BUSINESS ANALYTICS CLUB

Workshop Series 9.5

Intro to R – Applications in Trading & Macro Research

Janet Ye
Shantanu Joshi

Learning Objective

1. Learn the basics of foreign exchange (FX) or currency marketplace, and the role of a trader
2. Introduce basic programming concepts:
    - Variable and assignment
    - Reading CSV Files
    - Basic Types and Operations
    - Linear Regression Model
3. Give you the tools to teach yourself more R, Trading, and Research

You Are Apple

- Huge operations in China that generate significant portion of firm's revenue
- Revenues collected are in CNY, but salaries and expenses generated are paid in USD
- Multinational companies hedge their currency risks
  - Reasons
  - Benefits
- Apple calls a trader at a bank to hedge risks and transact

You Are A Hedge Fund

- After running quantitative analysis, your fund believes that CNY will depreciate against USD
- $2 mm if your speculation is correct
- You decide to bet on the movement of CNY

Generally Speaking

Agency	Principle
Client's Interest	Personal Interest
Sell Side	Buy Side
Hedging	Speculation

Today we act as a hedge fund

Creating Software

- A human programmer writes a series of instructions ("lines of code") for the computer to execute sequentially
- That code is written in some programming language
  - Programming languages share many common features, but also differ in their syntax and even their "expressiveness" for certain ideas

What is R?

- R is an open source statistical programming language
  - Focus is on data applications – organizing and cleaning data, fitting statistical models, making plots
  - Extensive user-created package "ecosystem"
- Widely used by statisticians, data-miners, economists, business analysts, etc.
  - A complement to SQL and Excel spreadsheet analysis, all of which will be taught in future workshops

Go here to download R

bit.ly/bacdata

"R and Trading" folder

The Command Line

This is a cursor. You can type commands and then see their output after hitting "Enter".

Examples

```
> 3 + 2
  [1] 5
> 100 / 5
  [1] 20
> 12 + 12 + 12
  [1] 36
```

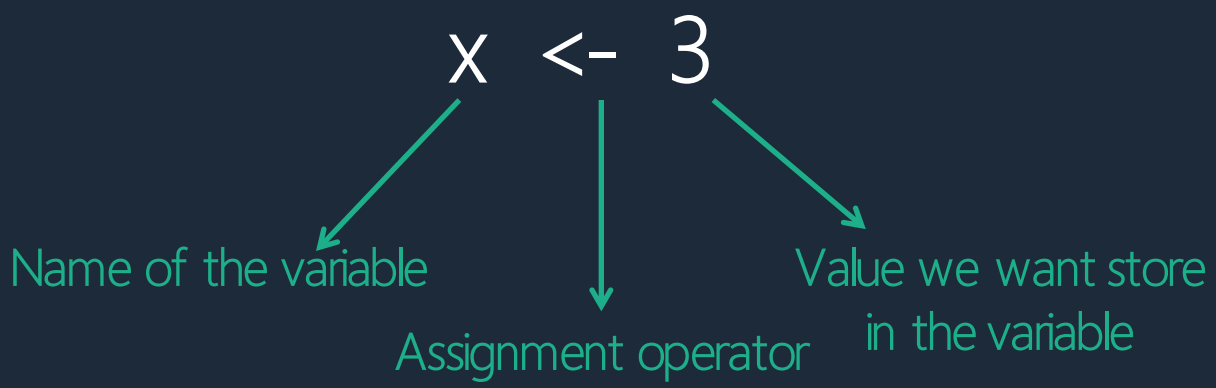
Arithmetic

+ addition
- subtraction
/ division
* multiplication
^ powers
%% modulus (remainder for division)

You try it:
What is 42 squared plus 7?

```
> 42 ^ 2 + 7
  [1] 1771
```

Assigning a Value to a Variable



You try it:
Assign a value of ‘42’ to the variable name ‘y’

```
> y <- 42
```

Accessing Value Later

```
> y
  [1] 42
```

Further Arithmetic Examples

```
> y + 2
  [1] 44
> y ^ 2
  [1] 1764
```

Update a Variable

```
> y <- 42
> y <- y - 1
> print(y)
  [1] 41
```

You try it:
Store ‘42’ to ‘y’ then
store ‘y’ to ‘x’

```
> y <- 42
> x <- y
```

Back to FX

- Goal is to predict the movement of CNY
  - Fit linear regression
  - Diagnostics
  - Prediction
- Develop a trading strategy
  - Risk appetite
  - How "sure" we are about our prediction
  - Take profit
  - Stop loss

Reading CSV Files

```
> before <- read.csv(file.choose(), head =
                TRUE, sep = ",")
```

Go to where the files are saved, and open before_deval.csv

Let's look at the first few rows in the `fxdata` data frame

```
> head(before)
```

	date	cny	dxy
1	7/30/14	6.1720	81.214
2	7/31/14	6.1738	81.432
3	8/1/14	6.1795	81.456
4	8/4/14	6.1793	81.302
5	8/5/14	6.1708	81.328
6	8/6/14	6.1634	81.328

CNY and DXY

- CNY
  - Onshore Chinese Yuan, not tradable
  - Tightly managed, rate set by the government
- DXY
  - Index measuring relative strength of USD
  - Consists of a basket of US trade partners' currencies
  - Goes up as USD becomes relatively stronger
- DXY gives us some clues as to where CNY is going

Let's investigate the data frame

First, we need to extract the column. There are three ways to do this


```
> cny <- before$cny
> cny <- before[["cny"]]
> cny <- before[,"cny"]

> dxy <- before$dxy
> dxy <- before[["dxy"]]
> dxy <- before[,"dxy"]
```

Vector

- Data frame columns are stored in a data type called vector
- A vector is a 1D array of values, indexed by integers starting at 1
- Most functions in R operate on vectors

Accessing elements of a vector

```
> cny[[1]]
[1] 6.172
```

Get the first element of the vector

```
> dxy[[5]]
[1] 81.328
```

Get the fifth value of the vector

```
> length(cny)
[1] 283
```

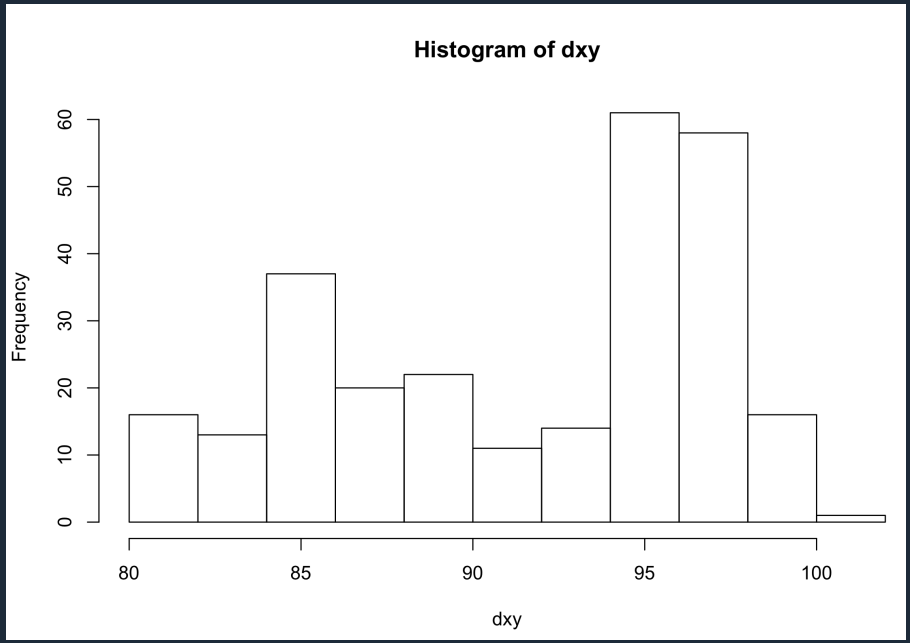
To see how many elements are contained in the vector

```
> cny[[length(cny)]]
[1] 6.2097
```

To see the last element

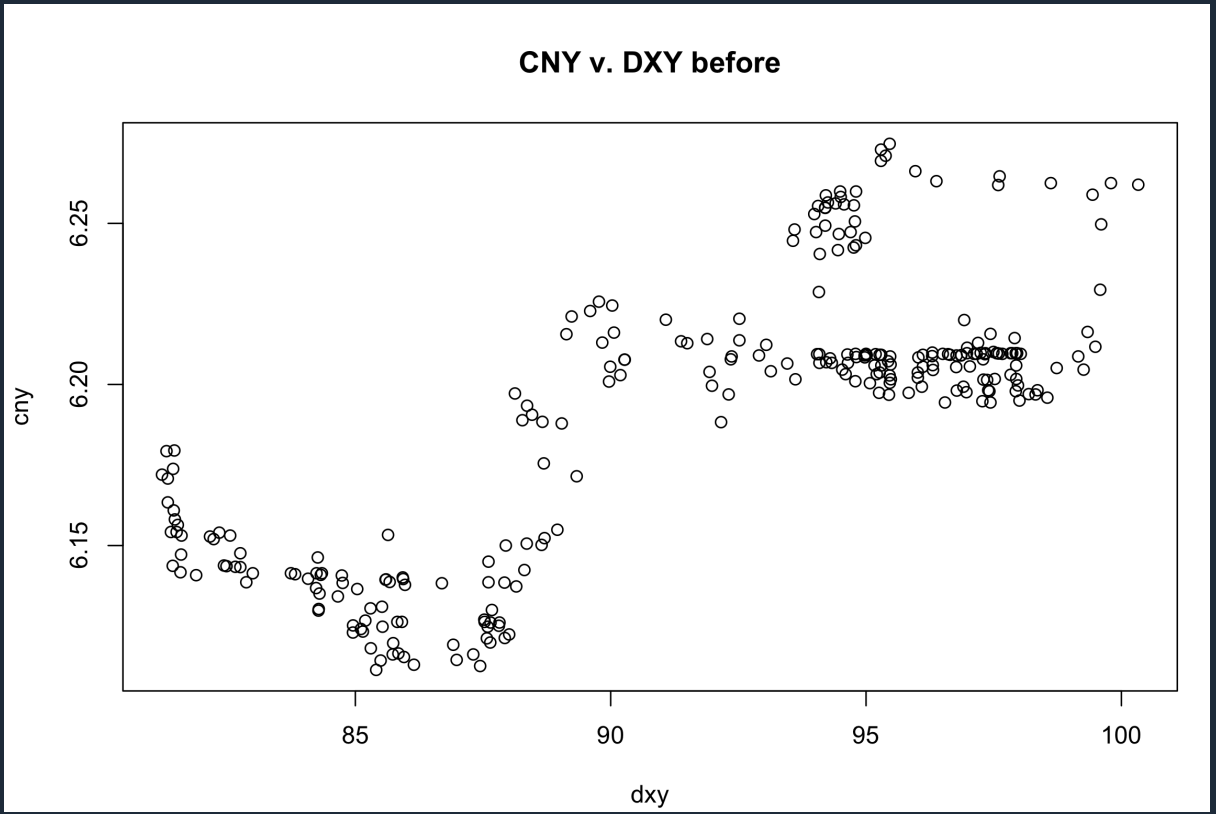
We can use the `hist` command to make a histogram of the values stored in a vector

```
> hist(dxy)
```



Plot DXY against CNY

```
> plot(dxy, cny, main="CNY v. DXY before”)
```



We want to fit a linear regression model
response variable: `cny`
predictors: `dxy`

```
> model <- lm(cny~dxy)
```

The formula syntax is:

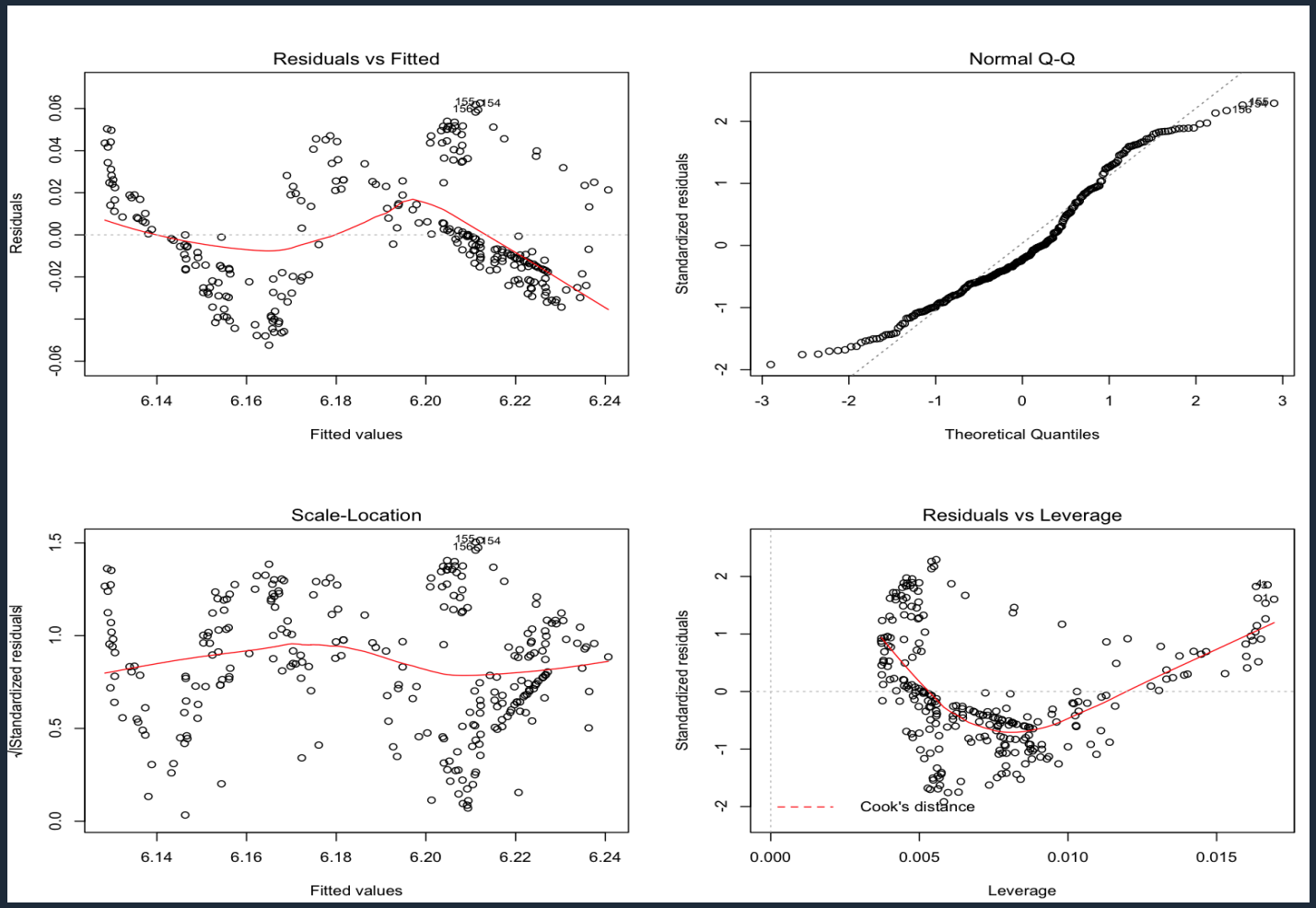
```
y ~ x1 + x2 + x3 + ...
```

We can get a fitted value, coefficient estimates, standard errors, t statistics, and p value with the `summary` command

```
> summary(model)
Call:
lm(formula = cny ~ dxy)

Residuals:
      Min        1Q    Median        3Q       Max
-0.052402 -0.018980 -0.005987  0.021084  0.062619

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 5.6515599  0.0278754  202.74   <2e-16 ***
dxy         0.0058717  0.0003038   19.33   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0274 on 267 degrees of freedom
Multiple R-squared:  0.5831,Adjusted R-squared:  0.5816
F-statistic: 373.5 on 1 and 267 DF,  p-value: < 2.2e-16
```

Plotting a fitted model shows us regression diagnostics

```
> plot(model)
```



We want to get a prediction for `cny` using second set of `dxy` data

```
> dxy_after <- read.csv(file.choose(),
head = TRUE, sep = ",")
```

Go to where the files are saved, and open after_deval_dxy.csv

We can then pass this data frame and the model to the `predict` command

```
> predicted_cny <- predict(model,  dxy_after)
> predicted_cny

        1        2        3        4        5
6.222045 6.222815 6.216779 6.217853 6.218293
        6        7        8        9       10
6.219973 6.221347 6.217348 6.215123 6.209416
       11       12       13       14
6.199575 6.206609 6.209956 6.212950
```

Let's take look at the actual `cny` dataframe

```
> actual_cny_df <-
read.csv(file.choose(), head=TRUE,
sep=",")
```

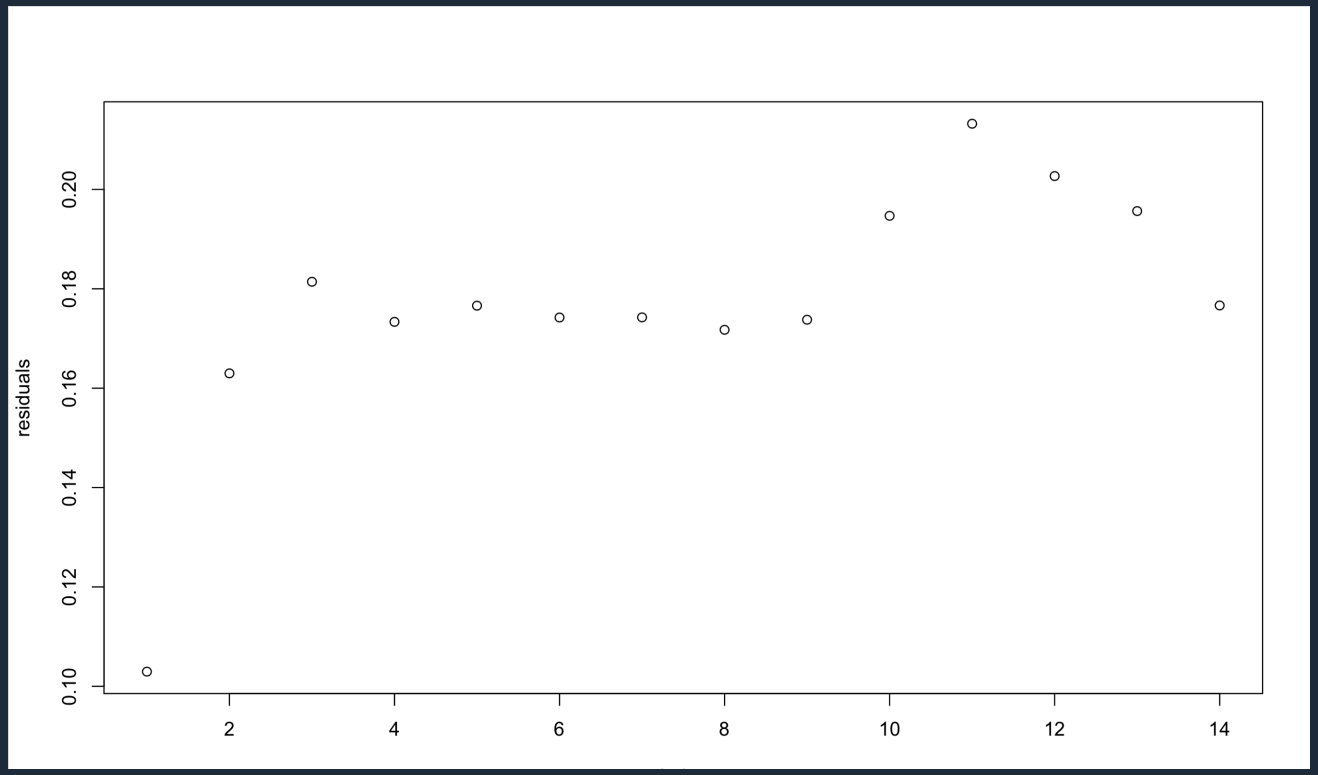
Go to where the files are saved, and open

after_deval_actual_cny.csv

```
> actual_cny <- actual_cny_df$cny
```

We can investigate the errors of our prediction of `cny`

```
> residuals <- actual_cny - predicted_cny
> plot(residuals)
```

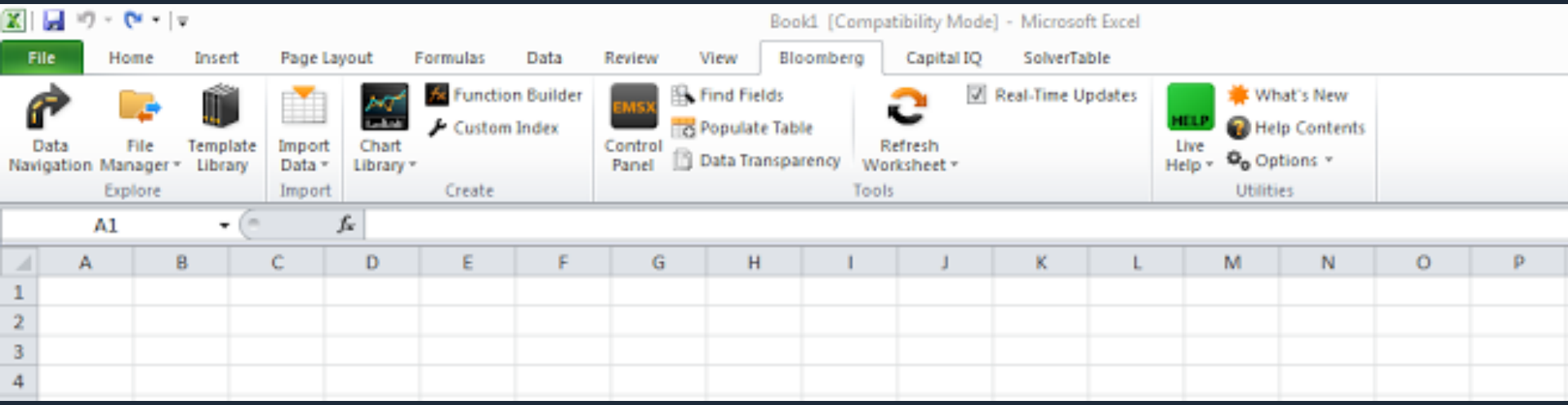


Our prediction is a bad fit. We should adjust our prediction

```
> shift <- mean(residuals)
> shift
[1] 0.17673

> adjusted_prediction <- predicted_cny +
shift
> mean(actual_cny - adjusted_prediction)
[1] 3.172085e-16
```

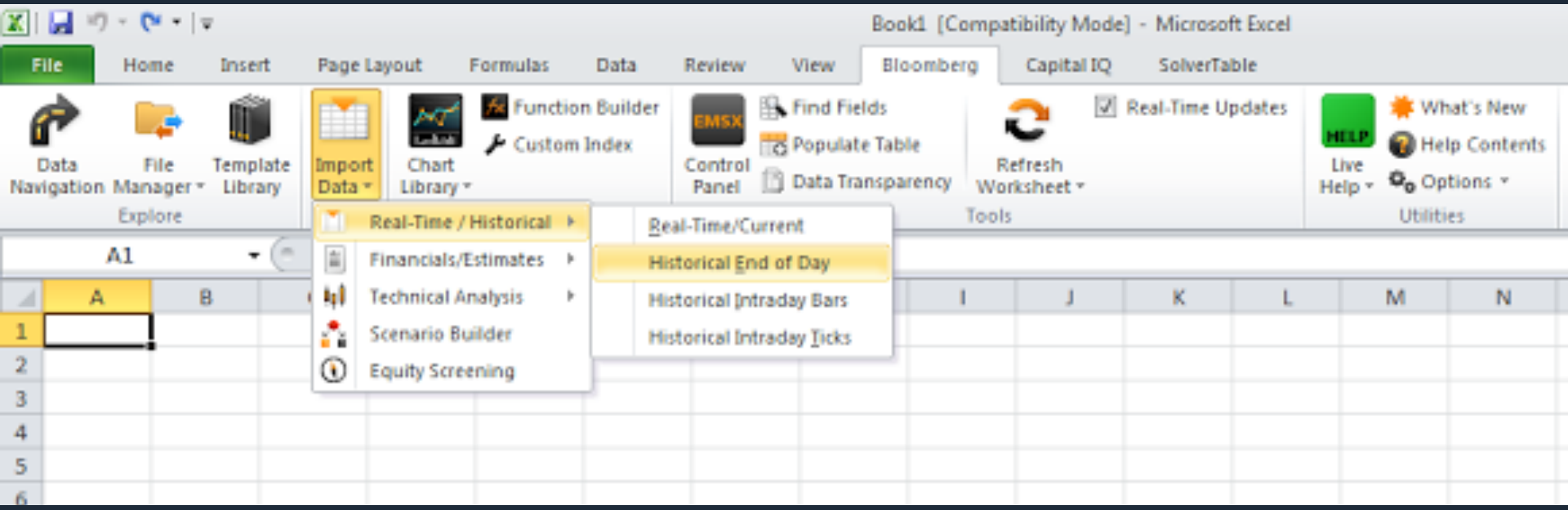
The shifted model looks much better with small residuals

Appendix: Accessing Data From Bloomberg Excel



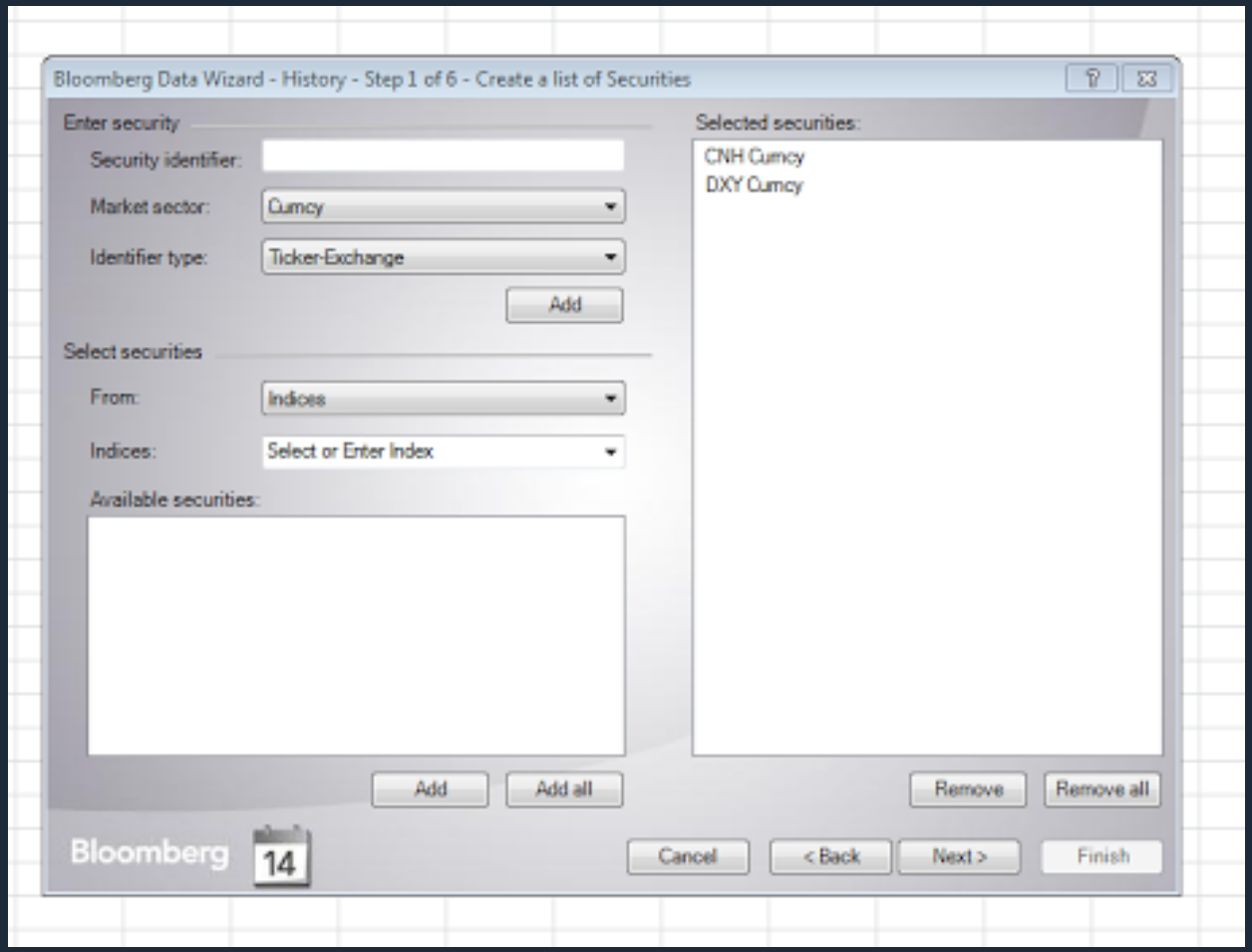
Click the Bloomberg pane in excel

Appendix: Accessing Data From Bloomberg Excel

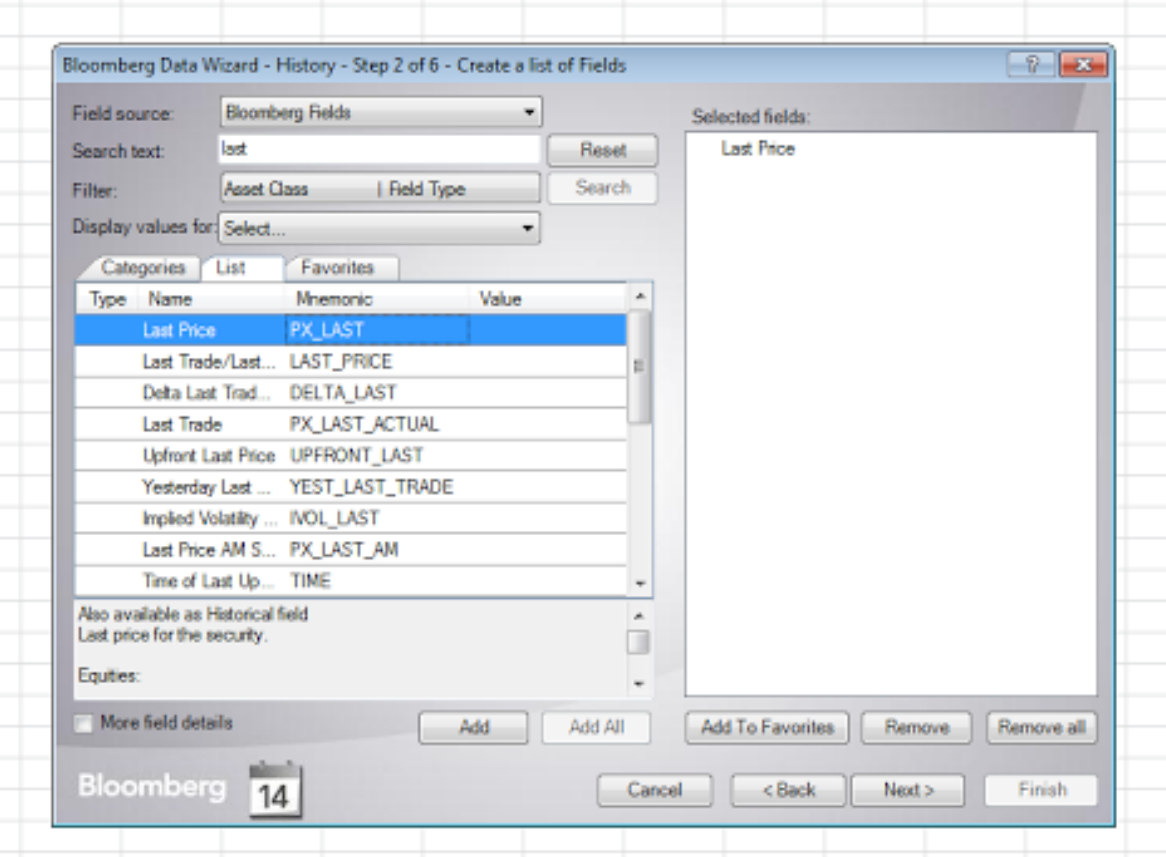


Historical End of Day in Drop Down

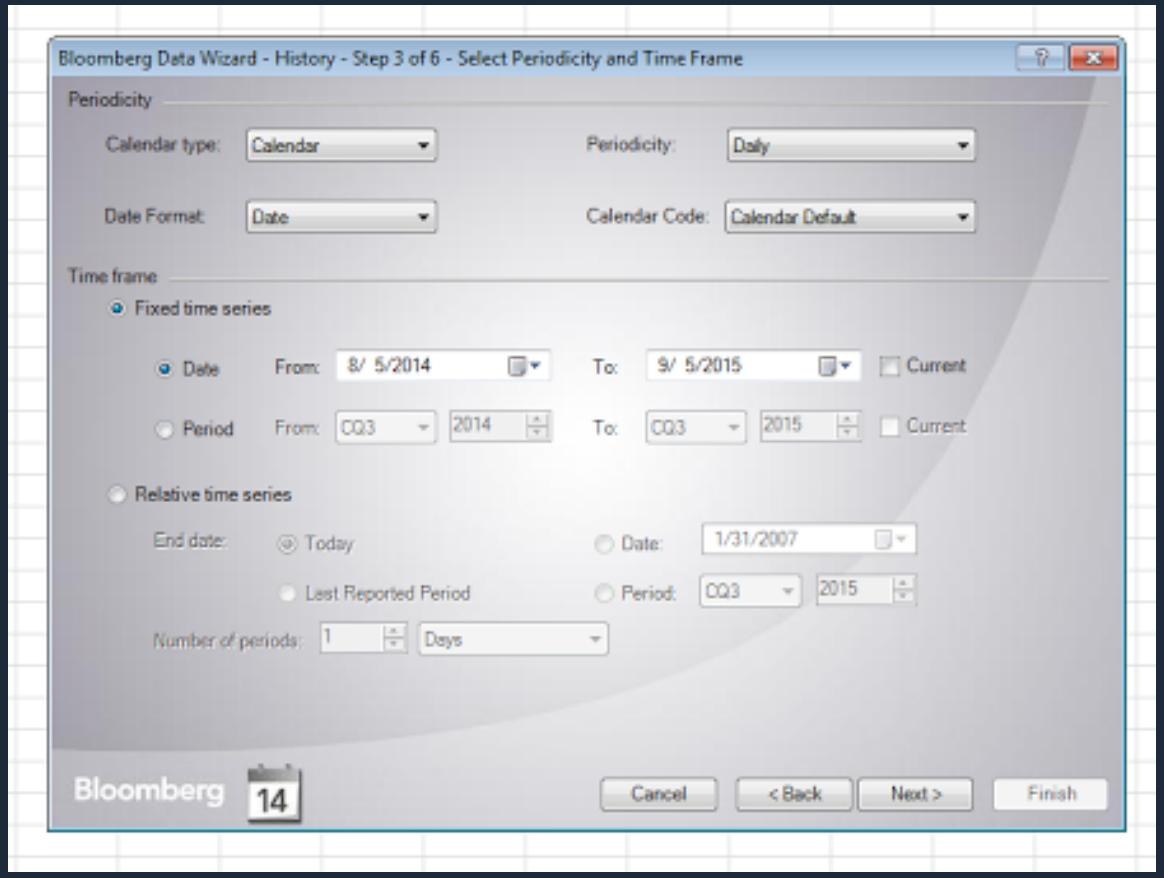
Appendix: Accessing Data From Bloomberg Excel



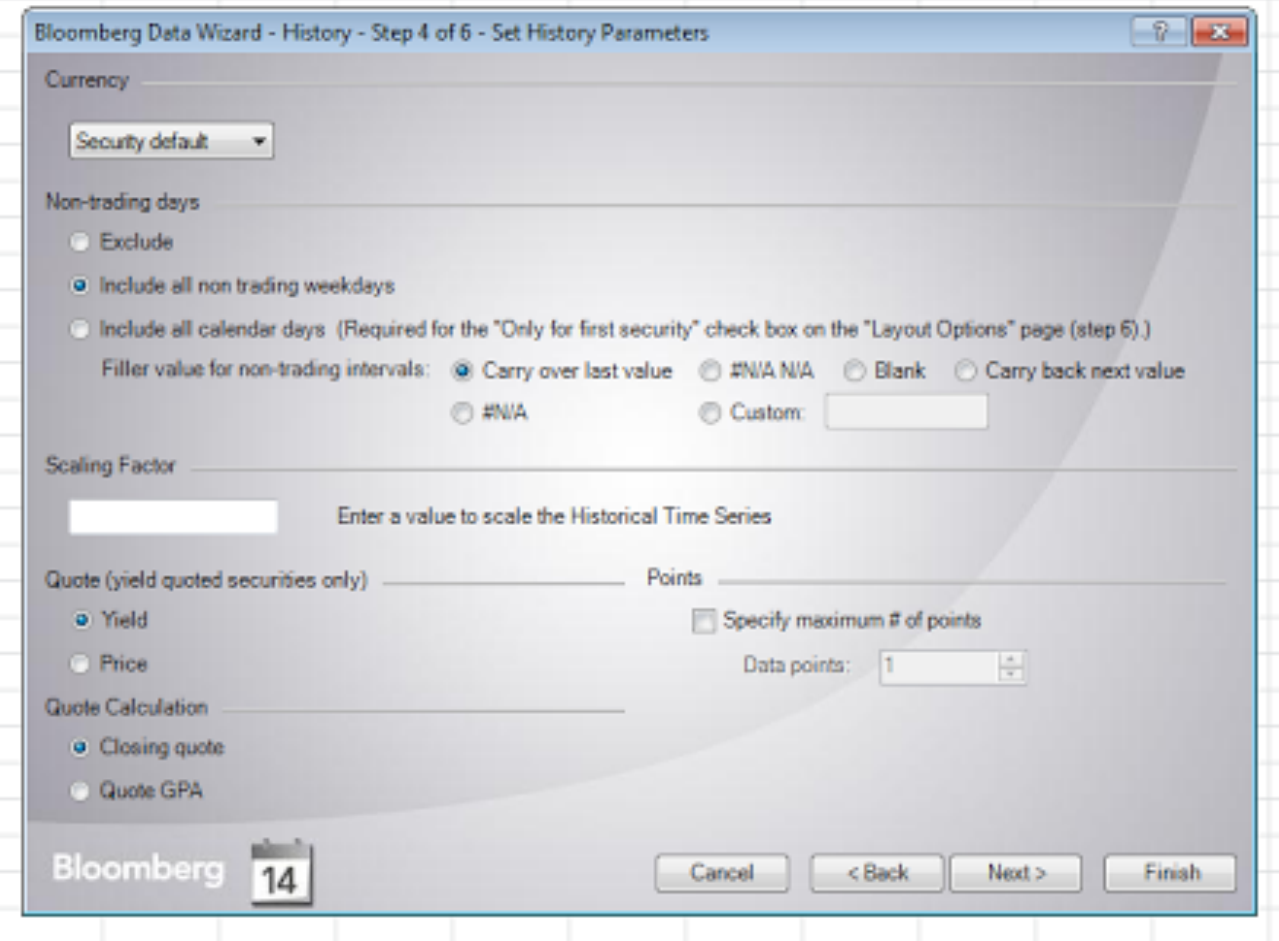
Appendix: Accessing Data From Bloomberg Excel



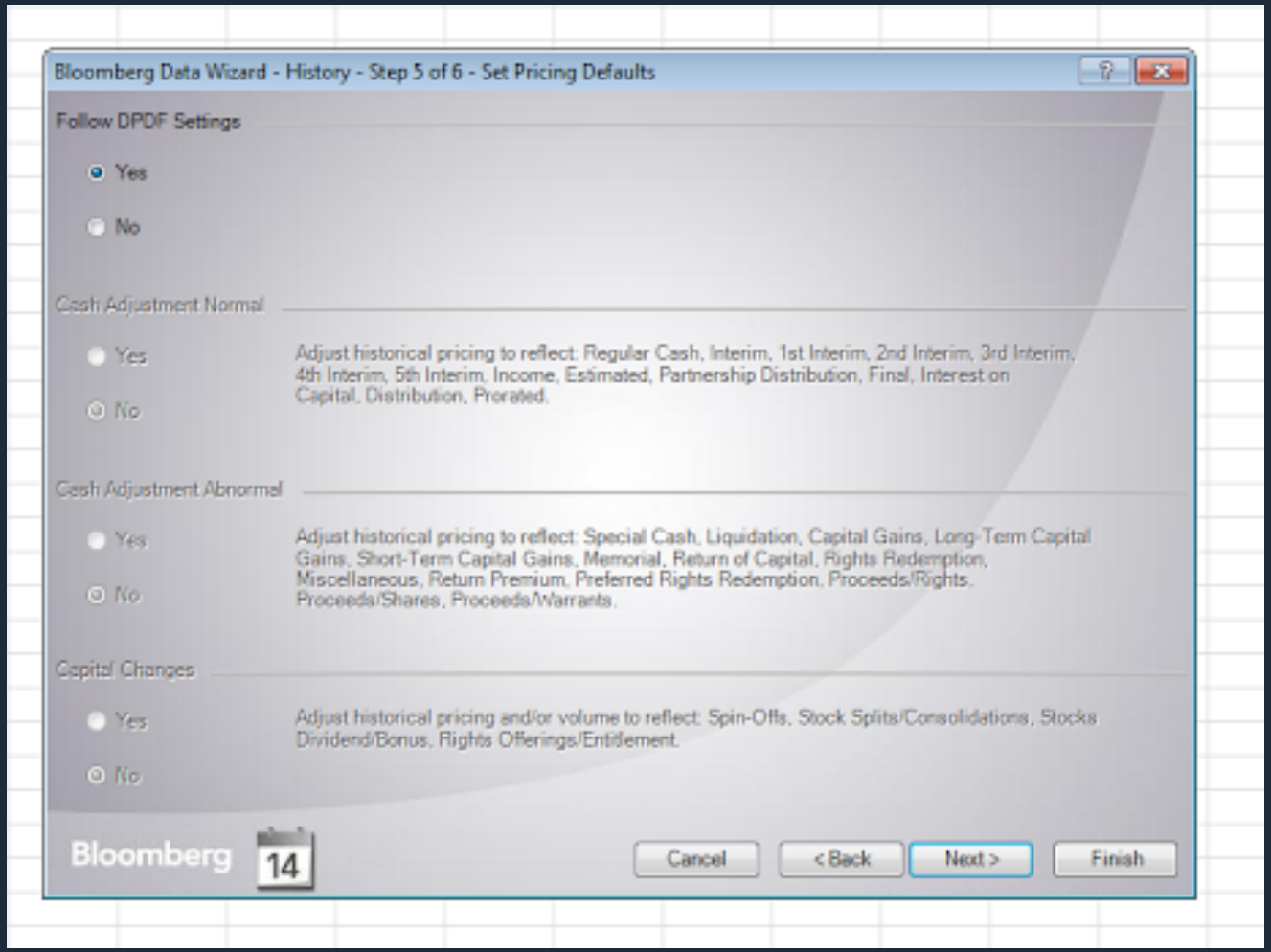
Appendix: Accessing Data From Bloomberg Excel



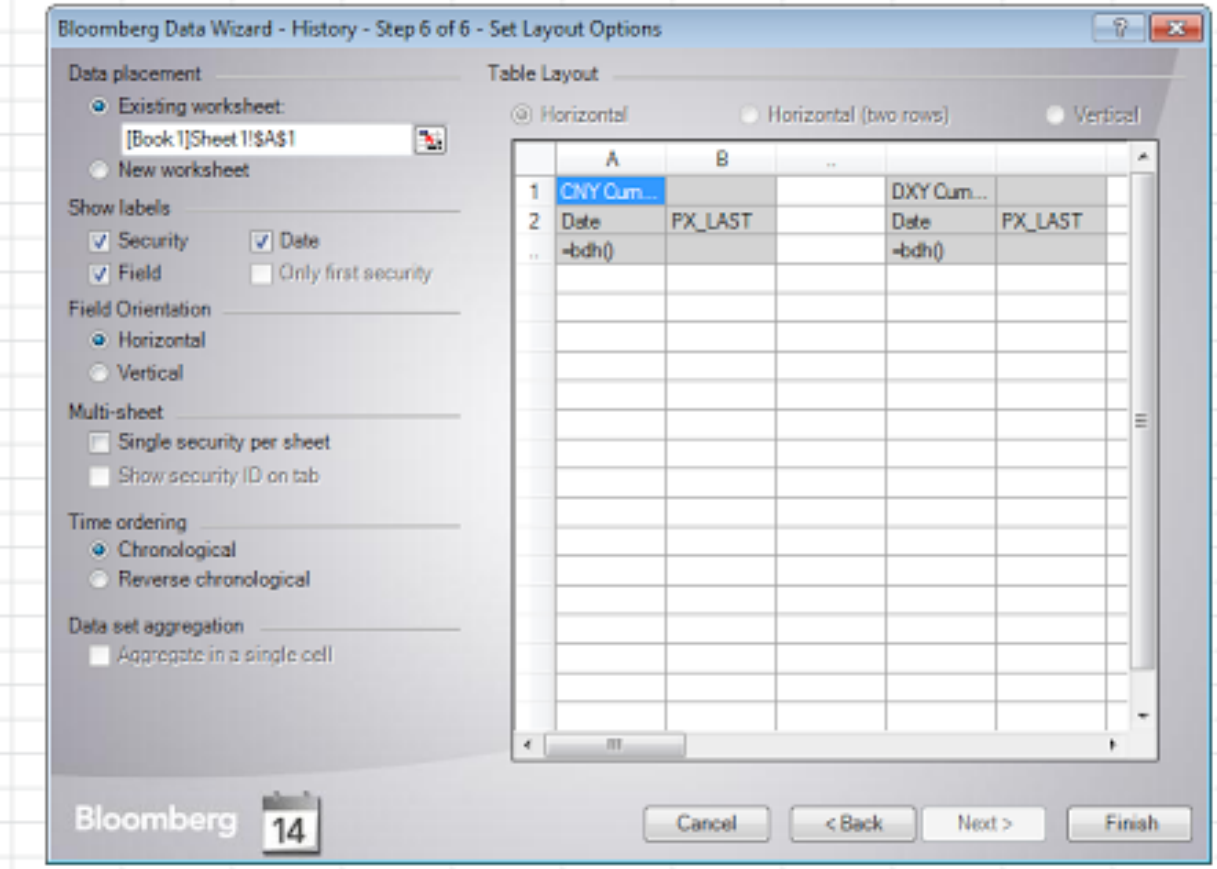
Appendix: Accessing Data From Bloomberg Excel



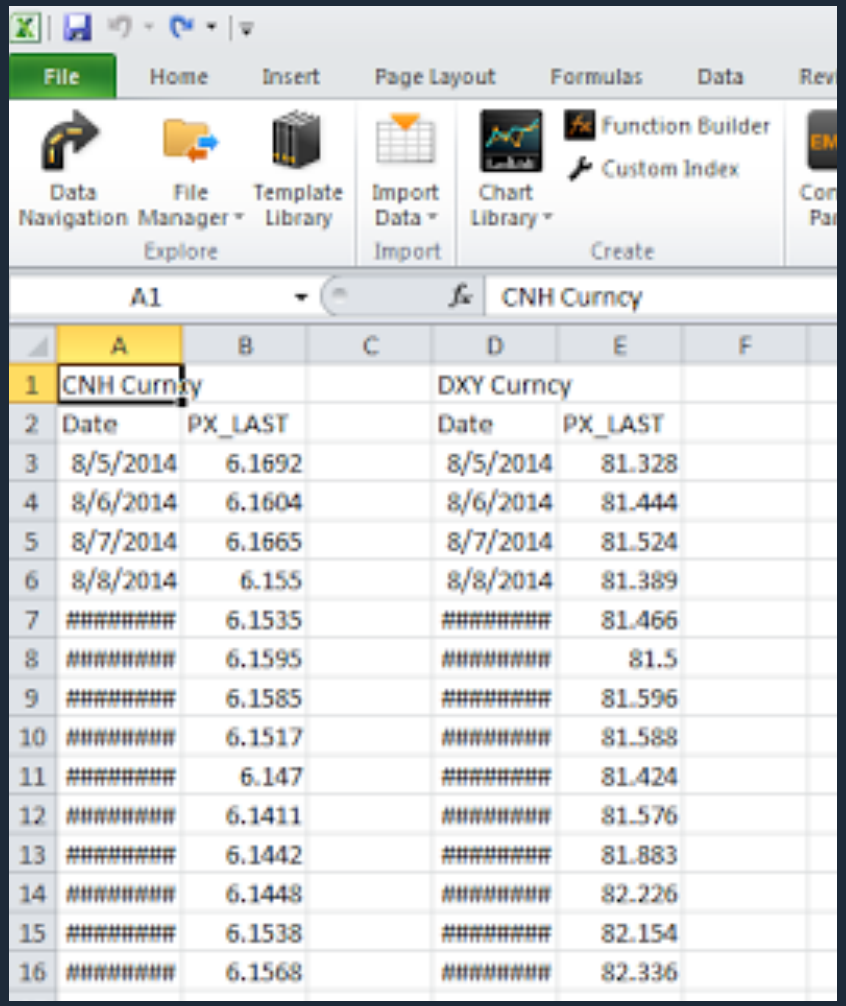
Appendix: Accessing Data From Bloomberg Excel



Appendix: Accessing Data From Bloomberg Excel



Appendix: Accessing Data From Bloomberg Excel



	A	B	C	D	E	F
1	CNH Curncy			DXY Curncy		
2	Date	PX_LAST		Date	PX_LAST	
3	8/5/2014	6.1692		8/5/2014	81.328	
4	8/6/2014	6.1604		8/6/2014	81.444	
5	8/7/2014	6.1665		8/7/2014	81.524	
6	8/8/2014	6.155		8/8/2014	81.389	
7	########	6.1535		########	81.466	
8	########	6.1595		########	81.5	
9	########	6.1585		########	81.596	
10	########	6.1517		########	81.588	
11	########	6.147		########	81.424	
12	########	6.1411		########	81.576	
13	########	6.1442		########	81.883	
14	########	6.1448		########	82.226	
15	########	6.1538		########	82.154	
16	########	6.1568		########	82.336	

Useful Resources

- bit.ly/bacdata BAC Workshop R.pdf
- R documentation
- Step by step R tutorial
- Great local integrated development environment (IDE) for R
  - http://www.rstudio.com/
- SWIRL in R
  - A software package that turns the R console into an interactive learning envrionment
  - In R, enter command

```
> install.packages("swirl")
> library(swirl)
> swirl()
> install_from_swirl("R Programming")
```

Acknowledgement

John Horton

Assistant Professor of Information, Operations and Management Sciences

Patrick Perry

Assistant Professor of Information, Operations and Management Sciences