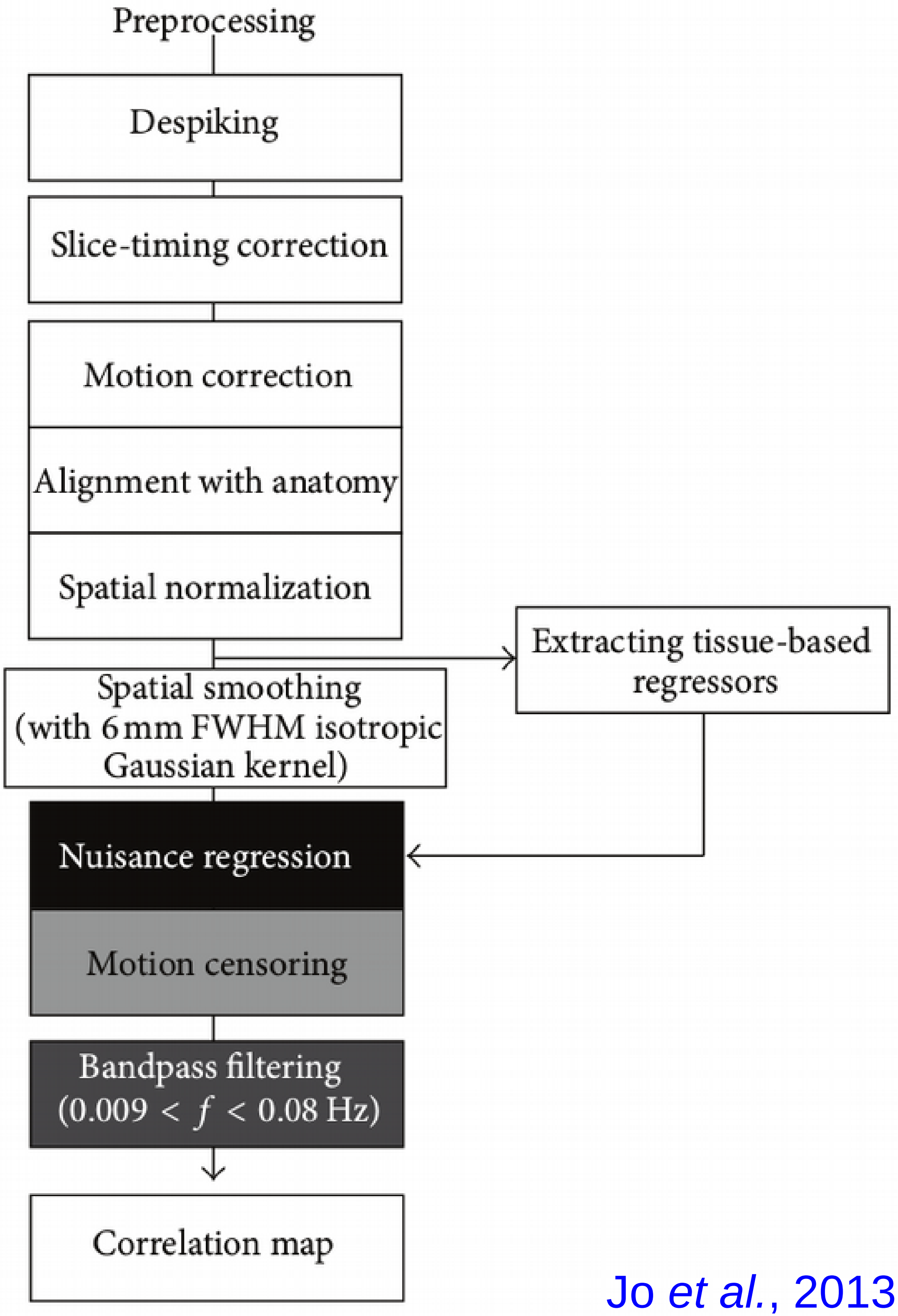
FMRI processing:

afni_proc.py *et al.*

(here, resting state analysis)

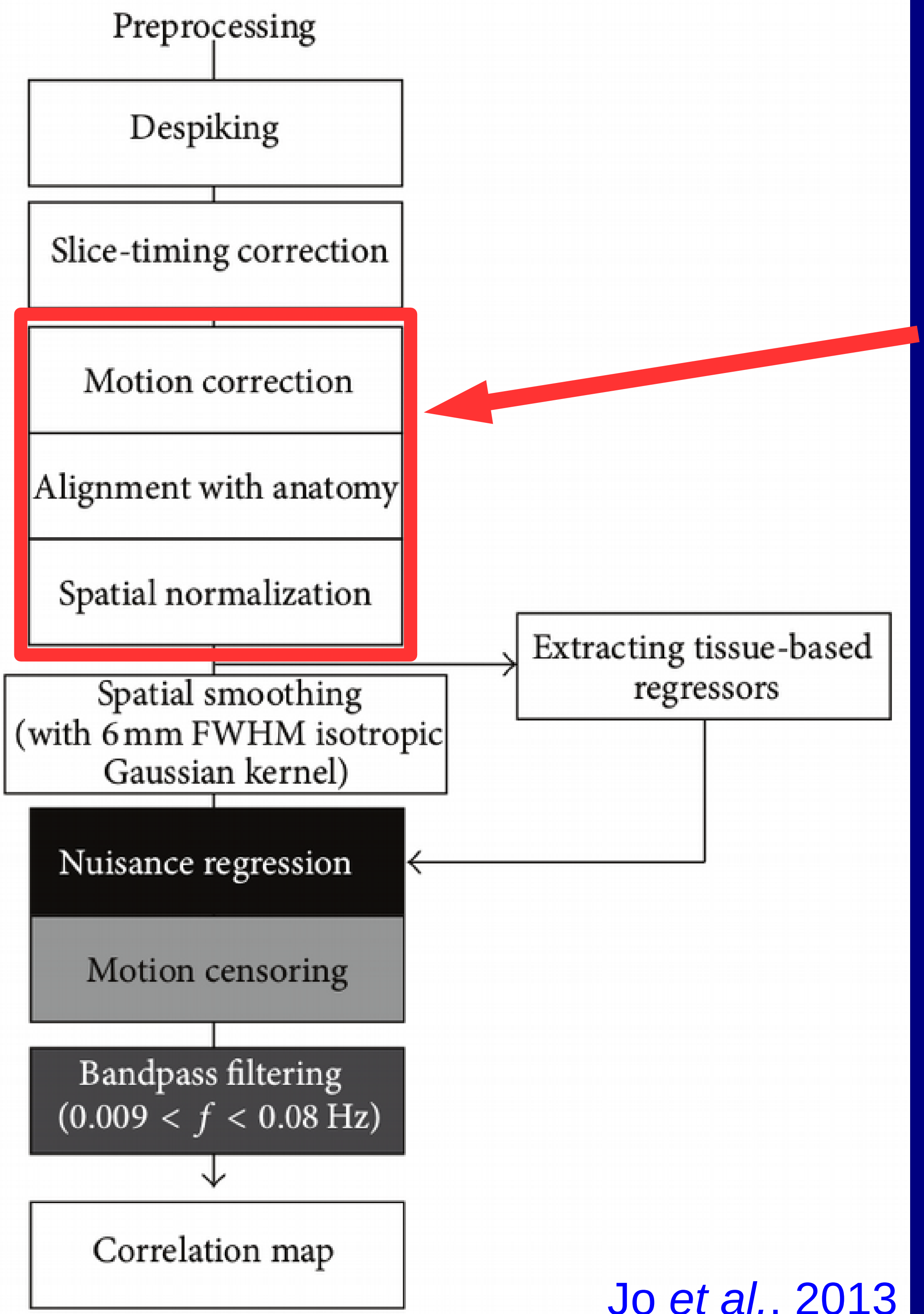


Jo *et al.*, 2013

Schematic of AFNI's recommended RS-FMRI pre-processing steps (HJ Jo *et al.*, 2010; 2013), but this is always adapting.

Pipeline most easily carried out using **afni_proc.py** (much credit to Rick Reynolds).

Starter examples in **afni_proc.py -help**



Preprocessing

Despiking

Slice-timing correction

Motion correction

Alignment with anatomy

Spatial normalization

Extracting tissue-based regressors

Spatial smoothing (with 6 mm FWHM isotropic Gaussian kernel)

Nuisance regression

Motion censoring

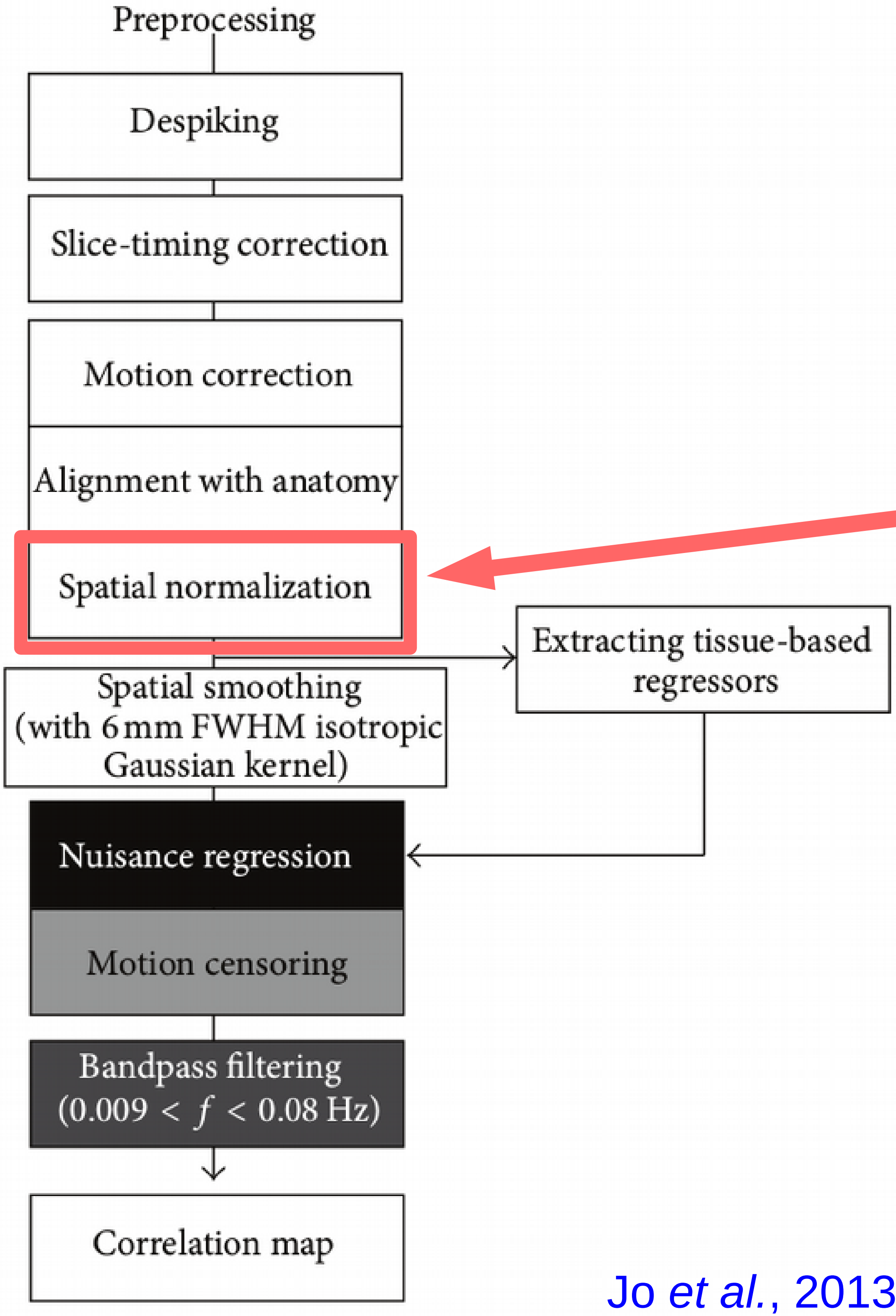
Bandpass filtering ($0.009 < f < 0.08$ Hz)

Correlation map

Jo *et al.*, 2013

Comments/updates:

Spatial transforms are calculated individually, but then concatenated and the single *result* is applied
→ to minimize smoothing.



Preprocessing

Despiking

Slice-timing correction

Motion correction

Alignment with anatomy

Spatial normalization

Extracting tissue-based regressors

Spatial smoothing (with 6 mm FWHM isotropic Gaussian kernel)

Nuisance regression

Motion censoring

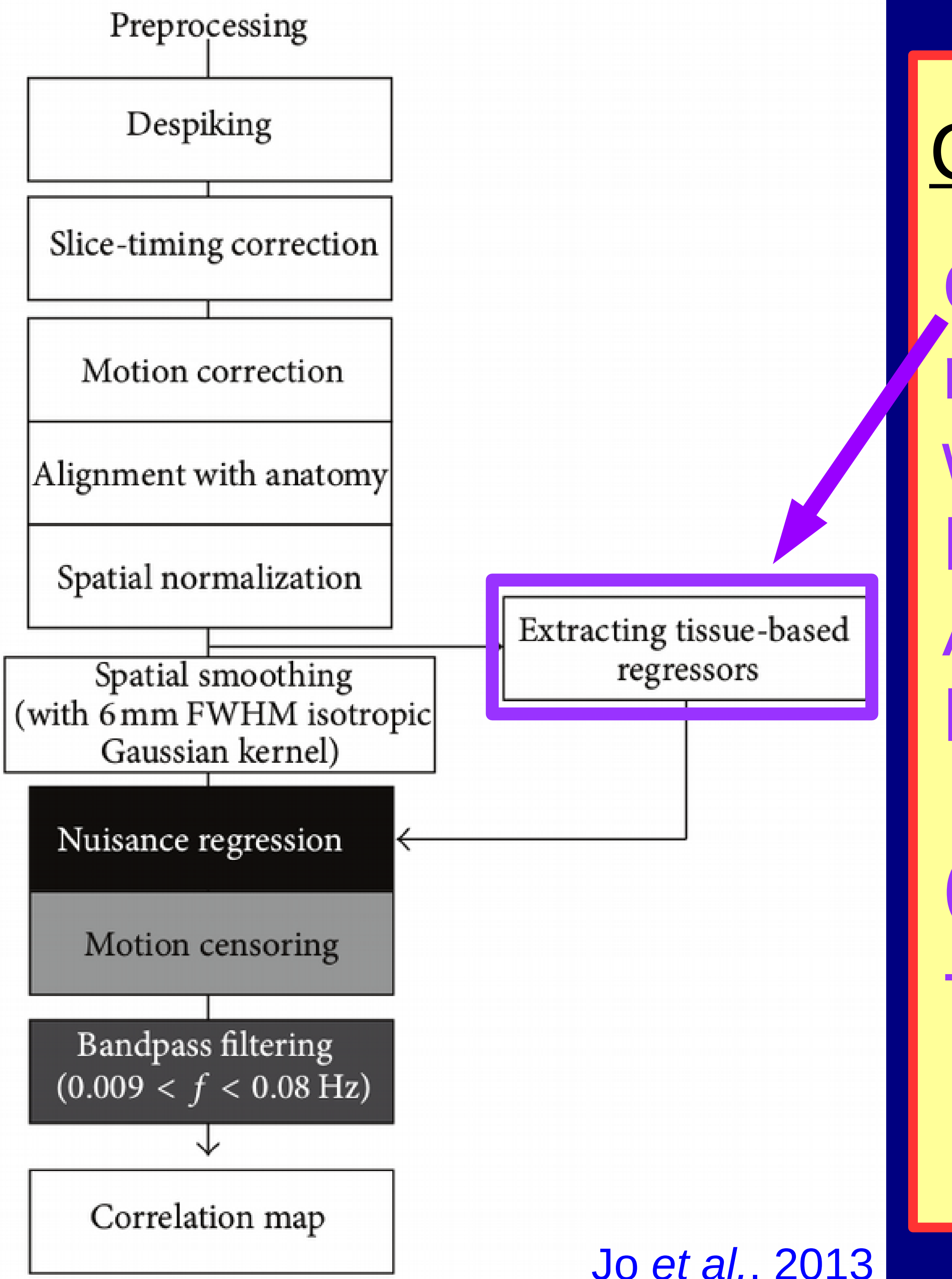
Bandpass filtering ($0.009 < f < 0.08$ Hz)

Correlation map

Jo *et al.*, 2013

Comments/updates:

Nonlinear registration available/recommended with **3dQwarp**.



Preprocessing

Despiking

Slice-timing correction

Motion correction

Alignment with anatomy

Spatial normalization

Extracting tissue-based regressors

Spatial smoothing (with 6 mm FWHM isotropic Gaussian kernel)

Nuisance regression

Motion censoring

Bandpass filtering ($0.009 < f < 0.08$ Hz)

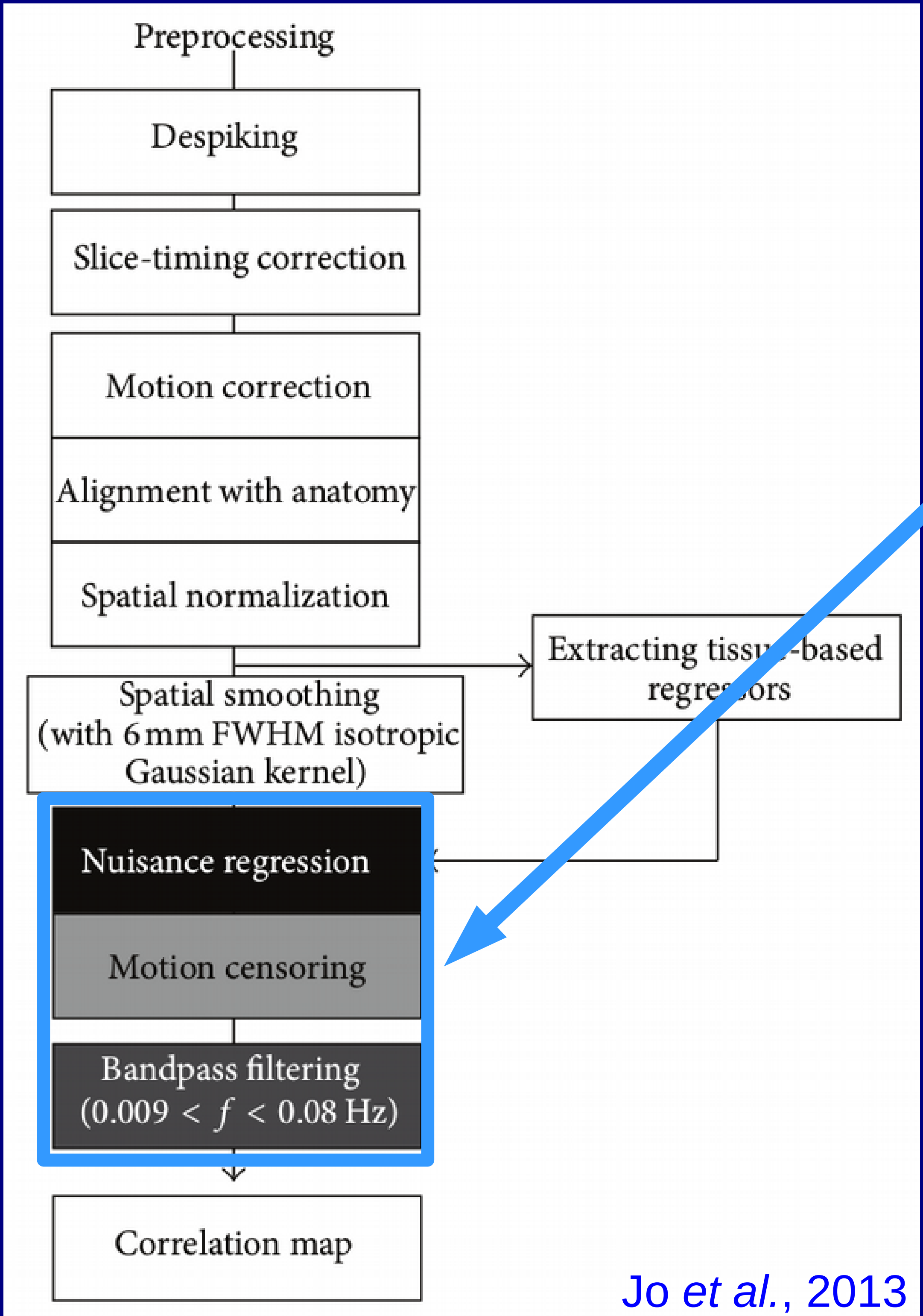
Correlation map

Jo *et al.*, 2013

Comments/updates:

Can calculate tissue regressors in many ways: 3dSeg, FreeSurfer output, ANATICOR, RETROICOR, etc.

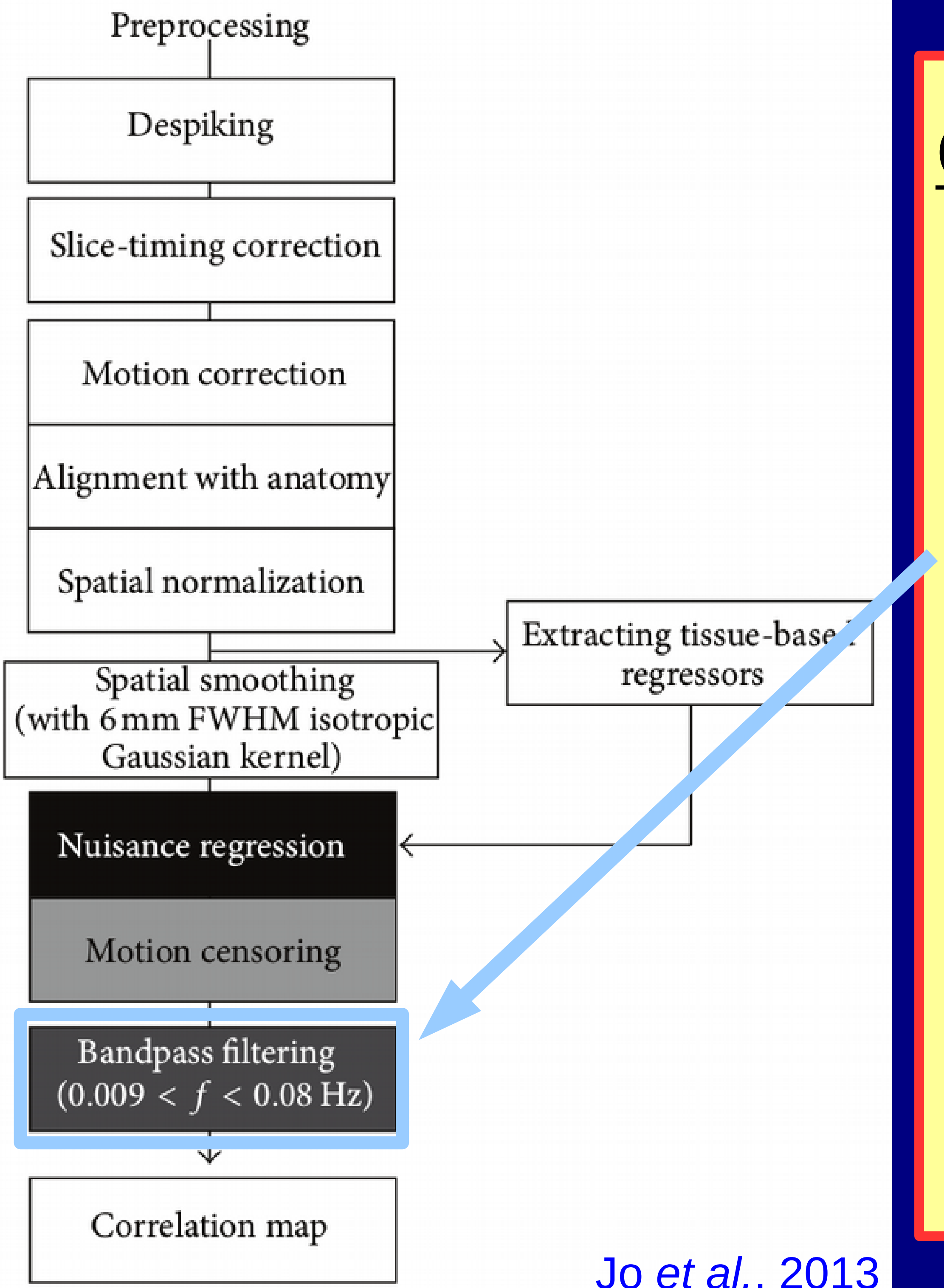
(various examples in -help file)



Jo *et al.*, 2013

Comments/updates:

All applied in a **single step** as regression, for mathematical correctness (if you care about that...). These should **not** be performed separately!



Preprocessing

Despiking

Slice-timing correction

Motion correction

Alignment with anatomy

Spatial normalization

Extracting tissue-based regressors

Spatial smoothing (with 6 mm FWHM isotropic Gaussian kernel)

Nuisance regression

Motion censoring

Bandpass filtering ($0.009 < f < 0.08$ Hz)

Correlation map

Jo *et al.*, 2013

Comments/updates:

NB: may not be necessary or even desired for RS-FMRI, esp. as TRs shorten-- have degree of freedom trouble. And, RS-FMRI has useful information above LFF range (see, e.g., Gohel & Biswal, 2015)

Preprocess via afni_proc.py: Setup

```
## Example 9b in 'afni_proc.py –help'
afni_proc.py -subj_id subj123          \
 -dsets epi_run1+orig.HEAD             \
 -copy_anat anat+orig                  \
 -blocks despike tshift align tlrc     \
         volreg blur mask regress      \
 -tcat_remove_first_trs 3              \
 -volreg_align_e2a                     \
 -volreg_tlrc_warp                     \
 -regress_anaticor                     \
 -regress_censor_motion 0.2            \
 -regress_censor_outliers 0.1          \
 -regress_bandpass 0.01 0.1            \
 -regress_apply_mot_types demean deriv \
 -regress_run_clustsim no              \
 -regress_est_blur_epits               \
 -regress_est_blur_errts
```

Preprocess via afni_proc.py: Setup

```
## Example 9b in 'afni_proc.py –help'
afni_proc.py -subj_id subj123          \
 -dsets epi_run1+orig.HEAD             \
 -copy_anat anat+orig                  \
 -blocks despike tshift align tlrc     \
         volreg blur mask regress      \
 -tcat_remove_first_trs 3              \
 -volreg_align_e2a                     \
 -volreg_tlrc_warp                     \
 -regress_anaticor                     \
 -regress_censor_motion 0.2            \
 -regress_censor_outliers 0.1          \
 -regress_bandpass 0.01 0.1            \
 -regress_apply_mot_types demean deriv \
 -regress_run_clustsim no              \
 -regress_est_blur_epits               \
 -regress_est_blur_errts
```

Complete pipeline! (16 lines!!)

Preprocess via afni_proc.py: Setup

```
## Example 9b in 'afni_proc.py –help'
afni_proc.py -subj_id subj123          \
 -dsets epi_run1+orig.HEAD             \
 -copy_anat anat+orig                  \
 -blocks despike tshift align tlrc     \
         volreg blur mask regress      \
 -tcat_remove_first_trs 3              \
 -volreg_align_e2a                     \
 -volreg_tlrc_warp                     \
 -regress_anaticor                     \
 -regress_censor_motion 0.2            \
 -regress_censor_outliers 0.1          \
 -regress_bandpass 0.01 0.1            \
 -regress_apply_mot_types demean deriv \
 -regress_run_clustsim no              \
 -regress_est_blur_epits               \
 -regress_est_blur_errts
```

Input
data set(s)

Preprocess via afni_proc.py: Setup

```
## Example 9b in 'afni_proc.py –help'
afni_proc.py -subj_id subj123          \
 -dsets epi_run1+orig.HEAD             \
 -copy_anat anat+orig                  \
 -blocks despike tshift align tlrc     \
         volreg blur mask regress      \
 -tcat_remove_first_trs 3              \
 -volreg_align_e2a                     \
 -volreg_tlrc_warp                     \
 -regress_anaticor                     \
 -regress_censor_motion 0.2            \
 -regress_censor_outliers 0.1          \
 -regress_bandpass 0.01 0.1            \
 -regress_apply_mot_types demean deriv \
 -regress_run_clustsim no              \
 -regress_est_blur_epits               \
 -regress_est_blur_errts
```

Choose major processing steps (blocks)

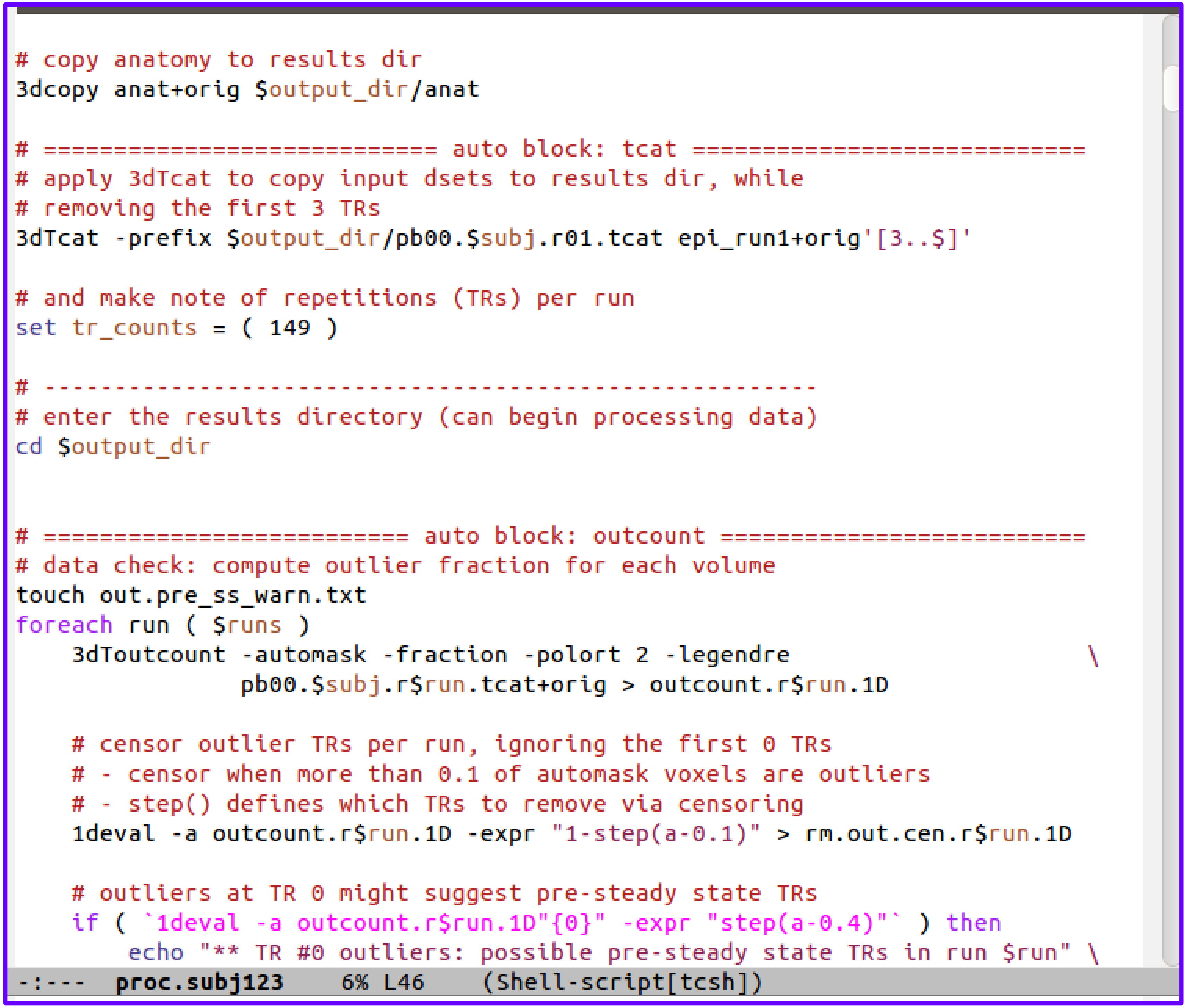
Preprocess via afni_proc.py: Setup

```
## Example 9b in 'afni_proc.py –help'
afni_proc.py -subj_id subj123          \
 -dsets epi_run1+orig.HEAD             \
 -copy_anat anat+orig                  \
 -blocks despike tshift align tlrc     \
         volreg blur mask regress      \
 -tcat_remove_first_trs 3              \
 -volreg_align_e2a                     \
 -volreg_tlrc_warp                     \
 -regress_anaticor                     \
 -regress_censor_motion 0.2            \
 -regress_censor_outliers 0.1          \
 -regress_bandpass 0.01 0.1            \
 -regress_apply_mot_types demean deriv \
 -regress_run_clustsim no              \
 -regress_est_blur_epits               \
 -regress_est_blur_errts
```

Specify details
and options
for each block

Preprocess via afni_proc.py: Script

→ **previous command generates a 484 line script file to process that subject:**



```
# copy anatomy to results dir
3dcopy anat+orig $output_dir/anat

# ============================ auto block: tcat ============================
# apply 3dTcat to copy input dsets to results dir, while
# removing the first 3 TRs
3dTcat -prefix $output_dir/pb00.$subj.r01.tcat epi_run1+orig'[3..$]'

# and make note of repetitions (TRs) per run
set tr_counts = ( 149 )

# -------------------------------------------------------
# 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 ( $runs )
    3dToutcount -automask -fraction -polort 2 -legendre                     \
                pb00.$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" \
-:---  proc.subj123    6% L46    (Shell-script[tcsh])
```

. . .

QC: free review/driver scripts - 1

At individual level:

afni_proc.py finishes processing volumes, outputting a summary →

and a set of scripts:

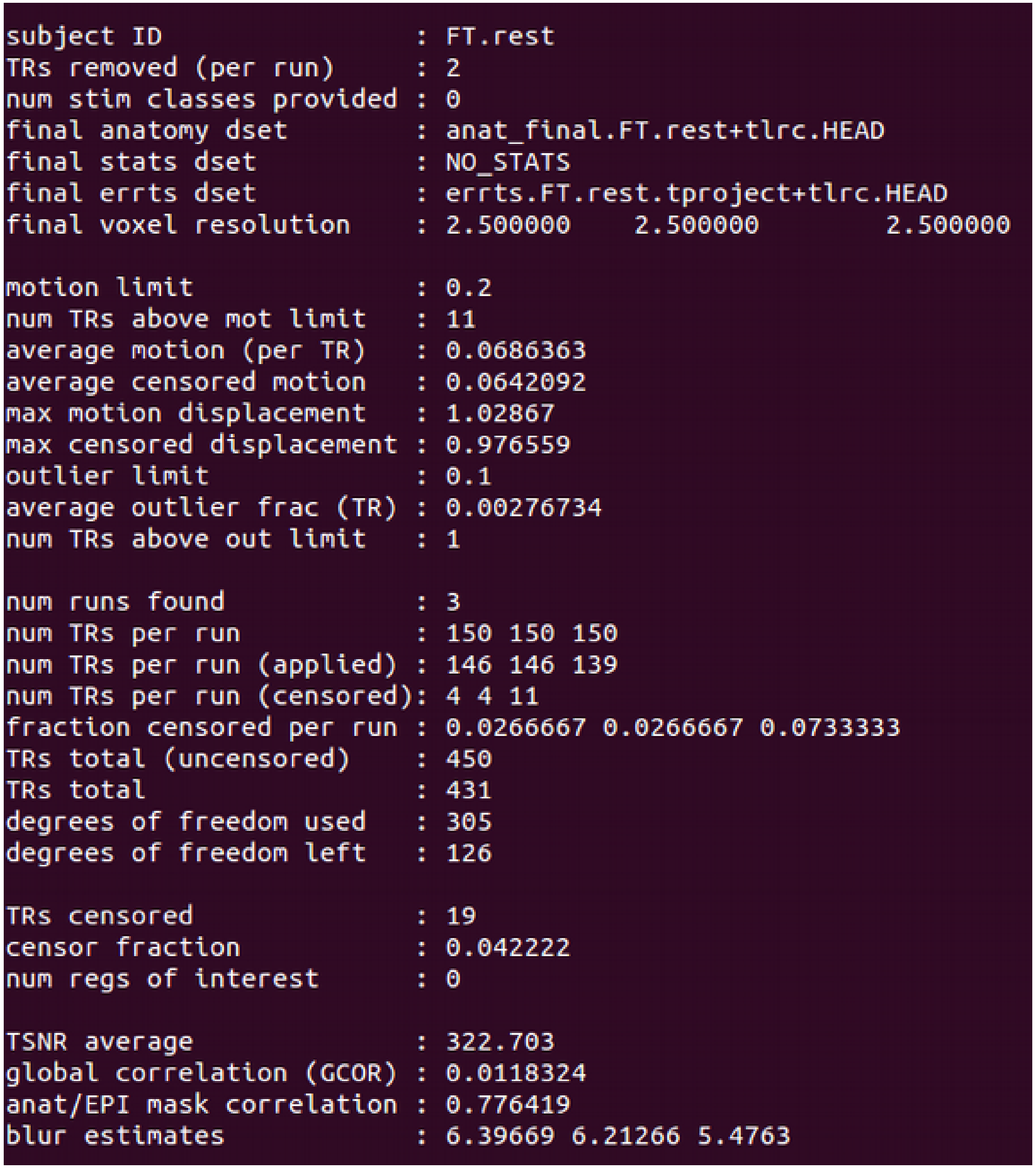
@ss_review_basic

@ss_review_driver

@ss_review_driver_commands

to automatically:

- **+ view alignment;**
- **+ view motion, censoring, etc.;**
- **+ store info in text file.**



```
subject ID                : FT.rest
TRs removed (per run)     : 2
num stim classes provided : 0
final anatomy dset        : anat_final.FT.rest+tlrc.HEAD
final stats dset          : NO_STATS
final errts dset          : errts.FT.rest.tproject+tlrc.HEAD
final voxel resolution    : 2.500000    2.500000        2.500000

motion limit              : 0.2
num TRs above mot limit   : 11
average motion (per TR)   : 0.0686363
average censored motion   : 0.0642092
max motion displacement   : 1.02867
max censored displacement : 0.976559
outlier limit             : 0.1
average outlier frac (TR) : 0.00276734
num TRs above out limit   : 1

num runs found            : 3
num TRs per run           : 150 150 150
num TRs per run (applied) : 146 146 139
num TRs per run (censored): 4 4 11
fraction censored per run : 0.0266667 0.0266667 0.0733333
TRs total (uncensored)    : 450
TRs total                 : 431
degrees of freedom used   : 305
degrees of freedom left   : 126

TRs censored              : 19
censor fraction           : 0.042222
num regs of interest      : 0

TSNR average              : 322.703
global correlation (GCOR) : 0.0118324
anat/EPI mask correlation : 0.776419
blur estimates            : 6.39669 6.21266 5.4763
```

QC: free review/driver scripts - 2

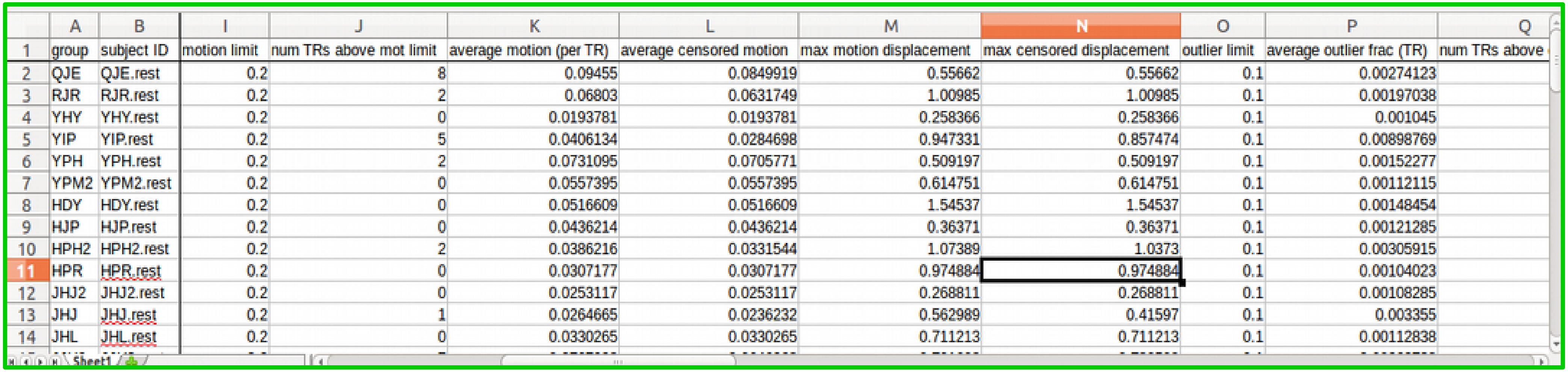
At group level:

gen_ss_review_table.py:

After running afni_proc.py on many subjects, use this to combine the summary/motion/censoring/etc. data into a single spreadsheet/text file.

Example:

```
gen_ss_review_table.py                                       \
    -tablefile review_table.xls                              \
    -infiles */*.rest.results/out.ss_review.*.txt
```



	A	B	I	J	K	L	M	N	O	P	Q
1	group	subject ID	motion limit	num TRs above mot limit	average motion (per TR)	average censored motion	max motion displacement	max censored displacement	outlier limit	average outlier frac (TR)	num TRs above
2	QJE	QJE.rest	0.2	8	0.09455	0.0849919	0.55662	0.55662	0.1	0.00274123	
3	RJR	RJR.rest	0.2	2	0.06803	0.0631749	1.00985	1.00985	0.1	0.00197038	
4	YHY	YHY.rest	0.2	0	0.0193781	0.0193781	0.258366	0.258366	0.1	0.001045	
5	YIP	YIP.rest	0.2	5	0.0406134	0.0284698	0.947331	0.857474	0.1	0.00898769	
6	YPH	YPH.rest	0.2	2	0.0731095	0.0705771	0.509197	0.509197	0.1	0.00152277	
7	YPM2	YPM2.rest	0.2	0	0.0557395	0.0557395	0.614751	0.614751	0.1	0.00112115	
8	HDY	HDY.rest	0.2	0	0.0516609	0.0516609	1.54537	1.54537	0.1	0.00148454	
9	HJP	HJP.rest	0.2	0	0.0436214	0.0436214	0.36371	0.36371	0.1	0.00121285	
10	HPH2	HPH2.rest	0.2	2	0.0386216	0.0331544	1.07389	1.0373	0.1	0.00305915	
11	HPR	HPR.rest	0.2	0	0.0307177	0.0307177	0.974884	0.974884	0.1	0.00104023	
12	JHJ2	JHJ2.rest	0.2	0	0.0253117	0.0253117	0.268811	0.268811	0.1	0.00108285	
13	JHJ	JHJ.rest	0.2	1	0.0264665	0.0236232	0.562989	0.41597	0.1	0.003355	
14	JHL	JHL.rest	0.2	0	0.0330265	0.0330265	0.711213	0.711213	0.1	0.00112838	

QC: @radial_correlate

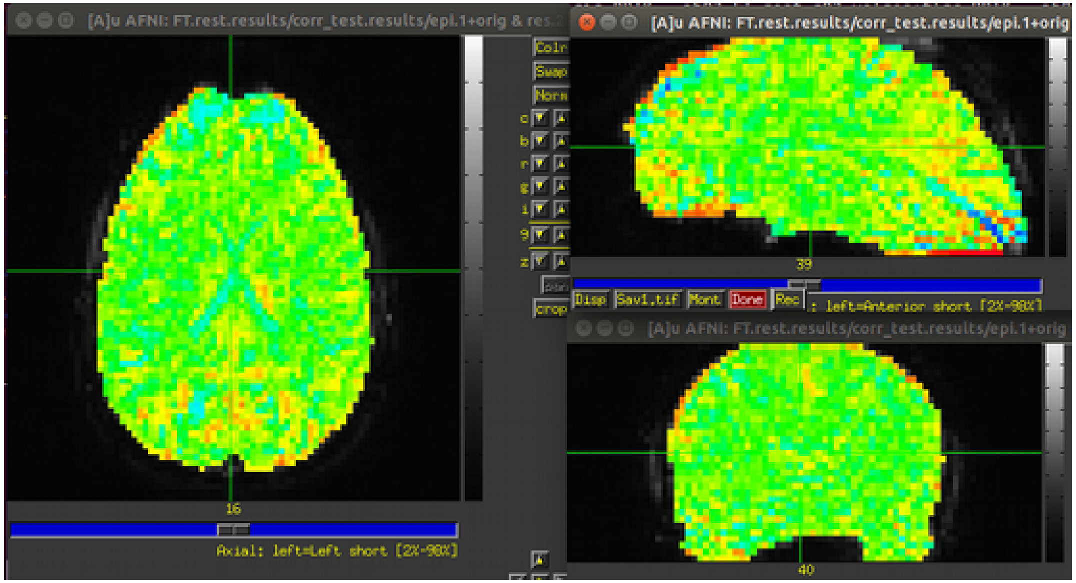
For scanner/coils/processing:

```
$ @radial_correlate EPI_DATASET
```

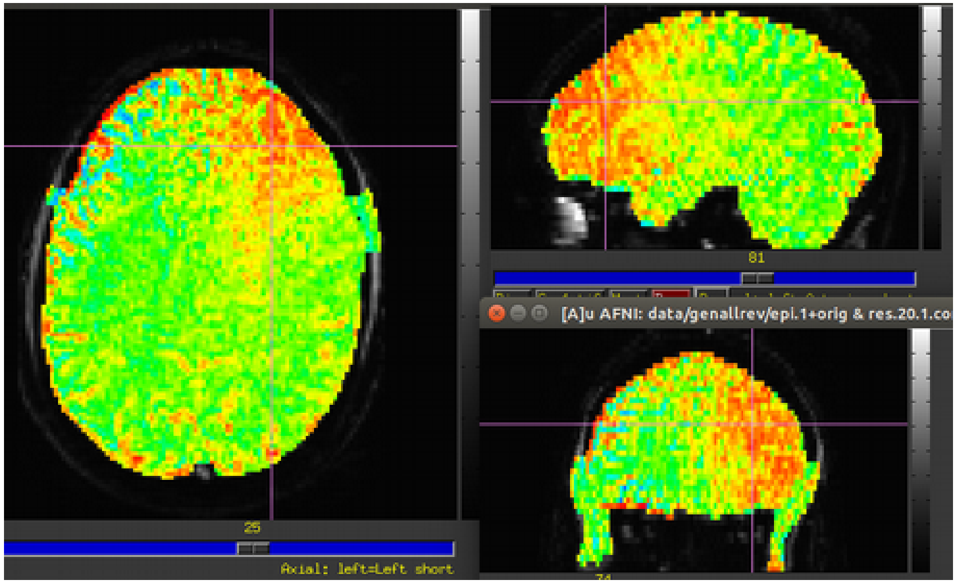
→ for each *i*th voxel, calculates the correlation of *i*'s time series with the average time series of a surrounding sphere (default rad: 20 mm).

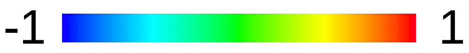
Creates a whole brain map where oddly homogeneous regions are highlighted:

No 'red zones'


[A]u AFNI: FT.rest.results/corr_test.results/epi.1+orig & res...
[A]u AFNI: FT.rest.results/corr_test.results/epi.1+orig
[A]u AFNI: FT.rest.results/corr_test.results/epi.1+orig
Colr
Swap
Norm
crop
Disp
Sav1.tif
Mont
Done
Rec
16
39
40
Axial: left=Left short [2%-98%]
: left=Anterior short [2%-98%]

Yes 'red zones'!





AFNI's InstaCorr (demo)

Fun with AFNI: (RS)FMRI edition

`afni_proc.py`

- + easy to customize+build a processing pipeline.
- + automatic QC scripts included-- for free!
- + one efficient command to report, script with, store, ...
- + been making 'reproducibility' easy for years.

`@radial_correlate`

- + simple QC method for scanner/coil artefacts.
- + "bad coils, bad coils! wha'ch'ya gonna' do when they fail for you?"

--RR

`InstaCorr`

- + best quality control: looking at your data!
- + best science ideas: looking at your data!
- + has a group-level analogue: Group InstaCorr (GIC).