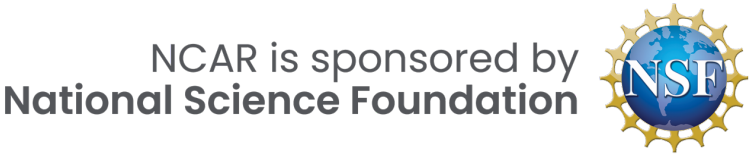
NCL and the Pivot to Python

Discussion and Roadmap

January 2019

Mary Haley
Rick Brownrigg
Kevin Hallock
Bill Ladwig
John Clyne (supervisor)

Visualization and Analysis Systems Technologies
National Center for Atmospheric Research



Any opinions, findings and conclusions or recommendations expressed in this material do not necessarily reflect the views of the National Science Foundation.

Table of Contents

Executive Summary

The NCAR Command Language (NCL) is a scripting language providing specialized visualization and analysis capabilities tailored for climate and weather data. PyNIO, WRF-Python, and PyNGL are Python modules built on a subset of NCL's capabilities.

Based in part on report recommendations from an NSF Site Visit Team (SVT) in May 2016 and a CISL Advisory Panel (CISLAP) Meeting in July 2017, NCAR has been evaluating strategies for future development and support of NCL in the context of widespread availability of popular open-source data languages with much larger user communities, such as Python and R. These ecosystems duplicate much of NCL's functionality while adding critical features that NCL doesn't have. Both reports stressed the importance of NCL's unique functionality in the geoscientific community. However, given decreasing budgets and staff reductions, they strongly recommended adopting a more open development software model to enable the geosciences community to play a more active role in its future development and support.

NCAR is committed to supporting visualization and analysis capabilities for atmospheric, oceanic, and earth science research. In order to get feedback from the geosciences community on their use of NCL, Python, and other open source tools, as well as their thoughts on open development, NCAR conducted an NCL survey in May 2018 and hosted an NCL Advisory Panel (NCLAP) meeting in August 2018.

Findings from the SVT, CISLAP, and NCLAP reports and the NCL survey—combined with the changing software landscapes and the enormous functionality that Python brings to the earth sciences—have led NCAR to arrive at these major decisions, effective immediately:

- Python will be adopted as the scripting language platform of choice for future visualization and analysis development.
- Unique and critical NCL functions will be ported to Python.
- NCL and PyNIO will be placed into maintenance mode.
- Development will continue on WRF-Python and PyNGL for the foreseeable future.
- All software, including NCL and PyNIO, will be moved to a more open development software platform to allow for continued community development.

This report provides an in-depth technical justification for these decisions and what steps will be taken to help transition NCL users to Python. Supporting materials included with this report are the "NCL, Python, and Related Software User Survey and Report" and the "NCL Advisory Panel Meeting and Report".

A two-year roadmap is included near the end of this document, which addresses the top immediate priorities that came out of the supporting materials:

- Wrapping NCL's computational routines into Python
- Training and support for transitioning NCL users to Python
- Moving the tools to a more open development model+

Introduction

A team of four software engineers in the Visualization and Analysis Systems Technologies (VAST) section at NCAR develops and supports four software packages for the analysis and visualization of climate and weather data: NCL, PyNIO, PyNGL, and WRF-Python. NCL is the flagship product supported by this team. The three Python packages do not cover the full range of NCL capabilities.

The NCL developers are at major crossroads as we evaluate the viability of developing and supporting NCL in the face of staffing and budget cuts, and the continued explosive growth and use of Python in the scientific community. NCL is supported on NSF core funds and it has become increasingly more difficult to justify funding development of NCL when Python and its wide collection of open source scientific modules (herein referred to as "Python") duplicates a lot of NCL's functionality and adds critical features that NCL doesn't have.

The budget challenges combined with the enormous functionality that Python brings to the earth sciences has led to the decision that development on NCL as a language needs to cease, and that we pivot to Python as the core language for further development of data analysis and visualization software. In tandem with this decision is the requirement that the tools be moved immediately to a more open development model, better enabling the scientific community to play a more active and relevant role in their development and future.

In order to better mitigate a transition to Python and get feedback on open development from the general NCL community and a select group of power users, we conducted an extensive survey in May-June 2018 and hosted a meeting with an NCL Advisory Panel (NCLAP) in August 2018. Based on feedback from the survey and the written report from the NCLAP, pivoting to Python is a sound strategic decision that should move forward immediately, but with careful planning, support, and training to reduce anxiety and ease the uncertainty that such a change will bring to the NCL user base. Moving to a more open development model was viewed more positively, as it is somewhat of a natural progression for software that is already open source and has welcomed contributions from the outside community for many years. It does come with the valid concern about how to maintain stability of the software and keep it connected with NCAR's mission to enable Earth system science.

The success of this transition requires that we need to scale back support of NCL and make the Python tools more of a priority in future development efforts.

This report discusses in more detail the case for pivoting NCL users to Python and provides a roadmap for this transition and for moving to a more open development model. A recent addition to the roadmap includes adopting the new requirements mandated by NSF for software maintenance, licensing, and branding.

Brief Development History of NCL and Python Tools

Since its initial release in 1995, NCL has been developed and supported in close collaboration with scientific groups at NCAR and UCAR, resulting in a robust "all-in-one" software package tailored for the analysis and visualization of climate and weather data. NCL is particularly known for its support of various complex data formats (NetCDF, GRIB, HDF, HDF-EOS, shapefile), its hundreds of specialized computational routines, and its highly-customizable and publication-quality graphics. Unique to NCL—in comparison with other similar scripting languages like Python and IDL—is the dedicated level of support in the Earth Sciences provided by the developers via email, online documentation and examples, manuals, and training. The user community has also contributed significantly in all areas of NCL support and development.

This "all-in-one" solution combined with the extensive user support has resulted in thousands of users world-wide adopting NCL as their main post-processing tool of choice. Metrics on the usage of NCL are based on the number of downloads of the software, the level of activity on the NCL email list, and website statistics. See Appendix A for these metrics.

In the mid 2000s, seeing the early rise and potential of Python as a robust open source scientific programming language, we developed and released PyNIO and PyNGL, which wrapped the file I/O and graphics capabilities of NCL into Python modules. The purpose of these modules was two-fold: to expose the components of NCL to a more mainstream programming language and to provide unique capabilities not currently available in the Python community. In 2017, the team developed and released WRF-Python, a Python module built on top of NCL's WRF diagnostics library.

As the functionality and usage of NCL and its Python tools has continued to grow, the staff developing and supporting this software has been reduced over the years, resulting in currently four core software developers supporting four packages. Supporting software for two programming languages has been a major challenge and resulted in fewer new features being added overall. The size and scope of NCL alone requires a team of multiple developers to keep it going, let alone additionally supporting three Python modules.

In the face of these challenges, both the SVT and the CISLAP have recommended that we move our software to a more open development platform. This will hopefully facilitate more community contributions. This and related quality assurance are discussed in the "Open Development" section below.

Why Python?

Python has gained widespread acceptance by universities and research organizations around the world and is being adopted as the programming language of choice for scientific computing. This is evidenced by several factors: 1) the availability of quality scientific Python modules via the SciPy ecosystem, 2) the continued and growing popularity of the annual SciPy conference, now in its 17th year, 3) the availability of books on Python for scientists, and 4) the increasing number of scientific graduate students who are learning Python in college as an open source

alternative to other non-free software like IDL and MATLAB. In September 2018—for the first time in history—Python entered the TIOBE index top 3 (www.tiobe.com), a measure of popularity of programming languages based on search engine results.

Python has picked up rapid steam in the geoscientific community as well. For the last eight years the American Meteorological Society Annual Meeting has hosted a popular and well-attended symposium on the "Advances in Modeling and Analysis Using Python". NCAR is a major partner in the Pangeo (pangeo.io) community, an NSF EarthCube funded effort that provides an "open source scientific Python ecosystem for ocean / atmosphere / land / climate science" and is focused on providing tools and support for handling petabyte-scale datasets on HPC and cloud platforms. There are hundreds of scientific Python modules that provide domain-specific functionality for reading/writing data, computational analyses, and visualization. The benefit of these individual packages is that they are usually specialized for a specific domain or class of problems, thus filling a critical need that a more general-purpose language cannot.

The Python language itself provides rich language features that NCL does not have, including optional arguments, a robust interactive interface, generators, exception handling, and built-in debugging and testing. The Python community has a rapidly growing base of scientific software developers that are able to address the growing needs of the geoscientific community much faster than we can in the areas of scalability, interfaces to other languages like R for statistical calculations, and support for a wider range of complex data formats. By replacing the NCL language with the Python language, the NCL user base will instantly gain access to these features, and we will be able to benefit from the already vibrant and active open development Python community. Python itself has been open developed since October 2000.

Last but not least, it is becoming harder to hire developers who want to work on a programming language with a narrow focus, versus a highly visible and mainstream language like Python.

Even with the clear benefits of Python, we are well aware that asking the NCL user community to transition to Python will likely raise many questions and concerns, for example:

- How to convert NCL scripts to Python scripts?
- Where to get help for Python-related questions?
- Will Python support GRIB as robustly as NCL?
- Can Python produce the same quality graphics as NCL?
- Can NCL users who don't want to switch still use NCL for the foreseeable future?

We address these concerns in this report and the included roadmap.

NCL, Python, and Related Software User Survey

In spring 2018 we conducted an extensive "NCL, Python, and Related Software User Survey" (herein referred to as the "NCL survey"). The survey was targeted mainly at NCL users and its purpose was to determine:

- How many NCL users are actively using Python or considering it
- What NCL functionality is critical to its users
- What kind of Python support and training is needed
- How do users feel about an open development model for software

The NCL survey resulted in 699 responses which have been analyzed in a separate detailed report. We used the survey feedback to refresh and prioritize items for the roadmap, which is included in this report.

NCL Advisory Panel

An NCL Advisory Panel with twelve members was formed in the early spring of 2018 comprising of climate and weather researchers, faculty members, and software engineers from NOAA, University of Nebraska-Lincoln, University of Maine, Cargill, ClimaCell, DKRZ / Hamburg, and three scientific labs at NCAR. These members were all either current NCL users or had used NCL in the past. Most had varying degrees of familiarity with Python.

The purpose of the panel was to:

- Provide guidance on the future of NCL in the scientific Python ecosystem
- Provide ideas on how we can actively engage the scientific community in open development efforts
- Begin a more formal governance of NCL and its related Python tools

NCAR hosted a 1.5-day meeting in Boulder with the panelists on August 2-3, 2018. We presented materials on the current state and challenges of NCL development, an overview of Python and the NCL-based Python tools, a plan for moving to a more open development model, an introduction to Pangeo, a discussion of the NCL survey results, and a first draft of a roadmap for the next three years.

The panel gave an informal verbal report out on the second day of the meeting and wrote a formal report a few weeks later. Information from this report is referenced heavily through the rest of this document and was used to fine-tune the roadmap, especially in the "pivot to Python" transition. The panel has stated a desire for a second meeting to be held sometime in 2019.

The report contained six findings on the following topics, listed in order of priority. The findings under each topic are summarized followed by a response from the NCL developers. In some cases, direct statements from the report were included and enclosed in quotes; and in other cases, the statements are slightly reworded for clarity and brevity.

1. Continued support for the NCL Core Language

The panel strongly recommended "that CISL/NCAR continue to provide support for the core NCL language, including outreach support for the NCL user community." There were four points in this finding:

1.1 "Maintain the NCL core language in its current state indefinitely, without necessarily adding new functionality."

1.2 "Maintain access to the existing NCL function library indefinitely."

1.3 "Continue to support and develop specific libraries that are critical to a range of community of users and unique to NCL."

1.4 "Continue to support the current state of online NCL documentation and support the NCL user community through continuation of ncl-talk."

Based on discussions with the August panel meeting, points 1.1 and 1.2 stem from their concern that there are some NCL users who will be unwilling or unable to transition to Python and hence NCL should "live" as long as possible. The definition of "core language" is understood to mean language features like do loops, if statements, the NCL NetCDF data model, etc. The "indefinite maintenance" means that users should be able to compile and run NCL indefinitely and the current computational and graphical functions should continue to exist in their current state so that NCL scripts that depend on them can retain backwards compatibility.

We believe we can meet both of these points. First, efforts by Stanislaw Jaroszynski (NCAR) and Edward Hartnett, (GSD, edward.hartnett@noaa.gov) are underway to replace the custom NCL build system with CMake and GNU Autotools, which will standardize the build system. This will make it significantly easier to compile NCL, a long-standing user complaint. Second, the team has no plans to change the behavior of existing routines. As functionality is ported to Python, some routines may be retooled for the new library, but backwards compatibility in NCL will continue to be maintained to the extent possible.

Point 1.3 refers to Python computational libraries like WRF-Python and new specialized computational routines that get developed. Libraries and functions that are identified as unique to NCL are likely to be ported to Python and hence will be supported and developed in the future. Efforts are already underway to refactor NCL's computational library into an "agnostic" library (herein referred to as "NCOMP"), which will allow interfaces to be developed for other languages in addition to Python.

Regarding point 1.4, the online NCL documentation (www.ncl.ucar.edu) is quite massive and some of it is out-of-date, but we have no immediate plans to address this. The NCL website is one of the top visited sites at NCAR, so it's important to maintain its content and structure for the near-term. It should be noted, however, with the new charge coming from NSF for open source software to be more consistent in its branding, that the NCL website may need to be

restructured at some point to meet NCAR/NSF web page templating requirements. Some of its content may get reorganized and potentially even dropped, but only with the careful consideration of whether it is still useful to the NCL user community. We will solicit community input to help drive these decisions.

The extensive support provided by the developers on ncl-talk will have to be scaled back in order to put more emphasis on Python tools support. We will try to answer questions where possible, but will be looking more to the NCL user community to contribute in this effort. We do plan to start a Python email support list (see the "NCL and Python Tools Roadmap" later in this document).

2. Training and support for the "Pivot to Python"

The panelists noted that the consensus from the NCL survey is that "NCL user support is unparalleled in the software open development world and, coupled with the heavy focus on meteorological, climatological, and model- centric algorithm and examples, provides an unparalleled resource for scientific research." The panel strongly recommended "developing a training and support system for guiding NCL users through the pivot to Python transition, with an eye toward maintaining the current user base while expanding it to include the Python user community." They went on to suggest working towards "an expanded user base that will eventually take some of the weight from the NCL developers by contributing to current and future Python libraries through the open development model." This finding included five points:

2.1 Provide the same high level of user support in the Python ecosystem that currently exists for NCL.

2.2 Ease community anxiety over a perceived "NCL code freeze" by "emphasizing that the pivot to Python indicates that NCL's capabilities continue to evolve and move forward, even as resources move away from expanding the core NCL language."

2.3 Augment support (website and ncl-talk) and training by adding information about the Python software to the NCL website and offering training on NCL-to-Python libraries, possibly as stand-alone workshops or added to existing ones.

2.4 Hold informational sessions about the benefits of a pivot to Python and advertise them on ncl-talk and the website.

2.5 Develop side-by-side NCL-Python scripts on the NCL website.

These five points, coupled with results from the NCL survey, indicate that user support is something that must be carried forward as the team pivots to Python. The points are addressed below in the order that we can likely implement them.

Point 2.2 is the one of the first that we plan to address. An open letter to NCL users will be drafted regarding the decision to pivot from NCL to Python. This report can provide some of the content for such a letter. During the NCLAP meeting, the panelists and the NCL developers

agreed that this letter needs to be made highly visible, by announcing it on ncl-talk and featuring it prominently on the NCL website home page. The panelists said they would help review the letter, which will be accompanied by an updated roadmap. These documents will be made available to users for a chance to comment, likely on GitHub and the NCL website.

Point 2.5 is also something that will be started right away. There are hundreds of NCL "application" examples (www.ncl.ucar.edu/Applications) which will be moved from a private git repository into a public GitHub repository. This decision comes via a suggestion in the 2017 CISLAP report to review the open development model followed by the MOM6 ocean model community. The MOM6 model examples exist on their own repository (https://github.com/NOAA-GFDL/MOM6-examples), which we believe may be a good development model given the size and number of the NCL examples. Moving the examples to GitHub provides a ripe opportunity to encourage the NCL user community to contribute in converting NCL scripts to Python or create new scripts. Karin Meier-Fleischer (DKRZ), the main author of the NCL User Guide and a member of the NCLAP, has offered to help with this effort. She has already written a first draft of a "NCL transition to Python" document, containing dozens of NCL-to-Python example scripts.

Point 2.4 was discussed during the meeting, but it has not yet been decided what an "informational session" will entail. One possible idea was to host a webinar where the transition to Python is discussed (along with its benefits), while giving users a chance to voice their concerns and ask questions. This point will be augmented by point 2.2, where users will have a chance to comment on the open letter and the roadmap.

Point 2.3 has two action items that are somewhat different in scope. The first item is for adding documentation to the NCL website on the Python libraries being developed. This item needs to be carefully addressed, as pivoting the NCL software to Python presents an opportunity to adopt more modern documentation generator tools like Sphinx and to host documentation under something like Read the Docs. The second item addresses training on the new software. This is something that is not likely to happen for the first year of the transition, as we need time to create the new Python modules and become comfortable enough with Python to start training on them. We hope that by converting NCL examples to Python (point 2.4) right away and offering support in the form of something like an email list (point 2.1), that the training piece can come later. In early September 2018, Damien Irving released the first ever Data Carpentry lesson for atmosphere and ocean scientists and is looking for instructors, students, and general comments (https://datacarpentry.org/blog/2018/09/atmos-ocean-launch). We have reached out to Dr. Irving and a discussion of a possible collaboration is in motion.

Point 2.1 is addressed somewhat by the other four points. Something that needs to be considered is whether a new Python-based email list like ncl-talk needs to be started or whether the current "pyaos" email list (hosted by Johnny Lin, the Director of Undergraduate Computing Education at the University of Washington Bothell) is an appropriate forum for user support. The Pangeo community might also be a good source to tap into. There is a pyngl-talk email list, but it is not very active and should probably be retired in favor of these other options.

3. NCL graphics

The panel pointed to "the unique role that NCL plays for data analysis and scientific visualization, particularly for atmospheric and related sciences" and offered two points for this topic:

3.1 Continue "development and support of NCL graphics within the PyNGL library and a mirroring of these changes in the NCL language as appropriate".

3.2 There was an observation that the NCL survey results indicated a high-level of importance of NCL graphics for general use and journal publications, and that several panelists felt that matplotlib—the de facto 2D graphics package for Python—does not quite match the quality of NCL graphics.

In the early discussions around pivoting to Python, we had initially considered dropping support for PyNGL given the wide landscape of 2D plotting packages available in Python (many of which are built on top of matplotlib and cartopy, like Seaborn, MetPy, and IRIS). However, based on the NCL survey results and the comments from the panelists, it has been decided to continue supporting PyNGL for now. As such, a new version of PyNGL with support for Python 3 was released in mid-August.

While PyNGL will continue to be developed and supported in the foreseeable future, we believe that a thorough review of PyNGL and the other Python plotting packages is warranted, as continuing the development of PyNGL may likely be seen by NSF and the Python community as a duplication of effort. There are certain features and capabilities of NCL/PyNGL graphics that are highly unique, and we think this could be a show-stopper for NCL users who try to produce equivalent visualizations in Python. Developers in the Pangeo community have offered their Python expertise in helping us evaluate the visualization capabilities. This review will likely occur at the same time the NCL examples are being converted to Python (see discussion under finding 1).

4. NCL computational functions

Based on the fact that NCL "was designed for scientific analysis and data visualization specific to the predominant sciences found at NCAR and the surrounding labs, e.g., Atmospheric Science, Climatology, Meteorology, and Oceanography", the panel strongly recommended "continued development and support for the NCL computational functions unique to this community." There were three points under this topic:

4.1 Add performance enhancements to the existing functions and port them to Python.

4.2 Leverage the Pangeo community to identify computational functions that are not already available in existing Python libraries and integrate the library into Pangeo's 'Python Data Stack'.

4.3 Generate a list of critical computational routines to start with, based on results of the "Porting NCL Computational Routines to Python" section of the NCL user survey.

The porting of NCL's computational routines to a Python library is one of our highest priorities. Work has already started to refactor the NCL computational libraries into their own "NCOMP" package on GitHub, in order to separate the code from NCL and make it more amenable to being open developed. Thin wrappers are being developed that will enable these routines to be called from multiple languages like R, Python, and Julia. Finally, a Python library is being developed (herein referred to as "PyNComp") on top of these thin wrappers that have the option to use xarray as the common data model, enabling this code to work within the Pangeo software stack. The xarray module (xarray.pydata.org) provides a data model similar to NCL's data model, which combines data and metadata in one object and allows for computations that support missing values on multi-dimensional arrays, a feature that the core Python language does not have. See the "NCL and Python Tools Dependency Diagrams" section which includes diagrams for the current architecture of the NCL and Python tools and for the planned refactoring effort.

Determining which of the hundreds of NCL's analysis routines to focus on first is a bit of a challenge. As point 4.3 states, the NCL survey results provided some useful guidance on this. In order to tighten the development focus even further and get familiar with real-world issues in porting NCL code to Python, we are participating in the "CVDP Pilot Project" – an effort to convert a subset of the NCL-based Climate Variability Diagnostics Package to Python. The CVDP is an analysis tool developed by Adam Phillips (NCAR/CGD), and is used by NCAR's Climate Earth System Model (CESM) to generate a number of climate metrics that are used to evaluate a simulation. The CVDP provides a nice snapshot representation of popular NCL analyses routines that are specific to climate research. Members of the core Pangeo community have also offered their expertise with this project in helping select routines to port and answering questions about xarray.

5. Open development

As mentioned earlier in this report, moving NCL and its Python tools to a more open development model are direct recommendations from the SVT and the CISLAP. The NCLAP was explicitly asked to address this topic, on which they came back with three points and several subpoints.

5.1 They "strongly support the movement toward an open development model that can augment continued support for the NCL core language through user contributions" which includes refactoring the code and formal governance in the form of a contributor's guide. They also recommended that the "repository gatekeepers include NCL developers in order to maintain the stability of the NCL code and keep it connected with UCAR/NCAR, which has a fundamental mission to enable community science".

5.2 They "support the recommendation to move the NCL open development platform to an issue tracking repository such as GitHub rather than the external/JIRA system that is being used currently."

5.3 "The Pangeo community should be leveraged to identify computational functions unique to NCL". In addition, the Pangeo community can serve as a model for "formulating a formal governance framework", "the process of vetting and integrating code within an open development framework", and "a list of libraries recommended by the community."

To address point 5.1, we are refactoring components of NCL into separate packages, turning them into individual software packages that can be maintained and used independently. This is in addition to the NCOMP project mentioned earlier. The refactoring effort should encourage more open development, as it will be easier to extend than a single monolithic package. Writing contributors' guides where appropriate will parallel the refactoring effort. Finally, we do plan to continue as repository gatekeepers, as we will be the main developers and maintainers of these packages for the unforeseeable future.

Point 5.2 is something that we are currently reviewing, as there are hundreds of tickets under JIRA. It is not yet clear if and how they should best be transcribed to GitHub. In the interest of open development, however, the team would like to move fully to GitHub for issue tracking and enhancement reporting.

For point 5.2, we have established a connection with the Pangeo community through the CVDP Pilot Project and a Pangeo workshop hosted in August 2018. As mentioned earlier, the Pangeo developers have expressed a specific interest in helping us with the pivot to Python, and they certainly will be used as a resource in developing a sound governance model. As part of the refactoring effort, we plan to develop a PEP8 style guide and to integrate automated testing. Any code contributed by the community would also be required to follow these guidelines.

6. File input / output support

In the NCLAP meeting, we discussed how NCL and PyNIO are duplicating file I/O capabilities that are already available in Python via modules like netcdf4-python, cfgrib, and PyTables, to name a few, and suggested that it might be time to retire PyNIO in favor of using these specialized packages. The NCLAP came back with three points on this topic:

6.1 They encouraged "continued PyNIO support for specific NCL capabilities that are not duplicated or well-supported by other Python packages. **In particular, the panel encourages the continued support of uncommon formats such as GRIB and HDF-EOS."** (Highlight theirs.)

6.2 The panel recognized that "existing I/O capability and development outside of NCL does not need to be duplicated by NCL" and that "active PyNIO development may best be accomplished as part of an open development community".

6.3 NCL file I/O functions that are already available in Python libraries should be leveraged where possible when converting the online NCL examples scripts to Python.
[This point was clarified with the NCLAP chair and co-chair and reworded as such here.]

Even though this particular finding is listed last, it is one that is a major issue for the developers. NCL and PyNIO are unique in their robust support of NetCDF, GRIB, HDF and HDF-EOS files. They provide a singular function to read all of these formats, placing them in a common data model that looks similar to the NetCDF data model.

Unfortunately, it has become increasingly unrealistic to support this array of complicated data formats in a single software package, especially as new features are added that requires dedicated expertise to understand and implement. This—coupled with the fact that the lead software developer for NCL and PyNIO's file I/O interface retired in late 2017 with no budget for a replacement—has led us to recommend to the panel that PyNIO be retired in favor of xarray and other specialized Python packages. We believe these data formats are best supported by the software engineers who develop them or whom are resident experts. Some examples include Unidata's netcdf4-python module, ECMWF's cfgrib for GRIB, NumFOCUS's PyTables for HDF5 and Raytheon's PyHDF module for HDF-EOS. Some of these packages tie nicely into the Pangeo Data Stack due to their support for xarray, which provides a similar common data model as NCL but with plug-in support for big data.

It should be noted that xarray supports GRIB via PyNIO as a backend, but its developers may replace PyNIO with cfgrib once a stable version has been released. Given the level of complexity and the lack of a strong standard in supporting GRIB, the NCLAP and the developers are cautious in assuming that xarray and the combination of other Python file I/O modules can provide a full replacement of NCL file I/O capability. This could be a serious show-stopper for getting NCL users to convert to Python, and is why this text was highlighted by the NCLAP in point 6.1. As part of the pivot to Python, we consider it a high priority to review how well the various Python packages can handle the same data formats as NCL, and may be able to provide some expertise in helping developers of the other Python file I/O modules fine-tune their readers.

NCL and Python Tools Dependency Diagrams

The two diagrams on the following page illustrate the current state of the NCL and Python software and the state after the refactoring effort. The top diagram shows the current three Python software packages and how they are dependent on the file I/O, computational, and graphical components of NCL. The bottom diagram shows the refactoring of the computational ("NCOMP") and graphical ("NVIZ") components of NCL into their own software packages, and the addition of the new "PyNComp" Python package that is being developed.

The WRF-Python / NCL link is shown as a two-way dependency in the top diagram because the WRF-Python computational routines were originally copied out of NCL, heavily modified for WRF-Python, and then incorporated back into NCL. These routines still exist in both packages, but as part of the refactoring effort they will be moved to NCOMP.

Splitting out NCOMP and NVIZ into their own software packages removes the Python packages' dependency on the full NCL package. It also makes it easier for users to extend NCOMP and / NVIZ, if desired, for other languages.

There are currently no plans to refactor the NCL file I/O library into a separate package, as it is tied directly with the NCL interpreter's data model. This decision may change after the evaluation of PyNIO (see the "File input / output support discussion" under the "NCL Advisory Panel" section).

Diagram of current suite of NCL and Python software

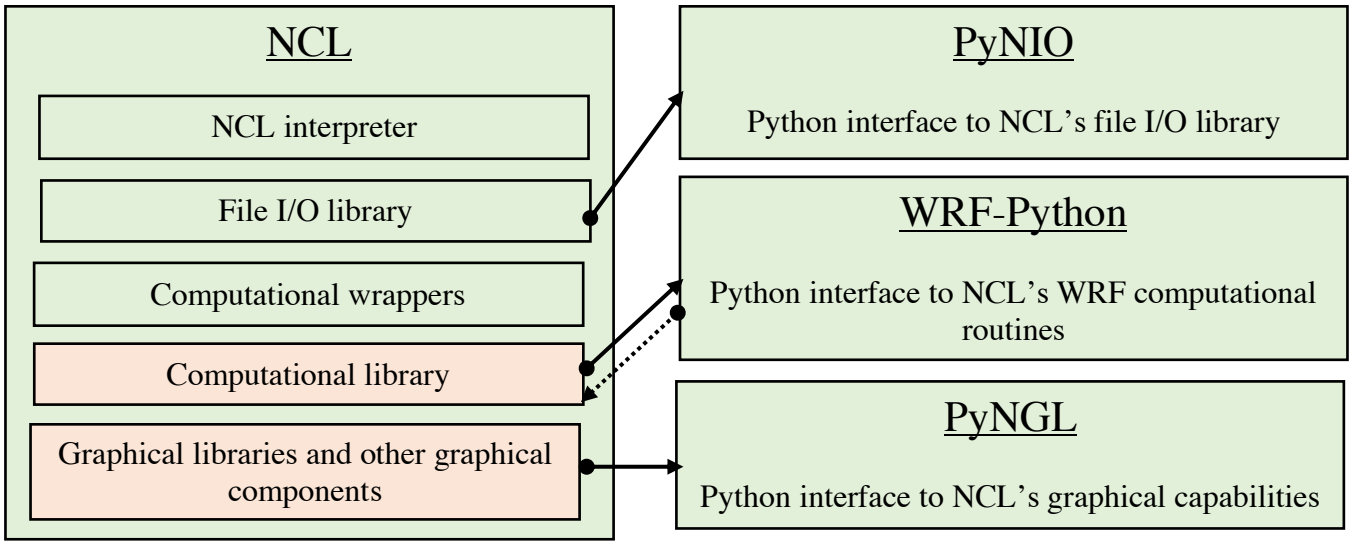
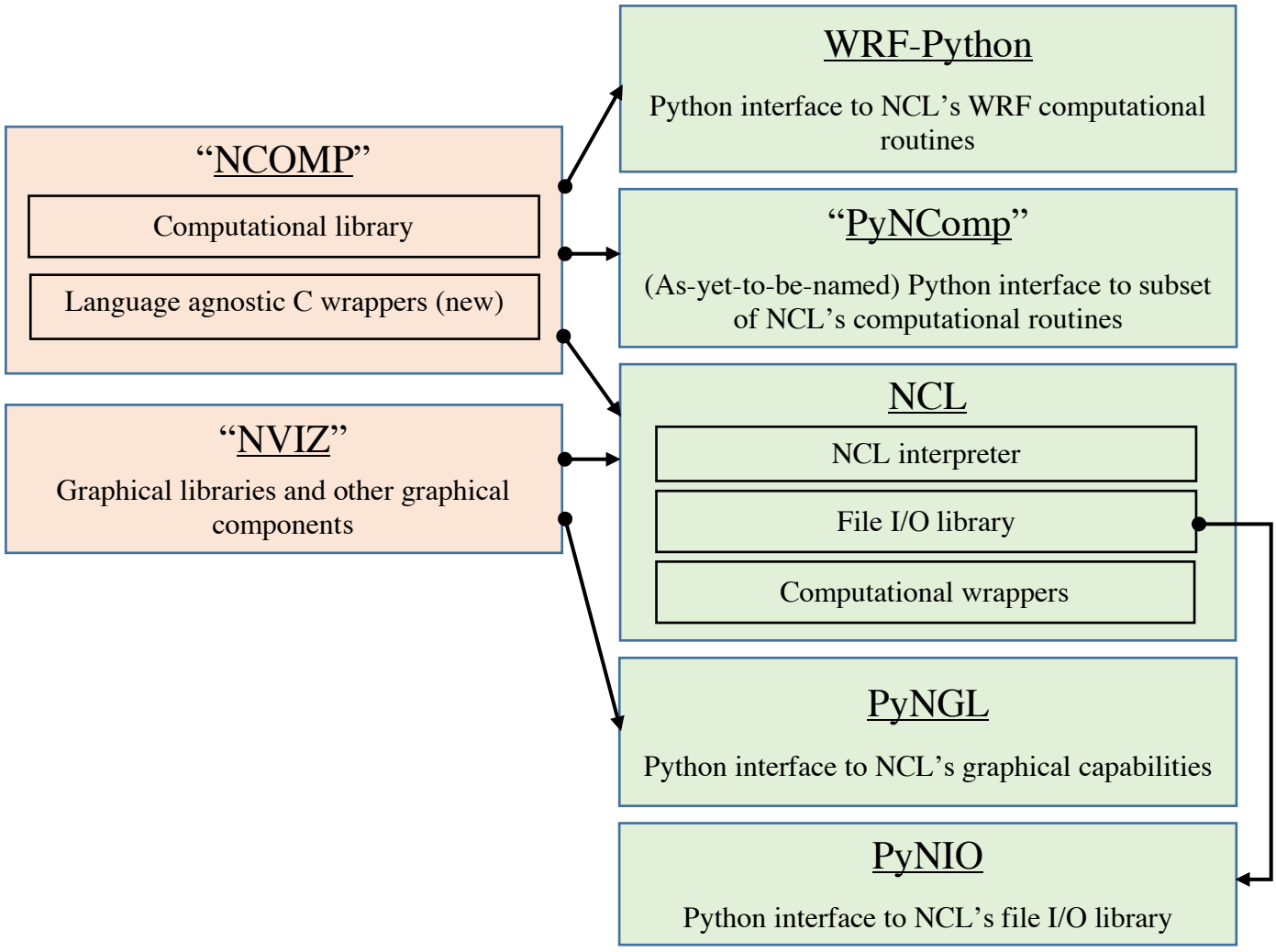


Diagram of refactored NCL and Python software



NCL and Python Tools 2018-2020 Roadmap

The NCL and Python tools roadmap has been heavily revised from an earlier version that was presented to the NCLAP meeting in August 2018 and had a more NCL-centric focus, to a newer version that has a more Python-centric focus. It's broken into two sections: a roadmap for the various software packages illustrated in the previous "refactored software" diagram, and a roadmap for the pivot to Python and training.

Input for this roadmap came from multiple sources: the NCL and Python tools developers, CISL upper management, discussions with NCLAP members, the SVT and CISLAP reports, the results from the NCL survey, the NCLAP written report, and the relatively new NSF branding requirements.

Roadmap Legend	
CM	Completed Milestone
SM	Scheduled Milestone
IP	In Progress
PW	Planned Work

NCL and Python Tools Software Release Roadmap									
NCL	**Jul-Sep 2018**	**Oct-Dec 2018**	**Jan-Mar 2019**	**Apr-Jun 2019**	**Jul-Sep 2019**	**Oct-Dec 2019**	**Jan-Mar 2020**	**Apr-Jun 2020**	**Jul-Sep 2020**
Version 6.5.0 • Subprocess module • Several new computational routines • Graphical enhancements	CM								
Version 6.6.0 * • New WRF-NCL functions • WRF-NCL performance enhancements • WRF-NCL bug fixes • Contributor's Guide • Apache 2.0 License	IP IP IP IP	IP IP IP IP	SM IP IP IP IP IP						
* This will be the last feature release of NCL. After this point it will go into maintenance mode and software development will be focused on the Python tools, "NCOMP", and open development efforts.									
"NCOMP"	**Jul-Sep 2018**	**Oct-Dec 2018**	**Jan-Mar 2019**	**Apr-Jun 2019**	**Jul-Sep 2019**	**Oct-Dec 2019**	**Jan-Mar 2020**	**Apr-Jun 2020**	**Jul-Sep 2020**
Version 1.0.0 • NCL computational library factored into its own package		IP	SM IP						

• Add language agnostic C wrappers to subset of computational routines • Contributor's Guide		IP PW	IP						
Version 1.1.0 • Add more language agnostic C wrappers • Performance enhancements (OpenMP, OpenACC)			PW PW	PW PW	SM PW PW				
"NVIZ"	**Jul-Sep 2018**	**Oct-Dec 2018**	**Jan-Mar 2019**	**Apr-Jun 2019**	**Jul-Sep 2019**	**Oct-Dec 2019**	**Jan-Mar 2020**	**Apr-Jun 2020**	**Jul-Sep 2020**
Version 1.0.0 • NCL graphical library and components factored into their own package • Contributor's Guide		IP	IP PW	SM PW PW					
Version 1.1.0 • GPU performance enhancements applied to contouring		IP	IP	PW	PW	SM PW			
"PyNComp"	**Jul-Sep 2018**	**Oct-Dec 2018**	**Jan-Mar 2019**	**Apr-Jun 2019**	**Jul-Sep 2019**	**Oct-Dec 2019**	**Jan-Mar 2020**	**Apr-Jun 2020**	**Jul-Sep 2020**
Version 0.1 – internal release • Subset of NCL computational routines ported to Python based on CVDP pilot project (running average, dtrend, some climatologies) • Contributor's Guide		IP	SM IP PW						
Version 0.2 – internal release • More NCL computational routines ported (EOFs, interpolation)			PW	SM PW					
Version 1.0 – public release • Performances enhancements (xarray/dask)				PW	SM PW				
WRF-Python	**Jul-Sep 2018**	**Oct-Dec 2018**	**Jan-Mar 2019**	**Apr-Jun 2019**	**Jul-Sep 2019**	**Oct-Dec 2019**	**Jan-Mar 2020**	**Apr-Jun 2020**	**Jul-Sep 2020**
Version 1.3.0		CM							

• Cross-section enhancements • Bug fixes									
Version 1.3.1 • Contributor's Guide		IP	SM IP						
Version 2.0.0 • Xarray input support • Performance enhancements (dask)			PW PW	PW PW	SM PW PW				
PyNIO	**Jul-Sep 2018**	**Oct-Dec 2018**	**Jan-Mar 2019**	**Apr-Jun 2019**	**Jul-Sep 2019**	**Oct-Dec 2019**	**Jan-Mar 2020**	**Apr-Jun 2020**	**Jul-Sep 2020**
Version 1.5.2 • Improved support for Python 3.x	CM								
Version 1.5.3 • Critical bug-fix release (if needed)		CM							
Version 1.6.0 • Improved support for Python 3.x • Apache 2.0 License		CM							
Versions 1.6.*x* * • Bug fix releases (as needed)			SM						
* PyNIO is in maintenance mode while being evaluated against other Python packages like xarray and cfgrib. Critical bug fix releases will be deployed as possible.									
PyNGL	**Jul-Sep 2018**	**Oct-Dec 2018**	**Jan-Mar 2019**	**Apr-Jun 2019**	**Jul-Sep 2019**	**Oct-Dec 2019**	**Jan-Mar 2020**	**Apr-Jun 2020**	**Jul-Sep 2020**
Version 1.6.0 • Support for Python 3.x	CM								
Version 1.6.1 • Minor Python 3.x bug fixes • Apache 2.0 License		CM							
Version 2.0.0 • Xarray input support • Extended suite of examples to include xarray • Contributor's Guide				PW PW PW	PW PW PW	SM PW PW PW			
Version 3.0.0 • Performance enhancements (GPU) applied to contouring		IP	IP	PW	PW	PW	SM PW		

Pivot to Python and Training Roadmap									
Pivot to Python	**Jul-Sep 2018**	**Oct-Dec 2018**	**Jan-Mar 2019**	**Apr-Jun 2019**	**Jul-Sep 2019**	**Oct-Dec 2019**	**Jan-Mar 2020**	**Apr-Jun 2020**	**Jul-Sep 2020**
Host NCL Advisory Panel Meeting	CM				SM				
Officially announce NCL-to-Python pivot			SM						
CVDP Pilot Project (Convert NCL CVDP scripts to Python)	IP	IP	IP	PW					
Convert NCL application examples to Python and create new Python examples		IP	IP	PW	PW	PW	PW	PW	PW
Write an "NCL-to-Python" transition guide	IP	IP	IP	PW	SM				
Evaluate PyNGL with other Python graphics packages (Iris, matplotlib, cartopy, xESMF, MetPy, others)	IP	IP	IP	PW	PW				
Evaluate PyNIO with other Python file I/O packages (xarray, cfgrib, netcdf4python, others)		IP	IP	PW	PW				
Evaluate Python computational packages (ESMPy, xESMF, Iris, others)			PW	PW	PW				
Establish online forum for Python tools user support					SM				
Host informational webinars on pivot to Python plans			PW	PW	PW				
Training	**Jul-Sep 2018**	**Oct-Dec 2018**	**Jan-Mar 2019**	**Apr-Jun 2019**	**Jul-Sep 2019**	**Oct-Dec 2019**	**Jan-Mar 2020**	**Apr-Jun 2020**	**Jul-Sep 2020**
Official end of NCL Workshops	CM								
WRF-Python / VAPOR Tutorial at Boise State	CM								
Develop Data Carpentry training modules						PW	PW	PW	PW
Develop and host NCL-to-Python transition tutorials				PW	PW	PW	PW	PW	PW
Conduct training on how to contribute software				PW	PW	PW	PW	PW	PW
Host Python training similar to NCL Workshops						PW	PW	PW	PW

Conclusion

The “all-in-one” nature of NCL has been an oft-stated reason why it is so popular with its user base, as compared to Python whose modules are by nature widely dispersed and not consistently supported. This feature has become a detriment to NCL, however, resulting in a monolithic package who size and scope has become difficult to maintain and support with limited budgets and resources. As a result, we have not been able to quickly address the increasing demands put on NCL by the rapidly growing size and complexity of climate and weather data.

As stated in the beginning of the NCL Advisory Panel report:

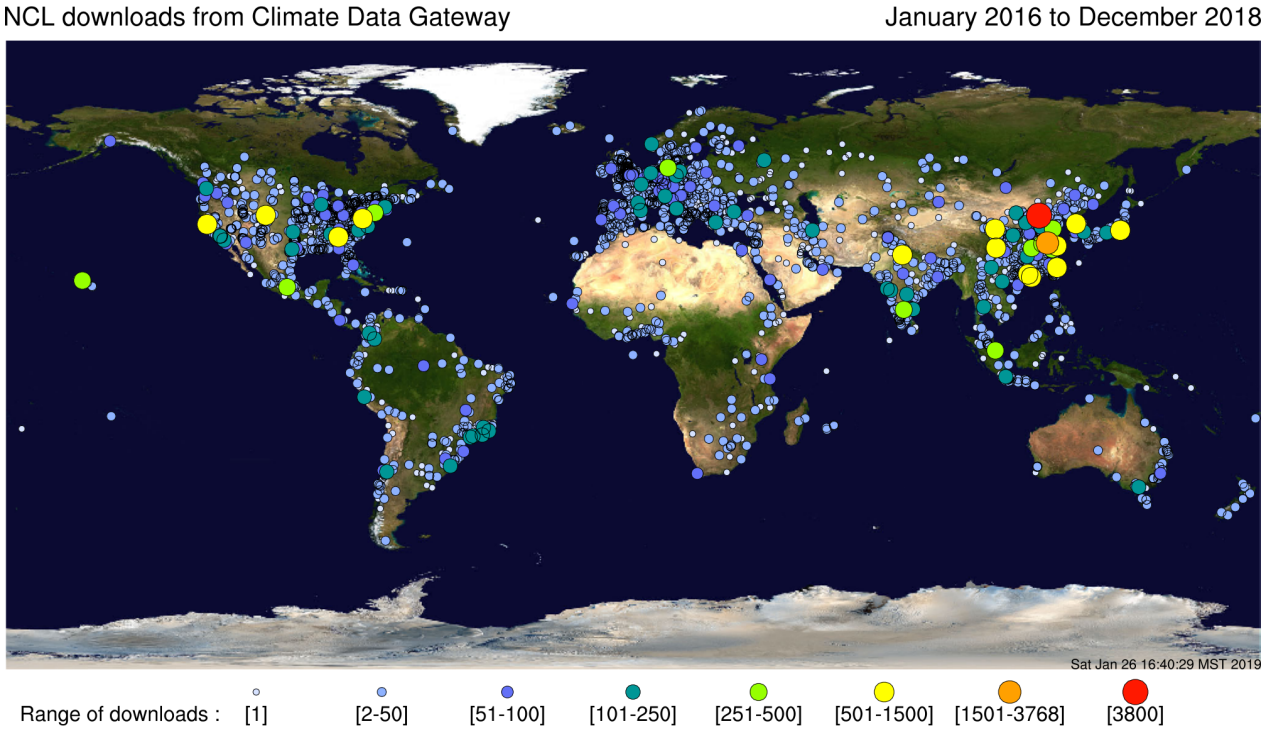
> “Integrating NCL into the ambient Python ecosystem is necessary for maintaining NCL’s utility and relevance as a tool for data analysis and visualization going forward.”

The panelists concluded by observing that “the NCL [team] is facing a period of significant change and transition which can be unsettling to users”, but also that “…the power and prevalence of Python, particularly with young scientists and coders, demands that NCL evolve to exist and thrive within the new Python ecosystem. This evolution will ensure that the graphical and computational abilities provided by NCL, particularly those unique to atmospheric and earth system sciences, are continued into the future irrespective of the programming language.”

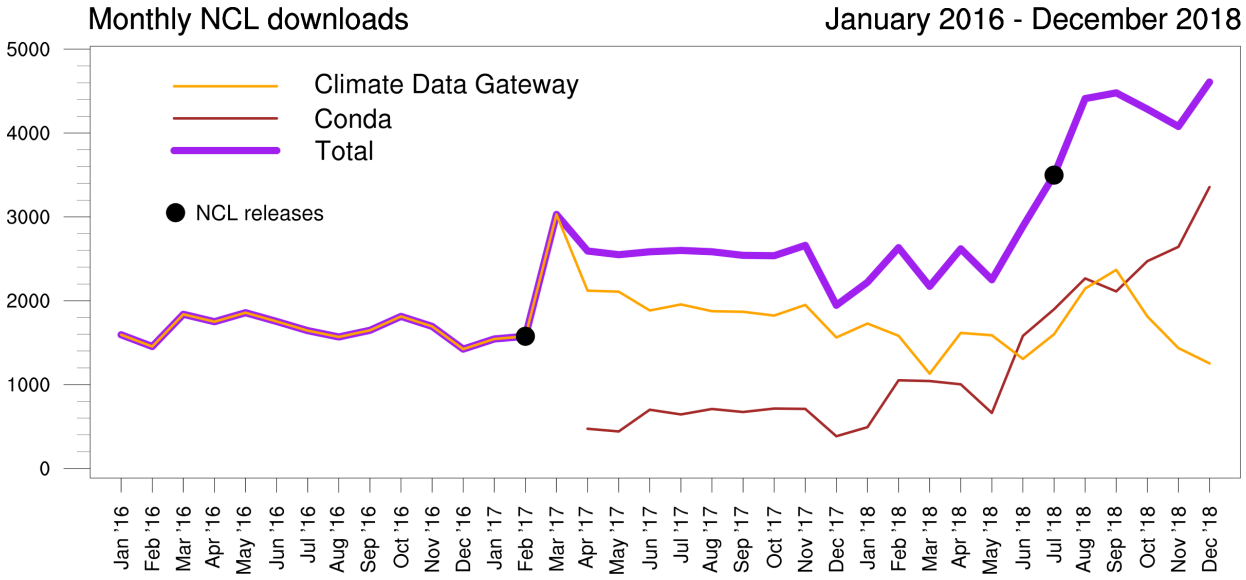
We hope that this report and roadmap helps ease the uncertainty around the plan to pivot to Python. We purposely did not include an “end-of-life” item for NCL itself in the roadmap, since the expectation is that NCL should be able to “live indefinitely” in open development mode for users who do not wish to transition to Python. This is similar to the way in that NCAR Graphics, which was retired in the mid 1990s, continues to live on with very minimal user support required.

The NCL development team is proud of the 20+ year of history of the NCAR Command Language. We are immensely grateful to the NCL user community that has helped us develop, improve, document, support, and train on this software. We're excited by the opportunities that this transition offers such as interoperability with the vast Python scientific ecosystem and easing adoption by new users.

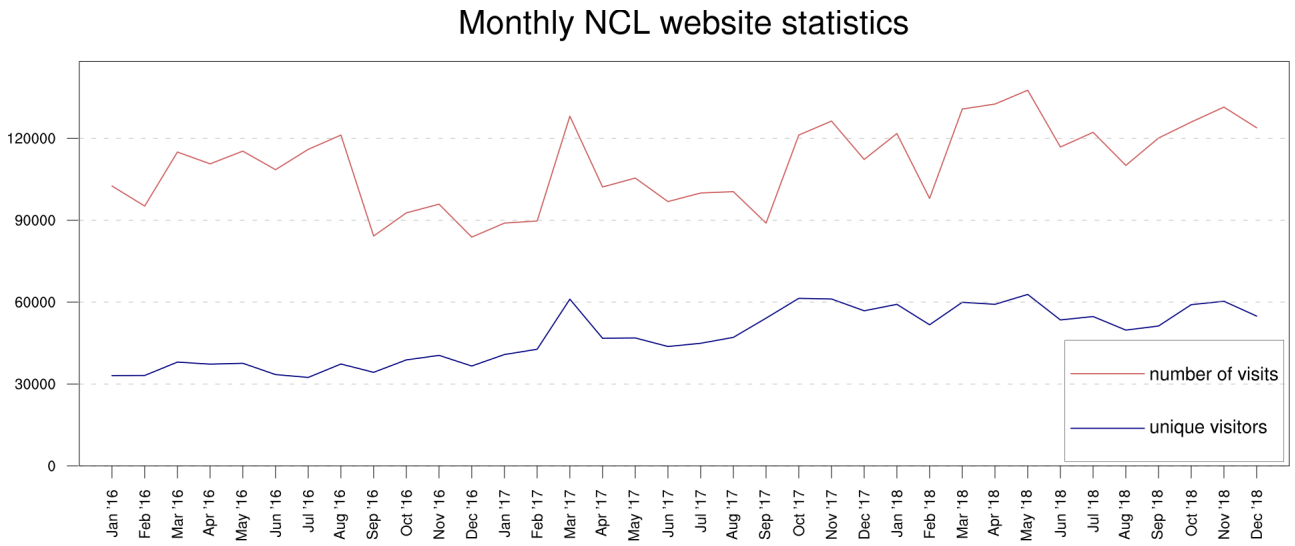
Appendix A – NCL and Python Tools Metrics



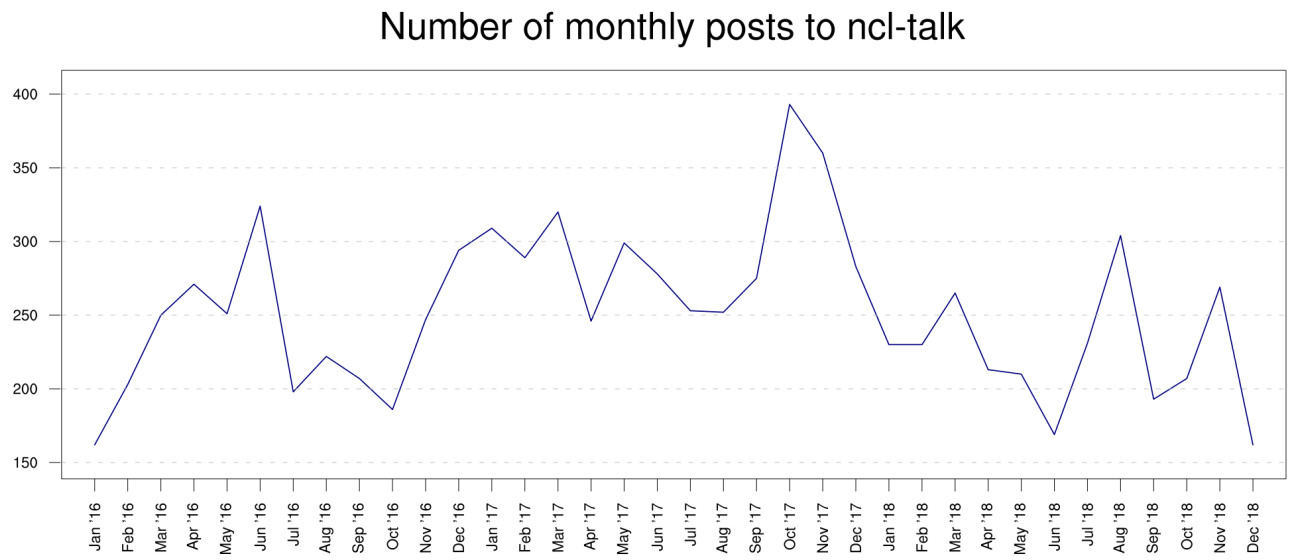
Each dot on this figure indicates the number of times NCL was downloaded from the Climate Data Gateway (CDG) (www.earthsystemgrid.org) from a particular location in the period January 2016 to December 2018. The total number of downloads of all locations in this time range is just over 62,898. This figure does not include downloads from "conda", because location metrics are not available. See the next figure for information on monthly CDG and conda downloads.



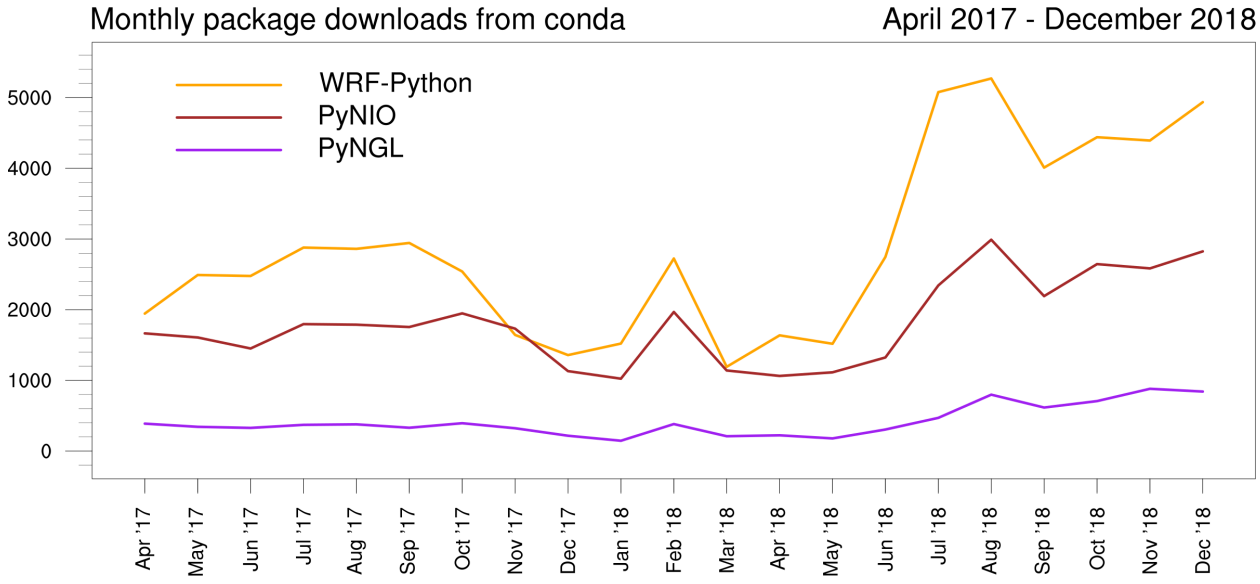
This figure shows the monthly download totals of NCL from both the CDG website and conda. NCL wasn't put under conda until April 2017, so the curve from January 2016 to March 2017 represents CDG downloads only.



This figure shows the monthly web statistics for the number of visits and unique visitors to the NCL website (www.ncl.ucar.edu). The statistics were obtained from webstats.ucar.edu, which is accessible only by UCAR employees.



This figure shows the number of monthly emails (both original posts and their responses) on the ncl-talk email list. Since January 2016, the list has averaged just over 250 posts per month.



This figure shows the monthly conda download totals of WRF-Python, PyNIO, and PyNGL. PyNIO and PyNGL used to be released on the Climate Data Gateway, but once conda became more stable, these two packages were moved exclusively to that system.