Forecasting Macroeconomic Indicators using Google Trend Data

Arrich, Maximilian — 15-605-017

Andrea, Longoni — Exchange Student

December 20, 2018

Contents

1 Executive Summary

Macroeconomics is a branch of economics with focus on the different aspects of the economy and how they are interrelated to each other. Some major macroeconomic indicators are among others unemployment, GDP, inflation and consumption . Economic indicators and their interpretation by knowledgeable sources are important in predicting how the state of the economy will take shape in the coming years. Furthermore, stock, bond, real estate, and other markets, are all affected in some ways by how leading economic indicators are performing. In this short working paper, we give a brief overview of how Google trend data can be used to infer about the development of such main macroeconomic aggregates.

1.1 Economic Indicators

Our project is focusing, in its first stage, on unemployment rate and consumption. The former is important because rising unemployment is seen as a sign of a weak economy, with slow growth and little spending. That might cause action by the Federal Reserve, which has a mandate to help reduce unemployment, like increasing the nation's money supply, or performing Quantitative Easing, a measure that has become more known in the weak of the last financial crisis. The latter is the single most important driving force of a lot of economies (e.g. USA). Keynesian economic theory says that the government should stimulate spending to end a recession. It reveals how much households spend on immediate consumption versus saving for the future. Higher consumption levels translate into greater GDP growth in the short term. On the other hand, a higher savings rate is good for long-term economic health. That's because banks use savings to fund loans for mortgages and business investments.

1.2 Lagging problem

Most, if not all, macroeconomic indicators are released by national statistic bureaus, in Switzerland for instance the SECO, with a certain lag. This lag may most commonly be due to slowly working bureaucratic procedures or simply the measuring process itself. Let $(\Omega, \mathcal{G}, \mathbb{P})$ describe the state space and let $\mathbb{F}_i = (\mathcal{F}_{i,t})_{t \geq t_o}$ describe the accumulation of information over time. Moreover, we partition $[t_0, \infty)$ into time intervals $(t_k, t_{k+1}], k \in \mathbb{N}$ and denote by $\mathcal{F}_T$ the filtration of the state space today. Denote by $\mathcal{G}_t$ the economic statistic released by some institution. Due to the signaling lag, we formally have that

$$\mathcal{G}_T \subset \mathcal{F}_{T-L} \subset \mathcal{F}_T$$

$L \in \mathbb{N}$ in this setting is denoting the lag. It is now easily visible that not all information that is available in the latest filtration of the state space is represented by the economic statistic. Depending on the lag L, there might be a more or less severe delay. Here it is important to mention that this information is already available, we therefore are not tackling a problem of extrapolation, but one of measuring the present state of the world. Illustrating by the example of the US, the most commonly used statistic on private consumption is the Personal Consumption Expenditures report which is based on the Consumer Expenditure Survey, released in August each year by the Bureau of Labor Statistics, including the data from the past year, thus incurring a log of up to 20 months. The second variable, unemployment rate, is released monthly with a lag of one month and revised consecutively. The BLS measures unemployment through monthly household surveys, called the Current Population Survey. It is easy to deduce that both procedures yield lagging indicators with a lag of at least one month (unemployment) and up to twenty month (consumption). Thus, information about unemployment and consumption that is already an element of the present filtration of the state space is not represented by these statistics.

1.3 Solution

Our project aims to solve the above mentioned lagging problem extracting information from Google trend queries and applying inference. Google queries are indeed continuously updated and thus are not subject to any lag. Using these as explanatory variables allows the user to produce an indicator without any lag. In order to do so, we use Google data as a proxy for the intentions of the most elementary subject of the economy: the individual. We believe that doing so in order to predict macro-indicator is a good strategy because of the extreme popularity of Google among all levels of society. For instance the first thing one might do when getting notice of dismissal is going on the web looking for information about subsidies or job vacancies. Note that this happens as soon as the individual is informed about the dismissal, which in most cases is three month before the official date of unemployment. This shows one more advantage of working with intentional data, that it might be used to conclude on future events before they have happened. This justifies the existence of a certain lead in the Google data with respect to the target macro economical variable, making possible a forecast exceeding the present state space horizon $\mathcal{F}_T$. The knowledge of those indicators (without any lag) can provide an advantage on the market (exactly like private information) to investors and facilitate the realization of positive alphas. Moreover, they can also be used by central banks and governments which would be able to adopt corrective strategies earlier, making them more efficient.

2 Code Structure

The code comes in the form of a shiny web application that consists of three main files which will be elaborated upon below. The ui.R file deals with the HMI of the application, passing information from the user to the logic defined in server.R and displaying information to the user. The third file, resources.R is a storage for self defined functions and is used to reduce the code of the other two files in order to improve readability.

2.1 resources.R

This is an auxiliary file holding the main functions that will be called by the two main files, especially by server.R.

2.2 ui.R

In order to allow for reactivity, some kind of front end user interface has to be established. Shiny gives control over this interface in the ui.R file that will fully define the HMI. The interface of this application is split into two main areas. One taking information from the user, the other one displaying the outputs calculated by server.R. The following inputs can be specified:

- **target:** Lets the user choose either "Unemployment" or "Private consumption" as the target macroeconomic series. The argument will be passed as a named numeric object.
- **keyword:** Takes up to five Google keywords that will be used as predictors to the desired target variable. The argument will be passed as a single string that needs to be split before processing.
- **region:** This input list the user choose from a predefined list of countries to specify the analysis to. At the moment only one country can be analyzed at once. The argument is passed as a named numeric object.
- **end_difflog:** Binary variable indicating if the transformation of the endogenous variable (the Eurostat economic series) to differenced logs is desired. The argument is passed as boolean.
- **end_difflog:** Binary variable indicating if the transformation of the exogenous variables (the Google trend series) to differenced logs is desired. The argument is passed as boolean.
- **go:** A button triggering the start of the processing. This button is deactivated while computation is performed to avoid repeated evaluation.

2.3 server.R

This file contains the server logic and does the main part of computations. It can be seen as the core code organized in two parts. The reactives that take inputs form the user and pass them on to functions stored in the resources.R file or do some other kind of parsing, described in detail below. The second, smaller, part of this code is rendering the results into an outputable format for the ui.R file to source. The so called "reactives" are blocks of code that get evaluated after the user is passing new information to the UI. The server logic consists of three such reactive expressions which will be explained below.

2.3.1 google_query

This reactive takes the following inputs directly from the user:

- **keyword:** Up to five Google keywords that will be used as predictors to the desired target variable

- **region:** The region to which the query will be specified.

First, the API is queried for the most related words to the first of the keywords. If less than five words were entered, the rest will be filled with the most related word according to the Google API query. This way, we obtain a lost of five words for which the interest over time is retrieved from the API. This series is then formatted into one multivariate ts object and aggregated additive to monthly data by to fit the Eurostat data we will download in the next step. This very object is the main output from this reactive.

2.3.2 eurostat_query

This reactive takes the following inputs directly from the user:

- **target:** The economic target series. For now, only two options "Unemployment" and "Private Consumption" are available.

- **region:** The region to which the query will be specified. This is the same as for the Google API.

Similarly to the google_query reactive, this chunk is downloading the desired economic target series form the eurostat API. After some formatting, this is the direct output of this reactive.

2.3.3 analysis

This is the core part of the code. Its inputs are the the directly forwarded outputs of the two query reactives. If needed, the two ts objects are harmonized before further processing. Before any analysis is done, the series are logged and the differences are taken. Next, a principal component analysis is performed and all five components are kept as final predictors. Before fitting the final OLS model, the five PCs are lagged for a minimum of zero to a maximum of twelve quarters. The number of lags depends on the user of the user. After lagging obtain a multivariate ts object M by binding together the separate univariate series $I_{Lag\,n}$:

$$\boldsymbol{M} = (\boldsymbol{I}_{Lag1}|\boldsymbol{I}_{Lag2}|\boldsymbol{I}_{Lag3}|\boldsymbol{I}_{Lag4})$$

On this M, a stepwise selection algorithm according to the Akaike information criterion (Akaike, 1973) is then performed. The criterion is defined as follows:

$$AIC = 2k - 2log(\hat{\boldsymbol{L}})$$

Where k is the number of parameters in the model and $(\hat{\boldsymbol{L}})$ is the maximum value of the likelihood function of the model. From this definition we can see that in dependence of the term multiplying k, the selection process is more or less penalizing new variables in the model. In a practical attempt to control the model size, the user could be given the option to to manipulate the selection process. The fitting process is structured in the following way: The starting point is a model only including the dependent macroeconomic varibale and the constant. The AIC of this model is computed and compared to the AIC of all models that can be obtained by adding one of the series contained in M. If a variable is found that yields a lower AIC after being included to the current model, it is added. After a new variable is added to the model, the AIC of all models that can be obtained by dropping any one variable out of the model is calculated. If there is a model with a lower AIC, that one is kept, dropping the variable that was excluded to produce it. This process is repeated until no model with a lower AIC can be produced by dropping or adding a single variable. The final model is then used to produce a forecast of the economic target variable. The desired lead is produced by only including variables with a high lag and by the superior availability of the Google data in comparison to the Eurostat statistics. If the target series was logged and differences were taken, the forecast is back transformed to the desired output in levels by the function multiply_recursive stored in the resources.R file. The object that is passed on by this reactive is a list containing the final predictions as well as the model that was used to produce it.

2.3.4 Outputs

The rest of the server.R file is processing the data forwarded by the three reactives and forwards the following plots to the ui.R file:

- **google_plot:** Plotting the five trend series retrieved from Google.
- **eurostat_plot:** Plotting the target series.
- **fitted_plot:** Plotting the forecast along with the original target series.

3 User Instructions

This section contains brief comments on the running and handling of the application.

If you do not have R installed on your machine, please install it. Please note that the following packages will automatically be installed on your machine if they are not already installed: "shiny","shinydashboard","gtrendsR","magrittr","reshape2","lubridate","zoo","tseries", "eurostat","dyn","ggplot2","ggfortify","shinyBS","countrycode","pander".

If you don't want this to happen please don't run the application. Please note that the scripts of the source directory WebTS may not be separated. For the application to work, an internet connection is required.

3.1 OS X / UNIX

If run on OSX or Linux, you can use the runApp.command file enclosed in the main directory of the application to start the app. Before using the shortcut the first time, you need to authorize it to access your shell. Please open a new terminal window, navigate to the directory you saved these files into and type:

$$chmod + x \quad runApp.command$$

Thereafter you should be able to run the program by clicking the .command file.

3.2 OS X / UNIX / Windows

On Windows, please run the app directly from R. To do so, start R and type:

$$shiny :: runApp('your\_dir', launch.browser = T)$$

Where "your_dir" has to be replaced by the path to the application directory "WebTS". Alternatively, you can open server.R or ui.R in RStudio and click the "run app" button on the top right corner.

References

Akaike, H. (1973). Information theory and an extensión of the maximum likelihood principle. *International symposium on information theory*, (1973):267–281.