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

Federico Girosi[2] Gary King[3]

September 4, 2008

[1]Available from http://GKing.Harvard.Edu/yourcast.

[2]Policy Researcher, RAND Corporation, (1700 Main Street, PO Box 2138, Santa Monica, CA 90407-2138; http://www.ai.mit.edu/people/girosi/, girosi@rand.org, (310) 393-0411 x7794)

[3]David Florence Professor of Government, 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).

Contents

1 Introduction

YourCast implements the methods for demographic forecasting discussed in:

> Federico Girosi and Gary King. 2008. *Demographic Forecasting.* Princeton: Princeton University Press.

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 does not require that you have the same covariates with the same meaning measured in every cross-section.

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 that *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, 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.

2 Installation

YourCast requires `R` version 1.71 or later; it also requires 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

3.1 YourCast basics

Normal use of the YourCast software involves three steps: (1) labeling and organizing the data in a directory to be read by the software, (2) running the `yourprep()` function to load the data into R and create an input object in the format read by `yourcast()`, (3) and running the `yourcast()` function to generate predicted values for the dependent variable of interest.

The goal of the first two steps in this process is to create the specialized 'list' object (referred to here as the 'dataobj') from raw data files supplied by the user. In this section we will first describe how the list must be organized to be read by `yourcast()` and then how to use the `yourprep()` function to create an object of class 'yourprep' that meets these criteria.

3.1.1 The 'dataobj' input

YourCast takes as input a 'list' object, which must be organized in a specific way to be read by the software. The first element of the list, which must be labeled "data", is a list of all the cross sections to be included in the analysis. Each cross section should be labeled by its cross section ID (CSID) code. See below for more details.

Each cross section element of the 'data' list must also be organized in a specific manner. Each needs to contain the dependent variable that will be used in the `formula` argument given to `yourcast()` and all the independent variables relevant to that cross section. All columns should be labeled with the names of the respective variables, and the rows should be labeled with the observation year, whether it is in sample or out of sample. For the out of sample range, values for many variables will obviously be missing. These gaps should all be filled in with `NA`s, and the dataframe should include all years from the first observed year to the last predicted year.

The `dataobj` list should also contain a number of other elements that specify how the data is organized and how it should be labeled. The first, `index.code`, is mandatory. This element should be a string that indicates how the CSID codes are organized. 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.

The 'dataobj' list can also contain several other elements that provide labeling information to make `yourcast()` output easier to interpret, as well information on the spatial relationships between different geographical areas. More information on these optional elements can be found in the `yourprep()` documentation.

3.1.2 Creating the 'dataobj' and running `yourcast()`

The `yourprep()` function will assist you in creating a 'dataobj' that can be read by `yourcast()`. Here we outline the basic steps for using the software, but please see section 3.2 as well for an illustrative example.

`yourprep()` assumes that the user has data saved as individual cross section files that are stored in your workspace in R or in a directory on your hard drive in one of three different formats: fixed width text files, comma separated value files, and Stata '.dta' files. In YourCast a cross section refers to a single age group for a given geographic area. For example, if the user has observations on cancer deaths for several age groups between 5 and 80 of U.S. citizens (geo code '2450'), data for each group from '245005' to '245080' should be saved in a separate file in the directory. When saving the files, it is important to include the file extension ('.txt', '.csv', or '.dta') so that the function knows how process each data format.

The `yourprep()` function works by scanning the chosen directory for files beginning with a `tag` specified by the user. This tag system allows the function to load possibly thousands of cross section files without needing to know the filename of every one of them. 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. If the user requests the 'm' tag, the function will only pull the observations for males from the directory. Thus, for the cross section of cancer deaths of males aged 45 from the U.S., the respective file might be titled 'cancerm245045.txt'.

Additional files that contain lableling or spatial relationship information should also be stored in the chosen directory in one of the three accepted file formats. The names of these files will be supplied to the function in separate arguments in the call.

Finally, `yourprep()` also allows the user to include cross section files already loaded in the workspace, provided they are gathered into a list with each element labeled with the full CSID code for that cross section. Please see the `yourprep()` documentation for more information.

The output of the `yourprep()` function is an object of class 'yourprep' that meets all of requirements for the `yourcast()` 'dataobj' input enumerated above. Once given data in a readable format, `yourcast()` can produce forecasts with several different estimation techniques and use this output to generate three different types of plots. Plots are created with the function `plot.yourcast()`. In order to familiarize users with the software, we have included a number of 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)`.

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

3.2 Extended Example: YourCast from start to finish

In this section we reconstruct the results of 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] "adjacency.txt"     "chp.11.10.RData"  "chp.11.11.RData"  "chp.11.12.RData"
 [5] "chp.11.13.RData"  "chp.11.1.RData"   "chp.11.2.RData"   "chp.11.3.RData"
 [9] "chp.11.4.RData"   "chp.11.5.RData"   "chp.11.7.1.RData" "chp.11.7.2.RData"
[13] "chp.11.8.1.RData" "chp.11.8.2.RData" "chp.11.8.3.RData" "chp.11.9.1.RData"
[17] "chp.11.9.2.RData" "chp.2.6.1.RData"  "chp.2.6.2.RData"  "chp.2.7.1.RData"
[21] "chp.2.7.2.RData"  "chp.2.7.3.RData"  "cntry.codes.txt"  "cs2s204545.txt"
[25] "csid204500.txt"   "csid204505.txt"   "csid204510.txt"   "csid204515.txt"
[29] "csid204520.txt"   "csid204525.txt"   "csid204530.txt"   "csid204535.txt"
[33] "csid204540.txt"   "csid204545.txt"   "csid204550.txt"   "csid204555.txt"
[37] "csid204560.txt"   "csid204565.txt"   "csid204570.txt"   "csid204575.txt"
[41] "csid204580.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 'adjacency.txt', a list of proximity scores for all the possible 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 `adjacency`. 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", adjacency="adjacency.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"    "adjacency"
```

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 `dvlabel` gives a title for the plots by describing the dependent variable. The second argument, `age.insamp.predict`, tells the function not to plot predicted 'yhat' values in sample in the `age` plots. You can see more of these options by typing `help(plot.yourcast)`.

```
plot(ylc, dvlabel="Respiratory Infections", age.insamp.predict=FALSE)
```

Since we did not specify which type of plot we wanted, the program gave us the default combination of `age` and `time` plots. However, the plotting function can also do either of these plots separately, as well as three dimensional plots. To see these, we need to use the `family` argument. For example, here is a call for the three dimensional plot:

```
plot(ylc, dvlabel="Respiratory Infections", family="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"`.

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.

4 Reference

4.1 yourcast: Time-series cross-sectional Forecasting

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=F,
                     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`

4.2 `yourprep`: Data object creation wizard for 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, adjacency=NULL, year.var=FALSE,
                     sample.frame=NULL, summary=FALSE, verbose=FALSE)
```

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. Default: `NULL`

`adjacency` 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. 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`

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.

Value

`dataobj` A list with several components:

data A list with the cross-sectional data matrices as elements.

proximity A symmetric matrix (geographic region by geographic region) of proximity scores for geographic smoothing used by the 'map' and 'bayes' methods. The larger each element of the matrix, 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 element of the symmetric matrix is created from one row of the proximity input to `yourprep()` (which is two country codes and a proximity score).

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

```
# 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(paste(.libPaths()[1],"/YourCast/data",sep=""))

# 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",adjacency="adjacency.txt",
year.var=TRUE,verbose=TRUE,sample.frame=c(1950,2000,2001,2030))

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

## Not run:
dta <- yourprep(G.names="cntry.codes.txt",adjacency="adjacency.txt",
year.var=TRUE,summary=TRUE)
## End(Not run)

# 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",adjacency="adjacency.txt",
year.var=TRUE,verbose=TRUE,datalist=datalist)

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

4.3 `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, dpath=getwd(), dvlabel=NULL, family="agetime",
                            time.insamp.obs=TRUE, time.insamp.predict=TRUE,
                            age.insamp.predict=TRUE,
                            threedim.insamp.predict=TRUE,
                            age.xlab=NULL, age.ylab=NULL,
                            time.xlab=NULL, time.ylab=NULL,
                            threedim.xlab=NULL, threedim.ylab=NULL,
                            threedim.zlab=NULL,
                            screen=list(z=-40, x=-60, y=0),
                            print="device", ...)
```

Arguments

`x` `yourcast` output object

`dpath` String. Directory where '`.pdf`' outputs are saved. Defaults to working directory if not specified. Will be ignored if `print="device"`. Default: `getwd()`

`dvlabel` String. Description of dependent variable that will be used as the main title for the plots. Default: `NULL`

`family` String. Specifies type of plot generated by the function. `time` creates a time series plot where each age cohort is plotted separately on the device. `age` creates a plot of forecasts on age where data from each each year is plotted separately. `agetime` creates a side-by-side presentation of the `age` and `time` plots. Finally, `threedim` creates a three-dimensional plot of the information in the 'age' and 'time' plots. Default: `agetime`

`time.insamp.obs` Boolean. For `time` and 'time' plots in `agetime` plots, specifies whether observed values should be plotted within the range of years with observations for the dependent variable. Default: `TRUE`

`time.insamp.predict` Boolean. For `time` and 'time' plots in `agetime` plots, specifies whether predicted values should be plotted within the range of years with observations for the dependent variable. For `time` plots, the default is to print both predicted and observed values; this option merely removes predicted values from this range if set to `FALSE`. Default: `TRUE`

`age.insamp.predict` Boolean. For `age` plots and the 'age' plots of `agetime` plots, specifies whether predicted values should be plotted within the range of years with observations for the dependent variable. If set to `FALSE`, predicted values are replaced by observed values in this range. Default: `TRUE`

`threedim.insamp.predict` Boolean. For `threedim` plots, specifies whether predicted values should be plotted within the range of years with observations for the dependent variable. If set to `FALSE`, predicted values are replaced by observed values in this range. Default: `TRUE`

`age.xlab` String. The label for the 'x' axis of `age` plots and the 'age' plot of `agetime` plots. Will be ignored if 'age' plot not created. Default: `"Age"`

`age.ylab`	String. The label for the 'y' axis of `age` plots and the 'age' plot of `agetime` plots. Will be ignored if 'age' plot not created. Default: `"Forecasts"`
`time.xlab`	String. The label for the 'x' axis of `time` plots and the 'time' plot of `agetime` plots. Will be ignored if 'time' plot not created. Default: `"Time"`
`time.ylab`	String. The label for the 'y' axis of `time` plots and the 'time' plot of `agetime` plots. Will be ignored if 'time' plot not created. Default: `"Data and Forecasts"`
`threedim.xlab`	String. The label for the 'x' axis of `threedim` plots. Will be ignored if 'threedim' plot not created. Default: `"Year"`
`threedim.ylab`	String. The label for the 'y' axis of `threedim` plots. Will be ignored if 'threedim' plot not created. Default: `"Age"`
`threedim.zlab`	String. The label for the 'z' axis of `threedim` plots. Will be ignored if 'threedim' plot not created. Default: `"Forecasts"`
`screen`	List. List with three elements 'x', 'y', and 'z' that rotate the viewing angle for three dimensional plots. Argument ignored for all other plot types. Default: `list(z=-40, x=-60, y=0)`
`print`	Boolean. Specifies whether graphical output should be displayed sequentially on a device window (`device`) or saved directly to a '`.pdf`' file in the `dpath`. Default: `"device"`
`...`	Arguments to be passed to par() for the purpose of setting graphical parameters for the plotting functions. See `help(par)` for more details.

Details

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. Three-dimensional plots are created with the `wireframe` function from the `lattice` library. For space considerations, axes are labeled with the numerical versions of the 'age' and 'time' vectors regardless of whether `A.names` and `T.names` are supplied to `yourcast`.

Plots are titled with the `dvlabel` and the `G.names` dataframe if it was supplied to `yourcast` in the `dataobj`. For example if `dvlabel="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.

Axis labels can be changed with the appropriate `xlab`, `ylab`, and `zlab` arguments.

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, `plot.yourcast` handles `agetime` plots differently than the other families by opening 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. This convenience makes it impossible to place multiple plots on the same device, however. Users seeking to create 1x2, 2x2, 3x2, etc., plot layouts for the purposes of comparison are advised to use the separate `age` and `time` plot families and print each plot indivudally to the device.

Below is some example code for a 2x2 plot layout with, in effect, two 'agetime' plots:

```
par(mfrow=c(2,2))
plot(y.out1,family="age")
plot(y.out1,family="time")
plot(y.out2,family="age")
plot(y.out2,family="time")
```

Value

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

Author(s)

Jon Bischof ⟨jbischof@fas.harvard.edu⟩

References

http://gking.harvard.edu/yourcast

See Also

yourcast function and documentation (help(yourcast))

4.4 `histograph`: Histograms for model ebayes

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`

References