Data Wrangling with `dplyr` - Exercises

Lachlan Deer Uli Bergmann

The goal of this document is to give you some experience using functions from `dplyr` with a little less guidance. Proceed at your own pace, and we will provide solutions later on.

We want to build up some experience with is `dplyr`.

Load it and tibble:

```
library("dplyr")
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library("tibble")
```

We are going to use a classic dataset from Graduate Econometrics texts that has gasoline consumption and some other variables by country for the 60's and 70's. The data is inside the package `plm` and called `Gasoline`. Load it as follows:

```
data(Gasoline, package = "plm")
```

Use the help to get a sense of what variable names mean in the data.

Now work through the following exercises:

1. Create a dataset `gasoline` which is a tibble

```
gasoline = as_tibble(Gasoline)
```

2. Create a subset of that only has data from the 1960s. Do this two ways, once where you don't use piping, %>%, and once where you do.

We can do this the normal way:

```
filter(gasoline, year < 1970)
```

```
## # A tibble: 180 x 6
##    country  year lgaspcar lincomep  lrpmg lcarpcap
##    <fct>   <int>    <dbl>    <dbl>  <dbl>    <dbl>
##  1 AUSTRIA  1960     4.17    -6.47 -0.335    -9.77
```

```
##  2 AUSTRIA  1961     4.10    -6.43 -0.351    -9.61
##  3 AUSTRIA  1962     4.07    -6.41 -0.380    -9.46
##  4 AUSTRIA  1963     4.06    -6.37 -0.414    -9.34
##  5 AUSTRIA  1964     4.04    -6.32 -0.445    -9.24
##  6 AUSTRIA  1965     4.03    -6.29 -0.497    -9.12
##  7 AUSTRIA  1966     4.05    -6.25 -0.467    -9.02
##  8 AUSTRIA  1967     4.05    -6.23 -0.506    -8.93
##  9 AUSTRIA  1968     4.05    -6.21 -0.522    -8.85
## 10 AUSTRIA  1969     4.05    -6.15 -0.559    -8.79
## # ... with 170 more rows
```

As we know, piping replaces the first argument of a function with the object we pipe into it, we can therefore repeat above's command via:

```
gasoline %>% filter(year < 1970)
```

```
## # A tibble: 180 x 6
##    country  year lgaspcar lincomep  lrpmg lcarpcap
##    <fct>   <int>    <dbl>    <dbl>  <dbl>    <dbl>
##  1 AUSTRIA  1960     4.17    -6.47 -0.335    -9.77
##  2 AUSTRIA  1961     4.10    -6.43 -0.351    -9.61
##  3 AUSTRIA  1962     4.07    -6.41 -0.380    -9.46
##  4 AUSTRIA  1963     4.06    -6.37 -0.414    -9.34
##  5 AUSTRIA  1964     4.04    -6.32 -0.445    -9.24
##  6 AUSTRIA  1965     4.03    -6.29 -0.497    -9.12
##  7 AUSTRIA  1966     4.05    -6.25 -0.467    -9.02
##  8 AUSTRIA  1967     4.05    -6.23 -0.506    -8.93
##  9 AUSTRIA  1968     4.05    -6.21 -0.522    -8.85
## 10 AUSTRIA  1969     4.05    -6.15 -0.559    -8.79
## # ... with 170 more rows
```

3. Create a subset that contains data from the years ranging from 1969 to 1973.

We have a few options to accomplish this.

1. Explicitly naming all years we want

```
filter(gasoline, year %in% c(1969,1970,1971,1972,1973))
```

```
## # A tibble: 90 x 6
##    country  year lgaspcar lincomep  lrpmg lcarpcap
##    <fct>   <int>    <dbl>    <dbl>  <dbl>    <dbl>
##  1 AUSTRIA  1969     4.05    -6.15 -0.559    -8.79
##  2 AUSTRIA  1970     4.08    -6.08 -0.597    -8.73
##  3 AUSTRIA  1971     4.11    -6.04 -0.654    -8.64
##  4 AUSTRIA  1972     4.13    -5.98 -0.596    -8.54
##  5 AUSTRIA  1973     4.20    -5.90 -0.594    -8.49
##  6 BELGIUM  1969     3.85    -5.86 -0.355    -8.52
##  7 BELGIUM  1970     3.87    -5.80 -0.378    -8.45
##  8 BELGIUM  1971     3.87    -5.76 -0.399    -8.41
##  9 BELGIUM  1972     3.91    -5.71 -0.311    -8.36
## 10 BELGIUM  1973     3.90    -5.64 -0.373    -8.31
## # ... with 80 more rows
```

2. Create the vector sequence like so:

```
filter(gasoline, year %in% 1969:1973)
```

```
## # A tibble: 90 x 6
##    country  year lgaspcar lincomep  lrpmg lcarpcap
##    <fct>   <int>    <dbl>    <dbl>  <dbl>    <dbl>
##  1 AUSTRIA  1969     4.05    -6.15 -0.559    -8.79
##  2 AUSTRIA  1970     4.08    -6.08 -0.597    -8.73
##  3 AUSTRIA  1971     4.11    -6.04 -0.654    -8.64
##  4 AUSTRIA  1972     4.13    -5.98 -0.596    -8.54
##  5 AUSTRIA  1973     4.20    -5.90 -0.594    -8.49
##  6 BELGIUM  1969     3.85    -5.86 -0.355    -8.52
##  7 BELGIUM  1970     3.87    -5.80 -0.378    -8.45
##  8 BELGIUM  1971     3.87    -5.76 -0.399    -8.41
##  9 BELGIUM  1972     3.91    -5.71 -0.311    -8.36
## 10 BELGIUM  1973     3.90    -5.64 -0.373    -8.31
## # ... with 80 more rows
```

3. Using small equal and bigger equal in the search

```
filter(gasoline, year >= 1969, year <= 1973)
```

```
## # A tibble: 90 x 6
##    country  year lgaspcar lincomep  lrpmg lcarpcap
##    <fct>   <int>    <dbl>    <dbl>  <dbl>    <dbl>
##  1 AUSTRIA  1969     4.05    -6.15 -0.559    -8.79
##  2 AUSTRIA  1970     4.08    -6.08 -0.597    -8.73
##  3 AUSTRIA  1971     4.11    -6.04 -0.654    -8.64
##  4 AUSTRIA  1972     4.13    -5.98 -0.596    -8.54
##  5 AUSTRIA  1973     4.20    -5.90 -0.594    -8.49
##  6 BELGIUM  1969     3.85    -5.86 -0.355    -8.52
##  7 BELGIUM  1970     3.87    -5.80 -0.378    -8.45
##  8 BELGIUM  1971     3.87    -5.76 -0.399    -8.41
##  9 BELGIUM  1972     3.91    -5.71 -0.311    -8.36
## 10 BELGIUM  1973     3.90    -5.64 -0.373    -8.31
## # ... with 80 more rows
```

4. Using dplyr's `between` function

```
filter(gasoline, between(year, 1969, 1973))
```

```
## # A tibble: 90 x 6
##    country  year lgaspcar lincomep  lrpmg lcarpcap
##    <fct>   <int>    <dbl>    <dbl>  <dbl>    <dbl>
##  1 AUSTRIA  1969     4.05    -6.15 -0.559    -8.79
##  2 AUSTRIA  1970     4.08    -6.08 -0.597    -8.73
##  3 AUSTRIA  1971     4.11    -6.04 -0.654    -8.64
##  4 AUSTRIA  1972     4.13    -5.98 -0.596    -8.54
##  5 AUSTRIA  1973     4.20    -5.90 -0.594    -8.49
##  6 BELGIUM  1969     3.85    -5.86 -0.355    -8.52
##  7 BELGIUM  1970     3.87    -5.80 -0.378    -8.45
##  8 BELGIUM  1971     3.87    -5.76 -0.399    -8.41
##  9 BELGIUM  1972     3.91    -5.71 -0.311    -8.36
## 10 BELGIUM  1973     3.90    -5.64 -0.373    -8.31
## # ... with 80 more rows
```

4. Create a subset that contains data for the years 1969, 1973 and 1977.

```
filter(gasoline, year %in% c(1969, 1973, 1977))
```

```
## # A tibble: 54 x 6
##    country  year lgaspcar lincomep  lrpmg lcarpcap
##    <fct>   <int>    <dbl>    <dbl>  <dbl>    <dbl>
##  1 AUSTRIA  1969     4.05    -6.15 -0.559    -8.79
##  2 AUSTRIA  1973     4.20    -5.90 -0.594    -8.49
##  3 AUSTRIA  1977     3.93    -5.83 -0.422    -8.25
##  4 BELGIUM  1969     3.85    -5.86 -0.355    -8.52
##  5 BELGIUM  1973     3.90    -5.64 -0.373    -8.31
##  6 BELGIUM  1977     3.85    -5.56 -0.432    -8.14
##  7 CANADA   1969     4.86    -5.56 -1.04     -8.10
##  8 CANADA   1973     4.90    -5.41 -1.13     -7.94
##  9 CANADA   1977     4.81    -5.34 -1.07     -7.77
## 10 DENMARK  1969     4.17    -5.72 -0.407    -8.47
## # ... with 44 more rows
```

5. Create a dataset that contains only the columns country, year, lrpmg.

```
select(gasoline, country, year, lrpmg)
```

```
## # A tibble: 342 x 3
##    country  year  lrpmg
##    <fct>   <int>  <dbl>
##  1 AUSTRIA  1960 -0.335
##  2 AUSTRIA  1961 -0.351
##  3 AUSTRIA  1962 -0.380
##  4 AUSTRIA  1963 -0.414
##  5 AUSTRIA  1964 -0.445
##  6 AUSTRIA  1965 -0.497
##  7 AUSTRIA  1966 -0.467
##  8 AUSTRIA  1967 -0.506
##  9 AUSTRIA  1968 -0.522
## 10 AUSTRIA  1969 -0.559
## # ... with 332 more rows
```

6. Create a dataset that does not contain the columns country, year, lrpmg.

```
select(gasoline, -country, -year, -lrpmg)
```

```
## # A tibble: 342 x 3
##    lgaspcar lincomep lcarpcap
##       <dbl>    <dbl>    <dbl>
##  1     4.17    -6.47    -9.77
##  2     4.10    -6.43    -9.61
##  3     4.07    -6.41    -9.46
##  4     4.06    -6.37    -9.34
##  5     4.04    -6.32    -9.24
##  6     4.03    -6.29    -9.12
```

```
##  7     4.05    -6.25    -9.02
##  8     4.05    -6.23    -8.93
##  9     4.05    -6.21    -8.85
## 10     4.05    -6.15    -8.79
## # ... with 332 more rows
```

7. Rename the column year to be called date.

```
rename(gasoline, date = year)
```

```
## # A tibble: 342 x 6
##    country  date lgaspcar lincomep  lrpmg lcarpcap
##    <fct>   <int>    <dbl>    <dbl>  <dbl>    <dbl>
##  1 AUSTRIA  1960     4.17    -6.47 -0.335    -9.77
##  2 AUSTRIA  1961     4.10    -6.43 -0.351    -9.61
##  3 AUSTRIA  1962     4.07    -6.41 -0.380    -9.46
##  4 AUSTRIA  1963     4.06    -6.37 -0.414    -9.34
##  5 AUSTRIA  1964     4.04    -6.32 -0.445    -9.24
##  6 AUSTRIA  1965     4.03    -6.29 -0.497    -9.12
##  7 AUSTRIA  1966     4.05    -6.25 -0.467    -9.02
##  8 AUSTRIA  1967     4.05    -6.23 -0.506    -8.93
##  9 AUSTRIA  1968     4.05    -6.21 -0.522    -8.85
## 10 AUSTRIA  1969     4.05    -6.15 -0.559    -8.79
## # ... with 332 more rows
```

8. Select all columns that start with "l".

The help of the `select()` function

```
? select
```

gives us a hint and tells us about the `starts_with()` function.

```
select(gasoline, starts_with("l"))
```

```
## # A tibble: 342 x 4
##    lgaspcar lincomep  lrpmg lcarpcap
##       <dbl>    <dbl>  <dbl>    <dbl>
##  1     4.17    -6.47 -0.335    -9.77
##  2     4.10    -6.43 -0.351    -9.61
##  3     4.07    -6.41 -0.380    -9.46
##  4     4.06    -6.37 -0.414    -9.34
##  5     4.04    -6.32 -0.445    -9.24
##  6     4.03    -6.29 -0.497    -9.12
##  7     4.05    -6.25 -0.467    -9.02
##  8     4.05    -6.23 -0.506    -8.93
##  9     4.05    -6.21 -0.522    -8.85
## 10     4.05    -6.15 -0.559    -8.79
## # ... with 332 more rows
```

9. Select the columns country, year, and all columns that contain the letters "car".

The help of the `select()` function also showed us that there is a `contains()` function:

```
select(gasoline, country, year, contains("car"))
```

```
## # A tibble: 342 x 4
##    country  year lgaspcar lcarpcap
##    <fct>   <int>    <dbl>    <dbl>
##  1 AUSTRIA  1960     4.17    -9.77
##  2 AUSTRIA  1961     4.10    -9.61
##  3 AUSTRIA  1962     4.07    -9.46
##  4 AUSTRIA  1963     4.06    -9.34
##  5 AUSTRIA  1964     4.04    -9.24
##  6 AUSTRIA  1965     4.03    -9.12
##  7 AUSTRIA  1966     4.05    -9.02
##  8 AUSTRIA  1967     4.05    -8.93
##  9 AUSTRIA  1968     4.05    -8.85
## 10 AUSTRIA  1969     4.05    -8.79
## # ... with 332 more rows
```

10. What does the function `pull()` do? Try it on the column `lrpmg`.

To find out about the `pull()` function, look at its help file

```
? pull()
```

Trying it out and showing the structure of its output

```
pull(gasoline, var = lrpmg)
str(pull(gasoline, var = lrpmg))
```

reveals that it just returns the `lrpmg` column of the dataframe as a vector in the correct format.

It is identical to the following two commands:

```
gasoline[["lrpmg"]]

gasoline$lrpmg
```

11. Create a grouped data set that groups the data by country.

```
gas_country = group_by(gasoline, country)
```

To see the grouping, we use the glimpse command and confirm it was successful:

```
glimpse(gas_country)
```

```
## Observations: 342
## Variables: 6
## Groups: country [18]
## $ country  <fct> AUSTRIA, AUSTRIA, AUSTRIA, AUSTRIA, AUSTRIA, AUSTRIA, AUST...
## $ year     <int> 1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969...
## $ lgaspcar <dbl> 4.173244, 4.100989, 4.073177, 4.059509, 4.037689, 4.033983...
```

```
## $ lincomep <dbl> -6.474277, -6.426006, -6.407308, -6.370679, -6.322247, -6....
## $ lrpmg    <dbl> -0.3345476, -0.3513276, -0.3795177, -0.4142514, -0.4453354...
## $ lcarpcap <dbl> -9.766840, -9.608622, -9.457257, -9.343155, -9.237739, -9....
```

12. Ungroup the dataset from 11.

```
gas_ungrouped = ungroup(gas_country)
glimpse(gas_ungrouped)
```

```
## Observations: 342
## Variables: 6
## $ country  <fct> AUSTRIA, AUSTRIA, AUSTRIA, AUSTRIA, AUSTRIA, AUSTRIA, AUST...
## $ year     <int> 1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969...
## $ lgaspcar <dbl> 4.173244, 4.100989, 4.073177, 4.059509, 4.037689, 4.033983...
## $ lincomep <dbl> -6.474277, -6.426006, -6.407308, -6.370679, -6.322247, -6....
## $ lrpmg    <dbl> -0.3345476, -0.3513276, -0.3795177, -0.4142514, -0.4453354...
## $ lcarpcap <dbl> -9.766840, -9.608622, -9.457257, -9.343155, -9.237739, -9....
```

13. Find the mean of lgaspcar by country. Call that variable `avg_lgaspcar`.

A naive approach via

```
avg_lgaspcar = mean(gas_country$lgaspcar)
```

only returns a single mean of the whole dataframe. To get the correct average per country, we use the `summarise()` function

```
summarise(gas_country, avg_lgaspcar = mean(lgaspcar))
```

```
## # A tibble: 18 x 2
##    country  avg_lgaspcar
##    <fct>           <dbl>
##  1 AUSTRIA          4.06
##  2 BELGIUM          3.92
##  3 CANADA           4.86
##  4 DENMARK          4.19
##  5 FRANCE           3.82
##  6 GERMANY          3.89
##  7 GREECE           4.88
##  8 IRELAND          4.23
##  9 ITALY            3.73
## 10 JAPAN            4.70
## 11 NETHERLA         4.08
## 12 NORWAY           4.11
## 13 SPAIN            4.06
## 14 SWEDEN           4.01
## 15 SWITZERL         4.24
## 16 TURKEY           5.77
## 17 U.K.             3.98
## 18 U.S.A.           4.82
```

14. Return a dataset that computes the mean of lgaspcar for france.

```
filter(gas_country, country == "FRANCE") %>%
  summarise(avg_lgaspcar = mean(lgaspcar))
```

```
## # A tibble: 1 x 2
##   country avg_lgaspcar
##   <fct>          <dbl>
## 1 FRANCE          3.82
```

15. Compute the mean, standard deviation, min and max of lgaspcar by country.

```
sum_gas_country = summarise(gas_country,
                  avg_lgaspcar = mean(lgaspcar),
                  sd_lgaspcar  = sd(lgaspcar),
                  min_lgaspcar = min(lgaspcar),
                  max_lgaspcar = max(lgaspcar))
```

16. Which country has the highest average gasoline consumption.

To answer this question, we order the dataset by avg consumption in descending order:

```
arrange(sum_gas_country, desc(avg_lgaspcar))
```

```
## # A tibble: 18 x 5
##    country  avg_lgaspcar sd_lgaspcar min_lgaspcar max_lgaspcar
##    <fct>           <dbl>       <dbl>        <dbl>        <dbl>
##  1 TURKEY           5.77      0.329          5.14         6.16
##  2 GREECE           4.88      0.255          4.48         5.38
##  3 CANADA           4.86      0.0262         4.81         4.90
##  4 U.S.A.           4.82      0.0219         4.79         4.86
##  5 JAPAN            4.70      0.684          3.95         6.00
##  6 SWITZERL         4.24      0.102          4.05         4.44
##  7 IRELAND          4.23      0.0437         4.16         4.33
##  8 DENMARK          4.19      0.158          4.00         4.50
##  9 NORWAY           4.11      0.123          3.96         4.44
## 10 NETHERLA         4.08      0.286          3.71         4.65
## 11 AUSTRIA          4.06      0.0693         3.92         4.20
## 12 SPAIN            4.06      0.317          3.62         4.75
## 13 SWEDEN           4.01      0.0364         3.91         4.07
## 14 U.K.             3.98      0.0479         3.91         4.10
## 15 BELGIUM          3.92      0.103          3.82         4.16
## 16 GERMANY          3.89      0.0239         3.85         3.93
## 17 FRANCE           3.82      0.0499         3.75         3.91
## 18 ITALY            3.73      0.220          3.38         4.05
```

17. Return a dataset that returns the countries with the highest and lowest average consumption.

To answer this, we can filter our dataset for values which are equal to the max and min of gas consumption.

```
filter(sum_gas_country, avg_lgaspcar == max(avg_lgaspcar) | avg_lgaspcar == min(avg_lgaspcar))
```

```
## # A tibble: 2 x 5
##   country avg_lgaspcar sd_lgaspcar min_lgaspcar max_lgaspcar
##   <fct>          <dbl>       <dbl>        <dbl>        <dbl>
## 1 ITALY           3.73       0.220         3.38         4.05
## 2 TURKEY          5.77       0.329         5.14         6.16
```

18. Add a variable `count` to the dataset that has the number of times each country appears in the data - Is it balanced?

For this we count the number of values for each country

```
country_count = summarise(gas_country, count = n())
country_count
```

```
## # A tibble: 18 x 2
##    country  count
##    <fct>    <int>
##  1 AUSTRIA     19
##  2 BELGIUM     19
##  3 CANADA      19
##  4 DENMARK     19
##  5 FRANCE      19
##  6 GERMANY     19
##  7 GREECE      19
##  8 IRELAND     19
##  9 ITALY       19
## 10 JAPAN       19
## 11 NETHERLA    19
## 12 NORWAY      19
## 13 SPAIN       19
## 14 SWEDEN      19
## 15 SWITZERL    19
## 16 TURKEY      19
## 17 U.K.        19
## 18 U.S.A.      19
```

19. Create a meaningless dataset called `spam` that is the exponential of the sum of lgaspcar and lincomep. Also check out what happens if you replace `mutate()` with `transmute()`.

`mutate()` includes the newly created values and all existing columns

```
spam = mutate(gasoline, my_exp = exp(lgaspcar + lincomep))
glimpse(spam)
```

```
## Observations: 342
## Variables: 7
## $ country  <fct> AUSTRIA, AUSTRIA, AUSTRIA, AUSTRIA, AUSTRIA, AUSTRIA, AUST...
## $ year     <int> 1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969...
## $ lgaspcar <dbl> 4.173244, 4.100989, 4.073177, 4.059509, 4.037689, 4.033983...
## $ lincomep <dbl> -6.474277, -6.426006, -6.407308, -6.370679, -6.322247, -6....
```

```
## $ lrpmg    <dbl> -0.3345476, -0.3513276, -0.3795177, -0.4142514, -0.4453354...
## $ lcarpcap <dbl> -9.766840, -9.608622, -9.457257, -9.343155, -9.237739, -9....
## $ my_exp   <dbl> 0.10015533, 0.09778181, 0.09689458, 0.09914524, 0.10181905...
```

While `transmute()` does not include existing columns

```
spam2 = transmute(gasoline, my_exp = exp(lgaspcar + lincomep))
glimpse(spam2)
```

```
## Observations: 342
## Variables: 1
## $ my_exp <dbl> 0.10015533, 0.09778181, 0.09689458, 0.09914524, 0.10181905, ...
```

20. Create the lead and lag of lgaspcar for each row of data. Call the new columns `lead_lgaspcar` and `lag_lgaspcar`.

```
gas_ts = mutate(gas_country,
                lead_lgaspcar = lead(lgaspcar, order_by = year),
                lag_lgaspcar  =  lag(lgaspcar, order_by = year))
```

NOTE:

1. We used the data which was grouped by `country`.

- Try what happens when we use the ungrouped data `gasoline` instead?
- Whats the problem?

2. We ordered the data by year. This makes sure that the lags are ordered correctly.

- What would happen if the data was in the incorrect order?
- Try e.g. to order it by `lrpmg` before creating lead and lag

21.

The following countries belong the to EU:

```
eu_countries <- c("austria", "belgium", "bulgaria", "croatia", "republic of cyprus",
                  "czech republic", "denmark", "estonia", "finland", "france", "germany",
                  "greece", "hungary", "ireland", "italy", "latvia", "lithuania", "luxembourg",
                  "malta", "netherla", "poland", "portugal", "romania", "slovakia", "slovenia",
                  "spain", "sweden", "u.k.")
```

Create a variable `in_eu` in the gasoline data which takes the value `TRUE` if a country is in the EU. (Note that the case of the string will matter!)

```
eu_countries <- c("austria", "belgium", "bulgaria", "croatia", "republic of cyprus",
                    "czech republic", "denmark", "estonia", "finland", "france", "germany",
                    "greece", "hungary", "ireland", "italy", "latvia", "lithuania", "luxembourg",
                    "malta", "netherla", "poland", "portugal", "romania", "slovakia", "slovenia",
                    "spain", "sweden", "u.k.")
```

Let's first ignore the hint and try to create the indicator varible for the supplied `eu_countries` vector as is:

```
transmute(gasoline, in_eu = (country %in% eu_countries))
```

```
## # A tibble: 342 x 1
##    in_eu
```

```
##    <lgl>
##  1 FALSE
##  2 FALSE
##  3 FALSE
##  4 FALSE
##  5 FALSE
##  6 FALSE
##  7 FALSE
##  8 FALSE
##  9 FALSE
## 10 FALSE
## # ... with 332 more rows
```

We see the value of `in_eu` is `FALSE` for all countries. This is because the country names in `gasoline` are written in ALLCAPS while the countries in `eu_countries` are written in lowercase.

Let us first transform the `eu_countries` vector to ALLCAPS with the `toupper()` function:

```
EU_COUNTRIES = toupper(eu_countries)
```

and now let's try to `mutate()` again

```
gasoline = mutate(gasoline, in_eu = (country %in% EU_COUNTRIES))
```

We see that this worked as expected

22.

Here's a different way to classify countries:

- Mediterranean: france, italy, turkey, greece, spain
- Central Europe: germany, austria, switzerl, belgium, netherla
- Anglosphere: canada, u.s.a. , u.k., ireland
- Nordic: denmark, norway, sweden
- asia: japan

  Create a new variable `region` that uses these definitions. (Hint: `case_when()`) will likely be your friend here.

Let's first create a list for each region:

```
medi   = toupper(c("france", "italy", "turkey", "greece", "spain"))
ce     = toupper(c("germany", "austria", "switzerl", "belgium", "netherla"))
anglo  = toupper(c("canada", "u.s.a." , "u.k.", "ireland"))
nordic = toupper(c("denmark", "norway", "sweden"))
asia   = toupper("japan")
```

now we will create the variable using `mutate()` and `case_when()`

```
gasoline = mutate(gasoline,
                  region = case_when(
                    country %in% medi   ~ "Mediterranean",
                    country %in% ce     ~ "Central Europe",
                    country %in% anglo  ~ "Anglosphere",
                    country %in% nordic ~ "Nordic",
                    country %in% asia   ~ "Asia"
```

```
                    )
)
```

23. Notice that in the country names switzerl and netherla are a little funky. Use the functions `mutate` and `recode` to replace the name with the full country name.

```
gasoline = mutate(gasoline,
                  country = recode_factor(country,
                                          NETHERLA = "NETHERLANDS",
                                          SWITZERL = "SWITZERLAND"
                  )
)
```

24. Compute the variable `quintile` that computes which quintile of lgaspcar each country is. Do this only for 1960. Repeat this to create a variable `decile` with the appropriate definiiton.

First we create a dataset only for 1960 where gas consumption is grouped by country

```
gas_country_1960 = filter(gasoline, year == 1960) %>%
  group_by(country) %>%
  summarise(avg_lgaspcar = mean(lgaspcar)
  )
```

Next, we use dplyr's ntile function on `gas_country_1960$avg_lgaspcar`. Note that quintiles have 5 cutoff values for the CDF of the distribution at 0%, 20%, 40%, 60%, 80% and 100%. Quintiles therefore have 4 groups between them.

```
ntile(gas_country_1960$avg_lgaspcar, 4)
```

```
##  [1] 3 2 2 2 4 3 1 1 4 2 1 4 3 3 1 4 1 3
```

We see that the ntile function returns a vector of the quintile in which a value for a country lies. As the order is equal to the one in `gas_country`, we can just add it as a new column to our data with the name `quintile`:

```
gas_country_1960$quintile = ntile(gas_country_1960$avg_lgaspcar, 4)
```

Next, we repeat this for `decile` which defines 9 intervals

```
gas_country_1960$decile = ntile(gas_country_1960$avg_lgaspcar, 9)
```

25. Create a variable `high_consumption` that takes the value `TRUE` if lgaspcar is higher than the yearly average for a given country.

The easiest way to do this is to match the country averages which we calculated in question 15 back into the main dataset and then do the comparison

```
gasoline = left_join(gasoline, select(sum_gas_country, country, avg_lgaspcar), by = "country") %>%
  mutate(high_consumption = (lgaspcar > avg_lgaspcar)) %>%
  select(-avg_lgaspcar)
```

```
## Warning: Column `country` joining factors with different levels, coercing to
## character vector
```

Note:

- we use `select()` because we only need the `country` and `avg_lgaspcar` columns from the grouped dataset
- after computing `high_consumption` we drop `avg_lgaspcar` because we don't need it anymore