Why Julia?

Proposal to adopt **Julia** for the **GHRSS survey** codebase.

Ujjwal Panda

: astrogewgaw.com
: github.com/astrogewgaw
: ujjwalpanda97@gmail.com

Last updated: April 12, 2022.

As radio telescopes produce data at both unprecedented rates and resolutions, the task of processing such data has been handed over to highly automated data processing pipelines. Initially, these pipelines were nothing more than shell scripts that essentially automated the same commands that would have been entered manually by the user; now, these have become large codebases. As many telescopes shift to conducting their searches for pulsars and radio transients commensal to other observations[1], the need for processing data in real-time has emerged; as a result, many data pipelines run entirely on GPUs or clusters of GPUs.

However, the need for high performance has given rise to a disturbing trend in the realm of pulsar astronomy software: in order to make data pipelines that are optimised for performance, many pulsar astronomy groups have taken to writing their entire pipelines from scratch[2]. While this guarantees performance (since all algorithms are optimised to work together and written in the same framework), it has lead to pipelines turning into large and monolithic codebases. These become difficult to extend, maintain, or even document.

The solution to these problems is not immediately apparent, since some of them harbor ancient demons from the domain of scientific computing. One of them is the notorious **two language problem**. Programmers start developing new analysis packages and algorithms in languages like **Python**. These **high level languages** tend to ease development by providing abstraction over the nitty-gritty details of the machine[3]. However, when deploying their code in production, they resort to **low level languages**, such as **C**, **C++**, or **Fortran**. Since these have access to a machine's internal mechanisms, they allow programmers to write highly performant code. The only solution for the problem, until recently, has been to program the bottlenecks of a algorithm in a low level language, and then wrap the code in a high level language. Pulsar astronomy is no stranger to this trend[4].

The aim of this document is to provide a solution to some of these problems, by introducing **Julia** to the domain of pulsar astronomy. To the author's

This document uses **Tufte-LaTeX**, LaTeX a class inspired by the works of Edward Tufte. Edward Tufte is a statistician and artist, and Professor Emeritus of Political Science, Statistics, and Computer Science at Yale University. He wrote, designed, and self-published 4 classic books on data visualization, which became the basis for the **Tufte style**.

[1] E.g.: the CRAFT and **realfast** surveys

[2] A notable exception to this is the **AstroAccelerate** organisation, which shares optimised versions of several pulsar data algorithms for the GPU, for use by other groups.

[3] An example of such details is memory management; such languages abstract this away by using **garbage collectors**, which deallocate unused memory automatically.

[4] A recent example is **RIPTIDE**, which wraps kernels written in C++ with Python, thus providing a high-level but performant abstraction over the Fast Folding Algorithm (**FFA**)

knowledge, this language has not been adopted by pulsar astronomers. It will be the author's attempt to make a case for doing so. Do note that this is a *living* document: it is meant to evolve over time, and it exists as a repository on GitHub here: **https://github.com/astrogewgaw/whyjulia**. Feel free to suggest changes and give feedback.