hw02: Analyze PWT with R and tidyverse

Kenji Sato
Kobe University
mail@kenjisato.jp

2017/12/07

1 Overview

Purpose

To become familiar with R and **tidyverse** and to play with the Penn World Table (Feenstra, Inklaar, and Timmer 2015).

Instructions

In this assignment, you will

- clone the assignment repository and make a working branch (eg. `solution` branch);
- solve the problems in Section 5;
- write the solutions in `solution.Rmd` and knit the file;
- commit `solution.Rmd` and `solution.pdf`; and
- open a Pull Request.

2 Set Up

Before you get started, please download the Penn World Table dataset and place it in an appropriate directory. You can use the helper script I provide. Look at the `R` folder, read throught the code in `R/pwt-setup.R` and then execute the following line of code in the console.[1]

```
source("R/pwt-setup.R")
```

Now you should have PWT dataset on your computer. To load this dataset in R, I would recommend using `haven::read_dta()` function from **haven** package, which comes with **tidyverse**.

```
pwt <- haven::read_dta("~/Data/pwt90.dta")
pwt
```

```
## # A tibble: 11,830 x 47
##    countrycode country  currency_unit  year rgdpe rgdpo   pop   emp   avh
##          <chr>   <chr>          <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
```

[1] `source()` function reads the file (R script) passed as the first argument and executes the R code written in the file. `"R/pwt-setup.R"` is a string that specifies a relative path from your working directory to the file. It assumes that there is an `R` folder under the working directory, and a file named `pwt-setup.R` exists in that `R` folder. If you see an error saying “No such file or directory,” your working directory is different from what I expect or you may have mistakenly removed the file.

```
##  1         ABW   Aruba Aruban Guilder  1950    NA     NA     NA     NA     NA
##  2         ABW   Aruba Aruban Guilder  1951    NA     NA     NA     NA     NA
##  3         ABW   Aruba Aruban Guilder  1952    NA     NA     NA     NA     NA
##  4         ABW   Aruba Aruban Guilder  1953    NA     NA     NA     NA     NA
##  5         ABW   Aruba Aruban Guilder  1954    NA     NA     NA     NA     NA
##  6         ABW   Aruba Aruban Guilder  1955    NA     NA     NA     NA     NA
##  7         ABW   Aruba Aruban Guilder  1956    NA     NA     NA     NA     NA
##  8         ABW   Aruba Aruban Guilder  1957    NA     NA     NA     NA     NA
##  9         ABW   Aruba Aruban Guilder  1958    NA     NA     NA     NA     NA
## 10         ABW   Aruba Aruban Guilder  1959    NA     NA     NA     NA     NA
## # ... with 11,820 more rows, and 38 more variables: hc <dbl>, ccon <dbl>,
## #   cda <dbl>, cgdpe <dbl>, cgdpo <dbl>, ck <dbl>, ctfp <dbl>,
## #   cwtfp <dbl>, rgdpna <dbl>, rconna <dbl>, rdana <dbl>, rkna <dbl>,
## #   rtfpna <dbl>, rwtfpna <dbl>, labsh <dbl>, delta <dbl>, xr <dbl>,
## #   pl_con <dbl>, pl_da <dbl>, pl_gdpo <dbl>, i_cig <dbl+lbl>,
## #   i_xm <dbl+lbl>, i_xr <dbl+lbl>, i_outlier <dbl+lbl>, cor_exp <dbl>,
## #   statcap <dbl>, csh_c <dbl>, csh_i <dbl>, csh_g <dbl>, csh_x <dbl>,
## #   csh_m <dbl>, csh_r <dbl>, pl_c <dbl>, pl_i <dbl>, pl_g <dbl>,
## #   pl_x <dbl>, pl_m <dbl>, pl_k <dbl>
```

If you see error saying `Error in loadNamespace(name) : there is no package called 'haven'` in any of your libraries, please install it by running the following code in the console.[2]

```r
install.packages("tidyverse")
```

In the following, we assume that **tidyverse** is loaded on memory. Do this:

```r
library(tidyverse)
```

You might be worried about the disturbing message that tells you there are conflicts of names but you do not have to be.

You see this message because both **dplyr** (loaded with **tidyverse**) and **stats** (loaded at start up) packages have functions with identical names. You can no longer (in this session) use `filter()` function of the **stats** package simply with `filter()`, because the name now points to `filter()` function defined in the **dplyr** package. It does not mean you can never use the former function; it does mean that you must use it with its full name `stats::filter()`.

3 dplyr primer

Table 1 shows all the variables the table has along with short descriptions for the variables.

Often times, we do not need all of these variables for analysis. To trim away unnecessary data, we will make use of **dplyr**, a package for data processing, which comes with **tidyverse**.

Since `pwt90` is too big to learn programming concepts with, let's make a smaller toy dataset with `tibble()`.[3]

```r
tbl <- tibble(
  id = letters[1:4],
  salary = 400 + rnorm(4, 0, 50),
  sex = c("M", "M", "F", "F")
```

[2] **haven** is a part of the **tidyverse** package family. Notice, however, that `library("tidyverse")` does not load **haven** automatically. You need to `library("haven")` separately or call functions in **haven** with the form of `haven::function_name()` like `haven::read_dta()`.

[3] `tibble` or `tbl_df` is an extension of `data.frame` of base R. Run `vignette("tibble")` for more information.

Table 1: pwt90.dta

name	label
countrycode	3-letter ISO country code
country	Country name
currency_unit	Currency unit
year	Year
rgdpe	Expenditure-side real GDP at chained PPPs (in mil. 2011US$)
rgdpo	Output-side real GDP at chained PPPs (in mil. 2011US$)
pop	Population (in millions)
emp	Number of persons engaged (in millions)
avh	Average annual hours worked by persons engaged (source: The Conference Board)
hc	Human capital index, see note hc
ccon	Real consumption of households and government, at current PPPs (in mil. 2011US$)
cda	Real domestic absorption, see note cda
cgdpe	Expenditure-side real GDP at current PPPs (in mil. 2011US$)
cgdpo	Output-side real GDP at current PPPs (in mil. 2011US$)
ck	Capital stock at current PPPs (in mil. 2011US$)
ctfp	TFP level at current PPPs (USA=1)
cwtfp	Welfare-relevant TFP levels at current PPPs (USA=1)
rgdpna	Real GDP at constant 2011 national prices (in mil. 2011US$)
rconna	Real consumption at constant 2011 national prices (in mil. 2011US$)
rdana	Real domestic absorption at constant 2011 national prices (in mil. 2011US$)
rkna	Capital stock at constant 2011 national prices (in mil. 2011US$)
rtfpna	TFP at constant national prices (2011=1)
rwtfpna	Welfare-relevant TFP at constant national prices (2011=1)
labsh	Share of labour compensation in GDP at current national prices
delta	Average depreciation rate of the capital stock
xr	Exchange rate, national currency/USD (market+estimated)
pl_con	Price level of CCON (PPP/XR), price level of USA GDPo in 2011=1
pl_da	Price level of CDA (PPP/XR), price level of USA GDPo in 2011=1
pl_gdpo	Price level of CGDPo (PPP/XR), price level of USA GDPo in 2011=1
i_cig	0/1/2, see note i_cig
i_xm	0/1/2, see note i_xm
i_xr	0/1: the exchange rate is market-based (0) or estimated (1)
i_outlier	0/1, see note i_outlier
cor_exp	Correlation between expenditure shares, see note cor_exp
statcap	Statistical capacity indicator (source: World Bank, developing countries only)
csh_c	Share of household consumption at current PPPs
csh_i	Share of gross capital formation at current PPPs
csh_g	Share of government consumption at current PPPs
csh_x	Share of merchandise exports at current PPPs
csh_m	Share of merchandise imports at current PPPs
csh_r	Share of residual trade and GDP statistical discrepancy at current PPPs
pl_c	Price level of household consumption, price level of USA GDPo in 2011=1
pl_i	Price level of capital formation, price level of USA GDPo in 2011=1
pl_g	Price level of government consumption, price level of USA GDPo in 2011=1
pl_x	Price level of exports, price level of USA GDPo in 2011=1
pl_m	Price level of imports, price level of USA GDPo in 2011=1
pl_k	Price level of the capital stock, price level of USA 2011=1

```r
)
tbl
```

```
## # A tibble: 4 x 3
##      id   salary   sex
##   <chr>    <dbl> <chr>
## 1     a 422.2181     M
## 2     b 488.3323     M
## 3     c 466.0057     F
## 4     d 409.7065     F
```

3.1 `filter`

`filter()` can be used to take rows that satisfy certain conditions. To retrieve rows with `salary` more than `400`, you can use the below code.

```r
filter(tbl, salary > 400)
```

```
## # A tibble: 4 x 3
##      id   salary   sex
##   <chr>    <dbl> <chr>
## 1     a 422.2181     M
## 2     b 488.3323     M
## 3     c 466.0057     F
## 4     d 409.7065     F
```

To retrieve rows that `sex` is `"M"`,

```r
filter(tbl, sex == "M")
```

```
## # A tibble: 2 x 3
##      id   salary   sex
##   <chr>    <dbl> <chr>
## 1     a 422.2181     M
## 2     b 488.3323     M
```

To get rows that `sex` is `"M"` and `salary` is more than `400`,

```r
filter(tbl, sex == "M" & salary > 400)
```

```
## # A tibble: 2 x 3
##      id   salary   sex
##   <chr>    <dbl> <chr>
## 1     a 422.2181     M
## 2     b 488.3323     M
```

To get rows that `sex` is `"F"` or `salary` is less than or equal to`400`,

```r
filter(tbl, sex == "F" | salary <= 400)
```

```
## # A tibble: 2 x 3
##      id   salary   sex
##   <chr>    <dbl> <chr>
## 1     c 466.0057     F
## 2     d 409.7065     F
```

3.2 `select`

To choose clumns, use `select`.

```
select(tbl, id, salary)
```

```
## # A tibble: 4 x 2
##      id   salary
##   <chr>    <dbl>
## 1     a 422.2181
## 2     b 488.3323
## 3     c 466.0057
## 4     d 409.7065
```

You can remove columns by appending negative sign.

```
select(tbl, - salary)
```

```
## # A tibble: 4 x 2
##      id   sex
##   <chr> <chr>
## 1     a     M
## 2     b     M
## 3     c     F
## 4     d     F
```

3.3 `mutate` and `transmute`

To manipulate data in columns, use `mutate` or `transmute`.

`mutate` adds new columns. Let's suppose that `salary` is measured in million yen unit and that we want to change the unit to thousand yen. This is achieved with the following code.

```
mutate(tbl, salary_in_thousand = 1000 * salary)
```

```
## # A tibble: 4 x 4
##      id   salary   sex salary_in_thousand
##   <chr>    <dbl> <chr>              <dbl>
## 1     a 422.2181     M           422218.1
## 2     b 488.3323     M           488332.3
## 3     c 466.0057     F           466005.7
## 4     d 409.7065     F           409706.5
```

`transmute` removes all variable other than those explicitly specified.

```
transmute(tbl, id, salary_in_thousand = 1000 * salary)
```

```
## # A tibble: 4 x 2
##      id salary_in_thousand
##   <chr>              <dbl>
## 1     a           422218.1
## 2     b           488332.3
## 3     c           466005.7
## 4     d           409706.5
```

3.4 `%>%`

You can combine the above functions (and many others) with pipe operator %>% from **magrittr** package, on which **dplyr** depends.

Let's see an example.

```
tbl %>%
  filter(salary > 400) %>%
  select(id, sex)
```

```
## # A tibble: 4 x 2
##       id   sex
##    <chr> <chr>
## 1      a     M
## 2      b     M
## 3      c     F
## 4      d     F
```

This is equivalent to the following.

```
tbl_tmp <- filter(tbl, salary > 400)
select(tbl_tmp, id, sex)
```

```
## # A tibble: 4 x 2
##       id   sex
##    <chr> <chr>
## 1      a     M
## 2      b     M
## 3      c     F
## 4      d     F
```

Piping makes a chain of commands look much neater.

3.5 `group_by` and `aggregate`

Another operation we might want to perform is to compute group-wise statistics. The following code computes the ration of the highest salary to the lowest within each of male and female groups.

```
tbl %>%
  group_by(sex) %>%
  summarise(mean = max(salary) / min(salary))
```

```
## # A tibble: 2 x 2
##      sex     mean
##    <chr>    <dbl>
## 1      F 1.137414
## 2      M 1.156588
```

4 PWT and plotting with ggplot2

Now is the time to work with PWT. Let's focus on the following ten countries.

```
countries <- c("United States", "United Kingdom", "Germany", "France",
               "Italy", "Japan", "Canada", "China", "Republic of Korea", "India")
```

We extract country, year, rgdpo, pop.

```r
pwt10 <-
  pwt %>%
  filter(country %in% countries) %>%
  select(country, year, rgdpo, pop)
pwt10
```

```
## # A tibble: 650 x 4
##    country  year    rgdpo      pop
##      <chr> <dbl>    <dbl>    <dbl>
##  1  Canada  1950 155053.0 13.81121
##  2  Canada  1951 160307.0 14.12590
##  3  Canada  1952 174147.9 14.57431
##  4  Canada  1953 182327.0 14.96642
##  5  Canada  1954 181436.9 15.41282
##  6  Canada  1955 197522.2 15.82101
##  7  Canada  1956 213976.1 16.21010
##  8  Canada  1957 219338.9 16.76710
##  9  Canada  1958 224430.2 17.21249
## 10  Canada  1959 233373.6 17.61666
## # ... with 640 more rows
```

To visualize the GDP growth of these countries, we use **ggplot2** package, which again comes with **tidyverse**. The following code produces Figure 1.

```r
ggplot(pwt10) + geom_line(aes(x = year, y = rgdpo, color = country))
```

```
## Warning: Removed 5 rows containing missing values (geom_path).
```

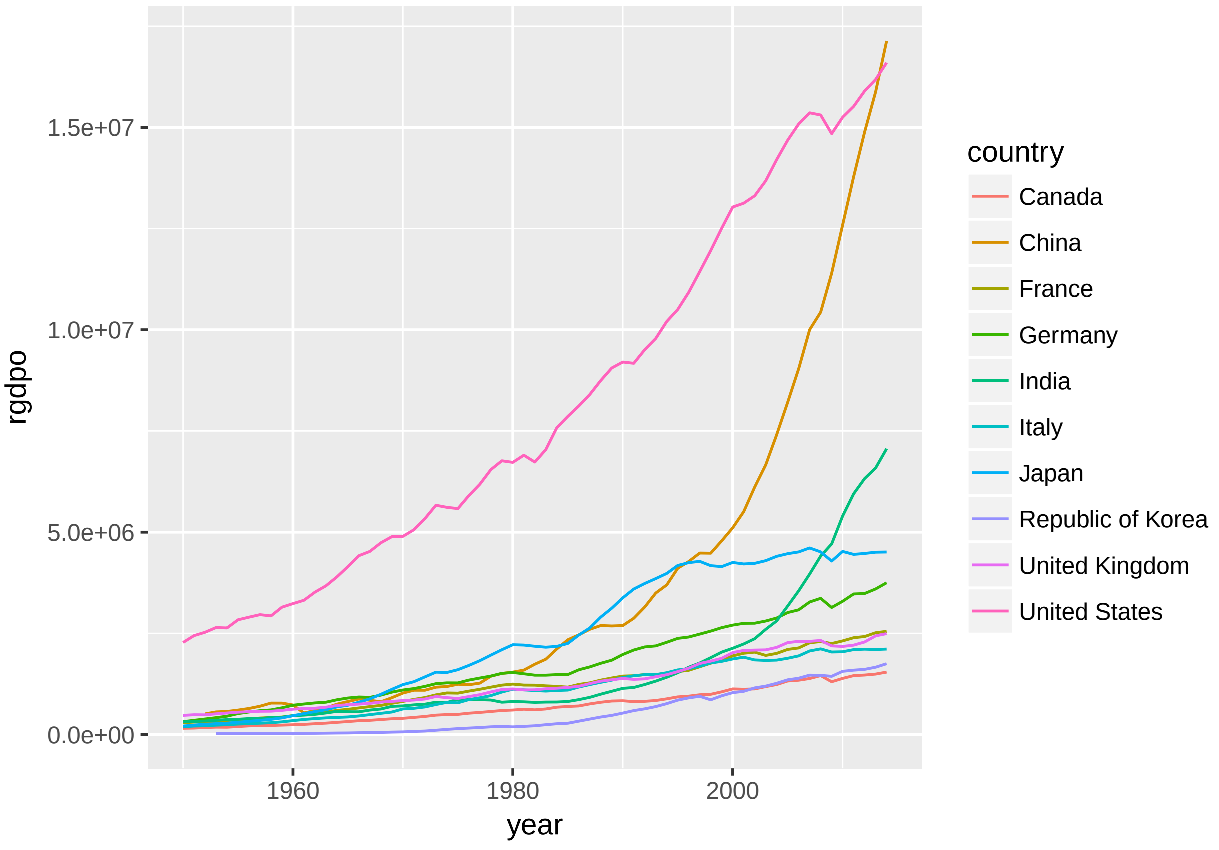

Figure 1: Real GDP

The following code produces Figure 2. The graphs show roughly constant growth of log real GDP.

```r
ggplot(pwt10) + geom_line(aes(x = year, y = rgdpo, color = country)) +
  scale_y_log10()
```

```
## Warning: Removed 5 rows containing missing values (geom_path).
```

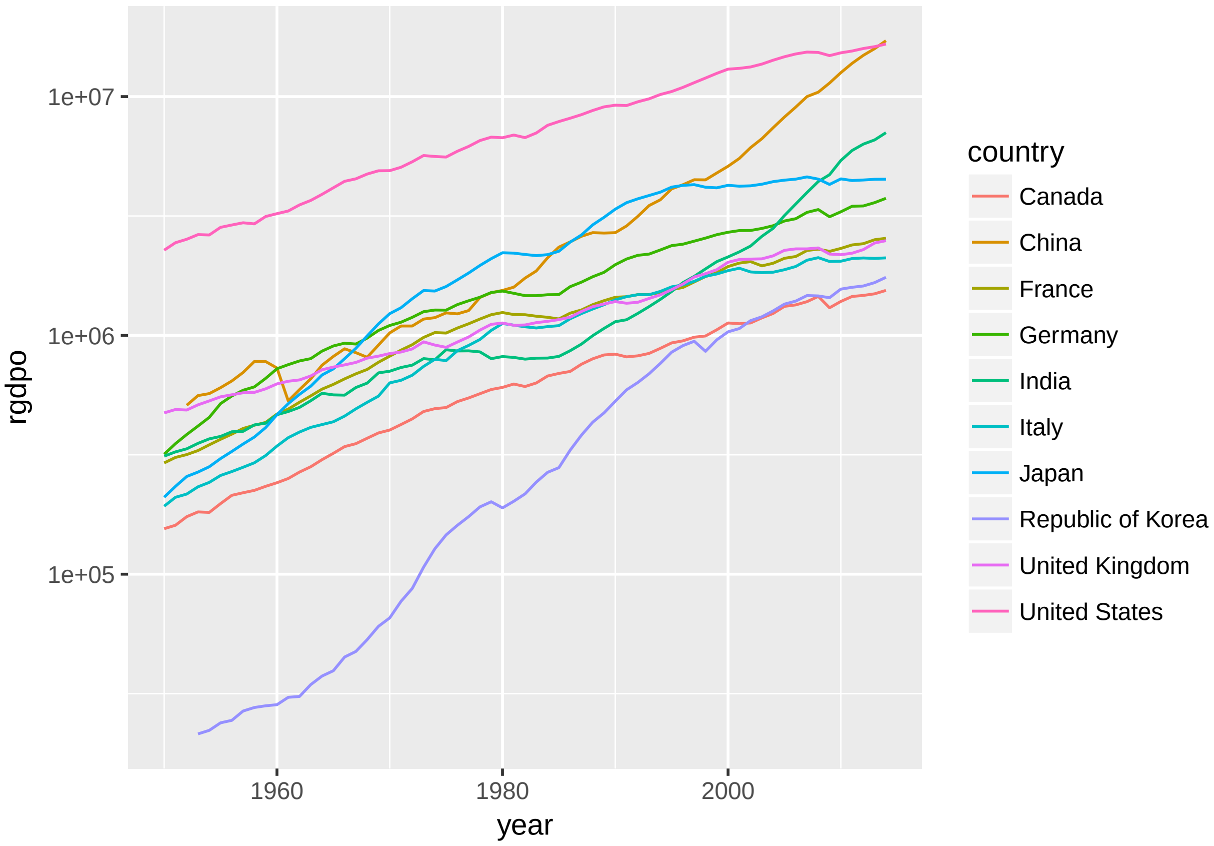

Figure 2: Real GDP on log scale

5 Problem

Consider the period between 1960 and 2014. Compute the average annual real GDP growth rates for these countries chosen earlier. Which country did grow the fastest?

How about the growth rates for real GDP per capita?

Write your answer along with code in `solution.Rmd`, knit it, and submit through a PR.

References

Feenstra, Robert C., Robert Inklaar, and Marcel P. Timmer. 2015. "The Next Generation of the Penn World Table." *American Economic Review* 105 (10): 3150–82.