One of the themes at useR 2017 in Brussels was "Get involved". People were encouraged to contribute to the community, even when they did not consider themselves R specialists (yet). This could be by writing a package or a blog post. But also by simply correcting typos through pull requests, or sending a tweet about a successful analysis. Bottom line, get your stuff out in the open, share your work!

I felt this urge of getting involved already a year ago, at the useR conference in 2016. Hearing all these people speak about the great work they had done was really inspiring and I wanted to be a part of it. I wanted to try what it was like to develop a package. I knew that what I could do was only minor, I have a full time job as a data scientist of which contributing to the community is not part of the job description. However, I could free up one day a week by working a few hours more on the other days of the week and I could do a little on the train to work every day. I was not sure if I was up to developing a full R package, but I could at least try.

The idea for the *padr* package came at work, where I was dealing with timestamped data a lot. Aggregating data to a higher level and filling missing values took me a lot of time each time I did it. I thought there must be a better way for doing this. I did not have a clear end product in mind, I just started trying stuff and stitched small ideas together to do more complex operations. In the mean time I worked my way through Hadley's *R packages*, and learned about the fundamentals of software design in R. Picking up skills while I went along.

Past January I published the first version of *padr* on CRAN, and I must say that it was quite scary. Putting your product out in the open comes with a lot of imposter feelings. I had thoroughly tested the functions but still was afraid all kinds of bugs would pop-up. Well bugs did pop-up and this appeared to be just fine. I had incredibly nice comments by people suggesting improvements or who even fixed bugs themselves. If there is one lesson I learned from writing package, it is that software does not have to be perfect to be useful. Of course you don't bring junk to CRAN that is full of errors, but once you have tried your best you should just get it in the open and wait for feedback. This would improve the software much faster than spending endless evenings trying to find bugs.

I benefit greatly from the skills I picked up in the process in my day job. My code in data science projects is much cleaner now, trying to write as many functions as I can. Data scientists that have no background in computer science, like myself, can become much better analysts by acquainting themselves with principles of software design. Elegant code will enhance the reproducibilty and shareability of your code. R's package structure is not only great for designing software, it can also be used as an environment for doing analyses. Especially when these have to be rerun on new datasets over and over.

There is still a lot of stuff for me to learn about writing better software and gaining a deeper understanding of the language. However, I no longer think this is intimidating. Rather, it is a great opportunity to improve and improve the software I am writing. You don't need complete mastery to start developing. On the contrary, mastery seems to come through developing. You only truly learn R by trying things you have not tried before.