IPB University
Bogor Indonesia

STK371
KOMPUTASI STATISTIK 1

Materi 5. Berhubungan dengan Data Eksternal

https://www.stat.ipb.ac.id/agusms/index.php/stk371/

Pendahuluan

- Terdapat beberapa paket untuk melakukan hubungan dengan data eksternal melalui koneksi data:
  - Paket utils
  - Paket foreign
  - Paket DBI:
    - Paket RSQLite
    - Paket RODBC
    - Paket RMySQL
  - dll

Paket utils

- Turunan dari read.table
- Membaca data
  - read.csv() : angka format internasional
  - read.csv2() : angka format Indonesia
- Menyimpan data
  - write.csv()
  - write.csv2()

Paket foreign

- Paket foreign digunakan untuk membaca atau menulis data pada dbase dan aplikasi statistik umum
- Baca atau Tulis data :

–dBase (DBF)	- SAS
–Epidata/EpiInfo	- SPSS (portable)
–Minitab	- Stata
–S-PLUS	- Systat

Paket foreign

- dbase
  - read.dbf()
  - write.dbf()
- •EpiData/EpiInfo
  - read.epiinfo()
- •Minitab
  - read.mtp()
- •S-Plus
  - read.S()

Paket foreign

- SAS
  - –read.ssd()
  - –read.xport()
- SPSS
  - –read.spss()
- Stata
  - –read.dta() - write.dta()

write.foreign(, package = c("SPSS", "Stata", "SAS"))

- Systat
  - –read.systat()

Paket Database Interface (DBI)

- Merupakan abstraksi dari class interface database yang menghubungkan dataframe dengan database relational
- Harus diimplementasikan dengan paket R<dbase>:
  - RODBC
  - RSQLite
  - RMySQL

Paket RODBC

- Membaca dan menulis data menggunakan ODBC
  - Excel
  - dBase
  - Access
  - dll yg memiliki database connector odbc

Paket RODBC - connect

- Syntax

```
odbcConnect(dsn, uid = "", pwd = "", ...)
odbcDriverConnect(connection = "", case, believeNRows
    = TRUE, colQuote, tabQuote = colQuote,
    interpretDot = TRUE, DBMSencoding = "",
    rows_at_time = 100, readOnlyOptimize = FALSE)
odbcReConnect(channel, ...)
odbcConnectAccess(access.file, uid = "", pwd = "",
    ...)
odbcConnectAccess2007(access.file, uid = "", pwd =
    "", ...)
odbcConnectDbase(dbf.file, ...)
odbcConnectExcel(xls.file, readOnly = TRUE, ...)
odbcConnectExcel2007(xls.file, readOnly = TRUE, ...)

close(nameconnection)
```

Paket RODBC - channel

- MySQL on Windows -- MySQL maps to lower case on Windows only

```
channel <- odbcConnect("testdb", uid="ripley",
    pwd="tolower")
```

- MS Access

```
channel <- odbcConnect("testacc")
channel2 <- odbcConnectAccess("test.mdb", uid="ripley")
```

- Excel

```
channel <- odbcConnect("bdr.xls")
channel2 <- odbcDriverConnect(paste("DRIVER=Microsoft
    Excel Driver (*.xls)", "DBQ=D:\bdr\hills.xls",
    "ReadOnly=False", sep = ";"))
Atau "DRIVER=Microsoft Excel Driver (*.xls *.xlsx,
    *.xlsm, *.xlsb)"
channel3 <- odbcConnectExcel("hills.xls")
```

Paket RODBC - query

- Syntax

```
sqlQuery(channel, query, errors = TRUE, ...,
    rows_at_time)
sqlGetResults(channel, as.is = FALSE, errors
    = FALSE, max = 0, buffsize = 1000,
    nullstring = NA_character_, na.strings =
    "NA", believeNRows = TRUE, dec =
    getOption("dec"), stringsAsFactors =
    default.stringsAsFactors())
```

- Ilustrasi

```
sqlQuery(channel, paste("select State,  Murder
    from USArrests", "where Rape > 30 order by
    Murder"))
```

Paket RODBC - fetch

- Syntax

```
sqlFetch(channel, sqtable, ..., colnames
   = FALSE, rownames = TRUE)
sqlFetchMore(channel, ..., colnames =
   FALSE, rownames = TRUE)
```

- Ilustrasi

```
sqlFetch(channel, "USArrests")
```

Paket RODBC - save

- Syntax

```
sqlSave(channel, dat, tablename = NULL, append =
  FALSE, rownames = TRUE, colnames = FALSE,
  verbose = FALSE, safer = TRUE, addPK = FALSE,
  typeInfo, varTypes, fast = TRUE, test = FALSE,
  nastring = NULL)
```

- Ilustrasi

```
sqlSave(channel, USArrests, rownames = "state",
    addPK=TRUE)
```

Paket RSQLite

Menciptakan Database Baru

- Menggunakan `dbConnect()`:

```
mydb <- dbConnect(RSQLite::SQLite(), "my-
db.sqlite")

dbDisconnect(mydb)
```

- Jika database sementara, gunakan "" (on-disk database) atau ":memory:" atau "file::memory:" (in-memory database). Database akan terhapus secara otomatis setelah disconnect.

```
mydb <- dbConnect(RSQLite::SQLite(), "")

dbDisconnect(mydb)
```

Mengcopy data frame

- Gunakan `dbWriteTable()`:

```
mydb <- dbConnect(RSQLite::SQLite(), "")
dbWriteTable(mydb, "mtcars", mtcars)
dbWriteTable(mydb, "iris", iris)
dbListTables(mydb)
#> [1] "iris"   "mtcars"
```

Queries

- Menggunakan `dbGetQuery()`:

```
dbGetQuery(mydb, 'SELECT * FROM mtcars LIMIT 5')
#>    mpg cyl disp  hp drat    wt  qsec vs am gear carb
#> 1 21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
#> 2 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
#> 3 22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
#> 4 21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
#> 5 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
```

Query

- Tidak semua nama variabel R valid dengan nama variabel SQL, dibutuhkan "namavar":

```
dbGetQuery(mydb, 'SELECT * FROM iris WHERE "Sepal.Length" < 4.6')
#>   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
#> 1          4.4         2.9          1.4         0.2  setosa
#> 2          4.3         3.0          1.1         0.1  setosa
#> 3          4.4         3.0          1.3         0.2  setosa
#> 4          4.5         2.3          1.3         0.3  setosa
#> 5          4.4         3.2          1.3         0.2  setosa
```

Query

- Cara lain yang aman menggunakan params:

```
dbGetQuery(mydb, 'SELECT * FROM iris WHERE "Sepal.Length" < :x',
params = list(x = 4.6))
```

```
#>   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
#> 1          4.4         2.9          1.4         0.2  setosa
#> 2          4.3         3.0          1.1         0.1  setosa
#> 3          4.4         3.0          1.3         0.2  setosa
#> 4          4.5         2.3          1.3         0.3  setosa
#> 5          4.4         3.2          1.3         0.2  setosa
```

Batched queries

- Jika hasil query tidak cukup dalam memory, dapat digunakan `dbSendQuery`(), `dbFetch`() dan `dbClearResults`().
- Default `dbFetch`() akan membaca semua baris:

  → gunakan n untuk maksimum baris

```
rs <- dbSendQuery(mydb, 'SELECT * FROM mtcars')
while (!dbHasCompleted(rs)) {
  dat.f <- dbFetch(rs, n = 10)
  print(nrow(dat.f))
}
#> [1] 10
#> [1] 10
#> [1] 10
#> [1] 2
dbClearResult(rs)
```

Multiple parameterised queries

- Dapat dilakukan hal yang sama pada parameterised query dengan parameter berbeda. Call `dbBind()` :

```
rs <- dbSendQuery(mydb, 'SELECT * FROM iris WHERE
"Sepal.Length" < :x')

dbBind(rs, param = list(x = 4.5))

nrow(dbFetch(rs))

#> [1] 4

dbBind(rs, param = list(x = 4))

nrow(dbFetch(rs))

#> [1] 0

dbClearResult(rs)
```

Multiple parameterised queries

- Dapat di lewati multiple parameters dalam satu panggilan `dbBind()`:

```
rs <- dbSendQuery(mydb, 'SELECT * FROM iris WHERE
"Sepal.Length" = :x')

dbBind(rs, param = list(x = seq(4, 4.4, by =
0.1)))

nrow(dbFetch(rs))

#> [1] 4

dbClearResult(rs)
```

Statements

- DBI memiliki fungsi `dbSendStatement`() and `dbExecute`(), yang merupakan counterparts dari `dbSendQuery`() and `dbGetQuery`() untuk statement SQL yang tidak menghasilkan tabular

```
dbExecute(mydb, 'DELETE FROM iris WHERE
"Sepal.Length" < 4')

#> [1] 0

rs <- dbSendStatement(mydb, 'DELETE FROM iris
WHERE "Sepal.Length" < :x')

dbBind(rs, param = list(x = 4.5))

dbGetRowsAffected(rs)

#> [1] 4

dbClearResult(rs)
```

SQL

Structured Query Language

SQL

```
SELECT columns or computations
    FROM table
    WHERE condition
    GROUP BY columns
    HAVING condition
    ORDER BY column [ASC | DESC]
    LIMIT offset,count;
```

Ilustrasi

- SELECT * FROM tablename;
- SELECT var1,var2,var2/var1 from tablename;
- SELECT var1,var2,var2/var1 AS ratio FROM tablename;
- SELECT * FROM tablename WHERE var1 > 10 AND var2 < var1;
- SELECT var1,var2,var2/var1 AS ratio FROM tablename HAVING ratio > 10;

Agregat

- SELECT type, AVG(x) AS mean FROM table GROUP BY type;

Task	SQL aggregation function
Count numbers of occurrences	`COUNT()`
Find the mean	`AVG()`
Find minimum	`MIN()`
Find maximum	`MAX()`
Find variance	`VAR_SAMP()`
Find standard deviation	`STDDEV_SAMP()`

Selesai...