



Populating an ALFRED-type Database

Federal Reserve Bank of St. Louis – Research Division (10/16/2006)

© 2006, The Federal Reserve Bank of St. Louis. Articles may be reprinted, reproduced, published, distributed, displayed, and transmitted in their entirety if copyright notice, author name(s), and full citation are included. Abstracts, synopses, and other derivative works may be made only with prior written permission of the Federal Reserve Bank of St. Louis.



TABLE OF CONTENTS

Section	Page
Step 1: Set Up Data Spreadsheet	1
Step 2: Enter Data and Check for Keying Errors	3
Step 3: Create and Check Plot of Levels	4
Step 4: Create and Check Revision Count	6
Step 5: Create Necessary Files for Final Check	8
Step 6: Create Text Files for Web Group	10
Step 7: Send Necessary Files to the Web Group	12
Step 8: Web Group Adds Information from Step 7 to the Development Server	13
Step 9: Work with Web Database Output	20
Step 10: Make Adjustments/Corrections	28
Step 11: Repeat Steps 8-10 Until No Corrections Are Required	28
Step 12: Web Group Makes Live on Public Website	29
Appendix: Create Text Files for Web Group from Haver Archives	30

After collecting the sources that contain the data you need as well as a complete list of release dates for the period that you are preparing to create, you are now ready to create the first step, the data spreadsheet.

If you will be creating vintage data for several series from the same release, you can use the spreadsheet that results from keying the first variable as a template for the rest of the series. Just make a copy of the workbook after step 4, open the file, remove the raw data, and save it as template.xls.

To create one vintage series, complete all of the steps below.

STEP 1: SET UP DATA SPREADSHEET

1. Open a new Excel workbook.

Document Series Definition

2. Create a “Series Info” sheet. (Double-click on the tab at the bottom to change the name from “Sheet1” to “Series Info”.)
3. Enter descriptive information about the series on the “Series Info” sheet. This information can include the series mnemonic, a short description, source, frequency, units, seasonal adjustment, notes about vintage sources, etc.

Title:	Real Final Sales
Series ID:	FINSLC1
Source:	U.S. Department of Commerce, Bureau of Economic Analysis
Seasonal Adjustment:	Seasonally Adjusted Annual Rate
Frequency:	Quarterly
Current Units:	Billions of Constant Dollars
BEA Release Date	Beginning with 1970, all BEA release dates were obtained directly from BEA, from an electronic file they sent us Virginia Mannering <Virginia.Mannering@bea.gov>, on 8/15/2005 1:32:00 PM Prior to 1970, BEA release dates were obtained by searching the Wall Street Journal and the New York Times, using the ProQuest Also, Katrina Stierholz visited the St. Louis Public Library to use their search engines for dates we could not obtain via ProQuest Where BEA release date = #N/A, it means that BEA did not release any data during that month, hence there was no BEA release
Benchmark revision date	are highlighted in bright yellow
Errata	July-82

Document Benchmark and Annual Revision Sources

4. Create a “Benchmark Info” sheet by selecting “Insert-Worksheet” from the pull-down menus. (Double-click on the tab at the bottom to change the name from “Sheet2” to “Benchmark Info”.)
5. Enter information about the sources that contain the complete benchmarks and annual revisions on the “Benchmark Info” sheet. This information will include release date, publication date, observation periods changed in the benchmark, any changes in units that occurred, etc.

Real NIPA Benchmark Documentation			
Column Date	Revision begins	Units	Reference 1
23-Jul-59	Mar-56	Gross National Product (in 1954 Billions of dollars)	Survey of Current Business. Jul-59 p. 10
22-Jul-60	Mar-57		Gross National Product, National Income and Personal Income, 1947-59. p.10-11
19-Jul-61	Mar-58		Gross National Product, National Income, and Personal Income 1947-60. p.9
20-Jul-62	Mar-59		Gross National Product, National Income and Personal Income, 1947-61 pp. 8-9
22-Jul-63	Mar-60		Survey of Current Business Jul-63, p. 15.
16-Jul-64	Mar-61		Gross National Product, National Income and Personal Income 1947-63. p.8-9
19-Aug-65	Mar-47	Gross National Product (in 1958 Billions of dollars)	Table 2 Gross National Product in Constant Dollars. Survey of Current Business. PP. 26-27. August 1965
15-Jul-66	Jun-60		Table 1.2. Gross National Product in Constant Dollars. Survey of Current Business. P. 11. July 1966

The data will be entered on a tab titled “1RawData”. First, you will need to set up this sheet.

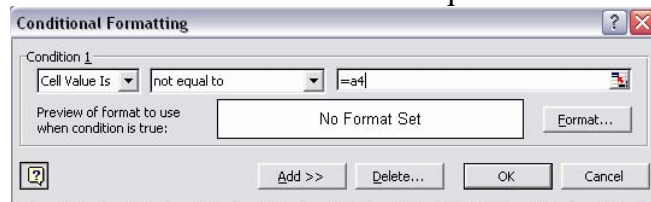
6. Create a sheet in your workbook named “1RawData” by selecting “Insert-Worksheet” from the pull-down menus. (Double-click on the tab at the bottom to change the name from “Sheet3” to “1RawData”.)

Set-up Column and Row Headers

7. In Row 1 enter information to document the source of the data for each column
8. In Row 2 enter the release dates for the periods that will be entered by hand. The first release date should be placed in cell B2.
9. Highlight in yellow any cell in Row 2 for columns that will contain benchmark or annual revision data.
10. Row 3 contains the release dates in a format that will be used in later steps.
 - a. Enter the word “Date” in A3.
 - b. Enter this formula to create the new date format in cell B3
=“_”&YEAR(B2)&“_”&TEXT(MONTH(B2),“00”)&“_”&TEXT(DAY(B2),“00”).
 - c. Copy this formula from B3 and paste to the right for all of the release dates.
11. In Column A, starting in Row 4, enter the observation periods for the data you will be keying in. Use the Excel date format *mm/dd/yy*.

Set-up Conditional Formatting to Mark Revisions from Column to Column

12. Set up the conditional formatting to mark changes in the data values from the previous column. This will help catch keying errors as you are entering the data.
 - a. Select B4
 - b. Select “Format – Conditional Formatting” from the pull-down menus.
 - c. Set “Condition 1” to be Cell Value Is – not equal to – =A4



- d. Click the “Format” button.
 - e. Select the “Pattern” tab and then select a color to highlight the changing cells.
 - f. Click “OK” to exit “Format Cells” dialog.
 - g. Click “OK” to exit “Conditional Formatting” dialog.
13. Copy cell B4, select the entire range that will contain data (to the right: to the end of your release date list; down: to the end of your observation period list), and then select “Edit-Paste” from the pull-down menus. This will copy the “conditional format” over all of the cells that will eventually contain data.

Create a Range for Use in Future Steps of the Process

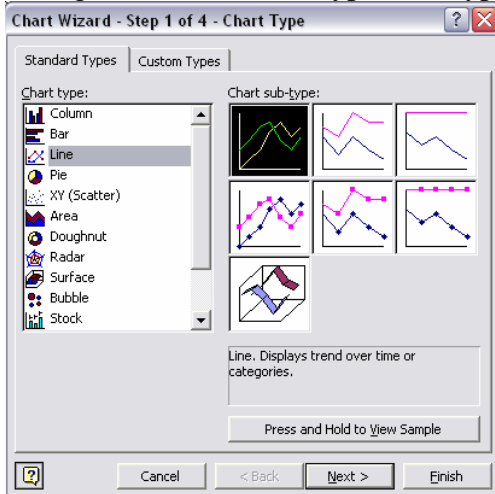
14. Create a range name starting with A3 and ending with the final observation on the final release date in this sheet. (The programs in this documentation will use the range name “rawdata”.)
 - a. Select the range that you wish to name
 - b. Select “Insert – Name – Define” from the pull-down menus.
 - c. Type the name of your range (rawdata) in the “Names in workbook:” box.
 - d. Click “OK”
15. Save this file with the name of the mnemonic you are working with.

STEP 3: CREATE and CHECK PLOT OF LEVELS

A quick plot of the levels will point out any large keying errors that may remain.

Create the Chart

1. Select the range “rawdata”.
 - a. Press F5 to open the Go To box
 - b. Select “rawdata” in the “Go To:” list
 - c. Click “OK”
2. Select “Insert-Chart” from the pull-down menus
3. Change the “Chart Type” to Line
4. Change the “Chart Sub-Type” to a type with no symbols



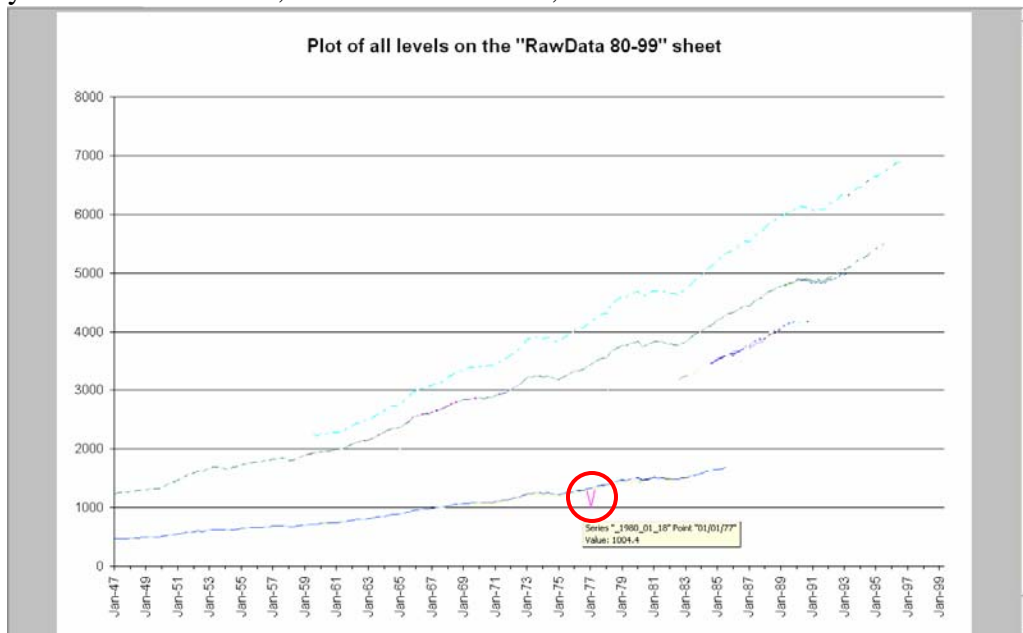
5. Click “Next” to complete Step 1 of the Chart Wizard
6. Click “Next” to complete Step 2 of the Chart Wizard
7. In Step 3 of the Chart Wizard, select the “Legend” tab and un-check the “Show legend” box.
8. Click “Next” to complete Step 3 of the Chart Wizard
9. In Step 4, select “As new sheet:” and give the sheet a name of “2Chart”
10. Click “Finish” to close the Chart Wizard
11. Save the file.

Check the Chart

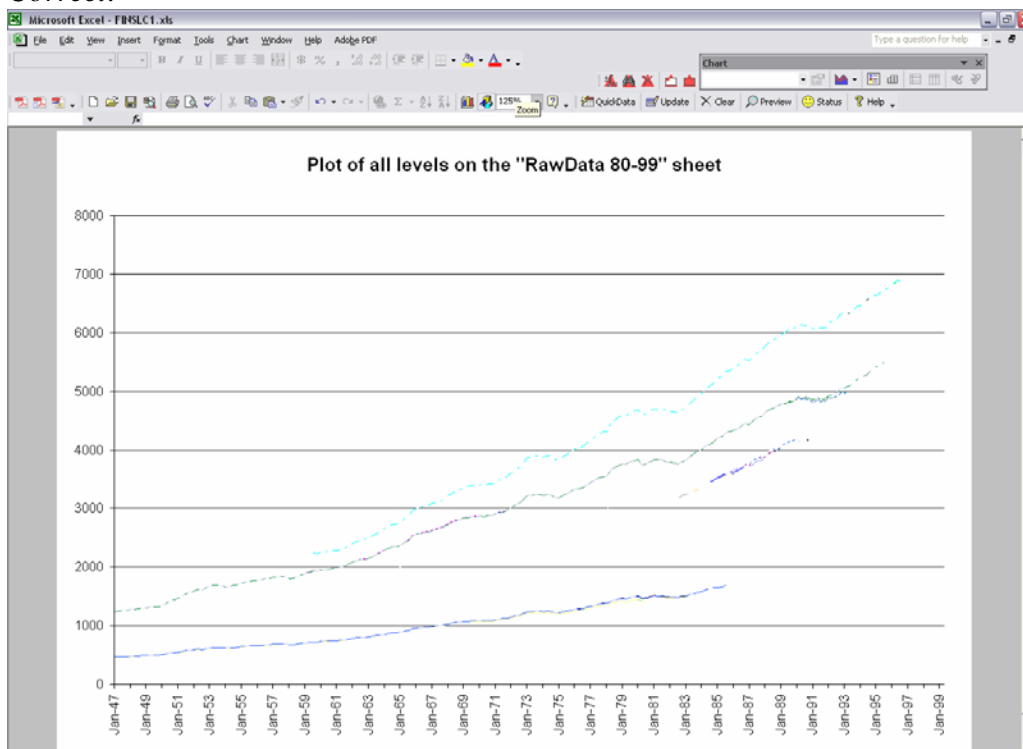
12. Verify that the curves follow the expected pattern for the series that you are working on.

Problem:

To pinpoint a “bad” observation, hover over the spot on the chart. A pop-up box will tell you the column label, the observation date, and the observation value.



Correct:

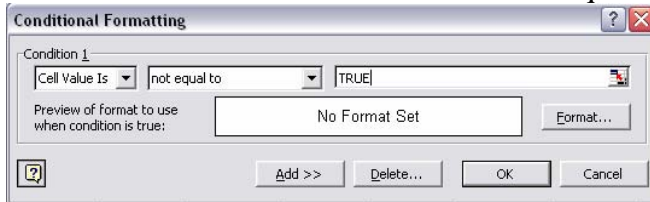


STEP 4: CREATE AND CHECK REVISION COUNT

This serves as a double-check for the conditional formatting from step #1.

Mark the Cells that Revise

1. Create a sheet in your workbook named “3Verify” by selecting “Insert-Worksheet” from the pull-down menus. (Double-click on the tab at the bottom to change the name from “Sheet4” to “3Verify”.)
2. Copy all of Row 1 from “1RawData” and paste to Row 1 in “3Verify”.
3. Copy the observation dates from Column A in “1RawData” and paste to Column A of “3Verify”. The first date should be place in Row 4.
4. In C4 of “3Verify”, enter this formula to mark changes from the previous release date -
EXACT('1RawData'!C4,'1RawData'!B4)
5. Set- up conditional formatting for cell C4 to mark a “FALSE” value.
 - a. Select C4
 - b. Select “Format – Conditional Formatting” from the pull-down menus.
 - c. Set “Condition 1” to be Cell Value Is – not equal to – TRUE



- d. Click the “Format” button.
 - e. Select the “Pattern” tab and then select a color to highlight the changing cells.
 - f. Click “OK” to exit “Format Cells” dialog.
 - g. Click “OK” to exit “Conditional Formatting” dialog.
6. Copy C4 and paste it down through the last observation period.
 7. Select the formulas you just created in column C, copy this column, and paste across to the right for all of the release dates shown in Row 1.
 8. Visually verify that the revision pattern for your series holds. For example, for the NIPA series, a new observation should be added (advance estimate), the last observation should be revised (preliminary and final estimates), 3-4 years should be revised (annual revision) or the entire column should revise (benchmark). If you see an odd revision, check the raw data you entered in “1RawData”.
 9. Save the file.

Create a sum of all the observations for each release date

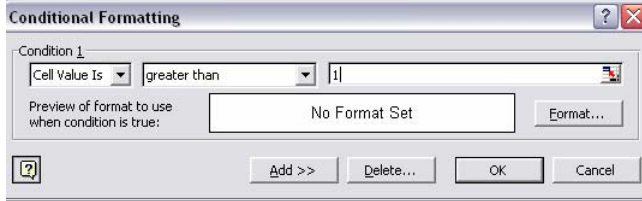
This will be used in a later step to compare the data after it is loaded on the web

10. In cell B2 of “3Verify”, enter a formula to sum all of the observations entered and stored for that release date in the sheet “1Rawdata”. (*=SUM('1RawData'!B4:B10000)*)
11. Copy B2 and paste across to the right for all of the release dates
12. Save the file.

Create a count of the number of revisions occurring on each release date

13. In Cell C3 of “3Verify”, enter a formula to count the number of “FALSE”s in each column
(*=COUNTIF(C4:C10000,FALSE)*)
14. Apply a conditional format to this formula. The condition will depend on the revision pattern of the series that you are working on. For example, with the NIPA series, except for benchmarks, all other columns should only show 1 revision.

- a. Select C3
- b. Select “Format – Conditional Formatting” from the pull-down menus.
- c. Set “Condition 1” to be Cell Value Is – greater than – 1



- d. Click the “Format” button.
 - e. Select the “Pattern” tab and then select a color to highlight the cells that don’t fit the revision pattern for your series.
 - f. Click “OK” to exit “Format Cells” dialog.
 - g. Click “OK” to exit “Conditional Formatting” dialog.
15. Copy C3 and paste across to the right for all of the release dates.
 16. Visually verify that the revision pattern for your series holds. In the NIPA series, if more than one observation revises, check to see if the release date is also an annual revision or a benchmark. If it’s not, check the raw data in the “1RawData” sheet.
 17. Save the file.

This is what the “3Verify” sheet will like.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	BEA Release Date	01/18/80	02/22/80	03/19/80	04/18/80	05/20/80	06/18/80	07/18/80	08/19/80	09/19/80	10/17/80	11/19/80	12/23/80	01/21/81	02/19/81	03/18/81
2	Sum of Values	116287.3	115990.3	116291.1	117735.4	117735.6	117735.5	119144.0	119143.1	119141.5	120560.4	120559.3	121998.6	123488.9	123491.0	123491.3
3	# of revisions		2	2	1	1	1	1	1	1	1	1	135	1	1	1
115	10/01/74		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE
116	01/01/75		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE
117	04/01/75		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE
118	07/01/75		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE
119	10/01/75		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE
120	01/01/76		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE
121	04/01/76		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE
122	07/01/76		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE
123	10/01/76		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE
124	01/01/77		FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE
125	04/01/77		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE
126	07/01/77		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE
127	10/01/77		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE
128	01/01/78		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE
129	04/01/78		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE
130	07/01/78		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE
131	10/01/78		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE
132	01/01/79		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE
133	04/01/79		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE
134	07/01/79		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE
135	10/01/79		FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE
136	01/01/80		TRUE	TRUE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE
137	04/01/80		TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE
138	07/01/80		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE
139	10/01/80		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE
140	01/01/81		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
141	04/01/81		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
142	07/01/81		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
143	10/01/81		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
144	01/01/82		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
145	04/01/82		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE

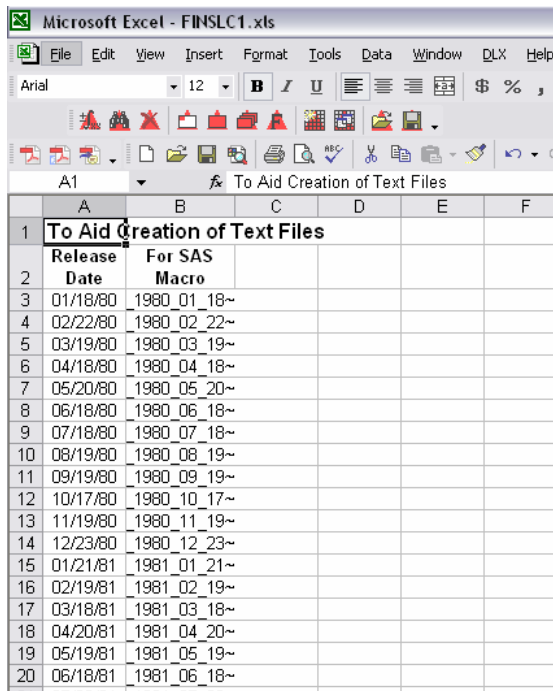
STEP 5: CREATE NECESSARY FILES FOR FINAL CHECK

There is nothing to check in this step. However, it is a necessary step that makes creating text files and checking your spreadsheet data against the data loaded on the web possible.

A) Create “4Aid Creation of Text Files” sheet

1. Insert another spreadsheet in your workbook – “4Aid Creation of Text Files” – by selecting “Insert-Worksheet” from the pull-down menus.
2. Copy the release dates from Row 2 of the “1RawData” sheet, starting with column B.
3. Paste the dates into column A of this sheet.
 - a. Select cell A3
 - b. Select “Edit-Paste Special”
 - c. Select “Values” and “Transpose”
 - d. Click “OK”
4. In cell B3 of “4Aid Creation of Text Files”, enter this formula
=“_”&YEAR(A3)&“_”&TEXT(MONTH(A3),“00”)&“_”&TEXT(DAY(A3),“00”)&“~”. This formula will create information needed by a SAS job to follow.
5. Copy and paste cell B3 down for all of the release dates in column A.
6. Save the file.

“4Aid Creation of Text Files” should look like this.



The screenshot shows a Microsoft Excel spreadsheet with the following data:

	A	B	C	D	E	F
1	To Aid Creation of Text Files					
2	Release Date	For SAS Macro				
3	01/18/80	1980_01_18~				
4	02/22/80	1980_02_22~				
5	03/19/80	1980_03_19~				
6	04/18/80	1980_04_18~				
7	05/20/80	1980_05_20~				
8	06/18/80	1980_06_18~				
9	07/18/80	1980_07_18~				
10	08/19/80	1980_08_19~				
11	09/19/80	1980_09_19~				
12	10/17/80	1980_10_17~				
13	11/19/80	1980_11_19~				
14	12/23/80	1980_12_23~				
15	01/21/81	1981_01_21~				
16	02/19/81	1981_02_19~				
17	03/18/81	1981_03_18~				
18	04/20/81	1981_04_20~				
19	05/19/81	1981_05_19~				
20	06/18/81	1981_06_18~				

B) To use this information to create text files

1. Open in a text editor the SAS program that creates individual text files
2. Come back to the sheet “4Aid Creation of Text Files”; select B3 through the row that contains the last release date, and copy.
3. Go back to the SAS program and paste this into the macro call at the end of the program (after the text “%outer (indate=” but before the final “);”)
4. Save the updated SAS program

C) Create “5Comparing to Web-Loaded Files” sheet

This will be used to compare the data in the spreadsheet file to the text files after they are loaded onto the web, to ensure that the files were transferred correctly and assigned to the correct release date.

1. Insert another spreadsheet in your Excel workbook – “5Comparing to Web-Loaded Files” – by selecting “Insert-Worksheet” from the pull-down menus.
2. Copy the release dates, starting in column B, from Row 1 of the “3Verify” sheet.
3. Transpose and paste the dates into this sheet.
 - a. Select cell A3
 - b. Select “Edit-Paste Special”
 - c. Select “Values” and “Transpose”
 - d. Click “OK”
4. Copy the observation sums, starting in column B, from Row 2 of the “3Verify” sheet.
5. Transpose and paste the sums into this sheet.
 - a. Select cell B3
 - b. Select “Edit-Paste Special”
 - c. Select “Values” and “Transpose”
 - d. Click “OK”
6. Copy the revision counts, starting in column B, from Row 3 of the “3Verify” sheet.
7. Transpose and paste the counts into this sheet.
 - a. Select cell C3
 - b. Select “Edit-Paste Special”
 - c. Select “Values” and “Transpose”
 - d. Click “OK”
8. Save the Excel file.

“5Comparing to Web-Loaded Files” sheet should look like this.

The screenshot shows a Microsoft Excel spreadsheet with the following data:

1	For Comparison After Web Load		
2	Release Date	Sum of values	Number of Revisions
3	01/18/80	116287.3	0
4	02/22/80	115990.3	2
5	03/19/80	116291.1	2
6	04/18/80	117735.4	1
7	05/20/80	117735.6	1
8	06/18/80	117735.5	1
9	07/18/80	119144	1
10	08/19/80	119143.1	1
11	09/19/80	119141.5	1
12	10/17/80	120560.4	1
13	11/19/80	120559.3	1
14	12/23/80	121998.6	135
15	01/21/81	123488.9	1
16	02/19/81	123491	1
17	03/18/81	123491.3	1
18	04/20/81	125006.3	1
19	05/19/81	125009.6	1
20	06/18/81	125009.1	1
21	07/22/81	126508.5	1

STEP 6 – CREATE TEXT FILES FOR WEB GROUP FROM SPREADSHEET DATA

Vintage data files are organized by directories containing the archive year and date.

Consider the file path: ~/yyyy/yyyy-mm-dd/data/**/filename. ‘yyyy’ represents the archive year, ‘yyyy-mm-dd’ represents the archive date, “**” represents the type of data- such as “gdp” and “monetary”, and ‘filename’ represents the series id/mnemonic.

This structure is shown in the image below. This image shows the files as they appear on the web development server. Note the subdirectory “test_ignore” (defined later). Filenames followed by “.~1~” are previous versions of files that have been renamed so that they are ignored. Also note the directory structure.

```

/2002/2002-01-04/data/gdp_ $ ls
afcedef  aslceed  cbica    expgs    finslcl  gdp92    gpdj     impgsc96  nripdca  pcedgc96  pnfi     slceca
afcehlt  aslcehlt  cofc    expgsc1  finslc96  gdp96    gpdic1   impgsca  pagel1.pdf  pcend    pnfic1  sldef
afceisd  aslceis  civa    expgsc96  finslca  gdpcl    gpdic96  jcxfe    pagel2.pdf  pcendc96  pnfic96  slexpnd
afceisr  aslcemc  cncf    expgsca  fndefx   gdp96    gpdica   jcxfe.~1~  pagel4.pdf  pcepi    pnfica  slinv
afcemc   aslcepos  coe     fdefx    fpi      gdpca    gpdictpi  ndgi     pagel6.pdf  pcepi.~1~  prfi    slinv96
afcemk   aslcet   cofc    FDL22.ZIP  fpicl    gdpctpi  gpdictpi.~1~  ndgic96  page7.pdf  pcepilfe  prfic1  test_ignore
afceneti  aslctax  cpatax  FDL7.ZIP  fpic96   gdpdef   gpdictpi.~2~  netexc   pcdg       pcepilfe.~1~  prfic96  tgdef
afceoasi  aslexpnd  cprofit  fgce     fpica    gdppt    gpsave    netexc92  pcdgcc96  pces    prfica  update
afcess   aslpitax  dcbcca  fgcecl   fpitax   gexpnd   gpsave.~1~  netexc96  pcdgcca   pcesc96  propinc  wascur
afdef    aslptax  dgi     fgcec96  fsdp     gmp      grecpt    netexp    pce       pcesv   psave
afexpnd  aslrecept  dgic96  fgceca   gce      gmp.~1~  gsave     netfi     pcec    pcesvc96  psave.~1~
afgsl    aslrsl  dividend  fgdef    gcecl    gmp92    gsave.~1~  netfi.~1~  pcec96   pcnd     psavert
afrecept  aslstax  dpi     fgexpnd  gcec96   gmp96    gupdate.html  nfcptax  pcecc96  pcndgc96  rentin
aslce    cbi      dpic96  fgrecept  gceca    gmpc96   ibt       nicur     pcecca  pi       slce
aslceaid  cbicl    dspic96  fgsl     gdb      gmpctpi  impgs     nripdcl  pcectpi  pincome  slcecl
aslcecs  cbic96   dspic96  finsal   gdp      gmpdef   impgsc1   nripdc96  pcedg    pmsave  slcec96
/2002/2002-01-04/data/gdp_ $

```

To create individual text files for each column in your data spreadsheet, run the SAS program below.¹

Some values in the program may need to be changed to coincide with your data (values to change are also highlighted in green in the complete program code below).

1. %let var = **finslcl**;
Change *finslcl* to whatever the mnemonic will be for the variable that you are working on.
2. DATAFILE= "H:\Alfred\Proces-1\&var..xls"
Change *H:\Alfred\Process Doc* to equal the name of the directory where you stored your raw data sheet in Step 1.15
3. SHEET="1RawData";
Change *rawdata* to equal the name that you gave your raw data sheet in Step 1.6
4. RANGE="rawdata";
Change *rawdata* to equal the name that you gave your data range in Step 1.14
5. %do i = 1 %to 201;
Change *201* to equal the number of release date columns in your 1RawData sheet.
6. file="H:\Alfred\Proces-1\webdir\&y\&y. -&m. -&d\data\gdp\&var";
command="md H:\Alfred\Proces-1\webdir\&y\&y. -&m. -&d\data\gdp";
Change *H:\Alfred\Process Doc\webdir* to equal the name of the directory where you want to store the raw text files on your PC.
7. format date **yyqp6.**;
Change *yyqp6.* to match the frequency of the data that you are working with. This is the date format for quarterly data.
For monthly data, change *yyqp6.* to *yymm7.*
For weekly and daily data, change *yyqp6.* to *yymmddn8.*

¹ If you have access to archived Haver databases, see appendix for information on using these archives to create the necessary text files.

For calendar year data, change *yyqp6.* to *year4.*

8. %outer (indate=

The values following this line should be the values that you pasted in from Step 5.B.15

```
options nocenter nodate nonumber pagesize=32767 linesize=256 noxwait;
*****;
**VARIABLE NAME/RESULTING TEXT FILE NAME      **;
*****;
  %let var = finslcl1;

*****;
**READ IN SPREADSHEET FILE                    **;
*****;
PROC IMPORT OUT= WORK.inner
  DATAFILE= "H:\Alfred\Proces~1\&var.xls"
  DBMS=EXCEL REPLACE;
  SHEET="1RawData";
  RANGE="rawdata";
  GETNAMES=yes;
  MIXED=YES;
  SCANTEXT=YES;
  USEDATE=YES;
  GUESSINGROWS=100;
RUN;

*****;
**OUTPUT TEXT FILES FOR WEB USE              **;
*****;
%macro outer(indate=);
%do i = 1 %to 201;
  %let y = %substr(%scan(&i, "~"), 2, 4);
  %let m = %substr(%scan(&i, "~"), 7, 2);
  %let d = %substr(%scan(&i, "~"), 10, 2);

  %let id = %scan(&i, "~");

  data newreal&i;
    set inner(keep=date &i where=(&i ne .));
  run;

  **Create directory to store text files**;
  data _null_;
    command="md H:\Alfred\Proces~1\webdir\&y\&y.-&m.-&d\data\gdp\";
    call system(command);
  run;

  **Write text files**;
  proc printto new file="H:\Alfred\Proces~1\webdir\&y\&y.-&m.-&d\data\gdp\&var";
  run;
    proc print data=newreal&i noobs label;
      var date &i;
      format date yyqp6.;
      title " ";
    run;
  proc printto;
  run;
%end;
%mend;

%outer (indate=
1980_01_18~
1980_02_22~
1980_03_19~
1980_04_18~
1980_05_20~
1980_06_18~
1980_07_18~
1980_08_19~
1980_09_19~
1980_10_17~
.
.
.);
```

Output: Your output files should look like the example below. Be sure that your date is in the correct format. Also be sure that the header row begins with the word “DATE”. These consistency of these items is essential for the programs that the web group runs.

```
finslc1 - Notepad
File Edit Format View Help

      _1980_
DATE   01_18
1947.1  463.9
1947.2  468.4
1947.3  470.8
1947.4  471.0
1948.1  473.2
1948.2  480.4
1948.3  483.9
1948.4  490.6
1949.1  491.3
1949.2  496.1
1949.3  495.9
1949.4  496.9
1950.1  507.0
1950.2  517.4
1950.3  534.4
1950.4  532.5
1951.1  549.0
1951.2  553.3
1951.3  570.1
1951.4  578.7
1952.1  584.0
1952.2  594.8
1952.3  592.9
1952.4  605.3
1953.1  618.2
1953.2  621.1
```

STEP 7: SEND NECESSARY FILES TO THE WEB GROUP

1. Using Winzip (or another compression software), zip together all of the text files that were created in the last step. Be sure that the directory structure is included in the zip file.
2. Create a text file that contains the complete list of release dates for the series that you are working on. This should include the release dates in Row 2 of the 1RawData sheet of your raw data spreadsheet as well as the expected set of release dates for the data pulled from the electronic archives.
3. Create a list of unit changes (where necessary). For example, the quarterly real final sales experienced a number of unit changes over all of the available vintages. (The start dates given below are release dates; the end dates are 1 day prior to the following start date.)

	Start	End
Bil. of 1958 \$	1965-08-19	1976-01-19
Bil. of 1972 \$	1976-01-20	1985-12-19
Bil. of 1982 \$	1985-12-20	1991-12-03
Bil. of 1987 \$	1991-12-04	1996-01-18
Bil. of Chained 1992 \$	1996-01-19	1999-10-27
Bil. of Chained 1996\$	1999-10-28	2003-12-09
Bil. of Chained 2000\$	2003-12-10	Current

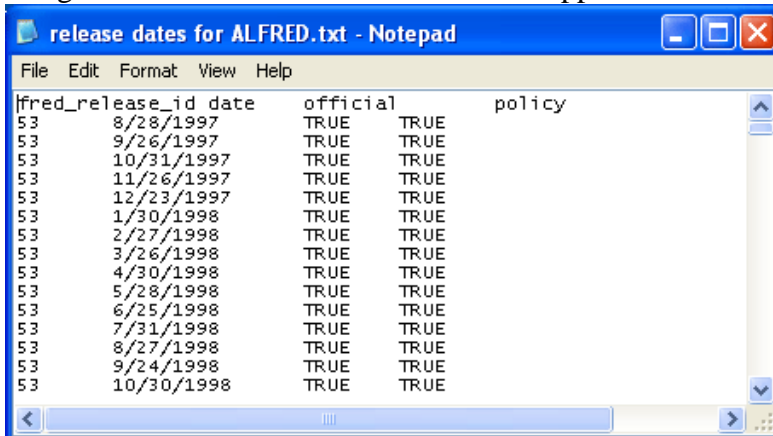
4. Email the zip file, the release date text file, and any unit changes to the web group member that transfers the data to the development server.

STEP 8: WEB GROUP ADDS INFORMATION FROM STEP 7 TO THE DEVELOPMENT SERVER²

A. Based on the files received, the following steps are taken to incorporate the data text files, the release dates and the unit changes.

1. Data files that need to be removed are moved to a subdirectory named “test_ignore,” which is created within the same directory as the original data file. In this way, we retain the original data files for possible future reference.
2. New files are copied into the appropriate existing archive directory. If there are existing files of the same name, they are either renamed or moved to a “test_ignore” sub-directory – again, to retain the original files.
3. New archive directories are created as needed, with the same structure as outlined above – this is done mainly for pre-FRED vintages.
4. Existing files may need to be edited.
5. Existing files may need to be moved to different archive directories.
6. Release dates are imported to the database table “fred_release_date” using phpPgAdmin on the development server and phpPgAdmin on the production (research) server. PhpPgAdmin is a browser-based software package that allows access to the database.

The following image gives an example of a list of release dates ready to be imported into the database, formatted as a tabbed-delimited text file. The first line of text lists the fields in the fred_release_date table. Each release in the database is identified by a unique “fred_release_id.” “Official” designates whether or not the release date is known with certainty, and “policy” designates whether or not the release date applies to all series on the release.



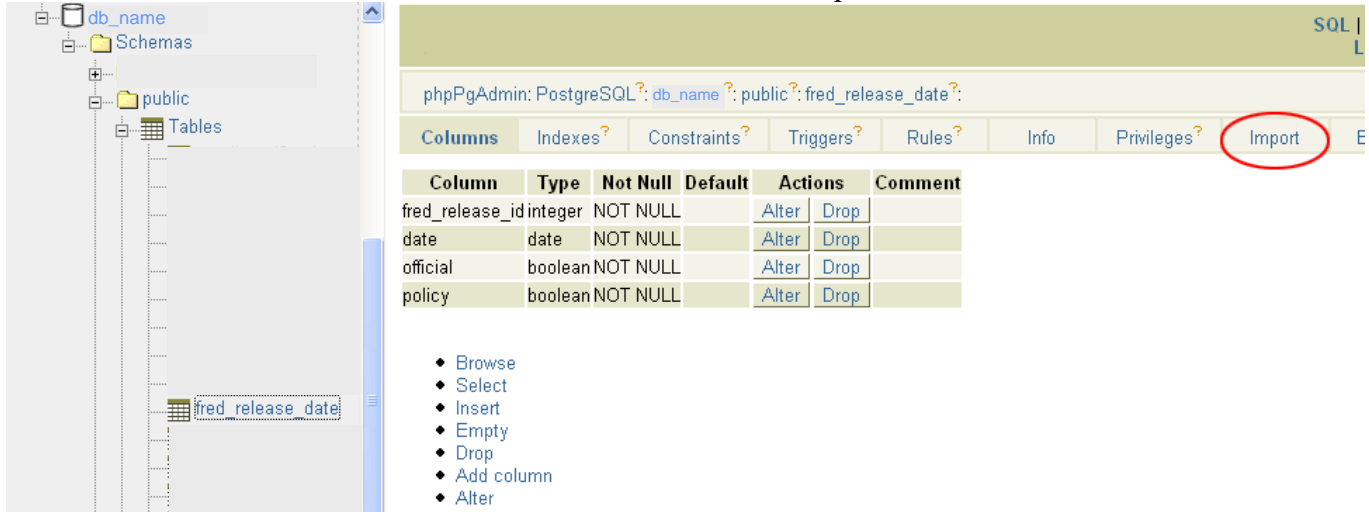
The image shows a Notepad window titled "release dates for ALFRED.txt". The text inside is a tabbed-delimited file with the following content:

fred_release_id	date	official	policy
53	8/28/1997	TRUE	TRUE
53	9/26/1997	TRUE	TRUE
53	10/31/1997	TRUE	TRUE
53	11/26/1997	TRUE	TRUE
53	12/23/1997	TRUE	TRUE
53	1/30/1998	TRUE	TRUE
53	2/27/1998	TRUE	TRUE
53	3/26/1998	TRUE	TRUE
53	4/30/1998	TRUE	TRUE
53	5/28/1998	TRUE	TRUE
53	6/25/1998	TRUE	TRUE
53	7/31/1998	TRUE	TRUE
53	8/27/1998	TRUE	TRUE
53	9/24/1998	TRUE	TRUE
53	10/30/1998	TRUE	TRUE

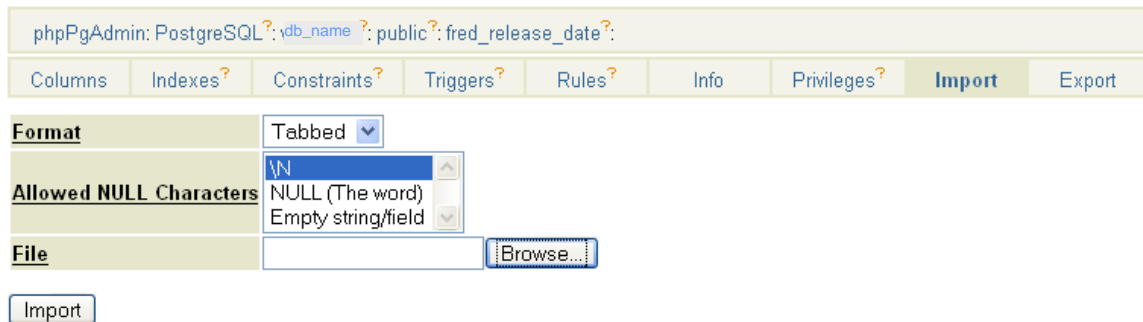
² The procedures in Steps 8 and 12 have been tested using the following software:

Red Hat Enterprise Linux 4
PostgreSQL 8.1.4
PHP 4.3.9
Apache 2.0.52

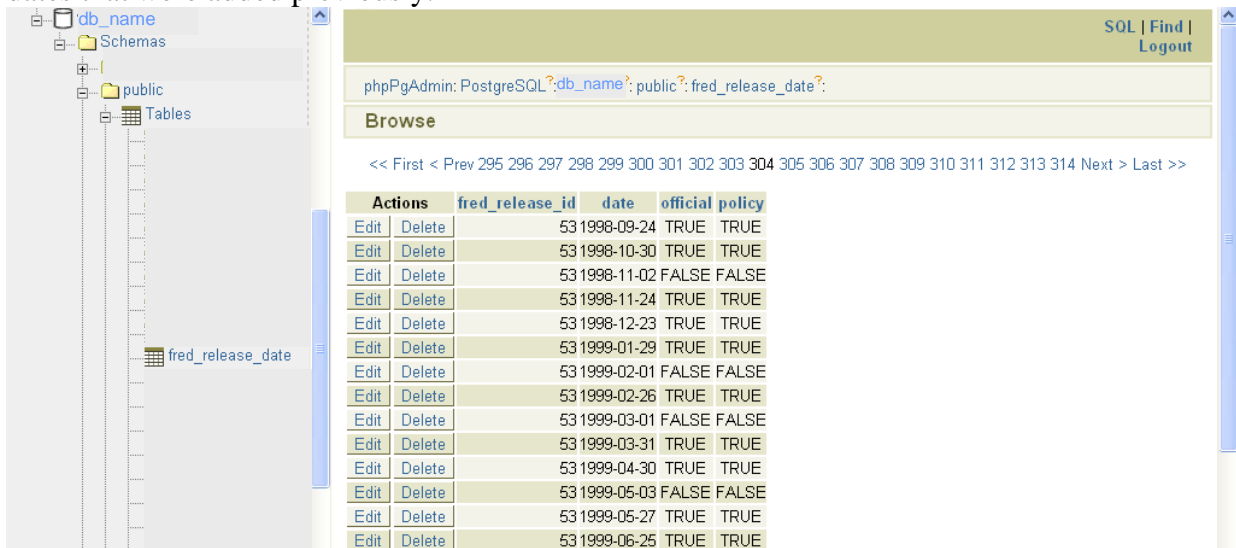
The page in phpPgAdmin that we will use to add release dates to the fred_release_date table is shown below. To add release dates to the table, click on the “Import” link.



This takes us to the following screen, from which we can choose the format of the file (we choose “Tabbed”). We locate the file to be imported by choosing “Browse.” Click “Import” to add the release dates to the table.



In the image below, some of the newly imported release dates can be seen, as well as release dates that were added previously.



Clicking on the “Edit” button next to a release date will allow you to edit it if necessary, as shown below.

phpPgAdmin: PostgreSQL?: db_name?: public?: fred_release_date?:

Edit row

Column	Type	Format	NULL (The word)	Value
fred_release_id	integer	Value		53
date	date	Value		1998-09-24
official	boolean	Value		TRUE
policy	boolean	Value		TRUE

Save Cancel

- Units changes for individual series, changes to the names of press releases, and other additional information are updated using the admin forms on the development server and the production server. (Please note that the forms shown below were developed by the St. Louis Fed.)

An example of units changes as shown in the admin forms:

Series ID	Valid Period	Title	Units
<u>FINSLC1</u>	1776-07-04 to 1959-02-18	Real Final Sales of Domestic Product, 1 Decimal	Bil. of 1957 \$
	1959-02-19 to 1965-08-18	Real Final Sales of Domestic Product, 1 Decimal	Bil. of 1954 \$
	1965-08-19 to 1976-01-19	Real Final Sales of Domestic Product, 1 Decimal	Bil. of 1958 \$
	1976-01-20 to 1985-12-19	Real Final Sales of Domestic Product, 1 Decimal	Bil. of 1972 \$
	1985-12-20 to 1991-12-03	Real Final Sales of Domestic Product, 1 Decimal	Bil. of 1982 \$
	1991-12-04 to 1996-01-18	Real Final Sales of Domestic Product, 1 Decimal	Bil. of 1987 \$
	1996-01-19 to 1999-10-27	Real Final Sales of Domestic Product, 1 Decimal	Bil. of Chained 1992 \$
	1999-10-28 to 2003-12-09	Real Final Sales of Domestic Product, 1 Decimal	Bil. of Chained 1996 \$
	2003-12-10 to 9999-12-31	Real Final Sales of Domestic Product, 1 Decimal	Bil. of Chained 2000 \$

Text highlighted in red indicates information that changed from a previous revision.

Edit Series ID: FINSLC1, Real Final Sales of Domestic Product, 1 Decimal

*Valid Period: 1776-07-04 to 1959-02-18
Use date format YYYY-MM-DD (e.g. 2004-12-31).

*Public: True False

*Vintages: True False

*Plot Growth Rates: True False
(Defaults to TRUE. Set this to FALSE if a series is an INTEREST RATE, EXCHANGE RATE, RATIO, or a series with NEGATIVE VALUES. The intuition is that growth rates should not be available for series that are stationary with stable means.)

*Title: Real Final Sales of Domestic Product 1 Decimal

*Units: Billions of 1957 Dollars

Other: _____

*Frequency: Quarterly

Other: _____

Frequency describes how observations recur (e.g. Annual, Quarterly, Monthly, Weekly, Bi-weekly, Daily). Do not enter here 'End of Period', 'Ending Friday', 'Ending Monday', 'Ending Wednesday', 'Fiscal Year', etc. Enter this type of information as the period description below.

B. Preparing and running *update_date_vintages.php* –

The program `~/update_data_vintages.php` reads in data from the archive directories, merges data revisions storing each revision only once, adjusts archive dates to release dates, and outputs database files, a log file, and other information. The following settings are edited prior to running the program:

1. Choose between running an entire FRED category or specific series. If running an entire category, the category ID number(s) must be listed in the program. If running specific series, the Series ID's must be listed in the program. In the example below, the empty array following “`only_ancestor_cat_ids`” indicates that the program will not be restricted to a specific category – and the “`only_series_ids`” array indicates that the program will run using the series “`FINSLC1`.”
2. Set “`only_vintages_public`” to “`true`” so that the program runs using release dates. The “`vintages_public`” field in the database table “`fred_series`” must be set to true for each series in the database.
3. Specify the range of archive directory dates (by year). In the example below, the program is set to use years 1996 through 2006 (other ranges are coded into the program for convenience, but are commented out in this example).
4. Specify the maximum archive date (yyyy-mm-dd) if needed. The maximum archive date is mainly used for pre-FRED vintages that will be merged with existing ALFRED vintages. In the example below, “`max_archive_date`” is set to false so that the program will use all available archives within the year range specified.
5. Choose whether or not to produce the `observation_by_vintage_date.sql` file. In the example below, the program will create this file.

The following two images show these settings in the `update_data_vintages.php` program. (Note that these images only represent part of the code for `update_date_vintages.php`.³)

```
//$only_ancestor_cat_ids = array(21, 110); // Restrict series to be in categories that have these ids as ancestors. Use an empty array to not restrict by category. # 1
//$only_ancestor_cat_ids = array(28); // Restrict series to be in categories that have these ids as ancestors. Use an empty array to not restrict by category.

$only_ancestor_cat_ids = array(); // Restrict series to be in categories that have these ids as ancestors. Use an empty array to not restrict by category.

$only_vintages_public = true; # 2
//$only_vintages_public = false; // Boolean (e.g. true or false) to restrict to series with $series->vintages_public equal to true.
// Boolean (e.g. true or false) to restrict to series with $series->vintages_public equal to true.

//$only_series_ids = array();
$only_series_ids = array('FINSLC1'); # 1
```

³ For complete `update_data_vintages.php` program and more detailed information regarding the environment set-up necessary, contact webmaster@research.stlouisfed.org.

```

$create_fred_observation_file = true; // Either true or false.
$create_fred_obs_meta_file = true; // Either true or false.
$create_fred_observation_by_vintage_date_file = true; // Either true or false.

// used in $series->is_file_old($file,$file_old_tolerance).
// $file_old_tolerance is added to the file modification time and compared to the last updated time for the series.
// $file_old_tolerance lets some old files te$datat as new if there newer than the series last update time minus $file_old_tolerance.
// $file_old_tolerance was added to be conservative and catch update that could occur between the time that
// a series update was detected and when it actually was stored in the database.
$file_old_tolerance = 0*60;
$max_dbh_age = 60; // Max database connection age in seconds. If older, old $dbh close and new $dbh created.

$sparse_preliminary_values = false;
//$max_archive_date = '1996-10-11'; // Set to last archive date to read in (YYYY-MM-DD) or set to false to disable;
$max_archive_date = false; // Set to last archive date to read in (YYYY-MM-DD) or set to false to disable;

$years = array(1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006);
//$years = array(1999, 2000, 2001, 2002, 2003, 2004, 2005);
//$years = array(1994, 1995, 1996);
/*
$years = array(
    1927, 1928, 1929,
    1930, 1931, 1932, 1933, 1934, 1935, 1936, 1937, 1938, 1939,
    1940, 1941, 1942, 1943, 1944, 1945, 1946, 1947, 1948, 1949,
    1950, 1951, 1952, 1953, 1954, 1955, 1956, 1957, 1958, 1959,
    1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969,
    1970, 1971, 1972, 1973, 1974, 1975, 1976, 1977, 1978, 1979,
    1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989,
    1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999,
    2000, 2001, 2002, 2003, 2004, 2005, 2006
);
*/

```

C. Results of the *update_data_vintages.php* Program –

When the *update_data_vintages.php* program completes, an email is sent that contains a link to the zipped output files- *readme.txt*, *fred_observation.sql*, *fred_obs_meta.sql*, and *fred_observation_by_vintage_date.sql* (optional). Below is an example email:



Updates available at:
http://.../2006-09-19_102905.zip
 Category IDs:
 Any
 Only Series With Public Vintages:
 true
 Series IDs:
 FINSLC1

The *README.txt* file is shown below. Public Vintages “true” indicates that the program was run using release dates for the series listed.



The file fred_update.log is shown below. “A” indicates the archive from which the vintage was created, “F” indicates the time stamp on the file, and “V” indicates the vintage date used to store this revision.

```

fred_update.log - Notepad
File Edit Format View Help

LOG START: 2006-09-19 10:29:05 AM

Updated FINSCL1 , 132 New Obs, A: 1980-01-18, F: 2006-08-31, V: 1980-01-18, A|F-V: 0 Days, Last Obs: 1979-10-01 - 1979-12-31 1435.2
Updated FINSCL1 , 1 New Obs, A: 1980-02-22, F: 2006-08-31, V: 1980-02-22, A|F-V: 0 Days, Last Obs: 1979-10-01 - 1979-12-31 1438.2
Updated FINSCL1 , 1 New Obs, A: 1980-03-19, F: 2006-08-31, V: 1980-03-19, A|F-V: 0 Days, Last Obs: 1979-10-01 - 1979-12-31 1439.0
Updated FINSCL1 , 1 New Obs, A: 1980-04-18, F: 2006-08-31, V: 1980-04-18, A|F-V: 0 Days, Last Obs: 1980-01-01 - 1980-03-31 1444.3
Updated FINSCL1 , 1 New Obs, A: 1980-05-20, F: 2006-08-31, V: 1980-05-20, A|F-V: 0 Days, Last Obs: 1980-01-01 - 1980-03-31 1444.5
Updated FINSCL1 , 1 New Obs, A: 1980-06-18, F: 2006-08-31, V: 1980-06-18, A|F-V: 0 Days, Last Obs: 1980-01-01 - 1980-03-31 1444.4
Updated FINSCL1 , 1 New Obs, A: 1980-07-18, F: 2006-08-31, V: 1980-07-18, A|F-V: 0 Days, Last Obs: 1980-04-01 - 1980-06-30 1408.5
Updated FINSCL1 , 1 New Obs, A: 1980-08-19, F: 2006-08-31, V: 1980-08-19, A|F-V: 0 Days, Last Obs: 1980-04-01 - 1980-06-30 1407.6
Updated FINSCL1 , 1 New Obs, A: 1980-09-19, F: 2006-08-31, V: 1980-09-19, A|F-V: 0 Days, Last Obs: 1980-04-01 - 1980-06-30 1406.0
Updated FINSCL1 , 1 New Obs, A: 1980-10-17, F: 2006-08-31, V: 1980-10-17, A|F-V: 0 Days, Last Obs: 1980-07-01 - 1980-09-30 1418.9
Updated FINSCL1 , 1 New Obs, A: 1980-11-19, F: 2006-08-31, V: 1980-11-19, A|F-V: 0 Days, Last Obs: 1980-07-01 - 1980-09-30 1417.8
Updated FINSCL1 , 135 New Obs, A: 1980-12-23, F: 2006-08-31, V: 1980-12-23, A|F-V: 0 Days, Last Obs: 1980-07-01 - 1980-09-30 1476.9
Updated FINSCL1 , 1 New Obs, A: 1981-01-21, F: 2006-08-31, V: 1981-01-21, A|F-V: 0 Days, Last Obs: 1980-10-01 - 1980-12-31 1490.3
Updated FINSCL1 , 1 New Obs, A: 1981-02-19, F: 2006-08-31, V: 1981-02-19, A|F-V: 0 Days, Last Obs: 1980-10-01 - 1980-12-31 1492.4
Updated FINSCL1 , 1 New Obs, A: 1981-03-18, F: 2006-08-31, V: 1981-03-18, A|F-V: 0 Days, Last Obs: 1980-10-01 - 1980-12-31 1492.7
Updated FINSCL1 , 1 New Obs, A: 1981-04-20, F: 2006-08-31, V: 1981-04-20, A|F-V: 0 Days, Last Obs: 1981-01-01 - 1981-03-31 1515.0
Updated FINSCL1 , 1 New Obs, A: 1981-05-19, F: 2006-08-31, V: 1981-05-19, A|F-V: 0 Days, Last Obs: 1981-01-01 - 1981-03-31 1518.3
Updated FINSCL1 , 1 New Obs, A: 1981-06-18, F: 2006-08-31, V: 1981-06-18, A|F-V: 0 Days, Last Obs: 1981-01-01 - 1981-03-31 1517.8

```

The file fred_observation.sql is a database dump of the fred_observation table for only the series and vintage dates processed by the update_data_vintages.php program. Below is an example:

```

fred_observation.sql - Notepad
File Edit Format View Help

--
-- PostgreSQL database dump
--

SET client_encoding = 'UNICODE';
SET check_function_bodies = false;
SET SESSION AUTHORIZATION 'xxxxxxxxxx';
SET search_path = "temp", pg_catalog;

--
-- Data for TOC entry 2 (OID 488324480)
-- Name: fred_observation; Type: TABLE DATA; Schema: temp; Owner: datadesk
--

COPY fred_observation (series_id, period_start_date, period_end_date, value, is_preliminary, valid_start_date, valid_end_date) FROM stdin;
FINSCL1 1947-01-01 1947-03-31 463.9 f 1980-01-18 1980-12-22
FINSCL1 1947-01-01 1947-03-31 463.9 f 1980-12-23 1985-12-15
FINSCL1 1947-01-01 1947-03-31 1056.9 f 1985-12-20 1991-12-03
FINSCL1 1947-01-01 1947-03-31 \N f 1991-12-04 1992-12-21
FINSCL1 1947-01-01 1947-03-31 1240.0 f 1992-12-22 1996-01-13
FINSCL1 1947-01-01 1947-03-31 \N f 1996-01-19 9999-12-31
FINSCL1 1947-04-01 1947-06-30 468.4 f 1980-01-18 1980-12-22
FINSCL1 1947-04-01 1947-06-30 470.4 f 1980-12-23 1985-12-19
FINSCL1 1947-04-01 1947-06-30 1066.0 f 1985-12-20 1991-12-03
FINSCL1 1947-04-01 1947-06-30 \N f 1991-12-04 1992-12-21
FINSCL1 1947-04-01 1947-06-30 1247.6 f 1992-12-22 1996-01-13
FINSCL1 1947-04-01 1947-06-30 \N f 1996-01-19 9999-12-31
FINSCL1 1947-07-01 1947-09-30 470.8 f 1980-01-18 1980-12-22
FINSCL1 1947-07-01 1947-09-30 473.0 f 1980-12-23 1985-12-19
FINSCL1 1947-07-01 1947-09-30 1074.3 f 1985-12-20 1991-12-03

```

The file fred_obs_meta.sql is a database dump of the fred_obs_meta table for only the series and vintage dates processed by the update_data_vintages.php program. The fred_obs_meta table stores summary statistics about observations such as the first observation period, last observation period, maximum number of digits, and maximum number of decimal places. This information is used to create and format public data files created on the production server. Below is an example of the file fred_obs_meta.sql file:

```

fred_obs_meta.sql - Notepad
File Edit Format View Help

--
-- PostgreSQL database dump
--

SET client_encoding = 'UNICODE';
SET check_function_bodies = false;
SET SESSION AUTHORIZATION 'xxxxxxxxxx';
SET search_path = "temp", pg_catalog;

--
-- Data for TOC entry 2 (OID 488325772)
-- Name: fred_obs_meta; Type: TABLE DATA; Schema: temp; Owner: datadesk
--

COPY fred_obs_meta
(series_id, last_updated, min_period_start_date, min_period_end_date, max_period_start_date, max_period_end_date,
value_width, value_decimal_width, valid_start_date, valid_end_date, files_timestamp, min_period_start_date_nn,
min_period_end_date_nn, max_period_start_date_nn, max_period_end_date_nn) FROM stdin;
FINSCL1 2006-09-19 10:29:06-05 1947-01-01 1947-03-31 1979-10-01 1979-12-31
FINSCL1 2006-09-19 10:29:07-05 1947-01-01 1947-03-31 1979-10-01 1979-12-31
FINSCL1 2006-09-19 10:29:07-05 1947-01-01 1947-03-31 1980-01-01 1980-03-31

1980-01-18 1980-02-21 2006-09-19 10:29:06-05 1947-01-01
1980-02-22 1980-04-17 2006-09-19 10:29:07-05 1947-01-01
1980-04-18 1980-05-19 2006-09-19 10:29:07-05 1947-01-01

```

The file fred_observation_by_vintage_date.sql is an optional file used for analysis only. This table does not exist on production. The fred_observation_by_vintage_date.sql contains all distinct vintages of data for the series and vintage dates processed by the program update_data_vintages.php. The file fred_observation_by_vintage_date.sql is an exploded or verbose representation of the vintage data contained in fred_observation.sql. The file fred_observation_by_vintage_date.sql contains the rows in fred_observation.sql repeated for each distinct vintage date that is between the row's valid start date and valid end date. Below is example output:

```

fred_observation_by_vintage_date.sql - Notepad
File Edit Format View Help
--
-- PostgreSQL database dump
--
SET client_encoding = 'UNICODE';
SET check_function_bodies = false;
SET SESSION AUTHORIZATION 'xxxxxxxxxx';
SET search_path = "temp", pg_catalog;
--
-- Data for TOC entry 2 (OID 488325906)
-- Name: fred_observation_by_vintage_date; Type: TABLE DATA; Schema: temp; Owner: datadesk
--
COPY fred_observation_by_vintage_date (series_id, period_start_date, period_end_date, value, is_preliminary, valid_start_date, valid_end_date,
|vintage_date) FROM stdin;
FINSCLI 1947-01-01 1947-03-31 463.9 f 1980-01-18 1980-12-22 1980-01-18
FINSCLI 1947-04-01 1947-06-30 468.4 f 1980-01-18 1980-12-22 1980-01-18
FINSCLI 1947-07-01 1947-09-30 470.8 f 1980-01-18 1980-12-22 1980-01-18
FINSCLI 1947-10-01 1947-12-31 471.0 f 1980-01-18 1980-12-22 1980-01-18
FINSCLI 1948-01-01 1948-03-31 473.2 f 1980-01-18 1980-12-22 1980-01-18
FINSCLI 1948-04-01 1948-06-30 480.4 f 1980-01-18 1980-12-22 1980-01-18
FINSCLI 1948-07-01 1948-09-30 483.9 f 1980-01-18 1980-12-22 1980-01-18
FINSCLI 1948-10-01 1948-12-31 490.6 f 1980-01-18 1980-12-22 1980-01-18
FINSCLI 1949-01-01 1949-03-31 491.3 f 1980-01-18 1980-12-22 1980-01-18
FINSCLI 1949-04-01 1949-06-30 496.1 f 1980-01-18 1980-12-22 1980-01-18
FINSCLI 1949-07-01 1949-09-30 495.9 f 1980-01-18 1980-12-22 1980-01-18
FINSCLI 1949-10-01 1949-12-31 496.9 f 1980-01-18 1980-12-22 1980-01-18
FINSCLI 1950-01-01 1950-03-31 507.0 f 1980-01-18 1980-12-22 1980-01-18
FINSCLI 1950-04-01 1950-06-30 517.4 f 1980-01-18 1980-12-22 1980-01-18

```

STEP 9: WORK WITH WEB DATABASE OUTPUT

Three files of the files shown above will be used to verify the data before it is made live to the public: fred_update.log, fred_observation.sql, and fred_observation_by_vintage_date.sql.

- A. Check fred_update.log for errors and fix them. These corrections can be made to the text files or to the spreadsheet. If the changes are made to the spreadsheet, you will need to rerun the SAS program from Step 6 in order to create the text files required for the web group programs. Do not move on until the error log comes back clean.

Errors and Explanations:

- Skipping *\$file*. Error parsing file.
This usually indicates that the date is in an unexpected format or the header of the text file doesn't begin with the word "DATE".
- Skipping *\$file*. Error. All observations have missing values.
- Skipping *\$file*. Error. The vintage date is not newer than the previous vintage.
This indicates that a file contains revisions but there is not a release date available to assign these changes to. This usually occurs if a release date is missing in the list.⁴
- Skipping *\$file*. Error. The last observation in the previous vintage is newer than the last observation in the file with archive date *archive_date* and vintage date *valid_start_date*.
This indicates that the previous file contained more observations than the current file. Check to make sure that the file shown in the error ends with the observation that is shown in the press release.
- Error storing *\$file* in database. ERROR: new row for relation "fred_observation" violates check constraint "fred_obs_ped_lteq_vsd"
This indicates that the last observation in the file has a date greater than the release date that it is being assigned to. For example, you shouldn't have a September 2006 CPI observation on the 2006-09-15 release date.
- Skipping *\$file*. Error. Unequal # of obs between the previous and file vintages from *period_start_date1* to *period_start_date2*. (*\$n_db_obs_middle* vs. *\$n_file_obs_middle*)
Check to make sure that there are no new missing rows in the middle of your text files. Also check to make sure that no single date has more than 1 value.

- B. Prepare the *.sql files for use in the SAS code below.

1. Replace all of the "\N" with a "." for missing values and
2. Remove the final line ("\.") of the file.

- C. The following SAS program (CheckDatabaseFiles.sas) will prepare several files and pieces of output that you can use to verify your work. In each section of code, you will need to adjust the directory names for the location of the files that you received from the web group and for the location of the output files that this code will produce.⁵

⁴ This error will also occur if there were changes released by the source between releases which were recorded in an electronic database (Haver), and this data was used to create some of the vintage text files. See appendix for more information.

⁵ The SAS programs in this document have been tested using version 9.1.3 Service Pack 2 (for the XP_PRO platform).

1. This section verifies that the **number of decimal places** being stored is consistent. A value of -1 denotes a missing value, 0 = no decimal places, 1 = 1 decimal place, etc.

Program:

```
options nocenter pagesize=32767 linesize=256 nonumber nodate;

PROC IMPORT OUT= fromweb
            DATAFILE= "h:\alfred\process doc\2006-09-19_102905\fred_observation.sql"
            DBMS=TAB REPLACE;
GETNAMES=NO;
DATAROW=18;
GUESSINGROWS=10000;
RUN;

data res;
  set fromweb;
  rename var1=varname
         var2=period_start_date
         var3=period_end_date
         var4=value
         var5=junk
         var6=valid_start_date
         var7=valid_end_date;
run;

data res; set res;
  deonly=value-int(value);
  if deonly ne 0 then numdec=length(compress(abs(deonly)))-2;
  else if deonly = 0 then numdec=0;

  psd = put(period_start_date, yymmdd10.);
  vsd = put(valid_start_date, yymmdd10.);
run;

proc sort data=res; by varname; run;
proc freq data=res noprint;
  tables numdec*varname/list sparse out=res_numdec;
run;

proc sort data=res_numdec; by numdec; run;
proc transpose data=res_numdec out=res_numdect;
  var count;
  id varname;
  idlabel varname;
  by numdec;
run;

proc print data=res_numdect noobs;
  title "DecimalPlaces - By Series";
run;
```

Output (from .LST file):

```
DecimalPlaces - By Series

numdec   _NAME_   _LABEL_   FINS1C1
      -1   COUNT   Frequency Count      98
       0   COUNT   Frequency Count     140
       1   COUNT   Frequency Count    1049
```

2. This section will **count the number of revisions** that took place for each series on each release date. Then compare these results to the output in sheet “5Comparing to Web-Loaded Files” of your original data spreadsheet to ensure that the data was transferred correctly.

Program:

```
*****;
**FINDING REVISION COUNTS**;
*****;
proc sort data=res; by varname; run;
proc freq data=res noprint;
```

```

tables valid_start_date*varname/list sparse out=res_varfreq;
run;

proc sort data=res_varfreq; by valid_start_date; run;
proc transpose data=res_varfreq out=res_varfreqt;
var count;
id varname;
idlabel varname;
by valid_start_date;
run;

data res_varfreqt; set res_varfreqt(drop=_name_ _label_); run;
proc print data=res_varfreqt noobs;
format valid_start_date yymmdd10.;
title "RevisionCount - By Series";
run;

```

Output (from .LST file):
RevisionCount - By Series

valid_start_date	FINSLC1
1980-01-18	132
1980-02-22	1
1980-03-19	1
1980-04-18	1
1980-05-20	1
1980-06-18	1
1980-07-18	1
1980-08-19	1
1980-09-19	1
1980-10-17	1
1980-11-19	1
1980-12-23	135
1981-01-21	1
1981-02-19	1
1981-03-18	1
1981-04-20	1
1981-05-19	1

Example of Comparison Spreadsheet:

	A	B	C	D	E	F	G	H	I
	Web Run 2006-09-19-102905			Original Spreadsheet Counts			Differences		
	Start_date	FINSLC1		Release Date	FINSLC1		FINSLC1		
4	01/18/80	132		01/18/80			-132		
5	02/22/80	1		02/22/80	1		0		
6	03/19/80	1		03/19/80	1		0		
7	04/18/80	1		04/18/80	1		0		
8	05/20/80	1		05/20/80	1		0		
9	06/18/80	1		06/18/80	1		0		
10	07/18/80	1		07/18/80	1		0		
11	08/19/80	1		08/19/80	1		0		
12	09/19/80	1		09/19/80	1		0		
13	10/17/80	1		10/17/80	1		0		
14	11/19/80	1		11/19/80	1		0		
15	12/23/80	135		12/23/80	135		0		
16	01/21/81	1		01/21/81	1		0		
17	02/19/81	1		02/19/81	1		0		
18	03/18/81	1		03/18/81	1		0		
19	04/20/81	1		04/20/81	1		0		
20	05/19/81	1		05/19/81	1		0		
21	06/18/81	1		06/18/81	1		0		
22	07/22/81	1		07/22/81	1		0		
23	08/19/81	1		08/19/81	1		0		
24	09/18/81	1		09/18/81	1		0		
25	10/20/81	1		10/20/81	1		0		
26	11/19/81	1		11/19/81	1		0		
27	12/18/81	1		12/18/81	1		0		
28	01/20/82	1		01/20/82	1		0		
29	02/22/82	1		02/22/82	1		0		
30	03/19/82	1		03/19/82	1		0		
31	04/21/82	1		04/21/82	1		0		
32	05/19/82	1		05/19/82	1		0		
33	06/21/82	1		06/21/82	1		0		
34	07/21/82	38		07/21/82	38		0		
35	08/19/82	1		08/19/82	1		0		

3. This section will place the **data into a spreadsheet**. You can use this spreadsheet to look at the revision history for a specific variable, specific observation date, or specific release date. To simplify this, use the Autofilter feature (see below) in Excel.

Program:

```
*****;
**CREATE ONE COLUMN DATA SPREADSHEET**;
*****;
proc export data=res(drop=period_start_date valid_start_date deonly numdec junk)
  dbms=excel2000
  outfile='H:\Alfred\Process Doc\Output\fred_observation.xls';
run;
```

Output (in output/fred_observation.xls):

The file that is output by SAS has been adjusted below for readability and usability.

1. The columns were rearranged to include only varname, psd, period_end_date, value, vsd, and valid_end_date (in that order).
2. Next, select cell A2. Then select “Window-Freeze Panes” from the Excel pull-down menus.
3. Finally, select cell A2. Then select “Data-Filter-AutoFilter” from the Excel pull-down menus.

	A	B	C	D	E	F	G	H
	varname	psd	period_end_date	value	vsd	valid_end_date		
1								
2	FINSLOC1	1947-01-01	3/31/1947	463.9	(All)	12/22/1980		
3	FINSLOC1	1947-01-01	3/31/1947	465.9	(Top 10...)	12/19/1985		
4	FINSLOC1	1947-01-01	3/31/1947	1056.9	(Custom...)	12/3/1991		
5	FINSLOC1	1947-01-01	3/31/1947		1980-01-18	12/21/1992		
6	FINSLOC1	1947-01-01	3/31/1947	1240	1980-02-22	1/18/1996		
7	FINSLOC1	1947-01-01	3/31/1947		1980-03-19	12/31/9999		
8	FINSLOC1	1947-04-01	6/30/1947	468.4	1980-04-18	12/22/1980		
9	FINSLOC1	1947-04-01	6/30/1947	470.4	1980-05-20	12/19/1985		
10	FINSLOC1	1947-04-01	6/30/1947	1066	1980-06-18	12/3/1991		
11	FINSLOC1	1947-04-01	6/30/1947		1980-07-18	12/21/1992		
12	FINSLOC1	1947-04-01	6/30/1947	1247.6	1980-08-19	1/18/1996		
13	FINSLOC1	1947-04-01	6/30/1947		1980-09-19	12/31/9999		
14	FINSLOC1	1947-07-01	9/30/1947	470.8	1980-10-17	12/22/1980		
15	FINSLOC1	1947-07-01	9/30/1947	473	1980-11-19	12/19/1985		
16	FINSLOC1	1947-07-01	9/30/1947	1074.3	1980-12-23	12/3/1991		
17	FINSLOC1	1947-07-01	9/30/1947		1981-01-21	12/21/1992		
					1981-02-19			
					1981-03-18			
					1981-04-20			
					1981-05-19			
					1991-12-04			

4. This section will **plot the levels** of each variable. The results should match the plots produced in your data spreadsheet in Step 3. Also plot the current data available from the source. The patterns should be similar to the charts produced in SAS.

Before running this section, adjust the start and end dates in the GPLOT. These dates should include the earliest and most recent observation available over all of the vintages.

Program:

```
*****;
**PLOTTING ALL LEVELS**;
*****;
proc sort data=res; by varname; run;

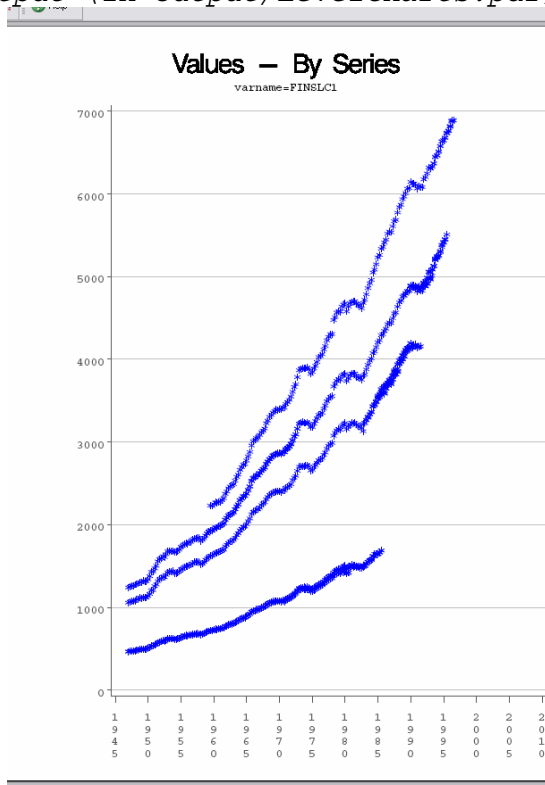
ods pdf file="H:\Alfred\Process Doc\Output\LevelCharts.pdf";
symbol1 l=1 w=1 c=blue i=none v=star;
proc gplot gout=plot data=res;
  axis1 order=("01jan1945"d to "31dec2010"d by year5)
  label=none
```

```

minor=none
color=dagr;
axis2 label=none
minor=none
color=dagr;
plot value*period_start_date=1/
overlay
haxis=axis1
vaxis=axis2
autovref
cvref=ligr
noframe;
by varname;
format period_start_date year4.;
title "Values - By Series";
run; quit;
ods pdf close;

```

Output (in output/LevelCharts.pdf):



5. This section will create a spreadsheet that contains the last observation that was revised for each variable for each release date. At this step, verify that the revision that is shown in the spreadsheet **matches the actual printed press release / source data**. This step is most important for any data pulled from an electronic archive as the archive date for these will most likely not occur on the same day as the release date.

Program:

```

*****;
**MANUAL CHECK OF LAST REVISION AGAINST ASSIGNED PRESS RELEASE**
*****;
proc sort data=res; by varname valid_start_date descending period_start_date;
run;

data revlist(drop=lvarname lvalid_start_date period_start_date);
set res;

```

```

label valid_start_date="Release Date";

lvarname=lag(varname);
lvalid_start_date=lag(valid_start_date);

tag=1;
if valid_start_date=lvalid_start_date and varname=lvarname then do;
    value=.; tag=0;
end;

tocheck=put(period_start_date, date7.)||" - "||compress(value);
run;

data revlist; set revlist(where=(tag ne 0)); run;

proc sort data=revlist; by valid_start_date; run;

proc transpose data=revlist out=revlistt;
var tocheck;
id varname;
idlabel varname;
by valid_start_date;
run;

proc export data=revlistt(drop=_name_)
dbms=excel2000
outfile='H:\Alfred\Process Doc\Output\ManualCheck.xls';
sheet="Last Revision - by R.D.";
run;

```

Output (in output/ManualCheck.xls):

	A	B	C	D	E	F	G	H
1	valid_start_date	FINSLC1						
2	1/18/1980	01OCT79 - 1435.2						
3	2/22/1980	01OCT79 - 1438.2						
4	3/19/1980	01OCT79 - 1439						
5	4/18/1980	01JAN80 - 1444.3						
6	5/20/1980	01JAN80 - 1444.5						
7	6/18/1980	01JAN80 - 1444.4						
8	7/18/1980	01APR80 - 1408.5						
9	8/19/1980	01APR80 - 1407.6						
10	9/19/1980	01APR80 - 1406						
11	10/17/1980	01JUL80 - 1418.9						
12	11/19/1980	01JUL80 - 1417.8						
13	12/23/1980	01JUL80 - 1476.9						
14	1/21/1981	01OCT80 - 1490.3						
15	2/19/1981	01OCT80 - 1492.4						
16	3/18/1981	01OCT80 - 1492.7						
17	4/20/1981	01JAN81 - 1515						

6. This section will **sum all of the observations** that took place for each series on each release date. Then compare these results to the output from Step 5D above to ensure that the data was transferred correctly.

Program:

```

*****;
**SUM OF ALL OBSERVATIONS**;
*****;
PROC IMPORT OUT= fromweb2
DATAFILE=
"H:\Alfred\Process Doc\2006-09-19_102905\fred_observation_by_vintage_date.sql"
DBMS=TAB REPLACE;
GETNAMES=NO;
DATAROW=18;
GUESSINGROWS=20;
RUN;

data category2;
set fromweb2;
rename var1=varname
var2=period_start_date

```

```

var3=period_end_date
var4=value
var5=junk
var6=valid_start_date
var7=valid_end_date
var8=vintage_date;

run;

proc means data=category2 sum noprint;
  var value;
  class varname vintage_date;
  output out=varsums sum=;
run;

proc sort data=varsums; by vintage_date; run;
proc transpose data=varsums(where=( _type_=3)) out=varsumst;
  var value;
  by vintage_date;
  id varname;
  idlabel varname;
run;

proc print data=varsumst noobs;
  title "Sum of All Obs - By Series";
run;

```

Output (from .LST file):
Sum of All Obs - By Series

vintage_date	_NAME_	FINSLC1
1980-01-18	value	116287.3
1980-02-22	value	116290.3
1980-03-19	value	116291.1
1980-04-18	value	117735.4
1980-05-20	value	117735.6
1980-06-18	value	117735.5
1980-07-18	value	119144
1980-08-19	value	119143.1
1980-09-19	value	119141.5
1980-10-17	value	120560.4
1980-11-19	value	120559.3
1980-12-23	value	121998.6
1981-01-21	value	123488.9
1981-02-19	value	123491

Example of Comparison Spreadsheet:

	A	B	C	D	E	F	G	H
1	Web Run 2006-09-19-102905			Original Spreadsheet Sums			Differences	
2	vintage_date	FINSLC1		R.D.	FINSLC1		R.D.	FINSLC1
3	1/18/1980	116287.3		1/18/1980	116287.3		1/18/1980	0
4	2/22/1980	116290.3		2/22/1980	116290.3		2/22/1980	0
5	3/19/1980	116291.1		3/19/1980	116291.1		3/19/1980	0
6	4/18/1980	117735.4		4/18/1980	117735.4		4/18/1980	0
7	5/20/1980	117735.6		5/20/1980	117735.6		5/20/1980	0
8	6/18/1980	117735.5		6/18/1980	117735.5		6/18/1980	0
9	7/18/1980	119144		7/18/1980	119144		7/18/1980	0
10	8/19/1980	119143.1		8/19/1980	119143.1		8/19/1980	0
11	9/19/1980	119141.5		9/19/1980	119141.5		9/19/1980	0
12	10/17/1980	120560.4		10/17/1980	120560.4		10/17/1980	0
13	11/19/1980	120559.3		11/19/1980	120559.3		11/19/1980	0
14	12/23/1980	121998.6		12/23/1980	121998.6		12/23/1980	0
15	1/21/1981	123488.9		1/21/1981	123488.9		1/21/1981	0
16	2/19/1981	123491		2/19/1981	123491		2/19/1981	0
17	3/18/1981	123491.3		3/18/1981	123491.3		3/18/1981	0
18	4/20/1981	125006.3		4/20/1981	125006.3		4/20/1981	0
19	5/19/1981	125009.6		5/19/1981	125009.6		5/19/1981	0
20	6/18/1981	125009.1		6/18/1981	125009.1		6/18/1981	0
21	7/22/1981	126508.5		7/22/1981	126508.5		7/22/1981	0
22	8/19/1981	126506.8		8/19/1981	126506.8		8/19/1981	0
23	9/18/1981	126508.7		9/18/1981	126508.7		9/18/1981	0

7. This section creates a **chart of the growth rates** of the complete series as it appears on each release date. You can also create a separate chart for each release date by changing the by statement in the proc gplot. Again create a plot of the current growth rates and compare them with these SAS charts.

Before running this section, adjust the growth rate calculation, the start and end dates in the GPLOT, and the title of the chart.

Program:

```
*****;
**GROWTH RATE CHARTS**;
*****;
data growth;
  set category2;

  llvintage_date =lag1(vintage_date);
  llvalue =lag1(value);
  /*Change this calculation to calculate the growth rate that you are interested in
  checking for this set of variables*/
  gro=((value/llvalue)**4)-1)*100;

  if vintage_date ne llvintage_date then gro=.;
run;

/* To print outlying observations, adjust this proc for the variable name in
question, subsetting for the extreme where the odd observation is showing up*/
/*proc print data=growth(where=(varname in("FINSLC1") and gro>200)); run;*/

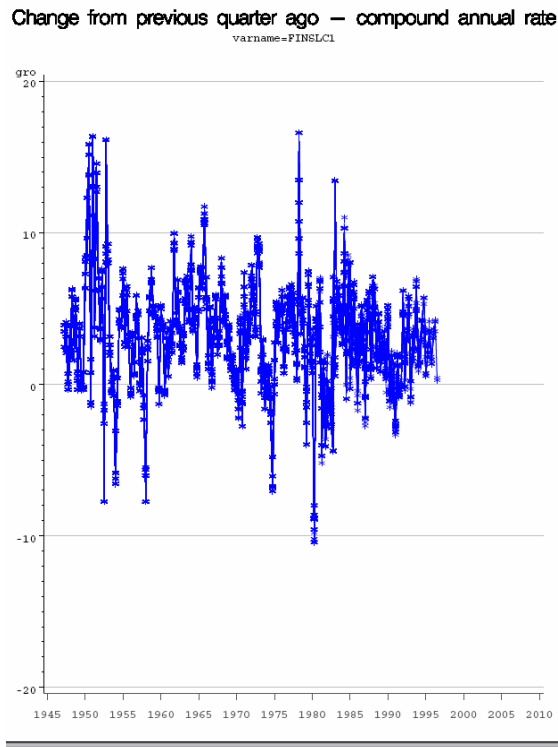
proc sort data=growth; by varname vintage_date ; run;

ods pdf file="H:\Alfred\Process Doc\Output\GrowthCharts.pdf";

symbol1 l=1 w=1 c=blue i=join v=star;
proc gplot gout=plot data=growth;
  axis1 order=("01jan1945"d to "31dec2010"d by year5)
    label=none
    minor=none
    color=dagr;
  plot gro*period_start_date=1/
    overlay
    haxis=axis1
    autovref
    cvref=ligr
    skipmiss
    noframe;
  by varname; *vintage_date; /*Uncomment for separate chart for each release date*/
  format period_start_date year4. vintage_date yymmdd10.;
  /*Adjust the title statement to match the growth calculation above*/
  title "Change from previous quarter - compound annual rate";
run;
quit;

ods pdf close;
```

Output (in output/GrowthCharts.pdf) :



STEP 10: MAKE ADJUSTMENTS/CORRECTIONS

Correct any errors that show up as a result of the checks in Step 9. These corrections can be made to the text files or to the spreadsheet. If the changes are made to the spreadsheet, you will need to rerun the SAS program from Step 6 in order to create the text files required for the web group programs.

Zip together the files that have been adjusted and email the file to the web group.

STEP 11: REPEAT STEPS 8-10 UNTIL NO CORRECTIONS ARE REQUIRED

STEP 12: WEB GROUP MAKES LIVE ON PUBLIC WEBSITE

Once work is done on the development server, the files `fred_observation.sql` and `fred_obs_meta.sql` need to be loaded onto the production server. On the production server, this requires deleting the existing series observations and observation meta information on the production and loading the contents of `fred_observation.sql` and `fred_obs_meta.sql`.

Below is a sample of `upgrade.sql` script.⁶

```
begin;

delete from fred_obs_meta where series_id in (
'FINSLC1'
);
\i /2006/20060927/pg/fred_obs_meta.sql
set search_path = "public", fred_file, tsearch, cassidi, geom ;

delete from fred_observation where series_id in (
'FINSLC1'
);
\i /2006/20060927/pg/fred_observation.sql
set search_path = "public", fred_file, tsearch, cassidi, geom ;

update fred_series set is_public = true, vintages_public = true where series_id in (
'FINSLC1'
) and exists (
select min(fo.valid_start_date) from fred_observation as fo
where fred_series.series_id = fo.series_id
group by fo.series_id
having fred_series.valid_end_date >= min(fo.valid_start_date));

--commit;
```

Before `upgrade.sql` is run:

1. Perform a final check to make sure everything is in place. On the production server, check that release dates have been loaded, check that units changes have been entered, and that the files from the development server contain the latest revisions currently available on the production server.
2. On the production server, create a new directory and copy `upgrade.sql`, `fred_observation.sql`, and `fred_obs_meta.sql` to this new directory.
3. Edit `upgrade.sql` to change series ids and the paths to files.
4. Edit `fred_observation.sql` and `fred_obs_meta.sql` and replace “temp” with “public”. This will load the files into the public schema in the database instead of the temp schema which was used on the development server.
5. At a command prompt, type ‘psql web_site’ to get a database command prompt.
6. Now type ‘\i ???/upgrade.sql’ (??? is the path to the file). This will start a transaction to delete the existing data, load the new data, and set the `vintages_public` boolean in the `fred_series` table to make vintage data available in ALFRED.

At this point, the transaction has not been committed and is not visible to the public. You can perform queries to check that everything is correct. If you find errors, type ‘rollback;’. Otherwise, type ‘commit;’.

⁶ For complete `upgrade.sql` program and more detailed information regarding the environment set-up necessary, contact webmaster@research.stlouisfed.org.

APPENDIX – CREATE TEXT FILES FOR WEB GROUP FROM HAVER ARCHIVES

If you are lucky enough to have copies of vintage Haver databases, the program below will pull the data from those databases and write text files that can be used by the web group's programs.

Some values in the program may need to be changed to coincide with your data (values to change are also highlighted in green in the complete program code below).

2. %let dbname=**usna**;
Change *usna* to match the Haver database that you are pulling the data from.
3. %let inv=**fsh**;
Change *fsh* to match the Haver mnemonic for the series that you are pulling.
4. %let outv=**finslcl1**;
Change *finslcl1* to match the vintage mnemonic that you are creating.
5. %let freq=**quarter**;
Change *quarter* to match the frequency of the data that you are pulling.
6. %let form=**yyqp6**;
Change *yyqp6* to match the frequency of the data that you are working with. This is the date format for quarterly data.
For monthly data, change *yyqp6* to *yymmp7*.
For weekly and daily data, change *yyqp6* to *yymmddn8*.
For calendar year data, change *yyqp6* to *year4*.
7. file="**H:\Alfred\Proces~1\webdir\&y\&y.-&m.-&d\data\gdp\&var**";
command="**md H:\Alfred\Proces~1\webdir\&y\&y.-&m.-&d\data\gdp**";
Change *H:\Alfred\Process Doc\webdir* to equal the name of the directory where you want to store the raw text files on your PC.
8. Change the macro call (*%haverpull2(...)*) at the bottom of the program to access the Haver archives that you are interested in.

Program:

```
options nocenter nodate nonumber pagesize=32767 linesize=256 noxwait;
proc datasets; delete _all_; run; quit;

*****;
**Haver Database to Pull From**;
*****;
%let dbname=usna;

*****;
**Haver Mnemonic**;
*****;
%let inv=fsh;

*****;
**FRED Mnemonic**;
*****;
%let outv=finslcl1;

*****;
**Frequency of data**;
*****;
%let freq=quarter;

*****;
**Date Format for Output**;
*****;
%let form=yyqp6;

*****;
**Macro to Pull From Haver Archives **;
** and Output to Text Files **;
*****;
%macro haverpull2(dir1=,start=);
```



```

%let y = %substr(&dir1,1,4);
%let m = %substr(&dir1,5,2);
%let d = %substr(&dir1,7,2);

libname _all_ clear;

**Pull data from Haver database**;
libname indata sasehavr "j:\&dir1\haver\dlx\data" freq=&freq start=&start;
data haver1;
  set indata.&dbname;
  keep date &inv;
  if &inv=. then delete;
  rename &inv=&outv;
run;

**Create directory to store text files by archive date**;
data _null_;
  command="md H:\Alfred\Proces~1\webdir\&y.\&y.-&m.-&d.\data\gdp\";
  call system(command);
run;

**Write text files**;
proc printto new file="H:\Alfred\Proces~1\webdir\&y.\&y.-&m.-&d.\data\gdp\&outv";
run;
proc print data=haver1 noobs;
  var date &outv;
  format date &form..;
  title " ";
run;
proc printto;
run;

**Combine all data in a SAS dataset**;
data comps;
  set haver1;
  archive="&dir1";
run;
proc append base=completeset data=comps; run; quit;

%mend;

*****;
**Call Macros to Pull Data from Archives**;
*****;
**more archives**;

*****;
**Count revisions in Haver archives**;
*****;
proc sort data=completeset; by date archive; run;

data counthelp(drop=l&outv ldate);
  set completeset;

  l&outv=lag(&outv);
  ldate=lag(date);

  if date=ldate and &outv=l&outv then delete;

  if archive="19990820" then delete;
run;

proc freq data=counthelp noprint;
  tables archive/list sparse out=haverrevcount;
run;

proc print data=haverrevcount noobs label;
  var archive count;
  label archive="Archive Date"
         count="Revision Count &outv";
run;

```

Output (from .LST file):

Archive Date	Revision Count finslc1
19990827	1
19991001	1
19991029	1
19991105	163
19991126	1
19991224	1
20000128	1
20000225	1
20000331	6
20000407	158
20000428	49
20000526	1

Remember, the dates listed in the output above are archive dates (not yet release dates). Compare the above list to the list of release dates that you were expecting between 1999-08-20 to the present.

For any dates above that will not map to a release date from your list (examples of problematic dates are highlighted in orange above), investigate to find an explanation. For data pulled from Haver, open the archive database in Haver's view mode, look at the data and the Haver update date and try to clean up these odd revisions. In some cases, you may need to add an "unofficial" release date to your list. This date usually will be set to the Haver update date and should only be included after you find documentation from the source that interrelease changes were made.

The revision counts calculated here may differ slightly from the final Alfred web output files as these don't count the change to (or from) a missing value. You can still use these counts as a pattern guide for what to expect when these data are loaded into the web database.

Remember that the rows in the spreadsheet that contain revision counts from the Haver archives are dated using the archive date while the results off the program in Step 9.C.2 will have converted these to release dates.