Service méthodes statistiques SMS INED

MUSÉUM NATIONAL D'HISTOIRE NATURELLE

SOCIETE FRANÇAISE DE STATISTIQUE

Groupe Enquêtes, Modèles et Applications

LES RENCONTRES DE STATISTIQUE APPLIQUÉE
Palette d'applications sous R
Jeudi 28 avril 2011

Accéder à une base de données avec 

Raymond Baudoin

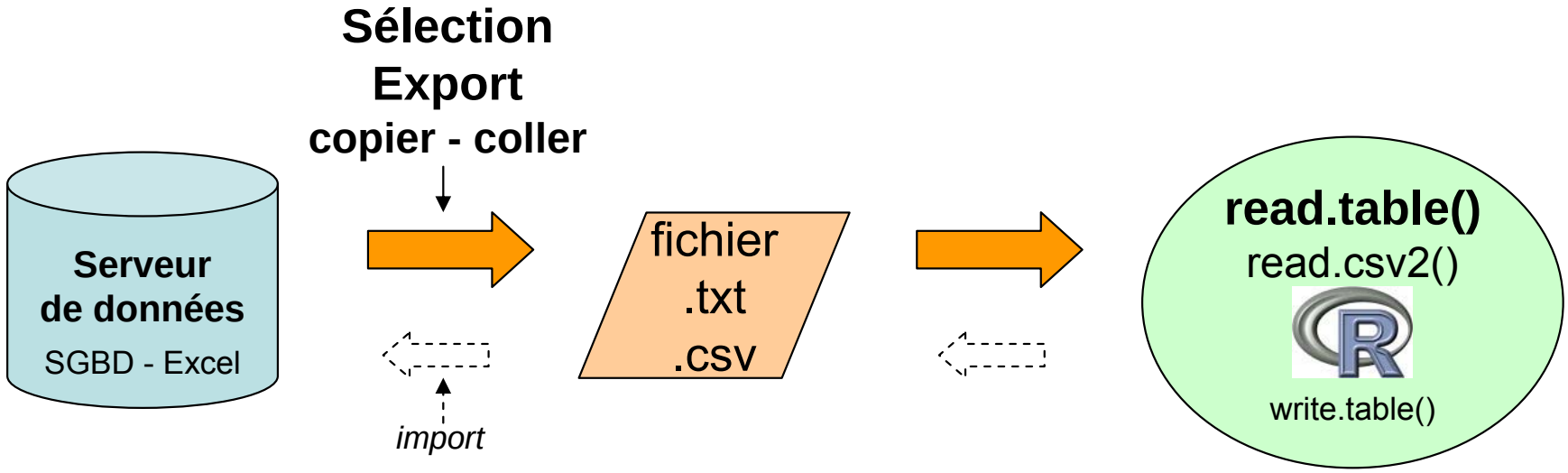
Muséum national d'Histoire naturelle
Département EGB
Inventaire et suivi de la biodiversité
Conservatoire botanique national du bassin parisien

baudoin@mnhn.fr

Avoir des données dans

Le chargement des données est la phase incontournable pour l'utilisation des puissants outils de traitements qu'offrent la majorité des librairies R. Mais bien souvent ces données sont créées et gérées par une autre application, tableur MS Excel, logiciel de base de données…

Le passage par un fichier intermédiaire issu d'une extraction qui est ensuite lu dans R est la procédure classique :



- L'opération est à refaire si les données sont modifiées ou si l'extraction n'est pas satisfaisante
- L'intégrité des données n'est plus respectée par l'existence du fichier intermédiaire

Fonctions de base pour lire un fichier texte

➢ Si le fichier est sous la forme d'un tableau statistique :

```
read.table (file = fich ,          # Nom complet du fichier ou "clipboard" pour le presse papier (copier/coller)
            sep = ";" ,            # Séparateur des valeurs, \t pour tabulation, \n : pas de séparateur
            dec = "," ,            # Symbole décimal
            header = FALSE ,
            stringsAsFactors = TRUE , # Conversion des chaines en facteur
            row.names = 1,         # colonne utilisée comme identifiant des lignes
            skip = 1, nrow = 3 )   # Restriction de lecture, ici saut de la première ligne et 3 lignes lues
```

```
> read.table (file = "Habitants.csv", sep=";", header=FALSE, nrow=2)
  V1      V2        V3     V4             V5    V6        V7        V8         V9
1 pk     Nom    Prénom numéro           voie  Code     Ville Téléphone     Depuis
2  1 Lecesve     André      9 rue Gay Lussac 92320 CHATILLON 146542856 19/10/1920
> read.csv2 (file = "Habitants.csv", row.names=1, nrow=2)
      Nom    Prénom numéro           voie  Code     Ville Téléphone     Depuis
1 Lecesve     André      9 rue Gay Lussac 92320 CHATILLON 146542856 19/10/1920
2 Larrouy Catherine     10 rue Gay Lussac 92320 CHATILLON 140920841 01/01/1999
> read.delim2 ("clipboard", row.names=1, nrow=2)
      Nom    Prénom numéro           voie  Code     Ville Téléphone     Depuis
1 Lecesve     André      9 rue Gay Lussac 92320 CHATILLON 146542856 19/10/1920
2 Larrouy Catherine     10 rue Gay Lussac 92320 CHATILLON 140920841 01/01/1999
```

Copier/Coller d'une feuille Excel

➢ Si le fichier est sous une autre forme :

scan (file = fich , sep=";", what = "character", skip = 1, nlines = 3)

```
> scan (file = "Habitants.csv", sep=";", what = "character",skip=1, nlines=2)
Read 18 item
 [1] "1"           "Lecesve"     "André"       "9"           "rue Gay Lussac" "92320"      "CHATILLON"
 [8] "146542856"   "19/10/1920"  "2"           "Larrouy"     "Catherine"      "10"         "rue Gay Lussac"
[15] "92320"       "CHATILLON"   "140920841"   "01/01/1999"
```

readLines (con = fich, n = 3)

```
> readLines (con = "Habitants.csv", n=2)
[1] "pk;Nom;Prénom;numéro;voie;Code;Ville;Téléphone;Depuis"
[2] "1;Lecesve;André;9;rue Gay Lussac;92320;CHATILLON;146542856;19/10/1920
```

Fonctions pour lire une feuille MS-Excel

R ≥ 2.12.2

au format .xls (Excel 97/2000/XP/2003) ou .xlsx (Excel 2007)

library (xlsx)

read.xlsx (

```
    file,                          # nom du fichier
    sheetIndex=,                   # numéro de la feuille
    sheetName=NULL,                # ou son nom
    as.data.frame=TRUE,
    header=TRUE,
    colClasses=NA,                 # numeric, integer, logical, character, Date
    rowIndex=NULL,
    colIndex=NULL,
    keepFormulas=FALSE,
    encoding="unknown",
    ...)
```

read.xlsx2 (

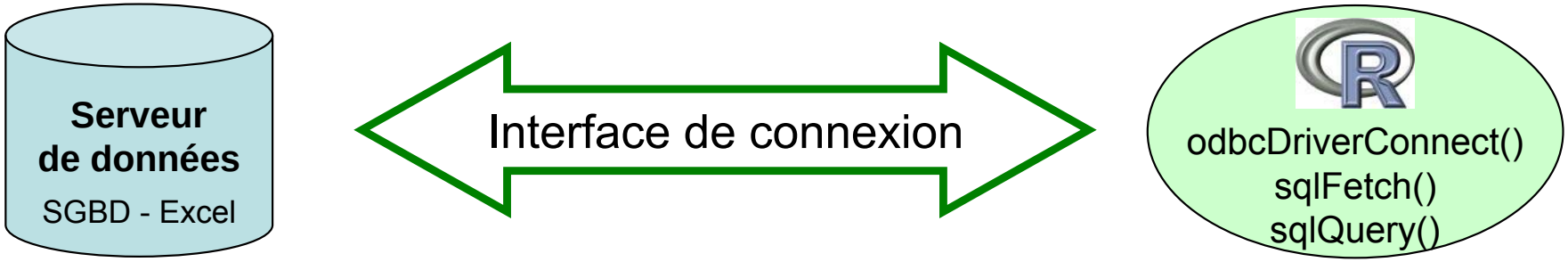
```
    file,                          # nom du fichier
    sheetIndex=,                   # numéro de la feuille
    sheetName=NULL,                # ou son nom
    as.data.frame=TRUE,
    header=TRUE,
    colClasses="character",        # ou "numeric"
    startRow=1,
    startColumn=1,
    noRows=NULL,                   # nombre de lignes à lire
    noColumns=NULL)
```

```
read.xlsx ("OCCSOL_1000.xlsx", sheetIndex=1, rowIndex=c(1:4,rep(0,135-4)) )
  NA.     BB      BH      BI      BM     BT       O     TR     VB     VG     VH     VM
1   0 0.0787 25.2158 30.0836 15.3149 1.1547  0.0465 6.5254 5.4042 3.2627 5.0906 7.8229
2   1 0.2431 22.2879 27.8258 22.7205 1.1578  0.0000 5.9316 5.1204 2.8737 3.8698 7.9693
3   2 0.1411 16.7779 23.4831  8.4136 0.7781 18.2162 7.0589 6.9271 2.5255 6.9970 8.6815
read.xlsx2 ("OCCSOL_1000.xlsx", sheetIndex=1, colClasses="numeric", startRow=1, noRows=3)
  c.0..1.     BB      BH      BI      BM     BT      O     TR     VB     VG     VH     VM
1       0 0.0787 25.2158 30.0836 15.3149 1.1547 0.0465 6.5254 5.4042 3.2627 5.0906 7.8229
2       1 0.2431 22.2879 27.8258 22.7205 1.1578 0.0000 5.9316 5.1204 2.8737 3.8698 7.9693
read.xlsx ("Habitants.xls", sheetIndex=1, rowIndex=c(1:3,rep(0,14-3)), encoding="UTF-8", stringsAsFactors=FALSE)
      Nom    Prénom numéro         Adresse  Code     Ville Téléphone     Depuis
1 Lecesve     André      9 rue Gay Lussac  92320 CHATILLON 146542856 1920-10-19
2 Larrouy Catherine     10  rue Gay Lussac 92320 CHATILLON 140920841 1999-01-01
```

Aussi : **library (xlsReadWrite)** pour lire et écrire au format .xls
library (gdata) : read.xls ()

Pourquoi connecter une base de données avec R ?

Le passage par un fichier intermédiaire peut être, dans bien des cas, évité par une connexion directe à l'application gérant les données.



SGBD : Système de gestion de base de données

- Sélection des données dans *R*
- Intégrité des données respectée

Avec un serveur de données comme Oracle, Postgres, MySQL, MS-Access, respectant la norme SQL (Structured Query Language - langage normalisé de contrôle et interrogation des bases de données relationnelles), l'utilisation de la commande SELECT offre la possibilité de ne prendre en compte que certaines variables (champs) et de sélectionner sur les contenus (valeurs) pour obtenir le tableau souhaité (data.frame) et unique même dans le cas d'un schéma relationnel complexe (n tables).

Les librairies utilisées:

- pour la connexion à des bases de données : RODBC, SQLiteDF, RMySQL, RPostgreSQL, ROracle, DBI
- pour accéder aux feuilles de classeurs MS-Excel : RODBC, xlsx
- pour utiliser la commande SELECT sur des fichiers ou des data frames : sqldf

SGBD : Système de gestion de base de données

Ou DBMS (Database management system)

Ensemble de services pour :

- permettre l'accès aux données de façon simple
- manipuler les données présentes dans la base (insertion, suppression, modification)
- autoriser un accès aux informations à de multiples utilisateurs

Se compose :

- D'un moteur pour le stockage des informations sur le support physique
- De sous-systèmes gérant :
  - La définition et la structure les données
  - L'administration les données
  - La manipulation les données
  - L'interface utilisateur

SQLite : SGBD utilisable sous 

- moteur de base de données relationnelles accessible par le langage SQL.
- la base de données (déclarations, index, données) stockée dans un fichier
- multi-plateforme
- utilisé dans de nombreuses applications : Firefox, Skype, Google Gears…

http://www.sqlite.org/

Une interface utilisateur :
SQLite Database Browser

http://sqlitebrowser.sourceforge.net/

La librairie sqldf permet à  **d'utiliser SQLite**

La table : élément de base pour les données dans un SGBD

- Une table est composée de **lignes**
- Une ligne est un ensemble fixe de **champs** (attributs)

≈ une feuille xls

Exemple :
la table Personnes

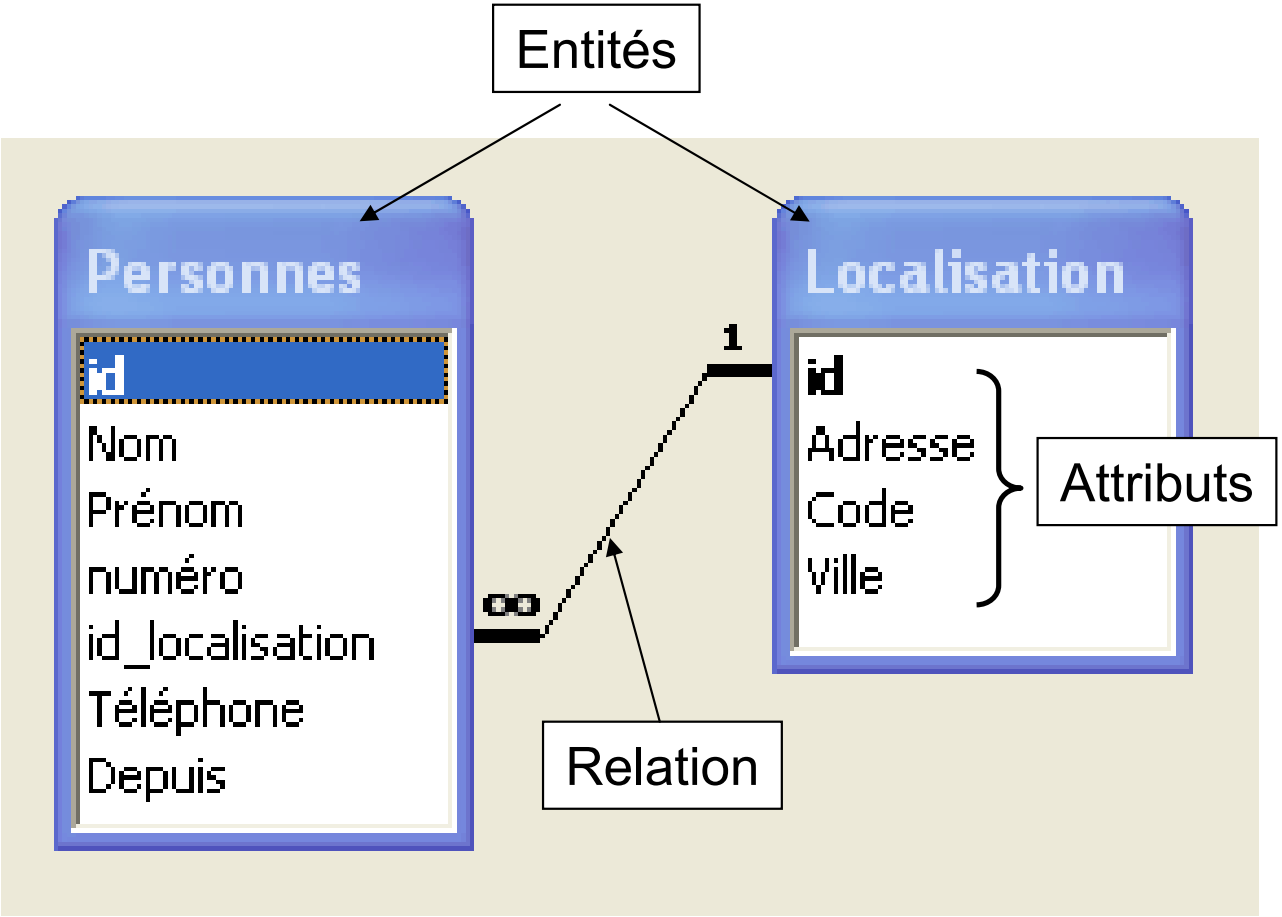
Chaque champ a :

- ***un nom*** ⇨
- ***un type*** ⇨

id	Nom	Prénom	numéro	id_localisation	Téléphone	Depuis
1	Lecesve	André	9	2	146542856	19/10/1920
2	Larrouy	Catherine	10	2	140920841	01/01/1999
3	Larrouy	Eric	10	2	140920841	01/01/1999
6	Meyer	Michel	15	3		05/06/1960
8	Auquier	Anne	21	3	157750048	05/09/2005
9	Auquier	Anne	21	3	636699001	05/09/2005
10	Auquier	Bernard	21	3	146428564	05/09/2005
INTEGER	CHARACTER		INTEGER		DECIMAL	DATE

Pour 
une table correspond à la définition d'un data.frame

Le modèle relationnel des SGBD r



Objectif du modèle : Pas d'information redondante

Les données sont réparties en n tables ou entités dont des champs (attributs) identiques assurent les liaisons (relations)

Relations possibles : 1/1, 1/n, m/1
Une relation n / m nécessite une table de jointure

La **clef primaire** d'une table est un attribut ou un groupe d'attributs dont la valeur identifie de manière unique une ligne de la table.

SQL (Structured Query Language)

Langage normalisé
pour décrire, manipuler, contrôler l'accès et interroger
les bases de données relationnelles

Caractéristiques

- Inventé en 1970
- Déclaratif c'est-à-dire indépendant du contexte d'exécution
- Régi par une norme ANSI/ISO
  - Première normalisation ISO en 1987
  - SQL 2 correspond à la norme SQL/92
  - SQL 3 en 1999, actuellement SQL 2003
- Portable sur différentes plates-formes aussi bien matérielles que logicielles.

Une commande SQL écrite dans un environnement Windows sous MS-ACCESS est utilisable directement dans un environnement ORACLE sous Unix.

SQL (Structured Query Language)

Langage structuré de requêtes

regroupe :

- un langage de contrôle des accès (DCL, Data Control Language)
- un langage de définition de données (DDL Data Definition Language)
  - Pour créer, modifier ou supprimer des tables dans la base
- un langage de manipulation de données (DML, Data Manipulation Language)
  - Pour insérer, modifier ou supprimer des données dans une table
  - **Pour sélectionner des données**
    Sélection des lignes, des colonnes, dans une ou plusieurs tables avec l'utilisation de la ***commande SELECT***

Syntaxe de la commande SELECT

```
SELECT [ALL | DISTINCT] { * | col | expr [AS alias], ... }
FROM table [alias], ...
[ WHERE { conditions de recherche | sous conditions} ]
[ GROUP BY col, ...] [HAVING conditions de recherche ]
[ ORDER BY { col | num } {ASC | DESC}, ...] ;
```

Légende :
{ } : Une des valeurs séparées par '|' obligatoire.
[ ] : Valeur optionnelle.
... : Répétition.
__ : Valeur par défaut.

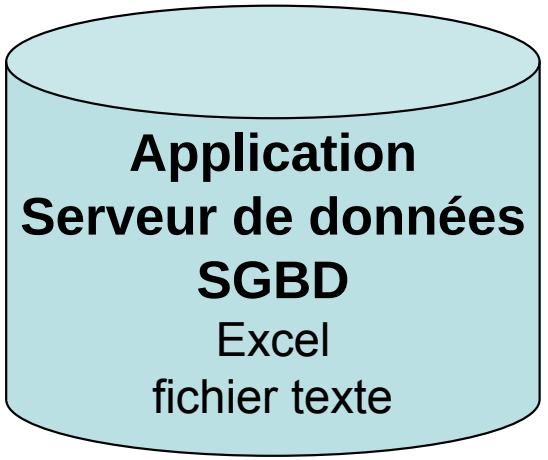
SELECT	Précise les colonnes qui vont apparaître dans la réponse
FROM	Précise la (ou les) table intervenant dans l'interrogation
WHERE	Précise les conditions à appliquer sur les lignes avec : - Des opérateurs de comparaison : =, >, <, >=, <=,<> - Des opérateurs logiques : AND, OR, NOT - Des prédicats : IN, LIKE, NULL, ALL, ANY...
GROUP BY	Précise la (ou les) colonne de regroupement
HAVING	Précise la (ou les) condition associée à un regroupement
ORDER BY	Précise l'ordre dans lequel vont apparaître les lignes de la réponse : - ASC : ordre ascendant (par défaut) - DESC : ordre descendant

Accès à l'application gérant les données

Interface de connexion

DBI

DataBase Interface



L'interface doit :

- **Identifier l'application**
- **Etablir la connexion**
- **Renvoyer les données**

rôle de

DBI Driver

DBI Connexion

DBI Résultats

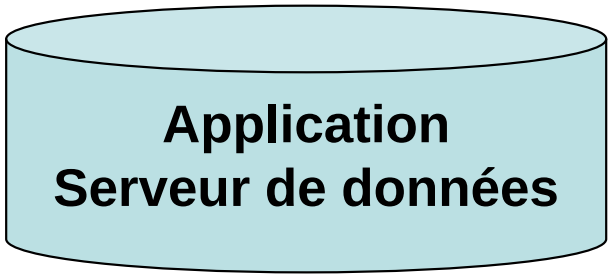
Une interface particulière définie par Microsoft

ODBC **Open DataBase Connectivity**

- Couche logiciel qui permet la connexion entre applications (Windows)
- L'application doit avoir un pilote ODBC
- La source de données est identifiée par son DSN = Data Source Name
- Dans ce cas la connexion se fait directement via le pilote ODBC de l'application

Des packages R spécifiques *

R ≥ 2.9



RMySQL	Package source: RMySQL_0.7-4.tar.gz MacOS X binary: RMySQL_0.7-4.tgz Windows binary: RMySQL_0.7-4.zip	MySQL
RPostgreSQL	Package source: RPostgreSQL_0.1-6.tar.gz MacOS X binary: Non disponible Windows binary: RPostgreSQL_0.1-6.zip	PostgreSQL
ROracle	Package source: ROracle_0.5-9.tar.gz MacOS X binary: Non disponible Windows binary: Non disponible	Oracle
RSQLite	Package source: RSQLite_0.8-2.tar.gz MacOS X binary: RSQLite_0.8-2.tgz Windows binary: RSQLite_0.8-2.zip	SQLite
RODBC	Package source: RODBC_1.3-1.tar.gz MacOS X binary: RODBC_1.3-1.tgz Windows binary: RODBC_1.3-1.zip	Applications ayant un pilote ODBC : Access, Excel, Oracle, SQLite...

** Contient à la fois le pilote et l'interface (DBI)*

Accéder à des données via une DBI

➢ Etape 1 : se connecter à la base

```
library (RSQLite)                                  # Exemple : accès à une base SQLite
db <- "SeminR.db"                                  # nom de la base uniquement - faire un setwd ()
drv <- dbDriver ("SQLite")                         # charge le pilote
con <- dbConnect (drv, dbname = db)                # connecte  la base db avec le pilote drv
```

Informations sur la connexion

```
drv
<SQLiteDriver:(2192)>
con
<SQLiteConnection:(2192,0)>

dbListConnections(drv )
[[1]]
<SQLiteConnection:(2192,0)>

[[2]]
<SQLiteConnection:(2192,1)>
```

```
dbGetInfo (con)
$dbname
[1] "SeminR.db"

$serverVersion
[1] "3.7.3"
```

```
summary (con)
<SQLiteConnection: DBI CON (2192, 5)>
  Database name: SeminR.db
  Loadable extensions: on
  File open flags:
  Virtual File System:
  SQLite engine version:  3.7.3
  Results Sets:
   No open result sets
```

```
dbDisconnect (con)                                 # Pour fermer la connexion
                                                   # OBLIGATOIRE pour libérer l'accès
```

➢ Etape 2 : voir la structuration des données

connexion

```
dbListTables (con)                    # liste des tables de la base connectée par con
[1] "Habitants"    "localisation" "personnes"

dbExistsTable (con, "localisation")# vérifie l'existence de la table
[1] TRUE

dbListFields (con, "localisation")   # liste les champs de la table localisation
[1] "id"      "Adresse" "Code"    "Ville"

 dbListFields (con, "personnes")         # liste les champs de la table personnes
 [1] "id"             "Nom"            "Prenom"         "numero"
 [5] "id_localisation" "Telephone"      "Depuis"
```

➢ Etape 3 : lire toutes les données d'une table

lecture de la table Habitants dans la base connectée par *con*

connexion ↓ nom de la table ↓

```
( dDBI <- dbReadTable ( con, "Habitants") )
```

	pk	Nom	Prenom	numero	voie	Code	Ville	Telephone	Depuis
1	1	Lecesve	AndrÃ©	9	rue Gay Lussac	92320	CHATILLON	146542856	19/10/1920
2	2	Larrouy	Catherine	10	rue Gay Lussac	92320	CHATILLON	140920841	01/01/1999
3	3	Larrouy	Eric	10	rue Gay Lussac	92320	CHATILLON	140920841	01/01/1999
4	4	Malher	Goerges	13	rue Gay Lussac	92320	CHATILLON	146576986	05/06/1960
5	5	Lipinski	Ludovic	15	rue Gay Lussac	92320	CHATILLON	147352329	23/09/1952
6	6	Meyer	Michel	15	rue Roissis	92140	CLAMART	NA	05/06/1960
7	7	Foucher	Georges	17	rue Roissis	92140	CLAMART	146449501	03/02/2000
8	8	Auquier	Anne	21	rue Roissis	92140	CLAMART	157750048	05/09/2005
9	9	Auquier	Anne	21	rue Roissis	92140	CLAMART	636699001	05/09/2005
10	10	Auquier	Bernard	21	rue Roissis	92140	CLAMART	146428564	05/09/2005
11	11	Mahier	Ludovic	3	avenue Verdun	92170	VANVES	147361266	07/05/1983
12	12	Berrue	Christiane	4	avenue Verdun	92170	VANVES	146381434	21/10/1985
13	13	Berrue	Christiane	4	avenue Verdun	92170	VANVES	954912355	21/10/1985

Problème de codage
Pour le modifier :

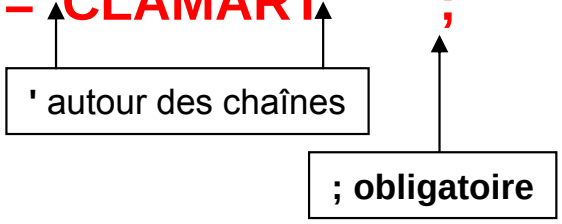
```
for (x in which
(sapply(dDBI,class)=="character") )
Encoding (dDBI [,x]) <-"UTF-8"
```

```
str(dDBI)
'data.frame':   13 obs. of  9 variables:
 $ pk       : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Nom      : chr  "Lecesve" "Larrouy" "Larrouy" "Malher" ...
 $ Prenom   : chr  "André" "Catherine" "Eric" "Goerges" ...
 $ numero   : int  9 10 10 13 15 15 17 21 21 21 ...
 $ voie     : chr  "rue Gay Lussac" "rue Gay Lussac" "rue Gay Lussac" "rue Gay Lussac" ...
 $ Code     : int  92320 92320 92320 92320 92320 92140 92140 92140 92140 92140 ...
 $ Ville    : chr  "CHATILLON" "CHATILLON" "CHATILLON" "CHATILLON" ...
 $ Telephone: chr  "146542856" "140920841" "140920841" "146576986" ...
 $ Depuis   : chr  "19/10/1920" "01/01/1999" "01/01/1999" "05/06/1960" ...
```

➢ Etape 3' : lire les données *par une sélection SQL*

Rédaction de la requête :

```
sql <- "SELECT nom, prenom, ville                          # champs sélectionnés
        FROM personnes, localisation                       # tables utilisées
        WHERE personnes.id_localisation = localisation.id  # jointure (relation)
        AND Ville = 'CLAMART'      ;"                     # critère de sélection des lignes
```



1ère procédure : lecture de l'intégralité de la sélection

connexion → ← requête SQL

```
dbGetQuery (con, sql) # exécution de la requête
```

```
      Nom  Prenom   Ville
1   Meyer  Michel CLAMART
2 Foucher Georges CLAMART
3 Auquier    Anne CLAMART
4 Auquier    Anne CLAMART
5 Auquier Bernard CLAMART
```

résultat du SELECT

Accéder à des données via une DBI

- Etape 3' : lire les données par une sélection SQL

2ème procédure : lecture avec un curseur

connexion, requête SQL

```
res <- dbSendQuery (con, sql)      # initialise le curseur (0) à partir de la requête
```

curseur

```
fetch (res, n = 2)                 # lecture de n lignes
                                   #     à partir de la position du curseur + 1
```

```
      Nom  Prenom   Ville
1   Meyer  Michel CLAMART
2 Foucher Georges CLAMART
```

```
fetch (res, n = -1)                # si n = -1 lecture jusqu'à la fin
                                   #     à partir de la position du curseur + 1
```

```
      Nom  Prenom   Ville
3 Auquier    Anne CLAMART
4 Auquier    Anne CLAMART
5 Auquier Bernard CLAMART
```

```
dbGetRowCount (res)                # affiche le nombre de lignes lues
[1] 5
dbHasCompleted (res)               # test de la fin de la table
[1] TRUE
```

Autres fonctions associées :

```
dbClearResult (res)    # libère le curseur. A faire pour un nouveau dbSendQuery (res) si dbHasCompleted (res) est FALSE
dbColumnInfo (res)     # affiche les caractéristiques des champs de la requête
dbGetStatement (res)   # affiche le SELECT
```

Accéder à des données via ODBC

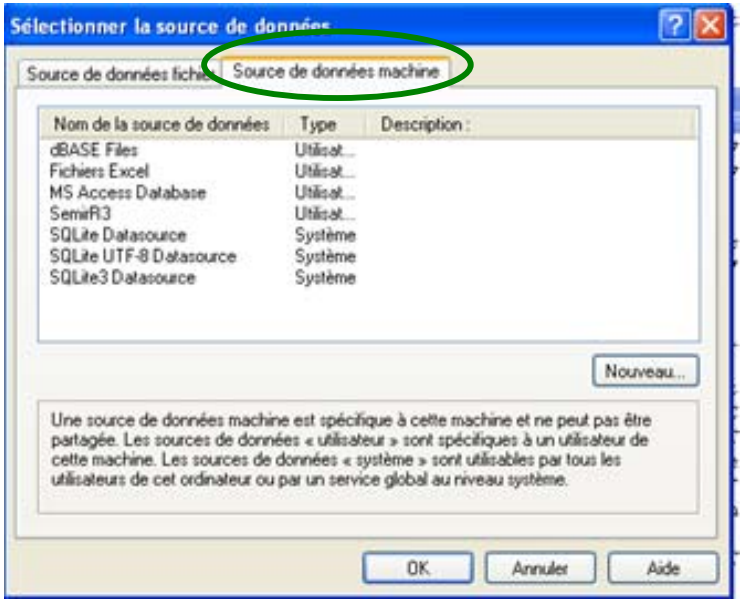
library (RODBC)

channel <- odbcDriverConnect ()

Faire : Panneau de configuration - Outils d'administration - Sources de données (ODBC)
C:\Documents and Settings\All Users\Menu Démarrer\Programmes\Outils d'administration

La base est dans un répertoire accessible

Choix du type pilote odbc de l'application

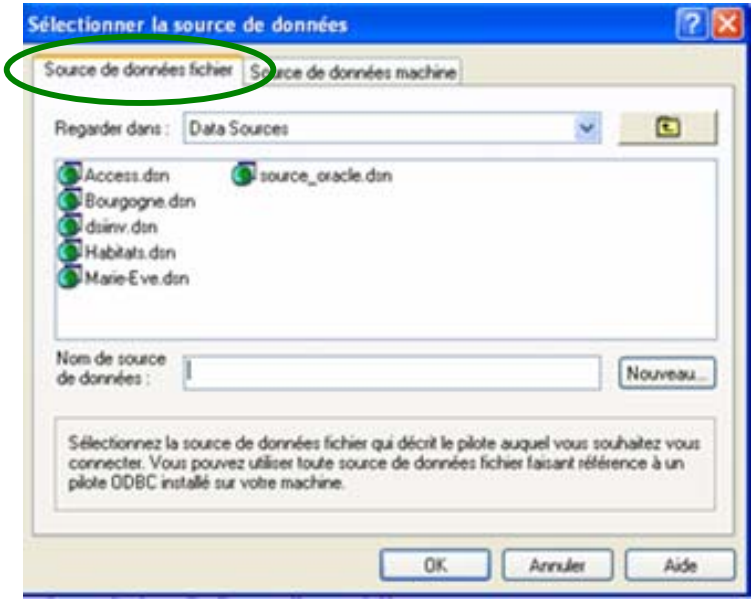


Choix de l'application

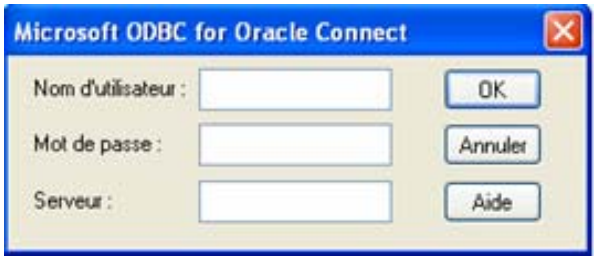
odbcConnectAccess (access.file, ...)
odbcConnectAccess2007 (access.file, ...)
odbcConnectDbase (dbf.file, ...)
odbcConnectExcel (xls.file, readOnly = TRUE, ...)
odbcConnectExcel2007 (xls.file, ...)

La base est sur un serveur distant

L'accès à la base est décrit dans le dsn associé



Si le contrôle de d'accès est activé



dsn dans:
C:\Program Files\Fichiers communs\ODBC\Data Sources)

Accéder à des données via ODBC

➢ Etape 1 : se connecter à la source de données

une base sqlite

```
library (RODBC)
chadb <- odbcDriverConnect()

chadb
RODBC Connection 7
Details:
case=nochange
DSN=SeminR3
Database=D:\Utilisateurs\Raymond\Enseignement\Séminaires R\
    INED\Présentation\SeminR.db
```

un classeur Excel

```
library (RODBC)
chaxls <- odbcConnectExcel ("Habitants.xls")

chaxls
RODBC Connection 3
Details:
case=nochange
DBQ=D:\Utilisateurs\Raymond\Enseignement\Séminaires R\
    INED\Présentation\Habitants.xl
DefaultDir=D:\Utilisateurs\Raymond\Enseignement\
    Séminaires R\INED\Présentation
Driver={Microsoft Excel Driver (*.xls)}
DriverId=790
MaxBufferSize=2048
PageTimeout=5
```

une base Access

```
library (RODBC)
(conn <- file.choose())
[1] "D:\\...Présentation\\RODBC.mdb"

chamdb <- odbcConnectAccess (conn)

chamdb
RODBC Connection 4
Details:
case=nochange
DBQ=D:\Utilisateurs\Raymond\Enseignement\Séminaires R\
   INED\Présentation\RODBC.mdb
Driver={Microsoft Access Driver (*.mdb)}
DriverId=25
FIL=MS Access
MaxBufferSize=2048
PageTimeout=5
UID=admin
```

Déconnexion

```
odbcCloseAll ()   # ferme toutes les connexions
close (channel)   # ferme la connexion indiquée
```

obligatoire pour accéder au classeur par Excel

➢ Etape 2 : voir la structure des données

connexion

sqlTables (chadb) [,-c(2,5)] **# pour une base sqlite**

```
  TABLE_CAT   TABLE_NAME TABLE_TYPE
1      <NA> localisation      TABLE
2      <NA>    personnes      TABLE
3      <NA>    Habitants      TABLE
```

sqlTables (chamdb) [,-c(2,5)] **# pour une base Access**

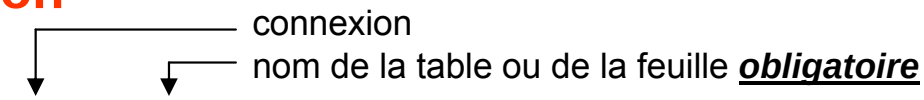
```
                        TABLE_CAT        TABLE_NAME   TABLE_TYPE
1  D:\\...\\Présentation\\RODBC MSysAccessObjects SYSTEM TABLE
2  D:\\...\\Présentation\\RODBC     MSysAccessXML SYSTEM TABLE
3  D:\\...\\Présentation\\RODBC          MSysACEs SYSTEM TABLE
4  D:\\...\\Présentation\\RODBC   MSysIMEXColumns SYSTEM TABLE
5  D:\\...\\Présentation\\RODBC    MSysIMEXSpecs  SYSTEM TABLE
6  D:\\...\\Présentation\\RODBC       MSysObjects SYSTEM TABLE
7  D:\\...\\Présentation\\RODBC       MSysQueries SYSTEM TABLE
8  D:\\...\\Présentation\\RODBC MSysRelationships SYSTEM TABLE
9  D:\\...\\Présentation\\RODBC         Habitants        TABLE
10 D:\\...\\Présentation\\RODBC     Localisation        TABLE
11 D:\\...\\Présentation\\RODBC         Personnes        TABLE
```

sqlTables (chaxls) [,-c(2,5)] **# pour un classeur Excel**

```
                                TABLE_CAT TABLE_NAME   TABLE_TYPE
1 D:\\...\\Présentation\\Habitantstants    Feuil1$ SYSTEM TABLE
2 D:\\...\\Présentation\\Habitantstants    Feuil2$ SYSTEM TABLE
3 D:\\...\\Présentation\\Habitantstants    Feuil3$ SYSTEM TABLE
```

➢ Etape 2' : voir la structure des tables

tbl = "localisation"



sqlColumns (chadb, tbl) [,3:6] **# table localisation de la base sqlite**

	TABLE_NAME	COLUMN_NAME	DATA_TYPE	TYPE_NAME
1	localisation	id	4	INTEGER
2	localisation	Adresse	-9	VARCHAR
3	localisation	Code	8	NUMERIC
4	localisation	Ville	-10	TEXT

sqlColumns (chamdb, tbl) [,3:6] **# table localisation de la base Access**

	TABLE_NAME	COLUMN_NAME	DATA_TYPE	TYPE_NAME
1	localisation	id	4	COUNTER
2	localisation	Adresse	12	VARCHAR
3	localisation	Code	4	INTEGER
4	localisation	Ville	12	VARCHAR

sqlColumns (chaxls, "Feuil1") [,3:6] **# feuille 1 du classeur Excel**

	TABLE_NAME	COLUMN_NAME	DATA_TYPE	TYPE_NAME
1	Feuil1$	Nom	12	VARCHAR
2	Feuil1$	Prénom	12	VARCHAR
3	Feuil1$	numéro	8	NUMBER
4	Feuil1$	Adresse	12	VARCHAR
5	Feuil1$	Code	8	NUMBER
6	Feuil1$	Ville	12	VARCHAR
7	Feuil1$	Téléphone	8	NUMBER
8	Feuil1$	Depuis	93	DATETIME

➢ Etape 3 : lire toutes les données d'une table

à l'aide d'un curseur (max)

connexion → ← nom de la table ou de la feuille

sqlFetch (chamdb,"personnes", max=3) # initialise le curseur et lecture de 3 lignes

```
     Nom    Prénom numéro          Adresse  Code     Ville Téléphone     Depuis
1 Lecesve     André      9 rue Gay Lussac  92320 CHATILLON 146542856 1920-10-19
2 Larrouy Catherine     10  rue Gay Lussac 92320 CHATILLON 140920841 1999-01-01
3 Larrouy      Eric     10  rue Gay Lussac 92320 CHATILLON 140920841 1999-01-01
```

par défaut max = 0

sqlFetchMore (chamdb, max=1) # lecture de n lignes à partir de la position du curseur +1

```
    Nom  Prénom numéro          Adresse  Code     Ville Téléphone     Depuis
1 Malher Goerges     13 rue Gay Lussac 92320 CHATILLON 146576986 1960-06-05
```

sqlFetchMore (chamdb) # par défaut max = 0 : lecture jusqu'à la fin

```
       Nom     Prénom numéro          Adresse  Code     Ville Téléphone     Depuis
1 Lipinski    Ludovic     15 rue Gay Lussac 92320 CHATILLON 147352329 1952-09-23
2    Meyer     Michel     15    rue Roissis 92140   CLAMART        NA 1960-06-05
3  Foucher    Georges     17    rue Roissis 92140   CLAMART 146449501 2000-02-03
4  Auquier       Anne     21    rue Roissis 92140   CLAMART 157750048 2005-09-05
5  Auquier       Anne     21    rue Roissis 92140   CLAMART 636699001 2005-09-05
6  Auquier    Bernard     21    rue Roissis 92140   CLAMART 146428564 2005-09-05
7   Mahier    Ludovic      3  avenue Verdun 92170    VANVES 147361266 1983-05-07
8   Berrue Christiane      4  avenue Verdun 92170    VANVES 146381434 1985-10-21
9   Berrue Christiane      4  avenue Verdun 92170    VANVES 954912355 1985-10-21
```

sqlFetchMore (chamdb, max=1)

```
[1] -1
```

← code de fin ou <0 lignes> (cas sqlite)

Fermer la connexion pour accéder au fichier par Excel
close (chaxl)

➢ Etape 3' : lire les données *par une sélection SQL*

Rédaction de la requête

```
sql <- "SELECT nom, prénom, code, ville                    # champs sélectionnés
          FROM personnes, localisation                     # tables utilisées
           WHERE personnes.id_localisation = localisation.id # jointure (relation)
             AND Code > 92300                              # critère de sélection des lignes
               ORDER BY nom ; "                            # tri alphabétique
```

obligatoire

Exécution de la requête

connexion → ; requête SQL →

```
sqlQuery (chamdb, sql)
```

```
       nom    prénom  code     ville
1  Larrouy      Eric 92320 CHATILLON
2  Larrouy Catherine 92320 CHATILLON
3  Lecesve     André 92320 CHATILLON
4 Lipinski   Ludovic 92320 CHATILLON
5   Malher   Goerges 92320 CHATILLON
```

Résumé de la démarche

via DBI → **Connexion** ← via ODBC

exemple : SGBD sqlite

Charger la librairie

library (sqlite)	library (RODBC)

Charger le pilote

drv <- dbDriver ("SQLite")

Ouvrir une connexion sur les données

con <- dbConnect (drv, dbname=*...)	channel <- odbcDriverConnect()
* nom de la base uniquement	channel <- odbcConnectExcel(…)
	channel <- odbcConnectAccess(…)

Voir la structure des données

dbListTables (con)	sqlTables (channel,…)
dbListFields (con, "table")	sqlColumns (channel, "table",…)
dbGetRowCount (res); dbColumnInfo (res)	

Charger les données avec tous les champs

dbReadTable (con, "table")	sqlFetch (channel, "table",…)
	sqlFetchMore (channel,…)

Charger les données avec un SELECT (sql)

dbGetQuery (con, sql,…)	sqlQuery (channel, sql,…)
res <- dbSendQuery (con, sql)	
fetch (res, n = -1)	

Fermer la connexion : OBLIGATOIRE pour libérer l'accès

dbDisconnect (con)	close (channel) ou odbcCloseAll ()

Libérer le pilote

dbUnloadDriver (drv)

Utiliser SQL pour manipuler des données dans
library (sqldf)

sqldf ("Requête SQL", dbname = ..., connexion = ...)

SELECT

nom* d'une base SQLite
par défaut :
ouvre une base virtuelle
en mémoire

** uniquement le nom de la base*

OU

nom d'une connexion
sur une base SQLite
ouverte par :
- dbConnect ()
- sqldf (dbname =)
par défaut

- **sur des tables d'une base SQLite** (ou MySQL)
- **sur des fichiers textes**
- **sur des data.frames**

Remarque sur l'ouverture / fermeture de la connexion

```
( sqldf() ) # ouverture d'une connexion sur une base virtuelle
<SQLiteConnection:(1036,0)>
( sqldf() ) # fermeture de la connexion au 2ème appel
NULL
```

sqldf () pour manipuler des données dans

1 - Utilisation avec une base SQLite

requête SQL

```
pers <- sqldf ("SELECT Nom, prenom, id_localisation FROM Personnes ;",
               dbname = "SeminR.db")                                  # nom de la base

condb <- dbConnect (drv = "SQLite", dbname = "SeminR.db")
loca <- sqldf ("SELECT * FROM localisation ;", connection = condb)    # connexion
```

2 - Utilisation avec des data.frames

```
sql <- "SELECT Nom, Ville FROM loca , pers          # data.frame
        WHERE loca.id = pers.id_localisation        # jointure
          AND Nom LIKE 'L%' ;"                      # troncature à droite
NV <- sqldf (sql)     # requête exécuté dans la base virtuelle
```

```
NV
       Nom     Ville
1  Lecesve CHATILLON
2  Larrouy CHATILLON
3  Larrouy CHATILLON
4 Lipinski CHATILLON
```

3 - Utilisation avec des fichiers textes

```
Loc <- file ("Localisation.txt")   # connexion au fichier texte
   attr (Loc ,"file.format") = list (sep = "\t", header=TRUE)

sql1 ="SELECT DISTINCT Nom, code, NV.Ville FROM NV, Loc
        WHERE Loc.Ville = NV.Ville ;"  # jointure
sqldf (sql1)
```

data.frame

fichier texte

```
       Nom  Code     Ville
1  Larrouy 92320 CHATILLON
2  Lecesve 92320 CHATILLON
3 Lipinski 92320 CHATILLON
```