解析fsl_motion_outliers脚本

Alex / 2018-03-17 / free_learner@163.com / learning-archive.org

更新于2023-07-06，主要是文字排版上的更新，内容基本保持不变。

`fsl_motion_outliers`是FSL提供的一个进行头动校正、计算头动指标（dvars, fd, refrms等）并检测离群值（outlier）的Bash脚本，我通过学习这个脚本，试图了解如何写好脚本以及如何计算常见头动参数。

一、Bash特殊变量

在Bash shell里，字符`$1, $2, $3...$9`有特殊意义，分别表示向脚本传递的第1到第9个参数。假设我有一个名为`my_script.sh`的脚本，内容如下：

```
#! /bin/bash
echo "my name is $1"
echo "my age is $2"
echo "I would like to say $3"
```

脚本里的`$1, $2, $3`就是三个位置参数，可以在运行脚本时对其赋值，比如我以下面的方式运行脚本

```
./my_script.sh Alex 28 "Hello World"
```

在终端里会输出：

```
my name is Alex
my age is 28
I would like to say Hello World
```

`Alex, 28, "Hello World"`就是传递给脚本的三个参数，参数之间用空格分隔，如果参数本身有空格，参数需要加引号。同样地，`fsl_motion_outliers`脚本也是通过这样的方式接受参数。另外，变量`$0`表示脚本本身，`$#`表示所有传递的参数的个数，比如上面的例子中`$#`等于3.

二、解析参数

在命令行里运行`fsl_motion_outliers`（我测试的FSL版本是5.0.10），可以看到这个脚本有15个选项，其中输入（`-i`）和输出（`-o`）是必选项，剩下13个选项是可选项；选项也分为带参数

和不带参数，比如`-i`选项后面需接参数，`--refrms`选项后不接参数；另外，从选项形式可分为短形式，如`-m`选项，和长形式，如`--dvars`选项，短形式的选项后的参数用空格分隔，长形式的选项后的参数用等号分隔。

`fsl_motion_outliers`脚本首先解析用户指定了哪些选项和参数，这个功能通过三个自定义的函数实现，函数名分别为`get_opt1`，`get_arg1`和`get_arg2`，功能分别是提取长形式选项的选项名、提取长形式选项的参数和提取短形式选项的参数。具体地，

长形式选项后用等号连接参数，通过sed命令去掉等号及等号后的内容，即提取选项名。比如`--thresh=0.5`，这个函数就提取`--thresh`这个部分。

```
get_opt1() {
    arg=`echo $1 | sed 's/=.*//'`
    echo $arg
}
```

长形式选项后用等号连接参数，首先使用grep命令检查是否有等号，如果没有等号，提示该选项需要参数；如果有等号，使用sed命令去掉等号及等号以前的字符，如果等号后没有其他字符，则提示该选项需要参数；如果等号后有其他字符，返回参数变量`arg`。

```
get_arg1() {
    if [ X`echo $1 | grep '='` = X ] ; then
        echo "Option $1 requires an argument" 1>&2
        exit 1
    else
        arg=`echo $1 | sed 's/.*=//'`
        if [ X$arg = X ] ; then
            echo "Option $1 requires an argument" 1>&2
            exit 1
        fi
        echo $arg
    fi
}
```

短形式选项后用空格连接参数，直接检查该参数是否为空；参数不为空，返回参数变量`$2`。

```
get_arg2() {
    if [ X$2 = X ] ; then
        echo "Option $1 requires an argument" 1>&2
        exit 1
    fi
    echo $2
}
```

`fsl_motion_outliers`脚本采用while循环和case条件语句逐个解析选项和参数：

以下截取了部分原始代码来说明`fsl_motion_outliers`脚本如何实现逐个提取选项和参数。当参数个数大于或等于1，则选项还没有解析完；使用`get_opt1`函数提取选项，使用case条件语句来判断该选项是否在给定选项中，如果在给定选项中，则进一步提取该选项的参数；`shift 2`的作用是把位置参数向左移动两个位置，这样`$3`就变成了`$1`，通过这样的方式来处理已经解析过的选项和参数。

```
while [ $# -ge 1 ]
do
  iarg=`get_opt1 $1`
  case "$iarg" in
       -i)
          mcf=`get_arg2 $1 $2`;
          shift 2;;
       --dummy)
          ndel=`get_arg1 $1`;
          shift;;
       *)
          echo “Unrecognised option $1” 1>$2
          exit 1
          shift;;
  esac
done
```

三、头动校正和mask

如果计算fd（framewise displacement），需要先进行头动校正：

```
mcflirt -in $mcf -out fmri_mcf -mats -plots -refvol $refnum -rmsrel -rmsabs
```

其中mcflirt是FSL进行头动校正的命令，`-in`和`-out`分别表示输入和输出图像，即校正前后的图像，`-mats`和`-plots`分别表示转换矩阵和参数，`-refvol`表示以某一个图像为基准进行校

正，`-rmsrel`和`-rmsabs`不清楚是什么意思（没有找到介绍资料）。

由于计算dvars和refrms需要将计算范围限制在脑内，需要提供mask；如果没有设置mask选项，脚本会自己估计一个mask，方法是计算图像信号值的2和98百分位数，以这个区间的10%为阈值。

```
thr2=`fslstats $mcf -P 2`;
thr98=`fslstats $mcf -P 98`;
robthr=`echo "$thr2 + 0.1 * ( $thr98 - $thr2 )" | bc -l`;
fslmaths $mcf -Tmean -thr $robthr -bin $mask
```

四、头动指标

1. dvars

dvars的原理就是计算前后两个图像之间每个体素的信号差异，这个信号差异取平方（避免负值的影响），再求mask内部的均值，最后这个均值的平方根就是dvars；在下面的代码中，由于`-Xmean`，`-Ymean`，`-Zmean`是在整个图像（而非mask内部）求均值，变量`$maskmean`的作用就是补偿体素总数的变化，校正这个均值；变量`$brainmed`是图像的中位数，由于BOLD信号本质上是无量纲的，将中位数调整到1000，使得不同站点采集的数据具有可比性。

```
tmax=`fslval ${mcf} dim4`;
tmax1=`echo $tmax - 1 | bc`;
fslroi $mcf ${mcf}1 0 $tmax1
fslroi $mcf ${mcf}2 1 $tmax1
brainmed=`fslstats ${mcf} -k ${mask} -P 50`;
maskmean=`fslstats ${mask} -m`;
sqrtcom="-sqrt"
fslmaths ${mcf}2 -sub ${mcf}1 -mas ${mask} -sqr -Xmean -Ymean -Zmean \
         -div $maskmean $sqrtcom res_mse_diff -odt float
fslmaths res_mse_diff -div $brainmed -mul 1000 res_mse_diff
```

2. refrms

refrms的计算原理是选取一个图像做参考点，计算其他每个图像与参考图像在每个体素上的差异，平方后求每个图像的均值，这个均值在前后两个图像之间的变化的绝对值即为refrms；注意不同于dvars，调整图像的中位数是在平方以前做的，至于为什么这样做我还不清楚，我自己的想法是完全可以在计算任何参数之前，先调整中位数。

```
fslroi $mcf exf $refnum 1
fslmaths $mcf -sub $exf -mas ${mask} -div $brainmed -sqr -Xmean -Ymean -Zmean \
         -div $maskmean $sqrtcom res_mse -odt float
fslroi res_mse res_mse0 0 1 0 1 0 1 0 $tmax1
fslroi res_mse res_mse1 0 1 0 1 0 1 1 $tmax1
fslmaths res_mse1 -sub res_mse0 -abs res_mse_diff
```

3. fd

```
## fmri_mcf.par是头动校正后生成的包含6个头动参数的文本文件（T*6的矩阵，T表示时间点个数），将文本文件转换成nifti的格式（1*1*T*6的四维数组）。
fslascii2img fmri_mcf.par 1 1 $tmax 6 1 1 1 1 res_mse_par
## 提取前三个时间点的数据，即三个转动参数；提取后三个时间点的数据，即三个平移参数。
fslroi res_mse_par res_mse_par_rot_full 0 3
fslroi res_mse_par res_mse_par_trans_full 3 3
## 计算6个参数前后两个点之间的差异。
fslroi res_mse_par_rot_full res_mse_par_rot0 0 1 0 1 0 $tmax1
fslroi res_mse_par_rot_full res_mse_par_rot1 0 1 0 1 1 $tmax1
fslmaths res_mse_par_rot1 -sub res_mse_par_rot0 res_mse_par_rot
fslroi res_mse_par_trans_full res_mse_par_trans0 0 1 0 1 0 $tmax1
fslroi res_mse_par_trans_full res_mse_par_trans1 0 1 0 1 1 $tmax1
fslmaths res_mse_par_trans1 -sub res_mse_par_trans0 res_mse_par_trans
## 三个转动参数的差异取绝对值乘上50，将转动参数转换为平移距离，假设大脑半径为50mm；三个平移参数的差异取绝对值；将6个参数的差异绝对值相加即为fd。
fslmaths res_mse_par_rot -abs -mul 50 res_mse_par_rot
fslmaths res_mse_par_rot -Tmean -mul 3 res_mse_par_rotsum
fslmaths res_mse_par_trans -abs -Tmean -mul 3 res_mse_par_transsum
fslmaths res_mse_par_transsum -add res_mse_par_rotsum res_mse_diffZ
## 上面生成的fd文件是1*1*(T-1)*1的四维数组，现在转换为1*1*1*(T-1)的四维数组。
fsl2ascii res_mse_diffZ res_mse_diff.txt
grep [0-9] res_mse_diff.txt0* > res_mse_diff.txt
fslascii2img res_mse_diff.txt 1 1 1 $tmax1 1 1 1 1 res_mse_diff
```

dvars和fd的参考文献：Power, J. D., Barnes, K. A., Snyder, A. Z., Schlaggar, B. L., & Petersen, S. E. (2012). Spurious but systematic correlations in functional connectivity MRI networks arise from subject motion. Neuroimage, 59(3), 2142-2154.

五、保存和作图

```
## 将nifti格式转换为文本文件，同时用fsl_tsplot画出头动指标的图。
fsl2ascii res_mse_diff vals.txt
echo "0" > ${savefile}
cat vals.txt[0-9]* | grep [0-9] >> ${savefile}
rm vals.txt[0-9]*
fsl_tsplot -i vals.txt -t "Motion outlier metric: $metric" -x "time" -y "metric value" -o
${plotfile}
```

六、检测离群值并生成矩阵

```
## 如果没有设置离群值的阈值，默认采用高于1.5倍IQR作为阈值。
pv=`fslstats res_mse_diff -p 25 -p 75`;
p25=`echo $pv | awk '{ print $1 }'`;
p75=`echo $pv | awk '{ print $2 }'`;
threshv=`echo "$p75 + 1.5 * ( $p75 - $p25 )" | bc -l`
## 上述头动指标都是计算的前后时间点之间的差异，第一个时间点的头动即为0，所以需要在前面添加一个
0；计算高于阈值的时间点个数。
fslmaths res_mse_diff -thr $threshv -bin outliers
fslroi outliers one 0 1 0 1 0 1 0 1
fslmaths one -mul 0 zero
fslmerge -t outliers zero outliers
nmax=`fslstats outliers -V | awk '{ print $1 }'`
## 生成离群值的矩阵，每个离群值一列，每列长度为T，在离群值出现的时间上为1，其余时间点为0；这个
矩阵就表明了在那几个时间上出现了较大的头动，该矩阵可用于线性回归方程去除头动对信号带来的影响。
注意这里fslmaths中-roi选项的作用是将某个时间点前后的时间点都赋值为0。
n=0;
while [ $n -lt $tmax ] ; do
fslmaths outliers -roi 0 1 0 1 0 1 $n 1 stp
val=`fslstats stp -V | awk '{ print $1 }'`;
if [ $val -gt 0 ] ; then
      fslmeants -i stp -o singleev;
      if [ -f $outfile ] ; then
          paste -d ' ' $outfile singleev > ${outfile}2
          cp ${outfile}2 $outfile
          rm -f ${outfile}2
      else
          cp ${outdir}_mc/singleev $outfile
      fi
  fi
  n=`echo "$n + 1" | bc`;
done
```

七、总结

通过学习`fsl_motion_outliers`脚本，我了解到如何自己定义函数解析参数以及如何使用FSL内置命令计算头动指标。理解上肯定有不少谬误之处，欢迎指正。