Intermediate tidyverse exercises

RUG @ HSG

Below, you will find some exercises to apply and improve your tidyverse skills. Generally, they increase in difficulty as you go along. Don't hesitate to ask questions in the Q&A WhatsApp group, we or your fellow students will be happy to answer them!

The data set

In these exercises we will be working with the `gapminder` data set. It includes historical data on the life expectancy, population, and GDP in different countries over the past decades. You can load it directly from the `gapminder` package like this:

```r
install.packages("gapminder")
gapminder <- gapminder::gapminder
```

Task 1: Getting an overview over the data

(a) In `tidyverse` there are multiple functions that allow you to gain a quick overview over the data that you are working with. Use one of them to find out:

- How many rows and columns are in the data
- Which variable types are in the data

(b) Find out for which time frame the data is available.

(c) Count the number of countries represented per continent. (*Hint:* the function `n_distinct()` gives you the number of unique values in a vector)

Task 2: Descriptive statistics

(a) We have the GDP per capita for a lot of countries, but how do they compare per continent? Calculate the mean GDP per capita for every continent in the year 2007. The mean GDP can be misleading if the distribution over the countries is skewed. Also add the median GDP per capita.

(b) The economic development of these continents seems to be quite different. Calculate the minimum, median, and maximum life expectancy per continent in 1952. Sort them from lowest to highest median life expectancy.

(c) The table from task (b) is great but a bit hard to read at a glance. Calculate the distance from the minimum life expectancy to the median and from the median to the maximum in the year 1952.

Task 3: More in-depth analysis

(a) So far we have compared certain variables in one year, now we would like to know how they developed over time. Calculate the median GDP per capita for every continent for every year since 1980. To make the resulting table easier to interpret, transform it to wide format.

Hint: Instead of typing out all years since 1980, try to use a suitable logical condition when filtering.

(b) We have looked at data on the continent level but how do things look in Switzerland (or your home country)? Extract the GDP per capita, population and life expectancy in Switzerland for all years since 1980. To make the table more readable, display the GDP per capita in thousands, the population in millions, and the life expectancy in years past 50. Additionally, round the population values to one decimal. Transform your result into wide format, so that there is a row per calculated value and a column per year.

Hint: You can use the pipe operator within another "pipe chain", e.g., to pass a value to the `round()` function.
Hint: To display your result in wide format, you will have to transform to a "real" long format first, before transforming to wide format.

(c) Now that we have analyzed Switzerland (or your home country), let's compare it to the rest of Europe (or your home continent). Assemble a table with one column for Switzerland's GDP per capita for every year in thousands and one column with Europe's (excluding Switzerland) median GDP per capita for every year in thousands.

Hint: You can use the `ifelse()` function to logically fill a column, i.e., to determine if a country is Switzerland or rest of Europe. Type `?ifelse` in the console to see how it works.

If you made it this far, we commend you! That was some serious data wrangling just then. You now have some important skills to apply to your own data analyses!

Solutions

Task 1

(a) There is more than one answer to this task, however, we prefer `glimpse()` because the overview is nice and condensed.

```
gapminder %>% glimpse()
```

```
## Rows: 1,704
## Columns: 6
## $ country   <fct> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", ~
## $ continent <fct> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, ~
## $ year      <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 1997, ~
## $ lifeExp   <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, 38.438, 39.854, 40.8~
## $ pop       <int> 8425333, 9240934, 10267083, 11537966, 13079460, 14880372, 12~
## $ gdpPercap <dbl> 779.4453, 820.8530, 853.1007, 836.1971, 739.9811, 786.1134, ~
```

We see that there are `1704` rows and `6` columns in the dataframe. We have `factor`, `integer` and `double` data types.

(b) We can use the `summarize()` function for this:

```
gapminder %>%
  summarize(starting_year = min(year),
            ending_year = max(year))
```

```
## # A tibble: 1 x 2
##   starting_year ending_year
##           <int>       <int>
## 1          1952        2007
```

(c) If we group by continent first, we can use the `n_distinct()` function to give us the number of countries per continent.

```
gapminder %>%
  group_by(continent) %>%
  summarize(nr_countries = n_distinct(country))
```

```
## # A tibble: 5 x 2
##   continent nr_countries
##   <fct>            <int>
## 1 Africa              52
## 2 Americas            25
## 3 Asia                33
## 4 Europe              30
## 5 Oceania              2
```

Task 2

(a) To analyze only the year 2007 and not the entire time frame, we first filter for the year 2007 before we group the countries.

```
gapminder %>%
  filter(year == 2007) %>%
  group_by(continent) %>%
  summarize(mean_gdpPercap = mean(gdpPercap),
            median_gdpPercap = median(gdpPercap))
```

```
## # A tibble: 5 x 3
##   continent mean_gdpPercap median_gdpPercap
##   <fct>              <dbl>            <dbl>
## 1 Africa              3089.            1452.
## 2 Americas           11003.            8948.
## 3 Asia               12473.            4471.
## 4 Europe             25054.           28054.
## 5 Oceania            29810.           29810.
```

(b) Similarly to the task before, we can calculate these values by filtering, grouping, and summarizing.

```
gapminder %>%
  filter(year == 1952) %>%
  group_by(continent) %>%
  summarize(min_lifeExp = min(lifeExp),
            median_lifeExp = median(lifeExp),
            max_lifeExp = max(lifeExp)) %>%
  arrange(median_lifeExp)
```

```
## # A tibble: 5 x 4
##   continent min_lifeExp median_lifeExp max_lifeExp
##   <fct>           <dbl>          <dbl>       <dbl>
## 1 Africa           30             38.8        52.7
## 2 Asia             28.8           44.9        65.4
## 3 Americas         37.6           54.7        68.8
## 4 Europe           43.6           65.9        72.7
## 5 Oceania          69.1           69.3        69.4
```

(c) We can pass more complex calculations with more than one function to `summarize`.

```
gapminder %>%
  filter(year == 1952) %>%
  group_by(continent) %>%
  summarize(neg_dev_lifeExp = median(lifeExp) - min(lifeExp),
            pos_dev_lifeExp = max(lifeExp) - median(lifeExp))
```

```
## # A tibble: 5 x 3
##   continent neg_dev_lifeExp pos_dev_lifeExp
##   <fct>               <dbl>           <dbl>
## 1 Africa               8.83          13.9
## 2 Americas            17.2           14.0
## 3 Asia                16.1           20.5
## 4 Europe              22.3            6.77
## 5 Oceania              0.135          0.135
```

Note: The deviation in Oceania is so small because there are only two countries included from this continent (Australia and New Zealand). We know this from our calculation in task 1 (c).

Task 3

(a) You already know how to calculate the medians from previous tasks. Additionally, we can transform the data to wide format with `pivot_wider()`.

```
gapminder %>%
  filter(year >= 1980) %>%
  group_by(continent, year) %>%
  summarize(median_gdpPercap = median(gdpPercap)) %>%
  pivot_wider(names_from = year,
              values_from = median_gdpPercap)
```

```
## `summarise()` has grouped output by 'continent'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 5 x 7
## # Groups:   continent [5]
##   continent `1982` `1987` `1992` `1997` `2002` `2007`
##   <fct>      <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1 Africa     1324.  1220.  1162.  1180.  1216.  1452.
## 2 Americas   6435.  6361.  6619.  7114.  6995.  8948.
## 3 Asia       4107.  4106.  3726.  3645.  4091.  4471.
## 4 Europe    15323. 16215. 17550. 19596. 23675. 28054.
## 5 Oceania   18555. 20448. 20894. 24024. 26939. 29810.
```

Note: The logical operator >= is equivalent to the ≥ sign. Similarly, we can use <= to signify ≤. We can also use the logical "not-equal-to" operator != to mean ≠.

(b) To preserve the year variable, we use `transmute()`. Of course it is perfectly fine to use `select()` first and then `mutate`. In order to achieve the rounding to one decimal, we pass the population values to `round()` with a pipe operator. Pipe in a pipe!

```
gapminder %>%
  filter(country == "Switzerland" & year >= 1980) %>%
  transmute(year,
            k_gdpPercap = gdpPercap / 1e3,
            mio_pop = (pop / 1e6) %>% round(digits = 1),
            lifeExp_past_50 = lifeExp - 50) %>%
  pivot_longer(cols = c(k_gdpPercap, mio_pop, lifeExp_past_50),
               names_to = "variables",
               values_to = "values") %>%
  pivot_wider(names_from = year,
              values_from = values)
```

```
## # A tibble: 3 x 7
##   variables       `1982` `1987` `1992` `1997` `2002` `2007`
##   <chr>            <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1 k_gdpPercap       28.4   30.3   31.9   32.1   34.5   37.5
## 2 mio_pop            6.5    6.6    7      7.2    7.4    7.6
## 3 lifeExp_past_50   26.2   27.4   28.0   29.4   30.6   31.7
```

Note: The `1e3` and `1e6` are scientific notation for 1'000 and 1'000'000. By using this notation, you can avoid mistakes coming from typing too many or too few zeros.

(c) We solve this task with a "helper" column called `region` which we create with the `ifelse()` function. We can then group by these different regions. If you solved this by first saving the Europe column in a separate variable, this is of course completely valid. This applies to coding in general: everyone has their own preferred way to do things and everyone will defend that approach as if their life depended on it.

```
gapminder %>%
  filter(continent == "Europe") %>%
  mutate(region = ifelse(country == "Switzerland", "Switzerland", "Europe")) %>%
  group_by(region, year) %>%
  summarise(k_median_GDP = median(gdpPercap) / 1e3) %>%
  pivot_wider(names_from = "region", values_from = "k_median_GDP")
```

```
## `summarise()` has grouped output by 'region'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 12 x 3
##     year Europe Switzerland
##    <int>  <dbl>       <dbl>
##  1  1952   5.07        14.7
##  2  1957   6.04        17.9
##  3  1962   7.48        20.4
##  4  1967   9.33        23.0
##  5  1972  12.3         27.2
##  6  1977  14.2         27.0
##  7  1982  15.3         28.4
##  8  1987  16.1         30.3
##  9  1992  17.5         31.9
## 10  1997  18.7         32.1
## 11  2002  22.5         34.5
## 12  2007  27.5         37.5
```