*cbd* Package Documentation

Created: David Kelley, 2014-2015

Updated:

[Introduction 1](#_Toc429740719)

[Data Retrieval 2](#_Toc429740720)

[CHIDATA 3](#_Toc429740721)

[Data Manipulation 3](#_Toc429740722)

[Transformations 3](#_Toc429740723)

[Summarizations 4](#_Toc429740724)

[Aggregation / Disaggregation 4](#_Toc429740725)

[Binary Operations of Series 4](#_Toc429740726)

[Merging tables 5](#_Toc429740727)

[File Handling & Misc. 5](#_Toc429740728)

[xlsfile 5](#_Toc429740729)

[Calendar Date Functions 5](#_Toc429740730)

[Trim 5](#_Toc429740731)

# Introduction

**cbd** (an abbreviation for Chartbook Data) is a Matlab [package](http://www.mathworks.com/help/matlab/matlab_oop/scoping-classes-with-packages.html) designed for general data management in Economic Research. Conceptually it breaks down into three major sections which make up the majority of this manual:

1. Data retrieval
2. Data manipulation
3. File handling & other

For individual function reference, see the Matlab documentation for the specific function (i.e. from Matlab, call **help cbd.data**).

The package is currently located at *O:\PROJ\_LIB\Presentations\Chartbook\Data\Dataset Creation\cbd*. This is the path that should be added to any project using cbd. All source files are contained in the *+cbd* directory and can be accessed with **cbd.*function*** once the top-level folder is on the Matlab path.

Within the **+cbd** folder, there is a **+private** folder that contains functions that are used internally. Feel free to use them; they have simply been placed there since they have no apparent user-facing purpose.

A suite of test routines are held in the *test* folder and can be run with the **execTests.m** script to ensure that the package is performing well against a reasonable set of test cases.

# Data Retrieval

Data retrieval in cbd is accomplished with the **cbd.data** function which currently supports three data sources: Haver, FRED, and the data saved in the CHIDATA folder (see later section).

|  |  |
| --- | --- |
| **Examples** | |
| **Code** | **Description** |
| gdp = cbd.data(‘GDPH’) gdp = cbd.data(‘GDPH@USECON’) gdp = cbd.data(‘GDPH’, ‘dbID’, ‘USECON’) | Level of real GDP from Haver |
| cfnai = cbd.data(‘FRBCNAI@SURVEYS’) | CFNAI from Haver |
| gdp2 = cbd.data(‘GDPC96@FRED’) | Level of real GDP from FRED |
| gdp3 = cbd.data(‘DIFA%(GDPH)’) | Annualized percent change in GDP |
| ptRate = cbd.data(‘LEPTE/LF\*100’) | Part-time rate |
| data = cbd.data({‘LR’, ‘YRYR%(JCXFEBM)’}) | Dataset of unemployment and inflation |
| lrData = cbd.data(‘LR’, ‘startDate’, ‘1/1/2000’) | Unemployment starting in January 2000 |
| lrDataRT = cbd.data(‘LR’, ‘asOfStart’, ‘1/1/2000’) | Real-time PCE with vintages beginning in 2000. |
| (UNRATE - UNRATE#asOf:"1/1/2010")#dbID:"FRED" | Revsions to the unemployment rate since the January 2010 values |

Each data series to be retrieved should be specified as a string. Each string should consist of a series name (for example, *GDPH*) that will be pulled from the database specified using the *@* symbol (*FRBCNAI@SURVEYS*). If no database is specified, it will default to the USECON Haver database.

Each string can also contain an expression of data manipulation functions (*DIFA%(GDPH)*) as specified in the next section. Data transformations can be nested as well as take other numeric arguments (*DIFA%(MOVV(JCBM,3),3)*). They can take string arguments provided that the string is specified in doublequotes. They can also use operators between two series (HPT + HST – HSPT). To ignore NaN values in basic math operations, specify the *ignoreNan* as *true* in a name-value pair option.

Several series can be pulled at one time by specifying a cell array of strings. If you're pulling several series from the same database, use a name-value pair option to set a default database with *dbID,* which defaults to the USECON Haver database if no database is specified.

Data is returned as a Matlab [table](http://www.mathworks.com/help/matlab/tables.html) object with the specification string as the name of the series (though sometimes this is not possible and the name defaults to *series1*, *series2 …*

To restrict the dates of data being pulled, use the *startDate* and *endDate* name-value pairs with a Matlab serial date or a string date. If these options are omitted, the full history of the series will be retrieved. To achieve the same effect on a data series once it has been retrieved, see the cbd.trim function.

To get real-time data (FRED only), use the option *asOf*, or a combination of *asOfStart* and *asOfEnd* with serial or string dates.

Any option can also be applied to only a portion of the data retrieval with the hash (#) operator. To specify an option, follow a portion of a string with *#option* in order to set a boolean option to *true* or with *#option:value* to set the option to a given value. Note that the scope of the option is limited to the preceding expression so that (UNRATE - UNRATE#asOf:"1/1/2010")#dbID:"FRED" would give the revisions in the unemployment rate since the beginning of 2010. Note that when specifying a value for the option, it must be in double quotes .

## CHIDATA

In order to use the cbd functions with data outside either Haver or FRED, you can use the CHIDATA database. This is database mimics a Haver database so that you can pull data from it similarly (e.g., MYSERIES@CHIDATA).

To add a series to the database, use the **chidata\_save** function. This function should give warnings if you ever try to save over another series or change any of the properties associated with a series, but still attempt to make sure you’re not writing over anything important before using it.

Also note that in the CHIDATA directory, all of the .csv files are written in a particular format expected by Matlab and should generally not be changed by hand.

# Data Manipulation

The data manipulation functions can be grouped conceptually into two categories: transformations and summarizations. The description of these functions will assume that the data has already been retrieved and the functions are operating on the output from the **cbd.data** function. They can all be equally used within a specification of the data retrieval using **cbd.data**. Note that when calling them from Matlab, some functions may contain *Pct* at the end of the function name – these may be called from within the **cbd.data** function using a percent sign as in Haver (but the percent sign is not allowed in Matlab function names).

## Transformations

The transformation functions are based off of the Haver functions with a few extensions. All of the functions here take a table of data in and return a table with the same dimensions. Note that some of these functions take an optional number of periods as a second input.

The standard 15 Haver functions (diff, difa, difv, diff%, difa%, difv%, yryr, yryr%, yryrl, diffl, difal, difvl, movv, mova, movt) are all supported and behave as they do in Haver.

There are also the following additional transformation functions:

* **lag** - computes the lag or lead of a series. Takes the series as the first input and (optionally) a second input of the number of periods. Input a negative number of periods to compute a lead. Note that lag(series, 0) returns the series untransformed.
* **stddm** - demeans and standardizes data
* **indexed** – create an index out of a data series that is normalized to 100 at a given date with percent changes applied from that date.
* **nan2zero –** convertsNaN values to zeros. Primary created for use in the *ignoreNan* option of the addition and subtraction functions.
* **nan2one –** converts NaN values to ones. Primarily created for use in the *ignoreNan* option of the multiplication and division functions.
* **interpNan** – linear interpolation of a series (really a thin wrapper around the **interp1** Matlab function). Useful for graphing since Matlab doesn’t plot NaN values.

## Summarizations

The summarization functions take a table of values as an input and return a table of values that has a column for every series but only one row of values.

* mean
* median
* max
* min
* last – return the last non-NaN valued observation for a series
* first – return the first non-NaN valued observation for a series
* change, changePct – return the (percent) change over the course of the series

## Aggregation / Disaggregation

Series can be aggregated and disaggregated by period using the **cbd.agg** and **cbd.disagg** functions. Both of these functions take three arguments: a data series, a new frequency to be returned as (Y, Q, M, W, or D) and an aggregation (EOP, AVG, SUM) or disaggregation (NAN, FILL, INTERP) method. The **cbd.disagg** function can also take a fourth argument of whether or not to extrapolate the data.

## Binary Operations of Series

There are also functions to perform operations on two series. These can be called with data series in Matlab as the following functions:

* addition
* subtraction
* multiplication
* division

These functions are however primarily provided so that when retrieving data with **cbd.data** the normal operators can be used (i.e. +, -, \*, /).

## Merging tables

Use **cbd.merge** to combine two series into a single table. If the frequencies of the two series are different, the lower frequency series is disaggregated (using the NAN method) to the higher frequency. Note that this function can take multiple series as inputs.

# File Handling & Misc.

## xlsfile

The Matlab functions for interacting with Excel (xlsread, xlswrite, etc.) force an Excel process to be created and shut down with each function call. It is much more efficient to create an object to manage the Excel process instead interact with the file through this object, especially for large files or many read-write operation on one file.

For example, the following example is about twice as fast as the equivalent use of **xlsread**:

dataFile = cbd.xlsfile(‘giantDataFile.xlsx’);  
 dataA = dataFile.read(‘Sheet1’);  
 dataB = dataFile.read(`Sheet2’);

Generally it should work the same as the Matlab functions you already know by simply replacing **xlsread** with **myXlsfileVariable.read** and similarly for the write commands. See the function documentation for more information and for how to use the small added functionality such as exporting Excel files to PDF from Matlab.

## Calendar Date Functions

There are also several date functions since the Matlab functions that do much of this are inexplicably in the Financial Toolbox (which we have limited licenses for). Each of these functions can take either a string date or a Matlab serial date.

* year – get the year of a date
* quarter – get the quarter number of a date
* month – get the month number of a date
* day – get the day of month of a date

## Trim

The function **trim** will take a table as an input and a name-value pairs of either *startDate* or *endDate* to trim the table to the table to the requested dates.