charles
SCHWAB

Creating an R Database

Patrick Howerter

Schwab Equity Ratings Department

Analysts spend a lot time preparing data...

- Analysts spend 60 to 90% of their time preparing their data
- Many analysts are not database experts
  - Develop research that is not shareable or reproducible
- Quants are expected to use near perfect data (SR 11-7)
- Models developed in R rarely translate into Production

Research Platform

- Free flowing data from multiple sources
- A lot of history being analyzed(20+ years)
- Quality not rigorously checked
- Timeliness of updates not critical (Generous lead time to fix data issues)

Production

- Limited amount of data sources
- Run time is usually faster than a backtest since a current slice of data is used compared to a backtest
- Regulated and requires verification and validation
- Requires redundancy as data delays/outages are disastrous

Common Use Case (ETL process)

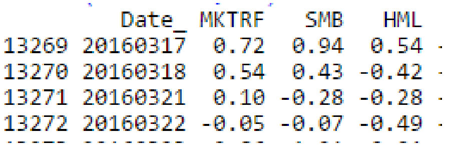
Basic R User

.csv

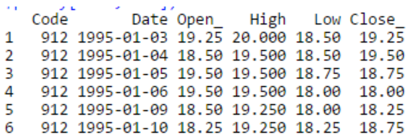
“Quant” GUI

utils::read.csv openxlsx::read.xlsx

utils:: download. File; utils::unzip



```
            Date_ MKTRF   SMB   HML
13269 20160317  0.72  0.94  0.54
13270 20160318  0.54  0.43 -0.42
13271 20160321  0.10 -0.28 -0.28
13272 20160322 -0.05 -0.07 -0.49
```



```
  Code       Date Open_   High   Low Close_
1  912 1995-01-03 19.25 20.000 18.50  19.25
2  912 1995-01-04 18.50 19.500 18.50  19.50
3  912 1995-01-05 19.50 19.500 18.75  18.75
4  912 1995-01-06 19.50 19.500 18.00  18.00
5  912 1995-01-09 18.50 19.250 18.00  18.25
6  912 1995-01-10 18.25 19.250 18.25  18.75
```

Advanced R User

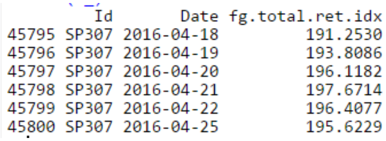
Vendor API

SQL DB

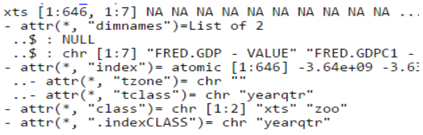
Quandl

VendorAPI::VendorDownload RODBC::sqlQuery

Quandl::Quandl.dataset.get



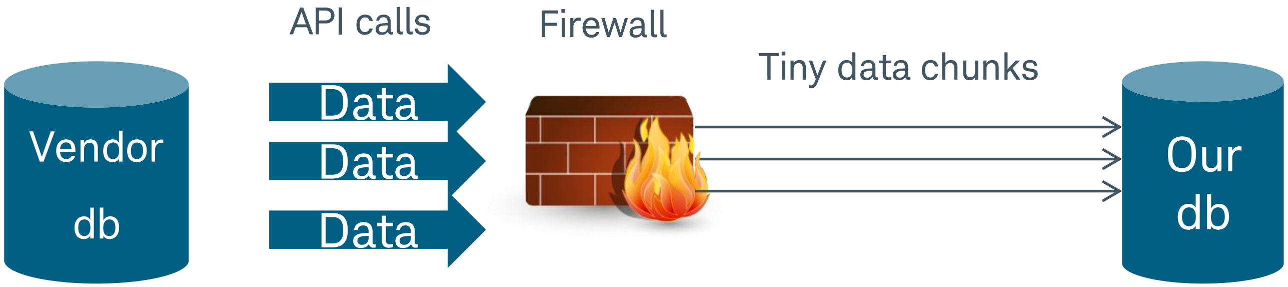
```
         Id       Date fg.total.ret.idx
45795 SP307 2016-04-18         191.2530
45796 SP307 2016-04-19         193.8086
45797 SP307 2016-04-20         196.1182
45798 SP307 2016-04-21         197.6714
45799 SP307 2016-04-22         196.4077
45800 SP307 2016-04-25         195.6229
```



```
xts [1:646, 1:7] NA NA NA NA NA NA NA NA NA NA ...
- attr(*, "dimnames")=List of 2
 ..$ : NULL
 ..$ : chr [1:7] "FRED.GDP - VALUE" "FRED.GDPC1 -
- attr(*, "index")= atomic [1:646] -3.64e+09 -3.6
 ..- attr(*, "tzone")= chr ""
 ..- attr(*, "tclass")= chr "yearqtr"
- attr(*, "class")= chr [1:2] "xts" "zoo"
- attr(*, ".indexCLASS")= chr "yearqtr"
```

Using the API for Production AND Research

Use the vendor's API to call organized data. Downloads via HTTPS



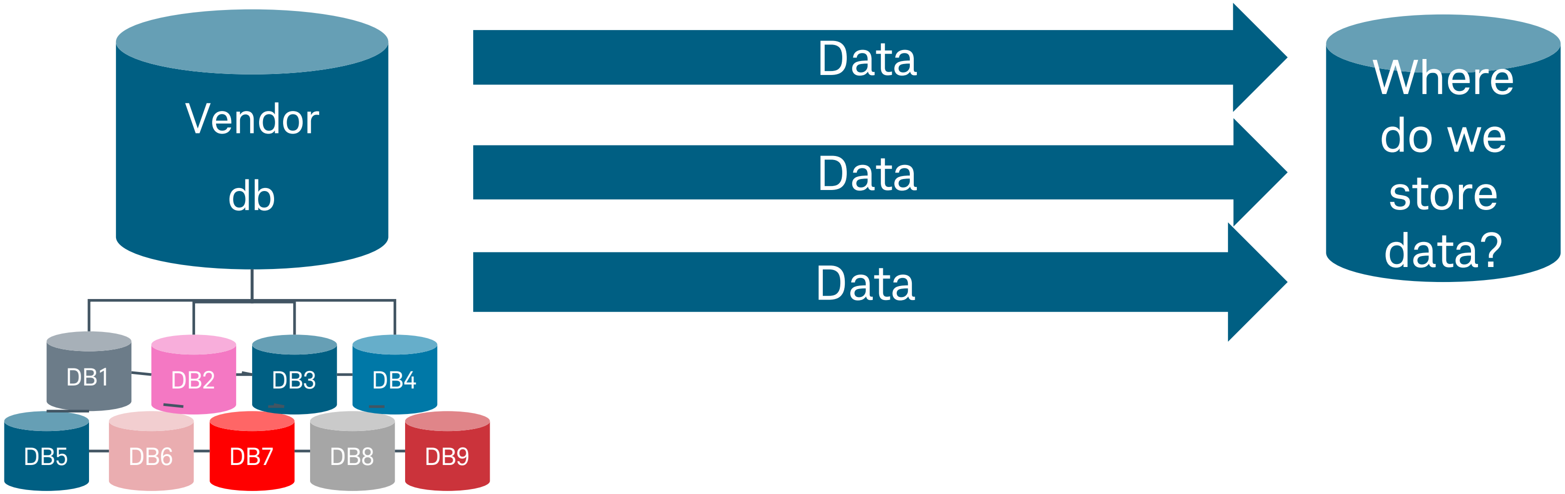
Issues

- Vendors limit download speeds to accommodate multiple clients
- No transparency of data inputs
- Download time outs happen
- Can take many days to re-build historical database
- Maintaining corporate actions becomes onerous (revisions and restatements might require a full rebuild)

Conclusion: Convenient to use but takes too long to download extensive history and not reliable enough to run in Production

Gathering the Raw Data

- Use a Database of the raw data inputs



- SQL Data is “tidy” but data attributes are different (mapping, corporate actions, date dimensions)
- Need a common way to gather our data from different formats so we can analyze all data in the same manner

Organizing input data into an Analytical database

Option 1

Use R functions to call SQL stored procedures:

Pros

- Always calling “fresh” data
- No database maintenance

Cons

- SQL and most relational database are intended to store rather than analyze data
- SQL inefficient when returning time series data
- Authentication every time that SQL is called
- Learning to develop SQL code and not R code

Option 2

Download raw SQL data then transform the data using R code:

Pros

- Mostly select statements (SQL’s Strengths)
- More time spent coding R!

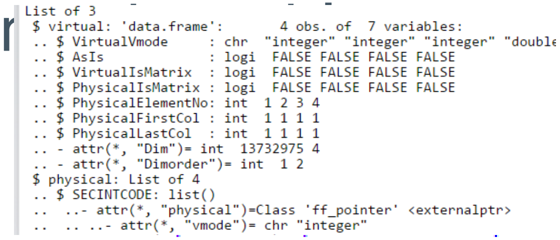
Cons

- Maintaining a database (what type?)
- Virtual memory easily overloaded when you download large chunks of data!

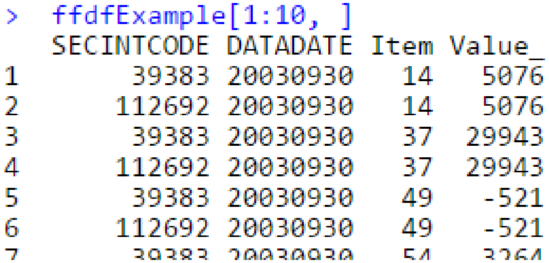
Using the ff and ETLUtils Packages

ff::ffdf

- Writes R objects to disk
- Use the object as an array (ff) or as a data.frame (ffdf)
- Can save the connections (structure attributes) to the objects and re-open them
- Has attributes such as read-only to allow multiple user to access the sar



```
List of 3
 $ virtual: 'data.frame':       4 obs. of  7 variables:
 .. $ VirtualVmode     : chr  "integer" "integer" "integer" "double
 .. $ AsIs             : logi  FALSE FALSE FALSE FALSE
 .. $ VirtualIsMatrix  : logi  FALSE FALSE FALSE FALSE
 .. $ PhysicalIsMatrix : logi  FALSE FALSE FALSE FALSE
 .. $ PhysicalElementNo: int  1 2 3 4
 .. $ PhysicalFirstCol : int  1 1 1 1
 .. $ PhysicalLastCol  : int  1 1 1 1
 .. - attr(*, "Dim")= int  13732975 4
 .. - attr(*, "Dimorder")= int  1 2
 $ physical: List of 4
 .. $ SECINTCODE: list()
 ..  ..- attr(*, "physical")=Class 'ff_pointer' <externalptr>
 ..  .. ..- attr(*, "vmode")= chr "integer"
```

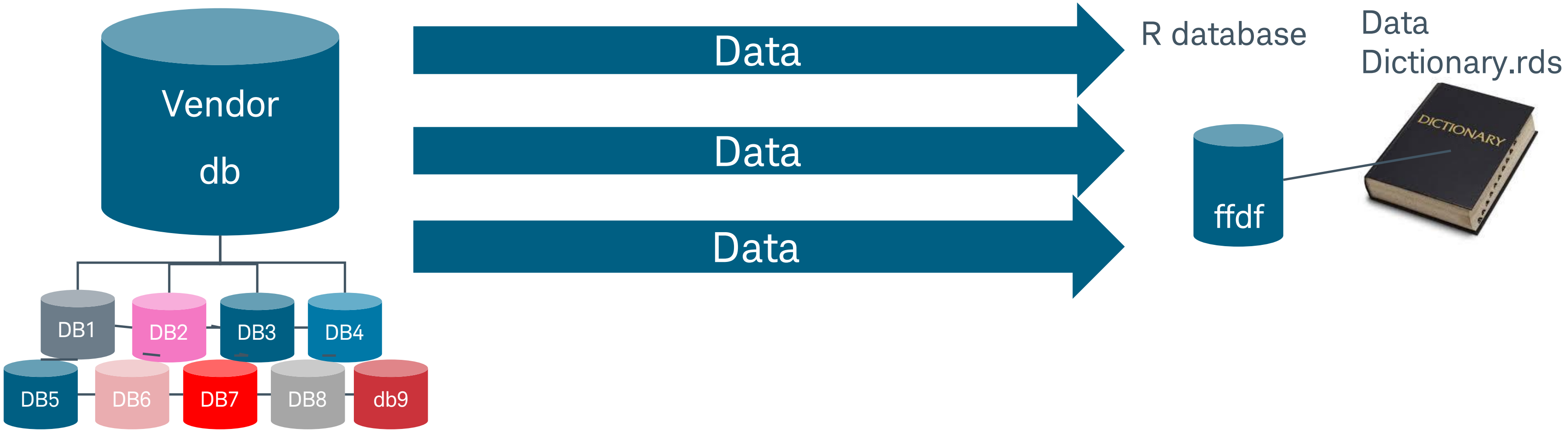


```
>  ffdfExample[1:10, ]
   SECINTCODE DATADATE Item Value_
1       39383 20030930   14   5076
2      112692 20030930   14   5076
3       39383 20030930   37  29943
4      112692 20030930   37  29943
5       39383 20030930   49   -521
6      112692 20030930   49   -521
7       39383 20030930   54   3264
```

ETLUtils::read.odbc.ffdf

- Executes the database syntax, creates an ffdf object, and loads the data
- Uses ODBC, JDBC, and DBI connections
- Can specify chunk sizes to manage virtual memory
- Can run in parallel processing because you are writing to disk!!!!
- Save your data connections in an RDS file and use them again
- Store the data back into a database for point-in-time research by using load.odbc.ffdf

Success!



- Organized over 10 vendors database schemas into one common r database
- Extracted 30 of 700 GBs of SQL data into an R database
- Can download our historical database daily in 1.5 hours with about 200 simple data calls running in parallel
- Multiple users accessing the same database just by opening a data dictionary Rds file.

Steps to Becoming an ETL *Ninja*

- Find the data source and write a download script to ff files.
  - If the data is in a database use the read.odbc.ffdf command
  - If you are reading in files from another source into a data frame, convert the object to ffdf using the as.ffdf() command.
- When you are satisfied with your ffdf structure then execute save.ffdf command and store the names of your ffdf files into .Rds files (persistent layer)
- The next time you log into your database use your Rds files to load in your ffdf objects!

Credits

- citation('ff') Daniel Adler, Christian Gläser, Oleg Nenadic, Jens Oehlschlägel and Walter Zucchini (2014). ff: memory-efficient storage of large data on disk and fast access functions. R package version 2.2-13. http://CRAN.R-project.org/package=ff

- citation('ETLUtils') Jan Wijffels (2015). ETLUtils: Utility Functions to Execute Standard Extract/Transform/Load Operations (using Package 'ff') on Large Data. R package version 1.3. http://CRAN.R-project.org/package=ETLUtils

Thank You!

charles SCHWAB

Own your tomorrow™