Eye Movement analysis with HMMs: EMHMM Tutorial

Antoni B. Chan
Dept. of Computer Science,
City University of Hong Kong

Janet H. Hsiao
Dept. of Psychology,
The University of Hong Kong

EMHMM Toolbox

v0.76

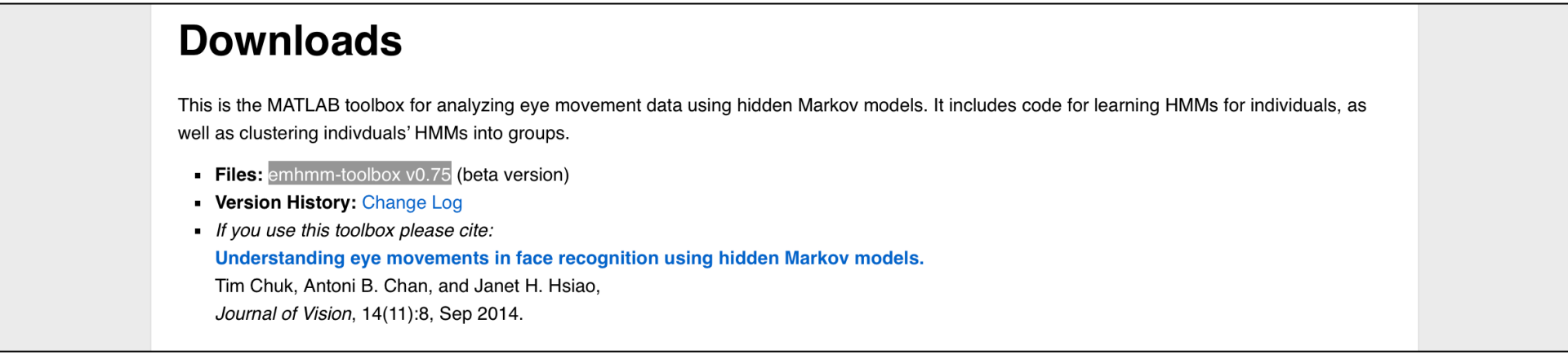
Last updated: 2019-06-16

EMHMM Toolbox Tutorial

- **Tutorial on how to use the EMHMM toolbox**
  - Obtaining and installing the toolbox
  - Setup in MATLAB
  - Reading in data
  - Estimating individual HMMs
  - Viewing individual HMMs
  - Clustering individuals' HMMs to form group HMMs
  - Viewing group HMMs
  - Analysis of group HMMs
  - Holistic/Analytic Models and H-A Scale

Obtaining EMHMM Toolbox

- Download from website:
  - http://visal.cs.cityu.edu.hk/research/emhmm/



Downloads

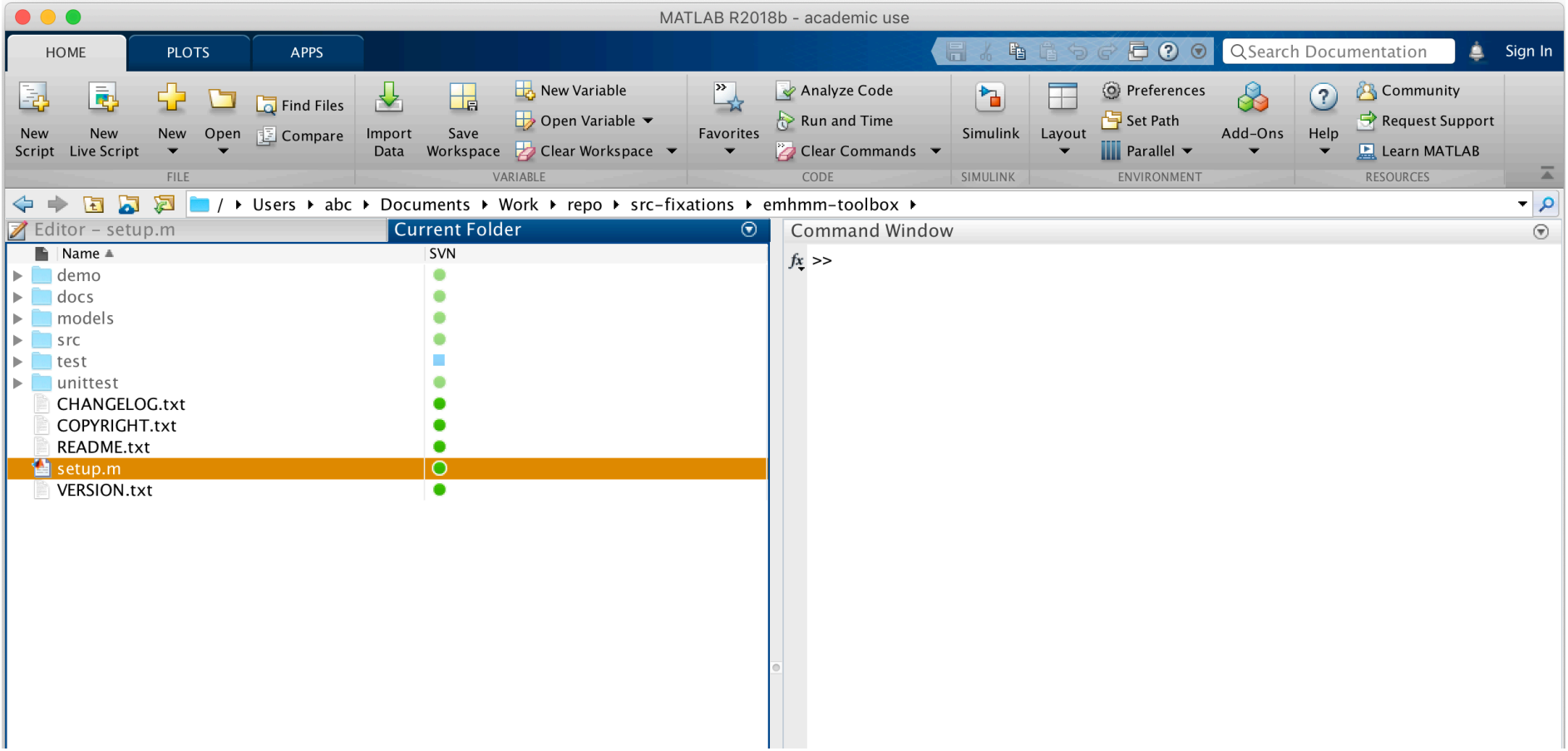
This is the MATLAB toolbox for analyzing eye movement data using hidden Markov models. It includes code for learning HMMs for individuals, as well as clustering indivduals' HMMs into groups.

- **Files:** emhmm-toolbox v0.75 (beta version)
- **Version History:** Change Log
- *If you use this toolbox please cite:*
  Understanding eye movements in face recognition using hidden Markov models.
  Tim Chuk, Antoni B. Chan, and Janet H. Hsiao,
  Journal of Vision, 14(11):8, Sep 2014.

- Installation:
  - Extract the contents of the `emhmm-toolbox.zip` file into your desired directory.

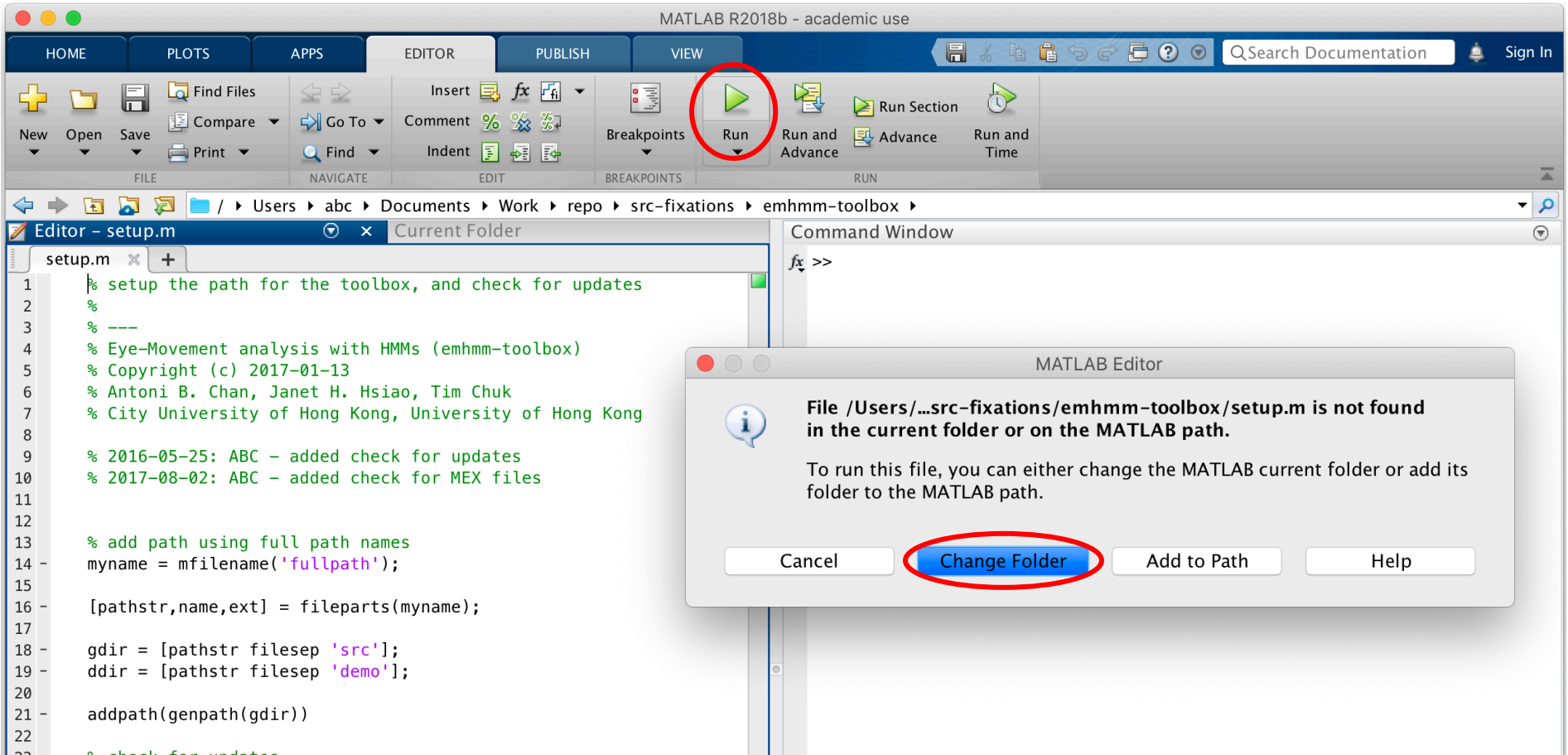
Setup in MATLAB (1)

- First, need to setup the EMHMM toolbox in MATLAB.
  - In MATLAB, navigate to the emhmm-toolbox directory.
  - Double-click on “setup.m” to open it.



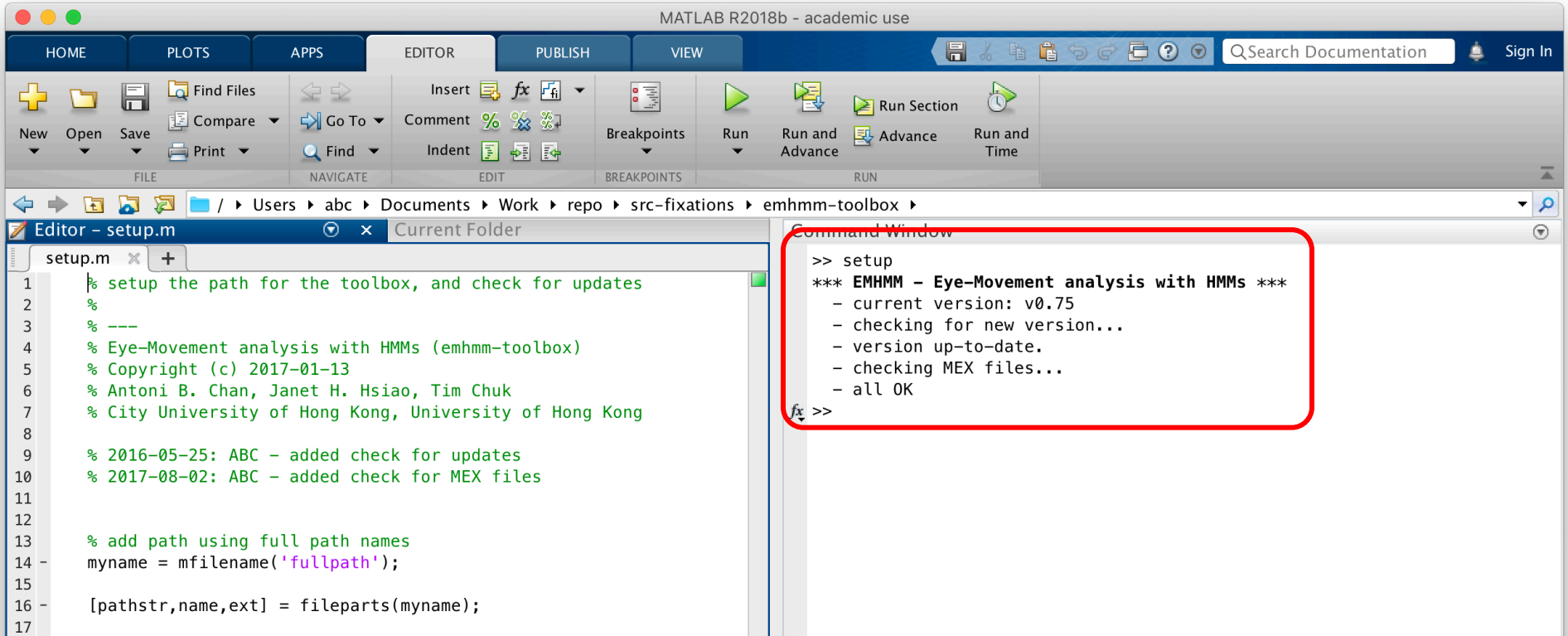
Setup in MATLAB (2)

- Click the “Run” button to run the script.
  - If prompted, click “Change Folder” to change to the script’s directory.



Setup in MATLAB (3)

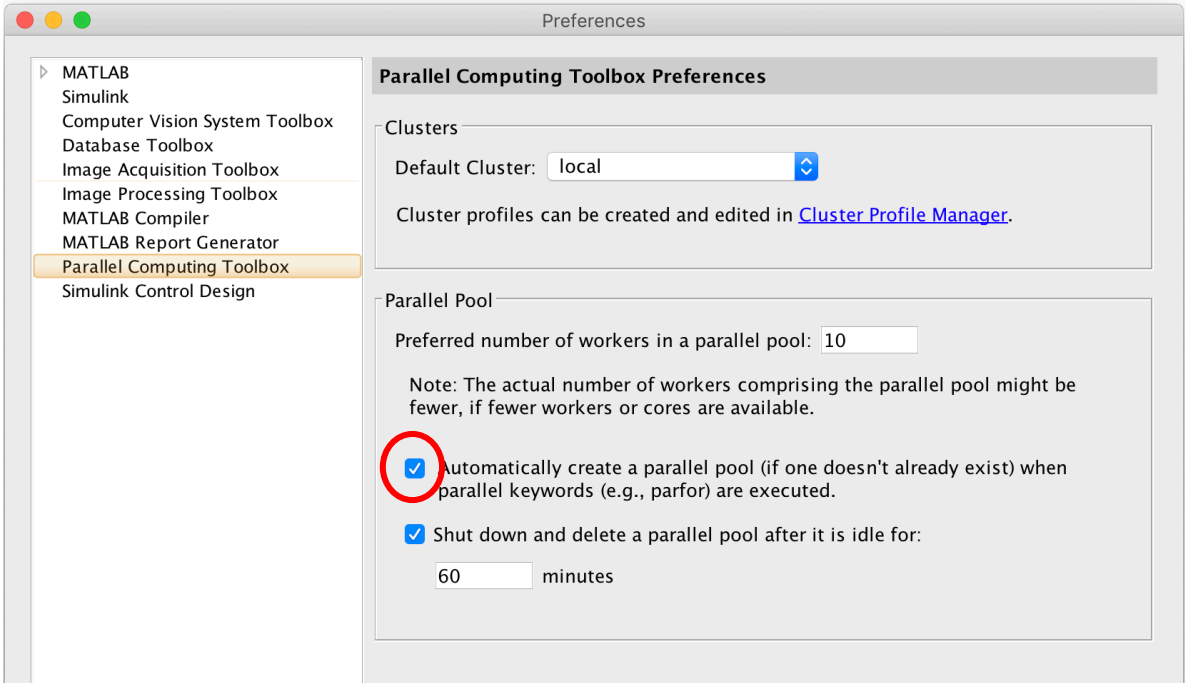
- The setup script will:
  - add the toolbox directories to the MATLAB path.
  - check for MEX files (compiles if necessary).
    - Includes MEX files for: macOS (10.13), Windows (7), Linux (Ubuntu).
    - MEX files make the algorithms 10-20 times faster.



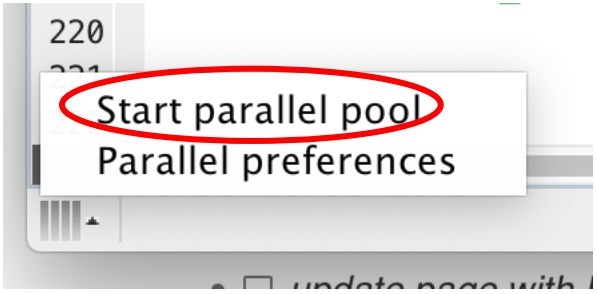
MATLAB toolbox dependencies

- MATLAB version: tested with **2014b** and **2018b**.
- *Statistics toolbox* (required)
- *Parallel computing toolbox* (optional)
  - Used for speeding up computations on multi-core CPUs.

To enable automatically, use Preferences:

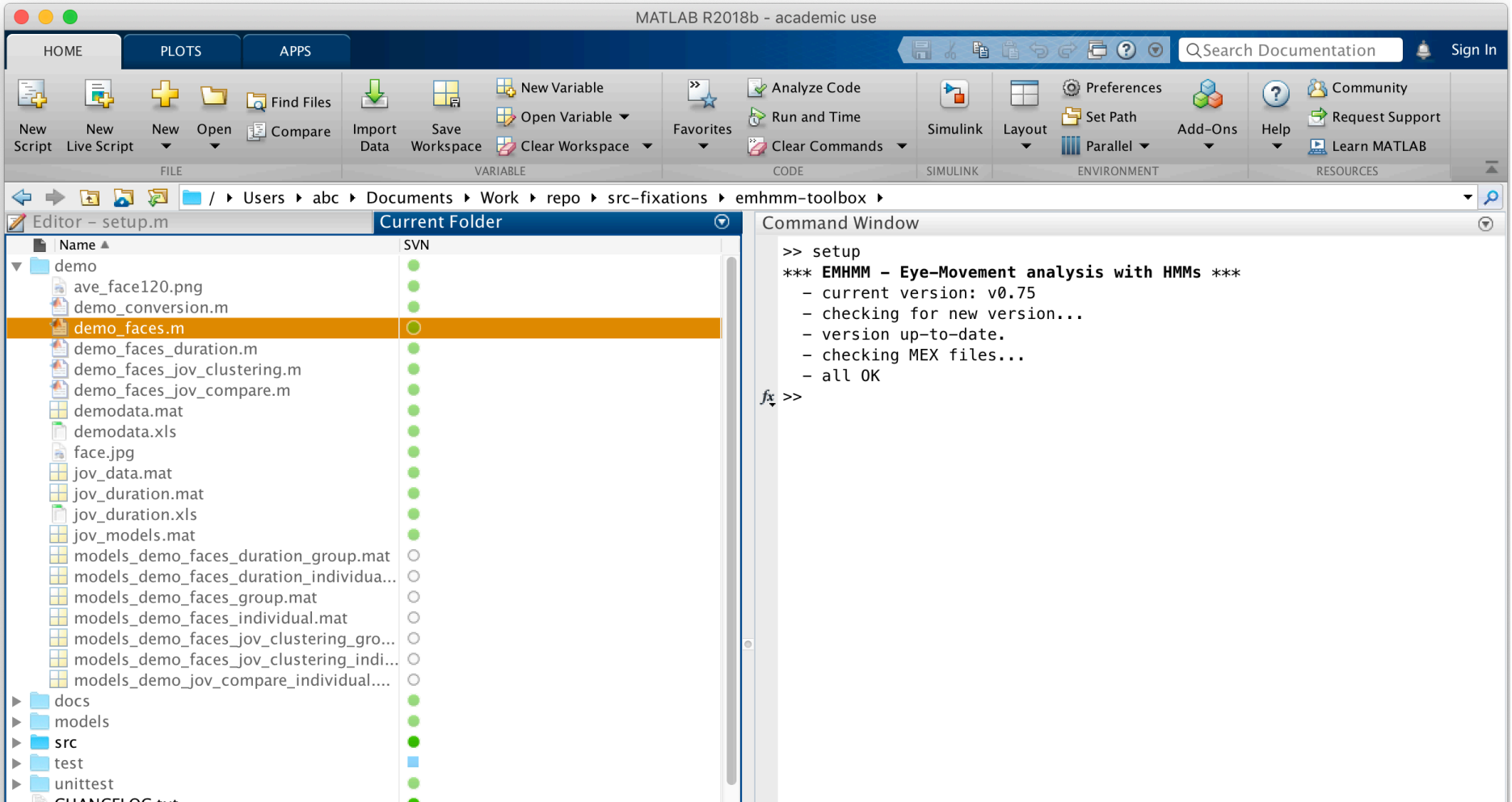


To enable manually, click small icon on bottom-left of the MATLAB window:



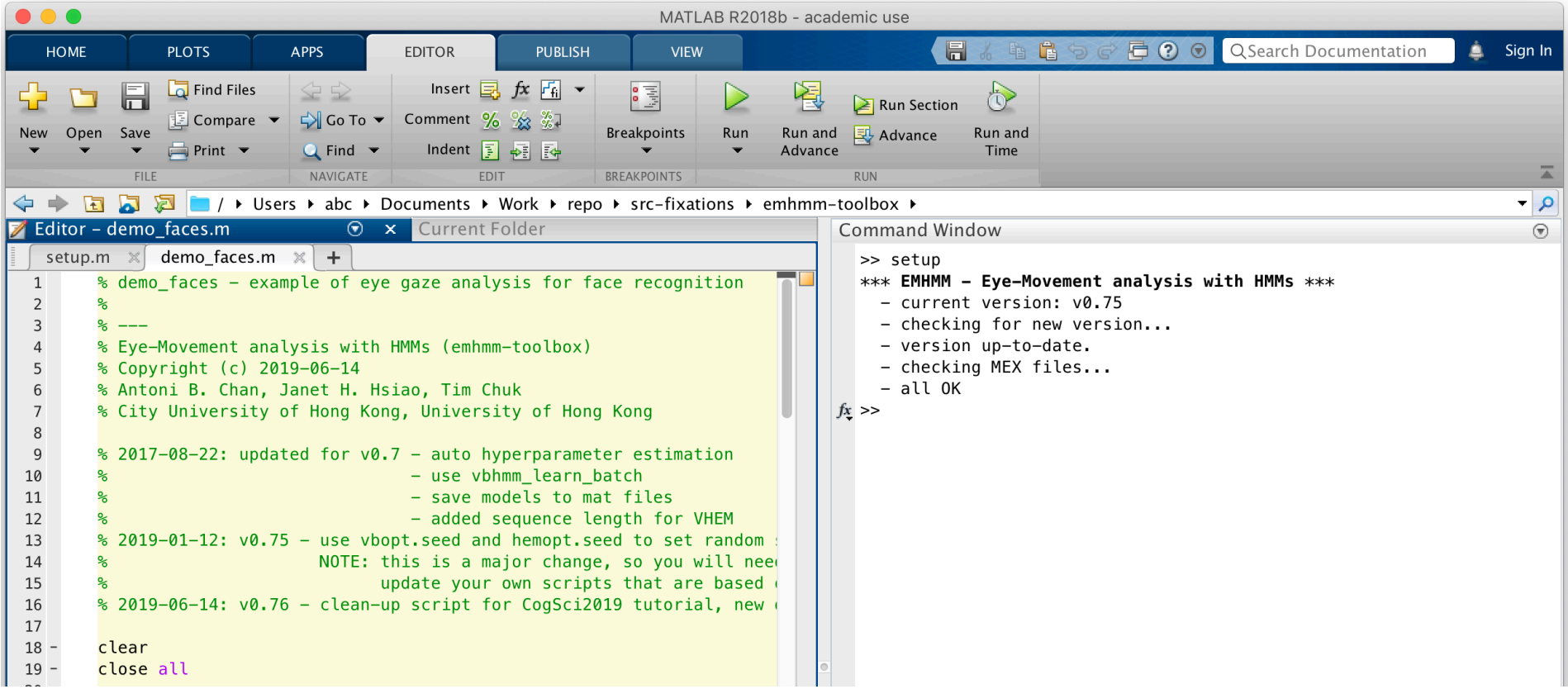
Demo Script

- In the “demo” folder, the script “`demo_faces.m`” shows most of the functionality of the toolbox.



Run Demo Script

- To run the `demo_faces` script:
  - Double-click to open the script.
  - Click “Run”, and select “Change Folder” when prompted.
  - Now we will go through the major parts of the script.



Setup Data Files

- File with the fixation data: `demodata.xls`
- File with the image stimuli: `ave_face120.png`
  - the average image used for visualization
- Files for saving results for saving results (.mat)

```
17
18 -    clear
19 -    close all
20
21      %% Data files %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
22      % names of files used
23 -    xlsname = 'demodata.xls';    % Excel File with fixation data
24 -    faceimg = 'ave_face120.png';  % average face image
25
26      % names of files for saving results
27 -    matfile_individual = 'models_demo_faces_individual.mat';
28 -    matfile_group      = 'models_demo_faces_group.mat';
29
30
31      % pause after each step in the analysis
32 -    do_pause = 1;
33      %do_pause = 0;  % uncomment this to skip pausing
```

Fixation Data Format

- Fixation data is in an Excel spreadsheet (`demodata.xls`).
  - Contains 4 columns:
    - *SubjectID*: subject ID (int or string)
    - *TrialID*: trial ID (int or string)
    - *FixX*: fixation X coordinate
    - *FixY*: fixation Y coordinate
  - Fixation data is automatically separated according to subject ID and trial ID.
  - In each trial, assumes fixations occur in order.
- Assumes that fixation locations have been aligned to the image.

	A	B	C	D
1	SubjectID	TrialID	FixX	FixY
2	1	1	182.16	209.52
3	1	1	201.68	192.13
4	1	1	134.48	186.12
5	1	2	166.08	192.21
6	1	2	188.96	200.38
7	1	2	122.88	208
8	1	3	147.84	275
9	1	3	172.72	212.24
10	1	3	109.36	217.45
11	1	4	168.96	202.39
12	1	4	180.56	223.39
13	1	4	109.68	226.91
14	1	5	170.88	231.32
15	1	5	120.16	234.53
16	1	5	180.88	239.58
17	1	6	172.32	204.71
18	1	6	98.16	206.87
19	1	6	180.64	203.03
20	1	7	161.76	242.7
21	1	7	173.92	222.98
22	1	7	137.12	216.97
23	1	8	169.36	222.02
24	1	8	178.88	222.9
25	1	8	128.48	230.68
26	1	9	182.56	205.51
27	1	9	190.08	213.29
28	1	9	142.64	217.53
29	1	10	171.44	210.96
30	1	10	123.28	217.53
31	1	10	183.44	202.31
32	1	11	112.8	200.3
33	1	11	145.12	235.97

Reading in Data

MATLAB R2018b - academic use

HOME | PLOTS | APPS | EDITOR | PUBLISH | VIEW

/ ▸ Users ▸ abc ▸ Documents ▸ Work ▸ repo ▸ src-fixations ▸ emhmm-toolbox ▸ demo

Editor - demo_faces.m

setup.m | demo_faces.m

```
20
21    %% Data files %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
22    % names of files used
23 -  xlsname = 'demodata.xls';    % Excel File with fixation data
24 -  faceimg = 'ave_face120.png';  % average face image
25
26    % names of files for saving results
27 -  matfile_individual = 'models_demo_faces_individual.mat';
28 -  matfile_group      = 'models_demo_faces_group.mat';
29
30
31    % pause after each step in the analysis
32 -  do_pause = 1;
33    %do_pause = 0;  % uncomment this to skip pausing
34
35
36    %% Load data from xls %%%%%%%%%%%%%%%%%%%%%%%%%
37    % see the xls file for the format
38 -  [data, SubjNames, TrialNames] = read_xls_fixations(xlsname);
39
40    % the data is read and separated by subject and trial, and stored in a cell
41    % data{i}         = i-th subject
42    % data{i}{j}      = ... j-th trial
43    % data{i}{j}(t,:) = ... [x y] location of t-th fixation
44
45    % the same data is stored in a mat file.
46    %load demodata.mat
47
48    % the number of subjects
49 -  N = length(data);
50
51    % load image
52 -  img0 = imread(faceimg);
```

Current Folder

Command Window

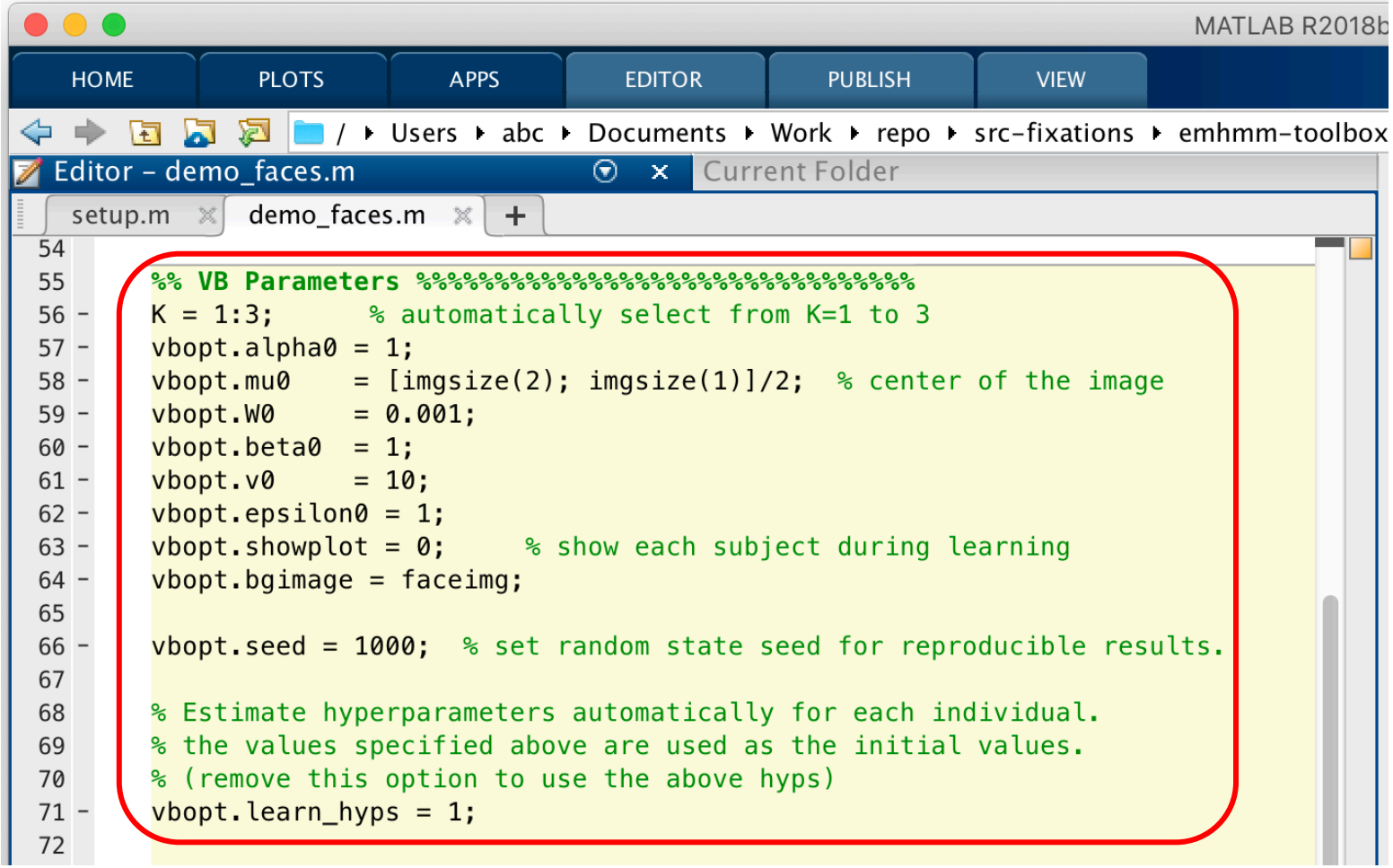
```
Reading demodata.xls
- found SubjectID in column 1
- found TrialID in column 2
- found FixX in column 3
- found FixY in column 4
- found 10 subjects:
1 2 3 4 5 6 7 8 9 10
    * subject 1 had 40 trials
    * subject 2 had 40 trials
    * subject 3 had 40 trials
    * subject 4 had 40 trials
    * subject 5 had 40 trials
    * subject 6 had 40 trials
    * subject 7 had 40 trials
    * subject 8 had 40 trials
    * subject 9 had 40 trials
    * subject 10 had 39 trials
=== running Subject 1 ===
=== running Subject 2 ===
=== running Subject 3 ===
=== running Subject 4 ===
=== running Subject 5 ===
=== running Subject 6 ===
[Subject 1] -- vbhmm K=1: (seed=1000).
[Subject 1] (K=1) optimizing trial 1: -1
[Subject 1] (K=1) best run=1; LL=-1104.6
[Subject 1] (K=1)  {alpha0=1; epsilon0=1
[Subject 2] -- vbhmm K=1: (seed=1000).
[Subject 2] (K=1) optimizing trial 1: -1
[Subject 2] (K=1) best run=1; LL=-993.39
[Subject 2] (K=1)  {alpha0=1; epsilon0=1
[Subject 3] -- vbhmm K=1: (seed=1000).
[Subject 3] (K=1) optimizing trial 1: -9
```

Estimating individual HMMs

- An individual HMM is estimated from each subject's fixation data using the variational Bayesian algorithm.
- Automatically selects the model hyperparameters by maximizing the log-likelihood of the data.
  - Best of 50 trials with random initializations.
- Automatically selects the number of ROIs, *K.*
  - The model with highest log-likelihood over all *K* is selected.

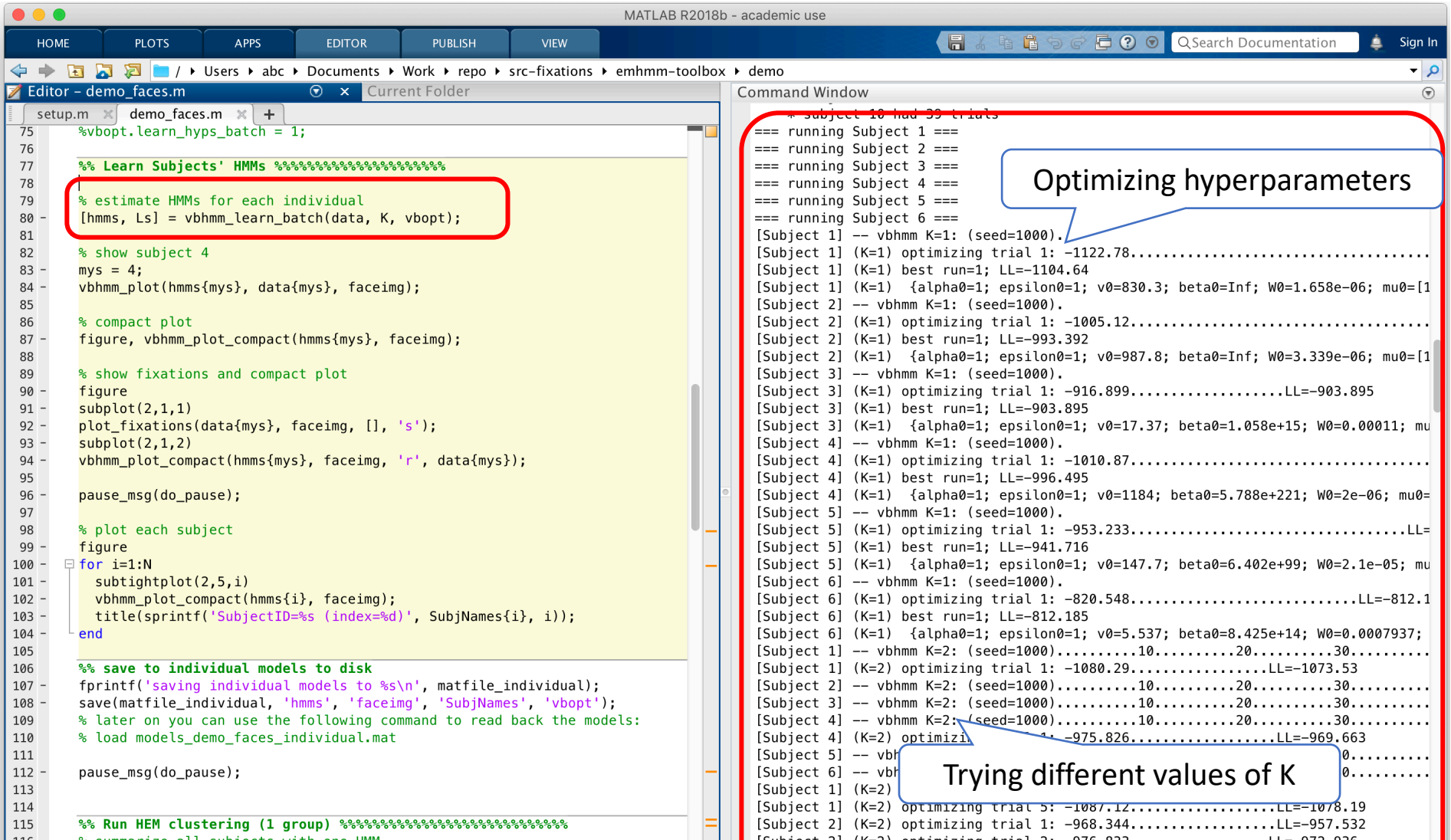
Setting up Options

- `K` contains an array of number of ROIs to try.
- `vbopt` is a structure with various options.



Estimating Individual HMMs

- Using data from 10 subjects, learn 10 HMMs

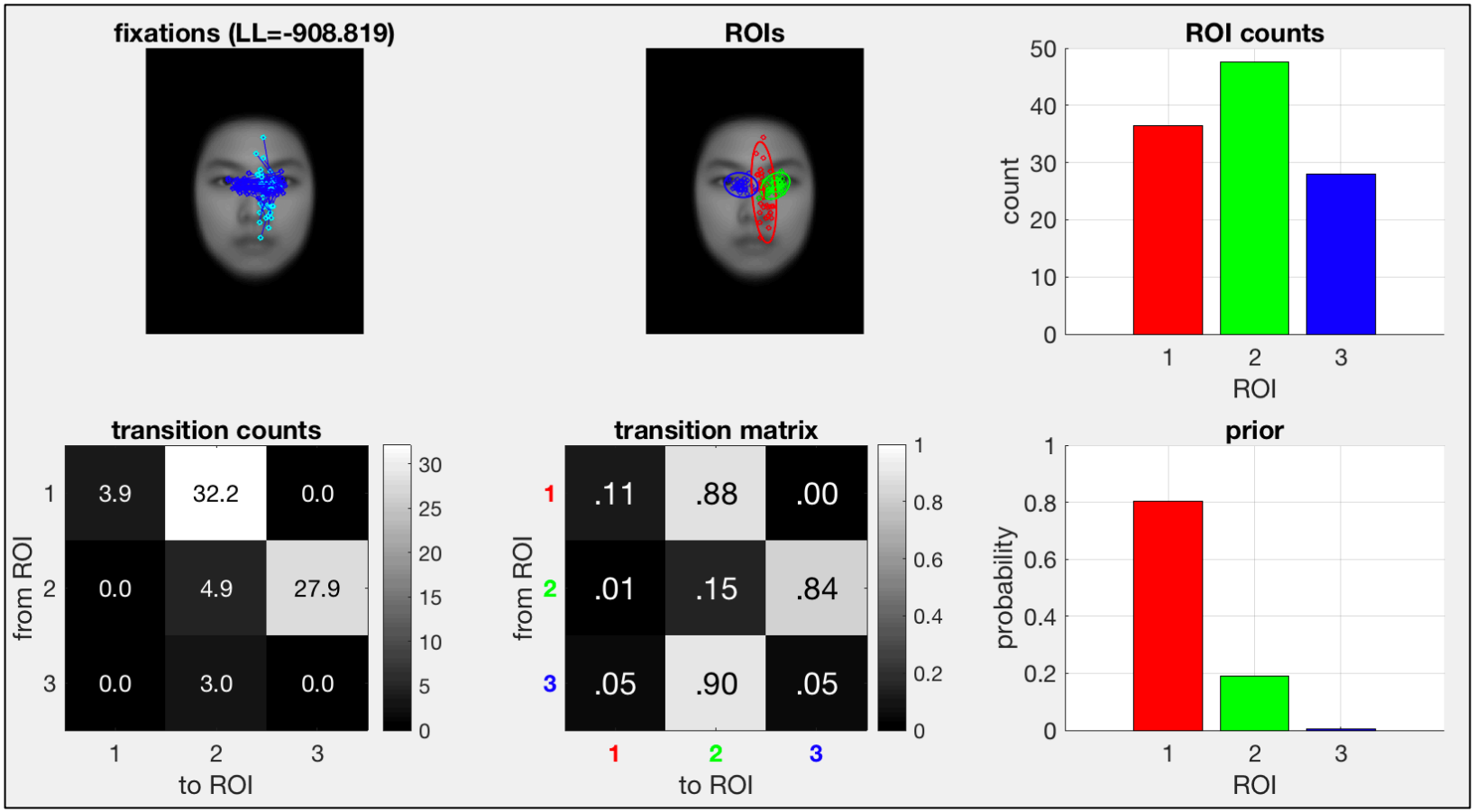


Visualizing an HMM

Figure 1

- Look at the 4th subject:
- Output Figure

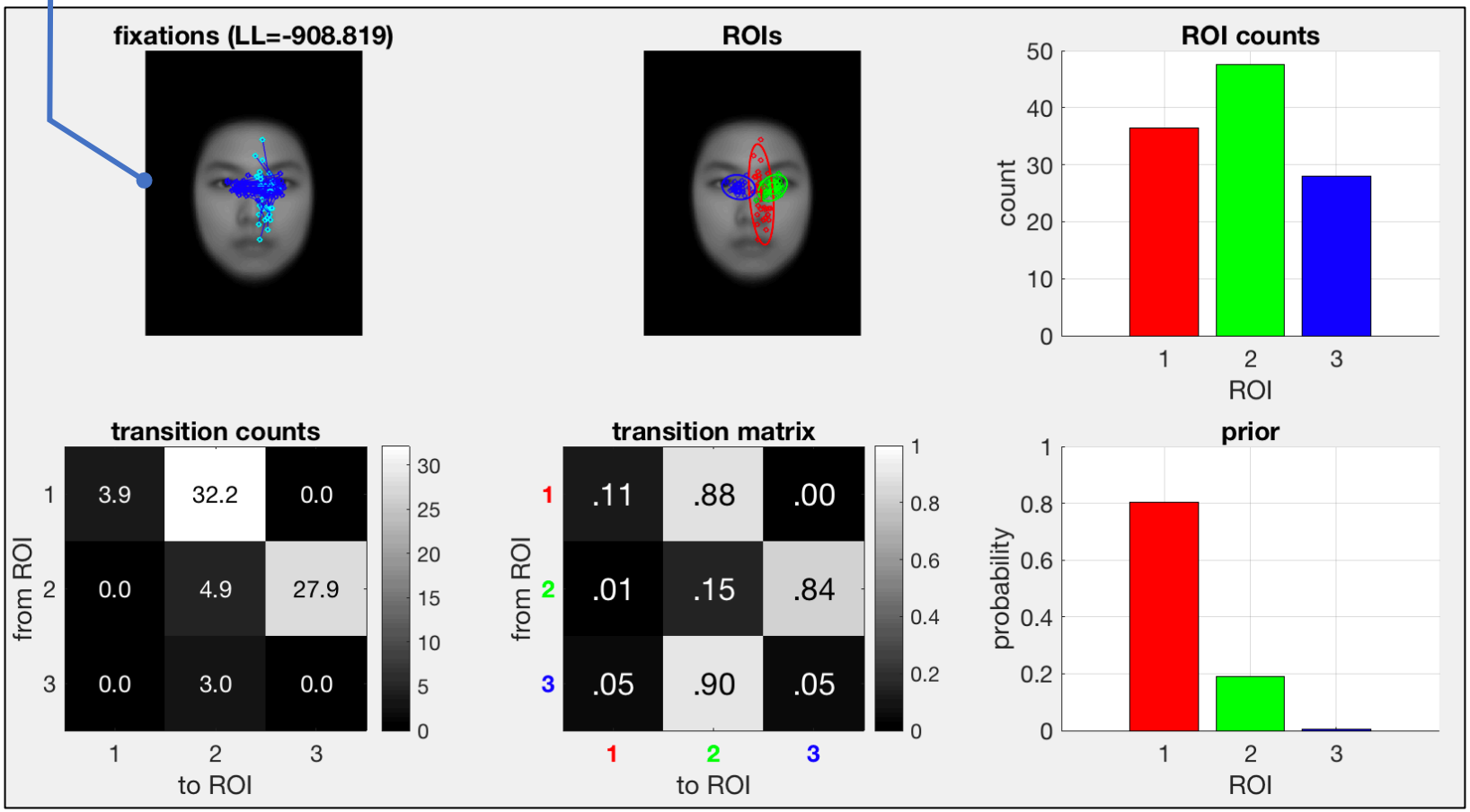
```
79
80 - % estimate HMMs for each individual
[hmms, Ls] = vbhmm_learn_batch(data, K, vbopt);
81
82 % show subject 4
83 - mys = 4;
84 - vbhmm_plot(hmms{mys}, data{mys}, faceimg);
85
```



Visualizing an HMM

Figure 1

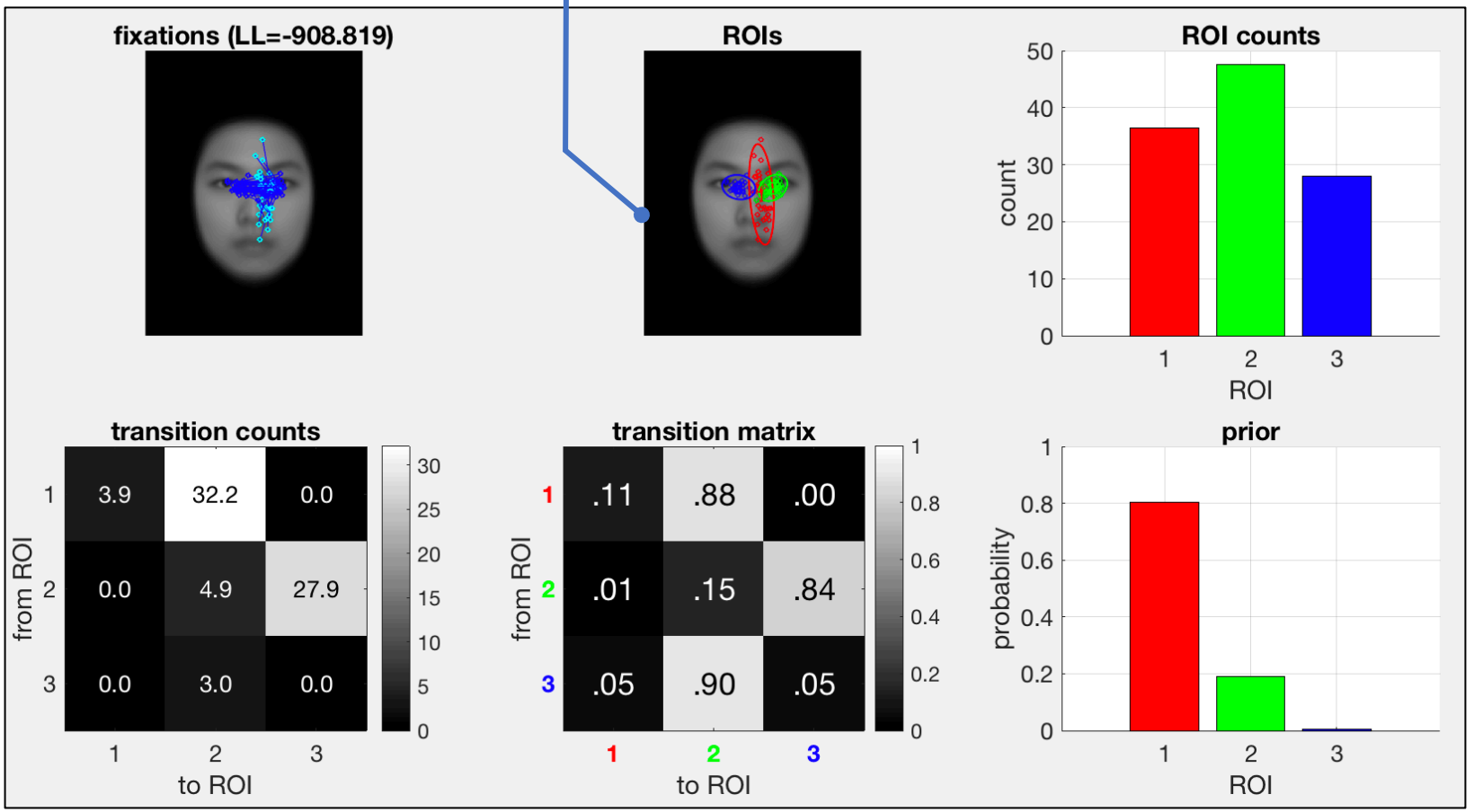
Fixation sequences: cyan "o" is the first fixation.



Visualizing an HMM

Figure 1

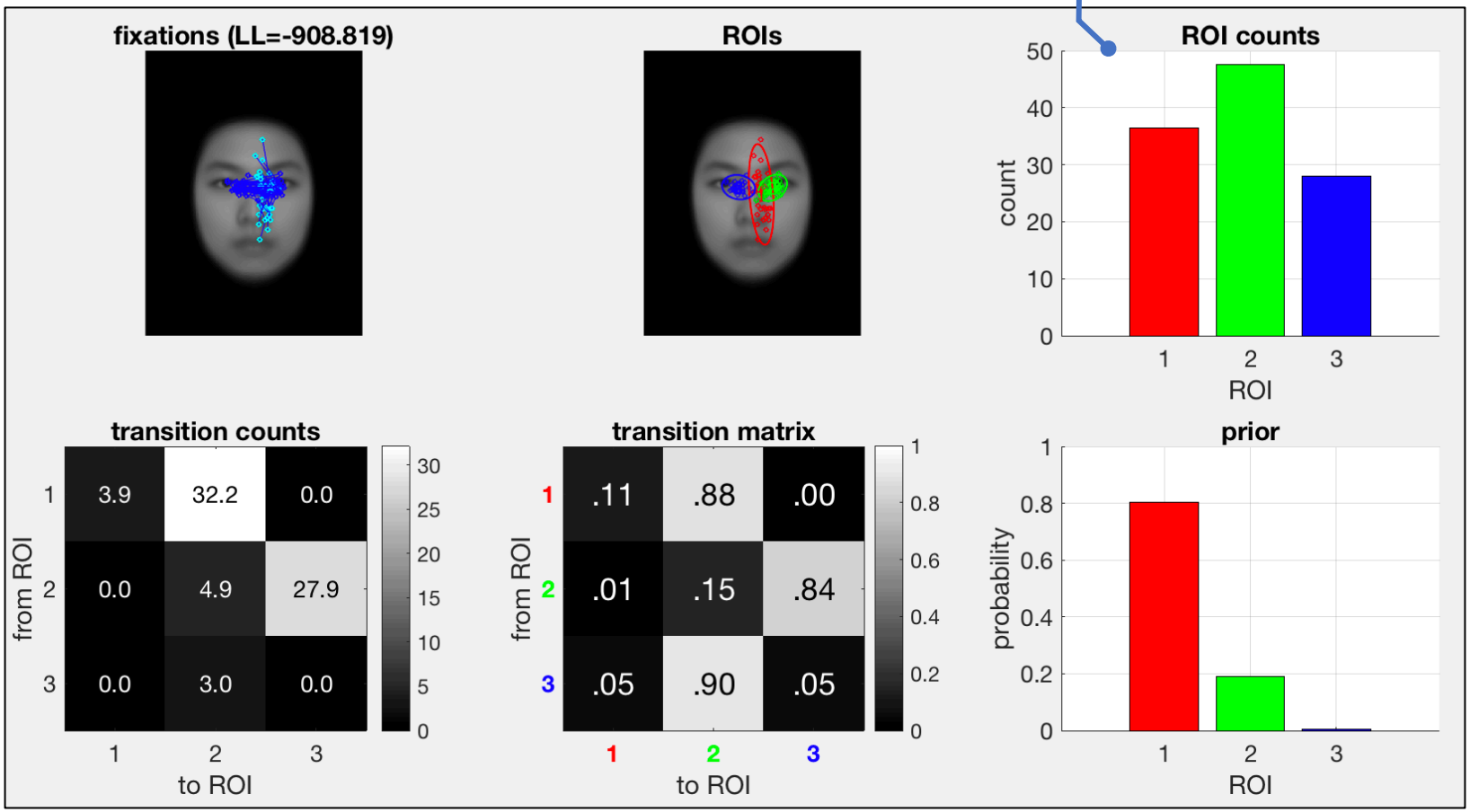
ROIs: ellipses represent 2 standard deviation contours from the mean of the 2D Gaussian (95% of the probability density). Color indicates the fixations assigned to an ROI.



Visualizing an HMM

Figure 1

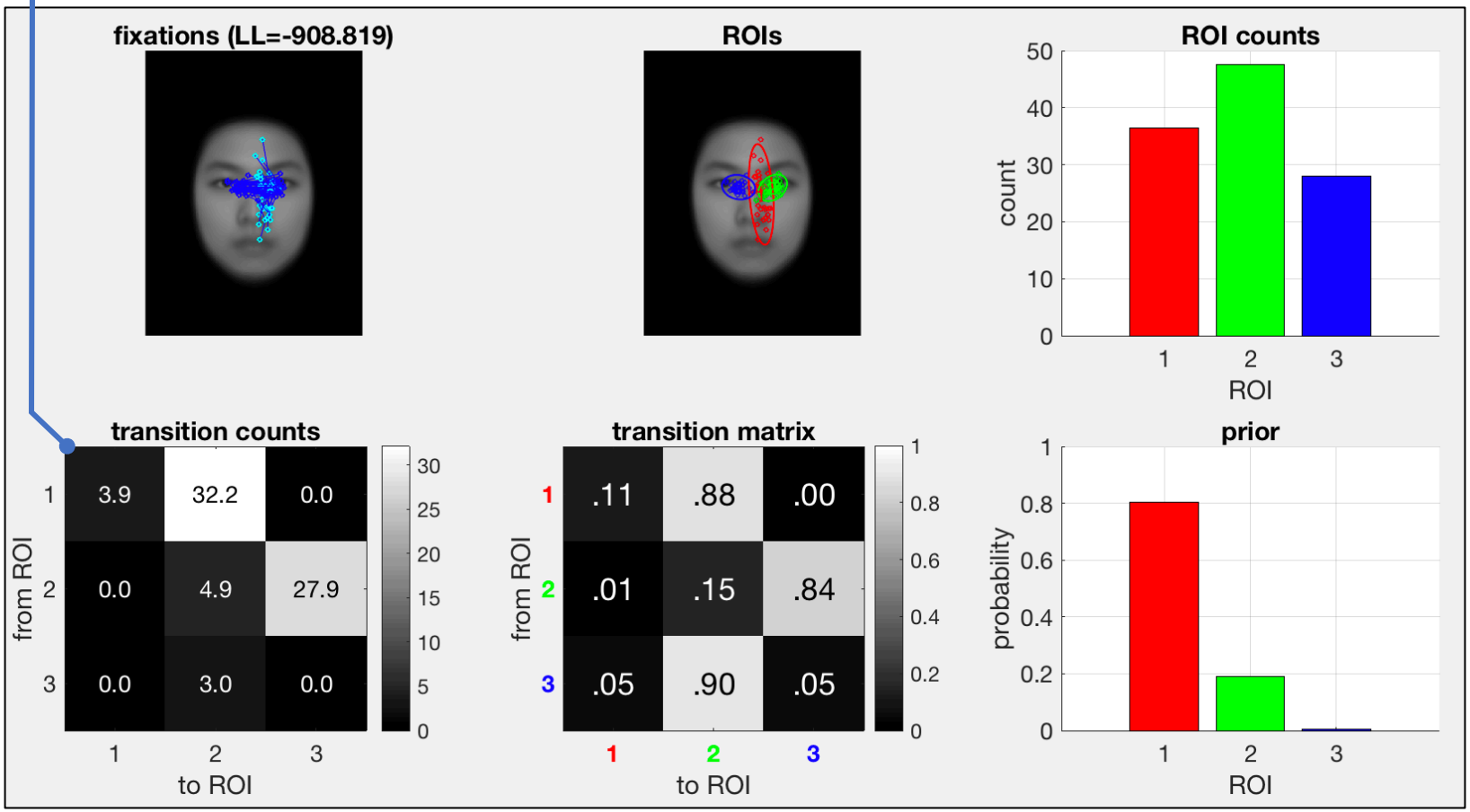
ROI counts: total number of fixations in each ROI.



Visualizing an HMM

Figure 1

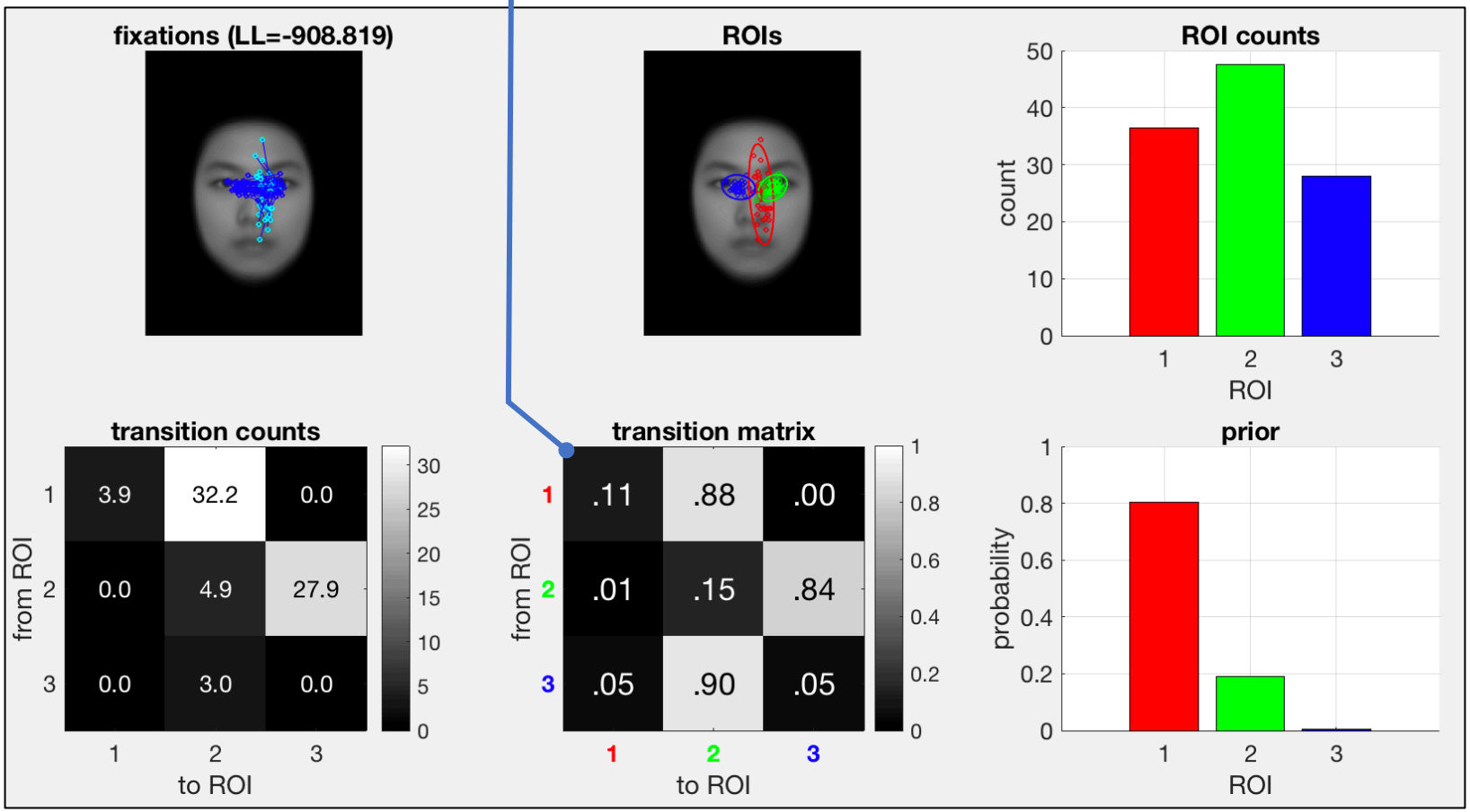
Transition counts: total number of transitions from one ROI to another ROI.



Visualizing an HMM

Figure 1

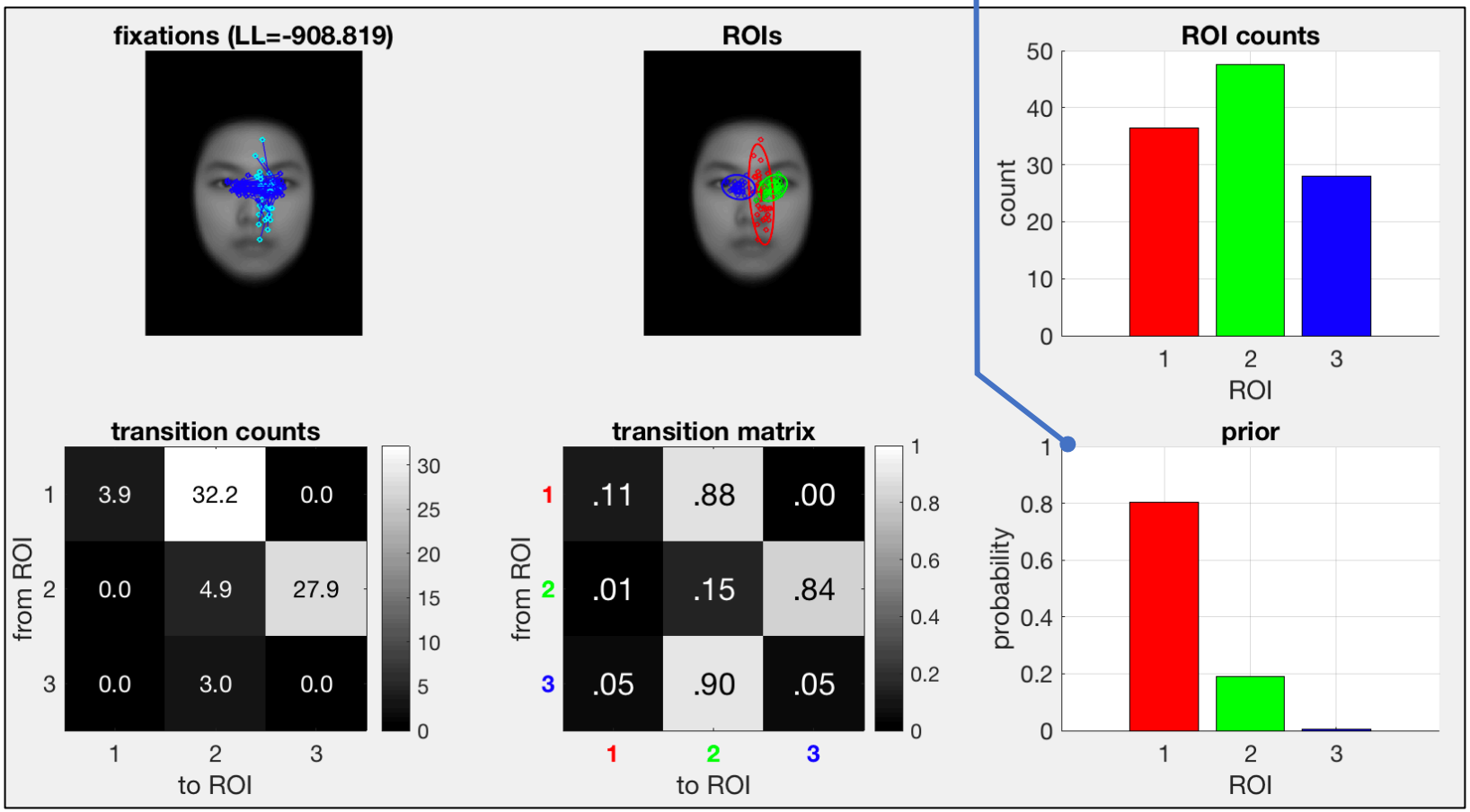
Transition matrix: given a particular ROI, probability of moving to another ROI.
Each row is a probability distribution (sums to one).



Visualizing an HMM

Figure 1

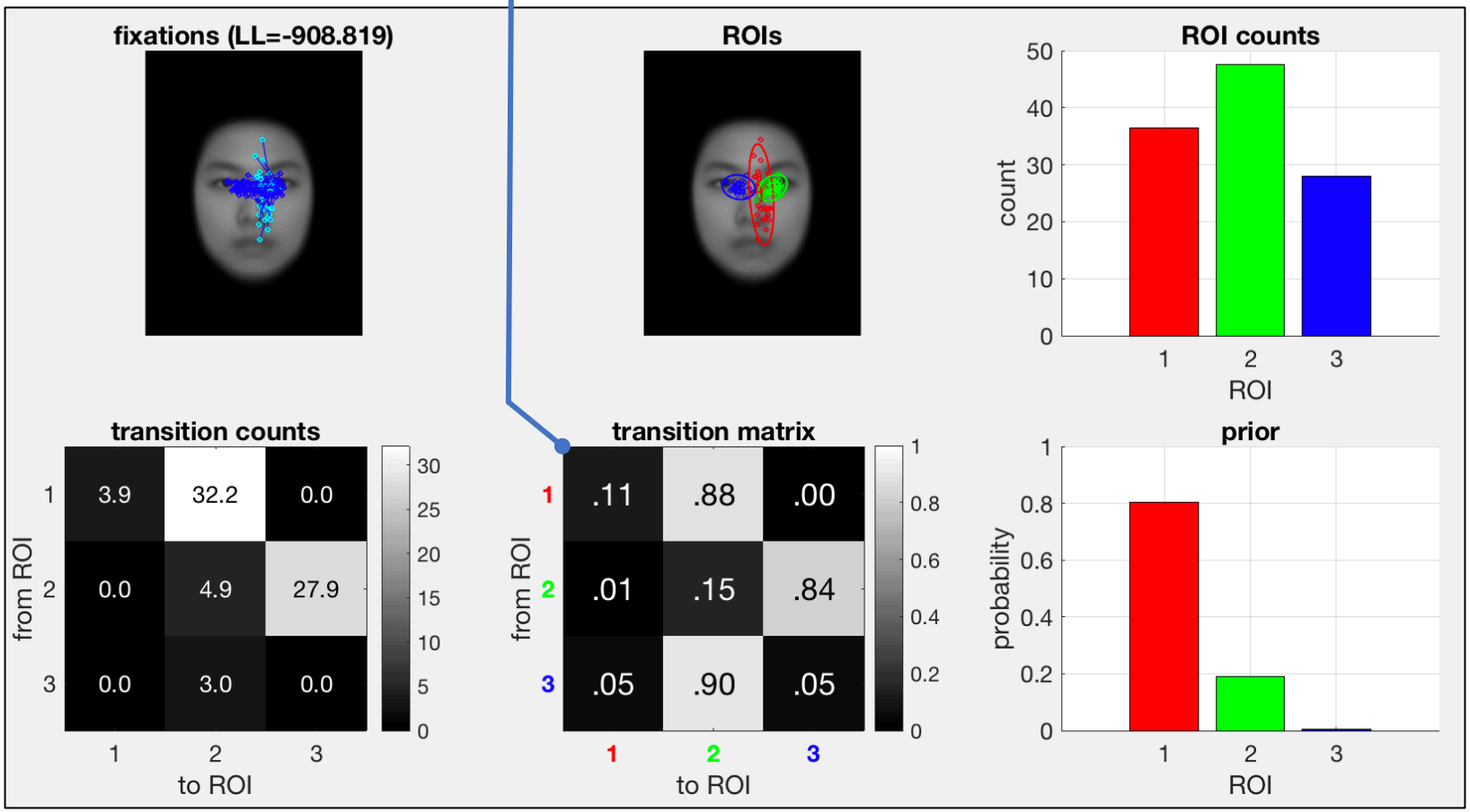
Prior Probabilities: probability of the ROI of the first fixation.



Visualizing an HMM

Figure 1

Roi Order: ROIs are automatically sorted according to the most likely fixation path: ROI-1 is the most probable initial fixation; ROI-2 is the most likely next fixation given ROI-1, etc.



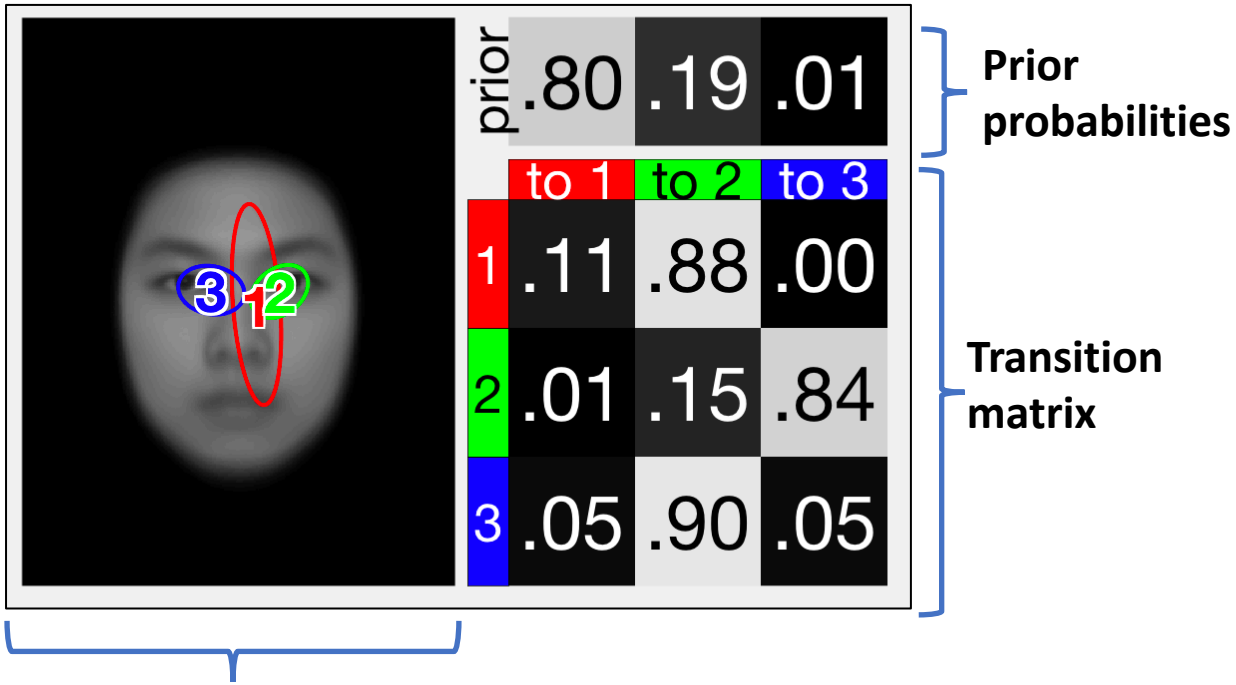
Compact Visualization

Figure 2

- Visualization without fixations:

```
83 -    mys = 4;
84 -    vbhmm_plot(hmms{mys}, data{mys}, faceimg);
85
86      % compact plot
87 -    figure, vbhmm_plot_compact(hmms{mys}, faceimg);
88
89      % show fixations and compact plot
90 -    figure
```

- Figure:



ROIs: numbers are the ROI centers,
ellipse represents 95% probability region

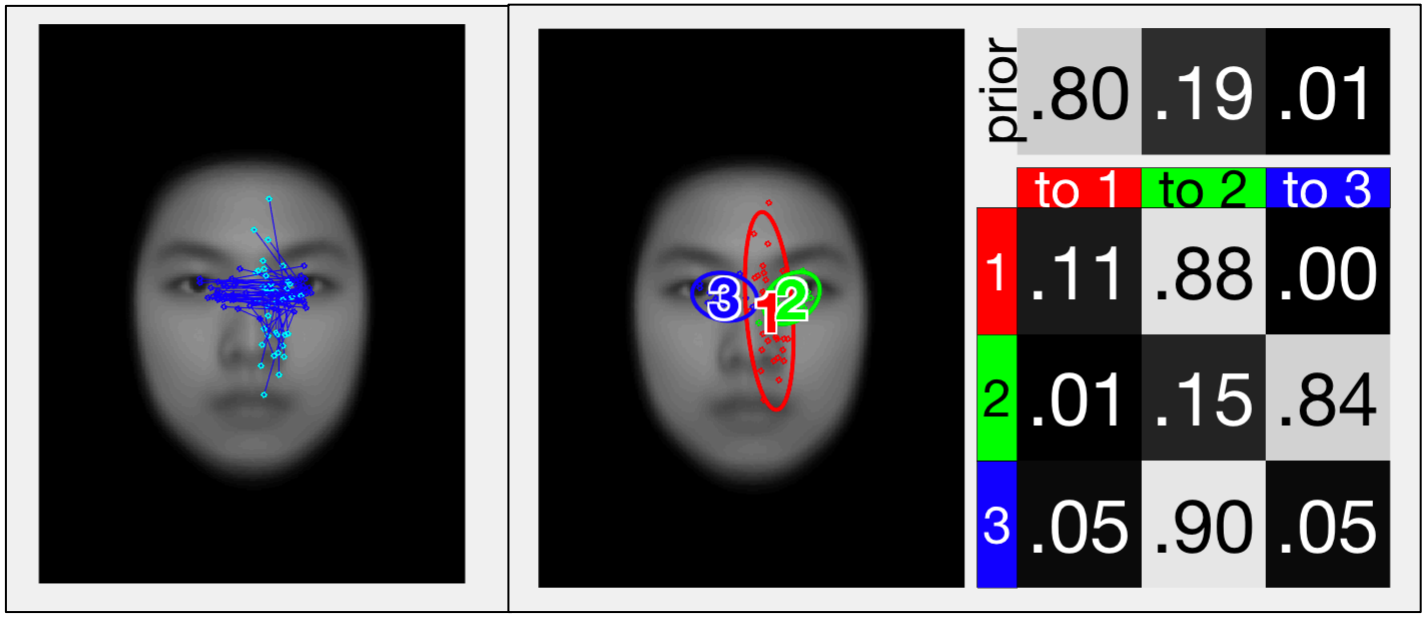
Visualization with Fixations

Figure 3

- Visualize fixations:

```
88
89
90 -  % show fixations and compact plot
91 -  figure
92 -  subplot(2,1,1)
93 -  plot_fixations(data{mys}, faceimg, [], 's');
94 -  subplot(2,1,2)
95 -  vbhmm_plot_compact(hmms{mys}, faceimg, 'r', data{mys});
```

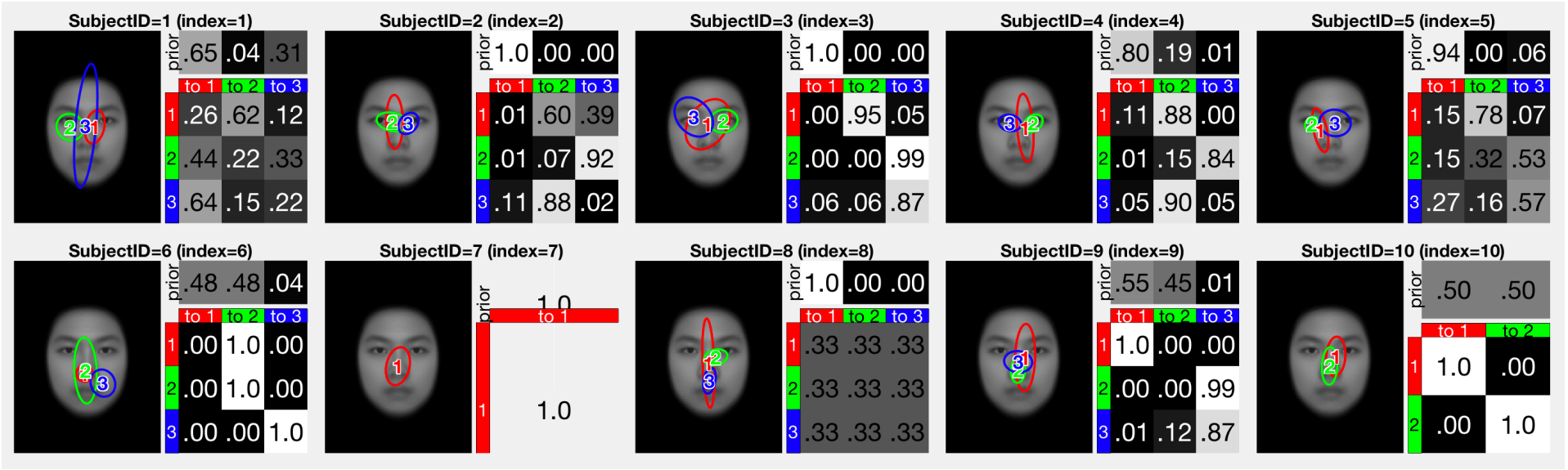
- Output Figure:



Individual HMMs

Figure 4

- HMMs for the 10 subjects

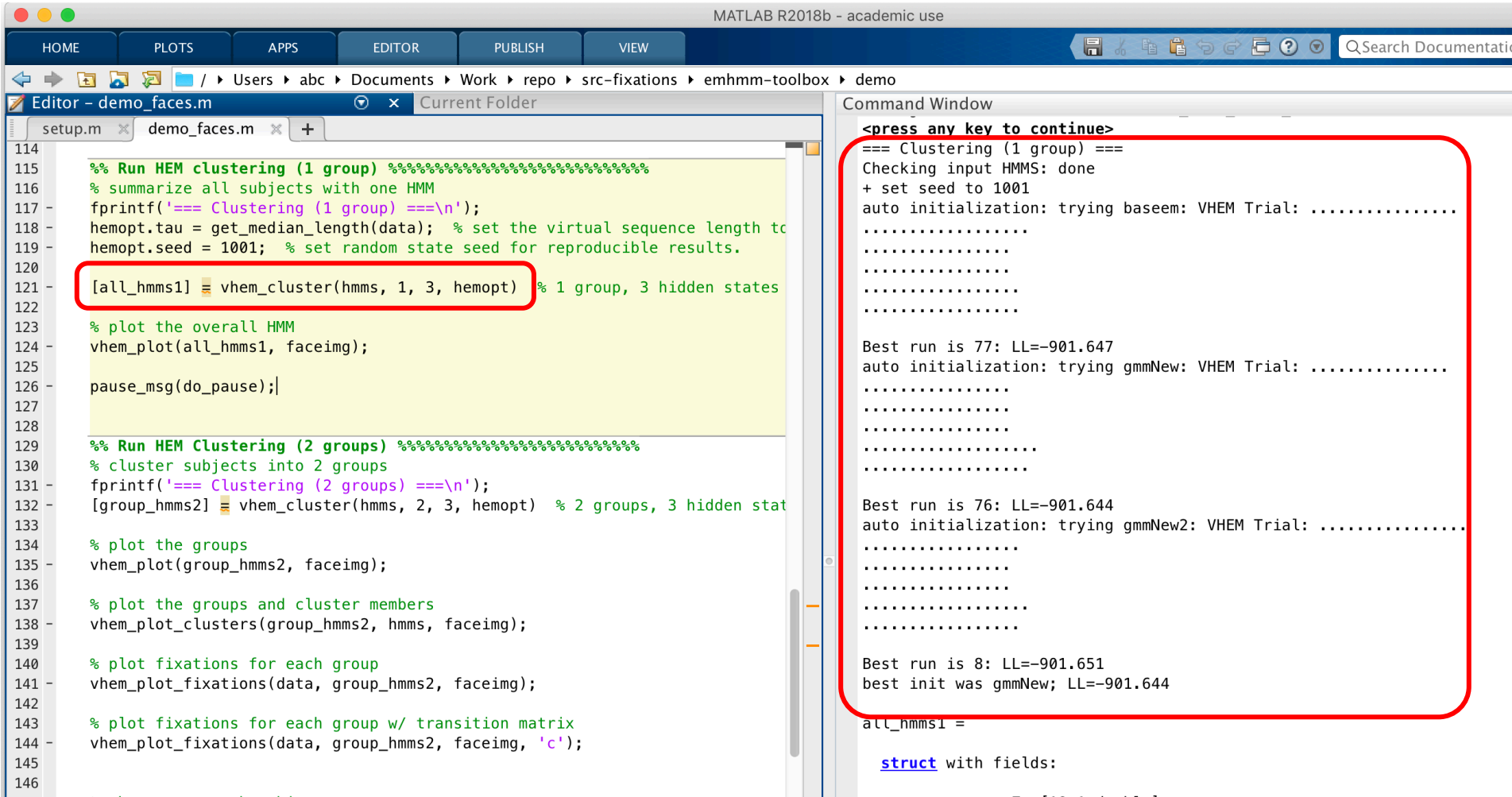


Clustering HMMs – One Group

- We use the variational HEM (VHEM) algorithm to cluster HMMs to discover group patterns.
  - 3 different initialization methods are used.
  - 100 trials each.
  - The run with the highest log-likelihood is kept.
- The overall eye gaze pattern is obtained by clustering into 1 group.
- Common eye gaze strategies are obtained by clustering into 2 groups.

Clustering – 1 Group

- Use **1** cluster to get overall strategy.
  - Set the number of ROIs to **3**



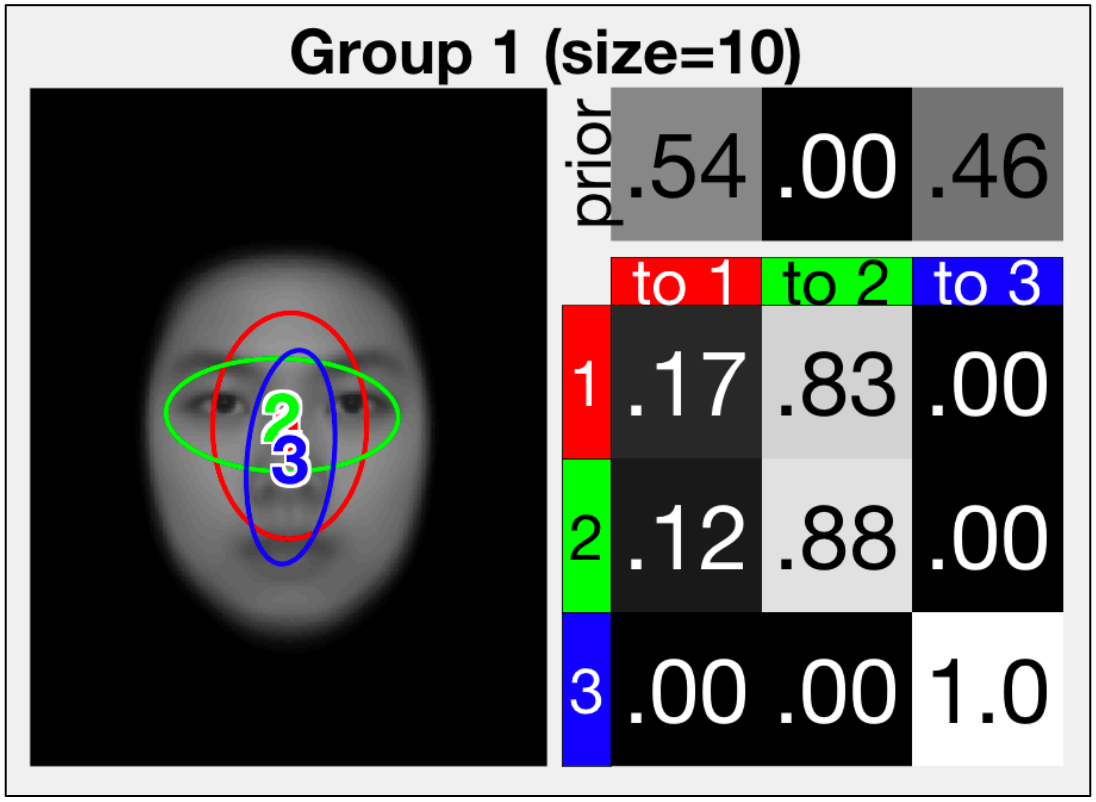
Viewing the Overall HMM

Figure 5

- Plot group HMM:

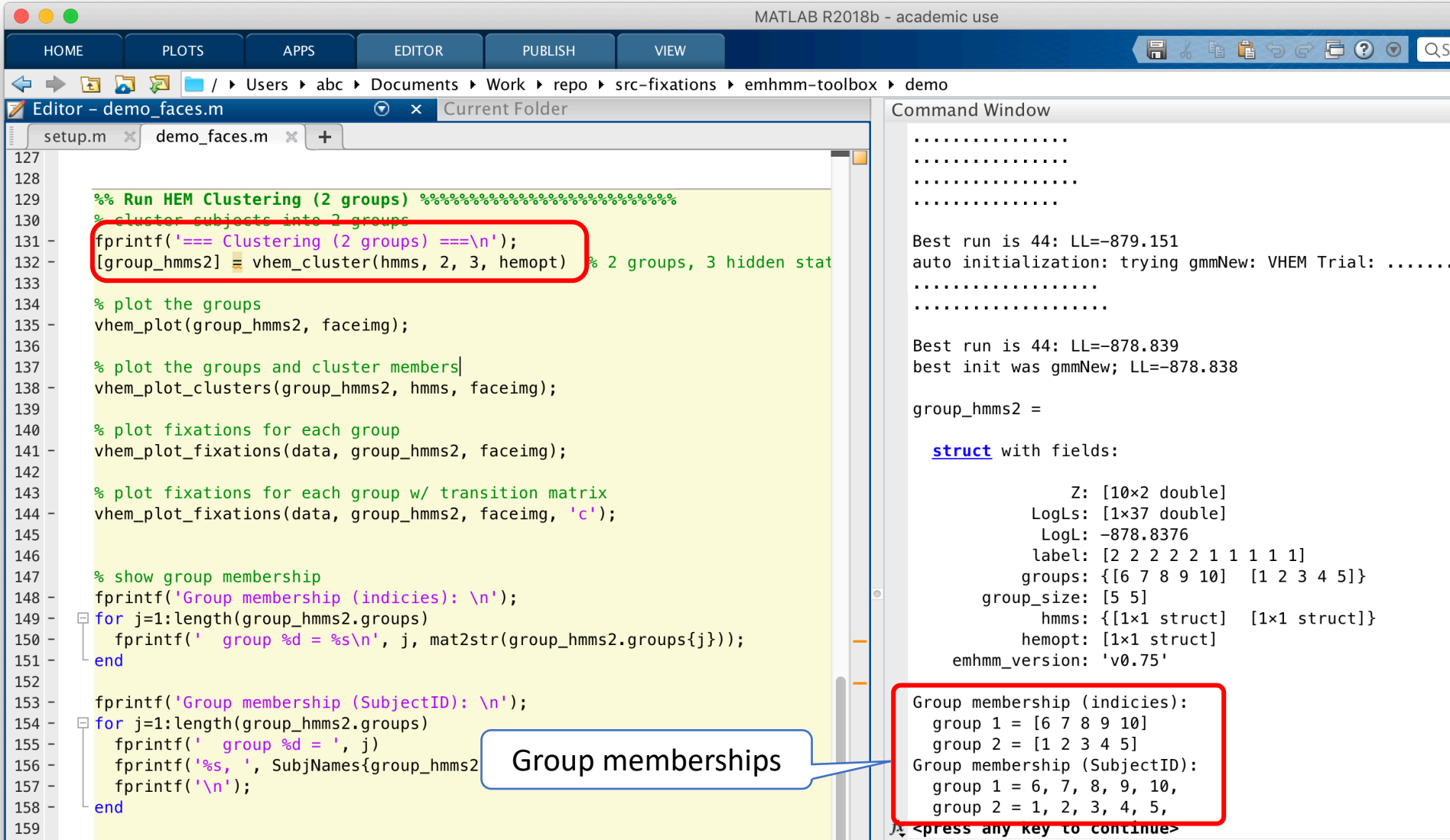
```
120
121 -   [all_hmms1] = vhem_cluster(hmms, 1, 3, hemopt)  % 1 g
122
123     % plot the overall HMM
124 -   vhem_plot(all_hmms1, faceimg);
125
```

- Output figure:



Clustering – 2 Groups

- Set number of clusters to **2**

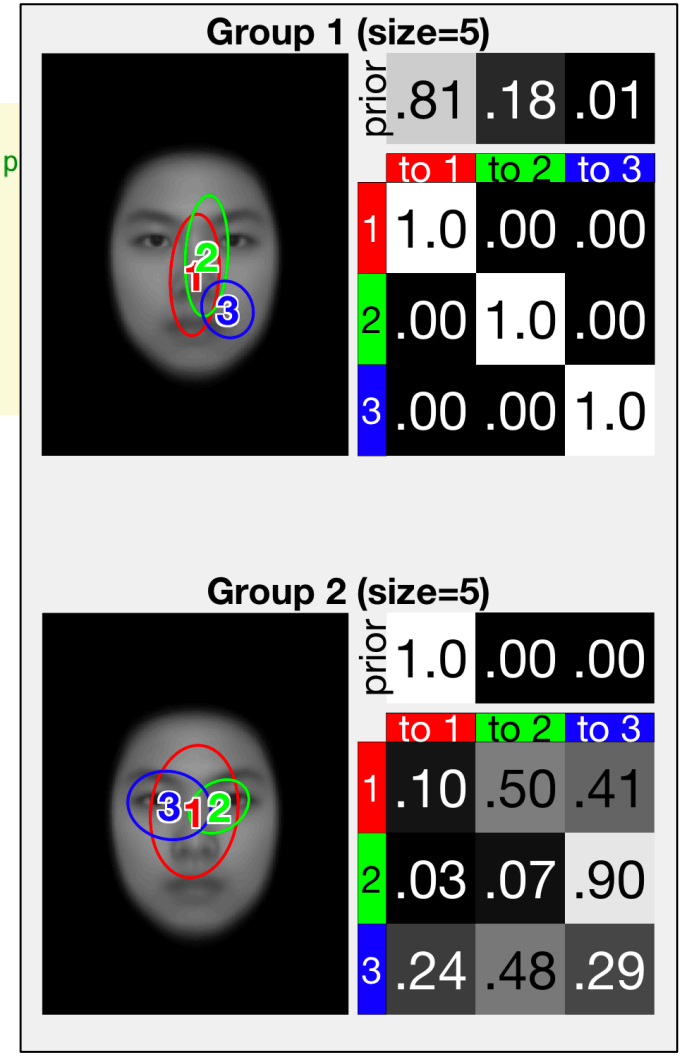


Viewing Group HMMs

Figure 6

- View 2 common stategies

```
131 -   fprintf('=== Clustering (2 groups) ===\n');
132 -   [group_hmms2] = vhem_cluster(hmms, 2, 3, hemopt)  % 2 group
133
134     % plot the groups
135 -   vhem_plot(group_hmms2, faceimg);
136
137     % plot the groups and cluster members
138 -   vhem_plot_clusters(group_hmms2, hmms, faceimg);
```



Visualize Groups with fixations

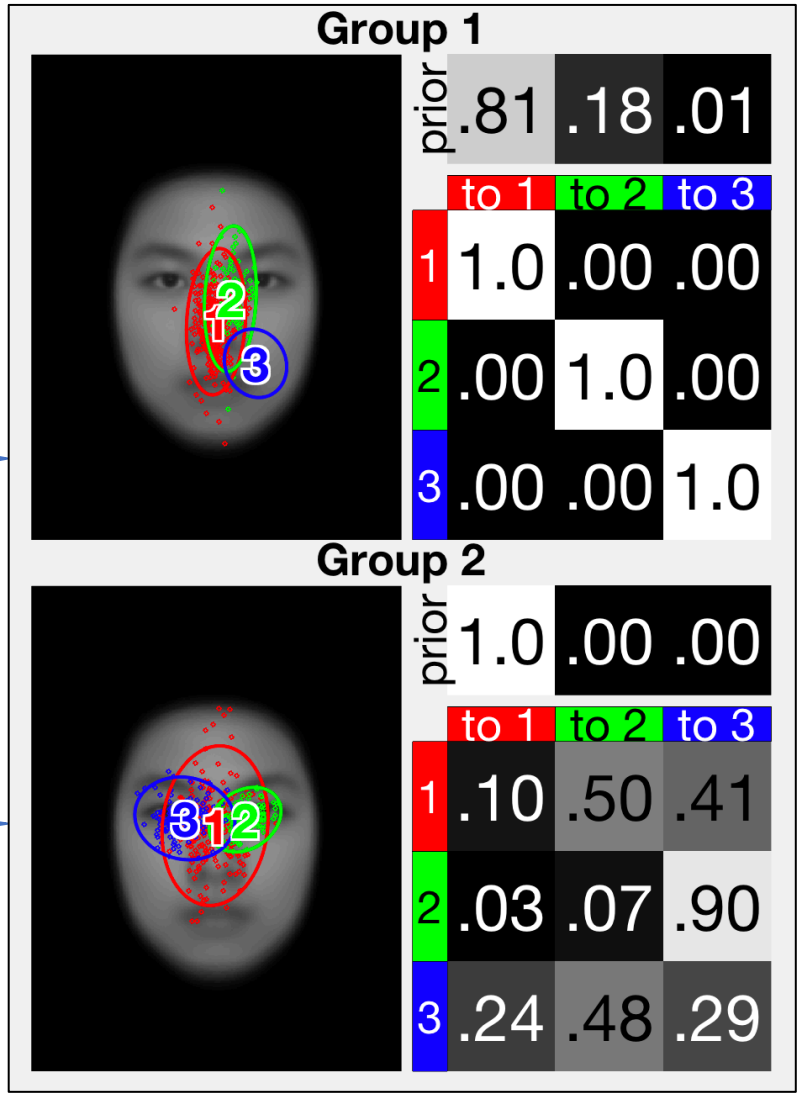
Figure 10

- Plot fixations with group HMMs:

```
142
143   % plot fixations for each group w/ transition matrix
144 - vhem_plot_fixations(data, group_hmms2, faceimg, 'c');
145
146
```

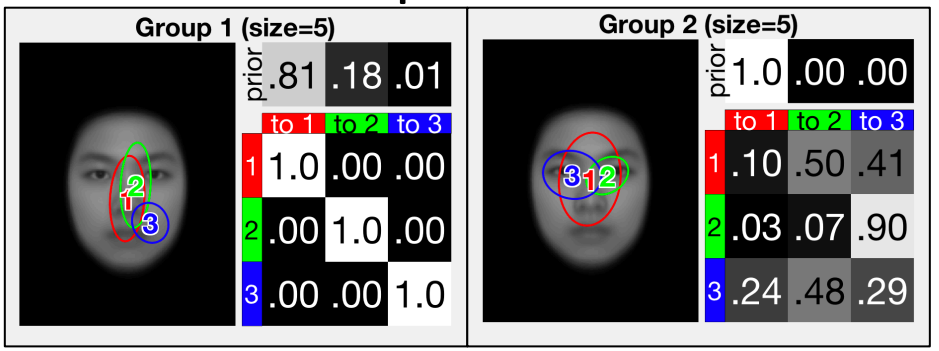
Shows fixations from all subjects assigned to that group.

Fixation color indicates the assigned ROI



Analysis: Most Probable ROI Sequence

- Compute most probable sequences (length **3**).



MATLAB R2018b - academic use

HOME | PLOTS | APPS | EDITOR | PUBLISH | VIEW

/ ▸ Users ▸ abc ▸ Documents ▸ Work ▸ repo ▸ src-fixations ▸ emhmm-toolbox ▸ demo

Editor - demo_faces.m

setup.m | demo_faces.m

```
161
162     %% get the most likely ROI sequences %%%%%%%%%
163 -   fprintf('\n*** top-5 most probable ROI sequences ***\n');
164 -   [seqs{1}, seqs_pr{1}] = stats_stateseq(group_hmms2.hmms{1}, 3);
165 -   [seqs{2}, seqs_pr{2}] = stats_stateseq(group_hmms2.hmms{2}, 3);
166
167     % show the sequences
168 -   fprintf('prob   : hmm1 ROI seq \n');
169 -   fprintf('-----------------\n');
170 -   for i=1:5
171 -     fprintf('%0.4f : ', seqs_pr{1}(i));
172 -     fprintf('%d ',     seqs{1}{i});
173 -     fprintf('\n');
174 -   end
175
176     % show the sequences
177 -   fprintf('\nprob   : hmm2 ROI seq \n');
178 -   fprintf('-----------------\n');
179 -   for i=1:5
180 -     fprintf('%0.4f : ', seqs_pr{2}(i));
181 -     fprintf('%d ',     seqs{2}{i});
182 -     fprintf('\n');
183 -   end
184
```

Current Folder

Command Window

```
  group 1 = [6 7 8 9 10]
  group 2 = [1 2 3 4 5]
Group membership (SubjectID):
  group 1 = 6, 7, 8, 9, 10,
  group 2 = 1, 2, 3, 4, 5,
<press any key to continue>

*** top-5 most probable ROI sequences ***
prob   : hmm1 ROI seq
-----------------
0.8067 : 1 1 1
0.1849 : 2 2 2
0.0084 : 3 3 3
0.0000 : 1 2 2
0.0000 : 1 1 2

prob   : hmm2 ROI seq
-----------------
0.4465 : 1 2 3
0.1934 : 1 3 2
0.1163 : 1 3 3
0.0954 : 1 3 1
0.0488 : 1 1 2
<press any key to continue>
```

Most-likely ROI sequence is 1-1-1, with 80.7% probability

Most-likely ROI sequence is 1-2-3, with 44.7% probability

Analysis: Group Differences

- Test if two group HMMs are different:
  - Using data from Group 1, calculate average log-likelihood difference under Group 1 and Group 2 HMMs.
    - This is an approximation of the KL divergence between Group 1 and Group 2.
  - Use t-test to check if the average log-likelihood difference is significantly different from 0 → the two HMMs are different.
  - Same process using data from Group 2.

Analysis: Group Differences

- Select Group data and perform t-test.

```
%% Statistical test %%%%%%%%%%%%%%%%%%%%%%%%%%%%
% collect data for group 1 and group 2
data1 = data(group_hmms2.groups{1});
data2 = data(group_hmms2.groups{2});

fprintf('\n*** t-test ***\n');

% run t-test for hmm1
[p, info, lld] = stats_ttest(group_hmms2.hmms{1}, group_hmms2.hmms{2}, data
fprintf('- test group hmm1 different from group hmm2: t(%d)=%g; p=%g\n', ir

% run t-test for hmm2
[p, info, lld] = stats_ttest(group_hmms2.hmms{2}, group_hmms2.hmms{1}, data
fprintf('- test group hmm2 different from group hmm1: t(%d)=%g; p=%g\n', ir


%% get mean log-likelihoods (e.g., to use with a correlation test) %%%%%%%%
fprintf('\n*** get mean LL for each subject ***\n');

[mLL1] = stats_meanll(group_hmms2.hmms{1}, data);
[mLL2] = stats_meanll(group_hmms2.hmms{2}, data);

fprintf('- mean LL for each subject under group hmm1:\n');
mLL1
fprintf('- mean LL for each subject under group hmm2:\n');
mLL2

%% save group models to file
fprintf('saving group models to %s\n', matfile_group);
save(matfile_group, 'hemopt', 'all_hmms1', 'group_hmms2', 'faceimg');
% later on you can use the following command to read back the models:
% load models_demo_faces_group.mat
```

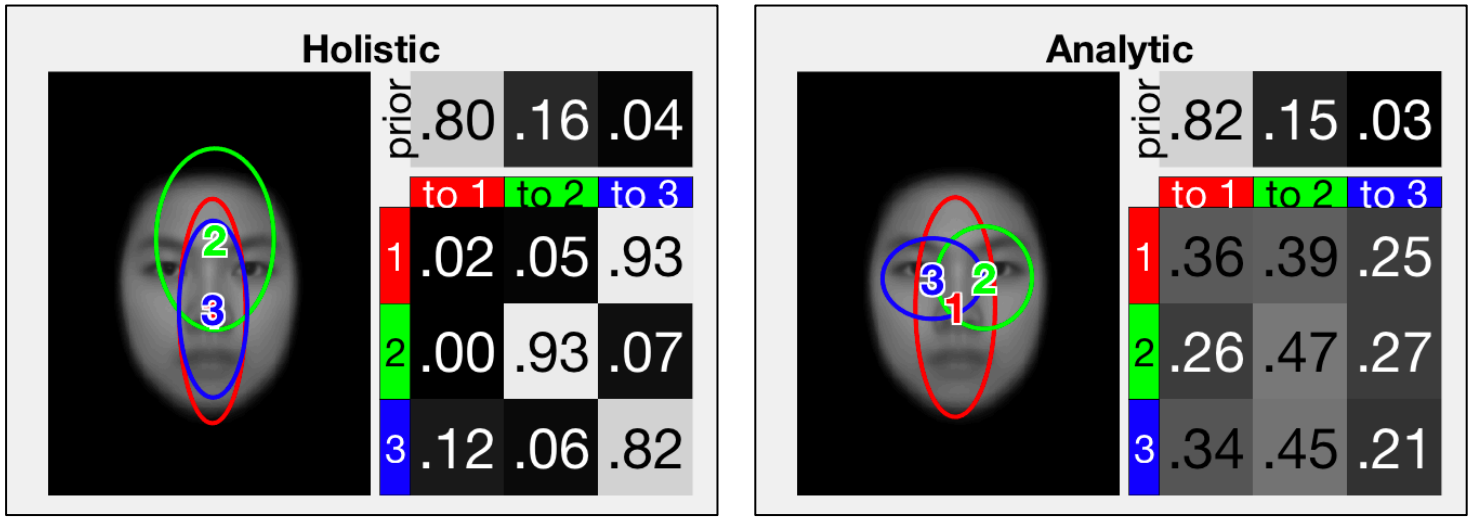
```
prob   : hmm1 ROI seq
-----------------
0.8067 : 1 1 1
0.1849 : 2 2 2
0.0084 : 3 3 3
0.0000 : 1 2 2
0.0000 : 1 1 2

prob   : hmm2 ROI seq
-----------------
0.4465 : 1 2 3
0.1934 : 1 3 2
0.1163 : 1 3 3
0.0954 : 1 3 1
0.0488 : 1 1 2
<press any key to continue>

*** t-test ***
- test group hmm1 different from group hmm2: t(4)=8.15202; p=0.000616171
- test group hmm2 different from group hmm1: t(4)=6.41628; p=0.00151614

*** get mean LL for each subject ***
- mean LL for each subject under group hmm1:

mLL1 =

  -11.5196   -9.9474  -12.3179  -10.6476  -12.3742   -8.5738   -8.1270   -

- mean LL for each subject under group hmm2:

mLL2 =

   -9.2684   -8.4376   -8.8677   -8.5927   -8.8478  -10.1217   -9.1723   -

saving group models to models_demo_faces_group.mat
>>
```

Both *p*-values show HMMs are significantly different.

script Ln 194

Representative Holistic/Analytic Models

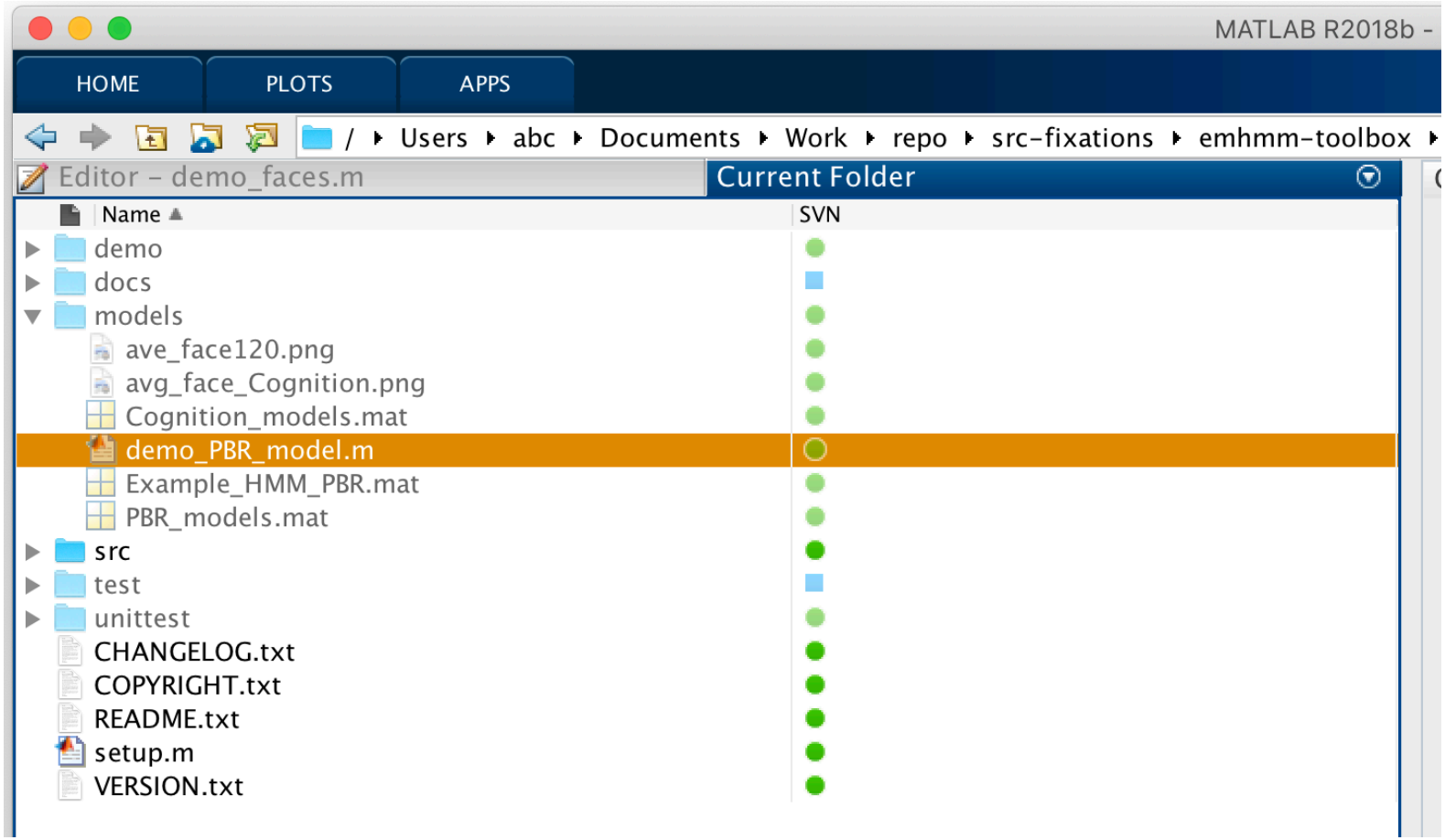
- Representative models for holistic/analytic strategies are available for computing H-A scale.



- Models are from: Cynthia Y.H. Chan, Antoni B. Chan, Tatia M.C. Lee, and Janet H. Hsiao, "Eye Movement Patterns in Face Recognition are Associated with Cognitive Decline in Older Adults." *Psychonomic Bulletin & Review*, 25(6): 2200-2207, Dec 2018.

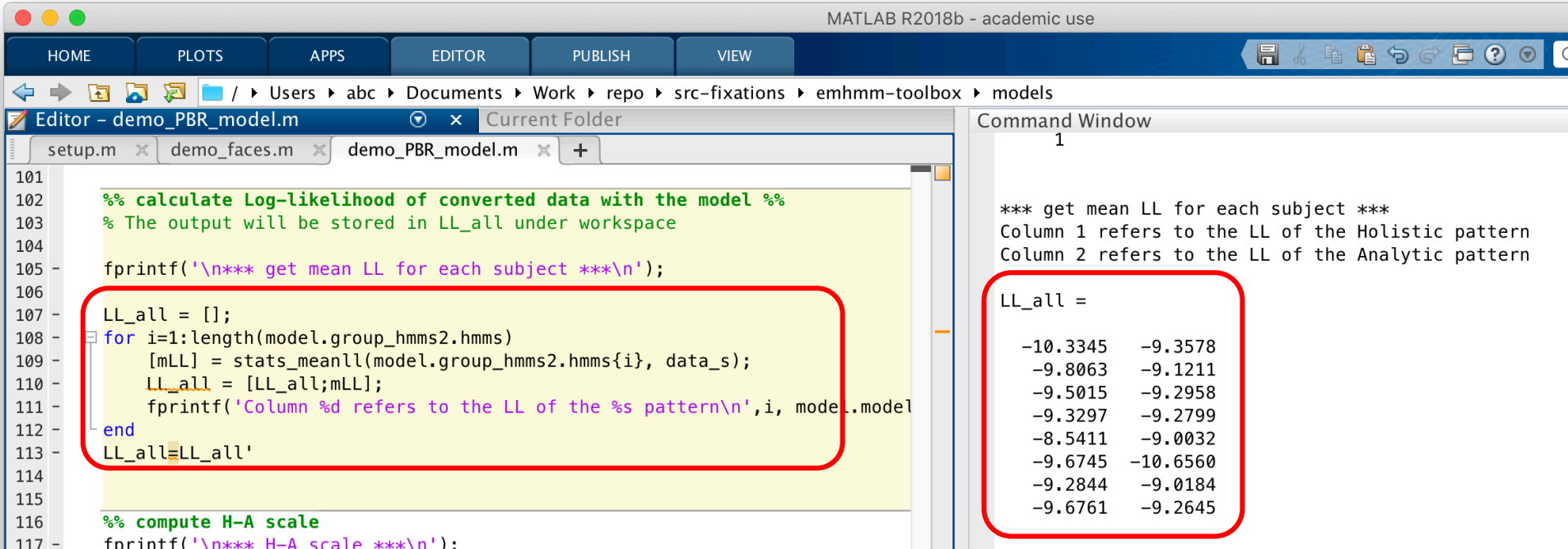
Demo Script for HA Models

- In the “models” directory of the toolbox:



Computing H-A Scale

- Using a eye fixation data from new subjects, compute H-A scale of each subject.
  - *Note: requires similar stimuli sizes and layout.*
- Compute log-likelihood under each group with `stats_meanll`.



Computing H-A Scale

- H-A scale: $HA = \dfrac{LLH - LLA}{|LLH| + |LLA|}$

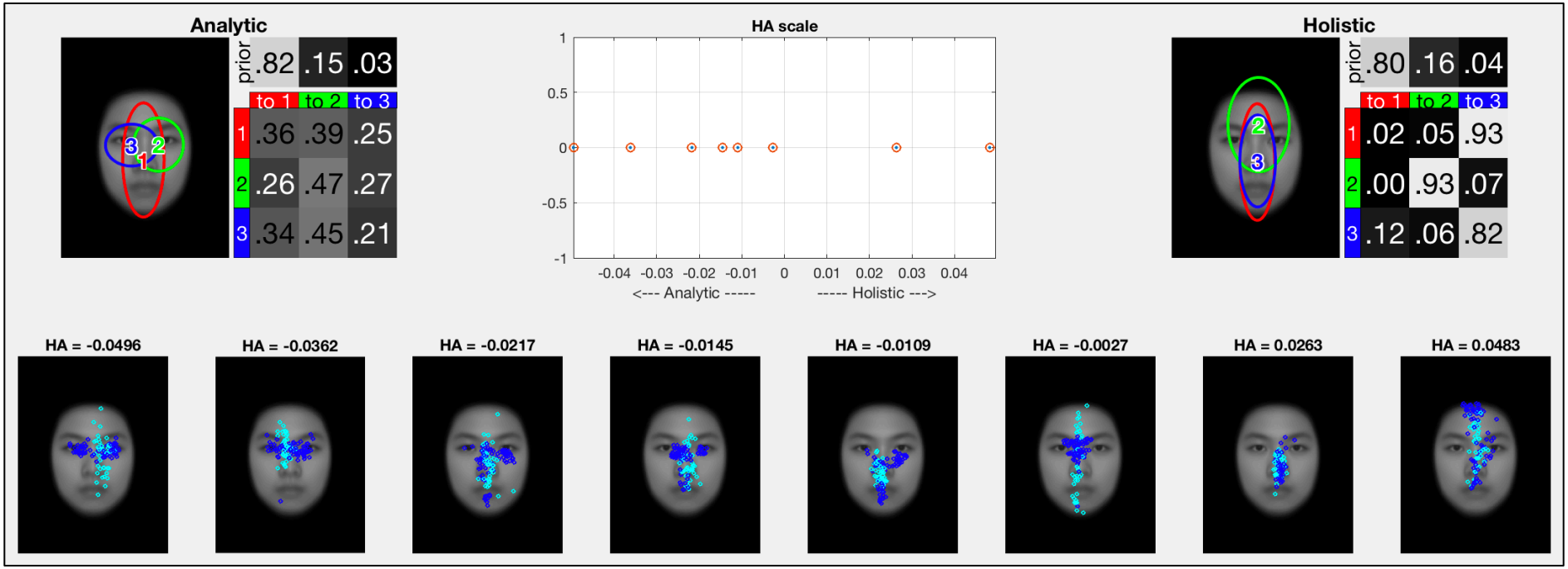
```
106
107 -   LL_all = [];
108 -   for i=1:length(model.group_hmms2.hmms)
109 -       [mLL] = stats_meanll(model.group_hmms2.hmms{i}, data_s);
110 -       LL_all = [LL_all;mLL];
111 -       fprintf('Column %d refers to the LL of the %s pattern\n',i, model.model
112 -   end
113 -   LL_all=LL_all'
114
115
116     %% compute H-A scale
117 -   fprintf('\n*** H-A scale ***\n');
118 -   HAscale = (LL_all(:,1) - LL_all(:,2)) ./ sum(abs(LL_all), 2)
119
120     % sort HA values in ascending order
121 -   [HAscale_sort, inds] = sort(HAscale);
122
123     % select 8 values
124 -   inds2 = inds(round(linspace(1,length(inds),8)));
125
126     %% make a nice plot
```

```
   -9.5015   -9.2958
   -9.3297   -9.2799
   -8.5411   -9.0032
   -9.6745  -10.6560
   -9.2844   -9.0184
   -9.6761   -9.2645


*** H-A scale ***

HAscale =

   -0.0496
   -0.0362
   -0.0109
   -0.0027
    0.0263
    0.0483
   -0.0145
   -0.0217
```

Positive HA indicate more holistic; negative HA is more analytic

Visualization

- Showing fixations of each subject and their HA value.

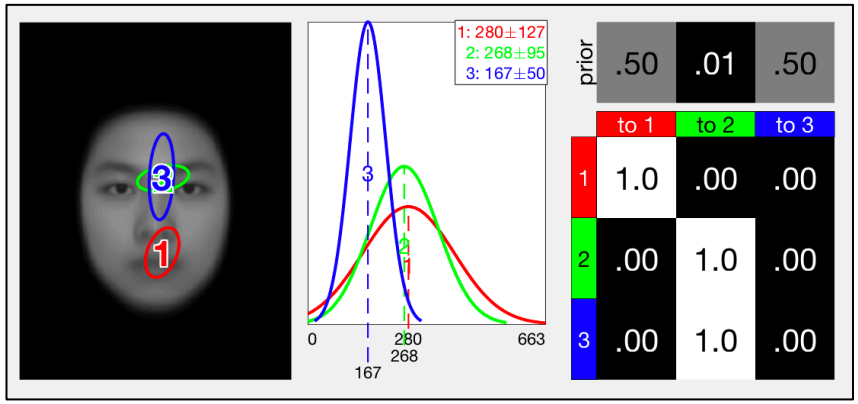


Most analytic

Most holistic

More Resources

- Analyzing eye fixation location & duration
  - `demo_faces_duration.m`



- JOV paper demo:
  - `demo_faces_jov_clustering.m`
  - `demo_faces_jov_compare.m`
- Converting coordinates between face images
  - `demo_conversion.m`
- More details about options for each function
  - See `emhmm-documentation.pdf`
  - MATLAB help: e.g., "`help vbhmm_learn`"

Questions?

- EMHMM toolbox:
  - http://visal.cs.cityu.edu.hk/research/emhmm/

- Acknowledgements:
  - This research was supported by the Research Grant Council of Hong Kong SAR: #17402814 and HKU 745210H for J.H. Hsiao; CityU 110513 and G-CityU109/14 for A.B. Chan. We also thank the HKU Seed Funding Programme for Basic Research (Project number 201311159131 to J.H. Hsiao.