Date: November 4, 2020
Place: office 102, School of Electrical Engineering, University of Belgrade, Bulevar kralja Aleksandra 73, 11000 Belgrade, Serbia

Interviewer (NM): Nadica Miljković
Interviewee (PP): Professor Predrag Pejović, School of Electrical Engineering, University of Belgrade, he was a guest lecturer at ETH Zurich and Universidad Politécnica de Madrid and PhD at the University of Colorado Boulder, personal webpage, http://tnt.etf.rs/~peja/

Scientific Discipline: Scientific Computing (Numerical and Symbolic Computing)

Transcript (translation in English, originally in Serbian, translated by Nadica Miljković)

NM: The CURE-FAIR questionnaire contains a user story and the first question is your role in relation to your experience in computational reproducibility. For example journal editor, researcher, reviewer, educator/lecturer, etc.
PP: I would say researcher only. For some time I have been using exclusively free software for education and since then I have no problems in computational reproducibility of teaching materials. Therefore, I do not have accumulated results in proprietary software, everything is translated. In my experience, the most important aspect in relation to reproducibility is in my research in numerical computing. I teach students about the importance of numerical reproducibility in master course. This is very important for example when we have to decide whether something is zero or close to zero. For example, to decide whether there is a voltage source loop. When you said reproducibility this came first to my mind.
The next thing that comes across my mind is that the majority of my code was written in Fortran. Then, when Fortran disappeared I had to translate everything in C. This was hard and cumbersome work back in 1993. After that experience I was careful when choosing a platform as I was afraid would I be able to reuse my own results in the future. Then, I had to translate my code in numerical computing in Matlab, then GNU Octave, and in the end Python. Right now I tend to use Julia and Python interchangeably.

NM: If I get it right, your major issue was related to cross-platform reproducibility? I am interested if you had a reproducibility problem as a result of versioning, for example translating code from Python 2 to Python 3?
PP: Yes, there is cross-versioning. However, I find that problem way too easy, so I had no intention to mention it. This year I am translating everything to Python 3 and there are only a few issues that need to be resolved (e.g., print function). Cross-versioning is easy and fast and uncomparable when I had to reuse Fortran code in C.

NM: What is exactly your scientific discipline? Numerical Computing?
PP: My main area is Scientific Computing and my sub-disciplines are Scientific Computing are Numerical Computing and Symbolic Computing.
I also had cross-platform issues in Symbolic Computing. I started to use Derive during my PhD in the USA and at some point it became unavailable. Derive was a very nice small program and

it worked phenomenal. Texas Instruments bought it and then they created derived Derive and then it was discontinued. Back then I had to reuse Derive code in Maxima and to start using Maxima. This happened in 2012 when I was a guest lecturer at Universidad Politécnica de Madrid and I translated everything that I had in Derive to Maxima. It was quite cumbersome.

NM: As I understood well, cross-platform is the greatest problem?
PP: Yes. Also, I have another cross-machine problem. If I have to use the same code that I wrote under Linux on a machine with Windows I need to spend quite a large amount of time in order to reuse a code. For me that is an issue.

NM: I understand. If someone would decide to solve that, what would be a solution? To describe methodology in all tiny details? Are there any scientific papers in your area where authors do not report the version of a program or operating system? Was there any situation when the missing data was important to you and prevented you from repeating someone's else result?
PP: That is the standard scenario. Unfortunately.
Let me show you something. My PhD student Marija Glišić and I uploaded experimental verification on Zenodo. We got a request for that from reviewers. You can see it at https://zenodo.org/record/3732925 for experiment 2, at https://zenodo.org/record/3601544 for experiment 1, and at https://zenodo.org/record/3601399 for simulation. The idea was to place it publicly available and accessible to anonymous reviewers.

NM: Obviously sharing code is an important part of computational reproducibility. Do you think that journals should provide code review (verification)? Would that help? Do you think that scientists send their erroneous code to journals or they just do not explain it?
PP: I think that there are some of them sharing the erroneous code. I placed this on Zenodo with the intention that someone can reuse it and check my results. I strongly believe in verification.

NM: Did your reviewers perform that check?
PP: I have no information on that.
Another thing related to this topic is the work of my student Miloš Novaković. He had an assignment to reuse Marija's Python code in Julia and to compare results. Julia performed expectedly fast, but the most interesting result is comparison - it was identical to the level of bit.

NM: That's perfect. So cross-platform reuse can be successful if you have all relevant data?
PP: Yes. His input data was this Zenodo link.

NM: This probably depends also on the programming language. Obviously Python and Julia are compatible. But, what if you choose some other programming language?
PP: Of course.

NM: Should be there a list of recommended programming languages for scientific computing?
PP: I don't think so. I think that it is related to the IEEE arithmetic standard and if you have this standard implemented then it should be OK. Of course this is not as simple as it seems. It is

because there are intermediate results and if you have a change from 32-bit to 64-bit word at some point then the results will be different. Those are not simple simulations and calculations, they usually last for a couple hours and there are many calculations. In order to get identical results, you practically need to perform all operations identically. This is also conditioned by word length. Python and Julia seem to be very compatible, but this is not necessarily the case with other languages.
To my students at course Software Tools in Electronics (all materials are available online, http://tnt.etf.bg.ac.rs/~oe4sae/) I show how factoriel performs in C and in Python. Algorithmically, those programs are identical. The problem with C is that flags are set in such a way that overflow is not detected. When you try to perform a factorial of 500 you will get something that makes no sense, such as negative value. In Python, the change of type will take place, from integer to long, and the result will be correct. This is something that I would classify as computational reproducibility.

NM: Do you think that we should teach PhD students computational reproducibility?
PP: I teach students at 2nd year of Bachelor studies this example with factoriel at course Software Tools in Electronics (all materials are available online, http://tnt.etf.bg.ac.rs/~oe4sae/). I do not call it reproducibility, that is true, but I show them, you can find it online in course materials.
I must admit that when you said reproducibility I wasn't quite sure what you think by that?

NM: Thank you for your honesty. Many researchers asked me that.
PP: What do you consider to be computational reproducibility? There is another thing that also crossed my mind. It is a problem with compatibility of Linux Ubuntu. I tried the code on two computers and it couldn't work, both Linux Ubuntu GCC compiler. It appeared that one computer was 32 bit OS and another 64 bit OS and that was the problem.

NM: To conclude OS and cross-platform are the biggest issues?
PP: Yes.

NM: If someone would tell you that your code on Zenodo should work in 200 years from now, what would you say? Python and Julia might not exist then.
PP: If you insist on backward compatibility you will be in a constant problem of advancement. That can be both an advantage and disadvantage. You cannot progress if you tie strongly your hardware and software to such strict requests. Simply, I do not think that you can ensure that. I would say that we can split reproducibility into reproducibility in time and in the current moment. We don't have reproducibility in the current moment and the future is on a completely different level. That future reproducibility is not computational, it is document reproducibility. For example, I would love to read my PhD thesis in digital form again. My PhD thesis was prepared in Word for Windows 2.0C 25 years ago.

NM: Do you still keep those files?
PP: Sure. I have it all on floppy disks. In order to transfer it to my current computer, I had to find an old computer (you can check behind yourself). Now, I decided to keep this old computer, just

in case.

… follows a talk on problems with CD-based storage …

PP: To conclude, I wouldn’t insist on backward compatibility. Rather, I would insist on reasonable decisions. Fine example of reasonable decision is done with Python versioning. They kept version 2 for a long time in parallel with 3 and the transition is very well documented. I think that is why my transition from 2 to 3 is simple. Emulator of and “old” and outdated programs together with well prepared and detailed documentation would enable reproducibility in time.

.... follows a sample case that Latex files from 70is and 80is can be reproduced (an experience from a colleague mathematician) ...

NM: Overall, this should be taken care of by institutions. Information Centres and Data Centres.
PP: This is a well known problem. However, there are other sides of the story. Many lobi groups would keep vendor-lock-in as they have an interest in it.

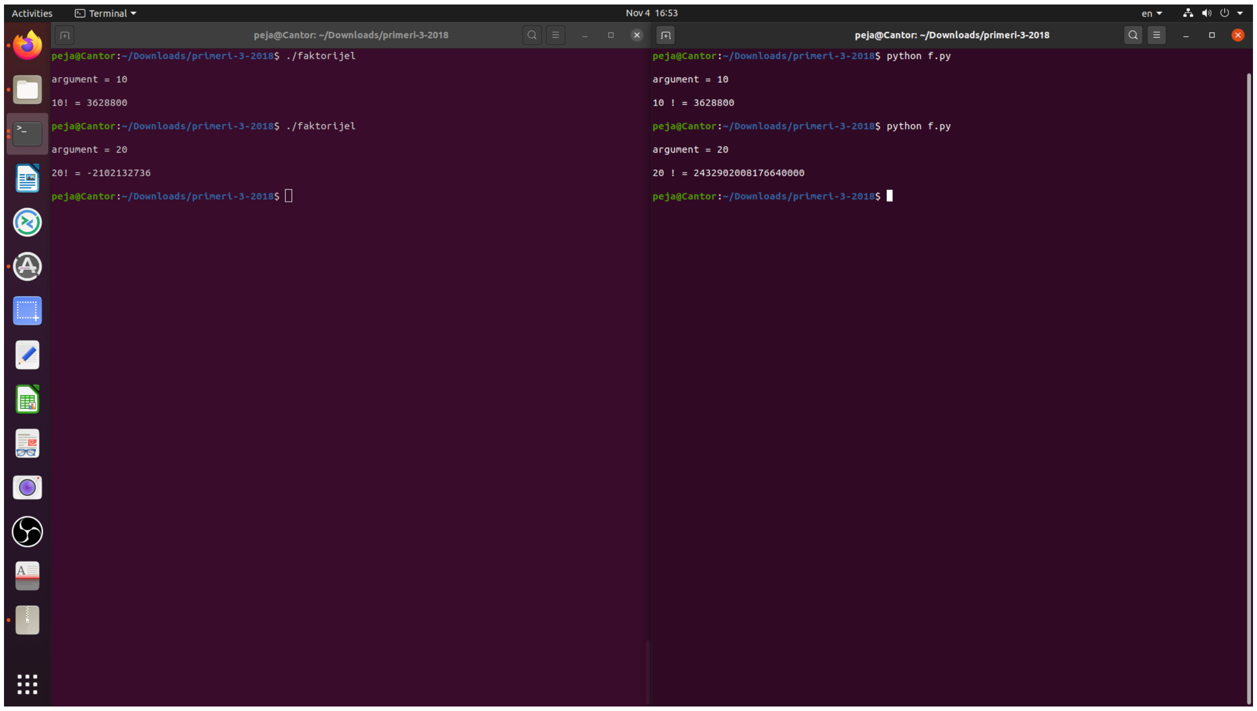
… follows a chit chat on author rights …

NM: Do you think that all scientific software should be free software?
PP: This is straightforward. Yes, I do. Many scientists do. My colleagues from ETH in Zurich even developed proprietary software under Linux as they found it more convenient. That is a personal decision. I decided not to use it and that is a consequence of my bad experience. I feel humiliated as their salespersons are insulting my intelligence. No nice way to say this. They do not have complete knowledge and they try to earn some profit from scientists that are mostly underpaid.
It is also cost/benefit. What do you get with proprietary software that you can not get with free software regardless ideology?

NM: I would like to ask something about review. You said that you shared your research on Zenodo for reviewers expecting that reviewers would use it. Have you ever as a reviewer tested someone's code and did you get the chance to test it?
PP: A large amount of my time is devoted to review. I just got two papers for review both in scientific computing. In most cases, I get diagrams in the paper that I cannot reproduce. I intentionally send my code in my papers so that reviewers can check it, and since I believe that this way we are creating a standard. I admit that I have never asked authors for a code. Maybe it is how I am raised as I consider such requests impolite. There are several factors, (1) the first one is compassion to authors as I know how much effort they have put on and (2) I always think what their disclosure could be on some commercial grant.
I perform my review visually and if I see something that appears unexpected then I ask them about that. I have never seen a programming code in all my reviews, although I consider the code in Scientific Computing the most important part.

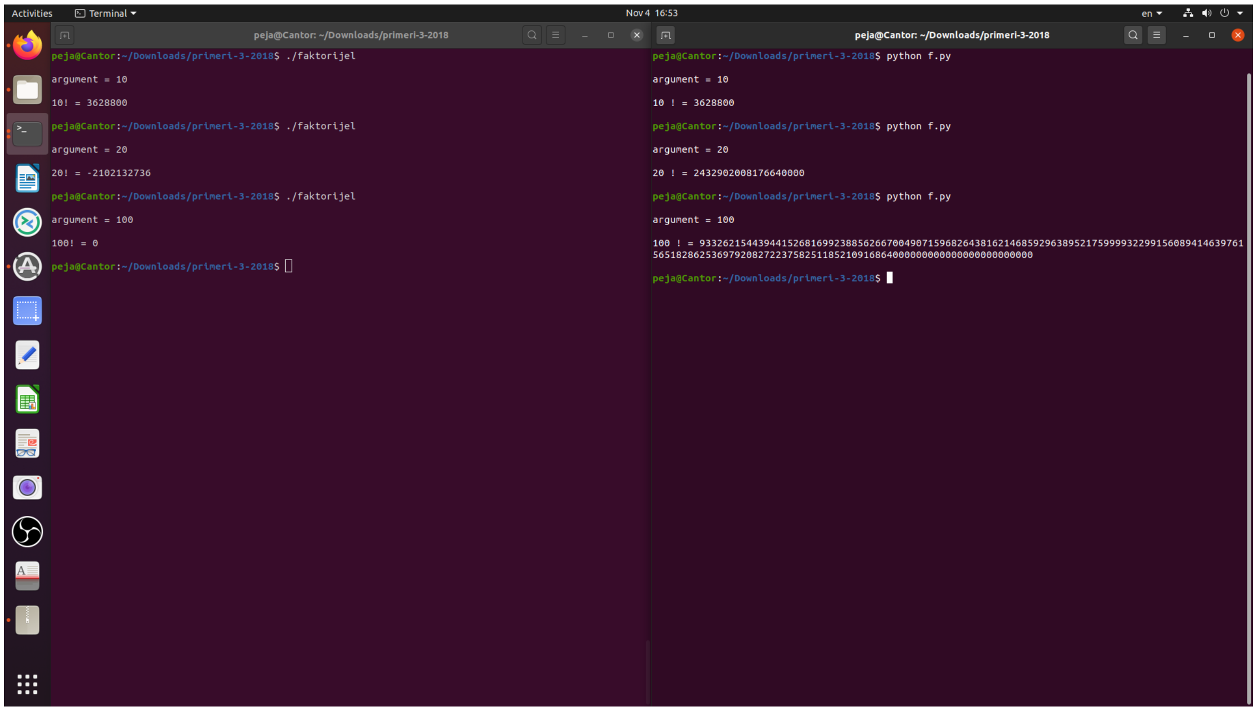
NM: I understand, you cannot force researchers to give their code, but it is desirable. Maybe journals can start asking them about code? Not forcing of course.
PP: I have concrete examples on Zenodo. There I share code under CC Attribution Share Alike - I consider it to be equivalent to GPL. In these examples, all results are worthless without a code.

In the following PrtScs, factorials in C and Python programming languages are compared.


Activities
Terminal
Nov 4 16:53
en
peja@Cantor: ~/Downloads/primeri-3-2018
peja@Cantor: ~/Downloads/primeri-3-2018

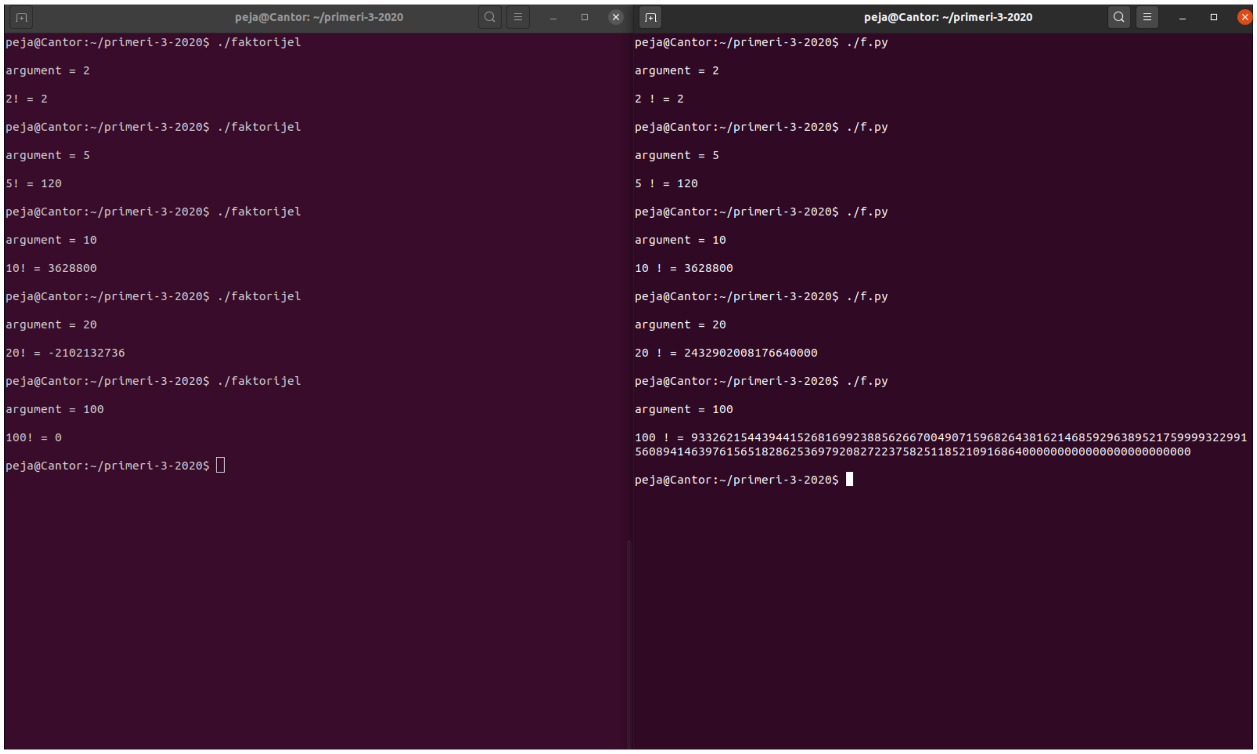
```
peja@Cantor:~/Downloads/primeri-3-2018$ ./faktorijel
argument = 10
10! = 3628800
peja@Cantor:~/Downloads/primeri-3-2018$ ./faktorijel
argument = 20
20! = -2102132736
peja@Cantor:~/Downloads/primeri-3-2018$
```

```
peja@Cantor:~/Downloads/primeri-3-2018$ python f.py
argument = 10
10 ! = 3628800
peja@Cantor:~/Downloads/primeri-3-2018$ python f.py
argument = 20
20 ! = 2432902008176640000
peja@Cantor:~/Downloads/primeri-3-2018$
```


Activities
Terminal
Nov 4 16:53
en
peja@Cantor: ~/Downloads/primeri-3-2018
peja@Cantor: ~/Downloads/primeri-3-2018

```
peja@Cantor:~/Downloads/primeri-3-2018$ ./faktorijel
argument = 10
10! = 3628800
peja@Cantor:~/Downloads/primeri-3-2018$ ./faktorijel
argument = 20
20! = -2102132736
peja@Cantor:~/Downloads/primeri-3-2018$ ./faktorijel
argument = 100
100! = 0
peja@Cantor:~/Downloads/primeri-3-2018$
```

```
peja@Cantor:~/Downloads/primeri-3-2018$ python f.py
argument = 10
10 ! = 3628800
peja@Cantor:~/Downloads/primeri-3-2018$ python f.py
argument = 20
20 ! = 2432902008176640000
peja@Cantor:~/Downloads/primeri-3-2018$ python f.py
argument = 100
100 ! = 93326215443944152681699238856266700490715968264381621468592963895217599993229915608941463976
156518286253697920827223758251185210916864000000000000000000000000
peja@Cantor:~/Downloads/primeri-3-2018$
```



```
peja@Cantor:~/primeri-3-2020$ ./faktorijel

argument = 2

2! = 2

peja@Cantor:~/primeri-3-2020$ ./faktorijel

argument = 5

5! = 120

peja@Cantor:~/primeri-3-2020$ ./faktorijel

argument = 10

10! = 3628800

peja@Cantor:~/primeri-3-2020$ ./faktorijel

argument = 20

20! = -2102132736

peja@Cantor:~/primeri-3-2020$ ./faktorijel

argument = 100

100! = 0

peja@Cantor:~/primeri-3-2020$
```

```
peja@Cantor:~/primeri-3-2020$ ./f.py

argument = 2

2 ! = 2

peja@Cantor:~/primeri-3-2020$ ./f.py

argument = 5

5 ! = 120

peja@Cantor:~/primeri-3-2020$ ./f.py

argument = 10

10 ! = 3628800

peja@Cantor:~/primeri-3-2020$ ./f.py

argument = 20

20 ! = 2432902008176640000

peja@Cantor:~/primeri-3-2020$ ./f.py

argument = 100

100 ! = 93326215443944152681699238856266700490715968264381621468592963895217599993229915
5608941463976156518286253697920827223758251185210916864000000000000000000000000

peja@Cantor:~/primeri-3-2020$
```

Date: November 5, 2020

NM: Do you approve my transcript of the interview? Would you like to add something?
PP: Yes, I do approve the transcript.
I would like to point out that I think that formats are crucial. Secondly, it is needed to archive software versions somehow. When you report program results, one should at least say what version of the program it is and on which platform it works, it is possible that under Windows and Linux the code works differently. With free software, archiving versions is legal execution, with proprietary it is not. For outdated OSs, it would be convenient to have an emulator as those OSs won't be here forever.
The idea that crosses my mind is could we use any software with a standard command to list "environment parameters", machine parameters and program version together with its modules in order to document the type of a machine (by configuration) and the version of the program used to create results? Could any central body archive the relevant ones program in terms of making it available to older versions? This can be done for free software only. The problem is clear, the lobby groups and the seller's vendor-lock-in ideology will not be supportive.

NM: Do you agree that we make this transcript available on Zenodo?
PP: Yes, I do.