#!/bin/tcsh -xef

echo "auto-generated by afni\_proc.py, Thu Sep 11 10:45:12 2014"

echo "(version 4.21, September 8, 2014)"

echo "execution started: `date`"

# execute via :

# tcsh -xef s1305.process2mm.script |& tee output.s1305.process2mm.script

# =========================== auto block: setup ============================

# script setup

# take note of the AFNI version

afni -ver

# check that the current AFNI version is recent enough

afni\_history -check\_date 13 May 2014

if ( $status ) then

echo "\*\* this script requires newer AFNI binaries (than 13 May 2014)"

echo " (consider: @update.afni.binaries -defaults)"

exit

endif

# the user may specify a single subject to run with

if ( $#argv > 0 ) then

set subj = $argv[1]

else

set subj = s1305

endif

# assign output directory name

set output\_dir = s1305.results2mm

# verify that the results directory does not yet exist

if ( -d $output\_dir ) then

echo output dir "$subj.results" already exists

exit

endif

# set list of runs

set runs = (`count -digits 2 1 2`)

# create results and stimuli directories

mkdir $output\_dir

mkdir $output\_dir/stimuli

# copy stim files into stimulus directory

cp /raid4/sdanny/workingdata/tau/behav\_data/onsets/s1305\_visit1\_AC\_onsets.txt \

/raid4/sdanny/workingdata/tau/behav\_data/onsets/s1305\_visit1\_AI\_onsets.txt \

/raid4/sdanny/workingdata/tau/behav\_data/onsets/s1305\_visit1\_Incorrect\_onsets.txt \

/raid4/sdanny/workingdata/tau/behav\_data/onsets/s1305\_visit1\_N\_onsets.txt \

$output\_dir/stimuli

# copy anatomy to results dir

3dcopy /raid4/sdanny/workingdata/tau/subjects/s1305/anat/s1305\_anat+orig \

$output\_dir/s1305\_anat

# copy over the external volreg base

3dbucket -prefix $output\_dir/external\_volreg\_base \

'/raid4/sdanny/workingdata/tau/subjects/s1305/afni/OutBrick\_r1+orig[1]'

# ============================ auto block: tcat ============================

# apply 3dTcat to copy input dsets to results dir, while

# removing the first 4 TRs

3dTcat -prefix $output\_dir/pb00.$subj.r01.tcat \

/raid4/sdanny/workingdata/tau/subjects/s1305/afni/OutBrick\_r1+orig'[4..$]'

3dTcat -prefix $output\_dir/pb00.$subj.r02.tcat \

/raid4/sdanny/workingdata/tau/subjects/s1305/afni/OutBrick\_r2+orig'[4..$]'

# and make note of repetitions (TRs) per run

set tr\_counts = ( 180 180 )

# -------------------------------------------------------

# enter the results directory (can begin processing data)

cd $output\_dir

# ========================== auto block: outcount ==========================

# data check: compute outlier fraction for each volume

touch out.pre\_ss\_warn.txt

foreach run ( $runs )

3dToutcount -automask -fraction -polort 3 -legendre \

pb00.$subj.r$run.tcat+orig > outcount.r$run.1D

# censor outlier TRs per run, ignoring the first 0 TRs

# - censor when more than 0.1 of automask voxels are outliers

# - step() defines which TRs to remove via censoring

1deval -a outcount.r$run.1D -expr "1-step(a-0.1)" > rm.out.cen.r$run.1D

# outliers at TR 0 might suggest pre-steady state TRs

if ( `1deval -a outcount.r$run.1D"{0}" -expr "step(a-0.4)"` ) then

echo "\*\* TR #0 outliers: possible pre-steady state TRs in run $run" \

>> out.pre\_ss\_warn.txt

endif

end

# catenate outlier counts into a single time series

cat outcount.r\*.1D > outcount\_rall.1D

# catenate outlier censor files into a single time series

cat rm.out.cen.r\*.1D > outcount\_${subj}\_censor.1D

# ================================= tshift =================================

# time shift data so all slice timing is the same

foreach run ( $runs )

3dTshift -tzero 0 -quintic -prefix pb01.$subj.r$run.tshift \

pb00.$subj.r$run.tcat+orig

end

# ================================= align ==================================

# for e2a: compute anat alignment transformation to EPI registration base

# (new anat will be intermediate, stripped, s1305\_anat\_ns+orig)

align\_epi\_anat.py -anat2epi -anat s1305\_anat+orig \

-save\_skullstrip -suffix \_al\_junk \

-epi external\_volreg\_base+orig -epi\_base 0 \

-epi\_strip 3dAutomask \

-giant\_move -cost lpc+ZZ -AddEdge \

-volreg off -tshift off

# ================================== tlrc ==================================

# warp anatomy to standard space

@auto\_tlrc -base TT\_N27+tlrc -input s1305\_anat\_ns+orig -no\_ss

# ================================= volreg =================================

# align each dset to base volume, align to anat, warp to tlrc space

# verify that we have a +tlrc warp dataset

if ( ! -f s1305\_anat\_ns+tlrc.HEAD ) then

echo "\*\* missing +tlrc warp dataset: s1305\_anat\_ns+tlrc.HEAD"

exit

endif

# register and warp

foreach run ( $runs )

# register each volume to the base

3dvolreg -verbose -zpad 1 -base external\_volreg\_base+orig \

-1Dfile dfile.r$run.1D -prefix rm.epi.volreg.r$run \

-cubic \

-1Dmatrix\_save mat.r$run.vr.aff12.1D \

pb01.$subj.r$run.tshift+orig

# create an all-1 dataset to mask the extents of the warp

3dcalc -overwrite -a pb01.$subj.r$run.tshift+orig -expr 1 \

-prefix rm.epi.all1

# catenate volreg, epi2anat and tlrc transformations

cat\_matvec -ONELINE \

s1305\_anat\_ns+tlrc::WARP\_DATA -I \

s1305\_anat\_al\_junk\_mat.aff12.1D -I \

mat.r$run.vr.aff12.1D > mat.r$run.warp.aff12.1D

# apply catenated xform : volreg, epi2anat and tlrc

3dAllineate -base s1305\_anat\_ns+tlrc \

-input pb01.$subj.r$run.tshift+orig \

-1Dmatrix\_apply mat.r$run.warp.aff12.1D \

-mast\_dxyz 2.5 \

-prefix rm.epi.nomask.r$run

# warp the all-1 dataset for extents masking

3dAllineate -base s1305\_anat\_ns+tlrc \

-input rm.epi.all1+orig \

-1Dmatrix\_apply mat.r$run.warp.aff12.1D \

-mast\_dxyz 2.5 -final NN -quiet \

-prefix rm.epi.1.r$run

# make an extents intersection mask of this run

3dTstat -min -prefix rm.epi.min.r$run rm.epi.1.r$run+tlrc

# if there was an error, exit so user can see

if ( $status ) exit

end

# make a single file of registration params

cat dfile.r\*.1D > dfile\_rall.1D

# ----------------------------------------

# create the extents mask: mask\_epi\_extents+tlrc

# (this is a mask of voxels that have valid data at every TR)

3dMean -datum short -prefix rm.epi.mean rm.epi.min.r\*.HEAD

3dcalc -a rm.epi.mean+tlrc -expr 'step(a-0.999)' -prefix mask\_epi\_extents

# and apply the extents mask to the EPI data

# (delete any time series with missing data)

foreach run ( $runs )

3dcalc -a rm.epi.nomask.r$run+tlrc -b mask\_epi\_extents+tlrc \

-expr 'a\*b' -prefix pb02.$subj.r$run.volreg

end

# create an anat\_final dataset, aligned with stats

3dcopy s1305\_anat\_ns+tlrc anat\_final.$subj

# ================================== blur ==================================

# blur each volume of each run

foreach run ( $runs )

3dmerge -1blur\_fwhm 6 -doall -prefix pb03.$subj.r$run.blur \

pb02.$subj.r$run.volreg+tlrc

end

# ================================== mask ==================================

# create 'full\_mask' dataset (union mask)

foreach run ( $runs )

3dAutomask -dilate 1 -prefix rm.mask\_r$run pb03.$subj.r$run.blur+tlrc

end

# create union of inputs, output type is byte

3dmask\_tool -inputs rm.mask\_r\*+tlrc.HEAD -union -prefix full\_mask.$subj

# ---- create subject anatomy mask, mask\_anat.$subj+tlrc ----

# (resampled from tlrc anat)

3dresample -master full\_mask.$subj+tlrc -input s1305\_anat\_ns+tlrc \

-prefix rm.resam.anat

# convert to binary anat mask; fill gaps and holes

3dmask\_tool -dilate\_input 5 -5 -fill\_holes -input rm.resam.anat+tlrc \

-prefix mask\_anat.$subj

# compute overlaps between anat and EPI masks

3dABoverlap -no\_automask full\_mask.$subj+tlrc mask\_anat.$subj+tlrc \

|& tee out.mask\_ae\_overlap.txt

# note correlation as well

3ddot full\_mask.$subj+tlrc mask\_anat.$subj+tlrc |& tee out.mask\_ae\_corr.txt

# ---- create group anatomy mask, mask\_group+tlrc ----

# (resampled from tlrc base anat, TT\_N27+tlrc)

3dresample -master full\_mask.$subj+tlrc -prefix ./rm.resam.group \

-input /opt/AFNINEW/TT\_N27+tlrc

# convert to binary group mask; fill gaps and holes

3dmask\_tool -dilate\_input 5 -5 -fill\_holes -input rm.resam.group+tlrc \

-prefix mask\_group

# ================================= scale ==================================

# scale each voxel time series to have a mean of 100

# (be sure no negatives creep in)

# (subject to a range of [0,200])

foreach run ( $runs )

3dTstat -prefix rm.mean\_r$run pb03.$subj.r$run.blur+tlrc

3dcalc -a pb03.$subj.r$run.blur+tlrc -b rm.mean\_r$run+tlrc \

-c mask\_epi\_extents+tlrc \

-expr 'c \* min(200, a/b\*100)\*step(a)\*step(b)' \

-prefix pb04.$subj.r$run.scale

end

# ================================ regress =================================

# compute de-meaned motion parameters (for use in regression)

1d\_tool.py -infile dfile\_rall.1D -set\_nruns 2 \

-demean -write motion\_demean.1D

# compute motion parameter derivatives (just to have)

1d\_tool.py -infile dfile\_rall.1D -set\_nruns 2 \

-derivative -demean -write motion\_deriv.1D

# create censor file motion\_${subj}\_censor.1D, for censoring motion

1d\_tool.py -infile dfile\_rall.1D -set\_nruns 2 \

-show\_censor\_count -censor\_prev\_TR \

-censor\_motion 2 motion\_${subj}

# combine multiple censor files

1deval -a motion\_${subj}\_censor.1D -b outcount\_${subj}\_censor.1D \

-expr "a\*b" > censor\_${subj}\_combined\_2.1D

# run the regression analysis

3dDeconvolve -input pb04.$subj.r\*.scale+tlrc.HEAD \

-censor censor\_${subj}\_combined\_2.1D \

-polort 3 \

-local\_times \

-num\_stimts 10 \

-stim\_times 1 stimuli/s1305\_visit1\_AC\_onsets.txt 'GAM' \

-stim\_label 1 AC \

-stim\_times 2 stimuli/s1305\_visit1\_AI\_onsets.txt 'GAM' \

-stim\_label 2 AI \

-stim\_times 3 stimuli/s1305\_visit1\_Incorrect\_onsets.txt 'GAM' \

-stim\_label 3 Incorrect \

-stim\_times 4 stimuli/s1305\_visit1\_N\_onsets.txt 'GAM' \

-stim\_label 4 N \

-stim\_file 5 motion\_demean.1D'[0]' -stim\_base 5 -stim\_label 5 roll \

-stim\_file 6 motion\_demean.1D'[1]' -stim\_base 6 -stim\_label 6 pitch \

-stim\_file 7 motion\_demean.1D'[2]' -stim\_base 7 -stim\_label 7 yaw \

-stim\_file 8 motion\_demean.1D'[3]' -stim\_base 8 -stim\_label 8 dS \

-stim\_file 9 motion\_demean.1D'[4]' -stim\_base 9 -stim\_label 9 dL \

-stim\_file 10 motion\_demean.1D'[5]' -stim\_base 10 -stim\_label 10 dP \

-num\_glt 8 \

-gltsym 'SYM: +AI -AC' \

-glt\_label 1 ABias \

-gltsym 'SYM: +AC -N' \

-glt\_label 2 ACvN \

-gltsym 'SYM: +AI -N' \

-glt\_label 3 AIvN \

-gltsym 'SYM: +AI +AC' \

-glt\_label 4 AngryvFix \

-gltsym 'SYM: +AI +AC +N' \

-glt\_label 5 FacevFix \

-gltsym 'SYM: +3\*Incorrect -AI -AC -N' \

-glt\_label 6 ErrorvCorrect \

-gltsym 'SYM: +AI +AC -2\*N' \

-glt\_label 7 ACAIvNEU \

-gltsym 'SYM: +.5\*AI +.5\*AC' \

-glt\_label 8 Angry \

-cbucket cbucket.stats.s1305 \

-xsave \

-jobs 4 \

-fout -tout -x1D X.xmat.1D -xjpeg X.jpg \

-x1D\_uncensored X.nocensor.xmat.1D \

-fitts fitts.$subj \

-errts errts.${subj} \

-bucket stats.$subj

# if 3dDeconvolve fails, terminate the script

if ( $status != 0 ) then

echo '---------------------------------------'

echo '\*\* 3dDeconvolve error, failing...'

echo ' (consider the file 3dDeconvolve.err)'

exit

endif

# display any large pariwise correlations from the X-matrix

1d\_tool.py -show\_cormat\_warnings -infile X.xmat.1D |& tee out.cormat\_warn.txt

# -- execute the 3dREMLfit script, written by 3dDeconvolve --

tcsh -x stats.REML\_cmd

# if 3dREMLfit fails, terminate the script

if ( $status != 0 ) then

echo '---------------------------------------'

echo '\*\* 3dREMLfit error, failing...'

exit

endif

# create an all\_runs dataset to match the fitts, errts, etc.

3dTcat -prefix all\_runs.$subj pb04.$subj.r\*.scale+tlrc.HEAD

# --------------------------------------------------

# create a temporal signal to noise ratio dataset

# signal: if 'scale' block, mean should be 100

# noise : compute standard deviation of errts

3dTstat -mean -prefix rm.signal.all all\_runs.$subj+tlrc

3dTstat -stdev -prefix rm.noise.all errts.${subj}\_REML+tlrc

3dcalc -a rm.signal.all+tlrc \

-b rm.noise.all+tlrc \

-c full\_mask.$subj+tlrc \

-expr 'c\*a/b' -prefix TSNR.$subj

# ---------------------------------------------------

# compute and store GCOR (global correlation average)

# (sum of squares of global mean of unit errts)

3dTnorm -norm2 -prefix rm.errts.unit errts.${subj}\_REML+tlrc

3dmaskave -quiet -mask full\_mask.$subj+tlrc rm.errts.unit+tlrc > \

gmean.errts.unit.1D

3dTstat -sos -prefix - gmean.errts.unit.1D\' > out.gcor.1D

echo "-- GCOR = `cat out.gcor.1D`"

# create ideal files for fixed response stim types

1dcat X.nocensor.xmat.1D'[8]' > ideal\_AC.1D

1dcat X.nocensor.xmat.1D'[9]' > ideal\_AI.1D

1dcat X.nocensor.xmat.1D'[10]' > ideal\_Incorrect.1D

1dcat X.nocensor.xmat.1D'[11]' > ideal\_N.1D

# --------------------------------------------------------

# compute sum of non-baseline regressors from the X-matrix

# (use 1d\_tool.py to get list of regressor colums)

set reg\_cols = `1d\_tool.py -infile X.nocensor.xmat.1D -show\_indices\_interest`

3dTstat -sum -prefix sum\_ideal.1D X.nocensor.xmat.1D"[$reg\_cols]"

# also, create a stimulus-only X-matrix, for easy review

1dcat X.nocensor.xmat.1D"[$reg\_cols]" > X.stim.xmat.1D

# ============================ blur estimation =============================

# compute blur estimates

touch blur\_est.$subj.1D # start with empty file

# -- estimate blur for each run in epits --

touch blur.epits.1D

# restrict to uncensored TRs, per run

foreach run ( $runs )

set trs = `1d\_tool.py -infile X.xmat.1D -show\_trs\_uncensored encoded \

-show\_trs\_run $run`

if ( $trs == "" ) continue

3dFWHMx -detrend -mask full\_mask.$subj+tlrc \

all\_runs.$subj+tlrc"[$trs]" >> blur.epits.1D

end

# compute average blur and append

set blurs = ( `3dTstat -mean -prefix - blur.epits.1D\'` )

echo average epits blurs: $blurs

echo "$blurs # epits blur estimates" >> blur\_est.$subj.1D

# -- estimate blur for each run in errts --

touch blur.errts.1D

# restrict to uncensored TRs, per run

foreach run ( $runs )

set trs = `1d\_tool.py -infile X.xmat.1D -show\_trs\_uncensored encoded \

-show\_trs\_run $run`

if ( $trs == "" ) continue

3dFWHMx -detrend -mask full\_mask.$subj+tlrc \

errts.${subj}+tlrc"[$trs]" >> blur.errts.1D

end

# compute average blur and append

set blurs = ( `3dTstat -mean -prefix - blur.errts.1D\'` )

echo average errts blurs: $blurs

echo "$blurs # errts blur estimates" >> blur\_est.$subj.1D

# -- estimate blur for each run in err\_reml --

touch blur.err\_reml.1D

# restrict to uncensored TRs, per run

foreach run ( $runs )

set trs = `1d\_tool.py -infile X.xmat.1D -show\_trs\_uncensored encoded \

-show\_trs\_run $run`

if ( $trs == "" ) continue

3dFWHMx -detrend -mask full\_mask.$subj+tlrc \

errts.${subj}\_REML+tlrc"[$trs]" >> blur.err\_reml.1D

end

# compute average blur and append

set blurs = ( `3dTstat -mean -prefix - blur.err\_reml.1D\'` )

echo average err\_reml blurs: $blurs

echo "$blurs # err\_reml blur estimates" >> blur\_est.$subj.1D

# add 3dClustSim results as attributes to the stats dset

set fxyz = ( `tail -1 blur\_est.$subj.1D` )

3dClustSim -both -NN 123 -mask full\_mask.$subj+tlrc \

-fwhmxyz $fxyz[1-3] -prefix ClustSim

3drefit -atrstring AFNI\_CLUSTSIM\_MASK file:ClustSim.mask \

-atrstring AFNI\_CLUSTSIM\_NN1 file:ClustSim.NN1.niml \

-atrstring AFNI\_CLUSTSIM\_NN2 file:ClustSim.NN2.niml \

-atrstring AFNI\_CLUSTSIM\_NN3 file:ClustSim.NN3.niml \

stats.$subj+tlrc stats.${subj}\_REML+tlrc

# ================== auto block: generate review scripts ===================

# generate a review script for the unprocessed EPI data

gen\_epi\_review.py -script @epi\_review.$subj \

-dsets pb00.$subj.r\*.tcat+orig.HEAD

# generate scripts to review single subject results

# (try with defaults, but do not allow bad exit status)

gen\_ss\_review\_scripts.py -mot\_limit 2.0 -out\_limit 0.1 -exit0

# ========================== auto block: finalize ==========================

# remove temporary files

\rm -f rm.\*

# if the basic subject review script is here, run it

# (want this to be the last text output)

if ( -e @ss\_review\_basic ) ./@ss\_review\_basic |& tee out.ss\_review.$subj.txt

# return to parent directory

cd ..

echo "execution finished: `date`"

# ==========================================================================

# script generated by the command:

#

# afni\_proc.py -subj\_id s1305 -dsets \

# /raid4/sdanny/workingdata/tau/subjects/s1305/afni/OutBrick\_r1+orig.HEAD \

# /raid4/sdanny/workingdata/tau/subjects/s1305/afni/OutBrick\_r2+orig.HEAD \

# -scr\_overwrite -copy\_anat \

# /raid4/sdanny/workingdata/tau/subjects/s1305/anat/s1305\_anat+orig.HEAD \

# -tcat\_remove\_first\_trs 4 -blocks tshift align tlrc volreg blur mask \

# scale regress -align\_opts\_aea -giant\_move -cost lpc+ZZ -AddEdge \

# -volreg\_base\_dset \

# '/raid4/sdanny/workingdata/tau/subjects/s1305/afni/OutBrick\_r1+orig[1]' \

# -volreg\_align\_e2a -volreg\_tlrc\_warp -blur\_size 6 -mask\_dilate 1 \

# -scale\_max\_val 200 -regress\_basis GAM -regress\_censor\_motion 2.0 \

# -regress\_censor\_outliers 0.1 -out\_dir s1305.results2mm -script \

# s1305.process2mm.script -regress\_stim\_times \

# /raid4/sdanny/workingdata/tau/behav\_data/onsets/s1305\_visit1\_AC\_onsets.txt \

# /raid4/sdanny/workingdata/tau/behav\_data/onsets/s1305\_visit1\_AI\_onsets.txt \

# /raid4/sdanny/workingdata/tau/behav\_data/onsets/s1305\_visit1\_Incorrect\_onsets.txt \

# /raid4/sdanny/workingdata/tau/behav\_data/onsets/s1305\_visit1\_N\_onsets.txt \

# -regress\_stim\_labels AC AI Incorrect N -regress\_local\_times \

# -regress\_make\_ideal\_sum sum\_ideal.1D -regress\_est\_blur\_epits \

# -regress\_est\_blur\_errts -regress\_reml\_exec -regress\_opts\_3dD -num\_glt 8 \

# -gltsym 'SYM: +AI -AC' -glt\_label 1 ABias -gltsym 'SYM: +AC -N' \

# -glt\_label 2 ACvN -gltsym 'SYM: +AI -N' -glt\_label 3 AIvN -gltsym 'SYM: \

# +AI +AC' -glt\_label 4 AngryvFix -gltsym 'SYM: +AI +AC +N' -glt\_label 5 \

# FacevFix -gltsym 'SYM: +3\*Incorrect -AI -AC -N' -glt\_label 6 \

# ErrorvCorrect -gltsym 'SYM: +AI +AC -2\*N' -glt\_label 7 ACAIvNEU -gltsym \

# 'SYM: +.5\*AI +.5\*AC' -glt\_label 8 Angry -cbucket cbucket.stats.s1305 \

# -xsave -jobs 4