Rijksuniversiteit Groningen

Contemporary Statistics with Applications

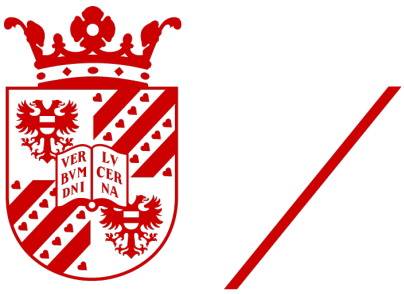
WMMA015-05

An introduction to splines

Author:
Olav Gruppen

Teacher:
Marco Grzegorczyk

November 1, 2022



rijksuniversiteit
groningen

1 Introduction

In this reader, I will provide an introduction to splines. The reader combines mathematical theory and an with programming exercises in R, and has an ultimate goal to build an implementation of splines together with the reader.

2 Data

In this project, we will work with speedskating times. The times are taken from the World Allround Speed Skating Championships organised by the International Skating Union (ISU). In ice-skating, the allround discipline is historically regarded as one of the most prestigious prizes to win, and therefore, the allround tournament was the first official tournament to be organised by the ISU. The allround tournament consists of 4 distances: 500 and 5000 meter on the first day, and 1500 and 10.000 meter on the second day. Every skater does the race individually and their times are recorded. In order to find the winner, all skaters get a score and the lowest-scoring skater wins. The score is simply the sum of rescaled times, where the time for every distance is rescaled to 500 meter.

In my eyes, an important skill for any data scientist is collecting data. As statisticians, we often work with provided data sets, and we never think about the collection of the data. Therefore, I have learned a technique called "webscraping" in this project. I have written a function in a Python notebook to scrape wikipedia pages in order to find winning times for every distance on the ISU allround speedskating championships. The code is included on my GitHub page[1] as the file `src/scrape_wiki.ipynb`. After scraping the files and manually adding the missing values in Excel, I have used the file `src/load_data.R` to load the data nicely formatted into R.

In my websprape, I have restricted to using only male skating times, since the first world championship allround was already in 1889 for men, but only in 1936 for women. There is simply more data available for men. In my eyes, the longest distance is the most beautiful, though not every ice-skater will agree. Therefore, I have chosen to work work only the data on the 10.000m in this reader "An introduction to splines". Of course, if you wish, you can load any of the 4 distance datasets. I refer to section 6 for results using different data than 10.000m.

I have visualized the speedskating times in Figure 1. Times vary from 1335 seconds in 1903 (Sint-Petersburg) to 770 seconds in 2007 (Heerenveen) - a gradual improvement over the years. Let $X = (1888, 1889, \ldots, 2020, 2021)^T$ be a vector of the years 1888 until 2021, and let $Y = (NA, NA, \ldots, 782.45, NA)^T$ be the winning time on the 10.000m during the allround speed skating championships, with NA for missing data. The first and last 10 rows of the data are shown in Table 1. Note that in our example, the vector X (years) is one-dimensional. This is not required for splines, but having a one dimensional X makes splines a lot easier to understand.

[1] https://github.com/Hallolav/introduction_to_splines

	year	time
1	1888	NA
2	1889	NA
3	1890	NA
4	1891	NA
5	1892	NA
6	1893	1221.40
7	1894	1152.40
8	1895	1076.00
9	1896	1132.40
10	1897	1202.40
...	...	...
125	2012	NA
126	2013	791.86
127	2014	793.93
128	2015	776.69
129	2016	787.19
130	2017	790.45
131	2018	820.38
132	2019	771.17
133	2020	782.45
134	2021	NA

Table 1: First and last rows of the speedskating dataset using 10.000m

Figure 1: development of finish times on 10.000 meter in the ISU allround speed skating championships

There are several reasons which lead to great development of speedskating times over the years. Firstly, artificial ice rinks were introduced in 1924, which lead to better ice quality and thus faster times. Secondly, the "nordic skate" was introduced in 1954 - where the blade was connected to the shoe, instead of being tied to shoes. Then, in 1974, aerodynamic lycra suits were introduced, decreasing air resistance of skaters. In 1986, artificial ice rinks with climate control were introduced, and the most recent technical improvement was in 1996, when the "clap skate" was introduced.

In this reader, we will use the years above as "knots", cutoff points that define partitions for a variable. In piecewise polynomials, for example, the knots are used in as borders for line segments, and in splines to fit a new basis function. This will become clear shortly.

3 Piecewise Polynomials

A *piecewise polynomial* is obtained by dividing the domain of X into intervals, and representing the prediction function $f(X)$ by a seperate polynomial in each interval. For example, the toy example in Figure 2 is sampled from $f(x) \begin{cases} -5 + 0.1x + \epsilon \\ 5 + \epsilon \\ 25 - 0.1x + \epsilon \end{cases}$, where ϵ is is noise sampled from the standard normal distribution. We could fit a quadratic function to the data and the results would be quite okay – but a piecewise linear equation would be better: we know that the "true" function consists of three linear segments.

In order to fit a piecewise polynomial to the toy data, we would fit three polynomials: one to the segment x between 0 and 100, a second one to the segment x between 100 and 200, and finally, a third polynomial to the segment x between 200 and 300.

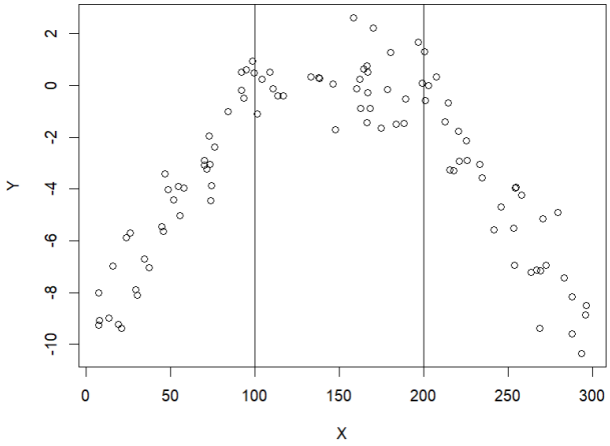


Figure 2: Toy data. In order to fit a piecewise polynomial to the function, the linear case is sufficient

The degree d of the piecewise polynomial, is the highest dimension in the regression model that we fit to the data plus 1. For example, if we go for a quadratic piecewise polynomial (with degree 3), we would fit a polynomial $f(x) = \beta_0 + \beta_1 x + \beta_2 x^2$ to every segment. Note that we need to fit 6 polynomials in this assignment, each with d β coefficients - leaving $6 \cdot d$ degrees of freedom for the total model.

3.1 Code

In this section of the reader, we aim to fit a piecewise polynomial to the speedskating data. As discussed before, our knots are given by $(1924, 1954, 1974, 1986, 1996)$. Start by loading the accompanying code to this reader, importing the speedskating data and setting the knots. Your assignment is to finish the code by finishing the helper functions in order to fit the piecewise polynomial.

In my eyes, any data scientist should be able to work with GitHub, as it is an industry standard way of distributing code and tracking changes. I have therefore included accompanying code to the exercises in this reader on GitHub. You can find the code on my GitHub page.

Assignment 0. *Clone the GitHub repository and open the file* `An introduction to splines.R`. *This is the file we will use for assignments throughout this reader. Try to do all assignments in this reader*

yourself as they contribute to your learning, but take inspiration from `An introduction to splines (sample solution).R` *if you're completely stuck.*

3.2 `shift()` and `shift_back()`

We work with $X = (1888, \ldots, 2020)^T$, a vector of years. Especially in the higher degrees, the polynomial basis can become too big to work with and taking the inverse can result in numerical underflow problems. Therefore, in order to fit polynomials, we will need to to a *min-max scaling*. The min-max scaling is defined as follows:

$$x_{scaled} = \frac{x - \min(X)}{\max(X) - \min(X)}$$

Assignment 1. *Finish the* `shift()` *function. As an input, your function takes the years X, and your function should return a list of three values: the minimum of X, the maximum of X, and finally, the min-max scaled version of X.*

Assignment 2. *Finish the* `shift_back()` *function. As an input, your function takes the previously shifted X, and the previously found minimum and maximum value. As an output, your function returns the original values of X.*

3.3 `add_piecewise_basis()`

A polynomial is fit to every segment in the domain of X. In order to fit a polynomial of degree d, we need the following basis functions

1. $h_0(X) = X^0 = 1$
2. $h_1(X) = X$
3. ...
4. $h_{d-1}(X) = X^{d-1}$.

Assignment 3. *Finish the* `add_piecewise_basis()` *function. Your function takes the degree of the polynomial as input, as well as the years X, and outputs a matrix with the basis functions of X. Note that the column names of the matrix are expected to be* `c('X^0', 'X^1', ...)`*. A first step is already implemented.*

3.4 `find_beta()`

Let X be the basis functions of a one-dimensional predictor variable for degree d, and let Y the predicted variable. We aim to fit an coefficient β so that $RSS(\beta) = (Y - \beta X)^T(Y - \beta X)$ is minimized. That is, we set $\frac{d}{d\beta} RSS(\beta)$ to 0.

$$\begin{aligned}
RSS(\beta) &= (Y - \beta X)^T (Y - \beta X) \\
\frac{d}{d\beta} RSS(\beta) &= X^T(Y - X\beta) \\
0 &\stackrel{!}{=} X^T(Y - X\beta) \\
\implies X^T X\beta &= X^T Y \\
\implies \beta &= (X^T X)^{-1} X^T Y
\end{aligned}$$

Assignment 4. *Finish the function* `find_beta()`. *Your function takes basis function X as input, as well as the predicted variable Y. As an output, your function should output the coefficient $\beta = (X^T X)^{-1} X^T Y$. Note that β should be outputted as a column vector, with row names equal to* `c('X^0', 'X^1', ...)`

3.5 `make_prediction()`

In the previous exercise, we have seen that β minimizes the residual sum of squares. The coefficient is used to predict values for Y, given X as follows: $Y = \beta X = \beta_0 X^0 + \beta_1 X^1 + \dots$.

Assignment 5. *Finish the function* `make_prediction()` *to give a prediction based on two input variables: the basis function X, and the coefficient β. As an output, your function gives a dataframe with two columns: the predicted speedskating time, and the year of the prediction.*

3.6 Conclusion on piecewise polynomials

Assignment 6. *Fit a piecewise cubic polynomial $d = 4$ to the speedskating data using* `my_piecewise_polynomial()`

After finishing your code, your R should resemble something like Figure 3. There are some things going on with our predictions that are not nice. For example, the function is not continuous at the knots. In the next chapter(s), the introduction of (natural) splines will help us prevent these issues.

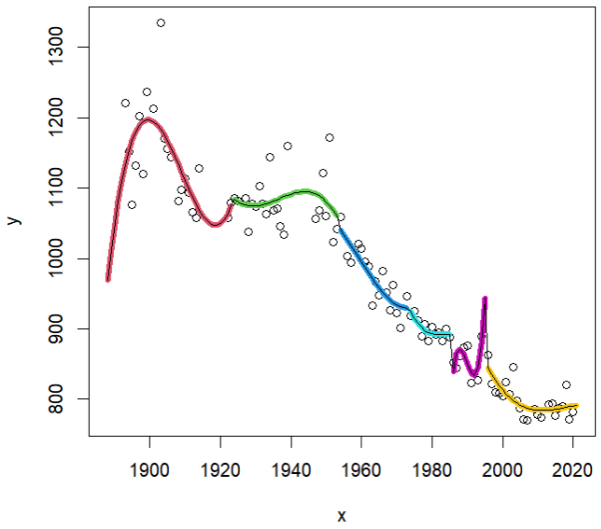


Figure 3: Piecewise polynomials fit to the speedskating dataset

4 Splines

In this section, we will fit splines to our dataset. Splines have the advantage over piecewise polynomials that they are "smooth" at the knots. Instead of repeatedly fitting polynomials to segments of the dataset, splines fit a function to the data. To do so, we will need more basis functions. Consider a spline of degree d with knots $\xi_1, \ldots, \xi_K$. We use the following basis functions:

- $h_0 = X^0 = 1$
- $\ldots$
- $h_{d-1} = X^{d-1}$
- $h_d = (X - \xi_1)_+^{d-1}$
- $\ldots$
- $h_{d+K-1} = (X - \xi_K)_+^{d-1}$

where $(X - \xi_j)_+ = \max(0, X - \xi_j)$.

Suppose that the degree of the spline is set at least to 1. To see that the spline is continuous, let $X^- = X \overset{\lim}{\nearrow} \xi_j$ let let $X^+ = X \overset{\lim}{\searrow} \xi_j$ (that is, letting X approach the knot from below and above respectively), and notice that continuity means that $f(X^+) = f(X^-)$. This is indeed the case, since $f(X^+) = \beta_0 1 + \cdots + \beta_{d-1}(X^+)^{d-1} + \beta_d(X^+ - \xi_1)_+^{d-1} + \cdots + \beta_{d+j-2}(X^+ - \xi_{j-1})_+^{d-1} + \beta_{d+j-1}(X^+ - \xi_j)_+^{d-1} = \beta_0 1 + \cdots + \beta_{d-1}(X^-)^{d-1} + \beta_d(X^- - \xi_1)_+^{d-1} + \cdots + \beta_{d+j-2}(X^- - \xi_{j-1})_+^{d-1} + 0 = f(X^-)$.

Similarly, in order to see that the spline is continuous in first and second derivative for degree > 3 (that is $\frac{\partial}{\partial X} f(X^+) = \frac{\partial}{\partial X} f(X^-)$ and $\frac{\partial^2}{\partial^2 X} f(X^+) = \frac{\partial^2}{\partial^2 X} f(X^-)$), notice again that that we can rewrite $f(X^+) = f(X^-) + \beta_{d+j}(X - \xi_j)_+^{d-1}$. Finally, noting that the first and second derivative of $\beta_{d+j}(X - \xi_j)_+^{d-1}$ are 0 at X^+, we have proven continuity at first and second derivative. It has been claimed that discontinuity in the second derivative is the highest discontinuity that humans can perceive, so that cubic splines are often chosen.

Using our speedskating data X, Y and knots $\xi_1 = 1924, \xi_2 = 1954, \xi_3 = 1974, \xi_4 = 1986$ and $\xi_5 = 1996$, we now have $d + 5$ degrees of freedom to fit a spline to our data. Recall that our piecewise polynomial had $6 \cdot d$ degrees of freedom, so this is a calculational improvement when $d > 1$.

4.1 `add_spline_basis()`

In this section, we have seen the basis function for splines. We will now implement it in R. Note that the bulk of the function `my_spline()` relies on your implementation of helper functions in previous assignments.

Assignment 7. *Finish the `add_spline_basis()` function in the code. The code takes the degree and knots as inputs, as well as the polynomial basis (which is calculated using your function from Assignment 3). As output, the function returns the spline bases. The column names are set for you in the code, you do not have to change them*

4.2 Conclusion on splines

Assignment 8. *Fit a cubic spline ($d = 4$) to the speedskating data using `my_spline()`*

Your output should resemble Figure 4. The prediction is a bit "wobbly" towards the ends. If we want to use our current model to predict future speedskating results, we would expect the times to go up. However, that is highly unlikely, as science (and sports with it) keeps improving. Luckily, natural splines come to the rescue.

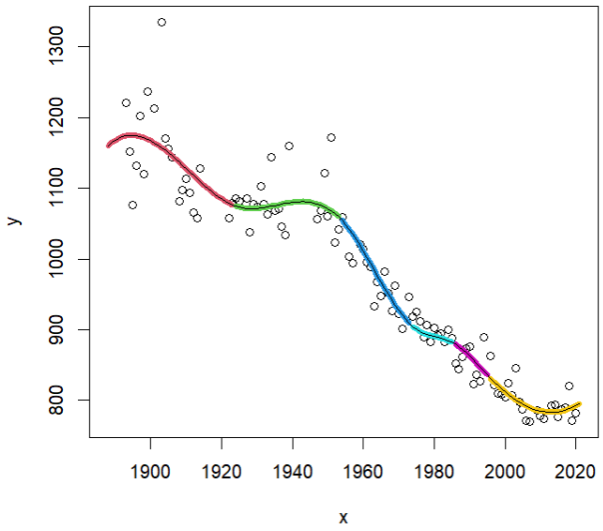


Figure 4: Cubic splines fit to the speedskating data

5 Natural splines

In the previous section on splines, we restrained our model so that the function is continuous at the knots. Natural splines add two more restrains: the model needs to be linear at the boundary knots. This is necessary, since the behaviour of polynomials can be erratic near boundaries, and extrapolation can therefore be dangerous. For a spline of degree d with knots $\xi_1, \ldots, \xi_K$, the basis functions for cubic splines are given by

1. $h_0(X) = X^0 = 1$
2. $h_1(X) = X^1 = X$
3. $h_2(X) = d_1(X) - d_{K-1}(X)$
4. $\ldots$
5. $h_d(X) = d_{K-2}(X) - d_{K-1}(X)$

where $d_j(X) = \frac{(X-\xi_j)_+^{d-1} - (X-\xi_K)_+^{d-1}}{\xi_K - \xi_j}$, again with $(X-\xi_j)_+ = \max(0, X-\xi_j)$.

Suppose that the degree of the natural spline is set at least to 1. To see that the natural spline is continuous, let $X^- = X \overset{\lim}{\nearrow} \xi_j$ and let $X^+ = X \overset{\lim}{\searrow} \xi_j$ again, and notice that we have continuity at the knot ξ_j since $f(X^+) = \beta_0 1 + \beta_1 X^+ + \beta_2 h_2(X^+) + \cdots + \beta_j h_j(X^+) + \beta_{j+1} h_{j+1}(X^+) = \beta_0 1 + \beta_1 X^- + \beta_2 h_2(X^-) + \cdots + \beta_j h_j(X^-) + 0 = f(X^-)$

Suppose now that the degree of the natural spline is > 3. To see continuity of the first and second derivative, start again by writing $f(X^+) = f(X^-) + \beta_{j+1} h_{j+1}(X^+)$, and consider the difference $f(X^+) - f(X^-) = \beta_{j+1} h_{j+1}(X^+) := d_j(X^+) - d_{d-1}(X^+) := \frac{(X^+ - \xi_j)_+^{d-1} - (X^+ - \xi_K)_+^{d-1}}{\xi_K - \xi_j} - \frac{(X^+ - \xi_{K-1})_+^{d-1} - (X^+ - \xi_K)_+^{d-1}}{\xi_K - \xi_{K-1}}$ and then noting the its first and second partial derivative w.r.t. X are 0 at X^+.

Using our speedskating data X, Y and knots $\xi_1 = 1924, \xi_2 = 1954, \xi_3 = 1974, \xi_4 = 1986$ and $\xi_5 = 1996$, we now have $d+1$ degrees of freedom to fit a spline to our data. Recall that our piecewise polynomial and splines had $6 \cdot d$ and $d + 5$ degrees of freedom respectively, so again we reach a calculational improvement when $d > 1$.

5.1 `add_natural_spline_basis()`

The natural spline basis function is given by $h_{j+1}(X) = d_j(X) - d_{K-1}(X)$, where $d_j(X) = \frac{(X - \xi_j)_+^{d-1} - (X - \xi_K)_+^{d-1}}{\xi_K - \xi_j}$. In this assignment, we will implement this in R.

Assignment 9. *Finish the function* ***`add_natural_spline_basis()`****. As input, the function takes the degree of the spline, as well as the X we are trying to predict, and the knots. The function $d_j(X)$ is already implemented in for you in R. Look closely at its implementation to find how to finish* ***`add_natural_spline_basis()`****.*

5.2 Conclusion on natural splines

Assignment 10. *Fit a cubic natural spline ($d = 4$) to the speedskating data using* ***`my_natural_spline()`***

Your output should resemble Figure 5. Note that our function is a lot less "wobbly", as we have enforced the first and last segment to be linear. Since the polynomials are less erratic at the boundaries, our new function $f(X)$ might prove when doing interpolation than our model for splines. However, the RSS of natural splines is higher than that of RSS, as that of regular splines, as pertubations are not followed as closely. This is a tradeoff that the reader should be aware of.

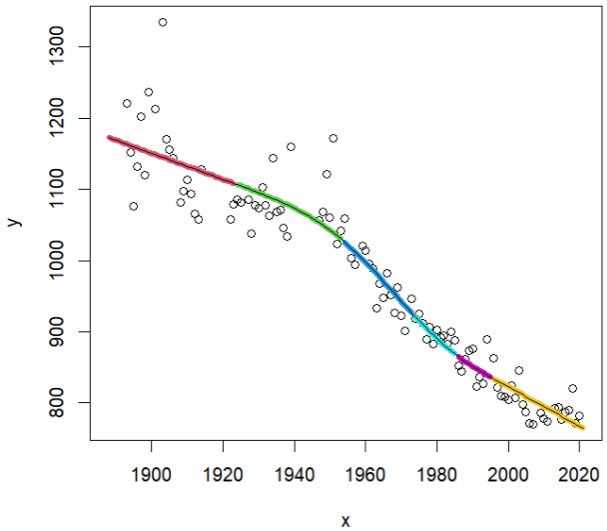
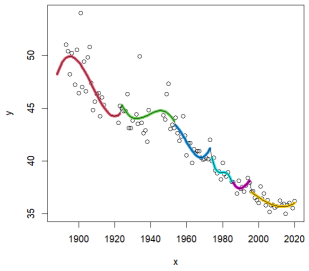
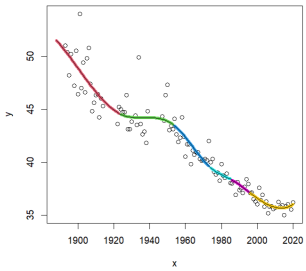
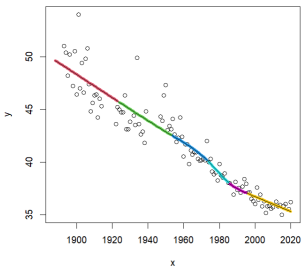
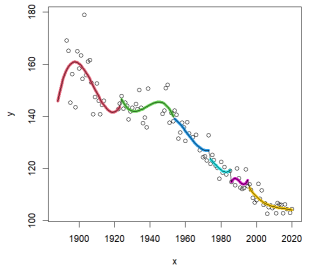
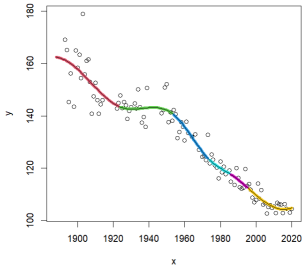
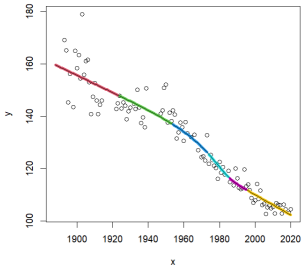
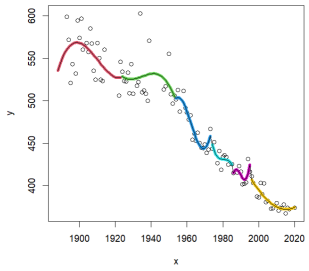
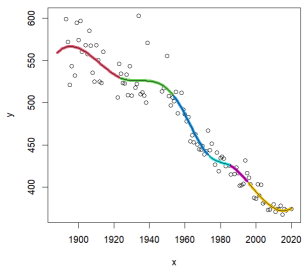
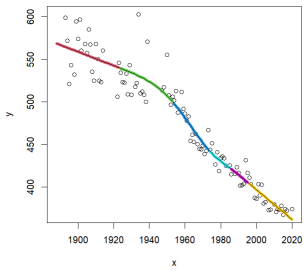
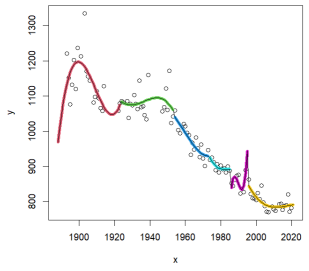
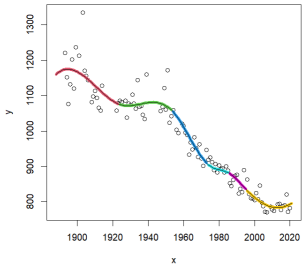
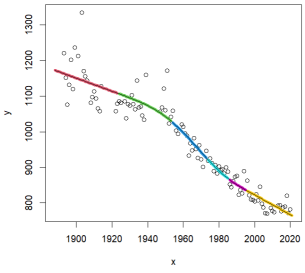


Figure 5: Natural cubic splines fit to the speedskating data. Enforcing the first and last segment to be linear, seems to make the function overall less "wobbly"

6 Appendix A: more results

As promised: the results when using the different datasets. If you have chosen to work with another dataset than the 10.000m which is chosen by default, you can find your answers here.

	piecewise polynomial	splines	natural splines
500m			
1500m			
5000m			
10.000m			

7 Appendix B: some amazing things that happened when working on this project

While working on this project, I came accross two really cool things. I wanted to share my insights, but there wasn't really a place in my report for them... So I made it into an appendix! :)

As I said, I have learned to do webscraping in this project. I had never done it before, but a collegue of mine had. I asked him what the best way would be to learn it, and he told me to use beautifulsoup. He was right after all: it is indeed super easy to collect your own data. To anybody who has the patience to go over this reading until the last appendix, I would advise to have a little more patience: do actually try it! As I said, I think this is quite a valuable skill for a data scientist to be able to collect data from the internet - and it's surprisingly easy to learn :)

While working on the project, I have contributed to a wikipedia page for the first time in my life. After scraping the webpages, I noticed outliers for ice-skating times in 1983: the result for 1500m was unexpectedly high, while the result for 5000m was unexpectedly low. What happened? The columns were switched on wikipedia. I corrected it, so that there were no problems in my data collection anymore.

Let's end this section with some funfacts I came across when I was analysing the data: if I lived a long time ago, I could have won individual distances at the allround championships. On the 500m distance (my personal best is 42.39) and 10.000m (PB: 17:14,70), I could have won until 1954 on the allround championships. On 1500 (PB: 2:06.20) and 5000m (PB: 7:41.85) races, I could have reached the top step of the podium until 1973. This is all to say how admirable it is that skaters were able to skate those times 50 years ago without all technical improvements I could enjoy. Amazing!