D01

讀取大型資料 in R

吳漢銘
國立政治大學 統計學系

https://hmwu.idv.tw

大綱

- 記憶體設置、 物件大小、計算執行(資料讀取)時間
- Handling Large Data Sets in R
- 讀取目錄下符合目標的(多個)檔案資料: list.files
- 直接讀取壓縮檔(zip)內之檔案
- 讀取HTML網頁表格，讀取XML表格
- 讀取影像檔案
- 從資料庫(MySQL)讀取資料
- GREA: read ALL the data into R/Importing Data with RStudio
- 讀取部份資料進入R計算(readbulk)
- fread {data.table}: Fast and friendly file finagler
- 讀取檔案部份欄位資料
- 如何讓read.table讀較大的資料速度更快

http://www.hmwu.idv.tw/index.php/r-software

Memory Allocation in R

- 當R啟動時，設定最大可獲得的記憶體:

```
"C:\Program Files\R\R-3.2.2\bin\x64\Rgui.exe" --max-mem-size=2040M
```

  - 最小需求是32MB.
  - R啟動後僅可設定更高值，不能再用**memory.limit**設定較低的值。

```
> report.memory <- function(size = 4095){
+   cat("current memory in use: ", memory.size(max = FALSE), "Mb \n")
+   cat("maximum memory obtained from the OS: ", memory.size(max = TRUE), "Mb \n")
+   cat("current memory limit: ", memory.size(max = NA), "Mb \n")
+   cat("current memory limit: ", memory.limit(size = NA), "Mb \n")
+   cat("increase memory limit: ", memory.limit(size = size), "Mb \n")
+ }
>
> report.memory()
current memory in use:  686.74 Mb
maximum memory obtained from the OS:  1558.81 Mb
current memory limit:  65408.91 Mb
current memory limit:  65408 Mb
increase memory limit:  65408 Mb
Warning message:
In memory.limit(size = size) : 無法減少記憶體限制：已忽略
```

R與Windows作業系統

最大可獲得的記憶體

- 32-bit R + 32-bit Windows: 2GB.
- 32-bit R + 64-bit Windows: 4GB.
- 64-bit R + 64-bit Windows: 8TB.

Report the Space Allocated for an Object: object.size{utils}

- 儲存R物件所佔用的記憶體估計。

```
object.size(x)
print(object.size(x), units = "Mb")
```

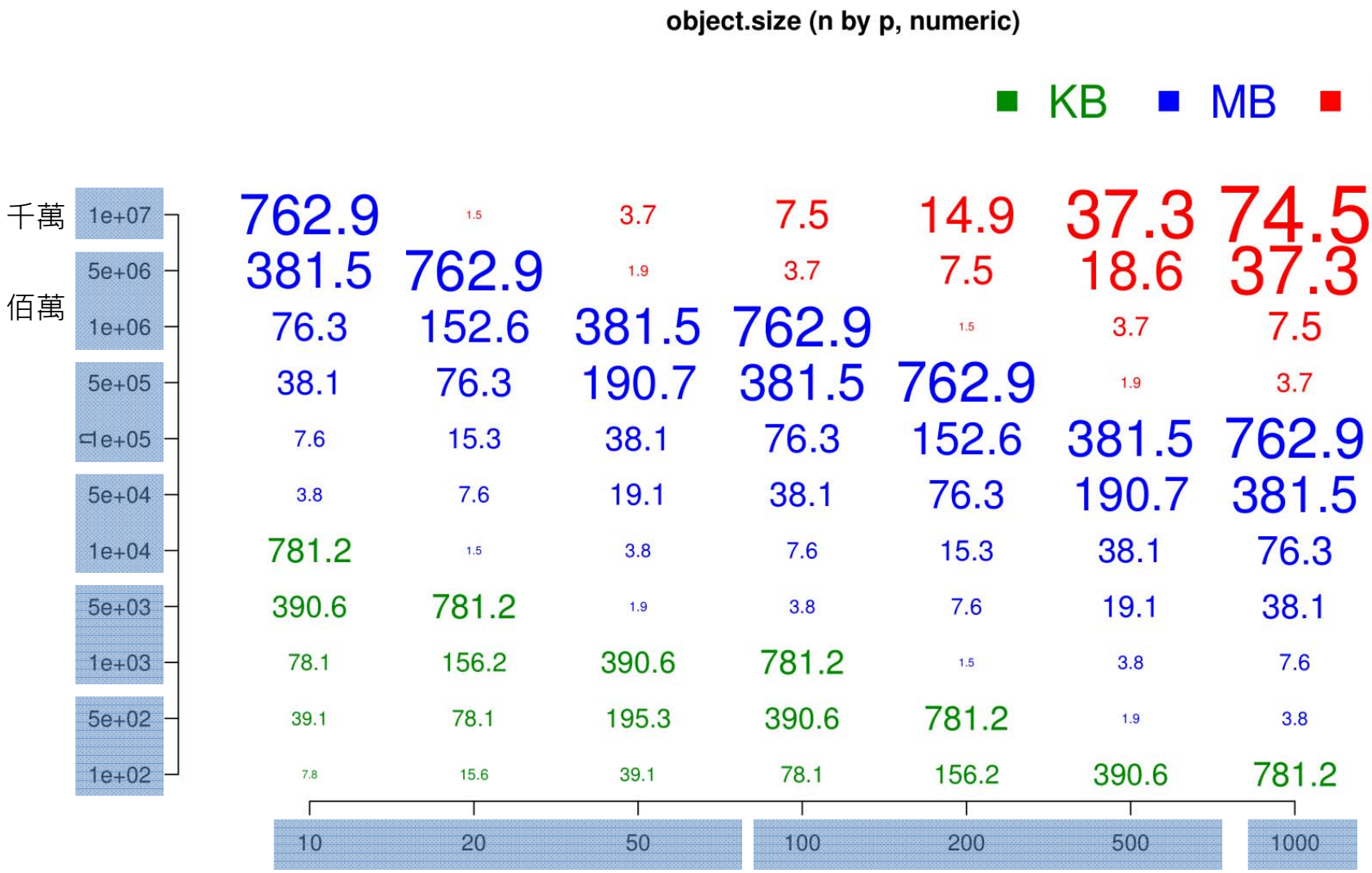
```
> n <- 10000
> p <- 200
> myData <- as.data.frame(matrix(rnorm(n*p), ncol = p, nrow=n))
> print(object.size(myData), units = "Mb")
15.3 Mb

> write.table(myData, "myData.txt") ## 約 34.7 MB

> InData <- read.table("myData.txt")
> print(object.size(InData), units = "Mb")
15.6 Mb
```

NOTE: Under any circumstances, you cannot have more than $2^{31}-1=2,147,483,647$ rows or columns.

object.size{utils}



(n*p*8)/(1024*1024) MB

1 Bit = Binary Digit; 8 Bits = 1 Byte; 1024 Bytes = 1 Kilobyte; 1024 Kilobytes = 1 Megabyte
1024 Megabytes = 1 Gigabyte; 1024 Gigabytes = 1 Terabyte; 1024 Terabytes = 1 Petabyte

Measuring execution time: system.time{base}

```
myFun <- function(n){
    for(i in 1:n){
        x <- x + i
    }
    x
}
```

```
> start.time <- Sys.time()
> ans <- myFun(10000)
> end.time <- Sys.time()
> end.time -start.time
Time difference of 0.0940001 secs
```

```
> system.time({
+     ans <- myFun(10000)
+ })
   user  system elapsed
   0.04    0.00    0.05
```

See also : microbenchmark, rbenchmark packages

```
myPlus <- function(n){
    x <- 0
    for(i in 1:n){
        x <- x + sum(rnorm(i))
    }
    x
}
```

```
myProduct <- function(n){
    x <- 1
    for(i in 1:n){
        x <- x * sum(rt(i, 2))
    }
    x
}
```

```
> system.time({
+     a <- myPlus(5000)
+ })
   user  system elapsed
   3.87    0.00    3.91
> system.time({
+     b <- myProduct(5000)
+ })
   user  system elapsed
  10.36    0.00   10.42
```

Handling Large Data Sets in R

- **The Problem with large data sets in R:**
  - R reads entire data set into RAM all at once.
  - R Objects live in memory entirely.
  - Does not have int64 datatype.
  - Not possible to index objects with huge numbers of rows & columns even in 64 bit systems (2 Billion vector index limit) .

類型名稱	位元組	其他名稱	值的範圍
_int32	4	signed、signed int、int	–2,147,483,648 到 2,147,483,647
_int64	8	long long、signed long long	–9,223,372,036,854,775,808 到 9,223,372,036,854,775,807
double	8	無	1.7E +/- 308 (15 位數)

https://msdn.microsoft.com/zh-tw/library/s3f49ktz.aspx

- **How big is a large data set:**
  - Medium sized files that can be loaded in R (within memory limit but processing is cumbersome (typically in the 1~2 GB range).
  - Large files that cannot be loaded in R due to R/OS limitations.
    - Large files (typically 2 ~ 10 GB) that can still be processed locally using some work around solutions.
    - Very Large files (> 10 GB) that needs distributed large scale computing.

Handling large data sets in R, Sundar Pradeep & Philip Moy, **April 10, 2015**
https://rstudio-pubs-static.s3.amazonaws.com/72295_692737b667614d369bd87cb0f51c9a4b.html

Strategy for Medium sized datasets (< 2 GB)

- Reduce the size of the file before loading it into R (select some columns).
- Pre-allocate number of rows (**`nrows`**) and pre-define column classes (**`colClasses`**), define **`comment.char`** parameter
- Use **`fread {data.table}`**.
- Use pipe operators to overwrite files with intermediate results and minimize data set duplication through process steps.
- Parallel Processing
  - Explicit Parallelism (user controlled): **`rmpi`** (Message Processing Interface), **`snow`** (Simple Network of Workstations)
  - Implicit parallelism (system abstraction): **`doMC`** (Foreach Parallel Adaptor for 'parallel'), **`foreach`** (Provides Foreach Looping Construct for R).

Handling large data sets in R, Sundar Pradeep & Philip Moy, **April 10, 2015**
https://rstudio-pubs-static.s3.amazonaws.com/72295_692737b667614d369bd87cb0f51c9a4b.html

Strategy for Medium sized datasets (2 ~10 GB) and Very Large datasets (> 10GB)

- **Medium sized datasets (2 ~ 10 GB)**
  - For medium sized data sets which are too-big for in-memory processing but too-small-for-distributed-computing files, following R Packages come in handy.
  - **`bigmemory`**: Manage Massive Matrices with Shared Memory and Memory-Mapped Files (http://www.bigmemory.org/)
  - **`ff`**: memory-efficient storage of large data on disk and fast access functions (http://ff.r-forge.r-project.org/)

- **Very Large datasets (> 10GB)**
  - Use integrated environment packages like **RHipe** to leverage **Hadoop MapReduce** framework.
  - Use **`RHadoop`** directly on hadoop distributed system. (https://github.com/RevolutionAnalytics/RHadoop/wiki)
  - Storing large files in databases and connecting through **DBI/ODBC** calls from R is also an option worth considering.

Handling large data sets in R, Sundar Pradeep & Philip Moy, **April 10, 2015**
https://rstudio-pubs-static.s3.amazonaws.com/72295_692737b667614d369bd87cb0f51c9a4b.html

11 Tips on How to Handle Big Data in R

1. Think in vectors: avoid for-loops if possible.
2. Use the **`data.table`** package.
3. Read csv-files with the **`fread`** function instead of **`read.csv`** (**`read.table`**).
4. Parse POSIX dates with the very fast package **`fasttime`**.
5. Avoid copying **`data.frames`** and remove, **`rm(yourdatacopy)`**.
6. Merge **`data.frames`** with the superior **`rbindlist {data.table}`**.
7. Use the **`stringr`** package instead of the regular expressions
8. Use the **`bigvis`** package for visualising big data sets.
9. Use a random sample for your exploratory analysis or to test code.
10. **`read.csv()`** for example has a **`nrows`** option, which only reads the first x number of lines.
11. Export your data set directly as gzip.

Fig Data: 11 Tips on How to Handle Big Data in R (and 1 Bad Pun) (2013-07-18 by Ulrich Atz)
https://theodi.org/blog/fig-data-11-tips-how-handle-big-data-r-and-1-bad-pun

讀取目錄下符合目標的資料檔案: list.files

```
list.files(path = ".", pattern = NULL, all.files = FALSE,
           full.names = FALSE, recursive = FALSE,
           ignore.case = FALSE, include.dirs = FALSE, no.. = FALSE)
```

score_quiz1.txt

```
"gender" "Calculus" "LinearAlgebra" "BasicMath" "Rprogramming" "English"
"student.5" "M" 69 93 83 79 95
"student.4" "M" 70 78 31 26 69
"student.9" "F" 57 26 21 99 51
"student.2" "F" 73 32 73 76 37
"student.6" "F" 56 6 50 98 33
"student.3" "M" 62 4 73 22 4
"student.7" "F" 76 5 8 95 62
"student.10" "F" 68 63 43 15 97
"student.8" "M" 73 97 23 60 66
```

10位學生(student.1~student.10)自由參加5次小考，各次小考之5科成績("Calculus" "LinearAlgebra" "BasicMath" "Rprogramming" "English")分別紀錄於5個檔案中。

```
> getwd() # setwd("F:/my_R")
[1] "D:/R/data/quiz"
> list.files() # dir()
[1] "score_quiz1.txt" "score_quiz2.txt" "score_quiz3.txt" "score_quiz4.txt"
[5] "score_quiz5.txt"
> list.dirs()
[1] "."
> (filenames <- list.files(".", pattern="*.txt"))
[1] "score_quiz1.txt" "score_quiz2.txt" "score_quiz3.txt" "score_quiz4.txt"
"score_quiz5.txt"
```

讀取多個資料檔案並計算摘要

```
> (quiz.data <- lapply(filenames, read.table))
[[1]]
           gender Calculus LinearAlgebra BasicMath Rprogramming English
student.5       M       69            93        83            79      95
student.4       M       70            78        31            26      69
...
[[5]]
          gender Calculus LinearAlgebra BasicMath Rprogramming English
student.3      M       53           100       100            33       1
student.8      M       81            69        75            86      27
...
> (quiz.data.summary <- lapply(quiz.data, summary))
[[1]]
 gender    Calculus      LinearAlgebra     BasicMath    Rprogramming       English
 F:5    Min.   :56.00   Min.   : 4.00   Min.   : 8   Min.   :15.00   Min.   : 4.00
 M:4    1st Qu.:62.00   1st Qu.: 6.00   1st Qu.:23   1st Qu.:26.00   1st Qu.:37.00
        Median :69.00   Median :32.00   Median :43   Median :76.00   Median :62.00
        Mean   :67.11   Mean   :44.89   Mean   :45   Mean   :63.33   Mean   :57.11
        3rd Qu.:73.00   3rd Qu.:78.00   3rd Qu.:73   3rd Qu.:95.00   3rd Qu.:69.00
        Max.   :76.00   Max.   :97.00   Max.   :83   Max.   :99.00   Max.   :97.00
...

[[5]]
 gender    Calculus    LinearAlgebra   BasicMath      Rprogramming     English
 F:3    Min.   :53.0   Min.   : 14   Min.   :  2.0   Min.   : 9.0   Min.   : 1
 M:2    1st Qu.:53.0   1st Qu.: 15   1st Qu.: 39.0   1st Qu.:12.0   1st Qu.: 6
        Median :70.0   Median : 32   Median : 55.0   Median :33.0   Median :27
        Mean   :66.2   Mean   : 46   Mean   : 54.2   Mean   :43.8   Mean   :31
        3rd Qu.:74.0   3rd Qu.: 69   3rd Qu.: 75.0   3rd Qu.:79.0   3rd Qu.:41
        Max.   :81.0   Max.   :100   Max.   :100.0   Max.   :86.0   Max.   :80
> names(quiz.data.summary) <- filenames
> quiz.data.summary$score_quiz2.txt
```

課堂練習

- 合併此5組資料使成一資料表格，並新增一 變數「小考次別(quiz.id)」
- 10位學生各參加哪幾次的小考?
- 各次小考，每科平均及變異數為多少? (未參加的同學不列入計算)
- 若此學期5次小考配分比重為(0.1, 0.1, 0.2, 0.2, 0.3)，試計算每位同學各科小考平均及變異數?
- 每位同學每科皆刪除最差的一次成績，試計算每位同學各科小考平均及變異數?
- 男女生各科小考平均及變異數為多少?

- 試讀取單數次小考成績檔案進入R。

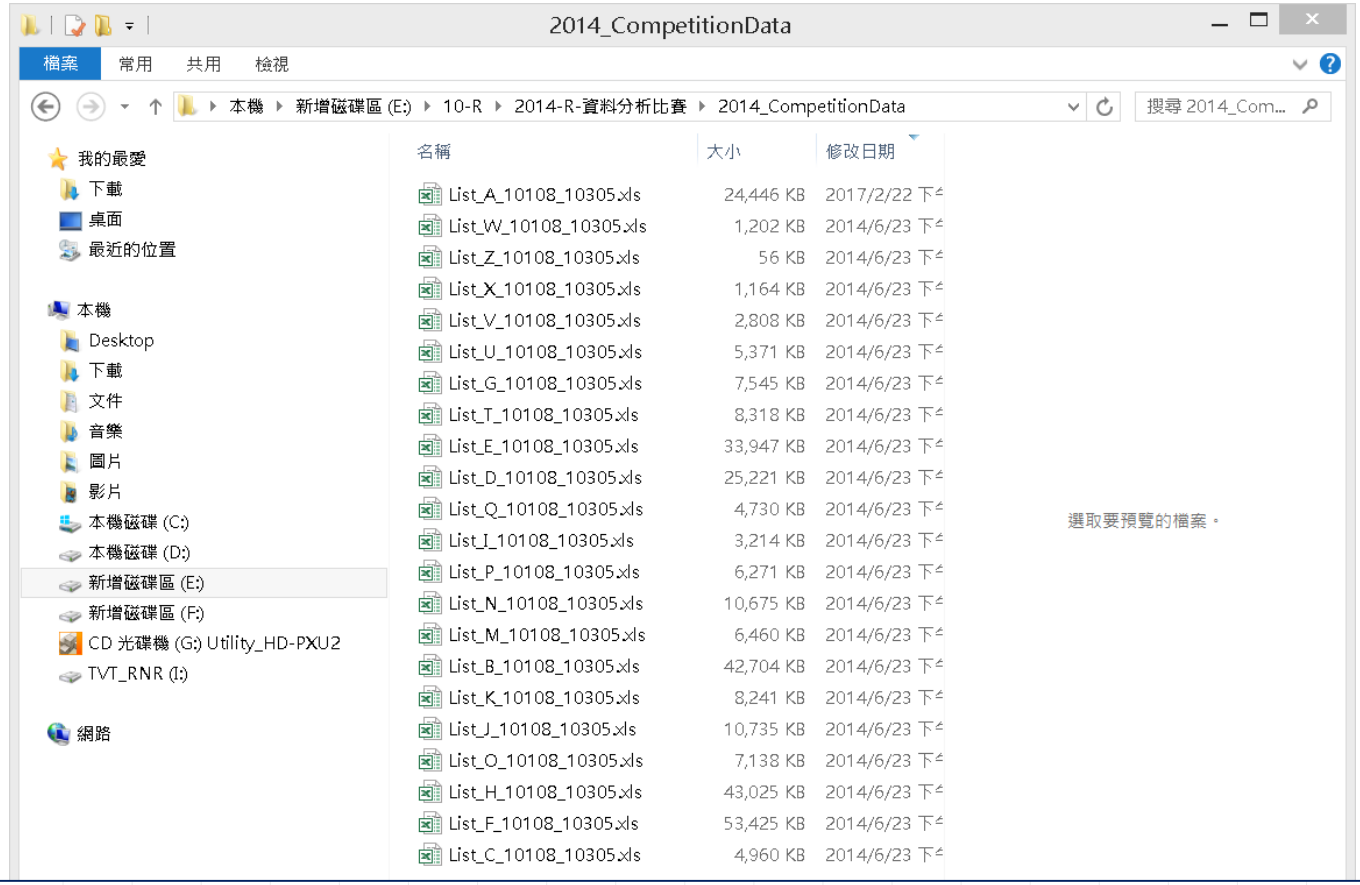
範例: 房屋實價登錄資料

2014年臺灣資料分析競賽資料 (使用R軟體):
大約 682724筆紀錄，28個變數

	檔案(xls)	縣市別	資料筆數	欄位數
1	List_A	臺北市	54111	28
2	List_B	臺中市	94683	28
3	List_C	基隆市	10833	28
4	List_D	臺南市	54643	28
5	List_E	高雄市	74565	28
6	List_F	新北市	119719	28
7	List_G	宜蘭縣	16104	28
8	List_H	桃園縣	95612	28
9	List_I	嘉義市	6840	28
10	List_J	新竹縣	23399	28
11	List_K	苗栗縣	17515	28
12	List_M	南投縣	13651	28
13	List_N	彰化縣	22613	28
14	List_O	新竹市	15664	28
15	List_P	雲林縣	13171	28
16	List_Q	嘉義縣	9847	28
17	List_T	屏東縣	17583	28
18	List_U	花蓮縣	11327	28
19	List_V	臺東縣	5825	28
20	List_W	金門縣	2509	28
21	List_X	澎湖縣	2423	28
22	List_Z	連江縣	87	28

1	鄉鎮市區	大安區	松山區
2	交易標的	房地(土地+建物)	房地(土地+建物)
3	土地區段位置/建物區段門牌	臺北市大安區和平東路xxx	臺北市松山區三民路xxx
4	土地移轉總面積(平方公尺)	19.39	35.53
5	使用分區或編定	住	住
6	非都市土地使用分區		
7	非都市土地使用地		
8	交易年月	10106	10107
9	交易筆棟數	土地1建物2車位0	土地1建物1車位0
10	移轉層次	五層	七層
11	總樓層數	017	007
12	建物型態	住宅大樓(11層含以上有電梯)	華廈(10層含以下有電梯)
13	主要用途	國民住宅	國民住宅
14	主要建材	鋼筋混凝土造	鋼筋混凝土造
15	建築完成年月	0740522	0810303
16	建物移轉總面積(平方公尺)	100.98	146.66
17	建物現況格局-房	3	3
18	建物現況格局-廳	2	2
19	建物現況格局-衛	1	2
20	建物現況格局-隔間	有	有
21	有無管理組織	有	有
22	總價(元)	18680000	25800000
23	單價(元/平方公尺)	184999	175917
24	車位類別		
25	車位移轉總面積(平方公尺)	0	0
26	車位總價(元)	0	0
27	交易標的橫坐標	305057	306980
28	交易標的縱坐標	2768793	2771975

範例: 房屋實價登錄資料



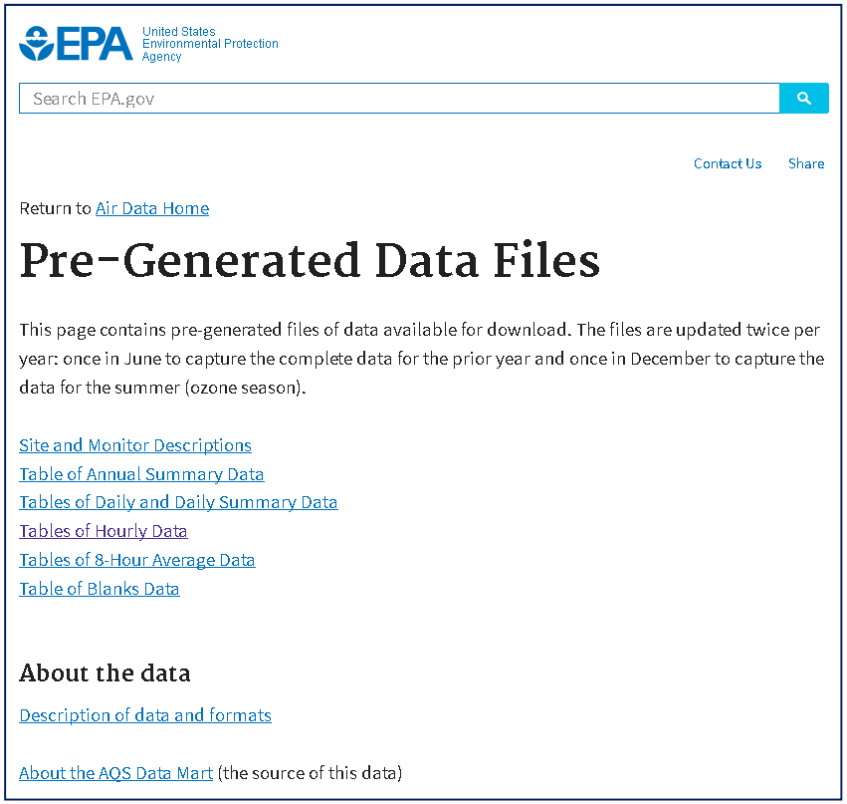
	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB
1	土地移轉総	使用分區或	非都市土地	非都市土地	交易年月	交易筆棟數	移轉層次	總樓層數	建物型態	主要用途	主要建材	建築完成年	建物移轉総	建物現況格	建物現況格	建物現況格	建物現況格	有無管理組	總價(元)	單價(元/平	車位類別	車位移轉総	車位總價(元	交易標的(	交易標的(
2	19.39	住			10106	土地1建物	五層	017	住宅大樓(含	國民住宅	鋼筋混凝土	0740522	100.98	3	2	1	有	有	1.9E+07	184999		0	0	305057	2768793
3	35.53	住			10107	土地1建物	七層	007	華廈(10層	國民住宅	鋼筋混凝土	0810303	146.66	3	2	2	有	有	2.6E+07	175917		0	0	306980	2771975
4	8.46	商			10107	土地3建物	九層	012	辦公商業大	商業用	鋼筋混凝土	0710408	93.42	0	0	0	有	有	2E+07	217307		0	0	303677	2770528
5	5.5	其他			10107	土地1建物			其他				0	0	0	0	有	無	132096	24017		0	0	301660	2772973
6	3.88	商			10107	土地4建物	六層	011	住宅大樓(含	商業用	鋼筋混凝土	0651207	36.74	1	1	1	有	有	4200000	114317		0	0	301888	2771743
7	32.41	住			10107	土地1建物	三層	005	公寓(5樓含	住家用	鋼筋混凝土	0691114	104.11	3	1	1	有	無	1.4E+07	134473		0	0	309098	2773459
8	9.37	其他			10107	土地3建物			其他				0	0	0	0	有	無	255000	27213		0	0	309231	2770317
9	1.02	其他			10107	土地1建物			其他				0	0	0	0	有	無	50000	49073		0	0	307197	2772172
10	1433.67	其他			10106	土地1建物			其他				0	0	0	0	有	無	9000000	6278		0	0	305450	2778288
11	10	住			10107	土地1建物	二層	006	套房(1房1	住家用	鋼筋混凝土	0860526	37.33	1	1	1	有	有	5960000	159670		0	0	304408	2772036
12	4.2	其他			10107	土地1建物			其他				0	0	0	0	有	無	1900000	452381		0	0	303843	2771492

Air Pollution Dataset from EPA

- **Dataset:** an air pollution (hourly ozone levels) dataset from the U.S. Environmental Protection Agency (EPA) for the year 2014.
  http://aqsdr1.epa.gov/aqsweb/aqstmp/airdata/download_files.html

- U.S. EPA on hourly ozone measurements in the entire U.S. for the year 2014. The data are available from the EPA' s Air Quality System web page.

- The dataset is a comma-separated value (CSV) file, where each row of the file contains one hourly measurement of ozone at some location in the country.



Hourly Data

Hourly Data

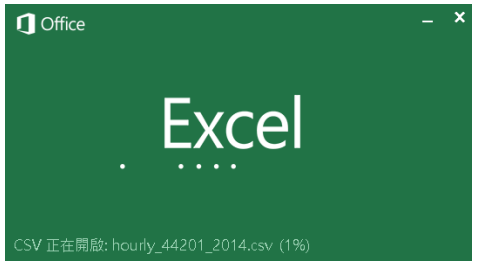
Criteria Gases

Year	Ozone (44201)	SO2 (42401)	CO (42101)	NO2 (
2015	hourly_44201_2015.zip 1,575,854 Rows 11,553 KB As of 2015-06-20	hourly_42401_2015.zip 813,370 Rows 5,510 KB As of 2015-06-20	hourly_42101_2015.zip 468,398 Rows 3,415 KB As of 2015-06-20	hourly_ 67 As of
2014	hourly_44201_2014.zip 8,967,571 Rows 66,326 KB As of 2015-06-20	hourly_42401_2014.zip 3,724,805 Rows 24,719 KB As of 2015-06-20	hourly_42101_2014.zip 2,457,531 Rows 16,890 KB As of 2015-06-20	hourly_ 3,3 As of

Hourly Data

Criteria Gases

Year	Ozone (44201)	SO2 (42401)	CO (42101)	NO2 (42602)
2016	hourly_44201_2016.zip 7,027,694 Rows 52,377 KB As of 2016-12-23	hourly_42401_2016.zip 2,709,398 Rows 17,768 KB As of 2016-12-23	hourly_42101_2016.zip 1,718,205 Rows 12,307 KB As of 2016-12-23	hourly_42602_2016.zip 2,554,440 Rows 19,669 KB As of 2016-12-23
2015	hourly_44201_2015.zip 9,071,460 Rows 67,419 KB As of 2016-12-23	hourly_42401_2015.zip 3,762,681 Rows 24,789 KB As of 2016-12-23	hourly_42101_2015.zip 2,454,720 Rows 17,435 KB As of 2016-12-23	hourly_42602_2015.zip 3,560,477 Rows 27,234 KB As of 2016-12-23
2014	hourly_44201_2014.zip 9,096,553 Rows 67,594 KB As of 2016-12-23	hourly_42401_2014.zip 3,763,752 Rows 24,960 KB As of 2016-12-23	hourly_42101_2014.zip 2,474,722 Rows 17,312 KB As of 2016-12-23	hourly_42602_2014.zip 3,429,015 Rows 25,960 KB As of 2016-12-23



dataset:
hourly_44201_2014.zip (64.7M)
hourly_44201_2014.csv (1.89G)

	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	Parameter N	Date Local	Time Local	Date GMT	Time GMT	Sample Mea	Units of Me	MDL	Uncertainty	Qualifier	Method Typ	Method Coc	Method Nar	State Name	County Nan	Date of Last Change		
1048572	Ozone	2014/1/22	19:00	2014/1/23	03:00	0.002	Parts per mi	0.005			FEM	87	INSTRUMI	California	Merced	2014/8/4		
1048573	Ozone	2014/1/22	21:00	2014/1/23	05:00	0.002	Parts per mi	0.005			FEM	87	INSTRUMI	California	Merced	2014/8/4		
1048574	Ozone	2014/1/22	22:00	2014/1/23	06:00	0.012	Parts per mi	0.005			FEM	87	INSTRUMI	California	Merced	2014/8/4		
1048575	Ozone	2014/1/22	23:00	2014/1/23	07:00	0.013	Parts per mi	0.005			FEM	87	INSTRUMI	California	Merced	2014/8/4		
1048576	Ozone	2014/1/23	00:00	2014/1/23	08:00	0.014	Parts per mi	0.005			FEM	87	INSTRUMI	California	Merced	2014/8/4		

(limited to 1,048,576 rows)

Hourly Data

hourly_44201_2014.csv

```
1  "State Code","County Code","Site Num","Parameter Code","POC","Latitude","Longitude","Datum","Parameter Name","Date Local","Time Local","Date GMT"
2  "01","003","0010","44201",1,30.498001,-87.881412,"NAD83","Ozone","2014-03-01","01:00","2014-03-01","07:00",0.047,"Parts per million",0.005,"","",
3  "01","003","0010","44201",1,30.498001,-87.881412,"NAD83","Ozone","2014-03-01","02:00","2014-03-01","08:00",0.047,"Parts per million",0.005,"","",
4  "01","003","0010","44201",1,30.498001,-87.881412,"NAD83","Ozone","2014-03-01","03:00","2014-03-01","09:00",0.043,"Parts per million",0.005,"","",
5  "01","003","0010","44201",1,30.498001,-87.881412,"NAD83","Ozone","2014-03-01","04:00","2014-03-01","10:00",0.038,"Parts per million",0.005,"","",
6  "01","003","0010","44201",1,30.498001,-87.881412,"NAD83","Ozone","2014-03-01","05:00","2014-03-01","11:00",0.035,"Parts per million",0.005,"","",
7  "01","003","0010","44201",1,30.498001,-87.881412,"NAD83","Ozone","2014-03-01","06:00","2014-03-01","12:00",0.035,"Parts per million",0.005,"","",
8  "01","003","0010","44201",1,30.498001,-87.881412,"NAD83","Ozone","2014-03-01","07:00","2014-03-01","13:00",0.034,"Parts per million",0.005,"","",
9  "01","003","0010","44201",1,30.498001,-87.881412,"NAD83","Ozone","2014-03-01","08:00","2014-03-01","14:00",0.037,"Parts per million",0.005,"","",
10 "01","003","0010","44201",1,30.498001,-87.881412,"NAD83","Ozone","2014-03-01","09:00","2014-03-01","15:00",0.044,"Parts per million",0.005,"","",
11 "01","003","0010","44201",1,30.498001,-87.881412,"NAD83","Ozone","2014-03-01","10:00","2014-03-01","16:00",0.046,"Parts per million",0.005,"","",

8967568 "80","006","0007","44201",1,31.7122,-106.3953,"NAD83","Ozone","2014-08-31","19:00","2014-09-01","02:00",0.019,"Parts per million",0.005,"","","FE
8967569 "80","006","0007","44201",1,31.7122,-106.3953,"NAD83","Ozone","2014-08-31","20:00","2014-09-01","03:00",0.021,"Parts per million",0.005,"","","FE
8967570 "80","006","0007","44201",1,31.7122,-106.3953,"NAD83","Ozone","2014-08-31","21:00","2014-09-01","04:00",0.024,"Parts per million",0.005,"","","FE
8967571 "80","006","0007","44201",1,31.7122,-106.3953,"NAD83","Ozone","2014-08-31","22:00","2014-09-01","05:00",0.002,"Parts per million",0.005,"","","FE
8967572 "80","006","0007","44201",1,31.7122,-106.3953,"NAD83","Ozone","2014-08-31","23:00","2014-09-01","06:00",0.002,"Parts per million",0.005,"","","FE
8967573
```

1.89 GB (2,034,887,869 字節)， 8,967,573 行。 TeX 行 8967573, 欄 1

There are 34 variables with 8967571 observations:

"State Code","County Code","Site Num","Parameter Code", "POC",
"Latitude","Longitude","Datum","Parameter Name","Date Local",
"Time Local","Date GMT","Time GMT","Sample Measurement","Units of Measure",
"MDL","Uncertainty","Qualifier","Method Type","Method Code",
"Method Name","State Name","County Name","Date of Last Change"

註: 如何呈現這些變數的內容及資訊?

直接讀取壓縮檔(zip)內之檔案

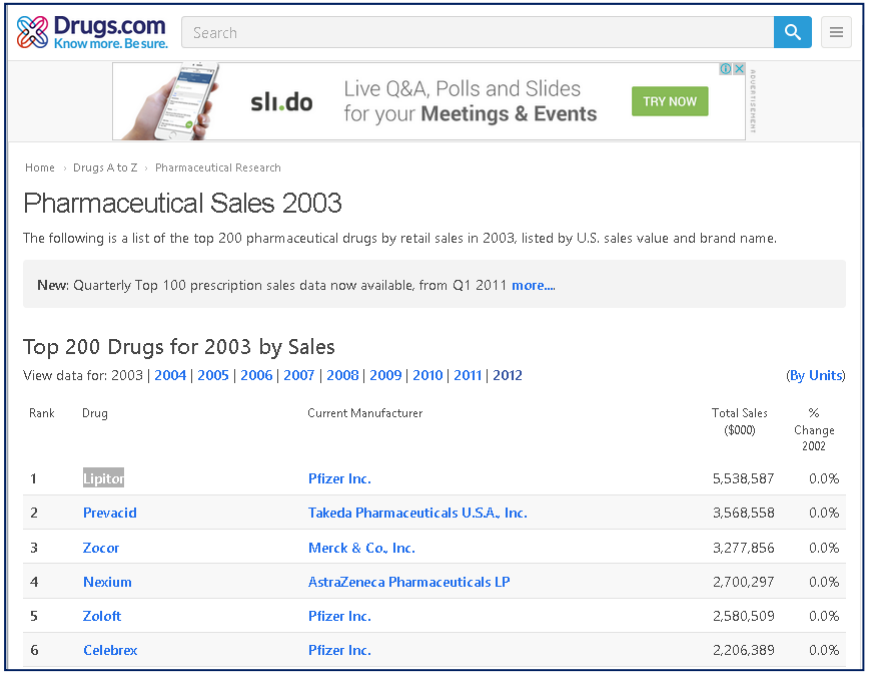
```
> library(readr)
> ozone <- read_csv("hourly_44201_2014.csv") # unzip and read
==================================================  |  96% 1900 MB
>
> # read without unzip
> # unz reads (only) single files within zip files, in binary mode.
> # The description is the full path to the zip file.
> # a zip file contains several files, create a connection to read one of the files
> # AirPollution-test.zip: hourly_44201_2014-test.csv, hourly_44201_2015-test.csv,
hourly_44201_2016-test.csv
>
> zz <- unz(description="AirPollution-test.zip", filename="hourly_44201_2014-test.csv")
> ozone.zip <- read.csv(zz, header=T)
> ozone.zip
   State.Code County.Code Site.Num Parameter.Code POC Latitude Longitude Datum Parameter.Name
1           1           3       10          44201   1 30.49748 -87.88026 NAD83          Ozone
2           1           3       10          44201   1 30.49748 -87.88026 NAD83          Ozone
...
> close(zz)
>
> # read from the web url
> location  <- "http://aqsdr1.epa.gov/aqsweb/aqstmp/airdata/hourly_44201_2014.zip"
> zz <- unz(location, filename="hourly_44201_2014.csv")
> ozone.url <- read.csv(zz, header=T)
> close(zz)
```

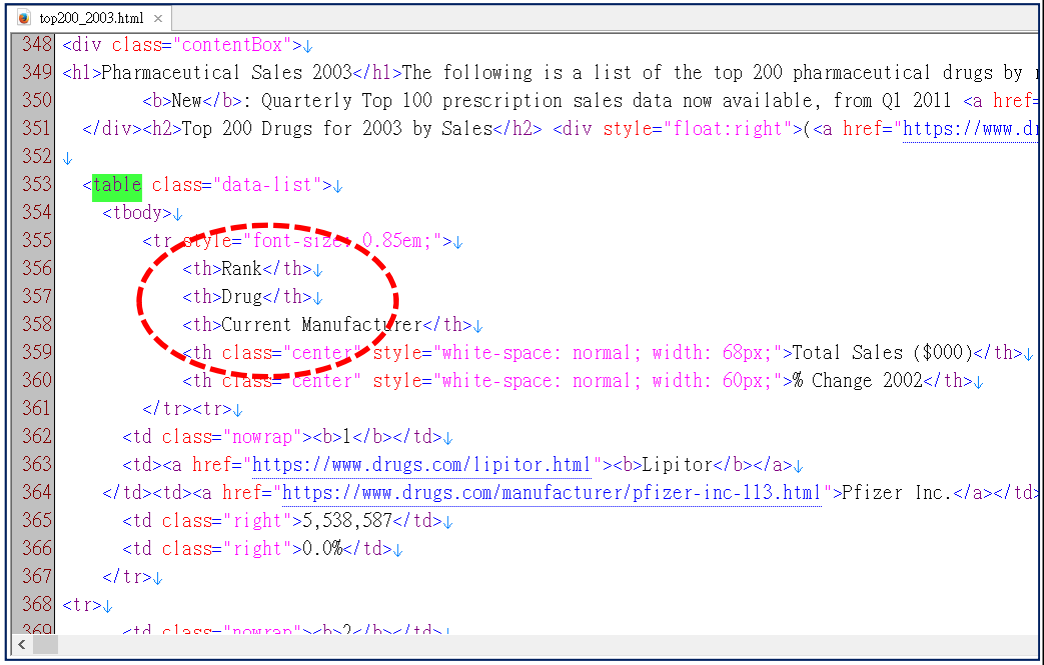
See also: `connections {base}: file(), url(), gzfile(), bzfile(), xzfile(), unz(), pipe()`

```
zz <- gzfile('file.csv.gz', 'rt')
mydata <- read.csv(zz, header=F)
```

讀取HTML網頁表格 (1)

https://www.drugs.com/top200_2003.html






```
> install.packages("XML", dep = T, repos="http://cran.csie.ntu.edu.tw")
> library(XML)
> library(RCurl)
```

```
readHTMLTable(doc, header = NA, colClasses = NULL, skip.rows = integer(),
              trim = TRUE, elFun = xmlValue, as.data.frame = TRUE,
              which = integer(), ...)
```

https://rstudio-pubs-static.s3.amazonaws.com/1776_dbaebbdbde8d46e693e5cb60c768ba92.html

讀取HTML網頁表格 (1)

```
> URL1 <- getURL("https://www.drugs.com/top200_2003.html")
> htmlTable1 <- readHTMLTable(URL1, header=T)
> str(htmlTable1)
> head(htmlTable1[[1]])
```

```
> str(htmlTable1)
List of 1
 $ NULL:'data.frame':   201 obs. of  5 variables:
  ..$ Rank                : Factor w/ 201 levels "","1","10","100",..: 2 113 125 136 147 158 169 180 191 3 ...
  ..$ Drug                : Factor w/ 201 levels "Abilify","Accupril",..: 108 140 194 120 196 43 198 119 67 9 ...
  ..$ Current Manufacturer: Factor w/ 46 levels "","Abbott Laboratories",..: 36 43 33 10 36 36 20 36 36 27 ...
  ..$ Total Sales ($000)  : Factor w/ 200 levels "0","1,046,145",..: 168 126 125 89 88 87 86 85 84 83 ...
  ..$ % Change 2002       : Factor w/ 1 level "0.0%": 1 1 1 1 1 1 1 1 1 1 ...
> head(htmlTable1[[1]])
  Rank     Drug                 Current Manufacturer Total Sales ($000) % Change 2002
1    1  Lipitor                       Pfizer Inc.          5,538,587          0.0%
2    2 Prevacid Takeda Pharmaceuticals U.S.A., Inc.          3,568,558          0.0%
3    3    Zocor                  Merck & Co., Inc.          3,277,856          0.0%
4    4   Nexium     AstraZeneca Pharmaceuticals  LP          2,700,297          0.0%
5    5   Zoloft                       Pfizer Inc.
6    6 Celebrex                       Pfizer Inc.
```

Top 200 Drugs for 2003 by Sales

View data for: 2003 | 2004 | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 | 2012 (By Units)

Rank	Drug	Current Manufacturer	Total Sales ($000)	% Change 2002
1	Lipitor	Pfizer Inc.	5,538,587	0.0%
2	Prevacid	Takeda Pharmaceuticals U.S.A., Inc.	3,568,558	0.0%
3	Zocor	Merck & Co., Inc.	3,277,856	0.0%
4	Nexium	AstraZeneca Pharmaceuticals LP	2,700,297	0.0%
5	Zoloft	Pfizer Inc.	2,580,509	0.0%
6	Celebrex	Pfizer Inc.	2,206,389	0.0%

讀取HTML網頁表格 (2)

https://en.wikipedia.org/wiki/List_of_countries_and_dependencies_by_population

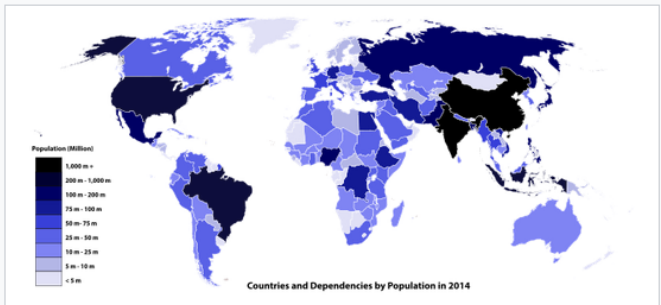
List of countries and dependencies by population

From Wikipedia, the free encyclopedia

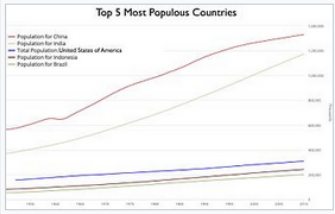
This article **needs additional citations for verification**. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed.
(January 2017) (Learn how and when to remove this template message)

This is a list of countries and dependent territories by population. It includes sovereign states, inhabited dependent territories, and, in some cases, constituent countries of sovereign states, with inclusion within the list being primarily based on the ISO standard ISO 3166-1. For instance, the United Kingdom is considered as a single entity while the constituent countries of the Kingdom of the Netherlands are considered separately. In addition, this list includes certain states with limited recognition not found in ISO 3166-1.

The population figures do not reflect the practice of



A map of world population in 2014



Top 5 most populous countries

Sovereign states and dependencies by population [edit]

Note: All dependent territories or constituent countries that are parts of sovereign states are shown in *italics*.

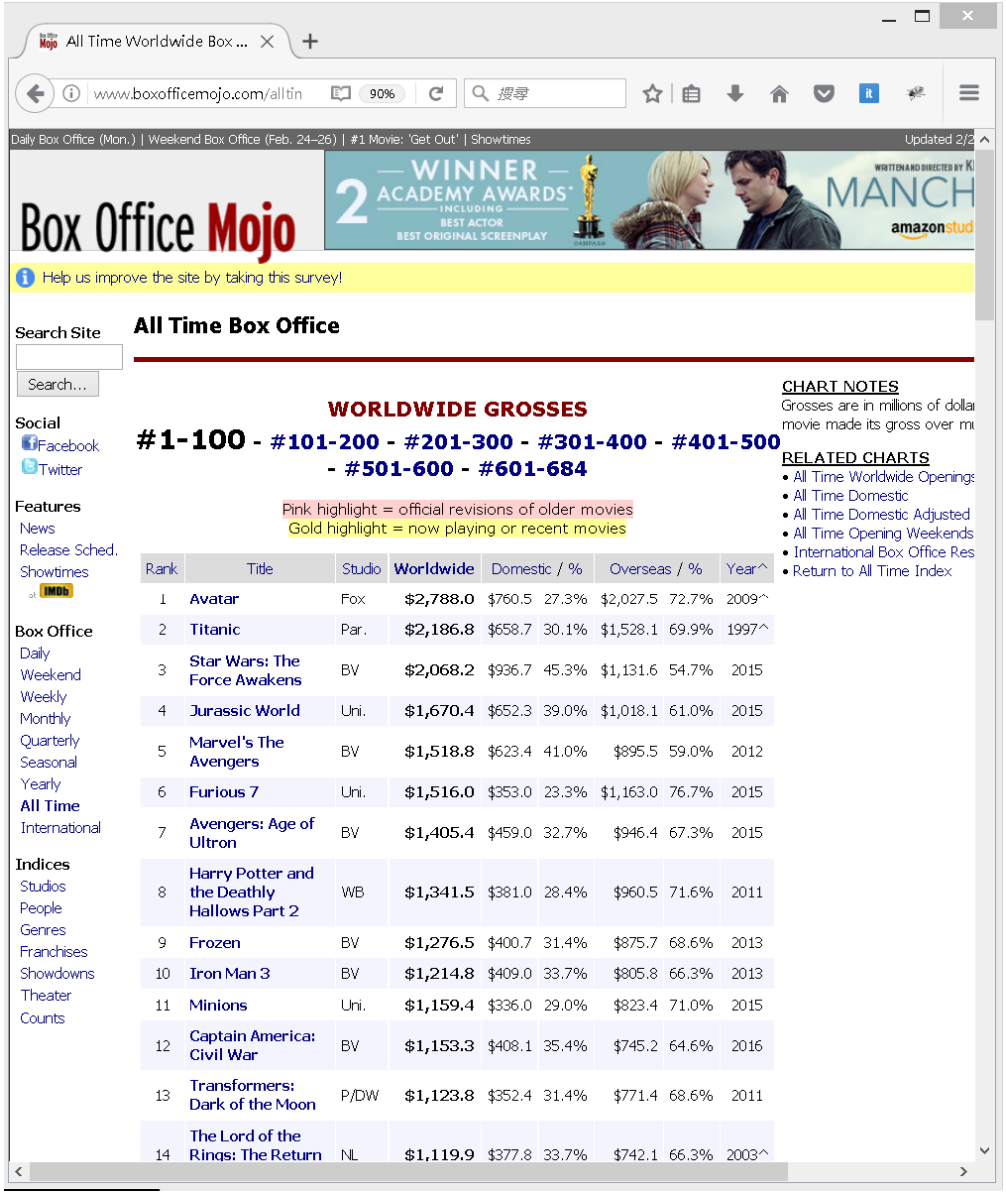
Rank	Country (or dependent territory)	Population	Date	% of world population	Source
1	China[Note 2]	1,381,680,000	February 22, 2017	18.5%	Official population clock
2	India	1,312,320,000	February 22, 2017	17.5%	Official population clock
3	United States[Note 3]	324,567,000	February 22, 2017	4.34%	Official population clock
4	Indonesia	260,581,000	July 1, 2016	3.48%	UN Projection
5	Brazil	207,132,000	February 22, 2017	2.77%	Official population clock
6	Pakistan	196,435,000	February 22, 2017	2.62%	Official population clock
7	Nigeria	186,988,000	July 1, 2016	2.5%	UN Projection
8	Bangladesh	161,986,000	February 22, 2017	2.16%	Official population clock
9	Russia[Note 4]	146,838,993	January 1, 2017	1.96%	Official estimate

讀取HTML網頁表格 (2)

```
> URL1 <- getURL("https://en.wikipedia.org/wiki/List_of_countries_and_
dependencies_by_population")
> htmlTable1 <- readHTMLTable(URL1, header = TRUE)
> str(htmlTable1)
> head(htmlTable1[[2]])
```

```
> str(htmlTable1)
List of 3
 $ NULL: NULL
 $ NULL:'data.frame':   243 obs. of  6 variables:
  ..$ Rank                            : Factor w/ 197 levels "-","1","10","100",..: 2 110 121 132 143 154 165 176 187 3 ...
  ..$ Country (or dependent territory): Factor w/ 243 levels "Abkhazia[Note 19]",..: 44 95 232 96 30 160 152 18 175 105 ...
  ..$ Population                      : Factor w/ 243 levels "1,132,657","1,167,242",..: 8 5 128 103 92 78 73 62 53 49 ...
  ..$ Date                            : Factor w/ 59 levels "April 15, 2012",..: 15 15 15 27 15 15 27 15 21 21 ...
  ..$ % of world
population          : Factor w/ 187 levels "0.00000075%",..: 181 180 187 186 185 184 183 182 179 178 ...
  ..$ Source                          : Factor w/ 31 levels "2011 census result",..: 15 15 15 31 15 15 31 15 14 12 ...
 $ NULL:'data.frame':   24 obs. of  2 variables:
  ..$ V1: Factor w/ 13 levels "","Cities","Continental",..: 1 13 1 3 1 12 1 2 1 10 ...
  ..$ V2: Factor w/ 11 levels "Age at first marriage\nDivorce rate\nDomestic citizens\nEthnic and cultural diversity level\n
> head(htmlTable1[[2]])
  Rank Country (or dependent territory)    Population              Date % of world\npopulation                    Source
1    1                     China[Note 2] 1,381,680,000 February 22, 2017                 18.5% Official population clock
2    2                             India 1,312,320,000 February 22, 2017                 17.5% Official population clock
3    3           United States[Note 3]   324,567,000 February 22, 2017                 4.34% Official population clock
4    4                         Indonesia   260,581,000      July 1, 2016                 3.48%             UN Projection
5    5                            Brazil   207,132,000 February 22, 2017                 2.77% Official population clock
6    6                          Pakistan   196,435,000 February 22, 2017                 2.62% Official population clock
```

課堂練習



Rank	Title	Studio	Worldwide	Domestic / %	Overseas / %	Year^
1	Avatar	Fox	$2,788.0	$760.5 27.3%	$2,027.5 72.7%	2009^
2	Titanic	Par.	$2,186.8	$658.7 30.1%	$1,528.1 69.9%	1997^
3	Star Wars: The Force Awakens	BV	$2,068.2	$936.7 45.3%	$1,131.6 54.7%	2015
4	Jurassic World	Uni.	$1,670.4	$652.3 39.0%	$1,018.1 61.0%	2015
5	Marvel's The Avengers	BV	$1,518.8	$623.4 41.0%	$895.5 59.0%	2012
6	Furious 7	Uni.	$1,516.0	$353.0 23.3%	$1,163.0 76.7%	2015
7	Avengers: Age of Ultron	BV	$1,405.4	$459.0 32.7%	$946.4 67.3%	2015
8	Harry Potter and the Deathly Hallows Part 2	WB	$1,341.5	$381.0 28.4%	$960.5 71.6%	2011
9	Frozen	BV	$1,276.5	$400.7 31.4%	$875.7 68.6%	2013
10	Iron Man 3	BV	$1,214.8	$409.0 33.7%	$805.8 66.3%	2013
11	Minions	Uni.	$1,159.4	$336.0 29.0%	$823.4 71.0%	2015
12	Captain America: Civil War	BV	$1,153.3	$408.1 35.4%	$745.2 64.6%	2016
13	Transformers: Dark of the Moon	P/DW	$1,123.8	$352.4 31.4%	$771.4 68.6%	2011
14	The Lord of the Rings: The Return	NL	$1,119.9	$377.8 33.7%	$742.1 66.3%	2003^

http://www.boxofficemojo.com/alltime/world/

下載全球最高電影票房前500大電影紀錄:

- 票房分佈為何?
- 各發行商(Studio)之發行之電影數量為何?
- 各發行商(Studio)之平均每部電影之票房如何?
- 各年代之電影發行數量為何?

讀取 XML 檔案

XML [編輯] https://zh.wikipedia.org/wiki/XML

維基百科，自由的百科全書

可延伸標記式語言（英語：Extensible Markup Language，簡稱：**XML**），是一種標記式語言。標記指電腦所能理解的資訊符號，通過此種標記，電腦之間可以處理包含各種資訊的文章等。如何定義這些標記，既可以選擇國際通用的標記式語言，比如HTML，也可以使用像XML這樣由相關人士自由決定的標記式語言，這就是語言的可延伸性。XML是從標準通用標記式語言（SGML）中簡化修改出來的。它主要用到的有可延伸標記式語言、可延伸樣式語言（XSL）、XBRL和XPath等。

用途 [編輯]

XML設計用來傳送及攜帶資料資訊，不用來表現或展示資料，HTML語言則用來表現資料，所以XML用途的焦點是它說明資料是什麼，以及攜帶資料資訊。

- 豐富檔案（Rich Documents）- 自定檔案描述並使其更豐富
  - 屬於檔案為主的XML技術應用
  - 標記是用來定義一份資料應該如何呈現
- 後設資料（Metadata）- 描述其它檔案或網路資訊
  - 屬於資料為主的XML技術應用
  - 標記是用來說明一份資料的意義
- 配置文件（Configuration Files）- 描述軟體設定的

```
<?xml ==?>version"1.0" encoding"UTF-8" "no"standalone=
<!DOCTYPE recipe PUBLIC "-//Happy-Monkey//DTD RecipeBook//EN
"http://www.happy-monkey.net/recipebook/recipebook.dtd">

<recipe>

<title>Peanutbutter On A Spoon</title>

<ingredientlist>
  <ingredient>Peanutbutter</ingredient>
</ingredientlist>

<preparation>Stick a spoon in a jar of peanutbutter, scoop
and pull out a big glob of peanutbutter.</preparation>
```

語法上的烹飪技術書刊。此標籤可轉ormat並使用程式語言或XSL。

例 [編輯]

XML定義結構、儲存資訊、傳送資訊。下例為小張傳送給大元的便條，儲存為XML。

```
<?xml version="1.0"?>
<小紙条>
 <收件人>大元</收件人>
 <發件人>小張</發件人>
 <主題>問候</主題>
 <具體內容>早啊，飯吃了沒？ </具體內容>
</小紙条>
```

這XML文件僅是純粹的資訊標籤，這些標籤意義的展開依賴於應用它的程式。

讀取 XML 檔案

```
> library(XML)
> book.data <- xmlToDataFrame("books.xml")
> str(book.data)
> head(book.data)
```

```
> str(book.data)
'data.frame':   12 obs. of  6 variables:
 $ author      : Factor w/ 9 levels "Corets, Eva",..: 3 7 1 1 1 8 9 4 5 6 ...
 $ title       : Factor w/ 12 levels "Creepy Crawlies",..: 12 5 3 7 10 2 9 1 8 4 ...
 $ genre       : Factor w/ 5 levels "Computer","Fantasy",..: 1 2 2 2 2 4 4 3 5 1 ...
 $ price       : Factor w/ 6 levels "36.95","4.95",..: 3 5 5 5 5 2 2 2 6 1 ...
 $ publish_date: Factor w/ 11 levels "2000-09-02","2000-10-01",..: 2 8 4 9 11 1 3 6 3 7 ...
 $ description : Factor w/ 12 levels "A deep sea diver finds true love twenty
> head(book.data)
               author                 title    genre price publish_date
1 Gambardella, Matthew XML Developer's Guide Computer 44.95   2000-10-01
2           Ralls, Kim         Midnight Rain  Fantasy  5.95   2000-12-16
3          Corets, Eva       Maeve Ascendant  Fantasy  5.95   2000-11-17
4          Corets, Eva       Oberon's Legacy  Fantasy  5.95   2001-03-10
5          Corets, Eva  The Sundered Grail  Fantasy  5.95   2001-09-10
6     Randall, Cynthia           Lover Birds  Romance  4.95   2000-09-02

1
2                                          A former architect battles cor
3                                          After the collapse of a nanote
4 In post-apocalypse England, the mysterious \n      agent known only as Oberc
5                                          The two daughters of Maev
6
```

Sample XML File (books.xml)

The following XML file is used in various samples throughout the Microsoft XML Core Serv

XML

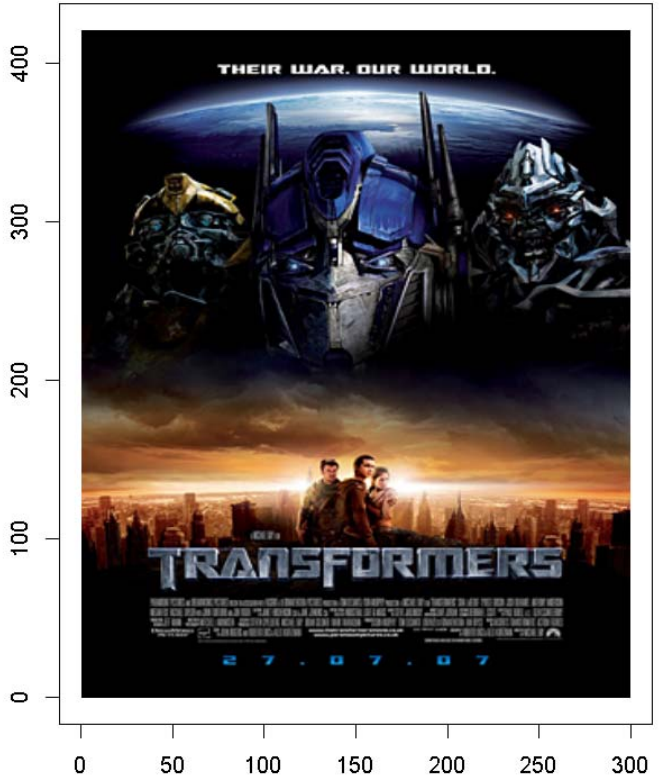
```
<?xml version="1.0"?>
<catalog>
   <book id="bk101">
      <author>Gambardella, Matthew</author>
      <title>XML Developer's Guide</title>
      <genre>Computer</genre>
      <price>44.95</price>
      <publish_date>2000-10-01</publish_date>
      <description>An in-depth look at creating applications
      with XML.</description>
   </book>
   <book id="bk102">
      <author>Ralls, Kim</author>
      <title>Midnight Rain</title>
      <genre>Fantasy</genre>
      <price>5.95</price>
      <publish_date>2000-12-16</publish_date>
      <description>A former architect battles corporate zombies,
      an evil sorceress, and her own childhood to become queen
      of the world.</description>
```

https://msdn.microsoft.com/en-us/library/ms762271(v=vs.85).aspx

讀取影像檔案

```
> install.packages(c("tiff", "jpeg", "png", "fftwtools"),
repos="http://cran.csie.ntu.edu.tw")
> library(EBImage) # (Repositories: BioC Software)
> Transformers <- readImage("Transformers07.jpg")
> (dims <- dim(Transformers))
[1] 300 421   3
> Transformers
Image
  colorMode    : Color
  storage.mode : double
  dim          : 300 421 3
  frames.total : 3
  frames.render: 1

imageData(object)[1:5,1:6,1]
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    0    0    0    0    0    0
[2,]    0    0    0    0    0    0
[3,]    0    0    0    0    0    0
[4,]    0    0    0    0    0    0
[5,]    0    0    0    0    0    0
> plot(c(0, dims[1]), c(0, dims[2]), type='n',
+ xlab="", ylab="")
> rasterImage(Transformers, 0, 0, dims[1], dims[2])
```



https://en.wikipedia.org/wiki/Transformers_(film)

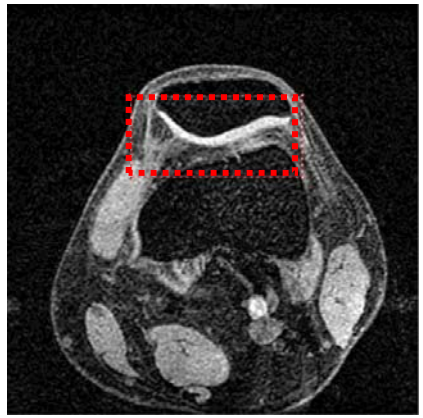
彩色影像轉成灰階

```
> Transformers.f <- Image(flip(Transformers))
> # convert RGB to grayscale
> rgb.weight <- c(0.2989, 0.587, 0.114)
> Transformers.gray <- rgb.weight[1] * imageData(Transformers.f)[,,1] +
+                      rgb.weight[2] * imageData(Transformers.f)[,,2] +
+                      rgb.weight[3] * imageData(Transformers.f)[,,3]
> dim(Transformers.gray)
[1] 300 421
> Transformers.gray[1:5, 1:5]
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    0    0    0    0
[2,]    0    0    0    0    0
[3,]    0    0    0    0    0
[4,]    0    0    0    0    0
[5,]    0    0    0    0    0
> par(mfrow=c(1,2), mai=c(0.1, 0.1, 0.1, 0.1))
> image(Transformers.gray, col = grey(
+ seq(0, 1, length = 256)), xaxt="n", yaxt="n")
> image(Transformers.gray, col = rainbow(256),
+ xaxt="n", yaxt="n")
```

Converting RGB to grayscale/intensity
http://stackoverflow.com/questions/687261/converting-rgb-to-grayscale-intensity

影像資料分析範例: Image Segmentation

- Segmentation: partition an image into homogeneous regions.
- Images: texture images, medical images, color images,...
- Medical Image Segmentation:
  - anatomical regions or pathological regions.
  - extract tumors.

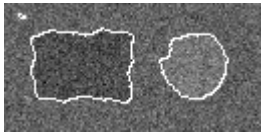


MRI images

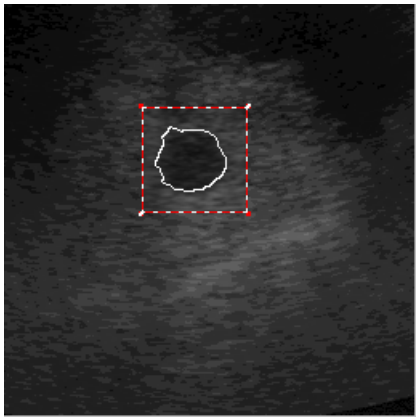
FCM+FSIR

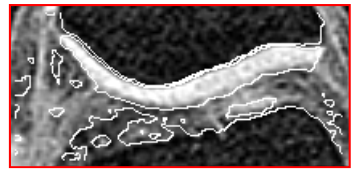
Simulated images

sd=20



FCM+FSIR





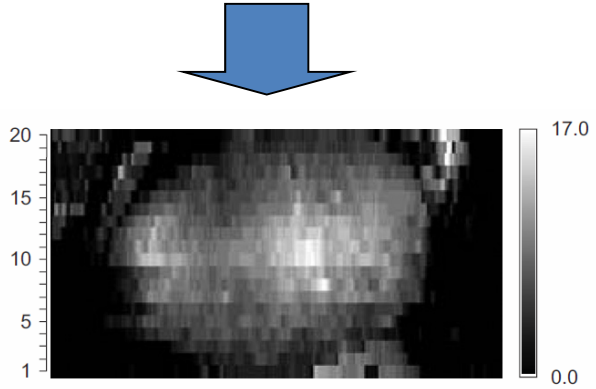
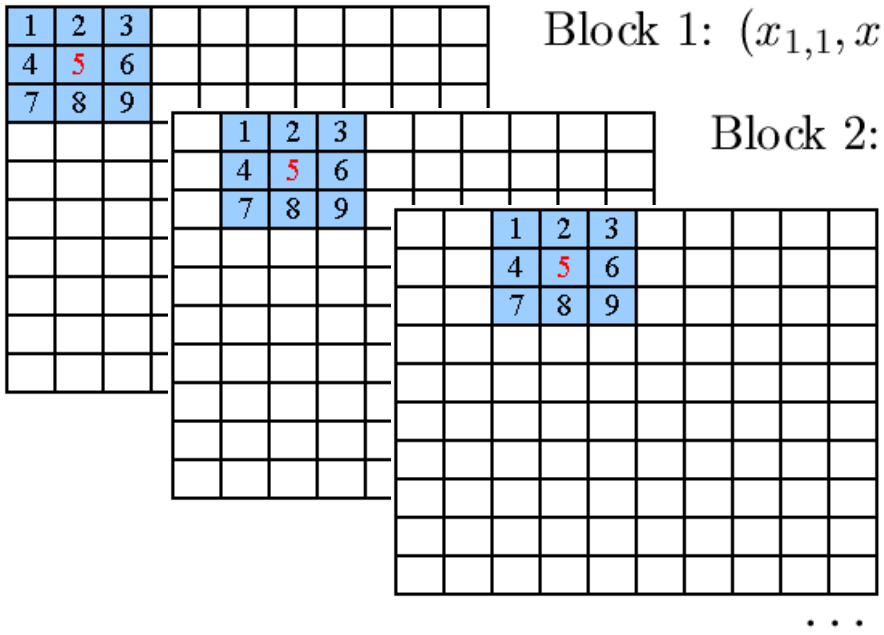


Image Feature Extraction: Local Blocks

Block length = 3



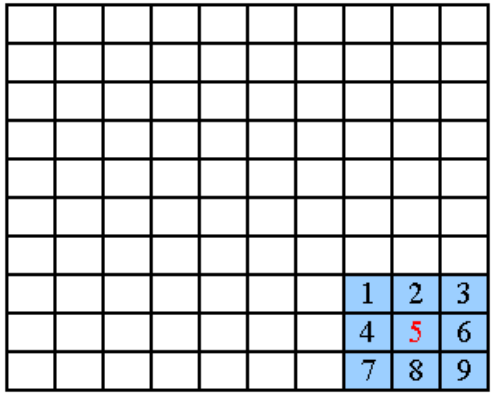
Block 1: $(x_{1,1}, x_{1,2}, \cdots, x_{1,9})$

Block 2: $(x_{2,1}, x_{2,2}, \cdots, x_{2,9})$

Block 3: $(x_{3,1}, x_{3,2}, \cdots, x_{3,9})$

$\cdots$

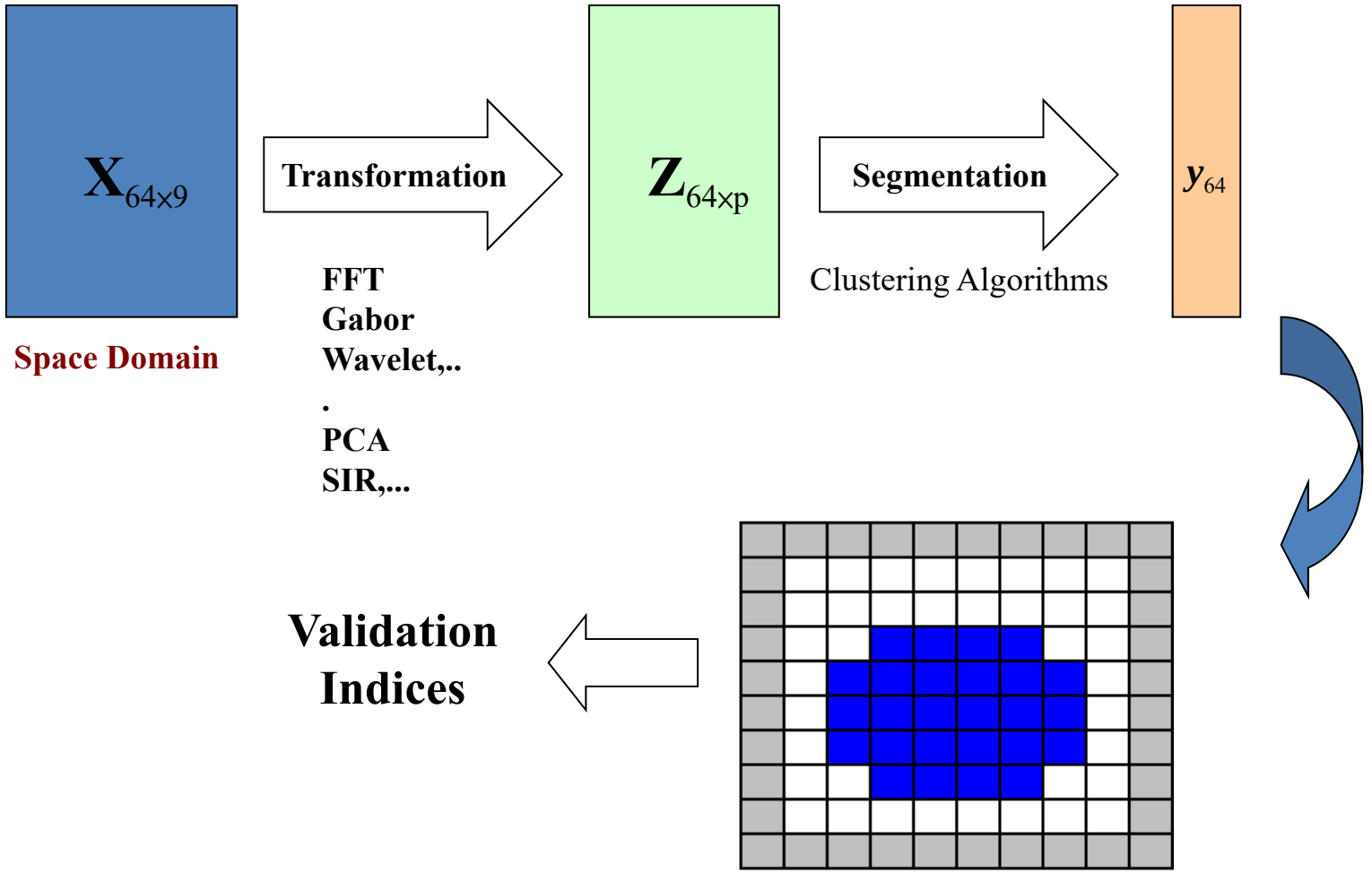
Block 64: $(x_{64,1}, x_{64,2}, \cdots, x_{64,9})$



grey level: *f(x,y)=0,...,255*

Image Features

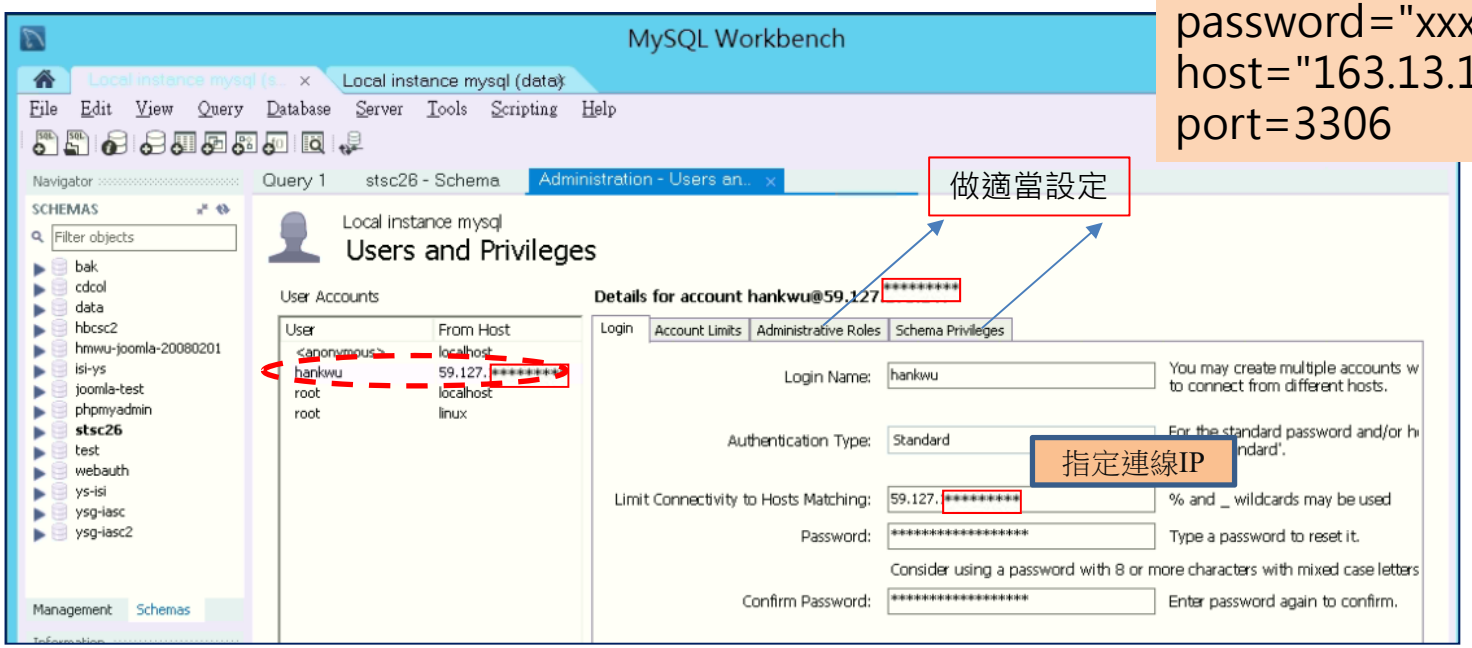
Block size $= 3 \times 3$



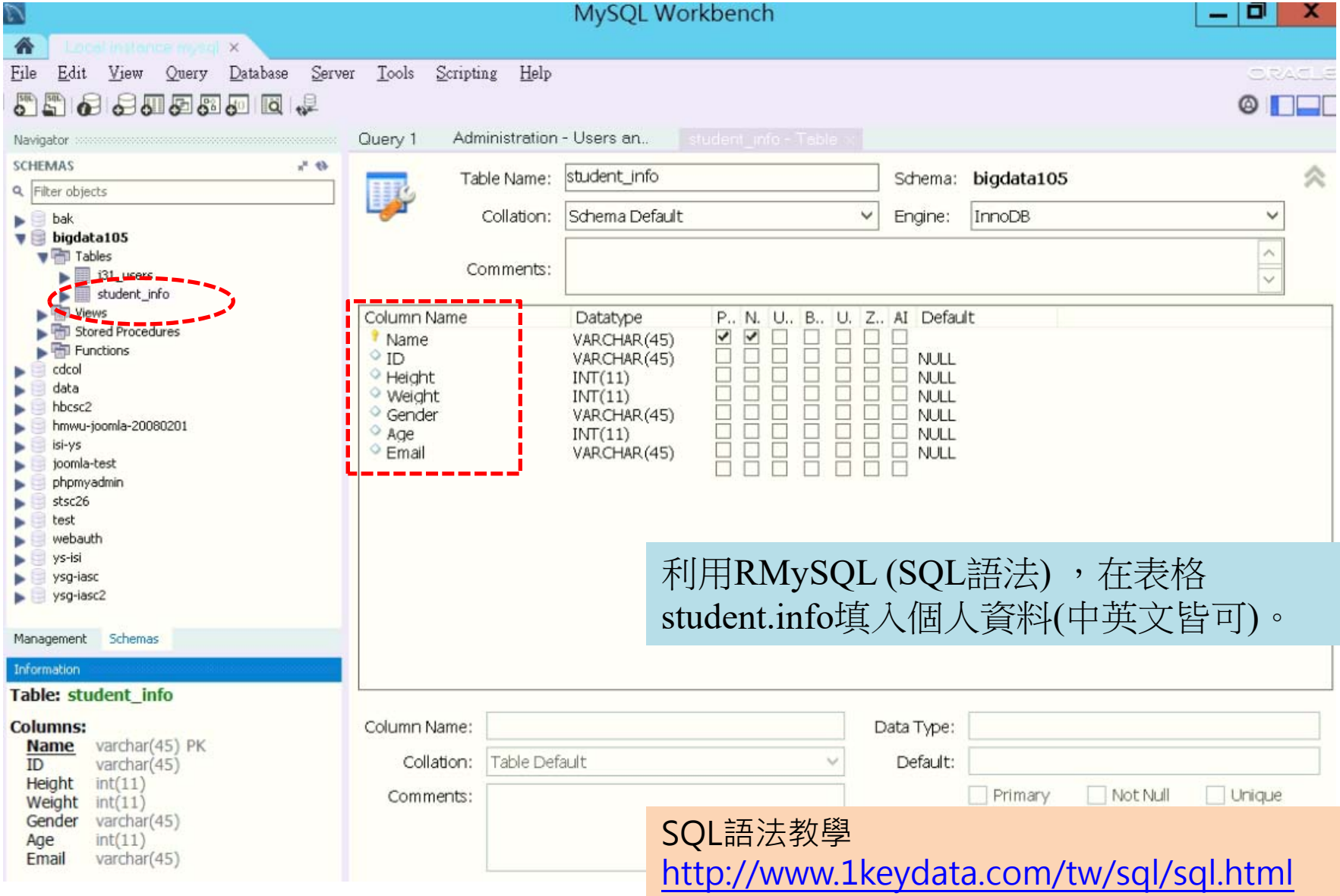
讀取MySQL資料

- 讀取Excel資料檔案
- 使用ODBC讀取 Excel 檔案 (Windows為例)
- 利用RMySQL
  讀取MySQL資料庫的資料 (localhost)
- 利用RMySQL
  讀取MySQL資料庫的資料 (remote host)

```
MySQL
dbname = "bigdata105",
username="student",
password="xxxxxx",
host="163.13.113.xxx",
port=3306
```



課堂練習



利用RMySQL (SQL語法) ，在表格student.info填入個人資料(中英文皆可)。

SQL語法教學
http://www.1keydata.com/tw/sql/sql.html

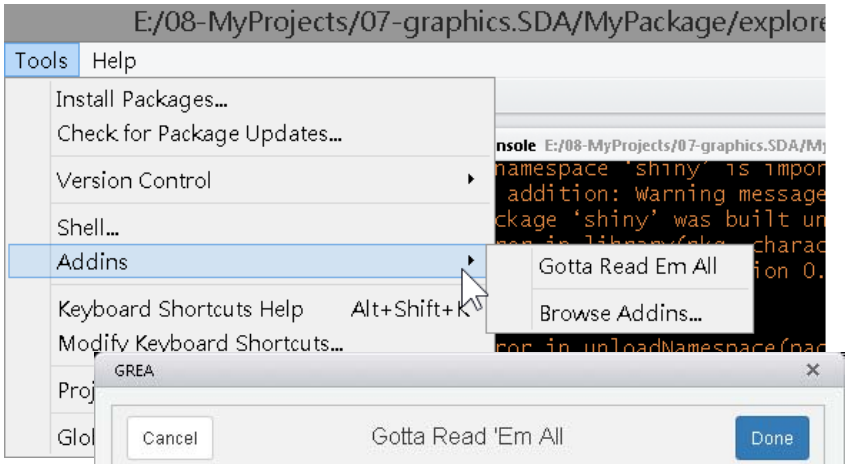
GREA: read ALL the data into R

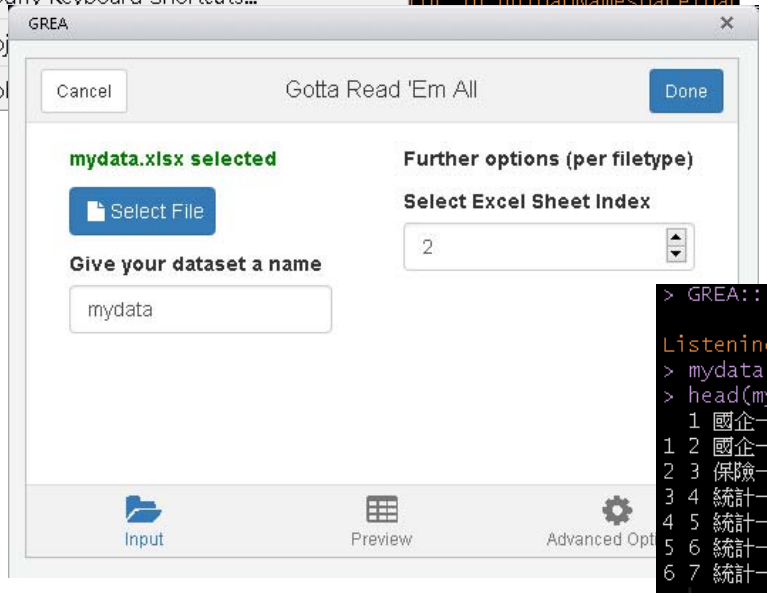
- GREA: The RStudio Add-In to read ALL the data into R!

https://www.r-bloggers.com/grea-the-rstudio-add-in-to-read-all-the-data-into-r/

- 在RStudio安裝

```
> devtools::install_github("Stan125/GREA", force = TRUE)
> install.packages(c("csvy", "miniUI", "openxlsx", "readODS", "urltools"))
```





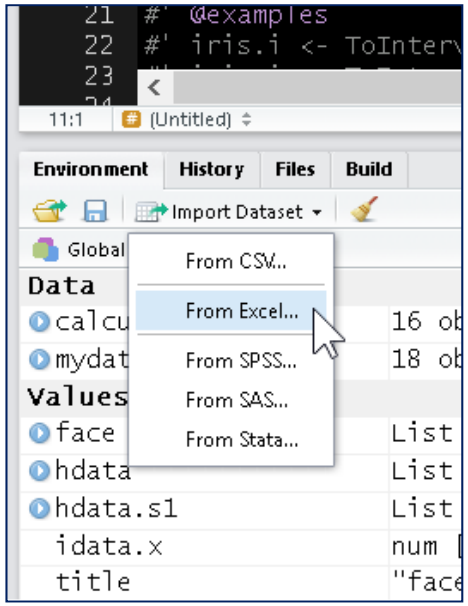
	A	B	C	D	E	F	G	H	I	J	K	L
1	Calculus				Quiz(1)	Quiz(2)	Quiz(3)	Quiz(4)		Midterm Exam		
2					10/15	11/12	12/10	1/7	TA	Core1	Core2	Sum
3	No	Departmen	ID	Name	7%	7%	8%	8%	15%	70%	30%	100%
4	1	國企一	981550867	張　勛	60	33	15	65	87	45	20	65
5	2	國企一	981555585	雷　逸	0				13			
6	3	保險一	983522324	張庭涵	0	0	5		73	5	0	5
7	4	統計一	984223018	張兆臻	30	25	30	10	60	13	8	21
8	5	統計一	984223026	柯品慧	25	10	10	15	73	5	8	13
9	6	統計一	984223034	謝欣逸	53	25	80	85	80	43	30	73
10	7	統計一	984223042	張儷誼	15	5	15	90	87	3	0	3
11	8	統計一	984223059	徐　詠	15	40	35	60	80	22	20	42
12	9	統計一	984223067	王莞宏	55	70	85	80	100	39	10	49
13	10	統計一	984223075	王易羽	20	28	10	70	80	31	5	36
14	11	數學一	984223083	高瓊萱	65	63	15	50	80	27	30	57
15	12	數學一	984223091	丁愛	95	86	85	75	100	60	20	80
16	13	數學一	984223109	張書樺	80	65	98	75	80	36	28	64
17	14	數學一	984223117	曾清瑄	15	0	5	0	73	0	7	7
18	15	數學一	984223125	劉倩怡	30	30	20	20	80	11	3	14
19	16	數學一	984223141	曾雲元	65	80	80	85	100	33	30	63
20	17	數學一	984223158	黃雅信	65	90	70	65	100	38	30	68
21	18	數學一	984223166	廖恭蓉	30	10	20	20	67	9	3	12

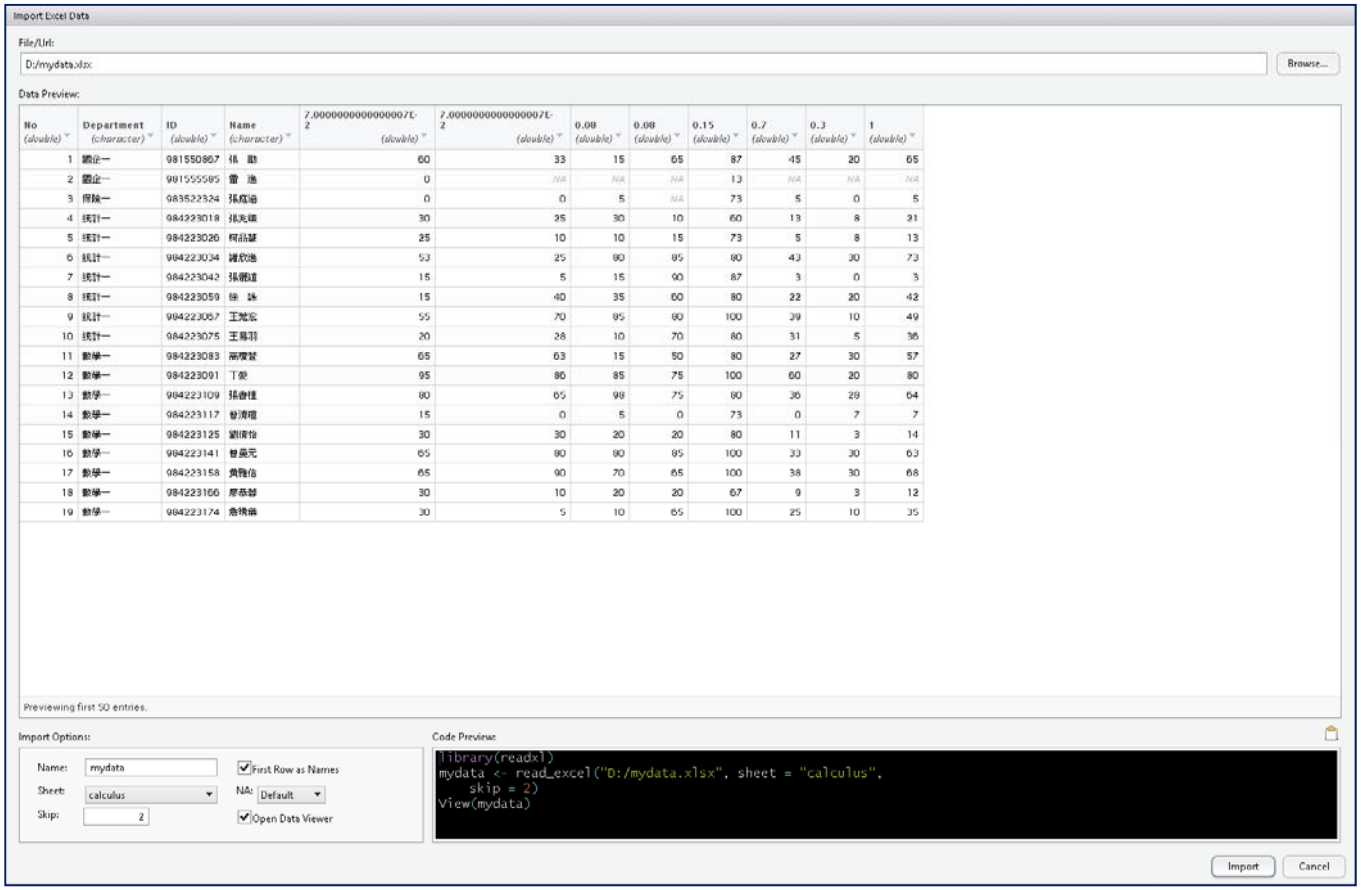
```
> GREA:::GREA()

Listening on http://127.0.0.1:4064
> mydata <- rio::import(file = "E:/10-R/01-主題/data/mydata.xlsx", which = 2L, skip = 3L)
> head(mydata)
  1 國企一 981550867 張　勛 60 33 15 65 87 45 20 65
1 2 國企一 981555585 雷　逸  0 NA NA NA 13 NA NA NA
2 3 保險一 983522324 張庭涵  0  0  5 NA 73  5  0  5
3 4 統計一 984223018 張兆臻 30 25 30 10 60 13  8 21
4 5 統計一 984223026 柯品慧 25 10 10 15 73  5  8 13
5 6 統計一 984223034 謝欣逸 53 25 80 85 80 43 30 73
6 7 統計一 984223042 張儷誼 15  5 15 90 87  3  0  3
```

Importing Data with RStudio

- Importing data into R is a necessary step that, at times, can become time intensive. To ease this task, RStudio includes new features to import data from: csv, xls, xlsx, sav, dta, por, sas and stata files.





https://support.rstudio.com/hc/en-us/articles/218611977-Importing-Data-with-RStudio
https://rstudio-pubs-static.s3.amazonaws.com/1776_dbaebbdbde8d46e693e5cb60c768ba92.html

readbulk: Read and Combine Multiple Data Files

```
read_bulk(directory = ".", subdirectories = FALSE, extension = NULL,
          data = NULL, verbose = TRUE, fun = utils::read.csv, ...)
```

```
> raw.data <- read_bulk(directory = ".", extension = ".txt", sep=" ")
Reading score_quiz1.txt
Reading score_quiz2.txt
Reading score_quiz3.txt
Reading score_quiz4.txt
Reading score_quiz5.txt
> str(raw.data)
'data.frame':   39 obs. of  7 variables:
 $ gender       : Factor w/ 2 levels "F","M": 2 2 1 1 1 2 1 1 2 1 ...
 $ Calculus     : int  69 70 57 73 56 62 76 68 73 74 ...
 $ LinearAlgebra: int  93 78 26 32 6 4 5 63 97 25 ...
 $ BasicMath    : int  83 31 21 73 50 73 8 43 23 0 ...
 $ Rprogramming : int  79 26 99 76 98 22 95 15 60 28 ...
 $ English      : int  95 69 51 37 33 4 62 97 66 9 ...
 $ File         : chr  "score_quiz1.txt" "score_quiz1.txt" "score_quiz1.txt"
"score_quiz1.txt" ...
> raw.data
   gender Calculus LinearAlgebra BasicMath Rprogramming English            File
1       M       69            93        83            79      95 score_quiz1.txt
2       M       70            78        31            26      69 score_quiz1.txt
...
38      F       70            32        55            79       6 score_quiz5.txt
39      F       53            15         2             9      80 score_quiz5.txt
```

fread {data.table}:
Fast and friendly file finagler

```
fread(input, sep="auto", sep2="auto", nrows=-1L, header="auto", na.strings="NA", file,
      stringsAsFactors=FALSE, verbose=getOption("datatable.verbose"), autostart=1L,
      skip=0L, select=NULL, drop=NULL, colClasses=NULL,
      integer64=getOption("datatable.integer64"),         # default: "integer64"
      dec=if (sep!=".") "." else ",", col.names,
      check.names=FALSE, encoding="unknown", quote="\"",
      strip.white=TRUE, fill=FALSE, blank.lines.skip=FALSE, key=NULL,
      showProgress=getOption("datatable.showProgress"),   # default: TRUE
      data.table=getOption("datatable.fread.datatable")   # default: TRUE
)
```

```
library(data.table)
mydata <- fread("mylargefile.txt")
```

Other features include :

- **fast and friendly file reader**: `?fread`. It accepts system commands directly (such as `grep` and `gunzip`) and other convenience features for *small* data.
- **fast and parallelized file writer**: `?fwrite` announced here and on CRAN in Nov 2016.
- **parallelized row subsets** - See this benchmark for timings
- fast **aggregation** of large data; e.g. 100GB in RAM (see benchmarks on up to **two billion rows**)

Amazon EC2 r3.8large (Ubuntu, CPU(s): 32, Mem: 240G)

http://brooksandrew.github.io/simpleblog/articles/advanced-data-table/
https://cran.r-project.org/web/packages/data.table/index.html
https://github.com/Rdatatable/data.table/wiki
https://www.datacamp.com/courses/data-table-data-manipulation-r-tutorial

Reference manual:	data.table.pdf
Vignettes:	Frequently asked questions
	Introduction to data.table
	Keys and fast binary search based subset
	Reference semantics
	Efficient reshaping using data.tables
	Secondary indices and auto indexing

Demo Speedup

https://www.rdocumentation.org/packages/data.table/versions/1.10.4/topics/fread

```
> n <- 1e6
> dt <- data.table(x1 = sample(1:1000, n, replace = TRUE),
+                  x2 = sample(1:1000, n, replace = TRUE),
+                  x3 = rnorm(n),
+                  x4 = sample(c("foo", "bar", "baz", "qux", "quux"), n, replace = TRUE),
+                  x5 = rnorm(n),
+                  x6 = sample(1:1000, n, replace = TRUE)
+ )
> write.table(dt, "Speedup-test.csv", sep = ",", row.names = FALSE, quote = FALSE)
> cat("File size (MB):", round(file.info("Speedup-test.csv")$size / 1024 ^ 2), "\n")
File size (MB): 51
>
> # read by read.csv
> system.time(data.rc <- read.csv("Speedup-test.csv", stringsAsFactors = FALSE))
   user  system elapsed
   6.86    0.13    7.00
>
> # read by read.table, (all known tricks and known nrows)
> system.time(data.rt <- read.table("Speedup-test.csv", header = TRUE,
+    sep = ",", quote = "",
+    stringsAsFactors = FALSE,
+    comment.char = "",
+    nrows = n, colClasses = c("integer", "integer", "numeric",
+                              "character", "numeric", "integer")
+     )
+ )
   user  system elapsed
   3.55    0.09    3.65
```

```
> # read by fread{data.table}
> system.time(data.fr <- fread("Speedup-test.csv"))
   user  system elapsed
   1.65    0.00    1.66
```

讀取部份資料進入R計算

HIGGS Data Set

http://archive.ics.uci.edu/ml/datasets/HIGGS

Download: Data Folder, Data Set Description

Abstract: This is a classification problem to distinguish between a signal process which produces Higgs bosons and a backgro

Data Set Characteristics:	N/A	**Number of Instances:**	11000000	**Area:**	Physical
Attribute Characteristics:	Real	**Number of Attributes:**	28	**Date Donated**	2014-02-12
Associated Tasks:	Classification	**Missing Values?**	N/A	**Number of Web Hits:**	49041

```
> cat(round(file.info('HIGGS.csv')$size /2^30, 2), "GB\n")
7.48 GB
>
> transactFile <- 'HIGGS.csv'
> readLines(transactFile, n=2)
[1] "1.0000000000000000e+00,8.692932128906250000e-01,-6.350818276405334473e-
01,2.256902605295181274e-01,3.274700641632080078e-01,-6.899932026863098145e-...
>
> variables <- c("label", "lepton_pT", "lepton_eta", "lepton_phi", "missing_energy_magnitude",
"missing_energy_phi", "jet_1_pt", "jet_1_eta", "jet_1_phi", "jet_1_b_tag", "jet_2_pt",
"jet_2_eta", "jet_2_phi", "jet_2_b_tag", "jet_3_pt", "jet_3_eta", "jet_3_phi", "jet_3_b-tag",
"jet_4_pt",  "jet_4_eta", "jet_4_phi", "jet_4_b_tag", "m_jj", "m_jjj", "m_lv", "m_jlv",
"m_bb", "m_wbb", "m_wwbb")
```

讀取部份資料進入R計算

```
> chunkSize <- 100000
> con <- file(description=transactFile, open="r")
> dataChunk <- read.table(con, nrows=chunkSize, header=T, fill=TRUE, sep=",",
+                         col.names=variables)
> index <- 0
> counter <- 0
> s <- 0
> repeat {
+     index <- index + 1
+     cat("Processing rows:", index * chunkSize, "\n")
+     s <- s + sum(dataChunk$lepton_pT)
+     counter <- counter + nrow(dataChunk)
+     if (nrow(dataChunk) != chunkSize){
+         print('Done!')
+         break
+     }
+     dataChunk <- read.table(con, nrows=chunkSize, skip=0, header=F,
+                             fill = TRUE, sep=",", col.names=variables)
+
+     if(index > 3) break # test this process in 3 times
+ }
Processing rows: 1e+05
Processing rows: 2e+05
Processing rows: 3e+05
Processing rows: 4e+05
> close(con)
> cat("number of observations: ", counter, "\n")
number of observations:  4e+05
> cat("mean of lepton_pT: ", s/counter, "\n")
mean of lepton_pT:  0.9923863
```

See also:

- **LaF**: Fast Access to Large ASCII Files
- **chunked**: Chunkwise Text-File Processing for 'dplyr'
- **ff**: memory-efficient storage of large data on disk and fast access functions.
- **readf {data.table}**
- **readbulk:** Read and Combine Multiple Data Files

讀取檔案部份欄位資料

```
> first.line <- readLines("HIGGS.csv", n=1)
> # Split the first line on the separator
> items <- strsplit(first.line, split=",", fixed=TRUE)[[1]]
> items
 [1] "1.00000000000000000e+00"  "8.69293128906250000e-01"  "-6.35081827640533447e-01" "2.25690260529518127e-01"
 [5] "3.27470064163208007e-01"  "-6.89993202686309814e-01" "7.54202246665954589e-01"  "-2.48573139309883117e-01"
 [9] "-1.09206390380859375e+00" "0.00000000000000000e+00"  "1.37499213218688965e+00"  "-6.53674185276031494e-01"
[13] "9.30349111557006836e-01"  "1.10743606090545654e+00"  "1.13890433311462402e+00"  "-1.57819831371307373e+00"
[17] "-1.04698538780212402e+00" "0.00000000000000000e+00"  "6.57929539680480957e-01"  "-1.04545699432492256e-02"
[21] "-4.57671694457530975e-02" "3.10196137428283691e+00"  "1.35376000404357910e+00"  "9.79563117027282715e-01"
[25] "9.78076159954071045e-01"  "9.20004844665527344e-01"  "7.21657454967498779e-01"  "9.88750934600830078e-01"
[29] "8.76678347587585449e-01"
> length(items)
[1] 29
```

```
> HIGGS.first2cols <- read.table("HIGGS.csv", header=F, fill=TRUE, sep=",",
+                                colClasses = c(rep("numeric", 2), rep("NULL", 27)))
> str(HIGGS.first2cols)
'data.frame':   11000000 obs. of  3 variables:
 $ V1 : num  1 1 1 0 1 0 1 1 1 1 ...
 $ V2 : num  0.869 0.908 0.799 1.344 1.105 ...
> head(HIGGS.first2cols)
  V1        V2
1  1 0.8692932
2  1 0.9075421
3  1 0.7988347
4  0 1.3443848
5  1 1.1050090
6  0 1.5958393
```

colClasses: "character", "complex", "factor", "integer", "numeric", "Date", "logical"

如何讓read.table讀較大的資料速度更快: 設定colClasses

- Specifying **colClasses** instead of using the default can make 'read.table' run MUCH faster, often twice as fast.
- If all of the columns are "numeric", just set '**colClasses = "numeric"**'.
- If the columns are all different classes, or perhaps you just don't know, then you can read in just a few rows of the table and then create a vector of classes from just the few rows.

```
> system.time(data.rt1 <- read.table("Speedup-test.csv", header = TRUE, sep = ","))
   user  system elapsed
   6.48    0.03    6.54
> system.time(tab5rows <- read.table("Speedup-test.csv", header = TRUE, sep = ",", nrows = 5))
   user  system elapsed
      0       0       0
> classes <- sapply(tab5rows, class)
> classes
       x1        x2        x3        x4        x5        x6
"integer" "integer" "numeric"  "factor" "numeric" "integer"
> system.time(data.rt2 <- read.table("Speedup-test.csv", header = TRUE, sep = ",",
+                                    colClasses = classes))
   user  system elapsed
   3.59    0.04    3.64
```

http://www.biostat.jhsph.edu/~rpeng/docs/R-large-tables.html

如何讓read.table讀較大的資料速度更快: 設定nrows, comment.char

- Specifying the '**nrows**' argument doesn't necessary make things go faster but it can help a lot with memory usage.
- If you know that the data rows are definitely less than, say, N rows, then you can specify '**nrows = N**' and things will still be okay. A mild overestimate for '**nrows**' is better than none at all.

```
> # install.packages("R.utils")
> library(R.utils)
> system.time(n1 <- countLines("HIGGS.csv"))
   user  system elapsed
  32.44   22.50   55.00
> system.time(n2 <- length(readLines("HIGGS.csv")))
   user  system elapsed
 308.24    7.36  315.78
> n1
[1] 11000000
attr(,"lastLineHasNewline")
[1] TRUE
> n2
[1] 11000000
```

comment.char: If the data file has no comments in it (e.g. lines starting with '#'), then setting 'comment.char = ""' will sometimes make 'read.table()' run faster.