R vs. Excel: A Simple Comparison

Bas Machielsen

February 13, 2020

Abstract

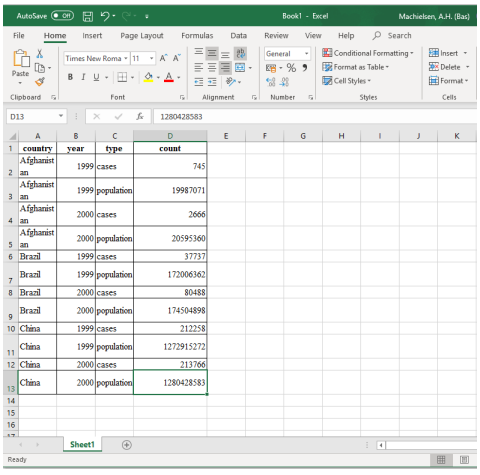
Microsoft Excel is often used by students and teachers alike to perform elementary data treatment, and generate descriptive statistics and accompanying graphs. Contrary to popular wisdom, however, Excel is not user-friendly, and the quality of its output is, by most standards, judged as inferior to the outputs of various alternatives. In this pamphlet, I contrast Excel with R. I proceed by showing the most common way of doing five elementary data operations in Excel, and then perform each of them in R: Data tidying, by subset statistics, data merging, creating elementary graphs, and string manipulation. By comparing and contrasting both approaches, I intend to show the approach using R is superior, and Microsoft Excel can be left aside in the future.

1 Data Tidying

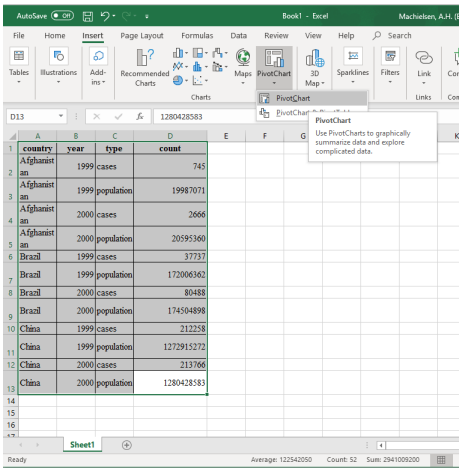
Data tidying is one of the most important skills for anyone intending to work with, and analyze, data. Data tidying involves the process of going from 'untidy' data, which can be in many formats, to tidy data, which can only be in one format: tidy data is data in which separate observations are stored in rows, and separate variables are stored in columns (Wickham et al. (2014)). As an example of untidy data, let us take an example from *R for Data Science* (Wickham and Grolemund, 2016):

country	year	type	count
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

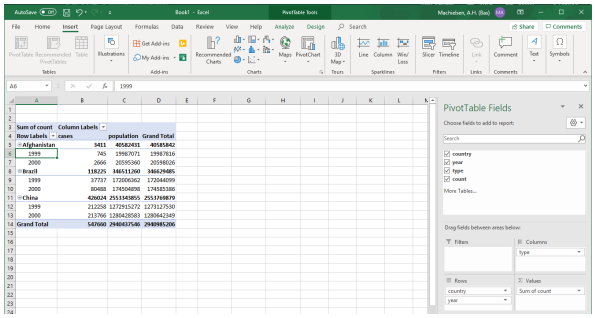
This dataset contains three countries, Afghanistan, China, and Brazil, their estimated population, and the number of cases (presumably, of a disease). This dataset is untidy because although every observation is in a separate row, not every variable is stored in a separate column. Both cases and population are stored in one variable. How would we go about this problem in Microsoft Excel? There are two alternatives. We could do it by hand, which is possible in case of tiny datasets like these, but what if the sample would contain many more countries? Secondly, we could use Excel's pivot function, which would encompass the following steps: we start out with the dataset in Excel, as depicted in figure 1.



Then, we should select the entire table, go to *Insert*, and select *PivotChart*, then *PivotChart and PivotTable*, as depicted in figure 1.



Then, we get the menu depicted in figure 3, in which we should select the intricate combination of existing variables leading to a recomposition of the table cells that at best requires loads of copy-and-pasting before you have a clean and tidy dataset.



Now, let's contrast this approach to an approach in R. We start up Rstudio, load the tidyverse (a set of packages which facilitate data tidying and other data treatment), and execute the following commands (the data as previously shown is loaded as *table2*):

```
library(tidyverse)

pivot_wider(table2,
    names_from = "type",
    values_from = "count"
    )
```

We immediately obtain the following table, which is a tidy dataset. Each observation is stored in a row, and each variable (each separate piece of information) is stored in a column.

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

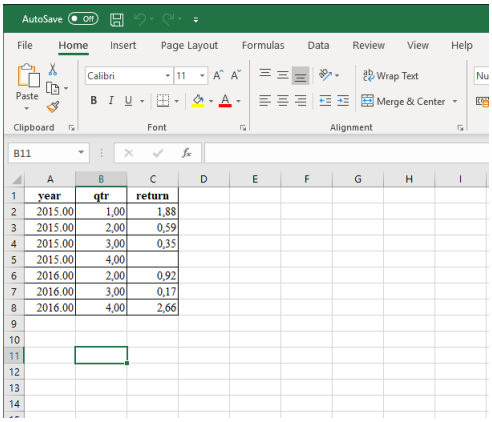
So far for tidying data. It seems clear that R is much more user-friendly, and also much faster. Let us now proceed with the second aspect, by subsetting.

2 By subsetting

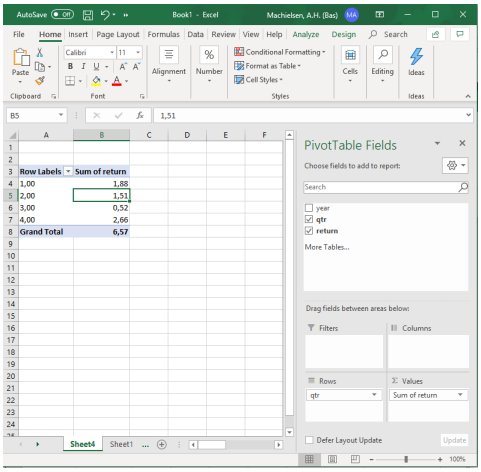
By subsetting refers to the process of summing, averaging, or computing any other statistic of a variable X per group/category of a variable Y. As an example, let us look at the following dataset:

year	qtr	return
2015	1.00	1.88
2015	2.00	0.59
2015	3.00	0.35
2015	4.00	NA
2016	2.00	0.92
2016	3.00	0.17
2016	4.00	2.66

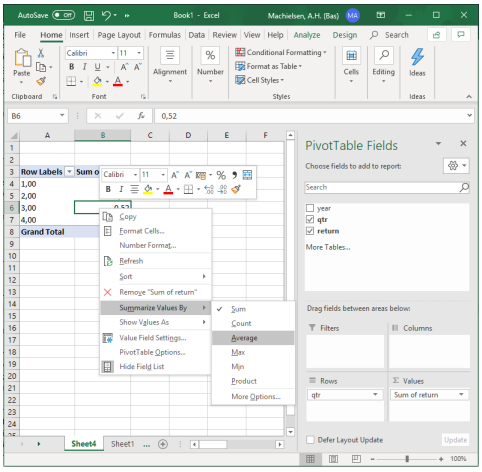
The dataset contains stock returns on a certain portfolio per quarter. If one looks at the data attentively, it is noticeable that there is one explicit NA, but there is also an implicit NA. I have added this for fun, to see how both R and Excel handle this. Suppose we want to know the average return per quarter. How do we go about this in Excel? First, we start off with the dataset in Excel. We have to make sure that Excel recognizes the data as numbers, which is always a tedious undertaking. In this case, we must find all dots and replace them by comma's to have Excel recognize them as numbers. This starting position is shown in the following figure:



Then, we once again apply pivot table, and we end up with the following menu, in which we want to categorize the quarters and summarise the returns:



So now we have the quarters where they belong, but Excel gives us the sum of both metrics by default, and not the average! Fortunately, we can still do this, by right-clicking in the table on one of the sum of the returns (see figure 2), we select *Summarize Values By*, and then select *Average*. Finally, we have obtained one simple group_by statistic using Excel.



Let us now do the same job in R. We again use the tidyverse, the aforementioned set of packages. More specifically, we use dplyr, which is specifically designed to 'ply' the data into the most commonly used forms, including subsetting. As we've loaded the packages before, there is no need to do it again. We can start right-off by instructing R to ply the data according to a group using the `group_by` command, and then instructing R to take the average:

```
stocks %>%
    group_by(qtr) %>%
    summarise(average =
        mean(return,
             na.rm = TRUE)
             )
```

In this chunk, I make use of R's 'piping' operator (%>%), which basically tells you to use the results of the previously entered commands to the next one. In particular, I tell R to take the stocks dataset, then group it by quarter (using `dplyr`'s `group_by`), and then summarise over the previous output (i.e. the dataset grouped by quarter). `Summarise` is very general, so in this example I took the mean, but I can take the median, standard deviation, maximum, minimum, or any combination of them, according to my liking. R also allows for more customization by specifying to remove NA's, instead of automatically removing them and offering no option, as Excel does. This is the output that I obtain:

qtr	average
1.00	1.88
2.00	0.76
3.00	0.26
4.00	2.66

group_by also enables the user to group the data according to multiple categorical variables, so you could compute means, medians, etc. for every group1-group2 combination. Finally, by specifying average in the previous chunk, I specified the name I wanted to give to my variable, something which is also difficult to do in Excel.

3 Merging data

The next part involves data merging. Merging data involves combining observations for the same units (for example, country-year units) on different variables in one new data frame. For example, I might have a time-series of government spending over GDP in the Netherlands and Germany from 1870-1940, and I might have another time series of educational spending in those two countries. I want to connect each country-year observation from the first dataset to the second, so I end up with a dataset with both variables. Now, let's take two datasets from Clio-Infra, a website featuring datasets frequently used in (cross-country) economic history analyses. To break with tradition, I will first show how to do this in R, which turns out to be extremely easy.

First, I load two datasets from Clio-infra, average years of education (educ), and GDP per capita (gdp). The datasets look as follows:

```
head(educ, 5)
```

ccode	country.name	year	value
528.00	Netherlands	1820.00	2.00
826.00	United Kingdom	1820.00	1.76
528.00	Netherlands	1830.00	2.00
826.00	United Kingdom	1830.00	1.93
528.00	Netherlands	1840.00	2.50

```
head(gdp, 5)
```

ccode	country.name	year	value
56.00	Belgium	1500.00	1467.00
818.00	Egypt	1500.00	680.00
276.00	Germany	1500.00	1146.00
380.00	Italy	1500.00	1532.85
528.00	Netherlands	1500.00	1454.00

Very nice! Both datasets are tidy: countries, years, and values of gdp and educ respectively are stored separately. Now, let's see how easy it is to merge these two datasets.

```
merge(gdp, educ,
    by.x = c("country.name",
             "year",
             "ccode"),
    by.y = c("country.name",
             "year",
             "ccode")
  )
```

That's right! We use the command merge, we specify the variables which we need to match for both the first dataset (by.x) and the second dataset (by.y), and R automatically matches all observations. The output (I called it merge) looks as follows:

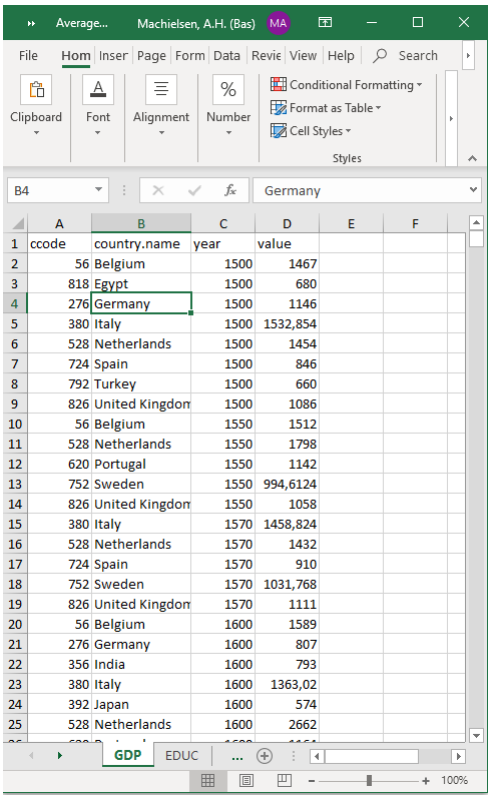
```
head(merge, 5)
```

country.name	year	ccode	value.x	value.y
Afghanistan	1950.00	4.00	645.00	0.22
Afghanistan	1960.00	4.00	739.00	0.31
Afghanistan	1970.00	4.00	709.00	0.62
Afghanistan	1980.00	4.00	690.00	1.09
Afghanistan	1990.00	4.00	604.00	1.60

Now, let's see how to do this in Excel. This will be much more difficult, as Excel has no built-in merge function. Excel only has VLOOKUP, which can be very useful, but is challenging to use if you want to match according to multiple variables. Oftentimes, the user is forced to generate a 'key' (a country-year variable in this example) to be able to match observations. This is unnecessary and user-unfriendly, in contrast to the straightforward approach just enunciated in R.

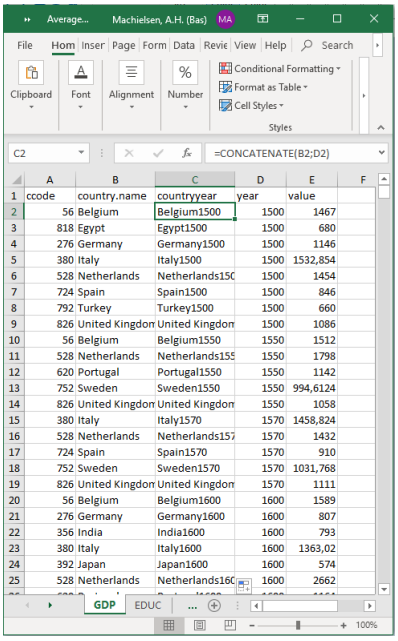
First, in Excel, it is useful to have both datasets in one Excel session, so we have to copy and paste one dataset into a separate sheet in the other file. In figure 7, we can see that I've pasted one dataset

('GDP') in the file of the other, but on a separate sheet. Then, we have to start making a concatenated variable in both dataset of country-year, so as to allow VLOOKUP match the variables later. VLOOKUP can only match on the basis of one variable, unlike R's merge function, so we have to concatenate all relevant variables for merging into one variable.



	A	B	C	D	E	F
1	ccode	country.name	year	value		
2	56	Belgium	1500	1467		
3	818	Egypt	1500	680		
4	276	Germany	1500	1146		
5	380	Italy	1500	1532,854		
6	528	Netherlands	1500	1454		
7	724	Spain	1500	846		
8	792	Turkey	1500	660		
9	826	United Kingdon	1500	1086		
10	56	Belgium	1550	1512		
11	528	Netherlands	1550	1798		
12	620	Portugal	1550	1142		
13	752	Sweden	1550	994,6124		
14	826	United Kingdon	1550	1058		
15	380	Italy	1570	1458,824		
16	528	Netherlands	1570	1432		
17	724	Spain	1570	910		
18	752	Sweden	1570	1031,768		
19	826	United Kingdon	1570	1111		
20	56	Belgium	1600	1589		
21	276	Germany	1600	807		
22	356	India	1600	793		
23	380	Italy	1600	1363,02		
24	392	Japan	1600	574		
25	528	Netherlands	1600	2662		

We insert a new column, and use the CONCATENATE function to concatenate the string of cell B2 with that of D2, with this command: =CONCATENATE(B2;D2). We then click on the small dot below to expand the command to all observations in GDP. Then, we repeat this procedure in EDUC. We end up with the following extra column in both datasets.

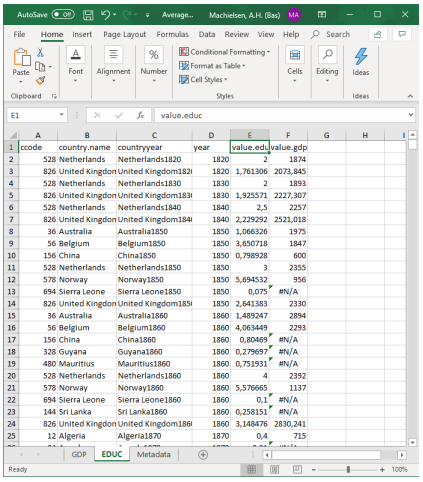


	A	B	C	D	E	F
1	ccode	country.name	countryyear	year	value	
2	56	Belgium	Belgium1500	1500	1467	
3	818	Egypt	Egypt1500	1500	680	
4	276	Germany	Germany1500	1500	1146	
5	380	Italy	Italy1500	1500	1532,854	
6	528	Netherlands	Netherlands150	1500	1454	
7	724	Spain	Spain1500	1500	846	
8	792	Turkey	Turkey1500	1500	660	
9	826	United Kingdon	United Kingdon	1500	1086	
10	56	Belgium	Belgium1550	1550	1512	
11	528	Netherlands	Netherlands155	1550	1798	
12	620	Portugal	Portugal1550	1550	1142	
13	752	Sweden	Sweden1550	1550	994,6124	
14	826	United Kingdon	United Kingdon	1550	1058	
15	380	Italy	Italy1570	1570	1458,824	
16	528	Netherlands	Netherlands157	1570	1432	
17	724	Spain	Spain1570	1570	910	
18	752	Sweden	Sweden1570	1570	1031,768	
19	826	United Kingdon	United Kingdon	1570	1111	
20	56	Belgium	Belgium1600	1600	1589	
21	276	Germany	Germany1600	1600	807	
22	356	India	India1600	1600	793	
23	380	Italy	Italy1600	1600	1363,02	
24	392	Japan	Japan1600	1600	574	
25	528	Netherlands	Netherlands160	1600	2662	

Then, we take either one of the datasets as our 'base' dataset (I use EDUC), create a new column (and call it GDP, for example), and then apply VLOOKUP in the following way:

```
=VLOOKUP(C2;GDP!C:E;3;FALSE)
```

where C2 refers to our 'key' column, a country-year observation. We can then extend this by clicking on the right dot at the lower right of the first cell. The syntax in VLOOKUP is confusing and user-unfriendly: first, you have to specify the variable on the basis of which you want to match (in this case the newly-created key, e.g. Netherlands1880), then, you specify the *array* over which you want to match. This array *must* have the key as the first variable, otherwise it will not work. Then, you specify the *relative* column from which you want to retrieve the matched value (which, in our case, is the 3rd column relative to the key column), and you specify approximate matching as FALSE, because approximate matching in Excel works very badly. Only in this way you can match variables on a one-by-one basis in Excel. This is about as stupid as it gets. Finally, you will end up with the following dataset, after having done a lot more work than you would have done if you would've used R:

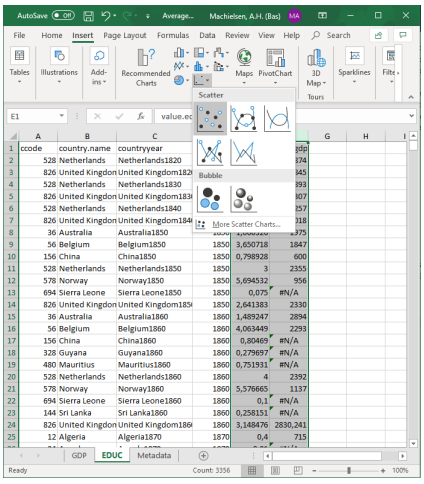


Next, we will talk about one of the most fun aspects of data treatment: creating elementary graphs.

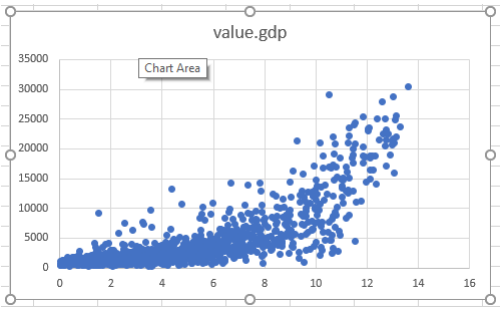
4 Graphs

Admittedly, it is quite easy to make graphs in Excel. However, Excel-generated graphs are notoriously ugly. In this section, I attempt to show that graphs using the ggplot system in R are *as* easy, if not easier, to create as those in Excel, and I will demonstrate that they are much prettier, and give more oversight, even when using the same underlying data. Let us use the newly-created dataset in the previous section, to attempt to construct a decent graph in Excel.

We start off by constructing a graph depicting the relationship between education and GDP. Do we find that better educated countries (or technically, country-years) have a higher GPD than lower educated countries?



The answer is.. yes! But the graph is quite ugly.

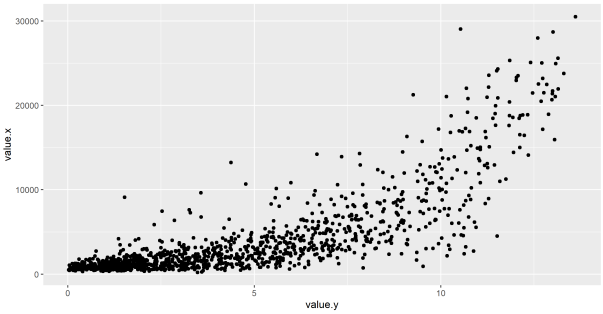


Let us now see how to do this in R. Could it be any easier than this? Probably not, but it is not much harder, and the graphs are prettier, and it is easier to customize them in a way which is suitable for presentations or journal articles, as I demonstrate below.

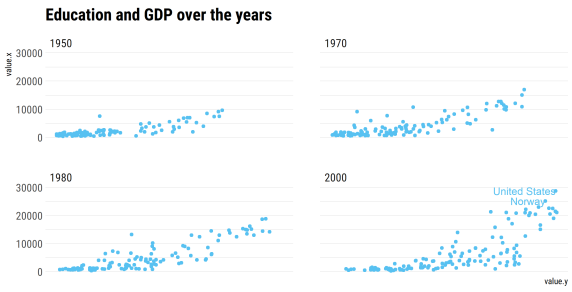
Remember that we had previously loaded `merge` into our R environment, so we can just call it using the basic `qplot` function from the ggplot2 package, which generates a default plot of the variables you specify.

```
qplot(data = merge,
      value.y,
      value.x)
```

You specify the dataset you use, the x variable, and the y variable (in this case, value.y is education and value.x is gdp), and `qplot` automatically generates a plot for you. The plot is shown below:



There are many more options through which you can customize your graphs in the ggplot2 package. As a demonstration, I attempt to use some different aesthetics to showcase the features of ggplot. I filter the dataset to only include observations from 1950, 1970, 1980 and 2000 to explore differences in relationship between education and GDP over time:



For the code, see the appendix.

5 String manipulation

In the final section, I want to briefly discuss string manipulation. String manipulation can be very easy in Excel, by using the `LEFT` and `RIGHT` commands to extract the left and right parts of a string. This is useful, for example, in the case of extracting a year from a long string. It is conceivable, however, that there are more tedious cases, and Excel offers no universal solution for them. In R, there is an integrated language called *Regular Expressions*, which is also supported in Python, Perl, and many other programming languages. Regular Expressions allow you to specify a particular pattern which you are looking for, so it comes in as extremely handly when extracting strings. Suppose we want to extract the year from the following string:

```
TOTAL_2018_""NL_ASSETS
```

In Excel, supposing the string were located in cell A1, we could do the following:

```
=MID(A1;7;4)
```

However, once we are unsure of the starting place and the length of the year (it could be a date), we would run into trouble. Alternatively, we could use R to match the same string. This could be done in the following way:

```
str_extract(str, "([0-9])+")
```

In which we specify we want to extract any sequence (indicated by `()+`) of numbers (indicated by `[0-9]`). Therefore, string matching is easier, more intuitive and more straightforward in R.

6 Conclusion

In this pamphlet, I compared five common data manipulation operations in Excel and R. I conclude that R is easier to use for these manipulations, and in the case it isn't, it is about as easy, but yields much better results. Thank you for reading.

A Code for Figure Education and GDP

The code which I used to generate the graph in section 4 is the following:

```
ggplot(data = merge2, aes(
    x = value.y, y = value.x)) +
geom_point() +
facet_wrap(~ year, nrow = 2) +
    geom_text(
        data = subset(merge2,
                      year == "2000"
                      & value.x > 25000),
            aes(value.y, value.x,
                label = country.name),
            hjust = "right") +
scale_x_continuous(breaks = NULL) +
ggtitle("Education and GDP over the years") +
theme_ipsum_rc()
```

References

Wickham, H. et al. (2014). Tidy data. *Journal of Statistical Software*, 59(10):1–23.

Wickham, H. and Grolemund, G. (2016). *R for data science: import, tidy, transform, visualize, and model data*. " O'Reilly Media, Inc.".