# More on Database Management with IRIS

You should be comfortable with the material through `basic_dbase.pdf` before proceeding.

## Contents


- Clear workspace
- More database functions
- Converting databases
- Metadata in IRIS
- Graphical representations in IRIS


## Clear workspace

```
clear all;
close all;
```

## More database functions

For help on any of these functions, just type

```
help data/function_name
```

```
d = struct();
d.aa = tseries(qq(2000,1):qq(2010,4),@rand);
d.aa_2 = diff(d.aa);
d.cc = cumsum(d.aa);
d.dd = tseries(qq(1990,1):qq(2008,1),@randn);

dbnames(d,'nameFilter','aa\w*') a

rng = dbrange(d); b
fprintf('%s:%s \n',dat2char(rng(1)),dat2char(rng(end)))
```

**a** List of database entries filtered by name or class. In this particular example, we are only interested in names which start with aa and are followed by any number of alphanumeric characters, including the underscore.

**b** Find a range that encompasses the ranges of the listed `tseries` objects.

```
ans =

    'aa'    'aa_2'

1990Q1:2010Q4
```

## Converting databases

```
x  = rand(3,10);
d2 = array2db(x,qq(2000,1:10),{'x1','x2','x3'}); c

a = db2array(d,{'aa','aa_2'},qq(2005,1):qq(2009,1)) d

tt = db2tseries(d,{'cc','dd'},qq(2000,1:5)) e
```

**c** Convert numeric array to database. Note the alternative representation of a quarterly range – a similar syntax may be used with the functions `mm` and `yy`.

**d** Converts a list of tseries objects from an IRIS database over a given range to a numeric array.

**e** Convert numeric array to database.

```
a =

    0.0787  -0.0813
    0.3107   0.2320
    0.3242   0.0136
    0.1769  -0.1473
    0.2460   0.0691
    0.9918   0.7458
    0.6113  -0.3805
    0.4830  -0.1283
    0.2443  -0.2387
    0.9739   0.7296
    0.7154  -0.2585
    0.6984  -0.0170
    0.1748  -0.5236
    0.4704   0.2955
    0.8051   0.3347
    0.1097  -0.6954
    0.7950   0.6853


tt =

        tseries object: 5-by-2

        2000Q1: 0.79821   0.61121
        2000Q2:  1.036     1.2149
        2000Q3: 1.6094    -1.9265
        2000Q4: 2.2288   0.066858
        2001Q1: 2.6825     -2.629
    'cc'    'dd'

        user data: empty
```

## Metadata in IRIS

One of the biggest strengths of IRIS as a database management system is the intuitive and flexible way in which it stores metadata. Each `tseries` is endowed with a simple descriptor through the `.comment` field. Moreover, the `.userdata` field allows you to assign excruciatingly detailed descriptions of your data in the form of Matlab structures. Such functionality, if used wisely, allows for great productivity gains through automation.

```
comment(tt) f
tt.comment

thisText = 'This is my favorite macro timeseries.';
d.aa = comment(d.aa,thisText); g
d.aa.comment

ud = struct();
ud.Description = 'Index of Global Happiness';
ud.Source = 'Murky Research Ltd.';
ud.LastUpdated = datestr(now);
d.aa = userdata(d.aa,ud); h
d.aa.userdata

dd = dbload('dataComments.csv'); i
dd
dd.LGDP.comment
```

**f** Note that when we applied `db2tseries` above it automatically assigned the former fieldnames of the input database as comments to the newly created `tseries`. These are two equivalent ways to access those comments.

**g** A `comment` function allows you to assign comments from scratch.

**h** A `userdata` function allows you to associate a Matlab structure with a `tseries`.

**i** Of course, this functionality would be greatly undermined if one had to manually input comments for each series. Thankfully, IRIS is smarter than that – not only does `dbsave` save existing comments on the output CSV file, but if you have a CSV file with a line for comments immediately underneath the variable names, those are taken onboard by `dbload`.

```
ans =

    'cc'    'dd'


ans =

    'cc'    'dd'


ans =
```

```
    'This is my favorite macro timeseries.'


ans =

    Description: 'Index of Global Happiness'
         Source: 'Murky Research Ltd.'
    LastUpdated: '28-Dec-2010 17:52:25'


dd =

        LGDP: [175x1 tseries]
        LCPI: [175x1 tseries]
    LCPICORE: [175x1 tseries]
         UNR: [175x1 tseries]
        CAPU: [175x1 tseries]


ans =

    'Log real GDP x100'
```

## Graphical representations in IRIS

What if you want to plot some data which is already stored in IRIS format? Not a problem. First off, you are free you use all of Matlab's native graphing capabilities. There is one big advantage to doing that – the x-axis displays proper dates! For anyone who has experience trying to plot timeseries in Matlab from scratch this alone might be reason enough to persuade them to switch to IRIS. Better yet, again there are IRIS functions which make plotting even simpler – we will see those momentarily. Finally, IRIS features an extremely powerful and versatile `report` object – please refer to the `Reporting` folder.

```
plotRange = qq(2000,1):qq(2011,4);

figure();
subplot(2,2,1);
plot(plotRange,d.aa);
title('aa');
grid on;

subplot(2,2,2);
plot(plotRange,d.aa_2);
title('aa_2');
grid on;

subplot(2,2,3);
plot(plotRange,d.cc);
title('cc');
```

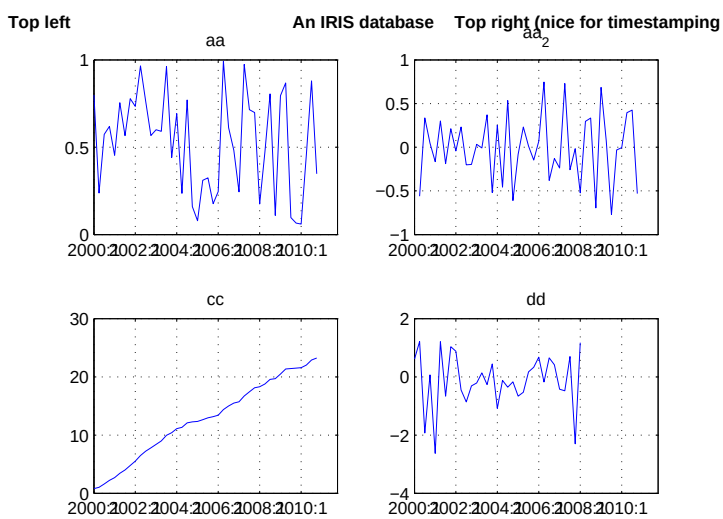
```
grid on;

subplot(2,2,4);
plot(plotRange,d.dd);
title('dd');
grid on;

qtitle({'Top left','An IRIS database','Top right (nice for timestamping'}); j

dbplot(d,{'aa','aa_2','cc','dd'},plotRange); k
qtitle({'Top left','An IRIS database',datestr(now)});
```

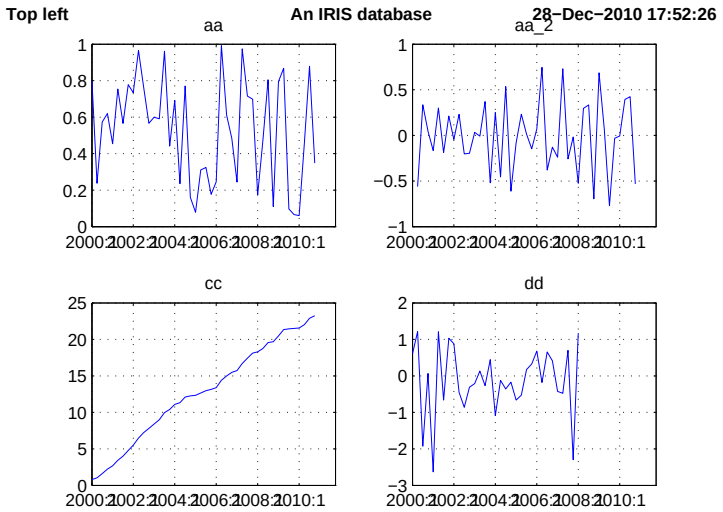
**j** This is an IRIS utility function which makes adding figure titles easy.

**k** Plot a batch of series from database. Notice how it reduced 18 lines of native Matlab code to 2 for the same graph! What is more, there are a multitude of great options a few keystrokes away, such as adding a zeroline, highlighting certain periods (eg. recessions), etc. Type

```
help data/dbplot
```

to see the complete list.




Top left
An IRIS database
28−Dec−2010 17:52:26
aa
aa_2
cc
dd