Group Analysis with AFNI - Hands On

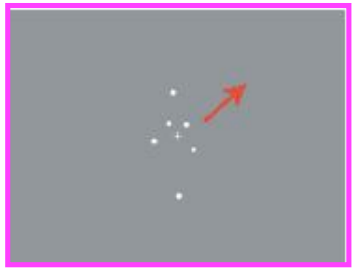
- The following sample group analysis comes from “How-to #5 -- Group Analysis: AFNI 3dANOVA3”, described in full detail on the AFNI website: http://afni.nimh.gov/pub/dist/HOWTO/howto/ht05_group/html

- **Brief description of experiment :**
  - Design:
    - Rapid event-related
  - “Stimulus Condition” has 4 levels:
    - TM = Tool Movies
    - HM = Human Movies
    - TP = Tool Point Light Displays
    - HP = Human Point Light Displays

Tool Movie

Human Movie

Tool Point Light



Human Point Light

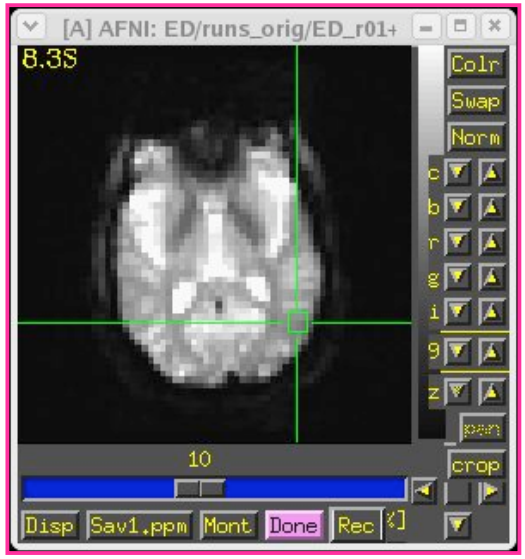
- Data Collected:
  - 1 Anatomical (SPGR) dataset for each subject
    - 124 sagittal slices
  - 10 Time Series (EPI) datasets for each subject
    - 23 axial slices x 138 volumes = 3174 volumes/timepoints per run
      - note: each run consists of random presentations of rest and all 4 stimulus condition levels
    - TR = 2 sec; voxel dimensions = 3.75 x 3.75 x 5 mm
  - Sample size, n=7 (subjects ED, EE, EF, FH, FK, FL, FN)

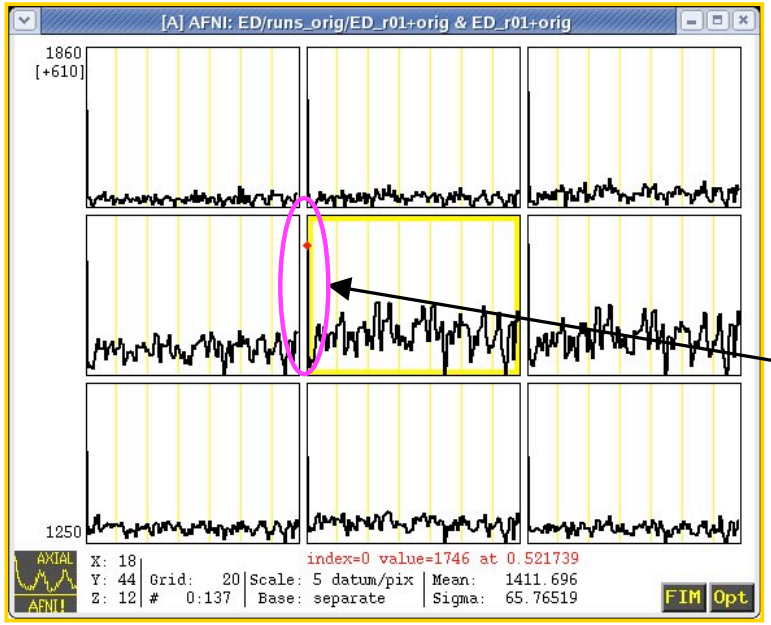
- **Analysis Steps:**
  - **Part I**: Process data for each subject first
    - Pre-process subjects' data ⇒ many steps involved here…
    - Run deconvolution analysis on each subject's dataset --- **3dDeconvolve**
  - **Part II**: Run group analysis
    - 3-way Analysis of Variance (ANOVA) --- **3dANOVA3**
    - i.e., Object Type (2) x Animation Type (2) x Subjects (7) = 3-way ANOVA

- **PART I ⇒ Process Data for each Subject First:**
  - ✧ Hands-on example: Subject **ED**
  - ✧ We will begin with ED's anat dataset and 10 time-series (3D+time) datasets:

EDspgr+orig, **EDspgr+tlrc**, **ED_r01+orig**, **ED_r02+orig** … **ED_r10+orig**

  - ➥ Below is ED's **ED_r01+orig** (3D+time) dataset. Notice the first two time points of the time series have relatively high intensities*. We will need to remove them later:





Timepoints 0 and 1 have high intensity values

✶ **Images obtained during the first 4-6 seconds of scanning will have much larger intensities than images in the rest of the timeseries, when magnetization (and therefore intensity) has decreased to its steady state value**

- **STEP 1: Check for possible "outliers" in each of the 10 time series datasets. The AFNI program to use is 3dToutcount (also run by default in `to3d`)**

  - An outlier is usually seen as an isolated spike in the data, which may be due to a number of factors, such as subject head motion or scanner irregularities.
  - In any case, the outlier is not a true signal that results from presentation of a stimulus event, but rather, an artifact from something else -- it is noise.

```
foreach run (01 02 03 04 05 06 07 08 09 10)
        3dToutcount -automask ED_r{$run}+orig  \
                    > toutcount_r{$run}.1D
end
```

  - How does this program work? For each time series, the trend and Mean Absolute Deviation are calculated. Points far away from the trend are considered outliers. "Far away" is mathematically defined.
    - See 3dToutcount -help for specifics.
  - `-automask`: Does the outlier check only on voxels within the brain and ignores background voxels (which are detected by the program because of their smaller intensity values).
  - `>` : This is the "redirect" symbol in UNIX. Instead of displaying the results onto the screen, they are saved into a text file. In this example, the text files are called `toutcount_r{$run}.1D`.

- Subject ED's outlier files:

  toutcount_r01.1D

  toutcount_r02.1D

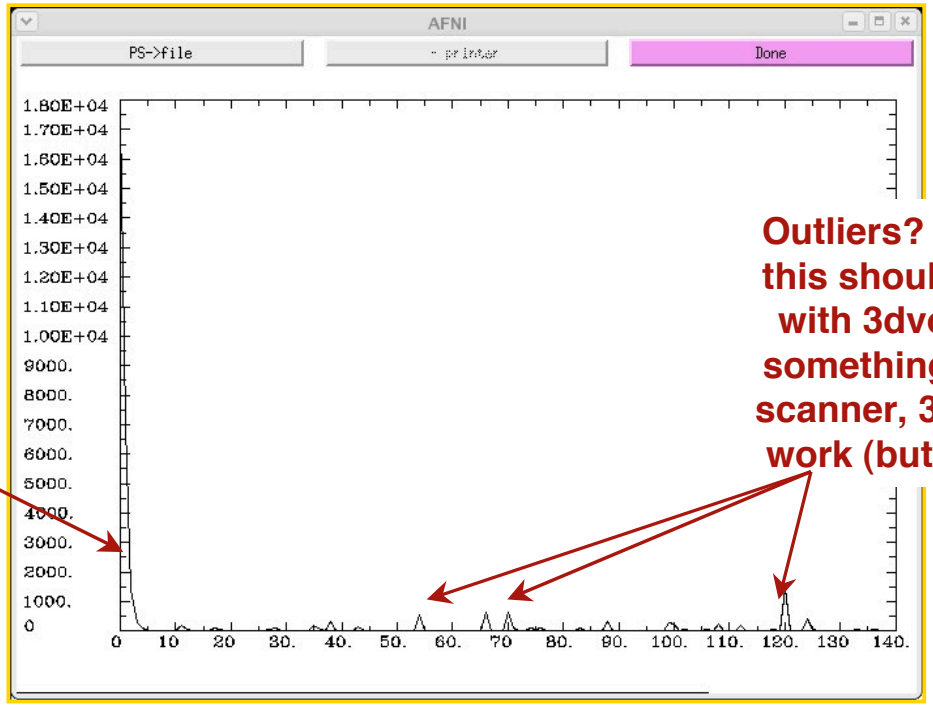
  ...

  toutcount_r10.1D

Note: "1D" is used to identify a text file. In this case, each file consists a column of 138 numbers (b/c of 138 time points).

- Use AFNI **1dplot** to display any one of ED's outlier files. For example:

```
1dplot toutcount_r04.1D
```



Num. of 'outlier' voxels

time

High intensity values in the beginning are usually due to scanner attempting to reach steady state.

Outliers? If head motion, this should be cleared up with 3dvolreg. If due to something weird with the scanner, 3dDespike might work (but use sparingly).

- **STEP 2: Shift voxel time series so that separate slices are aligned to the same temporal origin using `3dTshift`**

  - The temporal alignment is done so it seems that all slices were acquired at the same time, i.e., the beginning of each TR.
  - The output dataset time series will be interpolated from the input to a new temporal grid. There are several interpolation methods to choose from, including 'Fourier', 'linear', 'cubic', 'quintic', and 'heptic'.

```
foreach run (01 02 03 04 05 06 07 08 09 10)
      3dTshift -tzero 0  -heptic        \
               -prefix ED_r{$run}_ts    \
               ED_r{$run}+orig
end
```

  - `-tzero`: Tells the program which slice's time offset to align to. In this example, the slices are all aligned to the time offset of the first (`0`) slice.
  - `-heptic`: Use the 7th order Lagrange polynomial interpolation. Why 7th order? Bob Cox likes this (and that's good enough for me).

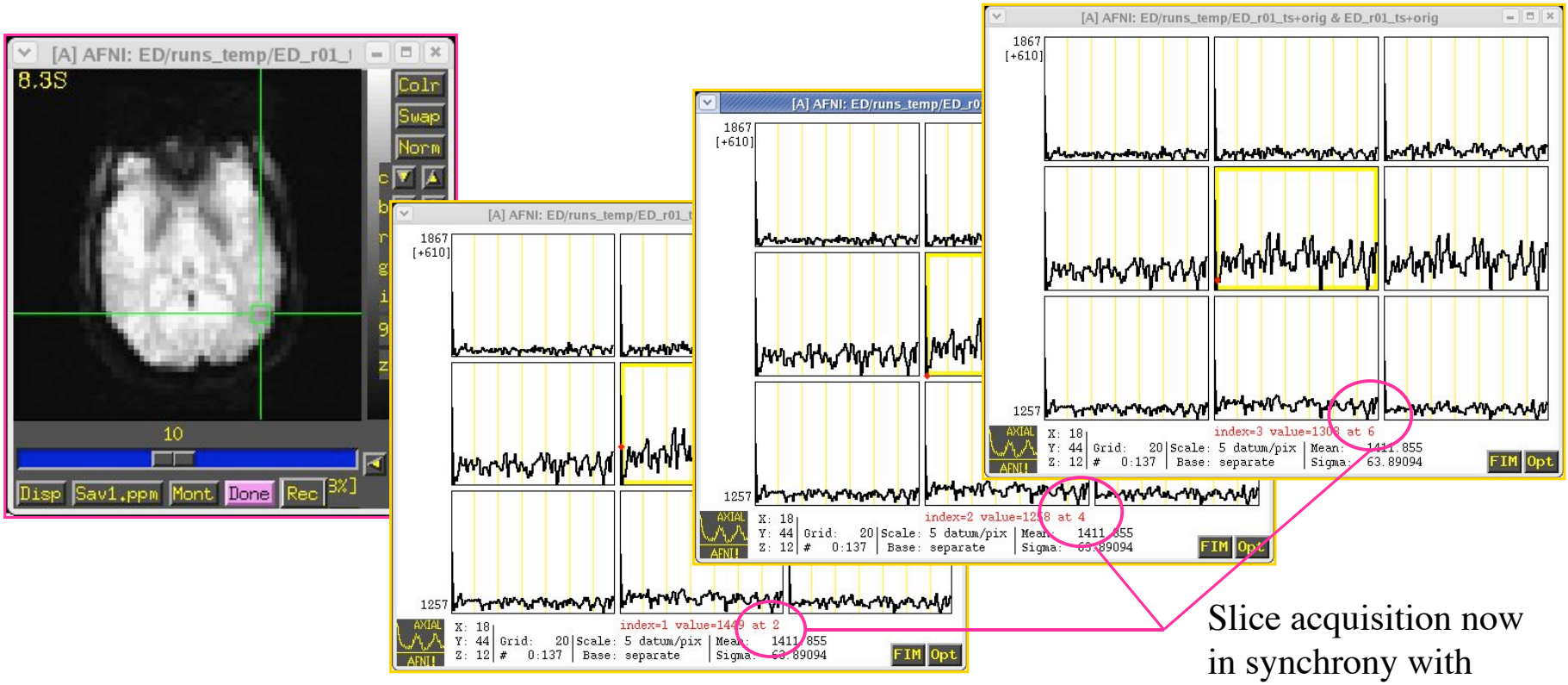
- Subject ED's newly created time shifted datasets:

  ED_r01_ts+orig.HEAD ED_r01_ts+orig.BRIK

  … …

  ED_r10_ts+orig.HEAD ED_r10_ts+orig.BRIK

- Below is run 01 of ED's time shifted dataset, **ED_r01_ts+orig**:



Slice acquisition now in synchrony with beginning of TR

- **STEP 3: Volume Register the voxel time series for each 3D+time dataset using AFNI program `3dvolreg`**

  - We will also remove the first 2 time points at this step

```
foreach run (01 02 03 04 05 06 07 08 09 10)
         3dvolreg  -verbose
                   -base ED_r01_ts+orig'[2]'   \
                   -prefix ED_r{$run}_vr       \
                   -1Dfile dfile.r{$run}.1D    \
                   ED_r{$run}_ts+orig'[2..137]'
  end
```

  - `-verbose`: Prints out progress report onto screen
  - `-base`: Timepoint **2** is our base/target volume to which the remaining timepoints (3-137) will be aligned. We are ignoring timepoints 0 and 1
  - `-prefix` gives our output files a new name, e.g., `ED_r01_vr+orig`
  - `-1Dfile`: Save motion parameters for each run (roll, pitch, yaw, dS, dL, dP) into a file containing 6 ASCII formatted columns.
  - `ED_r{$run}_ts+orig'[2..137]'` refers to our input datasets (runs 01-10) that will be volume registered. Notice that we are removing timepoints 0 and 1

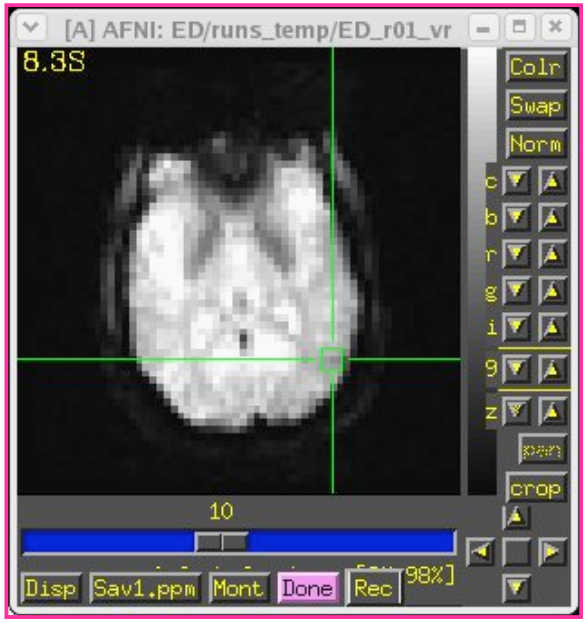
- Subject ED's newly created volume registered datasets:

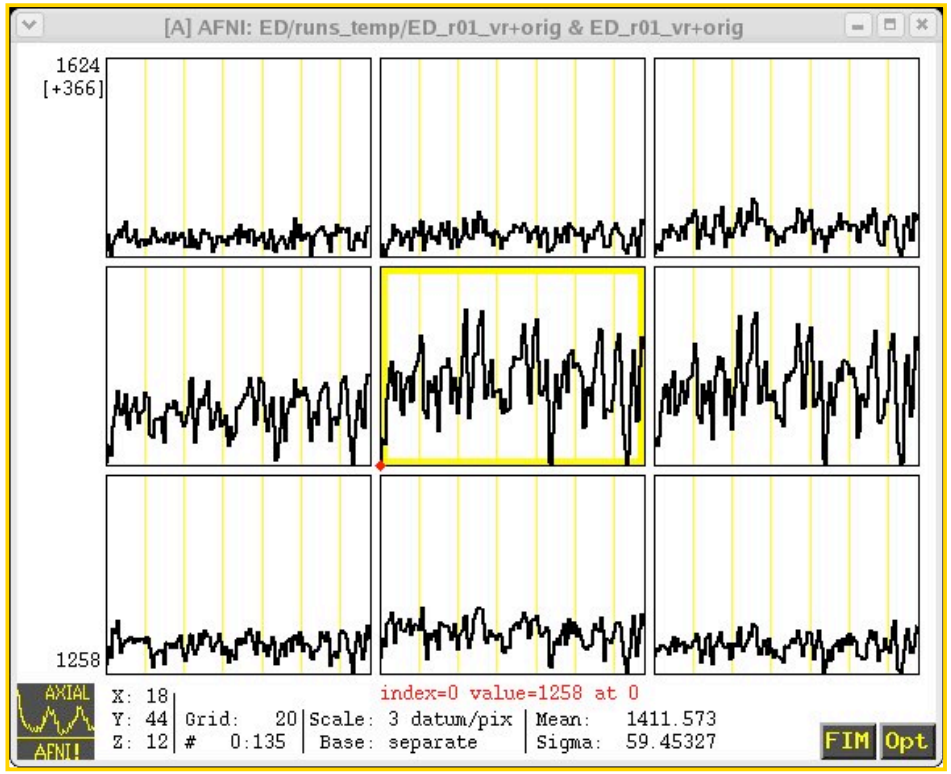
  ED_r01_vr+orig.HEAD ED_r01_vr+orig.BRIK

  … …

  ED_r10_vr+orig.HEAD ED_r10_vr+orig.BRIK

- Below is run 01 of ED's volume registered datasets, **ED_r01_vr+orig**:





- **STEP 4: Smooth 3D+time datasets with AFNI `3dmerge`**

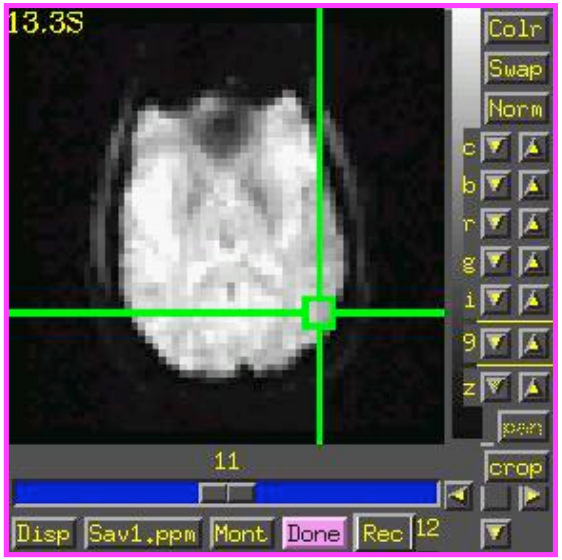
  - The result of spatial blurring (filtering) is somewhat cleaner, more contiguous activation blobs
  - Spatial blurring will be done on ED's time shifted, volume registered datasets:

```
foreach run (01 02 03 04 05 06 07 08 09 10)
        3dmerge -1blur_fwhm 4                \
                -doall                       \
                -prefix ED_r{$run}_vr_bl     \
                ED_r{$run}_vr+orig
end
```

  - `-1blur_fwhm 4` sets the Gaussian filter to have a full width half max of 4mm (You decide the type of filter and the width of the selected filter)
  - `-doall` applies the editing option (in this case the Gaussian filter) to all sub-bricks uniformly in each dataset

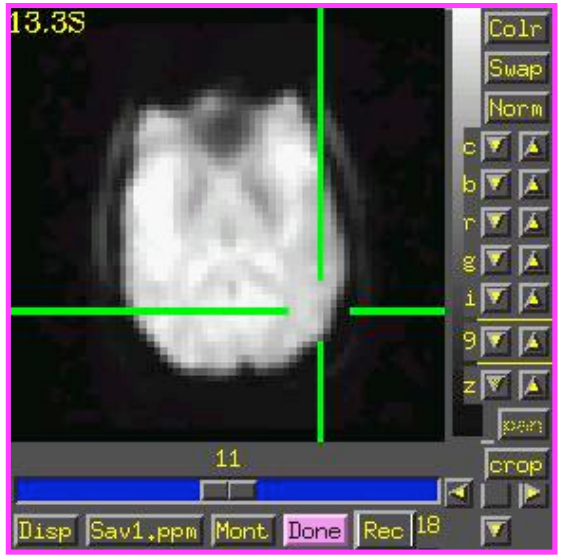
✧ Result from **3dmerge**:

ED_r01_vr+orig



Before blurring

ED_r01_vr_bl+orig



After blurring

- **STEP 5: Scaling the Data - i.e., Calculating Percent Change**

  - This particular step is a bit more involved, because it is comprised of three parts. Each part will be described in detail:

    A. Create a mask so that all background values (outside of the volume) are set to zero with **`3dAutomask`**
    B. Do a voxel-by-voxel calculation of the mean intensity value with **`3dTstat`**
    C. Do a voxel-by-voxel calculation of the percent signal change with **`3dcalc`**

  - Why should we scale our data?
    - Scaling becomes an important issue when comparing data *across* subjects, because baseline/rest states will vary from subject to subject
    - The amount of activation in response to a stimulus event will also vary from subject to subject
    - As a result, the baseline Impulse Response Function (IRF) and the stimulus IRF will vary from subject to subject -- we must account for this variability
    - By converting to percent change, we can compare the activation calibrated with the relative change of signal, instead of the arbitrary baseline of FMRI signal

- For example:

  Subject 1 - Signal in hippocampus goes from 1000 (baseline) to 1050 (stimulus condition)

  Difference = **50** IRF units

  Subject 2 - Signal in hippocampus goes from 500 (baseline) to 525 (stimulus condition)

  Difference = **25** IRF units

- Conclusion:
  - Subject 1 shows *twice* as much activation in response to the stimulus condition than does Subject 2 --- **WRONG!!**
  - If ANOVA were run on these difference scores, the change in baseline from subject to subject would add *variance* to the analysis
  - We must control for these differences in baseline across subjects by somehow normalizing the baseline so that a reliable comparison between subjects can be made

- Solution:
  - Compute **Percent Signal Change**
    - i.e., by what percent does the Impulse Response Function increase with presentation of the stimulus condition, relative to baseline?

  - Percent Change Calculation:
    - If **A** = Stimulus IRF
    - If **B** = Baseline IRF

Percent Signal Change = `(A/B) * 100%`

- ***Subject 1*** -- Stimulus (A) = 1050, Baseline (B) = 1000

  (1050/1000) * 100% = 105% or **5% increase in IRF**

- ***Subject 2*** -- Stimulus (A) = 525, Baseline (B) = 500

  (525/500) * 100% = 105% or **5% increase in IRF**

  - Conclusion:
    - Both subjects show a ***5% increase*** in signal change from baseline to stimulus condition
    - Therefore, no significant difference in signal change between these two subjects

- **STEP 5A: Ignore any background values in a dataset by creating a mask with `3dAutomask`**

  - Values in the background have very low baseline values, which can lead to artificially large percent signal change values. Let's remove them altogether by creating a mask of our dataset, where values inside the brain are assigned a value of "1" and values outside of the brain (e.g., noise) are assigned a value of "0"
  - This mask will be used later when the percent signal change in each voxel is calculated. A percent change will be computed only for voxels inside the mask
  - A mask will be created for each of Subject ED's time shifted/volume registered/blurred 3D+time datasets:

```
foreach run (01 02 03 04 05 06 07 08 09 10)
    3dAutomask -dilate 1 -prefix mask_r{$run} \
                ED_r{$run}_vr_bl+orig
end
```

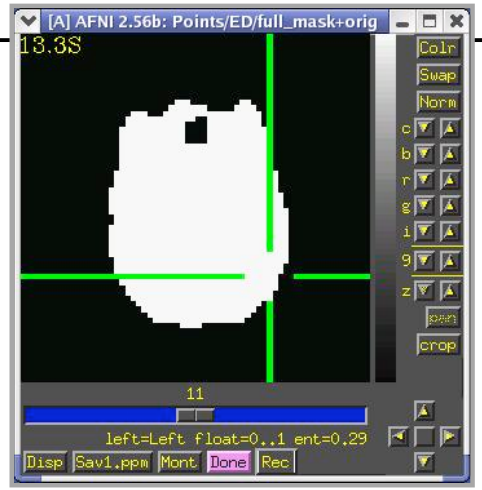
  - ➥ Output of **`3dAutomask`**: A mask dataset for each 3D+time dataset:

    `mask_r01+orig`, `mask_r02+orig` … `mask_r10+orig`

- Now let's take those 10 masks (we don't need 10 separate masks) and combine them to make one master or "full mask", which will be used to calculate the percent signal change only for values inside the mask (i.e., inside the brain).
- **3dcalc** -- one of the most versatile AFNI programs -- is used to combine the 10 masks into one:

```
3dcalc -a mask_r01+orig -b mask_r02+orig -c mask_r03+orig \
       -d mask_r04+orig -e mask_r05+orig -f mask_r06+orig \
       -g mask_r07+orig -h mask_r08+orig -i mask_r09+orig \
       -j mask_r10+orig                                   \
       -expr 'or(a+b+c+d+e+f+g+h+i+j)'                    \
       -prefix full_mask
```

Output: **full_mask+orig**:



➥ `-expr 'or'`: Used to determine whether voxels along the edges make it to the full mask or not. If an edge voxel has a "1" value in *any* of the individual masks, the 'or' keeps that voxel as part of the full mask.

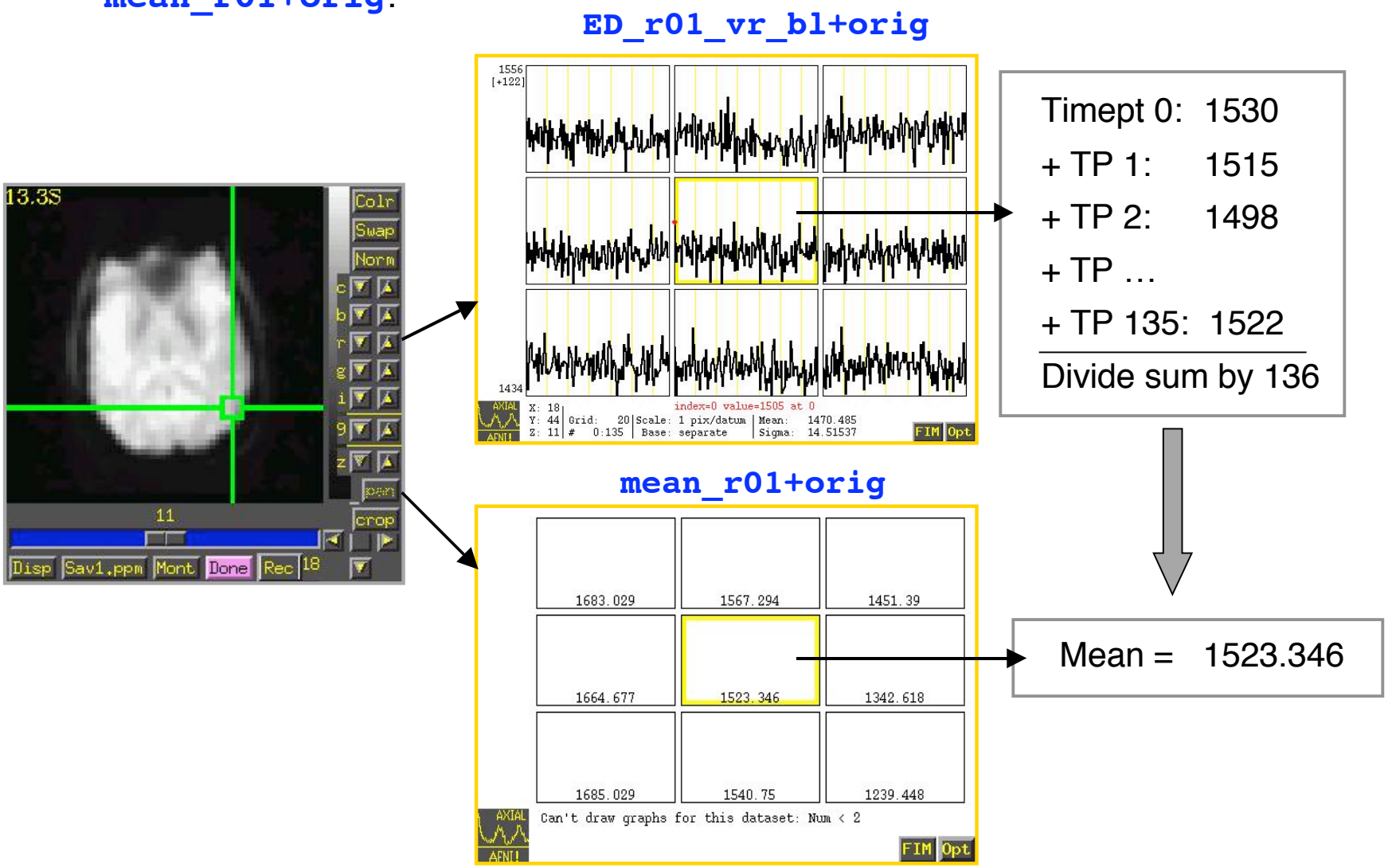
- **STEP 5B:** Create a voxel-by-voxel mean for each timeseries dataset with **3dTstat**

  - For each voxel, add the intensity values of the 136 time points and divide by 136
  - The resulting mean will be inserted into the “B” slot of our percent signal change equation (`A/B*100%`)

```
foreach run (01 02 03 04 05 06 07 08 09 10)
        3dTstat -prefix mean_r{$run}     \
        ED_r{$run}_vr_bl+orig
end
```

    - Unless otherwise specified, the default statistic for **3dTstat** is to compute a voxel-by-voxel mean
      - Other statistics run by **3dTstat** include a voxel-by-voxel standard deviation, slope, median, etc…

- The end result will be a dataset consisting of a single mean value in each voxel. Below is a graph of a 3x3 voxel matrix from subject ED's dataset **mean_r01+orig**:



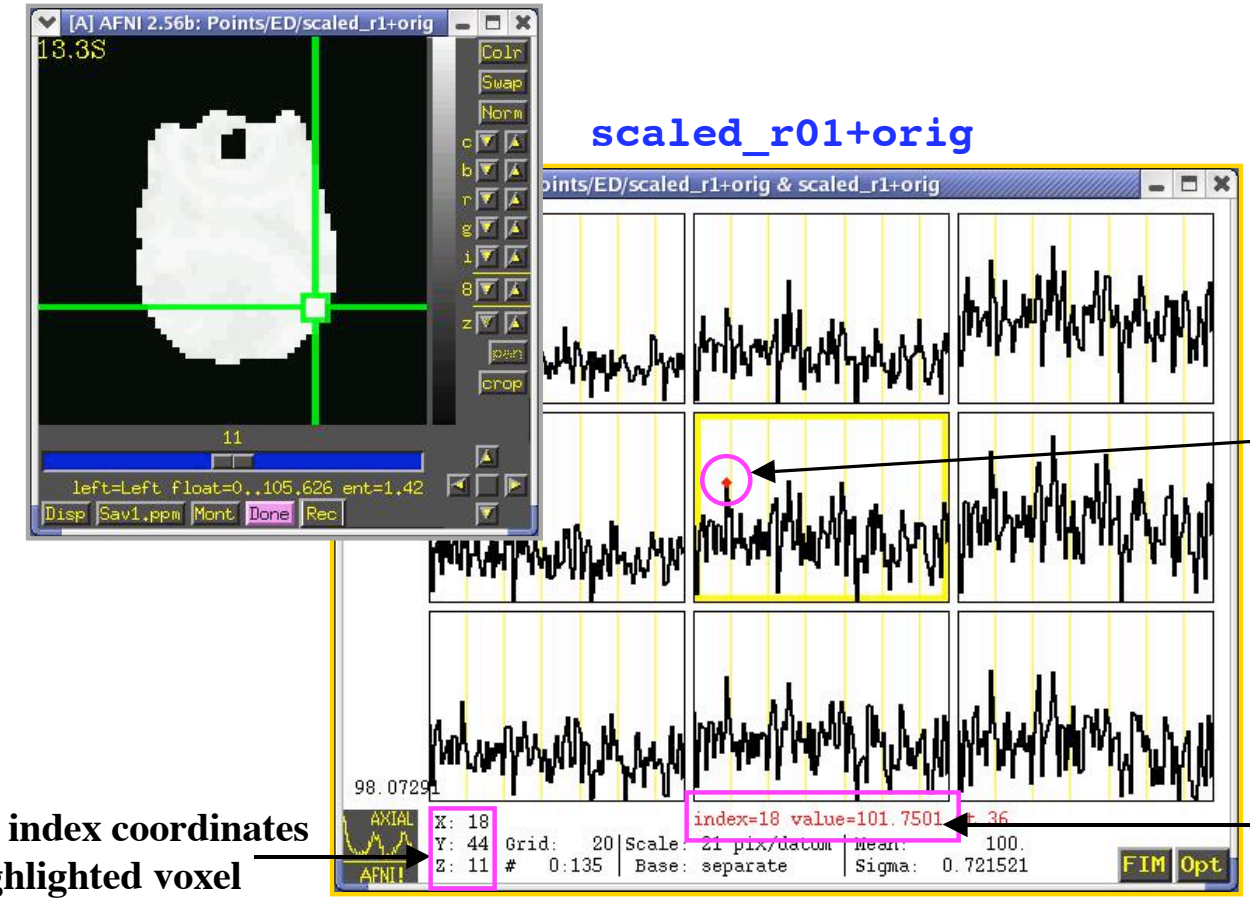
- **STEP 5C: Calculate a voxel-by-voxel percent signal change with `3dcalc`**

  - Take the 136 intensity values within each voxel, divide each one by the *mean* intensity value for that voxel (that we calculated in Step 3B), and multiply by 100 to get a percent signal change at each timepoint
  - This is where the `A/B*100` equation comes into play

```
foreach run (01 02 03 04 05 06 07 08 09 10)
       3dcalc -a ED_r{$run}_vr_bl+orig     \
              -b mean_r{$run}+orig         \
              -c full_mask+orig            \
              -expr "(a/b * 100) * c"      \
              -prefix scaled_r{$run}
end
```

    - Output of `3dcalc`: 10 scaled datasets for Subject ED, where the signal intensity value at each timepoint has now been replaced with a percent signal change value

      `scaled_r01+orig, scaled_r02+orig ... scaled_r10+orig`



Timepoint #18

Displays the timepoint highlighted in center voxel and its percent signal change value

Shows index coordinates for highlighted voxel

- E.g., Timepoint #18 above shows a percent signal change value of 101.7501
- i.e., relative to the baseline (of 100), the stimulus presentation (and noise too) resulted in a percent signal change of 1.7501% at that specific timepoint

- **STEP 6: Concatenate ED's 10 scaled datasets into one big dataset with `3dTcat`**

```
3dTcat -prefix ED_all_runs  \
       scaled_r??+orig.HEAD
```

  - The `??` Takes the place of having to type out each individual run, such as **`scaled_01+orig`**, **`scaled_r02+orig`**, etc. This is a helpful UNIX shortcut. You could also use the wildcard `*`

- The output from **`3dTcat`** is one big dataset -- **`ED_all_runs+orig`** -- which consists of 1360 volumes (i.e., 10 runs x 136 timepoints). Every voxel in this large dataset contains percent signal change values
- This output file will be inserted into the **`3dDeconvolve`** program

  - Do you recall those motion parameter files we created when running **`3dvolreg`**? (No? See page 8 of this handout). We need to concatenate those files too because they will be inserted into the **`3dDeconvolve`** command as Regressors of No Interest (RONI's).
    - The UNIX program **`cat`** will concatenate these ASCII files:

```
cat dfile.r??.1D > dfile.all.1D
```

- **STEP 7: Perform a deconvolution analysis on Subject ED's data with `3dDeconvolve`**

  - What is the difference between regular linear regression and deconvolution analysis?
    - With linear regression, the hemodynamic response is already assumed (we can get a fixed hemodynamic model by running the AFNI **`waver`** program)
    - With deconvolution analysis, the hemodynamic response is not assumed. Instead, it is computed by **`3dDeconvolve`** from the data
      - Once the HRF is modeled by **`3dDeconvolve`**, the program then runs a linear regression on the data
      - To compute the hemodynamic response function with **`3dDeconvolve`**, we include the "**`minlag`**" and "**`maxlag`**" options on the command line
        - The user (you) must determine the lag time of an input stimulus
        - 1 lag = 1 TR = 2 seconds
    - In this example, the lag time of the input stimulus has been determined to be about 15 lags (decided by the wise and all-knowing experimenter)
      - As such, we will add a "minlag" of **0** and a "maxlag" of **14** in our **`3dDeconvolve`** command

- **3dDeconvolve command - Part 1**

Our baseline is quadratic (default is "linear")

Concatenated 10 runs for subject ED

Our stim files for each stim condition

```
3dDeconvolve -polort 2
  -input ED_all_runs+orig -num_stimts 10                    \
  -concat ../misc_files/runs.1D                             \
  -stim_file 1  ../misc_files/all_stims.1D'[0]'             \
       -stim_label 1 ToolMovie                              \
       -stim_minlag 1 0  -stim_maxlag 1 14 -stim_nptr 1 2   \
  -stim_file 2 ../misc_files/all_stims.1D'[1]'              \
       -stim_label 2 HumanMovie                             \
       -stim_minlag 2 0  -stim_maxlag 2 14 -stim_nptr 2 2   \
  -stim_file 3 ../misc_files/all_stims.1D'[2]'              \
       -stim_label 3 ToolPoint                              \
       -stim_minlag 3 0  -stim_maxlag 3 14 -stim_nptr 3 2   \
  -stim_file 4 ../misc_files/all_stims.1D'[3]'              \
       -stim_label 4 HumanPoint                             \
       -stim_minlag 4 0  -stim_maxlag 4 14 -stim_nptr 4 2   \
```

Experimenter estimates the HRF will last ~15 sec

# of stimulus function pts per TR. Default = 1. Here it's set to 2

Continued on next page…

- **3dDeconvolve command - Part 2**

RONI's

RONI's are part of the baseline model

```
-stim_file 5   dfile.all.1D'[0]'  -stim_base 5          \
-stim_file 6   dfile.all.1D'[1]'  -stim_base 6          \
-stim_file 7   dfile.all.1D'[2]'  -stim_base 7          \
-stim_file 8   dfile.all.1D'[3]'  -stim_base 8          \
-stim_file 9   dfile.all.1D'[4]'  -stim_base 9          \
-stim_file 10  dfile.all.1D'[5]'  -stim_base 10         \
-gltsym ../misc_files/contrast1.1D -glt_label 1 FullF   \
-gltsym ../misc_files/contrast2.1D -glt_label 2 HvsT    \
-gltsym ../misc_files/contrast3.1D -glt_label 3 MvsP    \
-gltsym ../misc_files/contrast4.1D -glt_label 4 HMvsHP  \
-gltsym ../misc_files/contrast5.1D -glt_label 5 TMvsTP  \
-gltsym ../misc_files/contrast6.1D -glt_label 6 HPvsTP  \
-gltsym ../misc_files/contrast7.1D -glt_label 7 HMvsTM  \
```

General Linear Tests, "Symbolic" usage.

E.g., `+HumanMovie -ToolMovie`

rather than `-glt` option, e.g., `30@0 1 -1 0 0`

Continued on next page…

- **`3dDeconvolve` command - Part 3**

```
-iresp 1 TMirf
-iresp 2 HMirf
-iresp 3 TPirf
-iresp 4 HPirf
-full_first -fout -tout -nobout
-xpeg Xmat
-bucket ED_func
```

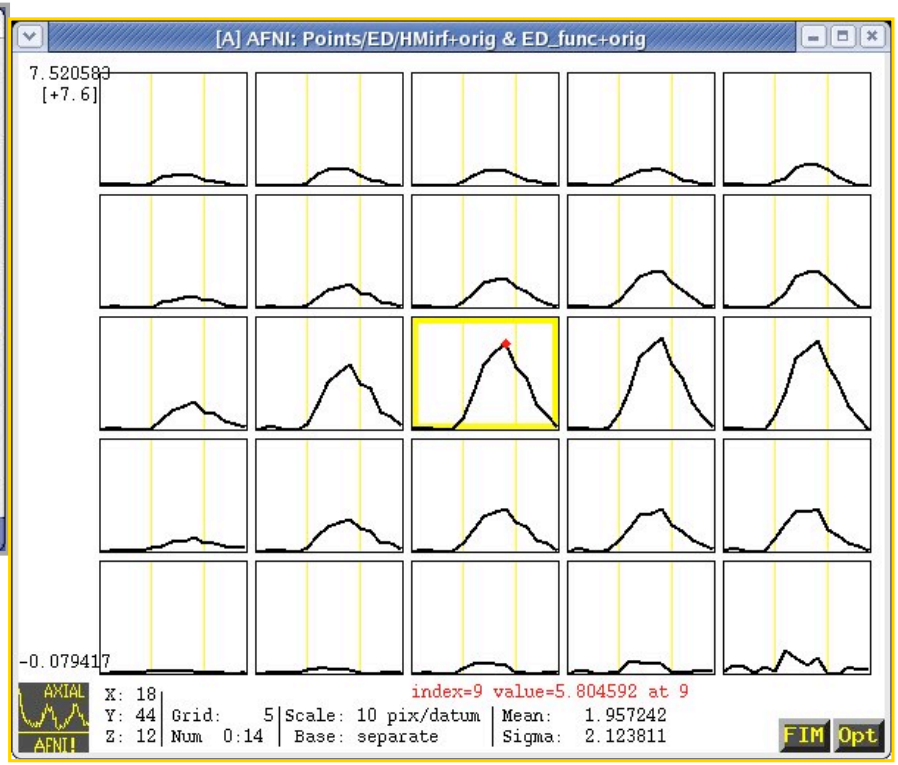
`irf` files show the voxel-by-voxel impulse response function for each stimulus condition. Recall that the IRF was modeled using 'min' and 'max' lag options (more explanation on p.27).

Writes a JPEG file graphing the X matrix

Show Full-F first in bucket dataset, compute F-tests, compute t-tests, don't show output of baseline coefficients in bucket dataset

Done with 3dDeconvolve command

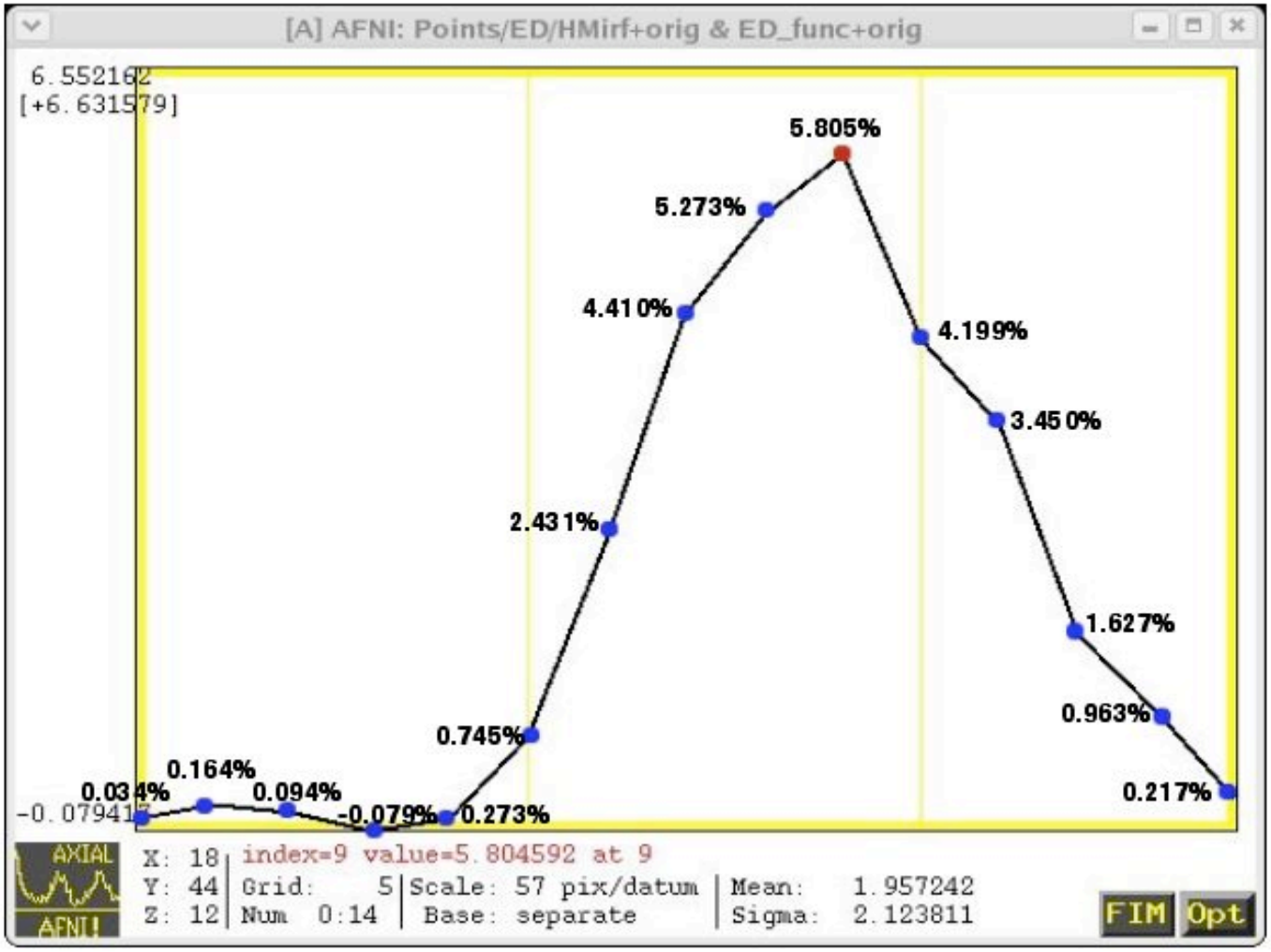
- **`-iresp 1 TMirf`**
- **`-iresp 2 HMirf`**
- **`-iresp 3 TPirf`**
- **`-iresp 4 HPirf`**
  - These output files are important because they contain the estimated Impulse Response Function for each stimulus type
  - The percent signal change is shown at each time lag
- Below is the estimated IRF for Subject ED's "Human Movies" (HM) condition:



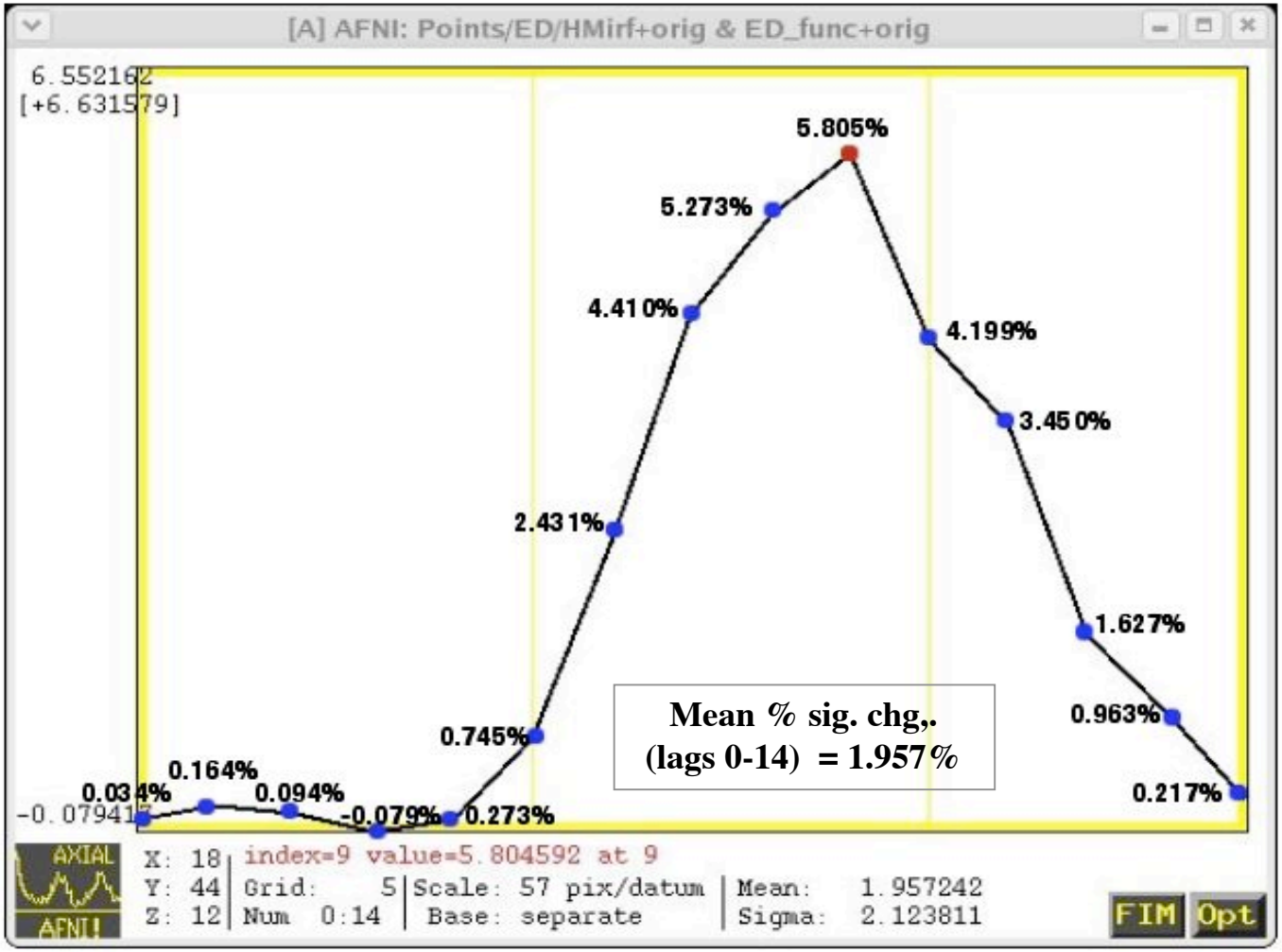
Switch UnderLay: **`HMirf+orig`**

Switch OverLay: **`ED_func+orig`**

- Focusing on a single voxel (from ED's **`HMirf+orig`** dataset), we can see that the IRF is made up of 15 time lags (0-14). Recall that this lag duration was determined in the **`3dDeconvolve`** command
- Each time lag consists of a percent signal change value:



- To run an ANOVA, only one data point can exist in each voxel
  - As such, the percent signal change values in the 15 lags must be averaged
  - In the voxel displayed below, the **mean percent signal change = 1.957%**



- **STEP 8: Use AFNI `3dbucket` to slim down the functional dataset bucket, i.e., create a mini-bucket that contains only the sub-bricks you're interested in**

  - There are 152 sub-bricks in **`ED_func+orig.BRIK`**. Select the most relevant ones for further analysis and ignore the rest for now.

```
#  0 Full F-stat          # 35 HumanMovie[1] t-st # 70 ToolPoint[3] t-st  #105 HumanPoint[5] t-st #140 HMvsHP LC[0] coef
#  1 ToolMovie[0] Coef    # 36 HumanMovie[2] Coef # 71 ToolPoint[4] Coef  #106 HumanPoint[6] Coef #141 HMvsHP LC[0] t-st
#  2 ToolMovie[0] t-st    # 37 HumanMovie[2] t-st # 72 ToolPoint[4] t-st  #107 HumanPoint[6] t-st #142 HMvsHP F-stat
#  3 ToolMovie[1] Coef    # 38 HumanMovie[3] Coef # 73 ToolPoint[5] Coef  #108 HumanPoint[7] Coef #143 TMvsTP LC[0] coef
#  4 ToolMovie[1] t-st    # 39 HumanMovie[3] t-st # 74 ToolPoint[5] t-st  #109 HumanPoint[7] t-st #144 TMvsTP LC[0] t-st
#  5 ToolMovie[2] Coef    # 40 HumanMovie[4] Coef # 75 ToolPoint[6] Coef  #110 HumanPoint[8] Coef #145 TMvsTP F-stat
#  6 ToolMovie[2] t-st    # 41 HumanMovie[4] t-st # 76 ToolPoint[6] t-st  #111 HumanPoint[8] t-st #146 HPvsTP LC[0] coef
#  7 ToolMovie[3] Coef    # 42 HumanMovie[5] Coef # 77 ToolPoint[7] Coef  #112 HumanPoint[9] Coef #147 HPvsTP LC[0] t-st
#  8 ToolMovie[3] t-st    # 43 HumanMovie[5] t-st # 78 ToolPoint[7] t-st  #113 HumanPoint[9] t-st #148 HPvsTP F-stat
#  9 ToolMovie[4] Coef    # 44 HumanMovie[6] Coef # 79 ToolPoint[8] Coef  #114 HumanPoint[10] Coef #149 HMvsTM LC[0] coef
# 10 ToolMovie[4] t-st    # 45 HumanMovie[6] t-st # 80 ToolPoint[8] t-st  #115 HumanPoint[10] t-st #150 HMvsTM LC[0] t-st
# 11 ToolMovie[5] Coef    # 46 HumanMovie[7] Coef # 81 ToolPoint[9] Coef  #116 HumanPoint[11] Coef #151 HMvsTM F-stat
# 12 ToolMovie[5] t-st    # 47 HumanMovie[7] t-st # 82 ToolPoint[9] t-st  #117 HumanPoint[11] t-st
# 13 ToolMovie[6] Coef    # 48 HumanMovie[8] Coef # 83 ToolPoint[10] Coef #118 HumanPoint[12] Coef
# 14 ToolMovie[6] t-st    # 49 HumanMovie[8] t-st # 84 ToolPoint[10] t-st #119 HumanPoint[12] t-st
# 15 ToolMovie[7] Coef    # 50 HumanMovie[9] Coef # 85 ToolPoint[11] Coef #120 HumanPoint[13] Coef
# 16 ToolMovie[7] t-st    # 51 HumanMovie[9] t-st # 86 ToolPoint[11] t-st #121 HumanPoint[13] t-st
# 17 ToolMovie[8] Coef    # 52 HumanMovie[10] Coef# 87 ToolPoint[12] Coef #122 HumanPoint[14] Coef
# 18 ToolMovie[8] t-st    # 53 HumanMovie[10] t-st# 88 ToolPoint[12] t-st #123 HumanPoint[14] t-st
# 19 ToolMovie[9] Coef    # 54 HumanMovie[11] Coef# 89 ToolPoint[13] Coef #124 HumanPoint F-stat
# 20 ToolMovie[9] t-st    # 55 HumanMovie[11] t-st# 90 ToolPoint[13] t-st #125 FullF LC[0] coef
# 21 ToolMovie[10] Coef   # 56 HumanMovie[12] Coef# 91 ToolPoint[14] Coef #126 FullF LC[0] t-st
# 22 ToolMovie[10] t-st   # 57 HumanMovie[12] t-st# 92 ToolPoint[14] t-st #127 FullF LC[1] coef
# 23 ToolMovie[11] Coef   # 58 HumanMovie[13] Coef# 93 ToolPoint F-stat   #128 FullF LC[1] t-st
# 24 ToolMovie[11] t-st   # 59 HumanMovie[13] t-st# 94 HumanPoint[0] Coef #129 FullF LC[2] coef
# 25 ToolMovie[12] Coef   # 60 HumanMovie[14] Coef# 95 HumanPoint[0] t-st #130 FullF LC[2] t-st
# 26 ToolMovie[12] t-st   # 61 HumanMovie[14] t-st# 96 HumanPoint[1] Coef #131 FullF LC[3] coef
# 27 ToolMovie[13] Coef   # 62 HumanMovie F-stat  # 97 HumanPoint[1] t-st #132 FullF LC[3] t-st
# 28 ToolMovie[13] t-st   # 63 ToolPoint[0] Coef  # 98 HumanPoint[2] Coef #133 FullF F-stat
# 29 ToolMovie[14] Coef   # 64 ToolPoint[0] t-st  # 99 HumanPoint[2] t-st #134 HvsT LC[0] coef
# 30 ToolMovie[14] t-st   # 65 ToolPoint[1] Coef  #100 HumanPoint[3] Coef #135 HvsT LC[0] t-st
# 31 ToolMovie F-stat     # 66 ToolPoint[1] t-st  #101 HumanPoint[3] t-st #136 HvsT F-stat
# 32 HumanMovie[0] Coef   # 67 ToolPoint[2] Coef  #102 HumanPoint[4] Coef #137 MvsP LC[0] coef
# 33 HumanMovie[0] t-st   # 68 ToolPoint[2] t-st  #103 HumanPoint[4] t-st #138 MvsP LC[0] t-st
# 34 HumanMovie[1] Coef   # 69 ToolPoint[3] Coef  #104 HumanPoint[5] Coef #139 MvsP F-stat
```

```
3dbucket -prefix ED_func_slim \
         -fbuc ED_func+orig'[0,125..151]'
```

- **STEP 9: Compute a voxel-by-voxel mean percent signal change with AFNI `3dTstat`**

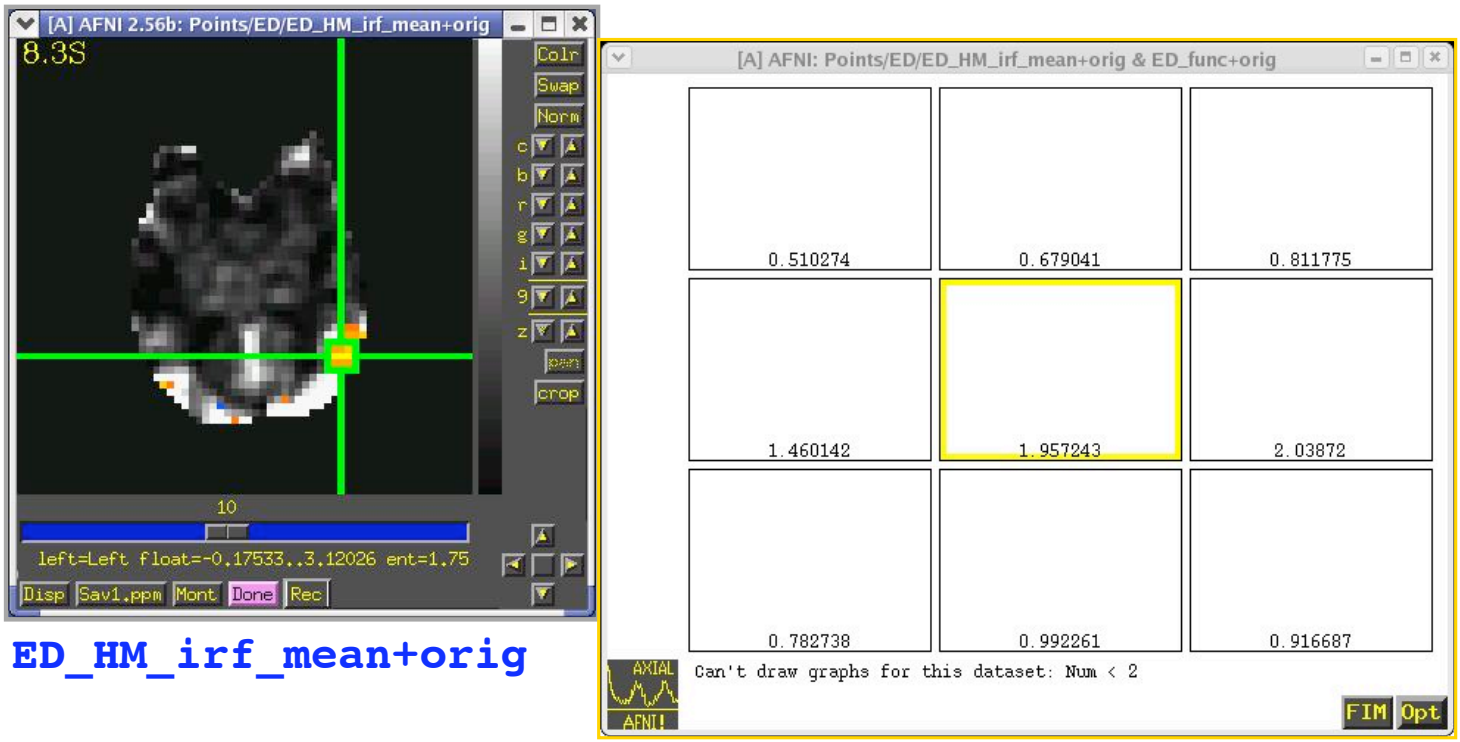
  - The following **`3dTstat`** command will compute a voxel-by-voxel mean for each IRF dataset, of which we have four: **`TMirf`**, **`HMirf`**, **`TPirf`**, **`HPirf`**

```
foreach cond (TM HM TP HP)
        3dTstat -prefix ED_{$cond}_irf_mean  \
                {$cond}irf+orig
end
```

➥ The output from **3dTstat** will be four `irf_mean` datasets, one for each stimulus type. Below are subject ED's averaged IRF datasets:

ED_TM_irf_mean+orig **ED_HM_irf_mean+orig**
ED_TP_irf_mean+orig **ED_HP_irf_mean+orig**

✧ Each voxel will now contain a single number (i.e., the mean percent signal change). For example:



ED_HM_irf_mean+orig

- **STEP 10: Resample the mean IRF datasets for each subject to the same grid as their Talairached anatomical datasets with `adwarp`**
  - For statistical comparisons made across subjects, all datasets -- including functional overlays -- should be standardized (e.g., Talairach format) to control for variability in brain shape and size

```
foreach cond (TM HM TP HP)
       adwarp -apar EDspgr+tlrc  -dxyz 3       \
              -dpar ED_{$cond}_irf_mean+orig  \
end
```

  - The output of **`adwarp`** will be four Talairach transformed IRF datasets.

    `ED_TM_irf_mean+tlrc` `ED_HM_irf_mean+tlrc`

    `ED_TP_irf_mean+tlrc` `ED_HP_irf_mean+tlrc`

- We are now done with Part 1-- Process Individual Subjects' Data -- for Subject ED
  - Go back and follow the same steps for remaining 6 subjects
- We can now move on to Part 2 -- RUN GROUP ANALYSIS (ANOVA)

- **PART 2 ⇒ Run Group Analysis (ANOVA3):**
  - In our sample experiment, we have 3 factors (or Independent Variables) for our analysis of variance: “Stimulus Condition” and “Subjects”

    - IV 1: OBJECT TYPE ⇒ 2 levels
      - Tools (T)
      - Humans (H)

    - IV 2: ANIMATION TYPE ⇒ 2 levels
      - Movies (M)
      - Point-light displays (P)

    - IV 3: SUBJECTS ⇒ 7 levels *(note: this is a small sample size!)*
      - Subjects ED, EE, EF, FH, FK, FL, FN

  - The mean IRF datasets from each subject will be needed for the ANOVA. Example:

```
ED_TM_irf_mean+tlrc   EE_TM_irf_mean+tlrc   EF_TM_irf_mean+tlrc
ED_HM_irf_mean+tlrc   EE_HM_irf_mean+tlrc   EF_HM_irf_mean+tlrc
ED_TP_irf_mean+tlrc   EE_TP_irf_mean+tlrc   EF_TP_irf_mean+tlrc
ED_HP_irf_mean+tlrc   EE_HP_irf_mean+tlrc   EF_HP_irf_mean+tlrc
```

- **3dANOVA3 Command - Part 1**

IV's A & B are fixed, C is random.
See `3dANOVA3 -help`

```
3dANOVA3 -type 4                                  \
         -alevels 2                               \
         -blevels 2                               \
         -clevels 7                               \
         -dset 1 1 1 ED_TM_irf_mean+tlrc \
         -dset 2 1 1 ED_HM_irf_mean+tlrc \
         -dset 1 2 1 ED_TP_irf_mean+tlrc \
         -dset 2 2 1 ED_HP_irf_mean+tlrc \
         -dset 1 1 2 EE_TM_irf_mean+tlrc \
         -dset 2 1 2 EE_HM_irf_mean+tlrc \
         -dset 1 2 2 EE_TP_irf_mean+tlrc \
         -dset 2 2 2 EE_HP_irf_mean+tlrc \
         -dset 1 1 3 EF_TM_irf_mean+tlrc \
         -dset 2 1 3 EF_HM_irf_mean+tlrc \
         -dset 1 2 3 EF_TP_irf_mean+tlrc \
         -dset 2 2 3 EF_HP_irf_mean+tlrc \
```

IV A: Object

IV B: Animation

IV C: Subjects

`irf` datasets, created for each subj with 3dDeconvolve (See p.26)

Continued on next page…

- **3dANOVA3 Command - Part 2**

```
-dset 1 1 4 FH_TM_irf_mean+tlrc \
-dset 2 1 4 FH_HM_irf_mean+tlrc \
-dset 1 2 4 FH_TP_irf_mean+tlrc \
-dset 2 2 4 FH_HP_irf_mean+tlrc \
-dset 1 1 5 FK_TM_irf_mean+tlrc \
-dset 2 1 5 FK_HM_irf_mean+tlrc \
-dset 1 2 5 FK_TP_irf_mean+tlrc \
-dset 2 2 5 FK_HP_irf_mean+tlrc \
-dset 1 1 6 FL_TM_irf_mean+tlrc \
-dset 2 1 6 FL_HM_irf_mean+tlrc \
-dset 1 2 6 FL_TP_irf_mean+tlrc \
-dset 2 2 6 FL_HP_irf_mean+tlrc \
-dset 1 1 7 FN_TM_irf_mean+tlrc \
-dset 2 1 7 FN_HM_irf_mean+tlrc \
-dset 1 2 7 FN_TP_irf_mean+tlrc \
-dset 2 2 7 FN_HP_irf_mean+tlrc \
```

more `irf` datasets

Continued on next page…

- **3dANOVA3 Command - Part 3**

Produces main effect for factor 'a' (Object type). I.e., which voxels show increases in % signal change that is sig. Different from zero?

Main effect for factor 'b', (Animation type)

```
-fa ObjEffect                    \
-fb AnimEffect                   \
-adiff 1 2 TvsH                  \
-bdiff 1 2 MvsP                  \
-acontr 1 -1 sameas.TvsH         \
-bcontr 1 -1 sameas.MvsP         \
-aBcontr 1 -1: 1 TMvsHM          \
-aBcontr -1 1: 2 HPvsTP          \
-Abcontr 1: 1 -1 TMvsTP          \
-Abcontr 2: 1 -1 HMvsHP          \
-bucket AvgAnova
```

These are contrasts (t-tests). Explained on pp 38-39

All F-tests, t-tests, etc will go into this dataset bucket

End of ANOVA command

- **-adiff**: Performs contrasts between levels of factor ‘a’ (or `-bdiff` for factor ‘b’, `-cdiff` for factor ‘c’, etc), with *no* collapsing across levels of factor ‘a’.

  E.g.1, Factor “Object Type” --> 2 levels: (1)Tools, (2)Humans:

  ```
  -adiff 1 2 TvsH
  ```

  E.g., 2, Factor “Faces” --> 3 levels: (1)Happy, (2)Sad, (3)Neutral

  ```
  -adiff 1 2 HvsS
  -adiff 2 3 SvsN
  -adiff 1 3 HvsN
  ```

  Simple paired t-tests, no collapsing across levels, like Happy vs. Sad/Neutral

- **-acontr**: Estimates contrasts among levels of factor ‘a’ (or `-bcontr` for factor ‘b’, `-ccontr` for factor ‘c’, etc). *Allows* for collapsing across levels of factor ‘a’
  - In our example, since we only have 2 levels for both factors ‘a’ and ‘b’, the **-diff** and **-contr** options can be used interchangeably. Their different usages can only be demonstrated with a factor that has 3 or more levels:
  - E.g.: factor ‘a’ = FACES --> 3 levels :(1) Happy, (2) Sad, (3) Neutral

  ```
  -acontr -1 .5 .5 HvsSN   ← Happy vs. Sad/Neutral
  -acontr .5 .5 -1 HSvsN   ← Happy/Sad vs. Neutral
  -acontr .5 -1 .5 HNvsS   ← Happy/Neutral vs. Sad
  ```

- **`-aBcontr`**: 2nd order contrast. Performs comparison between 2 levels of factor **'a'** at a *Fixed* level of factor **'B'**
  - E.g. factor **'a'** --> Tools(1) vs. Humans(-1),
    factor **'B'** --> Movies(1) vs. Points(2)
    - We want to compare 'Tools Movies' vs. 'Human Movies'. Ignore 'Points'

      `-aBcontr 1 -1 : 1 TMvsHM`
    - We want to compare "Tool Points' vs. 'Human Points'. Ignore 'Movies'

      `-aBcontr 1 -1 : 2 TPvsHP`

- **`-Abcontr`**: 2nd order contrast. Performs comparison between 2 levels of factor **'b'** at a *Fixed* level of factor **'A'**
  - E.g., E.g. factor **'b'** --> Movies(1) vs. Points(-1),
    factor **'A'** --> Tools(1) vs. Humans(2)
    - We want to compare 'Tools Movies' vs. 'Tool Points'. Ignore 'Humans

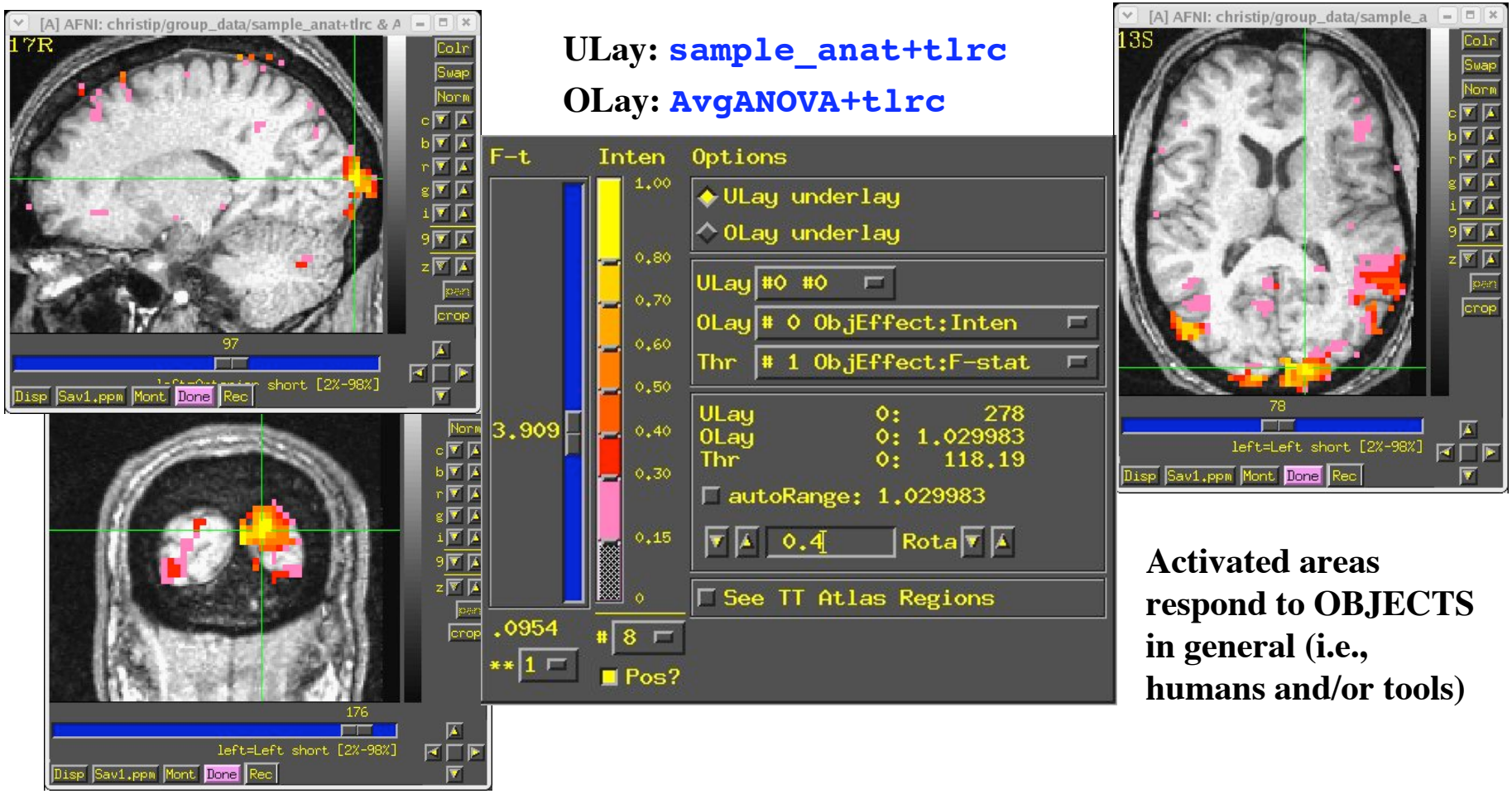
      `-Abcontr 1 : 1 -1 TMvsTP`
    - We want to compare "Human Movies vs. 'Human Points'. Ignore 'Tools'

      `-Abcontr 2 : 1 -1 HMvsHP`

- **In class -- Let's run the ANOVA together**:
  - **cd AFNI_data2**
    - This directory contains a script called `s3.anova.ht05` that will run `3dANOVA3`
    - This script can be viewed with a text editor, like **emacs**
  - **./s3.anova.ht05**
    - execute the ANOVA script from the command line
  - **cd group_data ; ls**
    - result from ANOVA script is a bucket dataset AvgANOVA+tlrc, stored in the group_data/ directory
  - **afni &**
    - launch AFNI to view the results

- The output from **3dANOVA3** is bucket dataset **AvgANOVA+tlrc**, which contains 20 sub-bricks of data:
  - i.e., main effect F-tests for factors A and B, 1st order contrasts, and 2nd order contrasts

- `-fa`: Produces a main effect for factor ‘a’
  - In this example, `-fa` determines which voxels show a percent signal change that is significantly different from zero when any level of factor “Object Type” is presented
  - **`-fa ObjEffect`**:



ULay: **`sample_anat+tlrc`**

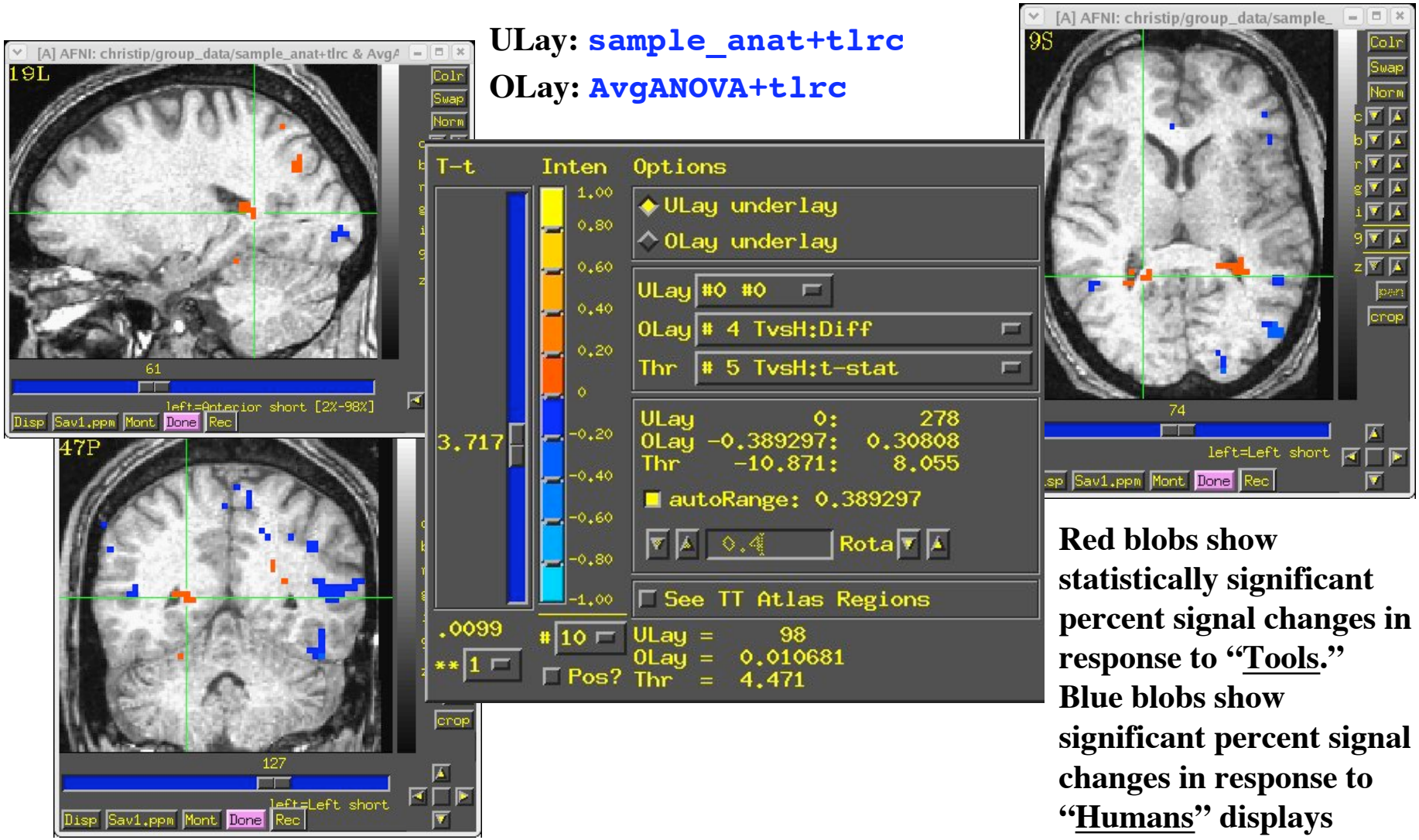
OLay: **`AvgANOVA+tlrc`**

Activated areas respond to OBJECTS in general (i.e., humans and/or tools)

- Brain areas corresponding to “Tools” (reds) vs. “Humans” (blues)
  - **`-diff 1 2 TvsH`** (or **`-acontr 1 -1 TvsH`**)

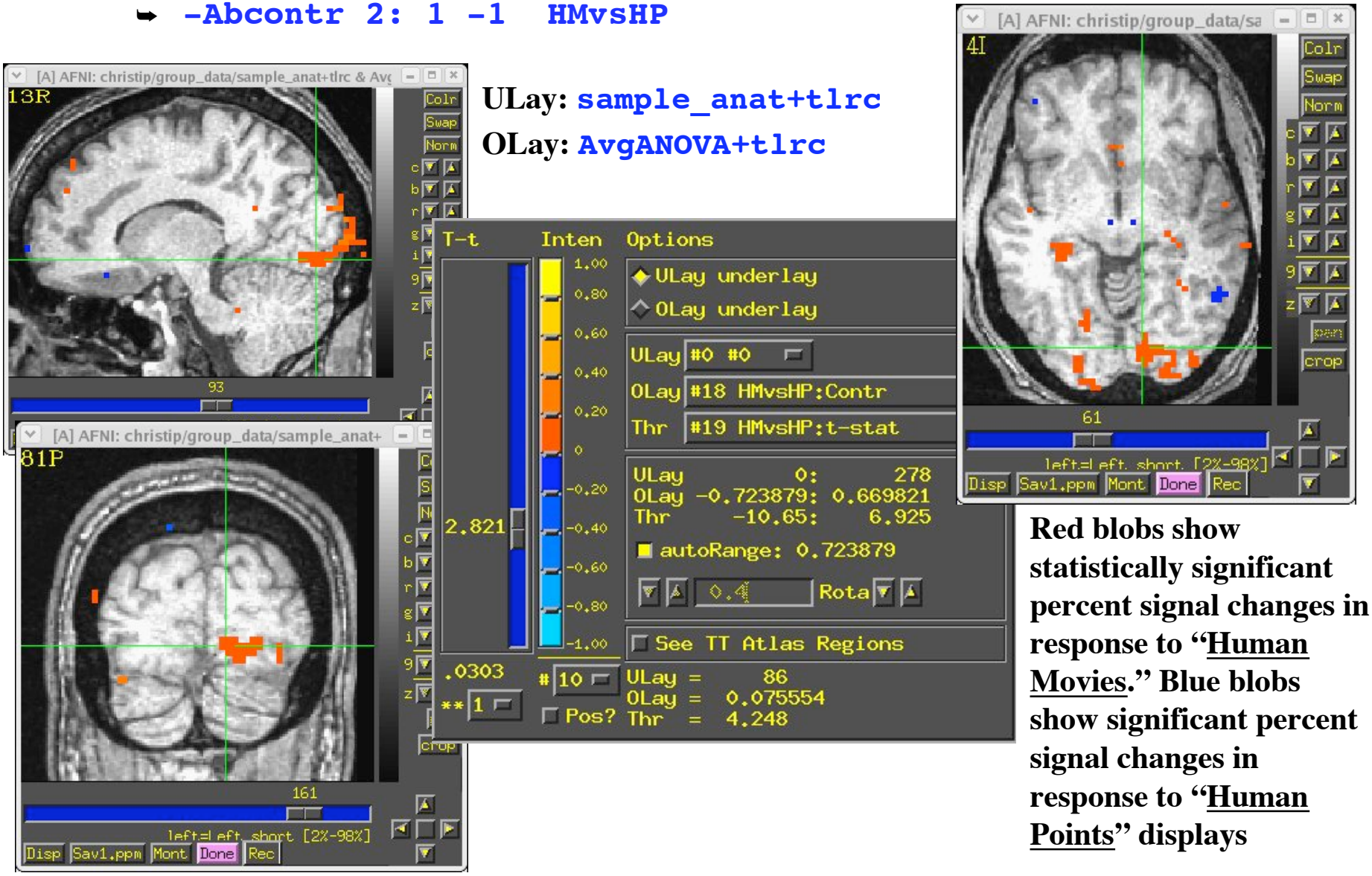
ULay: **`sample_anat+tlrc`**

OLay: **`AvgANOVA+tlrc`**



Red blobs show statistically significant percent signal changes in response to “<u>Tools</u>.” Blue blobs show significant percent signal changes in response to “<u>Humans</u>” displays

- Brain areas corresponding to “Human Movies” (reds) vs. “Humans Points” (blues)
  - **-Abcontr 2: 1 -1 HMvsHP**



ULay: sample_anat+tlrc

OLay: AvgANOVA+tlrc

Red blobs show statistically significant percent signal changes in response to “Human Movies.” Blue blobs show significant percent signal changes in response to “Human Points” displays

- Many thanks to Mike Beauchamp for donating the data used in this lecture and in the how-to#5

- For a full review of the experiment described in this lecture, see

  Beauchamp, M.S., Lee, K.E., Haxby, J.V., & Martin, A. (2003). FMRI responses to video and point-light displays of moving humans and manipulable objects. Journal of Cognitive Neuroscience, 15:7, 991-1001.

- For more information on AFNI ANOVA programs, visit the web page of Gang Chen, our wise and infinitely patient statistician:

  http//afni.nimh.gov/sscc/gangc