Modular Fermitools Proposal Draft

Our goal to improve the development workflow of the Fermitools proceeds with the following plan to modularize the package; to break the monolith of the fermitools into separate, discrete sub-packages, each of which may be developed and released independently.

The desire for such a plan is to reduce the scope of work required to update the fermitools and to release more frequently. We believe modularization will help by reducing build times, allowing faster iteration of development testing, and easier installation of dependent packages.

Modularization Schemes

The 3 schemes we consider are enumerated here:
* 2-Package (Fermitools + Extras)
* 3-Package (Base + Core + Extras)
* N-Package (Fermitools-meta)

2-Package (Fermitools & Extras)

This scheme would see a central block of the main Fermitools retained, while a group of less commonly used elements of the tools split off into a package named "Fermitools-extras".

The core "Fermitools" package would hold all base libraries and dependencies "tip", "likelihood", "irfs"... etc. and all important end-point utilities such as gtbin, gtselect, and the python interface modules such as gt_app, pyLikelihood etc. The "Fermitools-extras" package should contain less commonly used tools like pgwave, gtorbsim and gtobssim, along with their external dependencies, like ROOT. Ideally ROOT would only be a dependency of "Fermitools-extras", and would no longer be a dependency of "Fermitools". The now smaller "Fermitools" package would be a dependency of the "Fermitools-extras" package.

``[Fermitools] ==> [Fermitools-extras]``

3-Package (Base + Core + Extras)

This scheme builds upon the 2 package scheme by further partitioning off a set of "Base" modules from the core Fermitools. This base package would contain all the modules that are rarely updated dependencies of most other fermitools: "tip", "ST_APP", "facilities", etc. would all be in this "Fermitools-base" package. Libraries under active development (Likelihood) along with all user applications (gtbin, modeleditor) would remain in the "Fermitools" package, which we might internally refer to as Fermitools-core.

The "Fermitools-extra" package would not be modified in this scheme.

``[Fermitools-base] ==> [Fermitools] ==> [Fermitools-extras]``

N-Package

This segmentation could be further applied recursively until all of the Fermitools exist in conda as separate packages. A Conda package for Tip, a Conda package for Likelihood, ... etc. In this case each package would include its dependencies dynamically at both build and install time. The full suite of Fermitools could be installed in full as a conda meta-package, or separate meta-packages could be written for each tool this would provide the most support for users to customize their use of the Fermitools.

Implementation order

It makes sense that the 2-package scheme should be implemented first, as it is a strict subset of the 3-package scheme and lessons learned in its creation can be applied to the 3-pacakge scheme. At that point we may find there is **no need** for an N-pacakge scheme, or that the act of modularization has not solved the problems we had hoped it would.

Release Concerns

Version Number

With separate packages the issue of release synchronization grows. While packages may be built and tested separately we are obliged to create a single release of the tools, with a single version number. As we currently know of no way to update the existing version number of a built conda package the following situation will be quite common. A single package update to fermitools-core for example with no change to fermitools-extra would still require both packages to be rebuilt that both may have the same version in anaconda.

It may be feasible to decouple each package's version number and have the "Release" be a conda meta-package which includes both as a dependency, but this adds further complexity for our users who I expect would be confused by this change.

We may chose to abandon the need to keep these separate packages version-equivalent, and simply allow fermitools-core to run several versions ahead of fermitools-extras. However I expect internal push-back from institutional actors on this idea.