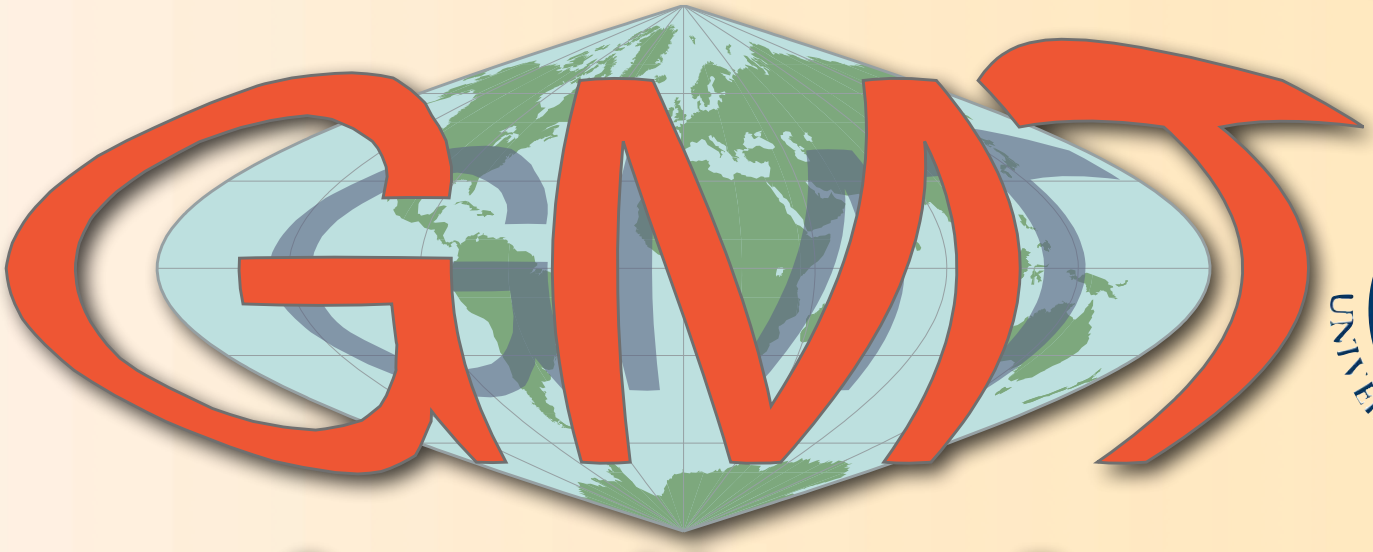
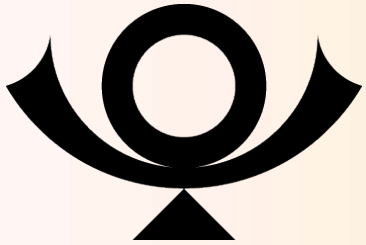
State of The Generic Mapping Tools



THE GENERIC MAPPING TOOLS



GMT SUMMIT AGENDA

Aug 15–19 August, 2016

Daily schedule: AM session 8:30 am – noon, Lunch, PM session 1:30 pm – 5:00 pm

MONDAY: Joint Meeting with the GMT Steering Committee

am: Sandwell: Introductions
Wessel: "State of the Generic Mapping Tools and Plans for the Future"
Q/A

pm: Steering committee member presentations (30 min each): Applications & needs
Caress, Soofi, Diggs, Basset, Sandwell
Q/A

TUESDAY: Joint Meeting, cont., Recommendation and Debriefing

am: Steering committee recommendations

Steering Committee Adjourns

pm: Developer meeting debriefing and prep for postdoc candidate interviews

WEDNESDAY: Search for Post-Doctoral Candidate

am: Skype and Live interviews of postdoc candidates.
pm: Deliberation on postdoc candidates, ranking, action plan

THURSDAY: Development Prioritizations

am: Discuss short and long-term goals
pm: Identify actions, set time-table and assign tasks

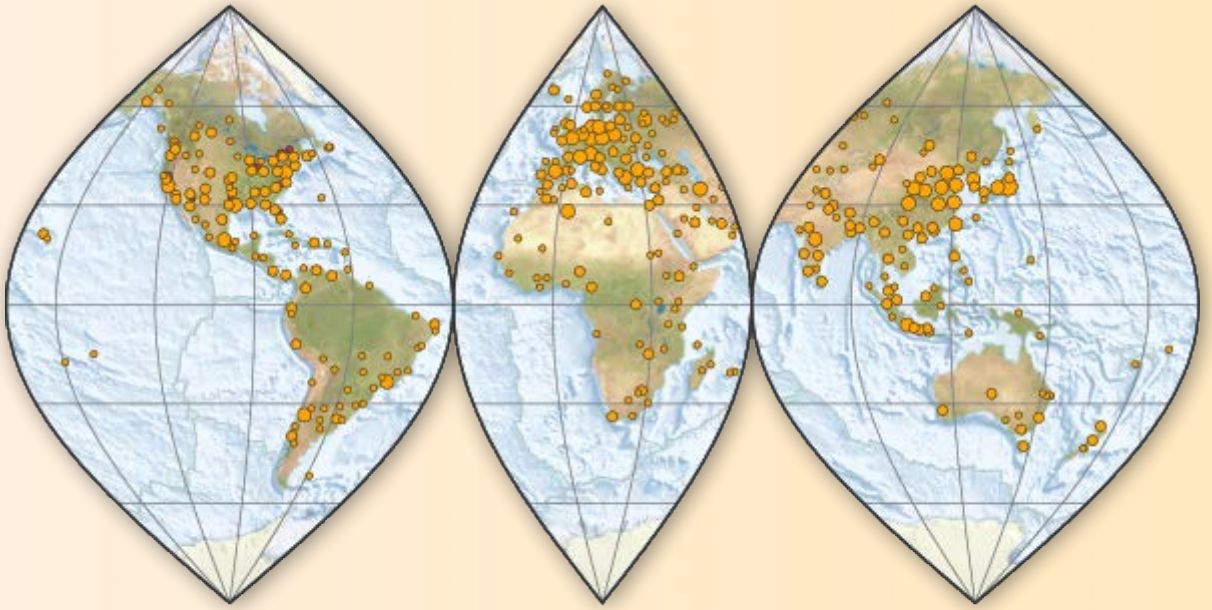
FRIDAY: Releasing a New GMT Version

am Release GSHHG 2.3.6 and document entire process
am: Release GMT 4.5.15 and document entire process
am: Release GMT 5.3 and document entire process
pm: Open

GMT Overview

- Status of GMT
- The Public GMT API
- GMT 5.3 Release Aug 19
- Funding Situation
- Development Plans
- Post-Doctoral Search
- Succession Plans

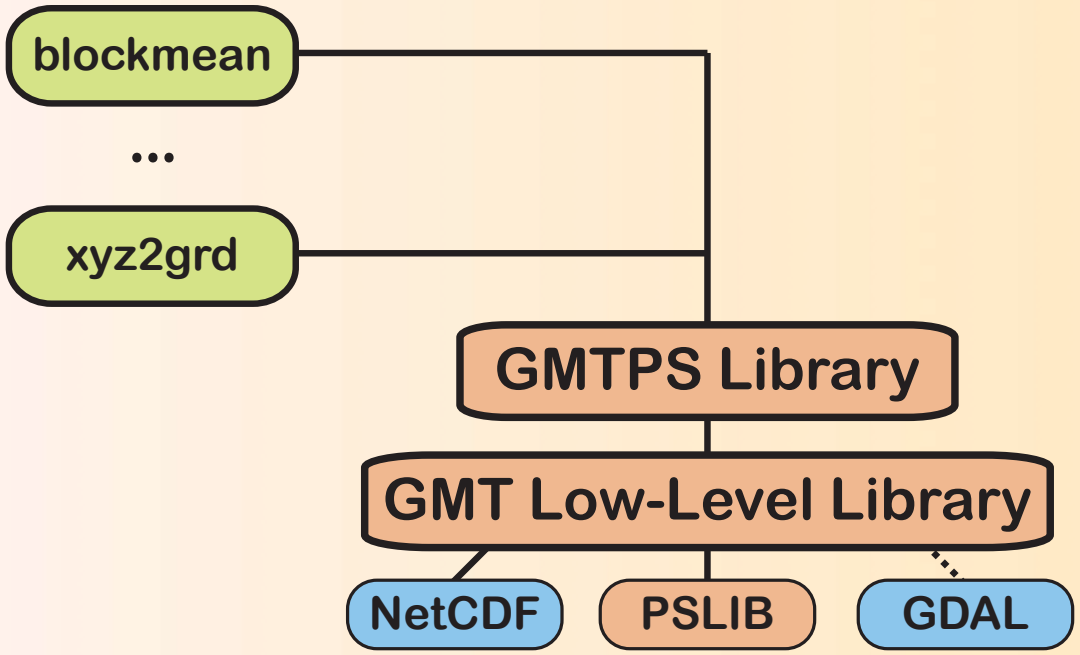
Status of GMT



- Tens of thousands of users worldwide
- 15% at US institutions [NSF cares]
  - 50% OSX, 40% Windows, 10% Linux
- 85% non-US [NSF cares less]
  - 25% OSX, 65% Windows, 10% Linux

Status of GMT

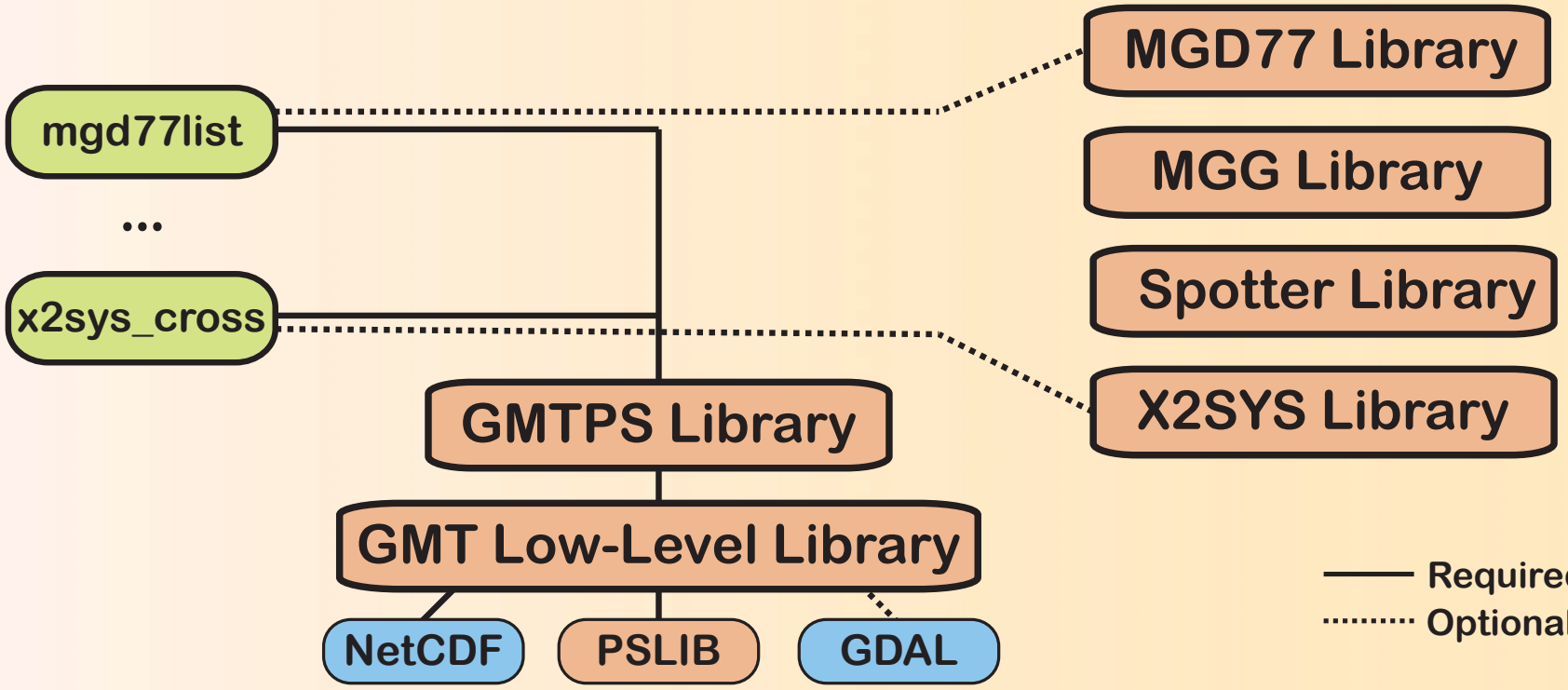
- 1988-2013 [GMT 1–4]: Each GMT module was a separate program linked to two low-level GMT libraries, the PostScript library, and some external libraries



GDAL since GMT 4.5.1 [9/2009]

Status of GMT

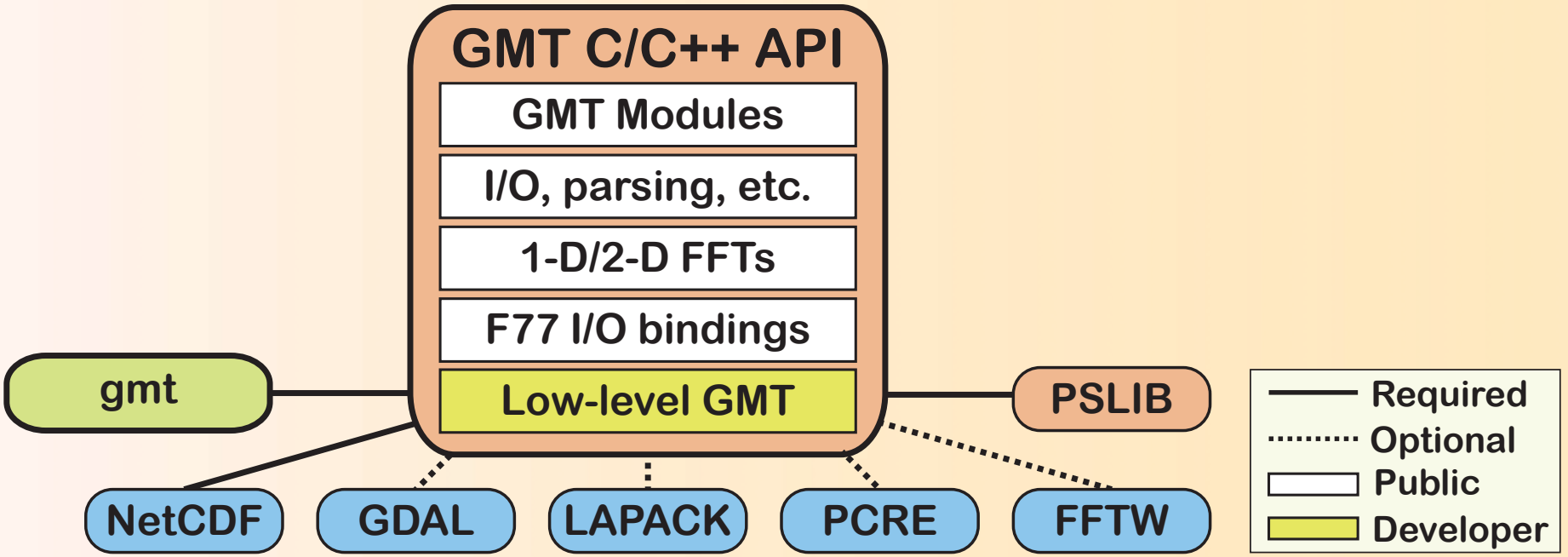
- 1988-2013 [GMT 1–4]: Each GMT module was a separate program linked to two low-level GMT libraries, the PostScript library, and some external libraries



GDAL since GMT 4.5.1 [9/2009]

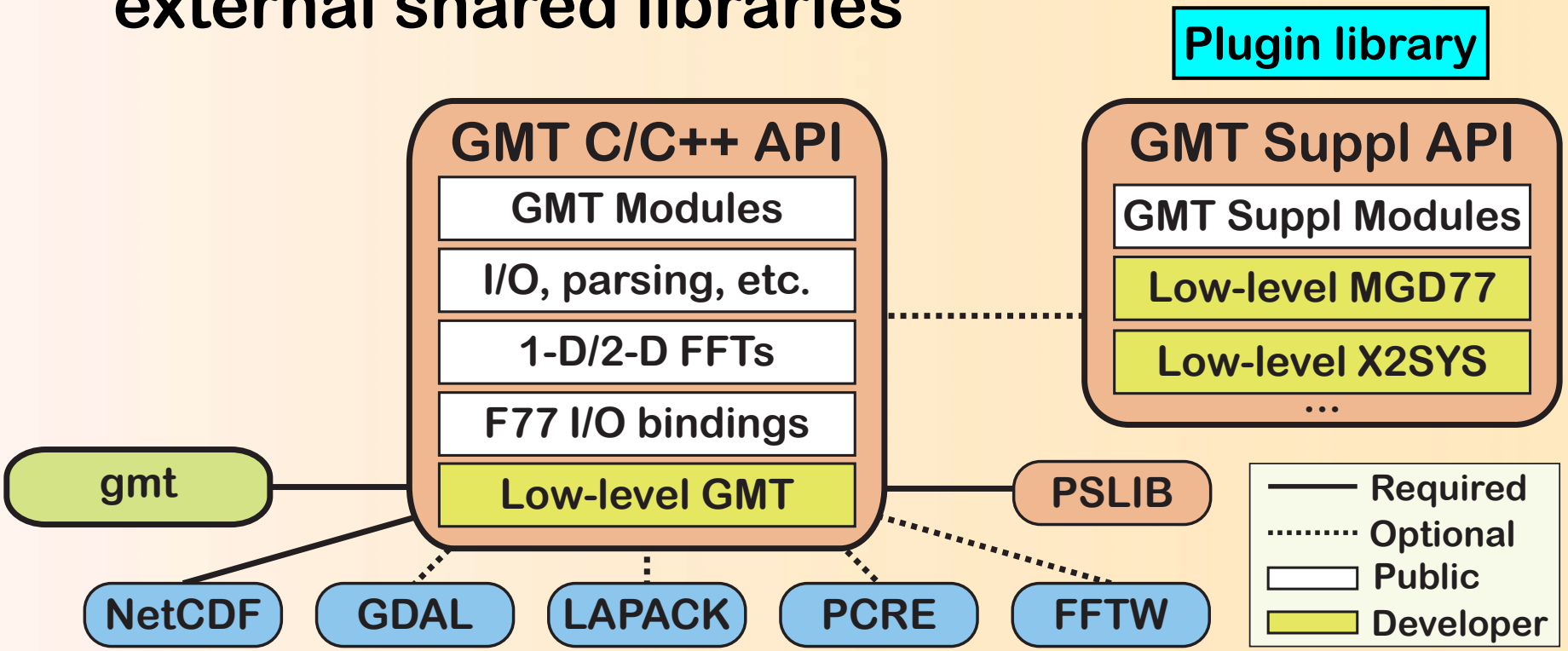
Status of GMT

- Nov 2013 [GMT 5]: There is only a single program (gmt), a shared library with high-level documented API modules and functions and low-level GMT functions, the shared/ updated PostScript library, and several external shared libraries



Status of GMT

- Nov 2013 [GMT 5]: There is only a single program (gmt), a shared library with high-level documented API modules and functions and low-level GMT functions, the shared/ updated PostScript library, and several external shared libraries



Status of GMT

- There are no longer programs called blockmean, pscoast, etc. These are now modules accessible via the single executable we have called gmt

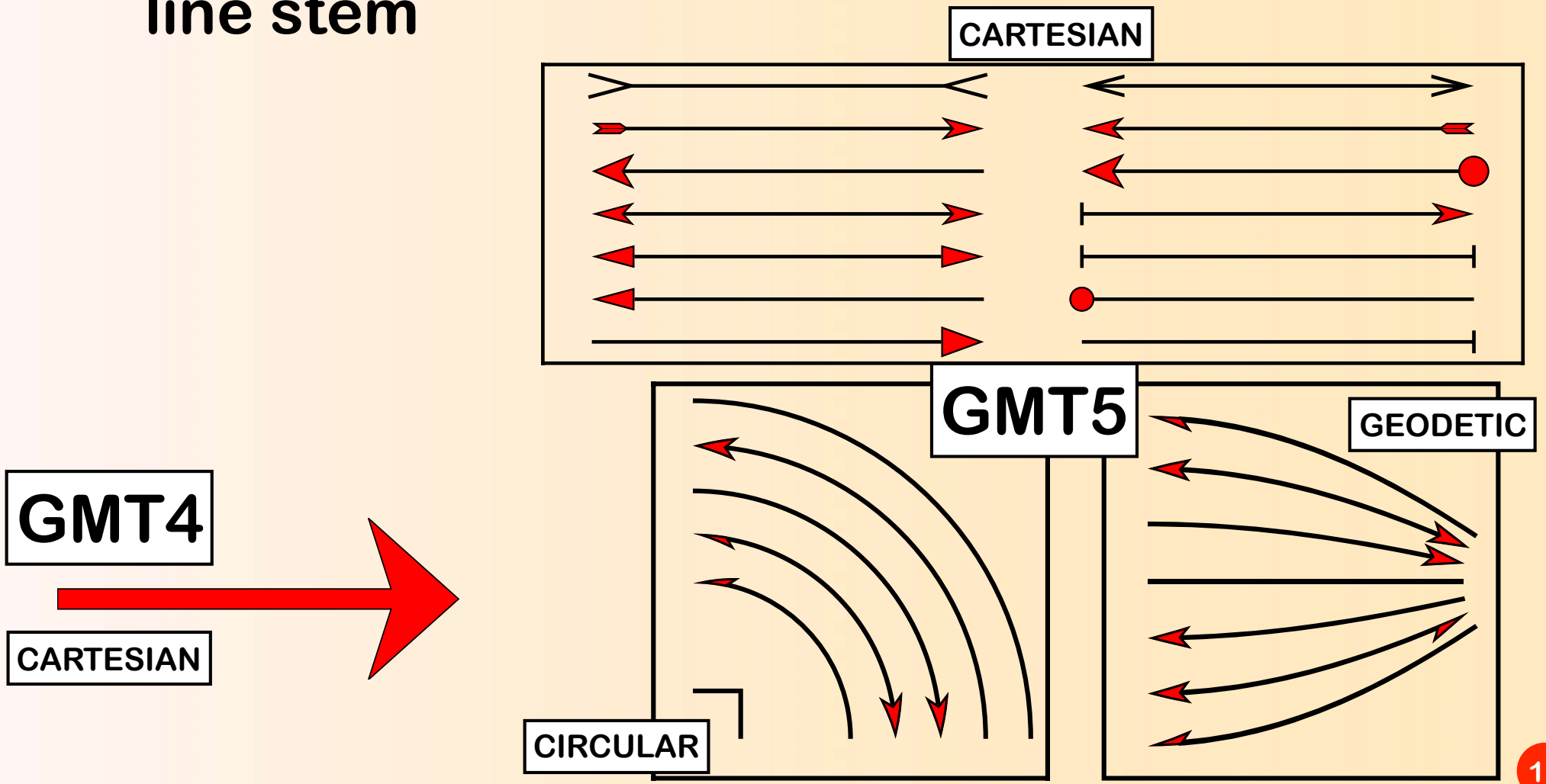
How does this affect users?

GMT 4: pscoast -Rg -JH180/6i -Gred > t.ps

GMT 5: gmt pscoast -Rg -JH180/6i -Gred > t.ps

Status of GMT

- The arrow symbol has been completely redesigned to use a polygonal head and a line stem



Status of GMT

How did this change affect users? Minimally.

How did we handle this crisis for existing scripts?

1. Install symbolic links called pscoast, grdimage, etc., that all point to gmt. When gmt is launched via a link it knows and calls the relevant module
2. Added csh and bash functions to handle automatic pre-pending of the gmt executable
3. Accept deprecated GMT4-syntax in compatibility mode, which is enabled by default

Arrows are not completely portable, but the new system is deemed superior to the old limited system

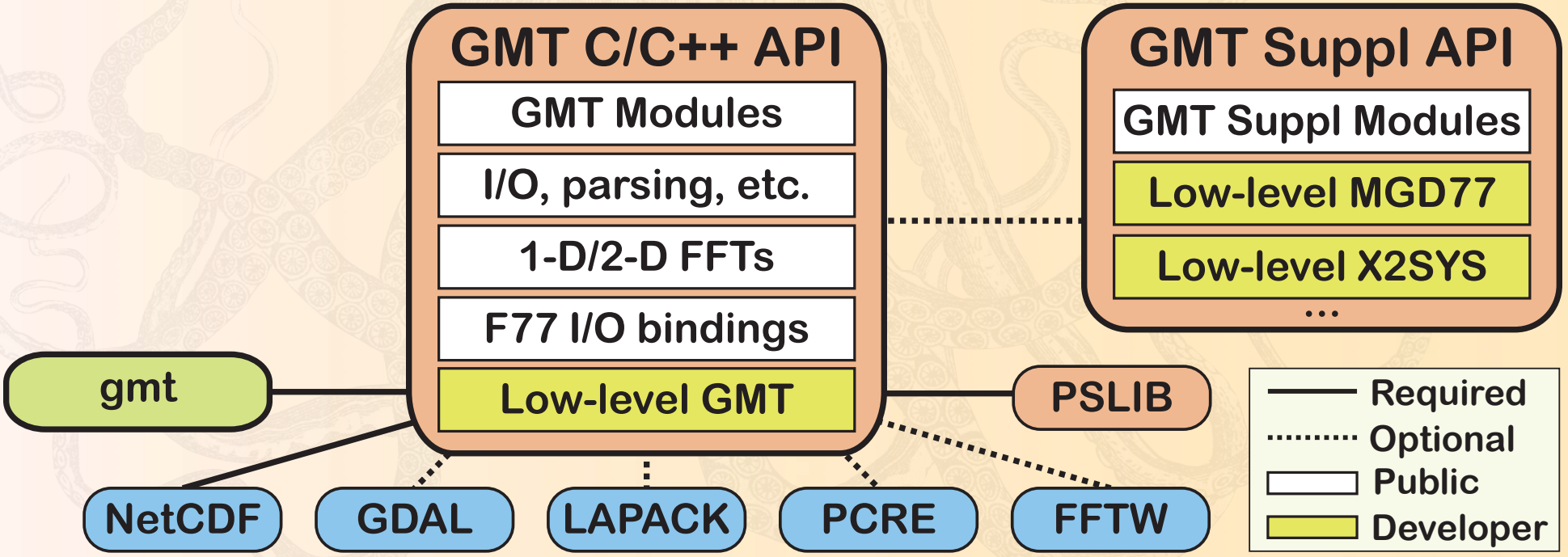
Status of GMT

Why did we change?

1. We needed GMT programs to be able to call other GMT programs, but we could only do so via system calls
2. We needed to avoid namespace collisions as other packages also have programs called triangulate, surface, etc.
3. We wanted to let developers build new tools on top of GMT, e.g., new supplements.
4. We wanted to provide access to GMT from external computational environments, such as Matlab, Python, Julia, R, …

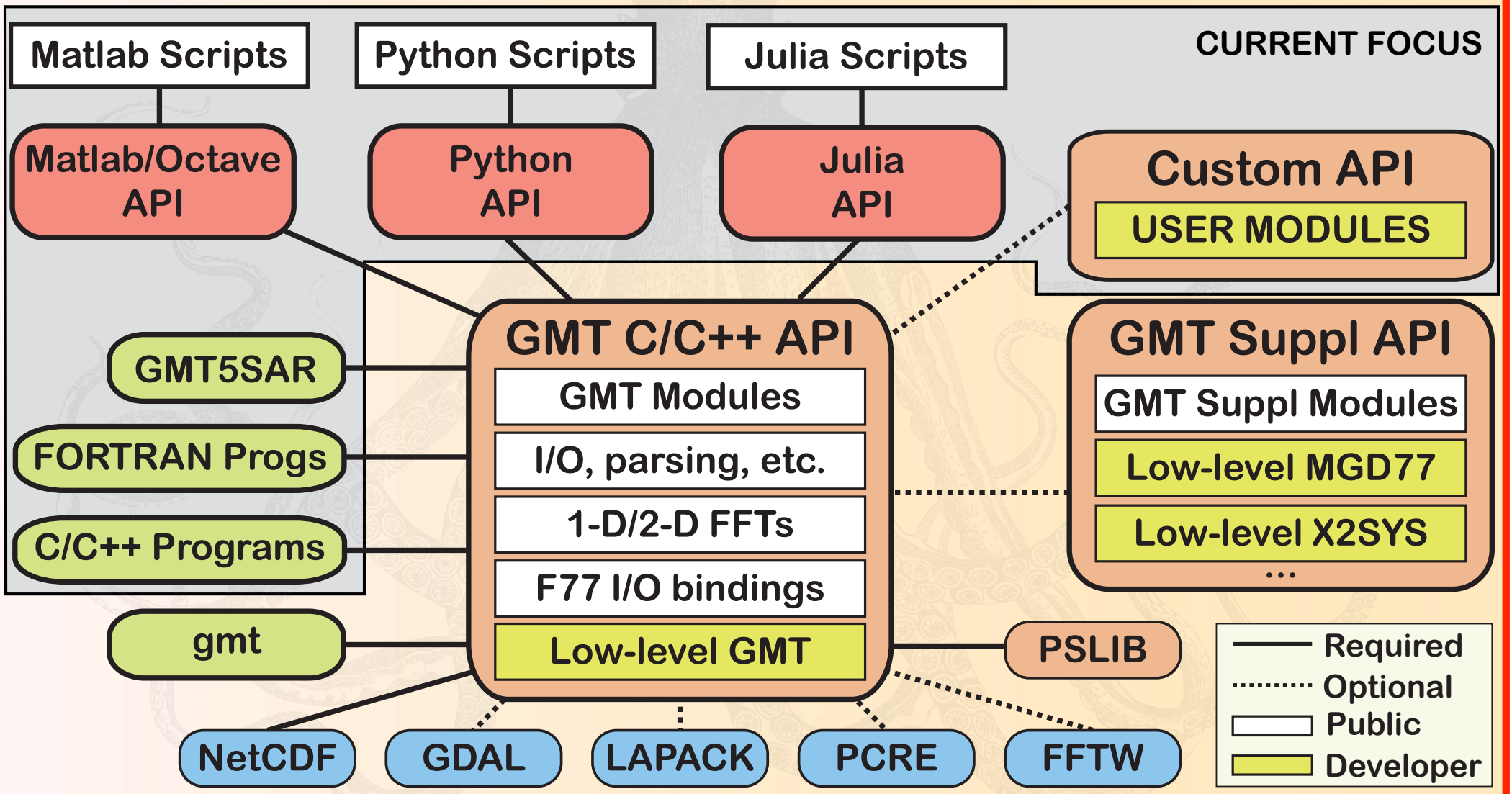
Status of GMT

The GMT Octopus



Status of GMT

The GMT Octopus



The Public GMT API

Why do we need a GMT API?

- Extend GMT to other computing environments [Matlab, Julia, Python]
- Simplify design of new core and supplemental modules
- Allow custom modules to be used by calling the gmt executable
- Simply use the API from external programs [GMT5SAR, MB-System]

The Public GMT API

What are the tasks of the GMT API?

- **Create** a self-contained, thread-safe **GMT session** (or several sessions, possibly running in parallel)
- **Represent** data-in-memory as **VirtualFiles** that GMT modules can read from or write to like ordinary files
- **Accept** program options via **linked option lists, text arrays** (e.g., argc, argv[]), or a **command string**
- **Call** any number of **GMT modules**
- **Export** results from **VirtualFiles** to actual files
- **Destroy** the session and all its resources

The Public GMT API

Hardwiring a "grdtrack.c" program

```
#include "gmt.h"

int main (int argc, char *argv[]) {
      void *API = NULL;       /* GMT API control structure */

      /* 1. Initializing new GMT session */
      API = GMT_Create_Session (argv[0], GMT_PAD_DEFAULT, GMT_SESSION_NORMAL, NULL);

      /* 2. Run the grdtrack module with given arguments */
      GMT_Call_Module (API, "grdtrack", argc-1, argv+1);

      /* 3. Destroy GMT session */
      GMT_Destroy_Session (API);

      exit (0);
}
```

The API defines functions, structures, and constants

The Public GMT API

Show VirtualFile usage

```
#include "gmt.h"
int main (int argc, char *argv[]) {
  void *API;                          /* The API control structure */
  struct GMT_DATASET *D = NULL;       /* Structure to hold input dataset */
  struct GMT_GRID *G = NULL;          /* Structure to hold output grid */
  char input[GMT_STR16] = {""};       /* String to hold virtual input filename */
  char output[GMT_STR16] = {""};      /* String to hold virtual output filename */
  char args[128] = {""};              /* String to hold module command arguments */

  /* 1. Initialize the GMT session */
  API = GMT_Create_Session ("test", 2U, 0, NULL);
  /* 2. Read in our data table to memory */
  D = GMT_Read_Data (API, GMT_IS_DATASET, GMT_IS_FILE, GMT_IS_PLP, GMT_READ_NORMAL,
      NULL, "A.txt", NULL);
  /* 3. Associate our data table with a virtual file */
  GMT_Open_VirtualFile (API, GMT_IS_DATASET, GMT_IS_PLP, D, input);
  /* 4. Create a virtual file to hold the resulting grid */
  GMT_Create_VirtualFile (API, GMT_IS_GRID, GMT_IS_SURFACE, output);
  /* 5. Prepare the greenspline module arguments */
  sprintf (args, "-R0/7/0/7 -I0.2 -D1 -St0.3 %s -G%s", input, output);
  /* 6. Call the greenspline module */
  GMT_Call_Module (API, "greenspline", GMT_MODULE_CMD, args);
  /* 7. Obtain the grid from the virtual file */
  G = GMT_Read_VirtualFile (API, output);
  /* 8. Write the grid to file */
  GMT_Write_Data (API, GMT_IS_GRID, GMT_IS_FILE, GMT_IS_SURFACE, GMT_GRID_ALL, NULL, "A.nc", G);
  /* 9. Destroy the GMT session */
  GMT_Destroy_Session (API);
};
```

The Public GMT API

Snippet from grdimage.c on auto-shading

```
/*-------------------- This is the grdimage main code --------------------*/

if (Ctrl->I.derive) {    /* Auto-create intensity grid from data grid */
    char virtual_file[GMT_STR16] = {""}, args[GMT_LEN256] = {""};
    GMT_Report (API, GMT_MSG_VERBOSE, "Derive intensity grid from data grid\n");
    /* Create a virtual file to hold the intensity grid */
    if (GMT_Create_VirtualFile (API, GMT_IS_GRID, GMT_IS_SURFACE, virtual_file))
        Return (API->error);
    /* Prepare the grdgradient arguments using default settings -A45 -Nt1 */
    sprintf (args, "%s -G%s -A45 -Nt1", Ctrl->In.file[0], virtual_file);
    /* Call the grdgradient module */
    if (GMT_Call_Module (API, "grdgradient", GMT_MODULE_CMD, args))
        Return (API->error);
    /* Obtain the data from the virtual file */
    if ((Intens_orig = GMT_Read_VirtualFile (API, virtual_file)) == NULL)
        Return (API->error);
}
...
```

The Public GMT API

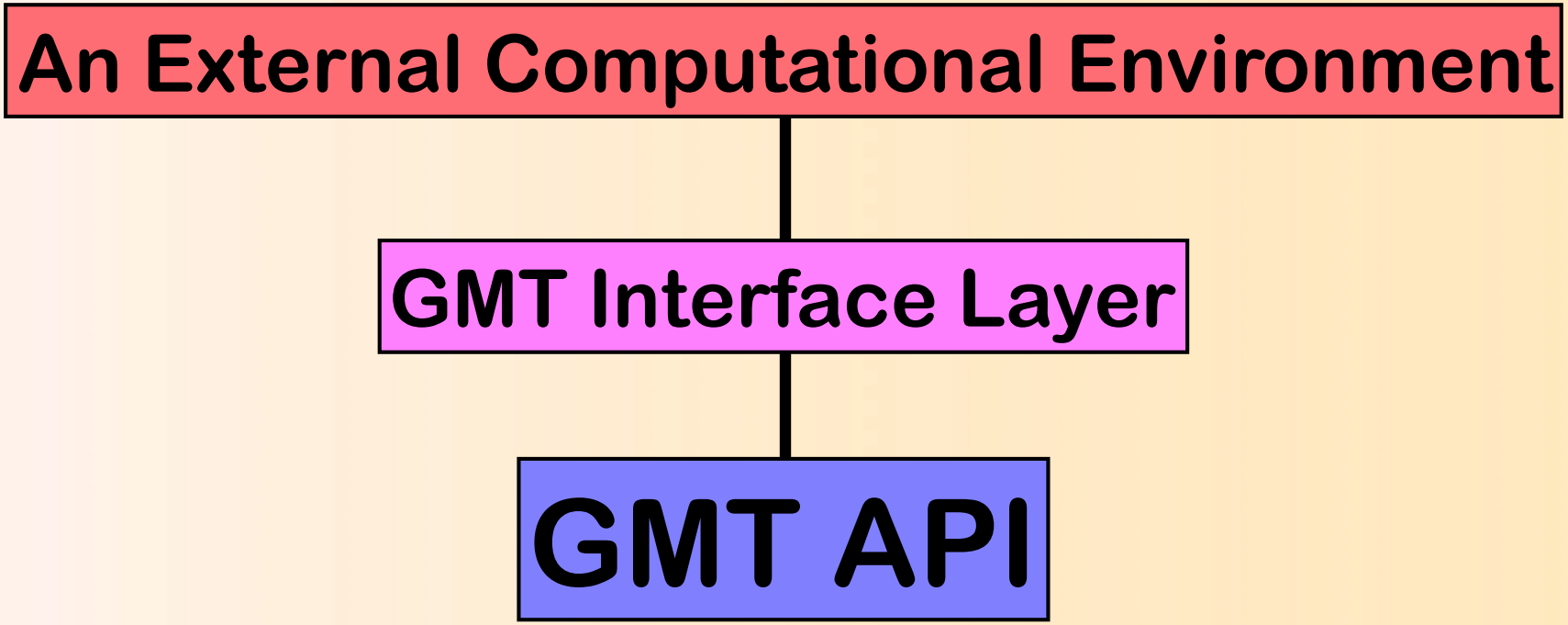
The Public GMT Data Structures

- GMT_DATASET: Data set (purely numerical)
- GMT_GRID: Grid
- GMT_IMAGE: Image
- GMT_PALETTE: Palette (CPT)
- GMT_POSTSCRIPT: PostScript
- GMT_TEXTSET: Text set (mixed numerical/text)

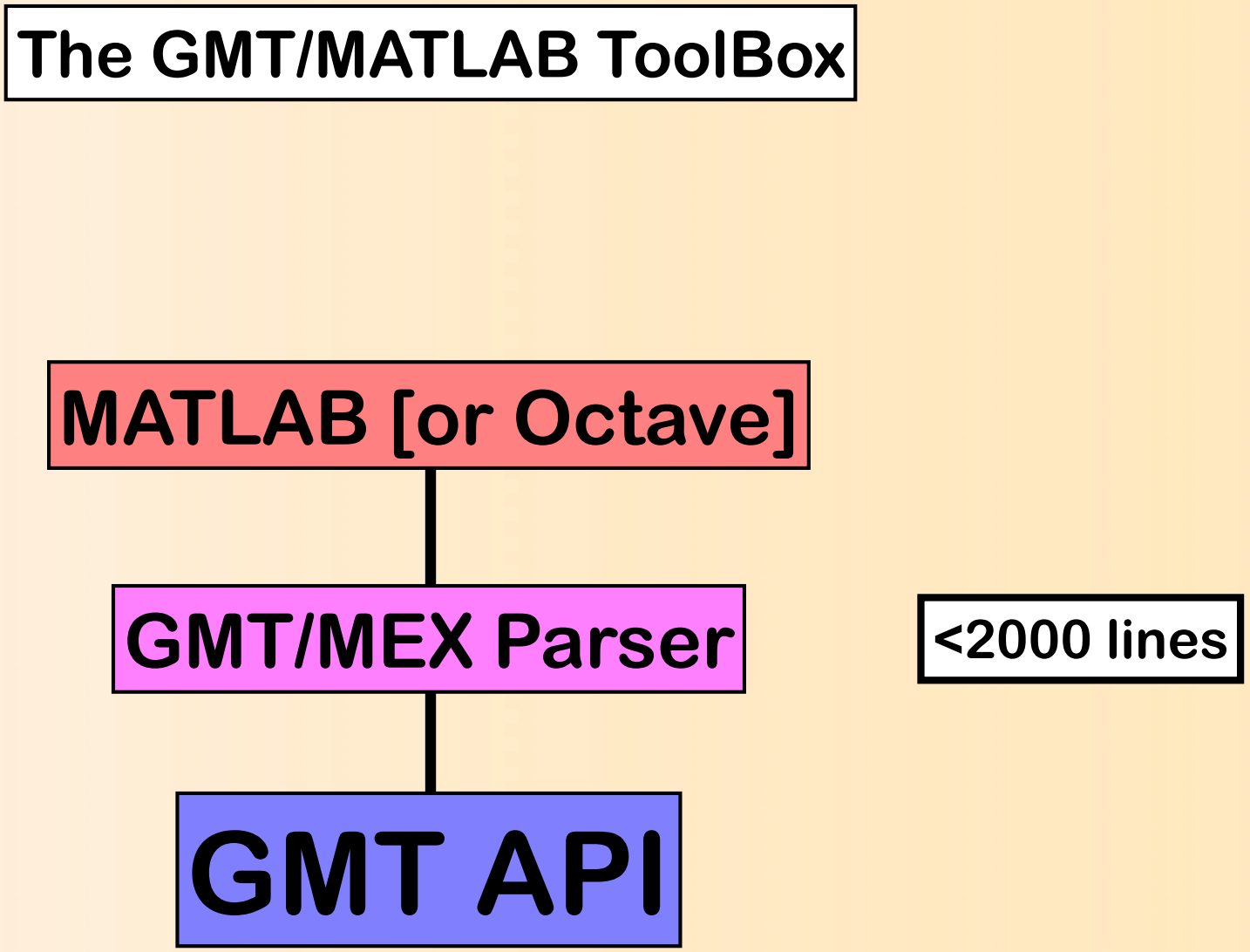
- GMT_MATRIX: Matrix (for plain user matrix)
- GMT_VECTOR: Vector (for array of user vectors)

The Public GMT API

How to use the API from other software?



The Public GMT API



[output objects] = gmt (*modulename*, *optionstring*[, input objects]);

The Public GMT API

How does the GMT/MATLAB Toolbox work?

1. **The GMT/MEX parser deals with:**
   a. Preparing MATLAB arrays and structures to be sent to GMT modules as virtual files
   b. Prepares MATLAB arrays and structures based on GMT module output in the form or virtual files
2. **The main gmt.m script calls gmtmex which deals with:**
   a. Check arguments and get input pointers
   b. Use the parser to build in/out structures
   c. Call the GMT module
   d. Convert GMT output to MATLAB structs and assign left-hand-side pointers

The Public GMT API

Example of GMT UNIX command-line

```
gmt filter1d raw_data.txt –Fm15 > filtered_data.txt
```

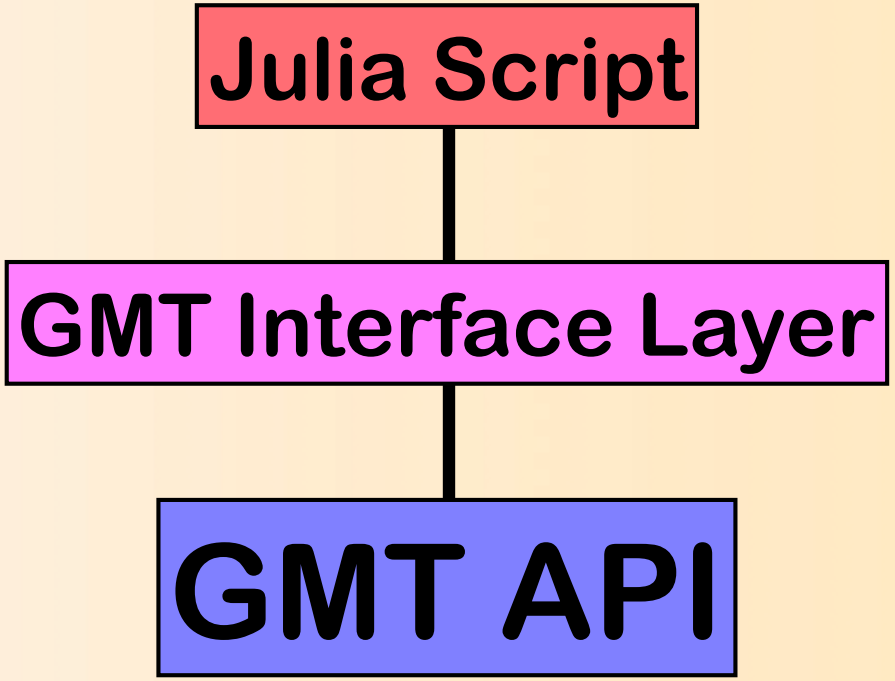
Example of GMT/MATLAB toolbox command

```
load raw_data; % Read in the data set
filtered = gmt ('filter1d', '-Fm15', raw_data);
```

DEMO

The Public GMT API

The Julia Interface



DEMO

GMT 5.3 Release

- As a group, we will release these three products on Friday, Aug 19:
  - GSHHG 2.3.6: Various improvements
  - GMT 4.5.15: Latest bug fixes for GMT4
  - GMT 5.3: Latest point-release for GMT5
- We will carefully document all steps so that it is reproducible by others

GMT 5.3 Release

What is new in GMT 5.3?

- Four new modules:
  1. pssolar [core]
  2. gpsgridder [potential]
  3. gmtpmodeler [spotter]
  4. rotsmoother [spotter]

GMT 5.3 Release

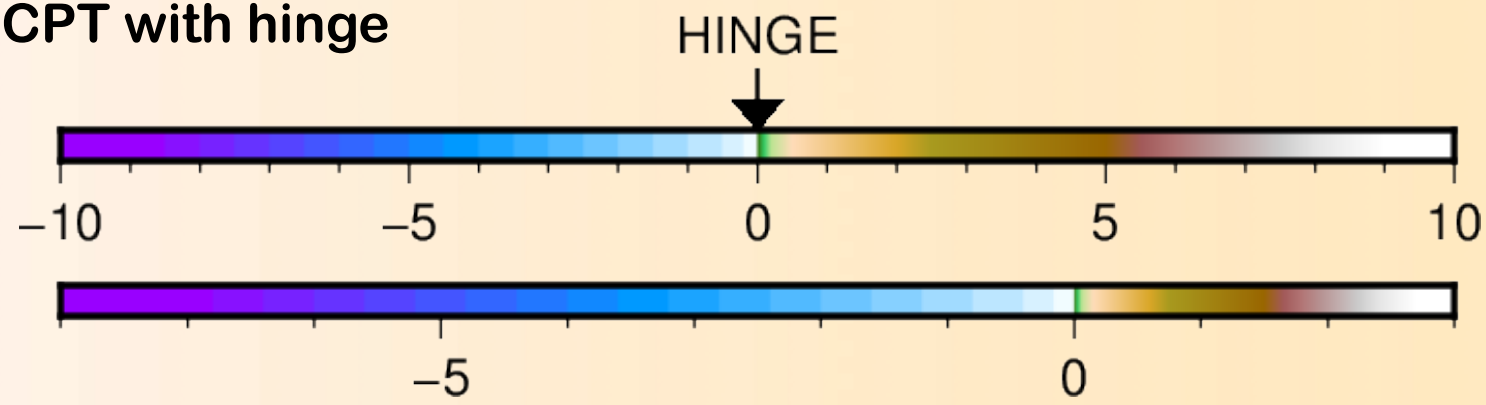
Lots of major and minor enhancements

- Dynamic & Hinged CPT + flexible custom CPT setup
- Standardized Map Embellishment Setup
- Decorated Lines and Line Ending Options (psxy)
- Auto-Labeling of Points (pstext)
- Line Segmentation and Networks
- Miscellaneous:
  - Geographic sizes for geo-symbols
  - EPS-based custom symbols
  - Spherical statistical operators
  - API expansion
  - More than 50 other new features

GMT 5.3 Release

CPT Improvements

Dynamic CPT with hinge



Flexible CPT generation for custom palettes

```
echo   0     red       100    red > seis.cpt
echo   100   green     300    green >> seis.cpt
echo   300   blue    10000    blue >> seis.cpt
```

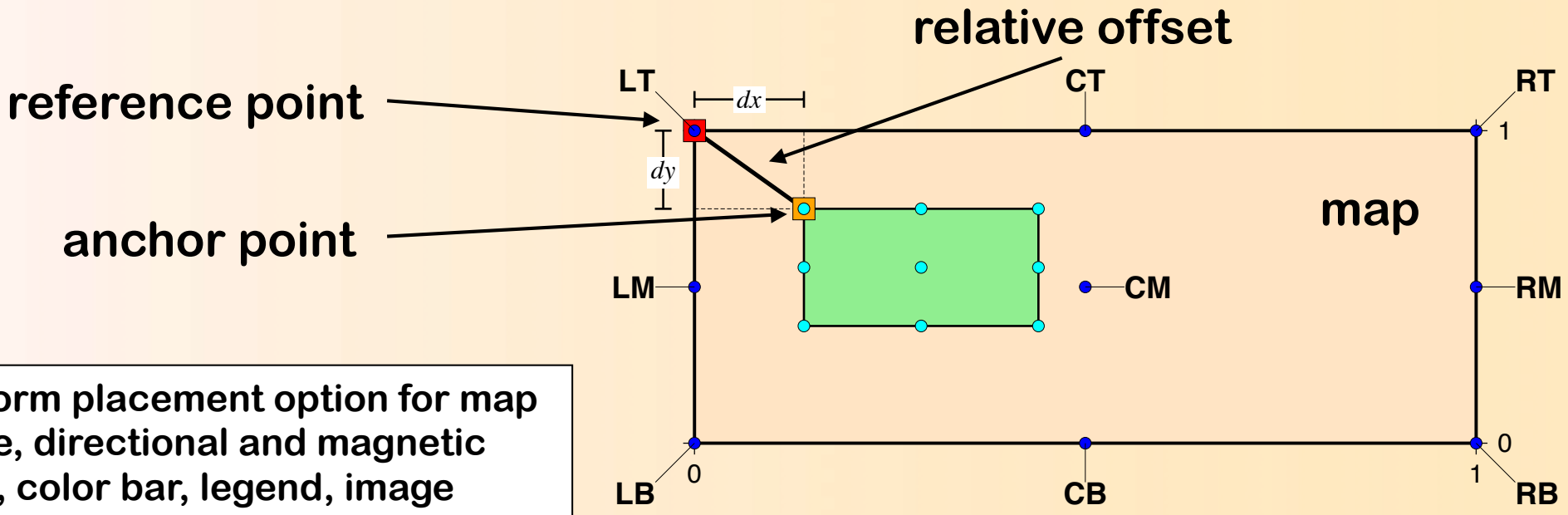
```
gmt makecpt -Cred,green,blue -T0,100,300,10000 -N > neis.cpt
```

GMT 5.3 Release

Embellishment Improvements

- Numerous module options were given new and more consistent syntax via option modifiers, reducing the variability across modules

Set anchor point for map embellishments

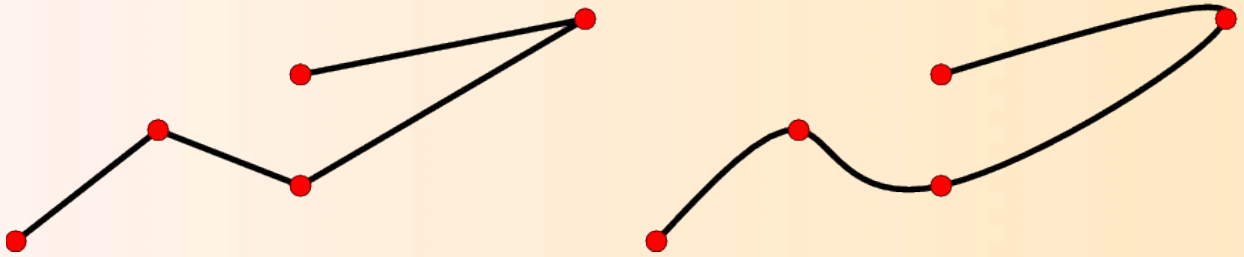


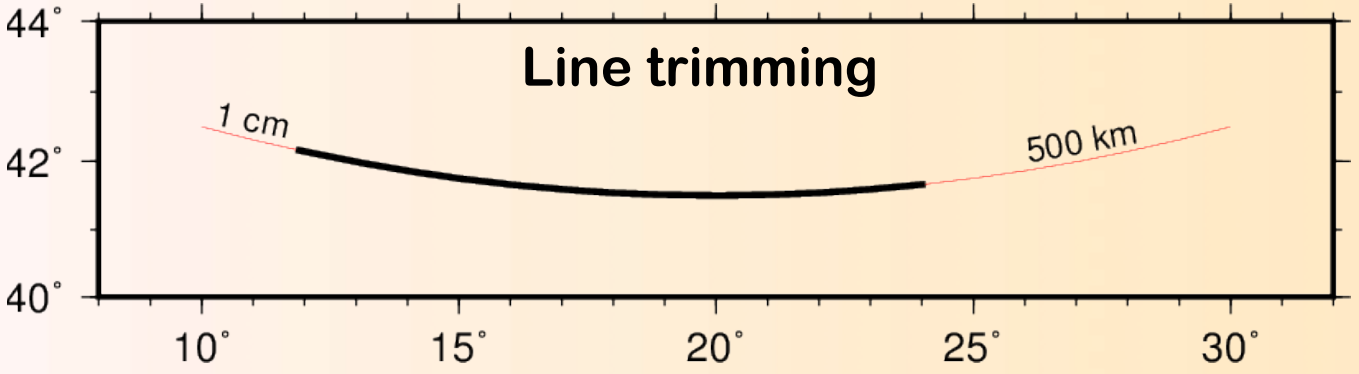
Uniform placement option for map scale, directional and magnetic rose, color bar, legend, image overlay, GMT logo, and map insert

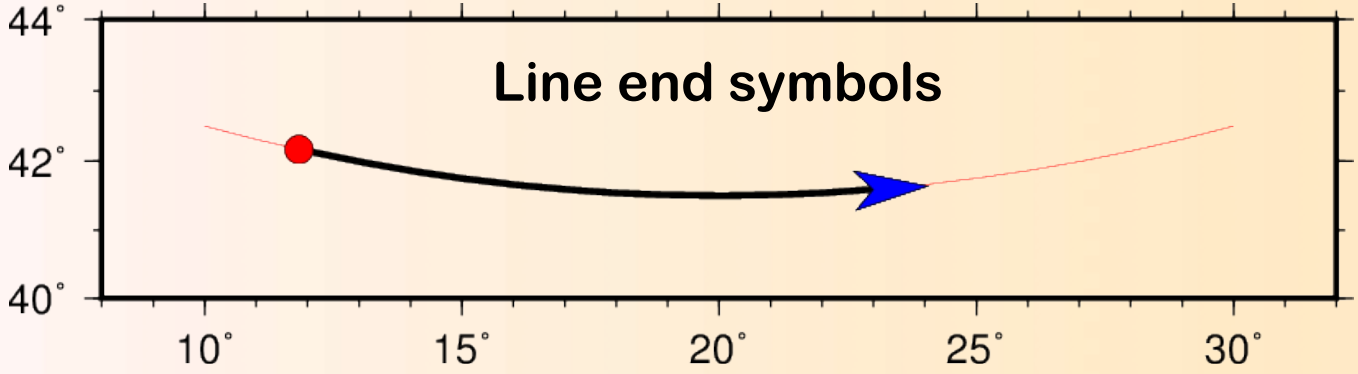
GMT 5.3 Release

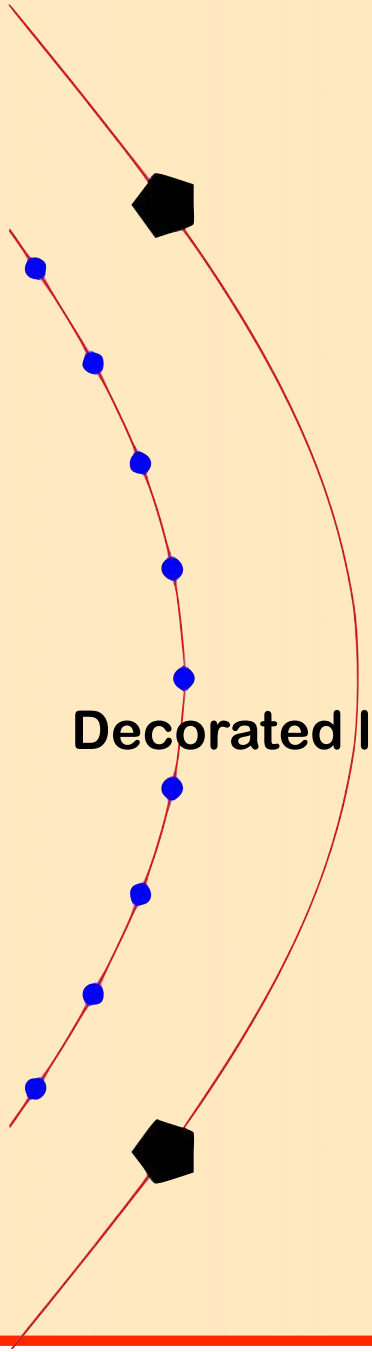
Line Decorations & Endings

Bezier interpolation



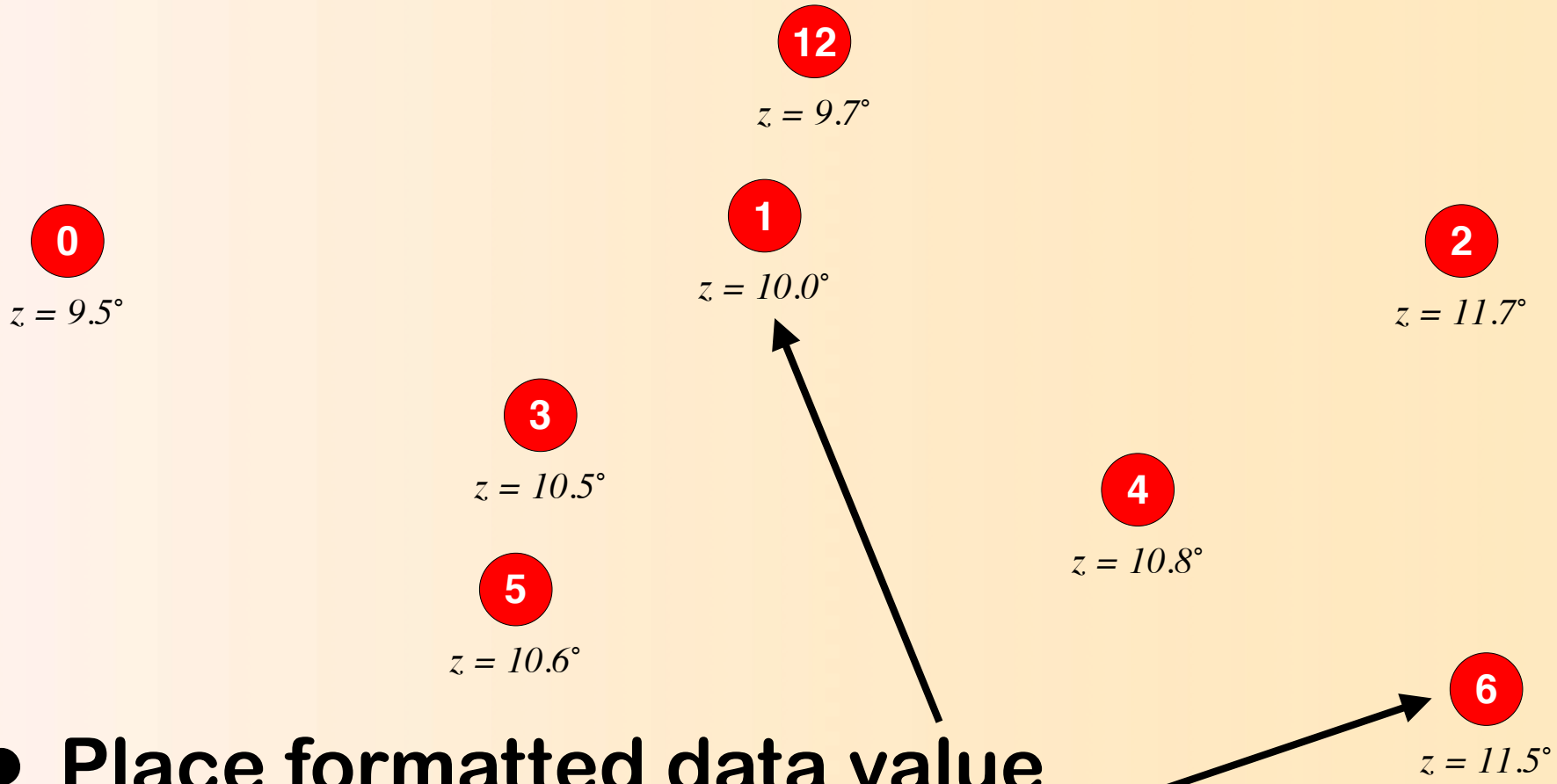






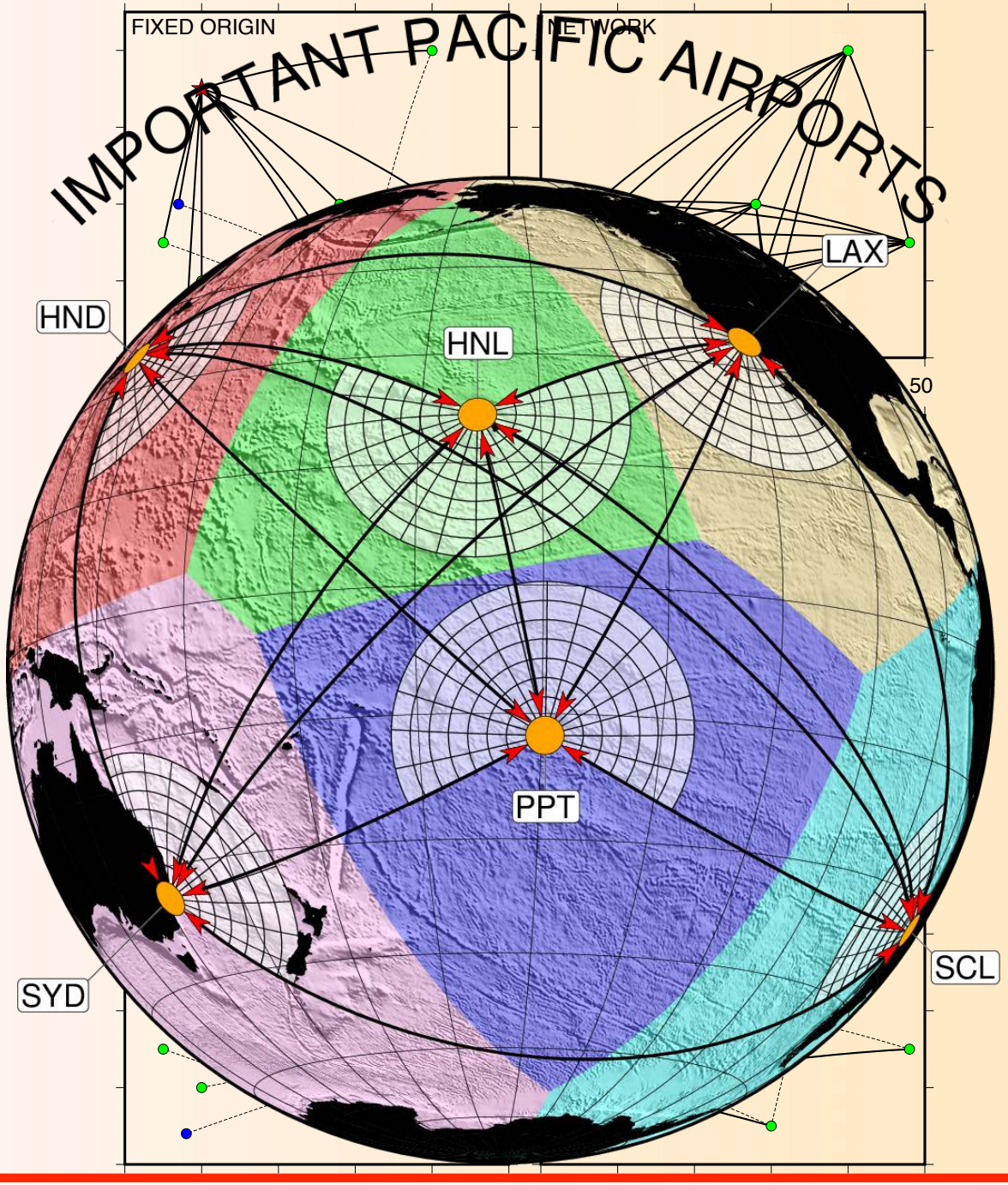
GMT 5.3 Release

Auto-Labeling



- Place formatted data value
- Place record number

GMT 5.3 Release



Line segmentation and networks

Funding Situation

- We have been funded ~continuously since 1993.
  - MGG funding 1993–1995, 1996–1999, 2001–2004, 2005–2010, 2010–2015
  - Always got funded on first try, with excellent reviews, etc.
- EAR/Geoinformatics funding 2013–2017 for collaboration between Wessel, Sandwell and Feigl
  - Top ranked proposal by panel. We took a 10% per-year budget cut but were promised a 1-year supplement, hence effectively a 4-year project
  - Focus on new capabilities for non-marine users:
    - Extend plot capability to geodesy, seismology [meca], geology [ternary, symbology]
    - Develop Python/GMT module
    - Build GMT5SAR
    - Deploy GMT on the Open Science Grid (OSG)

Funding Situation

- Latest MGG proposal 2016–2021
  - Highly-ranked proposal (all excellent ranks)
  - Funding recommended for 5 years
  - Further interactions with NSF lead to initial 2 year MGG funding while NSF plans for a sustainable funding model for the following 3 years
  - Obtained Venture Funds for Software Reuse (2 years) to target GMT/Python module and code hardening
  - UNAVCO's GMT/GMT5SAR Frontiers Activities under the National Geophysical Observatory for Geoscience (NGEO). This is presently most likely candidate for long-term maintenance - but awaits proposal

Development Plans

We proposed seven tasks:

1. Succession Planning
2. PROJ4 Dependency
3. OGR Bridge
4. MGG Focus
5. GSHHG 3
6. Code Hardening
7. PostScriptLight Detachment

Development Plans

Task 1: Succession Planning

The GMT leader is nearing retirement so we need to develop plans for how the baton can be passed. We need to start this now as it will be a long process

Actions:

1. Form a GMT Steering Committee
2. Assemble a GMT Maintainer's Handbook

Development Plans

Task 2: PROJ4 Dependency

GMT originated concurrently with PROJ4 so we developed our own mapping functions with incompatible syntax. Switching to PROJ4 eliminates map code maintenance, provides additional projections and improves interoperability

Actions:

1. Replace GMT's internal map engine with the industry-standard PROJ4 library
2. Provide backwards compatibility for GMT projection syntax

Development Plans

Task 3: OGR Bridge

We now use the GDAL library to read any grid format. Likewise, using OGR library we could read any vector file format (e.g., shapefiles) and maintain proper projection and geometry metadata

Actions:

1. Add OGR i/o as new vector file capability
2. Retain the compatibility with OGR/GMT format

Development Plans

Task 4: MGG Focus

Finalize, polish, and improve documentation for MGG-related tools for track data analysis, cross-overs, geopotential fields and isostasy

Actions:

1. Improve examples and documentation for mgd77, potential, and spotter
2. Merge x2sys generic tools into core

Development Plans

Task 5: GSHHG 3

Auto-derive a new, improved GSHHG from the Open Street Map projects, removing the burden of coastline editing from the GMT developers

Actions:

1. Develop a workflow to extract coastlines from the OSM database
2. Educate GMT users on how to contribute data to OSM

Development Plans

Task 6: Code Hardening

Increase number of test scripts, continue adding OpenMP and LAPACK to optimize calculations

Actions:

1. Develop a process by which we can quickly add more test scripts
2. Work with computational scientist on profiling, OpenMP, etc.

Development Plans

Task 7: PostScriptLight Detachment

Separate the PostScriptLight library (previously PSL or pslib) from GMT and let it become a required dependency. It changes so infrequently that a splitting from GMT reduces maintenance chores

Actions:

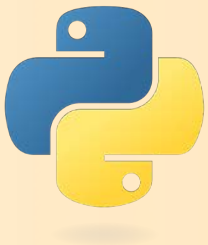
1. Split the PSL code from GMT code in svn and create another repository
2. Push PSL on package managers

Post-Doctoral Position

- The postdoc will help address these key tasks that cross the various GMT grants:

  1. Fast-track the GMT/Python module
  2. Code Hardening, with some assistance from computational scientists at UH
  3. Testing/Calibration

OpenMP



python™

Post-Doctoral Position

- **Search Status:**

1. Advertised across numerous geophysical mailing lists and forums
2. Received over a dozen inquiries, resulting in 10 serious candidates, of which 3 were dismissed for lack of relevant skills
3. Requested letters for remaining 7 candidates, all of which have been received. Dismissed 2 candidates as too weak.
4. Remaining 5 candidates will be interviewed via Skype or in person this Wednesday
5. We hope to select a person next week and have him report to me in Hawaii no later than early next year.

Succession Plan

- There are several aspects that need to be addressed, perhaps to various degrees:

1. Physical hosting and maintenance of code, provide web access to wiki, forums, bug tracker, support, documentation, etc.
2. Bug fixing, updates required due to OS changes, bits/bytes, libraries, etc.
3. Further development of GMT, extensions, new supplements, APIs, etc.

Succession Plan

Hosting Location

- The easiest problem to solve. Some possibilities are
  1. UNAVCO as part of the NGEO, or otherwise
  2. SOEST for historical reasons
  3. NOAA, IRIS, …

Succession Plan

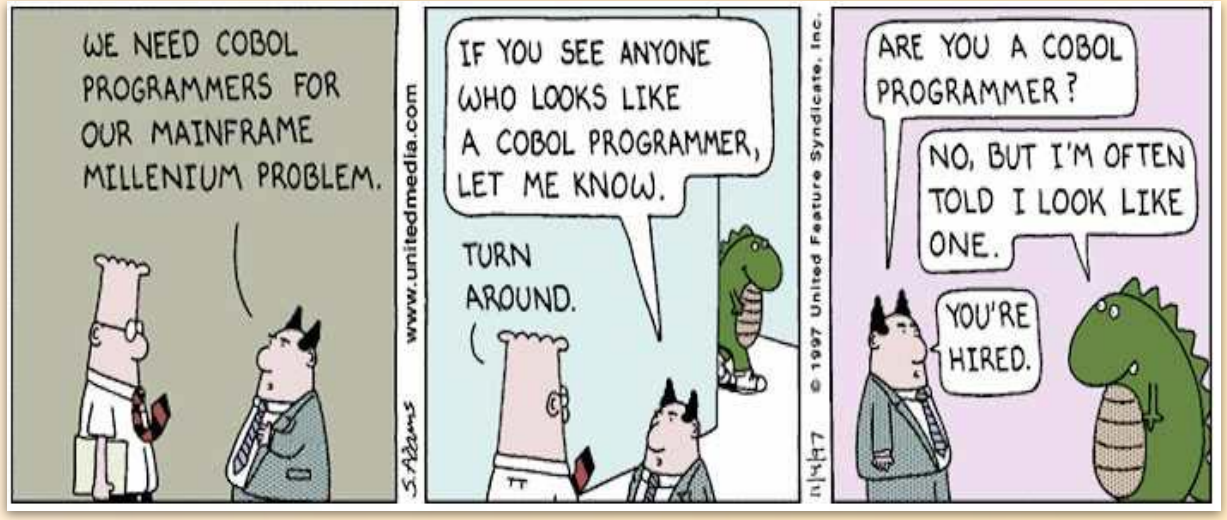
Maintenance

- This probably requires a combination of a salaried employee at the host and leveraging of the user community

1. Tasks are: Maintaining repositories, fixing reported bugs (or let trusted users do so via write-permission), maintain wiki, issue tracker and forums, issue new service releases, correct errors in documentation
2. Cost would be modest as GMT matures if no new features are added

Succession Plan

New Developments



- This would require a new leader with an agenda to step up.
  1. Requires funding for leader or underlings
  2. Need to engage the user community
  3. I am not necessarily an easy act to follow since C programmers are going the way of the Cobol

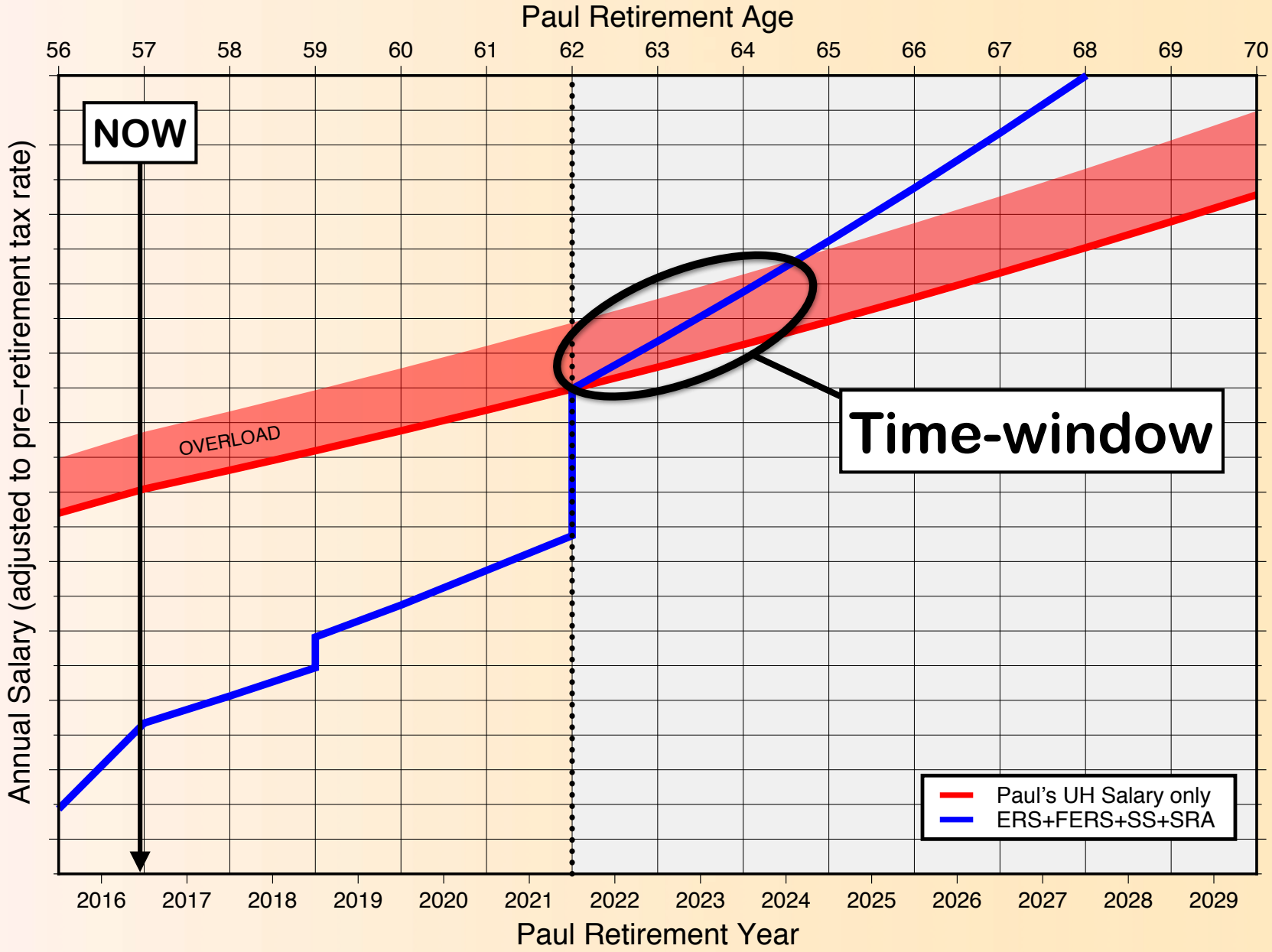
Succession Plan

Time Line

1. Our proposal covers 5 year, possibly 6 with one extension
2. We will know in ~1 year if NGEO is a viable solution, possibly with sub-contract support to UHM for years 3+.
3. Wessel is likely to retire “young” (i.e., to become Emeritus at end of current 5-6 cycle)
4. Also likely to be involved for years as needed

Succession Plan

Paul Wessel Retirement 7/1/2021 Income Planning



Succession Plan

- The floor is open for discussion