GMT 参考手册

GMT 中文社区

2016 年 08 月 31 日

目录

插图

表格

欢迎来到 GMT(Generic Mapping Tools) 的世界。

本项目是 GMT 中文社区维护的 GMT 中文手册，既可以作为入门读物，也可以作为日常参考。希望通过阅读本手册，能够让用户尽快掌握 GMT 的用法。

相关链接：

- 社区主页：http://gmt-china.org
- 项目主页：http://docs.gmt-china.org
- 项目源码：https://github.com/gmt-china/GMT_Docs
- GMT 官方主页：http://gmt.soest.hawaii.edu
- GMT 官方手册：http://gmt.soest.hawaii.edu/doc/latest/index.html

本手册的章节结构如下：

- 前言：介绍本文档的起源、文档约定以及阅读方式
- 简介：简单介绍 GMT 的基本信息
- 基础知识：介绍 GMT 中的基础知识
- 进阶知识：介绍 GMT 中的进阶知识
- 标准选项：介绍 GMT 所有模块的通用选项
- 配置 *GMT*：介绍 GMT 中的配置参数
- 投影方式：介绍 GMT 中的几十种投影方式
- 附录：与 GMT 相关但很少用到的一些知识
- *FAQ*：常见问题及解决方案
- 索引：关键词索引，方便通过关键字查找

除了本手册之外，还可以阅读 GMT 中文社区的其他资源：

- GMT 模块手册：http://modules.gmt-china.org
- GMT 示例手册：http://examples.gmt-china.org

第一章　前言

1.1 文档说明

本项目由 seisman 于 2014 年发起，并将文档源码在 GitHub 上开源。

在 2014 年，网络上能够找到的几本 GMT 中文文档，大多基于 GMT4 甚至 GMT3。这几本文档存在一些明显的缺陷：

1. 排版质量一般，大多是在 Word 里写好然后转换成 PDF 的
2. 缺乏维护，几乎都是写好发布之后就没有再更新
3. 不完整，很多有价值的内容都没有被包含在文档中
4. 不严谨，很多文档声称是基于 GMT4，但实际上某些命令却在使用 GMT3 的语法
5. 无法适配最新版的 GMT，某些时候对于新用户而言是一种障碍
6. 重复劳动，不同的作者，花费了大量的时间和精力，翻译整理出了几份相似的文档，这其中很多工作都是在重复劳动

因而，我于 2014 年开始写针对 GMT5 的中文文档。新的 GMT 中文文档具有如下特点：

1. 开源：源代码用 rST 语言写成，并在 GitHub 上开源
2. 高质量排版：文档有网页版和 PDF 版，保证高质量的排版
3. 完整性：尽可能覆盖 GMT 的方方面面，成为一个完整的参考手册
4. 协同合作：依托强大的 GitHub 和 Git，使得多人合作共同编辑同一份文档变得很简单
5. 持续维护：不仅适配 GMT 的新版本，而且不断修正文档中存在的错误
6. 严谨性：尽可能保证所有命令均通过 GMT 最新版本的测试

经过两年努力，文档的整体结构已经大体完成。我希望文档今后能够持续更新，不断完善，不管是接下来的 GMT 5.2.2、5.3.0 还是未来的 GMT 6.0 甚至 7.0，都能随着 GMT 版本的升级而不断更新，但这些已非我一人之力所能完成。故而成立 GMT 中文社区，将维护工作转移至社区，希望能够有更多的志愿者加入到文档的维护工作中来。

本文档会针对每一个 GMT 版本发布相应的文档。目前最新的版本是 GMT 5.2.1，读者可以从社区主页的 下载页面 下载网页版和 PDF 版的文档供离线阅读。

本文档会不断更新，项目主页 中总是展示最新最完善的网页版版本。

1.2 建议的阅读方式

磨刀不误砍柴工。GMT 的知识点虽然多，但最常用的却只有一点。把基础的东西掌握好，就可以解决日常遇到的大多数问题。

学习 GMT 的最好方式就是多尝试，因而本文档在介绍各个知识点时给出了尽量多的示例。读者应一边阅读正文一边运行相应示例，尝试修改示例中的参数并查看其效果。请注意，运行示例程序时最好不要复制而是自己手敲代码，这样更容易及早发现自己未来会犯的错误。本文档中的示例都经过测试，但不排除偶尔有示例出错的情况。遇到这种情况，请自行排查再提交 Issue。

对于本文档的每一章而言，建议的阅读方式时：

1. 简介 ： 了解 GMT 的相关背景
2. 基础知识：仔细阅读，多多理解，尽量记住，至少要有印象
3. 进阶知识：粗略浏览，对每节的大概内容要有印象，表数据 一节比较重要
4. 标准选项：非常重要，必须掌握选项 `-B` 、`-J` 、`-K` 、`-O` 、`-P` 、`-R` 、`-V` 、`-X` 和 `-Y` 的用法，其他选项的用法有印象即可
5. 配置 *GMT*：仔细阅读第一节的内容，并多看这一节的三张图。其他节的内容稍微浏览一遍即可
6. 投影方式：掌握线性投影 `-JX` 的用法，其他投影方式看看图即可，需要的时候再重新看图
7. 附录：可以阅读一下*GMT 风格指南* 一节

1.3 词汇表

在 GMT 中有不少与绘图有关的专业词汇。这些英文词汇没有固定的中文翻译，因而这里统一给出本文所使用的中文翻译，并在图中指出这次词汇的含义。

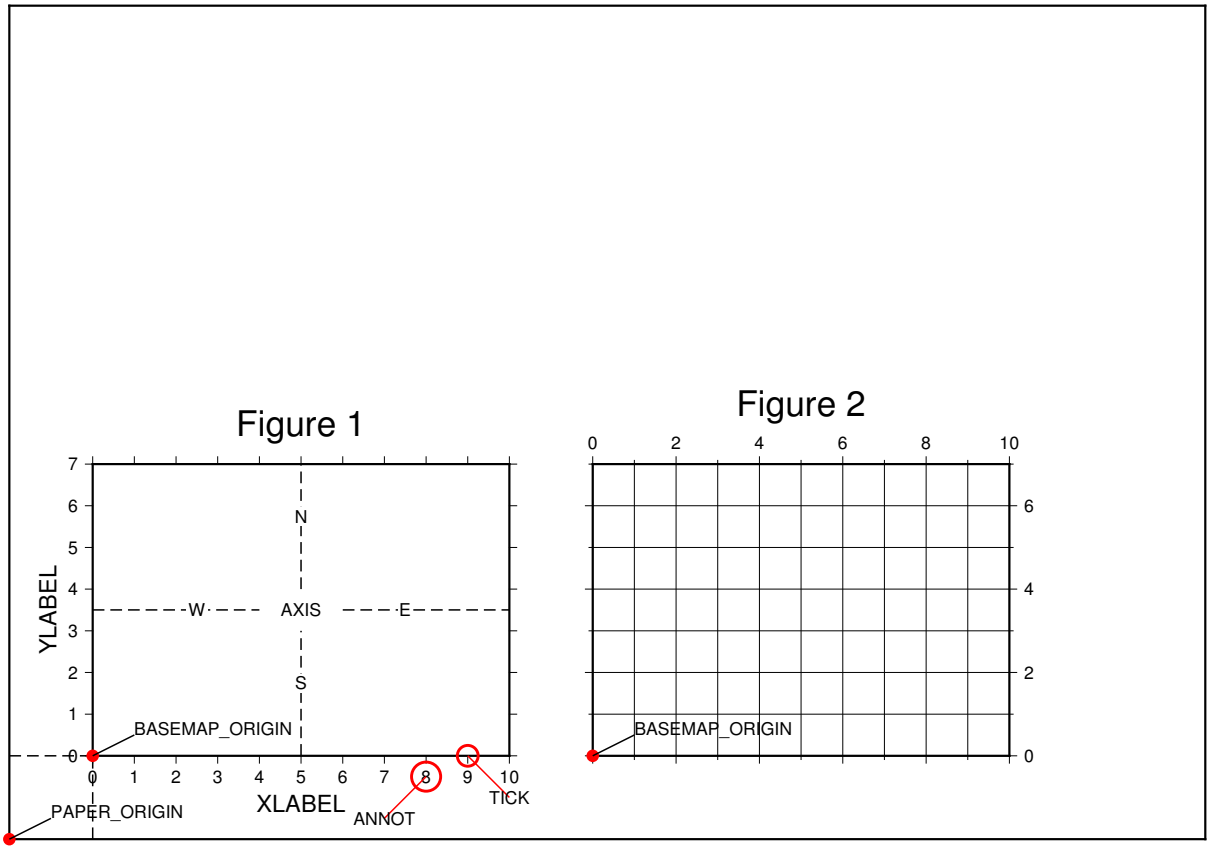


图 1.1: GMT 绘图元素

纸张原点（PAPER_ORIGIN） 一张纸，无论横着放还是竖着放，纸张的原点都位于纸张的最左下角。图中最外部的黑色边框代表纸张的边界。左下角的红点即纸张原点的位置。

底图（BASEMAP） 底图是一个抽象的概念，可以认为图 1 中黑色边框就是一个底图。

底图原点（BASEMAP_ORIGIN） 底图原点位于每一张底图的左下角，如图中红点所示。通过 `-X` 和 `-Y` 选项可以控制底图原点相对于纸张原点的位置。

坐标轴（AXIS） 每个底图都有四条边，分别称为 W、E、S、N。四条边可以显示也可以不显示。

标签（LABEL） 标签用于表明每个坐标轴的含义，即图中的 XLABEL 和 YLABEL。

标题（TITLE） 每张底图都可以有一个标题，标题默认位于底图的上方，即图中 Figure 1 所在位置。

标注（ANNOT） 如图中所示，即每个坐标轴上的数字

刻度（TICK） 如图所示，刻度指每个坐标轴上的刻度线

网格线（GRID） 如图 2 所示。

子图 在一张纸上可以绘制多个底图，每个底图称为一个子图，通过使用 `-X` 和 `-Y` 选项可以调

整每个子图相对于纸张原点的位置。图中图 1 和图 2 是整张图的子图。

第二章　简介

这一章将介绍 GMT 的一些背景知识。

2.1 GMT 简介

2.1.1 GMT 是什么

GMT，全称 Generic Mapping Tools，中文一般译为“通用制图工具”。GMT 不仅可以用于绘制不同类型的地图，还可以绘制常见的笛卡尔坐标轴（线性轴、对数轴和指数轴）。除此之外，GMT 还有一些数据处理和分析的功能，比如多项式拟合、数据滤波、线性回归分析等。

2.1.2 GMT 的历史

- 1988 年，Paul Wessel 和 Walter H.F. Smith 开发了 GMT 的最原始版本 GMT 1.0
- 1991 年 8 月 10 日，GMT 2.0 发布
- 1998 年 11 月 8 日，GMT 3.x 的第一个正式版发布
- 2005 年 10 月 1 日，GMT 4.x 的第一个正式版发布；目前最新版本 GMT 4.5.14 发布于 2015-11-01
- 2013 年 11 月 5 日，GMT 5.x 的第一个正式版发布；目前最新版本 GMT 5.2.1 发布于 2015-11-12

2.1.3 GMT 的特点

为什么选择 GMT 作为绘图软件呢？因为 GMT 有如下特点：

1. 开源免费

   GMT 是开源软件，任何人均可免费获取其源代码，并根据自己的需求修改、再发布。

2. 跨平台

   GMT 的源码采用了高度可移植的 ANSI C 语言，其完全兼容于 POSIX 标准，几乎不需修改即可运行在大多数类 UNIX 系统上。GMT 官方网站不仅提供了软件源码，还提供了供 Windows 和 Mac OS 使用的软件安装包。各大 Linux 发行版中也提供了预编译的二进制版本。

3. 模块化

   GMT 遵循 UNIX 的模块化设计思想，将 GMT 的绘图及数据处理功能划分到不同的模块中。这样的模块化设计有很多优点：

   - 只需要少量的模块
   - 各个模块之间相互独立且代码量少，易于更新和维护
   - 每一步均独立于独立于之前的步骤以及具体的数据类型，因而可以用于不同的应用中
   - 可以在 shell 脚本中调用一系列程序，或通过管道连接起来，进而绘制复杂图件

4. 高精度矢量图

   GMT 绘制得到的图件为 PS 格式，即 PostScript，是一种页面描述语言。PS 格式是矢量图片格式，可以任意放大缩小而不失真。GMT 充分利用 PostScript 语言的特性，可以生成高质量的矢量图件，并可以很容易地转换为其他图片格式。

2.1.4 GMT 版本号

GMT 的版本号的格式为:

```
major.minor.patch
```

其中 `major` 为主版本号，`minor` 为次版本号，`patch` 为补丁版本号。

一般而言，版本号的更新规则如下：

- 当有极大的更新，会增加主版本号 `major` ，因而 `major` 不同的两个版本在语法和功能上有很大的差异
- 当有较大的更新，比如个别命令的语法有变化，会更新次版本号 `minor`
- 若更新主要是修复错误，则会增加 `patch` 的版本号

因而，`GMT 5.x.x` 与 `GMT 4.x.x` 之间有很大差异，两个版本的语法是不完全兼容的，切勿混在一起使用。`GMT 4.5.x` 和 `GMT 4.4.x` ，部分命令的语法和用法可能有一点区别。而 `GMT 4.5.13` 相对于 `GMT 4.5.12` ，主要是修复了一些 BUG。

2.1.5 GMT4 vs GMT5

GMT 目前在同时维护 GMT4 和 GMT5 两个大版本。用户是选择 GMT4 还是 GMT5 呢?

GMT4 的优缺点:

- 已发布十年有余，功能相对成熟
- 仅修复 Bug，不再增加新功能，命令语法相对稳定
- 网络上的中文教程及示例多采用 GMT4 语法，新手学习起来更方便

GMT5 的优缺点:

- 相对于 GMT4 有很多改进，命令语法更统一，选项的设计更加合理
- 相对于 GMT4 加入了更多的新功能
- GMT5 是 GMT 的未来
- 由于重写了大量代码，因而可能有较多的 Bug
- 目前关于 GMT5 的中文教程太少

综上，对于用户的建议是:

- 新用户推荐学习 GMT5，毕竟 GMT5 是 GMT 的未来，更多新用户的加入也可以更好地促进 GMT 的发展
- 实验室可能有前人留下的 GMT4 的代码，若历史负担太重，建议使用 GMT4
- 若自己的大型项目中使用了 GMT 绘图，建议使用 GMT4，因为 GMT4 相对稳定，不至于因为 GMT 小版本的不兼容而影响到自己的项目

PS: GMT4 和 GMT5 可在系统中共存，因而完全可以同时安装两个版本，旧脚本不必修改，继续使用 GMT4，新脚本则使用 GMT5 语法。

2.1.6 跨平台特性

GMT 是跨平台的，可以运行在 Linux 及 Windows 下，当然也可以运行在 Mac 下，这里姑且认为 Mac 跟 Linux 是同一个东西。

推荐在 Linux 下使用 GMT，原因如下:

- GMT 是在 Linux 下开发再移植到 Windows 下的。因而，Windows 版本的 GMT 相对来说有更多的 bug
- Linux 自带了众多数据处理工具: `gawk` 、`cut` 、`paste` 等

- Windows 下的命令行及 bat 太难用，Linux 下的命令行和 Bash 脚本相对来说更易用

PS：Windows10 即将自带 Bash，未来有可能可以在 Windows 下写 Bash 脚本，值得期待。

2.1.7 GMT 替代品

在绘制地图方面，还有一些软件也可以实现类似的功能，可以作为 GMT 的替代品。

1. Matplotlib Basemap Toolkit for Python
2. M_Map for Matlab
3. ggmap for R

2.2 安装 GMT

这一节介绍如何在 Linux/Windows/Mac OS 下安装 GMT 5.2.1。

2.2.1 Linux

大多数 Linux 发行版的软件源中都有 GMT，因而可以通过系统自带的软件包管理器来安装。但通常系统软件源里自带的 GMT 都比较老，所以在 Linux 还是建议自己手动编译安装。

编译过程有些复杂，一般用户可以尝试使用为 CentOS 和 Ubuntu 用户准备的 快速安装脚本 。其他发行版的用户也可以测试一下。

2.2.1.1 解决依赖关系

GMT 主要依赖于 cmake(>=2.8.5)、fftw(>=3.3)、glib2(>=2.32)、netCDF、ghostscript 等。

对于 Ubuntu/Debian:

```
sudo apt-get update
sudo apt-get install gcc g++ cmake make libc6
sudo apt-get install ghostscript libnetcdf-dev
sudo apt-get install libgdal-dev python-gdal
sudo apt-get install liblapack3 libglib2.0-dev
sudo apt-get install libpcre3-dev libfftw3-dev
```

对于 CentOS/RHEL/Fedora:

```
sudo yum install gcc gcc-c++ cmake make glibc
sudo yum install ghostscript netcdf-devel
sudo yum install glib2-devel
sudo yum install gdal-devel gdal-python
sudo yum install lapack64-devel lapack-devel
sudo yum install pcre-devel fftw-devel
```

2.2.1.2 下载

Linux 安装 GMT 需要下载三个文件（这里提供的国内下载源）：

1. GMT 源码：http://mirrors.ustc.edu.cn/gmt/gmt-5.2.1-src.tar.gz
2. 全球海岸线数据 GSHHG：http://mirrors.ustc.edu.cn/gmt/gshhg-gmt-2.3.5.tar.gz
3. 全球数字图表 DCW：http://mirrors.ustc.edu.cn/gmt/dcw-gmt-1.1.2.tar.gz

也可以到社区主页的 下载页面 下载对应的文件。

2.2.1.3 安装 GMT

将下载的三个压缩文件放在同一个目录里，按照如下步骤进行安装：

```
# 解压三个压缩文件
$ tar -xvf gmt-5.2.1-src.tar.gz
$ tar -xvf gshhg-gmt-2.3.5.tar.gz
$ tar -xvf dcw-gmt-1.1.2.tar.gz

# 将 gshhg 和 dcw 数据复制到 gmt 的 share 目录下
$ mv gshhg-gmt-2.3.5 gmt-5.2.1/share/gshhg
$ mv dcw-gmt-1.1.2 gmt-5.2.1/share/dcw-gmt

# 切换到 gmt 源码目录下
$ cd gmt-5.2.1

# 新建用户配置文件
$ gedit cmake/ConfigUser.cmake
```

向 cmake/ConfigUser.cmake 文件中加入如下语句:

```
set (CMAKE_INSTALL_PREFIX "/opt/GMT-5.2.1")
set (GMT_INSTALL_MODULE_LINKS FALSE)
set (COPY_GSHHG TRUE)
```

```
set (COPY_DCW TRUE)
set (GMT_USE_THREADS TRUE)
```

- CMAKE_INSTALL_PREFIX 设置 GMT 的安装路径，可以修改为其他路径
- GMT_INSTALL_MODULE_LINKS 为 FALSE，表明不在 GMT 的 bin 目录下建立命令的软链接，也可设置为 TRUE
- COPY_GSHHG 为 TRUE 会将 GSHHG 数据复制到 GMT/share/coast 下
- COPY_DCW 为 TRUE 会将 DCW 数据复制到 GMT/share/dcw 下
- GMT_USE_THREADS 表示是否开启某些模块的并行功能

继续执行如下命令以检查 GMT 的依赖关系:

```
$ mkdir build
$ cd build/
$ cmake ..
```

cmake .. 会检查 GMT 对软件的依赖关系，我的检查结果如下:

```
*  Options:
*  Found GSHHG database       : /home/user/GMT/gmt-5.2.1/share/gshhg (2.3.5)
*  Found DCW-GMT database     : /home/user/GMT/gmt-5.2.1/share/dcw-gmt
*  NetCDF library             : /usr/lib64/libnetcdf.so
*  NetCDF include dir         : /usr/include
*  GDAL library               : /usr/lib64/libgdal.so
*  GDAL include dir           : /usr/include/gdal
*  FFTW library               : /usr/lib64/libfftw3f.so
*  FFTW include dir           : /usr/include
*  Accelerate Framework       :
*  Regex support              : PCRE (/usr/lib64/libpcre.so)
*  ZLIB library               : /usr/lib64/libz.so
*  ZLIB include dir           : /usr/include
*  LAPACK library             : yes
*  License restriction        : no
*  Triangulation method       : Shewchuk
*  OpenMP support             : enabled
*  GLIB GTHREAD support       : enabled
*  PTHREAD support            : enabled
*  Build mode                 : shared
*  Build GMT core             : always [libgmt.so]
*  Build PSL library          : always [libpostscriptlight.so]
*  Build GMT supplements      : yes [supplements.so]
```

```
*  Build GMT Developer        : yes
*  Build proto supplements    : none
*
*  Locations:
*  Installing GMT in          : /opt/GMT-5.2.1
*  GMT_DATADIR                : /opt/GMT-5.2.1/share
*  GMT_DOCDIR                 : /opt/GMT-5.2.1/share/doc
*  GMT_MANDIR                 : /opt/GMT-5.2.1/share/man
-- Configuring done
-- Generating done
```

正常情况下的检查结果应该与上面给出的类似。若出现问题，则需要检查之前的步骤是否有误，检查完毕后重新执行 `cmake ..` ，直到出现类似的检查结果。检查完毕后，开始编译和安装:

```
$ make
$ sudo make install
```

2.2.1.4 修改环境变量

修改环境变量并使其生效:

```
$ echo 'export GMT5HOME=/opt/GMT-5.2.1' >> ~/.bashrc
$ echo 'export PATH=${GMT5HOME}/bin:$PATH' >> ~/.bashrc
$ echo 'export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${GMT5HOME}/lib64' >> ~/.bashrc
$ exec $SHELL -l
```

2.2.1.5 测试是否安装成功

在终端键入 `gmt` ，若出现如下输出，则安装成功:

```
$ gmt --version
5.2.1
```

2.2.2 Windows

GMT 为 Windows 用户提供了安装包，可以直接安装使用。Windows 下需要按照 GMT、ghostscript 和 gsview。

1. 安装 GMT

   GMT 官方网站提供的安装包有问题，请到社区主页的 下载页面 下载可以使用的版本。

直接双击安装包即可安装，直接点击下一步，使用默认的选项即可，无须做任何修改。在“选择组件”页面，建议将四个选项都勾选上，然后点击下一步安装完成。

安装完成后，“开始” -> “所有程序” -> “附件” -> “命令提示符”以启动 cmd。在 cmd 窗口中执行:

```
C:\Users\xxxx> gmt --version
5.2.1
```

即表示安装成功。

2. 安装 ghostscript

   到 ghostscript 主页 下载安装包。

   安装的过程没什么可说的，在最后一步，记得勾选 “Generate cidfmap for Windows CJK TrueType fonts”。

3. 安装 gsview

   到 gsview 主页 下载安装包，直接安装即可。

2.2.3 Mac OS

Mac OS 下 GMT 的安装方法有很多，可以直接使用安装包，也可以使用各种软件管理工具。

1. 直接使用 GMT 提供的安装包

   到国内镜像网站下载 安装包 ，也可以到社区主页的 下载页面 下载安装包。

   安装完成后，在桌面会出现 GMT 的图标。点击该图标会启动一个终端，在终端内执行:

```
echo ${PATH%%:*}
```

   并将输出的目录添加到 ~/.bashrc 中:

```
echo 'PATH=${PATH}:/path/to/gmt' >> ~/.bashrc
```

2. 使用 macports 安装:

```
sudo port install gdal +curl +geos +hdf5 +netcdf +fftw3 +openmp
sudo port install gmt5
```

3. 使用 fink 安装:

```
sudo fink install gmt5
```

4. 使用 homebrew 安装:

```
brew update && brew upgrade
brew install gmt
```

2.3 目录树

本文用 `$GMTHOME` 表示 GMT 的安装路径，在 Linux 下一般是 `/usr/local/GMT5` 或 `/opt/GMT5`，在 Windows 下一般是 `C:\programs\gmt5` 。

`$GMTHOME` 中包含了 GMT 的全部文件。下面列出了 `$GMTHOME` 的目录树，并对其中重要的文件及目录做简单解释。GMT 在 Linux 和 Windows 下的目录树稍有不同，这里以 Linux 为准:

```
.
|-- bin
|-- include
|-- lib64
`-- share
```

2.3.1 `bin` 目录

`bin` 目录下包含如下文件:

```
bin
|-- gmt                          # GMT 主程序
|-- isogmt                       # GMT 主程序（隔离模式）
|-- gmt-config                   # GMT 安装信息获取工具
|-- gmt_shell_functions.sh       # bash 辅助脚本
`-- gmtswitch                    # GMT 版本切换工具
```

几点注意事项:

1. Linux 下，除了上面说到的几个可执行文件外，可能还有很多类似 `psxy` 、`grdimage` 这样的文件，其本质上是指向二进制文件 `gmt` 的软链接，这样设计的目的是为了部分兼容 GMT4 的语法，但建议在使用 GMT5 时严格遵循 GMT5 的语法，因而建议删除 `bin` 目录下除上面列出的 5 个可执行文件以外的所有其他文件。

2. Windows 下，除了上面说到的几个可执行文件外，还有一堆其他 exe 文件，以及一些 dll 文件。这些多余的 exe 文件是为了兼容 GMT4 语法而存在的，建议删除。可以将所有文

件按照文件大小排序，然后删除所有大小为 6 KB 的 exe 文件。

3. 上述 5 个可执行文件中，日常需要使用的只有 gmt 这一个，即 gmt 可以认为是 GMT5 中唯一的一个命令。

2.3.2 include 目录

inlcude 目录下有 gmt 子目录，gmt 子目录中包含了 GMT 的头文件:

```
include
`-- gmt
```

仅当自己写程序调用 GMT 函数库时才需要使用这些头文件，因而一般用户不需要关注。

2.3.3 lib 目录

32 位系统下目录名为 lib ，64 位系统下目录名为 lib64 或 lib ，以下统称为 lib 目录。lib 目录下包含了 GMT 的动态库文件:

```
lib
|-- gmt
|   `-- plugins
|       `-- supplements.so          # GMT 扩展函数库
|-- libgmt.so                       # GMT 主函数库
`-- libpostscriptlight.so           # PSL 绘图函数库
```

仅当自己写代码调用 GMT 函数库时才需要使用，因而一般用户不需要关注。

2.3.4 share 目录

share 目录中包含了 GMT 运行所需的众多辅助文件:

```
share
|-- VERSION                       # GMT 版本号
|-- coast                         # 含海岸线数据
|-- conf                          # 含 GMT 的配置文件
|   |-- gmt.conf                      # 全局配置文件
|   |-- gmt_SI.conf                   # 全局配置文件 (SI 单位制)
|   |-- gmt_US.conf                   # 全局配置文件 (US 单位制)
|   |-- gmt_cpt.conf                  # 全局 CPT 文件列表
|   |-- gmt_custom_media.conf         # 自定义纸张尺寸
|   |-- gmt_custom_symbols.conf       # 自定义特殊符号
```

```
|   |-- gmtlogo_letters.txt          # gmtlogo 使用的数据文件
|   `-- gmtlogo_title.txt            # gmtlogo 使用的数据文件
|-- cpt                            # 含全局 CPT 文件
|-- custom                         # 自定义特殊符号的 def 文件
|-- dbase                          # 含 grdraster 命令所使用的数据库（默认无数据）
|   `-- grdraster.info               # grdraster 命令的配置文件
|-- dcw                            # 含国界、省界数据
|-- doc                            # 官方文档
|   |-- examples                     # 官方示例
|   |-- html                         # 官方文档网页版
|   |-- pdf                          # 官方文档 PDF 版
|   |-- supplements                  # 补充包的说明文档
|   `-- tutorial                     # GMT 教程中所使用的数据
|-- localization                   # 不同语言的时间定义
|-- man                            # man 格式文档
|-- postscriptlight                # PS 所需要的文件
|   |-- PSL_custom_fonts.txt       # 用户自定义的 PS 字体
|   `-- PSL_standard_fonts.txt     # PS 自带的 35 个字体
|-- spotter                        # 板块运动相关数据
|-- tools                          # GMT 辅助工具
|-- mgd77                          # mgd77 辅助文件
|-- mgg                            # mgg 辅助文件
`-- x2sys                          # x2sys 辅助文件
```

2.4 设计哲学

2.4.1 命令行

与平常接触的 Adobe PhotoShop、Adobe Illustator 之类的绘图工具不同，GMT 是没有图形界面的，无法通过鼠标点击或拖动来执行绘图操作。

GMT 是纯命令行的，所有的绘图操作都是通过在命令行敲入命令来执行的。命令行相对于图形界面的优势在于：

- 内存占用更低
- 精确控制图形的显示，比如线条宽度、圆的大小
- 方便写成脚本，具有可批量处理、可重复、可自动化的特点

2.4.2 命令格式

在目录树一节中已经说过，GMT 的 bin 目录下几乎可以说是只有 gmt 这一个有用的二进制文件，所以一切的绘图操作都是通过 gmt 这个命令来完成的。

一个 GMT 命令由“gmt + 模块 + 选项 + 参数”构成，写成如下形式:

```
gmt module -Axx+bxxxx -Bxx+axxxx
```

其中，

- gmt 是 GMT 中“唯一”的一个二进制文件，所有 GMT 命令必须以 gmt 开头
- module 是用于完成某个特定操作的模块的名字
- -A 是模块 module 提供的选项，xx 为选项 -A 的参数
- +b 是选项 -A 的子选项，xxxx 为该子选项的参数

一个完整的示例:

```
gmt pscoast -R0/20/0/20 -JM6i -Ggray -Wthin -B5 -B+t"Title with spaces" -V -P > map.ps
```

其中，

- 命令以 gmt 开头
- pscoast 是用于绘制海岸线的模块
- -R 、-J 、-G 等都是 pscoast 模块的选项
- -B+t"Title with spaces" 中 +t 是选项 -B 的子选项，"Title with spaces" 是子选项 +t 的参数

几点说明:

- 若模块名以 gmt 开头，则模块名中的 gmt 可省略。比如 gmt gmtconvert xxx xxx 可以简写为 gmt convert xxx xxx
- 模块名、选项等均区分大小写
- 每个模块可以使用哪些选项由模块自己定义，具体参考每个模块的语法说明
- 选项以 - 开头，后接**单个字符**表示某个选项，字符后接选项的参数以及子选项
- 不以 - 开头的参数，都会被当做文件，GMT 会尝试去读取
- 子选项以 + 开头，后接**单个字符**以及子选项的参数

- 官方文档中子选项称为 modifier，可以译为“修饰符”，本文档中统一使用“子选项”
- 各选项间以空格分隔，选项内部不能有空格
- 选项内部的字符串，若存在空格，应用单引号或双引号括起来
- 注意 `-A` 、—A`` 以及 ``- A` 的细微区别，这通常是由于标点符号的全角和半角导致的，GMT 中只能使用第一种
- GMT4 官方文档的网页版、PDF 版以及网络上某些介绍 GMT 的博客中，该字符都是有问题的，直接复制运行通常都会报错。GMT5 的官方文档以及本文档不存在该问题。

2.4.3 模块分类

GMT 的所有绘图操作都是通过调用不同的模块来完成的，但 GMT 除了绘图之外，还有数据处理以及格式转换等功能。根据功能的不同，大致将 GMT 的众多模块分为如下几类：

- 参数设定类：如 `gmtset`
- 绘图操作类：如 `psxy` 、`grdimage`
- 数据处理类：如 `gmtmath` 、`grdmath`
- 格式转换类：如 `xyz2grd` 、`psconvert`

2.5 PostScript

前面已经多次提到了 PostScript（简写为 PS），这一节将简单介绍一下 PS 语言与 PS 格式。

2.5.1 PS 是什么

PostScript 是一种用于描述**矢量图形**的页面描述语言。简单的说，用 PostScript 语言写成的文件就是 PS 格式的图片，一般文件后缀用 `ps` ，简称为 PS 文件。

提到格式，很多人都会有畏惧感，总觉得格式是个很复杂的东西。其实，格式不过是一种定义，定义了将信息以何种方式保存到文件中，那些用于打开某种格式的软件，只不过恰好知道了格式的定义，并且根据定义将文件中的信息提取出来，然后呈现给用户，仅此而已。比如 `doc` ，用 MS Word 可以打开，用 WPS 也可以打开，如果你愿意，你也可以自己读 `doc` 的格式定义，自己写代码读取 `doc` 格式的文件。

下面教你怎样从零创建一个 PS 格式的文件，以消除对 PS 格式的陌生感。

1. 首先打开你最常用的一款**文本编辑器**（比如 `vim` 、`gedit` 或 `notepad++` ），新建一个空白文件；

2. 将如下内容复制到该空白文件中:

```
%! PS-Adobe-3.0
/Helvetica findfont 20 scalefont setfont
150 400 moveto
(PostScript is not that hard!) show

showpage
%%Trailer
%%EOF
```

3. 将该文件以文件名 `simple.ps` 保存

4. Linux 下用命令 `gs simple.ps` 查看该文件；Windows 下应该直接双击就可以看到

解释一下这个文件:

- `%! PS-Adobe-3.0` 表明该文件是 PostScript 格式，且是 PS3.0 版
- `Helvetica` 是 PS 内置的一种字体，`findfont` 命令用于寻找、缩放、设置字体
- `150 400 moveto` 将坐标原点移动到某位置
- `(text) show` 用于显示文字
- `showpage` 表示显示该页

是不是很简单? GMT 的绘图模块本质上就是生成一堆 PS 代码，只要把这些 PS 代码按照一定的规则保存到纯文本文件中，即可用 PS 阅读器查看绘图效果。

2.5.2 PS 的优点

GMT 所有的绘图模块都只能生成 PS 代码，将这些 PS 代码保存到 PS 文件中即可完成绘图。

不管当初 GMT 是以什么理由选择 PS 作为图像格式的，就今天来看，PS 文件具有如下优势:

1. 矢量图形格式

   PS 是矢量图形格式，即用点、直线或多边形等数学方程的几何元素来表示图像。因而可以任意旋转与缩放而不是出现图像失真。

2. 易于转换为其他格式

   GMT 提供了 `psconvert` 模块，可以很方便地将 PS 文件以任意精度转换为 jpeg、png、eps、pdf 等图片格式，以满足不同情形下的需求。

2.5.3 PS 解释器

PS 解释器，或称 PS 阅读器，是用于查看 PS 文件的软件。

- ghostscript
- gsview
- evince
- zathura (Linux only)
- SumatraPDF (Windows only)

2.6 Linux 命令行

本节会简单介绍一下在使用 GMT 的过程中需要掌握的一些与 GMT 无关的基础知识。这些基础知识多数是以平台无关的，既适用于 Linux 也适用于 Windows，个别仅适用于 Linux。由于我几乎不在 Windows 下用 GMT，所以这里所说的均以 Linux 为准，对 Windows 仅供参考。

2.6.1 终端

终端在 Linux 下称为 terminal。打开终端后，在终端键入如下命令:

```
$ gmt pscoast -R70/140/2/60 -Ggray -JM6i -Wthin -B5 -V -P > map.ps
pscoast: GSHHG version 2.3.4
Derived from World Vector Shoreline, CIA WDB-II, and Atlas of the Cryosphere
Processed by Paul Wessel and Walter H. F. Smith, 1994-2014
pscoast: Working on bin #   301
pscoast: Done
```

上面的命令会绘制中国及其周边的海岸线数据，图片保存到 `map.ps` 中，可以用 gs 命令查看:

```
$ gs map.ps
```

终端在 Windows 下称为 cmd。“开始” -> “附件” -> “命令提示符” 即可启动 cmd，也可以直接在开始按钮中的搜索框中直接搜索 “cmd”。在 cmd 中键入上面的绘图命令即可绘制图形。直接双击生成的 PS 文件即可用 gsview 查看绘图效果。

2.6.2 脚本

一张稍复杂的图通常都需要多个 GMT 命令来完成。你可以在终端键入如下两个命令来绘制海岸线并在某个点处加一个五角星:

```
gmt pscoast -R70/140/2/60 -Ggray -JM6i -W1/thin -B5 -K -P > map.ps
echo 115 40 | gmt psxy -R70/140/2/60 -JM6i -Sa0.5c -Gred -O >> map.ps
```

直接在终端敲命令是不是很麻烦? 一旦敲错了就得把光标移过去重新编辑，或者哪里不满意想修改，还得查找命令历史把命令找出来重新执行一遍，又或者一个不小心命令历史丢了，原来的画图命令再也找不到了。这一切都可以通过脚本来解决。

以 Linux 为例，新建一个名为 `test.sh` 的文件，将上面的两行命令复制到文件中并保存就完成了一个脚本，然后在终端中键入如下命令即可执行该脚本:

```
$ sh ./test.sh
```

对于 Windows 用户，可以将上面的两行命令复制到名为 `test.bat` 的文件内并保存，然后双击该 bat 文件即可执行脚本。

Linux 下最基础的脚本语言是 bash，Windows 下则是 bat，当然还有跨平台的 Perl 和 Python。本文档的所有示例都使用 bash 语法。

脚本最最简单的用法就是像上面的示例那样，把一堆要执行的命令复制进去即可，但脚本的功能可不仅仅如此，要学会善于利用脚本语言的强大功能，比如:

1. 定义变量，代替命令中重复的部分
2. 用脚本语言完成计算与数据处理
3. 利用脚本实现循环
4. 利用脚本的命令行参数实现脚本的可复用性

2.6.3 后缀

经常接触的 `.doc` 、`.ppt` ，以及上面提到的 `.sh` 、`.bat` 、`.ps` 等，都是文件后缀，文件后缀的主要作用是表明该文件是什么格式或什么类型的文件。

Windows 下，每一个文件后缀都与特定的应用程序相关联，比如后缀为 `.doc` 的文件默认都是用 MS Word 打开，当然也可以选择用其他应用程序打开。后缀为 `.bat` 的文件，在双击的时候会直接被执行，想要编辑的话就需要右键编辑。

Linux 下，文件后缀就没那么重要了。一个 bash 脚本可以用 `.sh` 结尾，也可以用 `.gmt` 结尾，甚至没有后缀。不管后缀是什么，如果用 `sh` 来执行该脚本，这个脚本就会被当成 bash 脚本；如

果用 `perl` 来执行该脚本，这个脚本就会被当成 Perl 脚本（bash 脚本被当成 Perl 脚本来解释，会直接报错）。因而在 Linux 下，对文件的后缀并没有严格的要求，后缀的作用仅仅是让用户一眼就可以看出来文件是什么格式而已。

2.6.4 标准输出流和标准错误流

标准输出流（STDOUT）用于显示输出数据，标准错误流（STDERR）用于显示错误消息。一般来说，标准输出流和标准错误流都是屏幕。

GMT 中，模块的正常输出都发送到标准输出流，模块的语法、警告、错误以及进程报告都发送到标准错误流，由于这两者默认情况下都是屏幕，所以标准输出流和标准错误流会混在一起，因而需要对输出进行重定向。常见的做法可以是下面的一种或多种的组合：

1. 将标准输出流重定向到数据文件中
2. 将标准错误流重定向到日志文件中
3. 将标准输出流通过管道传递给下一个命令

2.6.5 重定向

这里只介绍 GMT 中经常使用的最简单的重定向功能。

对于标准输出流：

- `>`：将标准输出流重定向到新文件中。若该文件已存在，则覆盖文件中原内容；若该文件不存在，则创建该文件
- `>>`：将标准输出流追加到文件中。若文件已存在，则将标准输出流追加到已有文件后面；若文件不存在，则创建该文件

对于标准错误流，重定向符号是 `2>` 和 `2>>` 。这里的 `2` 表示标准错误流，大于号的含义与标准输出流相同。

GMT 中最常用的是对标准输出流的重定向。前面说过，GMT 的绘图模块会生成一堆 PS 代码，并发送到标准输出流中。默认情况下这些代码会显示在屏幕上，因而需要对标准输出流做重定向:

```
gmt psxy -R70/140/2/60 -JM6i -T -K -P > map.ps
gmt pscoast -R70/140/2/60 -Ggray -JM6i -W1/thin -B5 -K -O >> map.ps
echo 115 40 | gmt psxy -R70/140/2/60 -JM6i -Sa0.5c -Gred -K -O >> map.ps
gmt psxy -R70/140/2/60 -JM6i -T -O >> map.ps
```

上例中的四个命令中都对标准输出流进行了重定向，第一个命令使用 > 将生成的 PS 代码发送到新文件中，其余命令则使用 >> 将 PS 代码追加到已有文件中。

2.6.6 管道

除了上面提到的重定向符号之外，还有一种常用的类似重定向的操作，即管道，用符号 | 表示。

管道的作用是，将上一个命令的标准输出作为下一个命令的标准输入。

举例如下，假设文件 input.dat 中包含了一系列地震的经度、纬度和震级共三列数据，想要在图上画很多圆表示地震的位置，圆的大小表示震级的大小。可以用类似如下命令:

```
gmt psxy input.dat -Rxxx ... > test.ps
```

此时 psxy 模块会读取 input.dat 文件的内容作为其输入。

也可以使用管道:

```
cat input.dat | gmt psxy ... > test.ps
```

cat 命令会读取 input.dat 的内容并将其发送到标准输出流，由于使用了管道，标准输出流中的内容被 gmt psxy 接收作为自己的标准输入流。

当然还可以使用 gawk

```
gawk '{print $1, $2, $3/10}' input.dat | gmt psxy ... > test.ps
```

gawk 会读取 input.dat 的内容，并对数据做简单处理并输出。

2.6.7 标准输入流

GMT 的某些模块需要数据才可以画图，这些数据可以来自于文件，或来自于标准输入流。

比如要绘制地震的分布，可以把地震的经纬度信息放在文件 event.loc 中，其内容如下:

```
100.0 40.0
110.0 45.0
```

将这些数据传给 GMT 有如下几种方法。

1. 直接在命令行指定文件名，命令会自动读取该文件的内容:

   ```
   gmt psxy event.loc -R70/140/20/60 -JM6i -B5 -Sc0.2c -Gred -P > map.ps
   ```

2. 直接从键盘输入

   标准输入流的默认设备是键盘。下面的例子中直接从键盘输入 GMT 所需的数据。首先执行 `gmt psxy` 命令，然后键盘键入两行数据，再按下 `Ctrl+C` 中断输入，GMT 会给出中断警告，然后按下回车键即可:

```
$ gmt psxy -R70/140/20/60 -JM6i -B5 -Sc0.2c -Gred -P > map.ps
100.0 40.0
110.0 45.0
Interrupt at /lib64/libc.so.6(__read+0x10)[0x7f8383e8d980]
Tuser: 0.004s Tsys: 0.004s VmRSS: 8340kB VmSize: 114268kB
Press return to continue, ctrl-c to quit.
$
```

3. 标准输入流重定向 <

   < 的作用是读取 < 后的文件的内容并将其作为标准输入流，与直接在命令行指定文件名类似:

```
gmt psxy -R70/140/20/60 -JM6i -B5 -Sc0.2c -Gred -P > map.ps < event.loc
```

4. 通过管道输入

   管道可以将前一个命令的标准输出作为后一个命令的标准输入:

```
cat event.loc | gmt psxy -R70/140/20/60 -JM6i -B5 -Sc0.2c -Gred -P > test.ps
```

5. Here Documents

   示例如下，两个 `EOF` 之间的所有数据都会被传递给 GMT:

```
gmt psxy -R70/140/20/60 -JM6i -B5 -Sc0.2c -Gred -P > map.ps << EOF
100.0 40.0
110.0 45.0
EOF
```

说明:

1. 上面列出的 5 种方式中，常用的是第 1、4、5 种;
2. Here Documents 方法中，`EOF` 可以被替换成其他任意字符（比如 `END` ），只要保证开始和结束的符号一致即可
3. Here Documents 方法仅适用于 bash，不适用于 bat

2.6.8 倒引号

倒引号，也称为反引号，英文为 backtick 或 backquote。倒引号的作用是将一个命令的标准输出插在另一个命令的任意位置。

例如，想要用 psxy 绘制某数据时，需要提供数据的范围 -R ，而 gmtinfo 模块可以用于计算并输出数据的范围，即需要将 gmtinfo 的输出作为 psxy 的一个选项。

比如:

```
$ gmt info in.dat -I1/1
-R0/10/0/10
$ gmt psxy in.dat -JX10c -R0/10/0/10 > map.ps
```

上面的做法需要人工干预，不适合脚本自动化，可以利用倒引号将 gmtinfo 的输出保存到变量中:

```
#!/bin/bash

R=`gmt info input -I1/1`
gmt psxy in.dat -JX10c $R > map.ps
```

上面的例子还可以进一步简化。此处变量 $R 只需要用一次，因而没有必要把 gmtinfo 的输出信息保存到变量中，可以直接在 psxy 命令中使用倒引号:

```
$ gmt psxy in.dat -JX10c `gmt info in.dat -I1/1` > map.ps
```

此处，bash 首先会执行倒引号内的命令，然后用 gmtinfo 的输出替换整个倒引号部分，再执行替换后的命令。这样的写法更易于自动化。

2.6.9 通配符

UNIX 下提供了通配符功能，使得可以基于文件名的模式选择一组文件。

UNIX 下的通配符包括:

表 2.1: 通配符

通配符	含义
*	匹配任意数目的任意字符
?	匹配任意单个字符
[ABC]	匹配中括号内的任意单个字符
[A-Z]	匹配给定范围内的任意单个字符

示例：

1. `data_*.d` 会匹配所有以 `data_` 开头，并以 `.d` 结尾的文件
2. `line_?.d` 会匹配所有以 `line_` 开头，后接任意一个字符，并以 `.d` 结尾的文件
3. `section_1[0-9]0.part` 会匹配 `section_1x0.part` 中 `x` 为 0 到 9 的文件
4. `section_[12].part` 会匹配 `section_1.part` 和 `section_2.par` 两个文件

2.7 GMT 学习资源

GMT 相关的学习资源列举如下。虽然官方文档都有对应的中文版本，但还是建议一切以官方英文版本为准。

1. GMT 官方入门教程
2. GMT 官方参考手册
3. GMT 中文参考手册
4. GMT 官方模块手册
5. GMT 中文模块手册
6. GMT 官方示例
7. GMT 中文社区示例
8. SeisMan 博客 GMT 示例
9. *GMT 开发版源码* 中的测试脚本
10. GMT 学习 QQ 群：218905582
11. GMT 官方论坛

2.8 引用

若你发表的文章中使用了 GMT 做的图或数据处理的结果，可以考虑引用 GMT 的如下系列文章：

GMT 发表在 EOS 上的文献：

- Wessel, P., W. H. F. Smith, R. Scharroo, J. Luis, and F. Wobbe, Generic Mapping Tools: Improved Version Released, *EOS Trans. AGU*, 94(45), p. 409-410, 2013. doi:10.1002/2013EO450001.

- Wessel, P., and W. H. F. Smith, New, improved version of Generic Mapping Tools released, *EOS Trans. AGU*, 79(47), p. 579, 1998. doi:10.1029/98EO00426.

- Wessel, P., and W. H. F. Smith, New version of the Generic Mapping Tools released, *EOS Trans. AGU*, 76(33), 329, 1995. doi:10.1029/95EO00198.

- Wessel, P., and W. H. F. Smith, Free software helps map and display data, *EOS Trans. AGU*, 72(41), 445–446, 1991. doi:10.1029/90EO00319.

GMT 的某些模块基于 GMT 团队发展并发表的算法。算法相关文章包括：

- Kim, S.-S., and P. Wessel, Directional median filtering for regional-residual separation of bathymetry, *Geochem. Geophys. Geosyst.*, 9, Q03005, 2008. doi:10.1029/2007GC001850. [`dimfilter`]

- Luis, J. F. and J. M. Miranda, i Reevaluation of magnetic chrons in the North Atlantic between 35°N and 47°N: Implications for the formation of the Azores Triple Junction and associated plateau, *J. Geophys. Res.*, 113, B10105, 2008. doi:10.1029/2007JB005573. [`grdredpol`]

- Smith, W. H. F., and P. Wessel, Gridding with continuous curvature splines in tension, *Geophysics*, 55(3), 293–305, 1990. doi:10.1190/1.1442837. [`surface`]

- Wessel, P., Tools for analyzing intersecting tracks: The x2sys package, *Computers & Geosciences*, 36, 348–354, 2010. doi:10.1016/j.cageo.2009.05.009. [`x2sys`]

- Wessel, P., A General-purpose Green's function-based interpolator, *Computers & Geosciences*, 35, 1247–1254, 2009. doi:10.1016/j.cageo.2008.08.012. [`greenspline`]

- Wessel, P. and J. M. Becker, Interpolation using a generalized Green's function for a spherical surface spline in tension, *Geophys. J. Int.*, 174, 21–28, 2008. doi:10.1111/j.1365-246X.2008.03829.x. [`greenspline`]

第三章　基础知识

这一章介绍 GMT 中所有程序通用的基础知识，这些知识点在 GMT 中会经常使用，因而需要全面掌握。

3.1 单位

GMT 中使用的单位分为两类：长度单位和距离单位。需要注意这两者的区别，长度单位一般用于度量纸张上的距离，而距离单位用于度量真实地球上的距离。

3.1.1 长度单位

GMT 中的长度量，可以使用厘米（cm）、英寸（inch）或点（point）为单位。它们之间的关系如下:

```
1 inch = 2.54 cm = 72 point
```

厘米、英寸和点，在 GMT 中分别用 `c` 、`i` 和 `p` 表示。例如 `5c` 表示 5 厘米，`3i` 表示 3 英寸，`2p` 表示 2 点。

GMT 中的长度量都有一个默认单位，该默认单位由 GMT 参数*PROJ_LENGTH_UNIT* 控制。通常，该默认单位是厘米。

若给定了长度量，但未显式指定长度单位时，则使用*PROJ_LENGTH_UNIT* 对该长度量进行解释。比如 `-X4` 中的长度量，会根据*PROJ_LENGTH_UNIT* 取值的不同而被解释为 4 厘米、4 英寸或 4 点。

也可以在长度量后加上单位，来显式地指定当前长度量要使用的单位。比如 `-X4c` 会被唯一地解释为 4 厘米，而不会依赖于*PROJ_LENGTH_UNIT* 的取值。

使用长度单位时，建议遵循如下几条：

- 显式指定长度量的单位，不依赖于*PROJ_LENGTH_UNIT* 的值，以免导致同一脚本在不同机器上跑的结果不同

- 单位 `p` 用于指定较小的长度量，比如线宽、字体大小
- 单位 `c` 和 `i` 用于指定较大的长度量，比如底图宽度、原点移动的具体、圆圈大小等
- 尽量使用国际单位制（`c` ）而不用美国单位制（`i` ），因为国人对于 1 厘米要比 1 英寸更有概念

3.1.2 距离单位

对于真实地球上的距离量而言，常用单位包括：

- `d`： 弧度（degree of arc）
- `m`： 弧分（minute of arc）
- `s`： 弧秒（second of arc）
- `k`： 千米（kilometer）
- `e`： 米（meter）；默认值

还有几个不常用的单位：

- `f`： 英尺（foot）
- `M`： Statute mile
- `n`： Nautical mile
- `u`： US Survey foot

对于一个距离量而言，若不指定单位，则默认其单位为 `e`（即 “米”），当然还是建议为每个距离量显式指定其单位，使得命令更加清晰。比如在地球上以某点为中心画一个特定半径的圆，半径为 `30` 等效于 `30e` 表示 30 米，半径为 `30k` 则表示 30 千米。

对于距离量而言，还涉及到如何计算地球上两点之间距离的问题。GMT 中提供了三种不同精度的距离计算方式，分别是 Flat Earth 距离、大圆弧距离和大地测量距离。

1. 大圆弧距离：将地球当做球体，是 GMT 中默认使用的距离计算方式。比如距离量 `50k` 就默认用大圆弧距离来计算
2. Flat Earth 距离：是大圆弧距离的一阶近似，通过在距离量前加上减号来使用这种计算方式。比如距离量 `-50k` 则使用 Flat Earth 方式计算距离
3. 大地测量距离：计算距离时考虑了地球椭率，可以通过在距离量前加上加号来使用这种计算方式。比如距离 `+50k` 则使用大地测量方式来计算距离

从计算精度上看：大地测量距离 > 大圆弧距离 >Flat Earth 距离；从计算速度上看，Flat Earth 距离 > 大圆弧距离 > 大地测量距离。

关于三种距离计算方式的详细介绍，见附录三种距离计算方式 。

3.2 画布

要画一张图，首先需要准备一个画布。前面已经说过，GMT 的画布就是 PS 文件。GMT 默认的画布是 A4 大小的横置的白色 PS 文件。

本节介绍的内容，涉及到后面的一些知识，因而这里只是做介绍，具体如何操作在后面的章节中会提到。

3.2.1 画布尺寸

GMT 默认的画布尺寸为 A4，即 `210mmx297mm` 。修改 GMT 参数*PS_MEDIA* 可以控制画布尺寸，GMT 提供了几十种预定义的画布尺寸可供选择，还支持自定义尺寸，见*PS_MEDIA* 的说明。

一般不建议随便修改纸张尺寸，一方面期刊对于图件的尺寸可能有要求，另一方面其他尺寸的纸张的打印效果可能很差。

3.2.2 画布颜色

默认的画布颜色为白色。可以通过设置*PS_PAGE_COLOR* 来修改画布颜色。

3.2.3 画布方向

取一张 A4（210mmx297mm）纸放在桌面上，有两种放置的方式：竖着放和横着放。如图所示：

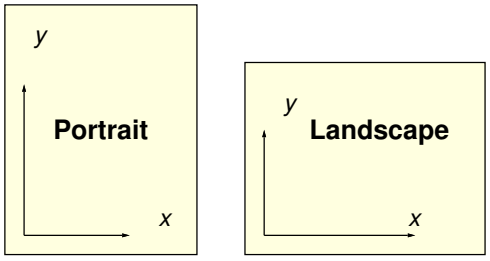


图 3.1: 画布方向

图中左边把纸张竖着放的称为 Portrait（肖像）模式，右边把纸张横着放的方式称为 Landscape（风景）模式。记起来也很简单，人是瘦长的，所以人的肖像照需要竖着拍，风景是矮胖的，就需要横着拍。

对于 Portrait 模式和 Landscape 模式，还有另外一种理解方式，即认为画布总是竖着放的。若 X 轴从左到右递增，与画布的短边平行，Y 轴从下往上递增，与画布的长边平行，则称之为 Portrait 模式；若 X 轴从下往上递增，与画布的长边平行，Y 轴从右往左递增，与画布的短边平行，则称为 Landscape 模式。Landscape 模式可以认为是在 Portrait 模式的基础上先将坐标原点沿 X 轴移动到右下角，在将坐标系旋转 90 度得到。

由于历史原因，GMT 中画布的默认放置方式是 Landscape 模式。修改画布的方向有两种方式：

1. 在第一个**绘图**命令中使用 `-P` 选项
2. 修改 GMT 参数中的*PS_PAGE_ORIENTATION*

3.2.4 底图原点

准备好画布之后，可不能随便找一个点就开始画了，还需要定义底图的原点。GMT 默认的底图原点距左下角（`1i,1i`），即坐标原点离纸张左下角的距离是水平方向偏移 1 英寸，垂直方向偏移 1 英寸。

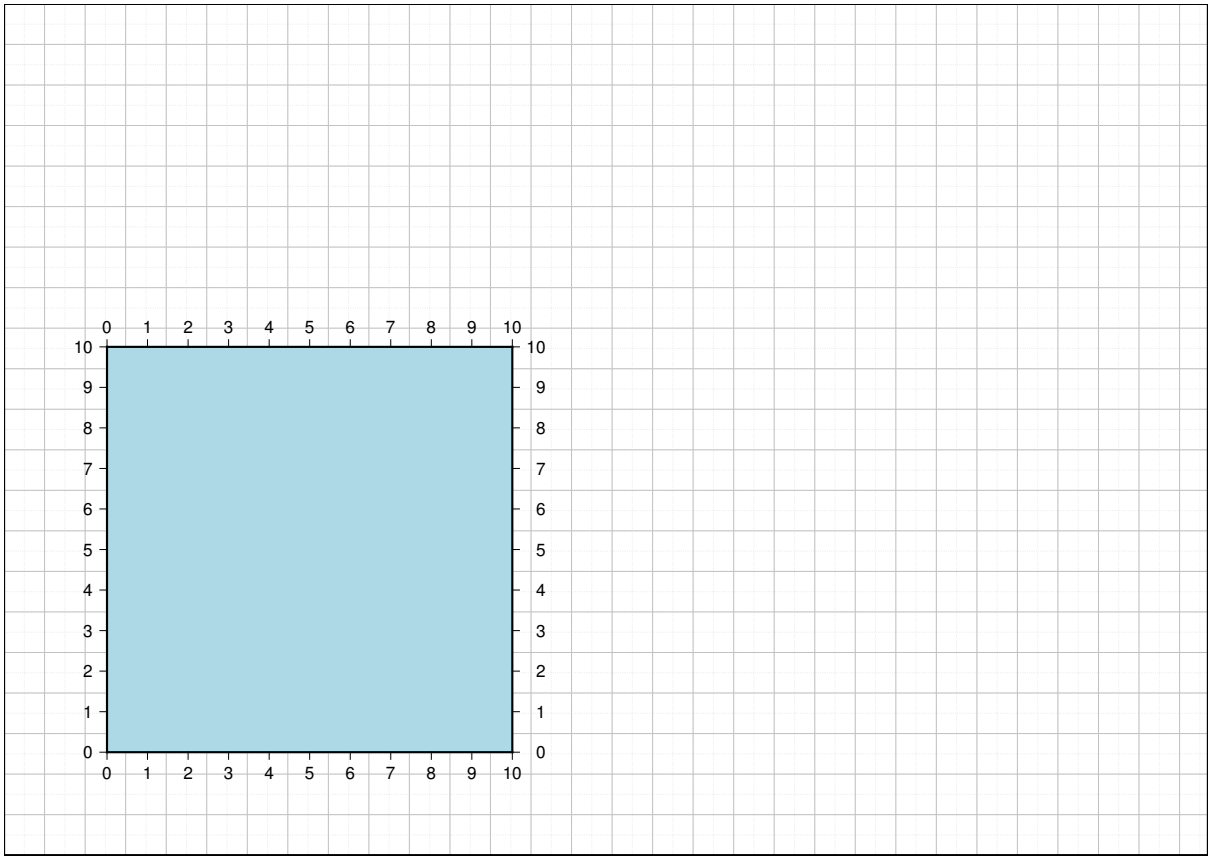


图 3.2: 底图原点

图中所示是一个完整的 A4 画布。其中灰色实线的间距是 1 cm，灰色虚线（右键查看大图）的间距是 0.5 cm。图中的底图用如下命令得到:

```
gmt psbasemap -R0/10/0/10 -JX10c/10c -B1 -B+glightblue > map.ps
```

该命令中，`-JX10c/10c` 规定了底图的宽和高都是 10 cm，可以看到，底图的左边界位于 `1i` ，右边界位于 `1i+10c` 处，下边界位于 `1i` ，上边界位于 `1i+10c` 处。需要说明的是，此处提到的底图原点是指矩形底图的坐标轴的左下角相对于画图左下角的位置，而不包括坐标轴的标注或标签部分。

可以通过两种方式修改底图的原点:

1. 修改 GMT 参数*MAP_ORIGIN_X* 和*MAP_ORIGIN_Y*
2. 命令中使用 `-X` 和 `-Y` 选项

3.3 颜色

既然是绘图，丰富的颜色是必须的。GMT 中，可以通过五种不同的方式来指定颜色。

3.3.1 颜色名

通过颜色名指定颜色是最直观的方式了。常见的颜色如 `white` 、`black` 、`red` 、`orange` 、`yellow` 、`green` 、`cyan` 、`blue` 、`magenta` 、`gray` （或 `grey` ）和 `brown` 等。除 `white` 和 `black` 之外，其余的几种常见颜色名还可以加上前缀 `light` 或 `dark` ，以表示浅色和深色。比如 `lightblue` 、`blue` 、`darkblue` 分别表示浅蓝、蓝色和深蓝。

GMT 共支持 663 种颜色名。Linux 下可以使用 `man gmtcolors` 命令查看 GMT 支持的颜色名列表，或直接查看 GMT 自带的颜色文档 `${GMTHOME}/share/doc/pdf/GMT_RGBchart_a4.pdf` 。

所有的颜色名都是不区分大小写的，所以 `lightblue` 、`LIGHTBLUE` 或者 `LightBlue` 都是合法的颜色名。

3.3.2 RGB

即三原色光模型，或又称 RGB 颜色模型，是一种加色模型，将红（**R**ed ）、绿（**G**reen）、蓝（**B**lue）三原色的色光以不同的比例相加，以产生多种多样的色光。

GMT 中可以通过指定 `r/g/b` 的格式来指定 RGB 颜色，其中 `r` 、`g` 、`b` 的取值范围都是 0 到 255，三者用反斜线 `/` 分开。

RGB 颜色示例：

- `0/0/0` ： 黑色；
- `255/255/255` ： 白色；
- `255/0/0` ： 红色；
- `0/255/0` ： 绿色；
- `0/0/255` ： 蓝色；

RGB 颜色除了可以用上面的表示法之外，还可以用 HMTL 中常用的表示法 `#RRGGBB` ，即分别用两位的十六进制数字表示每个颜色通道，0 对应的 16 进制是 `00` ，255 对应的 16 进制是 `FF` 。例如 `#000000` 即黑色，`#FF0000` 即红色。十六进制数用大小写表示均可。

3.3.3 HSV

通过 `h-s-v` 格式指定颜色，其中 HSV 分别代表色相（Hue）、饱和度（Saturation）、明度（Value）。

- 色相（H）是色彩的基本属性，就是平常所说的颜色名称，如红色、黄色等，取值范围为 0 到 360。
- 饱和度（S）是指色彩的纯度，越高色彩越纯，低则逐渐变灰，取值范围为 0 到 1。
- 明度（V）是色彩的亮度，取值范围为 0（dark）到 1（light）。

三者之间用破折号 `-` 分开，例如 `200-0.1-0.1` 。

3.3.4 CMYK

印刷四分色模式，是彩色印刷时采用的一种套色模式，利用色料的三原色混色原理，加上黑色油墨，共计四种颜色混合叠加，形成所谓“全彩印刷”。四种标准颜色是：

- **C**yan：青色，又称为天蓝色或是湛蓝
- **M**agenta：品红色，又称为洋红色
- **Y**ellow：黄色
- blac**K**：定位套版色（黑色）

四者的取值范围均为 0 到 1，用反斜线 `/` 分开，例如 `0.2/0.3/0.4/0.4` 。

3.3.5 灰色

灰色是日常常见的一种颜色，而灰色有可以根据灰的不同程度细分为不同的灰色。指定灰色的办法很简单，用一个数值表示灰度即可，其取值范围为 0 到 255。例如 0 表示黑色，255 表示白色，128 表示灰色。

除了用灰度表示之外，灰色还可以用前面提到的几种形式表示。

1. 用 RGB 表示灰度

   灰色本质上就是 R=G=B 的一种颜色。因而 128/128/128 代表灰度为 128 ，200/200/200 代表灰度是 200 。

2. 用 GMT 颜色名表示灰度

   GMT 自定义了多个名字来表示灰度，除了前面说过的 gray 、lightgray 和 darkgray 之外，还有 gray0 、gray1 一直到 gray100 。其中 gray0 即黑色，gray100 即白色。

3.3.6 颜色小结

GMT 中可以用五种方法指定颜色，分别是：

- 颜色名：red
- RGB 值：30/25/128 或 #00FA84
- HSV 值：200-0.1-0.1
- CMYK 值：0.2/0.3/0.4/0.5
- 灰度：30

3.3.7 透明色

每一种颜色都可以额外指定不同的透明度。GMT 中，可以通过在颜色后加上 @ 再加上透明度来得到不同程度的透明色。透明度的取值范围是 0 到 100，0 表示不透明，100 表示全透明。例如：red@25 、30/25/128@60 。

除了可以指定某个颜色的透明度之外，还可以指定整个图层的透明度。在使用透明特性时，还有一些额外的注意事项，见*-t 选项* 一节。

3.4 画笔

有画笔才能画线条，才可以画出三角形、圆形等各种复杂的形状。

GMT 中的画笔有三个属性：笔宽、颜色和线型，三者用逗号分隔，即 `<width>,<color>,<style>` 。在 GMT 模块的语法介绍中，一般用 `<pen>` 表示画笔属性，读者在见到 `<pen>` 时应自行脑补成 `<width>,<color>,<style>` 。

在指定画笔属性时，可以指定三个属性中的任意一个或多个属性，但要保证属性的相对顺序。

3.4.1 画笔线宽

可以通过两种方式指定画笔宽度。

1. 宽度值 + 单位，即 `<width>c|i|p`

   线宽的单位可以取 `p` 、`c` 或 `i` （见单位 一节），若不显式给定线宽单位，则默认线宽单位为 `p` 。

   推荐只使用 `p` 作为线宽单位，毕竟多数情况下线条的宽度都比较小，用“小”单位 `p` 作为单位更方便些。并建议总是显式给定线宽单位，以使得命令的参数更加清晰易读，比如 `1p` 、`0.25p` 。

2. 预定义画笔宽度名

   对于一些常用的画笔宽度，GMT 将其定义为特定的名字，以方便用户使用。下表中列出了这些预定义画笔宽度名及其对应的线宽。

表 3.1: GMT 预定义画笔宽度名

线宽名	线宽	线宽名	线宽
faint	0	thicker	1.5p
default	0.25p	thickest	2p
thinnest	0.25p	fat	3p
thinner	0.50p	fatter	6p
thin	0.75p	fattest	12p
thick	1.0p	obese	18p

说明: 最细的画笔宽度可以通过设置笔宽为 `0p` 或 `faint` 来实现，但画笔的实际宽度由具体的设备来决定。

3.4.2 画笔颜色

见颜色 一节。

3.4.3 画笔线型

画笔线型属性控制了线条的外观，可以用四种方式表示：

1. 预定义线型名

   GMT 预定义了几种线型名，包括 `solid` （实线）、`dashed` （虚线）和 `dotted` （点线）。

2. 简单符号

   - 句点号 `.` 表示点线
   - 破折号 `-` 表示虚线

3. 组合符号

   通过对简单符号的任意组合可以获得更多的线型，比如 `.-` 表示点划线，`..-` 表示两个点号与一个破折号交替出现。

4. 复杂线型

   可以通过 `string:offset` 的形式自定义更复杂的线型。其中 `string` 是一系列由下划线 `_` 分隔的数字组成，这一系列数字中，第奇数个数字表示实线的长度，第偶数个数字表示空白的长度。通过实线和空白的长度的不同组合，即可构成多种复杂的线型。`offset` 表示线段开始处整个线型的初始相位移动。

   例如，`4_8_5_8:2` 表示线型首先是长度为 `4p` 的实线，然后是长度为 `8p` 的空白，紧接着长度为 `5p` 的实线和长度为 `8p` 的空白，然后按照该模式不断重复。此处的 `offset` 值为 `2p` ，因而线段的最开始处，第一条实线的长度 `4p` 经过相移后长度为 `2p` 。线型中的这些数值，默认单位是 `p` ，也可以使用 `c` 或 `i` 。

说明：

1. `.` 和 `-` 的绝对长度由画笔宽度来决定。`.` 的长度等于画笔宽度；`-` 的长度为 8 倍画笔宽度；点线或虚线中段间空白的长度为 4 倍画笔宽度。

3.4.4 画笔小结

画笔包含三个属性：宽度、颜色、线型。通过不同的组合方式，可以定义无穷多种画笔。

下图中展示了不同风格的画笔，读者可以下面命令里 `-W` 选项中的 `<pen>` 修改成不同的值来理解这一节的内容：

```
gmt psxy -R0/10/0/5 -JX10c/5c -W<pen> > pens.ps << EOF
0 2
```

```
10 2
EOF
```

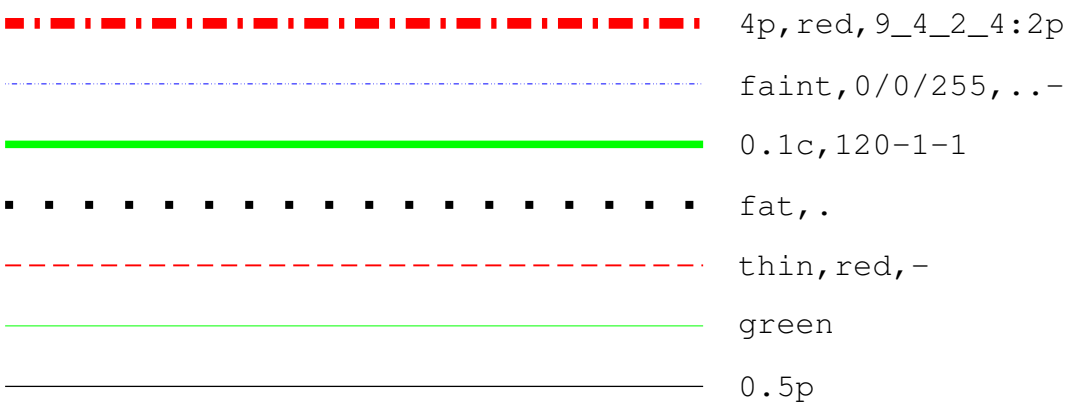


图 3.3: GMT 画笔示例

3.4.5 其他属性

除了上面提到的三个画笔属性之外，GMT 中还有一些参数可以影响线条的外观。这些参数包括：

- *PS_LINE_CAP* ： 控制线段顶端的绘制方式
- *PS_LINE_JOIN* ： 控制线段拐点/交点的绘制方式
- *PS_MITER_LIMIT* ： 控制线段拐点在 `miter` 模式下的阈值

3.5 填充

用彩色的画笔绘制了圆形或多边形之后，还需要为其填充颜色。在 GMT 模块的语法介绍中，一般用 `<fill>` 表示需要指定填充属性，读者在见到 `<fill>` 时应自动联想到本节介绍的内容。

填充 `<fill>` 有两种方式/形式：填充颜色和填充图案。GMT 中通常使用 `-G<fill>` 选项来填充颜色，本节以该选项为例。

3.5.1 填充颜色

给区域填充颜色很简单，直接用 `-G<color>` 即可。颜色在颜色 一节已经介绍过了。比如 `-Gred` 、`-G230/200/0` 。

3.5.2 填充图案

还可以给区域填充图案（pattern），比如地质填图里经常会给不同区域填充不同的图案以区分不同的地质时期。其语法为:

```
-G[p|P]<dpi>/<pattern>[:B<color>[F<color>]]
```

<pattern> 有两种取法:

- 取 1 到 90 内的整数，表示使用 GMT 提供的 90 种预定义的 64x64 位图图案
- 取文件名，表示使用自定义的 1、8、24 位 Sun 光栅文件作为位图图案

<dpi> 设置了位图图案在当前页面上的分辨率。dpi 越大，则区域内位图重复的次数越多；若设置 dpi 为 0，则根据 gmt.history 中的内容决定所能使用的最大分辨率。

若使用 -GP 而不是 -Gp ，则图案会发生位反转，即白色区域变成黑色，黑色区域变成白色（仅对 1 位位图或 GMT 预定义位图图案有效）。

对于 GMT 预定义的图案以及用户自定义的 1 位位图来说，可以用 :B<color>[F<color>] 设置图案的前景色和背景色，以分别替换默认的黑色和白色像素点。若设置前景色或背景色为 - ，则视为前景色或背景色为透明。

下图列出了 GMT 中预定义的 90 种位图图案（右键查看大图），所有图案都是使用默认的黑白色在 <dpi> 取 300 的环境下生成的。

3.5.3 填充小结

GMT 中通常使用 -G 选项填充多边形或符号。有两种填充方式，分别是:

- 填充颜色: -G<color>
- 填充图案: -Gp<dpi>/<pattern>

下面给出了一些填充的示例:

- -G128
- -G127/255/0
- -G#00ff00
- -G25-0.86-0.82
- -GDarkOliveGreen1
- -Gp300/7

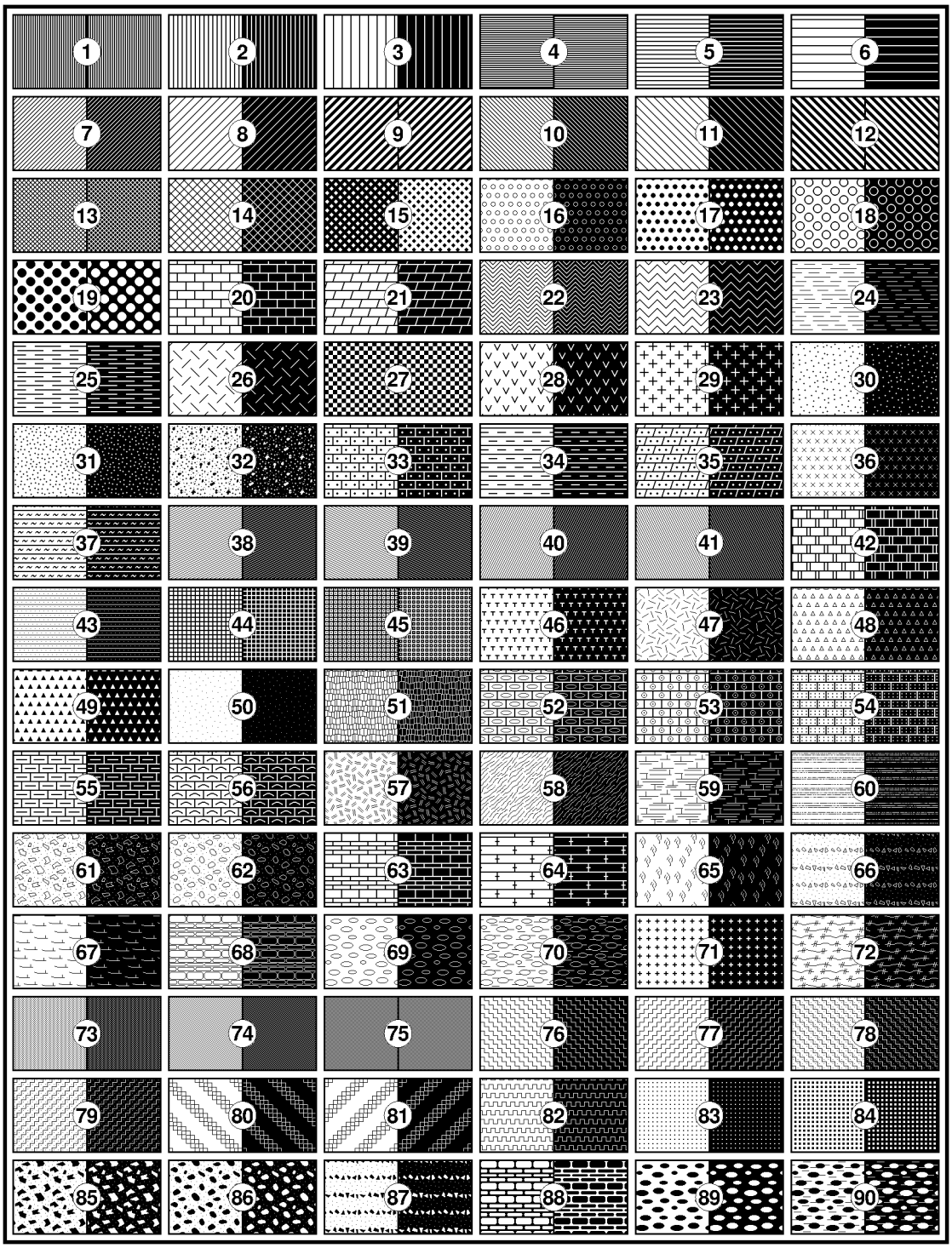


图 3.4: GMT 内置位图图案

- `-Gp300/7:Bred`
- `-Gp300/7:BredF-`
- `-Gp100/marble.ras`

说明：

1. 由于 PostScript 实现的限制，`-G` 选项里使用的光栅图片必须小于 146x146 像素；若要使用更大的图像，需要使用 `psimage`
2. 在 PostScript Level 1 下，图案填充是通过使用多边形做路径裁剪实现的。复杂的裁剪路径会需要更多的内存，因而可能导致某些 PS 解释器由于内存不足而退出。在这种情况下，建议使用灰度填充区域。

3.6 文字

前面几节说了 GMT 中如何画线条以及如何填充颜色，这节说说文字的那些事。

文字，或者称文本或字符串，主要由三个属性控制：文字大小、字体类型、填充色。三个属性之间用逗号分隔，即 `<size>,<fonttype>,<fill>` 。三者均是可选的，但先后顺序不可乱。若其中任意一个属性被省略，则使用该属性的默认值或上一次的设置值。

3.6.1 文字大小

文字大小，可以理解成字号，用数字加单位表示。在不指定单位的情况下默认单位为 `p` ，也可加上 `c` 、`p` 或者 `i` 显式指定单位，比如 `15p` 。

有些中文期刊可以会有类似“图片标题是四号字”这样的要求，这就需要知道 GMT 中的字体大小与 MS Word 中字号大小的对应关系。如下表所示：

表 3.2: Word 字号与 GMT 中字号对应关系

字号	p	字号	p
初号	42	小初	36
一号	26	小一	24
二号	22	小二	18
三号	16	小三	15
四号	14	小四	12
五号	10.5	小五	9
六号	7.5	小六	6.5
七号	5.5	八号	5

3.6.2 字体类型

字体类型控制了文字使用哪一种字体。大多数 PostScript 解释器都内置 35 种标准字体，GMT 默认支持且只支持这 35 种标准字体。若你使用的 PS 解释器不支持其中某些字体，则会自动用默认字体（一般是 Courier）替换需要的字体。

下图给出了 GMT 支持的 35 种字体的列表：

GMT 中可以用字体名（区分大小写）或对应的字体编号来指定字体。上图中给出了每种字体的字体编号以及字体名称。每个字体名称使用的是自己相对应的字体，所以可以从图中直观地看出不同字体的区别。

图中大多数字体都很直观，比较特别的字体有两个，Symbol（12 号）和 ZapfDingbats（34 号），在特殊字体 中会专门介绍。

3.6.3 填充色

可以为文字指定颜色或图案，也就是常说的文字颜色，见填充 一节。

3.6.4 描边

在给文字指定填充色的同时，可以在填充色 `<fill>` 后加上 `=<pen>` 来指定文本轮廓（即描边）的画笔属性。`<pen>` 的用法见画笔 一节。比如 `red=2p,blue` 表示将文字填充为红色，并使用宽度为 `2p` 的蓝色线条给文字描边。若填充色 `<fill>` 为 `-`，则不对文字做填充，即实现空心文字的效果。

#	Font Name	#	Font Name
0	Helvetica	**17**	**Bookman-Demi**
1	**Helvetica-Bold**	***18***	***Bookman-DemiItalic***
2	*Helvetica-Oblique*	19	Bookman-Light
3	***Helvetica-BoldOblique***	*20*	*Bookman-LightItalic*
4	Times-Roman	21	Helvetica-Narrow
5	**Times-Bold**	**22**	**Helvetica-Narrow-Bold**
6	*Times-Italic*	*23*	*Helvetica-Narrow-Oblique*
7	***Times-BoldItalic***	***24***	***Helvetica-Narrow-BoldOblique***
8	Courier	25	NewCenturySchlbk-Roman
9	**Courier-Bold**	*26*	*NewCenturySchlbk-Italic*
10	*Courier-Oblique*	**27**	**NewCenturySchlbk-Bold**
11	***Courier-BoldOblique***	***28***	***NewCenturySchlbk-BoldItalic***
12	Σψμβολ (Symbol)	29	Palatino-Roman
13	AvantGarde-Book	*30*	*Palatino-Italic*
14	*AvantGarde-BookOblique*	**31**	**Palatino-Bold**
15	**AvantGarde-Demi**	*32*	***Palatino-BoldItalic***
16	***AvantGarde-DemiOblique***	*33*	*ZapfChancery-MediumItalic*
		34	✺❁❒❄✤❇■✳✪❁▼▲ (ZapfDingbats)

图 3.5: GMT 中的 35 种 PS 标准字体

使用 `=<pen>` 语法绘制文本轮廓时，轮廓线条有一半宽度位于文字外部，另一半宽度会遮住字体。为了避免这一现象，可以使用 `=~<pen>` 语法，此时在绘制文字轮廓时只绘制文字外部的半个线宽的线条。

3.6.5 示例

下图给出了几种指定文本属性的方式：

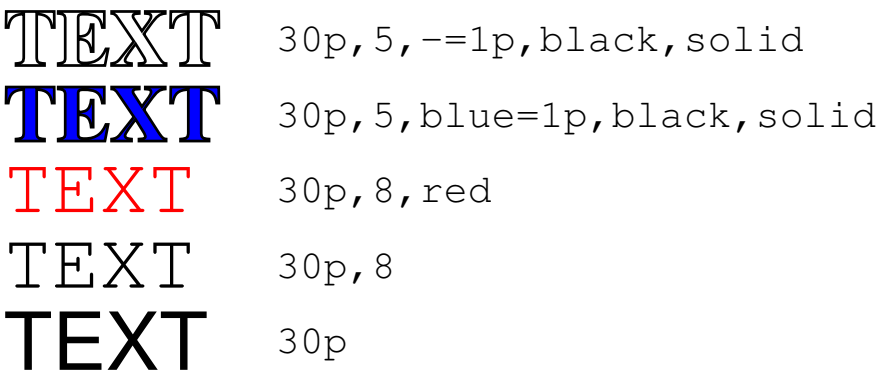


图 3.6: GMT 文本属性示例

从下往上，一一解释一下：

1. 字号为 30p ，其余使用默认值
2. 字号为 30p ，使用 8 号字体
3. 字号为 30p ，8 号字体，颜色为红色
4. 字号为 30p ，5 号字体，字色为蓝色，用宽度为 1p 的黑色实线描边
5. 与前一个相同，唯一区别在于字色为 - ，相当于透明色，产生空心文字

读者可以将下面命令中 -F+f 后的 <font> 修改为不同的值以帮助理解本节的内容:

```
echo 2.5 0.5 TEXT | gmt pstext -R0/5/0/1 -JX15c/2c -F+f<font> > text.ps
```

3.7 输入与输出

GMT 各个模块的输入和输出，可以分为如下几类:

1. 表数据
2. 2D 网格数据
3. CPT 文件
4. PS 代码
5. 警告和错误消息
6. 退出状态码

3.7.1 表数据

英文称为 table data，也称为列数据或多列数据，常用于表示点和线。

表数据中有 N 个记录，每个记录都有 M 个字段。表数据可以有三种形式：ASCII 表、二进制表和 netCDF 表。

最常用的表数据形式是 ASCII 表，可以用编辑器直接编辑。ASCII 表中有 N 行 M 列，每行是一个记录，每列是一个字段。一个记录内的字段之间用空格、制表符、逗号或分号分隔。

下面这个 ASCII 表数据有 10 行 3 列，三列数据分别代表经度、纬度和深度:

```
133.949  34.219  20
133.528  34.676  15
130.233  33.410  43
135.133  35.313  35
```

```
131.377  34.398  22
132.792  34.457  34
133.620  34.936  6
131.101  32.811  23
129.435  33.212  55
133.144  33.647  67
```

关于表数据的详细介绍，见表数据 。

3.7.2 2D 网格数据

GMT 可以绘制 2D 网格数据。通常，2D 网格文件的横轴是经度、纵轴是纬度，Z 值可以表示高程、重力值、温度、速度等。

GMT 可以使用的 2D 网格数据有三种格式：

1. NetCDF 格式
2. Sun 光栅文件
3. 自定义的二进制数据格式

最常见也最推荐的网格数据格式是 NetCDF 格式。

关于网格数据的详细介绍，见网格文件 。

3.7.3 CPT 文件

CPT 文件全称 color palette table，中文称为调色板文件或色标文件。

CPT 文件的作用在于定义了任意一个 Z 值所对应的颜色，常用于绘制彩图、灰度图，或根据数据的不同属性填充不同的颜色。

关于网格文件的详细介绍，见*CPT 文件* 。

3.7.4 PS 代码

GMT 的绘图模块会生成 PS 代码到标准输出流。为了将结果保存到 PS 文件中，需要将 PS 代码重定向到 PS 文件中。

PS 代码几乎只能作为 GMT 模块的输出，偶尔可以作为输入。

3.7.5 警告和错误消息

GMT 在运行时，会将语法信息、错误报告、警告信息等发送到标准错误流。详情见*-V 选项* 。

3.7.6 退出状态码

每个模块在退出时都会返回状态码，值为 1 表示成功，其他值表示失败。

3.8 数据格式

既然是个画图工具，肯定要输入一些数据，这就牵涉到数据格式的问题了。

GMT 可以绘制一般的笛卡尔坐标轴、地图的经纬度轴以及绝对时间轴、相对时间轴。对于不同的坐标轴，需要的数据格式也不同。

3.8.1 地理坐标

地理坐标（即经纬度）有两种表示方式：

1. 以浮点型的度数表示：比如 `-123.45` 代表-123.45 度
2. 度分秒表示：`ddd[:mm[:ss[.xxx[W|E|S|N]]]]` 。其中，`ddd` 、`mm` 、`ss` 、`xxx` 分别表示弧度、弧分、弧秒、弧毫秒。`W` 、`E` 、`S` 、`N` 分别代表西经、东经、北纬、南纬。例如 `123:27:15.120W` 表示西经 123 度 27 分 15.12 秒。

3.8.2 绝对时间坐标

绝对时间由两部分构成，即日期和时间，表示为 `<date>T<clock>` 。其中 `T` 是关键字，用于分隔日期和时间。

日期 `<date>` 可以是如下格式的一种：

1. `yyyy[-mm[-dd]]` ： 年-月-日，例如 `2013` 、`2015-10` 、`2015-01-02`
2. `yyyy[-jjj]` ： 年-一年中的第几日，例如 `2015-040`
3. `yyyy[-Www[-d]]` ： 年-第几周-该周内第几天，例如 `2014-W01-3` 、`2014-W01`

时间 `<clock>` 的格式为 `hh:mm:ss.xxx` ，例如 `10:10:35.120` 。

使用过程中需要注意：

1. GMT 的时间数据的输入/输出格式默认为 `yyyy-mm-ddThh:mm:ss.xxx` 。若想要输入其他格式的时间数据，需要修改*FORMAT_DATE_IN* 和*FORMAT_CLOCK_IN* ；若想要输出其他格式的时间数据，需要修改*FORMAT_DATE_OUT* 和*FORMAT_CLOCK_OUT*
2. 若未指定 `<date>` 则假定 `<date>` 为今日
3. 若未指定 `<clock>` 则认为是 `00:00:00`
4. 若指定了 `<clock>` 则必须要加 `T`
5. 所有绝对时间在内部都会被转换成相对于某一时刻的秒数

下面举几个绝对日期的例子：

- `2014-02-10T10:00:00.000`
- `T10:20:44.234`
- `2014-040T23:23:54.330`

3.8.3 相对时间坐标

相对时间坐标即相对于某个参考时刻的秒数、小时数、天数或年数。因而在使用相对时间时，首先要给定两个参数：参考时刻以及相对时间的单位。

GMT 参数*TIME_EPOCH* 用于指定参考时刻，*TIME_UNIT* 用于指定相对时间的单位。也可以用参数*TIME_SYSTEM* 同时指定这两个参数。默认的参考时刻为 1970 年 1 月 1 日午夜，默认的相对时间单位为秒。

在指定了参考时刻后，相对时间就跟一般的浮点数没什么区别了。那如何区分一般的浮点数与相对时间呢？有两种方式：

1. 在数据后加上小写的 `t` ，比如 `30t` 表示相对时间 30，单位由*TIME_UNIT* 决定
2. 在命令行中使用 `-ft` 选项表明当前数据是相对时间，此时不需要在数字后加 `t`

3.8.4 其他坐标

除了上面提到的地理坐标、绝对时间坐标、相对时间坐标之外的其他数据，都简单地用浮点数表示，而不去在意其物理含义及单位。比如，5 牛顿的力，5 千克的质量，在 GMT 看来都只是浮点数 `5` 。

这些浮点数坐标可以用两种方式表示：

1. 一般表示：`xxx.xxx` ，比如 `123.45`

2. 指数表示：xxx.xxExx ，其中 E 可以用 e 、D 、d 替换，比如 1.23E10

第四章　进阶知识

这一章介绍 GMT 中的一些进阶知识。这些知识点在入门的时候用处不大，但在实际绘图时却有可能会用到。

4.1 表数据

英文称为 table data，也称为列数据或多列数据。表数据中有 N 个记录，每个记录都有 M 个字段。

GMT 可以读取三种形式的表数据：ASCII 表、二进制表和 netCDF 表。

4.1.1 ASCII 表

4.1.1.1 ASCII 表简介

ASCII 表是最常见的数据形式，可以用编辑器直接编辑。数据中有 N 行 M 列，每行是一个记录，每列是一个字。一个记录内的字段之间用空格、制表符、逗号或分号分隔。每个字段可以是整数（`12` ）、浮点数（`20.34` ）、地理坐标（`12:23:44.5W` ）、绝对时间（`2010-10-20T10:30:53.250` ）、相对时间（`30t` ），同时，GMT 还可以处理大多数 CVS（Comma-Separated Values）文件，包括被双引号扩起来的数字。

例如:

```
# This is a comment line
# lon      lat  evdp
# This is another comment line
133.949  34.219  20
133.528  34.676  15
130.233  33.410  43
135.133  35.313  35
131.377  34.398  22
132.792  34.457  34
133.620  34.936  6
```

```
131.101  32.811  23
129.435  33.212  55
133.144  33.647  67
```

记录中以 # 开头的行，即第一列是 # 的记录，会被当做注释行直接忽略，不算在 N 个记录之内。所以这个 ASCII 表可以认为有 10 行 3 列，三列数据分别代表经度、纬度和深度。

不同的模块和选项的组合会对数据的列数以及每列的含义都有不同的定义，因而需要根据具体情况去准备数据。准备数据的过程中可能会用到 GMT 的 -i 、-o 选项以及 gawk 等工具。

4.1.1.2 文件头记录

在第一个记录前，可以有一个或多个与数据无关的记录，称为文件头记录 (file header records)。

记录中以 # 开头的行都被当做注释忽略，所以不算是文件头记录。其他不以 # 开头但与数据无关的行，则是文件头记录。要使用 -h 选项或设置参数*IO_N_HEADER_RECS* 跳过这些文件头记录。

下面的 ASCII 表有一个文件头记录，可以使用 -h1 选项跳过该文件头段记录:

```
# This is a comment line
# lon      lat  evdp
# This is another comment line
2015-01-05 10:20:30.456 15 45 60 6.0
133.949  34.219  20
133.528  34.676  15
130.233  33.410  43
135.133  35.313  35
131.377  34.398  22
132.792  34.457  34
133.620  34.936  6
131.101  32.811  23
129.435  33.212  55
133.144  33.647  67
```

4.1.1.3 数据段头记录

在绘制断层的时候，可以将每个断层的经纬度信息分别放在各自的文件中，但当断层数量很多时，这样做可能导致目录下有太多数据文件而混乱不堪。为了解决类似的问题，GMT 引入了多段数据的概念。

多段数据，顾名思义，就是一个文件中包含了多个数据段。为了区分每个数据段，需要在每段数据的开头加上一个额外的数据段头记录（segment header records）来标记一段新数据的开始。

数据段头记录可以是任意格式，但所有数据段头记录的第一列字符必须相同。GMT 中默认的数据段头记录的首字符为 `>` ，也可以通过修改*IO_SEGMENT_MARKER* 设置为其他字符。

IO_SEGMENT_MARKER 可以取两个特殊的值：

- 取 `B` 表示用空行作为数据段的分隔符
- 取 `N` 表示用一个所有列都是 NaN 的记录作为数据段分隔符

下面是一个包含两个数据段的多段数据，每段数据是一个多边形：

```
>
10  20
15  30
12  25
>
22  20
30  30
40  50
35  44
```

数据段头记录不仅仅用于标记数据段的开始，还可以额外指定该段数据的其他属性。比如：

- `-D` 指定某个距离值
- `-W` 指定当前数据段的画笔颜色
- `-G` 指定当前数据段的填充色
- `-Z` 设置当前数据对应的 Z 值，并从 CPT 文件中获取 Z 值对应的颜色
- `-L` 设置当前数据段的标签信息
- `-T` 设置当前数据段的一般描述信息
- `-Ph` 表明当前数据段构成的闭合多边形位于另一个闭合多边形的内部

这些数据段头记录中的选项参数将覆盖命令行中相应选项的值。

下面的多段数据，分别设置两段数据拥有不同的画笔颜色：

```
> -W2p,red
10  20
15  30
12  25
> -W2p,blue
```

```
22  20
30  30
40  50
35  44
```

4.1.2 二进制表

对于 IO 密集型任务，可以将用二进制表形式以加速。

简单地说，ASCII 表与二进制表的区别在于前者使用 `fprintf` 输出而后者使用 `fwrite` 输出。GMT 在读取二进制表数据时，无法直接判断数据中有多少个记录，每个记录有多少个字段。因而需要使用 `-bi` 选项指定二进制表数据的格式，详情见*-b 选项* 。

二进制表中也可以有文件头记录，用 `-h` 选项可以指定要跳过的字节数。二进制表也可以表示多段数据，此时用一个值为 `NaN` 的记录作为数据段头记录来标记每段数据的开始。

4.1.3 NetCDF 表

表数据也可以用 NetCDF 格式保存，该格式的好处在于通用。比如 GMT 自带的海岸线数据就是 NetCDF 的表数据。NetCDF 表数据中包含了一个或多个一维数组，每个一维数组都有对应的变量名（比如 `lon` 、`lat` 、`vel` 等等），由于 NetCDF 格式的数据中包含了很多元数据（meta data），所以读取就变得很容易。

默认情况下，GMT 在读入 NetCDF 表时会从第一个一维数组开始读，并将其作为输入的第一列，然后再读入第二个一维数组，将其作为输入的第二列，依次循环下去，直到读完自己所需要的字段数。

若需要手动指定要从 NetCDF 表中读入哪些变量，可以在 netCDF 表文件名后加上后缀 `?<var1>/<var2>/...` ，也可以直接使用 `-bic<var1>/<var2>/...` 选项。其中 `<var1>` 等是要从 NetCDF 表中读入的变量名。比如 `file.nc?lon/lat` 表示要从文件中读入 `lon` 和 `lat` 两个一维数组作为输入数据。

目前，GMT 只支持读取 netCDF 表数据，不支持写 netCDF 表数据。

4.2 网格文件

4.2.1 网格文件格式

GMT 支持多种格式的网格文件，包括 netCDF 格式、二进制网格文件以及 8 位 Sun 光栅文件。其中最常用的是 netCDF 格式。除此之外，GMT 还可以读取用户自定义的网格文件，只要用

户写好自定义网格文件的读写子程序，并将其与 GMT 函数库链接起来即可。详情参考源码中的 `gmt_customio.c` 文件。

4.2.1.1 NetCDF 网格文件

GMT 默认将 2D 网格保存为兼容 COARDS 的 netCDF 文件，一般以 `.nc` 或 `.grd` 作为文件后缀。

COARDS 是许多机构在分发网格文件时遵循的标准格式。GMT 兼容该格式，因而 GMT 可以直接读取其他机构或程序提供的网格文件，GMT 生成的网格文件也可以被其他程序读取。

netCDF 格式目前有两个版本，netCDF3 和 netCDF4。GMT 目前默认使用 netCDF4 版本的文件格式。

关于 netCDF 网格文件的详细信息，见本章的其他章节。

4.2.1.2 Native 二进制网格文件

旧版本的 GMT 不支持 netCDF 格式的文件，使用的是 GMT 自定义的二进制网格格式。新版本的 GMT 依然支持这种二进制网格格式，但不建议使用。该文件格式由两部分组成：892 个字节的头段区和长度不定的数据区。详细信息见附录*Native 二进制网格格式* 。

4.2.1.3 Sun 光栅文件

Sun 光栅文件格式包含了一个头段区以及一系列无符号一字节整型以表示 bit-pattern。详细信息见附录*Sun 光栅文件* 。

4.2.2 写网格文件

上面已经说过，GMT 可以读写三类格式的网格数据：netCDF 格式、二进制格式和 Sun 光栅文件。进一步细分，GMT 可以读取几十种不同格式的网格数据。

GMT 所支持的网格文件格式在下表列出，每种网格文件格式都有一个对应的两字符 ID。

表 4.1: 网格文件格式 ID

ID	说明
	GMT 4 netCDF standard formats
nb	GMT netCDF format (8-bit integer, COARDS, CF-1.5)
ns	GMT netCDF format (16-bit integer, COARDS, CF-1.5)
ni	GMT netCDF format (32-bit integer, COARDS, CF-1.5)
nf	GMT netCDF format (32-bit float, COARDS, CF-1.5)
nd	GMT netCDF format (64-bit float, COARDS, CF-1.5)
	GMT 3 netCDF legacy formats
cb	GMT netCDF format (8-bit integer, depreciated)
cs	GMT netCDF format (16-bit integer, depreciated)
ci	GMT netCDF format (32-bit integer, depreciated)
cf	GMT netCDF format (32-bit float, depreciated)
cd	GMT netCDF format (64-bit float, depreciated)
	GMT native binary formats
bm	GMT native, C-binary format (bit-mask)
bb	GMT native, C-binary format (8-bit integer)
bs	GMT native, C-binary format (16-bit integer)
bi	GMT native, C-binary format (32-bit integer)
bf	GMT native, C-binary format (32-bit float)
bd	GMT native, C-binary format (32-bit float)
	Miscellaneous grid formats
rb	SUN raster file format (8-bit standard)
rf	GEODAS grid format GRD98 (NGDC)
sf	Golden Software Surfer format 6 (32-bit float)
sd	Golden Software Surfer format 7 (64-bit float)
af	Atlantic Geoscience Center AGC (32-bit float)
ei	ESRI Arc/Info ASCII Grid Interchange format (ASCII integer)
ef	ESRI Arc/Info ASCII Grid Interchange format (ASCII float)
gd	Import/export via GDAL

除了上面列出的网格文件格式之外，有 C 编程经验的高级用户还可以自己自定义网格文件格式，并将读写该格式的子程序链接到 GMT 函数库中，使得 GMT 可以支持自定义网格文件格式的读取。详情见源码中的 `gmt_customio.c` 。

GMT 在读网格数据时，可以自动检测网格文件的格式，所以不用担心。而在写网格数据时，GMT 会默认使用 `nf` 格式（可以修改*IO_GRIDFILE_FORMAT* 以设置其他网格文件格式为默认值）。如果想要以其他格式写网格数据，则可以在网格文件名后加上 `=<ID>` 参数以指定要使用的网格文件格式。

无论是读还是写网格文件，都可以按照如下格式指定网格文件的文件名:

```
<name>[=<ID>[/<scale>/<offset>][/<nan>]]
```

其中

- `<name>` 是网格文件名
- `<ID>` 是写网格文件时要使用的网格文件格式
- `<scale>` 将所有数据乘以比例因子 `<scale>` ，默认值为 1
- `<offset>` 将所有数据加上一个常数 `<offset>` ，默认值为 0
- `<nan>` 表明将文件中值为 `<nan>` 认为是 NaN

`<scale>` 和 `<offset>` 都可以取为 `a` ，表明由程序自动决定比例因子和偏移量的值。在读网格文件时，会先乘以比例因子再加上偏移量；在写网格文件时，会先加上偏移量，再乘以比例因子。

举几个例子:

1. 读入 Golden 软件公司的 surfer 软件生成的网格文件，GMT 可以自动识别，故而直接用 `file.grd`
2. 读一个二进制短整型网格文件，先将所有值为 32767 的值设置为 NaN，再将数据乘以 10 并加上 32000，可以用 `myfile.i2=bs/10/32000/32767`
3. 将一个二进制短整数网格文件减去 32000 再除以 10，然后写到标准输出，可以用 `=bs/0.1/-3200`
4. 读一个 8 字节标准 Sun 光栅文件（其原始范围为 0 到 255），并将其归一化到正负 1 范围内，可以用 `rasterfile=rb/7.84313725e-3/-1` ，即先乘以因子使得数据范围从 0 到 255 变成 0 到 2，再减去 1，则数据范围变成-1 到 1
5. 写一个 8 字节整型 netCDF 网格文件，偏移距由 GMT 自动决定，可以用 `=nb//a`

GMT 还支持通过网格文件后缀自动识别网格文件格式，详情见附录网格文件后缀 一节。

4.2.3 读 netCDF 文件

netCDF 格式的设计相当灵活，可以包含多个多维变量。而 GMT 中与网格相关的模块，只能直接处理包含一个二维变量的 netCDF 文件。因而，GMT 在读取包含了多个多维变量的 netCDF 文件时，可以做一些特殊的处理。

4.2.3.1 多个二维变量的处理

当 netCDF 网格文件中包含多个二维变量时，GMT 默认会读取第一个二维变量作为 Z 值，并忽略其余的二维变量。如果用户想要自己指定读取某个特定的二维变量，可以在网格文件名后加上后缀 `?<varname>` 来实现，其中 `<varname>` 是 netCDF 文件中包含的变量名。

比如想要从文件中获取名为 `slp` 的二维变量的信息，可以用:

```
gmt grdinfo "file.nc?slp"
```

两点说明:

1. netCDF 中包含的变量名 `<varname>` 可以用 `ncdump -c file.nc` 得到
2. Linux 下问号会被解析为通配符，因而在命令行或 Bash 中使用时需要将问号转义，或者将整个文件名放在单引号或双引号内

4.2.3.2 三维变量的处理

偶尔会遇到三维网格文件，比如地球参考模型，三个维度分别是经度、纬度和深度，模型中的速度和密度等则是一个三维变量。

在遇到多维变量时，GMT 默认会读取第一层（即深度值最小的那一层）数据。可以通过如下两种方法来读取特定层的数据。

1. 文件名后加上 `[<index>]`

   `<index>` 是第三维度变量（比如深度）的索引值，第一层的索引值为 0

2. 文件名后加上 `(<level>)`

   `<level>` 是要获取数据的那一层的深度值。若 `<level>` 指定的深度与网格不重合，则 GMT 会找到离其最近的深度，而不会去做插值

假设有一个地球模型文件，`ncdump -c file.nc` 的结果为（只列出与深度有关的部分）:

```
dimensions:
    depth = 32 ;
variables:
    float depth(depth) ;
    depth:long_name = "depth below earth surface" ;
    depth:units = "km" ;
    depth:positive = "down" ;
data:
    depth = 50, 100, 200, 300, 400, 400, 500, 600, 600, 700, 800, 900, 1000,
```

```
        1100, 1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900, 2000, 2100, 2200,
        2300, 2400, 2500, 2600, 2700, 2800, 2850 ;
```

从中可以看到，该模型在深度方向上有 32 层，分别对应 50 千米、100 千米，一直到 2850 千米。`file.nc?vp[1]` 会读取第二层（即深度 100 km 处）的 P 波速度；而 `file.nc?vp(200)` 会读取深度 200 千米处的 P 波速度。

说明：

1. `ncdump -c file.nc` 命令可以查看 netCDF 网格文件中的信息
2. Linux 下中括号和小括号有特殊含义，因而在命令行或 Bash 中使用时需要进行转义，或者将整个文件名放在单引号或双引号内

4.2.3.3 四维变量的处理

对于四维变量，方法类似。假设有一个四维网格文件，四个维度分别是纬度、经度、深度、时间，变量为压强。利用 `ncdump` 可以查看四个纬度的取值范围：

```
lat(lat): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
lon(lon): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
depth(depth): 0, 10, 20, 30, 40, 50, 60, 70, 80, 90
time(time): 0, 12, 24, 36, 48
pressure(time,depth,lat,lon): 共 10x10x10x5=5000 个值
```

为了得到 depth=10，time=24 处的变量信息，可以用：

```
gmt grdinfo "file.nc?pressure[2,1]"
```

或者：

```
gmt grdinfo "file.nc?pressure(24,10)"
```

需要注意，时间在前，深度在后。

4.2.3.4 一维变量的处理

包含一维变量的 netCDF 文件，也就是前面所说的 netCDF 表。可以通过在文件名后加上变量名来使用一个一维变量，比如：

```
gmt psxy "file.nc?lon/lat" ...
gmt convert "file.nc?time/lat/lon"
```

4.2.3.5 修改坐标单位

某些 GMT 工具要求网格中的两个维度的单位必须是米，若输入数据中的维度的单位不是米，则需要对网格坐标做一些变换。

1. 如果使用的是地理网格数据（即两个维度是经度和纬度），可以加上 `-fg` 选项，则网格坐标会根据 Flat Earth 近似，自动转换成以米为单位。

2. 若使用的是笛卡尔坐标下的网格，但维度的单位不是米（比如是千米），则可以在网格文件名后加上 `+u<unit>` 选项来指定当前网格的维度单位，程序会在内部自动转换成以米为单位。比如，要读入一个维度单位为千米的网格文件，可以通过 `filename+uk` 将其转换成以米为单位。在输出网格时，会自动使用输入数据的原始单位，除非输出网格文件名中有额外的 `+u` 选项。也可以使用 `+U<unit>` 实现逆变换，将以米为单位的网格坐标变成以 `<unit>` 为单位。

4.2.4 网格配准

GMT 中的 2D 网格文件，在确定了网格范围和网格间隔后，网格线会出现在 $x = x_{min}, x_{min} + x_{inc}, x_{min} + 2 \cdot x_{inc}, \ldots, x_{max}$ 和 $y = y_{min}, y_{min} + y_{inc}, y_{min} + 2 \cdot y_{inc}, \ldots, y_{max}$ 处。而节点的位置有两种选择，即网格线配准（gridline registration）和像素配准（pixel registration）。GMT 默认使用的是网格线配准方式。

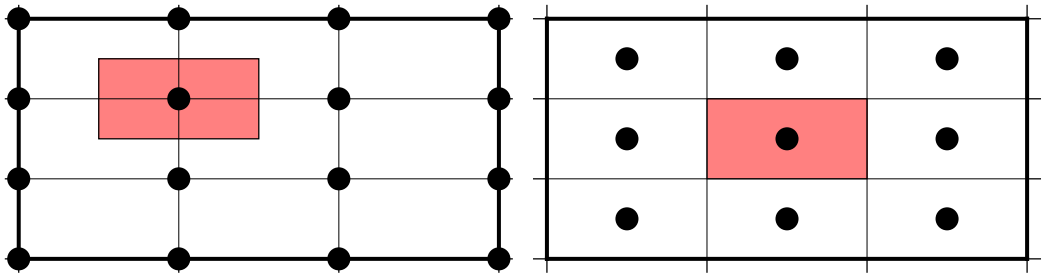


图 4.1: GMT 网格配准方式
（左）网格线配准；（右）像素配准。

4.2.4.1 网格线配准

在网格线配准方式下，节点（图中黑色圆圈）中心位于网格线的交叉点处，节点的值代表了长宽为 $x_{inc} \cdot y_{inc}$ 的单元（图中红色区域）内的平均值。这种情况下，节点数目与网格范围和间隔的关系为：

$$\begin{aligned} nx &= (x_{max} - x_{min})/x_{inc} + 1 \\ ny &= (y_{max} - y_{min})/y_{inc} + 1 \end{aligned}$$

左图中 nx=ny=4。

4.2.4.2 像素配准

在像素配准方式下，节点（图中黑色圆圈）位于网格单元的中心，即网格点之间的区域，节点的值代表了每个单元（图中红色区域）内的平均值。在这种情况下，节点数目与网格范围和间隔的关系为：

$$\begin{aligned} nx &= (x_{max} - x_{min})/x_{inc} \\ ny &= (y_{max} - y_{min})/y_{inc} \end{aligned}$$

因而，对于相同的网格区域和网格间隔而言，像素配准比网格线配准要少一列和一行数据。右图中 nx=ny=3。

4.2.5 边界条件

GMT 中的某些模块在对网格文件做某些操作（比如插值或计算偏导）时，在网格边界处需要指定网格的边界条件。边界条件的选取会影响到区域边界处的计算结果。GMT 中可以通过 `-n` 选项指定网格的边界条件。

GMT 中网格文件的边界条件有三类：

4.2.5.1 默认边界条件

默认的边界条件是：

$$\nabla^2 f = \frac{\partial}{\partial n} \nabla^2 f = 0$$

其中 $f(x, y)$ 是网格文件内的值，$\partial/\partial n$ 是垂直于这个方向的偏导。

$$\nabla^2 = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right)$$

是二维 Laplace 操作符。

4.2.5.2 周期边界条件

X 方向的周期边界条件表明数据是以周期 $x_{max} - x_{min}$ 重复的，数据每 $N = (x_{max} - x_{min})/x_{inc}$ 个点重复一次。Y 方向同理。

- 对于网格线配准的网格文件，共 N+1 列数据。第一列数据位于 $x = x_{min}$ 处，最后一列（N+1 列）数据位于 $x = x_{max}$ 处，周期边界条件意味着数据的第一列和最后一列是完全相同的
- 对于像素配准的网格文件，有 N 列数据，第一列位于 $x_{min} + x_{inc}/2$ ，最后一列（第 N 列）位于 $x_{max} - x_{inc}/2$ ，第一列和最后一列的数据是不同的。

4.2.5.3 地理边界条件

地理边界条件表明：

1. 若 $(x_{max} - x_{min}) \geq 360$ 且 180 是 x_{inc} 的整数倍，则在 X 方向使用周期为 360 的周期边界条件，否则使用默认边界条件
2. 若条件 1 为真，且 $y_{max} = 90$ 则 Y 方向上使用 “北极边界条件”，否则使用默认边界条件
3. 若条件 1 为真，且 $y_{min} = -90$ 则 Y 方向上使用 “南极边界条件”，否则使用默认边界条件

4.2.6 查看 netCDF 文件

某些软件可以直接用于查看 netCDF 文件的内容：

- ncview
- Panoply
- ncBrowse

更多相关工具，见 netCDF 网站上的列表 。

4.3 CPT 文件

在使用 2D 网格数据绘制彩图或灰度图时，需要为每个 Z 值或 Z 值范围指定要使用的颜色。CPT 文件的作用就是为每个 Z 值或 Z 值范围定义其对应的颜色。CPT 全称是 color palette table，也称为调色板或色标文件。

CPT 文件可以在 `grdimage` 、`psxy` 、`psxyz` 等命令中使用。通常，你可以使用 `makecpt` 或 `grd2cpt` 对已有的 CPT（比如 GMT 内置的 CPT）文件进行重采样，并适应你目前所使用的数据范围。当然，也可以自己手写 CPT 文件，或使用 `awk` 、`perl` 之类的文本处理工具自动生成 CPT 文件。

4.3.1 CPT 文件格式

CPT 文件有两种类型，一种适用于分类型数据，一种适用于常规数据。下面会逐一介绍。

4.3.1.1 分类 CPT 文件

分类 CPT 文件适用于有分类型数据，对于这些数据而言，常规的数值操作是未定义的。比如，将陆地分成不同的类型，沙漠、森林、冰川等等，定义沙漠对应的值是 1，森林对应的值是 2，冰川定义的值是 3。显然值取 1.5 是没有意义的。

对于这种分类型数据，需要给每个分类指定一个 `<key>` 。CPT 文件中则规定了每个 `<key>` 所对应的颜色或填充图案，以及一个可选的标签（通常是类型名）。

分类 CPT 文件的格式为:

```
<key1>      <fill1>     [;<label1>]
<key2>      <fill2>     [;<label2>]
...
<keyn>      <filln>     [;<labeln>]
```

几点说明:

1. `<key>` 必须单调递增但不必连续
2. `<fill>` 可以是颜色，也可以是图案，见填充 一节
3. 标签名以分号开头

`<key>` 可以取几个特殊的值:

- `B` ： 小于 `<key1>` 的值的颜色，即背景色，默认值由 *COLOR_BACKGROUND* 控制
- `F` ： 大于 `<keyn>` 的值的颜色，即前景色，默认值由 *COLOR_FOREGROUND* 控制
- `N` ： 值为 `NaN` 所对应的颜色，默认值由 *COLOR_NAN* 控制

下面是一个常规 CPT 文件的示例:

```
0  yellow  ;desert
1  green   ;forest
2  red     ;iceland
```

4.3.1.2 常规 CPT 文件

对于连续变化的数据而言，可以为几个特定值指定颜色，其他值的颜色则通过插值计算得到。此时可以使用常规 CPT 文件。常规 CPT 文件的格式为:

```
<z0>    <color_min_1>  <z1>   <color_max_1>    [<A>]    [;<label>]
<z1>    <color_min_2>  <z2>   <color_max_2>    [<A>]    [;<label>]
...
```

```
<zn-1>   <color_min_n>  <zn>   <color_max_n>    [<A>]    [;<label>]
B   <fill_back>
F   <fill_fore>
N   <fill_NaN>
```

以 CPT 文件中的第一行为例，其定义了一个 Z 值切片，切片范围为 <z0> 到 <z1> ，切片内每个 Z 值的颜色由 <color_min_1> 线性渐变为 <color_max_1> 。其他行同理。几点说明：

1. <z> 值必须单调递增
2. 每行的最大 <z> 必须与下一行的最小 <z> 相同，即 Z 切片之间不能存在间断
3. 若 <color_min_1> 与 <color_max_1> 相等，则 <z0> 到 <z1> 范围内的所有 Z 值均使用相同的颜色
4. 若 <color_min_1> 使用了图案，则 <color_max_1> 必须设置为 -
5. <A> 是可选的，用于表明在使用 psscale 命令绘制色标时要如何标注。<A> 可以取 L 、U 、B ，表示选择每个 Z 切片的下限、上限或者上下限作为标注。
6. ;<lable> 是切片的标签，当 psscale 使用 -L 选项时会在用 <label> 作为标注
7. B|F|N 语句分别用于设置背景色（默认值为*COLOR_BACKGROUND* ）、前景色（默认值为*COLOR_FOREGROUND* ）和 NaN 值的颜色（默认值为*COLOR_NAN* ）
8. B|F|N 语句要放在 CPT 文件的开头或结尾

下面是一个常规 CPT 文件的示例：

```
30    p200/16  80   -
80    -        100  -
100   200/0/0  200  255/255/0
200   yellow   300  green
```

本例中

- 30<z<80：以 200dpi 分辨率的 16 号图案填充
- 80<z<100：直接跳过
- 100<z<200：从深红色线性变化成黄色
- 200<z<300：从黄色线性变化成绿色

4.3.2 GMT 内置 CPT

GMT 内置了 35 个常规 CPT 文件和一个分类 CPT 文件，位于 `$GMTHOME/share/cpt` 目录中。

下图给出了 GMT 内置的 36 个 CPT 文件，每张图上边的色标为原始 CPT，用如下命令绘制:

```
gmt psscale -D5c/2c+w10c/1c+h+jTC -B0 -C<cpt> > test.ps
```

下边的色标是经过 `makecpt` 离散成 8 部分后的色标，用如下命令绘制:

```
gmt makecpt -C<cpt> -T-1/1/0.25 > new.cpt
gmt psscale -D5c/2c+w10c/1c+h+jTC -Bf0.25 -Cnew.cpt > test.ps
```

GMT 模块 `makecpt` 和 `grd2cpt` 可以以这些内置 CPT 文件为基础，针对用户自己的数据制作专门的 CPT 文件。比如某个内置 CPT 文件定义了从 0 到 1 颜色从蓝色变成红色，用 `makecpt` 可以制作一个从 1000 到 3000 颜色从蓝色变成红色的 CPT 文件。

4.3.3 使用 CPT

命令行指定 CPT 文件名后，GMT 会依次在当前目录、`~/.gmt` 和 `$GMTHOME/share/cpt/` 目录下寻找 CPT 文件，如果找不到还会加上后缀 `.cpt` 寻找。

在文件名后加上后缀 `+u|U<unit>` 还可以对 CPT 文件中的 Z 值进行缩放。

- `filename.cpt+u<unit>` 可以将 Z 值从 `<unit>` 变换为以米为单位
- `filename.cpt+U<unit>` 可以将 Z 值从以米为单位变换成 `<unit>`

4.3.4 其他 CPT

更多的 CPT 可以访问：http://soliton.vm.bytemark.co.uk/pub/cpt-city/

4.4 特殊字体

在介绍字体时说过，12 号字体（Symbol）和 34 号字体（ZapfDingbats）比较特殊，这一节专门介绍一下。

能从键盘直接输入的字符是有限的，只有大小写的 26 个字母、0-9 的阿拉伯数字以及几个常见的符号。有时候可能想要输入一些特殊的字符，比如希腊字母。

以字符 `a` 为例，使用除 12 号和 34 号外的其他字体时，显示的永远是字符 `a` ，只是样子可能有些不一样。如果将字符 `a` 设置为 12 号字体，显示的是 α ；如果将字符 `a` 设置为 34 号字体，则

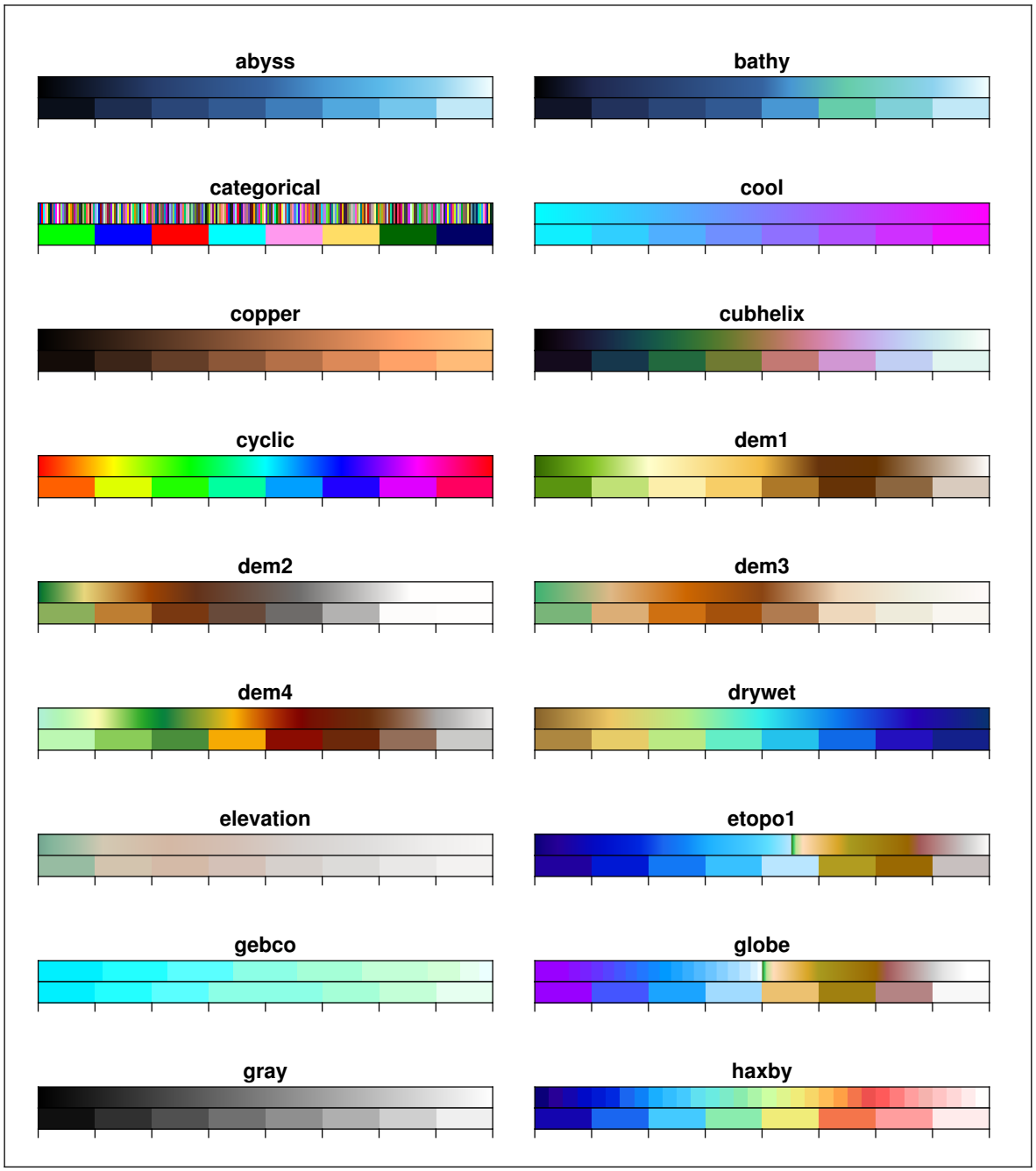


图 4.2: GMT 内置 CPT 示例 1

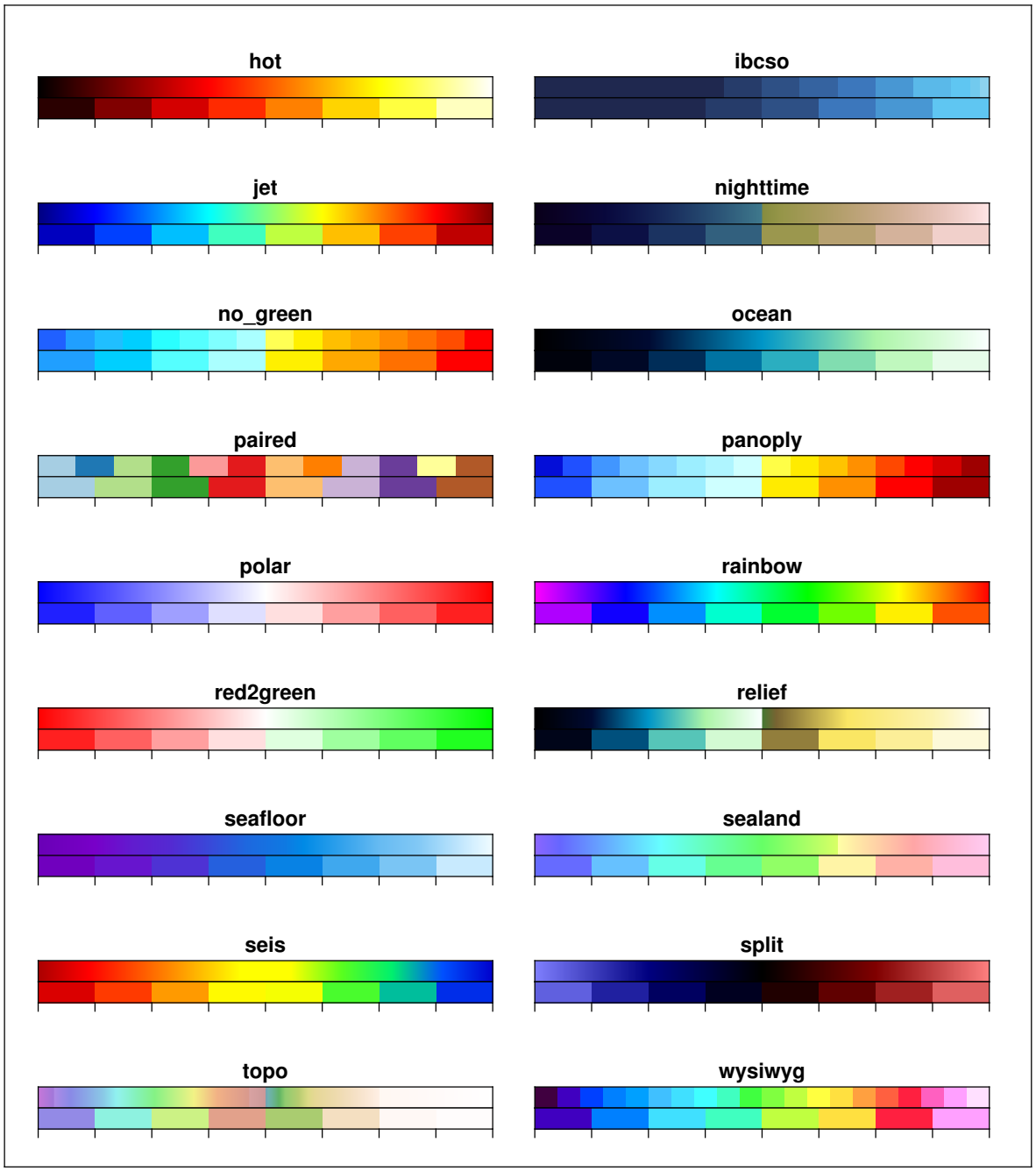


图 4.3: GMT 内置 CPT 示例 2

显示的是一个类似花的图案。本质上，12 号和 34 字体与其他字体并没有什么本质的区别，可以理解为这两个字体显示的时候变化比较大。

由于输入的字符 `a` 与实际显示的字符 α 的差别太大，所以要一一对应就变得很困难。因而在使用 12 号和 34 号字体时通常使用一个三位的八进制码表示一个字符，每个二进制码对应一个字符。

下图给出了 12 号字体（Symbol 字符集）和 34 字体（Pifont ZapfDingbats）的字符与八进制码对应关系：

Symbol

octal	0	1	2	3	4	5	6	7
\04x		!	∀	#	∃	%	&	∍
\05x	(	)	*	+	,	−	.	/
\06x	0	1	2	3	4	5	6	7
\07x	8	9	:	;	<	=	>	?
\10x	≅	Α	Β	Χ	Δ	Ε	Φ	Γ
\11x	Η	Ι	ϑ	Κ	Λ	Μ	Ν	Ο
\12x	Π	Θ	Ρ	Σ	Τ	Υ	ς	Ω
\13x	Ξ	Ψ	Ζ	[	∴	]	⊥	–
\14x	‾	α	β	χ	δ	ε	ϕ	γ
\15x	η	ι	φ	κ	λ	μ	ν	ο
\16x	π	θ	ρ	σ	τ	υ	ϖ	ω
\17x	ξ	ψ	ζ	{	|	}	~	

\24x	∈	ϒ	′	≤	⁄	∞	ƒ	♣
\25x	♦	♥	♠	↔	←	↑	→	↓
\26x	°	±	″	≥	×	∝	∂	•
\27x	÷	≠	≡	≈	…	|	—	↵
\30x	ℵ	ℑ	ℜ	℘	⊗	⊕	∅	∩
\31x	∪	⊃	⊇	⊄	⊂	⊆	∈	∉
\32x	∠	∇	®	©	™	∏	√	·
\33x	¬	∧	∨	⇔	⇐	⇑	⇒	⇓
\34x	◊	〈	®	©	™	∑	⎛	⎜
\35x	⎝	⎡	⎜	⎣	⎛	⎧	⎝	⎜
\36x		〉	∫	⌠	⎮	⌡	⎞	⎟
\37x	⎠	⎤	⎟	⌡	⎞	⎫	⎠	

ZapfDingbats

octal	0	1	2	3	4	5	6	7
\04x		✁	✂	✃	✄	☎	✆	✇
\05x	✈	✉	☛	☞	✌	✍	✎	✏
\06x	✐	✑	✒	✓	✔	✕	✖	✗
\07x	✘	✙	✚	✛	✜	✝	✞	✟
\10x	✠	✡	✢	✣	✤	✥	✦	✧
\11x	★	✩	✪	✫	✬	✭	✮	✯
\12x	✰	✱	✲	✳	✴	✵	✶	✷
\13x	✸	✹	✺	✻	✼	✽	✾	✿
\14x	❀	❁	❂	❃	❄	❅	❆	❇
\15x	❈	❉	❊	❋	●	❍	■	❑
\16x	❒	❑	❒	▲	▼	◆	❖	◗
\17x	❘	❙	❚	❛	❜	❝	❞	

\24x		❡	❢	❣	❤	❥	❦	❧
\25x	♣	♦	♥	♠	①	②	③	④
\26x	⑤	⑥	⑦	⑧	⑨	⑩	❶	❷
\27x	❸	❹	❺	❻	❼	❽	❾	❿
\30x	①	②	③	④	⑤	⑥	⑦	⑧
\31x	⑨	⑩	❶	❷	❸	❹	❺	❻
\32x	❼	❽	❾	❿	➔	→	↔	↕
\33x	➘	➙	➚	➛	➜	→	➙	➟
\34x	➠	➡	➢	➢	➤	➥	➦	➧
\35x	➨	➩	➪	➫	➬	➩	➩	➭
\36x		➮	➲	➳	➴	➵	➶	➷
\37x	➸	➹	➺	➻	➼	➽	➾	

图 4.4: Symbol 和 Pifont 字体八进制码

比如需要字符 β ，则首先设置文本的字体为 12 号字体，查上表可知，字符 β 位于 `\14x` 行 2 列，因而其对应的八进制码为 `\142` 。同理，θ 的八进制码为 `\161` 。同样是八进制码 `\161` ，如果设置字体为 34 号，则是右边图中对应的类似矩形的符号了。

下图给出了一些示例：

Input	Font	Output
abcd	10p,12,black	αβχδ
\141\142\143\144	10p,12,black	αβχδ
\141\142\143\144	15p,34,red	❀✪✳❄
\243\251\256\303	15p,34,blue	❣♦③④

图 4.5: 12 号和 34 号字体示例

对于 26 个希腊字母的大小写而言，由于其与 26 个英语字母的大小写一一对应，所以如果熟悉的话，完全可以用英文字母写文本，然后设置字体为 12 号字体，如第一行所示，这样的做法比第二行用八进制码 1 要简单些。但是对于其他特殊字符，由于不存在这种有规律的对应关系，所以最好使用八进制码表示。

4.5 特殊字符

所有的字符在计算机中都是用二进制表示的，不过二进制看起来比长，所以也可以用等效的八进制或十六进制表示。GMT 中用三位的八进制码表示一个字符。

对于任意一个三位的八进制码，可以对应字符 `a` ，也可以对应字符 `f` ，具体哪个八进制码对应哪个字符呢？这就由字符编码规范决定。

GMT 支持多种字符编码，包括 `Standard` 、`Standard+` 、`ISOLation1` 、`ISOLatin1+` 以及 `ISO-8859-x` （x 可以取 1 到 10 以及 13 到 15）。

GMT 当前使用的字符编码方式由 GMT 参数*PS_CHAR_ENCODING* 决定。若安装过程中使用的是国际单位制，则默认编码方式为 `ISOLatin1+` ，否则为 `Standard+` 。可以使用:

```
gmt get PS_CHAR_ENCODING
```

查看 GMT 当前使用的字符编码。

下图给出了 `Standard+` 和 `ISOLatin1+` 编码下八进制码与字符间的对应关系：

其中，浅红色区域是保留给控制字符的八进制码，浅绿色区域是扩展字符（扩展字符即 `Standard+` 编码相对于 `Standard` 编码多出的字符，`ISOLation1+` 同理。）。

这张图应该如何读呢？以 `ISOLation1+` 编码下的八进制码 `\144` 为例，`\14x` 行与 4 列的交界处就是该八进制码代表的字符，即 `d` 。

下面的示例展示了修改 GMT 的文本编码，以及不同编码下同一八进制码的效果:

Standard+

octal	0	1	2	3	4	5	6	7
\03x		¾	³	™	²	ý	ÿ	ž
\04x		!	"	#	$	%	&	’
\05x	(	)	*	+	,	-	·	/
\06x	0	1	2	3	4	5	6	7
\07x	8	9	:	;	<	=	>	?
\10x	@	A	B	C	D	E	F	G
\11x	H	I	J	K	L	M	N	O
\12x	P	Q	R	S	T	U	V	W
\13x	X	Y	Z	[	\	]	^	–
\14x	‘	a	b	c	d	e	f	g
\15x	h	i	j	k	l	m	n	o
\16x	p	q	r	s	t	u	v	w
\17x	x	y	z	{	|	}	~	ƒ
\20x	Ã	Ç	Đ	Ł	Ñ	Õ	Š	Þ
\21x	Ý	Ÿ	Ž	ã	¦	ç	©	°
\22x	÷	ð	¬	ł	−	µ	×	ñ
\23x	½	¼	¹	õ	±	®	š	þ
\24x		¡	¢	£	⁄	¥	ƒ	§
\25x	¤	'	“	«	‹	›	fi	fl
\26x	Á	–	†	‡	·	Â	¶	•
\27x	‚	„	”	»	…	‰	Ä	¿
\30x	À	`	´	ˆ	~	¯	˘	·
\31x	¨	É	°	¸	Ê	˝	˛	ˇ
\32x	—	Ë	È	Í	Î	Ï	Ì	Ó
\33x	Ô	Ö	Ò	Ú	Û	Ü	Ù	á
\34x	â	Æ	ä	ª	à	é	ê	ë
\35x	è	Ø	Œ	º	í	î	ï	ì
\36x	ó	æ	ô	ö	ò	ı	ú	û
\37x	ü	ø	œ	ß	ù	Å	å	ÿ

ISOLatin1+

octal	0	1	2	3	4	5	6	7
\03x		•	…	™	—	−	fi	ž
\04x		!	"	#	$	%	&	’
\05x	(	)	*	+	,	−	·	/
\06x	0	1	2	3	4	5	6	7
\07x	8	9	:	;	<	=	>	?
\10x	@	A	B	C	D	E	F	G
\11x	H	I	J	K	L	M	N	O
\12x	P	Q	R	S	T	U	V	W
\13x	X	Y	Z	[	\	]	^	–
\14x	‘	a	b	c	d	e	f	g
\15x	h	i	j	k	l	m	n	o
\16x	p	q	r	s	t	u	v	w
\17x	x	y	z	{	|	}	~	š
\20x	Œ	†	‡	Ł	⁄	‹	Š	›
\21x	œ	Ÿ	Ž	ł	‰	„	“	”
\22x	ı	`	´	ˆ	~	¯	˘	·
\23x	¨	¸	°	˛	'	˝	˛	ˇ
\24x		¡	¢	£	¤	¥	¦	§
\25x	¨	©	ª	«	¬	-	®	¯
\26x	°	±	²	³	´	µ	¶	·
\27x	¸	¹	º	»	¼	½	¾	¿
\30x	À	Á	Â	Ã	Ä	Å	Æ	Ç
\31x	È	É	Ê	Ë	Ì	Í	Î	Ï
\32x	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	×
\33x	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
\34x	à	á	â	ã	ä	å	æ	ç
\35x	è	é	ê	ë	ì	í	î	ï
\36x	ð	ñ	ò	ó	ô	õ	ö	÷
\37x	ø	ù	ú	û	ü	ý	þ	ÿ

图 4.6: Standard+ 和 ISOLatin1+ 编码下的八进制码

```
$ gmt set PS_CHAR_ENCODING Standard+
$ echo 1 1 '\260' | gmt pstext -JX2c/2c -R0/2/0/2 -B1 > standard.ps
$ gmt set PS_CHAR_ENCODING ISOLatin1+
$ echo 1 1 '\260' | gmt pstext -JX2c/2c -R0/2/0/2 -B1 > isolatin1.ps
```

其中，standard.ps 中显示的是类似 A 的字符，而 isolation1.ps 中显示的则是弧度 ° 符号。

需要注意，反斜杠在 GMT 中用于表示八进制码，同时，反斜杠也是大多数脚本语言的转义字符。因而会存在反斜杠先被脚本语言转义再传递给 GMT 的情况。故而此处需要用 '\260' 或 \\260 ，而不能直接用 \260 。

4.6 转义序列

前面已经介绍了如何设置字体属性以及如何输出特殊字符，但是这还不够，无法实现上标、下标，无法在一个字符串内随意切换字体和颜色。由此，GMT 引入了转义字符，给文本加入了更丰富的效果。

在 C 语言中，转义字符是 \ ；在 GMT 中，转义字符是 @ 。转义字符与一般字符合在一起构成了转义序列。GMT 可以识别的转义字符在下表列出：

表 4.2: GMT 转义字符

转义字符	说明
@~	打开/关闭 Symbol（12 号）字体
@+	打开/关闭上标
@-	打开/关闭下标
@#	打开/关闭大型小写字母
@_	打开/关闭下划线
@%<fontno>%	切换至另一字体；@%% 重置回前一字体
@:<size>:	切换至另一文本尺寸；@:: 重置回前一尺寸
@;<color>;	切换至另一文本颜色；@;; 重置回前一颜色
@!	用接下来的两个字符创建组合字符
@@	输出 @ 符号自身

对于一些常用的欧洲特殊字符，可以通过设置*PS_CHAR_ENCODING* 来实现，GMT 为其中的部分常见字符提供了转义序列，如下表：

表 4.3: 欧洲特殊字符

Code	*Effect*	*Code*	*Effect*
`@E`	Æ	`@e`	æ
`@O`	Ø	`@o`	ø
`@A`	Å	`@a`	å
`@C`	Ç	`@c`	ç
`@N`	Ñ	`@n`	ñ
`@U`	Ü	`@u`	ü
`@s`	ß	`@i`	í

4.6.1 转义字符示例

Input	Output
`abc@~def@~ghi`	abcdδεϕghi
`2@~p@~r@+2@+h@-0@-`	$2\pi r^2 h_0$
`S@#mall@# C@#aps@#`	SMALL CAPS
`Thi is @_underline@_`	This is underline
`@%1%Use@%% @%23%different@%% @%8%fonts@%%`	**Use** *different* `fonts`
`@:10:Use@:: @:20:different@:: @:15:size@::`	Use different size
`@;red;Colorful@;; @;blue;text@;;`	Colorful text
`@!CV @@`	𝕍 @
`Stresses are @~s@~@+*@+@-xx@- MPa`	Stresses are σ^*_{xx} MPa

图 4.7: GMT 转义序列示例

4.6.2 注意事项

1. 上标/下标不支持嵌套，即只支持一层上标/下标
2. `pstext` 命令中有选项可以在文本周围加上矩形框，该选项对转义序列无效
3. 转义序列需要成对存在，与括号类似，开启转义之后必须关闭转义；
4. 在 Windows 下，由于 bat 脚本中 `%` 表示变量，因此当你需要在 GMT 中使用百分号时，应使用 `%%` 来表示一个百分号，即 bat 脚本中的 `%%` 相当于字符 `%`；切换字体时 `@%%15%%` 相当于正常情况下的 `@%15%`；

4.7 线条属性

画笔 一节介绍了画笔的三个属性：线宽、颜色以及线型。对于线条而言，除了画笔的这三个属性之外，某些模块还可以为线条设置额外的属性，这些额外的属性可以通过在画笔属性后加上子选项来是实现。

线条的额外属性包括：端点偏移量、线条平滑和端点箭头。

4.7.1 端点偏移量

在给定若干个数据点绘制线条时，一般都是从起点（第一个点）一直画到终点（最后一个点）。可以为使用 `+o` 子选项为线段两端指定偏移量，使得绘制线段时的起点和终点与输入数据中指定的起点和终点间存在一定的偏移量。该子选项的语法是:

```
+o<offset>[<u>]
```

- 若只给了一个 `<offset>` ，则表示起点和终点共用同一个偏移量
- 也可以用 `<offset>/<offset>` 分别为起点和终点指定不同的偏移量
- 对于每个偏移量，都可以使用长度单位或距离单位

图中，细线和粗线使用了完全相同的输入数据，其中细线没有使用 `+o` 选项的结果，此时线段的起点和终点与数据指定的点重合，粗线在绘制线条时使用了 `-W2p+o1c/500k` 选项，即在起点处偏移 1 厘米，在终点处偏移 500 千米。

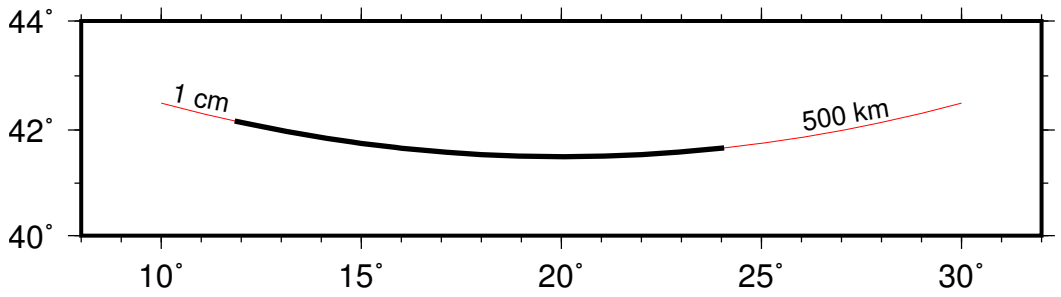


图 4.8: 线段起点偏移示意图

4.7.2 线条平滑

通常情况下，在绘制线条时，两点之间是用直线的（地图上两点之间默认用大圆弧连接。）使用 `+s` 子选项会使用 Bezier splines 在两点之间做样条插值以得到更光滑的曲线。

下图中，左图使用了 `-W2p` 选项，右边使用了 `-W2p+s` 。

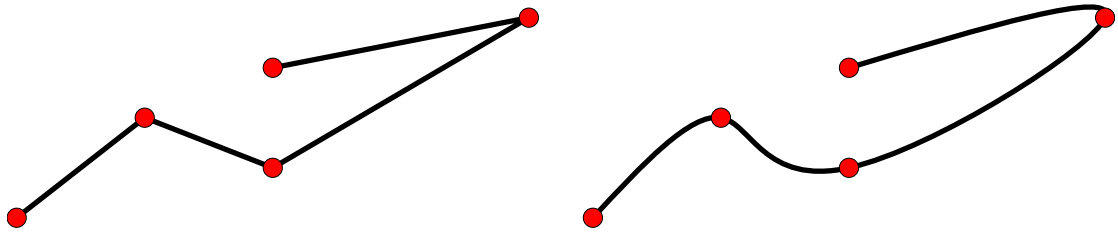


图 4.9: 线条自动样条插值示意图

4.7.3 端点箭头

默认情况下，在绘制线段时，线段的两个端点并没有什么特别的。使用 `+v` 子选项，可以在线段的一端或两段绘制一个指向端点的箭头。该子选项的语法为:

```
+v[b|e]<vspecs>
```

- 默认会在线段两端都加上箭头，`b|e` 表示只绘制开头或结尾的箭头
- 箭头属性 `<vspecs>` 的细节参考绘制矢量 一节的内容

下图中细线是通常绘制的线段,粗线使用的选项是 `-W2p+o1c/500k+vb0.2i+gred+pfaint+bc+ve0.3i+gblue` 。

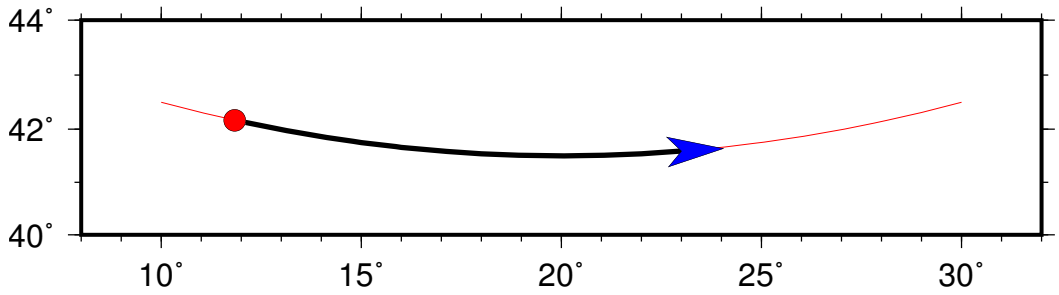


图 4.10: 线条端点箭头示意图

4.8 绘制矢量

GMT 中能够绘制矢量的模块包括 `psxy` 、`psxyz` 、`grdvector` 和 `psvelo` 等。除此之外，其他能够绘制线段的模块也会间接绘制矢量。

所有的矢量都可以分成两个独立的部分:

1. 矢量线部分，由画笔属性（画笔 ）和线条属性（线条属性 ）控制
2. 矢量箭头部分，具体属性接下来会介绍

4.8.1 矢量分类

GMT 中的矢量可以分为三类：

1. 笛卡尔矢量：矢量线是直线。可以通过如下几种方式指定
   (a) 给定起点和终点
   (b) 给定起点、方向和长度（使用长度单位）
   (c) 给定起点、方位角和长度（单位为 km）
2. 弧形矢量：矢量线是圆弧，通过给定圆心、半径以及圆弧起始和结束的角度来指定
3. 地理矢量：矢量线是大圆弧，通过两种方式指定
   (a) 给定起点和终点
   (b) 给定起点、方位角和长度（单位为 km）

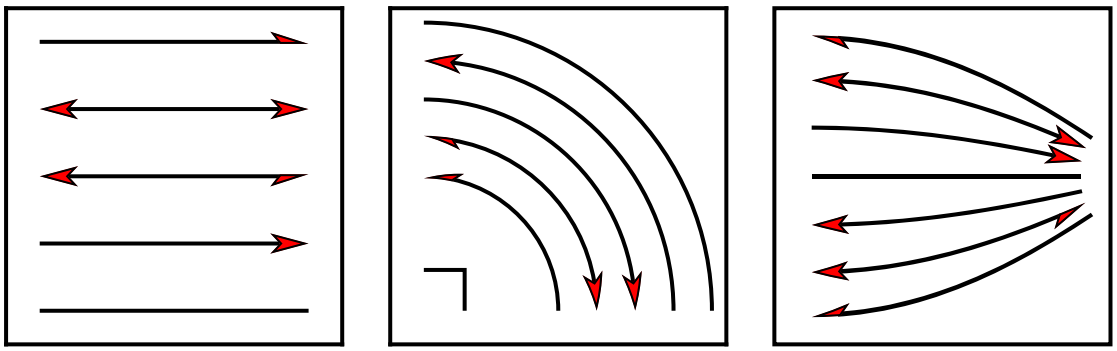


图 4.11: GMT 中的三种矢量箭头
（左）笛卡尔矢量；（中）弧形矢量；（右）地理矢量

4.8.2 矢量箭头属性

可以为矢量相关选项（比如 `psxy` 的 `-Sv` 和 `-Sm` 选项）加上额外的子选项来控制矢量箭头的属性。

绘制矢量时，默认只有矢量线没有矢量头，可以使用下面几个子选项为矢量加上矢量头：

- `+b[t|c|a][l|r]` ： 在矢量线的首端加上矢量头。`t|c|a` 用于控制矢量头的形状，分别表示矢量头为终点线（垂直于示例线的短线）、圆圈或者箭头，默认是箭头。`l|r` 表示只绘制矢量的头的左半边或右半边（默认两边都绘制）
- `+e[t|c|a][l|r]` ： 在矢量线的尾端加上矢量头。其他同上
- `+m[f|r][t|c|a][l|r]` ： 在矢量线的中间加上矢量头。`f|r` 表示矢量头沿着正方向或逆方向（默认值从首端指向尾端），其他同上。注意，`+m` 不能与 `+b` 或 `+e` 一起使用

下面的子选项可以进一步控制矢量头的外观：

- `+a<angle>` 控制矢量箭头顶端的夹角，默认值为 30 度
- `+l` 表示只绘制左半个箭头
- `+r` 表示只绘制右半个箭头
- `+g<fill>` 设置箭头填充色，`<fill>` 为 `-` 表示不填充（该选项似乎无效）
- `+p[-][<pen>]` 设置箭头轮廓的画笔属性（该选项似乎无效）

还有其他一些子选项：

- `+n<norm>` 默认情况下，矢量头的大小不随着矢量线的长度变化而变化，这可能会出现矢量长度太小时矢量头过大的情况。该选项使得矢量长度小于 `<norm>` 时，矢量头的属性（画笔宽度，箭头大小）会根据矢量长度按照 `length/<norm>` 缩放
- `+t[b|e]<trim>` 将矢量的首端或尾端偏移一定长度，`<trim>` 为正值表示矢量长度变短，为负值表示矢量长度变长。还可以使用 `+t<trim1>/<trim2>` 语法分别为首端和尾端指定偏移量
- `+o<plon>/<plat>` 见官方文档
- `+q` 表明输入数据中的 `angle` 和 `length` 被解释为矢量的开始和结束的角度

除了弧形矢量外，其他矢量还可以使用如下子选项：

- `+j<just>` 默认情况下，输入数据中的 XY 坐标会作为矢量的首端坐标，该选项可以修改这一行为，`<just>` 可以取 `b|e|c` 分别代表首端、尾端和中间
- `+s` 表明输入数据中的 `angle` 和 `length` 被解释为矢量的尾端 XY 坐标

对于笛卡尔矢量而言，还可以使用：

- `+z<scale>[<unit>]` 见官方文档

4.8.3 矢量示例

下图展示了部分 GMT 可以绘制的矢量。

4.9 锚点

要将一个矩形物体放在指定的坐标处，该怎么放呢？是把矩形中心放在指定坐标处？还是把矩形左上角放在坐标处？最好能够控制将矩形的任意一点放在指定坐标处。

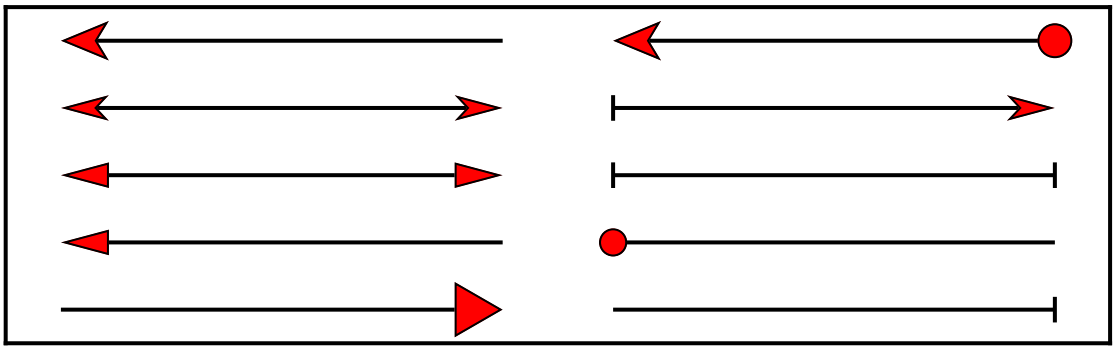


图 4.12: 矢量箭头类型

GMT 提供了这样一种机制，可以很方便地将矩形的任意一点放在指定坐标处。这里所说的矩形，并不是一个真正的矩形，而是一串文本、比例尺、指南针、颜色条等等。这些东西都可以被抽象成一个矩形，这一节仅以字符串为例，介绍 GMT 中矩形的锚点。

任意一个字符串，都有一个将其包起来的矩形。如图所示：

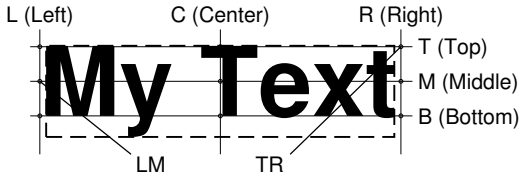


图 4.13: GMT 文本锚点

如上图所示，在水平方向上，定义 `L` 、`C` 、`R` 三个水平位置代码，表示左、中、右三个水平位置；在垂直方向上，定义 `T` 、`M` 、`B` 三个垂直位置代码，代表上、中、下三个垂直位置（图中 `B` 代表的是字符 `M` 的底部，而不是字符 `y` 的字体，这在印刷行业里称为 baseline）。

将任意一个水平位置代码与任意一个垂直位置组合起来，就构成了矩形的 9 个锚点（图中的 9 个圆圈），分别是 `LT` 、`LM` 、`LB` 、`CT` 、`CM` 、`CB` 、`RT` 、`RM` 、`RB` 。先后顺序不重要，`LT` 和 `TL` 是一个意思。

要指定一串文本相对于特定坐标的对齐方式，可以通过指定将文本的某个锚点放在特定坐标处来实现。下图展示了 9 种文本对齐方式的效果。图中将九个字符串按照相应的对齐方式放在坐标 `(1,1)` 、`(2,1)` 到 `(9,1)` 处，即将字符 `LT` 的 `LT` 锚点放在 `(1,1)` 处，其他同理。

从图中可以直观得看到，`CM` 对齐方式将字符串的中心放在了指定坐标处，`LT` 对齐方式将字符串的左上角放在了指定坐标处，其他同理。

4.10 绘制修饰物

GMT 除了可以绘制常规的数据外，还可以绘制 8 种修饰物：地图比例尺、方向玫瑰图、磁场玫瑰图、颜色条、图例、Image overlay、GMT logo、map insert。

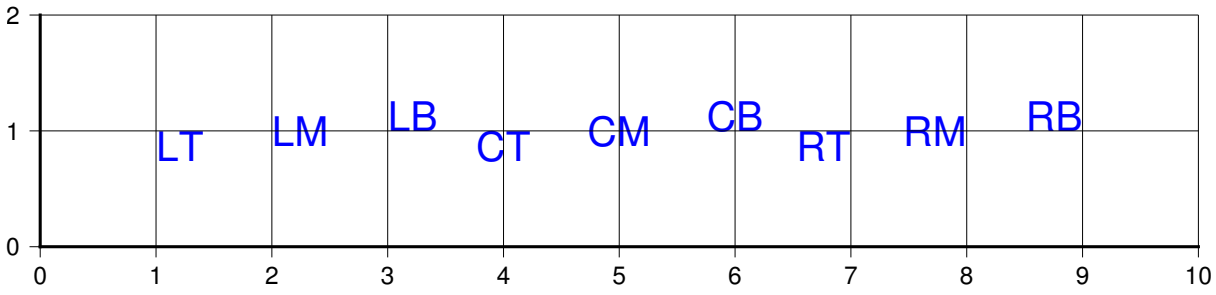


图 4.14: GMT 文本对齐方式

8 种修饰物的绘制有各自不同的语法，但这些修饰物都可以抽象为一个矩形。在如何指定修饰符在图上的位置方面，拥有共同的语法。

下图中，大矩形代表底图，小矩形代表修饰物。要将一个修饰物放在底图上，首先要在底图上指定一个参考点，然后在修饰物上找到一个锚点，将修饰物的锚点放在图上的参考点上即可。

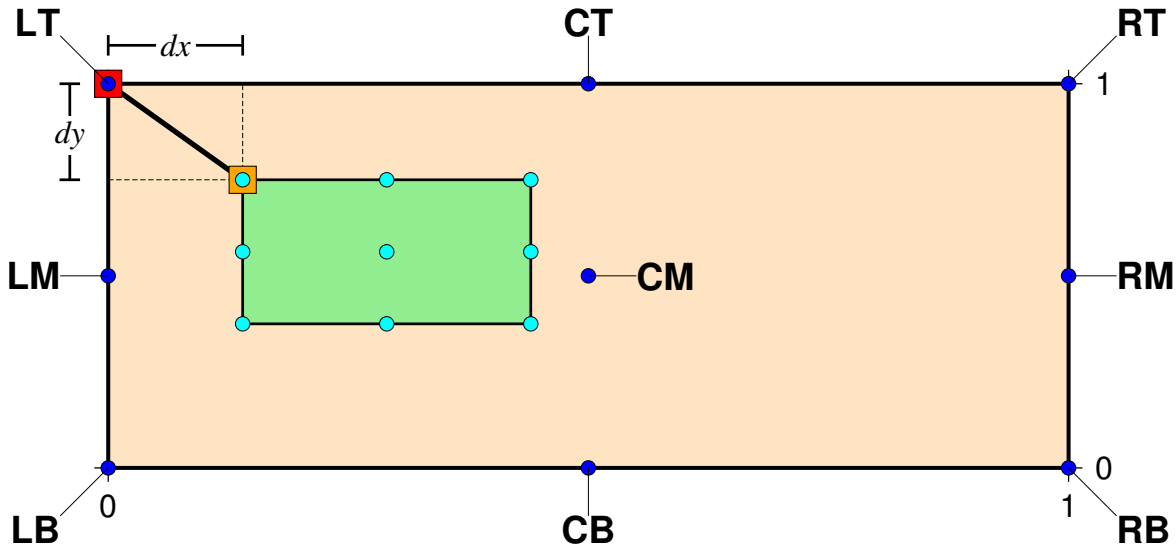


图 4.15: 参考点与锚点

4.10.1 参考点

可以用五种方式在图上指定一个参考点，其语法为:

```
[g|j|J|n|x]<refpoint>
```

1. `g` 用数据坐标指定参考点，比如 `g135W/20N` 表示参考点是西经 135 度北纬 20 度
2. `j` 指定底图的某个锚点作为参考点。关于锚点，见锚点 一节，在这种方式下，修饰物的锚点也默认使用相同的对齐方式代码。比如 `jTL` 将底图的左上角作为参考点，同时修饰物的锚点也是左上角，此时修饰物的左上角与底图左上角重合

3. `J` 与 `j` 类似，唯一的区别在于，修饰物的锚点默认使用镜像相反的锚点。比如 `JTL` 将底图的左上角作为参考点，同时修饰物的锚点是 `BR` 即右下角
4. `x` 用绘图坐标指定参考点。即相对于绘图原点的偏移距离。例如 `x2.75i/2c`
5. `n` 用归一化坐标指定参考点，X 方向和 Y 方向的坐标范围都是 0 到 1。比如 `n0.2/0.1`

若未指定使用何种方式，则默认使用 `x` 指定参考点。

4.10.2 锚点

在底图上指定了参考点之后，还需要在修饰物上指定锚点。可以使用子选项 `+j<anchor>` 来选定锚点。

在不使用该选项的情况下，若使用 `j<refpoint>` 指定参考点，则锚点与参考点使用相同的代码；若使用 `J<refpoint>` 指定参考点，则锚点使用与参考点呈镜像相反的代码；对于其他指定参考点的方式而言，锚点默认为 `MC` （map rose 和 map scale）或 `BL` （其他修饰物）。

4.10.3 偏移量

在指定参考点后，还可以对参考点指定额外的偏移量，尤其是使用 `j` 和 `J` 时。

可以使用 `+o<dx>/<dy>` 选项指定参考点的额外偏移量，并且正偏移量所对应的偏移方向与两字符锚点代码所指定的方向相同，比如使用 `jBL` 时，`+o1c/1c` 表示首先将参考点放在左下角，在此基础上，向左下角偏移量 `1c/1c` 。

4.10.4 背景面板

可以在这些修饰物的后面加上背景面板。面板的位置和大小由修饰物决定，除此之外，面板还有一些其他属性。通常，面板的属性由 `-F` 选项的子选项决定：

1. `+g<fill>` 指定填充色，默认不填充
2. `+p<pen>` 指定边框画笔属性，默认使用*MAP_FRAME_PEN* 的值
3. `+r<radius>` 绘制圆角边框，`<radius>` 为圆角的半径
4. `+i<gap>/<pen>` 在边框内部绘制一个内边框，`<gap>` 是内外边框的空白距离，`<pen>` 为内边框的画笔属性，比如 `+i0.1c/thin,dahsed` ，默认使用*MAP_DEFAULT_PEN*
5. `+c<clearance>` 默认情况下面板的大小由修饰物的大小决定，可以使用该子选项为面板增加额外的尺寸。
    - `+c<gap>` 为四个方向增加相同的空白距离

- +c<xgap>/<ygap> 分别为 X 方向和 Y 方向指定不同的空白距离
- +c<lgap>/<rgap>/<bgap>/<tgap> 分别为四个方向指定不同的空白距离

6. +s<dx>/<dy>/<shade> 下拉阴影区。<dx>/<dy> 是阴影区相对于面板的偏移量，<shade> 是阴影区的颜色，默认值为 4p/-4p/gray50 。

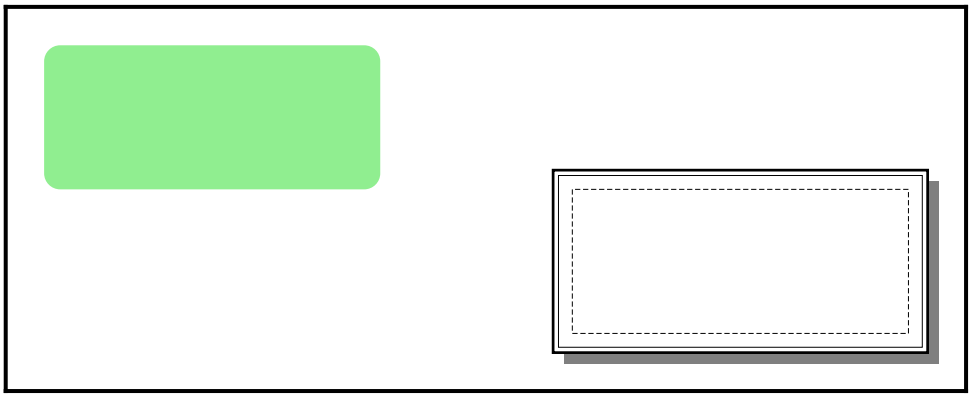


图 4.16: GMT 修饰物背景面板

左图使用了 -F+glightgreen+r ，右图使用了 -F+p1p+i+s+gwhite+c0.1i

4.11 命令行历史

在使用多个命令绘制一张图时，这些命令可能使用了完全相同的标准选项参数，如果对于每个命令都需要把标准选项的参数写一遍，则太麻烦了，在这种情况下，GMT 允许你省略标准选项的参数。例如:

```
gmt psbasemap -JX10c/10c -R0/10/0/10 -B1 -K > text.ps
gmt psxy station.dat -J -R -O >> text.ps
```

GMT 每个命令在执行的时候，会将当前命令所使用的标准选项的参数保存到命令行历史文件 gmt.history 中，比如上面第一个命令指定了 -JX10c/10c -R0/10/0/10 ，会生成如下内容:

```
# GMT 5 Session common arguments shelf
BEGIN GMT 5.1.2
B   1
J   X
JX  X10c/10c
R   0/10/0/10
L   1
END
```

在执行第二个命令时使用了 -J -R ，省略了具体的参数值，此时命令会读取 gmt.history 文件中需要的参数值。

抛开具体的细节不看，可以认为后面的命令继承了前面命令的标准选项参数，如果之后的命令显式指定了标准选项的参数，则后面命令中的参数会覆盖 `gmt.history` 中保存的值。

需要注意，如果在一个命令中通过管道或其他手段调用了多个 GMT 模块，GMT 无法保证 `gmt.history` 文件是按照从左到右的顺序处理的。

不建议依赖命令行历史这一特性，建议将常用的标准选项的参数定义成变量使用。

4.12 特殊目录

GMT 定义了一些环境变量和 GMT 参数，用于指定某些特殊用途的目录。这些环境变量和参数包括：

- `$GMT_SHAREDIR` GMT 的 share 目录所在位置，通常不用设置，GMT 会自动猜测其所在位置
- `$GMT_DATADIR` 或*DIR_DATA* 可以指向一个或多个目录，常用户放置常用的数据文件。目录之间用冒号分隔（Windows 下用分号分隔）。任何以斜杠 `/` 结尾的目录会被递归搜索（Windows 下不会）。若二者同时有值，以*DIR_DATA* 的值优先
- `$GMT_USERDIR` 用户放置自定义配置文件的地方，比如用户自定义的 `gmt.conf` 文件、自定义符号、CPT 文件、数学宏、网格文件后缀文件等。若该变量未定义，则默认值为 `$`*HOME*`/.gmt` 。
- `$GMT_TMPDIR` 临时文件（比如 `gmt.history` 和 `gmt.conf` ）放置的目录。若未设置，则默认为当前目录
- *DIR_DCW* DCW 数据放置的目录
- *DIR_GSHHG* 海岸线数据放置的目录

当命令行中有文件需要读入时，GMT 不仅仅会在当前目录下寻找文件，还会到这些特殊变量中寻找。GMT 会**依次**到下列目录中寻找文件：

1. 当前目录
2. GMT 参数*DIR_DATA* 所定义的目录
3. 环境变量 `$GMT_DATADIR` 所定义的目录
4. 环境变量 `$GMT_USERDIR` 所定义的目录

第五章　标准选项

GMT 模块众多，每个模块的具体效果，由模块的众多选项来决定。不同的模块有不同的选项，这其中有一些选项是通用选项，或称为标准选项，即这些选项在所有的命令里都具有完全相同的意义。故而把这些通用的选项单独拿出来介绍。

GMT 中的选项都是以 - 加一个字符的形式构成，通常这个字符是经过精心挑选的，使得用户很容易根据字符记住该选项的作用。

下面会介绍 GMT 中的通用选项的用法，由于这些选项在所有 GMT 模块中的用法是一样的，所以在介绍 GMT 模块时不会再出现介绍这些选项。

表 5.1: GMT 标准选项

-B	定义底图边框和轴的刻度、标注、标签等属性
-J	选择地图投影方式或坐标变换
-K	省略 PS 文件尾以追加更多 PS 代码
-O	省略文件头以将 PS 代码追加到已有的文件中
-P	设置纸张方向为 Portrait 模式
-R	指定区域范围
-U	在图上绘制时间戳
-V	详细报告模式
-X	移动 X 方向上的绘图原点
-Y	移动 Y 方向上的绘图原点
-a	将非空间数据与某些列联系在一起
-b	控制二进制的输入或输出
-c	设置当前 PS 文件打印时的份数
-d	将输入或输出中的 *nodata* 替换成 NaN
-f	设置 ASCII 输入或输出的格式
-g	根据数据间断对数据进行分段
-h	跳过数据的文件头段记录
-i	选择输入列
-n	设置网格插值方式
-o	选择输出列
-p	控制 3D 视角图
-r	设置网格配准方式
-s	控制 NaN 记录的处理方式
-t	设置图层透明度
-x	设置并行的核数（仅限于支持并行的模块）
-:	输入数据是 y/x 而不是 x/y

5.1 -R 选项

-R 选项用于指定要绘制的数据范围或地图区域。

5.1.1 四种方法

可以用四种方式指定数据范围。

1. -R<xmin>/<xmax>/<ymin>/<ymax>

通过给定 X 方向和 Y 方向的最大最小值来指定数据范围，例如 `-R0/360/-90/90` 表示 X 方向范围是 0 到 360，Y 方向范围是-90 到 90。

2. `-R<xlleft>/<ylleft>/<xuright>/<yuright>r`

   通过给定矩形区域的左下角坐标（`xlleft,ylleft`）和右上角坐标（`xuright,yuright`）来指定数据范围，例如 `-R-90/20/-65/30r` 。这种形式通常用于倾斜的地图投影中，此时不适合将经线和纬线作为地图边界。

3. `-R<gridfile>`

   `<gridfile>` 为网格文件名，即从该网格文件中读取范围信息。对于某些命令，这种方式不仅会从网格中读取范围信息 `-R` ，还会读入网格间隔 `-I` 和网格配准信息。

4. `-R<code><x0>/<y0>/<nx>/<ny>`

   该方法可以在创建网格文件时使用。`<code>` 用于指定网格区域的锚点（见锚点 一节）。`<code>` 所指定的锚点对应的坐标为（`x0,y0`），`<nx>` 、`<ny>` 是 X 和 Y 方向的网格数，通过 `-I` 选项指定的网格间隔即可计算得到区域范围。

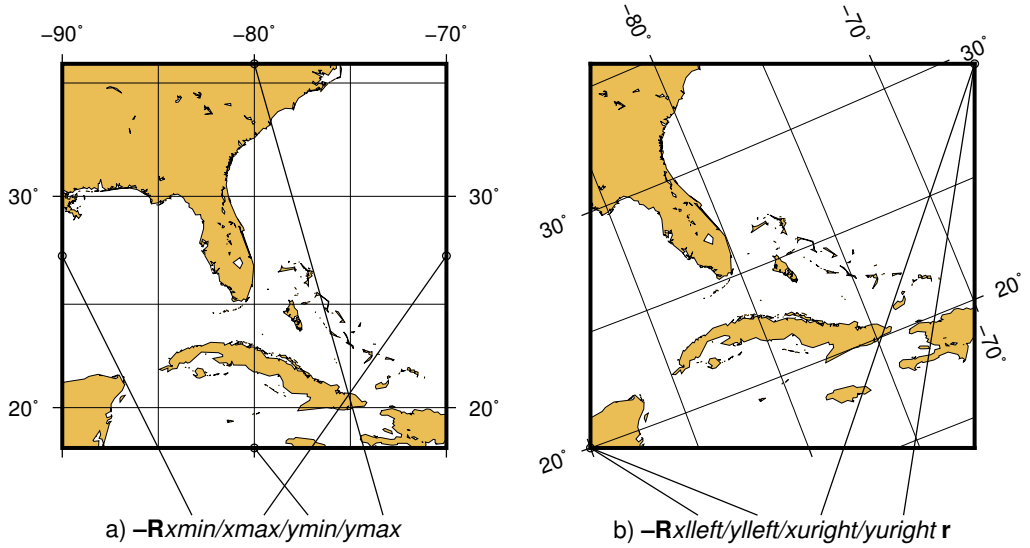


图 5.1: `-R` 选项指定数据范围

(a) 指定每个维度的极值；(b) 指定左下角和右上角的坐标

5.1.2 五种坐标

GMT 的坐标轴可以是多种类型，比如地图的经纬度轴、一般的笛卡尔轴、时间轴等等。不同类型的轴需要用不同的方式来指定数据的范围。

5.1.2.1 地理坐标

在数据格式 中已经介绍过地理坐标的格式，既可以用浮点数表示也可以用度分秒表示。

对于两个常用的范围，可以使用其简写形式，`-Rg` 等效于 `-R0/360/-90/90` ，`-Rd` 等效于 `-R-180/180/-90/90` 。

GMT 对于地理投影和线性投影的默认设置有很大区别，有些时候数据是地理坐标，但是因为某些原因不能选择地理投影，只能选择线性投影（`-JX` 或 `-Jx` ），此时可以通过如下几种方式表明当前数据是地理坐标下的数据，而不是简单的笛卡尔坐标：

1. 使用 `-Rg` 而不是 `0/360/-90/90`
2. 使用 `-Rd` 而不是 `-180/180/-90/90`
3. 使用 `-Rg<xmin>/<xmax>/<ymin>/<ymax>` 表明某个有限范围的地理区域
4. 在范围后加后缀 `W|E|S|N` 或更通用的 `D|G` ，比如 `-R0/360G/-90/90N`
5. 使用 `-fg` 选项表明输入数据是地理坐标

5.1.2.2 投影后坐标

地理坐标可以通过选择投影方式投影成笛卡尔坐标，可以通过在范围前加上一个前置的长度单位来表明这是一个经过投影的地理坐标。

例如 `-Rk-200/200/-300/300` 表示位于投影中心 `(0,0)` 处的一个 `400km x 600km` 的矩形区域。这些坐标在 GMT 内部会被转换成对应的地理坐标。当你想要用投影单位指定区域时用这种方式会比较方便（例如 UTM meters）。

5.1.2.3 绝对时间坐标

见数据格式 一节。

5.1.2.4 相对时间坐标

见数据格式 一节。

5.1.2.5 笛卡尔坐标

见数据格式 一节。

5.2 -J 选项

-J 选项用于指定将数据投影到画布上所采用的函数，即投影方式。-J 选项的用法有两种：

- -J<δ>[<pars>/]<scale>
- -J<Δ>[<pars>/]<width>

此处 <δ> 是一个小写字母，<Δ> 是一个大写字母，代表某种投影方式。<pars> 代表零个或多个由斜杠分隔的投影参数，具体的参数数目由投影方式决定。

投影方式使用大写字母时，其最后一个参数 <width> 表示地图宽度。投影方式使用小写字母时，最后一个参数 <scale> 表示比例尺，可以只跟一个数字 xxxx ，也可以是 1:xxxx 格式。前者表示地球上的 1 度在图上为 xxxx 厘米（当然也可以是其他单位），而后者表示地图上的 1 厘米代表真实地球的 xxxx 厘米。

即，可以使用 -J 选项选择投影方式，每个投影方式有大写和小写两种形式，其中小写形式指定的是地图的比例尺，大写形式指定的是图在画布上的宽度。

例如：

- -Jm1c 表示使用墨卡托投影，地图上的 1 度距离投影到画布上为 1 厘米
- -Jm1:10000000 表示使用墨卡托投影，画布上的 1 cm 代表实际距离中的 10000000 cm，即 100 km
- -JM15c 也表示使用墨卡托投影，整个地图的宽度是 15 厘米，地图的高度由 -R 和 -J 自动
- -JX10c/5c 使用线性投影，地图的宽度是 10 厘米，高度为 5 厘米

下表列出了 GMT 所支持的全部投影方式，详细介绍见投影方式 。

表 5.2: GMT -J Codes

-JA$\mathrm{lon_0/lat_0}[/horizon]/width$	Lambert azimuthal equal area
-JB$\mathrm{lon_0/lat_0/lat_1/lat_2}width$	Albers conic equal area
-JC$\mathrm{lon_0/lat_0}width$	Cassini cylindrical
-JCyl_stere/$[\mathrm{lon_0}[/\mathrm{lat_0}/]]width$	Cylindrical stereographic
-JD$\mathrm{lon_0/lat_0/lat_1/lat_2}width$	Equidistant conic
-JE$\mathrm{lon_0/lat_0}[/horizon]/width$	Azimuthal equidistant
-JF$\mathrm{lon_0/lat_0}[/horizon]/width$	Azimuthal gnomonic
-JG$\mathrm{lon_0/lat_0}[/horizon]/width$	Azimuthal orthographic
-JG$\mathrm{lon_0/lat_0}alt/azim/tilt/twist/W/H/width$	General perspective
	下页继续

表 5.2 – 续上页

-JHlon$_0$*width*	Hammer equal area
-JIlon$_0$*width*	Sinusoidal equal area
-JJlon$_0$*width*	Miller cylindrical
-JKflon$_0$*width*	Eckert IV equal area
-JKslon$_0$*width*	Eckert VI equal area
-JLlon$_0$/lat$_0$/lat$_1$/lat$_2$*width*	Lambert conic conformal
-JM[lon$_0$[/lat$_0$/]]*width*	Mercator cylindrical
-JN[lon$_0$/]*width*	Robinson
-JOalon$_0$/lat$_0$*azim*/*width*	Oblique Mercator, 1: origin and azimuth
-JOblon$_0$/lat$_0$/lon$_1$/lat$_1$*width*	Oblique Mercator, 2: two points
-JOclon$_0$/lat$_0$/lon$_\mathrm{p}$/lat$_\mathrm{p}$*width*	Oblique Mercator, 3: origin and pole
-JP[**a**]*width*[/*origin*]	Polar [azimuthal] (θ, r) (or cylindrical)
-JPoly[lon$_0$[/lat$_0$/]]*width*	(American) polyconic
-JQ[lon$_0$[/lat$_0$/]]*width*	Equidistant cylindrical
-JR[lon$_0$/]*width*	Winkel Tripel
-JSlon$_0$/lat$_0$[/*horizon*]/*width*	General stereographic
-JT[lon$_0$[/lat$_0$/]]*width*	Transverse Mercator
-JU*zone*/*width*	Universal Transverse Mercator (UTM)
-JV[lon$_0$/]*width*	Van der Grinten
-JW[lon$_0$/]*width*	Mollweide
-JX*width*[**l**|**p***exp*|**T**|**t**][/*height*[**l**|**p***exp*|**T**|**t**]][**d**]	Linear, $\log_{10}$, $x^a - y^b$, and time
-JYlon$_0$/lat$_0$*width*	Cylindrical equal area

GMT 用单个字母指定投影方式，但英文字母只有 26 个，而投影方式却不止 26 个，因而，从 GMT 4.3.0 开始，GMT 开始支持 `Proj4` 包的命名方式。与上面介绍的不同，`Proj4` 包不是使用单个字符指定投影方式，而是通过一个单词指定。比如墨卡托投影既可以用 `-Jm` 指定也可以用 `-Jmerc` 指定。

表 5.3: Proj4 -J codes

-Jaea/lon$_0$/lat$_0$/lat$_1$/lat$_2$*scale*	Albers conic equal area
-Jaeqd/lon$_0$/lat$_0$[/*horizon*]/*scale*	Azimuthal equidistant
-Jcass/lon$_0$/lat$_0$*scale*	Cassini cylindrical
-Jcea/lon$_0$/lat$_0$*scale*	Cylindrical equal area
-Jcyl_stere/[lon$_0$[/lat$_0$/]]*scale*	Cylindrical stereographic
-Jeqc/[lon$_0$[/lat$_0$/]]*scale*	Equidistant cylindrical
-Jeqdc/lon$_0$/lat$_0$/lat$_1$/lat$_2$*scale*	Equidistant conic
	下页继续

表 5.3 – 续上页

-Jgnom/lon_0/lat_0[/*horizon*]/*scale*	Azimuthal gnomonic
-Jhammer/lon_0*scale*	Hammer equal area
-Jeck4/lon_0*scale*	Eckert IV equal area
-Jeck6/lon_0*scale*	Eckert VI equal area
-Jlaea/lon_0/lat_0[/*horizon*]/*scale*	Lambert azimuthal equal area
-Jlcc/lon_0/lat_0/lat_1/lat_2*scale*	Lambert conic conformal
-Jmerc/[lon_0[/lat_0/]]*scale*	Mercator cylindrical
-Jmill/lon_0*scale*	Miller cylindrical
-Jmoll/[lon_0/]*scale*	Mollweide
-Jnsper/lon_0/lat_0*alt/azim/tilt/twist/W/H/scale*	General perspective
-Jomerc/lon_0/lat_0*azim*/*scale*	Oblique Mercator, 1: origin and azimuth
-Jomerc/lon_0/lat_0/lon_1/lat_1*scale*	Oblique Mercator, 2: two points
-Jomercp/:lon_0/lat_0/lon_p/lat_p*scale*	Oblique Mercator, 3: origin and pole
-Jortho/lon_0/lat_0[/*horizon*]/*scale*	Azimuthal orthographic
-Jpolar/[**a**]*scale*[/*origin*]	Polar [azimuthal] (θ, r) (or cylindrical)
-Jpoly/[lon_0[/lat_0/]]*scale*	(American) polyconic
-Jrobin/[lon_0/]*scale*	Robinson
-Jsinu/lat_0*scale*	Sinusoidal equal area
-Jstere/lon_0/lat_0[/*horizon*]/*scale*	General stereographic
-Jtmerc/[lon_0[/lat_0/]]*scale*	Transverse Mercator
-Jutm/*zone*/*scale*	Universal Transverse Mercator (UTM)
-Jvandg/[lon_0/]*scale*	Van der Grinten
-Jwintri/[lon_0/]*scale*	Winkel Tripel
-Jxy*xscale*[**l**|**p***exp*|**T**|**t**][/*yscale*[**l**|**p***exp*|**T**|**t**]][**d**]	Linear, $\log_{10}$, $x^a - y^b$, and time

5.3 -B 选项

-B 选项用于控制底图边框的显示。

-B 选项有两套语法，分别用于设置底图的边框以及每条轴的属性，因而在一个命令中可能需要多次使用 -B 选项。若命令中没有出现 -B 选项，则不绘制底图边框。

5.3.1 边框设置

-B 选项在设置边框属性时的语法为:

```
-B[<axes>][+b][+g<fill>][+n][+o<lon>/<lat>][+t<title]
```

其中：

- <axes> 控制显示底图的哪几条边
- +b 在 3D 绘图中根据 -R 选项指定的范围绘制长方体的 12 条边
- +g<fill> 在底图内部填色，见填充 一节
- +n 表示不绘制边框和标注
- +o<lon>/<lat> 指定网格线的参考点。默认情况下，网格线是以北极点作为参考的，如果你想要以另一个点作为参考绘制倾斜的网格线，则可以使用 +o 子选项
- +t<title> 指定当前底图的标题。该标题位于底图的上方中部。标题可以是任意字符串，如果是字符串中有空格，则必须用引号将字符串括起来

通常情况下，只需要使用 <axes> 和 +t<title> 选项。

5.3.1.1 <axes>

<axes> 控制要绘制哪些边以及这些边是否显示标注。对于二维图而言，有上下左右四条边，分别用东西南北四个方向的英文单词首字母表示。对于每条边都有三种状态：

1. 不出现该字母表示不绘制这条边
2. 用大写字母表示绘制这条边，且该边有刻度、有标注
3. 用小写字母表示绘制这条边，但该边有刻度、无标注

下面两个命令，分别使用了不同的 -B 选项，可以自己执行，查看绘图效果并试着理解 <axes> 的用法：

```
gmt psbasemap -R0/10/0/10 -JX5c -B2 -BWSEN > test1.ps
gmt psbasemap -R0/10/0/10 -JX5c -B2 -BWSn > test2.ps
```

对于 3D 绘图来说，<axes> 还可以加上一个 Z 用于控制 Z 轴。同理，大写的 Z 表示有刻度和标注，小写的 z 表示无标注。默认情况下，只会绘制一条 Z 轴，可以使用 1234 的任意组合来表示要绘制哪些 Z 轴。其中 1 表示左下角的 Z 轴，其他 Z 轴按逆时针顺序编号。加上 +b 子选项，会绘制一个由 -R 选项范围决定的长方体的 12 条边，即相当于一个 box。如果 Z 轴有指定网格间距，则会在 xz 和 yz 平面内显示网格线。

下面的命令展示了 3D 绘图中 -B 选项的不同用法，读者可以自己一一测试，根据绘图效果理解 -B` 选项中各字母的含义。命令中的某些选项还没有介绍过，暂时可以不必理会其含义：

```
gmt psbasemap -R0/10/0/10/0/10 -JX5c -JZ5c -Bz2 -BWSENZ -p45/45 > test1.ps
gmt psbasemap -R0/10/0/10/0/10 -JX5c -JZ5c -Bz2 -BWSENZ1234 -p45/45 > test2.ps
gmt psbasemap -R0/10/0/10/0/10 -JX5c -JZ5c -Bz2 -BWSEN+b -p45/45 > test3.ps
gmt psbasemap -R0/10/0/10/0/10 -JX5c -JZ5c -Bz2 -B+b -p45/45 > test4.ps
gmt psbasemap -R0/10/0/10/0/10 -JX5c -JZ5c -Bz2 -BWESNZ+b -p45/45 > test5.ps
gmt psbasemap -R0/10/0/10/0/10 -JX5c -JZ5c -B2 -Bz2 -BwSEnZ+b -p45/45 > test6.ps
```

5.3.1.2 示例

```
gmt psbasemap -R0/10/0/10 -JX5c -Ba2g2 -BWSen+glightblue+ttitle > test.ps
gmt psbasemap -R0/10/0/10 -JX5c -Ba2g2 -BWS+glightblue+t"This is title" > test2.ps
```

5.3.2 轴设置

X 轴、Y 轴、Z 轴，每条轴都有很多属性，包括刻度间隔、网格线间隔、轴标签以及标注的间隔、前缀和单位。轴属性可以用如下语法控制:

```
-B[p|s][x|y|z]<intervals>[+l|L<label>][+p<prefix>][+u<unit>]
```

为了更加清晰，以上的语法也可以被分为两部分:

```
-B[p|s][x|y|z]<intervals>
-B[p|s][x|y|z][+l|L<label>][+p<prefix>][+u<unit>]
```

其中，

- `p|s` 主属性或次属性
- `x|y|z` 设置哪一条轴的属性
- `<interval>` 设置刻度、网格线、标注的间隔
- `+l|L<label>` 给选中的轴加标签，
  - `+l<label>` 标签文字方向平行于 Y 轴方向
  - `+L<label>` 标签文字方向平行于 X 轴方向（仅适用于标签文字很短的情况）
- `+p<prefix>` 选中的轴的标注加前缀
- `+u<unit>` 给选中的轴的标注加单位。对于地图而言，标注的单位为度，该符号是自动添加的，由*FORMAT_GEO_MAP* 控制

5.3.2.1 p|s

对于每个轴来说，都有两个等级的属性可以设置，分别称为 p（Primary）和 s（Secondary）。

对于地理坐标而言，通常只需要使用默认的 Primary 属性即可，而 Secondary 则主要用于坐标轴为时间轴的情况下，此时 p 和 s 分别用于指定不同尺度的时间间隔。在 GMT 默认的情况下，p 属性的标注比较靠近坐标轴，而 s 属性的标注离坐标轴稍远。p 和 s 的用法与区别，可以参考后面给出的例子。

5.3.2.2 x|y|z

要设置哪些边的信息，默认值为 xy ，即同时设置 X 轴和 Y 轴的信息。可以指定单个轴（比如只有 x ），也可以同时指定多个轴（比如 xy 和 xyz ）。如果想要不同轴有不同的设置，则需要多次使用 -B 选项，每个指定不同的轴。

5.3.2.3 <interval>

每个轴都有三个属性，分别是标注（annotation）、刻度（frame）和网格线（grid）。下图展示了这三个名词在绘图时的具体含义。

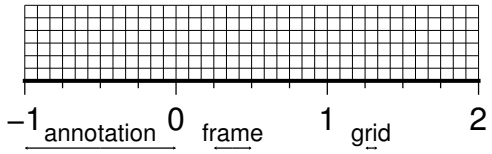


图 5.2: GMT 坐标轴中的标注、刻度和网格线

<interval> 可以用于设置这三个属性的间隔，它是一个或多个 [<t>]<stride>[<phase>][<u>] 的组合。

- <t> 可以取 a （标注）、f （刻度）、g （网格线），表明了要设置轴的哪部分的间隔
- <stride> 用于设置间隔，stride 为 0，表示不绘制
- <phase> 可以用于控制标注、刻度或网格线的起算点
- <u> 是间隔的单位，主要用于指定时间间隔

比如：-Ba30f15g15 ， -Bxa10 -Bya15 等等。

-B 选项还有一个可以自动计算间隔的功能，-Bafg 会根据当前的区域大小等信息自动计算合适的间隔，-Bafg/afg 则会对 X 轴和 Y 轴分别计算合适的间隔。

5.3.3 地理底图

地理底图与一般的坐标轴不同，其底图类型*MAP_FRAME_TYPE* 使用 `fancy` 形式。

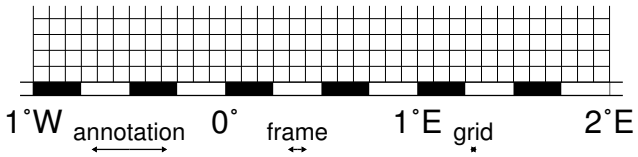


图 5.3: 地理底图示例 1

`-Ba1f15mg5m -BS`

下图同时使用了 `p` 和 `s` 两级属性。这里 `p` 属性用于显示弧度，`s` 属性用于显示弧分。

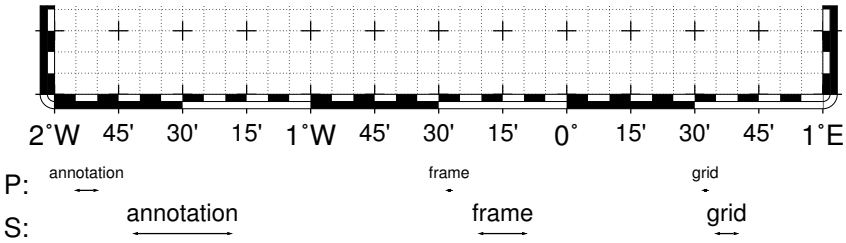


图 5.4: 地理底图示例 2

同时使用 P 和 S 两级属性 `-Bpa15mf5mg5m -BwSe -Bs1f30mg15m`

5.3.4 笛卡尔线性轴

对于一般的线性轴而言，标注的格式由*FORMAT_FLOAT_OUT* 决定，其默认值为 `%g` ，即根据数据的大小决定用一般表示还是指数表示，小数位的数目会根据 `<stride>` 自动决定。若设置*FORMAT_FLOAT_OUT* 为其他值，则会严格使用其定义的格式，比如 `%.2f` 表示显示两位小数。

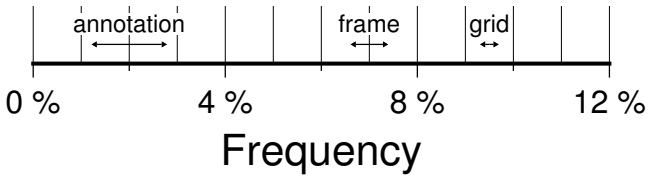


图 5.5: 笛卡尔线性轴

`-R0/12/0/0.95 -JX3i/0.3i -Ba4f2g1+lFrequency+u" %" -BS`

5.3.5 笛卡尔 $\log_{10}$ 轴

由于对数坐标的特殊性，`<stride>` 参数具有特殊的含义。下面说明 `<stride>` 在对数坐标下的特殊性：

- `<stride>` 必须是 1、2、3 或负整数-n。
  - `1`： 每 10 的指数
  - `2`： 每 10 的指数的 1、2、5 倍
  - `3`： 每 10 的指数的 0.1 倍
  - `-n`： 每 10 的 n 次方出现一次
- 在 `<stride>` 后加 `l` ，则标注会以 $\log_{10}$ 的值显示，比如 100 会显示成 2
- 在 `<stride>` 后加 `p` ，则标注会以 10 的 n 次方的形式显示，比如 10^{-5}

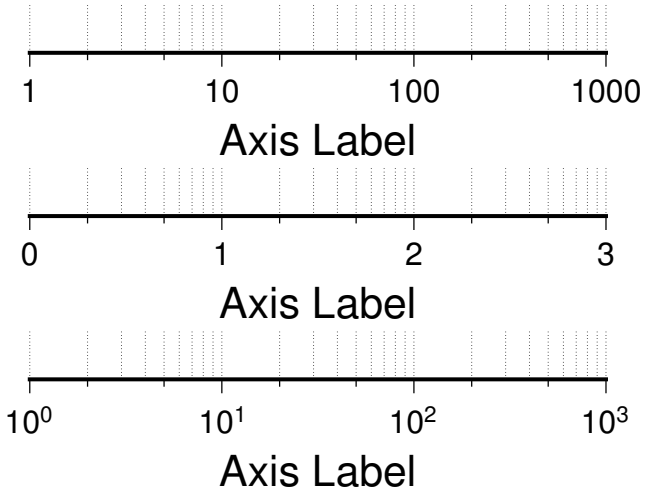


图 5.6: 对数坐标轴

(上) `-R1/1000/0/1 -JX3il/0.25i -Ba1f2g3`(中) `-R1/1000/0/1 -JX3il/0.25i -Ba1f2g3l`(下) `-R1/1000/0/1 -JX3il/0.25i -Ba1f2g3p`

5.3.6 笛卡尔指数轴

正常情况下，`<stride>` 用于生成等间隔的标注或刻度，但是由于指数函数的特性，这样的标注会在坐标轴的某一端挤在一起。为了避免这个问题，可以在 `<stride>` 后加 `p` ，则标注会按照转换后的值等间隔出现，而标注本身依然使用未转换的值。比如，若 stride=1，pow=0.5（即 sqrt），则在 1、4、处会出现标注。

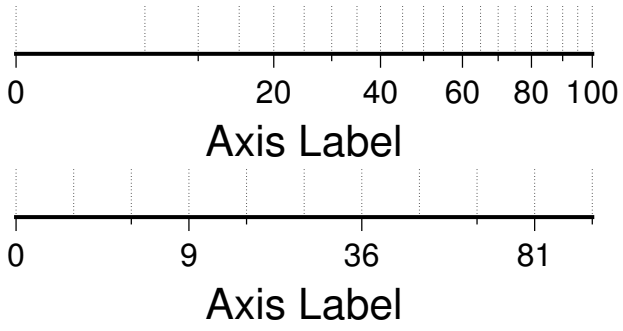


图 5.7: 指数投影坐标轴

(上) `-R0/100/0/0.9 -JX3ip0.5/0.25i -Ba20f10g5` (下) `-R0/100/0/0.9 -JX3ip0.5/0.25i -Ba3f2g1p`

5.3.7 时间轴

时间轴与其他轴不同的地方在于，时间轴可以有多种不同的标注方式。下面会用一系列示例来演示时间轴的灵活性。在下面的例子中，尽管只绘制了 X 轴（绘图时使用了 `-BS` ），实际上时间轴标注的各种用法使用于全部轴。

在绘制时间轴时，需要指定时间间隔，时间间隔的单位可以取如下值:

表 5.4: GMT 时间单位

Flag	Unit	Description
Y	year	Plot using all 4 digits
y	year	Plot using last 2 digits
O	month	Format annotation using FORMAT_DATE_MAP
o	month	Plot as 2-digit integer (1–12)
U	ISO week	Format annotation using FORMAT_DATE_MAP
u	ISO week	Plot as 2-digit integer (1–53)
r	Gregorian week	7-day stride from start of week (see TIME_WEEK_START)
K	ISO weekday	Plot name of weekday in selected language
k	weekday	Plot number of day in the week (1–7) (see TIME_WEEK_START)
D	date	Format annotation using FORMAT_DATE_MAP
d	day	Plot day of month (1–31) or day of year (1–366) (FORMAT_DATE_MAP)
R	day	Same as d; annotations aligned with week (see TIME_WEEK_START)
H	hour	Format annotation using FORMAT_CLOCK_MAP
h	hour	Plot as 2-digit integer (0–24)
M	minute	Format annotation using FORMAT_CLOCK_MAP
m	minute	Plot as 2-digit integer (0–60)
S	seconds	Format annotation using FORMAT_CLOCK_MAP
s	seconds	Plot as 2-digit integer (0–60)

第一个例子展示了 2000 年春天的两个月，想要将这两个月的每周的第一天的日期标注出来:

```
gmt set FORMAT_DATE_MAP=-o FONT_ANNOT_PRIMARY +9p
gmt psbasemap -R2000-4-1T/2000-5-25T/0/1 -JX5i/0.2i -Bpa7Rf1d -Bsa1O -BS -P > GMT_-B_
 ↪time1.ps
```

绘图效果如下图所示，需要注意*FORMAT_DATE_MAP* 前的破折号会去掉日期前面的前置零(即 02 变成 2)。

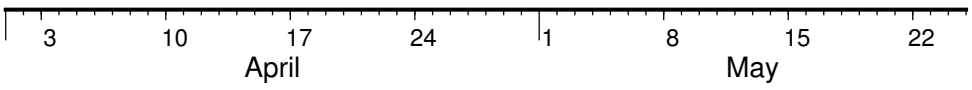


图 5.8: 时间轴示例 1

下面的例子用两种不同的方式标注了 1969 年的两天:

```
gmt set FORMAT_DATE_MAP "o dd" FORMAT_CLOCK_MAP hh:mm FONT_ANNOT_PRIMARY +9p
gmt psbasemap -R1969-7-21T/1969-7-23T/0/1 -JX5i/0.2i -Bpa6Hf1h -Bsa1K -BS -P -K > GMT_
 ↪-B_time2.ps
gmt psbasemap -R -J -Bpa6Hf1h -Bsa1D -BS -O -Y0.65i >> GMT_-B_time2.ps
```

绘图效果如下图所示。图中下面的例子使用周来标注，上面的例子使用日期来标注。

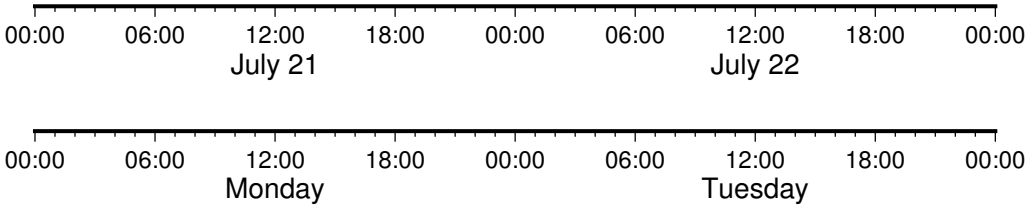


图 5.9: 时间轴示例 2

第三个例子展示了两年的时间，并标注了每年以及每三个月:

```
gmt set FORMAT_DATE_MAP o FORMAT_TIME_PRIMARY_MAP Character FONT_ANNOT_PRIMARY +9p
gmt psbasemap -R1997T/1999T/0/1 -JX5i/0.2i -Bpa3Of1o -Bsa1Y -BS -P > GMT_-B_time3.ps
```

年标注位于一年间隔的中间，月标注位于对应月的中间而不是三个月间隔的中间。

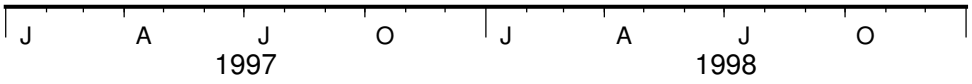


图 5.10: 时间示例 3

第四个例子展示了一天中的几个小时，通过在 R 选项中指定 t 来使用相对时间坐标。这里使用了 p 属性和 s 属性，12 小时制，时间从右向左增加:

```
gmt set FORMAT_CLOCK_MAP=-hham FONT_ANNOT_PRIMARY +9p
gmt psbasemap -R0.2t/0.35t/0/1 -JX-5i/0.2i -Bpa15mf5m -Bsa1H -BS -P > GMT_-B_time4.ps
```

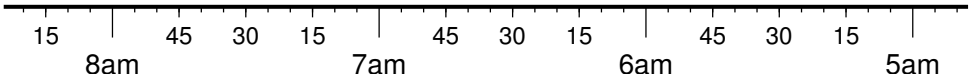


图 5.11: 时间轴示例 4

第五个例子用两种方式展示了几周的时间:

```
gmt set FORMAT_DATE_MAP u FORMAT_TIME_PRIMARY_MAP Character \
       FORMAT_TIME_SECONDARY_MAP full FONT_ANNOT_PRIMARY +9p
gmt psbasemap -R1969-7-21T/1969-8-9T/0/1 -JX5i/0.2i -Bpa1K -Bsa1U -BS -P -K > GMT_-B_
 ↪time5.ps
```

```
gmt set FORMAT_DATE_MAP o TIME_WEEK_START Sunday FORMAT_TIME_SECONDARY_MAP Chararacter
gmt psbasemap -R -J -Bpa3Kf1k -Bsa1r -BS -O -Y0.65i >> GMT_-B_time5.ps
```

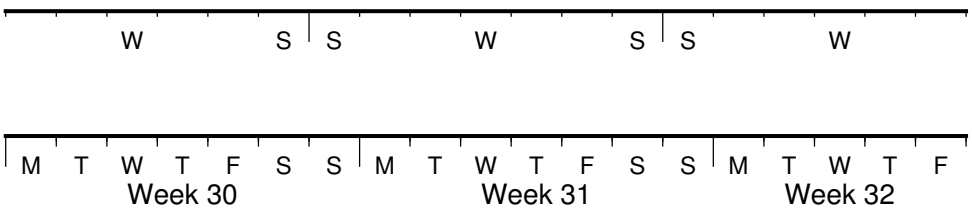


图 5.12: 时间轴示例 5

第六个例子展示了 1996 年的前 5 个月，每个月用月份的简写以及两位年份标注:

```
gmt set FORMAT_DATE_MAP "o yy" FORMAT_TIME_PRIMARY_MAP Abbreviated
gmt psbasemap -R1996T/1996-6T/0/1 -JX5i/0.2i -Ba1Of1d -BS -P > GMT_-B_time6.ps
```

图 5.13: 时间轴示例 6

第七个例子:

```
gmt set FORMAT_DATE_MAP jjj TIME_INTERVAL_FRACTION 0.05 FONT_ANNOT_PRIMARY +9p
gmt psbasemap -R2000-12-15T/2001-1-15T/0/1 -JX5i/0.2i -Bpa5Df1d -Bsa1Y -BS -P > GMT_-
↪B_time7.ps
```

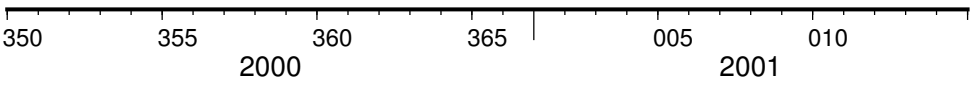


图 5.14: 时间轴示例 7

5.3.8 自定义轴

GMT 允许用户定义标注来实现不规则间隔的标注，用法是 `-Bc` 后接标注文件名。

标注文件中以“#”开头的行为注释行，其余为记录行，记录行的格式为:

```
coord   type   [label]
```

- `coord` 是需要标注、刻度或网格线的位置
- `type` 是如下几个字符的组合
  - `a` 或 `i` 前者为 annotation，后者表示 interval annotation

  - 在一个标注文件中，`a` 和 `i` 只能出现其中的任意一个
  - `f` 表示刻度，即 frame tick
  - `g` 表示网格线，即 gridline
- `label` 默认的标注为 `coord` 的值，若指定 `label` ，则使用 `label` 的值

需要注意，`coord` 必须按递增顺序排列。

下面的例子展示中展示了自定义标注的用法，`xannots.txt` 和 `yannots.txt` 分别是 X 轴和 Y 轴的标注文件。

```
cat << EOF > xannots.txt
416.0 ig Devonian
443.7 ig Silurian
488.3 ig Ordovician
542 ig Cambrian
EOF
cat << EOF > yannots.txt
0 a
1 a
2 f
2.71828 ag e
3 f
3.1415926 ag @~p@~
4 f
5 f
6 f
6.2831852 ag 2@~p@~
EOF
gmt psbasemap -R416/542/0/6.2831852 -JX-5i/2.5i -Bpx25f5g25+u" Ma" -Bpycyannots.txt \
              -BWS+glightblue -P -K > GMT_-B_custom.ps
gmt psbasemap -R416/542/0/6.2831852 -JX-5i/2.5i -Bsxcxannots.txt -BWS -O \
              --MAP_ANNOT_OFFSET_SECONDARY=10p --MAP_GRID_PEN_SECONDARY=2p >> GMT_-B_
↪custom.ps
rm -f [xy]annots.txt
```

5.4 -P 选项

在画布 一节中说过，画布有两种放置方式：Portrait 模式和 Landscape 模式。

由于历史原因，GMT 默认使用 Landscape 模式，可以通过 `-P` 选项设定使用 Portrait 模式，也可以通过设置 GMT 参数*PS_PAGE_ORIENTATION* 来修改纸张方向。

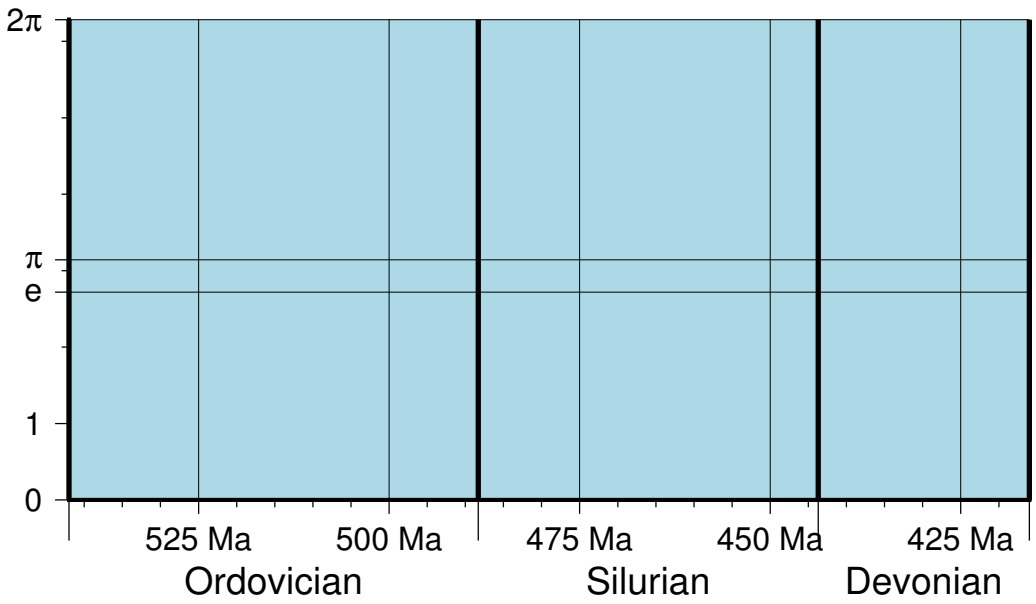


图 5.15: 自定义坐标轴

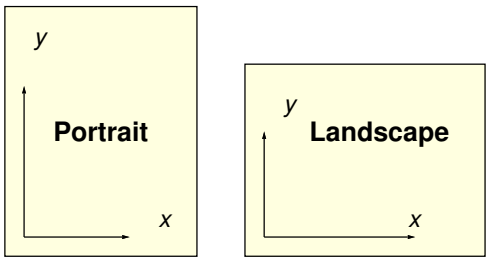


图 5.16: -P 选项指定画布方向

`-P` 选项设置的是**画布的属性**，对于由多个命令绘制而成的图片来说，只有第一个命令的 `-P` 选项是有效的。因而其他绘图命令中使用或不使用 `-P` 选项，不会影响绘图效果。

Linux 下默认的 PS 阅读器 evince 可以自动识别两种模式，Landscape 模式的 PS 文件横着放，Portrait 模式的文件竖着放，所以看上去总是对的。而 `gs` 无法自动识别两种模式，总是将文件竖着放，因而用 gs 看 Landscape 模式的 PS 文件时就会很别扭。

以下是进阶内容，仅供高级用户阅读。

将一个 Landscape 模式的 PS 文件与一个 Portait 模式的 PS 文件对比:

```
$ gmt psbasemap -R0/10/0/10 -JX10c/10c -B1 > 01.ps
$ gmt psbasemap -R0/10/0/10 -JX10c/10c -B1 -P > 02.ps
```

两个文件的差异如下:

```
$ diff 01.ps 02.ps
11c11
< %%Orientation: Landscape
---
> %%Orientation: Portrait
658c658
< V 595 0 T 90 R 0.06 0.06 scale
---
> V 0.06 0.06 scale
661,662c661,662
< /PSL_page_xsize 14033 def
< /PSL_page_ysize 9917 def
---
> /PSL_page_xsize 9917 def
> /PSL_page_ysize 14033 def
669c669
< %%GMT: psbasemap -R0/10/0/10 -JX10c/10c -B1
---
> %%GMT: psbasemap -R0/10/0/10 -JX10c/10c -B1 -P
```

可以发现，两个 PS 文件的差异主要在于 `595 0 T 90 R` 以及交换了 `PSL_page_xsize` 和 `PSL_page_ysize` 的值。`595 0 T 90 R` 的含义应该是将坐标系移动（**T**ransition）到 `(595,0)` 再旋转（**R**otate）90 度，即由 Portrait 模式变成 Landscape 模式。

5.5 -V 选项

`-V` 选项使得命令进入 verbose 模式，即会输出进程报告到标准错误流。

verbose 模式有 6 个等级，等级越高输出的信息越多，高等级会在低等级的基础上加上更多的输出信息。6 个等级从低到高分别为：

- `-Vq` ： quiet；不输出任何错误和警告；
- `-Vn` ： nomral；仅输出致命错误信息；即不使用 `-V` 选项时的默认值；
- `-Vc` ： compatibility；输出兼容性相关的警告信息；
- `-Vv` 或 `-V` ： verbose；即输出错误、警告以及数据处理的基本信息；
- `-Vl` ： long；详细的进程报告；
- `-Vd` ： debug；包含了大量调试信息；

该选项仅对当前命令有效，若希望接下来所有的命令都具有某个 verbose 级别，可以修改 GMT 参数*GMT_VERBOSE* 。

5.6 -U 选项

`-U` 选项会在纸上绘制一个带有 UNIX 系统时间戳的 GMT logo。其语法为：

```
-U[<just>/<dx>/<dy>/][c|<label>]
```

- `-U` 不加任何参数时会在当前图的左下角添加一个带时间戳的 logo
- `-U<just>/<dx>/<dy>` 可以调整 logo 的对齐方式（见锚点 ）以及相对于当前坐标原点的位置，比如 `-UBL/-54p/-54p`
- `-U<label>` 会在时间戳后打印字符串 `<label>`，比如 `-U"This is string"`
- `-Uc` 会在时间戳后打印当前命令

GMT 参数中有一些与 logo 相关的参数：

- *MAP_LOGO* 控制是否绘制时间戳，默认值为 `FALSE`
- *MAP_LOGO_POS* 用于控制时间戳的位置
- *FORMAT_TIME_STAMP* 用于控制时间戳的显示格式
- *FONT_LOGO* 时间戳中文本的字体

命令：

```
gmt psbasemap -R0/10/0/5 -JX10c/3c -Bx1 -By1 -P -UBL/-1.5c/-1.5c/"This is a GMT logo"␣
↪> GMT_-U.ps
```

显示效果如下图：

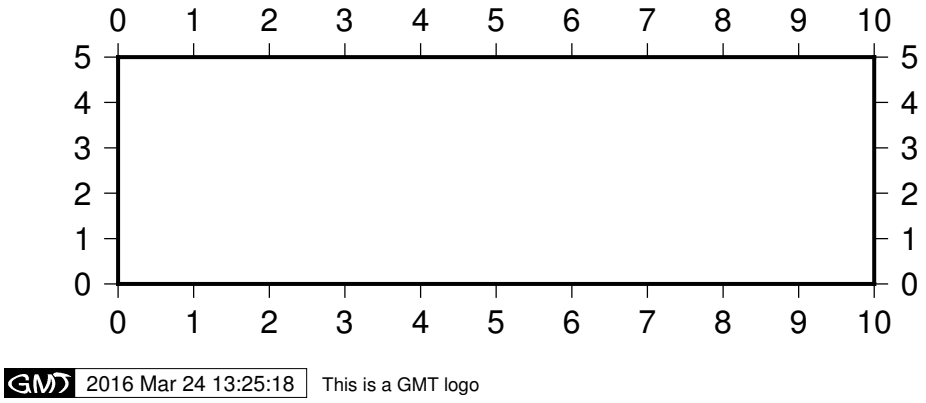


图 5.17: -U 选项加时间戳

说明：

1. 每个带有 -U 选项的绘图命令都会绘制一遍 logo，通常只需要一个绘图命令中使用 -U 即可

5.7 -K 和 -O 选项

-K 选项和 -O 选项是 GMT 绘图中常用的两个选项，也是经常产生错误的两个选项。

5.7.1 图层

要理解 -K 选项和 -O 选项，首先要理解图层的概念。对于使用过 PhotoShop 或者是在 PowerPoint 中画过图的人，应该很容易理解这一概念。所谓图层，可以理解成一张画有线条与文字的透明胶片，将这些透明胶片按照顺序一张张重叠起来，即构成了最终的图片。以 PhotoShop 中的图层为例，使用图层的好处在于，可以将某一图层单独拿出来修改，而不影响到其他图层。

图层的概念也适用于 GMT。通常来说，用 GMT 绘图时是不可能用一个命令就画出想要的效果的。比如要画一个震中和台站的分布图，通常需要将如下几个命令组合起来：

```
# 本示例中命令不完整，仅供演示用途

# 绘制海岸线
gmt pscoast ... > map.ps
# 用五角星表示地震位置
gmt psxy event.dat -Sa0.5c ... >> map.ps
# 用三角形表示台站位置
gmt psxy station.in -St0.5c ... >> map.ps
```

每一个绘图命令本质上都是在向 PS 文件中新增了一个图层，所以上面生成的 PS 文件中共有三个图层，第一个图层仅包含了海岸线，第二个图层仅包含一些五角星，第三个图层仅包含一些三角形。这些图层按照顺序叠加起来，就组成了我们最终看到的图片。在图层叠加过程中，后面的图层中的非透明区域会覆盖前面的图层，比如第二个图层中填充了颜色的五角星就有可能把前一图层中的某一小段海岸线给覆盖掉。

5.7.2 PS 文件结构

在理解了图层的概念之后在来说说 PS 文件的结构。

任何一个 PS 文件都由 header、body 和 trailer 组成，可以理解成头、身体和尾巴。其中头部的作用是初始化 PS 文件，比如设置纸张大小、纸张方向、载入字体等；身体部分则是真正的绘图部分，包含了每个绘图命令绘制的图层；尾巴部分则用于控制 PS 文件的显示。

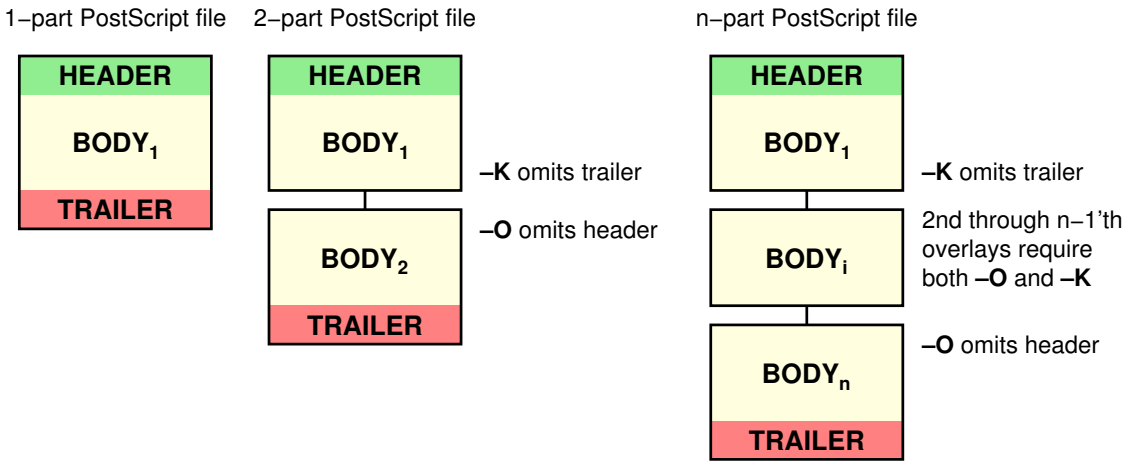


图 5.18: `-K` 和 `-O` 选项的原理

因而，一个 PS 文件中，只有一个头和尾巴，可以有零个、一个或多个身体。

5.7.3 `-K` 和 `-O` 的作用

在不使用 `-K` 和 `-O` 选项的情况下，GMT 的每个绘图命令都会产生完整的 PS 代码，即包含了头部、身体和尾部。而在使用多个命令绘制一张图片的多个图层时，则要保证第一个绘图命令必须省略尾部，中间的绘图命令必须省略头部和尾部，最后一个绘图命令必须省略头部。

`-K` 选项的作用是省略尾部，即意味着后面还有追加新的图层。`-O` 选项的作用是省略头部，即这个图层要覆盖在前一图层上。

因而，在一张图中绘制多个图层时，绘制第一个图层的命令需要使用 `-K` 选项省略尾部，绘制中间图层的命令需要使用 `-K -O` 选项，绘制最后一个图层的命令则需要使用 `-O` 选项。

因而，对于含多个图层的图片来说，绘图命令如下所示:

```
gmt pscmd1 ... -K > out.ps
gmt pscmd2 ... -K -O >> out.ps
...
gmt pscmd3 ... -K -O >> out.ps
gmt pscmd4 ... -O >> out.ps
```

5.7.4 -K 和 -O 的使用

总结一下 -K 和 -O 选项的用法。对于有 n 个图层的 PS 文件:

- 第 1 个绘图命令需要使用 -K 和重定向符号 >
- 第 2 到 n-1 个绘图命令需要使用 -K -O 和重定向符号 >>
- 第 n 个绘图命令需要使用 -O 和重定向符号 >>

上面总结的 -K 和 -O 选项的用法看上去很简单，但实际绘图中却非常容易出错。

想象一下这样一个场景，你需要画一张稍复杂的图，所以需要在脚本中写一堆命令来完成绘图。当你写脚本的时候正确地使用了 -K 和 -O 选项以及重定向符号，然后也初步完成了绘图。且不说绘图的过程有多艰辛，在完成初步绘图后你可能需要对这张图做一些微调，比如下面这些情形:

- 由于后面的图层会覆盖前面的图层，而有些覆盖是你不想要的，所以你可能会希望调换两个绘图命令的顺序。比如需要调整最开始的两个命令的顺序，或者要调整最后两个命令的顺序，你是否记得在调整顺序的时候要修改 -K 和 -O 选项以及重定向符号?
- 比如需要在第一个绘图命令前再加一个绘图命令，你是否记得要把原来的第一个绘图命令加上 -O 选项以及 >> ?
- 比如需要在最后一个绘图后面再加一个绘图命令，你是否记得要把原来的最后一个绘图命令加上 -K 选项?

上面列举的一些情形，即便是对于 GMT 比较熟悉的人，也偶尔会因为一时粗心而弄错。画图已经很不容易啦，还要时时注意 -K 、-O 和重定向有没有用错，还能不能安心的画图了?

在被 -K 和 -O 的用法坑了几次之后，就得想一想，有没有办法可以避免这个问题呢? 下面展示了一些小技巧:

```
#!/bin/bash

PS=map.ps
J=JX5c/5c
R=0/10/0/10
```

```
# 写入文件头
gmt psxy -J$J -R$R -T -K > $PS

# 真正的绘图命令
gmt xxxx -J$J -R$R ... -K -O >> $PS
gmt xxxx -J$J -R$R ... -K -O >> $PS

# 写入文件尾
gmt psxy -J$J -R$R -T -O >> $PS
```

解释一下：

- 对于需要用多个命令绘图的图片，最好将命令写到脚本文件中，这样方便记录和调试命令
- 上面的脚本是 bash 脚本，并将常出现的值定义成变量，以方便使用和修改
- psxy 模块的 `-T` 选项表示空输入，即该命令不会绘制任何实际的图形
- `gmt psxy ... -T -K` 只向 PS 文件中写入头部
- `gmt psxy ... -T -O` 只向 PS 文件中写入尾部
- 中间的全部绘图命令统一用 `-K -O >>` 。这样的统一使得，任意调整命令顺序或删减命令，都不需要修改 `-K` 、`-O` 和重定向符号！

5.8 -X 和 -Y 选项

在纸张上画图时，首先要确定原点在哪里。对于刚拿到的纸而言，最原始的原点位于纸张左下角。

很多情况下，想要自己控制底图原点的位置，`-X` 和 `-Y` 选项就用于移动底图原点。`-X` 和 `-Y` 的用法是一样的，所以下面仅以 `-X` 选项为例介绍其用法，其语法为:

```
-X[a|c|f|r][<xshift>[<u>]]
```

其中 `<xshift>` 新原点相对于当前原点的 X 方向偏移量，`<u>` 为偏移量的单位。

在偏移量之前加上不同的字符表示不同的含义：

- `-X2i` 或 `-Xr2i` ： 在**原底图原点**的基础上沿 X 方向偏移 2 英寸得到新底图原点
- `-Xa5c` ： 在**原底图原点**的基础上沿 X 方向偏移 5 厘米得到临时底图坐标，当前命令执行完成后，底图原点复原到**原底图原点**
- `-Xc3c` ： 在**纸张中心**的基础上沿 X 方向偏移 3 厘米得到新底图原点

- `-Xf4c` ： 在**纸张左下角**的基础上沿着 X 方向偏移 4 厘米得到新底图原点

在不显式使用 `-X` 和 `-Y` 的情况下，可以这样认为：

1. 若命令中未使用 `-O` 选项，即第一个绘图命令，相当于 `-Xr1i -Yr1i` ，即第一个绘图命令的绘图原点是距离纸张左下角（`1i,1i`）处，这样做是为了给底图左边和下边的刻度和标注信息留空间
2. 若命令中使用了 `-O` 选项，则相当于 `-Xr0 -Yr0` ，即不移动底图原点
3. 若使用了 `-X` 和 `-Y` 但不指定偏移量，则使用之前命令的偏移量

说明：

1. 第一个绘图命令的默认偏移量由*MAP_ORIGIN_X* 和*MAP_ORIGIN_Y* 控制
2. 当 PS 文件中有多个底图时才需要使用 `-X` 和 `-Y` 选项，不要仅仅为了在某个地方加几个文字就使用这两个选项

`-X` 和 `-Y` 选项的用法介绍起来有些难度，多试试就好，下面举个简单的例子：

```
gmt psbasemap -JX5c/2c -R0/5/0/2 -B1 -K > test.ps
gmt psbasemap -J -R -B1 -K -O -X7c >> test.ps
gmt psbasemap -J -R -B1 -K -O -X-7c -Y4c >> test.ps
gmt psbasemap -J -R -B1 -K -O -X7c >> test.ps
```

上图用四个 `psbasemap` 命令绘制了四张底图，绘图效果如下：

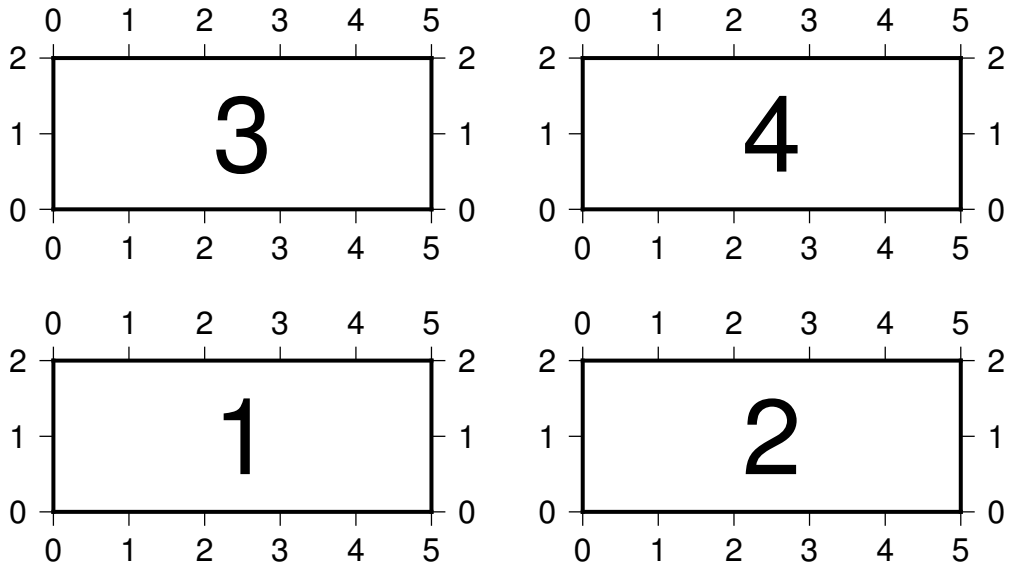


图 5.19: `-X` 和 `-Y` 移动绘图原点

解释：

1. 第一个命令的绘图原点位于纸张左下角（`1i,1i`）处，绘制底图 1

2. 第二个将绘图原点右移了 6 cm，绘制底图 2
3. 为了绘制底图 3，第三个命令将底图左移了 6 cm，并上移 6 cm
4. 第四个命令在底图 3 的基础上右移 6 cm，绘制底图 4

5.9 -a 选项

GMT5 支持读取 OGR/GMT 格式的数据文件，关于该格式的介绍，见*兼容 OGR 的 GMT 矢量数据格式* 。

该选项可以控制 GMT 在读写 OGR/GMT 格式的数据时对非空间数据的处理方式。该选项的语法为:

```
-a[<col>=]<name>[,...]
```

`-a` 选项后接一个或多个用逗号分隔的 `<col>=<name>` ，其作用在于将 OGR/GMT 格式的数据文件中非空间数据 `<name>` 字段作为输入/输出数据的第 `<col>` 列。若不指定 `<col>` ，则默认列数为 2，并依次增加。

例如 `-a2=depth` 会从数据文件中读取 X 和 Y 列信息，并从非空间数据的 `depth` 字段中读取值作为输入的第三列。

也可以通过将 `<col>` 设置成 `D|G|L|T|W|Z` 来将非空间数据与其他属性联系起来，比如标签、填充色、画笔属性、用于查找颜色的 Z 值等。该机制与在多段数据的段头记录中加上参数是等效的。

GMT 也可以用于输出 OGR/GMT 格式的数据文件，此时可以使用 `<col>=<name>[:<type>]` 来指定将输出数据的第 `<col>` 列以 `<type>` 数据类型保存到非空间字段 `<name>` 中。与输入类似，`<col>` 也可以取 `D|G|L|T|W|Z` 中的一个。对于输出而言，还需要加上 `+g<geometry>` 来指定数据的几何类型，`<geometry>` 可以取为 `[M]POINT|LINE|POLY` 。若加上 `+G` ，则程序会自动将跨越国际日期变更线的线段或多边形分成多段。

5.10 -b 选项

GMT 中的很多程序都需要读入表数据，表数据可以是 ASCII 格式、二进制格式，也可以是 netCDF 格式。

ASCII 格式的数据很直观，可以直接看到有多少列数据。而二进制格式的数据，GMT 在读入数据时是无法知道有多少列数据，也无法知道每一列数据的格式的。因而就需要使用 `-b` 选项告诉 GMT 输入或输出数据的具体格式。

-b 的语法是:

```
-b[i|o][<ncols>][<type>][+L|+B]
```

-bi 表示对输入有效，-bo 表示对输出有效。其后可以跟一个或多个用逗号分隔的 <ncols><type>，表示有 <ncols> 列类型为是 <type> 的数据。其中 <type> 可以取:

- c：即 int8_t，字符型
- u：即 uint8_t，无符号字符型
- h：即 int16_t，两字节有符号整型
- H：即 uint16_t，两字节无符号整型
- i：即 int32_t，四字节有符号整型
- I：即 uint32_t，四字节无符号整型
- l：即 int64_t，八字节有符号整型
- L：即 uint64_t，八字节无符号整型
- f：四字节单精度浮点型
- d：八字节双精度浮点型
- x：不代表某种数据类型，表示跳过 <ncols> 个字节

例如，要读入的二进制数据中有 3 列，前两列是单精度浮点型，最后一列是 int32_t 型，则在读入时可以用 -bi2f,1i 。

对于每一个 <ncols><type> 还可以加上 w 表示对数据强制做字节序转换。

+L 或 +B 表示整个数据按照 little-endian 或 big-endian 字节序读入。

若未指定 <ncols> 则假定 <type> 适用于所有列，读数据时 <ncols> 的值由数据期望得到的值决定。

5.11 -c 选项

-c 选项用于指定当前 PS 文件在打印时要被打印多少份，默认值为 1。

例如，当命令中使用 -c3 时，3 这个值会被写入到 PS 文件中，当用打印机打印该 PS 文件时，会自动打印 3 份。

说明:

1. 与 -P 选项类似，仅对第一个绘图命令有效，即仅当绘图命令中无 -K 和 -O ，或只有 -K 时才有效

5.12 -d 选项

GMT 中缺失的值会用 NaN 表示。某些情况下，用户可能会用类似-99999 这样的数据表示数据缺失，对于 GMT 而言，-99999 并不是一个特殊的值，因而就需要在将数据传递给 GMT 时告诉 GMT 某个特殊值表示数据缺失。

-d 选项的语法是:

```
-d[i|o]<nodata>
```

其作用是，对于输入数据，将数据中的 <nodata> 替换成 NaN，对于输出数据，将 NaN 替换成 <nodata> 。-di 和 -do 表示仅对输入或输出有效。

5.13 -f 选项

在某些情况下，需要显式指明当前输入/输出数据中每一列的含义，这可以通过 -f 选项来实现。

默认情况下，-f 选项对输入输出同时有效，-fi 表明该选项仅对输入数据有效，-fo 表明该选项仅对输出数据有效。

输入/输出数据中每一列的含义可以通过一系列以逗号分隔的字符串来表示。每个字符串包括两部分：列号以及数据类型。数据类型可以取 x 、y 、T 、t 、f ，分别表示这一列是经度、纬度、绝对时间、相对时间或普通的浮点数。列号是一个从零开始起算的整数（比如第 6 列的列号为 5 ），当多个连续的列有相同的数据类型时，也可以指定一个列号的范围，列号范围的格式为 <start>[:<inc>]:<stop> ，若未给定 <inc> 则默认为 1，比如 1:5 表示第 2 至 6 列数据。

例如，-fi0y,1x,3:4T 表明输入数据中第一列是纬度，第二列是经度，第 3、4 列是绝对时间，其他列数据则假定是默认的浮点数类型。

-fg 是 -f0x,1y 的缩写，表明输入/输出是地理坐标。除此之外，还可以使用 -fp<unit> ，该选项可以使用投影之后的坐标作为输入，在读入数据时投影后的坐标值会自动转换为经纬度值。

5.14 -g 选项

在处理多段数据时，GMT 提供了三种机制来决定文件中数据的分段情况：

1. 在表数据 中已经介绍了，使用数据段头记录来标记一段数据的开头

2. 若输入数据中，某个记录的某个关键列的值为 NaN，则也可以用于将该记录作为数据段的开始标识
    - 当*IO_NAN_RECORDS* 为 `skip` 时，这些包含 NaN 值的记录会被自动跳过
    - 当*IO_NAN_RECORDS* 为 `pass` 时，这些包含 NaN 值的记录会被当做数据段的开始标识
3. 也可以使用 `-g` 选项，通过判断两个相邻的数据点是否符合某个准则来决定数据分段

`-g` 选项的完整语法为:

```
-g[a]x|y|d|X|Y|D|[<col>]z[+|-]<gap>[u]
```

- `x` 两点的 X 坐标跳变超过 `<gap>`
- `y` 两点的 Y 坐标跳变超过 `<gap>`
- `d` 两点的距离超过 `<gap>`
- `<u>` 是单位
- `X|Y|D` 用于表示数据投影到纸上后 X 坐标、Y 坐标和纸上距离的跳变
- 若想要检查特定列是否满足分段准则，可以用 `[<col>]z` ，`<col>` 的默认值为 2 ，即第三列
- `+<gap>` 表示前一数据减去当前数据超过 `<gap>` 则分段
- `-<gap>` 表示当前数据减去前一数据超过 `<gap>` 则分段
- `<gap>` 表示两个数据的绝对值超过 `<gap>` 则分段

该选项可重复多次，以指定多个分段准则，默认情况下，若符合任意一个准则则分段，可以使用 `-ga` 选项，表明仅当所有准则都满足时才分段。

5.15 -h 选项

`-h` 选项用于指定在读入数据时需要跳过的头段记录数。其语法为:

```
-h[i|o][<n>][+c][+d][+r<remark>][+t<title>]
```

其中 `<n>` 表示数据中的头段记录数。`i` 和 `o` 表示仅对输入/输出有效。

输入文件中的空行以及以“#”开头的行都会被自动当做头段记录，因而会被自动跳过，在指定 `<n>` 时不需要再考虑空行以及以“#”开头的行。

几种常见的用法：

1. `-h` ： 使用 GMT 参数*IO_N_HEADER_RECS* 的值作为头段记录数（默认值为 0）
2. `-h3` ： 表示跳过 3 个头段记录

对于二进制输入文件，`<n>` 表示输入数据中要跳过的字节数，或输出数据中用空白字符补充的字节数。

对于输出文件而言，默认情况下，会将输入文件中的头段信息原样输出。使用 `+d` 以删除之前的头段信息，使用 `+c` 会将列名写到输出的头段记录中，`+r<remark>` 和 `+t<title>` 则可以分别加一个 remark 和 title 语句到输出的头段记录。

5.16 -i 和 -o 选项

经常遇到的情况是，已有的数据有很多列，而某个命令只需要其中的某几列；或者某个命令的输出有很多列，却只想要输出其中的某几列。

常见的做法是使用 `gawk` 以及管道对输入输出数据做处理。其实，GMT 提供的 `-i` 和 `-o` 选项可以满足日常的大多数需求了。

`-i` 选项的语法是:

```
-i<col>[l][s<scale>][o<offset>][,...]
```

默认情况下，GMT 会读取输入数据中的全部列。`-i` 选项后接以逗号分隔的列号或列号范围，以指定输入数据中要使用哪些数据列以及这些数据列的顺序，例如 `-i3,6,2` 表示读入数据中的第 4、7、3 列。

列号范围的格式为 `<start>[:<inc>]:[<stop>]` ，若省略 `<inc>` 则默认其值为 1，比如 `1:3` 。

每个列号后还可以加上额外的字符以对每个数据做简单的处理。比如 `l` 表示对当前列取 $\log_{10}$ ，`s` 表示将当前列乘以比例因子 `<scale>` ，`o` 表示将当前列的值加上 `<offset>` 。比如 `-i2s2o10,6,3` 表示读入数据的第 3、7、4 列，并对第 3 列数据乘以 2 再加上 10。

`-o` 选项的语法为:

```
-o<col>[,...]
```

即 `-o` 只能控制输出哪些列，不能对输出的数据做进一步处理。

5.17 -n 选项

-n 选项用于控制二维网格数据重采样过程中的插值算法。其语法为:

```
-n[b|c|l|n][+a][+b<BC>][+c][+t<threshold>]
```

b 表示 B-sphine 平滑算法，c 表示使用 bicubic 插值算法（默认值），l 表示 bilinear 插值算法，n 表示 nearest-neighbor 值。

- +a： 关闭抗混淆（仅在算法支持的前提下有效）
- +b<BC>： 设置网格的边界条件。<BC> 可以取 g 、p 、n ，分别代表地理边界条件、周期边界条件和自然边界条件。对于后两种边界条件，可以在后面加上 x 或 y 表示边界条件仅对一个方向有效。比如 -nb+bnxpy 表明 X 方向使用自然边界条件，Y 方向使用周期边界条件
- +c： 假设原网格的 Z 值范围为 zmin 到 zmax，插值后的网格范围可能为超过这一范围，使用 +c 则将超过 zmin 和 zmax 的部分裁剪掉
- +t<threshold> 见官方文档

5.18 -r 选项

GMT 中的 2D 网格文件有网格线配准和像素配准两种配准方式，详情见网格配准 一节。

默认情况下，GMT 认为输入/输出的网格文件都采用网格线配准方式，使用 -r 选项，则 GMT 会认为输入/输出的网格文件为像素配准方式。

对于相同的网格区域（-R ）和网格间隔（-I ）而言，像素配准会比网格线配准少一列和一行数据。

5.19 -p 选项

在一张纸上画一个 2D 矩形，当从不同的方向去看这个矩形时，可以看到不同的形态。-p 选项可以用于选择从怎样的视角去看一张图。该选项的语法为:

```
-p[x|y|z]<azim>/<elev>[/<zlevel>][+w<lon0>/<lat0>][/<z0>]][+v<x0>/<y0>]
```

<azim> 和 <elev> 指定视角的方位角和海拔，默认值是 180/90 ，即视角在正上方。此处方位角是相对于北方向顺时针旋转的角度，<evel> 是相对于纸张平面旋转的角度，向上为正向下为负。

对于 3D 绘图而言（使用 -JZ|z ），默认情况下会在 Z 轴底部绘制 XY 平面内边框。使用 -p 选项再加上 <zlevel> 可以指定在某个 Z=<zlevel> 处绘制 2D 边框及标注。也可以用 -px|y 在 X=<xlevel> 或 Y=<ylevel> 处绘制 YZ 平面或 XZ 平面。默认使用 -pz ，即在 Z=<zlevel> 处绘制 XY 屏幕。

对于使用 -p 选项画出来的图，如果是作为动画的一帧，可以加上 + 以规定数据域的中心，也可以用 +w<lon0>/<lat0>/<z> 指定一个特殊的世界坐标点，该点会被投影到纸张的中心；也可以用 +v<x0>/<y0> 指定投影后视点的坐标。

5.20 -s 选项

-s 选项用于压制 Z 值等于 NaN 的记录的输出。其语法为:

```
-s[<cols>][a|r]
```

默认情况下，命令会输出所有记录，包括那些 Z 值等于 NaN 的记录。使用该选项可以抑制 Z 值为 NaNN 的记录的输出。

<cols> 是一系列用逗号分隔的列号或者列号范围，其中列号范围的格式为 <start>[:<inc>]:<stop> ，若省略 <inc> 则默认其值为 1。

比如 2,5,7 和 2:4 ，表示检测这些列数据是否是 NaN 。

r 表示 reverse，即只输出满足 NaN 的记录行。-sa 表示所有列中只要有一列为 NaN 即抑制其输出。

5.21 -t 选项

-t 用于设置某个图层的透明度，其后接 0 到 100 内的数字，默认值为 0，即不透明，100 表示全透明。

由于 PostScript 语言本身是不支持透明的，所以即便使用了 -t 选项，也无法在生成的 PS 文件中看到透明效果。想要看到透明效果，需要先用 psconvert 将 PS 文件转换为其他图片格式。

5.22 -x 选项

GMT 中的某些模块支持 OpenMP 多线程并行。默认情况下，GMT 会尝试使用所有可用的核。使用 -x 选项可以限制 GMT 所使用的核数，其语法为:

```
-x[[-]<n>]
```

- <n> 表示仅使用 <n> 个核（若 <n> 太大，则设置 <n> 为计算机的最大核数）
- -<n> 表示使用 all -<n> 个核（若 <n> 太小，则设置 <n> 为 1）

支持该选项的模块包括：greenspline 、grdmask 、grdmath 、grdfilter 、grdsample 、sph2grd 、grdgravmag3d 、talwani2d 、talwani3d 、x2sys_solve 。

5.23 -: 选项

GMT 在读入数据时，会认为第一列是 X 值，第二列是 Y 值。对于地理数据而言，即第一列是经度、第二列是纬度。

若要读入的数据中第一列是纬度、第二列是经度，则需要先交换第一列和第二列再读入，可以使用 -: 选项实现前两列数据的交换。

默认情况下，该选项同时对输入输出生效：

- -:i 表明该选项仅交换输入数据的前两列，等效于 -i1,0
- -:o 表明该选项仅交换输出数据的前两列，等效于 -o1,0

说明：

1. 该选项仅用于交换输入/输出的前两列数据，-R 选项中始终是经度在前，网格文件中经度始终是第一维度

第六章　配置 GMT

6.1 GMT 配置简介

6.1.1 简介

GMT 的每个模块都有众多选项，除此之外，GMT 还有 100 多个参数，用于控制图像的外观和数据的处理方式。

所有 GMT 参数都有一个系统默认值，这些默认值位于 GMT 全局配置文件 $*GMTHOME*/share/conf/gmt.conf 中。

当执行一个 GMT 程序时，首先读取 GMT 全局配置文件，将所有参数初始化为系统默认值，然后去搜索 gmt.conf 文件。若找到该文件，则会读取该文件中的参数值并覆盖系统默认值。

gmt.conf 文件的搜索路径按优先级排序为：./gmt.conf > ~/.gmt/gmt.conf > ~/gmt.conf 。即 GMT 程序首先会在当前目录寻找配置文件；若未找到，则到 ~/.gmt 目录下寻找；若仍为找到，则在家目录下寻找。

6.1.2 设计思路

GMT 将这 100 多个参数单独放置在配置文件 gmt.conf 中，主要是基于两方面的考虑：

1. 让每个模块的命令行语法中都涵盖如此多的参数是不实际的，这些参数有些很少或几乎不需要改变（比如地图投影过程中使用的地球椭率）

2. 针对不同的应用环境，设置不同的 GMT 配置文件，保存到不同的 gmt.conf 文件中，当需要使用一组特定的 GMT 配置时，可以在 GMT 命令行中使用 +<defaultfile> 来指定要使用的配置文件。比如，不同的期刊可能要求不同的字体和字号，你可以使用同样的脚本加上不同的配置文件来满足不同期刊的要求；又比如，用于出版的图件和用于 PowerPoint 演示的图件通常需要用不同的颜色和字号。将这些参数保存到单独的 gmt.conf 配置文件中，可以尽可能的避免由于环境不同而导致的图片微调。

6.1.3 修改 GMT 配置

GMT 系统自带的 gmt.conf 文件中对每个参数都给了一个合理的默认值，但有时用户需要修改这些参数值，GMT 提供了多种方法来实现修改。

1. 在单个 GMT 命令中使用 +<defaultfile> 的语法，来指定使用配置文件 <defaultfile>，该方法仅对单个命令有效:

```
gmt psxy ... +custom_gmt.conf > test.ps
```

2. 在脚本开始执行之前，将要使用的配置文件 gmt.conf 复制到当前目录，待脚本执行完毕后，删除该配置文件。该方法要求写脚本时要非常小心，因为若脚本因为错误出现中断，会影响到配置文件的删除，进而可能导致反效果的出现。

```
cp ~/somewhere/gmt.conf ./gmt.conf
gmt psxy ...
gmt pscoast ...
gmt psxy ...
rm gmt.conf
```

3. 用 gmtset 模块在脚本执行的过程中显式地修改 GMT 参数值

   比如，需要设置主标注的字体为 12p,Times-Bold,red

```
gmt gmtset FONT_ANNOT_PRIMARY 12p,Times-Bold,red
```

   gmtset 会修改当前目录下的 gmt.conf 文件中的相应参数值，进而影响到接下来其它 GMT 程序的执行。若当前目录没有 gmt.conf 文件，则 gmtset 会先复制系统自带的 gmt.conf 文件，再修改之。该命令修改的参数将一直生效，直到被新值覆盖。

4. 若你需要修改某些参数值，使得其在执行单个命令时有效，而不影响其他命令的执行效果，可以考虑在该命令行上使用 --PAR=value 语法。

   比如，针对某个 GMT 命令，为了临时设置浮点数的输出格式包含更多的小数位，而不影响其他命令的浮点数输出格式，可以在该命令中加上 --FORMAT_FLOAT_OUT=%.16lg 。

5. GMT 提供了“隔离”模式，使得仅在单个脚本执行的过程中修改配置，当脚本执行完毕后自动恢复到原始配置，见附录隔离模式 一节

一般情况下，仅推荐使用方法三和方法四。

在使用方法三的时候，需要注意一个潜在的问题。假如一个脚本中，只有三个命令，首先执行了 GMT 命令 A，然后使用 gmtset 将字体由默认字体 a 修改为字体 b，然后又执行了 GMT 命令 B。则命令 A 使用的是字体 a，命令 B 使用的是字体 b，这是自己想要的效果，到此为止都是没有问题的。若再次执行该脚本，由于当前目录下已经有了上一次执行生成的 gmt.conf 文件，

且文件中使用的是字体 b，则命令 A 受到该参数文件的影响使用了字体 b，`gmtset` 将字体 b 修改为字体 b，命令 B 使用字体 b。这导致了执行同一个脚本出现了不同的结果，经常会浪费很多的时间用来调试和排错。最好的做法是在脚本结束时删除当前目录下的参数文件，甚至删除其他一些中间文件。

```
gmt psxy ...
gmt pscoast ..
gmt gmtset ...
gmt grdimage ...
gmt psxy ...

rm gmt.conf gmt.history # 要养成删除临时文件的习惯
```

6.1.4 GMT 配置示例

下面列出部分会影响到绘图效果的 GMT 参数。

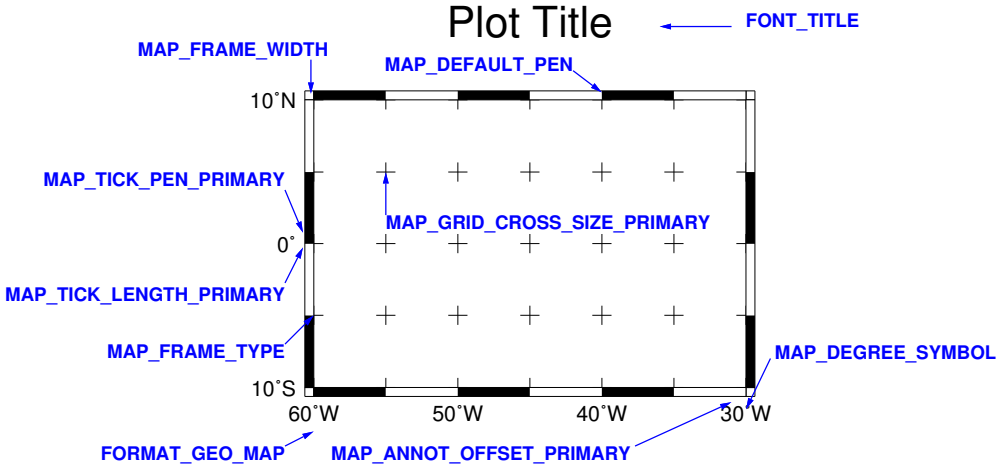


图 6.1: GMT 配置参数示例 1

6.2 COLOR 参数

下面列出所有与颜色相关的参数，参数的默认值在中括号内列出。

COLOR_BACKGROUND 图片背景色，数据 Z 值小于 CPT 文件中最小值所对应的背景色 [black]

COLOR_FOREGROUND 图片前景色，数据 Z 值大于 CPT 文件中最大值所对应的前景色 [white]

COLOR_NAN Z 值等于 `NaN` 时所使用的颜色 [127.5]

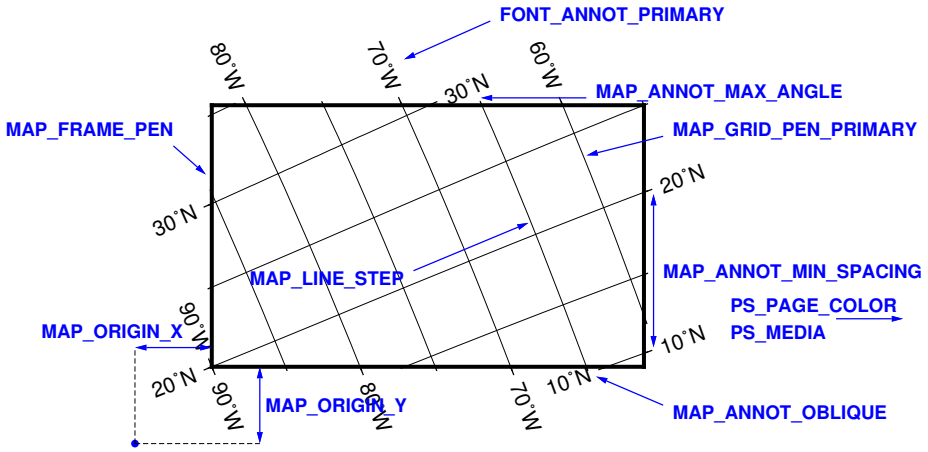


图 6.2: GMT 配置参数示例 2

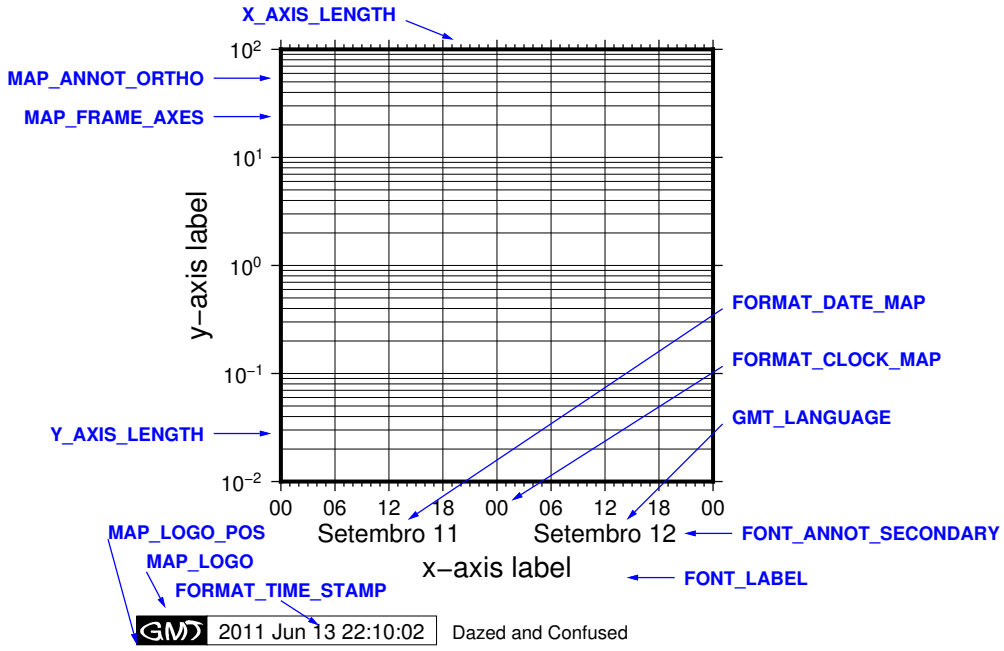


图 6.3: GMT 配置参数示例 3

某些绘图命令（比如 `grdimage` 、`psscale` ）可以使用强度文件（intensity file）来模拟光照效果。光照效果的实现，本质上是先将任意颜色转换成 HSV 模型，然后根据强度的正负，增大/减小 HSV 模型中的 S（饱和度）和 V（明度），以达到模拟光照的效果。下面的四个参数控制了模拟光照过程中 S 和 V 变化的极限值，估计是为了避免过亮或过暗的情况。

COLOR_HSV_MAX_S 最大正强度对应的 S 值，取值范围 0 到 1 [0.1]

COLOR_HSV_MIN_S 最小负强度对应的 S 值，取值范围 0 到 1 [1.0]

COLOR_HSV_MAX_V 最大正强度对应的 V 值，取值范围 0 到 1 [1.0]

COLOR_HSV_MIN_V 最小负强度对应的 V 值，取值范围 0 到 1 [0.3]

COLOR_MODEL 对 CPT 文件做插值时所使用的颜色空间 [none]

可以取如下值:

- `none` ： 使用 CPT 文件的注释行中指定的 `COLOR_MODEL`
- `rgb` ： 在 RGB 颜色空间中插值
- `hsv` ： 在 HSV 颜色空间中插值
- `cmyk` ： 假定文件是 CMYK 颜色，但在 RGB 空间插值

在 RGB 颜色空间内插值可能会产生异常的色调，而直接在 HSV 颜色空间插值则可以更好地保留色调值。

6.3 DIR 参数

下面列出所有与目录相关的参数，参数的默认值在中括号内列出。

DIR_DCW DCW 数据文件的路径，默认值为空，GMT 会自动猜测合理的路径值。

DIR_GSHHG 海岸线数据所在路径 [`$GMTHOME/_SHAREDIR/coast`]

DIR_DATA 存放数据文件的目录，默认值为空。

GMT 在命令中遇到文件名时，会首先在当前目录下寻找，若找不到，则会到参数 `DIR_DATA` 指定的目录中寻找，若找不到，则到环境变量 `${GMT_DATADIR}` 所指定的目录中寻找。

6.4 FONT 参数

本页面列出所有字体相关参数，参数的默认值在中括号内列出。

FONT 一次性设置所有 FONT 类参数（FONT_LOGO 除外）的默认字体，该参数不在 gmt.conf 中。

FONT_ANNOT 同时设置 FONT_ANNOT_PRIMARY 和 FONT_ANNOT_SECONDARY 的值。

FONT_ANNOT_PRIMARY 一级（Primary）标注的字体 [12p,Helvetica,black]

若在该参数的值前加上 + ，则其他字体、偏移量、刻度长度等参数值会相对于 FONT_ANNOT_PRIMARY 成比例缩放。

FONT_ANNOT_SECONDARY 二级（Secondary）标注所使用的字体 [14p,Helvetica,black]

FONT_LABEL 轴标签所使用的字体 [16p,Helvetica,black]

FONT_TITLE 图上方标题所使用的字体 [24p,Helvetica,black]

FONT_LOGO GMT 时间戳中字符串所使用的字体 [8p,Helvetica,black]

该参数中仅字体 ID 有效，字号及颜色均无效。

6.5 FORMAT 参数

下面列出所有与格式相关的参数，通常包括输入格式、输出格式和绘图格式三类，参数的默认值在中括号内列出。

日期数据的输入/输出/绘图格式：

FORMAT_DATE_IN 输入数据中日期字符串的格式模板 [yyyy-mm-dd]

日期字符串可以用公历表示，也可以用 ISO 周历表示。

对于公历而言，可以将如下模板做合理组合：

- yyyy： 四位年份
- yy： 两位年份
- mm： 两位月份
- o： 月份的名称简写
- dd： 两位日期
- jjj： 一年中的第几天

比如 ddmmyyy 、yy-mm-dd 、dd-o-yyyy 、yyyy/dd/mm 、yyyy-jjj 等。

对于 ISO 周历而言，其格式为 `yyyy[-]W[-]ww[-]d` 模板，表示某年的第 ww 周的第 d 天。比如 `yyyyWwwd` 或 `yyyy-Www` 等。

FORMAT_DATE_OUT 输出日期字符串时所使用的格式 [`yyyy-mm-dd`]

参考*FORMAT_DATE_IN* 的相关说明。除此之外：

- 若模板开头有一个 `-`，则所有的整数年月日在输出时会省略前置零。比如若使用模板 `-yyyy-mm-dd` 则输出类似于 `2012-1-3` 而不是 `2012-01-03`
- 若模板为 `-`，则输出时省略日期，日期和时间中的 `T` 也会被省略

FORMAT_DATE_MAP 绘制日期字符串时所使用的格式 [`yyyy-mm-dd`]

参考*FORMAT_DATE_IN* 和*FORMAT_DATE_OUT* 中的相关说明。除此之外，

- 绘制月份名时的 `mm` 可以用 `o` 替代，即图上显示 `Jan` 而不是 `01`
- 用 `u` 代替 `W[-]ww` ，即图上显示 `Week 10` 而不是 `W10`

所有的日期文本字符串都由*GMT_LANGUAGE* 、*FORMAT_TIME_PRIMARY_MAP* 和*FORMAT_TIME_SECONDARY_MAP* 控制。

时间数据的输入/输出/绘图格式：

FORMAT_CLOCK_OUT 输出时间字符串时所使用的格式 [`hh:mm:ss`]

默认使用 24 小时制。若要使用 12 小时制，可以在字符串的最后加上 `am` 、`AM` 、`a.m.` 、`A.M.` 。比如 `hh:mm:ss.xxx` 、`hh:mm` 、`hh:mm.xxx` 、`hha.m.` 等等。

- 若时间格式模板以 `-` 开头，则输出时间字符串时不会输出前置 0
- 若时间格式模板为 `-` ，则在输出日期时间时不输出时间字符串

FORMAT_CLOCK_IN 输入数据中时间数据的格式 [`hh:mm:ss`]

默认使用 24 小时制，即 `hh:mm:ss` ，若要使用 12 小时制，则在参数后加上 `am|pm|AM|PM` 。比如 `hh:mm` 或 `hh:mm:ssAM`

FORMAT_CLOCK_MAP 图上绘制时间字符串时所使用的格式 [`hh:mm:ss`]

地理坐标的输出/绘图格式：

FORMAT_GEO_OUT 控制地理坐标数据的输出格式 [`D`]

格式的通用形式有两类：

- `[+|-]D` ：表示将地理数据以浮点数的形式输出，浮点数的格式由*FORMAT_FLOAT_OUT* 决定

  - D：经度输出范围为 -180 到 180
  - +D：经度输出范围为 0 到 360
  - -D：经度输出范围为 -360 到 0
- [+|-]ddd[:mm[:ss]][.xxx][F|G]
  - ddd：固定格式的整型弧度
  - :：分隔符
  - mm：固定格式的整型弧分
  - ss：固定格式的整型弧秒
  - .xxx：前一个量的小数部分
  - F：用 WSEN 后缀来表示正负号
  - G：与 F 相同，但后缀前有一空格
  - +|-：默认经度范围为-180 到 180，若加正号则范围为 0 到 360，加负号则范围为-360 到 0

示例及效果：

- ddd:mmF => 35:45W
- ddd:mmG => 35:45 W
- ddd:mm:ss => 40:34:24
- ddd.xxx => 36.250

FORMAT_GEO_MAP 绘图时地理坐标的显示格式 [ddd.mm.ss]

格式的具体定义参考*FORMAT_GEO_OUT*，但具体格式会进一步由 -B 选项中的值控制。除此之外，还可以在格式后面加上 A 以表示绘制坐标的绝对值。

浮点数的输出/绘图：

FORMAT_FLOAT_OUT 双精度浮点数在输出时所使用的格式 [%.12lg]

具体的格式遵循 C 语言 printf 函数的格式定义，比如 %.3lf 。

若需要为不同列指定不同的输出格式，可以使用多个逗号分隔的 cols:format 形式。其中，cols 可以是列号（比如 5 代表数据的第六列），也可以是列范围（比如 3-7 表示第 4 到 8 列），不指定 cols 的格式将用于其他余下的列。比如 0:%.3lf,1-3:%.12lg,%lf

也可以列出 N 个用空格分隔的格式，这些格式分别应用到数据的前 N 列中，比如 `%.3lf %.2lf %lf` 。

FORMAT_FLOAT_MAP 以双精度浮点数形式绘制地图边框标注或等值线标注时所使用的格式 [`%.12lg`]

见*FORMAT_FLOAT_OUT* 中的相关说明。

FORMAT_TIME_MAP 同时设置 `FORMAT_TIME_PRIMARY_MAP` 和 `FORMAT_TIME_SECONDARY_MAP` 的值

FORMAT_TIME_PRIMARY_MAP 一级标注中月份、周名的格式 [full]

可以取如下值:

- `full` : 显示全称，比如 `January`
- `abbreviate` : 显示简称，比如 `Jan`
- `character` : 显示单个字符，比如 `J`

还可以使用 `Full` 、`Abbreviate` 、`Character` 表示所有名字均大写。

全称、简称以及单字符的定义，见 `$`*GMTHOME*`/share/localization/gmt_us.locale`

FORMAT_TIME_SECONDARY_MAP 二级标注中月份、周名的格式 [full]

见*FORMAT_TIME_PRIMARY_MAP* 中的相关说明。

FORMAT_TIME_STAMP GMT 时间戳中时间信息的显示格式 [`%Y %b %d %H:%M:%S`]

该选项的值用 C 函数 strftime 解析，故而理论上可以包含任意文本。

6.6 IO 参数

IO_HEADER 指定输入/输出的表文件中是否有文件头记录 [false]

可以取 `true|false` 。若值为 `true` ，则相当于使用了 `-h` 选项

IO_N_HEADER_RECS 在使用 `-h` 选项时，要跳过的文件头记录的数目 [0]

见*-h 选项* 和*表数据* 。

IO_LONLAT_TOGGLE 数据的前两列是纬度、经度而不是经度、纬度 [false]

该参数的作用与 `-:` 选项相同，见*-: 选项* 。其可以取如下值:

1. `false` 默认值，输入/输出数据均为 (x, y)

2. `true` 输入/输出数据均为 (y, x)

3. `IN` 仅输入数据为 (y, x)

4. `OUT` 仅输出数据为 (y, x)

IO_NAN_RECORDS 控制当读入的记录中的 X、Y 或 Z 包含 NaN 记录时的处理方式 [pass]

可以取如下值:

- `skip`： 直接跳过 NaN 记录，并报告 NaN 记录的数目
- `pass`： 将所有记录传递给程序

IO_COL_SEPARATOR GMT 输出 ASCII 表数据时列与列之间的分隔符 [tab]

可以取 `tab` 、`space` 、`comma` 和 `none`

IO_SEGMENT_MARKER 多段数据中每段数据开始的标识符 [`>`]

见*表数据* 中的相关介绍。若希望输入和输出数据中使用不同的数据段标识符，则可以使用逗号分隔输入和输出数据的段标识符，比如 `>,:` 。

有两个特殊的标识符:

1. `B` 表示将空行作为数据段开始的标识符

2. `N` 表示将一个 NaN 记录作为数据段开始的标识符

IO_NC4_CHUNK_SIZE 控制写 netCDF 文件时的分块大小 [auto]

见*netCDF 文件格式* 中的相关介绍，以及官方文档对该参数的说明。

IO_NC4_DEFLATION_LEVEL 输出 netCDF4 格式的数据时所使用的压缩等级 [3]

可以取 0 到 9 的整数，0 表示不压缩，9 表示最大压缩。低压缩率可以提高性能并减少文件尺寸，而高压缩率虽然可以进一步减小文件尺寸，但却需要更多的处理时间。

IO_SEGMENT_BINARY 二进制数据中，某个记录的所有值都是 NaN 时该如何解释 [2]

默认情况下，当二进制数据中某个记录的值为 NaN 的列数超过了 `IO_SEGMENT_BINARY` 的值时，则将该记录解释为二进制数据中的数据段头记录。将该参数赋值为 0 或 off 可以关闭这一特性。

IO_GRIDFILE_SHORTHAND 是否支持自动识别网格文件后缀的功能 [false]

见*网格文件后缀* 一节。若设置为 `true` ，则会检测每个网格文件的后缀是否在用户自定义文件后缀中；若为 `false` ，则不检测用户自定义文件后缀。

IO_GRIDFILE_FORMAT GMT 默认使用的网格文件格式 [nf]

见网格文件 一节。

6.7 MAP 参数

MAP_DEFAULT_PEN 设置所有与 `-W` 选项相关的画笔属性的默认值 [default,pen]

在参数值的前面加上 `+` 可以覆盖其他 PEN 相关参数中的颜色。

MAP_FRAME_AXES 要绘制/标注的轴 [WSEN]

默认值为 WSEN，绘制并标注四条边，可以通过 `-B` 选项控制实际绘制的边。详情见*-B 选项* 选项。

MAP_FRAME_TYPE 底图边框类型 [fancy]

可选值包括 `inside|plain|graph|fancy|fancy+` 。一般情况下，`fancy` 边框类型仅适用于投影后的 X、Y 方向平行于经度纬度方向的情况下，比如 rectangular 投影、polar 投影。对于某些投影，只能使用 plain 底图，即便 `MAP_BASEMAP_TYPE` 被设置为 fancy。

下图给出了不同的底图边框类型的效果：

MAP_FRAME_PEN 绘制底图类型为 plain 时边框的画笔属性 [`thicker,black`]

MAP_FRAME_WIDTH 设置底图类型为 fancy 时的边框宽度 [`5p`]

MAP_ORIGIN_X 新绘图在纸张上的原点的 X 坐标 [1i]

MAP_ORIGIN_Y 设置新绘图在纸张上的原点的 Y 坐标 [1i]

MAP_LOGO 是否在左下角绘制 GMT 时间戳 [false]

可以取 `true|false` ，等效于在命令行中使用 `-U` 选项。

MAP_LOGO_POS GMT 时间戳相对于当前绘图原点的对齐方式与位置 [`BL/-54p/-54p`]

MAP_TITLE_OFFSET 图标题的底部与轴标注（或轴标签）的顶部之间的距离 [14p]

MAP_SCALE_HEIGHT 地图比例尺的高度 [5p]

MAP_TICK_PEN 同时设置 `MAP_TICK_PEN_PRIMARY` 和 `MAP_TICK_PEN_SECONDARY` 的值

MAP_TICK_PEN_PRIMARY 一级刻度的画笔属性 [thinner,black]

MAP_TICK_PEN_SECONDARY 二级刻度的画笔属性 [thinner,black]

MAP_TICK_LENGTH 同时设置 `MAP_TICK_LENGTH_PRIMARY` 和 `MAP_TICK_LENGTH_SECONDARY` 的值

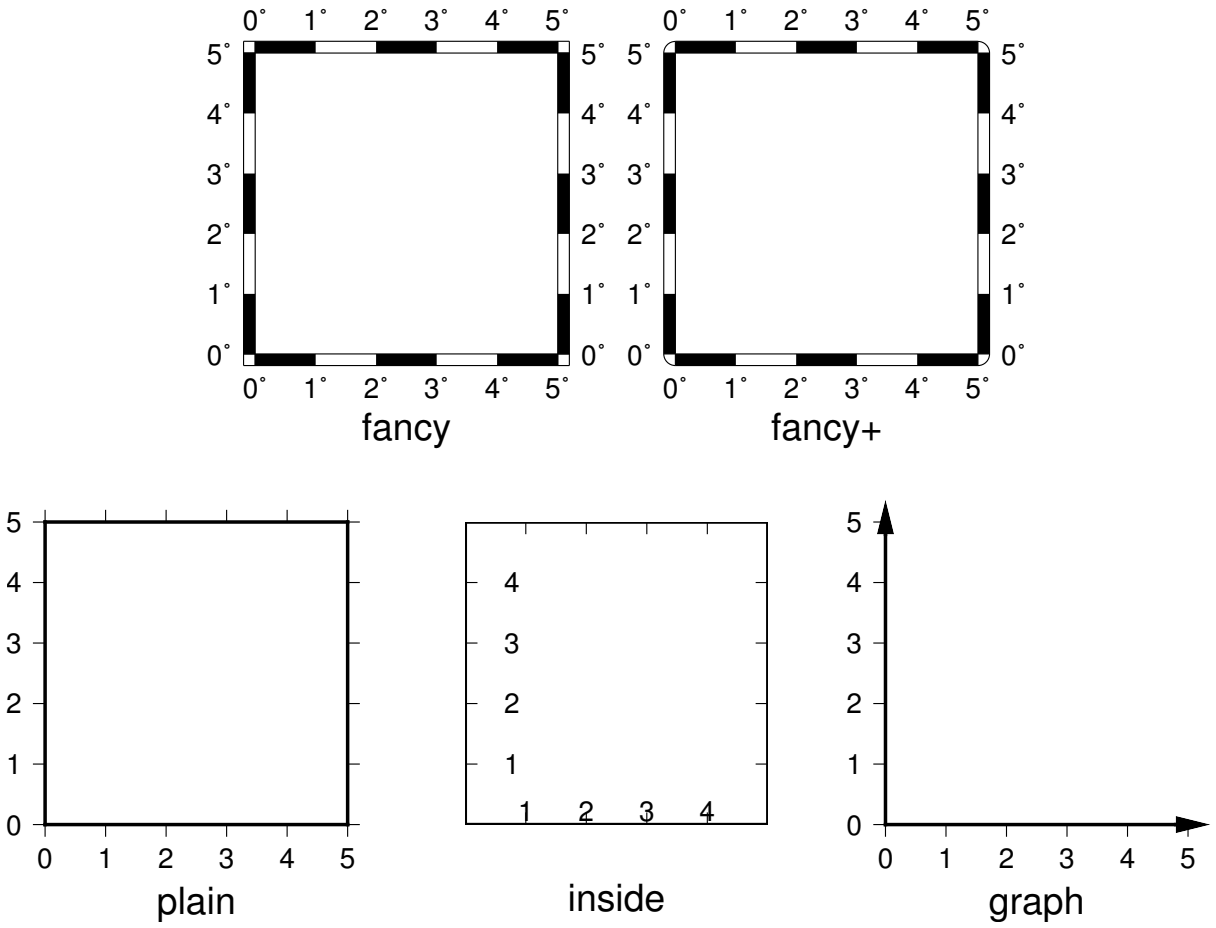


图 6.4: GMT 底图边框类型

MAP_TICK_LENGTH_PRIMARY 一级刻度的主刻度和次刻度的长度 [5p/2.5p]

若只给定一个长度值，则次刻度的长度假定为主刻度的一半

MAP_TICK_LENGTH_SECONDARY 二级刻度的主刻度和次刻度的长度 [15p/3.75p]

若只给定一个长度值，则次刻度的长度假定为主刻度的 25%

MAP_LINE_STEP 绘制线段时所使用的最大步长 [0.75p]

地理投影下，两点之间会用大圆路径连接，因而 GMT 需要先计算大圆路径上的其他中间点的坐标，并将这些点用直线连起来。若该步长太大，会导致大圆路径看上去很不光滑。

MAP_GRID_PEN 同时设置 `MAP_GRID_PEN_PRIMARY` 和 `MAP_GRID_PEN_SECONDARY` 的值

MAP_GRID_PEN_PRIMARY 一级网格线的线条属性 [default,black]

MAP_GRID_PEN_SECONDARY 二级网格线的线条属性 [thinner,black]

MAP_GRID_CROSS_SIZE 同时设置 `MAP_GRID_CROSS_SIZE_PRIMARY` 和 `MAP_GRID_CROSS_SIZE_SECONDARY` 的值

MAP_GRID_CROSS_SIZE_PRIMARY 一级网格交叉线的大小，0 表示绘制连续的网格线 [0p]

MAP_GRID_CROSS_SIZE_SECONDARY 二级网格交叉线的大小，0 表示绘制连续的网格线 [0p]

MAP_ANNOT_OFFSET 同时设置 `MAP_ANNOT_OFFSET_PRIMARY` 和 `MAP_ANNOT_OFFSET_SECONDARY` 的值

MAP_ANNOT_OFFSET_PRIMARY 一级标注的开始位置与刻度尾端间的距离 [5p]

MAP_ANNOT_OFFSET_SECONDARY 二级标注的底部与 secondary 标注的顶部之间的距离 [5p]

MAP_LABEL_OFFSET 轴标注底部与轴标签顶部间的距离 [8p]

MAP_VECTOR_SHAPE 矢量箭头的形状 [0]

取值范围为-2 到 2。0 表示矢量头为三角形，1 表示箭头形状，2 表示打开的 V 字。其他的中间值代表了两种形状的中间值。

MAP_DEGREE_SYMBOL 在地图上绘制“度”时所使用的符号 [ring]

可以取 `ring|degree|colon|none`

MAP_ANNOT_MIN_ANGLE 对于某些倾斜投影方式而言，如果标注的基线与地图的边界之间的夹角小于该值，则不绘制标注。合理的取值范围为 `[0,90]` [20]

MAP_ANNOT_MIN_SPACING 在某些倾斜投影中，相邻两个标注之间的最小距离，若标注的距离小于该值，则不绘制 [0p]

MAP_ANNOT_ORTHO 控制哪些轴的标注垂直于轴 [we]

该参数可以将 `wesnz` 做任意组合

MAP_ANNOT_OBLIQUE 见官方文档

MAP_POLAR_CAP 控制网格线在两极附近的显示，见官方文档

6.8 PROJ 参数

本节列出投影相关参数，参数的默认值在中括号内列出。

PROJ_LENGTH_UNIT 长度量的默认单位 [c]

见单位 一节。

PROJ_ELLIPSOID 地图投影中使用的地球椭球标准 [WGS-84]

GMT 支持几十种地球椭球标准，具体可取值见官方文档。除此之外，GMT 还支持自定义椭球，用户只需按照固定的格式对椭球命名，GMT 会从椭球名字中提取半长轴以及扁率。可用的格式如下：

- `a`：球半径为 a，单位为 `m`，扁率为零。比如 `6378137`
- `a,inv_f`：`inv_f` 为扁率的倒数，比如 `6378137,298.257223563`
- `a,b=semi_minor`：`semi_minor` 为半短轴长度，单位为 `m`。比如 `6378137,b=6356752.3142`
- `a,f=flattening`：`flattening` 为扁率，比如 `6378137,f=0.0033528`

需要注意，对于某些全球投影，GMT 会对选中的地球椭球做球状近似，将扁率设为零，并使用其平均半径。当 GMT 做此类近似时，会给出警告信息。

PROJ_AUX_LATITUDE 球体近似时的辅助纬线 [authalic]

在使用大圆弧距离计算方式时，需要将真实地球近似为一个半径为*PROJ_MEAN_RADIUS* 的球体，在做球体近似时需要选择合适的辅助纬线。可选值包括

- `authalic`
- `geocentric`
- `conformal`

- `meridional`
- `parametric`
- `none`

当设置为除 `none` 外的其他值时，GMT 会在计算距离前，将大圆弧距离计算时使用的两点中任意一点的纬度转换成辅助纬度。

PROJ_MEAN_RADIUS 地球/行星的平均半径 [authalic]

在计算两点间的大圆弧距离或区域的表面积时才会被使用。可选值包括

- `mean (R_1)`
- `authalic (R_2)`
- `volumetric(R_3)`
- `meridional`
- `quadratic`

PROJ_SCALE_FACTOR 修改某些投影的地图缩放因子以减小面积失真

- Polar Stereographic：默认值为 0.9996
- UTM：默认值为 0.9996
- Transverse Mercator：默认值为 1

PROJ_GEODESIC 指定大地测量距离中所使用的算法 [Vincenty]

可以取：

1. `Vincenty` 默认值，精确到 0.5mm
2. `Rudoe` given for legacy purpose
3. `Andoyer` 精度为 10 米量级，比 `Vincenty` 快 5 倍

6.9 PS 参数

本节列出所有与 PS 相关的参数，参数的默认值在中括号内列出。

PS_CHAR_ENCODING 字符集编码方式 [ISOLatin1+|Standard+]

在处理文本中的八进制码时所使用的字符集编码方式，可选值包括：Standard 、Standard+ 、ISOLatin1 、ISOLatin1+ 和 ISO-8859-x （其中 x 取值为 [1-10] 或 [13-15]）。若安装 GMT 时使用 SI 单位制，则默认值为 ISOLatin1 编码；否则使用 Standard+ 编码。

PS_LINE_CAP 控制线段的**端点**的绘制方式 [butt]

可以取如下值：

- butt ： 不对端点做特殊处理，即端点是矩形（默认值）
- round ： 端点处为直径与线宽相等的半圆弧
- square ： 端点处为边长与线宽相等的半个正方形

下图展示了 PS_LINE_CAP 取不同值时线段端点的区别。需要注意，图中三条线段的长度是相同的，但因为参数设置不同而导致实际线段长度看上去有些不一样。

图 6.5: PS_LINE_CAP 控制线段端点绘图效果

说明：当 PS_LINE_CAP=round 时，若线段长度为零，则线段将以圆的形式存在，这可以用于创建圆点线，此时将同一条线绘制两次，每次使用不同的相移和颜色，则可以创建颜色变化的线条。

PS_LINE_JOIN 控制线段拐点的绘制方式 [miter]

可以取 miter 、round 、bevel

下图展示了 PS_LINE_JOIN 取不同值时线段拐点的绘图效果。当线宽较小时，几乎看不出来区别，这里为了显示的效果，将线宽设置为 20p 。

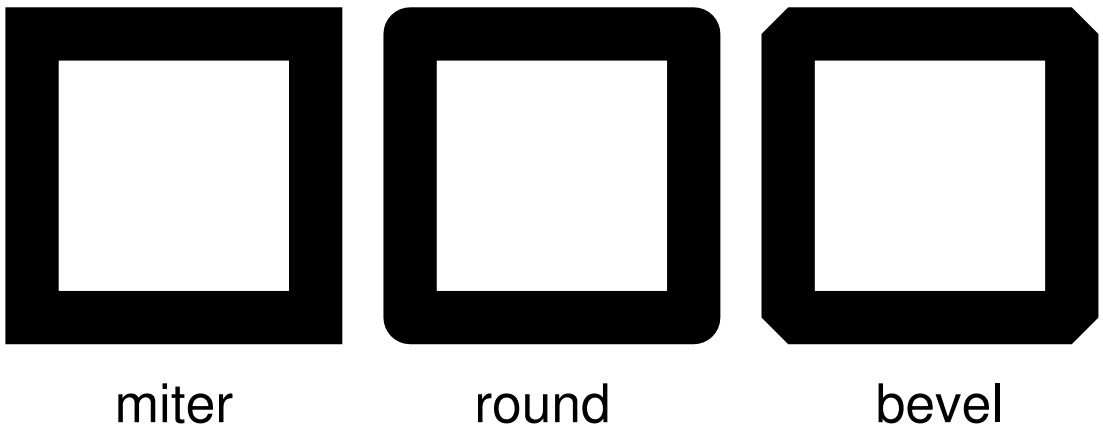


图 6.6: PS_LINE_JOIN 控制线段拐点绘制效果

PS_MITER_LIMIT 设置 mitered 拐点的角度阈值 [35]

当两个相交的线段之间的夹角小于该阈值时，则该拐角会被 bevelled 而不是被 mitered。该参数的取值范围为 0 到 180。若设置为 0，则使用 PS 的默认值（11 度），若设置为 180，则所有拐角都会被 beveled。

PS_MEDIA 设置当前纸张尺寸 [a4|letter]

下表列出了 GMT 预定义的若干种纸张尺寸及其对应的宽度和高度（单位为 points）。

表 6.1: GMT 预定义纸张大小

Media	width	height	Media	width	height
A0	2380	3368	archA	648	864
A1	1684	2380	archB	864	1296
A2	1190	1684	archC	1296	1728
A3	842	1190	archD	1728	2592
A4	595	842	archE	2592	3456
A5	421	595	flsa	612	936
A6	297	421	halfletter	396	612
A7	210	297	statement	396	612
A8	148	210	note	540	720
A9	105	148	letter	612	792
A10	74	105	legal	612	1008
B0	2836	4008	11x17	792	1224
B1	2004	2836	tabloid	792	1224
B2	1418	2004	ledger	1224	792
B3	1002	1418			
B4	709	1002			
B5	501	709			

用户还可以用 `WxH` 的格式完全自定义纸张尺寸，其中 `W` 和 `H` 分别为纸张的宽度和高度。比如 `12cx12c` 表示纸张为宽度和高度都为 12 厘米。

若某些尺寸经常使用，用户还可以在 `${GMT}/share/conf/gmt_custom_media.conf` 中添加自定义的纸张尺寸。

PS_PAGE_COLOR 设置纸张的背景色 [white]

PS_PAGE_ORIENTATION 设置纸张方向 [landscape]

可以取 `portrait` 或 `landscape` ，见画布 一节。

PS_SCALE_X 绘图时 X 方向的全局比例 [1.0]

用于实现图像的整体缩放

PS_SCALE_Y 绘图时 Y 方向的全局比例 [1.0]

用于实现图像的整体缩放

PS_TRANSPARENCY 设置生成 PS 文件所使用的透明模式 [Normal]

可取值包括 Color、ColorBurn、ColorDodge、Darken、Difference、Exclusion、HardLight、Hue、Lighten、Luminosity、Multiply、Normal、Overlay、Saturation、SoftLight、Screen

PS_COLOR_MODEL 设置生成 PS 代码时颜色所使用的颜色模型 [rgb]

可以取 RGB、HSV、CMYK 或 GRAY。若设置为 HSV，其不会影响绘图过程中使用 RGB 指定的颜色；若设置为 GRAY，则所有的颜色都将使用 YIQ 方法转换成灰度。

PS_COMMENTS 生成的 PS 代码中是否包含注释信息 [false]

若为 `true` ，则生成的 PS 文件中会包含注释，用于解释文件中操作的逻辑，当你需要手动编辑 PS 文件时比较有用。默认情况下，其值为 `false` ，即 PS 文件中不会包含注释，此时生成的 PS 文件更小。

PS_IMAGE_COMPRESS 设置 PS 中的图像压缩算法 [deflate,5]

可以取值为

- `rle`： Run-Length Encoding scheme
- `lzw`： Lempel-Ziv-Welch compression
- `deflate[,level]`： DEFLATE compression，`level` 可以取 1 到 9；
- `none`： 不压缩，相当于 `deflate,5` 。

6.10 TIME 参数

本节列出所有时间相关参数，参数的默认值在中括号内列出。

TIME_EPOCH 指定所有相对时间的参考时刻 [1970-01-01T00:00:00]

其格式为 `yyyy-mm-ddT[hh:mm:ss]` 或 `yyyy-Www-ddTT[hh:mm:ss]` 。

TIME_UNIT 指定相对时间数据相对于参考时间的单位 [s]

可以取：

1. y：年；假定一年 365.2425 天；
2. o：月；假定所有月是等长的；

3. d：天；

4. h：时；

5. m：分钟；

6. s：秒；

TIME_SYSTEM TIME_EPOCH 和 TIME_UNIT 的合并版

即指定 TIME_SYSTEM 相当于同时指定了 TIME_EPOCH 和 TIME_UNIT 。可取如下值：

1. JD ： 等效于-4713-11-25T12:00:00 d

2. MJD ： 等效于 1858-11-17T00:00:00 d

3. J2000 ： 等效于 2000-01-01T12:00:00 d

4. S1985 ： 等效于 1985-01-01T00:00:00 s

5. UNIX ： 等效于 1970-01-01T00:00:00 s

6. RD0001 ： 等效于 0001-01-01T00:00:00 s

7. RATA ： 等效于 0000-12-31T00:00:00 d

该参数并不存在于 gmt.conf 中，当指定该参数时，其会被自动转换为 TIME_EPOCH 和 TIME_UNIT 对应的值。

TIME_WEEK_START 指定周几是一周的第一天，可取值为 Monday 或 Sunday。

TIME_Y2K_OFFSET_YEAR 当用两位数字表示四位数字的年份时，TIME_Y2K_OFFSET_YEAR 给定了 100 年序列的第一年 [1950]

比如，若 TIME_Y2K_OFFSET_YEAR=1729，则数字 29 到 99 分别表示 1729 到 1799，而数字 00 到 28 则表示 1800 到 1828。默认值为 1950，即 00 到 99 表示的年份范围为 1950 到 2049。

TIME_INTERVAL_FRACTION 见官方文档中的说明

TIME_REPORT 见官方文档中的说明

TIME_IS_INTERVAL 见官方文档中的说明

6.11 其他参数

本节介绍 GMT 中的其他参数，参数的默认值在中括号内列出。

GMT_COMPATIBILITY 是否开启兼容模式 [4]

- 若做值为 4，表示兼容 GMT4 语法并给出警告
- 若值为 5，则表示不兼容 GMT4 语法，严格遵守 GMT5 语法，遇到 GMT4 语法时直接报错

GMT_VERBOSE 控制 GMT 命令的 verbose 级别 [c]

可选值包括 `quiet` 、`normal` 、`compat` 、`verbose` 、`long` 、`debug` ，也可以直接使用每个级别的第一个字母。参考*-V 选项* 一节。

GMT_TRIANGULATE 选择 triangulate 命令中源码的来源 [Watson]

`triangulate` 命令有两个版本的源码，`Watson` 的版本遵循 GPL，`Shewchuk` 的版本不遵循 GPL。该选项用于控制要使用哪个版本，`Shewchuk` 版本拥有更多功能。

GMT_LANGUAGE 绘制月份、星期几等时所使用的语言 [US]

该参数的默认值为 `US` ，所以在绘图时一月会显示 `January` 而不是 一月 。可以通过修改该参数设置 GMT 所使用的语言。该参数可以取值包含 CN1、CN2、UK、US 等。详细列表见 GMT 官方文档 gmt.conf

每种语言的定义位于 `$GMTHOME/share/localization` 中。读者可以自定义自己的语言，并将其放在该目录或 `~/.gmt` 目录下。

实际使用时，还涉及到编码和字体问题，所以想做出中文效果还需要一些特殊处理。

GMT_HISTORY GMT 历史文件 `gmt.history` 的处理方式 [true]

- `true` 可以读写
- `readonly` 只能读不能写
- `false` 不显示历史文件

GMT_INTERPOLANT 程序中一维插值所使用的算法 [akima]

1. `linear` ： 线性插值
2. `akima` ： akima's spline
3. `cubic` ： natural cubic spline
4. `none` ： 不插值

GMT_EXPORT_TYPE 控制表数据的数据类型，仅被外部接口使用 [double]

可以取 `double` 、`single` 、`[u]long` 、`[u]int` 、`[u]short` 、`[u]char`

GMT_EXTRAPOLATE_VAL 外插时超过数据区时如何处理 [NaN]

可选值包括：

1. `NaN`： 区域范围外的值一律为 NaN
2. `extrap`： 使用外插算法计算的区域外的值
3. `extrapval,val`： 设置区域外的值为 `val`

GMT_CUSTOM_LIBS 自定义的 GMT 共享库文件，默认值为空

GMT 支持自定义模块。用户可以自己写一个 GMT 模块，并将其编译成 GMT 兼容的动态函数库，并告知 GMT 该函数库的信息，就可以通过 `gmt xxx` 的语法调用自定义的模块，以实现扩充 GMT 功能的目的。

该参数用于指定自定义动态库函数的路径，多个路径之间用逗号分隔。路径可以是共享库文件的绝对路径，也可以是其所在的目录。若路径是一个目录名，该目录必须需斜杠或反斜杠结尾，表明使用该目录下的全部共享库文件。在 Windows 下，若目录名是 `/`，则在 `${GMTHOME}/bin/gmt_plugins` 目录下寻找库文件。

GMT_FFT 要使用的 FFT 算法 [auto]

可以取：

1. `auto`： 自动选择合适的算法
2. `fftw[,planner]`：FFTW 算法，其中 `planner` 可以取 `measure|patient|exhaustive`
3. `accelerate` OS X 下使用 Accelerate Framework
4. `kiss`： kiss FFT
5. `brenner`： Brenner Legacy FFT

第七章　投影方式

GMT 中的投影方式大致可以分为三大类：

1. 笛卡尔线性投影
2. 极坐标投影
3. 地图投影

其中地图投影又可以细分为几十种不同的投影方式。

GMT 支持的投影方式如下图所示。

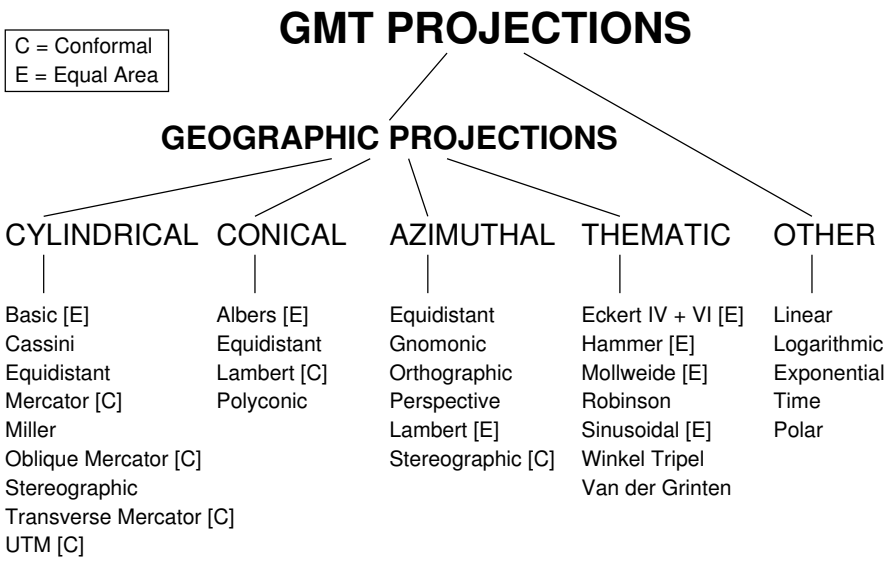


图 7.1: GMT 支持的地图投影及坐标变换

这一章会给出所有投影方式，但地图投影部分不会翻译。

7.1 -Jx：笛卡尔变换

GMT 中笛卡尔坐标变换分为三类：

- 线性坐标变换

- Log_{10} 坐标变换
- 指数坐标变换

在开始之前，先用 `gmtmath` 命令生成两个数据以供接下来示例使用:

```
gmt gmtmath -T0/100/1  T SQRT = sqrt.d
gmt gmtmath -T0/100/10 T SQRT = sqrt.d10
```

7.1.1 笛卡尔线性变换

笛卡尔线性变换的使用场景可以分为三类:

1. 常规的浮点数坐标
2. 地理坐标
3. 日期时间坐标

7.1.1.1 常规浮点数坐标

对于常规的浮点型数据而言，选择笛卡尔线性坐标意味着对输入坐标做简单的线性变换 `u'=au+b` ，将输入坐标 `u` 投影到纸张坐标 `u'` 。

线性投影可以通过四种方式指定:

- `-Jx<scale>` X 轴和 Y 轴拥有相同的比例尺 `<scale>`
- `-JX<width>` X 轴和 Y 轴拥有相同的长度 `<width>`
- `-Jx<xscale>/<yscale>` 分别为 X 轴和 Y 轴指定不同的比例尺
- `-JX<width>/<height>` 分别为 X 轴和 Y 轴指定不同的长度

下面的命令用线性投影将函数 $y = \sqrt{x}$ 用笛卡尔线性变换画在图上:

```
gmt psxy -R0/100/0/10 -JX3i/1.5i -Bag -BWSne+gsnow -Wthick,blue,- -P -K sqrt.d > GMT_
↪linear.ps
gmt psxy -R -J -St0.1i -N -Gred -Wfaint -O sqrt.d10 >> GMT_linear.ps
```

绘图效果如下图所示:

说明:

- 正常情况下，X 轴向右递增，Y 轴向上递增。有些时候可能需要 X 轴向左递增或者 Y 轴向下递增（比如 Y 轴是深度时），只要将轴的比例尺或者轴长度设置为负值即可。

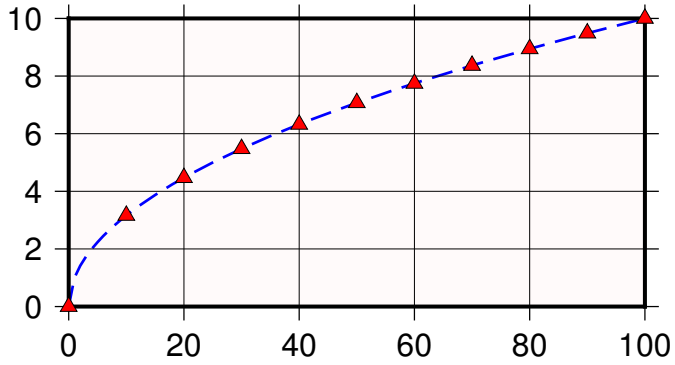


图 7.2: 笛卡尔坐标的线性变换

- 若指定 X 轴的长度，并设置 Y 轴的长度为 0，则会根据 X 轴的长度和范围计算出 X 轴的比例尺，并对 Y 轴使用相同的比例尺，进而计算出 Y 轴的长度，即 `-JX10c/0c` ，`-JX0c/10c` 同理。

7.1.1.2 地理坐标

理论上地理坐标应该用地理投影画，而不应该用线性投影，但是有时候可能的确需要使用线性投影。用线性投影绘制地理坐标时会碰到一个问题，即经度有一个 360 度的周期性。因而在使用线性投影时需要通知 GMT 数据实际上是地理坐标。有三种办法：

1. 在 `-R` 后、数据范围前加上 `g` 或 `d` ，比如 `-Rg-55/305/-90/90`
2. 在 `-Jx` 或 `-JX` 选项的最后加上 `g` 或 `d` ，比如 `-JX10c/6cd`
3. 使用 `-fg` 选项

下面的例子用线性投影绘制了一个中心位于 125ºE 的世界地图：

```
gmt set MAP_GRID_CROSS_SIZE_PRIMARY 0.1i MAP_FRAME_TYPE FANCY FORMAT_GEO_MAP ddd:mm:ssF
gmt pscoast -Rg-55/305/-90/90 -Jx0.014i -Bagf -BWSen -Dc -A1000 -Glightbrown -
↪Wthinnest \
        -P -Slightblue > GMT_linear_d.ps
```

7.1.1.3 日期时间坐标

时间日期坐标也可以用线性投影绘制，此时需要告诉 GMT 输入坐标是绝对时间还是相对时间。

可以通过在 `-Jx` 或 `-JX` 的最后加上 `T` 或 `t` ，不过实际上 `-R` 选择中已经指定了时间范围，所以没有必要在 `-J` 和 `-R` 选项中都指定。当 `-R` 和 `-J` 选项给出的坐标类型相冲突时，GMT 会给出警告，并以 `-JX` 选项为准。

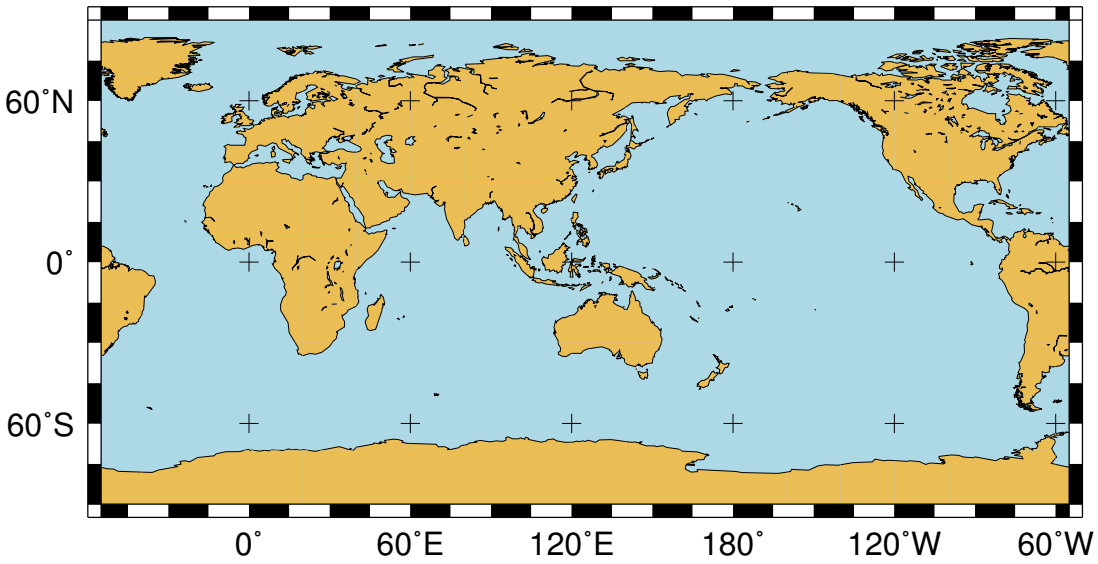


图 7.3: 地理坐标的线性变换

```
gmt set FORMAT_DATE_MAP o TIME_WEEK_START Sunday FORMAT_CLOCK_MAP=-hham \
        FORMAT_TIME_PRIMARY_MAP full
gmt psbasemap -R2001-9-24T/2001-9-29T/T07:0/T15:0 -JX4i/-2i -Bxa1Kf1kg1d \
              -Bya1Hg1h -BWsNe+glightyellow -P > GMT_linear_cal.ps
```

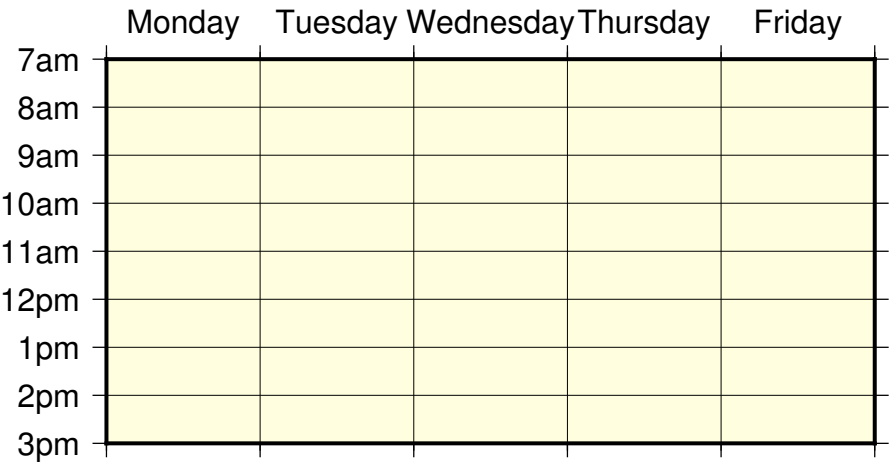


图 7.4: 日期时间坐标的线性变换

7.1.2 笛卡尔对数投影

对数变换 $\log_{10}$ 的数学表示是 $u' = a\log_{10}(u) + b$ ，可以通过在比例尺或轴长度后加上 `l` 指定。

下面的命令绘制了一个 X 轴为对数轴 Y 轴为线性轴的图:

```
gmt psxy -R1/100/0/10 -Jx1.5il/0.15i -Bx2g3 -Bya2f1g2 -BWSne+gbisque \
         -Wthick,blue,- -P -K -h sqrt.d > GMT_log.ps
```

```
gmt psxy -R -J -Ss0.1i -N -Gred -W -O -h sqrt.d10 >> GMT_log.ps
```

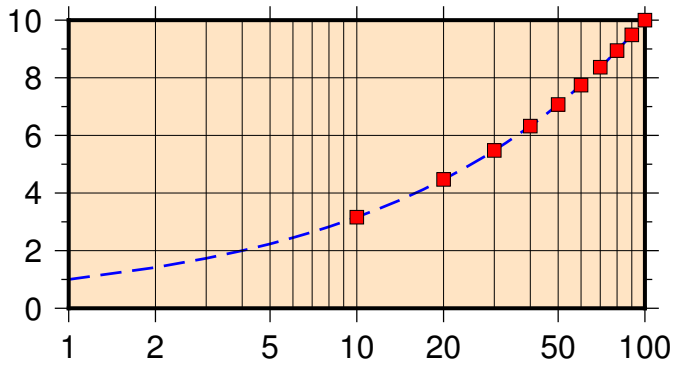


图 7.5: 对数投影

注意：若想要 X 轴和 Y 轴都使用对数投影，且 X 轴和 Y 轴比例尺不同，则必须在指定每个轴的比例尺时分别加上 `l` ，例如 `-JX10cl/6cl` 。

7.1.3 笛卡尔指数投影

指数投影的函数表示是 $u' = au^b + c$ ，使得用户可以绘制类似 x^p vs y^q 这样的函数关系。如果选 `p=0.5` 、`q=1` 则相对于绘制 `x` 与 $\sqrt{x}$ 的函数曲线。

要使用指数投影，需要在比例尺或轴长度后加上 `p<exp>` ，其中 `<exp>` 是要使用的指数。例如:

```
gmt psxy -R0/100/0/10 -Jx0.3ip0.5/0.15i -Bxa1p -Bya2f1 -BWSne+givory \
         -Wthick -P -K sqrt.d > GMT_pow.ps
gmt psxy -R -J -Sc0.075i -Ggreen -W -O sqrt.d10 >> GMT_pow.ps
```

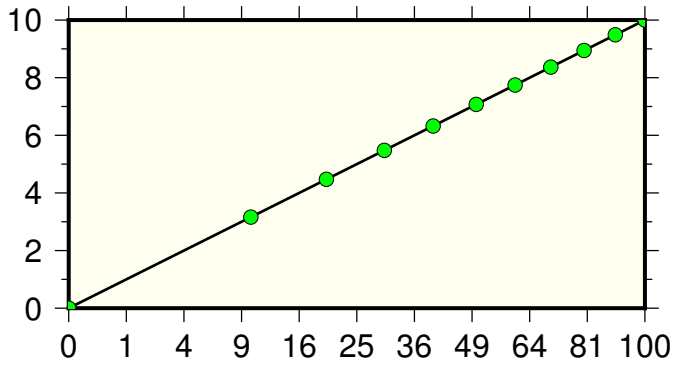


图 7.6: 指数变换

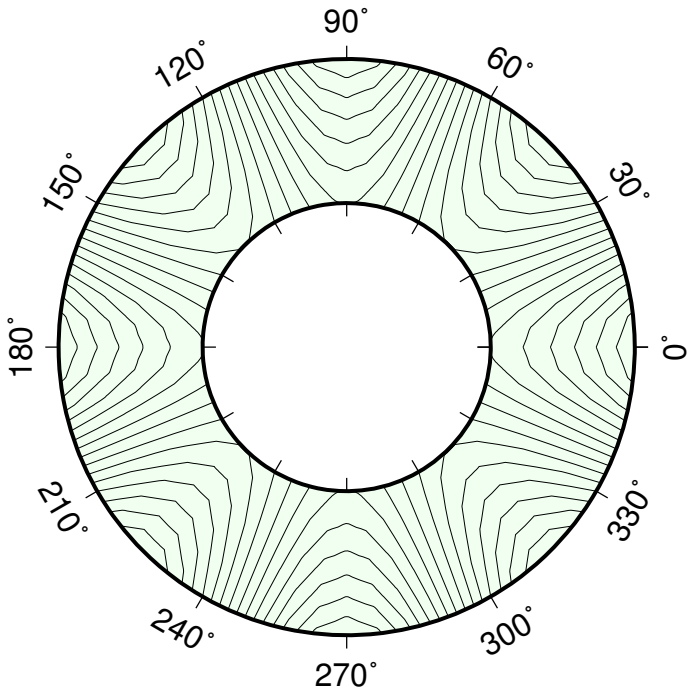


图 7.7: 极坐标（θ, r）的线性投影

7.2 -Jp：极坐标线性投影

该投影方式将极坐标（角度 θ 和半径 r ）转换到纸张上的位置。此时投影函数为 $x' = f(\theta, r)$ 和 $y' = g(\theta, r)$ ，X 和 Y 是相互耦合的，且具有 360 度的周期性，因而与地图投影类似。

1. 通常，θ 是相对于水平方向逆时针旋转的角度，但是也可以加一个 $theta_0$ 作为所有角度的共同偏移量。即 $x' = f(\theta, r) = ar\cos(\theta - \theta_0) + b$ 和 $y' = g(\theta, r) = ar\sin(\theta - \theta_0) + c$ 。
2. 或者，θ 也可以解释成相对于北方向顺时针旋转的角度，当然也可以为所有角度指定一个共同的偏移量 θ_0 ，即 $x' = f(\theta, r) = ar\cos(90 - (\theta - \theta_0)) + b$ 和 $y' = g(\theta, r) = ar\sin(90 - (\theta - \theta_0)) + c$ 。

极坐标投影可以通过如下方式定义：

- 用 `-Jp<scale>` 指定比例尺或用 `-JP<width>` 指定整张图的宽度
- 在 `p` 或 `P` 后插入 `a` 表明输入数据是顺时针的方位角而不是逆时针的角度
- 在后面加上 `/<theta_0>` 表明输入数据的偏移量，默认值为 0
- 在后面加上 `r` 可以反转径向的方向
- 在后面加上 `z` to annotate depths rather than radius.

下面的示例，用 `grdmath` 命令生成了一个 $z(\theta, r) = r^2 \cdot \cos 4\theta$ 的网格文件，并用 `grdcontour` 绘图:

```
gmt grdmath -R0/360/2/4 -I6/0.1 X 4 MUL PI MUL 180 DIV COS Y 2 POW MUL = tt.nc
gmt grdcontour tt.nc -JP3i -B30 -BNs+ghoneydew -P -C2 -S4 --FORMAT_GEO_MAP=+ddd > GMT_
↪polar.ps
```

7.3 -Ja：Lambert 方位等面积投影

维基链接: https://en.wikipedia.org/wiki/Lambert_azimuthal_equal-area_projection

该投影由 Lambert 于 1772 年发展得到，通常用于绘制大区域地图（例如整个大洲或半球），该投影是方位等面积投影，但不支持透视。在投影的中心畸变为 0，离投影中心距离越远，畸变越大。该投影的参数为:

```
-JA<lon>/<lat>/[<distance>/]<width>
-Ja<lon>/<lat>/[<distance>/]<scale>
```

- <lon>/<lat> 投影中心坐标
- <distance> 投影中心到边界的角度，默认值为 90，即距离投影中心各 90 度，即整个半球
- <width> 地图宽度
- <scale> 地图比例尺 1:xxxx 或 <radius>/<latitude> （<radius> 是纬线 <latitude> 与投影中心在图上的距离）

7.3.1 矩形地图

对于此投影而言，经线和纬线通常不是直线，因而不适合用于指定地图边界。因而本例中通过指定区域的左下角（0ºE/40ºS）和右上角（60ºE/10ºS）的坐标来指定区域范围:

```
gmt set FORMAT_GEO_MAP ddd:mm:ssF MAP_GRID_CROSS_SIZE_PRIMARY 0
gmt pscoast -R0/-40/60/-10r -JA30/-30/4.5i -Bag -Dl -A500 -Gp300/10 \
            -Wthinnest -P > GMT_lambert_az_rect.ps
```

7.3.2 半球地图

要绘制半球地图，需要指定区域范围为整个地球。下图绘制了以南美洲为中心的半球图:

```
gmt pscoast -Rg -JA280/30/3.5i -Bg -Dc -A1000 -Gnavy -P > GMT_lambert_az_hemi.ps
```

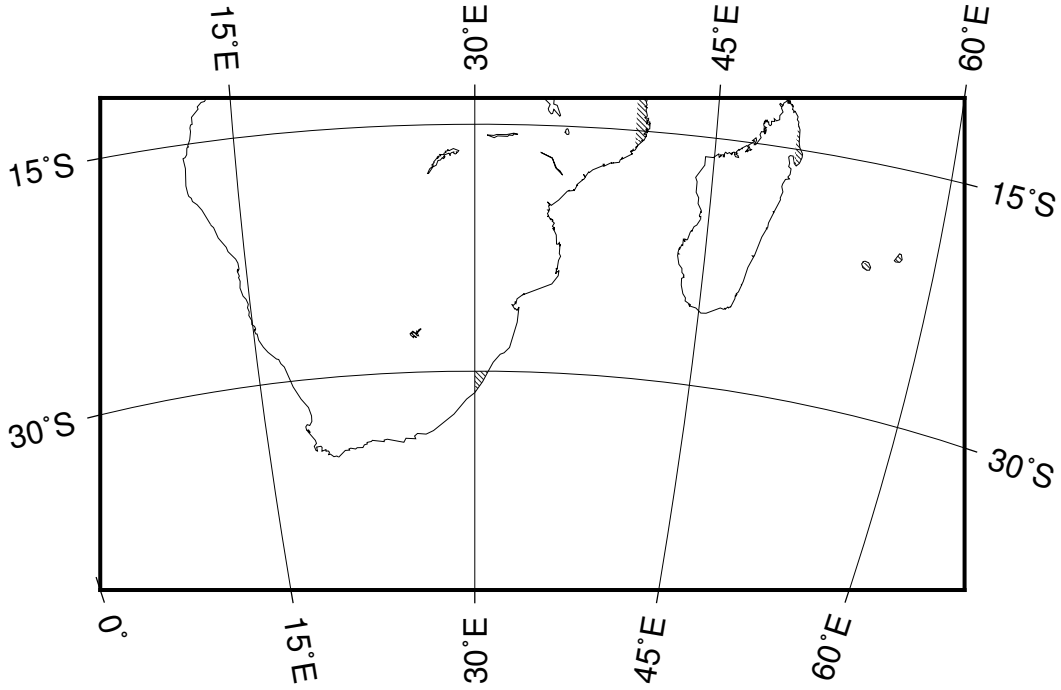


图 7.8: 使用 Lambert 方位等面积投影绘制矩形地图

图 7.9: 使用 Lambert 方位等面积投影绘制半球地图

地球物理学中，在绘制震源机制解时，就是将三维的辐射花样信息投影到一个水平面内。投影的方式有两种：Schmidt 网和 Wulff 网。其中 Schmidt 网使用的就是 Lambert 方位等面积投影（中心经纬度为 0/0），Wulff 网使用的则是等角度的立体投影。两种震源球投影方式如下图所示：

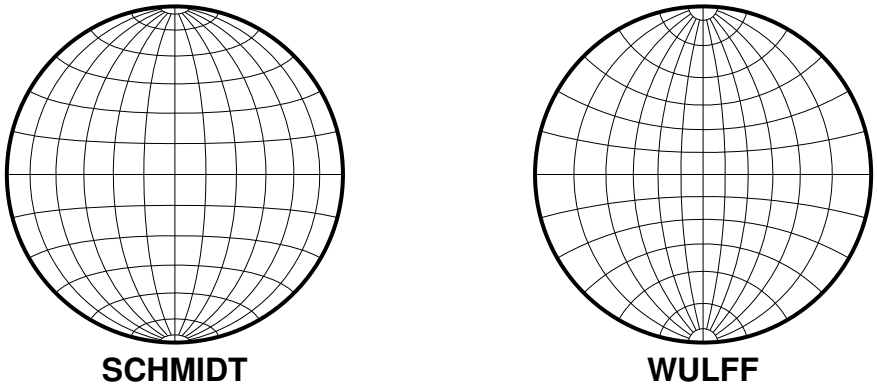


图 7.10: 震源球投影：等面积的 Schmidt 网和等角度的 Wulff 网

7.4 -Jb：Albers 圆锥等面积投影

维基链接：https://en.wikipedia.org/wiki/Albers_projection

此投影由 Albers 于 1805 年提出，主要用于绘制东西方向范围很大的地图，尤其是美国地图。

此投影是圆锥、等面积投影。纬线是不等间隔分布的同心圆，在地球南北极处分布较密。经线则是等间隔分隔，并垂直切割纬线。

在两条标准纬线处，比例尺和形状的畸变最小；在两者之间，沿着纬线的比例尺太小；在两者外部，沿着经线的比例尺则太大。沿着经线，则完全相反。

该投影方式的参数为:

```
-JB<lon>/<lat>/<lat1>/<lat2>/<width>
-Jb<lon>/<lat>/<lat1>/<lat2>/<scale>
```

- `<lon>` 和 `<lat>` 是投影中心的位置
- `<lat1>` 和 `<lat2>` 是两条标准纬线

下图绘制了台湾附近的区域，投影中心位于 125ºE/20ºN，25 度和 45 度纬线是两条标准纬线:

```
gmt set MAP_GRID_CROSS_SIZE_PRIMARY 0
gmt pscoast -R110/140/20/35 -JB125/20/25/45/5i -Bag -Dl -Ggreen -Wthinnest \
            -A250 -P > GMT_albers.ps
```

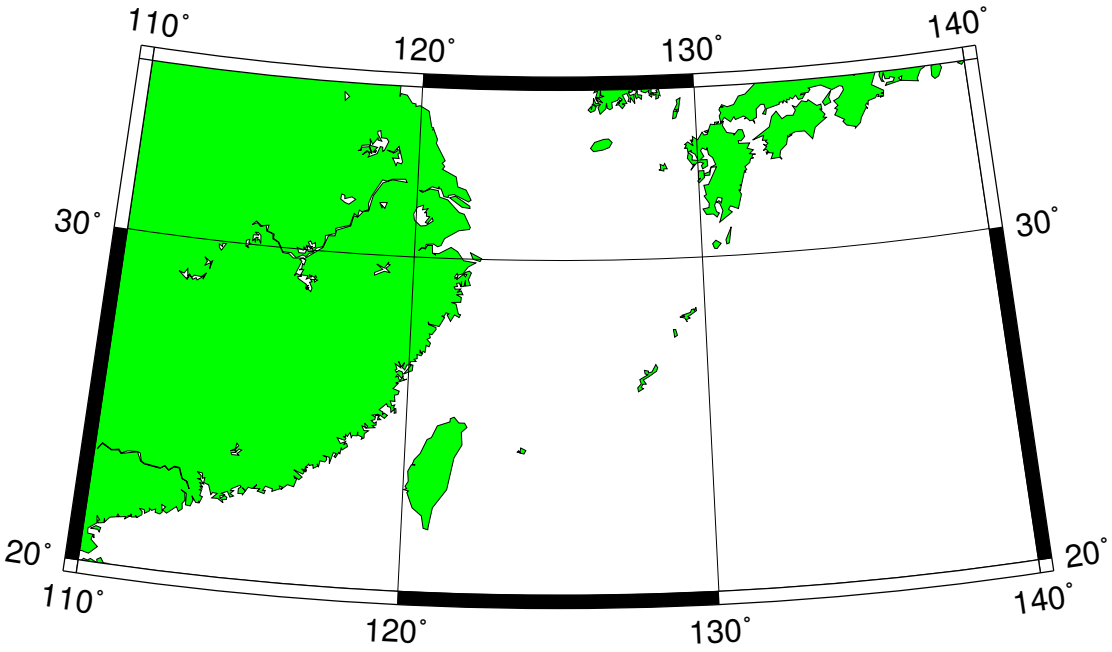


图 7.11: Albers 圆锥等面积投影

7.5 -Jc：Cassini 圆柱投影

维基链接：https://en.wikipedia.org/wiki/Cassini_projection

此圆柱投影由 César-François Cassini de Thury 于 1745 年在调查法国时提出。其偶尔也被称为 Cassini-Soldner 投影，因为后者提供了更加精确的数学分析得到了椭球下的公式。

此投影既不保角也不等面积，而是介于二者之间的一种投影方式。沿着中心经线的畸变最小，适合绘制南北方向区域范围较大的地图。其中，中心经线、距离中心经线 90 度的两条经线以及赤道是直线，其余经线和纬线都是复杂的曲线。

该投影方式的参数为:

```
-JC<lon>/<lat>/<width>
-JC<lon>/<lat>/<scale>
```

其中，`<lon>/<lat>` 为中心的经纬度。

示例:

```
gmt pscoast -R7:30/38:30/10:30/41:30r -JC8.75/40/2.5i -Bafg -LjBR+c40+w100+f+o0.15i/0.
↪2i \
    -Gspringgreen  -Dh -Sazure -Wthinnest -Ia/thinner -P --FONT_LABEL=12p > GMT_
↪cassini.ps
```

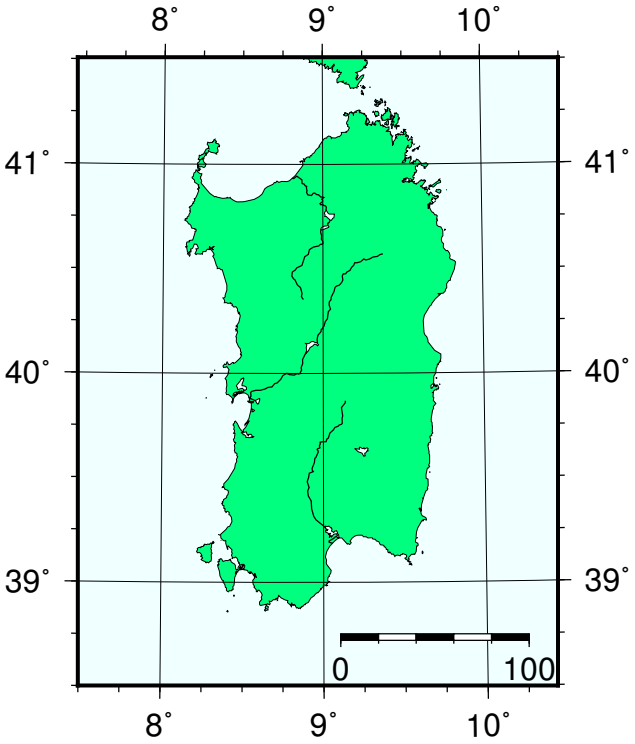


图 7.12: Cassini 投影绘制 Sardinia 岛

7.6 -Jcyl_stere：圆柱立体投影

维基链接：https://en.wikipedia.org/wiki/Gall_stereographic_projection

圆柱立体投影不像其他的圆柱投影那样显著，但由于其相对简单且能够克服其他圆柱投影的缺点（比如高纬度的畸变），故而仍然被使用。立体投影是透视投影，将整个球沿着赤道上的对跖点投影到一个圆柱上。该圆柱于两条距赤道等间距的标准纬线处穿过球体。

该投影的参数为:

```
-JCyl_stere/[<lon>[/<lat>]]/<width>
-Jcyl_stere/[<lon>[/<lat>]]/<scale>
```

- `<lon>` 中心经线，若省略则使用区域的中心经线
- `<lat>` 标准纬线，默认是赤道。若使用，则必须指定中心经线

一些比较流行的标准纬线的选择如下：

Miller's modified Gall	66.159467º
Kamenetskiy's First	55º
Gall's stereographic	45º
Bolshoi Sovietskii Atlas Mira or Kamenetskiy's Second	30º
Braun's cylindrical	0º

示例:

```
gmt set FORMAT_GEO_MAP dddA
gmt pscoast -R-180/180/-60/80 -JCyl_stere/0/45/4.5i -Bxa60f30g30 -Bya30g30 -Dc -A5000 \
            -Wblack -Gseashell4 -Santiquewhite1 -P > GMT_gall_stereo.ps
```

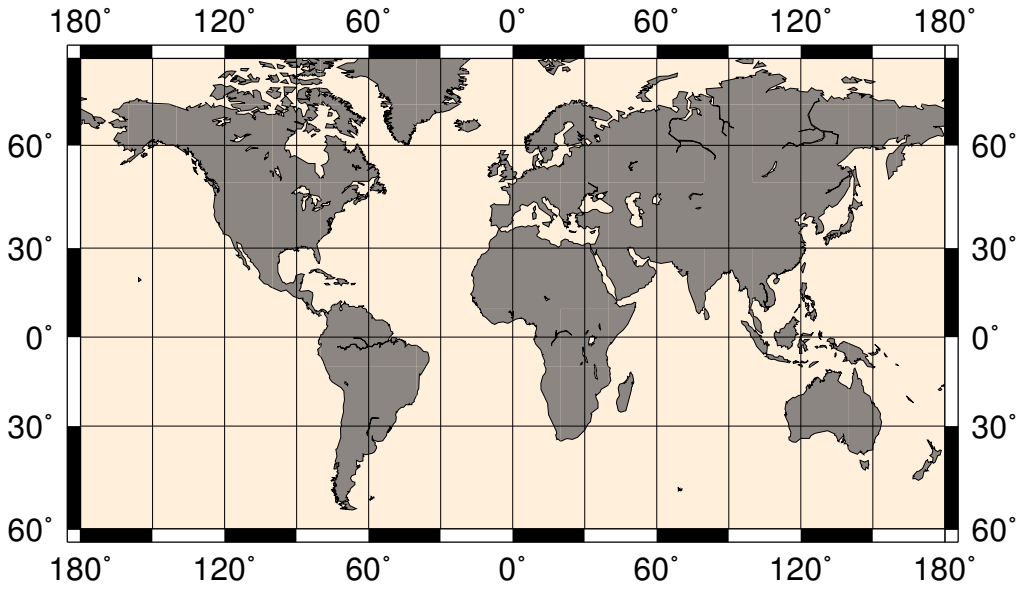


图 7.13: 使用 Gall 立体投影绘制世界地图

7.7 -Jd：等距圆锥投影

维基链接：https://en.wikipedia.org/wiki/Equidistant_conic_projection

等距圆锥投影由希腊哲学家 Claudius Ptolemy 于公元 150 年提出。其既不是保角也不是等面积，二者两种的折衷。在所有经线以及标准纬线上比例尺没有畸变。

该投影的参数为:

```
-JD<lon>/<lat>/<lat1>/<lat2>/<width>
-Jd<lon>/<lat>/<lat1>/<lat2>/<scale>
```

- `<lon>/<lat>` 投影中心位置

- <lat1>/<lat2> 两条标准纬线

等距圆锥投影常用于绘制小国家的地图集:

```
gmt set FORMAT_GEO_MAP ddd:mm:ssF MAP_GRID_CROSS_SIZE_PRIMARY 0.05i
gmt pscoast -R-88/-70/18/24 -JD-79/21/19/23/4.5i -Bag -Di -N1/thick,red \
            -Glightgreen -Wthinnest -P > GMT_equidistant_conic.ps
```

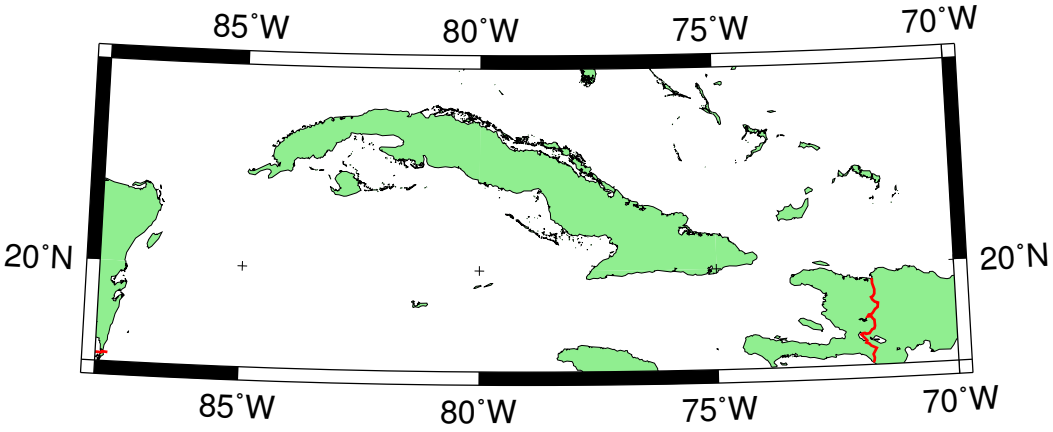


图 7.14: 等距圆锥地图投影

7.8 -Je：方位等距投影

维基链接：https://en.wikipedia.org/wiki/Azimuthal_equidistant_projection

方位投影最显著的特征是在图上测量的从中心到任意一点的距离是真实的。因而，地图上以投影中心为圆心的圆在真实地球上与投影中心是等距离的。同时，从中心出发的任意方向也是真实的。该投影常用于展示多个地理位置与指定点的距离图。

该投影的参数为:

```
-JE<lon>/<lat>[/<distance>]/<width>
-Je<lon>/<lat>[/<distance>]/<scale>
```

- <lon>/<lat> 投影中心的经纬度
- <distance> 是边界距离投影中心的度数，默认值为 180，即绘制全球图
- <scale> 可以取 1:xxxx 格式，也可以是 <radius>/<latitude> （表示从投影中心到纬线 <latitude> 在图上的距离为 <radius> ）

下图中，投影中心为 100ºW/40ºN，离投影中心 180 度的点在图中的最外边界处:

```
gmt pscoast -Rg -JE-100/40/4.5i -Bg -Dc -A10000 -Glightgray -Wthinnest -P > GMT_az_
↪equidistant.ps
```

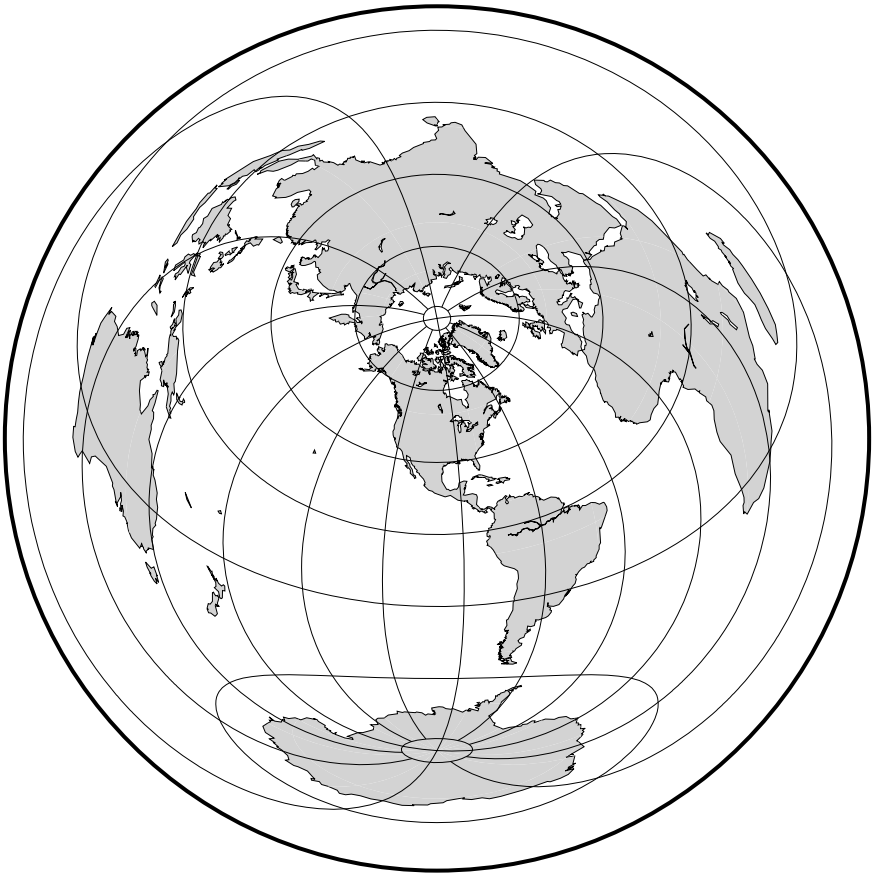


图 7.15: 使用等距方位投影绘制全球图

7.9 -Jf：球心方位投影

维基链接：https://en.wikipedia.org/wiki/Gnomonic_projection

此投影是一个从中心投影到与表面相切的一个平面的透视投影。此投影既不等面积也不保角，且在半球的边界处有很大畸变，但从投影中心出发的方向是真实的。大圆会被投影成直线。

该投影的参数为:

```
-JF<lon>/<lat>[/<distance>]/<width>
-Jf<lon>/<lat>[/<distance>]/<scale>
```

- `<lon>/<lat>` 投影中心的经纬度
- `<distance>` 地图边界到投影中心的角度，默认值为 60 度

- <scale> 可以是 1:xxxx 也可以是 <radius>/<latitude> （<radius> 是投影中心到纬线 <latitude> 在图上的距离）

示例:

```
gmt pscoast -Rg -JF-120/35/60/4.5i -B30g15 -Dc -A10000 -Gtan -Scyan -Wthinnest \
            -P > GMT_gnomonic.ps
```

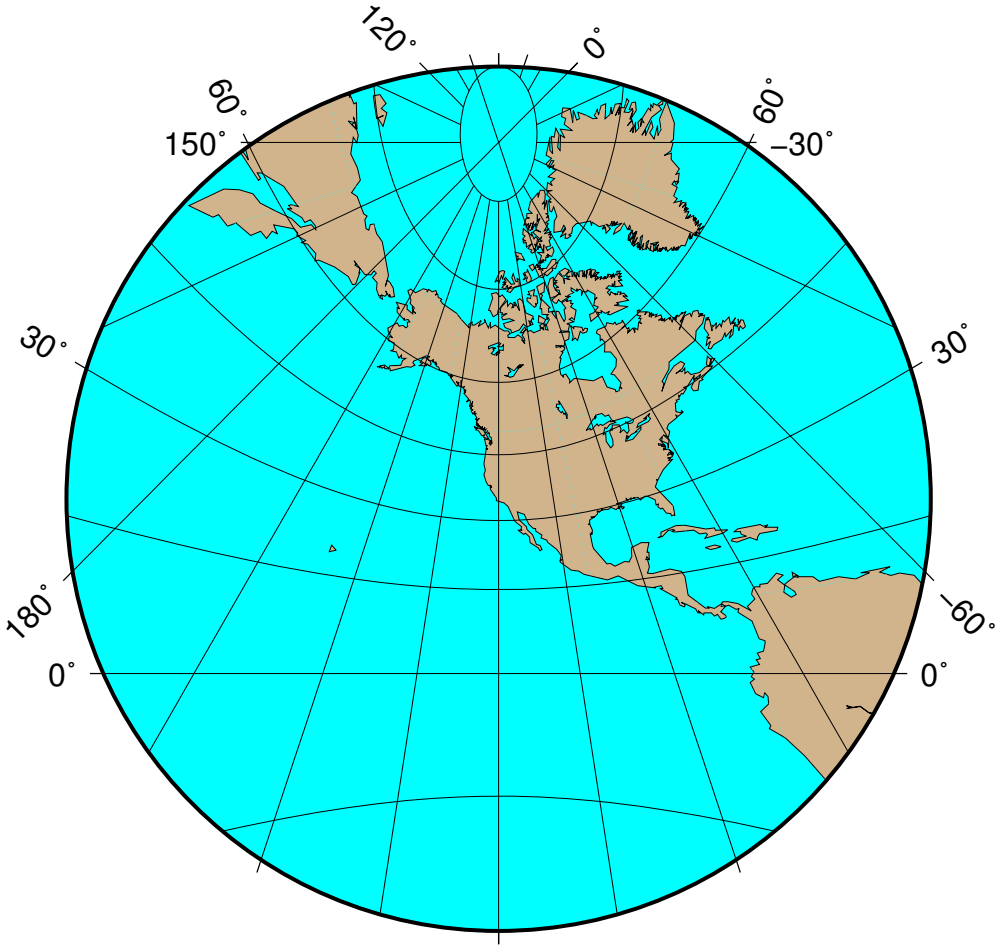


图 7.16: 球心方位投影

7.10 -Jg：正交投影

正交方位投影是一种从无穷远距离处的透视投影，因而常用于绘制从外太空看地球。与 Lambert 等面积投影以及立体投影类似，一次只能看到一个半球。该投影既不是等面积也不是保角，在半球边界处有较大得畸变。从投影中心出发的任意方向是真实的。

该投影的参数为:

```
-JG<lon>/<lat>[/<distance>]/<width>
-Jg<lon>/<lat>[/<distance>]/<scale>
```

- <lon>/<lat> 是投影中心位置
- <distance> 是边界离投影中心的度数，默认值为 90
- <scale> 地图比例尺 1:xxxx 或 <radius>/<latitude> （<radius> 是纬线 <latitude> 与投影中心在图上的距离）

示例:

```
gmt pscoast -Rg -JG-75/41/4.5i -Bg -Dc -A5000 -Gpink -Sthistle -P > GMT_orthographic.ps
```

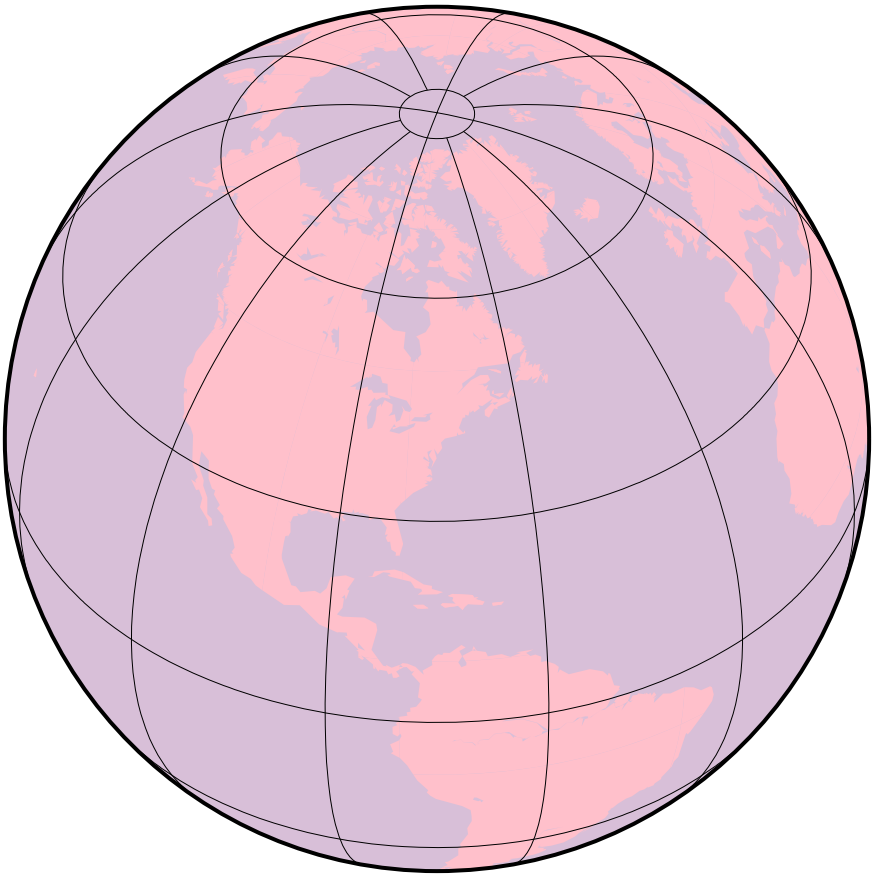


图 7.17: 使用正交投影绘制半球

-Jg 加上更多的参数时还可以用于绘制透视投影，以在二维平面内模拟从太空看三维的地球。具体的参数为:

```
-JG<lon>/<lat>/<alt>/<az>/<tilt>/<twist>/<width>/<height>
```

- <lon>/<lat> 投影中心的经纬度

- `<alt>` 是观察者所处的海拔，单位为 km。若该值小于 10，则假定是观察者相对于地心的距离，若距离后加了 `r` ，则表示观察者与地心的距离（单位为 km）。
- `<az>` 观察者的方位角，默认值为 90 度，即从东向观测
- `<tilt>` 倾角（单位为度），默认值为 60 度。若值为 0 则表示在顶点直接向下看，值为 60 则表示在顶点处沿着水平方向 30 度角的方向观察
- `<twist>` 扭转角度，默认值为 180 度。This is the boresight rotation (clockwise) of the image. The twist of 180º in the example mimics the fact that the Space Shuttle flies upside down.
- `<width>/<height>` 是视角的角度，单位为度，默认值为 60。This number depends on whether you are looking with the naked eye (in which case you view is about 60º wide), or with binoculars, for example.
- `<scale>` as 1:xxxxx or as radius/latitude where radius is distance on map in inches from projection center to a particular oblique latitude

示例:

```
gmt pscoast -Rg -JG4/52/230/90/60/180/60/60/5i -Bx2g2 -By1g1 -Ia -Di -Glightbrown \
            -Wthinnest -P -Slightblue --MAP_ANNOT_MIN_SPACING=0.25i > GMT_perspective.ps
```

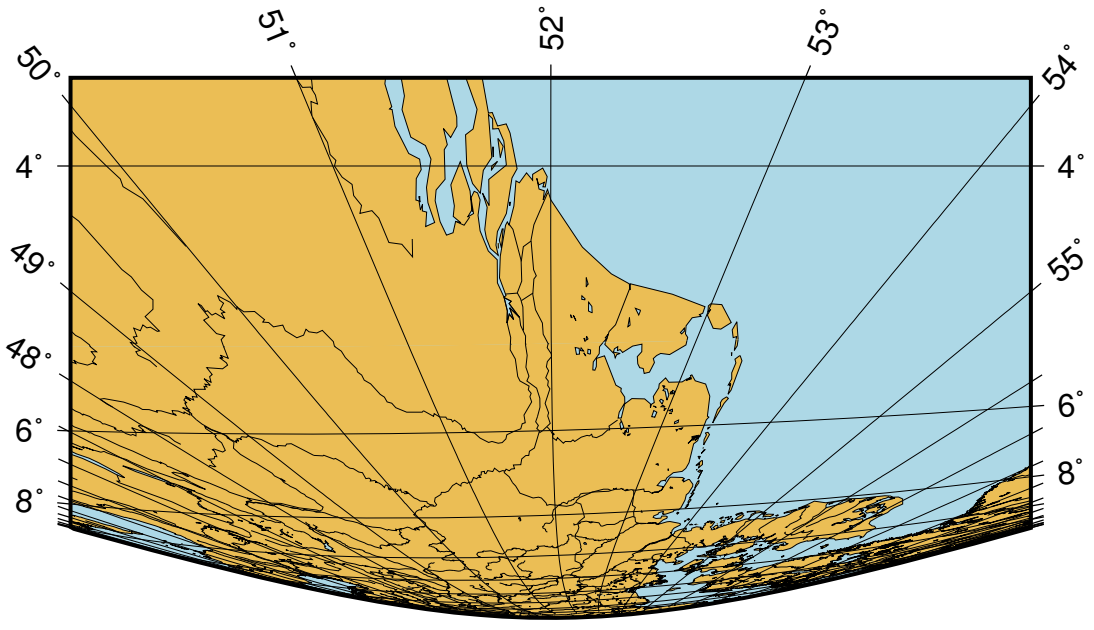


图 7.18: 透视投影

7.11 -Jh：等面积 Hammer 投影

维基链接：https://en.wikipedia.org/wiki/Hammer_projection

等面积 Hammer 投影由 Ernst von Hammer 于 1892 年提出，也被称为 Hammer-Aitoff 投影（Aitoff 投影与之看起来相似，但不等面积）。投影后的边界是一个椭圆，赤道和中心经线是执行，其余纬线和经线都是复杂曲线。

该投影的参数为:

```
-JH[<lon>/]<width>
-Jh[<lon>/]<scale>
```

<lon> 是中心经线，默认位于地图区域的中心。

示例:

```
gmt pscoast -Rg -JH4.5i -Bg -Dc -A10000 -Gblack -Scornsilk -P > GMT_hammer.ps
```

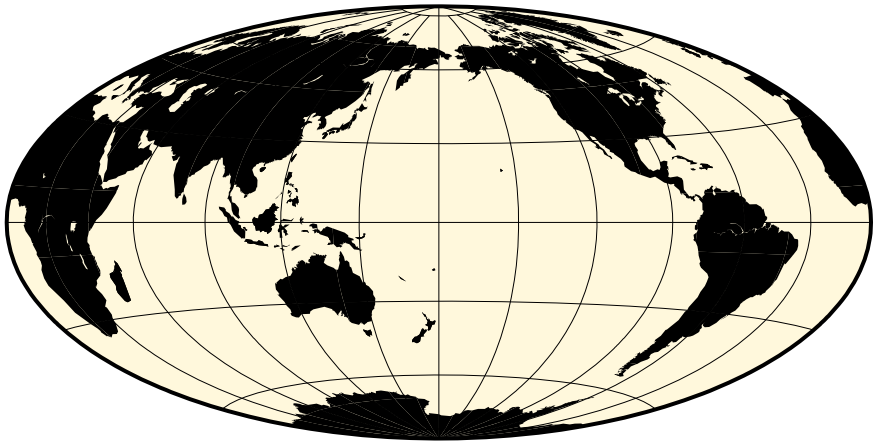


图 7.19: 使用 Hammer 投影绘制全球地图

7.12 -Ji：正弦曲线投影

维基链接：https://en.wikipedia.org/wiki/Sinusoidal_projection

正弦曲线投影是等面积投影，是已知的最古老的投影之一，也被称为等面积 Mercator 投影。其中心经线是直线，其余经线是正弦曲线，纬线是等间距的直线。在所有纬线和中心经线处比例尺是真实的。

该投影的参数为:

```
-JI[<lon>/]<width>        -Ji[<lon>/]<scale>
```

<lon> 是中心经线，默认值为地图区域的中心。

示例:

```
gmt pscoast -Rd -JI4.5i -Bxg30 -Byg15 -Dc -A10000 -Ggray -P > GMT_sinusoidal.ps
```

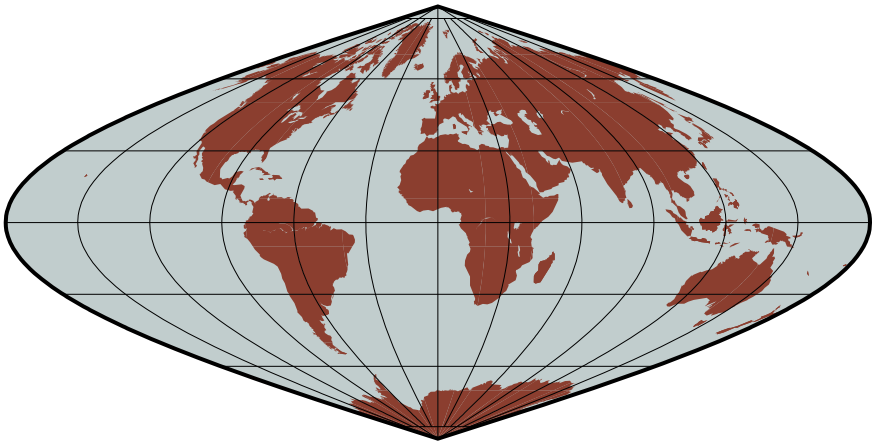


图 7.20: 使用正弦曲线投影绘制世界地图

为了减少形状的畸变，1927 年引入了间断正弦曲线投影，即用三个对称的段来覆盖全球。传统上，间断出现在 160°W、20°W 和 60°E 处。为了生成间断地图，必须调用 pscoast 命令三次以分别绘制每段地图并叠加起来。间断正弦曲线投影一般仅用于显示全球不连续数据分布。

为了生成一个宽度为 5.04 英寸的间断世界地图，需要设置比例尺为 5.04/360 = 0.014，并将每段图沿水平方向偏移其对应的宽度 (140·0.014 and 80·0.014):

```
gmt pscoast -R200/340/-90/90 -Ji0.014i -Bxg30 -Byg15 -A10000 -Dc -Gblack -K -P > GMT_
↪sinus_int.ps
gmt pscoast -R-20/60/-90/90 -Ji0.014i -Bxg30 -Byg15 -Dc -A10000 -Gblack -X1.96i -O -K␣
↪>> GMT_sinus_int.ps
gmt pscoast -R60/200/-90/90 -Ji0.014i -Bxg30 -Byg15 -Dc -A10000 -Gblack -X1.12i -O >>␣
↪GMT_sinus_int.ps
```

7.13 -Jj: Miller 圆柱投影

维基链接：https://en.wikipedia.org/wiki/Miller_cylindrical_projection

此投影由 Osborn Maitland Miller 于 1942 年提出，该投影既不是保角也不是等面积。所有的经线和纬线都是直线。该投影是 Mercator 与其他圆柱投影之间的折衷。在此投影中，纬线之间的

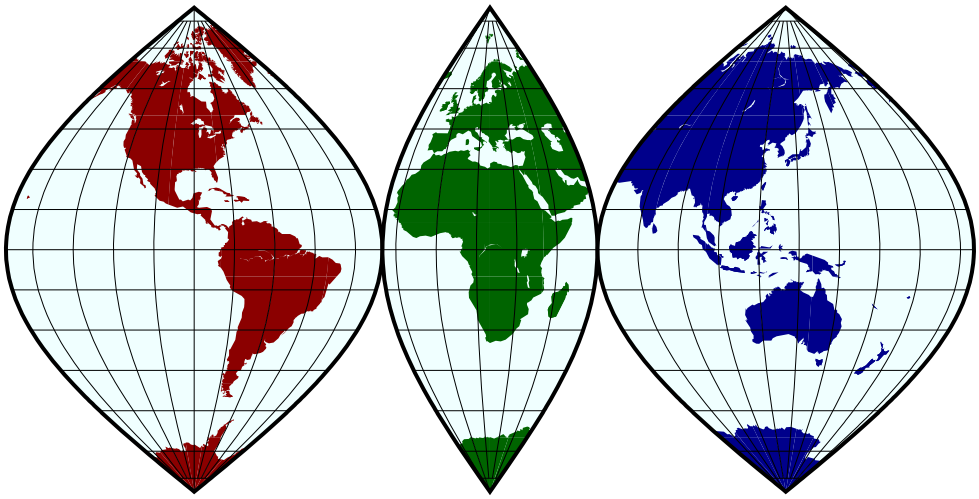


图 7.21: 使用间断正弦曲线投影绘制世界地图

间距使用了 Mercator 公式并乘以 0.8 倍的真实纬度，因而避免了极点的奇点，然后再将结果除以 0.8。

该投影的参数为:

```
-JJ<lon>/<width>        -Jj<lon>/<scale>
```

<lon> 为中心经度，默认为地图区域的中心。

示例:

```
gmt pscoast -R-90/270/-80/90 -Jj1:400000000 -Bx45g45 -By30g30 -Dc -A10000 \
            -Gkhaki -Wthinnest -P -Sazure > GMT_miller.ps
```

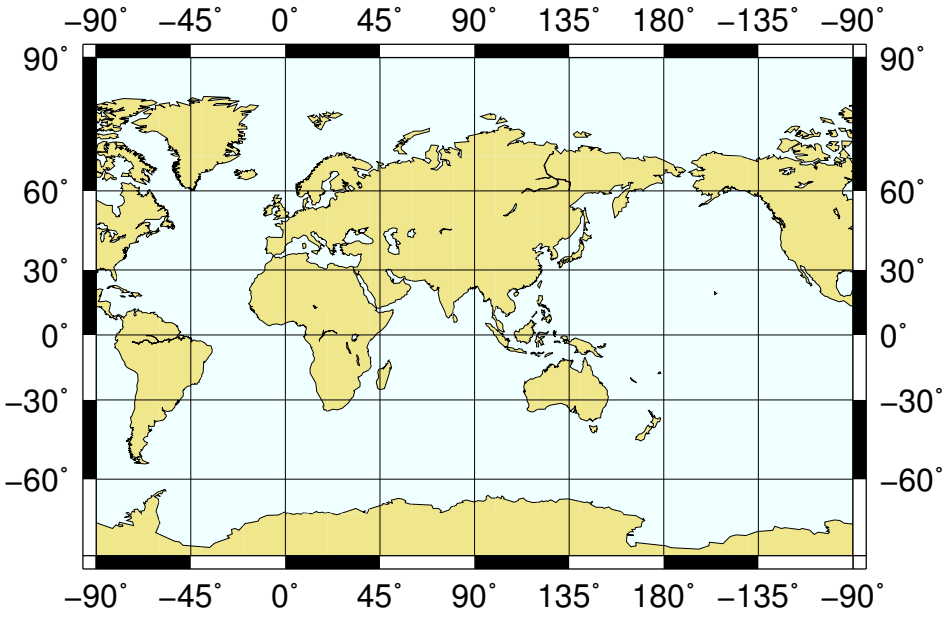


图 7.22: 使用 Miller 圆柱投影绘制世界地图

7.14 -Jk：Eckert 投影

维基链接：

- https://en.wikipedia.org/wiki/Eckert_IV_projection
- https://en.wikipedia.org/wiki/Eckert_VI_projection

Eckert IV 和 VI 投影由 Max Eckert-Greiffendorff 于 1906 年提出，是伪圆柱等面积投影。中心经线以及所有的纬线都是执行，其余经线是等间隔分布的椭圆弧（IV）或正弦曲线（VI）。比例尺在纬线 40°30’（IV）和 49°16’（VI）是真实的。`-JKf`（f 代表 four）表示使用 Eckert IV 投影，`-JKs`（s 代表 six）表示使用 Eckert VI 投影。若不指定 f 或 s，则默认使用 Eckert VI 投影。

该选项的参数为:

```
-JK[f|s][<lon>/]<width>       -Jk[f|s][<lon>/]<scale>
```

`<lon>` 为中心经线，默认值为地图区域的中心。

Eckert IV 示例:

```
gmt pscoast -Rg -JKf4.5i -Bg -Dc -A10000 -Wthinnest -Givory -Sbisque3 -P > GMT_
↪eckert4.ps
```

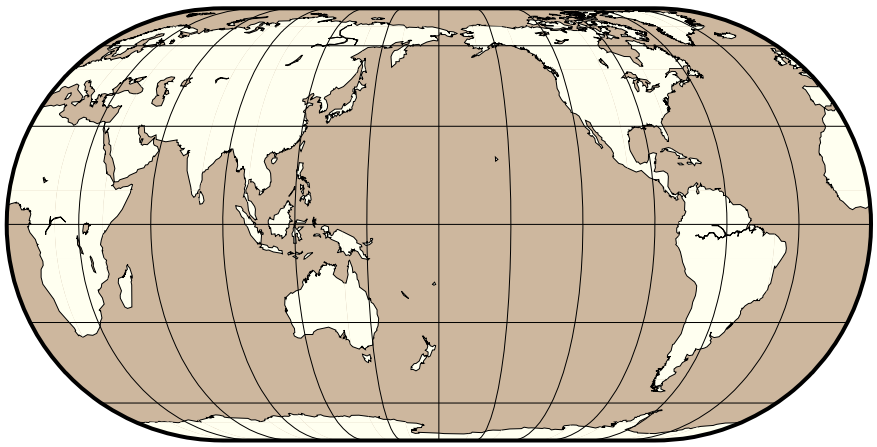


图 7.23: Eckert IV 投影绘制全球图

Eckert VI 示例:

```
gmt pscoast -Rg -JKs4.5i -Bg -Dc -A10000 -Wthinnest -Givory -Sbisque3 -P > GMT_
↪eckert4.ps
```

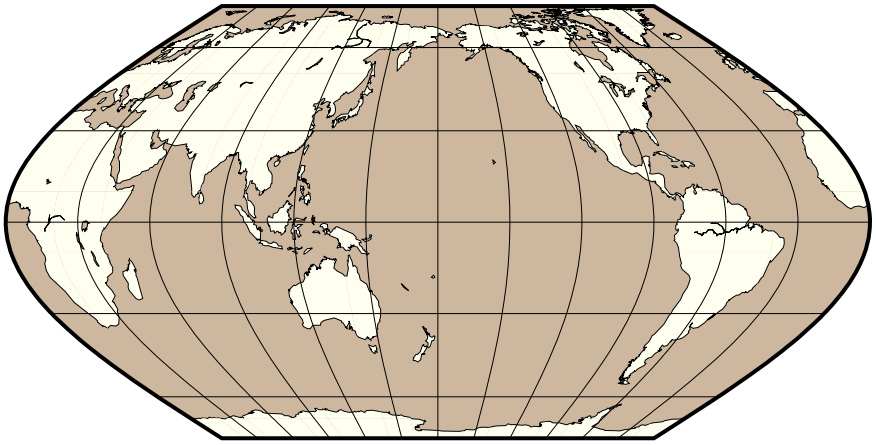


图 7.24: Eckert VI 投影绘制全球图

7.15 -JI：Lambert 圆锥保角投影

维基链接：https://en.wikipedia.org/wiki/Lambert_conformal_conic_projection

此投影由 Heinrich Lambert 于 1772 年提出，主要用于绘制东西方向范围很大的地图。与 Albers 投影不同的是，Lambert 投影不是等面积的。纬线是共圆心的圆弧，经线是这些圆的等间隔分布的半径。与 Albers 投影类似，只有两条标准纬线是无畸变的。

该投影的参数为:

```
-JB<lon>/<lat>/<lat1>/<lat2>/<width>
-Jb<lon>/<lat>/<lat1>/<lat2>/<scale>
```

- `<lon>` 和 `<lat>` 是投影中心的位置
- `<lat1>` 和 `<lat2>` 是两条标准纬线

Lambert 保角投影场用于绘制美国地图，两个固定的标准纬线是 33ºN 和 45ºN。

示例:

```
gmt set MAP_FRAME_TYPE FANCY FORMAT_GEO_MAP ddd:mm:ssF MAP_GRID_CROSS_SIZE_PRIMARY 0.
↪05i
gmt pscoast -R-130/-70/24/52 -Jl-100/35/33/45/1:50000000 -Bag -Dl -N1/thick,red \
            -N2/thinner -A500 -Gtan -Wthinnest,white -Sblue -P > GMT_lambert_conic.ps
```

投影中心的选取并不影响投影，但其指定了哪一条经线垂直于地图。

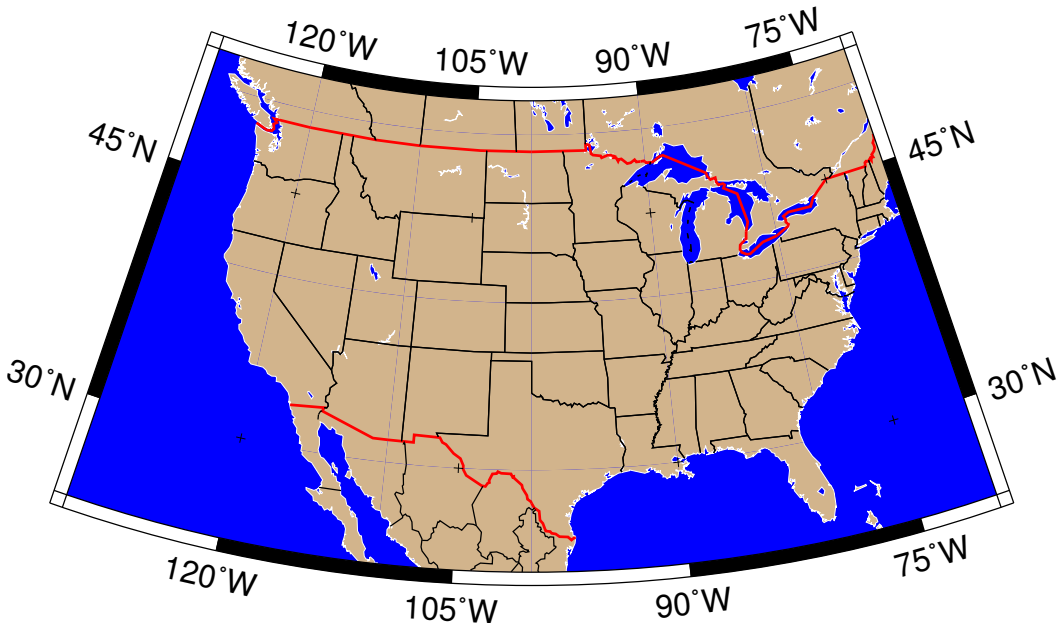


图 7.25: Lambert 保角圆锥投影

7.16 -Jm：Mercator 投影

维基链接：https://en.wikipedia.org/wiki/Mercator_projection

此投影是圆柱保角投影，沿着赤道无畸变，但两极畸变严重。此投影的主要特点是等方位角的线是一条直线，这样一条线称为 rhumb 线或 loxodrome。

在常规 Mercator 投影中，圆柱与赤道相切。若圆柱沿着其他方向与地球相切，则称为横向 Mercator 投影或倾斜 Mercator 投影。

常规的 Mercator 投影需要的参数如下:

```
-JM[<lon>[/<lat>]/]<width>
-Jm[<lon>[/<lat>]/]<scale>
```

- `<lon>` 中心经线，默认为地图区域的中心
- `<lat>` 标准纬线，默认值为赤道。若要指定标准纬线，则必须同时指定中心经线

示例:

```
gmt set MAP_FRAME_TYPE fancy
gmt pscoast -R0/360/-70/70 -Jm1.2e-2i -Bxa60f15 -Bya30f15 -Dc -A5000 -Gred \
            -P > GMT_mercator.ps
```

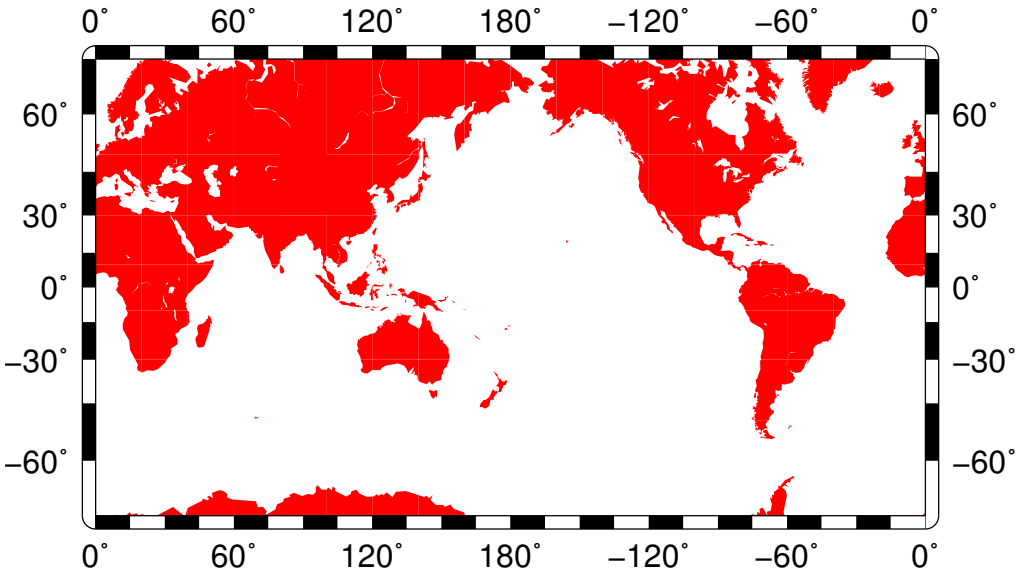


图 7.26: Mercator 投影

7.17 -Jn：Robinson 投影

维基链接：https://en.wikipedia.org/wiki/Robinson_projection

此投影 Arthur H. Robinson 于 1963 年提出，是一个修改后的圆柱投影，既不是保角也不是等面积。中心经线以及所有纬线都是执行，其余经线都是曲线。其使用查找表的方式而不是解析表达式来使得全球看上去比较正常。比例尺在经线 38 度是真实的。

该投影的参数为:

```
-JN[<lon>/]<width>       -Jn[<lon>/]<scale>
```

<lon> 是中心经线，默认值为地图区域的中心。

示例:

```
gmt pscoast -Rd -JN4.5i -Bg -Dc -A10000 -Ggoldenrod -Ssnow2 -P > GMT_robinson.ps
```

7.18 -Jo：倾斜 Mercator 投影

维基链接：https://en.wikipedia.org/wiki/Space-oblique_Mercator_projection

倾斜 Mercator 投影常用于绘制沿着倾斜方向横向范围较大的地图，其经线和纬线都是复杂曲线。

其有多种定义方式:

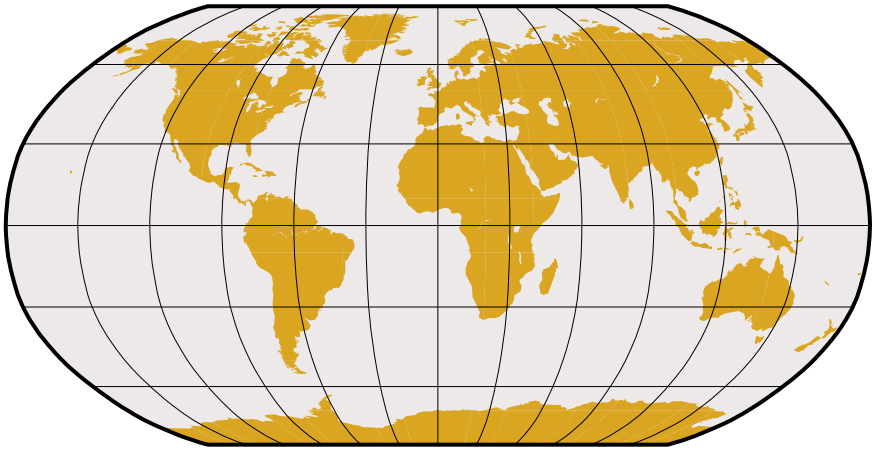


图 7.27: 使用 Robinson 投影绘制全球地图

```
-JO[a|A]<lon>/<lat>/<azi>/<width>
-Jo[a|A]<lon>/<lat>/<azi>/<scale>

-JO[b|B]<lon>/<lat>/<lon2>/<lat2>/<width>
-Jo[b|B]<lon>/<lat>/<lon2>/<lat2>/<scale>

-JO[c|C]<lon>/<lat>/<lonp>/<latp>/<width>
-Jo[c|C]<lon>/<lat>/<lonp>/<latp>/<scale>
```

- `<lon>/<lat>` 投影中心的经纬度
- `<azi>` 倾斜赤道的方位角
- `<lon2>/<lat2>` 倾斜赤道另一个点的经纬度
- `<lonp>/<latp>` 投影极点的经纬度

在三种定义中，大写的 `A|B|C` 表示允许投影极点位于南半球。

示例:

```
gmt pscoast -R270/20/305/25r -JOc280/25.5/22/69/4.8i -Bag -Di -A250 -Gburlywood \
            -Wthinnest -P -TdjTR+w0.4i+f2+l+o0.15i -Sazure --FONT_TITLE=8p \
            --MAP_TITLE_OFFSET=0.05i > GMT_obl_merc.ps
```

在使用倾斜头时，直接指定整个区域相对地图中心的相对投影坐标更为方便，下面的示例中使用了 `-Rk-1000/1000/-500/500` 来指定相对投影坐标。

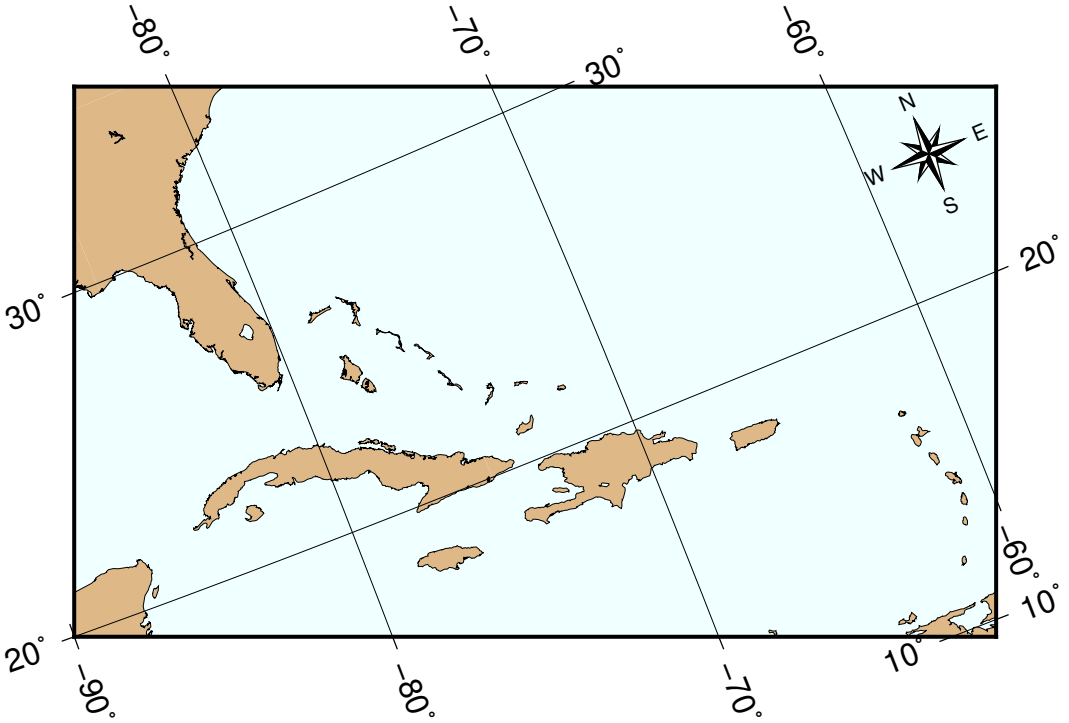


图 7.28: 使用 `-Joc` 倾斜 Mercator 投影

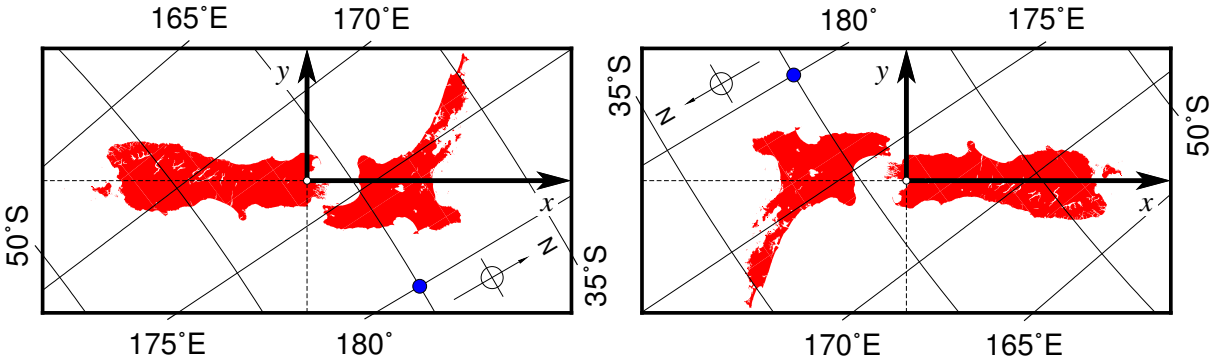


图 7.29: 使用 `-JOa` 倾斜 Mercator 投影

(左) `-JOa173:17:02E/41:16:15S/35/3i` (右) `-JOA173:17:02E/41:16:15S/215/3i`

7.19 -Jpoly：多圆锥投影

维基链接：https://en.wikipedia.org/wiki/Polyconic_projection

此投影既不是等面积也不是保角投影，沿着中心经线处畸变为 0。所有纬线的比例尺都是真实的，但其余经线则存在畸变。

示例:

```
gmt pscoast -R-180/-20/0/90 -JPoly/4i -Bx30g10 -By10g10 -Dc -A1000 -Glightgray \
            -Wthinnest -P > GMT_polyconic.ps
```

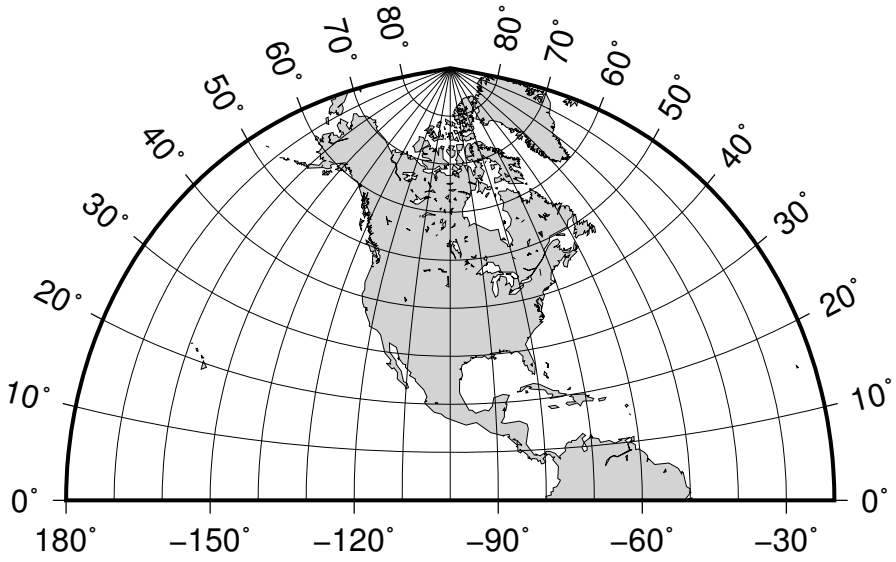


图 7.30: 多圆锥投影

7.20 -Jq：圆柱等距投影

维基链接：https://en.wikipedia.org/wiki/Equirectangular_projection

这个简单的圆柱投影是一个经度和纬度的线性缩放。最常用的形式是 Plate Carrée 投影，其中对经线和纬线的缩放比例是相同的。所有的经纬线都是直线。

该投影的参数为:

```
-JQ[<lon>/[<lat>]/]<width>
-Jq[<lon>/[<lat>]/]<scale>
```

- `<lon>` 是中心经线，默认为地图区域的中心
- `<lat>` 是标准纬线，默认为赤道，若指定了标准纬线，则必须指定中心经线

示例:

```
gmt pscoast -Rg -JQ4.5i -B60f30g30 -Dc -A5000 -Gtan4 -Slightcyan -P > GMT_equi_cyl.ps
```

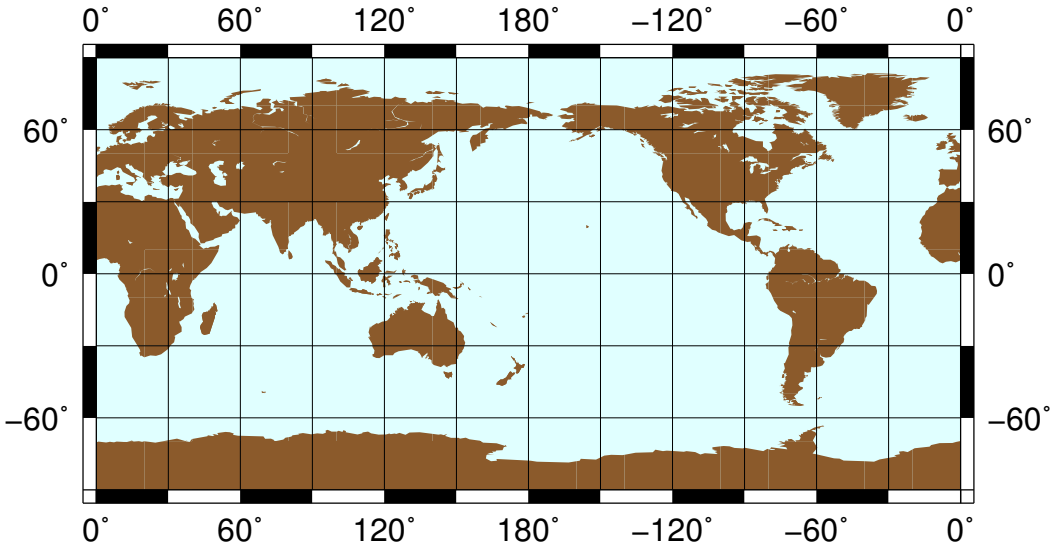


图 7.31: 使用 Plate Carrée 投影绘制全球地图

选择不同的标准纬线，则可以获取经度和纬度的不同缩放比例。流行的几个标准纬线如下:

Grafarend and Niermann, minimum linear distortion	61.7º
Ronald Miller Equirectangular	50.5º
Ronald Miller, minimum continental distortion	43.5º
Grafarend and Niermann	42º
Ronald Miller, minimum overall distortion	37.5º
Plate Carrée, Simple Cylindrical, Plain/Plane	0º

7.21 -Jr: Winkel Tripel 投影

维基链接: https://en.wikipedia.org/wiki/Winkel_tripel_projection

1921 年 Oswald Winkel 设计了该投影，以在三个元素（面积、角度、距离）之间折衷，是的在绘制全球地图时，这三个元素的畸变最小。此投影不是保角也不是等面积投影。中心经线和赤道是直线，其他经线和纬线是曲线。该投影取等距圆柱投影和 Aitoff 投影的坐标的平均值。极点处投影为 0.4 倍赤道长度的直线。

该投影的参数为:

```
-JR[<lon>/]<width>        -Jr[<lon>/]<scale>
```

<lon> 是中心经线，默认值为地图区域的中心。

示例:

```
gmt pscoast -Rd -JR4.5i -Bg -Dc -A10000 -Gburlywood4 -Swheat1 -P > GMT_winkel.ps
```

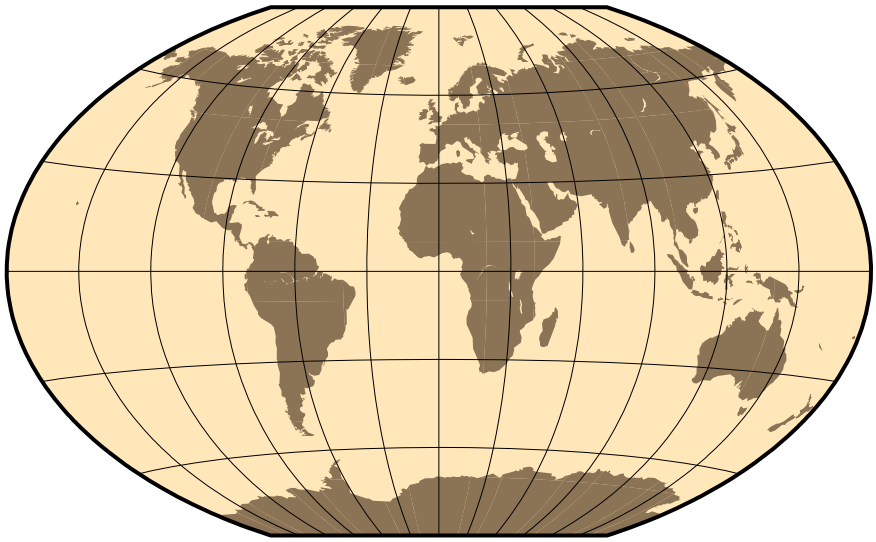


图 7.32: 使用 Winkel Tripel 投影绘制全球地图

7.22 -Js：立体等角投影

维基链接：https://en.wikipedia.org/wiki/Stereographic_projection

此投影是保角方位投影，主要用于绘制南北极区域。在两极，所有经线都是直线，纬线则是圆弧。

该投影的参数:

```
-JS<lon>/<lat>[/<distance>]/<width>
-Js<lon>/<lat>[/<distance>]/<scale>
```

- <lon>/<lat> 投影中心的经纬度
- <distance> 地图边界到投影中心的角度，默认值为 90 度
- <scale> 可以是 1:xxxx 也可以是 <radius>/<latitude> （<radius> 是投影中心到纬线 <latitude> 在图上的距离），还可以是 <slat>/1:xxxx （指定在标准纬线 <slat> 处的比例尺）

7.22.1 极区立体地图

下面的示例中，投影中心为北极，地图边界与经线和纬线完全重合:

```
gmt pscoast -R-30/30/60/72 -Js0/90/4.5i/60 -B10g -Dl -A250 -Groyalblue \
            -Sseashell -P > GMT_stereographic_polar.ps
```

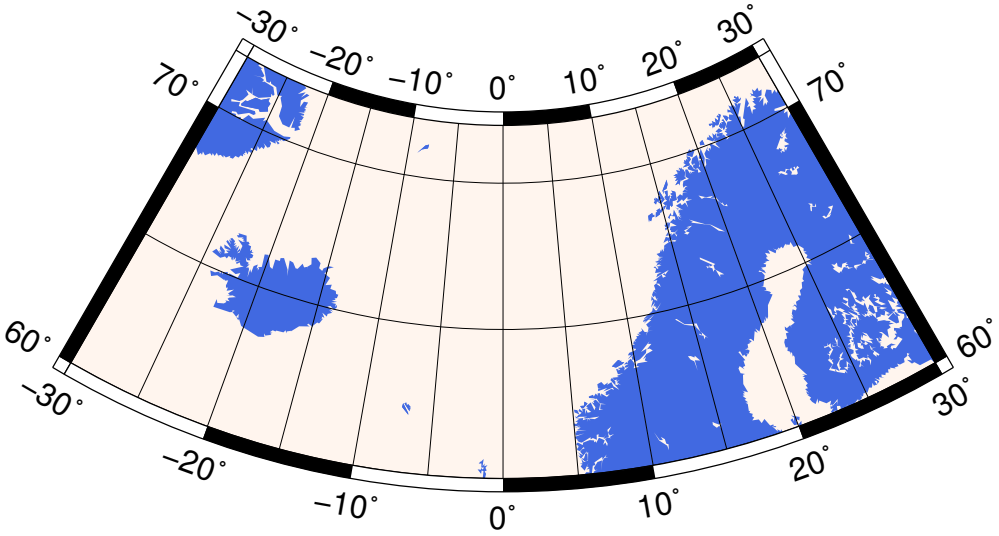


图 7.33: 极区立体保角投影

7.22.2 矩形立体地图

与 Lambert 方位等面积投影类似，也可以通过指定地图区域左下角和右上角的坐标来绘制一个矩形区域:

```
gmt set MAP_ANNOT_OBLIQUE 30
gmt pscoast -R-25/59/70/72r -JS10/90/11c -B20g -Dl -A250 -Gdarkbrown -Wthinnest \
            -Slightgray -P > GMT_stereographic_rect.ps
```

7.22.3 一般立体地图

示例:

```
gmt set MAP_ANNOT_OBLIQUE 0
gmt pscoast -R100/-42/160/-8r -JS130/-30/4i -Bag -Dl -A500 -Ggreen -Slightblue \
            -Wthinnest -P > GMT_stereographic_general.ps
```

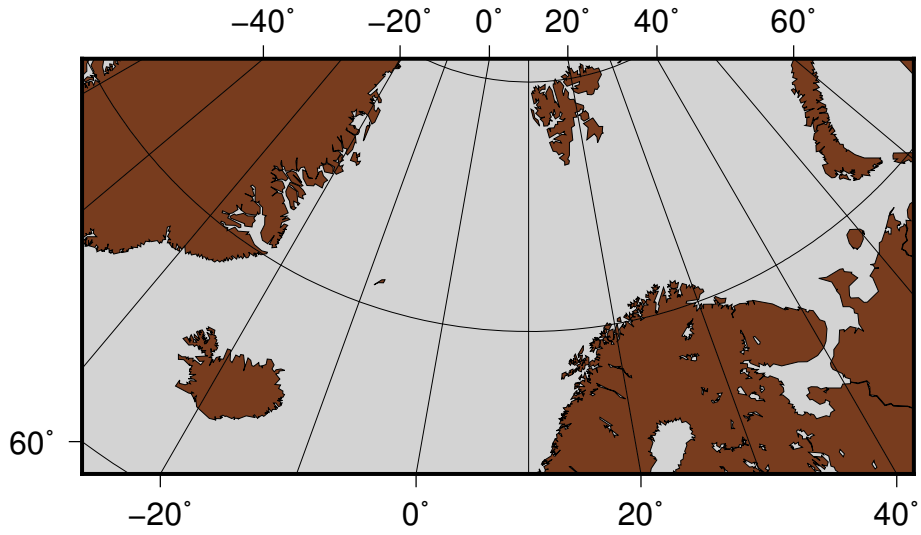


图 7.34: 矩形边界下的极区立体保角投影

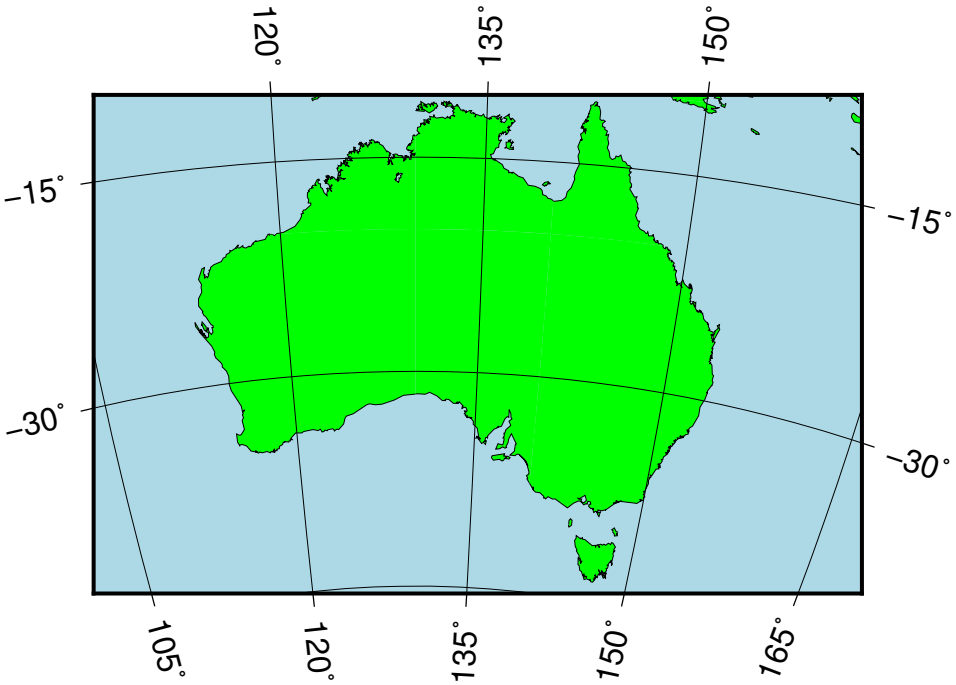


图 7.35: 一般立体投影

7.23 -Jt：横向 Mercator 投影

维基链接：https://en.wikipedia.org/wiki/Transverse_Mercator_projection

此投影由 Lambert 于 1772 年提出。该投影中，圆柱与某条经线相切。在该经线处无畸变。离中心经线越远畸变越大，距离中心经线 90 度处的经线畸变达到无穷。中心经线和赤道都是直线，其余经线和纬线则是复杂曲线。

该投影的参数:

```
-JT<lon>[/<lat>]/<width>
-Jt<lon>[/<lat>]/<scale>
```

<lon> 中心经线，<lat> 原点的纬度，默认值为赤道。

地图缩放因子默认值为 1，可以通过修改参数*PROJ_SCALE_FACTOR* 以实现自定义。

示例:

```
gmt pscoast -R20/30/50/45r -Jt35/0.18i -Bag -Dl -A250 -Glightbrown -Wthinnest \
            -P -Sseashell > GMT_transverse_merc.ps
```

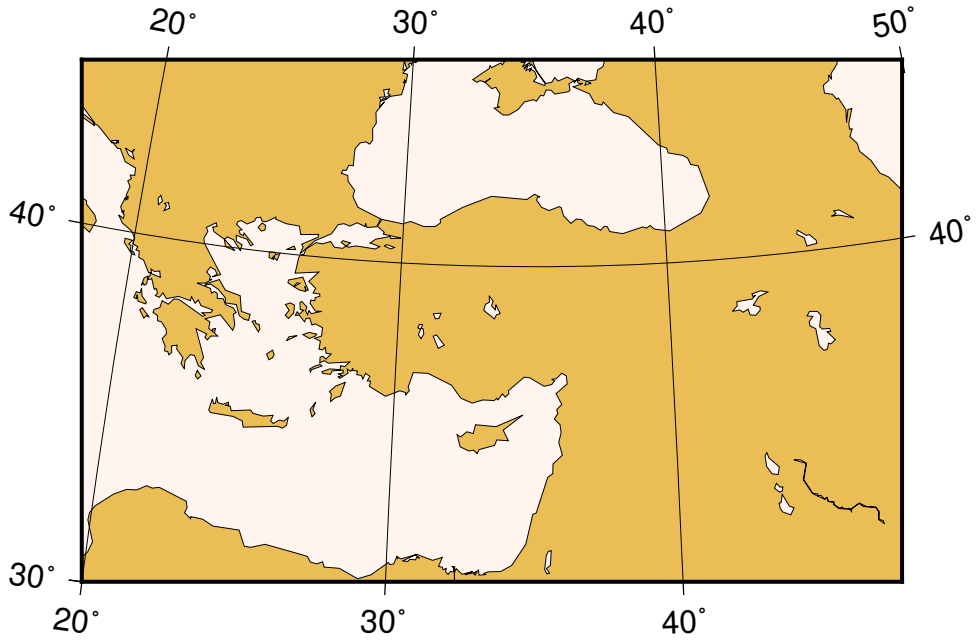


图 7.36: 矩形横向 Mercator 地图

示例:

```
gmt pscoast -R0/360/-80/80 -JT330/-45/3.5i -Ba30g -BWSne -Dc -A2000 \
            -Slightblue -G0 -P > GMT_TM.ps
```

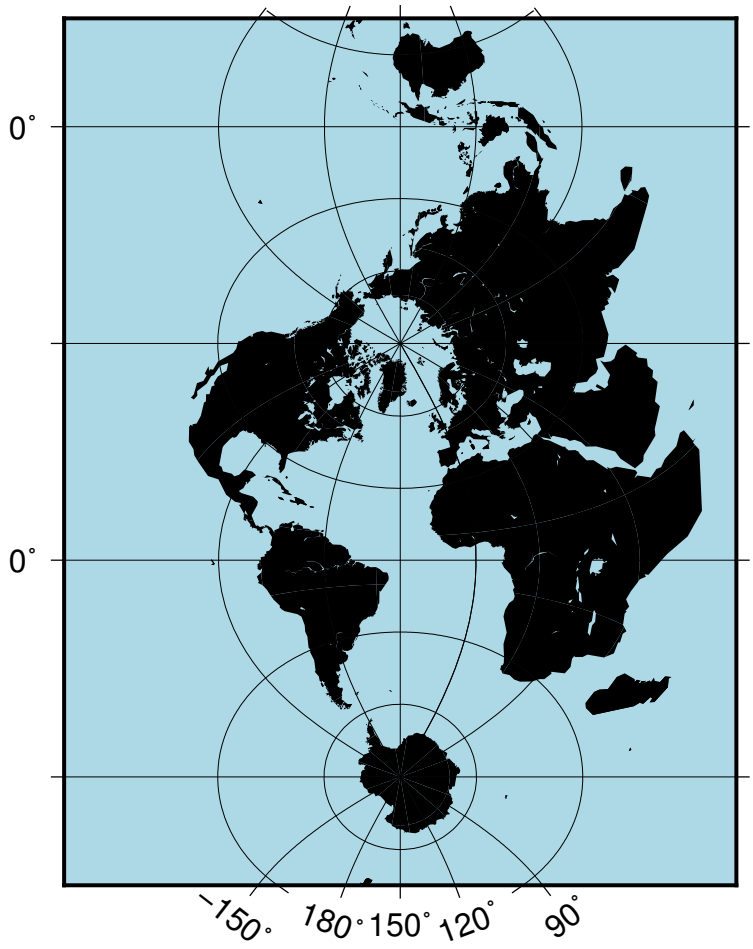


图 7.37: 全球横向 Mercator 地图

7.24 -Ju：通用横向 Mercator(UTM) 投影

维基链接：https://en.wikipedia.org/wiki/Universal_Transverse_Mercator_coordinate_system

通用横向 Mercator(UTM) 投影是横向 Mercator 投影的一个特殊子集。此处，全球在南北纬 84 度之间被划分为 60 个区域，大多数区域的宽度都是 6 度。每一个区域都有各自位移的中心经线。进一步，每个区域都被划分为纬度带。

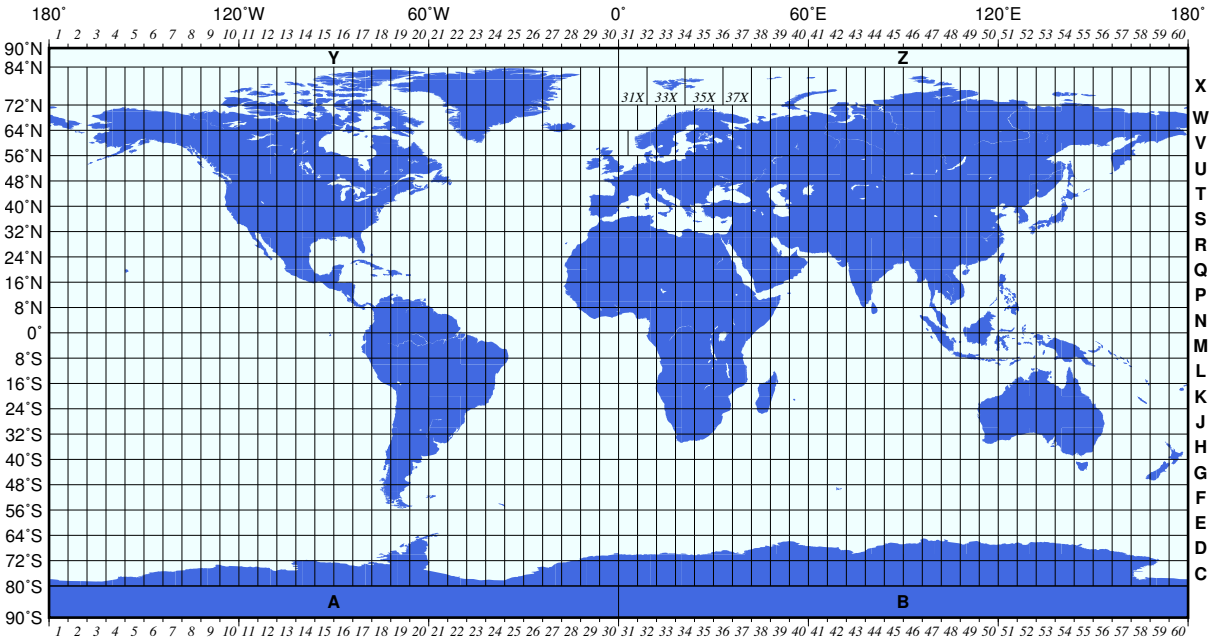


图 7.38: 通用横向 Mercator 区域布局

该投影的参数为:

```
-JU<zone>/<width>        -Ju<zone>/<scale>
```

其中 `<zone>` 可以取 1–60、A、B、Y、Z，负值表示南半球的区域，也可以加上 C–H 以及 J–N 来指定纬度带。

为了让任意指定区域的畸变最小化，公式中乘以了比例因子 0.9996，这个值可以通过修改*PROJ_SCALE_FACTOR* 以自定义。这是的 UTM 投影是割线投影而不是切线投影，在赤道处比例尺的畸变只有千分之一。在中心经线附近 10 度范围内的椭球投影表达式都是精确的。对于更大的区域，则在一般球状公式中使用保角纬度作为代替。

7.25 -Jv：Van der Grinten 投影

维基链接：https://en.wikipedia.org/wiki/Van_der_Grinten_projection

此投影由 Alphons J. van der Grinten 于 1904 年提出，其既不等面积也不保角。中心经线和赤道都是直线，其余经线则是圆弧，仅在赤道处比例尺是真实的，主要用于在一个圆内展示整个世界地图。

该投影的参数为:

```
-JV<lon>/<width>       -Jv<lon>/<scale>
```

<lon> 是投影中心经线，默认值为地图区域的中心。

示例:

```
gmt pscoast -Rg -JV4i -Bxg30 -Byg15 -Dc -Glightgray -A10000 -Wthinnest -P > GMT_
↪grinten.ps
```

图 7.39: 使用 Van der Grinten 投影绘制全球图

7.26 -Jw：Mollweide 投影

维基链接：https://en.wikipedia.org/wiki/Mollweide_projection

此投影是伪圆柱等面积投影，由 Karl Brandan Mollweide 于 1805 年提出。纬线是不等间隔分布的直线，经线是等间隔分布的椭圆弧。比例尺仅在南北纬 40 度 44 分纬线上才是真实的。此投影主要用于绘制全球的数据分布图。

该投影的参数为:

```
-JW[<lon>/]<width>      -Jw[<lon>/]<scale>
```

<lon> 为中心经线，默认值为地图区域的中心。

示例:

```
gmt pscoast -Rd -JW4.5i -Bg -Dc -A10000 -Gtomato1 -Sskyblue -P > GMT_mollweide.ps
```

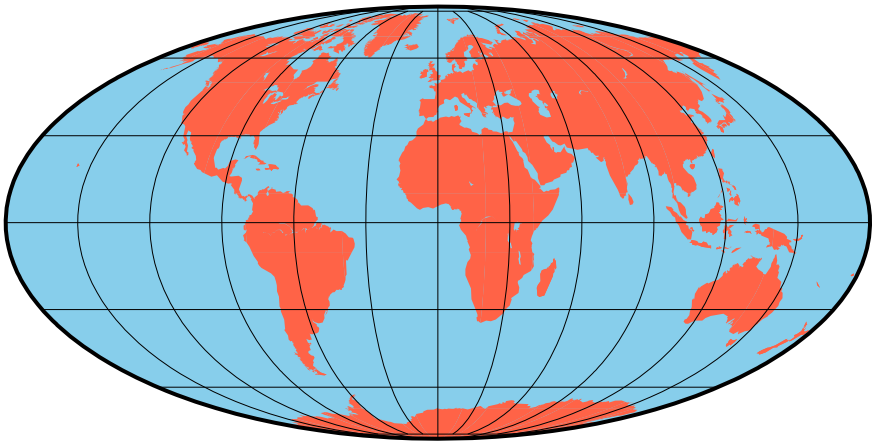


图 7.40: 使用 Mollweide 投影绘制全球地图

7.27 -Jy：圆柱等面积投影

维基链接：https://en.wikipedia.org/wiki/Cylindrical_equal-area_projection

选择不同的标准纬线，则对应不同的圆柱投影。所有的这些圆柱投影都是等面积且不保角的。所有经线和纬线都是直线。在高纬度处畸变很大。

该投影的参数为:

```
-JY<lon>/<lat>/<width>
-Jy<lon>/<lat>/<scale>
```

<lon> 是中心经线，<lat> 是标准纬线。

标准纬线可以取任意值，下面列出了一些比较流行的标准纬线的选择:

Balthasart	50º
Gall-Peters	45º
Hobo-Dyer	37º30’ (= 37.5º)
Trystan Edwards	37º24’ (= 37.4º)
Caster	37º04’ (= 37.0666º)
Behrman	30º
Lambert	0º

示例:

```
gmt pscoast -R-145/215/-90/90 -JY35/30/4.5i -B45g45 -Dc -A10000 -Sdodgerblue \
            -Wthinnest -P > GMT_general_cyl.ps
```

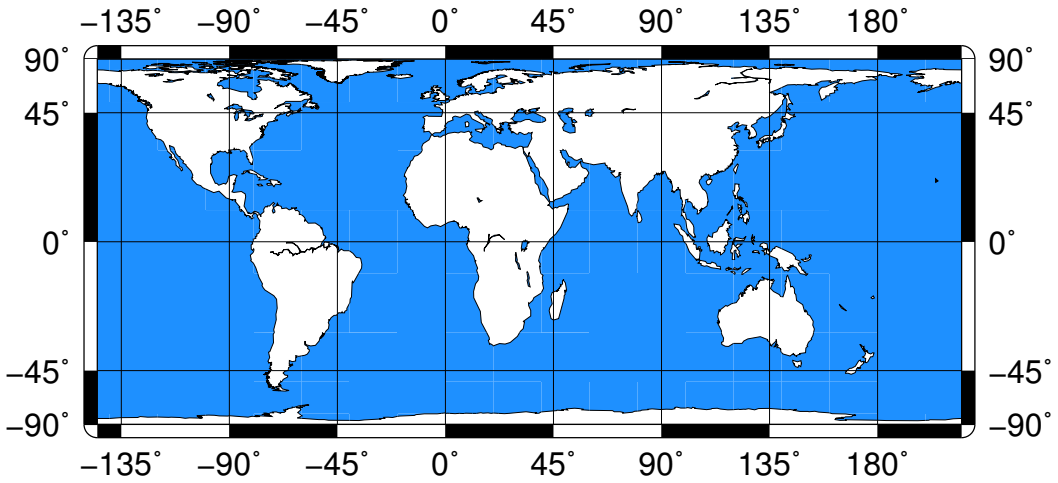


图 7.41: 使用 Behrman 圆柱等面积投影绘制地图

第八章　附录

8.1 三种距离计算方式

在计算地球任意两点间的距离时，GMT 提供了三种不同的计算方式。这三种方式在精度和效率上各有权衡，用户可以根据自己的需求选择适合的距离计算方式。

8.1.1 Flat Earth 距离

地球上任意两点 A 和 B 的 Flat Earth 距离计算公式：

$$d_f = R\sqrt{(\theta_A - \theta_B)^2 + \cos\left[\frac{\theta_A + \theta_B}{2}\right]\Delta\lambda^2}$$

其中 R 是地球平均半径（由参数*PROJ_MEAN_RADIUS* 控制），θ 是纬度，$\Delta\lambda = \lambda_A - \lambda_B$ 是经度差。式中地理坐标的单位均是弧度。

该方法的特点计算速度快，但精度不高，适用于纬度相差不大且对计算效率要求比较高的情况。

可以通过在距离量前加上前缀 - 指定使用该方法计算两点间的距离。某些情况下，只需要指定距离的单位而不需要指定具体的距离值，则可以在距离单位前加上前缀 - 表示该距离用 Flat Earth 方法计算。

比如，`-S-50M` 表示设定搜索半径为 50 海里，其中距离用 Flat Earth 方法计算。

8.1.2 大圆弧距离

地球上任意两点 A 和 B 的大圆弧距离可以用 Haversine 公式计算：

$$d_g = 2R\sin^{-1}\sqrt{\sin^2\frac{\theta_A - \theta_B}{2} + \cos\theta_A\cos\theta_B\sin^2\frac{\lambda_A - \lambda_B}{2}},$$

该方法是 GMT 默认使用的计算距离方法，适用于大多数情况。该方法将地球近似为一个半径为 R 的球，

有两个 GMT 参数可以控制大圆弧距离的计算：*PROJ_MEAN_RADIUS* 和*PROJ_AUX_LATITUDE*，参数的具体含义见相关页面中的说明。

8.1.3 大地测量距离

地球上两点间的距离用 Vincenty(1975) 的完全椭球公式计算。该方法计算得到的距离精度最高，精确到精确到 0.5 毫米，同时也是计算速度的最慢的方式。

可以通过在距离或距离单位前加上前缀 `+` 来指定用该方法计算距离。比如，`-S+20k` 表示用该方法计算的 20 千米的距离。

除了 Vincenty 完全椭球公式外，还可以将参数*PROJ_GEODESIC* 设置成 `Rudoe` （GMT4 所使用的计算公式）或 `Andoyer` （近似公式，精确到 10 米量级）以使用不同的计算公式。

8.1.4 总结

GMT 在计算距离时有三种算法：FLAT Earth 距离、大圆弧距离和大地测量距离。三种方法的计算精度由低到高，计算速度由高到低。

对于一个距离量，比如 `40k` ，默认使用大圆弧距离，可以使用 `-40k` 表示 FLAT Earth 距离，`+40k` 表示大地测量距离。

8.2 `-J` 和 `-R` 的重要性

`-R` 选项用于指定数据的范围，`-J` 选项用于指定数据的投影方式。这两个选项是 GMT 中最基础、最常用也是最重要的选项。这两个选项的用法很简单，因而经常被用户忽视。但理解了 `-R` 选项和 `-J` 选项背后的机制，对于以后绘制稍复杂的图大有帮助。

先用最简单的笛卡尔投影 `-JX` 来解释。假设使用 `-JX10c/5c -R0/10/0/20` ，即笛卡尔线性投影。X 方向的数据范围是 0 到 10，底图宽度为 10 cm，即 X 方向比例尺为 1 单位每厘米。Y 方向的数据范围是 0 到 20，高度为 5 cm，即 Y 方向的比例尺为 4 单位每厘米。

假定坐标原点位于纸张的左下角（实际绘图的时候坐标原点离左下角是有一定距离的，这里为了简单起见，将坐标原点定于左下角），即数据的 `(0,0)` 位于纸张 `(0c,0c)` 处，数据 `(10,0)` 位于纸张 `(10c,0c)` 处，数据 `(0,20)` 位于纸张 `(0c,5c)` ，数据 `(10,20)` 位于纸张 `(10c,5c)` 处，这样就在纸上确定了矩形的四个顶点。对于任意一点数据，都可以通过简单的计算，得到该数据点在纸张上的位置，比如数据点 `(5,5)` ，简单计算可知该点位于纸张 `(5c,1.25c)` 处。

对于非笛卡尔线性投影，比如各种复杂的地理投影，本质上也就是定义了一个将 `(经度，纬度)` 投影到平面上 `(x,y)` 的函数而已。当然，这个函数比较复杂，所以就不像笛卡尔线性投影那样举例了，具体函数的实现以及数据的投影，GMT 都已经做好了。用户需要做的只是理解。

所以，`-J` 选项后面接的投影方式，定义了经纬度到 XY 的函数。投影方式后面如果接的是宽度，则根据 `-R` 指定的数据范围即可计算投影的比例尺；如果投影方式后接的就是比例尺，那就直接用这个比例尺啦。

8.3 环境变量

8.3.1 $GMT_SHAREDIR

指定 GMT 的 share 目录的位置。若该变量为空，则 GMT 会使用默认值 `${GMTHOME}/share` 。

8.3.2 $GMT_DATADIR

用于指定用户自己的数据文件的放置目录。当在命令中给定文件名时，GMT 首先会在当前目录寻找该文件。若找不到，则会到环境变量 `$GMT_DATADIR` 所指定的目录中寻找。

对于一些常用的数据文件，可以放在特定的目录中，并将任意一个环境变量指向该目录，则在命令中使用该文件时只需要指定文件名而不必给出完整路径。

Linux 下多个目录之间用冒号分隔；Windows 下用分号分隔。

Linux 下以 / 结尾的目录会被递归搜索。(此句存疑)。

8.3.3 $GMT_USERDIR

用于指定用户自定义的配置文件的放置目录。比如 `gmt.conf` 、自定义的符号、CPT、数学宏、网格文件自定义后缀 `gmt.io` 等。

若 `$GMT_USERDIR` 未定义，则使用默认值 `${HOME}/.gmt` 。

8.3.4 $GMT_TMPDIR

GMT 写临时文件 `gmt.history` 和 `gmt.conf` 的路径。若未指定，则默认写到当前目录。

8.4 命令行补全

GMT 为 bash 提供了基本的命令行补全功能。在终端敲 GMT 命令时，可以使用 `Tab` 键补全部分选项。不过补全功能很弱，不用也罢。

使用步骤如下：

1. 安装 bash-completion

   Ubuntu/Debian 下:

```
sudo apt-get install bash-completion
```

CentOS/Fedora 下:

```
sudo yum install bash-completion
```

2. 在 ~/.bashrc 中加入如下语句

```
# Use bash-completion, if available
if [ -r /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
fi

. ${GMTHOME}/share/tools/gmt_completion.bash
```

3. 执行 source ~/.bashrc 使修改生效

4. 在命令行键入即可看到补全效果:

```
$ gmt psxy -[Tab]
-^  -?  -A  -B  -C  -E  -g  -h  -I  -K  -N  -p  -R  -S  -T  -V  -X
-:  -a  -b  -c  -D  -f  -G  -i  -J  -L  -O  -P  -s  -t  -U  -W  -Y
```

8.5 自定义字体

GMT 默认支持了 35 种字体，如果想要使用额外的字体，则需要自己配置。可以修改 $*GMTHOME*/share/postscriptlight/PSL_custom_fonts.txt 文件或在当前目录下新建配置文件 PSL_custom_fonts.txt 。

文件中每行包含三列数据，分别是：PS 字体名、字体的高度（单位为 p ）以及布尔值 0 或 1 告诉 GMT 是否要对文字重新编码。例如:

```
LinBiolinumO      0.700    0
LinLibertineOB    0.700    0
```

使用 gmt pstext -L 可查看所有字体对应的编号，在 GMT 中即可直接使用这些编号指定字体。

实际使用的时候有几个问题需要注意：

1. 第一列是 PostScript Name，与字体文件的文件名不同，但尚不清楚如何提取某个字体文件的 PS 文件名

2. 是否要对文字进行编码？取 0 还是 1，这个尚不确定

3. 调用该字体生成 PS 文件后，直接用 `gs` 打开会看不到效果，需要给 `gs` 加上 `-sFONTPATH=dirname` 选项指定字体文件名，或指定系统环境变量 `$GS_FONTPATH`

4. 在用 `psconvert` 将 PS 文件转换成其他格式时，需要使用 `-C-sFONTPATH=dirname` 将自定义字体的路径传递给 gs，使得字体可以被嵌入到新格式中

8.6 GMT 风格指南

在用 GMT 绘图的时候，遵循一定的风格指南，可以让画图的过程更轻松，让脚本更易读、易改、更健壮、可移植性更高。

8.6.1 使用脚本来执行 GMT 命令

GMT 遵循了 UNIX 的设计思想，将不同的功能分别放在不同的命令中，因而在绘图时需要执行一系列命令。

若使用命令行来执行一系列命令，很容易弄混前一个命令是什么。将所有的绘图命令放在脚本中可以很方便地重复执行一系列命令，以对绘图的细节进行微调。

除非是一两个命令就可以解决的图，否则应一律使用脚本而非命令行。

Windows 下常用的脚本是 bat，Linux 常用的是 Bash、Perl 和 Python。使用什么脚本语言完全依赖于用户个人的需求与喜好，这里以 Bash 脚本为例，其他脚本同理。

```
#!/bin/bash

gmt psxy ...
gmt pscoast ...
gmt grdraster ...
gmt grdimage ...
gmt psxy ...
```

8.6.2 不要跨平台写脚本

不要在 Windows 下写 Bash 脚本然后复制到 Linux 下运行；也不要在 Linux 下写 Bat 脚本放在 Windows 下运行。这其中会遇到很多坑，包括但不限于：

- 默认编码不同，Windows 用 GBK，Linux 用 UTF8；
- 换行符不同，Windows 用 `\r\n` ，Linux 用 `\n` ；

如果你真的跨平台写了脚本并遇到各种奇怪的问题时，尝试着新建一个文件，然后把脚本重新手敲一遍。

8.6.3 使用变量

脚本不仅仅只是将一系列命令放在一个文件而已。绘图时有很多需要在多个命令中重复使用的东西，比如设置投影方式的 `-J` 、设置绘图范围的 `-R` 、文件名 `xxx.ps` 。

关于如何使用变量，一般有两种定义方式，这两种方法各有利弊，尚待权衡：

1. 将参数作为变量的值

```
#!/bin/bash
J=M6i
R=0/360/-60/60
Bx=60
By=30
PS=map.ps

gmt pscoast -J$J -R$R -B$Bx -B$By -W1p -A1000 -K > $PS
gmt psxy -J -R -Sa0.5c -Gred -O >> $PS << EOF
160 20
150 30
EOF
```

2. 将选项和参数作为变量的值

```
#!/bin/bash
J=-JM6i
R=-R0/360/-60/60
B=-Bx60 -By30
PS=map.ps

gmt pscoast $J $R $B -W1p -A1000 -K > $PS
gmt psxy -J -R -Sa0.5c -Gred -O >> $PS << EOF
160 20
150 30
EOF
```

8.6.4 不要省略参数

GMT 的一个特性是后面的命令可以继承前面命令的一些参数，比如前面的命令中指定了 `-JM10c -R0/360/-60/60` ，后面的命令可以直接使用 `-J -R` 而不用重复给出更多的参数。这样

的设计减少了用户的键入。

省略参数虽然带来了一点点方便，但也可能会造成一些麻烦：

1. 写 GMT 脚本时由于需要经常修改、增添命令或调整各个命令之间的顺序。在省略了部分参数的情况下，调整各个命令之间的顺序就可能导致 `-J -R` 出现在第一个，有时会造成意想不到的错误。

2. 参数可以省略本质上是因为之前的命令将标准选项的参数写到了 GMT 历史文件 `gmt.history` 中，因而当在同一个目录里同时运行两个相同或不同的脚本时，两个脚本就会读写同一个 `gmt.history` 文件，进而可能导致一个脚本读到的内容是另外一个脚本写的。

因而，尽量不要省略参数。相同的参数在多个命令里要写很多遍，这样很麻烦，但是因为前面已经把这些参数定义成变量了，所以只是多敲了几个字符而已，因此带来的好处可不少。

```
#!/bin/bash
J=M6i
R=0/360/-60/60
Bx=x60
By=y30
PS=map.ps

gmt pscoast -J$J -R$R -B$Bx -B$By -W1p -A1000 -K > $PS
gmt psxy -J$J -R$R -Sa0.5c -Gred -O >> $PS << EOF
160 20
150 30
EOF
```

8.6.5 开始与结束

多个绘图命令会将 PS 代码依次写入到一个 PS 文件中。绘图命令的顺序有时会影响到成图的效果，最常见的例子就是，如果先 `pscoast` 再 `grdimage` ，则 `grdimage` 的效果就会覆盖 `pscoast` 的效果。因而在绘制一张稍复杂的图时，经常需要在原有的代码中增添、删除或修改已有命令的顺序，这个时候尤其需要注意 `-K` 、`-O` 以及重定向符号的使用。

下面的代码解决了这个问题：

```
#!/bin/bash
J=M6i
R=0/360/-60/60
Bx=x60
By=y30
PS=map.ps
```

```
# 写入 PS 文件头
gmt psxy -J$J -R$R -T -K > $PS

# 一系列绘图命令
gmt pscoast -J$J -R$R -B$Bx -B$By -W1p -A1000 -K -O >> $PS

# 写入 PS 文件尾
gmt psxy -J$J -R$R -T -O >> $PS
```

此处使用了专门的两个命令用于开始和结束一个 PS 绘图。这样做的好处在于：中间的所有绘图命令都使用 `-K -O >>`，不必再考虑这个命令是第一个还是最后一个了，也可以随意删除或修改任何一个命令而不必担心造成其它效果。

因而，实际写绘图脚本时，先把开始和结束这两个命令写对，然后在两个命令的中间写入真正的绘图命令。每新增一个绘图命令，都可以执行一下脚本，以检查绘图效果，若效果正确，则继续添加下一个绘图命令。

8.6.6 使用 SI 单位制

GMT 支持 SI 单位制和 US 单位制，默认是 SI 单位制。由于 GMT 的开发者是美国人，官方的文档使用的是 US 单位制，因而国内的 GMT 用户在学习的过程中也就习惯性地使用了 US 单位制。

实际上，国内用户对于 US 单位制没有太多的概念，`-X1i` 远远没有 `-X2.5c` 直观。SI 单位制是国际标准单位，也是中国人熟悉的单位，使用 SI 单位制会使得微调更简单。

8.6.7 不要依赖于 GMT 的系统设置

你所写的每一个脚本，将来都可能传给后来人使用，可能在任一台机器上使用。要保证脚本每次运行的结果完全一致，并不是一个简单的事情。

8.6.7.1 不要修改 GMT 系统设置

有些人喜欢使用特定字体，或者喜欢使用特定尺寸的纸张，这可以通过修改 `$GMTHOME/share/conf` 下的一堆系统配置文件来实现。但是，不要这样做，这会导致脚本在别人的机器上跑出来完全不一样的结果。

8.6.7.2 不要省略单位

当使用 `-JM10` 时，GMT 会默认使用当前的系统默认单位（一般来说是 `c` ，也就是厘米），当脚本在另一台系统默认单位为 `i` 的机器上运行时，绘图的结果会完全不同。

8.6.7.3 conf 文件的使用

不要手动修改 `gmt.conf` 文件!

GMT 中提供了 `gmtset` 模块可以用于修改缺省参数，比如标题的字体、大小等等。该命令会在当前工作目录下生成一个 `gmt.conf` 文件，进而影响到接下来绘图命令的执行效果。

合理的使用方式如下：

```
#!/bin/bash

# 用 gmtset 修改默认参数
gmt gmtset MAP_FRAME_TYPE plain

# 绘图
gmt psxy ...
gmt pscoast ...
gmt psxy ...

# 删除参数文件
rm gmt.*
```

在脚本的最后 `rm gmt.*` 删除了两个临时文件，一个是 `gmt.history` ，其记录了标准选项的命历史，另一个是 `gmt.conf` ，记录了当前的参数。

删除这些文件的原因在于：

- 临时文件，应该删除
- 脚本已经执行完毕，不应该遗留下无用的文件
- 保留 `gmt.conf` 文件，可能会导致下次执行脚本时产生不同的效果

有这样一种可怕的情况：假如你在 `$HOME` 下执行了 `gmtset` 命令，然后画了一个简单的图，但是却忘记删除 `$HOME` 下生成的 `gmt.conf` 文件，这会影响到其它目录中几乎所有 GMT 脚本的执行效果，而且这个问题很难排查。要避免这种情况的发生需要遵循几个原则：

1. 尽量不要在 `$HOME` 下执行 GMT 命令（可能会产生临时文件，难以清理）
2. 尽量不要使用命令行执行 GMT 命令（因为你很可能会忘记你刚刚执行过哪些命令）

3. 使用 gmtset 的脚本，最后一定要记得删除 gmt.conf

8.6.8 -P 选项的使用

只有第一个绘图命令中的 -P 选项是起作用的，所以不需要在每个绘图命令里都使用 -P 选项，当然若是每个绘图命令都使用了 -P 选项也没有问题，只是不够简洁而已。

两种推荐的使用方式：

1. 在开始 PS 文件时使用该选项：

```
#!/bin/bash
J=M20c
R=0/360/-60/60
Bx=x60
By=y30
PS=map.ps

gmt psxy -J$J -R$R -T -K -P > $PS
gmt pscoast -J$J -R$R -B$Bx -B$By -W1p -A1000 -K -O >> $PS
gmt psxy -J$J -R$R -T -O >> $PS
rm gmt.*
```

2. 修改 PS_PAGE_ORIENTATION ，不使用 -P 选项

```
#!/bin/bash
J=M20c
R=0/360/-60/60
Bx=x60
By=y30
PS=map.ps

gmt set PS_PAGE_ORIENTATION portrait
gmt psxy -J$J -R$R -T -K > $PS
gmt pscoast -J$J -R$R -B$Bx -B$By -W1p -A1000 -K -O >> $PS
gmt psxy -J$J -R$R -T -O >> $PS
rm gmt.*
```

8.6.9 不要滥用-B 选项

-B 选项用于绘制边框并控制边框的绘制效果，即每个使用 -B 选项的命令都会绘制一次边框，在没有使用 -X 和 -Y 的情况下，多个命令重复使用 -B 选项会绘制多次边框，但由于边框是重合的，所以会看不出来区别。

对于 -B 选项，合理的用法是仅在第一个命令中使用。

8.6.10 verbose 模式

GMT 命令的输出信息常用于在写脚本时判断命令执行是否正确，而在真正执行时过多的输出信息反而会扰乱用户的屏幕输出。合理的使用 verbose 模式的方式有三种：

1. 写脚本时每个命令都加上 -V 选项，待确认脚本正确无误之后删除所有 -V
2. 定义 Verbose 变量

```
#!/bin/bash

J=M20c
R=0/360/-60/60
Bx=x60
By=y30
PS=map.ps
V=-V        # 调试时用这个
#V=         # 调试完成用这个

gmt psxy -J$J -R$R -T -K -P $V > $PS
gmt pscoast -J$J -R$R -B$Bx -B$By -W1p -A1000 -K -O $V >> $PS
gmt psxy -J$J -R$R -T -O $V >> $PS
rm gmt.*
```

3. 修改缺省参数

```
#!/bin/bash
J=M20c
R=0/360/-60/60
Bx=x60
By=y30
PS=map.ps

gmt gmtset GMT_VERBOSE TRUE
gmt psxy -J$J -R$R -T -K > $PS
gmt pscoast -J$J -R$R -B$Bx -B$By -W1p -A1000 -K -O >> $PS
gmt psxy -J$J -R$R -T -O >> $PS
rm gmt.*
```

从使用上的简洁来看，最简单的是第三种方法。

8.6.11 慎用-X 和-Y

使用这两个选项会导致坐标原点的移动。因而使用的时候需要相当慎重。

1. 除极个别的情况外，`-X` 和 `-Y` 选项应该仅在绘制组合图（即一张图多个子图）时使用
2. 对于非组合图，也可以在第一个绘图命令中使用 `-Xc -Yc` 使得整个绘图框架位于纸张的中央
3. 不要仅仅为了将某个符号或文字移动到某个位置就使用这两个选项，如果真的有这种需求的话，应该使用绝对坐标 `-Xa1c -Ya1c` ，其仅影响当前命令的绘图位置

8.6.12 网格文件后缀

GMT 主要使用 netCDF 格式作为网格数据的格式，其标准后缀名为 `.nc` 。

需要注意以下两个事实：

1. GMT 不会对后缀进行检测，所以后缀是什么都不重要
2. GMT 之前的版本中曾经自定义了一种网格数据格式，并使用后缀 `.grd` ，因而很多脚本中都使用了 `.grd` 作为后缀。

8.7 netCDF 文件格式

8.7.1 格式说明

netCDF 的非数据部分，包含了众多属性，这些属性完整地描述了 netCDF 文件的内容。下表列出了 netCDF 的众多属性：

表 8.1: netCDF 格式说明

Attribute	**Description**
	Global attributes
Conventions	COARDS, CF-1.5 (optional)
title	Title (optional)
source	How file was created (optional)
node_offset	0 for gridline node registration (default), 1 for pixel registration
	x- and y-variable attributes
long_name	Coordinate name (e.g., "Longitude" and "Latitude")
units	Unit of the coordinate (e.g., "degrees_east" and "degrees_north")
actual range (or valid range)	Minimum and maximum x and y of region; if absent the first and last x- and y-values are queried
	z-variable attributes
long_name	Name of the variable (default: "z")
units	Unit of the variable
scale_factor	Factor to multiply z with (default: 1)
add_offset	Offset to add to scaled z (default: 0)
actual_range	Minimum and maximum z (in unpacked units, optional) and z
_FillValue (or missing_value)	Value associated with missing or invalid data points; if absent an appropriate default value is assumed, depending on data type.

默认情况下，GMT 会将 netCDF 文件中的第一个 2D 变量作为 Z 变量，而坐标轴 X 和 Y 的范围则从属性中提取出来。

8.7.2 分块与压缩

出于性能的考虑，GMT 在输出超过 16384 个网格单元的网格文件时，会打开分块功能。所谓分块，即数据不是按照一行一行序列存储的，而是将整个网格分成若干个区块，然后依次存储每个区块的数据。

下图描绘了一个分块的 netCDF 文件的布局。为了读取数据的一部分（比如左下角的四个蓝色区块），netCDF 只需要读取相应的区块即可，不用先读取整个数据。

由于数据的压缩和解压比磁盘 IO 要快，因而可以对 netCDF 数据进行压缩，使得磁盘占用更少，IO 负载更少。netCDF 的压缩可以分为若干等级，压缩级别越高，文件越小，读写数据越快，但压缩/解压越耗时。通常，压缩级别取 1 到 3 效果比较好。

GMT 参数*IO_NC4_CHUNK_SIZE* 可以控制分块的大小，*IO_NC4_DEFLATION_LEVEL* 可以控制压缩等级。

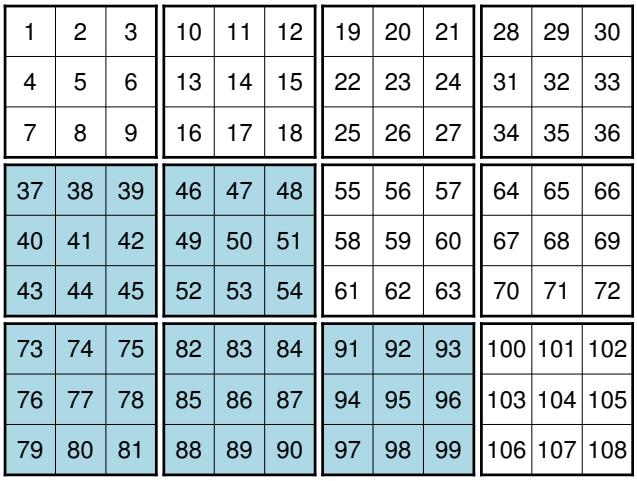


图 8.1: 网格分块

8.8 Native 二进制网格格式

头段区中包含了用于描述网格文件的变量，这些变量在文件中的存储顺序及其含义如下表:

表 8.2: GMT 自定义二进制网格文件结构

参数	类型	说明
nx	int	X 方向节点数目
ny	int	Y 方向节点数目
registration	int	配准方式：0 代表网格线配准，1 代表像素配准
x_min	double	区域的 X 最小值
x_max	double	区域的 X 最大值
ymin	double	区域的 Y 最小值
y_max	double	区域的 Y 最大值
z_min	double	数据的 Z 最小值
z_max	double	数据的 Z 最大值
x_inc	double	X 方向的节点间隔
y_inc	double	Y 方向的节点间隔
z_scale_factor	double	读取 Z 值后要乘以的因子
z_add_offset	double	Z 值乘以因子后要加上的偏移量
x_units	char[80]	X 方向的单位
y_units	char[80]	Y 方向的单位
z_units	char[80]	Z 方向的单位
title	char[80]	对数据集的描述
command	char[320]	生成该数据的命令
remark	char[160]	额外的注释
z	TYPE [nx*ny]	Z 值数组

8.9 Sun 光栅文件

Sun 光栅文件头段区的结构如下表：

表 8.3: Sun 光栅文件头段区

变量	类型	说明
ras_magic	int	魔法数，用于唯一标记数据格式
ras_width	int	图片宽度（像素数）
ras_height	int	图片高度（像素数）
ras_depth	int	像素深度（1、8、24、32 位）
ras_length	int	图片长度（字节数）
ras_type	int	文件类型
ras_maptype	int	colormap 类型
ras_maplength	int	接下来的 map 的长度（字节数）

在头段区后，若 ras_maptype 不等于 RMT_NONE ，则紧接着是 ras_maplength 个字节的 color map 区，然后是 ras_length 个字节的图片区。

相关的一些宏定义如下表:

表 8.4: Sun 头段区的宏定义

宏	说明
RAS_MAGIC	0x59a66a95
RT_STANDARD	1 (Raw pixrect image in 68000 byte order)
RT_BYTE_ENCODED	2 (Run-length compression of bytes)
RT_FORMAT_RGB	3 ([X]RGB instead of [X]BGR)
RMT_NONE	0 (ras_maplength is expected to be 0)
RMT_EQUAL_RGB	1 (red[ras_maplength/3],green[],blue[])

8.10 网格文件后缀

GMT 中也可以将网格文件的后缀与网格文件格式关联起来，就像 Windows 下 docx 后缀的文件与 MS Word 关联一样，这样 GMT 就可以直接根据文件后缀确定网格文件的格式了，这样更加直观也更加易用。

这一特性通过一个叫 gmt.io 的文件来实现。GMT 会依次在当前目录、家目录或 ~/.gmt 目录下，寻找 gmt.io 。

gmt.io 的示例格式如下:

```
# suffix format_id scale offset NaNxxxComments # GMT i/o shorthand file
# It can have any number of comment lines like this one anywhere
```

```
# suffix format_id scale offset  NaN   Comments
grd        nf        -     -      -    Default format
b          bf        -     -      -    Native binary floats
i2         bs        -     -    32767  2-byte integers with NaN value
ras        rb        -     -      -    Sun raster files
byte       bb        -     -     255   Native binary 1-byte grids
bit        bm        -     -      -    Native binary 0 or 1 grids
mask       bm        -     -      0    Native binary 1 or NaN masks
faa        bs      0.1     -    32767  Native binary gravity in 0.1 mGal
ns         ns        a     a      -    16-bit integer netCDF grid with auto-scale and␣
 ↪auto-offset
```

要使用这一特性，需要将参数*IO_GRIDFILE_SHORTHAND* 设置为 `true` 。此时，文件名 `file.i2` 等效于 `file.i2=bs///32767` ，`wet.mask` 等效于 `wet.mask=bm/1/0/0` 。

8.11 隔离模式

GMT 在运行的过程中会生成临时文件，比如 `gmt.conf` 和 `gmt.history` ，这两个文件用于在同一个图件的多个绘图命令之间交流共同的参数或选项。在使用这两个文件时可能会遇到一些问题，比如 `gmt.conf` 在脚本执行的中途被修改所造成的问题，以及在同一个目录下运行多个绘图脚本导致的 `gmt.history` 出现冲突等问题。

GMT 提供了隔离模式，使得每次脚本的每次运行，都会将临时文件放在不同的目录中，进而避免了可能出现的冲突。下面的脚本展示了隔离模式的使用:

```
ps=GMT_isolation.ps

# 创建临时文件夹
export GMT_TMPDIR=`mktemp -d /tmp/gmt.XXXXXX`

# 生成的 gmt.conf 位于 $GMT_TMPDIR 目录下
gmt gmtset COLOR_MODEL rgb FONT_ANNOT_PRIMARY 14p

# 生成临时文件到 $GMT_TMPDIR 目录
gmt grdmath -Rd -I1 Y = $GMT_TMPDIR/lat.nc
gmt makecpt -Crainbow -T-90/90/180 -Z > $GMT_TMPDIR/lat.cpt

# 生成的 gmt.history 文件位于 $GMT_TMPDIR 目录下
gmt grdimage $GMT_TMPDIR/lat.nc -JK6.5i -C$GMT_TMPDIR/lat.cpt -P -K -nl > $ps
gmt pscoast -R -J -O -Dc -A5000 -Gwhite -Bx60g30 -By30g30 >> $ps

# 清理临时目录
```

```
rm -rf $GMT_TMPDIR
```

8.12 获取 GMT 开发版

通常，从 GMT 官方网站下载得到的源码都是发布版源码，即是面向用户的源码。而 GMT 维护者在开发时所用所使用的源码是开发版源码，相对于发布版源码要多一些内容，比如开发版源码中多了大量的命令测试代码，是了解命令用法的好地方。

GMT 开发版源码通过 `svn` 管理，因而要先安装 `svn` ，然后执行如下命令即可获取 GMT 当前最新的开发源码:

```
svn checkout svn://gmtserver.soest.hawaii.edu/gmt5/trunk gmt5-dev
```

当然，也可以使用如下命令获取 GMT 的开发版的全部源码（包括所有分支）:

```
svn checkout svn://gmtserver.soest.hawaii.edu/gmt5/ gmt5-dev-full
```

Git 用户可以使用 git 提供的 svn 插件来获取源码:

```
git svn clone svn://gmtserver.soest.hawaii.edu/gmt5/trunk gmt5-dev
```

8.13 用 GMT 制作动画

GMT 只能生成 PS 格式的图片，所以 GMT 自身并不具有制作动画的功能。但动画本质上就是将一系列独立的图片按照顺序快速播放，所以可以利用 GMT 绘制多张独立的图片，并将这些图片作为动画的每一帧。

在制作一个动画时，通常需要按照顺序考虑如下问题:

1. 一个动画需要多少帧? 太少的帧数会导致动画存在不连续感，太多的帧数则可能需要绘制/播放很长时间。建议定义相关变量，使得可以根据实际情况进行微调
2. 通常一张动图中有很多元素是不变的。比如海岸线底图、一些图例信息等等。这些不变的元素，没有必要为每一帧都重复绘制。可以直接将这些不变的元素绘制完，然后将未关闭的文件复制到新文件中，并在新文件中绘制余下的部分
3. 帧计数器应初始化为 0，并随着循环不断递增。
4. 根据帧计数器中的值构建文件名，文件名必须有规律且唯一。例如 `plot_0001.ps` 就是不错的命名方式（0001 中的前置 0 不可省略，因为在将一系列图片转换成动画时需要涉及到各帧之间的先后顺序问题）

5. 将 PS 文件转换成图片，建议使用 GMT 自带的 `psconvert` 模块
6. 使用 ImageMagick 的 `convert` 命令将图片转换成 gif 或者 avi 等格式
7. 删除临时文件

以下是制作动画的整个流程的示意代码：

```
#!/bin/bash
PS=plot.ps

# 绘制图中的静态元素
gmt cmd1 ... -K > $PS
gmt cmd2 ... -K -O >> $PS

# 开始循环，i 为帧计数器，MAX 为最大帧数
for ((i=0; i<MAX; i++))
do
    # 构建新文件名 plot_xxxx.ps
    file = `echo $i | awk '{printf "plot_%04d.ps", $1}'`
    # 将已经绘制好静态元素的文件复制到新文件
    # 此时 file 中已经包含了静态元素
    cp $PS $file
    # 向 file 中添加动态元素
    gmt cmd3 ... -K -O >> $file
    gmt cmd4 ... -K -O >> $file
    gmt cmd5 ... -O >> $file
    # 将 PS 文件转换为某种图片格式
    gmt psconvert -A -P $file
done

# 利用 convert 命令将图片转换为 GIF 动图
convert *.jpg plot.gif

# 清理临时文件
rm *.ps *.jpg
```

8.14 兼容 OGR 的 GMT 矢量数据格式

8.14.1 简介

地理空间数据有多种格式，按照类型划分，可以大致分为光栅型（raster）和矢量型（vector）。

- 光栅型数据格式不完整列表：http://www.gdal.org/formats_list.html

- 矢量型数据格式不完整列表：http://www.gdal.org/ogr_formats.html

简单的说，在 GMT 中，netCDF 格式的网格文件属于光栅型地理空间数据，而一般的表数据则属于矢量型地理空间数据。

GDAL 是一个可以实现多种光栅型或矢量型地理空间数据格式间互相转换的库/工具，其全称为 Geospatial Data Abstraction Library。历史上，GDAL 仅用于处理光栅型数据格式，而 OGR 则仅用于处理矢量型数据格式。从 GDAL 2.0 开始，二者相互集成在一起，即 GDAL 已经具备了处理光栅型和矢量型地理空间数据格式的能力。本文中，提到 OGR 时，仅表示地理空间矢量数据格式。

一个矢量数据中，不仅仅有地理空间数据（地理坐标数据，点、线、多边形等），也可以有非地理空间数据（城市名等）。老版本的 GMT 只能处理地理空间数据，而不能利用非地理空间数据。GMT5 定义了一种兼容 OGR 的 GMT 矢量数据格式，通常称为 OGR/GMT 格式。这种格式中包含了地理空间和非地理空间数据，所有的非地理空间数据都以注释的形式写到文件中，因而 GMT4 也可以正常读取 OGR/GMT 格式的数据。OGR/GMT 格式中包含了非空间数据，使得 GMT 的输出可以很容易地被其他 GIS 或绘图软件所使用。

8.14.2 OGR/GMT 格式

OGR/GMT 格式的一些重要性质列举如下：

- 所有非空间数据都以注释行的形式写到文件中，这些注释行在 GMT4 中会被直接忽略
- 非空间数据的各个字段之间用空格分隔，每个字段均以字符 `@` 作为前缀，紧接着一个用于表征该字段内容的字符。每个字段内部的多个字符串之间用字符 `|` 分隔
- 字符 `\` 作为转义字符，比如字符串内 `\n` 表示换行
- 文件中，非空间数据均保存在空间数据之前。因而 GMT 在处理地理空间数据之前，以及解析了非地理空间信息，这些非地理空间信息可能会影响到地理空间数据的处理
- 数据文件的第一个注释行必须指定 OGR/GMT 格式的版本号，即 `@VGMT1.0`
- 为了兼容其他 GIS 格式（比如 shapefiles），OGR/GMT 格式中显式包含了一个字段，用于指定接下来的地理空间数据是点、线还是多边形
- 每个文件有一个头段注释，其中指定了当前文件所包含的地理特征，以及每个特征所对应的非地理属性（比如区域范围，投影方式等）
- 同一个 OGR/GMT 格式的文件中，所有数据段必须具有相同类型的特征（都是点或线或多边形）

8.14.3 OGR/GMT 元数据

在 OGR/GMT 格式的文件头部，需要包含一系列元数据信息。元数据用于描述整个文件的共同信息，比如版本号、几何类型、区域范围、投影方式、非空间数据的格式等信息。

8.14.3.1 格式版本号 @V

OGR/GMT 格式的版本号用 @V 来指定。因而 OGR/GMT 格式的文件的第一行的内容必须是:

```
# @VGMT1.0
```

其中 GMT1.0 是 OGR/GMT 格式的版本号。

8.14.3.2 几何类型 @G

@G 用于指定当前数据文件的几何类型，其后接的参数可以是:

- POINT: 包含多个数据点（每个点都可以有自己的头段记录）
- MULTIPOINT: 多点数据（所有的点共用同一个头段记录）
- LINESTRING: 包含多个独立的线段（即 GMT 中的多段数据，每条线段可以有自己的头段记录）
- MULTILINESTRING: 多线数据（文件中的所有线段是一个特性，共用同一个头段记录）
- POLYGON: 包含多个闭合多边形（每个多边形可以有自己的头段记录）
- MULTIPOLYGON: 多多边形数据（所有多边形共用同一个头段记录）

例如:

```
# @VGMT1.0 @GPOLYGON
```

8.14.3.3 区域范围 @R

@R 用于指定区域范围，其格式与 -R 选项类似。例如:

```
# @R150/190/-45/-54
```

8.14.3.4 投影信息 @J

投影信息用四个可选的字符串表示，每个字符串以 @J 开头。

- @Je: 投影的 EPSG 代码
- @Jg: GMT 中所使用的投影参数
- @Jp: 投影参数的 Proj.4 表示
- @Jw: 投影参数的 OGR WKT (well known text) 表示

示例:

```
# @Je4326 @JgX @Jp"+proj=longlat +ellps=WGS84+datum=WGS84 +no_defs"
# @Jw"GEOGCS[\"WGS84\",DATUM[\"WGS_1984\",SPHEROID\"WGS84\",6378137,\
298.257223563,AUTHORITY[\"EPSG\",\"7030\"]],TOWGS84[0,0,0,0,0,0,0],
AUTHORITY[\"EPSG\",\"6326\"]],PRIMEM[\"Greenwich\",0,\
AUTHORITY[\"EPSG\",\"8901\"]],UNIT[\"degree\",0.01745329251994328,\
AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",\"4326\"]]"
```

8.14.3.5 声明非空间字段 @N

@N 后接一个用于描述非空间字段名称的字符串，各个字段名称之间用 | 分隔。若字段名称中有空格，则必须用引号括起来。@N 必须有一个与之对应的 @T 。其中 @T 用于指定每个字段名称的数据类型。可取的数据类型包括 string 、interger 、double 、datetime 和 logical 。

示例:

```
# @VGMT1.0 @GPOLYGON @Nname|depth|id @Tstring|double|integer
```

表明数据文件中包含了多个多边形，每个多边形都可以有独立的头段记录以指定非空间信息，非空间信息有三个，分别是 name、depth 和 id，三个字段分别是字符串、浮点型和整型。

8.14.4 OGR/GMT 数据

元数据之后即是真正的数据，包括非空间数据和空间数据。

8.14.4.1 非空间数据

非空间数据用 @D 表示，紧跟着一系列以 | 分隔的字符串，每个字段的含义以及格式由 @N 和 @T 决定。

非空间数据所在的注释行应放在每段数据的坐标数据前。对于几何类型为 LINE 、POLYGON 、MULTILINE 或 MULTIPOLYGON 的数据而言，每段数据之间用特定的字符分隔，默认分隔符是 > 。非空间数据紧跟在 > 行之后。对于几何类型为 POINT 或 MULTIPOINT 的数据而言，则不需要分隔符。

@N 和 @D 中的字符串中若包含空格，则必须用双引号括起来。若字符串中本身包含双引号或 | ，则需要使用转义字符进行转义。若两个 | 之间为空，则表示对应的字段为空值。

一个点数据的头段示例:

```
# @VGMT1.0 @GPOINT @Nname|depth|id @Tstring|double|integer
# @D"Point 1"|-34.5|1
```

一个多边形数据的头段示例:

```
# @VGMT1.0 @GPOLYGON @Nname|depth|id @Tstring|double|integer
>
# @D"Area 1"|-34.5|1
```

8.14.4.2 多边形拓扑

旧版本的 GMT 只支持常规的多边形，不支持一个多边形内有个洞的情况。

GMT 通过在多边形数据前加上 @P 和 @H 来指定当前的数据段是外环还是内环，即是真正的多边形，还是多边形内的洞。@H 必须紧跟在对应的 @P 之后。

@H 所指定的洞不应该有任何 @D 值，因为非空间数据适用于整个特性，而 @H 所指定的多边形只是多边形的一部分，并不是一个新的多边形。

8.14.5 示例

点数据示例:

```
# @VGMT1.0 @GPOINT @Nname|depth|id
# @Tstring|double|integer
# @R178.43/178.5/-57.98/-34.5
# @Je4326
# @Jp"+proj=longlat +ellps=WGS84 +datum=WGS84+no_defs"
# FEATURE_DATA
# @D"point 1"|-34.5|1
178.5 -45.7
# @D"Point 2"|-57.98|2
```

```
178.43 -46.8
...
```

线数据示例:

```
# @VGMT1.0 @GLINESTRING @Nname|depth|id
# @Tstring|double|integer
# @R178.1/178.6/-48.7/-45.6
# @Jp"+proj=longlat +ellps=WGS84 +datum=WGS84+no_defs"
# FEATURE_DATA
> -W0.25p
# @D"Line 1"|-50|1
178.5 -45.7
178.6 -48.2
178.4 -48.7
178.1 -45.6
> -W0.25p
# @D"Line 2"|-57.98|$
178.43 -46.8
...
```

多边形数据示例:

```
# @VGMT1.0 @GPOLYGON @N"Polygon name"|substrate|id @Tstring|string|integer
# @R178.1/178.6/-48.7/-45.6
# @Jj@Jp"+proj=longlat +ellps=WGS84 +datum=WGS84+no_defs"
# FEATURE_DATA
> -Gblue -W0.25p
# @P
# @D"Area 1"|finesand|1
178.1 -45.6
178.1 -48.2
178.5 -48.2
178.5 -45.6
178.1 -45.6
>
# @H
# First hole in the preceding perimeter, so is technically still
# part of the same geometry, despite the preceding > character.
# No attribute data is provided, as this is inherited.
178.2 -45.4
178.2 -46.5
178.4 -46.5
178.4 -45.4
```

```
178.2 -45.4
>
# @P
...
```

8.15 GMT C API

GMT 为 C/Fortran 程序提供了两套 API, 包括 GMT API 和 postscriptlight 绘图库。C/Fortran 用户可以在自己的程序中直接调用这两套 API，以实现在程序中调用 GMT 模块或底层绘图库的功能。

由于这部分内容涉及到底层 API，所以通常只有高级用户才会涉及到，因而不会翻译这部分内容，高级用户若有需要，请自行阅读相应文档。

- GMT API
- postscriptlight

8.16 Linux 下的 GMT 中文支持

GMT 原生并不支持中文。为了让 GMT 支持中文，需要修改 ghostscript 和 GMT 的配置文件。本文的修改流程将以 CentOS7 为准，其他发行版与 CentOS7 的区别将在文末给出。

8.16.1 准备工作

8.16.1.1 安装 gs 中文配置文件

大多数发行版都已经默认安装了 gs。除此之外，还需要安装简体中文配置文件。CentOS 7 下中文配置文件可以通过如下命令安装:

```
$ sudo yum install ghostscript-chinese-zh_CN
```

安装完成后，中文配置文件的路径为 `/usr/share/ghostscript/conf.d/cidfmap.zh_CN` ，以下称为 ghostscript 中文配置文件。

8.16.1.2 GMT 字体配置文件

假定 GMT 的安装路径为 `/opt/GMT-5.2.1` ，则字体配置文件的路径为 `/opt/GMT-5.2.1/share/postscriptlight/PSL_standard_fonts.txt` 。

8.16.2 使 gs 支持中文

8.16.2.1 gs 中文配置文件

CentOS 7 中 ghostscript 中文配置文件的默认内容为:

```
/BousungEG-Light-GB <</FileType /TrueType /Path (/usr/share/fonts/wqy-zenhei/wqy-
↪zenhei.ttc) /SubfontId 0 /CSI [(GB1) 4] >> ;
/GBZenKai-Medium    <</FileType /TrueType /Path (/usr/share/fonts/wqy-zenhei/wqy-
↪zenhei.ttc) /SubfontId 0 /CSI [(GB1) 4] >> ;
/MSungGBK-Light     /BousungEG-Light-GB ;
/Adobe-GB1      /BousungEG-Light-GB ;
```

其中的细节可能看不懂，但是可以大概总结如下:

- 第一行定义了字体名为 `/BousungEG-Light-GB` ，对应的字体文件为 `/usr/share/fonts/wqy-zenhei/wqy-zenhei.ttc` ，也就是文泉驿正黑;
- 第二行定义了字体名为 `/GBZenKai-Medium` ，对应的字体文件也是文泉驿正黑;
- 第三行和第四行分别定义了字体名 `/MSungGBK-Light` 和 `/Adobe-GB1` ，这两种都对应于 `/BousungEG-Light-GB` ，相当于给字体定义了别名。

关于配置文件的几点说明:

- 字体名是任意的，比如字体名可以取为 `/ABC` ;
- 字体文件似乎只能是 `ttc` 或 `ttf` 格式的，当然修改参数也有可能可以使用其他格式的字体;
- 要注意确认字体文件是否存在，比如 CentOS7 下的 `wqy-zenhei.ttc` 字体实际上位于软件包 `wqy-zenhei-fonts` 中。若字体不存在，则需要安装相应软件包。

8.16.2.2 测试 gs 对 Linux 默认字体的支持

CentOS7 的 ghostscript 中文配置文件中，默认有四行，分别定义了四个字体名，尽管本质上这四个字体名都指向同一个字体。下面先测试一下如何让 gs 显示 Linux 的默认字体。

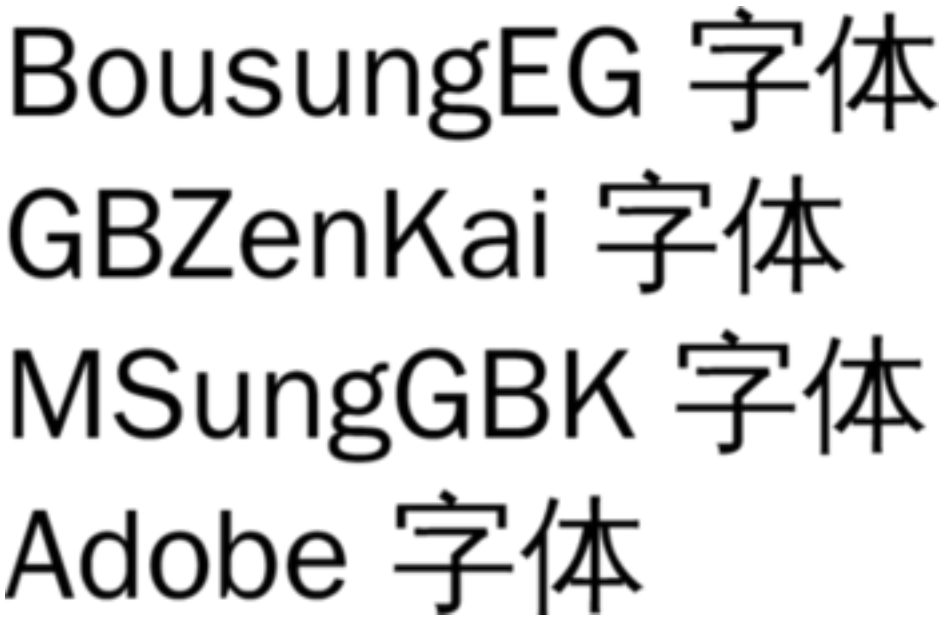
用 **编辑器**新建一个 PS 文件（是的，PS 文件其中就是纯文本，可以直接用编辑器编辑!），名为 `linux_fonts.ps` ，其内容为:

```
%! PS-Adobe-3. 0
/BousungEG-Light-GB--UniGB-UTF8-H findfont 20 scalefont setfont
150 400 moveto
(BousungEG 字体) show
```

```
/GBZenKai-Medium--UniGB-UTF8-H findfont 20 scalefont setfont
150 375 moveto
(GBZenKai 字体) show

/MSungGBK-Light--UniGB-UTF8-H findfont 20 scalefont setfont
150 350 moveto
(MSungGBK 字体) show

/Adobe-GB1--UniGB-UTF8-H findfont 20 scalefont setfont
150 325 moveto
(Adobe 字体) show

showpage
%%Trailer
%%EOF
```

简单解释一下，PS 文件中要使用某个中文字体，需要用 `FontName--CMap` 的格式来调用。其中 `FontName` 即 gs 中文配置文件中给定的字体名。CMap 可以取 `UniGB-UTF8-H` 和 `GB-EUC-H`，Linux 下一般用前者，Windows 下一般用后者，应该是用于指定汉字或中文字体的编码，具体原理不知。

用 gs 查看该 PS 文件，正常情况下显示如下图，表明 gs 可以正常显示 Linux 下的默认中文字体。



8.16.2.3 添加 Windows 中文字体

Linux 的中文字体较少，所以这里使用 Windows 下中的中文字体，这里只考虑 Windows 下的宋体、仿宋、黑体和楷体四个基本字体。将这四个字体文件复制到 `/usr/share/fonts/winfonts/` 目录下，然后对 gs 的中文配置文件做如下修改:

```
% 原内容保持不变
/BousungEG-Light-GB <</FileType /TrueType /Path (/usr/share/fonts/wqy-zenhei/wqy-
 ↪zenhei.ttc) /SubfontId 0 /CSI [(GB1) 4] >> ;
/GBZenKai-Medium    <</FileType /TrueType /Path (/usr/share/fonts/wqy-zenhei/wqy-
 ↪zenhei.ttc) /SubfontId 0 /CSI [(GB1) 4] >> ;
/MSungGBK-Light     /BousungEG-Light-GB ;
/Adobe-GB1      /BousungEG-Light-GB ;

% 新增 Windows 字体的支持
/STSong-Light <</FileType /TrueType /Path (/usr/share/fonts/winfonts/simsun.ttc) /
 ↪SubfontId 0 /CSI [(GB1) 4] >> ;
/STFangsong-Light <</FileType /TrueType /Path (/usr/share/fonts/winfonts/simfang.ttf)␣
 ↪/SubfontId 0 /CSI [(GB1) 4] >> ;
/STHeiti-Regular <</FileType /TrueType /Path (/usr/share/fonts/winfonts/simhei.ttf) /
 ↪SubfontId 0 /CSI [(GB1) 4] >> ;
/STKaiti-Regular <</FileType /TrueType /Path (/usr/share/fonts/winfonts/simkai.ttf) /
 ↪SubfontId 0 /CSI [(GB1) 4] >> ;
```

这里仅以 Windows 下的常用四大字体为例。对于 Windows 下的其他中文字体、Linux 的其他中文字体甚至日韩字体来说，方法类似。

8.16.2.4 测试 gs 对 Windows 中文字体的支持

用 **编辑器**新建一个 PS 文件，名为 `windows_fonts.ps` ，其内容为:

```
%! PS-Adobe-3. 0
/STSong-Light--UniGB-UTF8-H findfont 20 scalefont setfont
150 400 moveto
(Song Typeface 宋体) show

/STFangsong-Light--UniGB-UTF8-H findfont 20 scalefont setfont
150 375 moveto
(Fangsong Typeface 仿宋体) show

/STHeiti-Regular--UniGB-UTF8-H findfont 20 scalefont setfont
150 350 moveto
(Hei Typeface 黑体) show
```

```
/STKaiti-Regular--UniGB-UTF8-H findfont 20 scalefont setfont
150 325 moveto
（Kai Typeface 楷体）show

showpage
%%Trailer
%%EOF
```

用 gs 查看该 PS 文件，若正确显示中文如下图，则表明 gs 已支持 Windows 字体。

8.16.3 使 GMT 支持中文

8.16.3.1 修改 GMT 字体配置文件

打开 GMT 字体配置文件，在文件最后加入如下语句（以 Windows 下的四大常用字体为例）：

```
STSong-Light--UniGB-UTF8-H  0.700    1
STFangsong-Light--UniGB-UTF8-H  0.700    1
STHeiti-Regular--UniGB-UTF8-H   0.700   1
STKaiti-Regular--UniGB-UTF8-H   0.700   1
```

第一列为字体名，第二列为字母 A 的高度，第三列与编码有关。

8.16.3.2 查看 GMT 当前支持的字体

用 `pstext -L` 命令查看 GMT 当前的字体配置:

```
$ pstext -L
Font #  Font Name
-------------------------------------
0   Helvetica
1   Helvetica-Bold
...     ......
32  Palatino-BoldItalic
33  ZapfChancery-MediumItalic
34  ZapfDingbats
35 STSong-Light--UniGB-UTF8-H
36 STFangsong-Light--UniGB-UTF8-H
37 STHeiti-Regular--UniGB-UTF8-H
38 STKaiti-Regular--UniGB-UTF8-H
```

其中 0-34 为 GMT/gs 默认支持的西文字体，35 至 38 为新添加的中文字体。

8.16.3.3 GMT 中文测试

GMT5 测试脚本：

```
#!/bin/bash
gmt gmtset FONT_TITLE 40p,35,black

gmt pstext -R0/10/0/3 -JX15c/3c -Bafg -B+t"GMT 中文支持" -F+a+c+f -P > gmt5_cn.ps << EOF
3 2.1 0 LM 35p,35,red   GMT 宋体
3 0.9 0 LM 35p,36,blue  GMT 仿宋
7 2.1 0 LM 35p,37,black GMT 黑体
7 0.9 0 LM 35p,38,green GMT 楷体
EOF

rm gmt.*
```

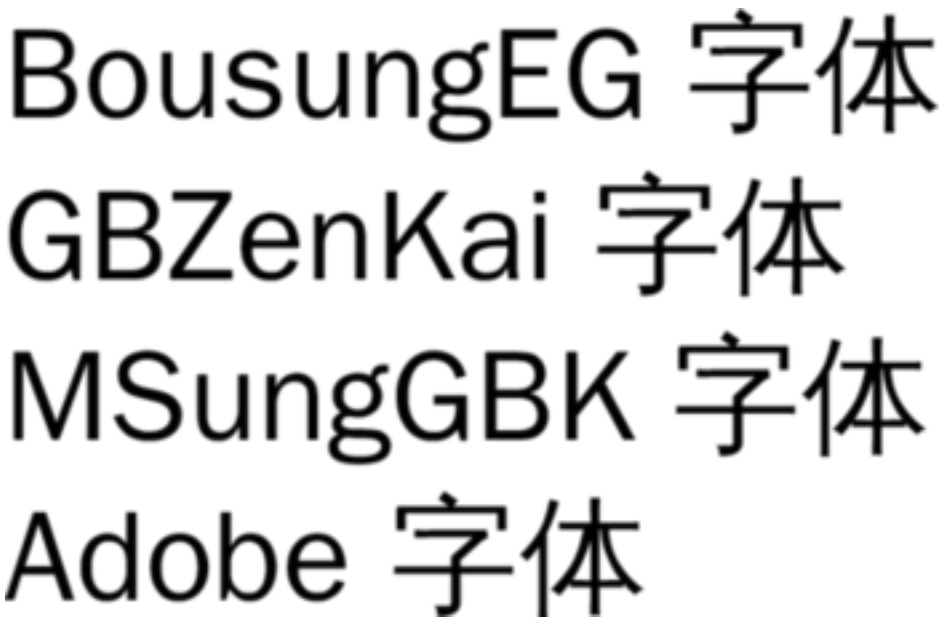
成图效果如下

8.16.4 对其他发行版的若干说明

其他发行版与 CentOS 7 之间或多或少有一些区别，列举如下。

8.16.4.1 CentOS 6

1. gs 中文配置文件需要用如下命令安装：




```
sudo yum install cjkuni-fonts-ghostscript
```

在安装配置文件的同时会安装中文字体 uming 和 ukai

2. gs 中文配置文件中给定的字体路径: /usr/share/fonts/cjkuni/uming.ttc 和 /usr/share/fonts/cjkuni/ukai.ttc 是错误的。正确的字体路径是 /usr/share/fonts/cjkui-uming/uming.ttc 和 /usr/share/fonts/cjkuni-ukai/ukai.ttc，要注意改正。

8.16.4.2 Ubuntu 14.04/15.04

1. gs 中文配置文件可以用如下命令安装（默认已安装）:

```
sudo apt-get install poppler-data
```

2. gs 中文配置文件路径为: /etc/ghostscript/cidfmap.d/90gs-cjk-resource-gb1.conf
3. gs 中文配置文件中默认使用的 Linux 字体为 uming 和 ukai，需要通过如下命令安装:

```
sudo apt-get install fonts-arphic-uming fonts-arphic-ukai
```

4. gs 中文配置文件的默认内容为:

```
/BousungEG-Light-GB <</FileType /TrueType /Path (/usr/share/fonts/truetype/
 ↪arphic/uming.ttc) /SubfontId 0 /CSI [(GB1) 4] >> ;
/GBZenKai-Medium    <</FileType /TrueType /Path (/usr/share/fonts/truetype/
 ↪arphic/ukai.ttc) /SubfontId 0 /CSI [(GB1) 4] >> ;
```

```
/Song-Medium /GBZenKai-Medium ;
/STSong-Light /BousungEG-Light-GB ;
/STFangsong-Light /BousungEG-Light-GB ;
/STHeiti-Regular /BousungEG-Light-GB ;
/STKaiti-Regular /BousungEG-Light-GB ;
/Adobe-GB1      /BousungEG-Light-GB ;
/Adobe-GB1-Bold /GBZenKai-Medium ;
```

需要将该文件改成:

```
% 原配置文件的内容，与 STSong-Light 等相关的四行必须删除
/BousungEG-Light-GB <</FileType /TrueType /Path (/usr/share/fonts/truetype/
 ↪arphic/uming.ttc) /SubfontId 0 /CSI [(GB1) 4] >> ;
/GBZenKai-Medium    <</FileType /TrueType /Path (/usr/share/fonts/truetype/
 ↪arphic/ukai.ttc) /SubfontId 0 /CSI [(GB1) 4] >> ;
/Song-Medium /GBZenKai-Medium ;
/Adobe-GB1      /BousungEG-Light-GB ;
/Adobe-GB1-Bold /GBZenKai-Medium ;

% 新增 Windows 字体的支持
/STSong-Light <</FileType /TrueType /Path (/usr/share/fonts/winfonts/simsun.ttc)␣
 ↪/SubfontId 0 /CSI [(GB1) 4] >> ;
/STFangsong-Light <</FileType /TrueType /Path (/usr/share/fonts/winfonts/simfang.
 ↪ttf) /SubfontId 0 /CSI [(GB1) 4] >> ;
/STHeiti-Regular <</FileType /TrueType /Path (/usr/share/fonts/winfonts/simhei.
 ↪ttf) /SubfontId 0 /CSI [(GB1) 4] >> ;
/STKaiti-Regular <</FileType /TrueType /Path (/usr/share/fonts/winfonts/simkai.
 ↪ttf) /SubfontId 0 /CSI [(GB1) 4] >> ;
```

修改完 gs 中文配置文件后，必须要执行如下命令:

```
$ sudo update-gsfontmap
```

该命令会将 `/etc/ghostscript/cidfmap.d/*.conf` 合并成单独的文件 `/var/lib/ghostscript/fonts/cidfmap`。gs 在需要中文字体时会读取 `/var/lib/ghostscript/fonts/cidfmap` 而不是 `/etc/ghostscript/cidfmap.d/*.conf`。这是 Ubuntu/Debian 和 CentOS 的一个很大不同。

8.16.4.3 Ubuntu 12.04

1. gs 中文配置文件需要用如下命令安装:

```
sudo apt-get install gs-cjk-resource
```

2. 其他部分未做测试，估计跟 Ubuntu 15.05 差不多。

8.16.5 参考资料

1. GMT 软件显示汉字的技术原理与实现，赵桂儒，《测绘通报》
2. ghostscript 中文打印经验
3. GMT 中文支持
4. 维基词条：PostScript
5. Debian Wiki

8.17 Windows 下的 GMT 中文支持

本文介绍如何让 GMT 在 Windows 下支持中文。

8.17.1 准备工作

1. 安装 Windows 版 GMT；
2. 安装 ghostscript；

   需要注意在安装的最后，会有一个生成 cidmap 的选项，选中该选项则表示会为当前系统自动生成中文所需的 cidmap 文件。默认该选项是被选中的，一定 **不要**将该选项取消；
3. 安装 gsview；

8.17.1.1 ghostscript 的中文支持

若 ghostscript 的安装没有问题，则在 `C:\Program Files\gs\gs9.15\examples\cjk` 目录下可以找到文件 `gscjk_ag.ps` 。

启动 cmd，键入如下命令:

```
cd "C:\Program Files\gs\gs9.16\bin"
gswin64.exe ..\examples\cjk\gscjk_ag.ps
```

该命令用命令行版本的 gswin64c 打开 gscjk_ag.ps ，若能看到中文，则说明 ghostscript 是可以正常支持中文的。

8.17.1.2 gsview 的中文支持

安装好 gsview 之后，PS 格式会自动与 gsview 关联。一般情况下，直接双击 PS 文件，就会用 gsview 打开该 PS 文件。

双击打开 gscjk_ag.ps ，一般情况下不会正确显示汉字。这是因为 gsview 在打开 PS 文件时没有找到汉字所对应的字体文件。

在 gsview 的“选项”->“高级配置”中，将 Ghostscript Options 由 -dNOPLATFONTS -sFONTPATH="c:\psfonts" 改成 -dNOPLATFONTS -sFONTPATH="C:\Windows\Fonts" ，此时 gsview 在调用 gswin64 时会将选项传递给 gswin64，gswin64 则会在 FONTPATH 中搜索字体。

配置完毕后，重新打开 gscjk_ag.ps ，若中文正常显示，则表示 gsview 已支持中文。

8.17.2 GMT 的中文支持

用 **编辑器**打开 gscjk_ag.ps ，会看到如下内容:

```
/STSong-Light--GB-EUC-H *findfont 20 scalefont setfont
150 400 moveto
(Song Typeface 宋体) show
/STFangsong-Light--GB-EUC-H *findfont 20 scalefont setfont
150 375 moveto
(Fangsong Typeface 仿宋体) show
/STHeiti-Regular--GB-EUC-H *findfont 20 scalefont setfont
150 350 moveto
(Hei Typeface 黑体) show
/STKaiti-Regular--GB-EUC-H *findfont 20 scalefont setfont
150 325 moveto
(Kai Typeface 楷体) show
%
/Times-Roman findfont 13 scalefont setfont
50 200 moveto
(* Chinese translation of"Ghostscript" is merely associative \
characters of these meanings.) show
50 200 13 sub moveto
(In Simplified Chinese articles, customarily we use just"Ghostscript" \
as it is.) show
```

其中 STSong-Light--GB-EUC-H 即为宋体，GB-EUC 是文字编码方式，H 表示水平字体，V 表示垂直向字体，这里给出了四种常见字体的名称

1. STSong-Light--GB-EUC-H
2. STFangsong-Light--GB-EUC-H
3. STHeiti-Regular--GB-EUC-H
4. STKaiti-Regular--GB-EUC-H

将这四种中文字体添加到 GMT 的字体配置文件中，GMT 版本不同，配置文件的位置也不同：

- GMT 5.1.2 及其之前版本：C:\programs\gmt5\share\pslib\PS_font_info.d
- GMT 5.2.1 及其之后版本:C:\programs\gmt5\share\postscriptlight\PSL_standard_fonts.txt

字体配置文件修改后，最后几行如下：

```
ZapfChancery-MediumItalic   0.610       0
ZapfDingbats            0.700       1
STSong-Light--GB-EUC-H  0.700    1
STFangsong-Light--GB-EUC-H  0.700    1
STHeiti-Regular--GB-EUC-H   0.700   1
STKaiti-Regular--GB-EUC-H   0.700   1
```

用 gmt pstext -L 查看 GMT 字体，可以看到，新添加的四种中文字体对应的字体编号为 35 到 38。

8.17.3 测试脚本

```
gmt gmtset FONT_TITLE 40p,35,black

echo 3.5 5 0 LM 45p,35,red  GMT 宋体 > tmp
echo 3.5 4 0 LM 45p,36,blue GMT 仿宋 >> tmp
echo 3.5 3 0 LM 45p,37,yellow GMT 黑体 >> tmp
echo 3.5 2 0 LM 45p,38,green GMT 楷体 >> tmp

gmt pstext tmp -R0/7/0/7 -JX6i/6i -Bafg -B+t"GMT 中文" -F+a+c+f -P > cn.ps
```

若生成的 PS 文件正常显示汉字，则表示 GMT 已经可以支持中文。

需要注意，若使用记事本编辑 bat 文件，则保存时应注意编码方式为 ANSI、Unicode 或 Unicode big endian，若使用 UTF-8 编码则会出现乱码；另外，很多编辑器默认将文本文件以 UTF-8 编码保存，因而可能需要修改编辑器的默认编码。

8.18 GMT 的 Matlab 接口

8.18.1 简介

GMT 的 Matlab 接口，顾名思义，提供了在 Matlab 中调用 GMT 命令的功能。通过该接口，GMT 的所有模块命令都可以在 Matlab 脚本中嵌入执行。GMT 命令生成的结果（grid 格网数据、table 表格数据、CPT 颜色表、文本文件、图片等）都可以作为 Matlab 变量进行运算；Matlab 中的矩阵变量也可以直接作为 GMT 的输入，执行 GMT 的命令。

8.18.2 安装

8.18.2.1 Windows 平台

GMT5.2 用户在 GMT 执行路径（默认为 `C:\programs\gmt5\bin` ）下已经存在 `gmt.m` 和 `gmtmex.mexw64` 两个文件，只要确保如下两点即可在 Windows 下使用该接口了。

- GMT 的执行路径已经加入了系统环境变量 path 中，保证系统可调用 GMT 命令；
- GMT 的执行路径已经加入 Matlab 的搜索路径下，保证 Matlab 可调用 GMT 命令，如下图所示。

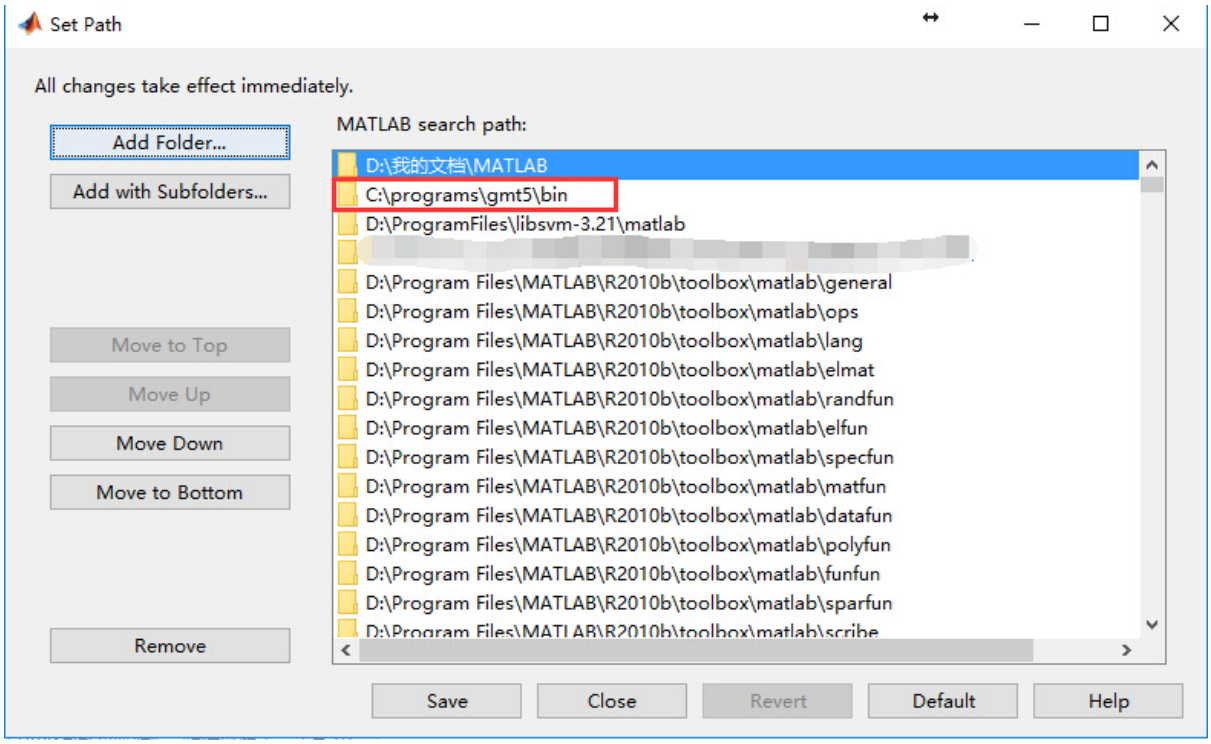


图 8.2: Matlab PATH 设置

8.18.2.2 OS X 平台

在 OS X 上按照如下流程可以成功编译 GMT 的 Matlab 接口。但由于 Matlab 处理动态链接库的方式很特别，因而编译接口的流程也很复杂。GMT 开发者正试图与 MathWorks 合作以简化安装流程。

1. 安装 OS X 平台下最新版本的 GMT；
2. 运行安装目录下 `share/tools` 下的 `gmt_prepmex.sh` 文件。此操作会复制 GMT 的已安装文件到 `/opt/gmt` 目录下，并且会重新检查所有的共享库；
3. 使用 `gmtswitch` 切换当前使用的 GMT 版本，确保 `/opt/gmt` 下的 GMT 为当前激活版本；
4. 使用 svn 获取 `gmt-mex` 项目文件到本地:

```
svn checkout svn://gmtserver.soest.hawaii.edu/gmt-mex gmt-mex
```

5. 进入 `get-mex` 目录并编译生成 `gmtmex.mexmaci64`

```
cd gmt-mex/trunk/
autoconf
./configure --enable-matlab
make
```

6. 将 `gmt.m` 和 `gmtmex.mexmaci64` 所在目录添加到 MTATLAB 路径中
7. 确保 `gmt.conf` 文件中包含选项:`GMT_CUSTOM_LIBS=/opt/gmt/lib/gmt/plugins/supplements.so`

经测试，该项目在 2015a，b 的 MATLAB 版本中可使用，对于更老版本的 MATLAB，还未进行测试。

8.18.2.3 Unix/Linux 平台

正在努力开发中，还望有志之士加入...

8.18.3 使用方法

GMT 接口完全模仿了传统的 matlab 命令，可以在命令行、m 文件或 IDE 中使用。形式是:

```
返回参数 = gmt('<module> <module-options>', 输入数据)
```

其中 **输入数据**可以为 Matlab 的矩阵、结构体或数组等；**返回参数**可直接在 Matlab 中参与后续的计算。调用 GMT 完毕后，清空缓存:

```
gmt('destroy')
```

8.18.3.1 入门级示例

在 matlab 环境中调用 pscoast 绘制地图:

```
gmt('pscoast -Rg -JA280/30/3.5i -Bg -Dc -A1000 -Gnavy -P > GMT_lambert_az_hemi.ps')
```

上例中，并不存在输入数据，也就是不存在与 Matlab 变量的交互，生成的 ps 文件在 Matlab 当前路径下。

8.18.3.2 进阶级示例

在 Matlab 环境中，绘制文字:

```
% 创建字符串数组
lines = {'5 6 Some label', '6 7 Another label'};
% 绘制
gmt('pstext -R0/10/0/10 -JM6i -Bafg -F+f18p -P > text.ps ', lines);
gmt('destroy');
```

上例中，字符串数组 lines 可以直接作为 pstext 的输入参数。

以上为单个输入参数，若需要多个输入参数，如何确定参数的先后顺序?

8.18.3.3 高手级示例

对一个矩阵数组进行格网化并绘图:

```
% 创建一个 100*3 矩阵，xyz 值均为 0~150 之间的随机数
t= rand(100,3)*150
% 利用 GMT 的 surface 命令对 t 进行格网化，输出为结构体 G，数组结构见附录
G = gmt('surface -R0/150/0/150 -I1', t );
% 利用 grd2cpt 创建颜色表文件，输出为颜色表结构体 cpt
cpt = gmt('grd2cpt -Cjet', G);
% 利用 grdimage 绘制格网化结果
gmt('grdimage -JX8c -Ba -P -C -G > crap_img.ps', cpt, G);
gmt('destroy');
```

上例中，grdimage 命令需要两个输入参数：颜色表 cpt 和格网数据 G ，两者先后顺序不可交换。cpt （选项 -C 的参数）要先于 G （grdimage 的强制性参数）。若有多个选项参数，则选项

的顺序决定参数的先后顺序，强制性输入参数要写在最后。

8.18.3.4 大神级示例

另一个多参数的例子：

```
x = linspace(-pi, pi)';            % 创建 x 值
seno = sin(x);                     % 创建 y 值
xyz  = [x seno seno];              % 创建 xyz 三列数据，其中 y=z
cpt  = gmt('makecpt -T-1/1/0.1');  % 创建 rainbow 颜色表
% 绘制函数曲线，以 z 值赋颜色。cpt 和 xyz 先后顺序不可交换。
gmt('psxy -R-3.2/3.2/-1.1/1.1 -JX12c -Sc0.1c -C -P -Ba > seno.ps', cpt, xyz);
gmt('destroy');
```

8.18.4 常见问题

- 使用完 GMT 接口后要记得 `gmt('destroy')`，不然有可能出现不可预知错误。
- gmt 括号内直接写 module 名，看似 GMT4 语句，实际只支持 GMT5 的语法。

8.18.5 附录

网格数据结构体说明:

```
ProjectionRefPROJ4      % Proj4 投影（Optional）
ProjectionRefWKT        % WKT 投影（Optional）
range                   % 1x6 向量表示数值范围:  [x_min x_max y_min y_max z_min z_max]
inc                     % 1x2 向量表示采样间隔:  [x_inc y_inc]
n_rows                  % 行数
n_columns               % 列数
n_bands                 % 波段数（维数）（目前未启用，恒 1）
registration            % 格网表达方式: 0 -> Grid registration; 1 -> Pixel registration
NoDataValue             % 空值
title                   % 标题
remark                  % Remark（Optional）
command                 % 生成命令（Optional）
DataType                % 数据格式'float' or 'double'
x                       % [1 x n 列]表示 X 坐标值
y                       % [1 x n 行]表示 y 坐标值
z                       % [n 行 x n 列]格网点值
x_units                 % x 轴单位（Optional）
y_units                 % y 轴单位（Optional）
```

8.19 GMT 的 Julia 接口

8.19.1 简介

Julia 是一门为科学计算设计的编程语言，简单易学。其与 Matlab、Python 等编程语言都有相似之处。GMT 提供了 Julia 接口，使得 Julia 用户可以直接在 Julia 脚本中调用 GMT 的相关模块。

8.19.2 安装

关于 Julia 的安装请参考 Julia 官方网站 的相关说明。

启动 Julia　并按照如下方式即可安装 GMT 的 Julia 接口（即 Julia 下的 GMT 模块）：

```
$ julia
               _
   _       _ _(_)_     |  A fresh approach to technical computing
  (_)     | (_) (_)    |  Documentation: http://docs.julialang.org
   _ _   _| |_  __ _   |  Type "?help" for help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 0.4.6 (2016-06-19 17:16 UTC)
 _/ |\__'_|_|_|\__'_|  |
|__/                   |  x86_64-redhat-linux

julia> Pkg.init()
INFO: Initializing package repository /home/seisman/.julia/v0.4
INFO: Cloning METADATA from git://github.com/JuliaLang/METADATA.jl

julia> Pkg.clone("git://github.com/joa-quim/GMT.jl.git")
INFO: Cloning GMT from git://github.com/joa-quim/GMT.jl.git
INFO: Computing changes...
INFO: No packages to install, update or remove
INFO: Package database updated
```

安装完成后，还需要通过如下命令:

```
echo 'push!(Libdl.DL_LOAD_PATH, "/opt/GMT-5.2.1/lib64")' >> ~/.juliarc.jl
```

将 GMT 的动态库文件所在目录添加到 Julia 的搜索路径中。

8.19.3 使用

在 Julia 中调用 GMT 的方式，与直接在命令行中调用 GMT 非常类似。通常来说，一个调用 GMT 的 Julia 脚本具有如下形式：

```
using GMT

返回值=gmt("<module> <options>", 输入参数)
gmt("destroy")
```

说明：

1. `using GMT` 的作用是在 Julia 中导入 GMT 模块，使得可以在 Julia 中通过 `gmt()` 函数调用 GMT 的所有模块。
2. 安装完 GMT 即可后第一次使用 `using GMT` 时，Julia 会对 GMT 即可进行预编译，因而会消耗一段时间，但之后再调用时，速度就非常快了
3. 函数 `gmt()` 用于调用 GMT 模块，其第一个参数与 GMT 命令行版本的参数几乎一致，之后的参数是当前命令所需的输入数据
4. 最后，调用函数 `gmt("destroy")` 以清理不需要的内存

8.19.3.1 最简单的例子

最简单的情况是，只是简单的绘图，不需要输入数据，也不生成任何数据。

```
using GMT
gmt("pscoast -Rg -JA280/30/3.5i -Bg -Dc -A1000 -Gnavy -P > GMT_lambert_az_hemi.ps")
```

该 Julia 命令等效于命令行版本的：

```
gmt pscoast -Rg -JA280/30/3.5i -Bg -Dc -A1000 -Gnavy -P > GMT_lambert_az_hemi.ps
```

8.19.3.2 向 GMT 传递数据

GMT 的 `surface` 命令会读入一个文本数据，对其进行插值以生成一个网格文件并绘图，例如：

```
gmt surface input.txt -Goutput.grd -R0/150/0/150 -I1
grdimage -Goutput.grd -JX8c -Ba -P -Cblue,red > crap_img.ps
```

下面看看在 Julia 中如何调用 `surface` 模块：

```
using GMT
t = rand(100,3) * 150;
G = gmt("surface -R0/150/0/150 -I1", t);
gmt("grdimage -JX8c -Ba -P -Cblue,red > crap_img.ps", G)
```

本例生成了一个 *100x3* 的随机数数组（矩阵）t ，并将其作为 gmt() 函数的第二个参数，即将数组 t 作为 surface 命令的输入数据（即命令行中的 input.txt ）。同时，将 surface 命令的输出数据（即命令行中生成的网格数据 -Goutput.grd ）保存到网格变量 G 中。

紧接着调用了 grdimage 模块绘制网格变量 G 。注意，在命令中使用或不使用 -G 选项是完全等效的。即上面例子中的最后一个命令也可以写成:

```
gmt("grdimage -JX8c -Ba -P -Cblue,red -G > crap_img.ps", G)
```

8.19.3.3 向 GMT 传递多个数据

若需要向 GMT 命令传递多个数据，则输入参数的顺序就变得很重要。

下面的例子在之前例子的基础上先生成了一个 CPT 文件，再利用 -C<cpt> 选项绘图。

```
using GMT
t = rand(100,3) * 150;
G = gmt("surface -R0/150/0/150 -I1", t);
cpt = gmt("grd2cpt -Cblue,red", G);
gmt("grdimage -JX8c -Ba -P -C -G > crap_img.ps", cpt, G)
```

命令行版本中的命令应该是:

```
gmt grdimage -JX8c -Ba -P -Cbluered.cpt -Goutput.grd > crap_img.ps
```

在 Julia 中，CPT 文件和网格文件作为输入数据，其在参数列表中的顺序由命令中 -C 和 -G 选项的先后顺序决定。

8.19.3.4 其他示例

下面的例子展示了如何绘制一条彩色渐变的正弦函数曲线:

```
using GMT
x = linspace(-pi, pi);              # The *xx* var
seno = sin(x);                      # *yy*
xyz  = [x seno seno];               # Duplicate *yy* so that it can be colored
```

```
cpt  = gmt("makecpt -T-1/1/0.1");  # Create a CPT
gmt("psxy -R-3.2/3.2/-1.1/1.1 -JX12c -Sc0.1c -C -P -Ba > seno.ps", cpt, xyz)
```

注意，由于变量 cpt 对应的是 -C 选项，而变量 xyz 是 psxy 模块的直接输入数据，所以，此处输入参数的顺序必须是 cpt,xyz 而不能是 xyz,cpt 。

下面的例子展示了如何写字符串：

```
using GMT
lines = Any["5 6 Some label", "6 7 Another label"];
gmt("pstext -R0/10/0/10 -JM6i -Bafg -F+f18p -P > text.ps", lines)
```

8.19.4 附录

gmt() 函数会返回多种类型的变量，比如上面例子中涉及到的 CPT 类型和网格类型的变量。因而需要在 Julia 中专门定义相关类型的变量。

Julia 中网格变量 GMTJL_GRID 的定义为:

```
type GMTJL_GRID      # The type holding a local header and data of a GMT grid
   ProjectionRefPROJ4::ASCIIString    # Projection string in PROJ4 syntax (Optional)
   ProjectionRefWKT::ASCIIString      # Projection string in WKT syntax (Optional)
   range::Array{Float64,1}            # 1x6 vector with [x_min x_max y_min y_max z_min␣
↪z_max]
   inc::Array{Float64,1}              # 1x2 vector with [x_inc y_inc]
   n_rows::Int                        # Number of rows in grid
   n_columns::Int                     # Number of columns in grid
   n_bands::Int                       # Not-yet used (always == 1)
   registration::Int                  # Registration type: 0 -> Grid registration; 1 ->
↪ Pixel registration
   NoDataValue::Float64               # The value of nodata
   title::ASCIIString                 # Title (Optional)
   remark::ASCIIString                # Remark (Optional)
   command::ASCIIString               # Command used to create the grid (Optional)
   DataType::ASCIIString              # 'float' or 'double'
   x::Array{Float64,1}                # [1 x n_columns] vector with XX coordinates
   y::Array{Float64,1}                # [1 x n_rows]    vector with YY coordinates
   z::Array{Float32,2}                # [n_rows x n_columns] grid array
   x_units::ASCIIString               # Units of XX axis (Optional)
   y_units::ASCIIString               # Units of YY axis (Optional)
   z_units::ASCIIString               # Units of ZZ axis (Optional)
end
```

图片变量 GMTJL_IMAGE 的定义为:

```
type GMTJL_IMAGE     # The type holding a local header and data of a GMT image
   ProjectionRefPROJ4::ASCIIString    # Projection string in PROJ4 syntax (Optional)
   ProjectionRefWKT::ASCIIString      # Projection string in WKT syntax (Optional)
   range::Array{Float64,1}            # 1x6 vector with [x_min x_max y_min y_max z_min
↪z_max]
   inc::Array{Float64,1}              # 1x2 vector with [x_inc y_inc]
   n_rows::Int                        # Number of rows in image
   n_columns::Int                     # Number of columns in image
   n_bands::Int                       # Number of bands in image
   registration::Int                  # Registration type: 0 -> Grid registration; 1 ->
↪ Pixel registration
   NoDataValue::Float64               # The value of nodata
   title::ASCIIString                 # Title (Optional)
   remark::ASCIIString                # Remark (Optional)
   command::ASCIIString               # Command used to create the image (Optional)
   DataType::ASCIIString              # 'uint8' or 'int8' (needs checking)
   x::Array{Float64,1}                # [1 x n_columns] vector with XX coordinates
   y::Array{Float64,1}                # [1 x n_rows]    vector with YY coordinates
   image::Array{UInt8,3}              # [n_rows x n_columns x n_bands] image array
   x_units::ASCIIString               # Units of XX axis (Optional)
   y_units::ASCIIString               # Units of YY axis (Optional)
   z_units::ASCIIString               # Units of ZZ axis (Optional) ==> MAKES NO SENSE
   colormap::Array{Clong,1}           #
   alpha::Array{UInt8,2}              # A [n_rows x n_columns] alpha array
end
```

CPT 变量 GMTJL_CPT 的定义为:

```
type GMTJL_CPT
    colormap::Array{Float64,2}
    alpha::Array{Float64,1}
    range::Array{Float64,2}
    rangeMinMax::Array{Float64,1}
end
```

第九章　FAQ

1. 使用 psxy 或其他类似命令时，出现:

```
psxy: Mismatch between actual (1) and expected (2) fields near line 10 (skipped)
```

出现该错误的原因是当前命令需要两列数据，而在第 10 行左右数据只有一列。此时需要人工检查一下第 10 行附近的数据。需要注意，这里的第 10 行不一定是准备数字。

2. 在使用地图投影时，为何无法给坐标轴加标签？命令如下:

```
gmt psbasemap -R0/10/0/10 -JM10c -B1 -Bx+l"Latitude" -By+"Longtitude" > a.ps
```

从绘图效果中可以看出，“Latitude” 和 “Longtitude” 是没有加进去的。这不是 Bug，而是一个 Feature。因为对于使用地图投影绘制的地图，从 “火车道” 边框或标注上的 “度” 符号都可以直观的知道这是一个地图投影而不是笛卡尔投影。而对于地图投影来说，X 方向是经度、Y 方向是纬度，是基本常识，给坐标轴加 label 反而是多此一举，故而对于地图投影来说，坐标轴是不能加 label 的。

如果执意要给坐标轴加 label，只能用 pstext 手动添加。

3. 在 Windows 下写了 bat 脚本，执行 bat 脚本时黑屏一闪而过，看不到出错信息。

解决办法：在脚本的最后加上 `pause` 。

4. 出现如下错误:

```
Long input record (xxx bytes) was truncated to first xx bytes!
```

出现该错误的原因是输入数据的一行超过特定的长度（一般是 512 字符），此时 GMT 会对输入数据做截断。出现该错误的常见原因是在将多个记录通过管道传递给命令时，记录末尾的换行符丢失，导致多个记录连成一行传递给命令。

5. 安装 GMT 后用 `pscoast` 绘图时出现如下错误:

```
pscoast: GSHHG version 2.2.0 or newer is needed to use coastlines with GMT.
Get and install GSHHG from ftp://ftp.soest.hawaii.edu/gshhg/.
```

```
pscoast: Could not find file [GSHHG crude resolution shorelines]
pscoast: No GSHHG databases available - must abort
```

`pscoast` 能够正常运行的前提是：

(a) 在 `$GMTHOME/share/coast` 目录下能够找到 GSHHG 数据文件（一堆以.nc 结尾的文件）

(b) GSHHG 数据文件有可读权限

(c) GSHHG 数据文件的版本号大于 2.2.0

(d) 系统的 netCDF 软件在编译时支持 netCDF4 格式

针对以上几点，如果出现了题中所说的错误，可以通过如下方式逐一排除：

(a) 检查 `$GMTHOME/share/coast` 目录下是否有 GSHHG 数据文件

(b) 在 `share/coast` 目录下执行 `ls -l` 检查数据文件是否有可读权限

(c) 打开 `README.TXT` 里，版本号是否大于 2.2.0

(d) 终端执行 `nc-config --has-nc4` 以检查 netCDF 软件是否支持 netCDF4 格式，若为 `yes` 则表示支持

索引