Qipeng Bai's Blog

Qipeng Bai

2018 年 4 月 19 日

目录

第一部分　地震学、计算机科学 **2**

第一章　2016 年 **3**

第二章　### 你好! **4**

2.1　韩国地震 CAP 机制解 4

第三章　### 地震时间处理 **7**

第四章　2017 年 **10**

第五章　### 新西兰地震余震可视化 **11**

第六章　2018 年 **14**

6.1　顶级域名绑定 GitHub Pages 14

第七章　### Hugo 安装配置 **15**

第八章　### 二级域名绑定 GitHub Pages **18**

第一部分

地震学、计算机科学

第一章　2016 年

第二章　### 你好!

欢迎来到 **白起鹏**的个人博客。

本博客收集资料与 2016 年，整理发布于 2018 年，主要关注地震学、计算机科学、深度学习、音乐与心理学的知识整理与分享。

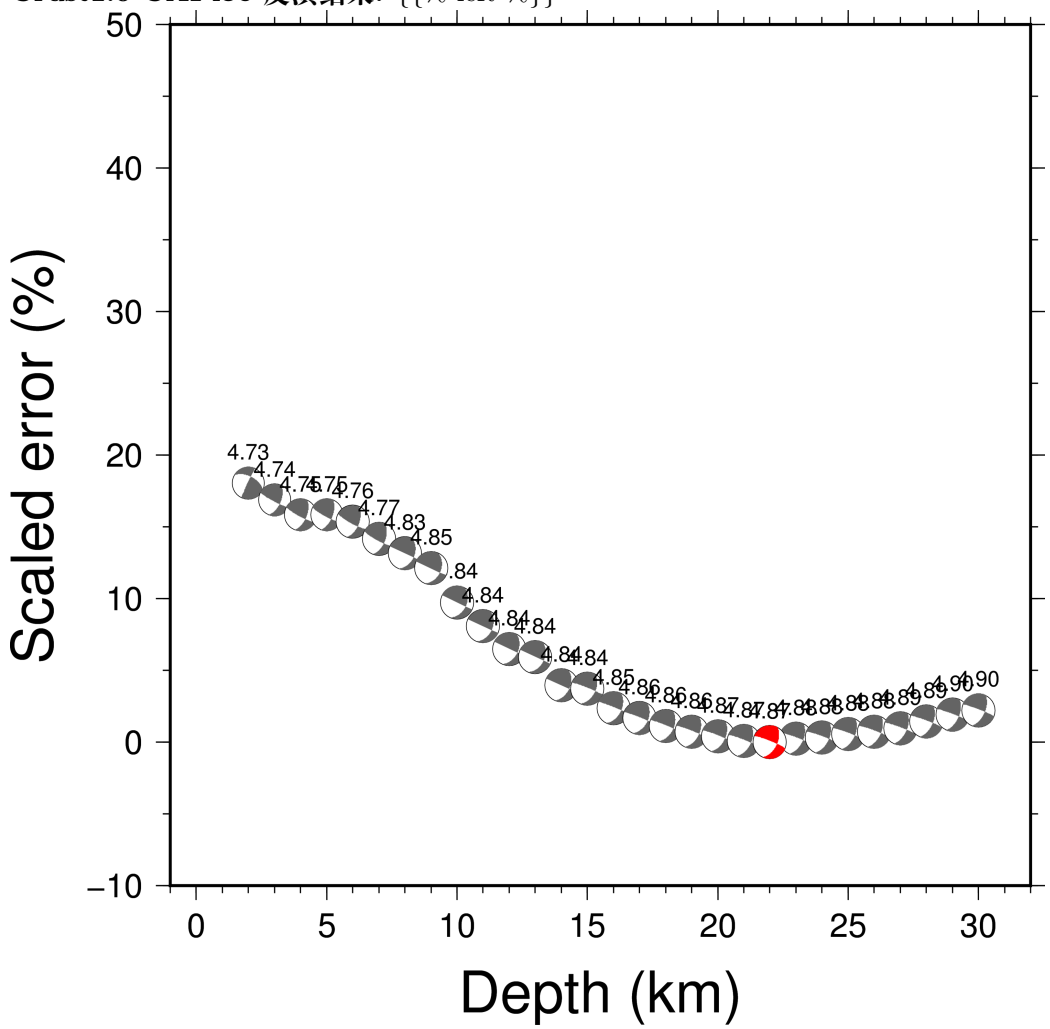
2.1　韩国地震 CAP 机制解

2016-09-12 在韩国庆州境内发生两次地震，本文给出 **CAP** 近震的机制解反演结果。

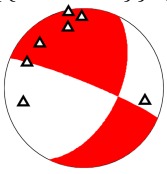
此反演结果使用的数据来源于 IRIS M4.9 [IRIS M5.4]
(http://ds.iris.edu/wilber3/find_stations/5192991)，地震基本信息来源于USGS M4.9 [USGS M5.4]
(http://earthquake.usgs.gov/earthquakes/eventpage/us10006p1f#executive)。
1. 地震速报 ***NEWS***
【正式测定：韩国庆州 4.9 级和 5.4 级地震】USGS 正式测定：2016-09-12 在韩国庆州境内发生两次地震，M4.9 级震源深度 10.0km，M5.4 级震源深度 10.0km。
SCIENTIFIC **Neic earthquake location information:** Magnitude: 5.4 Mb ±0.0 Location: 35.769°N 129.227°E ±4.9km Depth: 10.0 km ±1.8 Origin Time: 2016-09-12 11:32:54.940 UTC
USGS moment tensors

2. CAPloc 机制解
M4.9

Crust1.0 CAPloc 反演结果: {{% left %}}

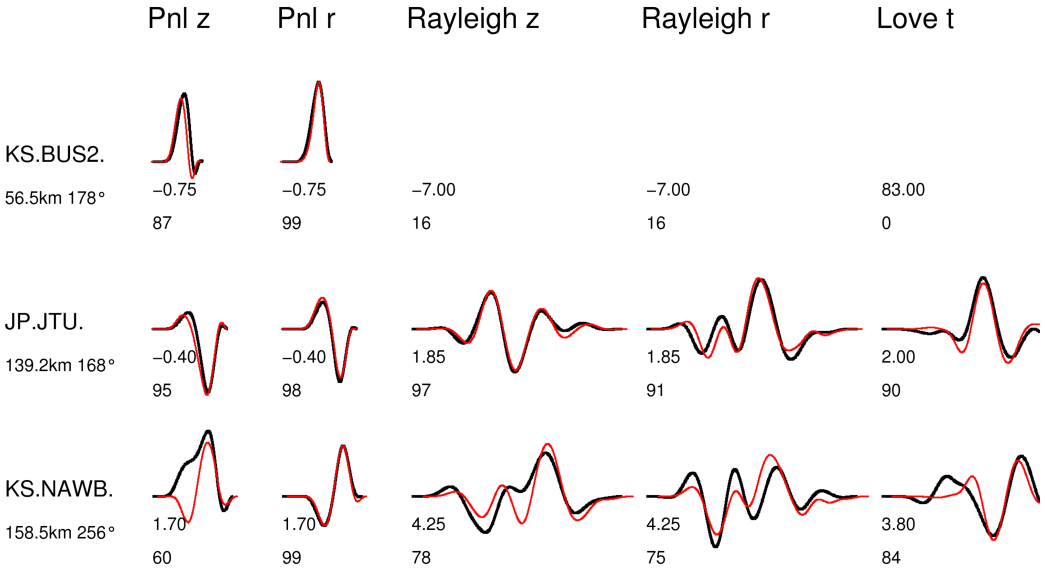


{{% left %}} {{% right %}}



Event: Mw4.9_SouthKoreadata Depth: 22km

FM 292 84 −36 Mw 4.87 Error 1.263e−02



(Kim et al., 1999) 文章中 model C 速度模型 CAPloc 反演结果:
Crust1.0 gCAPloc 反演结果:
M5.4
Crust1.0 CAPloc<img 全台站反演结果:
Crust1.0 CAPloc 挑选通道后反演结果:
Crust1.0 gCAPloc 全台站反演结果:
Crust1.0 gCAPloc 挑选通道反演结果:

© USTC 白起鹏 baiqp@mail.ustc.edu.cn

第三章　### 地震时间处理

通常在处理地震数据（尤其是 SAC 数据）时需要计算不同时间戳（Time Stamp），本文给出一些示例。

C 中的时间计算、转换

地震的时间处理，主要是对不同格式的转换，这里我先给出一个转换的 C 程序。

```c
#include "stdio.h"
main()
{
int day,month,year,sum,leap;
scanf("%d-%d-%d",&year,&month,&day);
switch(month) /*先计算某月以前月份的总天数*/
{
  case 1:sum=0;break;
  case 2:sum=31;break;
  case 3:sum=59;break;
  case 4:sum=90;break;
  case 5:sum=120;break;
  case 6:sum=151;break;
  case 7:sum=181;break;
  case 8:sum=212;break;
  case 9:sum=243;break;
  case 10:sum=273;break;
printf("%d",sum);
}
printf("%d",sum);
}
```

一般来说，时间处理主要是进行以下几件工作：

- 时间格式转换
- 时间差计算
- 时间统计-程序运行

地震学中的问题主要是时间的格式转换和时差计算，

Shell 中的时间格式转换、时差计算

date 通常情况下可以分：年月日时分秒 一般情况下需要对 date 做四则运算。shell 中命令的空格要严格控制，等号两边无空格，有多余空格会出现错误，这中限制在日期 date 命令中就比较明显。

命令格式：

```
date [-u] [-d datestr] [-s datestr] [--utc] [--universal] [--date=datestr] [--set=
    datestr] [--help] [-version] [+FORMAT] [MMDDhhmm[[CC]YY][.ss]]
```

通常我们拿到的时间格式会有以下几种：

- CSV: %Y-%m-%dT%H:%M:%SZ 例如： 2016-10-21T05:07:23.000Z
- SAC: %Y %j %H %M %S 例如：2016 295 05 07 23.00，或者是 saclst kzdate kztime f sacfile 得到 %Y/%m/%d %H:%M:%S 的时间格式
- 其他: 例如 2016年 10月 21日 星期五 05: 07: 23 CST等

date 显示指定格式的时间 date -d "STRING" +"FORMAT"，尽量加上双引号，避免空格。由此我们可以这样计算，取出时间处理成%Y-%m-%d %H:%M:%S格式，然后转换成自 UTC 时间1970-01-01 00:00:00 以来所有时间的秒数，然后通过秒数进行四则运算，最后再转换为想要的格式。这里暂且定义 %Y-%m-%d %H:%M:%S 为标准格式，其他时间都以此为对比标准。

```
#! /bin/bash
catalog=query.csv
outfile=LocalTime
localtimezone=$1

gawk -F "," 'NR>=2 {print $1}' $catalog | sed 's/Z/ /g' | sed 's/T/ /g' >
    StandardTime
echo -e "\033[35m\tStandard time: \033[0m"
cat StandardTime
cat StandardTime | while read line
do
        UTCtimestamp=`date -d "$line" +%s`
        localtimestamp=`expr $UTCtimestamp + $localtimezone \* 60 \* 60 | bc` #
            oneday=86400(s)
        echo `date -d @$localtimestamp +"%Y %j %H %M %S"` >> LocalTime
done
echo -e "\033[35m\tLocal time: \033[0m"
cat LocalTime
rm -rf StandardTime
```

```
#! /bin/bash
if [ $# -ne 2 ];
then
        echo 'Usage: SAC2standTime.sh filepath timezone(hour)'
else
sacpath=$1
localtimezone=$2 # 指定时区校正值，
```

```
sac="*.U"
cd $sacpath/
line=`saclst kzdate kztime f $sac | head -1 | gawk '{split($2,aa,"/");print aa[1]"
    -"aa[2]"-"aa[3],$3}'`
localtimestamp=`date -d "$line" +%s`
UTCtimestamp=`expr $localtimestamp - $localtimezone \* 60 \* 60 | bc` # oneday
    =86400(s) onehour=3600(s)
echo `date -d @$UTCtimestamp +"%Y %j %H %M %S"`
fi
```

通过上述两个 shell 脚本将标准时间和 SAC 中的时间进行互相转换，有效!

Python 中的 date 运算;

Perl 中的 date 运算;

Reference:

第四章　2017 年

第五章　### 新西兰地震余震可视化

地震学中的可视化工作多种多样，这里我们首先对新西兰地震余震震中随时间演化的可视化给出一个示例。

Python 具有十分丰富的扩展库来进行可视化工作，例如 Matplotlib, Mayavi, VTK 等等，这里我们主要关注的是地震学的数据可视化。

Aftershock

> The smaller aftershocks following a mainshock have a characteristic distribution in size and time. As Fig. 4.5-9, most aftershock occur on or near the mainshock's fault plane, so their location are used to distinguish between the fault and auxiliary planes and to estimate the fault area.
>
> – Seth Stein, An Introduction to Seismology Earthquake and Earth Structure

作为例子，我们先选取 **2016-11-20-M7.8 新西兰地震**作为主震，从`USGS earthquake feed`下载 csv 格式的余震目录数据，这里使用了 Python3 的 `urllib` 包作为获取目录的模块，原始的 csv 文件是后发生地震在前，为了画出随主震后时间演化的地震，我们将数据从后往前取值。

```
import urllib
import numpy as np
import matplotlib
matplotlib.rcParams['toolbar'] = 'None'
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
#from matplotlib.animation import FuncAnimation
import matplotlib.animation as animation


# Open the earthquake data
# ------------------------
# -> http://earthquake.usgs.gov/earthquakes/feed/v1.0/csv.php
feed = "http://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/"

# Significant earthquakes in the past 30 days
# url = urllib.request.urlopen(feed + "significant_month.csv")

# Earthquakes of magnitude > 4.5 in the past 30 days
```

```
# url = urllib.request.urlopen(feed + "4.5_month.csv")
url = urllib.request.urlopen(feed + "4.5_week.csv")

# Earthquakes of magnitude > 2.5 in the past 30 days
# url = urllib.request.urlopen(feed + "2.5_month.csv")
# url = urllib.request.urlopen(feed + "2.5_week.csv")

# Earthquakes of magnitude > 1.0 in the past 30 days
# url = urllib.request.urlopen(feed + "1.0_month.csv")

# Set earthquake data
data = url.read()
data = data.split(b'\n')[+1:-1]
E = np.zeros(len(data), dtype=[('position', float, 2),
                               ('magnitude', float, 1)])
for i in range(len(data)):
    row = data[i].split(b',')
    E['position'][i] = float(row[2]),float(row[1])
    E['magnitude'][i] = float(row[4])


fig = plt.figure()
ax = plt.subplot(1,1,1)
P = np.zeros(50, dtype=[('position', float, 2),
                        ('size',   float, 1),
                        ('growth', float, 1),
                        ('color',  float, 4)])

# Basemap projection
map = Basemap(projection='mill')
map.drawcoastlines(color='0.40', linewidth=0.25)
map.fillcontinents(color='0.60')
scat = ax.scatter(P['position'][:,0], P['position'][:,1], P['size'], lw=0.5,
                  edgecolors = P['color'], facecolors='None', zorder=10)


def update(frame):
    current = frame % len(E)
    i = frame % len(P)

    P['color'][:,3] = np.maximum(0, P['color'][:,3] - 1.0/len(P))
    P['size'] += P['growth']

    magnitude = E['magnitude'][current]
    P['position'][i] = map(*E['position'][current])
    P['size'][i] = 5
    P['growth'][i]= np.exp(magnitude) * 0.1

```

```
    if magnitude < 6:
        P['color'][i]  = 0,0,1,1
    else:
        P['color'][i]  = 1,0,0,1
    scat.set_edgecolors(P['color'])
    scat.set_facecolors(P['color']*(1,1,1,0.25))
    scat.set_sizes(P['size'])
    scat.set_offsets(P['position'])

plt.title("Earthquakes > 4.5 in the last 30 days")
anim = animation.FuncAnimation(fig, update, interval=100)
anim.save('line.gif', dpi=100, writer='imagemagick')
```

演示效果:

第六章　2018 年

6.1　顶级域名绑定 GitHub Pages

使用 Hugo 编译后的静态博客，在 GitHub Pages 上发布，可以直接在 https://username.github.io 查看。当然一些人想使用自定义域名，可以通过 Godaddy 申请一个域名。

申请域名后，可以通过将域名和GitHub的仓库绑定，则GitHub Pages的网页域名解析为自定义域名。具体做法如下，进入 DNS 管理，修改记录如下：
| 类型 | 名称 | 值 | TTL | |:————|:————|:————————-|:————| | A | @ | 192.30.252.153 | 1 小时 | | CNAME | www | bqpseismology.github.io | 1 小时 | | … | | | |
访问我的博客，请戳这里

:heavy_exclamation_mark: 注意：

- 不要使用默认的 CNAME 的记录，将默认的删去，否则会影响GitHub Pages的正确解析。
- 使用 http:// 访问，使用 https:// 访问会出现错误。

链接：

- GoDaddy
- GitHub

第七章　### Hugo 安装配置

Hugo 是由 Go 语言实现的静态网站生成器。简单易用、高效、易扩展、快速部署。

7.0.1　1.Hugo 安装配置

具体可参考官方中文参考文档
直接安装步骤如下：

- 从Hugo Releases下载对应操作系统的Hugo二进制文件，放在/bin/目录下运行
- 安装依赖工具：Git，Mercruial，Go(1.3+)
- 安装 Visual Studio Code ，方便修改代码

注意 Go 的高版本编译需要 Go 1.4 作为初始编译器，因此要先安装后再安装高版本，可参考 Go 高版本安装 。

1.1 让 go get 显示进度

在使用 Go 下载 github 的包比较大时，需要让屏幕显示进度，可以通过修改 Go 源码来实现。打开 /usr/local/go/src/cmd/go/vcs.go 文件，如下修改：

```
// vcsGit describes how to use Git.
var vcsGit = &vcsCmd{
    name: "Git",
    cmd: "git",

    createCmd: "clone {repo} {dir}", // 此处修改为 clone --progress {repo} {dir}
        update --init --recursive"},
```

```
var buf bytes.Buffer
    cmd.Stdout = &buf
    cmd.Stderr = &buf
    cmd.Stdout = os.Stdout // 重定向标准输出
    cmd.Stderr = os.Stderr // 重定向标准输出
    err = cmd.Run()
```

然后运行 src/all.bash 重新编译 Go，等编译完成后使用 go get 可以看到进度条。

1.2 安装 pandoc

使用二进制包安装，具体流程如下：

```
$ wget -c https://github.com/jgm/pandoc/releases/download/2.1.3/pandoc-2.1.3-linux
    .tar.gz
$ tar -zxvf pandoc-2.1.3-linux.tar.gz
$ su
# mv pandoc-2.1.3 /usr/local/
$ vim ~/.bashrc
At the end of the file, add  “export PATH=/usr/local/pandoc-2.1.3/bin:${PATH}”
$ source ~/.bashrc
```

7.0.2 2. 主题安装

参考 even

- 修改 /themes/even/src/css/ 目录下的版本文件，调整了网页的样式。
- 修改 /themes/even/src/fonts/ 目录下的字体文件及 /themes/even/src/css/_custom/ 的 _custom .scss 文件个性化定制页面。

注意修改 /themes/even/src/ 内任一文件时，要再次编译，编译命令如下：

```
$ cd src/
$ npm install
$ npm run build
```

7.0.3 3. 字体修改

3.1 字体安装

字体转换工具可使用 Everthing Fonts

Abode Source Code Pro

```
$ sudo yum install adobe-source-code-pro-fonts
$ cd /usr/share/fonts/adobe-source-code-pro
```

3.2 字体修改

字体修改可参考 CSS3@font-face，修改 /themes/even/src/css/_custom/_custom.scss 的字体变量列表，具体不再列出。

执行 npm install，npm run build 会报错 Unexpected character '' (1:0)，可通过如下办法解决：

```
$ vim /themes/even_m/src/webpack.config.js
// 添加如下语句
{
  test: /OpenSans-BoldItalic\.(woff|woff2|eot|ttf|otf|svg)$/,
  use: 'file-loader?name=[path][name].[ext]'
}
```

7.0.4 4. 网站图标修改

修改网站的图表,可修改 `themes/even/static/` 文件夹下的几个图片。也可在根目录 `static/` 中放置此类文件进行覆盖，优先级高于主题模版文件。

- android-chrome-192x192.png
- android-chrome-512x512.png
- apple-touch-icon.png
- browserconfig.xml
- favicon.ico
- favicon-16x16.png
- favicon-32x32.png
- manifest.json
- mstile-150x150.png
- safari-pinned-tab.svg

7.0.5 5. Hugo 站点说明

参考博客

- `archetypes`： 存放 default.md，头文件格式
- `content`： content 目录存放博客文章（.markdown/.md 文件）
- `data`： 存放自定义模版，导入的 toml 文件（或 json，yaml）
- `layouts`： layouts 目录存放的是网站的模板文件
- `static`： static 目录存放 js/css/img 等静态资源
- `config.toml`： config.toml 是网站的配置文件

第八章　### 二级域名绑定 GitHub Pages

申请到顶级域名后，可以将二级域名和 GitHub Pages 的项目仓库进行绑定，做到分仓库管理网站。这样可以在不修改主站的情况下对不同工作进行分类汇总。

8.0.1　1. DNS 二级域名设置

在 GoDaddy 域名管理中修改 DNS 如下：

类型	名称	值	TTL
A	@	192.30.252.153	1 小时
CNAME	www	@	1 小时
CNAME	blog	@	1 小时
…			

8.0.2　2. 绑定 GitHub Pages

在二级域名设置好后，在 GitHub 页面上传建成的网站，这里可以使用三种方式放置页面，`master`、`gh-pages`、`master/docs`，推荐使用第二种方式。

项目仓库根目录下放置一 `CNAME` 文件，里面写二级域名，例如 `blog.baiqp.info`，然后在仓库设置页面调整好 GitHub Pages 的设置。等待一定时间提醒配置好后进入页面。

这里列出常用的几个二级链接：

1. blog
2. music
3. wiki

2.2 项目撰写注意事项

因为这里使用的 GitHub Pages 进行发布，需要符合标准，请仔细阅读错误帮助手册，完整的 GitHub 帮助手册请参考 Help github。格式有错误会导致发布报错。

- Markdown 语法参考[01]
- 图片引用的名称和 `images/` 下的命名保持一致；
- 表格的格式要注意，具体格式如下：使用 `Markdown` 画的表格，如下表（**冒号表示对齐**）

col. 1	col. 2	col. 3
col 3 is	right-aligned	$1600
col 2 is	centered	$12
zebra stripes	are neat	$1

或

col. 1	col. 2	col. 3
col 3 is	right-aligned	$1600
col 2 is	centered	$12
zebra stripes	are neat	$1

- 在上传到 GitHub Pages 项目 `gh-pages` 时，可以先清空远程分支，再上传（效果还要验证）；

8.0.3　参考资料

1. Even Markdown 帮助