Journal of Global Economic Analysis, Volume 1 (2016), No. 1, pp. 251-283.

Using R to extract data from the World Bank's World Development Indicators

BY DOMINIQUE VAN DER MENSBRUGGHE[a]

This article describes three applications which use a new R-based interface that facilitates extraction of data from the World Bank's World Development Indicators (WDI) database. WDI contains over 1300 time series of socio-economic indicators and provides important inputs for the Global Trade Analysis Project (GTAP) Data Base and, as well, can assist in the development of socio-economic scenarios. The first application is a stand-alone R application that extracts per capita historical GDP for all countries and summarizes growth episodes across regions in a box plot. The two other applications highlight the increasing use of interoperability across software packages to take advantage of the specific features of each. The first couples the R-based WDI extraction with a 'Beamer'-type LaTeX presentation. This is a potentially powerful combination for a quick (and programmatic) turnaround from data to presentation. With little effort, the application can be extended to generate a large number of slides. The second example of interoperability is centered on a General Algebraic Modeling System (GAMS) program that calls R as a sub-process to extract the data for further processing in GAMS. The R-script developed for this last application could be readily coupled with other software packages.

JEL codes: C10, C82, C88

Keywords: Economic database; Software; R; Latex; GAMS

1. Introduction

With a strong push from then-President Robert Zoellick, the World Bank made publicly and freely available its major database known as the World Development Indicators (WDI). WDI contains a large host of socio-economic indicators, from the widely used such as population and GDP, to the more esoteric such as the percent of households that consume iodized salt. At last count WDI contained over 1300 indicators covering most of the countries of the world, and with the earliest indicators starting in 1960. Providing easy access to these indicators significantly cuts the cost of undertaking detailed quantitative analysis. Assessing historical trends—with either time series analysis or structural econometrics—is a *sina qua non* for

[a] The Center for Global Trade Analysis, Purdue University, 403 West State Street, West Lafayette, IN 47906 (e-mail: vandermd@purdue.edu).

shaping long-term scenarios of growth, development and sustainability, such as frequently done in the context of climate change analysis, for example.

There are two methods for accessing WDI. The first is to directly use the World Bank's web-based graphical user interface (GUI).[1] The user points and clicks with a mouse to select countries, indicators and years. The selected data can then be downloaded in alternative formats, for example Excel. This access method is fine for the occasional use of WDI but quickly becomes tedious for large selections and/or when access is routine. The second method uses a so-called Application Programming Interface (API) that can be embedded in computer code to programmatically extract data from WDI. The API requires inputs—selected countries, indicators and years—and returns the desired data 'cube'. The API has been integrated into an R package[2] that simplifies the extraction process and allows for the downloaded data to be directly treated in R or to be saved as a datafile for use with another programming environment.

Each new version of the Global Trade Analysis Project (GTAP)[3] database extensively uses data in the WDI database for shaping the macroeconomic framework—this includes indicators such as GDP and its components (private consumption, government and investment expenditures, etc.), population, the stock of capital and additional accounts required for satellite data sets and/or for historical purposes. For example, WDI includes the share of agriculture in GDP, the rate of urbanization, the share of the working age population in total population, energy use and CO_2 emissions, further decomposition of the balance of payments such as remittances, transfers and aid to governments, and poverty and distributional statistics.

This article provides a broad description of the type of data series available in WDI and how to use the new R-based extraction package to extract the data. A first application extracts real per capita growth for all countries since 1960 and plots long-run growth episodes in a box plot where countries are classified using the World Bank's regional definitions. This application illustrates the basic functionality of the extraction routine and how to combine the extraction with statistical analysis in R. It also shows how to use R's plotting features to save figures that can

[1] The GUI-interface to WDI is available at the following link: http://databank.worldbank.org/data/reports.aspx?source=world-development-indicators.

[2] R is open source software designed specifically for statistical analysis but with significant additional functionality for data and graphical analysis. More information, including how to download the software, is available at https://www.r-project.org/.

[3] The GTAP Data Base is a database of global economic transactions that integrates and harmonizes national input-output (IO) tables with a consistent database of bilateral trade flows. The latest release (Version 9) includes IO tables for 120 countries (with all remaining countries aggregated in 20 geographical regions) and economic activity is divided into 57 sectors. The database is described in greater detail in Aguiar et al. in this issue. Additional information on GTAP is available at https://www.gtap.agecon.purdue.edu/.

be imported into documents. To highlight the increasingly wide use of interoperability across different types of software packages, two additional applications are developed. The first application links R with LaTeX's 'Beamer' document class—a PDF slide presentation.[4] It illustrates how to quickly generate a presentation from the extracted WDI data, bypassing the at-times tedious steps of using Excel and/or Powerpoint packages. The second illustration embeds the R extraction package in a General Algebraic Modeling Systems (GAMS)[5] program that lets the user select the countries, indicators and range of years to be extracted. The selection is then passed on to R and the data returned to GAMS for additional processing.

2. World Development Indicators

The World Development Indicators database is one of the most significant international databases and is developed and maintained at the World Bank. The indicators stored in the database are largely socio-economic indicators at a national level. The most widely used are probably GDP and its components, and population. The indicators are typically annual time series many of which go back to 1960. A large number of the indicators are developed directly at the World Bank (at times in coordination with other international agencies such as the International Monetary Fund (IMF) and the United Nations (UN) Population Division), but a significant number are integrated from external databases. WDI includes, for example, energy data from the International Energy Agency (IEA), labor data from the International Labor Organization (ILO) and agricultural data from the United Nations Food and Agriculture Organization (FAO). Depending on specific needs, WDI can be used for one-stop shopping rather than having to refer to the more specialized databases. Moreover, with open access to WDI complemented with its API, data extraction can be done at extremely low cost.

The WDI data can be used for a variety of purposes—illustrating historical trends, parameter estimation, calibration of trends, assessing structural changes over time—for example population cohorts, the share of agriculture in GDP and employment, and a number of indicators related the so-called Millennium Development Goals (MDGs).[6]

---

[4] LaTeX is open source software designed for the preparation of documents. Traditionally it has been used to convert LaTeX input files into PDF files such as papers, reports, articles and books. More recent extensions include the preparation of slide presentations and web-based documents. LaTeX is freely available for a number of computing platforms and more information is available at https://latex-project.org/.

[5] GAMS is widely used in economics and other disciplines for mathematical programming. It provides an intuitive language for the development of mathematical models for the purpose of optimization and simulation. More information on GAMS, including extensive documentation, is available at www.gams.com. GAMS is freely available, though its use is limited without acquiring a license.

[6] More recently called the Sustainable Development Goals (SDGs).

Extraction from WDI requires selecting countries and/or regions, indicators, and the years to extract. The GUI provides the easiest access as the extraction is based on point and click and the user does not need to know the underlying arcane nomenclature for the countries/regions and the indicators. However, the API depends on 'codes' and these must be provided with exactitude to extract the desired data. Various systems have been devised for country codes. Perhaps the most widely used system is the ISO-3 coding.[7] For example, the ISO-3 code for Switzerland is `CHE`.[8] The World Bank country codes hone closely to the ISO-3 codes, but it does not necessarily update codes when they change.[9] The R WDI extraction package is based on a different set of codes known as ISO-2 that rely on 2-letter country codes.[10]

The indicator codes[11] have been developed by the World Bank and are typically short descriptions of the underlying indicator—divided into as many as five fields. The first field reflects the topic. For example, 'SP' refers to social indicators and population. The indicator for total population is `SP.POP.TOTL`. The GDP indicators are in the broad topic of national income (NY). The second field is the type of national income indicator—for example GDP or GNP. The third field will indicate the type of income indicator, for example per capita (PCAP) or total at market price (MKTP). The fourth indicator will most often indicate the pricing regime—current local currency, dollar, purchasing power parity (PPP) dollars, or at constant prices at some base year. For example, GDP at constant dollars at market exchange rates is given by `NY.GDP.MKTP.KD`.[12]

3. A practical illustration of the WDI extraction routine in R

This section describes an R script that plots long-run growth episodes since 1960 across all countries organized into the World Bank's regional classifications.

Figure 1 represents a box plot of all 10-year country growth episodes from the WDI database from 1960 through 2014 with the growth episodes classified according to the World Bank's classification: East Asia & Pacific (EAP), South Asia (SAS), Europe & Central Asia (ECA), Middle East & North Africa (MNA), Sub-Saharan Africa (SSA) and Latin America & Caribbean (LAC). The six geographic regions only include developing countries using the World Bank's definition. All of the

[7] See https://en.wikipedia.org/wiki/ISO_3166-1_alpha-3.
[8] Coming from the Latin *Confoederatio Helvetica* or *Confédération Helvétique* in French.
[9] For example, the World Bank continues to use the old `ZAR` code for the Democratic Republic of the Congo instead of the ISO-3 code of `COD`. Similarly for Romania, the ISO-3 code is `ROU`, and not the World Bank's `ROM`.
[10] See https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2. Note that these codes are widely used for Internet country suffixes.
[11] Also known as Catalog of Economic Time Series or CETS codes.
[12] A mostly complete list of CETS and country codes is available from the author.

developed economies are included in HIC, i.e. the high-income country classification. China has been assigned to its own region (CHN) and excluded from the EAP region. The maximum number of 10-year growth episodes for any single country over the period 1960-2014 is 46.[13] The red line represents the median for the given region. China has the highest, with a median of over 7.5 percent. The HIC region has a median of around 2.5 percent but with a significant number of outliers on both the low- and high-sides.[14] Note that the HIC region now includes a number of European transition economies that experienced a sharp decline after 1990. Not surprisingly, Sub-Saharan Africa shows the lowest median, at around 1 percent, with an inter-quartile range that includes the origin.

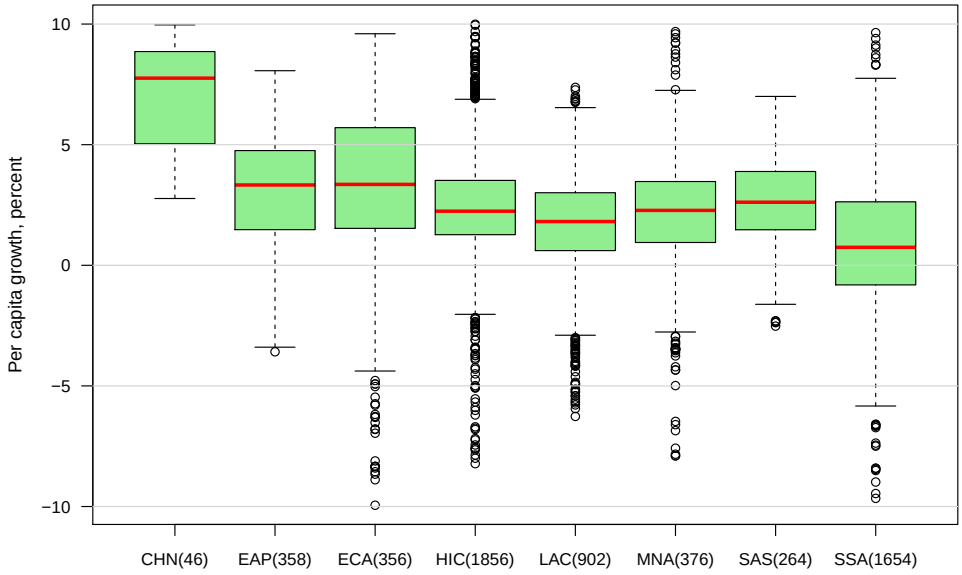

Figure 1. Growth periods across regions (count in parenthesis).

Source: Author calculations.

These numbers suggest some possibilities for projecting forward. The inter-quartile ranges could be used to bound low- and high-growth scenarios—perhaps attached to specific storylines. An alternative approach is to use the underlying probability distribution to bootstrap growth episodes for the future.

The box plot is created in R using a script that uses the WDI package to directly

[13] The historic time span, the length of a growth episode and the outlier cutoff are all user-defined options. Herein the time span has been set to 1960-2014, the length of a growth episode is 10 years and the outlier cutoff has been set to 10 percent.
[14] Outliers are represented by hollow circles in the chart.

Table 1. User options.

Name	Description
`wDir`	Path for the working directory. This directory will contain both the input file and the box plot figure. To use the default directory, use `'./'`.
`fName`	Name of the input file that contains the country codes to extract and their associated region.
`bpName`	Name of the output file containing the box plot figure.
`fType`	The file type for the box plot. The choices are `PDF`, `JPG` and `WMF` (Windows Metafile format).
`begYear`	First year of extraction (the earliest is 1960).
`endYear`	The final year of extraction.
`gap`	The length of a growth episode.
`threshold`	Growth episodes with a growth rate (in absolute terms) greater than the threshold are dropped. This affects the span of the box plot. If the threshold is set to a high number, the number of outliers increases.

Source: Author.

extract the relevant data. The script performs the following actions:

- It loads a short file with all of the country codes and the countries classified according to the World Bank's regional definitions as described above. In total there are 8 regions, with an unbalanced number of countries in each.
- For each country in each region, a growth episode is defined over a fixed period—given by a (user-specified) 10-year span. For each ten-year time span, the 'average' growth is calculated using a log-linear regression of real per capita income as a function of time: $\ln(y) = \alpha + \beta t$, where y is per capita GDP (in constant dollars) and t is time. The parameter β is the 'average' estimated growth rate and is saved as the value of a given growth episode. For a given time range, $t1$ through tn, there will be $tn - t1 - span + 2$ growth episodes, where *span* is the span of a growth episode, for example 10 years. A range of 1960 through 2014 would lead to 46 periods of 10 years—assuming no missing data. The maximum number of growth periods for each region is then $N(tn - t1 - span + 2)$, where N is the number of countries in the region.
- The growth episodes are stored in a matrix with R columns, where R is the number of regions and the rows contain the individual growth episodes for all countries in the region for all possible growth episodes.
- The growth episodes are plotted in a box plot that is stored in a graphics file that can subsequently be inserted in a document.

The complete R script is provided in Listing B.1.[15] For those familiar with R, most of the code should be relatively self-explanatory. The user's input is handled in the first part—though line 31. Table 1 explains the key options the user can modify.

The second part initializes the script. It loads the WDI library (see Appendix A on how to install the WDI package). It next reads the country codes to be extracted. The codes are stored in a comma separated CSV file. The first line of the file contains a header line. Each subsequent line contains four fields. The first field is a sequence number (that is ignored). The second and third columns contain respectively the ISO-3 and ISO-2 country codes. And the fourth column contains a region acronym. In our example, there are 8 regions as described above. The lines below show an example of the first few lines of what the CSV file could look like:

```
Seq,ISO3,ISO2,WBREG
44,CHN,CN,CHN
82,FJI,FJ,EAP
107,IDN,ID,EAP
123,KHM,KH,EAP
130,LAO,LA,EAP
163,MMR,MM,EAP
166,MNG,MN,EAP
172,MYS,MY,EAP
192,PHL,PH,EAP
194,PNG,PG,EAP
236,THA,TH,EAP
256,VNM,VN,EAP
5,ALB,AL,ECA
```

Line 47 creates a frequency table with the unique region identifiers. R's `table` function is used to build the frequency using column 4 as the key. The table has two columns. The first contains the unique region identifiers and the second the frequency count of countries in the region. Line 50 then determines the number of regions. Line 54 determines the maximum number of countries in any given region. This information is used to calculate the maximum number of possible growth episodes in a region.

Starting in line 64, there is a lengthy loop over all of the regions. Line 67 extracts all of the ISO-2 codes that have been assigned to the region. Lines 71 and 72 perform the critical extraction from the WDI database. The extraction uses the `WDI` function that comes with the WDI R package. The first argument is the vector containing the ISO-2 codes. The second argument is a list of one or more indicators to extract.[16] The final two arguments are the beginning and end years for the extraction.

Once the data is extracted, the growth episodes are calculated for all countries in the region (line 80). The extracted data is ordered by year in ascending order

[15] All the code listings described in this article are detailed in full in Appendex B.

[16] The example extracts one indicator at a time, though the WDI function is capable of extracting multiple indicators at a time.

Table 2. Summary table for box plot application.

Name	**Description**
Purpose	The R-based WDI extraction routine is used to extract and plot historical per capita GDP.
Input files	
gdppcplot.R	R-script containing instructions for data extraction and saving box plot chart.
wbreg.csv	User configured input file containing countries to extract and their classification by region. (User can change input file name.)
Output file	
gdppcBoxPlot.pdf	Box plot of per capita income growth episodes. (User can change output file name.)
Instructions	
	Copy the two input files to a user-specified directory.
	Change the user inputs in the file `gdppcplot.R`—particularly the directory information.
	Run the R-script. In R, type `source("userDirectory/gdppcPlot.R")` replacing *userDirectory* with the correct path to the directory containing the R script (unless it is in a default directory in which case it can be dropped). The default directory can be set by using the command `setwd("userDirectory")`.
Software requirements	
R	R statistical software.

Source: Author.

and per capita GDP is calculated (line 91). Growth rates are then calculated for each possible growth episode (line 99). Working vectors y and x are filled with the log of per capita GDP and time, respectively. If the sample is full, i.e. there are no missing data, a linear regression is performed to estimate the growth for the episode (line 115). R's `lm` function is used to fit linear models. In this case, the natural logarithm of per capita GDP (y) is regressed on time (x).[17] The `coef` function is used to extract the second coefficient, i.e. the growth rate, which is converted to a percent growth rate. The coefficient is kept if it is within the user-provided threshold.

At the end of the loop for each region, the vector column of growth rates for the growth episodes is concatenated to the previous (regional) column (line 137), except of course for the first column. The regional acronyms are assigned as the column names. The length of the columns will vary across regions. However, they are padded out by setting the missing cells to `NA`.

The final part of the script creates the box plot and saves it to a file using the desired format.

4. Interoperability Case 1: WDI extraction and a LaTeX 'Beamer' presentation

This section illustrates how to quickly produce a presentation of multiple (line) charts based on an extraction from WDI using R. A single R script is used to select the countries/regions, indicators and time span for using the R-based WDI extraction. The extracted data is then saved as a LaTeX file that will create a 'Beamer'-type presentation that includes a title slide. In generalized form, this script can be used to generate tens, if not hundreds of slides, with a few short key strokes, and could also be converted to produce other types of charts such as bar charts. LaTeX has been around for decades and is a very powerful markup language for producing documents, web pages and presentations. It is available for free on many different platforms and has a large user group. This application is relatively self-contained, i.e. it can be used with very little knowledge of LaTeX, though it requires LaTeX processing to convert the LaTeX input into a PDF presentation. Listing B.2 provides the R-script that will produce the LaTeX based input.[18] Virtually all of the user input is contained in the first part of script (between lines 10 and 90). Table 3 describes the user inputs available for the R-script. The descriptions should be relatively self-explanatory. The presentation will have one slide per extracted indicator (plus the title slide). Each chart will contain one trend line per extracted region, hence it is best to limit the number of countries/regions to extract.[19] Somewhat embedded in the script, the user can also change the theme and theme color of the Beamer

[17] By default, the `lm` function includes an intercept term.
[18] LaTeX files typically have the extension `.tex`.
[19] The script could readily be expanded to chart a much larger set of countries by dividing them in appropriate categories.

presentation.[20]

After the user options, the script initializes a number of variables and loads the WDI library. The preamble for the LaTeX file starts at line 115. The LaTeX package *pfgplots* will be used to generate the line charts. Line 128 ends the preamble, which produces the title slide. Subsequently, the script loops over all chosen indicators and will produce one slide with a line chart for each indicator. The loop starts in line 132. The extraction from WDI occurs first (line 136). Each chart has a preamble that is contained in lines 140-155. The information used in the preamble is the description of the indicator for the slide title and the beginning and end years for the x-axis. The following section loops over all of the regions for which data has been extracted. Each vector is extracted from the cube (line 170), sorted by year (line 173), and then plotted (lines 180-187) after scaling (line 184). The legend is added at line 187. The slide is terminated with lines 192-196.

[20] *CambridgeUS* is the default theme with a default color of *orchid*. Note that the presentation preamble and the charts contain other values that users are free to modify and test.

Table 3. User options for creating the WDI-based Beamer presentation.

Name	Description
`outwd`	Path for the output directory. The LaTeX ready file will be saved to this directory. To use the default directory, use `'./'`.
`fName`	Name of the LaTeX file (with `.tex` extension).
`pTitle`	Title for title page of presentation.
`shortTitle`	Short title of presentation—will be used for footer on slides.
`author`	Author's name.
`nickName`	Abbreviation for author's name—will be used for footer on slides.
`address`	Address—only used on title page. New lines are given by `'\\'`, but must be in pairs because they need to be 'escaped' in R.
`institute`	Short name for institute, will become part of the footer on slides.
`begYear`	First year of extraction (the earliest is 1960).
`endYear`	The final year of extraction.
`regions`	This contains the regions for which the data will be extracted. It has one row and three columns per region. The first two columns are the ISO-2 and ISO-3 codes respectively. The ISO-2 codes are used for the extraction and the ISO-3 codes for labeling the lines (and legend). The user can change the ISO-3 codes as they are not used for the extraction. The third column is a long-name for the region. It is currently ignored.
`indTable`	This contains the indicators that will be extracted from WDI. It has one row and four columns per indicator. The first column has a user-specified short name for the indicator and is currently ignored. The second column contains the World Bank CETS code for the indicator to extract. The third column contains a scaling factor for the extracted data. The fourth column contains a long-name for the indicator. This will be used as a slide title.
`Colors`	A list of colors for the lines on the charts. The list can be as long as desired. A full list of colors is available with the LaTeX *xcolor* package.

Source: Author.

Table 4. Summary table for Beamer application.

Name	Description
Purpose	The R-based WDI extraction routine is used to extract data series for a number of countries/regions. The data is saved in LaTeX format to create a Beamer-style presentation of the extracted series as line charts.
Input file	
extractWDIBeamer.R	R-script containing instructions for data extraction and saving data for input into LaTeX to create a Beamer-style presentation.
Intermediate files	
pptExample.tex	LaTeX file containing instructions for creating Beamer-style presentation of extracted data. (User can change file name.)
Final file	
pptExample.pdf	PDF presentation of extracted data including title slide. (User can change file name.)
Instructions	
	Copy the file `extractWDIBeamer.R` to a user-specified directory.
	Change the user inputs in the file `extractWDIBeamer.R`.
	Run the R-script. In R, type `source("userDirectory/extractWDIBeamer.R")` replacing **userDirectory** with the correct path to the directory containing the R script (unless it is in a default directory in which case it can be dropped). The default directory can be set by using the command `setwd("userDirectory")`.
	The R-script will have created the input file for LaTeX. Open the LaTeX input file in LaTeX and process it (this will be specific to the LaTeX installation). If there are no errors, LaTeX will have created the presentation as a PDF file.
Software requirements	
R	R statistical software.
Latex	LaTeX software for converting LaTeX markup files into a PDF file.

Source: Author.

5. Interoperability Case 2: Combining GAMS with the R-based extraction routine

The second case of interoperability describes how to combine GAMS and R to extract data and read it back into GAMS for additional analysis. Other packages similar to GAMS may have the same tools to link to other packages—for both input and output. The driving software is a GAMS program where the user selects the extraction options—countries/regions, indicators and range of years. The GAMS program saves these options in a file that is subsequently loaded by an R-script that is called from within the GAMS program as a sub-process and that executes the extraction and saves the data. The R sub-process stores the data in a CSV-formatted file. The CSV-formatted file is converted to the GAMS Data eXchange (GDX) format that is read by the GAMS program.[21] The GAMS program continues after reading in the extracted data.

The example GAMS program is provided in listing B.3. The code requires an auxiliary file, called `isocodes.gms`, which contains the set of ISO-2 and ISO-3 codes and their corresponding mapping.

The GAMS code is divided into 5 parts. The first part is a series of global options set by the user. These are described in detail in the preamble of the code. The second part is where the user enters information regarding which countries and indicators to select.[22] The set `c2` contains the list of countries to extract—using ISO-2 codes. It is ignored if the global option `'EXTALL'` is set to *true*. The indicators to extract require three set definitions—the user name of the indicators (that can be the same as the World Bank mnemonics), the World Bank mnemonics (or CETS code) of the indicators and then a mapping between the two sets. These are defined respectively in the sets `'var'`,`'wbind'` and `'mapv'`. The parameter `'scale'` is not needed for the extraction, but is used after the extraction to display the data.

The third part of the code saves the user input in a file that will be used as input to the R script. The format of the output file is a CSV file with each row containing a descriptor and a value.[23] The file name of the input file for the R script is set by the user in the global variable called `'BASENAME'`.

The fourth part of the code uses the `execute` command in GAMS to run R as a sub-task of GAMS using R's *RScript* program. RScript works just like R but takes all of its inputs from an R script file and it never opens the graphical user interface of R. It works more or less silently and when finished, the resulting extracted data

[21] This additional step is necessary because CSV files are read at compile time. However, the extraction is done at execution time. GDX files are read at execution time and thus this extra step solves the sequencing issue.

[22] The years to extract are selected in the global options section.

[23] Note that the file and the R script are most likely not bullet proof and more error checking would be needed. For example, the R script initializes the number of indicators relative to the number of lines it reads—thus blank lines could affect the code.

will be saved in a CSV file. The next step converts the CSV-formatted file into GDX formatted file.[24] A utility, developed at Wageningen University in the Netherlands, is used to convert the R-extracted CSV file into GAMS' GDX format—it is called `CSV_GDX_Tools`.[25]

The fifth part of the code reads the data from the GDX file into the parameter called `wdiData`. Once the data has been read, it can be used for subsequent processing by GAMS. In the listing file, the data is simply scaled and then displayed.

The rest of this section describes the R-script that is coupled with the GAMS program. It is a generic script in the sense that it not dependent on the calling program—in this case GAMS—, i.e. it could be coupled with any other package that has the ability to write the input file for the R-script and read the output CSV file (with or without the optional header line).[26]

The R-script is divided into two parts. The first part is an R function that performs the routine extraction, does some (slight) checking of the extracted data and saves the data in a CSV file. The function uses two imported functions from the WDI package—the `WDI` extraction function (already described above) and the `countrycode` function that is used to convert the ISO-2 codes to ISO-3 codes.[27]

Listing B.4 contains the R code defining the function 'WDI.Retrieve'. The first executable line of the function performs the extraction using the `WDI` extraction function. The function 'WDI.Retrieve' is written to retrieve only one indicator at a time. The purpose of this is so that the resulting CSV file is essentially 'vectorized' with each extracted indicator appended to the previous one. The 'country' argument in the function call contains the list of countries to extract (using ISO-2) codes or alternatively the text `all` in which cases data for all of the countries is extracted. The 'indicator' argument contains a single indicator code using the World Bank's mnemonics. The 'start' and 'end' arguments represent the first and final years to extract.

The rest of the 'WDI.Retrieve' function essentially performs a series of adjustments to the extracted cube. The first adjustment appends the ISO-3 codes to the

[24] GAMS users could bypass directly the CSV formatted-file and instead use the *gdxrrw* package that has been developed to read and write GDX files in R. We have kept this version as the CSV-formatted file can be useful in many other contexts such as Stata, Excel, etc. The *gdxrrw* package for R can by downloaded from https://support.gams.com/gdxrrw:interfacing_gams_and_r.

[25] Note that this data conversion utility also requires the file 'forbiddenwords.txt' that should be downloaded with the utility. Both files should be stored in a folder that is accessible (through the system path). Since it is meant to be coupled with GAMS, the GAMS folder would be a logical place to put the utility. The software is available for download at http://www3.lei.wur.nl/gamstools/index.htm.

[26] The author has coupled this script with a VBA macro in Excel that facilitates the choice of indicators and reads the subsequent CSV file into an Excel pivot table.

[27] The `countrycode` function comes in its own package and must be installed with the WDI package. Additional details are available in Section A.

Table 5. Summary table for GAMS application.

Name	Description
Purpose	The GAMS based program prepares a file with WDI selection options that are passed to R as a sub-process. The R-extracted data is saved as a CSV file that is converted to the GDX format. The latter is read into GAMS for further processing.
Input files	
gdpr.gms	The GAMS program that calls for the R extraction and then processes the extracted data.
isocodes.gms	A GAMS file that contains set definitions for the ISO-2 and ISO-3 country codes and their mapping. Also contains a mapping of the ISO-3 codes to the GTAP regions.
extractWDI.R	R script that reads the options for the extraction, extracts each indicator and saves the extracted data to a CSV file.
Intermediate files	
gdpr.opt	File containing the WDI selections for the R-based extraction.
gdpr.csv	File containing the data extracted by R.
gdpr.gref	File created by the CSV to GDX conversion tool. It is ignored.
Final file	
gdpr.gdx	The extracted data in GDX format
Instructions	
	Copy the files `gdpr.gms`, `isocodes.gms` and `extractWDI.R` to a user-specified directory.
	Change the user inputs in the file `gdpr.gms`.
	Run the GAMS program `gdpr.gms`. If using the GAMS Integrated Development Environment (IDE), open the file gdpr.gms and run it.
	If the program runs without error, the extracted data will be printed in the GAMS listing file and is saved in a GDX file.
Software requirements	
GAMS	GAMS mathematical programming software.
R	R statistical software.
CSV_GDX_Tools	Data conversion package that converts CSV files to GDX format (and vice versa).

Source: Author.

data cube using the `countrycode` translation function. Not all ISO-2 codes have an ISO-3 code. The second adjustment allows for the replacement of ISO-3 codes—existing or not. It is a two-step procedure. In the first step, the column indices of the ISO codes are located using the 'grep' (or search function). The second step replaces the relevant ISO-3 codes for the selected ISO-2 codes. In the listing example two ISO-3 codes are replaced (or created)—for Jersey and Guernsey Islands and for Kosovo.[28] The third adjustment is to delete all rows with an unknown ISO-3 code. The fourth adjustment appends the user-specified name of the indicator to each row of the cube. The user-specified name is typically a shorter name of the indicator compared to the World Bank mnemonic (or CETS code). For example `pop` might be used for total population rather than `SP.POP.TOTL`. The final adjustment re-arranges selected columns of the cubes. The ISO-2 column is dropped and the country name column is dropped if the `ifName` flag is set to *false*. Typically the country name can be retrieved from a lookup table given the ISO-3 code and its output in the data cube is redundant and space consuming. The final step is to save the data. The output is a CSV-formatted file. Each row contains the name of the indicator, the ISO-3 country code, the year and the value of the indicator. The country name is an optional field (see above).

The 'WDI.Retrieve' function is relatively generic and can be used multiple times in a single R script. In the scripts below, it is called for each individual indicator.[29]

The R code in listing B.5 is used to read the input file prepared by GAMS and to extract the selected data. It will call the 'wdi.Retrieve' function described above.

The first set of executable lines load the *WDI* and *countrycode* packages from the library. The next section of the code opens the GAMS-prepared file with the options and reads and parses each option. These include the name of the output file, the first and final years, a true/false flag for the header, a true/false flag to output or not the country name in the CSV file and a comma delimited string that contains the countries to be extracted (or else 'all' if all countries are to be extracted). The parsing functions use a few tricks in R. The 'grep' function is used to find the name of the relevant option in the file, i.e. the options can be in any order in the file (except for the indicators that must be the final lines in the file). The 'as.numeric' and 'as.logical' functions are used to convert the options into numerical or boolean values. The 'toupper' function is used to ensure that the ISO-2 country codes are in upper case (a requirement of the *WDI* routine), but it checks first to see if `'all'` countries have been selected. The final part of the parsing reads the indicators that are the last lines in the file. The indicators are loaded into a matrix called

[28] This adjustment is typically only relevant when `all` countries are selected. There is also no indication that the World Bank will continue to maintain these 2-letter ISO codes or add new ones not yet officially recognized.

[29] There might be some cost in speed as the `WDI` function will be called for each separate indicator rather than doing one call with all of the indicators. However, it is easy to create a CSV cube using this methodology.

`'indTable'`.

The next section of the R script initializes the CSV file—only in the case a header row has been requested. The final part of the R script loops over each of the indicators and uses the 'wdi.Retrieve' function to retrieve each indicator, one at a time, and save them in the CSV file.

Acknowledgements

The author would like to thank Angel Aguiar and Erwin Corong for beta-testing the software. Software advice was provided by Steven Dirkse from the GAMS Development Corporation and Wietse Dol from Wageningen University. Very useful comments were also provided by the editors and two anonomous reviewers.

Appendix A. Installing the R packages and their features

The R scripts rely on two packages developed by Vincent Arel-Bundock. The first is called *WDI* and it contains the actual WDI extraction routine. The second is called *countrycode*. It contains lists of country codes developed by different agencies—such as the World Bank, United Nations and others. The *countrycode* function translates one coding system into another. The *WDI* function returns the data cube using ISO-2 character codes.[30] These are converted to ISO-3 character codes before saving the data.

Installation of the packages is only required once.[31] To install the packages from within R, enter the following commands:

```
install.packages("WDI")
install.packages("countrycode")
```

A full description of the functionality of both packages can be downloaded with the packages.[32]

This section highlights some of the other features of the *WDI* package. The *WDI* function has an optional parameter, 'extra', that is a Boolean parameter taking `TRUE` and `FALSE` values. If the parameter is set to `TRUE`, the data cube will contain additional country-specific information such as the ISO-3 code, the name of the World Bank region and the income status of the country. The *WDIcache* function will return the most recent list of indicators available in WDI. The *WDIsearch* function can be used to search for indicators containing a specific string, for example 'gdp'. The returned cache contains a wealth of other information such as capitals and latitude and longitude. The following R commands show how to save the information in the returned cache in CSV files that can readily be loaded into an Excel workbook. The cache is returned as two (text) matrices. The first contains the complete list of indicators with codes, description and source database. The second contains the full list of countries and regions with sundry additional information.

```
Cache <- WDIcache()%
write.table(Cache[1],"indTab.csv",row.names=TRUE,sep=",")
write.table(Cache[2],"regionTab.csv",row.names=TRUE,sep=",")
```

[30] The function can optionally also return the ISO-3 codes, so the use of the *countrycode* function is not strictly needed, but it is maintained to illustrate its functionality and users can modify the R scripts below to convert to different coding conventions.

[31] The WDI package also requires the *RJSONIO*, which is normally automatically loaded when the WDI package is initialized.

[32] Available at https://cran.r-project.org/web/packages/WDI/WDI.pdf and https://cran.r-project.org/web/packages/countrycode/countrycode.pdf.

Appendix B. The program listings

Listing B.1. Plotting historical growth periods (*gdppcPlot.R*).

```
# User data

# Set the input/output file names

# Windows
wDir <- "v:/data/wdi/"

# Mac
# wDir <- "~/Documents/Dominique/data/wdi/"

# Name of input file with country and region codes
fName <- "wbreg.csv"

# Output file name
bpName <- "gdppcBoxPlot"

# Type of output for box plot (valid options are PDF, WMF, JPG)

fType <- "pdf"

# First year
begYear <- 1960

# Final year
endYear <- 2014

# Growth interval (in years)
gap <- 10

# Threshold for dropping growth rates (in percent)
threshold <- 10.1

# END OF USER SECTION

# Initialize files
fName <- paste(wDir, fName, sep="")
bpName <- paste(wDir, bpName, sep="")
fType <- toupper(fType)

# Load the WDI library
library(WDI)

# Read the countries/regions to analyze
wb <- read.csv(fName, sep=",", header=TRUE, na.strings="")

# Get the codes for the WB regions
wbReg <- as.data.frame(lapply(wb, function(x) (table(x)))[4])

# Get the number of WB regions
nReg <- nrow(wbReg)

# Get the greatest number of countries in any given region
# And calculate the maximum number of growth episodes for any region
maxC <- max(wbReg$WBREG.Freq)
maxLen <- maxC*(endYear-begYear-gap+2)
```

```
# Initialize the data vectors for the growth rate estimations

y <- vector(mode="double", length=gap)
x <- vector(mode="double", length=gap)

# Loop over all WB regions

for(r in 1:nReg){

# Get the country ISO-2 codes for this region and get the number of countries
  iso <- wb[wb$WBREG==wbReg[r,1],3]
  nc <- length(iso)

# Extract the data from WDI
  pop <- WDI(iso, "sp.pop.totl", start=begYear, endYear)
  gdp <- WDI(iso, "ny.gdp.mktp.kd", start=begYear, endYear)

# Initialize the growth episode vector
  gr <- vector(mode = "double", length=maxLen)
  gr <- NA
  smpl <- 0

# Loop over all countries in this region
  for(c in 1:nc) {

#   Extract the data for this country from the regional database
    popc <- pop[pop$iso2c==iso[c],c("iso2c", "sp.pop.totl", "year")]
    gdpc <- gdp[gdp$iso2c==iso[c],c("iso2c", "ny.gdp.mktp.kd", "year")]

#   Sort the data by year
    popc <- popc[order(popc$year),]
    gdpc <- gdpc[order(gdpc$year),]

#   Calculate per capita GDP
    gdppc<- gdpc[,2]/popc[,2]

#   Calculate length of this vector
    maxYears <- length(gdppc)

#   For every available growth episode of size gap, calculate the growth rate
#     using OLS

    for(t in gap:maxYears) {
      i <- 0
      t1 <- t-gap+1
      t2 <- t

#     Fill the x and y vectors for OLS
      for(tt in t1:t2) {
        if(!is.na(gdppc[tt])) {
          i <-i+1
          y[i] = log(gdppc[tt])
          x[i] = i
        }
      }
#     Calculate the growth rate if we have a full vector
      if(i == gap){
#       Do the regression and extract the growth rate
        grRate <- 100*coef(lm(y ~ x))[2]
```

```
#         Only use this growth episode if growth is below the threshold
#           in absolute terms
          if(abs(grRate) <= threshold) {
            smpl <- smpl+1
            gr[smpl] <- grRate
          }
        }
      }
    }
# Set the name for this region and append the number of growth episodes
    regName <- paste(toString(wbReg[r,1]),"(",smpl,")",sep="")

# To concatenate the vectors, they have to be of identical length
#   Fill out the vector with NA
    s1 <- smpl+1
    for(z in s1:maxLen) {
      gr[z] = NA
    }

    if(r == 1) {
      result <- cbind(gr)
    }
    else {
      result <- cbind(result,gr)
    }
    colnames(result)[r] <- regName
}

# Create the box plot
if (fType == "PDF") {
    pdf(paste(bpName, ".pdf", sep=""), height=6.5, width=9)
} else if(fType == "WMF") {
    win.metafile(paste(bpName, ".wmf", sep=""), height=5.5, width=8.5)
} else if (fType == "JPG") {
    jpeg(paste(bpName, ".jpg", sep=""), height=480, width=580)
} else {
    stop(paste("Wrong file type <", ftype, ">", sep=""))
}

# Create the box plot

par(cex.axis=0.75, las=1)
boxplot(result, medcol="red", col="lightgreen",
          ylab="Per capita growth, percent")
grid(nx=NA, ny=NULL, col="lightgray", lty=1, lwd=1)

dev.off()
```

Listing B.2. Using R with LaTeX to create a WDI-based presentation (*extractWDIBeamer.R*).

```
# ------------------------------------------------------------------
#
#    Code to extract data series from WDI and create line charts
#              for a Latex-based Beamer presentation
#
#    Original code by: Dominique van der Mensbrugghe
#
# ------------------------------------------------------------------

# >>>>> BEGINNING OF USER INPUT

# Set output directory (use './' for default directory)

# Windows
outwd <- "v:/data/wdi/"

# MAC
# outwd <- "~/Documents/Dominique/data/wdi/"

# Set file name

fName <- "pptExample.tex"
fName <- paste(outwd, fName, sep="")

# Set title, author and address (Use \\ to separate lines)
pTitle <- "Example of creating slides from an R-based WDI extraction procedure"
shortTitle <- "WDI Charts"
author <- "Dominique van der Mensbrugghe"
nickName <- "DvdM"
address <- "Center for Global Trade Analysis (GTAP) \\\\ Purdue University \\\\
 West Lafayette, IN"
institute <- "GTAP"

# Set the beginning year

begYear <- 1990

# Set the final year

endYear <- 2013

# Select the regions to be extracted

# The regions are an r x 3 matrix were r is the number of regions
# Column 1 is the region ISO-2 code and is used for the extraction
# Column 2 is the region ISO-3 code
#      (and will be used for the legend, and otherwise can be changed)
# Column 3 is the region name (and currently ignored)

regions <- rbind(cbind("4e","EAP","East Asia & Pacific"),
                 cbind("8S","SAS","South Asia"),
                 cbind("7E","ECA","Europe & Central Asia"),
                 cbind("XQ","MNA","Middle East & North Africa"),
                 cbind("ZF","SSA","Sub-Saharan Africa"),
                 cbind("XJ","LAC","Latin America & Caribbean"),
                 cbind("XD","HIC","High-income countries")
                )
```

```
# Select the indicators

# The indicators are an n x 4 matrix were n is the number of indicators
# Column 1 is an indicator name (ignored for the moment)
# Column 2 is the WB CETS code and is used for the extraction
# Column 3 is a scale that will be used to scale the output data
# Column 4 describes the indicator -- it will be used as the chart title

indTable <- rbind(
  cbind("pop","sp.pop.totl",1e6,"Population (million)"),
  cbind("gdpkd","ny.gnp.mktp.kd",1e9, "GDP (\\$2005 billion)"),
  cbind("gdpkdpp","ny.gdp.mktp.pp.kd",1e12, "GDP (\\$2005PPP trillion)"),
  cbind("gdppckd","ny.gnp.pcap.kd",1e0, "GDP per capita (\\$2005)"),
  cbind("gdppckdpp","ny.gdp.pcap.pp.kd",1e0, "GDP per capita (\\$2005PPP)")
)


# Set the colors for the lines -- see Latex's 'xcolor' package for choices

Colors <- rbind(
     "blue",
     "LightSkyBlue",
     "DarkGrey",
     "ForestGreen",
     "red",
     "black",
     "Orchid",
     "Goldenrod",
     "ForestGreen",
     "red",
     "black",
     "Orchid",
     "Goldenrod"
)

# END OF USER INPUT <<<<<

# ------------------------------------------------------------------------
#
# Main part of WDI extraction
#
# ------------------------------------------------------------------------

# Load the WDI library

library(WDI)

# Get the number of indicators and regions

nInd <- nrow(indTable)
nReg <- nrow(regions)
maxColor <- nrow(Colors)

# Get the iso2 codes

iso2 <- toupper(regions[,1])

# Write the Latex file preamble
# N.B. It is possible to change 'beamer' themes and theme colors
```

```
sink(fName,append=FALSE, split=FALSE)
cat("\\documentclass[xcolor={dvipsnames,svgnames}]{beamer}\n")
cat("\\usetheme{CambridgeUS}\n")
cat("\\usecolortheme{orchid}\n\n")
cat("\\usepackage{pgfplots}\n\n")
cat("\\begin{document}\n\n")
cat("\\pgfplotsset{every axis/.append style={line width=0.75pt},
    axis x line*=bottom, axis y line*=left}\n")
cat("\\pgfsetplotmarksize{0pt}\n\n")
cat("\\title[",shortTitle,"]{",pTitle,"}\n",sep="")
cat("\\author[",nickName,"]{",author,"}\n",sep="")
cat("\\institute[",institute,"]{",address,"}\n",sep="")
cat("\\date{\\today}\n")
cat("\\frame{\\titlepage}\n\n")
sink()

# Loop over all indicators

for (i in 1:nInd) {

# Get the data from the WDI database

  cube <- WDI(iso2, indicator=indTable[i,2], start=begYear, endYear)

# Save the chart pre-amble

  sink(fName,append=TRUE,split=FALSE)
  cat("\\frame{\\frametitle{",indTable[i,4],"}\n", sep="")
  cat("\\begin{figure}[h]\n")
  cat("\\centering\n")
  cat("\\begin{tikzpicture}[scale=0.5]\n")
  cat("\\begin{axis}[width=19.0cm, height=13.5cm,\n")
  cat(" xlabel={Year},\n")
  cat(" xmin={",begYear-1,"},\n",sep="")
  cat(" xmax={",endYear,"},\n",sep="")
  cat(" ymajorgrids,\n")
  cat(" legend style={draw=none},\n")
  cat(" x tick label style={/pgf/number format/.cd, scaled x ticks = false,set
  thousands separator={},fixed},\n")
  cat(" y tick label style={/pgf/number format/.cd, scaled y ticks = false,set
  decimal separator={.},fixed},\n")
  cat(" legend style={at={(1.03,0.5)},anchor=west,font=\\footnotesize},\n")
  cat(" legend cell align=left\n")
  cat(" ]\n")


# Set the scale for this indicator and initialize the color table
  indScale <- as.numeric(indTable[i,3])
  nColor <- 0

# Loop over all regions and write out the data
  for(r in 1:nReg) {

    nColor <- nColor + 1
    if(nColor > maxColor) nColor <- 1

#   Get the data for the selected region

    regData <- cube[cube$iso2c==iso2[r],c("iso2c", indTable[i,2], "year")]
```

```
#    Sort the data by year
     x <- regData[order(regData$year),]

#    Get the number of years
     nYear = nrow(x)

#    Save the data for this indicator

     cat("\\addplot+[",Colors[nColor],",smooth,line width=2.0pt] coordinates\n",
     sep="")
     cat("{\n")

     for(t in 1:nYear) {
       cat("(",x[t,3],",",x[t,2]/(indScale),")\n",sep="")
     }
     cat("};\n")
     cat("\\addlegendentry{",regions[r,2],"}\n",sep="")
   }

# End the graph

   cat("\\end{axis}\n")
   cat("\\end{tikzpicture}\n")
   cat("\\end{figure}\n")
   cat("}\n\n")
   sink()
}

# Write the Latex closing

sink(fName,append=TRUE,split=FALSE)
cat("\\end{document}\n\n")
sink()
```

Listing B.3. A GAMS file to retrieve data from WDI (*gdpr.gms*).

```
*
*-------------------------------------------------------------------------------
*
*    Example of a GAMS file that calls R to extract data from
*      the World Bank's WDI database
*
*    Description:
*      This GAMS file provides an example of how to combine GAMS with R to
*      extract data from the World Bank's World Development Indicators
*      Database (WDI). The user prepares a list of countries and indicators to
*      extract from WDI, including the range of years. The GAMS program will
*      save all of this information in a text file and then invoke R as a
*      sub-process (using R's RScript program). If successful, the R script
*      will save the extracted data as a CSV file. The file is converted to
*      a GDX gile before being read back in to the GAMS file. This is
*      necessary because CSV files are read-in at compile time, but GDX files
*      are read in at execution time. A utility called CSV_GDX_Tools is used
*      to convert the R-outputted CSV file, to the GDX format.
*      N.B. When reading CSV files GAMS does not want a header line in the
*      data file that describes the different fields in the CSV file.
*      However the CSV to GDX converter, and many database-type translators
*      of CSV files, such as Excel's pivot table feature require
*      the header line. This is one of the user options provided below.
*
*    Requirements:
*
*      isocodes.gms: A file containing a current set of ISO-2 and ISO-3 codes
*                    with their mapping. In addition, the file contains a
*                    mapping from ISO-3 to the GTAP regions.
*
*      R:            The RScript function is used to run R in background.
*                    When invoked, RScript should be in the system's path.
*                    If not, invoke R with full path name.
*
*      CSV_GDX_Tools The CSV to GDX conversion program. It needs to be
*                    in the system's path, or else, the full path name
*                    needs to be provided. (N.B. The tool also requires
*                    a file called 'forbiddenwords.txt' that should be
*                    in the same folder as the conversion tool.)
*
*      extractWDI.R The name of the R script file that performs the
*                    extraction from WDI and saves the data as a CSV file.
*                    Users are welcome to modify and/or rename the file.
*
*    User inputs:
*
*      BASENAME    This will be the base file name for all of the
*                  files created by the program.
*
*      FIRSTYR     First year to extract. The extraction routine requires a
*                  continuous range.
*
*      LASTYR      The last year to extract.
*
*      EXTALL      A true or false flag. If 'true' the extraction routine
*                  will download the data for all countries in the WDI
*                  database for the selected indicators. Any country
*                  information (for example as defined in the subset 'c2'
```

```
58  *                 will be ignored. If 'false', only data for countries
59  *                 identified in the subset 'c2' will be extracted.
60  *
61  *      IFHEADER   A true or false flag. If 'true' a header line will be
62  *                 included at the top of the CSV file with labels for the
63  *                 CSV file fields. If 'false', no header line will be
64  *                 output. For reading the CSV data into GAMS, the flag
65  *                 should be set to 'false'.
66  *
67  *      IFNAME     A true or false flag. The WDI extraction also retrieves
68  *                 the country names. If this flag is set to 'false', the
69  *                 country names will not be saved with the data. These can
70  *                 nonetheless be retrieved by the GAMS code since the
71  *                 country names are provided with the set definitions. The
72  *                 information is therefore redundant and takes up storage
73  *                 space (and can also not be readily read into a GAMS
74  *                 matrix). Set to 'true' to save the country names with
75  *                 the CSV data.
76  *
77  *      c2         C2 is a subset of the ISO-2 codes (that are used by the
78  *                 WDI extraction routine). The subset is user-defined and
79  *                 contains the ISO-2 codes for the countries which are to
80  *                 be extracted. The subset is ignored if 'EXTALL' is set
81  *                 to 'true'.
82  *
83  *      var        Var is a set of indicators to extract. The user provides
84  *                 short names for the extracted indicators--though is free
85  *                 to use the same mnemonic as the World Bank database. The
86  *                 set Var must be paired with the set wbind and the mapv
87  *                 mapping (see below).
88  *
89  *      wbind      WBInd is a set of indicators to extract using the WB's
90  *                 mnemonics. It is typically paired with the set Var and
91  *                 the mapping mapv.
92  *
93  *      mapv       The set mapv pairs the user-named indicators (Var) with
94  *                 the WB-named indicators (WBInd). It should be a
95  *                 one-to-one mapping.
96  *
97  *
    ------------------------------------------------------------------------------

99  acronym true, false ;

101 * Read the ISO-2, ISO-3 and GTAP codes and mappings

103 $offlisting
104 $include "isocodes.gms"
105 $onlisting

107 *
    ------------------------------------------------------------------------------
108 *
109 * BEGINNNING OF USER INPUT
110 *
111 *
    ------------------------------------------------------------------------------

113 * System options
114 $setglobal R_EXE C:\Program Files\R\R-3.2.2\bin\RScript
```

```
$setglobal CSVGDX "V:\bin\CSV_GDX_Tools"
$setglobal RSCRIPT "extractWDI.R"

* User options
$setglobal BASENAME "GDPR"
$setglobal FIRSTYR 2000
$setglobal LASTYR 2014
$setglobal EXTALL false
$setglobal IFHEADER true
$setglobal IFNAME false

* NOTE: If EXTALL is set to true, the following subset will be ignored
*      no matter how it is defined.

set c2(iso2) "Countries to extract" / fr, us, gb, be, bz / ;

set var "Indicators to extract" /
   "pop"
   "gdpcd"
   "gdpkd"
   "gnpppcd"
   "gnpppkd"
/ ;

set wbind "WB indicators to extract" /
   "SP.POP.TOTL"
   "NY.GDP.MKTP.CD"
   "NY.GDP.MKTP.KD"
   "NY.GDP.PCAP.PP.CD"
   "NY.GDP.PCAP.PP.KD"
/ ;

set mapv(var, wbind) "Mapping of variable names to WB indicators" /
   "pop"    . "SP.POP.TOTL"
   "gdpcd" . "NY.GDP.MKTP.CD"
   "gdpkd" . "NY.GDP.MKTP.KD"
   "gnpppcd" . "NY.GDP.PCAP.PP.CD"
   "gnpppkd" . "NY.GDP.PCAP.PP.KD"
/ ;

parameter scale(var) "Scaling factor for indicators" /
   "pop"    1e-6
   "gdpcd"  1e-6
   "gdpkd"  1e-6
   "gnpppcd" 1e-0
   "gnpppkd" 1e-0
/ ;

*
*-------------------------------------------------------------------------------
*
* END OF USER INPUT
*
*
*-------------------------------------------------------------------------------

sets
   year   "Years to extract" / %FIRSTYR%*%LASTYR% /
   y0(year) "First extraction year" / %FIRSTYR% /
   yf(year) "Last extraction year" / %LASTYR% /
```

```
;

set c(iso3) "Countries being extracted with ISO-3 codes" ;

if(%EXTALL% eq false,
  loop(mapISO(iso2,iso3)$c2(iso2),
    c(iso3) = yes ;
  ) ;
else
  c(iso3) = yes ;
) ;

* Delete intermediate files if they exist

$if exist "%BASENAME%.opt" $call 'del "%BASENAME%.opt"'
$if exist "%BASENAME%.csv" $call 'del "%BASENAME%.csv"'

file fopt / "%BASENAME%.opt" / ;
put fopt ;
put '"Name","Value"' / ;

put '"Output file", "%BASENAME%.csv"' / ;

loop(y0, put '"Start year", ', y0.tl / ; ) ;
loop(yf, put '"End year", ', yf.tl / ; ) ;

if(%IFHEADER% eq false,
  put '"Header", "FALSE"' / ;
else
  put '"Header", "TRUE"' / ;
) ;

if(%IFNAME% eq false,
  put '"Country name", "FALSE"' / ;
else
  put '"Country name", "TRUE"' / ;
) ;

scalar ifFirst / 1 / ;

put '"Regions", "' ;
if(%EXTALL% eq false,
  loop(c2,
    if(ifFirst,
      put c2.tl:2 ;
      ifFirst = 0 ;
    else
      put ',', c2.tl:2 ;
    ) ;
  ) ;
else
  put "all" ;
) ;

put '"' / ;
put '"Indicators","wbName"' / ;
fopt.pc=5 ;
loop(mapv(var,wbind),
  put var.tl, wbind.tl / ;
) ;
```

```
putclose fopt ;

* Extract the data using R
execute '"%R_EXE%" %RSCRIPT% %BASENAME%.opt'

* Convert the data to GDX format
execute '%CSVGDX% "%BASENAME%.csv" comma "All" "Var,iso3,Year" /method=csvgdx /
PARNAME=wdiData /GDX=%BASENAME%.gdx'

* Load the GDX data

parameter
   wdiData(var,iso3,year)
;
execute_load "%BASENAME%.gdx", wdiData ;

* Process the read-in data
wdiData(var,c,year) = scale(var)*wdiData(var,c,year) ;
option decimals=0 ;
display wdiData ;

* Cleanup intermediate files
if(1,
$if exist "%BASENAME%.opt" $call 'del "%BASENAME%.opt"'
$if exist "%BASENAME%.csv" $call 'del "%BASENAME%.csv"'
$if exist "%BASENAME%.gref" $call 'del "%BASENAME%.gref"'
) ;
```

Listing B.4. WDI.Retrieve function (*extractWDI.R*).

```
wdi.Retrieve <- function(cName, sName, indName, startYear, endYear,
    fName, ifAppend, ifName) {
#
#   Arguments:
#      cName:   List of countries/regions being extracted (could be "all")
#      sName:   Short series name (will be output to CSV file)
#      indName: Mnemonic of WDI indicator
#      startYear: Beginning year
#      endYear: End year
#      fName:   Name of output file
#      ifAppend: FALSE for first record (normally header), otherwise TRUE
#      ifName:  FALSE to delete country name from output, otherwise TRUE
#

# Retrieve the indicator

  cube <- WDI(country=cName, indicator=indName, start=startYear, end=endYear)

# Convert the ISO-2 labels to ISO-3 labels -- requires the countrycode package

  cube$iso3c <- countrycode(cube$iso2c,"iso2c","iso3c")

# Fix ISO-3 codes for missing countries

  iso2Ndx <- grep('iso2c', colnames(cube))
  iso3Ndx <- grep('iso3c', colnames(cube))

# !!!! May need to add other countries and/or change ISO-3 codes

  cube[iso3Ndx][cube[iso2Ndx]=='JG']<-"JGY"
  cube[iso3Ndx][cube[iso2Ndx]=='XK']<-"KOS"

# Delete regions with no iso code

  cube <- cube[!(is.na(cube$iso3c)),]

# Append the variable name to the dataframe

  cube[,"Var"] <- sName

#  Create the final data frame by re-ordering columns

  if(ifName) {

    cube <- cube[,c("Var", "iso3c", "country", "year", indName)]

  }
  else {

    cube <- cube[,c("Var", "iso3c", "year", indName)]

  }

# Save the data in CSV format

  write.table(cube, fName, append=ifAppend, row.names=FALSE,
      col.names=FALSE, na="", sep=",")
}
```

Listing B.5. Main part of R WDI extraction script used with GAMS (*extractWDI.R*).

```
#
#------------------------------------------------------------------------------
#
# Main part of WDI extraction
#
#
#------------------------------------------------------------------------------

# Load the needed libraries (after installation on the local machine)

library(WDI)
library(countrycode)

# Read the command line arguments

args <- commandArgs(TRUE)

if(length(args) < 1) {
  srcFile <- "extractWDI.opt"
} else {
   srcFile <- args[1]
}

# Get the options from the options file

options <- read.csv(srcFile, stringsAsFactors = FALSE,
     strip.white=TRUE, quote = "\"'")

# Get the name of the output file

outFileName <- options[grep("Output file", options$Name),2]

# Get the beginning year

begYear <- as.numeric(options[grep("Start year", options$Name),2])

# Get the final year

endYear <- as.numeric(options[grep("End year", options$Name),2])

# Flag for header

ifHeader <- as.logical(options[grep("Header", options$Name),2])

# Flag for outputting country name

ifName <- as.logical(options[grep("Country name", options$Name),2])

# Get the regions -- 'all' is to get all regions,must use iso2c codes

regions <- options[grep("Regions", options$Name),2]

if(regions != "all") {
   regions <- toupper(regions)
}

# Parse the regions
```

```
58  regions <- strsplit(regions,",")
59  regions <- c(do.call("cbind",regions))

61  # Get the indicators -- assumes input file has no blank lines

63  row1 <- 8
64  row2 <- nrow(options)
65  indTable <- options[row1:row2,]

67  # Get the number of indicators

69  nInd <- nrow(indTable)

71  if(ifHeader) {

73  # Use header for CSV file
74    if(ifName) {
75      header <- cbind("Var", "iso3", "regName","Year", "Val")
76    }
77    else {
78      header <- cbind("Var", "iso3", "Year", "Val")
79    }

81    write.table(header, outFileName, append=FALSE, row.names=FALSE,
82        col.names=FALSE, sep=",")
83  }
84  ifAppend <- ifHeader

86  # Loop over all indicators

88  for (i in 1:nInd) {

90  # Get the short and World Bank name for the indicators

92    var <- indTable[i,1]
93    wbName <- indTable[i,2]

95  # Use the function above to retrieve the data, convert iso2 to iso3, drop the
96  #   regions, and append to the CSV file

98    wdi.Retrieve(regions, var, wbName, begYear, endYear,
99        outFileName, ifAppend, ifName)

101 # After the first indicator, ifAppend must be TRUE

103   ifAppend <- TRUE
104 }
```