解析cat_batch_cat.sh脚本

Alex / 2021-03-21 / free_learner@163.com / AlexBrain.cn

更新于2023-09-13，主要是文字排版上的更新，内容基本保持不变。

`cat_batch_cat.sh`是CAT12工具包提供的一个通过命令行调用CAT12分析数据的Bash脚本，学习一下这个脚本。整个脚本包含 8 个函数，下面分别说明每个函数的目的和实现方法。

一、main函数

这个函数的内容就是顺序执行另外 5 个函数，即这个函数的目的就是把其他函数串起来。

```
main ()
{
  ## 解析参数，$@表示所有参数，${1+“$@”}这种用法我第一次见到，查了很久也没有找到这种用法的含义。
  parse_args ${1+"$@"}
  ## 检查matlab是否存在
  check_matlab
  ## 检查输入文件是否存在
  check_files
  ## 查看CPU的数量并设置任务数
  get_no_of_cpus
  ## 调用CAT12的函数进行具体的数据分析，比如Segment
  run_vbm
  exit 0
}
```

二、parse_args函数

`parse_args`函数的目的是解析用户输入的选项名字和内容。

```
parse_args ()
{
  ## 定义两个局部变量来存放当前选项名称和选项内容。
  local optname optarg
  ## 这个变量用来存放总的输入文件（即T1图像）的数量。
  count=0
  ## 这个变量用来存放解析得到的所有选项名称和选项内容。
  ## 对于Bash脚本，这里所谓选项名称和内容都是用位置参数来传递。
  paras=

  ## 如果参数总数目（$#）小于１，运行help函数并退出脚本。
  if [ $# -lt 1 ]; then
    ## help函数的作用是输出脚本用法和选项。
    help
    exit 1
  fi

  ## 如果参数数目大于０，则还有选项没有解析。
  while [ $# -gt 0 ]; do
    ## 如果第一个变量有等号，则保留等号前面的内容。
    ## 一般sed s/str1/str2/，这里逗号相当于/。
    optname="`echo $1 | sed 's,=.*,,'`"
    ## 如果变量里有等号，则保留等号后面的内容。
    optarg="`echo $2 | sed 's,^[^=]*=,,'`"
    ## 这里的假设似乎是第一个位置变量（$1）是选项名称，
    ## 第二个位置变量（$2）是选项内容，
    ## 但是实际上有些参数只有内容，没有名称，比如输入文件，后面看看它是如何处理的。
    paras="$paras $optname $optarg"

    case "$1" in
        --matlab* | -m*)
            ## 检查选项内容即变量optarg是否为空。
            exit_if_empty "$optname" "$optarg"
            matlab=$optarg
            ## 改变位置参数，比如$2变为$1，注意到这里只移动了一个位置，
            ## 配合esc后面的shift，则移动了两个位置，也就是完成了一对参数（名称＋内容）的
解析。
            shift
            ;;
...
        --f* | -f*)
            ## 输入文件可以放在一个文本文件中。
            exit_if_empty "$optname" "$optarg"
            listfile=$optarg
            shift
            list=$(< $listfile);
            ## 读取文本文件中每行的内容作为输入文件。
            for F in $list; do
              ARRAY[$count]=$F
              ((count++))
```

```
            done
            ;;
...
        -h | --help | -v | --version | -V)
            ## 如果是-h等选项，则输出帮助内容并退出。
            help
            exit 1
            ;;
        -*)
            ## 如果不能匹配以上选项名称的-开头的选项则认为是无法识别的选项。
            echo "`basename $0`: ERROR: Unrecognized option \"$1\"" >&2
            ;;
        *)
            ## 如果不能匹配以上选项名称的则认为是输入文件，注意这里没有shift，
            ## 因为这个选项只有内容没有名称。之所以没有设置选项名称，
            ## 我猜测是因为方便处理输入是多个文件的情况。
            ARRAY[$count]=$1
            ((count++))
            ;;
    esac
    shift
  done
}
```

三、help函数

这个函数的目的是输出该脚本的用法和可选项。

```
help ()
{
cat <<__EOM__

USAGE:
   cat_batch_cat.sh filename|filepattern [-m matlab_command] [-w] [-p
number_of_processes] [-d default_file] [-l log_folder]

   -n      nice level
...

This script was written by Christian Gaser (christian.gaser@uni-jena.de).

__EOM__
}
```

四、exit_if_empty函数

这个函数的目的就是检查选项内容是否为空。

```
exit_if_empty ()
{
  local desc val

  desc="$1"
  shift
  val="$*"

  if [ -z "$val" ]; then
    echo 'ERROR: No argument given with \"$desc\" command line argument!' >&2
    exit 1
  fi
}
```

五、check_files函数

这个函数的目的是检查输入文件是否存在。

```
check_files ()
{

  SIZE_OF_ARRAY="${#ARRAY[@]}"
  ## 如果输入文件数目为0，则退出。
  if [ "$SIZE_OF_ARRAY" -eq 0 ]; then
      echo 'ERROR: No files given!' >&2
      help
      exit 1
  fi

  i=0
  ## 循环检查每个输入文件是否存在或者是否是链接文件。
  while [ "$i" -lt "$SIZE_OF_ARRAY" ]; do
    if [ ! -f "${ARRAY[$i]}" ]; then
      if [ ! -L "${ARRAY[$i]}" ]; then
        echo ERROR: File ${ARRAY[$i]} not found
        help
        exit 1
      fi
    fi
    ((i++))
  done
}
```

六、get_no_of_cpus函数

这个函数的目的是查看电脑CPU的数目并设置任务数。

```
get_no_of_cpus () {

  if [ "$ARCH" == "Linux" ]; then
    NUMBER_OF_PROC=`grep ^processor $CPUINFO | wc -l`
  elif [ "$ARCH" == "Darwin" ]; then
    NUMBER_OF_PROC=`sysctl -a hw | grep -w logicalcpu | awk '{ print $2 }'`
  elif [ "$ARCH" == "FreeBSD" ]; then
    NUMBER_OF_PROC=`sysctl hw.ncpu | awk '{ print $2 }'`
  else
    NUMBER_OF_PROC=`grep ^processor $CPUINFO | wc -l`
  fi

  ## 如果没有获得CPU的数目，则退出脚本。
  if [ -z "$NUMBER_OF_PROC" ]; then
      echo "$FUNCNAME ERROR - number of CPUs not obtained. Use -p to define
number of processes."
      exit 1
  fi

  ##  如果没有设置任务数，则任务数为CPU的数目。
  # use all processors if not other defined
  if [ "$NUMBER_OF_JOBS" == "" ]; then
      NUMBER_OF_JOBS=$NUMBER_OF_PROC
  fi

  ## 如果任务数小于-1，则任务数为CPU的数目加上设置的任务数。
  if [ $NUMBER_OF_JOBS -le -1 ]; then
    NUMBER_OF_JOBS=$(echo "$NUMBER_OF_PROC + $NUMBER_OF_JOBS" | bc)
    ## 如果任务数小于1，则任务数设置为１。
    if [ "$NUMBER_OF_JOBS" -lt 1 ]; then
        NUMBER_OF_JOBS=1
    fi
  fi
  ##  如果任务数大于CPU的数目，则任务数为CPU的数目。
  if [ "$NUMBER_OF_JOBS" -gt "$NUMBER_OF_PROC" ]; then
      NUMBER_OF_JOBS=$NUMBER_OF_PROC
  fi
  echo "Found $NUMBER_OF_PROC processors. Use $NUMBER_OF_JOBS."
  echo
}
```

七、check_matlab函数

这个函数检查matlab是否存在。

```
check_matlab ()
{
  found=`which ${matlab} 2>/dev/null`
  if [ ! -n "$found" ]; then
    echo $matlab not found.
    exit 1
  fi
}
```

八、run_vbm函数

这个函数调用CAT12的函数对输入图像进行一些处理。

```
run_vbm ()
{
    ## 这个脚本默认的位置在CAT12的安装目录。
    cwd=`dirname $0`
    pwd=$PWD

    # we have to go into the toolbox folder to find matlab files
    cd $cwd

    spm12=`dirname $cwd`
    spm12=`dirname $spm12`

    ## 如果没有设置LOGDIR，则设置为输入文件所在目录。
    if [ "${LOGDIR}" == "" ]; then
        LOGDIR=`dirname ${ARRAY[0]}`
    fi

    export MATLABPATH=$spm12

    ## 根据输入文件的数目和任务数将输入文件分块，之所以乘以10000，应该是保证商不为0。
    SIZE_OF_ARRAY="${#ARRAY[@]}"
    BLOCK=$((10000* $SIZE_OF_ARRAY / $NUMBER_OF_JOBS ))

    # argument empty?
    if [ ! "${defaults_file}" == "" ]; then
        # check wether absolute or relative names were given
        if [ ! -f ${defaults_file} -a -f ${pwd}/${defaults_file} ]; then
            defaults_file=${pwd}/${defaults_file}
        fi

        # check whether defaults file exist
        if [ ! -f ${defaults_file} ];  then
            echo Default file $defaults_file not found.
            exit
        fi
    fi

    ## $$表示进程ID，这样可以确保文件名是独一无二的。
    # split files and prepare tmp-file with filenames
    TMP=/tmp/cat_$$
    i=0
    while [ "$i" -lt "$SIZE_OF_ARRAY" ]; do
        ## 如果商为0，则表示第一个block还有空余。
        count=$((10000* $i / $BLOCK ))

        # check wether absolute or relative names were given
        if [ ! -f ${ARRAY[$i]} ];  then
            if [ -f ${pwd}/${ARRAY[$i]} ]; then
                FILE=${pwd}/${ARRAY[$i]}
            fi
```

```
        else
            FILE=${ARRAY[$i]}
        fi

        ## 把属于同一个block的输入文件存放到向量的一个元素里，索引就是block。
        if [ -z "${ARG_LIST[$count]}" ]; then
            ARG_LIST[$count]="$FILE"
        else
            ARG_LIST[$count]="${ARG_LIST[$count]} $FILE"
        fi

        ## 将属于同一个block的输入文件存放到一个文本文件里。
        echo ${FILE} >> ${TMP}${count}
        FDIR=$(dirname $FILE)
        ((i++))
    done

    ## 存放输出信息的文本文件。
    vbmlog=${LOGDIR}/cat_${HOSTNAME}_${time}

    i=0
    while [ "$i" -lt "$NUMBER_OF_JOBS" ]; do
        if [ ! "${ARG_LIST[$i]}" == "" ]; then
            j=$(($i+1))
            ## 如果matlabcommand为空，则调用cat_batch_cat进行CAT12的Segment的模块的
分析。
            if [ -z "$matlabcommand" ]; then
              COMMAND="cat_batch_cat('${TMP}${i}','${defaults_file}')"
            else
              CFILES=""
              for F in ${ARG_LIST[$i]} ; do
                ## 将属于同一个block的输入文件用分号分隔。
                CFILES=$CFILES";"\'$F\';
              done
              ##  去掉第一个多余的分号。
              CFILES=$(echo $CFILES | cut -c 2-);
              ## 构造一个Matlab的cell变量。
              CFILES="{"$CFILES"}";
              ## 将CFILES替换为实际的输入文件。
              matlabcommand2=$matlabcommand
              matlabcommand2=$(echo $matlabcommand2 |sed 's/CFILES/$CFILES/g');
              eval "COMMAND=\"$matlabcommand2\";" # fprintf('ERROR');
e=lasterror; e.message,
              ## 如果调用CAT12的一些函数处理数据。
              COMMAND="try, spm; spm_get_defaults; cat_get_defaults; global
defaults vbm matlabbatch; $COMMAND;
catch caterr, sprintf('\n%s\nVBM Preprocessing error: %s:\n%s\n', repmat('-
',1,72),caterr.identifier,caterr.message,repmat('-',1,72)); for
si=1:numel(caterr.stack), cat_io_cprintf('err',sprintf('%5d -
%s\n',caterr.stack(si).line,caterr.stack(si).name)); end;
cat_io_cprintf('err',sprintf('%s\\n',repmat('-',1,72))); exit; end;
```

```
fprintf('VBM batch processing done.'); exit;";
            fi
            ## 对输入文件进行一些其他操作。
            SHCOMMAND="$shellcommand ${ARG_LIST[$i]}"

            echo Calculate
            for F in ${ARG_LIST[$i]}; do echo $F; done
            # File Output
            echo ---------------------------------- >> ${vbmlog}_${j}.log
            date                                    >> ${vbmlog}_${j}.log
            echo ---------------------------------- >> ${vbmlog}_${j}.log
            echo                                    >> ${vbmlog}_${j}.log
            echo Calling string of this batch:      >> ${vbmlog}_${j}.log
            echo "  $0 $paras"                      >> ${vbmlog}_${j}.log
            echo                                    >> ${vbmlog}_${j}.log
            echo MATLAB command of this batch:      >> ${vbmlog}_${j}.log
            echo "  $COMMAND"                       >> ${vbmlog}_${j}.log
            echo                                    >> ${vbmlog}_${j}.log
            echo Shell command of this batch:       >> ${vbmlog}_${j}.log
            echo "  $SHCOMMAND"                     >> ${vbmlog}_${j}.log
            echo                                    >> ${vbmlog}_${j}.log
            ## shellcommand和COMMAND不能同时执行。
            if [ -z "$shellcommand" ]; then
              # do nohup in background or not
              if [ -z "$fg" ]; then
                nohup nice -n $nicelevel ${matlab} -nodisplay "$nojvm" -
nosplash -r "$COMMAND" >> ${vbmlog}_${j}.log 2>&1 &
              else
                nohup nice -n $nicelevel ${matlab} -nodisplay "$nojvm" -
nosplash -r "$COMMAND" >> ${vbmlog}_${j}.log 2>&1
              fi
            else
              # do nohup in background or not
              if [ -z "$fg" ]; then
                nohup nice -n $nicelevel $SHCOMMAND >> ${vbmlog}_${j}.log 2>&1
&
              else
                nohup nice -n $nicelevel $SHCOMMAND >> ${vbmlog}_${j}.log 2>&1
              fi
            fi
            echo Check ${vbmlog}_${j}.log for logging information
            echo
        fi
        ((i++))
    done

    exit 0
}
```

九、小结

通过学习这个脚本，了解了如何在命令行调用CAT12分析数据以及一些编程小技巧（解析参数、多任务并行计算等）。