Seminar 3 – week 37

I. Computing π

In this exercise, we compute an approximation of the number π. We all know from school that $\pi \approx 3.14$, but as it is an irrational number the number of decimals is infinite. Many approaches have been suggested, some more efficient than others. We are going to compare two approaches. A method suggested by Gregory and Leibnitz in the 1670s is to compute π using the series

$$\pi = \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \cdots$$

Another approach, initially suggested by Nilakantha around 1500, is to use the series

$$\pi = 3 + \frac{4}{2 \times 3 \times 4} - \frac{4}{4 \times 5 \times 6} + \frac{4}{6 \times 7 \times 8} - \cdots$$

Notice that the sign is switching between + and – in both approaches.

1) Write a for-loop to compute π using the Gregory-Leibnitz approach with 10 terms.
   Hint: To obtain the alternating sign, you can use a variable `sign` *that is multiplied by -1 in each iteration.*
2) Write a similar loop to compute π using the Nilakantha approach. Compare the precision of the two approaches
   Hint: A good estimate of π can be found in the R constant `pi`*.*
3) How does the performance of the two approaches change if you use 1000 terms instead of 10?
4) We want to study how precision improves in the two approaches as we increase the number of terms in the sums. Construct a loop that approximates π using the two algorithms with 10, 100, 1000, 10 000 and 100 000 iterations and stores the approximations of π in separate vectors. Feel free to add more values in the loop, but a very high number of terms can be time consuming.
5) Put the vectors of approximations and a vector with the number of terms in a tidyverse tibble. Compute the absolute value of the error from the true value of π (given in the constant `pi`). Use `ggplot` to print how the error changes when the number of terms increases. To enhance readability, use log10 axes. Which of the two approaches is the best?

II. Importing and cleaning data

In this exercise, we'll go through some steps in importing data and reshaping the data into a format we can work on. The data we use are the outcomes of the regional elections (Fylkestingvalg) in 2019, downloaded from `valgresultater.no`. They can be found in the file `2019-09-11_partifordeling_1_fy_2019.csv` on Canvas. This is a text file separated by semicolons. We want to transform the data into something like the file `election.csv` (also on Canvas).

1) Import the data into a tibble. Remove data from Oslo, as it's not a proper region.
   The import data function in RStudio (see File ->Import Dataset) may be useful to generate the appropriate code. The sign that tells R what to treat as two separate columns is called the "delimiter", and you may need to change it from the default, as our data are separated by semicolons, and not commas as is more common.
   `Filter` is useful to keep or remove observations.
2) The importing gives variable names with spaces, which are difficult to use. Define a new variable `votes` with the number of votes ("`Antall stemmer totalt`") for a party in a region. (The Norwegian word for region is fylke. Fylkenavn=region name, Fylkenummer=region ID.)
   To refer to variable names with spaces, enclose the name between left quotes (`).
3) Keep the variables `Fylkenavn`, `Fylkenummer`, `Partikode`, and `votes`, and store them in a new tibble `votes`.
   `Select` is useful to select some variables
4) We do not want to work with data on the smallest parties. Remove all parties except those with Partikode 'A', 'SV', 'RØDT', 'SP', 'FRP', 'KRF', 'MDG', 'H', 'FNB', 'V'.
5) Reshape the data tibble so each line is a region with the electoral results for each party as a variable.
   These reshapes can be performed by `pivot_wider`