Advanced Adventures in FMRI Time Series Analysis

On the off chance that you weren’t confused enough already

IM Regression

- **IM** = **I**ndividual **M**odulation
  - Compute *separate* amplitude of response for each stimulus time given in input file
    - Instead of computing average amplitude of responses to multiple stimuli in the same class
  - Response amplitudes (βs) for each individual block/event will be highly noisy
    - Can't use individual activation map for much
    - Must pool the computed βs in some further statistical analysis (*t*-test via **`3dttest`**? inter-voxel correlations in the βs? correlate βs with something else?)
  - Usage: **`-stim_times_IM k tname model`**
    - Like **`-stim_times`**, but creates a separate regression matrix column for each time given

AM Regression - 1

- **AM** = **A**mplitude **M**odulated (SPM: Parametric Modulation)
  - Have extra data measured about *each* response to a stimulus
    - Reaction time; Galvanic skin response; Pain level perception; …
- Want to find active voxels whose activation level also depends on ABI
  - `3dDeconvolve` is a linear program, so must assume that change in FMRI signal as ABI changes is proportional to change in ABI values
- Need to make 2 separate regressors
  - One to find the mean FMRI response (the usual `-stim_times` analysis)
  - One to find the variations in the FMRI response as the ABI data varies; Second regressor has the form $r_{\text{AM2}}(t) = \sum_{k=1}^{K} h(t-\tau_k)\cdot(a_k - \bar{a})$
  - Where a_k =value of k th ABI value, and $\bar{a}$ = average ABI value
- Response (β) for first regressor is standard activation map
- Statistics and β for second regressor make activation map of places whose BOLD response changes with changes in ABI
  - Using 2 regressors allows separation of voxels that are active but are *not* detectably modulated by the ABI from voxels that *are* ABI-sensitive

AM Regression - 2

- New-ish feature of **3dDeconvolve**: **-stim_times_AM2**
- Usage is very similar to standard **-stim_times**
  - **-stim_times_AM2 1 times_ABI.1D 'BLOCK(2,1)'**
  - **times_ABI.1D** file has time entries that are “married” to ABI values:

```
10*5 23*4 27*2 39*5
17*2 32*5
*
16*2 24*3 37*5 41*4
```

  - Such files can be created from 2 standard ASCII .1D files using the **1dMarry** program
    - The **-divorce** option can be used to split them up
- **3dDeconvolve** automatically creates the two regressors (unmodulated and amplitude modulated)
  - Use **-fout** option to get statistics for activation of pair of regressors (i.e., testing null hypothesis that *both* β weights are zero: that there is no ABI-independent *or* ABI-proportional signal change)
  - Use **-tout** option to test each β weight separately
  - Can **1dplot** **X** matrix columns to see each regressor
  - Can have more than one ABI parameter per event (polygamy?)

AM Regression - 3

- Alternative to **AM**: use **IM** to get individual βs for each block/event and then do external regression and/or statistics on those values
- Could do nonlinear fitting (to these βs) via **`3dNLfim`**, or inter-class contrasts via **`3dttest`**, **`3dLME`**, **`3dANOVA`**, or intra-class correlations via **`3dICC`**, etc.
- What is better: **AM** or **IM**+*something more* ?
  - We don't know – experience with these options is limited – you can always try both!
  - If **AM** doesn't fit your models/ideas, then **IM**+ is clearly the way to go
  - Probably need to consult with SSCC to get some hints/advice

DM Regression

- Solving a visually presented puzzle:
  a) subject sees puzzle
  b) subject cogitates a while
  c) subject responds with solution

timing of events is measured

- We expect some voxels to be significant in phase (b) as well as phases (a) and/or (c) – (b) is probably what you care about!
- Variable length of phase (b) means that shape for its response varies between trials
  - Which is contrary to the whole idea of averaging trials together to get decent statistics
- Could assume response ***amplitude*** in phase (b) is constant across trials, and response ***duration*** of (b) equals the time between phases (a) and (c)
  - Need to use three HRFs
  - HRF (b): use the `dmBLOCK` response function
  - Can combine **DM** with **AM** (or **IM**) if needed

Allowing for Serial Correlation

- *t*- and *F*-statistics denominators: estimates of noise variance
  - White noise estimate of variance:

$$\hat{\sigma}^2 = \frac{1}{N-m}\sum_{i=0}^{N-1}[\text{data}_i - \text{fit}_i]^2$$

    - *N* = number of time points
    - *m* = number of fit parameters
    - *N*−*m* = degrees of freedom = how many equal-variance independent random values are left after time series is fit with *m* regressors
- **Problem**: if noise values at successive time points are correlated, this estimate of variance is biased to be too small, since there aren't really *N*−*m* independent random values left
  - Denominator too small implies *t*- and *F*-statistics are too large!
  - And number of degrees of freedom is also too large.
  - So significance (*p*-value) of activations in individuals is overstated.
- **Solution #1**: estimate correlation structure of noise and then adjust statistics (downwards) appropriately
- **Solution #2**: estimate correlation structure of noise ***and*** also estimate β fit parameters using more efficient "generalized least squares", using this correlation, all at once

AFNI's Program: **`3dREMLfit`**

- Implements Solution #2
  - REML is a method for simultaneously estimating variance + correlation parameters ***and*** estimating regression fit parameters (βs)
  - Correlation structure of noise is ARMA(1,1)
    - 2 parameters ***a*** (AR) and ***b*** (MA) in each voxel
      - ***a*** describes how fast the noise de-correlates over time
      - ***b*** describes the short-range correlation in time (1 lag)
    - Unlike SPM and FSL, *each voxel* gets a separate estimate of its own correlation parameters
- Inputs to **`3dREMLfit`**
  - run **`3dDeconvolve`** first to setup **`.xmat.1D`** matrix file and GLTs (don't have to let **`3dDeconvolve`** finish analysis: **`-x1D_stop`**)
    - **`3dDeconvolve`** also outputs a command line to run **`3dREMLfit`**
  - then, input matrix file and 3D+time dataset to **`3dREMLfit`**
- Output datasets are structured as if from **`3dDeconvolve`**

Nonlinear Regression

- Linear models aren't the only possibility
  - e.g., could try to fit HRF of the form $h(t) = a \cdot t^{b} \cdot e^{-t/c}$
  - Unknowns b and c appear nonlinearly in this formula
- Program `3dNLfim` can do nonlinear regression (including nonlinear deconvolution)
  - User must provide a C function that computes the model time series, given a set of parameters (e.g., a, b, c)
    - We could help you develop this C model function
    - Several sample model functions in the **AFNI** source code distribution
  - Program then drives this C function repeatedly, searching for the set of parameters that best fit each voxel
  - Has been used to fit pharmacological models to FMRI data acquired during pharmacological challenges
    - e.g., injection of nicotine, cocaine, ethanol, etc.
      - these are difficult experiments to do ***and*** to analyze
    - e.g., <u>D</u>ynamic <u>C</u>ontrast <u>E</u>nhanced MRI (for brain tumor analyses)

Deconvolution: The Other Direction

- Signal model: $Z(t) = H(t) * A(t)$ + baseline model + noise
- $H(t)$ = HRF = response magnitude t seconds after activation
  - $H(t)$ is **causal** = zero for $t < 0$
  - “$*$” is symbol for convolution, not multiplication!
- **`3dDeconvolve`**: find out something about $H(t)$ given $A(t)$
- Sometimes (**PPI, DSC**) want to solve the problem in the other direction: assume a model for $H(t)$ and find time series $A(t)$
  - Convolution is commutative: $H(t)*A(t) = A(t)*H(t)$
  - So the other direction looks to be the same problem
  - But isn't, since $H(t)$ is causal but $A(t)$ is not
    - Also, $H(t)*A(t)$ smooths out rough spots in $A(t)$, so un-doing this deconvolution adds roughness — including noise, which is already rough — which must be controlled or output $A(t)$ will be horrible junk
- Program **`3dTfitter`** can solve this type of problem
  - Also can allow for *per voxel* model components
  - Unlike **`3dDeconvolve`**, where each voxel has same model

Multi-Voxel Statistics

Spatial Clustering & False Discovery Rate:

“Correcting” the Significance

Basic Problem

- Usually have 50-200K FMRI voxels in the brain
- Have to make at least one decision about each one:
  - Is it “active”?
    - That is, does its time series match the temporal pattern of activity we expect?
  - Is it differentially active?
    - That is, is the BOLD signal change in task #1 different from task #2?
- Statistical analysis is designed to control the error rate of these decisions
  - Making ***lots*** of decisions: hard to get perfection in statistical testing

Two Approaches to the "Curse of Multiple Comparisons"

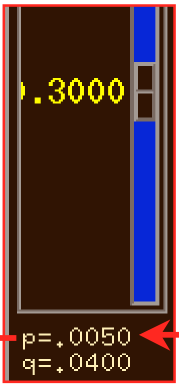
- Control FWE to keep expected total number of false positives below 1
  - Overall significance: α_{FW} = Prob(≥ one false positive voxel in whole brain)
  - **Bonferroni correction**: $\alpha_{\text{FW}} = 1-(1-p)^N \approx Np$, if $p << N^{-1}$
    - Use $p = \alpha/N$ as individual voxel significance level to achieve $\alpha_{\text{FW}} = \alpha$
    - Too stringent and overly conservative: $p = 10^{-8}\ldots10^{-6}$
  - What can rescue us from this hell of statistical super-conservatism?
    - Correlation: Voxels in the brain are not independent
      - Especially after we smooth them together!
      - Means that Bonferroni correction is *way* ***way*** ***way*** too stringent
    - Contiguity: Structures in the brain activation map
      - We are looking for activated "blobs": the chance that pure noise (H_0) will give a set of seemingly-activated voxels next to each other is lower than getting false positives that are scattered around far apart
    - Control FWE based on spatial correlation (smoothness of image noise) **and** minimum cluster size we are willing to accept
- Control false discovery rate (FDR) — More on this a little later!
  - FDR = expected proportion of false positive voxels among all detected voxels
    - Give up on the idea of having (almost) no false positives at all

Cluster Analysis: `3dClustSim`

- **FWE control in AFNI**
  - Monte Carlo simulations with program `3dClustSim` [supersedes `AlphaSim`]
    - o Named for a place where primary attractions are randomization experiments
    - o Randomly generate some number (*e.g.*, 10,000) of brain volumes with white noise (spatially uncorrelated)
      - That is, each "brain" volume is purely in $\mathbf{H_0}$ = no activation
      - Noise images can be blurred to mimic the smoothness of real data
    - o Count number of voxels that are false positives in each simulated volume
      - Including how many are false positives that are spatially together in clusters of various sizes (1, 2, 3, …)
    - o Parameters to program
      - Size of dataset to simulate
      - Mask (e.g., to consider only brain-shaped regions in the simulated 3D brick)
      - Spatial correlation FWHM: from `3dBlurToFWHM` or `3dFWHMx`
      - Connectivity radius: how to identify voxels belonging to a cluster?
        - Default = NN connection = touching faces
      - Individual voxel significance level = uncorrected *p*-value
    - o Output
      - Simulated (estimated) overall significance level (corrected *p*-value ≡ α)
      - Corresponding minimum cluster size at the input uncorrected *p*-value

- **Example:** `3dClustSim -nxyz 64 64 30 -dxyz 3 3 3 -fwhm 7`

```
# 3dClustSim -nxyz 64 64 30 -dxyz 3 3 3 -fwhm 7
# Grid: 64x64x30 3.00x3.00x3.00 mm^3 (122880 voxels)
# CLUSTER SIZE THRESHOLD(pthr,alpha) in Voxels
# -NN 1  | alpha = Prob(Cluster >= given size)
#  pthr  |  0.100  0.050  0.020  0.010
# ------ | ------ ------ ------ ------
 0.020000    89.4   99.9  114.0  123.0
 0.010000    56.1   62.1   70.5   76.6
 0.005000    38.4   43.3   49.4   53.6
 0.002000    25.6   28.8   33.3   37.0
 0.001000    19.7   22.2   26.0   28.6
 0.000500    15.5   17.6   20.5   22.9
 0.000200    11.5   13.2   16.0   17.7
 0.000100     9.3   10.9   13.0   14.8
```



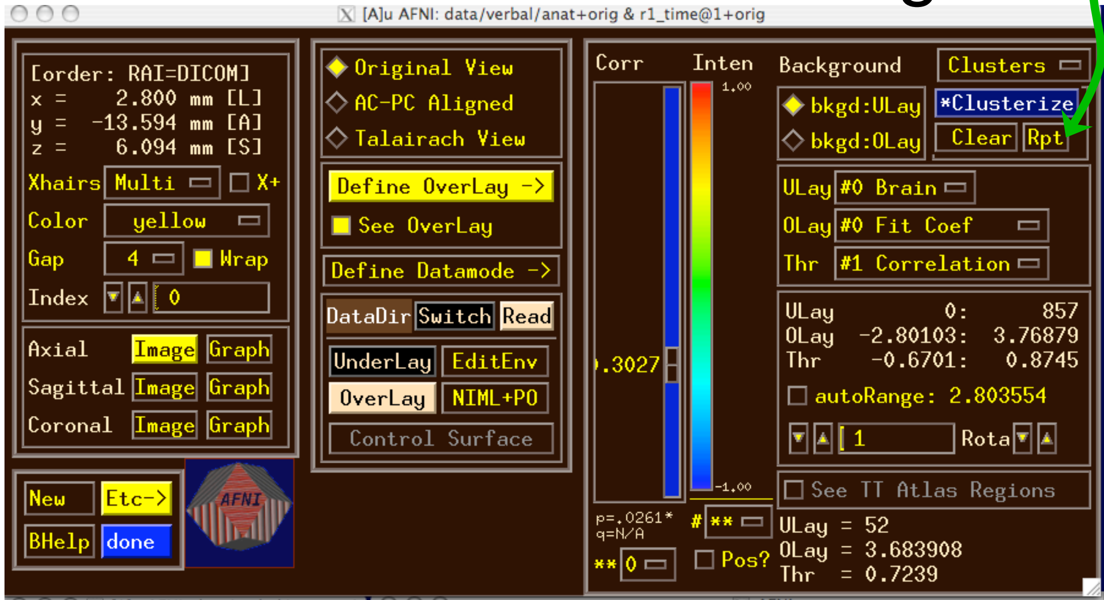
p-value of threshold

At a per-voxel *p*=0.005, a cluster should have **44+** voxels to occur with $\alpha < 0.05$ from noise *only*

`3dClustSim` can be run by `afni_proc.py`: results get stored into statistics dataset, and then used in **AFNI Clusterize** GUI

• Interactive Clustering

Report on clusters of above-threshold voxels



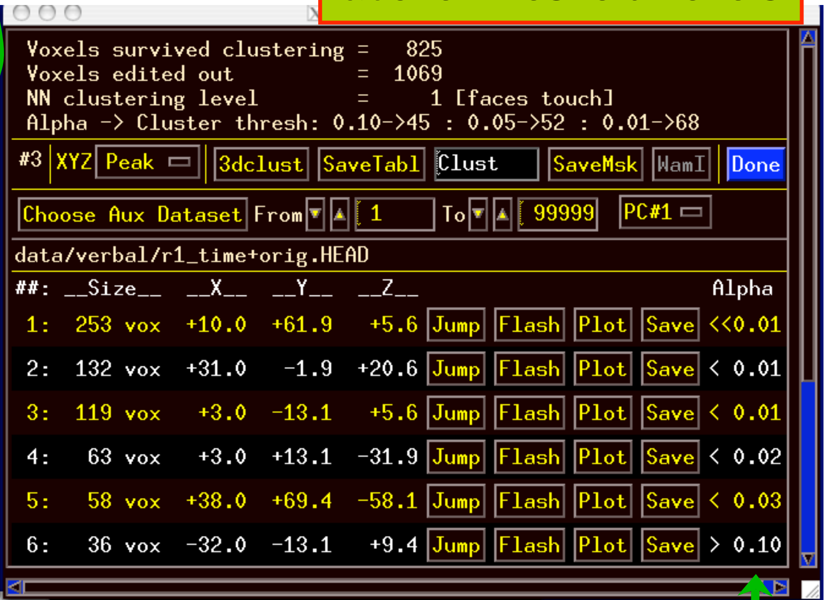
```
[order: RAI=DICOM]
x =     2.800 mm [L]
y =   -13.594 mm [A]
z =     6.094 mm [S]
```

```
ULay #0 Brain
OLay #0 Fit Coef
Thr  #1 Correlation

ULay         0:      857
OLay  -2.80103:  3.76879
Thr    -0.6701:   0.8745

ULay = 52
OLay = 3.683908
Thr  = 0.7239
```

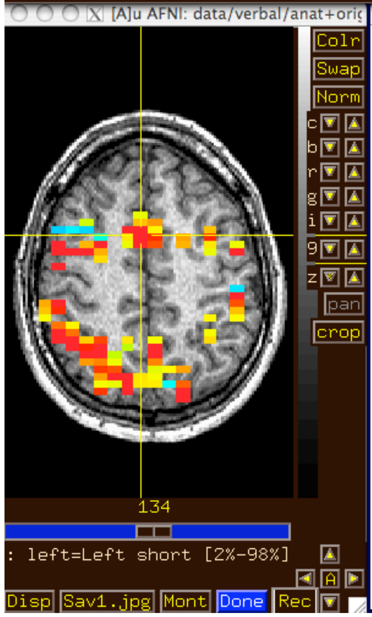
```
.3027    p=.0261*  q=N/A    ** 0
```

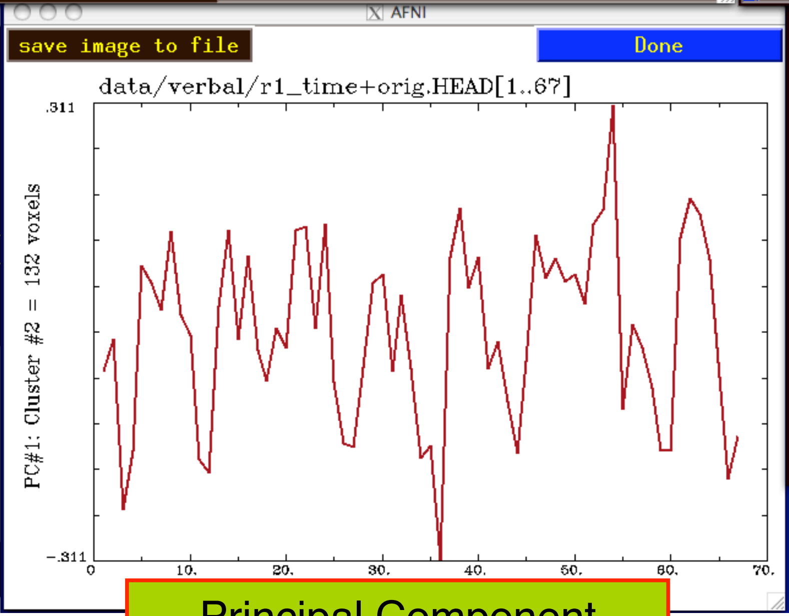


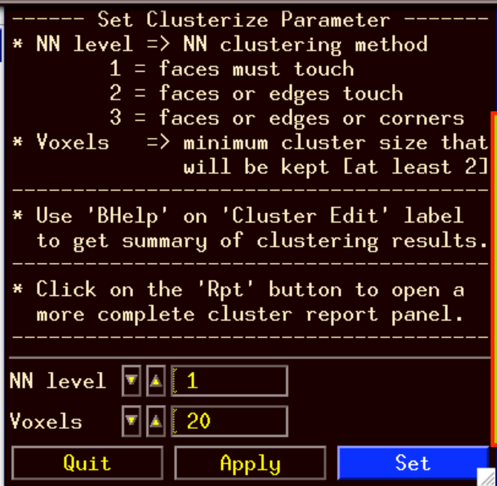
```
Voxels survived clustering =    825
Voxels edited out          =   1069
NN clustering level        =      1 [faces touch]
Alpha -> Cluster thresh: 0.10->45 : 0.05->52 : 0.01->68
```

```
From 1   To 99999
data/verbal/r1_time+orig.HEAD
```

##:	__Size__	__X__	__Y__	__Z__					Alpha
1:	253 vox	+10.0	+61.9	+5.6	Jump	Flash	Plot	Save	<<0.01
2:	132 vox	+31.0	-1.9	+20.6	Jump	Flash	Plot	Save	< 0.01
3:	119 vox	+3.0	-13.1	+5.6	Jump	Flash	Plot	Save	< 0.01
4:	63 vox	+3.0	+13.1	-31.9	Jump	Flash	Plot	Save	< 0.02
5:	58 vox	+38.0	+69.4	-58.1	Jump	Flash	Plot	Save	< 0.03
6:	36 vox	-32.0	-13.1	+9.4	Jump	Flash	Plot	Save	> 0.10







```
------ Set Clusterize Parameter -------
* NN level => NN clustering method
        1 = faces must touch
        2 = faces or edges touch
        3 = faces or edges or corners
* Voxels   => minimum cluster size that
              will be kept [at least 2]
----------------------------------------
* Use 'BHelp' on 'Cluster Edit' label
  to get summary of clustering results.
----------------------------------------
* Click on the 'Rpt' button to open a
  more complete cluster report panel.
----------------------------------------
NN level   1
Voxels     20
```

Cluster α level: interpolated from 3dClustSim table

This panel controls the cluster operation

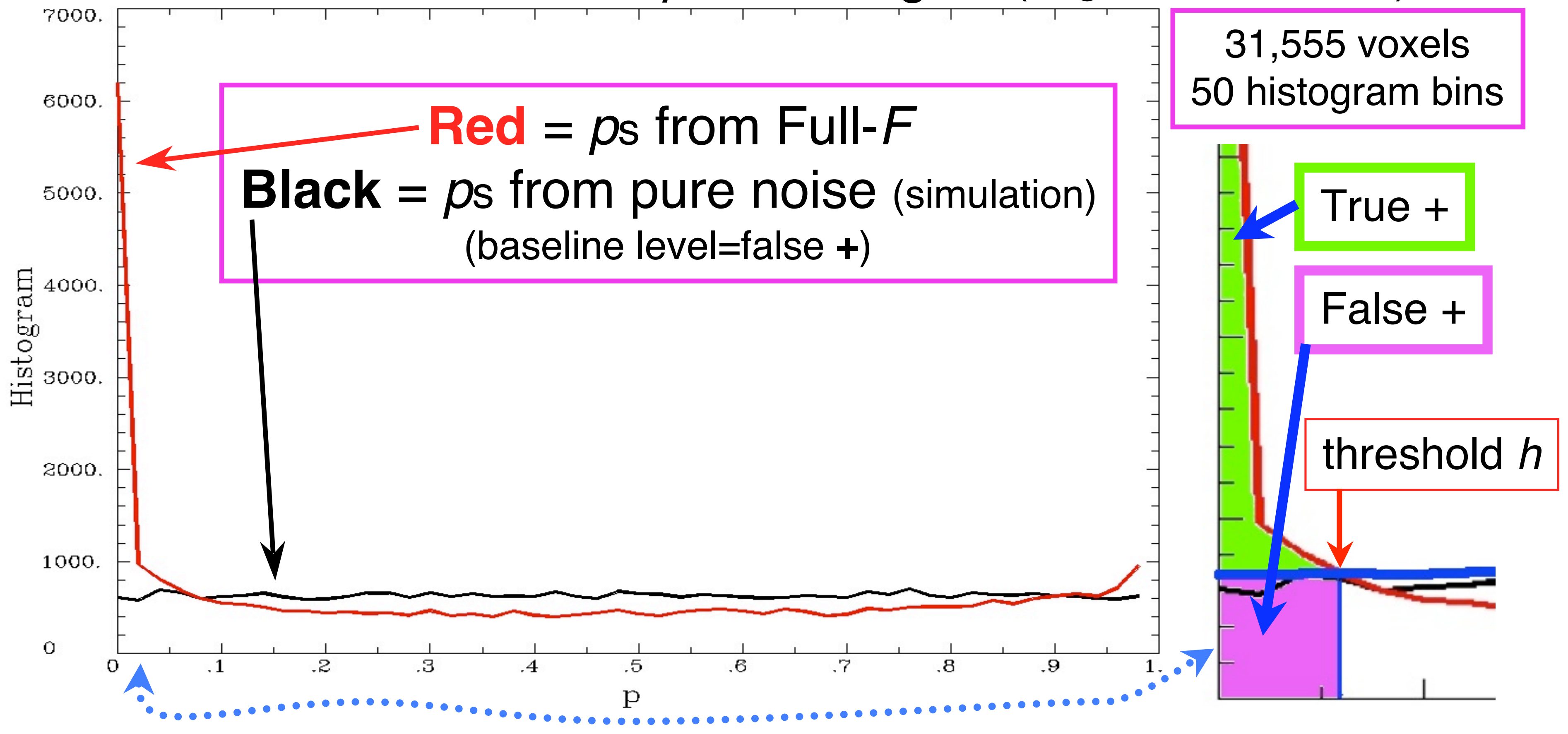
Principal Component time series over cluster #2

False Discovery Rate in AFNI!

- Situation: making *many* statistical tests at once
  - e.g, Image voxels in FMRI; associating genes with disease
- Want to set threshold on statistic (e.g., *F*- or *t*-value) to control ***false positive*** error rate
- Traditionally: set threshold to control probability of making a ***single*** false positive detection
  - But if we are doing 1000s (or more) of tests at once, we have to be very stringent to keep this probability low
- **FDR**: accept the fact that there will be multiple erroneous detections when making lots of decisions
  - Control the ***fraction*** of positive detections that are wrong
    - Of course, no way to tell which individual detections are right!
  - Or at least: control the *expected value* of this fraction

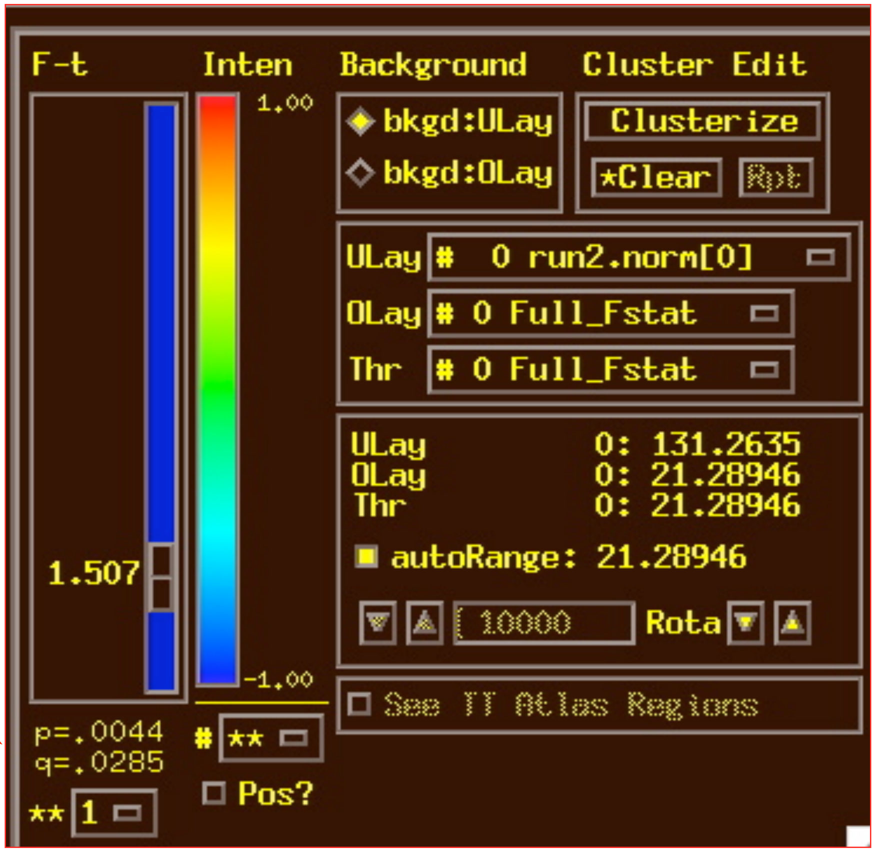
Basic Ideas Behind FDR *q*

- ***If*** all the null hypotheses are true, ***then*** the statistical distribution of the *p*-values will be uniform
  - Deviations from uniformity at low *p*-values ⇒ true positives
  - Baseline of uniformity indicates how many true negatives are hidden in the low *p*-value region ("significant" voxels)



FDR curves in AFNI Datasets

- **3dDeconvolve**, **3dANOVAx**, **3dttest**, and **3dNLfim** now compute FDR curves for all statistical sub-bricks and store them in output header
- **3drefit -addFDR** does same for other datasets
  - **3drefit -unFDR** can be used to delete such info
- **AFNI** now shows *p*- **and** *q*-values below the threshold slider bar
  - Interpolates FDR curve from header (threshold→*z*→*q*)
    - Can be used to adjust threshold by "eyeball"



q = N/A means it's Not Available

FWE or FDR?

- These 2 methods control Type I error in different senses
  - FWE: α_{FW} = Prob (≥ one false positive voxel/cluster in the whole brain)
    - Frequentist's perspective: Probability among **many** hypothetical activation maps gathered under identical conditions
    - Advantage: can directly incorporate smoothness into estimate of α_{FW}
  - FDR = expected fraction of false positive voxels among all detected voxels
    - Focus: controlling false positives among detected voxels in **one** activation map, as given by the experiment at hand
    - Advantage: not afraid of making a few Type I errors in a large field of true positives
  - Concrete example
    - Individual voxel $p = 0.001$ for a brain of 50,000 EPI voxels
    - Uncorrected → ≈50 false positive voxels in the brain
    - FWE: corrected $p = 0.05$ → ≈5% of the time would expect one or more purely false positive clusters in the entire volume of interest
    - FDR: $q = 0.05$ → ≈5% of voxels among those positively labeled ones are false positive
- What if your favorite blob (activation area) fails to survive correction?
  - Tricks (don't tell anyone we told you about these; we'll lie and say we never heard of you)
    - One-tail *t*-test? NN=3 clustering?
    - ROI-based statistics – e.g., grey matter mask, or whatever regions you focus on
  - Analysis on surface; *or*, Use better group analysis tool (**`3dLME`**, **`3dMEMA`**, etc.)