AN OPEN SOURCE FRAMEWORK FOR INTERACTIVE, COLLABORATIVE AND REPRODUCIBLE SCIENTIFIC COMPUTING AND EDUCATION

Fernando Perez
UC Berkeley

Brian E Granger
Cal Poly San Luis Obispo

We propose to build open source tools to support the various phases of computational work that are typical in scientific research and education. Our tools will span the entire life-cycle of a research idea, from initial exploration to publication and teaching. They will enable reproducible research as a natural outcome and will bridge the gaps between code, published results and educational materials. This project is based on existing, proven open source technologies developed by our team over the last decade that have been widely adopted in academia and industry.

1. TOOLS FOR THE LIFECYCLE OF COMPUTATIONAL RESEARCH

Scientific research has become pervasively computational. In addition to experiment and theory, the notions of simulation and data-intensive discovery have emerged as third and fourth pillars of science [5]. Today, even theory and experiment are computational, as virtually all experimental work requires computing (whether in data collection, pre-processing or analysis) and most theoretical work requires symbolic and numerical support to develop and refine models. Scanning the pages of any major scientific journal, one is hard-pressed to find a publication in any discipline that doesn't depend on computing for its findings.

And yet, for all its importance, computing is often treated as an afterthought both in the training of our scientists and in the conduct of everyday research. Most working scientists have witnessed how computing is seen as a task of secondary importance that students and postdocs learn "on the go" with little training to ensure that results are trustworthy, comprehensible and ultimately a solid foundation for reproducible outcomes. Software and data are stored with poor organization, documentation and tests. A patchwork of software tools is used with limited attention paid to capturing the complex workflows that emerge, and the evolution of code is often not tracked over time, making it difficult to understand how a result was obtained. Finally, many of the software packages used by scientists in research

are proprietary and closed-source, preventing the community from having a complete understanding of the final scientific results. The consequences of this cavalier approach are serious. Consider, just to name two widely publicized cases, the loss of public confidence in the "Climategate" fiasco [4] or the Duke cancer trials scandal, where sloppy computational practices likely led to severe health consequences for several patients [3].

This is a large and complex problem that requires changing the educational process for new scientists, the incentive models for promotions and rewards, the publication system, and more. We do not aim to tackle all of these issues here, but our belief is that a central element of this problem is the nature and quality of the software tools available for computational work in science. Based on our experience over the last decade as practicing researchers, educators and software developers, we propose an integrated approach to computing where the entire life-cycle of scientific research is considered, from the initial exploration of ideas and data to the presentation of final results. Briefly, this life-cycle can be broken down into the following phases:

- **Individual exploration:** a single investigator tests an idea, algorithm or question, likely with a small-scale test data set or simulation.
- **Collaboration:** if the initial exploration appears promising, more often than not some kind of collaborative effort ensues.
- **Production-scale execution:** large data sets and complex simulations often require the use of clusters, supercomputers or cloud resources in parallel.
- **Publication:** whether as a paper or an internal report for discussion with colleagues, results need to be presented to others in a coherent form.
- **Education:** ultimately, research results become part of the corpus of a discipline that is shared with students and colleagues, thus seeding the next cycle of research.

In this project, we tackle the following problem. **There are no software tools capable of spanning the entire lifecycle of computational research.** The result is that researchers are forced to use a large number of disjoint software tools in each of these phases in an awkward workflow that hinders collaboration and reduces efficiency, quality, robustness and reproducibility.

These can be illustrated with an example: a researcher might use Matlab for prototyping, develop high-performance code in C, run post-processing by twiddling controls in a Graphical User Interface (GUI), import data back into Matlab for generating plots, polish the resulting plots by hand in Adobe Illustrator, and finally paste the plots into a publication manuscript or PowerPoint presentation. But what if months later the researcher realizes there is a problem with the results? What are the chances they will be able to know what buttons they clicked, to reproduce the workflow that can generate the updated plots, manuscript and presentation? What are the chances that other researchers or students could reproduce these steps to learn the new method or understand how the result was obtained? How can reviewers validate that the programs and overall workflow are free of errors? Even if the researcher successfully documents each program and the entire workflow, they have to carry an immense cognitive burden just to keep track of everything.

We propose that the open source IPython project [9] offers a solution to these problems; a single software tool capable of spanning the entire life-cycle of computational research. Amongst high-level open source programming languages, Python is today the leading tool for general-purpose source scientific computing (along with R for statistics), finding wide adoption across research disciplines, education and industry and being a core infrastructure tool at institutions such as CERN and the Hubble Space Telescope Science Institute [10, 2, 12]. The PIs created IPython as a system for interactive and parallel computing that is the *de facto* environment for scientific Python. In the last year we have developed the IPython Notebook, a web-based *interactive computational notebook* that combines code, text, mathematics, plots and rich media into a single document format (see Fig. 1.1). The IPython Notebook was designed to enable researchers to move fluidly between all the phases of the research life-cycle and has gained rapid adoption. It provides an integrated environment for all computation, without locking scientists into a specific tool or format: Notebooks can always be exported into regular scripts and IPython supports the execution of code in other languages such as R, Octave, bash, etc. In this project we will expand its capabilities and relevance in the following phases of the research cycle: interactive exploration, collaboration, publication and education.

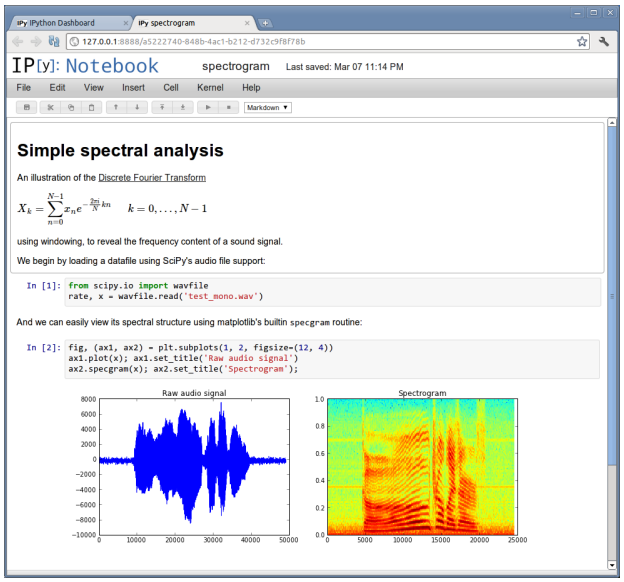

FIGURE 1.1. The web-based IPython Notebook combines explanatory text, mathematics, multimedia, code and the results from executing the code.

2. PRIOR WORK

In this section we describe the existing landscape of software tools that researchers use in computational work. We highlight the central problem this project will address, namely, the large number of disjoint software tools researchers are forced to use as they move through the different phases of research. We then detail prior work we have done in developing IPython, setting the stage for our proposed future work.

2.1. **The patchwork of existing software tools.** For **individual exploratory work**, researchers use various interactive computing environments: Microsoft Excel, Matlab, Mathematica, Sage [13], and more specialized systems like R, SPSS and STATA for statistics. These environments combine interactive, high-level programming languages with a rich set of numerical and visualization libraries. The impact of these environments cannot be overstated; they are used almost universally by researchers for rapid prototyping, interactive exploration and data analysis and visualization. However, these environments have a number of limitations: (a) some of them are proprietary and/or expensive (Excel, Matlab, Mathematica), (b) most (except for Sage) are focused on coding in a single, relatively slow, programming language and (c) most (except for Sage and Mathematica) do not have a document format that is rich, i.e., that can include text, equations, images and video in addition to source

code. While the use of proprietary tools isn't a problem *per se* and may be a good solution in industry, it is a barrier to scientific collaboration and to the construction of a common scientific heritage. Scientists can't share work unless all colleagues can purchase the same package, students are forced to work with black boxes they are legally prevented from inspecting (spectacularly defeating the very essence of scientific inquiry), and years down the road we may not be able to reproduce a result that relied on a proprietary package. Furthermore, because of their limitations in performance and handling large, complex code bases, these tools are mostly used for prototyping: researchers eventually have to switch tools for building production systems.

For **collaboration**, researchers currently use a mix of email, version control systems and shared network folders (Dropbox, etc.). Version control systems (Git, SVN, CVS, etc.) are critically important in making research collaborative and reproducible. They allow groups to work collaboratively on documents and track how those documents evolve over time. Ideally, all aspects of computational research would be hosted on publicly available version control repositories, such GitHub or Google Code. Unfortunately, the most common approach is still for researchers to email documents to each other. This form of collaboration makes it nearly impossible to track the development of a large project and establish reproducible and testable workflows. When it works at all, it most certainly doesn't scale beyond a very small group, as painfully experienced by anyone who has participated in the madness of a flurry of email attachments.

For **production-scale execution**, researchers are forced to turn away from the convenient interactive computing environments to compiled code (C/C++/Fortran) and parallel computing libraries (MPI, Hadoop), as most interactive systems don't provide the performance necessary for large-scale work and have primitive parallel support. These tools are difficult to learn and use and require large time investments. We emphasize that before production-scale computations begin, the researchers have already developed a mostly functional prototype in an interactive computing environment. Turning to C/C++/Fortran for production means starting over from scratch and maintaining at least two versions of the code moving forward. Furthermore, data produced by the compiled version has to be imported back into

the interactive environment for visualization and analysis. The resulting back-and-forth, complex workflow is nearly impossible to capture and put into version control systems, again making the computational research difficult to reproduce.

For **publications** and **presentations**, researchers use tools such as LaTeX, Google Docs or Microsoft Word/PowerPoint. The most important attribute of these tools in this context is that they don't integrate well with version control systems (LaTeX excepted) and with other computational tools. Digital artifacts (code, data and visualizations) have to be manually pasted into these documents, so that the same content is duplicated in many different places. When the artifacts change, the documents quickly become out of sync.

2.2. **The IPython Notebook.** The open-source IPython project is the primary focus of this project's proposed activities. PI Perez created IPython in 2001 and was joined by PI Granger in 2004; both continue to lead the project today. Together, they have grown the project into a vibrant open source community that has an active development team of over 150 contributors from academia and industry that collaborate via the GitHub website[1] and release new versions of the project approximately every 6 months.

IPython has had a significant impact on scientific computing across a wide range of disciplines, a fact that is seen in the expansive user base [2] that includes individuals and small groups from nearly every discipline, large scientific collaborations (Hubble Space Telescope, Chandra X-Ray Telescope, Square Kilometer Array, CERN, etc.), companies (Microsoft Azure, Visual Numerics PyIMSL, Enthought, etc.) and educational initiatives such as the Sloan Foundation funded Software Carpentry Project.

IPython provides open source tools for interactive computing in Python. Historically, IPython has provided an enhanced interactive shell that is now the *de facto* working environment for scientific and technical computing in Python. More recently, the IPython development team has expanded its efforts to develop the IPython Notebook, a web-based environment for Python, R, shell scripts and other languages. At its core, the IPython Notebook is a system for writing and running code in a web browser, with support for convenient code

[1] http://github.com/ipython
[2] http://wiki.ipython.org/Projects_using_IPython

development (e.g. syntax coloring). However, the Notebook goes beyond mere code, by enabling users to build documents that combine live, runnable code with visualizations, text, equations, images and videos. The Notebook provides everything needed for **interactive exploration** at the user's fingertips.

The Notebook document format has been carefully designed to support **collaboration**. Most importantly, by being version control friendly, users can preserve with the Notebook a full historical record of a computation, its results and accompanying material including embedded images and visualizations. Posting Notebooks on public version control repositories, such as GitHub, enables large groups of people to collaborate on the documents.

Because the Notebook integrates code with text, equations and multimedia, it is also an ideal platform for **publication** and **presentation**. The same document that is used for interactive exploration and production-scale computing can also be used to generate publications, documentation and presentations. Initial work has begun to enable Notebook documents to be exported to a wide range of formats. Work is also underway to enable Notebooks to be converted to PowerPoint style presentations with the click of a button, with the added twist that these presentations support live computations. This makes the Notebook an ideal teaching tool: multiple courses and workshops now use the Notebook both as the execution environment and as the file format for sharing and publishing; e.g. the Berkeley Python boot-camp, a new course for computational genomics taught at the BEACON Center at Michigan State University[3], an NIH-funded summer workshop also at MSU on the analysis of next-generation sequencing data[4] and the Software Carpentry Project [5].

Even today, the Notebook is transforming the workflow of computational research. As the Notebook supports multiple programming languages (Python, R, Octave, Bash, Perl, etc.) and integrates code with text, equations and rich media, researchers can use a single tool throughout the different phases of research. This enables collaboration and reproducibility to emerge as natural research outcomes. Since the Notebook captures the entire computational workflow, even if it includes multiple languages and system shell commands, it

[3] http://ged.msu.edu/angus/beacon-2012
[4] https://github.com/ngs-docs/ngs-notebooks
[5] http://software-carpentry.org

is much easier to share it with others, who can easily re-run an entire computation. Full reproducibility can be obtained by combining notebooks with virtual machine images deployed on cloud resources, as demonstrated in a recent collaboration between the IPython team, computer scientists at MIT and microbial ecologists at the University of Colorado that resulted in an "executable paper" that can be run by anyone to replicate the results [11].

3. Team Background

3.1. **The IPython team.** PIs Perez and Granger are both physicists whose research interests span a broad range of problems, from neuroscience and numerical algorithms to atomic physics and quantum computing. A constant theme of their research career has been a preoccupation with building high-quality computational tools. F. Perez and B. Granger met in graduate school at the University of Colorado, Boulder, and have collaborated closely since 2004. Perez started the IPython project in 2001, and in 2004 Granger joined the project by leading the development of parallel computing capabilities in IPython while a professor at Santa Clara University. Under his supervision, B. Ragan-Kelley completed a senior thesis project in computational physics on the design and implementation of IPython's parallel architecture. B. Ragan-Kelley has continued to work closely with the PIs since, and he will be the project's lead development engineer once he completes his PhD at UC Berkeley in December 2012. The three of us (Perez, Granger and Ragan-Kelley) continue to actively lead the development of the IPython project.

While we were all trained as physicists without any software engineering education, in our interaction with the world of open source developers we have learned and adopted rigorous software engineering practices that we follow to ensure IPython remains a robust and high-quality project even as it grows. All proposed contributions to IPython (even those of the core team) go through a rigorous peer-review process using the *pull request* mechanism on the GitHub website and no code can be committed to the project until it has passed an automated battery of almost 1600 tests. The project has extensive documentation, and it continues to attract both avid users and a growing community of developers; for version 0.13 we worked for 6 months and made a release on 7/1/2012:

- $3\frac{1}{2}$ months later, this version has been downloaded over 133,000 times[6].
- In this cycle, over 1100 separate issues (bugs and new features) were closed.
- We received contributions from 62 separate authors.
- These changes combined represent over 114,000 lines.

IPython is estimated to require 18 person-years and $2,400,000 to develop[7]. These results have been obtained with minimal funding and pushing our team beyond sustainable limits; IPython has received only one formal grant in 2011-2012, plus a few small consulting contracts over the years. But this is not a sensible strategy for the long term, and we are convinced that with robust funding for the core team we can have an even more significant impact in scientific computing for all disciplines.

In addition to the above three individuals, our budget also names explicitly one postdoctoral scholar and two scientists from the UC Berkeley Brain Imaging Center (BIC), where PI Perez works. Paul Ivanov is currently a Berkeley PhD student in Vision Science who is also a core IPython developer; part of his PhD thesis involves the development of reproducible research tools in modeling the visual system using IPython. We expect him to be hired as a postdoc for this project after his graduation (planned for December 2012). P. Ivanov has a long track record of engaging our user community very effectively, and we foresee his role in the project as not only doing core development, but continuing to play this critically important role of community engagement and evangelism. We expect he will work especially on user-facing areas of the project such as tutorials, documentation and the website, as well as traveling more than other members to conferences and workshops.

The two BIC scientists who are named in the project, Jean-Baptiste Poline and Matthew Brett, have a long track record in statistical analysis of neuroimaging data and in open source development [1, 8, 15]. They founded the open source Neuroimaging in Python project[8] which M. Brett continues to lead, and have collaborated with F. Perez on multiple projects involving open source Python tools for scientific computing since 2005. They will collaborate

[6] This number is a significant *undercount* of actual utilization, as many users can download the project via alternate channels we have no statistics for.

[7] Values generated using David A. Wheeler's 'SLOCCount' open source code analysis program.

[8] http://nipy.org

with Professor Jonathan Taylor (Statistics Dept., Stanford) on the development of executable lecture notes in applied statistics; Prof. Taylor is the author of the R language support in IPython and has agreed to participate in this project as a consultant.

We have budgeted room for one more postdoctoral scholar to work on the project, and have potential candidates that we can draw from for this position, from a number of late-stage PhD students who have made high-quality contributions to IPython over the last few years.

In summary, we have a team with extensive experience producing proven results on the class of problems we aim to tackle in this project, and with an established track record of high-functioning collaboration that has led to the current success of the IPython project. However we must stress that, for all our success in building IPython into a robust project with limited resources, the team regularly feels the strain of not having any stable support. We can't fund any of our talented developers to spend dedicated time on the project, to travel to conferences or meet us for development work, and must therefore rely strictly on their willingness to spend their spare time and resources on IPython. We are critically concerned about the potential loss of several key developers whom, once they graduate from their PhD studies, will likely find much less time to devote to the project. But a number of them have expressed a keen interest in continuing to develop IPython, if only we could offer them postdoctoral/engineering positions for at least a few years. Similarly, PIs Perez and Granger must juggle their regular research and teaching responsibilities on other topics against working on IPython, which causes us to often become unresponsive towards our developer community. This has a serious negative impact on the project, as contributors who find no feedback from the core team are likely to simply disengage and search other projects with more responsive teams. Finally, despite our success so far, it is clear that the really high-impact problems are yet to be tackled, and for those we need the ability to spend serious, dedicated time working. The objectives that form this proposal have a lot of value, but they are beyond our ability to tackle by simply scraping spare time in between other commitments and sporadic bursts of activity during holiday breaks.

3.2. **Ecosystem of collaborators.** Our team has also established a number of important collaborations with partners in academia and industry, that will be important assets in this effort. To name a few:

- The software team at the Hubble Space Telescope Science Institute currently leads the development for the Python data visualization library matplotlib [6]. Since most data visualizations in IPython are done with matplotlib, we maintain a close working relation with the matplotlib team that dates back to 2002.
- Enthought Inc. is an Austin, Tx. company founded by one of the creators of the SciPy project [7] that has funded IPython development in the past and which distributes IPython as part of their product *Enthought Python Distribution.*
- Microsoft Corporation started deploying IPython in 2010 as part of their open source Python Tools for Visual Studio project. We have continued collaborating with them, and currently provide tutorials and documentation on how to use the IPython Notebook as a comprehensive analysis environment in the Microsoft Azure cloud computing platform[9].
- Continuum Analytics is another scientific Python, Austin-based company focused on a web-based Big Data analysis platform, that also distributes IPython as part of their *Anaconda* Python distribution.
- The Software Carpentry project that teaches best computational practices to scientists across the world has adopted IPython Notebook as its core teaching and delivery platform, and provides constant critical feedback to us based on their field experience[10].
- The NumFOCUS Foundation[11] was created in 2012 to promote the use of accessible and reproducible computing in science and technology. PI Perez is a founding member of NumFOCUS and member of its board of directors, and IPython is one of the core projects that NumFOCUS aims to support and promote.

[9] http://www.windowsazure.com/en-us/develop/python/tutorials/ipython-notebook
[10] http://software-carpentry.org/2012/10/transitioning-to-the-ipython-notebook
[11] http://numfocus.org

This (incomplete) list shows how our team, in addition to our daily activities as scientists at UC Berkeley and Cal Poly San Luis Obispo, has very strong connections with major actors in the space of open source scientific computing. These ongoing partnerships will strengthen our work in IPython as they have in the past.

4. Research Approach

We will develop new capabilities in IPython so scientists can fluidly transition between the various phases of computational work as the problem demands, without artificial barriers as imposed by today's software tools. Our **specific aims** are to:

(1) Develop the IPython Notebook Server into a multi-user application with sharing, collaboration and publishing features.
(2) Provide support for interactive widgets embedded in IPython Notebook documents that allow users to visually manipulate computational results.
(3) Develop the IPython Notebook file format and supporting tools to facilitate the sharing, reuse and dissemination of computational work.
(4) Produce a collection of executable lecture notes written as IPython Notebooks, as companions to the *Introduction to Applied Statistics* course at Stanford University.
(5) Continue the maintenance and stewardship of the IPython project as a vibrant and active example of open source development.

We now describe our approach to these objectives and highlight how they address the core problems in scientific computing described in §1.

4.1. **A collaborative multiuser IPython Notebook server.**

We will enhance the Notebook server to provide a platform for **collaborative computing** between scientists that also facilitates the **publication** and **education** stages of computational work described in §1.

The IPython Notebook Server is a single-user application that runs a web server to which the user's browser connects. The web browser is the user interface where code is executed and the notebook document is edited. While the current version allows multiple users to

connect simultaneously, it does not distinguish between those users in any meaningful manner: users cannot create their own private notebooks and projects, there is minimal security to segregate/sandbox user activity, and sharing and collaboration options are primitive.

In this project, we will redesign the Notebook Server to be a true multi-user application. This will not prevent single-user mode from working, e.g. when a user just wants to work on a local project on his or her laptop. But a proper multiuser server will allow users to securely deploy the Notebook in settings such as research groups or classrooms. This server will allow users to log in with their authentication credentials and create new projects, where a project corresponds to a version-controlled directory on the server's file system. The user interface of the Notebook will then include a dashboard page, which shows all of a user's projects, and a project overview page, which shows the Notebooks for any given project, along with other scripts and data files.

By providing each user with private projects and notebooks, it will be possible to expose more sophisticated sharing options. A user will have the ability to share any project or Notebook in her account with other users, with fine-grained control over the level of visibility and access permissions granted. Furthermore, users will be able to publish static versions of any notebook as an HTML page, a slideshow or a rendered PDF.

With these developments, the IPython Notebook will become an enabler of collaborative work in scientific computing, whether for the members of a research group, for colleagues across the world or for educators wanting to deploy the system in a classroom setting.

4.2. **Interactive computational elements in IPython Notebooks.** Since the IPython Notebook is implemented as an application that renders inside of a web browser, we have at our disposal all of the rich media and interactive capabilities that modern browsers provide. The Notebook document format and architecture have been designed from the start to take advantage of this fact. As part of this project, we will add support for notebooks to contain interactive graphical user interface (GUI) elements such as sliders, buttons, selection lists, etc., that can control computations. We will provide a library that enables non-expert users

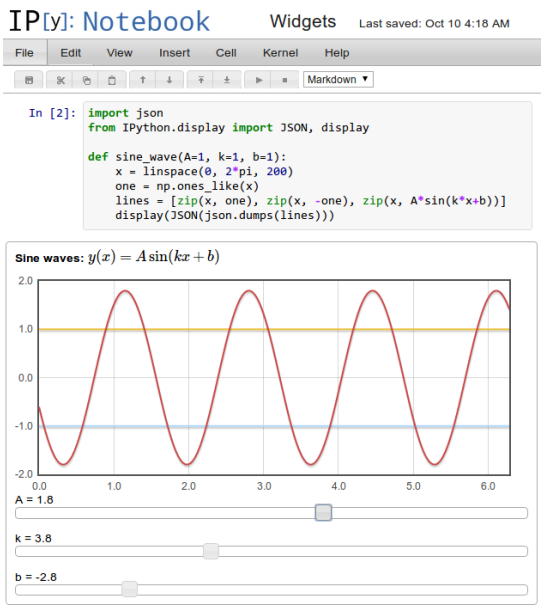

FIGURE 4.1. Prototype of interactive widgets in the IPython notebook.

to add interactive graphical support to any of their analysis codes, simple enough to be used even during ad-hoc exploratory work.

To illustrate this capability, consider wanting to see how the values of the parameters A, k and b affect the plot of a simple sine wave of the form $y(x) = A\sin(kx + b)$. With this new capability in place, users will be able to write a small plotting routine indicating the range of values to offer for A, k and b, and IPython will automatically display sliders for all three with these ranges, enabling the user to manipulate the values with the mouse and immediately see the updated plot. We have already prototyped this functionality, as illustrated in Fig. 4.1, but the current implementation requires the writing of delicate and hard to debug low-level code, making it a poor solution for general use.

This has immediate applications both for the **individual exploration** and **education** phases of the computational cycle, as it makes it very easy to write and deploy customized GUIs that control a specific fragment of code. Furthermore, this will provide the interactive benefits of GUIs for exploration and education, while remaining embedded in a file format that is amenable to validation, version control and sharing. Notebook files with these controls can be shared just like any other file, and the recipient will immediately be able to also perform the same interactive explorations without having to install any additional tools.

4.3. **An open file format for the sharing of computational results.** The IPython Notebook file uses a format that is human-readable yet easy to automatically parse, stored on-disk as a JSON data structure. It contains in a single file a representation of all the text, code, equations and images. To make this file format more useful for **publication** and **education**, we propose to develop tools that can export Notebooks to other formats: LaTeX/PDFs for written publications, HTML for posting on the Internet and blogs, slideshow presentations, etc. In addition to these export capabilities, we will create modular filters that can transform Notebook documents in various ways. For example, one might want to remove all source code from a Notebook, leaving only the explanatory text, which is useful in a classroom setting to provide exercises without the solution code. We have already developed prototypes of this export and filter architecture, but this draft needs to be turned into robust, tested, production code that is included as part of IPython itself, rather than a separate mini-project. This work will transform the Notebook document format into a universal format for the publication and presentation of computational results across disciplines and even programming languages.

It is important to stress that, while we originally developed the Notebook as a Python project, we have taken great care in ensuring that the Notebook can be useful for storing and sharing computational work in other programming languages. Recently, the developers of the new Julia language for scientific computing[12] have decided to adopt the IPython Notebook as the default web-based system for interactive computing in Julia. As part of this project, we will collaborate with the Julia team to ensure that the file format and supporting architecture of the Notebook system truly meets the needs of other languages. At the 2012 SciPy conference, PI Perez already had detailed conversations with the core Julia authors, and we remain in regular contact on the respective project mailing lists.

This is a central part of our mission: while the scientific computing world may use many tools beyond Python, it is important that we find ways of sharing computational work via open, freely available formats that do not depend on proprietary or patented technologies. This objective will ensure that the IPython Notebook format is up to that task.

[12] http://julialang.org

4.4. **An open collection of executable lecture notes on applied statistics.** We will validate the IPython Notebook's ability to support novel **publication** models for **education**, by developing a collection of *executable lecture notes* that will accompany the *Introduction to Applied Statistics* course at Stanford University[13]. The development of these lecture notes will provide direct feedback on our tools for using the notebook as an academic publishing format and educational platform, as we will be working in direct collaboration with Professor Jonathan Taylor from the Statistics Department at Stanford University. Prof. Taylor has already built a preliminary version of these lecture notes using the IPython Notebook, but due to current limitations in our tools, they are only deployed to the students in the form of static HTML pages and a PDF slide deck. Our collaboration with him will: (a) improve the toolchain in IPython to enable the delivery of executable notebooks that can be used for lecturing and for students as work material; (b) add new material to the lecture notes that illustrates the statistical concepts from the course with recently developed libraries in Python that will complement the existing R-based examples; (c) provide a free set of self-contained executable educational materials on applied statistics; (d) serve as a template for others interested in building similar materials in their own field.

Our inclusion of this objective stems from the fact that education is a critical part of the cycle of research. If the notebook is to gain wide traction as a research tool, it also needs to be useful to teachers developing educational materials and students using those materials as a starting point in their own research. But getting authors to commit to using a new format is a tall order, as people are understandably concerned about spending their limited time writing in an unproven format. This is why this objective is important to the success of our whole vision: by having a battle-tested, real-world example of the use of our tools in an important academic institution, we will be able to show that this is indeed a robust approach to the problem of providing modern lecture materials with a computational component. The fact that Prof. Taylor has already prototyped his lecture notes with IPython tools gives us a proven first cut to base our efforts on. This is consistent with the approach always taken by the IPython team: an iterative cycle where new features are put into the hands of users

[13] Stats 191: http://www.stanford.edu/class/stats191

quickly, and then feedback is gathered to guide future development. In this objective we will turn Prof. Taylor's collection of personal scripts into a robust, production-ready solution that is officially part of IPython itself, documented and tested to ensure reliability.

We have chosen statistics teaching as our driving application because of its strong relationship to computation and its ubiquity across many disciplines: in the age of Big Data, every scientific and industrial field needs statistical analysis. Furthermore, Prof. Taylor is already using the IPython notebook and related tools for his lecture notes, giving us a tested prototype to start from. He will collaborate with team members Poline and Brett, with whom he has interacted since contributing the original codes that led to the creation of the NiPy project.

4.5. **Continued stewardship of the open source IPython project.** In addition to the four specific new development objectives listed above, it is important to consider, as an objective in its own right, the continued effort to lead the IPython project. IPython is a very active project, with constant discussions on the mailing lists and IRC channels, new contributions arriving daily from volunteer developers worldwide and a large user base requiring support. This shows that IPython is a healthy and growing open source project, but it also places strong demands on the leadership team. A significant amount of work (testing, documentation, builds) also goes into releasing the software every 6 months. Furthermore, the core developers travel extensively to promote and teach users about the software at conferences and workshops. For the project to remain healthy, it is important that the core developers allocate time to these pursuits; our project budget reflects these priorities.

To help the IPython development team tackle the ambitious objectives of this grant proposal, we will host week long coding "sprints" every six months, at the beginning of each release cycle. These will bring together the core IPython developers (all personnel in this grant plus other developers from our community) for a week of intensive design discussions and coding. This modality of "coding sprints" (also referred to as "hackathons") has proven to be a remarkably efficient and cost-effective way of providing an intensely focused

effort from the whole team that provides enough energy for tackling complex problems with face-to-face discussion and dedicated implementation time.

Another aspect of the long term growth of the project is increasing its user base. Since its release in the summer of 2011, the Notebook has become extremely popular within the Python scientific computing community. However, outside this immediate community, there is clearly a much larger group of potential users that have not been exposed to it. The features of the Notebook (multiple languages, universal open file format) and the specific objectives of this project are specifically designed to target this larger user group, which, for example might use R or Octave. To reach this group, the project staff will travel to new conferences and workshops to "evangelize" the project. Specifically, we will use P. Ivanov to help in this community building work.

5. Project Deliverables and Assessment

5.1. **Multiuser Notebook Server.** The architecture of the IPython Notebook Server will be restructured to support multiple simultaneous users with authentication and the ability for users to control the sharing and publication of the projects hosted on the server.

5.2. **Interactive widgets.** We will develop a library of interactive HTML/JavaScript User Interface (UI) controls (sliders, check-boxes, etc) in the IPython Notebook bound to Python objects in the IPython Kernel. With this library, output, plots and data will automatically update as users manipulate the UI controls, enabling interactive data exploration, especially for non-technical users. We will provide a base set of widgets for common tasks and will begin to investigate more sophisticated widgets such as a spreadsheet-style control for working with tabular data.

5.3. **File format specification and conversion tools.** We will provide a fully documented specification of the IPython Notebook file format suitable for third-parties to implement compatible tools. This specification will support Notebooks written in other programming languages beyond Python. We will work with the Julia development team and establish

collaborations with R users, thanks to our existing contacts in the Statistics departments at Stanford and UC Berkeley.

We will build into the IPython codebase tools that will enable: (a) seamless conversion to other important formats: LaTeX, PDF, Markdown, reStructuredText, HTML; (b) an integrated slide show mode that turns a notebook into a presentation for dissemination and educational purposes; (c) the transformation of Notebooks based on metadata attached to parts of a notebook.

5.4. **Domain-specific use case: executable lecture notes to accompany a course in Applied Statistics.** We will develop and publish a collection of open source notebooks that will accompany Stanford University's *Introduction to Applied Statistics* course (Stats 191). These notebooks will be available in native IPython format as well as a website and in electronic book format for reading. We will also provide the skeleton of these notes as a template: this will serve as a starting point for educators to produce materials with similar features in other disciplines. These materials will be released according to the generous licensing terms of the Reproducible Research Standard [14] (i.e. CC-BY license for text, BSD license for code and public domain terms for data).

5.5. **Project stewardship.** The concrete deliverable for this objective will be the maintenance of a regular release schedule for IPython, with each new version being made available to the public roughly at 6 month intervals. We will also hold two annual development sprints at UC Berkeley, as well as presenting the project outcomes at the annual Scientific Computing in Python conferences, PyCon, Strata and other relevant conferences, such as the annual SuperComputing events.

5.6. **Success metrics and assessment.** While forecasting adoption of new tools is difficult, we consider the following as reasonable indicators of the impact of our proposed work:

- Deployment of 5 instances of the multiuser notebook server in research groups, classroom or company settings (by groups not affiliated to us).
- Development of 5 websites or projects that integrate the interactive widgets capability for educational or data exploration purposes.

- Adoption of the IPython Notebook as the teaching tool for 3 university courses.
- Release online of 5 sets of learning materials or tutorials by independent parties.
- Publication of 2 scientific articles by authors beyond our team, using IPython Notebooks to provide reproducible results.

We will conduct periodic user surveys online to assess progress on these metrics. We note that by virtue of being free software users can download IPython and set it up without asking us for permission or verification of any kind. Therefore, any metrics we provide are always an undercount.

6. Budget Justification

We request budget to fund a team with a long track record of success in leading the IPython project. The majority of the budget goes to support these established members, enabling them to devote their full attention the complex problems outlined earlier. We have a provision for one yet to be named postdoctoral researcher, allowing for new blood to enter the project, but everyone else listed in our budget is a known contributor. Our travel budget will enable us to participate not only in scientific Python conferences, but to engage scientists at discipline-specific events to reach a larger audience. Our workshop budget finances four week-long, intensely focused developer meetings (often referred to as "coding sprints") that are central to our success: from past experience, we know this kind of meeting is an extremely effective way to make significant inroads on difficult problems by "locking up" the whole team without distractions.

References

1. M. Brett, J.-L. Anton, R. Valabregue, and J.-B. Poline, *Region of interest analysis using an SPM toolbox*, Proc. 8th HBM (Sendai, Japan), June 2002.
2. Frederic Brochu, Ulrik Egede, J. Elmsheuser, K. Harrison, R. W. L. Jones, H. C. Lee, Dietrich Liko, A. Maier, Jakub T. Moscicki, A. Muraru, Glen N. Patrick, Katarina Pajchel, W. Reece, B. H. Samset, M. W. Slater, A. Soroko, C. L. Tan, and Daniel C. Vanderster, *Ganga: a tool for computational-task management and easy access to grid resources*, CoRR **abs/0902.2685** (2009).
3. J. Couzin-Frankel, *Cancer research. As questions grow, Duke halts trials, launches investigation.*, Science **329** (2010), no. 5992, 614–5 (eng).
4. O. Heffernan, *'Climategate' scientist speaks out.*, Nature **463** (2010), no. 7283, 860 (eng).
5. Tony Hey, Stewart Tansley, and Kristin Tolle (eds.), *The fourth paradigm: Data-intensive scientific discovery*, Microsoft Research, Redmond, Washington, 2009.
6. J. D. Hunter, *Matplotlib: A 2d graphics environment*, Computing In Science & Engineering **9** (2007), no. 3, 90–95.
7. E. Jones, T. Oliphant, P. Peterson, et al., *SciPy: open source scientific tools for Python*, 2001–, URL: http://www.scipy.org.
8. K. J. Millman and M. Brett, *Analysis of Functional Magnetic Resonance Imaging in Python*, Comput. Sci. Eng. **9** (2007), no. 3, 52–55, URL: http://neuroimaging.scipy.org.
9. F. Pérez and B. E. Granger, *IPython: a System for Interactive Scientific Computing*, Computing in Science & Engineering **9** (2007), no. 3, 21–29, URL: http://ipython.org.
10. F. Pérez, B. E. Granger, and J. D. Hunter, *Python: an ecosystem for scientific computing*, Computing in Science & Engineering **13** (2011), no. 2, 13–21.
11. B. Ragan-Kelley, W. A. Walters, D. McDonald, J. Riley, B. E. Granger, A. Gonzalez, R. Knight, F. Pérez, and J. G. Caporaso, *Collaborative cloud-enabled tools allow rapid, reproducible biological insights*, ISME Journal (2012), doi:10.1038/ismej.2012.123, URL: http://qiime.org/home_static/nih-cloud-apr2012.

12. Science Software Branch at the Space Telescope Science Institute, *Space Telescope Science Institute stsci_python*, URL: http://www.stsci.edu/institute/software_hardware/pyraf/stsci_python.
13. W. A. Stein et al., *Sage Mathematics Software*, The Sage Development Team, 2011, URL: http://www.sagemath.org.
14. V. Stodden, *Enabling Reproducible Research: Open Licensing For Scientific Innovation*, International Journal of Communications Law and Policy **13** (2009).
15. J. Taylor, K. Worsley, M. Brett, Y. Cointepas, J. D. Hunter, K. J. Millman, J-B. Poline, and F. Pérez, *BrainPy: an open source environment for the analysis and visualization of human brain data*, Human Brain Mapping, 2005.