The final project - Animated changes of GDP PPP 1990-2018

Bartosz Czernecki

1/14/2020

Overall aim

The animated charts are increasingly interested in many social media podcasts. For example, the popularity among programming languages is shown in this link: https://www.youtube.com/watch?v=Og847HVwRSI

In the final project we will try to create an animated chart that will show us changes of Growth Domestic Product (GDP) standardized by Purchasing Power Paritity (PPP) in all countries that participate in our course. The values are given in USD.

Dataset

The World Bank publishes GDP PPP reports on annual basis (https://data.worldbank.org/indicator/NY.GDP.PCAP.PP.CD) for all countries around the world. This dataset has been pre-formated to the form readable in spreadsheet-like file.

Please download the file from http://iqdata.pl/dataprocessing/data/gdp_ppp.xlsx and save it in a known location on your laptop.

Reading dataset

Set your working directory to the location of downloaded file. Please use from the upper menu `SESSION` then `SET WORKING DIRECTORY` and `CHOOSE DIRECTORY`.

Activate library that allows reading "XLSX" format (e.g. `readxl`) and name the read data as `df` object. The structure of the created `df` object should be as shown below for the first 6 rows:

```
## # A tibble: 6 x 31
##   country code  `1990` `1991` `1992` `1993` `1994` `1995` `1996` `1997`
##   <chr>   <chr>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1 Aruba   ABW   24101. 25871. 26533. 27431. 28657. 28649. 28499. 30216.
## 2 Afghan... AFG      NA     NA     NA     NA     NA     NA     NA     NA
## 3 Angola  AGO    3090.  3120.  2908.  2191.  2196.  2496.  2795.  2953.
## 4 Albania ALB    2549.  1909.  1823.  2057.  2290.  2666.  2980.  2717.
## 5 Andorra AND      NA     NA     NA     NA     NA     NA     NA     NA
## 6 Arab W... ARB    6808.  6872.  7255.  7459.  7646.  7774.  8094.  8398.
## # ... with 21 more variables: `1998` <dbl>, `1999` <dbl>, `2000` <dbl>,
## #   `2001` <dbl>, `2002` <dbl>, `2003` <dbl>, `2004` <dbl>, `2005` <dbl>,
## #   `2006` <dbl>, `2007` <dbl>, `2008` <dbl>, `2009` <dbl>, `2010` <dbl>,
## #   `2011` <dbl>, `2012` <dbl>, `2013` <dbl>, `2014` <dbl>, `2015` <dbl>,
## #   `2016` <dbl>, `2017` <dbl>, `2018` <dbl>
```

Re-shaping data

In the final phase of the project we will use `ggplot2` package to create our charts. This package requires so called narrow format of the data frame. Therefore our data frame needs to be reshaped to the form where we will have 4-5 columns with: `country`, `code`, `year` and `gdp`.

It can be done *manually* or with the use of a pivot-like functions. One of my favourie is the `gather` function from the `tidyr` package, which probably will have to be installed on your laptop if not used before.

Activate the `tidyr` and `dplyr` (or `tidyverse`) libraries and launch the `?gather` command to familiarize yourself with the way it works. Try to reshape our `df` object to the following structure:

```
head(df2)
```

```
## # A tibble: 6 x 4
##   country     code  year     gdp
##   <chr>       <chr> <chr>  <dbl>
## 1 Aruba       ABW   1990  24101.
## 2 Afghanistan AFG   1990     NA
## 3 Angola      AGO   1990   3090.
## 4 Albania     ALB   1990   2549.
## 5 Andorra     AND   1990     NA
## 6 Arab World  ARB   1990   6808.
```

Hints: Use the `year` column as the **key** argument, and `value = "gdp"`. Save the obtained results as `df2`.

Choosing countries

Right now our data frame `df2` contains data for all countries. We'd like to limit our animation only to countries that participate in our class, i.e.: Czech Rep., Germany, Poland South Korea and Spain. Additionally, we'd like to add average for the entire world which is given as *WLD* in the column `code`.

Find out codes for the chosen countries and use the `filter` command to clip the dataset for these 6 codes. Fill in the following command and save results as `gdp_tidy`

```
gdp_tidy = filter(df2, code %in% c("COUNTRY_1", "COUNTRY_2", ...) )
```

The first 10 rows of our `gdp_tidy` should return:

```
## # A tibble: 10 x 4
##    country        code  year     gdp
##    <chr>          <chr> <chr>  <dbl>
##  1 Czech Republic CZE   1990  12660.
##  2 Germany        DEU   1990  19497.
##  3 Spain          ESP   1990  13664.
##  4 Korea, Rep.    KOR   1990   8273.
##  5 Poland         POL   1990   4450
##  6 World          WLD   1990   5498.
##  7 Czech Republic CZE   1991  11596.
##  8 Germany        DEU   1991  21032.
##  9 Spain          ESP   1991  14449.
## 10 Korea, Rep.    KOR   1991   9346.
```

Creating test data for a single year

In the next steps we will be trying to create a bar chart showing differences in GDP PPP / capita. For testing purposes we can clip our dataset to the last year (2018) available in the `gdp_tidy` object. Use again the `filter` command for column `year` where it is equal to 2018. Save the results as `test`.

```
## # A tibble: 6 x 4
##   country        code  year     gdp
##   <chr>          <chr> <chr>  <dbl>
## 1 Czech Republic CZE   2018  39744.
## 2 Germany        DEU   2018  53075.
```

```
## 3 Spain          ESP   2018  39715.
## 4 Korea, Rep.    KOR   2018  40112.
## 5 Poland         POL   2018  31337.
## 6 World          WLD   2018  17948.
```

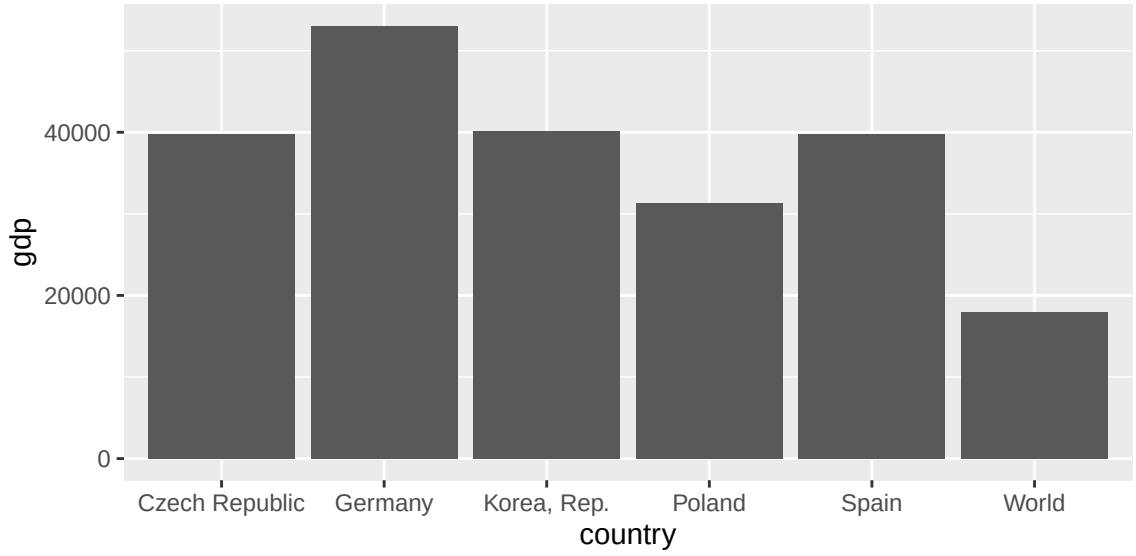
Plotting data with ggplot2 library

Intro

Activate the `ggplot2` package and try to create a basic plot with any of the chosen geom_. The list of available geoms can be found in the cheat-sheet (https://rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf). Let's start with *geom_bar* as it is the most generic geom_ type for data structured as our `test` data frame.

Fill in the following blank fields in a below attached pseudocode to create a chart that will be similar to our plot shown in a figure below:

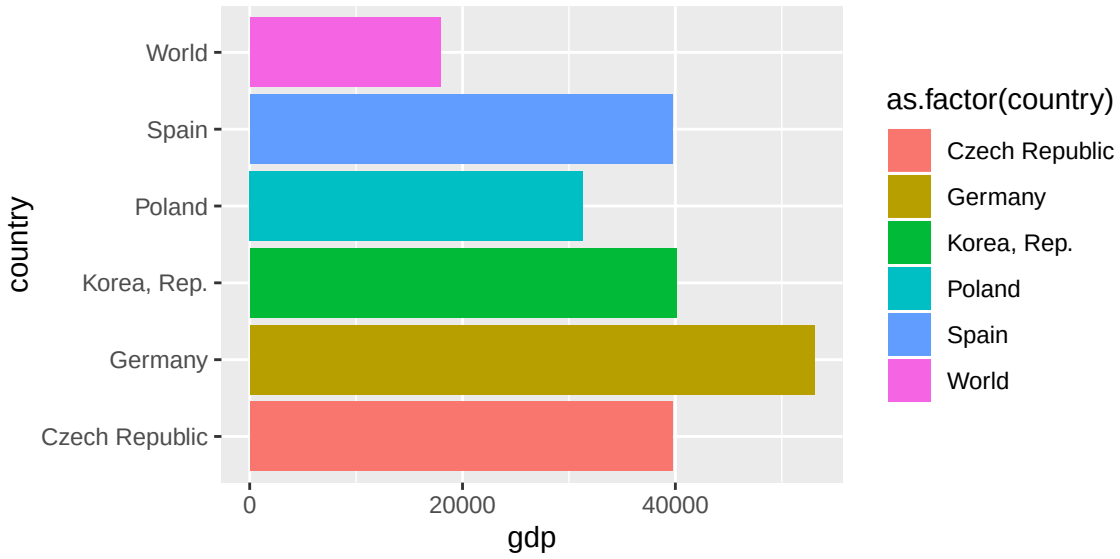
```
ggplot(test, aes(x = ________, y = ____ )) +
  geom_bar(stat = 'identity')
```



Plot customization

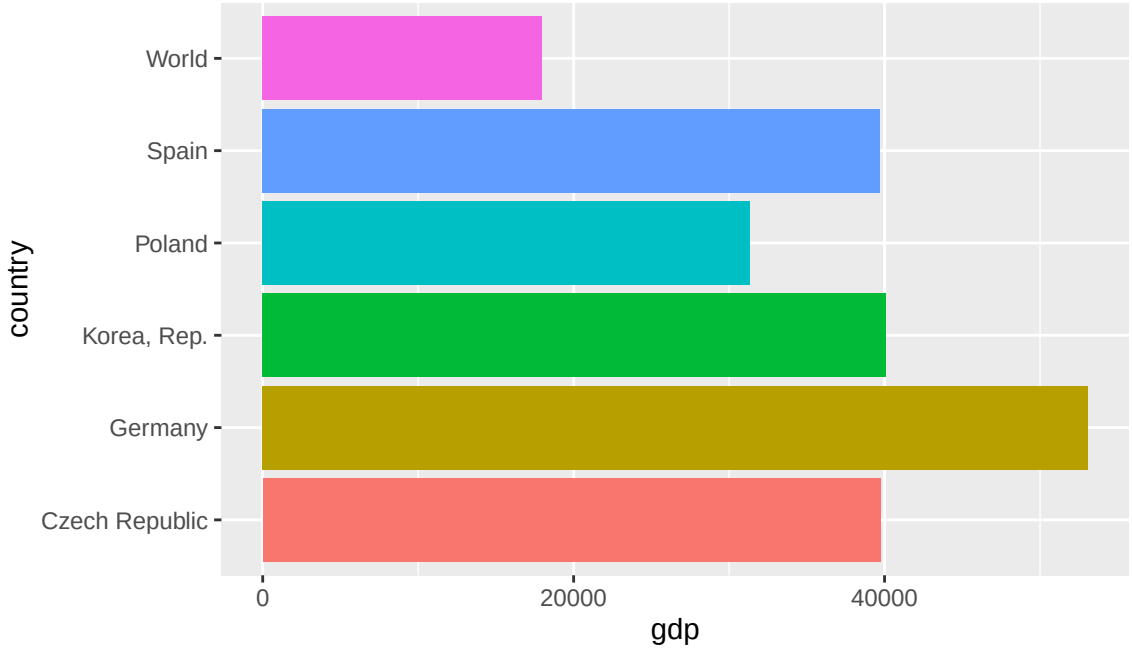
We can change the default plotting options by modyfying or adding extra arguments to the ggplot syntax. Try to add argument `fill = as.factor(country)` inside the `aes()` statement.

At the end of the created code add (i.e. with `+`) `coord_flip()` function which rotates the chart by 90 degrees. The results should be similar to the below chart:



Removing legend

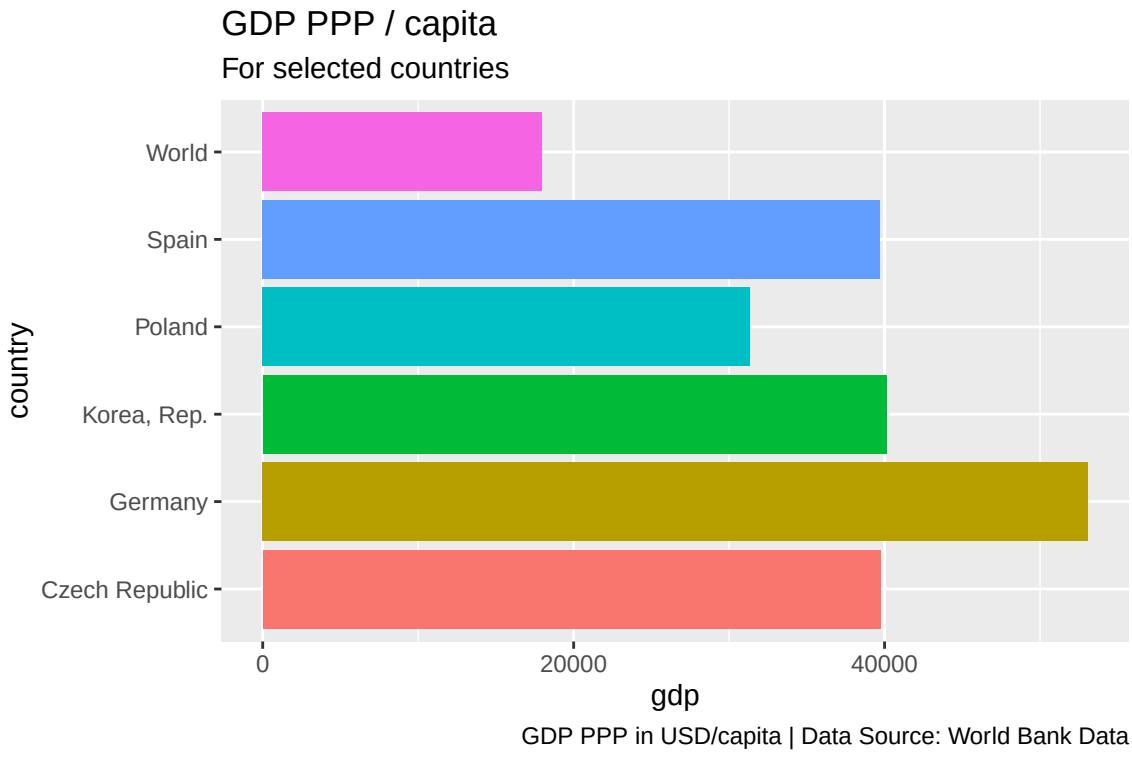
The default legend is redundant as we only show country names which are also labeled on the Y axis. Legend as many other ggplot items are controlled by the `theme()` function. In order to get rid off the legend please add to the plotting syntax: `theme(legend.position = "none")`



Titles & Labels

Our chart miss titles and labels. We can add them with `labs()` function at the end of `ggplot` syntax. For example:

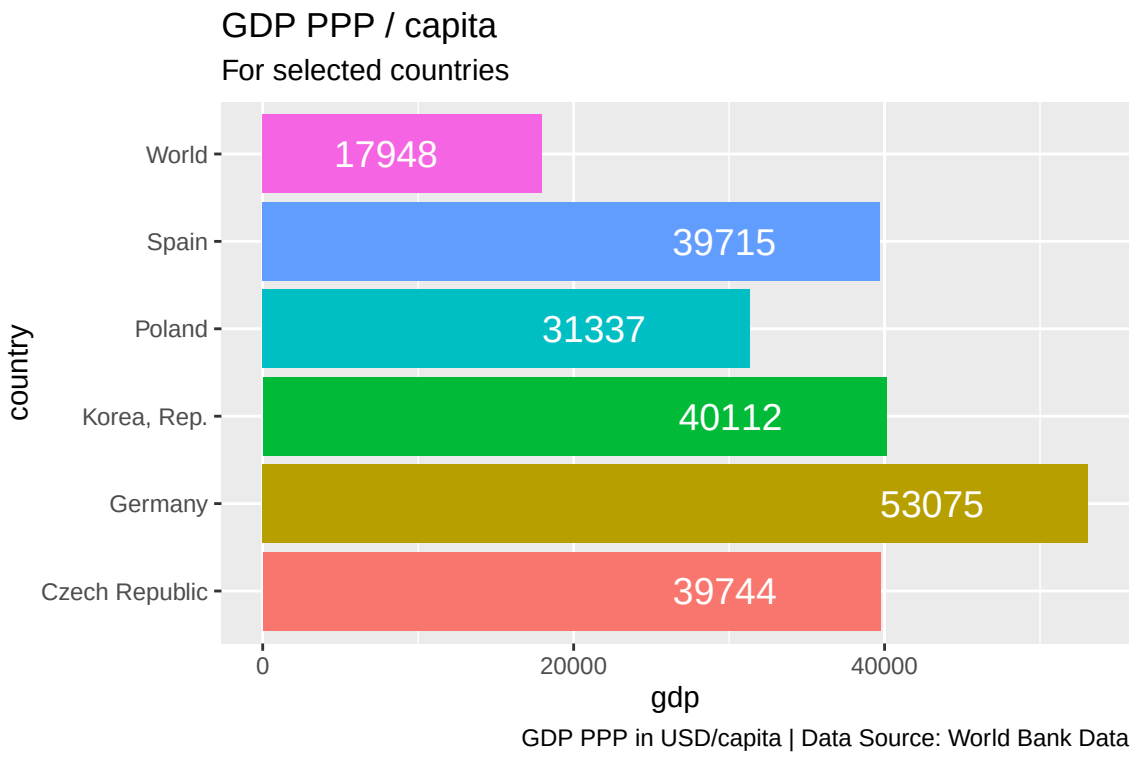
```
labs(title = 'GDP PPP / capita',
       subtitle  =  "For selected countries",
       caption  = "GDP PPP in USD/capita | Data Source: World Bank Data")
```



Adding text

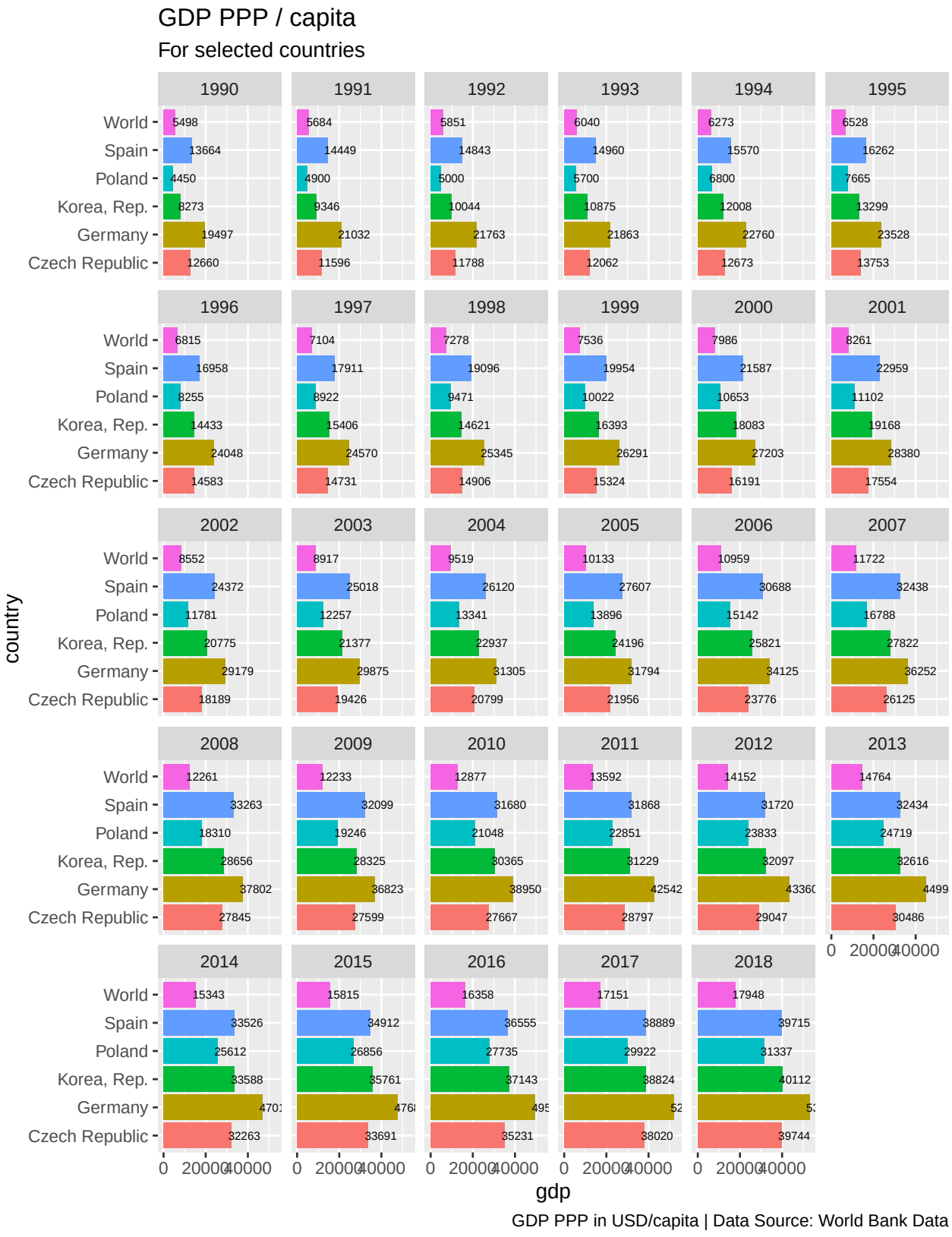
We can also add another `geom_` to plot to show as text/label the real (or rounded) value for GDP that will be visible on the height of each bar. Try to play a bit with the `geom_text()` template given below and modify it to get the exemplary result:

```
geom_text(aes(y = ____, label = _____), hjust = 2, size= 5, col = '______')+
```



Plotting the full dataset

So far we have worked on a clipped dataset for year 2018 only. We can look at the whole dataset in a similar way by creating a separate charts for each individual year. It can be very easily done with the `facet_grid()` or `facet_wrap()` functions that are added to the created grammar/template of our chart. As the argument for this command please use the column name that will be used for the division with tilde (~) as prefix. Do not forget to change the `test` data frame with the full dataset.


GDP PPP / capita
For selected countries
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
World
Spain
Poland
Korea, Rep.
Germany
Czech Republic
country
0
20000
40000
gdp
GDP PPP in USD/capita | Data Source: World Bank Data

Making animation with gganimate package

Data Pre-processing

Right now our geom_bar always displays the same order for all selected countries. However, there are years when countries rank can change. Therefore we will have to extend our data frame by one more column describing current rank for our countries.

There are plenty of possibilities that allow us to create rank column, but the quickest one might be coupled application of `group_by()` and `arrange()` functions. We can also add one more column that will be used for labeling that has country name and GDP glued together in one field.

```
library(dplyr)
gdp_tidy2 = gdp_tidy %>% group_by(year) %>%   arrange(year, gdp)
gdp_tidy2$rank = 6:1
gdp_tidy2$label = paste(gdp_tidy2$country, round(gdp_tidy2$gdp))
tail(gdp_tidy2)
```

```
## # A tibble: 6 x 6
## # Groups:   year [1]
##   country        code  year     gdp  rank label
##   <chr>          <chr> <chr>  <dbl> <int> <chr>
## 1 World          WLD   2018  17948.     6 World 17948
## 2 Poland         POL   2018  31337.     5 Poland 31337
## 3 Spain          ESP   2018  39715.     4 Spain 39715
## 4 Czech Republic CZE   2018  39744.     3 Czech Republic 39744
## 5 Korea, Rep.    KOR   2018  40112.     2 Korea, Rep. 40112
## 6 Germany        DEU   2018  53075.     1 Germany 53075
```

Creating animation

Required packages

The same template as we used previously with `geom_bar()` can be applied for creating animation with `gganimate` package. This package is based on other packages, so please install the `png`, `gifski` and other required dependencies.

Plot template

The previously created template can be now used with small modification, which will include the rank position. This might be dane by adding `as.factor(rank)` to the `aes()` introductory part of the code.

In the `geom_text()` we can use our label column that will show country name and value of GDP.

Setting up for animiation

One new thing that we will be added to the plotting code is the `transition_states()`, that is responsible for creating multiples slides of animation. The first argument should be given as unquoted name of column that will be responsible for creating new state. There are more parameters to be set. We will set arguments for: transition_states, transition_length and state length.

We can also modify the `labs()` to show in the title a current year on animation. If we put name **closest_state** in the curly brackets it will be read as value set in the `transition_states` above (i.e. year in our case). When all is set we can save the syntax as new object (e.g. **anim**).

This is how the code for animation setup might look like:

```
library(gganimate)
anim = ggplot(gdp_tidy2, aes(x = as.factor(rank), y = gdp, fill = as.factor(country))) +
```

```
  geom_bar(stat = 'identity') +
  coord_flip() +
  geom_text(aes(y = gdp, label = label), hjust = 2, size= 5, col = 'white')+
  theme(axis.text=element_text(size=14),
        legend.position = "none") +
    transition_states(states = year,
    transition_length = 2,
    state_length = 1) +
  labs(title = 'GDP PPP per Year : {closest_state}',
       subtitle  =  "For selected 6 countries",
       caption  = "GDP PPP in USD/capita | Data Source: World Bank Data")
```

Rendering animation

There are at least to way to generate our animation. The most popular are GIFs and MP4 files. Let's start with the first one.

The created previously `anim` object can be used as first argument for `animate()` function that will require setting up a list of arguments such as: number of frames to be rendered, fps, duration, render engine, resolution, file name, etc.. We will focus only on the most elementary settings that can be used with gifski package.

If your computer has ffmpeg codecs / drivers available (typically pre-installed on Linux and Mac computers) you may also try to create MP4 file:

```
#install.packages("gifski")
library(gifski)
animate(anim, 100, fps = 10,  width = 1200, height = 1000,
        renderer = gifski_renderer("gganim.gif"))

# For MP4
animate(anim, 200, fps = 10,  width = 900, height = 750,
        renderer = ffmpeg_renderer()) -> for_mp4

anim_save("animation.mp4", animation = for_mp4 )
```

Improving animation layout

The `geom_bar()` was not created with an intention to be used for animated plots. We can replace `geom_bar` statement with `geom_tile()` that is more appropriate for this kind of tasks.

```
  geom_tile(aes(y = gdp/2, height = gdp, width = 0.9), alpha = 0.8, color = NA)
```

We can also add labels for countries with `geom_text()` on the height of GDP = 0:

```
geom_text(aes(y = 0, label = country), vjust = 0.2, hjust = 1, size=3)
```

The whole procedure for creating animation is exactly the same as done previously, i.e. creating `anim` object, rendering animation and saving as GIF/MP4 file. Try to run the modified code and play the created animation.

The final code:

I have prepared some further modification to `theme()` function that modifies axis, labels, fonts, etc... We haven't covered this settings during our intro to the `ggplot` package. However, you can try to get the same animation with the below attached code:

```
anim = ggplot(gdp_tidy2, aes(rank, group = country,
                fill = as.factor(country), color = as.factor(country))) +
  geom_tile(aes(y = gdp/2, height = gdp, width = 0.9), alpha = 0.8, color = NA) +
  geom_text(aes(y = 0, label = country), vjust = 0.2, hjust = 1, size=5) +
  geom_text(aes(y=gdp,label = round(gdp), hjust=0, size = 6)) +
  coord_flip(clip = "off", expand = FALSE) +
  scale_y_continuous(labels = scales::comma) +
  scale_x_reverse() +
  guides(color = FALSE, fill = FALSE) +
  theme(axis.line=element_blank(),
        axis.text.x=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks=element_blank(),
        axis.title.x=element_blank(),
        axis.title.y=element_blank(),
        legend.position="none",
        panel.background=element_blank(),
        panel.border=element_blank(),
        panel.grid.major=element_blank(),
        panel.grid.minor=element_blank(),
        panel.grid.major.x = element_line( size=.1, color="grey" ),
        panel.grid.minor.x = element_line( size=.1, color="grey" ),
        plot.title=element_text(size=25, hjust=0.5, face="bold", colour="grey", vjust=-1),
        plot.subtitle=element_text(size=18, hjust=0.5, face="italic", color="grey"),
        plot.caption =element_text(size=12, hjust=0.5, face="italic", color="grey"),
        plot.background=element_blank(),
        plot.margin = margin(2,2, 2, 4, "cm")) +
  transition_states(year, transition_length = 4, state_length = 1) +
  view_follow(fixed_x = TRUE)  +
  labs(title = 'GDP PPP per Year : {closest_state}',
       subtitle  =  "Chosen 5 countries on the background of world average",
       caption  = "GDP PPP in USD/capita | Data Source: World Bank Data")

# alternatively use GIF animation if MP4 is not supported on your machine:
for_mp4 = animate(anim, 200, fps = 10,  width = 900, height = 750,
        renderer = ffmpeg_renderer())
anim_save("animation.mp4", animation = for_mp4 )
```

Hint: - this code can be copy-paste from this link: http://iqdata.pl/dataprocessing/data/final.R