Financial Data Access
with SQL, Excel & VBA

Guy Yollin
Instructor, Applied Mathematics
University of Washington
1. Introduction to QueryTables
2. Downloading equity data from Yahoo Finance
3. Read data from SQL databases
4. Implementing VBA worksheet formulas
5. VBA chart creation
Outline

1. Introduction to QueryTables
2. Downloading equity data from Yahoo Finance
3. Read data from SQL databases
4. Implementing VBA worksheet formulas
5. VBA chart creation
Section references

J. Green, S. Bullen, R. Bovey, M. Alexander
Excel 2007 VBA Programmer’s Reference
Wiley, 2007
  • Chapter 21
The QueryTables object is a collection of QueryTable objects

- Use the Add method to create a new query table and add it to the QueryTables collection
- Each QueryTable object represents a worksheet table built from data returned from an external data source
The QueryTables.Add Method

The QueryTables.Add Method creates a new query table

Syntax:

expression .Add(Connection, Destination, Sql)

**Connection**  The data source for the query table

- "ODBC;<connection string>"
- "URL;<url>"
- "TEXT;<text file path and name>"

**Destination**  The cell in the upper-left corner of the query table destination range

**Sql**  An optional SQL query string for an ODBC data source
The QueryTable.Refresh Method

The Refresh method causes Excel to connect to the data source of the QueryTable object, execute the SQL query, and return data to the range that is based on the QueryTable object.

Syntax:

expression.Refresh(BackgroundQuery)

**BackgroundQuery**

Used only with QueryTables that are based on the results of a SQL query

- True to return control to the procedure as soon as a database connection is made and the query is submitted
- False to return control to the procedure only after all data has been fetched to the worksheet
Properties and methods of the QueryTable object

Properties

**BackgroundQuery**
- True if queries for the query table are performed asynchronously (in the background)

**CommandText**
- Returns or sets the command string for the specified data source

**CommandType**
- The CommandType describes the value of the CommandText property

**TextFileParseType**
- Gets/sets text file column format

**TextFileCommaDelimiter**
- True if the comma is the delimiter when you import a text file

Methods

**Refresh**
- Updates an external data range (QueryTable)
  
  see QueryTable Object (Excel)


Guy Yollin (Copyright © 2012)
Steps to retrieve data from an external data source

Most data fetches will follow this basic procedure:

1. Define the connection string
2. Define the destination
3. Call QueryTables.Add to create the query table object
4. Set the command (i.e. specify the data to be retrieved)
5. Call QueryTable.Refresh to get the data
1. Introduction to QueryTables
2. Downloading equity data from Yahoo Finance
3. Read data from SQL databases
4. Implementing VBA worksheet formulas
5. VBA chart creation
Section references

Simon Beginninga
Financial Modeling, 3rd Edition
Massachusetts Institute of Technology, 2008
  • Chapter 41

J. Green, S. Bullen, R. Bovey, M. Alexander
Excel 2007 VBA Programmer’s Reference
Wiley, 2007
  • Chapter 21
Browser initiated download

It is extremely easy to download historic equity data from Yahoo Finance directly into Excel.

Just enter the following URL into a browser:

- The parameter "s=" specifies the Yahoo ticker symbol.
Since only the ticker symbol and no date range was specified, all available data is downloaded.

- S&P 500 Index from Jan-1950 to most recent trading day
Full Yahoo finance download string

Download S&P 500 index for 2011:

http://ichart.yahoo.com/table.csv?s=^GSPC&a=0&b=01&c=2011&d=11&e=31&f=2011&g=d

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>ticker symbol</td>
</tr>
<tr>
<td>a</td>
<td>fromMonth - 1</td>
</tr>
<tr>
<td>b</td>
<td>fromDat (2 digits)</td>
</tr>
<tr>
<td>c</td>
<td>fromYear</td>
</tr>
<tr>
<td>d</td>
<td>toMonth - 1</td>
</tr>
<tr>
<td>e</td>
<td>toDay (2 digits)</td>
</tr>
<tr>
<td>f</td>
<td>toYear</td>
</tr>
<tr>
<td>g</td>
<td>d=day, m=month</td>
</tr>
</tbody>
</table>
Sub GetPricesVer1()
    Dim Ws As Worksheet
    Dim Wq As QueryTable
    Dim Url As String
    Url = 
        "URL;" & _
        "s=^GSPC&" & _
        "g=d&" & _
        "ignore=.csv"
    Set Ws = ActiveSheet
    Set Wq = Ws.QueryTables.Add( _
        Connection:=Url, _
        Destination:=Ws.Range("A1")
    )
    Wq.Refresh BackgroundQuery:=False
End Sub

adapted from S. Beginninga, Financial Modeling, 3rd Ed.
The VBA query inserts the CSV data as one-cell-per-row
Sub GetPricesVer2()
    Dim Ws As Worksheet
    Dim Wq As QueryTable
    Dim Url As String
    Url = "TEXT; " & 
        "s=~GSPC&" & _
        "g=d&" & _
        "ignore=.csv"
    Set Ws = ActiveSheet
    Set Wq = Ws.QueryTables.Add( _
        Connection:=Url, _
        Destination:=Ws.Range("A1"))
    Wq.TextFileParseType = xlDelimited
    Wq.TextFileCommaDelimiter = True
    Wq.Refresh BackgroundQuery:=False
End Sub

adapted from S. Beginninga, Financial Modeling, 3rd Ed.
Properly formatted Yahoo download results

<table>
<thead>
<tr>
<th>Date</th>
<th>Open</th>
<th>High</th>
<th>Low</th>
<th>Close</th>
<th>Volume</th>
<th>Adj Close</th>
</tr>
</thead>
<tbody>
<tr>
<td>9/7/2012</td>
<td>1432.12</td>
<td>1437.92</td>
<td>1431.45</td>
<td>1437.92</td>
<td>3717620000</td>
<td>1437.92</td>
</tr>
<tr>
<td>9/6/2012</td>
<td>1403.74</td>
<td>1432.12</td>
<td>1403.74</td>
<td>1432.12</td>
<td>3952870000</td>
<td>1432.12</td>
</tr>
<tr>
<td>9/5/2012</td>
<td>1404.94</td>
<td>1408.81</td>
<td>1401.25</td>
<td>1403.44</td>
<td>3389110000</td>
<td>1403.44</td>
</tr>
<tr>
<td>9/4/2012</td>
<td>1406.54</td>
<td>1409.31</td>
<td>1396.56</td>
<td>1404.94</td>
<td>3200310000</td>
<td>1404.94</td>
</tr>
<tr>
<td>8/31/2012</td>
<td>1400.07</td>
<td>1413.09</td>
<td>1398.96</td>
<td>1406.58</td>
<td>2938250000</td>
<td>1406.58</td>
</tr>
<tr>
<td>8/30/2012</td>
<td>1410.08</td>
<td>1410.08</td>
<td>1397.01</td>
<td>1399.48</td>
<td>2530280000</td>
<td>1399.48</td>
</tr>
<tr>
<td>8/29/2012</td>
<td>1409.32</td>
<td>1413.95</td>
<td>1406.57</td>
<td>1410.49</td>
<td>2571220000</td>
<td>1410.49</td>
</tr>
<tr>
<td>8/28/2012</td>
<td>1410.44</td>
<td>1413.63</td>
<td>1405.59</td>
<td>1409.3</td>
<td>2629090000</td>
<td>1409.3</td>
</tr>
<tr>
<td>8/27/2012</td>
<td>1411.13</td>
<td>1416.17</td>
<td>1409.11</td>
<td>1410.44</td>
<td>2472500000</td>
<td>1410.44</td>
</tr>
<tr>
<td>8/24/2012</td>
<td>1401.99</td>
<td>1413.46</td>
<td>1398.04</td>
<td>1411.13</td>
<td>2598790000</td>
<td>1411.13</td>
</tr>
<tr>
<td>8/23/2012</td>
<td>1413.49</td>
<td>1413.49</td>
<td>1400.5</td>
<td>1402.08</td>
<td>3008240000</td>
<td>1402.08</td>
</tr>
<tr>
<td>8/22/2012</td>
<td>1413.09</td>
<td>1416.12</td>
<td>1406.78</td>
<td>1413.49</td>
<td>3062690000</td>
<td>1413.49</td>
</tr>
<tr>
<td>8/21/2012</td>
<td>1418.13</td>
<td>1426.68</td>
<td>1410.86</td>
<td>1413.17</td>
<td>3282950000</td>
<td>1413.17</td>
</tr>
<tr>
<td>8/20/2012</td>
<td>1417.85</td>
<td>1418.13</td>
<td>1412.12</td>
<td>1418.13</td>
<td>2766320000</td>
<td>1418.13</td>
</tr>
<tr>
<td>8/17/2012</td>
<td>1415.84</td>
<td>1418.71</td>
<td>1414.67</td>
<td>1418.16</td>
<td>2922990000</td>
<td>1418.16</td>
</tr>
<tr>
<td>8/16/2012</td>
<td>1405.57</td>
<td>1417.44</td>
<td>1404.15</td>
<td>1415.51</td>
<td>3114100000</td>
<td>1415.51</td>
</tr>
</tbody>
</table>
Outline

1. Introduction to QueryTables
2. Downloading equity data from Yahoo Finance
3. Read data from SQL databases
4. Implementing VBA worksheet formulas
5. VBA chart creation
J. Green, S. Bullen, R. Bovey, M. Alexander

*Excel 2007 VBA Programmer's Reference*

Wiley, 2007

- Chapter 21
ODBC Setup

- ODBC Data Source Administrator
  - User Data Sources
    - Name: AdventureWorks
      - Driver: Microsoft Access Driver (*.mdb, *.accdb)
    - Name: dBASE Files
      - Driver: Microsoft Access dBASE Driver (*.dbf, *.nkr)
    - Name: Excel Files
      - Driver: Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm)
    - Name: Iahman
      - Driver: Microsoft Access Driver (*.mdb, *.accdb)
    - Name: IahmanSQLite
      - Driver: SQLite3 ODBC Driver
    - Name: Visio Database Samples
      - Driver: Microsoft Access Driver (*.mdb)

- ODBC Microsoft Access Setup
  - Data Source Name: Iahman
  - Description:
  - Database: C:\AMATH500\SQL\Iahman591.mdb
    - Select
    - Create
    - Repair
    - Compact
  - System Database
    - None
    - Database

- Create New Data Source
  - Select a driver for which you want to set up a data source.
    - Name
      - Driver para a Microsoft Visual FoxPro
      - Microsoft Access dBASE Driver (*.dbf, *.nkr, *.db)
      - Microsoft Access Driver (*.mdb)
      - Microsoft Access Driver (*.mdb, *.accdb)
      - Microsoft Access Paradox Driver (*.db)
      - Microsoft Access Text Driver (*.txt, *.csv)
      - Microsoft Access-TreeView (*.mdb)
      - Microsoft dBASE Driver (*.dbf)

  - Back
  - Finish
  - Cancel

Guy Yollin  (Copyright © 2012)  Data Access with SQL, Excel & VBA  Data access with VBA 22 / 45
Sub goatHitters()
Dim varConn As String
Dim varSQL As String

Range("A1").CurrentRegion.ClearContents

varConn = "ODBC;DSN=lahman"

varSQL = _
"SELECT " & _
"Master.nameFirst, Master.nameLast, SUM(H) AS Hits " & _
"FROM Batting, Master " & _
"WHERE Master.playerID=Batting.playerID " & _
"GROUP BY Master.playerID, Master.nameLast, Master.nameFirst " & _
"HAVING SUM(H) > 3000 " & _
"ORDER BY SUM(H) DESC;"

With ActiveSheet.QueryTables.Add(_
    Connection:=varConn, _
    Destination:=Range("A1"))
    .CommandText = varSQL
    .Refresh BackgroundQuery:=False
End With
Sub goatHomers()
Dim varConn As String
Dim varSQL As String

Cells.Clear

varConn = "ODBC;DBQ=C:\Projects\VBA\lahman591.mdb;Driver={Microsoft Access Driver (*.mdb)}"

varSQL = "SELECT " & 
"  "nameFirst, nameLast, yearID, HR " & 
"  FROM Batting, Master " & 
"  WHERE Master.playerID=Batting.playerID " & 
"  AND HR>51 " & 
"  ORDER BY HR DESC;"

With ActiveSheet.QueryTables.Add(Connection:=varConn, Destination:=Range("A1"))
  .CommandText = varSQL
  .Refresh BackgroundQuery:=False
End With

End Sub

Note DSN-less ODBC connection
Sub goatStealers()

Dim varConn As String
Dim varSQL As String
Cells.Clear

varConn = "ODBC;DSN=lahmanSQLite"

varSQL = _
  "SELECT " & _
  "Master.nameFirst, Master.nameLast, SUM(SB) AS StolenBases " & _
  "FROM Batting, Master " & _
  "WHERE Master.playerID=Batting.playerID " & _
  "GROUP BY Master.playerID " & _
  "HAVING SUM(SB) > 600 " & _
  "ORDER BY SUM(SB) DESC;"

With ActiveSheet.QueryTables.Add( _
  Connection:=varConn, _
  Destination:=Range("A1"))
  .CommandText = varSQL
  .Refresh BackgroundQuery:=False
End With
Sub goatHomersADO()

' Requires reference to Microsoft ActiveX Data Objects xx Library

Dim Cn As ADODB.Connection, Rs As ADODB.Recordset
Dim MyConn, varSQL As String

Dim Rw As Long, Col As Long, c As Long
Dim MyField, Location As Range

Cells.Select
Selection.Clear
Range("A1").Select
'Set destination
Set Location = [A1]
'Set source
MyConn = "C:\Projects\VBA\lahman591.mdb"
'Create query
varSQL = __
"SELECT " & __
"Master.nameFirst, Master.nameLast, SUM(HR) AS HomeRuns " & __
"FROM Batting, Master " & __
"WHERE Master.playerID=Batting.playerID " & __
"GROUP BY Master.playerID, Master.nameFirst, Master.nameLast " & __
"ORDER BY SUM(HR) DESC;"

...
Read from Access database via ADO

..."Create recordset"
Set Cn = New ADODB.Connection
With Cn
  .Provider = "Microsoft.Jet.OLEDB.4.0"
  .Open MyConn
  Set Rs = .Execute(varSQL)
End With

"Write recordset to results area"
Rw = Location.Row
Col = Location.Column
c = Col
Do Until Rs.EOF
  For Each MyField In Rs.Fields
    Cells(Rw, c) = MyField
    c = c + 1
  Next MyField
  Rs.MoveNext
  Rw = Rw + 1
  c = Col
Loop
Set Location = Nothing
Set Cn = Nothing
End Sub
Outline

1. Introduction to Query Tables
2. Downloading equity data from Yahoo Finance
3. Read data from SQL databases
4. Implementing VBA worksheet formulas
5. VBA chart creation
Section references

John Walkenbach
*Excel 2010 Power Programming with VBA*
Sams, 2010

- Chapter 8
Black-Scholes pricing formulas

Price of a call option:

\[ c = S_0 N(d_1) - X e^{-rT} N(d_2) \]

Price of a put option:

\[ p = X e^{-rT} N(-d_2) - S_0 N(-d_1) \]

Where:

\[ d_1 = \frac{\ln(S_0/X) + (r + \sigma^2/2)T}{\sigma \sqrt{T}} \]

\[ d_2 = d_1 - \sigma \sqrt{T} \]

\( N \equiv \text{The cumulative normal distribution function} \)

\( S_0 = \text{spot price} \)
\( X = \text{strike price} \)
\( \sigma = \text{volatility} \)
\( T = \text{time to expiration} \)
\( r = \text{risk-free rate} \)
Public Function BlackScholes(CallPutFlag As String, _
S As Double, X As Double, T As Double, r As Double, _
v As Double) As Double

Dim d1 As Double, d2 As Double

d1 = (Log(S / X) + (r + v ^ 2 / 2) * T) / (v * Sqr(T))
d2 = d1 - v * Sqr(T)

If CallPutFlag = "c" Then
    BlackScholes = S * CND(d1) - X * Exp(-r * T) * CND(d2)
ElseIf CallPutFlag = "p" Then
    BlackScholes = X * Exp(-r * T) * CND(-d2) - S * CND(-d1)
End If

End Function

Source: http://www.espenhaug.com/black_scholes.html
Polynomial approximation to normal CDF

\[
N(x) = \begin{cases} 
1 - N'(x)(a_1k + a_2k^2 + a_3k^3 + a_4k^4 + a_5k^5) & \text{when } x \geq 0 \\
1 - N(-x) & \text{when } x < 0 
\end{cases}
\]

where

\[
k = \frac{1}{1 + \gamma x}
\]

\[
\gamma = 0.2316419
\]

\[
a_1 = 0.319381530
\]

\[
a_2 = -0.356563782
\]

\[
a_3 = 1.781477937
\]

\[
a_4 = -1.821255978
\]

\[
a_5 = 1.330274429
\]
The cumulative normal distribution function

Public Function CND(X As Double) As Double

Dim L As Double, K As Double
Const a1 = 0.31938153: Const a2 = −0.356563782:
Const a3 = 1.781477937: Const a4 = −1.821255978:
Const a5 = 1.330274429

L = Abs(X)
K = 1 / (1 + 0.2316419 * L)
CND = 1 − 1 / Sqr(2 * Application.Pi()) * Exp(−L^2/2) * (_
    (a1 * K + a2 * K^2 + a3 * K^3 + a4 * K^4 + a5 * K^5)

If X < 0 Then
    CND = 1 − CND
End If

End Function
Black-Shoales option pricing

<table>
<thead>
<tr>
<th>Option Parameters</th>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call/Put</td>
<td>CallPutFlag</td>
<td>c</td>
</tr>
<tr>
<td>Spot Price</td>
<td>S</td>
<td>100</td>
</tr>
<tr>
<td>Strike Price</td>
<td>X</td>
<td>105</td>
</tr>
<tr>
<td>time to expiration</td>
<td>T</td>
<td>0.25</td>
</tr>
<tr>
<td>Risk-Free Rate</td>
<td>r</td>
<td>0.05</td>
</tr>
<tr>
<td>Volatility</td>
<td>v</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Option Price: $2.48

<table>
<thead>
<tr>
<th>Option Parameters</th>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call/Put</td>
<td>CallPutFlag</td>
<td>p</td>
</tr>
<tr>
<td>Spot Price</td>
<td>S</td>
<td>100</td>
</tr>
<tr>
<td>Strike Price</td>
<td>X</td>
<td>95</td>
</tr>
<tr>
<td>time to expiration</td>
<td>T</td>
<td>0.5</td>
</tr>
<tr>
<td>Risk-Free Rate</td>
<td>r</td>
<td>0.1</td>
</tr>
<tr>
<td>Volatility</td>
<td>v</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Option Price: $2.95
You can use many (but not all) of Excel’s worksheet functions in your VBA code

- The WorksheetFunction object, which is contained in the Application object, holds all the worksheet functions that you can call from your VBA procedures.

- To use a worksheet function in a VBA statement, just precede the function name with the object reference:
  - Application.WorksheetFunction
Public Function BlackScholesVer2(CallPutFlag As String, _
S As Double, X As Double, T As Double, r As Double, _
v As Double) As Double

Dim d1 As Double, d2 As Double

d1 = (Log(S / X) + (r + v ^ 2 / 2) * T) / (v * Sqr(T))
d2 = d1 - v * Sqr(T)

If CallPutFlag = "c" Then
    BlackScholesVer2 = S * WorksheetFunction.NormSDist(d1) - _
    X * Exp(-r * T) * WorksheetFunction.NormSDist(d2)
ElseIf CallPutFlag = "p" Then
    BlackScholesVer2 = X * Exp(-r * T) * _
    WorksheetFunction.NormSDist(-d2) - _
    S * WorksheetFunction.NormSDist(-d1)
End If

End Function
1. Introduction to QueryTables
2. Downloading equity data from Yahoo Finance
3. Read data from SQL databases
4. Implementing VBA worksheet formulas
5. VBA chart creation
Section references

John Walkenbach
*Excel 2010 Power Programming with VBA*
Sams, 2010
- Chapter 18

Duane Birnbaum and Michael Vine
*Excel VBA Programming for the Absolute Beginner, 3rd Edition*
Thomson Course Technology, 2007
- Chapter 9
Record column chart creation process

1. Select A1:A17, H1:H17
2. Click record macro
3. Insert clustered columns chart
4. Right click and Format Axis
5. Change Axis Labels to Low
6. Align text to rotate 270
7. Delete legend
8. Click stop recording

Guy Yollin (Copyright © 2012)
Sub RecordPlotReturns()
    ActiveSheet.Shapes.AddChart.Select
    ActiveChart.SetSourceData
        Source := Range("'msft'!$A$1:$A$17,'msft'!$H$1:$H$17")
    ActiveChart.ChartType = xlColumnClustered
    ActiveChart.Axes(xlCategory).Select
    ActiveSheet.ChartObjects("Chart 10").Activate
    Selection.TickLabelPosition = xlLow
    ActiveSheet.ChartObjects("Chart 10").Activate
    ActiveChart.Legend.Select
    Selection.Delete
End Sub

- Hard-coded range
- Hard-coded chart name (will not be correct the next time)
- Nothing recorded regarding the text alignment
Sub RecordedMod1()
    Dim DataRange As Range
    Set DataRange = Selection
    ActiveSheet.Shapes.AddChart.Select
    ActiveChart.SetSourceData Source:=DataRange
    ActiveChart.ChartType = xlColumnClustered
    ActiveChart.Axes(xlCategory).Select
    Selection.TickLabelPosition = xlLow
    ActiveChart.Legend.Select
    Selection.Delete
End Sub

To-Do:
- align x-axis labels
- add a y-axis label
- change the title
- change chart position
Sub PlotReturns()
    Dim MyChart As Chart
    Dim DataRange As Range
    Set MyChart = ActiveSheet.Shapes.AddChart(xlColumnClustered, 375, 5, 350, 250).Chart
    With MyChart
        .ChartTitle.Text = "Daily Stock Returns"
        .Axes(xlCategory).TickLabels.Orientation = 90
        .Axes(xlCategory).TickLabelPosition = xlLow
        .HasLegend = False
        .Axes(xlValue, xlPrimary).HasTitle = True
    End With
End Sub
Sub PlotPrices()
    Dim MyChart As Chart
    Dim DataRange As Range
    Set DataRange = Selection
    Set MyChart = ActiveSheet.ChartObjects.Add(375, 5, 350, 250).Chart
    With MyChart
        .SetSourceData Source:=DataRange
        .ChartType = xlLineMarkers
        .ChartTitle.Text = "Daily Stock Price"
        .Axes(xlCategory).TickLabels.Orientations = 90
        .HasLegend = False
        .Axes(xlValue, xlPrimary).HasTitle = True
    End With
End Sub
Plotting candle stick charts

Sub PlotOHLC()
    Dim MyChart As Chart
    With MyChart
        .ChartStyle = 4
        .HasTitle = True
        .ChartTitle.Text = "Daily Stock Price"
        .Axes(xlCategory).TickLabels.Orientation = 90
        .Axes(xlCategory).TickLabelPosition = xlLow
        .HasLegend = False
        .Axes(xlValue, xlPrimary).HasTitle = True
    End With
End Sub
http://depts.washington.edu/compfin