YourCast: Software for Simultaneous Time Series Forecasting with Your Assumptions[1]

Federico Girosi[2], Gary King[3], and Konstantin Kashin[4]

January 16, 2014

[1]Available from http://GKing.Harvard.Edu/yourcast via a Creative Commons Attribution-Noncommercial-No Derivative Works 3.0, for academic use only. For expert programming assistance and many helpful comments, our thanks goes to Jon Bishof, Elena Villalon, and Nirmala Ravishankar.

[2]Associate Professor in Population Health, University of Western Sidney , (Centre for Health Research, Building 3, Campbelltown Campus, Locked Bag 1797, Penrith NSW 2751, Australia; F.Girosi@uws.edu.au, +61 2 9852 5222 x3605).

[3]Albert J. Weatherhead III University Professor, Harvard University (Institute for Quantitative Social Science, 1737 Cambridge Street, Harvard University, Cambridge MA 02138; http://GKing.Harvard.Edu, King@Harvard.edu, (617) 495-2027).

[4]Ph.D. Candidate, Harvard University (Institute for Quantitative Social Science, 1737 Cambridge Street, Harvard University, Cambridge MA 02138; http://konstantinkashin.com, kkashin@fas.harvard.edu, (617) 871-9884).

Contents

1 Introduction

YourCast implements the methods for demographic forecasting discussed in:

> Federico Girosi and Gary King. 2008. *Demographic Forecasting.* Princeton: Princeton University Press, `http://gking.harvard.edu/files/abs/smooth-abs.shtml`.

Please read at least Chapter 1 of the book before attempting to use YourCast.

At its most basic, YourCast runs linear regressions, and estimates the usual quantities of interest, such as forecasts, causal effects, etc. The benefit of running YourCast over standard linear regression software comes from the improved performance due to estimating sets of regressions together in sophisticated ways.

YourCast avoids the bias that results from stacking datasets from separate cross-sections and assuming constant parameters, and the inefficiency that results from running independent regressions in each cross-section. YourCast instead allows you to tie the different regressions together probabilistically in ways consistent with what you know about the world and your data. The model allows you to have different covariates with different meanings measured in different cross-sections.

For example, one might assume that the separate time series regressions in neighboring (or "similar") countries are more alike. Our approach is fully Bayesian, but you need not assume as the standard Bayesian approach does that the *coefficients* (which are never observed) in neighboring countries are similar. YourCast makes it possible to assume instead that neighboring countries are similar in their values or trends in the *expected value of the dependent variable.* This approach is advantageous because prior knowledge almost always exists about the dependent variable (such as that the age profile of mortality looks like the Nike swoosh), and the expected value is always on the same metric even when including explanatory variables that differ in number or meaning in each country.

The power of YourCast to improve forecasts comes from allowing one to smooth in many sophisticated ways, in addition to across countries. You can thus decide whether to smooth over indices that are geographic, grouped versions of underlying continuous variables (such as age groups), time, or interactions among these. For example, you can assume that, unless contradicted by the data, forecasts should be relatively smooth over time, or that the forecast time trends should be similar in adjacent age groups, or even that the differences in time trends between adjacent age groups stay roughly similar as they vary over countries. The model works with time-series-cross-sectional (TS-CS) data but also data for which the time series varies over more than one cross-section (TS-CS-CS-CS... data such as log-mortality over time by age, country, sex, and cause). The specific notion of "smoothness" or "similarity" used in YourCast is also your choice. The assumptions made by the statistical model are therefore governed your choices, and the sophistication of those assumptions and the degree to which they match empirical reality are, for the most part, limited only by what you may know or are willing to assume rather than arbitrary choices embedded in a mathematical model. In our work, we have found that YourCast makes it possible to improve forecasts well beyond that possible with traditional regression (or autoregression) strategies, although of course we make no promises about the future except that your performance may vary.

2 Installation

YourCast requires the current version of R as well as the packages `sma`, `lattice`, and `foreign` which are loaded automatically by YourCast. Installation of YourCast differs slightly by operating system.

2.1 Linux/Unix

From the R command line, type

```
> install.packages("YourCast", repos="http://gking.harvard.edu", destdir="~/") .
```

Alternatively, from a Linux/Unix shell, download the current Unix bundle by typing in the same directory as the downloaded file:

```
> R CMD INSTALL --library=.R/library YourCast.tar.gz
```

2.2 Windows

From the R command line, type

```
> install.packages("YourCast", repos="http://gking.harvard.edu") .
```

Alternatively, download the Windows bundle from `http://gking.harvard.edu/yourcast`. From the GUI, click on the menu "Packages", click on the option "Install package(s) from local zip files", and select the zip file that you downloaded.

3 User's Guide

YourCast works with multiple data sets in the same model. Thus, we require a more complicated data structure than the usual single data frame used for most statistical models in R. Although you can create this object yourself and input it directly into the `yourcast()` function, it is normally easier to use the tools we created to build this data structure automatically. To do this requires following three steps: (1) labeling and organizing the data in a format we describe; (2) running the `yourprep()` function to prepare your data, create an input object in the format we need; and load it into R; and (3) running the `yourcast()` function to generate forecasts. We describe these steps in the following subsections, and follow it with a detailed example that illustrates all steps from start to end.

3.1 Data Preparations

YourCast operates on time series cross-sectional data indexed by (1) a time period such as a year, (2) a grouped continuous variable such as an age group, and (3) a spatial or geographic variable such as geographic region or country. To fix ideas, we refer to these as time, age, and geography, respectively, but obviously they may change in other applications. (Either the age or geography indexes, but not both, may be dropped if desired.) We require a single dependent variable, such as mortality rates, to be the same and have the same meaning for all units **(author?)** [see 1, §8.4 for an exeception]. Covariates may differ in number, meaning, and content across both age and geography.

Thus, YourCast analyzes a set of data sets, each defined for one cross-sectional unit indexed by age and/or geography. Inside the data set corresponding to each cross-sectional unit is a time series with measures on the dependent variable and the covariates observed in this cross section. An example would be an annual time series (say 1952-1996) with the dependent variable of mortality rates and several covariates, all within the cross-section of 15-20 year olds in Uganda. All cross-sections should have the same time indices (1952-1996), possibly with some different overlapping observation periods, the same dependent variable, and covariates that are the same, completely different, or overlapping from cross-section to cross-section.

All the individual cross-section data sets (each containing a time series) must be on in a single subdirectory on disk in fixed width text files ('`.txt`'), comma separated value files ('`.csv`'), or Stata data files ('`.dta`'). (Alternatively, they may be in memory, in your R workspace.) Each file must be named with one string in three parts: an alphanumeric tag of the user's choice, a geography code of between zero (for no geography index) and four digits, and an age group code of between zero (for no age group index) to four digits. For example, if you have observations on cancer deaths for age group 45 (which might represent 45-50 year olds) for U.S. citizens (e.g., geography code '2450'), you may decide to choose tag "cancer". We would add a file extension as well and so if the data are in a plain text file, we put these elements together and the file name would be '`cancer245045.txt`'.

So we can understand your coding scheme, include an extra file in the same directory called '`tag.index.code`', where "tag" is the actual alphanumeric tag you chose (not the word t-a-g). For the example above, the filename would be '`cancer.index.code`'. The contents of the file should be 0–4 letters `g` followed by 0–4 letters `a`. In the example above, the entire contents of the file is: `ggggaa`.

Optionally, you may also add files that contain labels for each of the time, age, and geography codes. If these are included, they will make text and graphics output easier to interpret (and they may be useful documentation for you separate from yourcast). The files are `tag.T.names` for time periods, `tag.A.names` for age groups, and `tag.G.names` for geographic regions, where again "tag" is your chosen alphanumeric tag. The contents of each file should be ASCII text with all valid numerical codes in the first column and a corresponding label in the second column. Include column labels in the first row. So for geography, the second column might be country names and the columns would be labeled "region" and "name". If the codes are interpretable as is, such as is often the case for age groups and time periods, then you can omit the corresponding file.

Finally, if you wish to smooth over geographic regions, which the 'map' and 'bayes' methods allow, you must also include a file called `tag.proximity.txt` where "tag" is your chosen alphanumeric tag and "'`.txt`'" is used for text files but can also be used with comma separated files ('`.csv`'), or Stata data files ('`.dta`'). The larger the proximity score, the more proximate that pair of countries is in the prior; a zero element means the two geographic areas are unrelated, and the diagonal is ignored. Each row of the `proximity` file has three columns, consisting of geographic codes for two countries followed by a score indicating the proximity of the two geographic regions; please include column labels. For convenience, geographic regions that are unrelated (and would have zero entries in the symmetric matrix) may be omitted from `proximity`. In addition, `proximity` may include rows corresponding to geographic regions not included in the present analysis.

3.2 Loading in the Data

We load in all the data described in the previous section at once by using the function `yourprep()`. The only required argument is the chosen alphanumeric tag (and the subdirectory name if its not the working directory). The program will then attempt to load all files in that directory beginning with the chosen tag and will ignore the rest. Then run the function. For example:

```
ydata <- yourprep(tag="cancerMales")
```

The output object `ydata` (of class 'yourprep') now includes all the data and associated information needed for making forecasts.

3.3 Making Forecasts

To make forecasts, we require the data object, name the variables with a standard R formula, and the model. A regression is estimated for each cross section (tied together by any chosen priors), and so an explanatory variable listed in the formula is used for a particular cross-section if it is in the formula and it is present in that cross-section's data set. (That is, a variable not measured for a cross-section is dropped only for that cross-section.) As an example:

```
ylc <- yourcast(formula=log(rspi2/popu2) ~ time, dataobj=dta, model="LC")
```

Finally, the output object `ylc` (of class yourcast) can be plotted with function `plot.yourcast(ylc)` summarized with function `summarize(ylc)`, or accessed directly (use `names(ylc)` to see the contents).

3.4 Example

We now reconstruct the demo `chp.11.1` from start to finish to illustrate the capabilities of the YourCast software and provide further illustration to the user on how to take advantage of them.

Most users will not have their data in a format easily readable by `yourcast()`. Thus for this example we will start with the raw '`.txt`' files and take advantage of the `yourprep()` software designed to help users construct the 'dataobj' list easily.

We have stored the original files we used to create the 'dataobj' returned by typing `data(chp.11.1)` in YourCast's 'data' folder. You can view these files, which have the extension '`.txt`', by typing:

```
> dir(paste(.libPaths()[1],"/YourCast/data",sep=""))
 [1] "chp.11.1.RData"     "chp.11.10.RData"    "chp.11.11.RData"    "chp.11.12.RData"
 [5] "chp.11.13.RData"    "chp.11.2.RData"     "chp.11.3.RData"     "chp.11.4.RData"
 [9] "chp.11.5.RData"     "chp.11.7.1.RData"   "chp.11.7.2.RData"   "chp.11.8.1.RData"
[13] "chp.11.8.2.RData"   "chp.11.8.3.RData"   "chp.11.9.1.RData"   "chp.11.9.2.RData"
[17] "chp.2.6.1.RData"    "chp.2.6.2.RData"    "chp.2.7.1.RData"    "chp.2.7.2.RData"
[21] "chp.2.7.3.RData"    "cntry.codes.txt"    "csid204500.txt"     "csid204505.txt"
[25] "csid204510.txt"     "csid204515.txt"     "csid204520.txt"     "csid204525.txt"
[29] "csid204530.txt"     "csid204535.txt"     "csid204540.txt"     "csid204545.txt"
[33] "csid204550.txt"     "csid204555.txt"     "csid204560.txt"     "csid204565.txt"
[37] "csid204570.txt"     "csid204575.txt"     "csid204580.txt"     "proximity.txt"
```

We could load these files using the `data()` command, but for this example we will pretend they are files loaded into our working directory that we want `yourcast()` to be able to read.

The function `yourprep()` in the YourCast package is designed to help you turn these raw files into a 'dataobj' that `yourcast()` can read. The `yourprep()` function works by scanning either the working directory or another directory you specify for files beginning with the tag 'csid'. In the 'data' folder we scanned above, there are several files whose names consist of the 'csid' tag and a CSID code in the format we will specify to the function. These are the labels `yourprep()` needs to be able to recognize and process the files. All files should have an extension so that `yourprep()` knows how to read them; currently the function supports fixed-width '`.txt`' files, comma-separated value files, and Stata '`.dta`' files.

Let's examine the first of these cross section text files, '`csid204500.txt`'. As we can see below, this file contains all the years from the first observed year to the last predicted year, with missing values replaced by `NA`s. Because it was created in the R software, this file already has the years written in as rownames in a way that R can read (and for this reason has only three column labels).

```
"rspi2" "popu2" "time"
"1950" NA 5457 1920
"1951" NA 6319 1921
"1952" NA 7009 1922
"1953" NA 7553 1923
"1954" NA 7978 1924
...
"2026" NA NA 1996
"2027" NA NA 1997
"2028" NA NA 1998
"2029" NA NA 1999
"2030" NA NA 2000
```

However, we expect that most users will have input that looks like the next file in the directory, '`csid204505.txt`'. Below we can see that the observation year is an extra variable rather than a rowname.

```
year rspi2 popu2 time
1950 NA 3978 1920
1951 NA 4091 1921
1952 NA 4306 1922
1953 NA 4583 1923
1954 NA 4889 1924
...
2026 NA NA 1996
2027 NA NA 1997
2028 NA NA 1998
2029 NA NA 1999
2030 NA NA 2000
```

If this is the case, you should set the argument `year.var` to `TRUE` in `yourprep()`; this will automatically convert the 'year' variable to a rowname as long as it is labeled 'year'.

The 'data' directory also includes some of the optional files that we included in our 'dataobj' for the `chp.11.1` demo. The first is '`proximity.txt`', a list of proximity scores for pairs of the geographic units. The second is '`cntry.codes.txt`', a list of all the CSID codes for the geographic units and their respective labels. We will load these using arguments in the `yourprep()` function.

We're now ready to run the `yourprep()` function. Since the function already grabs all files beginning with 'csid' tag, we only need to specify the directory where the files are stored and the names of the optional files, `G.names` and `proximity`. Note that we have set `year.var=TRUE` since one of our files has the observation year as a separate variable rather than as the rowname:

```
dta <- yourprep(dpath=paste(.libPaths()[1],"/YourCast/data",sep=""),
                year.var=TRUE, sample.frame=c(1950,2000,2001,2030),
                G.names="cntry.codes.txt", proximity="proximity.txt",
                verbose=TRUE)
```

We have now created a 'dataobj' called `dta`. Examining the 'dataobj', we can see that it includes the two required elements, 'data' and 'index.code', as well as two optional elements.

```
> names(dta)
[1] "data"       "index.code" "G.names"    "proximity"
```

Examining the 'data' element, we can see that it includes all the cross section files that were in the `dpath`:

```
> names(dta$data)
 [1] "204500" "204505" "204510" "204515" "204520" "204525" "204530" "204535"
 [9] "204540" "204545" "204550" "204555" "204560" "204565" "204570" "204575"
[17] "204580"
```

We're now ready for a run of `yourcast()`. The first run of the program in the `chp.11.1` demo file uses the Lee-Carter model. This model uses few of the capacities of the YourCast package since it does no smoothing, but is good for a quick run of the function. Use of the smoothing options can be seen in many of the demos and is explained the `yourcast()` documentation. The code below produces an output object called `ylc` that is of class 'yourcast':

```
ylc <- yourcast(formula=log(rspi2/popu2) ~ time, dataobj=dta, model="LC")
```

The main output from the `yourcast()` function is the 'yhat' element of the output list, which contains the observed and predicted values for every cross section. This output is difficult to appreciate without graphics, but we can get a quick summary of our run of the function by typing `summary(ylc)`:

```
> summary(ylc)
Model: LC
Number of cross sections: 17
Formula: log(rspi2/popu2) ~ time

Observed period: 1950-2000
Forecast period: 2001-2030

Smoothing parameters:
 Ha.sigma  Ht.sigma Hat.sigma
      0.3       0.3       0.2

Geo units included:
[1] "2045"

See 'help(plot.yourcast)' for instructions on how to plot observed and predicted 'y' values
```

Here we can see basic information about the output object. More information not printed automatically is available by typing `names(summary(ylc))`.

We're now ready to plot the observed and predicted values to study the model output. This can be done simply by typing `plot(ylc)`, but we have added a few arguments here to enhance the graphical output. The argument `title` gives a title for the plots by describing the dependent variable. The argument `age.opts` allows us to pass options to the `age` plot in the form of a list object. For example, we can choose to not plot the predicted 'yhat' values in-sample through the `insamp.predict=FALSE` option. You can see more of these options by typing `help(plot.yourcast)`.

```
plot(ylc, title="Respiratory Infections", age.opts=list(insamp.predict=FALSE))
```

Since we did not specify which type of plot we wanted, the default combination of `age` and `time` plots is returned. However, the plotting function can also do either of these plots separately, as well as three dimensional age-time plots, total count plots, and life expectancy at birth plots. To see these, we need to use the `plots` argument. For example, here is a call for the three dimensional plot:

```
plot(ylc,title="Respiratory Infections", plots="threedim")
```

Finally, if your analysis includes a large number of geographic areas such that viewing output sequentially on the device is inconvenient, there an option in the plotting function to save the output for each geographic code as a '`.pdf`' file in the working directory rather than printing it to the device window. Just set `print="pdf"` and pass filename and output directory options to the function as a list object using the `print.args` argument.

This ends the example section of the users guide. Please visit the help files for individual functions or send an email to the YourCast listserv if you have problems.

3.5 More Information

We have included demos that provide step-by-step instructions on how to reproduce to graphs in Chapters 2 and 11 of *Demographic Forecasting.* A list of these demos can be found by typing

```
demo(package="YourCast")
```

at the command prompt. You can also access the preassembled 'dataobj's used in these demos directly by typing

```
data(package="YourCast")
```

To either run the demos or load these datasets, replace the package name in the respective argument with the name of the demo or dataset of interest; i.e., `demo(chp.11.1)`. The next section goes through this particular demo in detail.

For more information on the statistical methods implemented in this software, please refer to *Demographic Forecasting.*

4 Reference

The following pages list the main functions in YourCast with detailed reference information. These can also be loaded from R with the standard help command, such as `help(yourprep)`.

Description

Runs a set of regression models to forecast time-series cross-sectional data by either considering independent regressions in each cross-sectional unit or by using a variety of techniques to smooth across units.

Usage

```
yourcast(formula=NULL, dataobj=NULL,sample.frame=c(1950,2000,2001,2030),
                     standardize=TRUE, elim.collinear=FALSE,
                     tol=0.9999, solve.tol = 1.e-10,svdtol=10^(-10),
                     userfile=NULL, savetmp = T, model.frame=FALSE,
                     debug = F,  rerun= "yourcast.savetmp",
          ### specific to models
                     model="OLS",zero.mean=FALSE,
          #### smooth over ages
                     Ha.sigma = 0.3,
                     Ha.sigma.sd= 0.1, Ha.deriv=c(0,0,1),
                     Ha.age.weight=0, Ha.time.weight=0,
          #### smooth over time
                     Ht.sigma= 0.3,
                     Ht.sigma.sd=0.1,  Ht.deriv=c(0,0,1),
                     Ht.age.weight=0, Ht.time.weight=0,
          #### smooth over age-time
                     Hat.sigma=0.2,
                     Hat.sigma.sd=0.1,Hat.a.deriv=c(0,1),Hat.t.deriv=c(0,1),
                     Hat.age.weight=0,Hat.time.weight=0,
          #### smooth over cntry-time
                     Hct.sigma=0.3, Hct.sigma.sd =0.1,
                     Hct.t.deriv=1, Hct.time.weight = 0,
                     LI.sigma.mean=0.2,LI.sigma.sd = 0.1, nsample= 500,
                     low.pow=T, verbose=TRUE)
```

Arguments

`formula` A standard R formula of the form $y \sim x_1 + x_2$, except that an explanatory variable is included for a particular cross-section only if it is both listed in the formula and available in that cross-section's data set (see `dataobj`). Explanatory variables in the formula but not available for a cross-section (or in a cross-sectional dataset but not in the formula) are excluded. (For mortality forecasting, the specification looks like log(deaths/population) $\sim x_1 + x_2$, with deaths and population stored as separate variables in each dataframe.) (May be set to `NULL` if `savetmp` was set to `TRUE` on the last run, in which case the value of formula will come from the saved file.)

`dataobj` A object of class 'yourcast' or equivalent. See `help(yourprep)` for more details.

The `dataobj` may be supplied in one of four ways. Most commonly, the argument will specify (1) an object (in working memory) or (2) a string with the name of a file in the working directory. However, if (3) `dataobj` is a string referring to a directory on disk, then each element of the list above should be stored in a file in that directory, with element 'data' consisting of a subdirectory containing separate ASCII data files. (If this option is chosen, a complete data object, called 'dataobj.Rdata', will be stored in the directory named, and it will be loaded automatically if yourcast is run again with this chosen option.) (4) The last option is for dataobj to be set to NULL, after which the function will look for a 'yourcast.savetmp' file in the working directory from a previous run of the function where the argument savetmp was set to TRUE.

The function yourprep is available to help construct the dataobj in the proper format from individual cross section files in the working directory or the workspace. This function also performs a number of diagnostics to ensure that the data is entered properly and can be read by yourcast. See help(yourprep) for more information

sample.frame — Vector. A four element vector containing, in order, the start and end time periods to be used for the observed data and the start and end time periods to be forecast. Years identified here that are not available for a cross-section are ignored. Default: c(1950,2000,2001,2030).

standardize — Boolean. Should the covariates in each cross-sectional unit be standardized (to zero mean and standard deviation of 1)? Standardization is performed for both the in- and out-of-sample periods. Default: TRUE.

elim.collinear — Boolean. Whether collinearity among covariates should be tested and those that are collinear shoul be eliminated. Default: FALSE.

tol — Double scalar. Tolerance to find collinearities among covariates. Default: 0.9999.

solve.tol — A real number smaller than one that is used in the argument of the R-function solve to invert matrices (see description for tol). Default: 1^{-10}.

svdtol — A scalar; the tolerance used in inverting a matrix by SVD. Default: 10^{-10}.

userfile — A string with the name of a file that contains your values for some or all of yourcast's arguments. This file contains R code that changes default values of arguments. E.g., the file might contain:

```
index.code <- 30
data <- "WHOmortalityData"
```

If an option is specified in userfile, it takes precidence over command line options, so it is normally best to specify each option in either the userfile or the command line but not both. Default: NULL

savetmp — If TRUE, yourcast saves a file in the default directory (called 'yourcast.savetmp') with preliminary calculations. If the value of formula or dataobj is missing when yourcast is called, yourcast will get their values from this file, if it exists. This saves a minute or so of computing time for large data sets and is useful for multiple runs on the same data with different formulas specified or different prior values. If FALSE, no file is saved. (The structure of 'yourcast.savetmp' is for the convenience of yourcast and is not intended to be read by the user or saved for more than one run.) Default: TRUE.

model.frame — If TRUE, include entire input dataobj in the output object. Default: FALSE.

debug — Boolean. It puts the environment that contains parameters and arguments of the simulation in the user workspace. Default FALSE.

rerun — String. The name of the file that is saved in the default directory with preliminary calculations; see savetmp. Default: yourcast.savetmp

model — A string indicating the forecasting method, including: Bayes maximum a posteriori (map), Bayes with Gibbs sampling (bayes), Ordinary Least Squares (`ols`), Poisson (`poisson`), and Lee-Carter (`LC`). Default: `ols`. (We usually recommend `map`.)

`yourcast` also includes a procedure to help users set the sigma parameters below automatically for the case of model=`map`, and smoothing over age, time, or age and time, but for only one country. You may do this by running a preprocessing instance of `yourcast` first by setting this parameter to `ebayes` and using either the data to be analyzed or a larger data set which is likely to have similar or related parameter values. When `ebayes` is chosen, the `yourcast` output object will contain only the parameter values to feed into the next run of `yourcast`.

`zero.mean` A boolean or named vector with a value of $\bar{\mu}$ for each age group. If `TRUE`, the prior has zero mean. If `FALSE`, the prior has nonzero mean centered around the observed mean age profile (i.e., the average of Y over time and levels of the geographic index for each age group). Default: `FALSE`.

`Ha.sigma` This can be set in one of three ways: (1) a scalar which sets σ_a, the prior standard deviation of $E(Y)$, indicating how much to smooth $E(Y)$ over age groups (which may vary over geographic areas and time periods, and with the standard deviations averaged over age groups). A larger standard deviation represents more prior uncertainty, which allows the data to play a greater role. (2) `NA` to not smooth in this way. (3) To have `yourcast` search for a good value based on a target value of the derivative of $E(Y)$ with respect to age, set to a vector of elements containing the start and end of a range in sigma in which to look (such as 0.05 and 1.5), the number of values to look at within this range (such as 5), and the target value of the derivative of $E(Y)$ with respect to age (such as 0.05). The vector may also include a fifth element, which is the target value of the total standard deviation of $E(Y)$ over all dimensions of the prior (such as 0.1). (You may choose to run `yourcast` with model=`ebayes` on a related data set to find an approximate target value of the derivative and standard deviation automatically.) Default: `0.30`.

`Ha.sigma.sd` A scalar; the standard deviation of parameter Ha.sigma (for Gibbs sampling only). Default: `0.1`.

`Ha.deriv` A numeric vector, each element of which is n,the degree of a (discrete) derivative of the smoothness functional with respect to the age group. Element k of this vector refers to the $(k-1)$th derivative, where 0 excludes the derviative, 1 includes it, and values in between include the derivative but weight it down proportionally. The first element of the vector corresponds to the weight on the derivative with respect to age of order 0 (the identity operator), the second to the weight on the derivative of order 1 (the 1st derivative), etc. For example, c(0, 1, 1) corresponds to a mixed functional that penalizes the first and second derivatives equally. The higher the order of derivative, the more local smoothness over age groups; and lowest specified derivative controls the form of prior indifference. Default: `c(0, 0, 1)`, which usually works well.

`Ha.age.weight` A scalar or a numeric vector with weights that determine how much smoothing occurs for different age groups. If set to 0 or NA, age groups are weighted equally; if set to a nonzero scalar, the weight for age group a is set proportional to $a^H a.age.weight$; if a vector of length A, the ath element is the weight of age group a. Default: `0`.

`Ha.time.weight`

A scalar or a numeric vector with weights that determine how much smoothing occurs for different time periods when smoothing over age groups. If `0` or `NA`, time periods are weighted equally; if set to a nonzero scalar value, the weight for time period t in smoothing age groups is proportional to $t^H a.time.weight$; if the argument is a vector of length T, the tth element is the weight of time period t. Default: `0`.

`Ht.sigma` This can be set in one of three ways: (1) a scalar which sets σ_t, the prior standard deviation of $E(Y)$, indicating how much to smooth $E(Y)$ over time periods (which may vary over geographic areas and age groups, and with the standard deviations averaged over time periods). A larger standard deviation represents more prior uncertainty, which allows the data to play a greater role. (2) NA to not smooth in this way. (3) To have `yourcast` search for a good value based on a target value of the derivative of $E(Y)$ with respect to time, set to a vector of elements containing the start and end of a range in sigma in which to look (such as 0.05 and 1.5), the number of values to look at within this range (such as 5), and the target value of the derivative of $E(Y)$ with respect to time (such as 0.05). The vector may also include a fifth element, which is the target value of the total standard deviation of $E(Y)$ over all dimensions of the prior (such as 0.1). (You may choose to run `yourcast` with model=`ebayes` on a related data set to find an approximate target value of the derivative and standard deviation automatically.) Default: `0.30`.

`Ht.sigma.sd` A scalar; the standard deviation of parameter `Ht.sigma` (for Gibbs sampling only). Default: `0.1`.

`Ht.deriv` A numeric vector, each element of which is n, the degree of a (discrete) derivative of the smoothness functional with respect to time. Element k of this vector refers to the $(k-1)$th derivative, where 0 excludes the derviative, 1 includes it, and values in between include the derivative but weight it down proportionally. The first element of the vector corresponds to the weight on the derivative with respect to time of order 0 (the identity operator), the second to the weight on the derivative of order 1 (the 1st derivative), etc. For example, `c(0, 1, 1)` corresponds to a mixed functional that penalizes the first and second derivatives equally. The higher the order of derivative, the more local smoothness over time; and lowest specified derivative controls the form of prior indifference. Default: `c(0, 0, 1)`, which usually works well.

`Ht.age.weight` A scalar or a numeric vector with weights that determine how much smoothing occurs for different age groups when smoothing over time. If set to `0` or `NA`, age groups are weighted equally in smoothing over time; if set to a nonzero scalar, the weight for age group a is set proportional to $a^H t.age.weight$; if a vector of length A, the ath element is the weight of age group a. Default: 0.

`Ht.time.weight`

A scalar or a numeric vector with weights that determine how much smoothing occurs for different time periods when smoothing over time. If `0` or `NA`, time periods are weighted equally; if set to a nonzero scalar value, the weight for time period t in smoothing time periods is proportional to $t^H t.time.weight$; if the argument is a vector of length T, the tth element is the weight of time period t. Default: 0.

`Hat.sigma` This can be set in one of three ways: (1) a scalar which sets σ_{at}, the prior standard deviation of $E(Y)$, indicating how much to smooth the time trend in $E(Y)$ over age groups. A larger standard deviation represents more prior uncertainty, which allows the data to play a greater role. (2) NA to not smooth in this way. (3) To have `yourcast` search for a good value based on a target value of the derivative of $E(Y)$ with respect to age and time, set to a vector of elements containing the start and end of a range in sigma in which to look (such as 0.05 and 1.5), the number of values to look at within this range (such as 5), and the target value of the derivative of $E(Y)$ with respect to age and time (such as 0.05). The vector may also include a fifth element, which is the target value of the total standard deviation of $E(Y)$ over all dimensions of the prior (such as 0.1). (You may choose to run `yourcast` with model=`ebayes` on a related data set to find an approximate target value of the derivative and standard deviation automatically.) Default: `0.2`.

`Hat.sigma.sd` A scalar; the standard deviation of parameter `Hat.sigma` (for Gibbs sampling only). Default: `0.1`.

`Hat.a.deriv` A numeric vector, each element of which is n, the degree of a (discrete) derivative of the smoothness functional of time trends with respect to age groups. Element k of this vector refers to the $(k-1)$th derivative of the time trend v with respect to age, where 0 excludes the derviative, 1 includes it, and values in between include the derivative but weight it down proportionally. The first element of the vector corresponds to the weight on the derivative of the time trend with respect to age of order 0 (the identity operator), the second to the weight on the derivative of order 1 (the 1st derivative), etc. For example, `c(0, 1, 1)` corresponds to a mixed functional that penalizes the first and second derivatives equally. The higher the order of derivative, the more local smoothness over time; and lowest specified derivative controls the form of prior indifference. Default: `c(0, 0, 1)`, which usually works well.

`Hat.t.deriv` A numeric vector, each element of which is n, the degree of a (discrete) derivative of the smoothness functional of age derivative with respect to time. Element k of this vector refers to the $(k-1)$th derivative of the age derivative with respect to time, where 0 excludes the derviative, 1 includes it, and values in between include the derivative but weight it down proportionally. The first element of the vector corresponds to the weight on the age derivative with respect to time of order 0 (the identity operator), the second to the weight on the derivative of order 1 (the 1st derivative), etc. For example, `c(0, 1, 1)` corresponds to a mixed functional that penalizes the first and second derivatives equally. The higher the order of derivative, the more local smoothness over time; and lowest specified derivative controls the form of prior indifference. Default: `c(0, 0, 1)`, which usually works well.

`Hat.age.weight`

A scalar or a numeric vector with weights that determines how much smoothing occurs for different age groups when smoothing over age and time. If set to `0` or `NA`, age groups are weighted equally in smoothing over time; if set to a nonzero scalar, the weight for age group a is set proportional to $a^H t.age.weight$; if a vector of length A, the ath element is the weight of age group a. Default: `0`.

`Hat.time.weight`

A scalar or a numeric vector with weights that determine how much smoothing occurs for different time periods when smoothing over age and time. If `0` or `NA`, time periods are weighted equally; if set to a nonzero scalar value, the weight for time period t in smoothing time periods is proportional to $t^H t.time.weight$; if the argument is a vector of length T, the tth element is the weight of time period t. Default: `0`.

`Hct.sigma` A scalar which sets σ_t, the prior standard deviation of $E(Y)$, which indicates how to smooth $E(Y)$ over geographic areas, or NA to not smooth in this way. The parameter $\sigma_c t$ is the expected prior standard deviation of $E(Y)$ for a geographic area (varying over time periods and age groups, and with the standard deviations averaged over geographic areas). (A larger standard deviation represents more prior uncertainty, which allows the data to play a greater role.) Default: `0.3`.

`Hct.sigma.sd` A scalar; the standard deviation of parameter Ht.sigma (for Gibbs sampling only). Default: `0.1`.

`Hct.t.deriv` A numeric vector; controls whether smoothing the level or the time trend of $E(Y)$ over geographic areas (both cannot presently be done simultaneously). To smooth the level of $E(Y)$ over geographic areas, set to 1, the identity.

To smooth the time trend, set this (as in `Hat.t.deriv`) to the weight of the partial derivative taken with respect to time in the standard smoothness functional for the prior. The use of the first or higher order partial derivatives are supported. Default: `1`.

`Hct.time.weight`
: A scalar or a numeric vector with weights that determine how much smoothing occurs for different time periods when smoothing over geographic areas. If `0` or `NA`, time periods are weighted equally; if set to a nonzero scalar value, the weight for time period t in smoothing over areas is proportional to $t^H ct.time.weight$; if the argument is a vector of length T, the tth element is the weight of time period t. Default: `0`.

`LI.sigma.mean` A scalar; used in the likelihood and in the calculation of the priors in conjunction with `Ha.sigma.sd`, `Hat.sigma.sd`, `Ht.sigma.sd`, and `Hct.sigma.sd`. Default: `0.2`.

`LI.sigma.sd` A scalar; the standard deviation of `LI.sigma.mean` used in the calculation of the priors. Default: `0.1`.

`nsample` A scalar; represents the number of iterations in the Gibbs algorithm `bayes`. Default: `500`.

`low.pow` Boolean. Whether to include lower-power of explanatory variables in the simulation as derived from `formula`. For example $y \sim x^4$, if `low.pow` = `TRUE`, then x, x^2, x^3, x^4 will be included. Default: `TRUE`.

`verbose` Boolean. Suppress verbose output. Default: `FALSE`

Value

Returns a list of class 'yourcast' containing the following components:

`call` The full call, including all command line options when yourcast was called.

`userfile` The full userfile if it was specified.

`yhat` A list with the same cross-sectional elements as the input data, but with two columns: 'y' for the observed dependent variable and 'yhat' for the predicted values. These include both in-sample and out-of-sample values, as distinguished by the values of `sample.frame`.

`coeff` A list with the same cross-sectional elements as the input data, elements of which are the estimated coefficients if calculated by the chosen model.

`sigma` A list with the same cross-sectional elements as the input data, elements of which are the estimated standard error of the estimate of the regression (the standard deviation of the dependent variable given the explanatory variables).

`aux` List. A list of summary information about the yourcast analysis used by `plot.yourcast`

`params` Vector. Smoothing parameters used in model.

Author(s)

Federico Girosi `<girosi@rand.org>`; Elena Villalon `<evillalon@iq.harvard.edu>`; Gary King `<king@harvard.edu>`

References

`http://gking.harvard.edu/yourcast`

Description

Builds the data object for `yourcast` function from files in working directory or other specified directory and checks for errors

Usage

```
yourprep(dpath=getwd(),tag="csid",index.code="ggggaa",
                        datalist=NULL,G.names=NULL,A.names=NULL,
                        T.names=NULL,proximity=NULL,year.var=FALSE,
                        sample.frame=NULL,summary=FALSE,verbose=FALSE,

                        #lagging utility
                        lag=NULL,formula=NULL,vars.nolag=NULL)
```

Arguments

`dpath` String. Name of the directory where data files are stored. If `NULL` then defaults to working directory. Default: `NULL`

`tag` String. Group of characters placed before CSID code in filenames to indicate which files in `dpath` function should load. The `tag` can also be used to differentiate between different groups to be considered in separate analysis; for example, 'm' for male deaths and 'f' for female deaths. Default: `"csid"`

`index.code` String indicating how the CSID index variable is coded in the input data. Between 0 and 4 of the following two characters are used in this order: `g` for the geographic index (such as country) and `a` for a grouped continuous variable like an age group. For example, `ggggaa` would have the function interpret '245045' by using '2450' as the country code and '45' as the age group. Default: `"ggggaa"`

`datalist` A list of cross section dataframes already loaded into the workspace to be added to the `dataobj`. Names of list elements should be the numerical CSID code for each cross section, and dataframes should be formated identically to files loaded from an external directory (see Details)

`A.names, G.names, T.names`
String. Filename of optional two-column data files that list all valid numerical codes (in the first column) and corresponding alphanumeric names (optionally in the second column) for the indices corresponding to geographic areas in `G.names`, age groups in `A.names`, and time periods in `T.names`. Function will search `dpath` for file with specified name; please include column labels. The optional alphanumeric identifiers are most commonly only used for geographic areas since numerical values for age groups and time periods are usually meaingful on their own. However, if other grouped continuous variable used in place of ages, for example, specifying these labels will be important for output to be meaningful. NOTE: Auxiliary files will loaded automatically by `yourprep()` if they are saved in the `dpath` and labeled with the `tag` specified by the user. See 'Details' section for more infromation. Default: `NULL`

`proximity` Data file with codes to construct the symmetric matrix (geographic region by geographic region) of proximity scores for geographic smoothing used by the 'map' and 'bayes' methods. The larger the relative score, the more proximate that pair of countries is in the prior; a zero element means the two geographic areas are unrelated (the diagonal is ignored). Each row of the `proximity` file has three columns, consisting of geographic codes for two countries and a score indicating the proximity or similarity of the two geographic regions; please include column labels. For convenience, geographic regions that are unrelated (and would have zero entries in the symmetric matrix) may be omitted from `proximity`. In addition, `proximity` may include rows corresponding to geographic regions not included in the present analysis. Default: `NULL`

`year.var` Boolean. Should be `TRUE` if `year` coded as separate variable rather than as rowname for cross section data files. Function will look for `year` variable to use as rownames and then drop it from the dataframe. Change will only be made to dataframe if it does not already have rownames or if exisiting rownames are merely a '1...N' index of row numbers, so it is possible to apply correction even if some cross sections do not have a `year` variable and already have the correct rownames. Default: `FALSE`

`sample.frame` Optional four element vector containing, in order, the start and end time periods to be used for the observed data and the start and end time periods to be forecast. All cross sections do not have to begin at starting date, but must contain all years after the first observed value. Variables to be forecasted should be coded as `NA` in the out-of-sample period. Note that this makes it easy to reserve a range of values of the dependent variable for out-of-sample forecasting evaluation; our `summary` and `plot` functions in `yourcast` will make these comparisons automatically if the out-of-sample data are included. `yourprep()` uses this information only to verify that cross sections are correctly constructed, but it should also be included if one wants to use the lag utility. Default: `NULL`

`summary` Boolean. If `TRUE`, means for available observations on each variable are displayed for the cross sections read by `yourprep()`. Default: `FALSE`

`verbose` Boolean. If `TRUE`, function prints name of each cross section or auxiliary file as it is read into the `dataobj`. Default: `FALSE`

`lag` Number of years covariate data needs to be lagged from current position is cross section files. See 'Details' for more information. Default: `NULL`

`formula` Formula. The formula that one will use in the subsequent run of `yourcast()`. This helps the lagging utility distinguish between the response variable (which will not be shifted between cross sections) and the covariates of interest that should be lagged and included in the final cross sections of the dataobj. If the covariate 'index' is included in the formula, the lagging utility will include a variable in the cross sections that starts from 1 and counts the number of time periods since the start of the cross section. If a lag is requested, the formula argument must be specified. Default: `NULL`

`vars.nolag` Vector of strings. Vector of variables to be included in the dataobj but not lagged. These variables do not need to be included in the formula, and if found there will not ignored when the other covariates are lagged.

Details

Creates `dataobj` input for `yourcast` from files in working directory or other specified directory. Checks that all cross sections in `data` list titled properly and if all years up to last predicted year included in the dataframes (if `sample.frame` argument specified). Please note, however, that all cross sections from the same geographic area must have the same observation and prediction years in the dataframe (even if `NA`) for the graphing software `plot.yourcast` to work.

The cross section files must be named according to the CSID identifiers for country code and age group, preceeded by the specified tag (default: "csid") so that yourprep() can identify the file from other files in the dpath. For example, for the USA (country code 2450) time series of 45 year old individuals, the file name should be 'csid245045.txt' if the tag is left as the default. Files must have an extension so that the program can recognize how the data is coded. Currently, fixed width text files ('*.txt'), comma-separated values ('*.csv'), and Stata v.5-10 ('*.dta') files are supported, and multiple file types may be used in the same run of the program. '*.Rdata' objects can be included with the datalist option after they are loaded to a list in the workspace. yourprep() includes diagnostics to ensure that objects are properly named and not included accidentally, but users should examine the specified dpath before running yourprep() to minimize errors.

Each cross section file should be labeled columns of time-series data for the dependent variable(s) (e.g., disease, pop) and the covariates that will be used in the forecast. The rownames for the dataframe should be the observation year (if the year is coded as a separate variable, set year.var=TRUE). The files must contain the full time series that will be specified in the sample.frame argument in yourcast after the first observed year. For instance, if sample.frame=c(1950,2000,2001,2030), then files would have observations that start between 1950 and 2000 and include all other years (even if the entries are NA) up to the last year of prediction, i.e., 2030.

Optional auxiliary files such as G.names should be named according to the filename specified in the respective arguments. If specified, these files must have extensions and be coded in one of the three supported file types. However, these files will be automatically loaded by yourprep() if they are saved in the dpath and labeled with the tag specified by the user. The default names for these files must be used (e.g., 'G.names' and 'proximity'). For example, if the tag is left as the default and there is a file in the dpath labeled 'csid.G.names.txt', yourprep() will load this automatically and save the input as the G.names element of the 'dataobj' list. yourprep() arguments such as G.names take precedence over 'TAG.*' files in thedpath.

yourprep() also includes a lagging utility (activated once one specifies a lag length with the 'lag' argument). This utility is useful for when the data in each cross section is, for example, the response and covariates for 50 year olds in each year but the desired content for each cross section is the response for 50 year olds and the covariates for 25 year olds 25 years prior to each year (implying a lag of 25 years). In order to have yourprep() perform this lagging automatically, include cross sections for each age group with data starting the same number of years before the first observation year as the requested lag period. Thus if lag=25 and the first observation year is 1950, then the cross sections should all start at 1925. Age groups younger than the length of the lag will not retain covariate data (except perhaps an 'index' variable) in the output object. The covariates lagged are the predictor variables specified in the formula argument.

If data for a cohort 25 years (in this case) younger is not available for some cohort over age 25, yourprep() will look for the closest cohort available and issue a warning message.

Value

dataobj — A list with several components:

- **data** A list with the cross-sectional data matrices as elements.
- **proximity** A three-column matrix of proximity scores for geographic smoothing used by the 'map' and 'bayes' methods. For each row, the first two columns indicate the country pair. The third column indicates the proximity score. The larger each score, the more proximate that pair of countries is in the prior; a zero element means the two geographic areas are unrelated (the diagonal is ignored).
- **G.names, A.names, T.names** Optional two-column dataframes that list all valid numerical codes (in the first column, labeled codes) and corresponding alphanumeric names (optionally in the second column, labeled

name) for the indices corresponding to the geographic areas in `G.names`, age groups in `A.names`, and time periods in `T.names`.

index.code A string indicating how the index variable is coded in the input data.

Author(s)

Jon Bischof <jbischof@fas.harvard.edu>

References

http://gking.harvard.edu/yourcast

See Also

`yourcast` function and documentation (`help(yourcast)`)

Examples

```
## Not run:
# Working directory automatically set to directory with cross
# section and auxiliary files to begin. Files for this example
# in 'data' folder of YourCast library.

#Old working directory to be restored later
oldwd <- getwd()
# Now setting wd to 'data' folder in YourCast library
setwd(system.file("data",package="YourCast"))

# Simple run of the function, using option that turns year variable
# into label in each cs. Use sample.frame argument for all diagnostics
# to work

dta <- yourprep(G.names="cntry.codes.txt", proximity="proximity.txt",
year.var=TRUE,verbose=TRUE,sample.frame=c(1950,2000,2001,2030))


# With summary output (means of variables in each cross section)


dta <- yourprep(G.names="cntry.codes.txt", proximity ="proximity.txt",
year.var=TRUE,summary=TRUE)


# Function can also add datafiles already loaded into R as objects in
# the workspace with "datalist" option if put into a list and properly
# labeled. All diagnostics still performed
# 'csid204545', etc., are dataframes in workspace

# Labels changed to nonsense ones so as not to confuse with other files

data(csid204545)
data(csid204550)
data(csid204555)

datalist <- list("123456"=csid204545,"234567"=csid204550,
"345678"=csid204555)

# Verbose option turned on and datalist argument added
```

```
dta <- yourprep(G.names="cntry.codes.txt", proximity="proximity.txt",
year.var=TRUE,verbose=TRUE,datalist=datalist)

# Setting working directory back
setwd(oldwd)
rm(oldwd)

## End(Not run)
```

plot.yourcast	*Plot generation tool for YourCast*

Description

Creates graphics from YourCast output for each geographical unit and prints to the device window or a .pdf file in the specified directory.

Usage

```
## S3 method for class 'yourcast'
plot(x, plots=c("age","time"), title=NULL, subtitle=NULL,
     age.opts=list(), time.opts=list(), threedim.opts=list(),totcount.opts=list(),
     ex0.opts=list(),print="device", print.args=list(), dv.log=NA, ...)
```

Arguments

x	A `yourcast` output object.
plots	Vector of strings specifying the plot types to be generated. The available plots are age profile (`"age"`), time profile (`"time"`), 3D age/time profile (`"threedim"`), total counts by year (`"totcounts"`), and life expectancy at birth (`"ex0"`). The order in which the plots are specified is the order in which they are generated in the device. A maximum of 3 plot types is allowed. Default: `c("age","time")`.
title	String. Main title for the plots. Concatenated with country name if `dataobj$G.names` exists. Default: `NULL`.
subtitle	String. Subtitle for the plots. Default: `NULL`.
age.opts	A list of options for the age profile plot. See 'Details'.
time.opts	A list of options for the time profile plot. See 'Details'.
threedim.opts	A list of options for the 3D plot. See 'Details'.
totcount.opts	A list of options for the total counts plot. See 'Details'.
ex0.opts	A list of options for the life expectancy at birth plot. See 'Details'.
print	String specifying whether graphical output should be displayed sequentially on a device window (`"device"`) or saved directly to a '`.pdf`' file in the `dpath` (`"pdf"`). Default: `"device"`.
print.args	A list of options for the device. See 'Details'.
dv.log	Logical. Is the dependent variable logged? If `NA`, the function parses the formula in the `yourcast` object to determine if the dependent variable is logged. Default: `NA`.
...	Further arguments passed to or from other methods.

Details

Plots age profiles, time profiles, 3D age/time profiles, total counts (by time), and life expectancy at birth from `yourcast` output. Total count (by time) plots should only be used if forecasting counts (eg. population or death counts) and life expectancy at birth should only be used if forecasting mortality rates - both are common quantities of interest calculated from these forecasts and are thus included in the function. See `lifetable` for details of life expectancy at birth calculations from mortality rates. All plots are created using the `ggplot2` package, except for the 3D age/time profile, which is created using the `wireframe` function from the `lattice` library.

The function supports multiple plots in each call (up to 3) and arranges them horizontally in the order specified in `plots` argument. The device window is automatically resized to accomodate different number of plots, albeit device width and height may be specified manually through the `print.args` option (see below).

Plots are titled with the `title` and the `G.names` dataframe if it was supplied to `yourcast` in the `dataobj`. For example if `title="Respiratory Infections"` and the geographic identifier for that region is matched with `"Belize"`, the plot will be titled "Respiratory Infections, Belize". One or both labels will be utilized by the function if available. A subtitle may also be specified through `subtitle`.

It is important to note that `plot.yourcast` will only work if all cross sections within the same geographic unit are of the same dimensions. If, for example, a cross section for one age group has fewer yearly observations than another from the same group, these missing years must be filled in with `NA`, even if they occur in the beginning of the sample period. This does not hold across geographic units, however.

Finally, note that `plot.yourcast` opens a new device window for each new plot. This is done so that the size of the device can be controlled to keep the side-by-side plots from appearing distorted when launched.

Options for 'age', 'time', 'totcount', and 'ex0' plots

Options for the age profile, time profile, total count, and life expectancy at birth plots may be specified using the `age.opts`, `time.opts`, `totcount.opts`, and `ex0.opts` arguments, respectively, which are lists with any of following components:

`xlab` String specifying the x-axis label. Default is `"Age"` for 'age' plots and `"Time"` for 'time', 'totcount', and 'ex0' plots.

`ylab` String specifying the y-axis label. Default is `"Data and Forecasts"` for 'age' plots and `"Forecasts"` for 'time' plots.

`insamp.obs` Logical. Should the observed data be plotted? Default for 'age', 'totcounts', 'ex0' is `TRUE`; default for 'time' is `FALSE`.

`insamp.predict` Logical. Should the predicted data be plotted for the in-sample period? Default: `TRUE`.

Additionally, the following options may be passed to `age.opts` and `time.opts` lists:

`age.select` A numeric vector listing the ages to be plotted. If a scalar is supplied, it is used as the step size in a sequence of ages between the minimum and the maximum age. If `NULL`, all ages are plotted. Default: `NULL`.

`time.select` A numeric vector listing the times to be plotted. If a scalar is supplied, it is used as the step size in a sequence of times between the minimum and the maximum time. If `NULL`, all times are plotted. Default: `NULL`.

`unlog` Logical. Should the dependent variable be unlogged for 'age' and 'time' plots? Ignored if `dv.log` is set to `FALSE`. Default: `FALSE`.

Option `age.select` may be included in the `totcount.opts` list, but it functions differently than for the 'age' and 'time' plots. Instead of designating the ages to plot, `age.select` should be a numeric vector listing the ages to be summed over in the calculation of total counts over time.

For 'ex0' plots, options `ax`, `a0`, and `nx` that control the calculation of life tables from mortality rates in `yourcast` objects may be included in the `ex0.opts` list. By default, `ax` is set to 0.5, `a0` is set to 0.07, and `nx` is set to `NULL`. See `lifetable` for additional information.

Options for 'threedim' plot

Options for the 3D age/time profile may be specified through `threedim.opts`. The argument must be a list with any of the following components:

xlab String specifying the x-axis label. This corresponds to the time dimension. Default: "Time".

ylab String specifying the y-axis label. This corresponds to the age dimension. Default: "Age".

zlab String specifying the z-axis label. This corresponds to the dependent variable. Default: "Forecasts"

insamp.pred Logical. If TRUE, the forecasted data is plotted for the in-sample period. If FALSE, the observed data is plotted instead. Default: TRUE.

unlog Should the dependent variable be unlogged? Ignored if dv.log is set to FALSE. Default: FALSE.

args.wireframe A list of arguments (must be labeled) to be passed to wireframe. Note that some arguments to wireframe such as zlab and screen should be made as arguments to threedim.opts; they will be overwritten if found in args.wireframe.

screen List with three elements 'x', 'y', and 'z' that rotate the viewing angle for the 3D plot. Default: list(z=-40, x=-60, y=0).

Print options

The function prints sequentially to the device or saves '.pdf' files with the requested plot for each geographic unit in the sample. If requested, '.pdf' files will be saved in a specified directory or the working directory. Use print.args to set print options:

height Height of device. Default is 5 inches.

width Width of device. Default width adjusts automatically according to number of plots.

filename A string or vector of strings denoting the filename(s) of '.pdf's created for each geographical area if print="pdf". If a string, the csid code is automatically appended if multiple geographies so as to ensure unique filenames. If the vector is of the length equal to the number of geographies, the csid is not appended, albeit the order of the labels should be the same as the order of the areas in dataobj$data. Default: NULL, in which case plots are titled as plotgggg.pdf, where gggg denotes the csid.

dpath A string giving the directory where '.pdf' files will be output if print="pdf". Default: getwd().

Value

Device windows with requested plots or '.pdf' files saved in the dpath.

Author(s)

Konstantin Kashin <kkashin@fas.harvard.edu>, based on an earlier version by Jon Bischof.

References

http://gking.harvard.edu/yourcast

See Also

yourcast function and documentation (help(yourcast))

Examples

```
# Run Lee-Carter model for Figure 2.6 in Demographic Forecasting
data(chp.2.6.1)
ff.allc <- log(allc2/popu2) ~  time
ylc.allc <- yourcast(formula=ff.allc, dataobj=chp.2.6.1, model="LC",
                       elim.collinear=FALSE,
```

```
                       sample.frame=c(1950,2000,2001,2060))
# default
plot(ylc.allc,title="All causes of death")

## Not run:
# output plots to pdf
plot(ylc.allc,title="All causes of death", print="pdf")

# output plots to pdf with names "acd"+csid
plot(ylc.allc,title="All causes of death", print="pdf",
print.args=list(filename="acd"))

# order matters
plot(ylc.allc,plots=c("time","age"),title="All causes of death")
plot(ylc.allc,plots=c("age","time"),title="All causes of death")
## End(Not run)

# remove in-sampled predicted data from age profile plot
plot(ylc.allc,title="All causes of death",age.opts=list(insamp.predict=FALSE))

# plot every 10th age instead of every 5th age for age profile plot
plot(ylc.allc,title="All causes of death",
age.opts=list(insamp.predict=FALSE,age.select=10))

## Not run:
# plot every 5th year instead of every year for time series plot
plot(ylc.allc,title="All causes of death",
age.opts=list(insamp.predict=FALSE),time.opts=list(time.select=5))

# only plot data from 2000+
plot(ylc.allc,title="All causes of death",
age.opts=list(insamp.predict=FALSE, time.select=seq(2001,2060,5)),
time.opts=list(time.select=seq(2001,2060,5)))

# plot unlogged mortality rates
plot(ylc.allc,title="All causes of death",
age.opts=list(insamp.predict=FALSE, unlog=TRUE),time.opts=list(unlog=TRUE))

# plot `threedim' plot in addition to age and time profiles
plot(ylc.allc,plots=c("threedim","age","time"),
title="All causes of death",age.opts=list(insamp.predict=FALSE))

# plot life expectancy at birth in addition to age and time profiles
# since these are 5-year age groups, assume ax=2.5 and a0=1.907
plot(ylc.allc,plots=c("ex0","age","time"),title="All causes of death",
age.opts=list(insamp.predict=FALSE), ex0.opts=list(a0=1.907,ax=2.5))
## End(Not run)
```

Description

Draws histograms for priors calculated by model ebayes.

Usage

```
histograph(d1.a, d1.t, dt.da, SD, depvar=" ",
           model="ebayes", graphics.file=NA)
```

Arguments

`d1.a`	Numeric vector. First derivative respect to age.
`d1.t`	Numeric vector. First derivative respect to time.
`dt.da`	Numeric vector. Second derivative respect to age and time.
`SD`	Numeric vector. Standard deviation.
`depvar`	String with the name of the dependent variable. Default: `" "`
`model`	String with the name of the model. Default: `"ebayes"`.
`graphics.file`	String or `NA`. If string the name of a file to be appended to a directory path where the graphics will be saved, if `NA` it is displayed in the screen and not saved. Default: `NA`.

Value

Histograms of the vectors `d1.a`, `d1.t`, `dt.da`, and `SD`; see demos `chp.11.7`, `chp.11.8`, or `chp.11.9`.

Author(s)

Federico Girosi `<girosi@rand.org>`

References

`http://gking.harvard.edu/yourcast`

summary.yourcast	*Summarize yourcast output*

Description

yourcast class summary function

Usage

```
## S3 method for class 'yourcast'
summary(object, ...)
```

Arguments

`object`	`yourcast` output object to be summarized
`...`	Arguments to be passed to or from other methods.

Value

A list of summary objects of class 'summary.yourcast':

`sample.frame`	Four element vector containing, in order, the start and end time periods to be used for the observed data and the start and end time periods to be forecast
`params`	Vector. Smoothing parameters used in model.
`model`	String. Estimation technique used.
`formula`	Language. Formula used in analysis.
`numcs`	Integer. Number of cross sections used in analysis.
`cntry.codes`	Vector. Unique geographical codes of cross sections used in analysis.
`cntry.names`	Vector. If `G.names` dataframe submitted to `yourcast`, the names of all unique geographical codes of cross sections used in analysis.
`coef`	List. List of matrices with estimated coefficents for each cross section. For Lee-Carter models, only one set of coefficents calculated. Standard errors are not calculated for these coefficients.

Author(s)

Jon Bischof `<jbischof@fas.harvard.edu>`

References

`http://gking.harvard.edu/yourcast`

See Also

`yourcast`,`print.summary.yourcast`

lifetable	*Life table calculation for YourCast*

Description

Creates a list of observed and predicted period life tables across from YourCast output for each geographical unit. This function should only be used when forecasting all-cause mortality rates.

Usage

```
lifetable(obj,ax=0.5,a0=0.07,nx=NULL,dv.log=NA)
```

Arguments

`obj`	A `yourcast` output object containing all-cause mortality rates.
`ax`	A scalar, numeric vector of length A, or a numeric T by A matrix specifying the average years lived within a given age interval $[x, x+n)$ by individuals who die within that age interval. If a scalar, `ax` is assumed to be constant across all age intervals (except for the first age interval; see `a0`) and across time. If a vector, `ax` is assumed to be constant across time. See 'Details'. Default: 0.5.
`a0`	A scalar indicating the average years lived within the first age interval by individuals who die within that age interval. Ignored if `ax` is a vector or a matrix. Default: 0.07.
`nx`	A scalar or vector of length A defining the age interval widths. If a scalar, the age intervals have identical widths. If `NULL`, the age interval widths are calculated using the ages in the `yourcast` object. The last age interval is assumed to be open-ended so its length is set to `NA`. Default: `NULL`.
`dv.log`	Logical. Is the dependent variable logged? If `NA`, the function parses the formula in the `yourcast` object to determine if the dependent variable was logged. Default: `NA`.

Details

The input must be a `yourcast` object containing forecasts of all-cause mortality rates. The death rates do not have to be for one-year age intervals, albeit `ax` and `a0` should be adjusted accordingly based on the age interval size. The rates may either be logged or unlogged. If `dv.log` is set to `NULL`, the function parses the formula automatically to determine whether the rates are logged. Setting `dv.log` to `TRUE` or `FALSE` manually sets whether the rates are logged or not.

Specifying `ax` is necessary to convert mortality rates into conditional probabilities of death. Suppose that m_x is the mortality rate for the n-year age interval $[x, x+n)$. a_x is then defined as the average years lived within the age interval $[x, x+n)$ by individuals who die at that age. The default value of a_x is 0.5, which means that individuals on average live half the year within the age that they die (for single-year age intervals). The default value of a_0 is set to 0.07, following the reported values within the Human Mortality Database for adult females in the United States for the $[0, 1)$ age interval in the last decade.

Alternatively, one can specify `ax` as a vector of length A, where A is the number of age intervals. In this case, a_x is assumed to be constant across time for each age interval. One can also pass a T by A matrix, in which case a_x is allowed to vary over time.

The last age interval is open-ended: $[x, \infty)$. The conditional probability of death is set to 1 and a_x is automatically set to $1/m_x$.

Value

If a single geographic area, a list with names `obs` and `pred` is returned, where the elements are life table arrays calculated using observed and predicted mortality rates, respectively. If multiple geographic areas, a named list of geographic areas is returned, where each element contains a list of observed and predicted life tables.

Each life table array has dimensions T by A by 10, where the third dimension contains the following variables:

`x`	Age interval.
`nx`	Length of age interval.
`ax`	Average years lived within age interval by individuals who die during that age interval.
`qx`	Conditional probability of death for the age interval.
`px`	Conditional probability of survival for the age interval.
`lx`	Number of survivors at the start of the age interval. Set to 100,000 for the first age interval.
`dx`	Number of individuals dying within the age interval.
`Lx`	Person-years lived by individuals in the age interval.
`Tx`	Person-years remaining at start of age interval.
`ex`	Remaining life expectancy at start of age interval.

Author(s)

Konstantin Kashin `<kkashin@fas.harvard.edu>`

References

Preston, S.H., Patrick Heuveline, Michel Guillot. (2000) *Demography: Measuring and Modeling Population Processes.* Blackwell Publishing.

Wilmoth, J.R. et al. "Methods Protocol for the Human Mortality Database" `http://www.mortality.org/`.

`http://gking.harvard.edu/yourcast`

See Also

`yourcast` function and documentation (`help(yourcast)`)

Examples

```
# Run Lee-Carter model for Figure 2.6 in Demographic Forecasting
data(chp.2.6.1)
ff.allc <- log(allc2/popu2) ~  time
ylc.allc <- yourcast(formula=ff.allc, dataobj=chp.2.6.1, model="LC",
                       elim.collinear=FALSE,
                       sample.frame=c(1950,2000,2001,2060))

# calculate lifetable function assuming ax=2.5 and a0=1.907
# a0 calculated for both males and females from
# Human Mortality Database for New Zealand for 2000
lt <- lifetable(ylc.allc,ax=2.5,a0=1.907)

# observed lifetable for csid 5150 (New Zealand) for 2000
lt[["5150"]]$obs["2000",,]

# predicted lifetable for csid 5150 (New Zealand) for 2015
lt[["5150"]]$pred["2015",,]
```

`array.yourcast` *Array generation tool for YourCast*

Description

Creates array from YourCast output for each geographical unit.

Usage

```
array.yourcast(x, unlog=FALSE)
```

Arguments

`x`	A `yourcast` output object.
`unlog`	Logical. Should the dependent variable be unlogged? Default: `FALSE`.

Value

Creates array(s) from `yourcast` output. One array is created per geographic area. If there is one geographic area, an array is returned; else, a named list of arrays is returned. Each array is of size T (number of times) by A (number of ages) by 3, where the last dimension captures the type of data: `y` (observed values), `yhat` (predicted values), and `comb` (observed values in-sample and predicted values out-of-sample).

Author(s)

Konstantin Kashin `<kkashin@fas.harvard.edu>`.

References

`http://gking.harvard.edu/yourcast`

See Also

`yourcast` function and documentation (`help(yourcast)`)

Examples

```
# Run Lee-Carter model for Figure 2.6 in Demographic Forecasting
data(chp.2.6.1)
ff.allc <- log(allc2/popu2) ~  time
ylc.allc <- yourcast(formula=ff.allc, dataobj=chp.2.6.1, model="LC",
                       elim.collinear=FALSE,
                       sample.frame=c(1950,2000,2001,2060))

yc.array <- array.yourcast(ylc.allc)
dimnames(yc.array)

# predicted mortality rates (observed in-sample and predicted out-of-sample)
yc.array[,,"comb"]
```

References

[1] Federico Girosi and Gary King. *Demographic Forecasting*. Princeton University Press, Princeton, 2008. http://gking.harvard.edu/files/smooth/.