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

Federico Girosi[2] Gary King[3]

March 23, 2006

[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 Girosi and King (2005):[GirKin05]

> Federico Girosi and Gary King. 2005. *Demographic Forecasting.* Book manuscript in progress available from `http://gking.harvard.edu/`.

Please read at least Chapter 1 from this manuscript 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 package sma, which is 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 Reference

3.1 Function 'yourcast'

sec:refyour

3.1.1 Usage

```
y.out <- yourcast(formula, model=OLS,data, sample.frame=c(1950,2000,2001,2020),
          zero.mean=FALSE, Ha.sigma=0.3,Ha.sigma.sd=0.1,Ha.deriv=c(0,0,1),
          Ha.age.weight=0,Ha.time.weight=0,
          Ht.sigma=0.3, Ht.sigma.sd=0.1,Ht.deriv=c(0,0,1),Ht.age.weight=0,
          Ht.time.weight=0,
          Hat.sigma=0.2,Hat.sigma.sd=0.1,Hat.a.deriv=c(0,0,1),Hat.t.deriv=c(0,0,1),
          Hat.age.weight=0,Hat.time.weight=0,
          Hct.sigma=0.3,Hct.sigma.sd=0.1,
          Hct.t.deriv=1, Hct.time.weight=0,
          userfile=NULL,nsample=500,svtol=10e-10,solve.tol=1e-10,standardize=TRUE,
          LI.sigma.mean=0.2,LI.sigma.mean=0.1,model.frame=FALSE)
```

3.1.2 Inputs

Model and Data

formula A standard R formula of the form `y ~ x1 + x2`, 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)~ x1 + x2`, 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.)

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.)

dataobj Four types of inputs are allowed. If (1) `dataobj` is an object (in working memory) or (2) a string with the name of a file on disk, the object must contain a list with the following items (the first two of which are required):

index.code A string indicating how the index variable is coded in the input data. Between 0 and 4 of each of the following characters are used in order: `-` to ignore a character, `G` for the geographic index (such as country), `A` for a grouped continuous variable like an age group, and `T` for a time period. For example, `--GGGGAATTTT` means parse 920004172005 by ignoring 92, using 0004 as the country code, 17 as the age group, and 2005 as the year. Default: –GGGGAATTTT.

data A list of dataframes, one for each cross-sectional unit, with names corresponding to the geographic areas and age group cross-sectional indices `GGGGAA`, and rows labeled according to time periods `TTTT` (using notation in `index.code`, so one item in the list might be labeled 432101 for geographic area 4321 and age group 01). Columns must include at least one variable common to all dataframes, to be used as a dependent variable, and a possibly different set of explanatory variables in each cross-sectional unit.

G.names, A.names, T.names Optional two-column dataframes that list all valid numerical codes (in the first column, labeled `code`) and corresponding alphanumeric names (optionally in the second column, labeled `name`) for the indices corresponding to the geographic areas in `G.names`, age groups `A.names`, and time periods `T.names`. The alphanumeric names are most commonly only used for geographic areas, since numerical values for age groups and time periods are usually meaingful on their own.

proximity Includes codes to construct the symmetric matrix (geographic region by geographic region) of proximity scores for geographic smoothing used by methods MAP and Bayes. 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). This symmetric matrix is constructed by `yourcast()` from the `proximity` object in `dataobj`. Each row of `proximity` has three entries, consisting of geographic codes for two countries (corresponding to the row and column, and column and row, of the symmetric matrix to be built) and a score indicating how proximate or similar are the two geographic regions; thus, each row represents one element of the symmetric matrix. For convenience, geographic regions that are unrelated (and have zero entries in the symmetric matrix) may be omitted from `proximity`. In addition, `proximity` may include rows corresponding to geographic regions not in the present analysis.

(3) If `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, which an be useful if `savetmp` was set to TRUE on the last run, in which case the value of `dataobj` will come from the saved file.

sample.frame A 4 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(1950,2000,2001,2020). (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 will make these comparisons automatically if the out-of-sample data are included.)

Smoothing over Age Groups

zero.mean A boolean or string to set $\bar{\mu}$. 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). If a string, this is the name of a file that contains the mean profile to be used in constructing a non-zero mean prior. Default: FALSE.

Ha.sigma A scalar which sets σ_a, the prior standard deviation of $E(Y)$, which indicates how to smooth $E(Y)$ over age groups, or NA to not smooth in this way. The parameter σ_a is the expected prior standard deviation of $E(Y)$ for an age group (varying 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.) 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 $\mathbb{n}$, 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 $a^{\text{Ha.age.weight}}$; 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 $t^{\text{Ha.time.weight}}$; if the argument is a vector of length T, the tth element is the weight of time period t. Default: 0.

Smoothing over Time

Ht.sigma A scalar which sets σ_t, the prior standard deviation of $E(Y)$, which indicates how to smooth $E(Y)$ over time periods, or NA to not smooth in this way. The parameter σ_t is the expected prior standard deviation of $E(Y)$ for a time period (varying 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.) Default: 0.3.

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 $\mathbb{n}$, 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 $a^{\text{Ht.age.weight}}$; 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 $t^{\text{Ht.time.weight}}$; if the argument is a vector of length T, the tth element is the weight of time period t. Default: 0.

Smoothing over Age and Time

Hat.sigma A scalar which sets σ_{at}, the prior standard deviation of $E(Y)$, which indicates how to smooth the time trend in $E(Y)$ over age groups, or NA to not smooth in this way. The parameter σ_{at} is the expected prior standard deviation of $E(Y)$ for a time trend over age groups (varying over geographic areas, 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.) 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 $a^{\text{Ht.age.weight}}$; 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 $t^{\text{Ht.time.weight}}$; if the argument is a vector of length T, the tth element is the weight of time period t. Default: 0.

Smoothing Levels or Trends over Geographic Areas If this option is used, a proximity matrix, called `proximity`, to be used in smoothing over the geographic areas, must be stored in `dataobj`.

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 is 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 $t^{\text{Hct.time.weight}}$; if the argument is a vector of length T, the tth element is the weight of time period t. Default: 0.

Miscellaneous

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. Default: NULL, 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 its normally best to specify each option in either the userfile or the command line but not both.

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

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

solve.tol A real number smaller than 1 that is used in the argument of the R-function "solve" to invert matrices (see description for `tol`). Default is $1.e-10$.

standardize A 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.

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 is 0.2.

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

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

savetmp If TRUE, yourcast saves a file in the default directory (called `yourcast.savetmp`) with preliminary calculations. If the value of `formula` or `dataobj` are missing when yourcast is called, yourcast will get their values from this file, if it exists. This saves a minute or so of compute 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.

3.1.3 Value

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.

sterror a list with the same cross-sectional elements as the input `data`, elements of which are the estimated standard errors of the coefficients.

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).

References

GirKin05 Girosi, Federico and Gary King. 2005. "Demographic Forecasting." http://gking.harvard.edu/files/abs/smooth-abs.shtml.

4 Example

One data object is included with YourCast and is available by running `dta<-dataobj()`. Here is an example run, using the least squares regression model (the default):

```
library(YourCast)
dta <- dataobj()     # loads in data object included with YourCast
names(dta)
y.out <- yourcast(log(allc/population) ~ gdp^2 + tobacco + time, dataobj=dta)
names(y.out)         # see available output values, accessible as y.out$...
```

Continuing with the example, we can run the same specification and data with the MAP model as

```
y.out <- yourcast(log(allc/population) ~ gdp^2 + tobacco + time, dataobj=dta,
+                 model="MAP")
```

or, if we have run one of the other models first, as above, we could instead use the simpler form by dropping the formula and data object:

```
y.out2 <- yourcast(model="map")
```

This simpler form also uses some processing performed during the previous run and so is faster as well.

We can also change some other modeling options:

```
y.out2 <- yourcast(model="map", Hct.sigma=NA, Hat.sigma=NA)
```