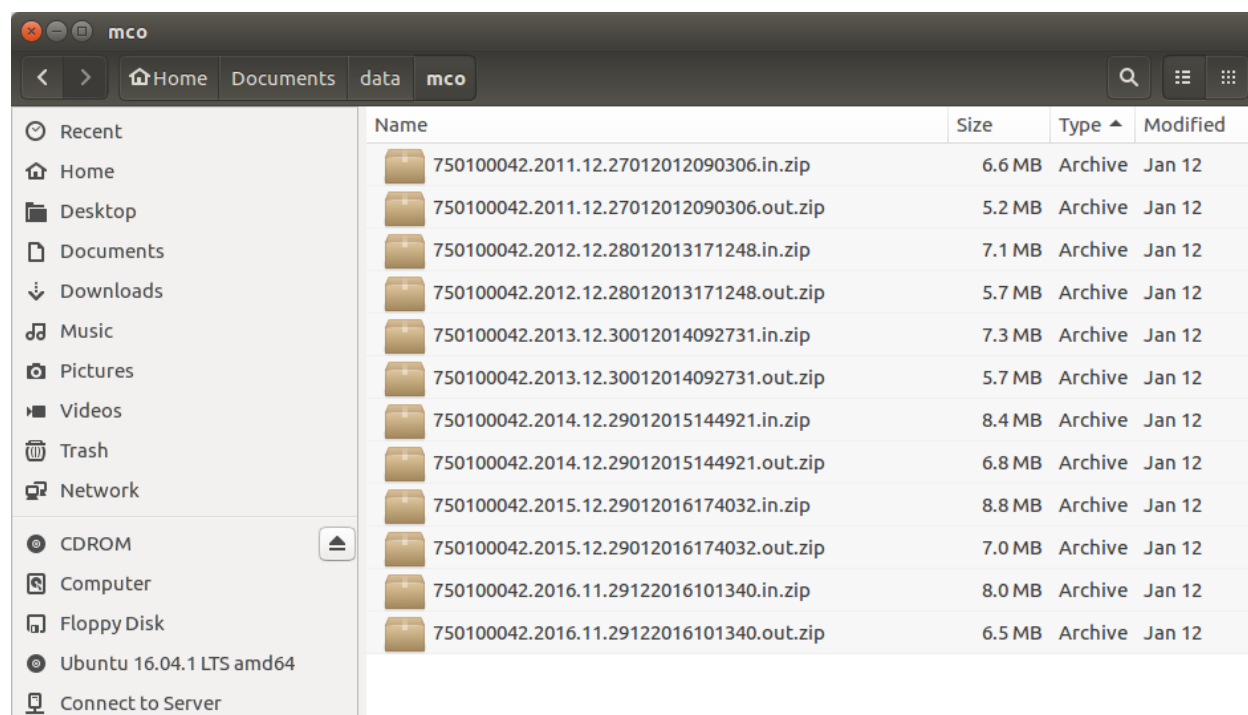


Figure 3.1: Archives MCO

Vous noterez que pour chaque champ PMSI il est conseillé d'utiliser un répertoire indépendant, ceci est nécessaire dans la mesure où le nom des archives PMSI ne contient pas l'information champ MCO, RSF, etc., il faut organiser l'archivage champ par champ, dans des répertoires différents.

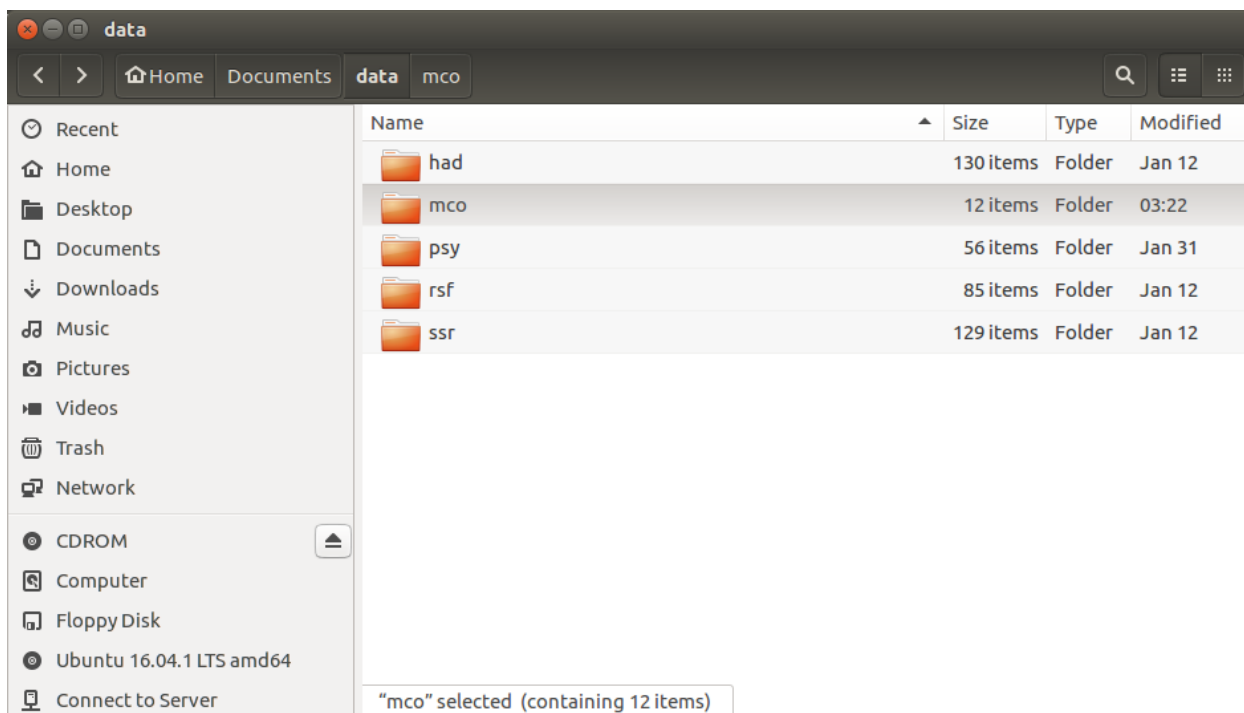


Figure 3.2: Un répertoire par champ PMSI

```
# Créer l'arborescence à partir de R
champs = c('mco', 'ssr', 'had', 'psy', 'rsf')
emplacement <- "~/Documents/data"
sapply(champs, function(x){dir.create(file.path(emplacement, x))})
```

3.2 Informations sur les archives

Le nom des fonctions dont l'objectif est de manipuler les archives commence par **a**.

La fonction `astat` permet d'éditer des statistiques sommaires sur les fichiers contenus dans une archive.

Nom	Fonction
<code>astat</code>	~ *.zip - Liste et volume des fichiers d'une archive PMSI

```
# Informations sur les fichiers : Date de creation, Taille
pmeasyr::astat(path = '~/Documents/data/mco/',
               file = '750100042.2015.12.29012016174032.out.zip',
               view = F)
```

3.3 Dézippage

Cette partie du package facilite la manipulation des archives PMSI, fichiers de type :

- `finess.annee.mois.date_et_heure_de_creation.in.zip`
- `finess.annee.mois.date_et_heure_de_creation.out.zip`

Les fonctions permettent de dézipper les fichiers depuis R en ligne de commande, sans intervention manuelle de l'utilisateur. L'avantage est d'obtenir un processus ne relevant pas d'interventions externes au logiciel R (pour pouvoir garder trace des étapes, et faciliter la reproduction, tout est inscrit dans un programme, dans un flux de processus). Une fois que les traitements et analyses sur les fichiers sont faits, il est possible d'effacer les archives également en ligne de commande.

Nom	Fonction
<code>adezip</code>	~ *.zip - Dézippe des fichiers de l'archive PMSI
<code>adezip2</code>	~ *.zip - Dézippe des fichiers de l'archive PMSI, avec en paramètre le nom de l'archive

```
# Dézippage uniquement des fichiers rsa, ano et tra du out 2015
# Ex: 750100042.2015.12.20160130.153012.out.zip
pmeasyr::adezip(finess = 750100042,
  annee = 2015,
  mois = 12,
  path = '~/Documents/data/mco',
  liste = c("rsa", "ano", "tra"),
  type = "out")
```

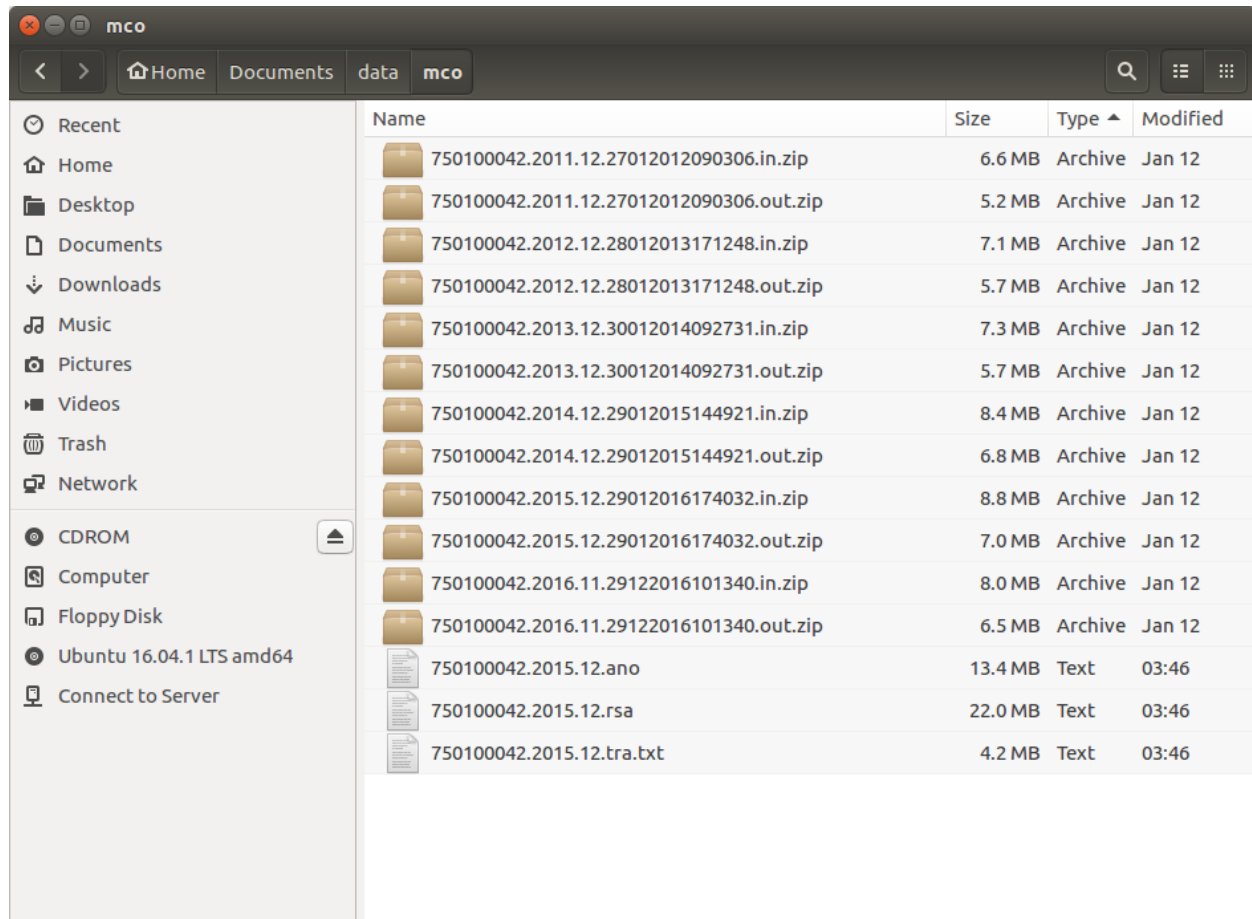
```
# Dézippage uniquement des fichiers rss, dmi et med du in 2015
# Ex: 750100042.2015.12.20160130.153012.out.zip
pmeasyr::adezip(finess = 750100042,
  annee = 2015,
  mois = 12,
  path = '~/Documents/data/mco',
  liste = c("rss", "dmi", "med"),
  type = "in")
```

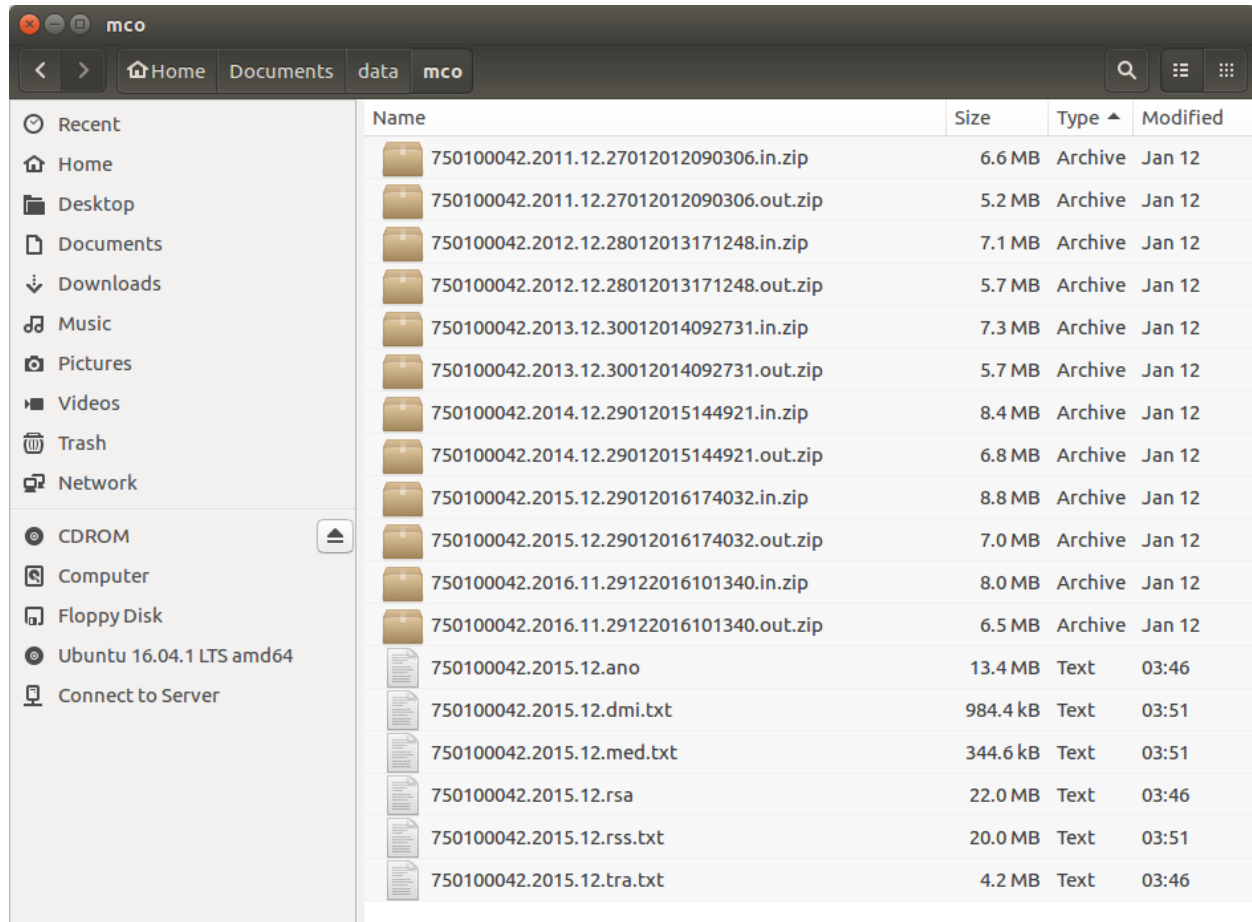
3.4 Suppression

À la fin d'une étude, il est inutile de garder les fichiers dézippés hors de l'archive, on peut les effacer : c'est ce que permet la fonction `adelete()`.

```
# Effacer les fichiers
pmeasyr::adelete(finess = 750100042,
  annee = 2015,
  mois = 12,
  path = '~/Documents/data/mco',
  liste = c("rsa", "ano", "tra"),
  type = "out")

pmeasyr::adelete(finess = 750100042,
  annee = 2015,
  mois = 12,
  path = '~/Documents/data/mco',
```

Figure 3.3: Après exécution de `adezip()` sur des fichiers du `out`

Figure 3.4: Après exécution de `adezip()` sur des fichiers du *in*

```
liste = c("rss", "med", "dmi"),
type = "in")
```

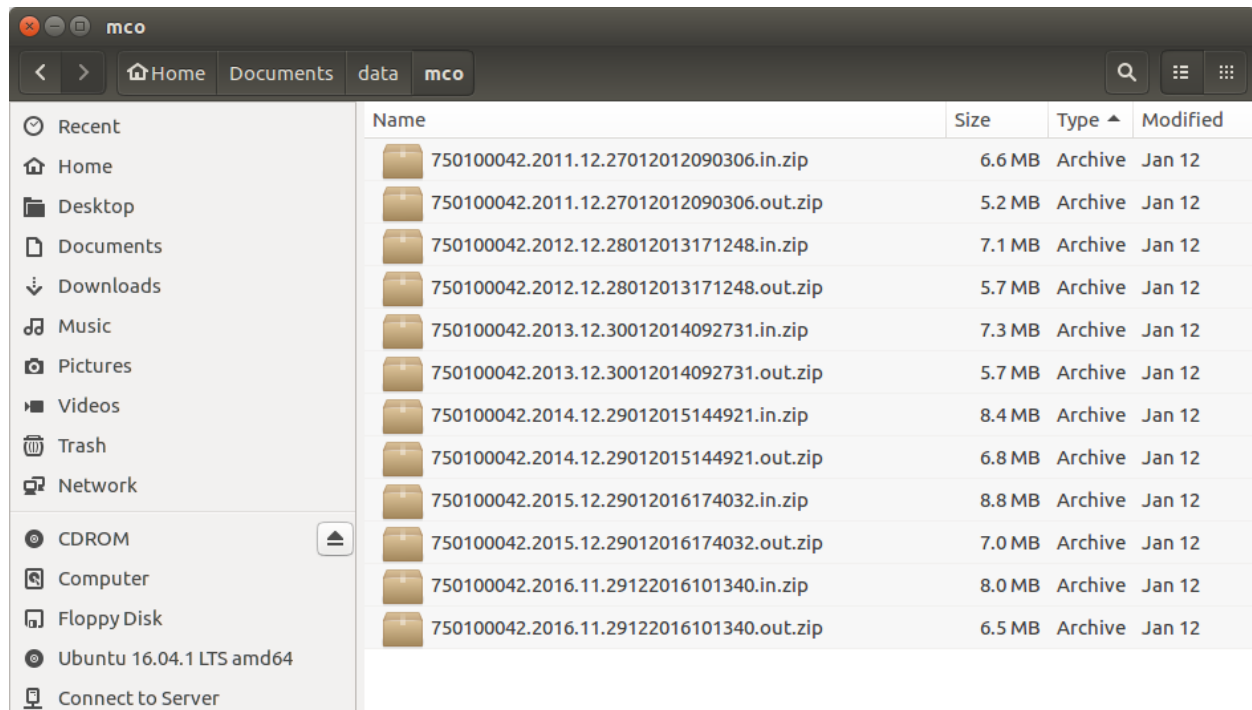


Figure 3.5: Après exécution de `delete()`

4

Import des données

4.1 MCO

Nom	Fonction
irsa	~ MCO - Import des RSA
irum	~ MCO - Import des RUM
idiap	~ MCO - Import des DIAP
idmi_mco	~ MCO - Import des DMI
iium	~ MCO - Import des donnees UM
ileg_mco	~ MCO - Import des erreurs Leg
imed_mco	~ MCO - Import des Med
ipo	~ MCO - Import des PO
iano_mco	~ MCO - Import des Anohosp

Les données in / out sont prises en charge.

4.1.1 RSA

Selon la nature des analyses à produire, plusieurs types d'imports sont possibles :

Type	Import
1	Light : Partie fixe
2	Light+ : Partie fixe + stream en ligne (+) actes et das
3	Light++ : Partie fixe + stream en ligne (++) actes, das, typaut um et dpdr des um
4	Standard : Partie fixe + creation des tables actes, das et rsa_um
5	Standard+ : Partie fixe + creation des tables actes, das et rsa_um + stream (+)
6	Standard++ : Partie fixe + creation des tables actes, das et rsa_um + stream (++)

```
library(pmeasyr)
# Import des rsa 2015 type 6
irsa(finess = 750100042,
     annee = 2015,
     mois = 12,
```

```

path = '~/Documents/data/mco',
typi = 6) -> rsa15
View(rsa15$rsa)
View(rsa15$rsa_um)
View(rsa15$actes)
View(rsa15$das)

```

Les tables sont par défaut avec des libellés :

NOFINESS Finess de l'établissement	NOVRSA Numéro de version du RSA	CLE_RSA Clé RSA	NOVRSS Numéro de version du RSS groupé	NOSEQTA Numéro séquentiel des tarifs
750100042	222	0000000001	116	002
750100042	222	0000000002	116	002
750100042	222	0000000003	116	002
750100042	222	0000000004	116	002
750100042	222	0000000005	116	002
750100042	222	0000000006	116	002
750100042	222	0000000007	116	002
750100042	222	0000000008	116	002
750100042	222	0000000009	116	002
750100042	222	0000000010	116	002
750100042	222	0000000011	116	002
750100042	222	0000000012	116	002
750100042	222	0000000013	116	002
750100042	222	0000000014	116	002
750100042	222	0000000015	116	002
750100042	222	0000000016	116	002
750100042	222	0000000017	116	002

Figure 4.1: Capture d'une portion de la table `rsa15$rsa`

4.1.2 RUM

```

# Import des rum 2015
irum(finess = 750100042,
     annee = 2015,
     mois = 12,
     path = '~/Documents/data/mco')

```

Selon la nature des analyses à produire, plusieurs types d'imports sont possibles :

Type	Import
1	XLight : Partie fixe
2	Light : Partie fixe + stream en ligne des actes, das et dad
3	Standard : Partie fixe + table actes, das, dad
4	Standard+ : Partie fixe + stream + table actes, das, dad

4.1.3 Colonne stream

Exemples sur quelques rsa :

- actes : Actes CCAM du Rsa

Cle RSA	actes
0000000001	EDSF004, EDSF004, JQGA004, JQGA004
0000000002	EPLF002, DEQP003, DEQP007, DZQM006
0000000003	EBQH002, EEQH002, YYYYY180

- dpdrum : zones diagnostics des passages UM du Rsa

Cle RSA	dpdrum
0000000004	Z098 I671
0000000005	Z380, P741, Z380

- das : zones diagnostics associes du Rsa

Cle RSA	das
0000000006	Z9580, Z9588
0000000007	P011, P032, P036, P011, P032, P700, P011, P032, P036

- um : types autorisations T2A des um de passage par ordre chronologique

Cle RSA	um
0000000009	01AC, 53 C
0000000010	51 C
0000000011	71 C, 04 C, 71 C

Pour les quatre autres champs PMSI, seules les données du *out* sont prises en charge par le package pour le moment.

Les fonctions d'imports pour ces champs PMSI reposent sur le même principe qu'en MCO.

4.2 HAD

Nom	Fonction
<code>iano_had</code>	~ HAD - Import des Anohosp
<code>imed_had</code>	~ HAD - Import des Med
<code>irapss</code>	~ HAD - Import des RAPSS
<code>ileg_had</code>	~ HAD - Import des erreurs LEG

```
library(pmeasyr)
# Import des rapss 2015
irapss(finess = 750712184,
```

actes Stream Actes	um Parcours Typaut UM	dpdrum Stream DP/DR des UM	das Stream Das
JLD001, DEQP003	07AC	N179	
DEQP003, DEQP003, DEQP003	07AC	I480	
	07AC	F100	
	07AC	T406	
YYYY600, ACQJ002	07AC	R410	B230, B582
DEQP003	07AC	T509	F603
YYYY009, DEQP007, ZZEP004	04 C	P599	
LFQK002, NEQK010, NAQK023	07AC	M796	
EAQH002, DEQP003, ZBQK001, YYYY600	07AC	J449	Z030, I618
BGQP003, BLQP010, DEQP003, BJQP005, EQQP008, ...	18 C	G459	
	07AC	F119	F209
DEQP003, JAQM003	07AC	N179	
DEQP003	07AC	I509	
DEQP003, NAQK023	07AC	R53+0	F023, G20, I480
DEQP003, LFQK002, LEQK001	07AC	I501	
	07AC	J189	J458
DEQP003, AFHB002, YYYY600, EBQH004	07AC	A879	
DEQP003, DEQP003	07AC	J209	
QAGA003, ZZLP054, ZZLP054, ZZQX180	53 P	M7958	
	29 P	Z502	
EEAF003, EEAF003, EEAF003, EEAF003, YYYY200, N...	29 C	I7021	E1140, E1150, Z713, Z718, L039, L030, Z740, E440...
FCFA008, FCFA008, GDFA011, GDFA011, ZZQX175	53 C	C328	I10, Z921, J384, T814, B956
ZCQM001, JAQM003	29 C	E1120	Z713, N083, E559
NDQK002, DEQP003	29 C	R02	R2630, E1170, Z713

Figure 4.2: Capture des zones *stream* de la table *rsa15\$rsa*

```
annee = 2015,
mois = 12,
path = '~/Documents/data/had') -> data_had
```

4.3 SSR

Nom	Fonction
<code>iano_ssr</code>	~ SSR - Import des Anohosp
<code>irha</code>	~ SSR - Import des RHA
<code>issrha</code>	~ SSR - Import des SSRHA
<code>imed_ssr</code>	~ SSR - Import des MED
<code>iium_ssr</code>	~ SSR - Import des UM
<code>ileg_ssr</code>	~ SSR - Import des erreurs LEG

```
# Import des rha 2015
irha(finess = 750041543,
annee = 2015,
mois = 12,
path = '~/Documents/data/ssr') -> data_ssr
```

4.4 PSY

Nom	Fonction
<code>iano_psy</code>	~ PSY - Import des Anohosp
<code>ir3a</code>	~ PSY - Import des R3A
<code>irpsa</code>	~ PSY - Import des RPSA

```
# Import des rpsa 2015
irpsa(finess = 750803454,
annee = 2015,
mois = 12,
path = '~/Documents/data/psy') -> rpsa_psy

# Import des r3a 2015
ir3a(finess = 750803454,
annee = 2015,
mois = 12,
path = '~/Documents/data/psy') -> r3a_psy
```

4.5 RSF

Nom	Fonction
<code>irafael</code>	~ RSF - Import des RSFA / Rafael

Nom	Fonction
iano_rafael	~ RSF - Import des RSFA / ANO

```
# Import des rsfa 2015
irafael(finess = 750712184,
        annee = 2015,
        mois = 12,
        path = '~/Documents/data/rsf') -> rsfa
```

4.6 Dictionnaire de variables

```
# Obtenir les noms, labels et types de variables (character, numeric, integer, date, ...)
dico(rsa15$rsa)
```

```
# Charger les formats de toutes les tables prises en charge par le package
pmeasyr::formats
```

4.7 Labels

```
# Obtenir le libelle d'une variable du PMSI
labeleasier(rsa15$rsa$SEXE, Sexe = T)
labeleasier(rsa15$rsa$ECHPMSI, Mode_entree = T)
```

5

Requêtes sur des pathologies / actes

5.1 Transposition des codes diagnostics

Les analyses sur les diagnostics CIM-10 sont parfois fastidieuses du fait des multiples positions de diagnostics : DP principal du séjour, DR principal du séjour, DPUM, DRUM, DAS. La fonction *tdiag* permet de rassembler tous les diagnostics dans une seule table.

```
# Pour les objets rsa et rum du MCO
# Transbahuter tous les diagnostics dans une seule table
tdiag(rsa15) -> rsa15 # "Tidy diagnostics"
View(rsa15$diags)
# Tous les diagnostics sont dans une table, avec un numero selon leur position
# 1:DP, 2:DR, 3:DPUM, 4:DRUM, 5:DAS
```

Exemple de résultat :

CLE_RSA	NSEQRUM	position	diag
0000000001	01	1	Z511
0000000001	01	2	C18
0000000002	01	1	C501
0000000002	01	3	C501
0000000002	02	1	D051
0000000002	02	5	E109

5.2 Recherche de codes diagnostics

L'objectif est de récupérer les séjours présentant un code diagnostic de la liste

```
# Liste D-0103 de la fonction groupage 2016 : Epilepsies
liste_diag = c('F803', 'G400', 'G401', 'G402', 'G403', 'G404',
               'G405', 'G406', 'G407', 'G408', 'G409', 'G410',
               'G411', 'G412', 'G418', 'G419', 'R568')

# En passant par la table diags
tdiag(rsa15) -> rsa15
```

```

library(dplyr)
# quelle que soit la position du diagnostic
rsa15$diags %>% filter(diag %in% liste_diag)
# position en das
rsa15$diags %>% filter(diag %in% liste_diag, position == 5)
# position en dp dr
rsa15$diags %>% filter(diag %in% liste_diag, position < 5)

# En passant par les zones stream
string_diags =
  'F803|G400|G401|G402|G403|G404|G405|G406|G407|G408|G409|G410|G411|G412|418|G419|R568'

# quelle que soit la position du diagnostic
rsa15$rsa %>% fiter(grepl(string_diags, dpdrum)|grepl(string_diags, das))
# position en das
rsa15$rsa %>% fiter(grepl(string_diags, das))
# position en dpdr
rsa15$rsa %>% fiter(grepl(string_diags, dpdrum))

```

5.3 Recherche de codes actes

```

# Code EBLA003

library(dplyr)
# En passant par la table actes
rsa15$actes %>% filter(CDCCAM == 'EBLA003')
# En passant par la zone stream
rsa15$rsa %>% filter(grepl('EBLA003', actes))

```

6

Étude des files actives

6.1 Import des données Anohosp

```
# Import des données anohosp du out
iano_mco(finess = 750100042,
  annee = 2015,
  mois = 12,
  path = '~/Documents/data/mco') -> ano

# Filtrer sur les patients chainables avec la variable cok
library(dplyr)
ano %>% filter(cok) -> ano

# File active globale établissement
distinct(ano, NOANON) %>% nrow()
```

6.2 File active d'une pathologie

```
# Codes diagnostics obésité
string_diags = 'E66'

library(dplyr)
# position en dpdr
rsa15$rsa %>% filter(grepl(string_diags, dpdr)) -> ob

# File active obésité globale établissement
inner_join(ano, ob, by = c('CLE_RSA')) -> patients_ob
distinct(patients_ob, NOANON) %>% nrow()
```

6.3 File active d'une chirurgie

```
# Codes actes chirurgie bariatrique
liste_actes = c('HFCA001', 'HFCC003', 'HFCC004', 'HFFA001', 'HFMA009',
```

```
      'HFMC007', 'HFKA001', 'HFKC001', 'HFKA002', 'HFMA011',  
      'HFMC008', 'HFMA010', 'HFMC006', 'HFLE002', 'HFGC900',  
      'HFLC900', 'HFFA011', 'HFFC018', 'HGCA009', 'HGCC027')  
  
library(dplyr)  
# acte codé activité 1 (chirurgical)  
rsa15$actes %>% filter(CDCCAM %in% liste_actes, ACT == '1') -> cob  
  
# File active chirurgie bariatrique globale établissement  
inner_join(ano, cob, by = c('CLE_RSA')) -> patients_cob  
distinct(patients_cob, NOANON) %>% nrow()
```


7

Fichier TRA

Le fichier TRA est un fichier du *out* qui permet de relier les données anonymes du *out* aux données du *in*, il comprend un lien entre :

- MCO : clé rsa, numéro de rss, numéro de séjour (nas), date d'entrée et date de sortie du séjour
- SSR : numéro séquentiel du séjour + noseqrhs et numéro de séjour + numéro de semaine
- PSY RPSA : ipp, date d'entrée et de fin du séjour, numéro séquentiel du séjour, numéro de séquence et numéro de séjour, dates de début et fin de séquence
- PSY R3A : ipp, date de l'acte, numéro d'ordre, forme activité, um, nature et lieu de l'acte
- HAD : numéro séquentiel de séjour, numéro de séquence, sous-séquence et numéro de séjour, dates de début et fin des séquences et sous-séquences, dates d'entrée et de sortie du séjour, modes d'entrée sortie provenance destination

Type	Import
<code>itra</code>	~ TRA - Import du TRA
<code>inner_tra</code>	~ TRA - Ajout du TRA aux données Out

7.1 Ajout du TRA en MCO

```
# lecture du fichier tra et jointure aux rsa
itra(750100042, 2015, 12, '~/Documents/data/mco') -> tra

# Ajout du tra aux rsa :
inner_tra(rsa15$rsa, tra) -> rsa15$rsa
# Ajout du tra à la partie um des rsa :
inner_tra(rsa15$rsa_um, tra) -> rsa15$rsa_um
# Ajout du tra à la partie actes des rsa :
inner_tra(rsa15$actes, tra) -> rsa15$actes
# Ajout du tra à la partie das des rsa :
inner_tra(rsa15$das, tra) -> rsa15$das
```

7.2 Ajout du TRA en HAD

```
# Import du TRA HAD
itra(finess = 750712184,
     annee = 2015,
     mois = 12,
     path = '~/Documents/data/had',
     champ = "had") -> tra

# Ajout du tra
inner_tra(data_had$rapss, tra, champ = "had") -> data_had$rapss
inner_tra(data_had$acdi, tra, champ = "had") -> data_had$acdi
inner_tra(data_had$ght, tra, champ = "had") -> data_had$ght
```

7.3 Ajout du TRA en SSR

```
# Import du TRA SSR
itra(finess = 750712184,
     annee = 2015,
     mois = 12,
     path = '~/Documents/data/ssr',
     champ = "ssr") -> tra

# Ajout du tra
inner_tra(data_ssr$rha, tra, champ = "ssr") -> data_ssr$rha
inner_tra(data_ssr$acdi, tra, champ = "ssr") -> data_ssr$acdi
```

7.4 Ajout du TRA en PSY

```
# Import du TRA PSY : fichiers RPSA
itra(finess = 750803454,
     annee = 2015,
     mois = 12,
     path = '~/Documents/data/psy',
     champ = "tra_psy_rpsa") -> tra

# Ajout du tra
inner_tra(rpsa_psy$rpsa, tra, champ = "psyrpsa") -> rpsa_psy$rpsa
inner_tra(rpsa_psy$das, tra, champ = "psyrpsa") -> rpsa_psy$das
```

```
# Import du TRA PSY : fichiers R3A
itra(finess = 750803454,
     annee = 2015,
     mois = 12,
     path = '~/Documents/data/psy',
     champ = "tra_psy_r3a") -> tra

# Ajout du tra
```

```
inner_tra(r3a_psy$r3a, tra, champ = "psyr3a") -> r3a_psy$r3a  
inner_tra(r3a_psy$das, tra, champ = "psyr3a") -> r3a_psy$das
```


8

Statistiques du PMSI

8.1 Âge et durée de séjour

```
library(dplyr)

# Age moyen et DMS sur les plus de 0 jour
rsa15$rsa %>%
  summarise(age_moyen = mean(AGEAN, na.rm = T),
            dms = mean(DUREE[DUREE > 0]),
            effectif = n(),
            effectif_sup0 = sum(DUREE > 0))

# Age moyen en prenant en compte les séjours des bébés
# (variable age en jour)
rsa15$rsa %>%
  mutate(Age = if_else(is.na(AGEAN), as.integer(AGEJR) / 365.25, as.numeric(AGEAN))) %>%
  summarise(age_moyen = mean(Age),
            effectif = n())
```

8.2 Nombre de séjours par catégorie majeure de diagnostics

```
# Nombre de séjours par catégorie majeure de diagnostics
rsa15$rsa %>% count(RSACMD)
```

8.3 Case-mix MCO, DMS par GHM / GHS

```
# Construire la variable GHM
rsa15$rsa %>% tidyr::unite(GHM,
                          RSACMD, RSATYPE, RSANUM, RSACOMPX,
                          sep = "") -> rsa15$rsa

# Case-mix par GHM
rsa15$rsa %>% count(GHM)
```

```
# Case-mix par GHM / GHS
rsa15$rsa %>% count(GHM, NOGHS)

# DMS par GHM / GHS
rsa15$rsa %>% group_by(GHM, NOGHS) %>%
  summarise(dms = mean(DUREE[DUREE > 0]),
            effectif = n(),
            effectif_sup0 = sum(DUREE > 0))
```

9

Help

Toutes les fonctions du package ont une page d'aide :

```
# Exemple : aide sur la fonction d'import des rsa  
?irsa
```