Bringing GMT to Python

Leonardo Uieda*

Paul Wessel

leouieda.com/talks/scipy2017.html

thank you

organizers

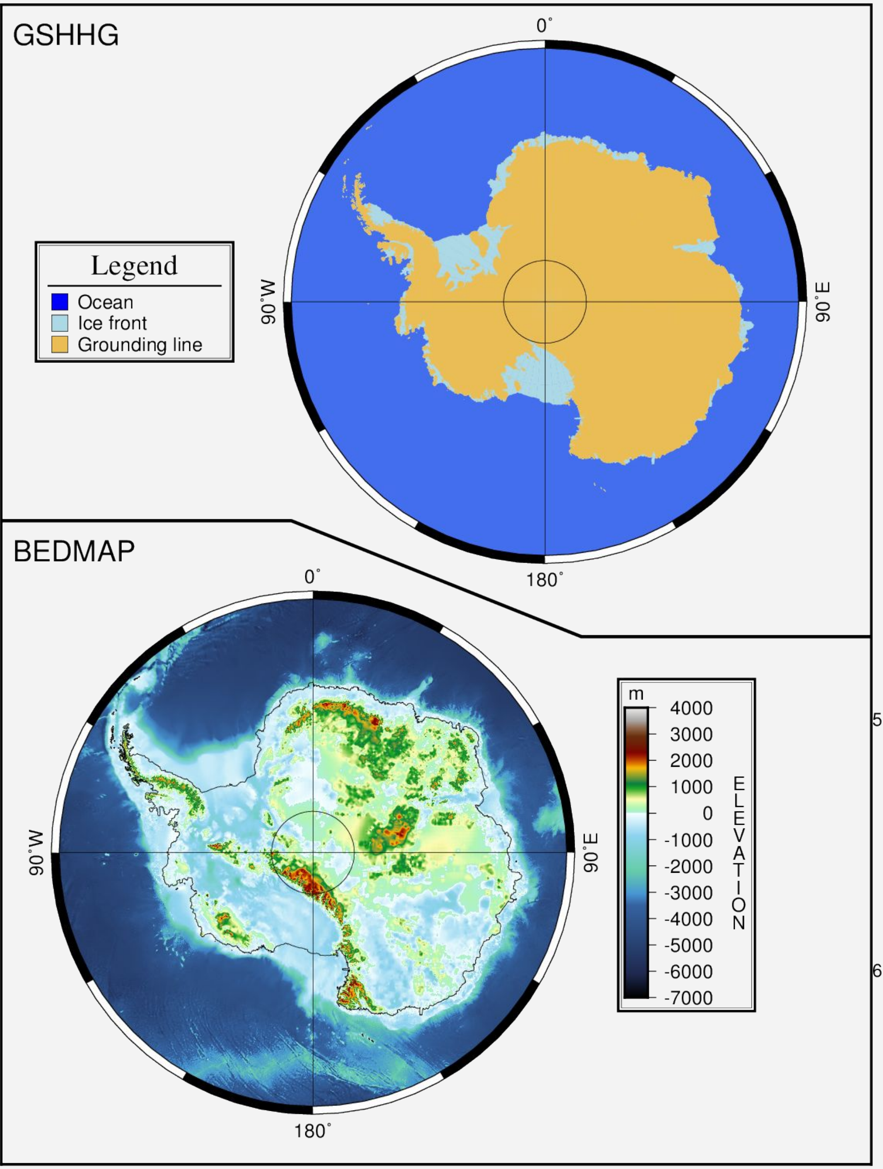
reviewers

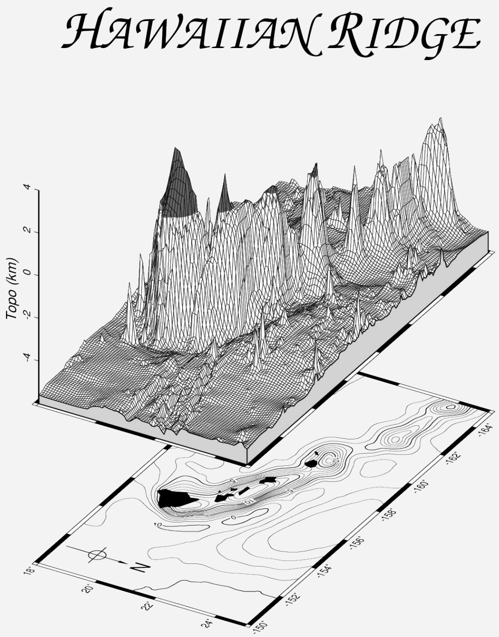
THE GENERIC MAPPING TOOLS

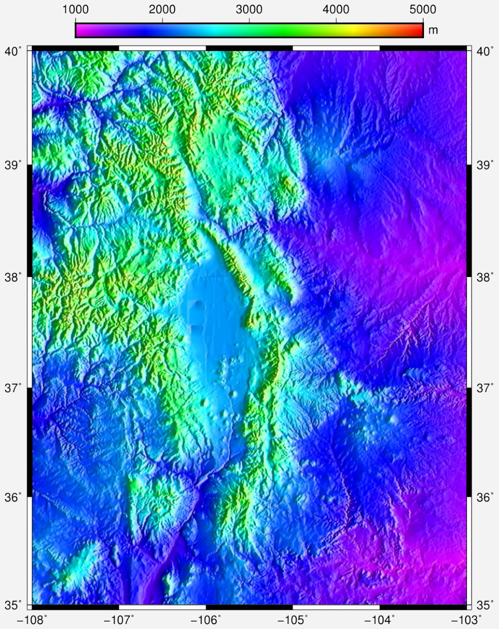
C cmd programs

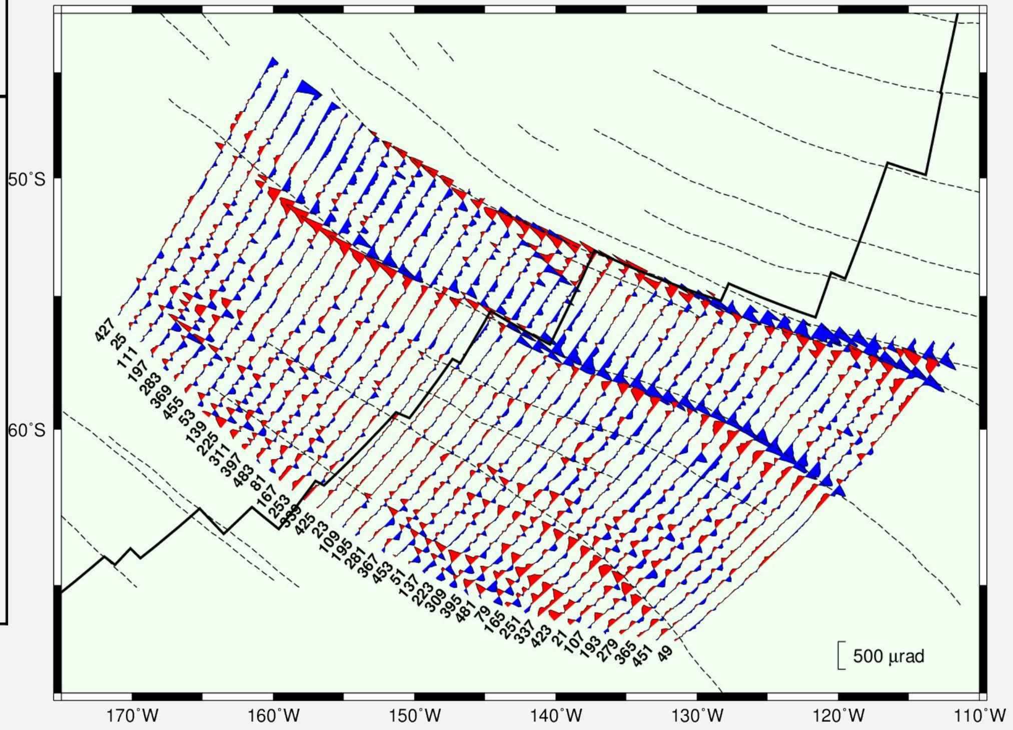
Process spatial data

Math on the sphere


GSHHG
0˚
90˚W
90˚E
180˚
Legend
🟦 Ocean
▢ Ice front
🟧 Grounding line
BEDMAP
0˚
90˚W
90˚E
180˚
m
4000
3000
2000
1000
0
-1000
-2000
-3000
-4000
-5000
-6000
-7000
ELEVATION


HAWAIIAN RIDGE
Topo (km)
4
2
0
-2
-4
N
18˚
20˚
22˚
24˚
-150˚
-152˚
-154˚
-156˚
-158˚
-160˚
-162˚
-164˚


1000
2000
3000
4000
5000
m
40˚
39˚
38˚
37˚
36˚
35˚
−108˚
−107˚
−106˚
−105˚
−104˚
−103˚


50˚S
60˚S
170˚W
160˚W
150˚W
140˚W
130˚W
120˚W
110˚W
427
25
111
197
283
369
455
53
139
225
311
397
483
81
167
253
339
425
23
109
195
281
367
453
51
137
223
309
395
481
79
165
251
337
423
21
107
193
279
365
451
49
500 µrad

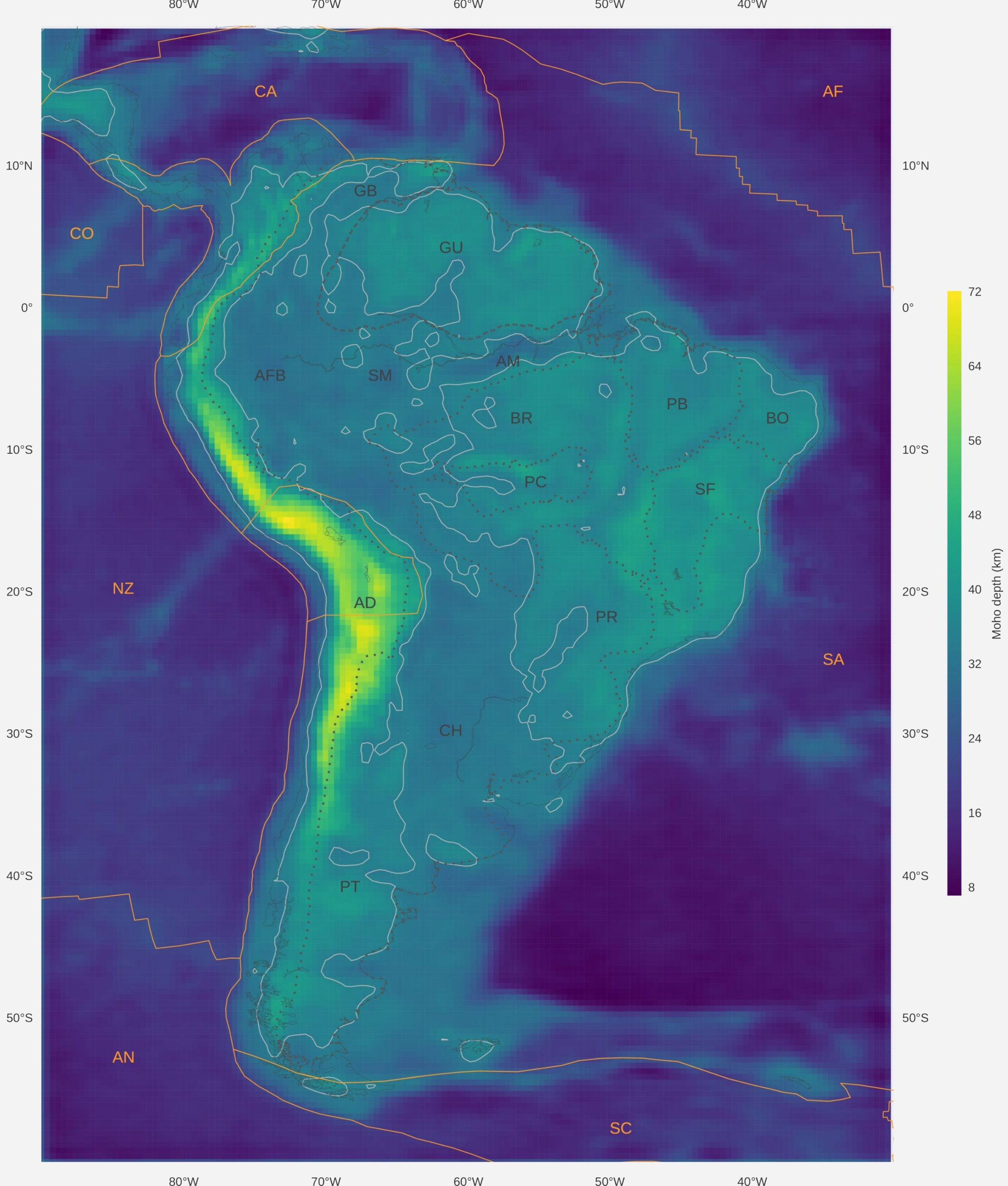
history

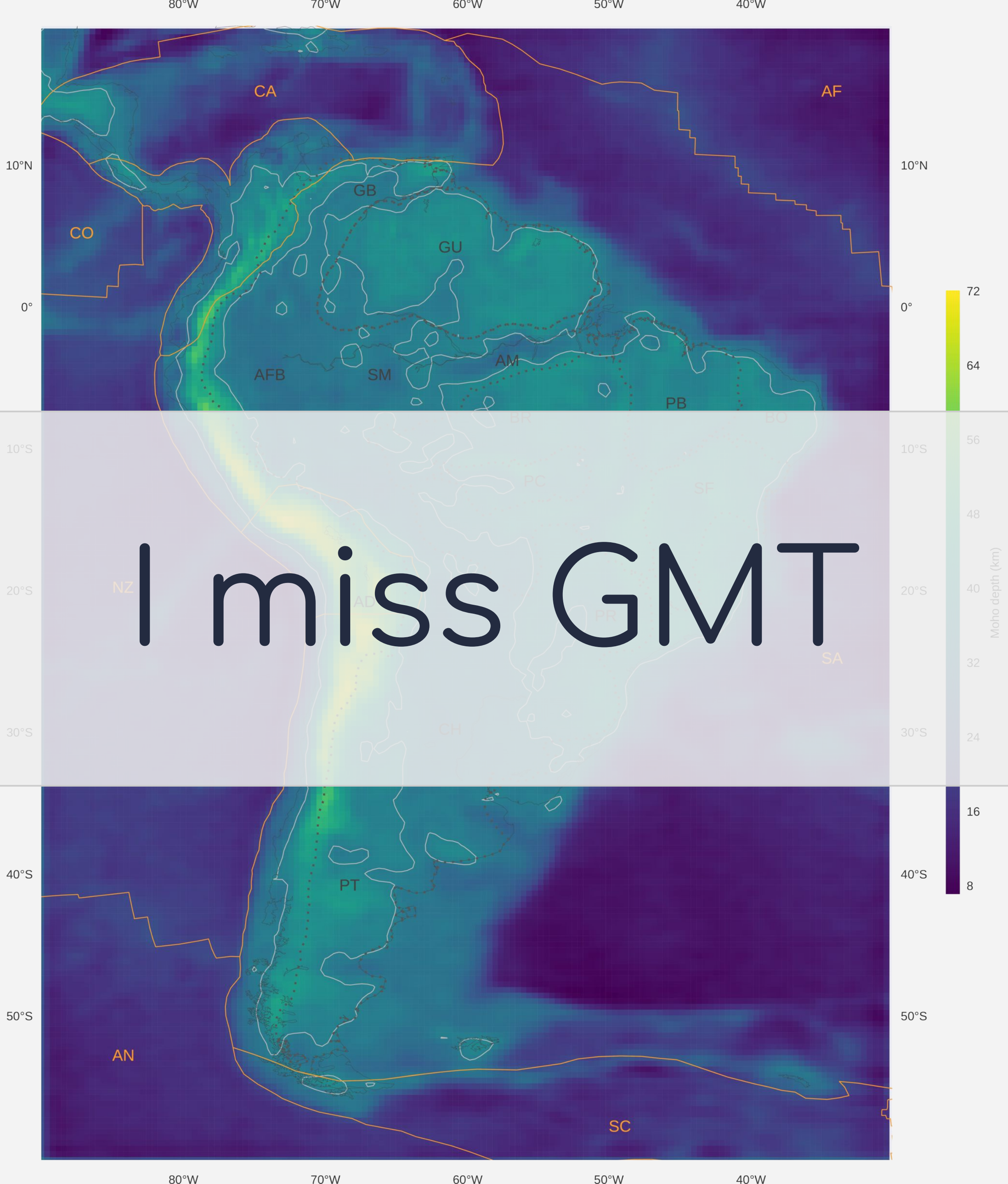
GMT 1.0 (1988)

C API in GMT 5

Matlab and Julia

confession


80°W
70°W
60°W
50°W
40°W
10°N
0°
10°S
20°S
30°S
40°S
50°S
CA
AF
CO
NZ
SA
AN
SC
GB
GU
AM
AFB
SM
BR
PB
BO
PC
SF
AD
PR
CH
PT
72
64
56
48
40
32
24
16
8
Moho depth (km)


80°W
70°W
60°W
50°W
40°W
10°N
0°
10°S
20°S
30°S
40°S
50°S
CA
AF
CO
GB
GU
AM
AFB
SM
PB
BR
BO
PC
SF
NZ
AD
PR
SA
CH
PT
AN
SC
72
64
56
48
40
32
24
16
8
Moho depth (km)

I miss GMT

gmtpy (Sebastian Heimann)

pygmt (Ian Rose)

PyGMT (Magnus Hagdorn)

Official GMT/Python

(NSF funded postdoc)

Official GMT/Python

(NSF funded postdoc)

In Hawaii…

Aloha
WELCOME TO HAWAII

goals

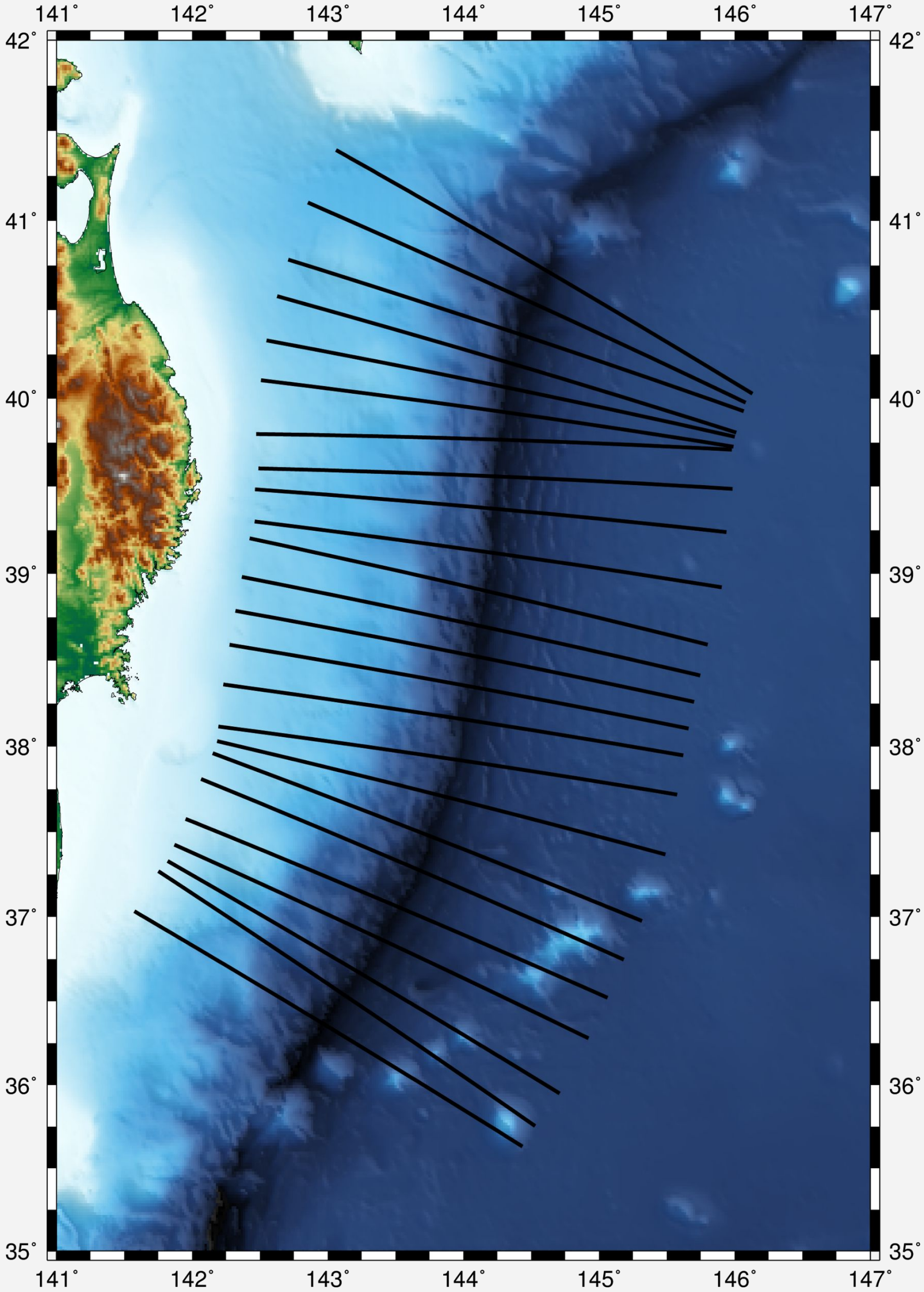
Use the C API

Pythonic

Scipy stack

Modern (py3k + GMT6)

modern mode


141°
142°
143°
144°
145°
146°
147°
42°
41°
40°
39°
38°
37°
36°
35°

```
gmt grdgradient -Nt0.2 -A45 data.nc \
   -Ggrd.nc
gmt makecpt -Cgeo -T-8000/2000 > t.cpt
gmt grdimage -JM6i  -Ct.cpt -Igrd.nc \
   data.nc -P -K > jp.ps
gmt pscoast -Rdata.nc  -Baf -W0.75p \
   -J -Dh -K -O >> jp.ps
gmt psxy -W2p lines.txt -R -J -O \
   >> jp.ps
gmt psconvert jp.ps -TG -A -P
gmt grdgradient -Nt0.2 -A45 data.nc \
  -Ggrd.nc
gmt makecpt -Cgeo -T-8000/2000 > t.cpt
gmt grdimage -JM6i  -Ct.cpt -Igrd.nc \
  data.nc -P -K > jp.ps
gmt pscoast -Rdata.nc  -Baf -W0.75p \
  -J -Dh -K -O >> jp.ps
gmt psxy -W2p lines.txt -R -J -O \
  >> jp.ps
gmt psconvert jp.ps -TG -A -P
gmt grdgradient -Nt0.2 -A45 data.nc \
  -Ggrd.nc
gmt makecpt -Cgeo -T-8000/2000 > t.cpt
gmt grdimage -JM6i  -Ct.cpt -Igrd.nc \
  data.nc -P -K > jp.ps
gmt pscoast -Rdata.nc  -Baf -W0.75p \
  -J -Dh -K -O >> jp.ps
gmt psxy -W2p lines.txt -R -J -O \
  >> jp.ps
gmt psconvert jp.ps -TG -A -P
gmt begin
  gmt figure japan-trench png
  gmt grdgradient -Nt0.2 -A45 data.nc \
    -Ggrd.nc
  gmt makecpt -Cgeo -T-8000/2000 > t.cpt
  gmt grdimage -JM6i -Ct.cpt -Igrd.nc \
    data.nc -P
  gmt pscoast -B -W0.75p
  gmt psxy -W2p lines.txt
gmt end
gmt begin
  gmt figure japan-trench png
  gmt grdgradient -Nt0.2 -A45 data.nc \
    -Ggrd.nc
  gmt makecpt -Cgeo -T-8000/2000 > t.cpt
  gmt grdimage -JM6i -Ct.cpt -Igrd.nc \
    data.nc -P
  gmt pscoast -B -W0.75p
  gmt psxy -W2p lines.txt
gmt end
gmt begin
  gmt figure japan-trench png
  gmt grdgradient -Nt0.2 -A45 data.nc \
     -Ggrd.nc
  gmt makecpt -Cgeo -T-8000/2000 > t.cpt
  gmt grdimage -JM6i -Ct.cpt -Igrd.nc \
     data.nc -P ~~> jp.ps~~
  gmt pscoast -B -W0.75p ~~>> jp.ps~~
  gmt psxy -W2p lines.txt ~~>> jp.ps~~
gmt end
gmt begin
  gmt figure japan-trench png
  gmt grdgradient -Nt0.2 -A45 data.nc \
     -Ggrd.nc
  gmt makecpt -Cgeo -T-8000/2000 > t.cpt
  gmt grdimage -JM6i -Ct.cpt -Igrd.nc \
     data.nc -P ~~-K > jp.ps~~
  gmt pscoast <…> -B -W0.75p ~~>> jp.ps~~
  gmt psxy -W2p lines.txt <…> ~~>> jp.ps~~
  ~~gmt psconvert <...>~~
gmt end
```

demo

what

where

how

pure Python

+

ctypes

github.com/
GenericMappingTools/
gmt-python

```
gmt/
    clib/
    ps_modules.py
    session_management.py
    extra_modules.py
    generators.py
    utils.py
    tests/
```

```python
@fmt_docstring
@use_alias(R='region', J='projection', ...)
@kwargs_to_strings(R='sequence')
def psbasemap(**kwargs):
    """
    Produce a basemap for the figure.
    {gmt_module_docs}
    {aliases}
    Parameters
    ----------
    {J}
    D : str
          ...
    """
    assert …
    call_module('psbasemap',
                build_arg_string(kwargs))
```

```
@fmt_docstring
@use_alias(R='region', J='projection', ...)
@kwargs_to_strings(R='sequence')
def psbasemap(**kwargs):
    """
    Produce a basemap for the figure.
    {gmt_module_docs}
    {aliases}
    Parameters
    ----------
    {J}
    D : str
          ...
    """
    assert …
    call_module('psbasemap',
                build_arg_string(kwargs))
```

```
@fmt_docstring
@use_alias(R='region', J='projection', ...)
@kwargs_to_strings(R='sequence')
def psbasemap(**kwargs):
   """
   Produce a basemap for the figure.
   {gmt_module_docs}
   {aliases}
   Parameters
   ----------
   {J}
   D : str
         ...
   """
   assert …
   call_module('psbasemap',
               build_arg_string(kwargs))
```

```
@fmt_docstring
@use_alias(R='region', J='projection', ...)
@kwargs_to_strings(R='sequence')
def psbasemap(**kwargs):
   """
   Produce a basemap for the figure.
   {gmt_module_docs}
   {aliases}
   Parameters
   ----------
   {J}
   D : str
         ...
   """
   assert …
   call_module('psbasemap',
               build_arg_string(kwargs))
```

```
@fmt_docstring
@use_alias(R='region', J='projection', ...)
@kwargs_to_strings(R='sequence')
def psbasemap(**kwargs):
   """
   Produce a basemap for the figure.
   {gmt_module_docs}
   {aliases}
   Parameters
   ----------
   {J}
   D : str
         ...
   """
   assert …
   call_module('psbasemap',
               build_arg_string(kwargs))
```

```python
@fmt_docstring
@use_alias(R='region', J='projection', ...)
@kwargs_to_strings(R='sequence')
def psbasemap(**kwargs):
    """
    Produce a basemap for the figure.
    {gmt_module_docs}
    {aliases}
    Parameters
    ----------
    {J}
    D : str
          ...
    """
    assert …
    call_module('psbasemap',
                build_arg_string(kwargs))
```

tests

```python
@figure_comparison_test
def test_psbasemap():
    figure()
    psbasemap(region=[0, 360, -90, 90],
              projection='W7i',
              frame=True,
              portrait=True)
```

hacked pytest-mpl

```
@figure_comparison_test
def test_psbasemap():
    figure()
    psbasemap(region=[0, 360, -90, 90],
              projection='W7i',
              frame=True,
              portrait=True)
```

GMT 6.0 trunk

(conda-forge package)

the plan

data exchange

```
my_data = np.loadtxt(‘...’)
gmt.figure()
gmt.psxy(data=my_data, ...)
```

aliases

```
gmt.pscoast(R='-30/30/-40/40',
            J='m0.1i',
            I='1/1p,blue',
            N='1/0.25p,-',
            W='0.25p,white', ...)
```

gallery

(hack sphinx-gallery?)

Python exclusives

new APIs
utility functions
notebook integration

help!

Wrap a module

1. Copy docs
2. Define aliases
3. Copy tests

conda package

github.com/conda-forge/gmt-feedstock

What do you want?

(~~backward compatibility~~)

Slides and contact

leouieda.com

Code

github.com/
GenericMappingTools