Matlab for Neuroscientist 2:

Random number and psychtoolbox for visual stimulation

Sunghyon Kyeong

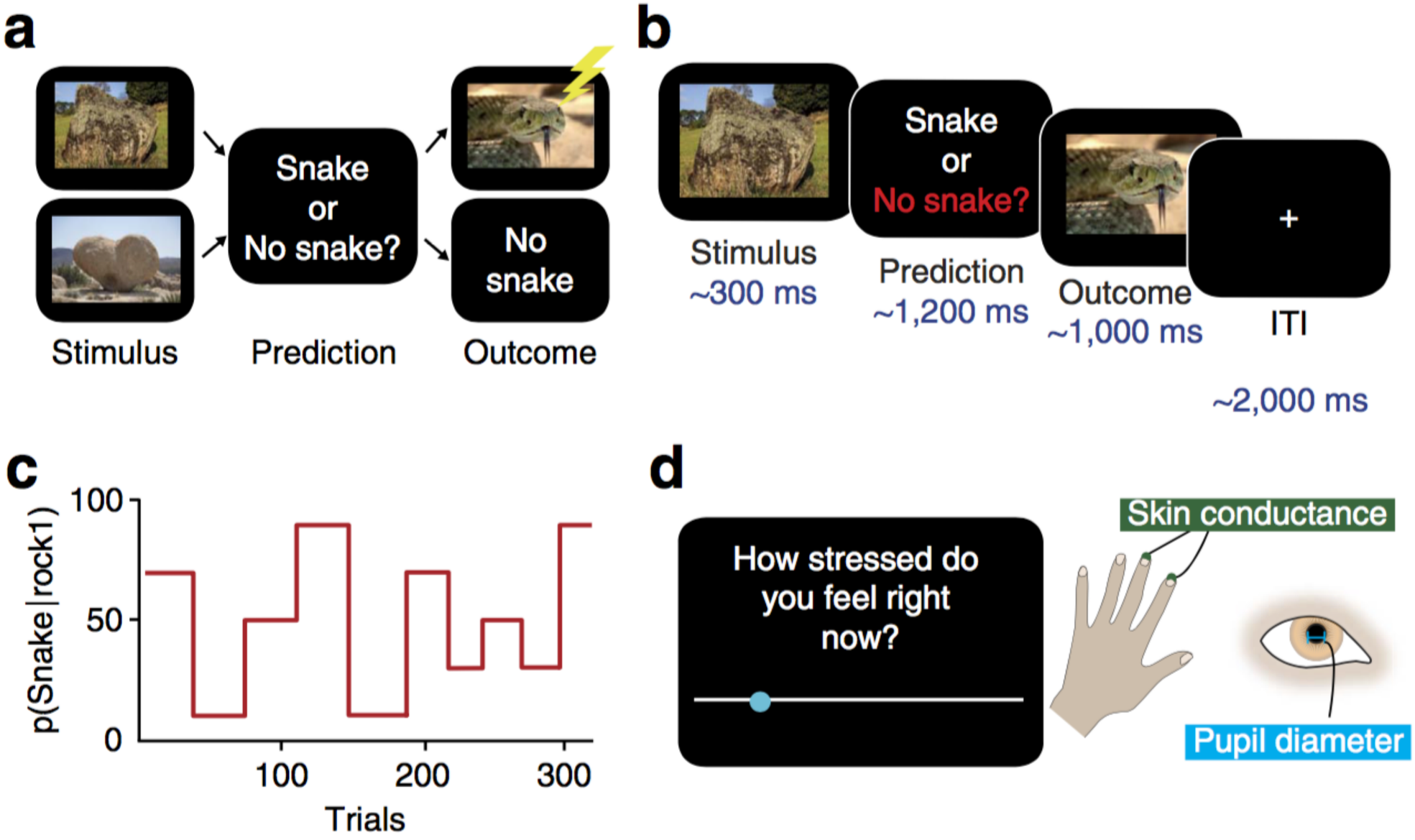
Severance Biomedical Science Institute,

Yonsei University College of Medicine

sunghyon.kyeong@gmail.com

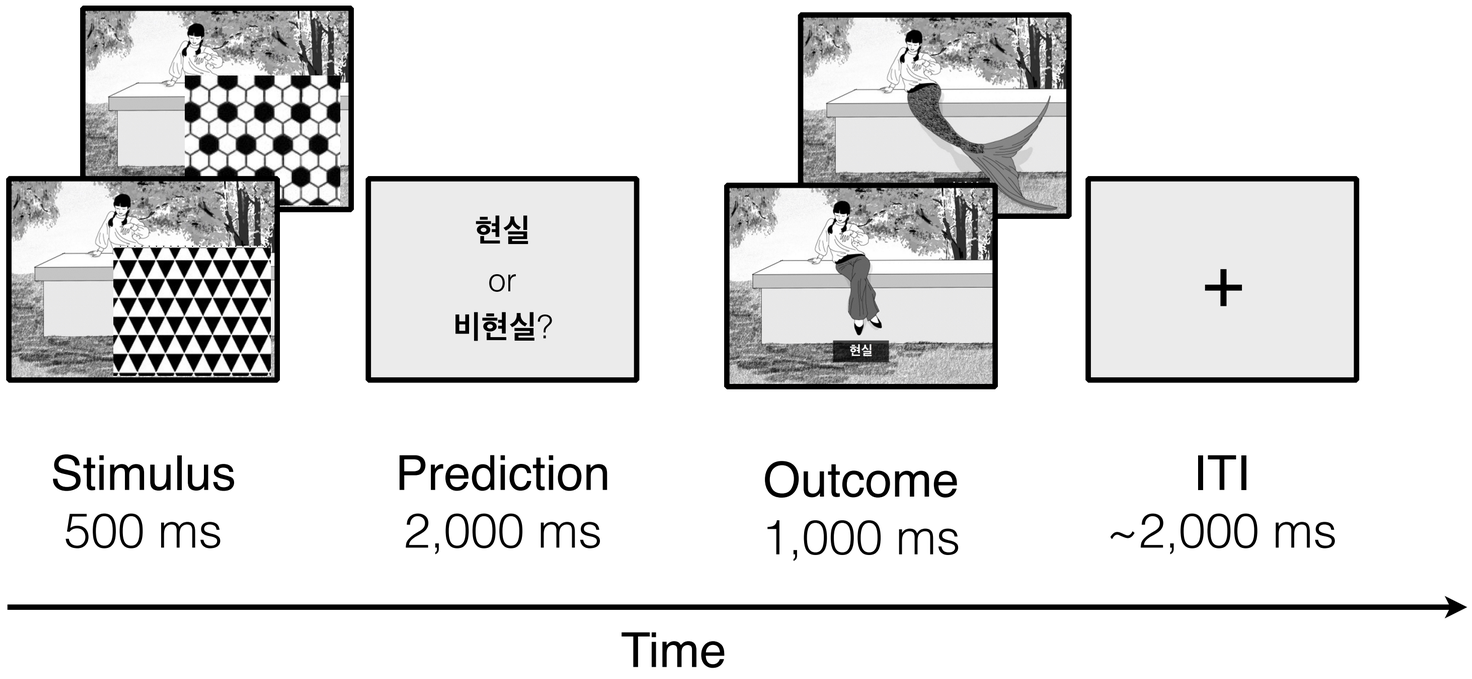
Contents

- Event related design (journal review)
- Random number generation
- Basics of Psychtoolbox
- Simple audio-visual stimulation
- Prediction Error Task using Psychtoolbox

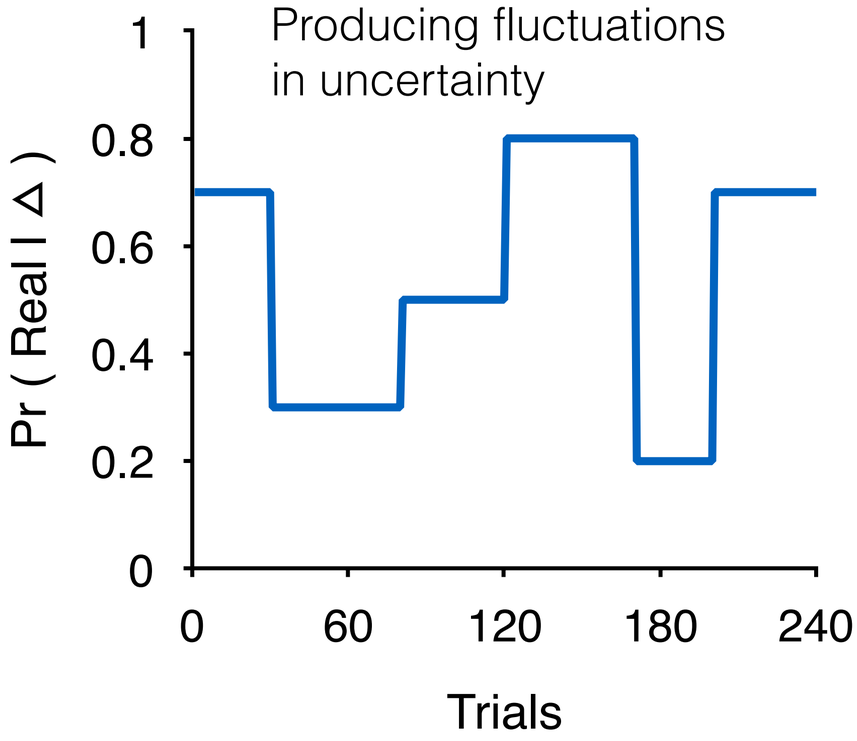

a
Snake
or
No snake?
No
snake
Stimulus
Prediction
Outcome
b
Snake
or
No snake?
+
Stimulus
~300 ms
Prediction
~1,200 ms
Outcome
~1,000 ms
ITI
~2,000 ms
c
p(Snake|rock1)
100
50
0
100
200
300
Trials
d
How stressed do
you feel right
now?
Skin conductance
Pupil diameter

Experimental design

a) Example trial



b) Probability curve



Journal review

◆ Human Brain Mapping 8:109–114(1999) ◆

Optimal Experimental Design for Event-Related fMRI

Anders M. Dale*

Nuclear Magnetic Resonance Center, Massachusetts General Hospital, Charlestown, Massachusetts

This analysis shows that statistical efficiency falls off dramatically as the ISI gets sufficiently short, if the ISI is kept fixed for all trials.

However, if the ISI is properly jittered or randomized from trial to trial, the efficiency improves monotonically with decreasing mean ISI.

Importantly, the efficiency afforded by such variable ISI designs can be more than 10 times greater than that which can be achieved by fixed ISI designs.

Journal review

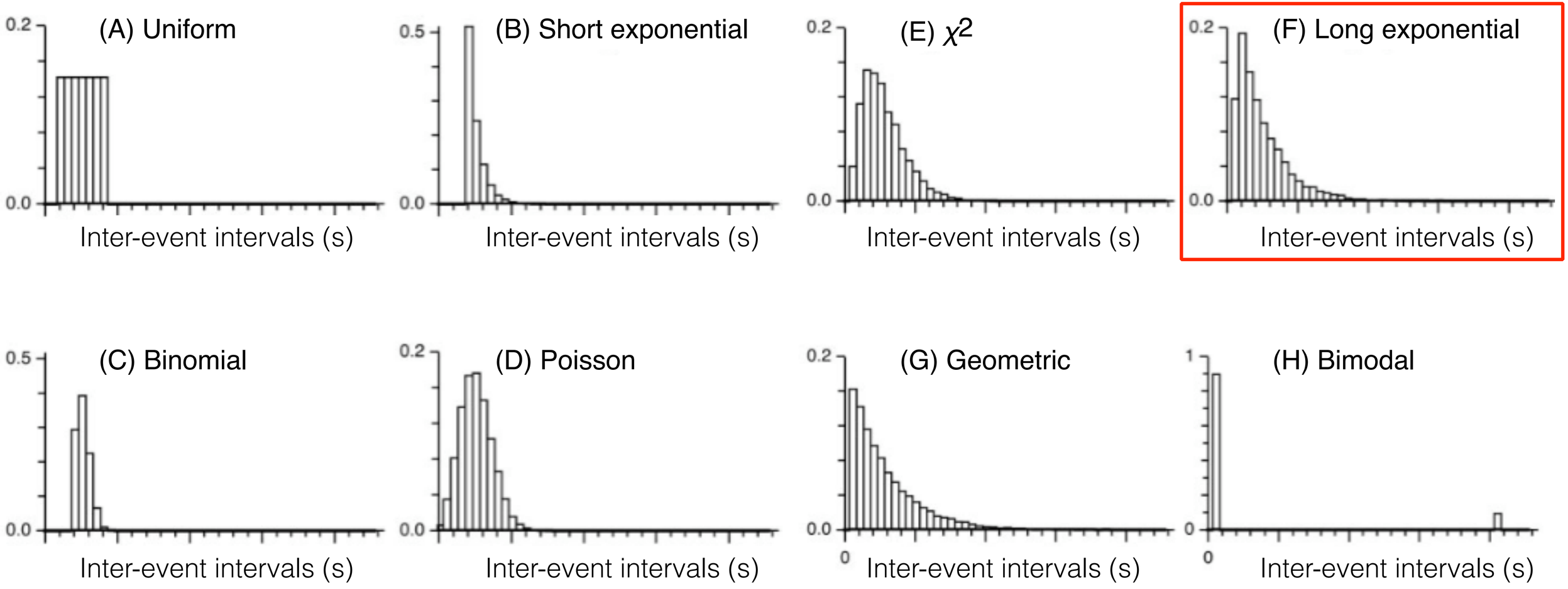
NeuroImage **14,** 1193–1205 (2001)
doi:10.1006/nimg.2001.0880, available online at http://www.idealibrary.com on IDEAL®

Improved Detection of Event-Related Functional MRI Signals Using Probability Functions

Gisela E. Hagberg,*,[1] Giancarlo Zito,*,† Fabiana Patria,* and Jerome N. Sanes*,‡,[2]

**Laboratory of Functional Neuroimaging, Fondazione Santa Lucia IRCCS, Rome, Italy; †Department of Neuroscience, Section of Neurology, University of Siena, Siena, Italy; and ‡Department of Neuroscience, Brown Medical School, Providence, Rhode Island 02912*

Received September 15, 2000



Probability density functions used for the selection of IEI. (A) Uniform; (B) short-decay exponential; (C) binomial; (D) Poisson; (E) χ^2; (F) long-decay exponential; (G) Geometric; and (H) bimodal distribution. Besides these distributions, IEI were selected by uniform permutation and according to a Latin square design.

Event sequences from the bimodal distribution, like **block designs**, had **the best performance for detection** and the poorest for estimation,

while **high estimation and detectability** occurred for the **long-decay exponential distribution.**

※ Efficiency indicates a measure of the expected accuracy of the estimator.

optseq2

Homepage: https://surfer.nmr.mgh.harvard.edu/optseq/

Generate optimal random sequences of events

Generate null time based on exponential distribution

Download the Linux version of optseq2.
Download the Linux x86_64 version of optseq2.
Download the MacOSX-PowerPC version of optseq2.
Download the MacOSX-Intel version of optseq2.
Download the Cygwin version of optseq2.

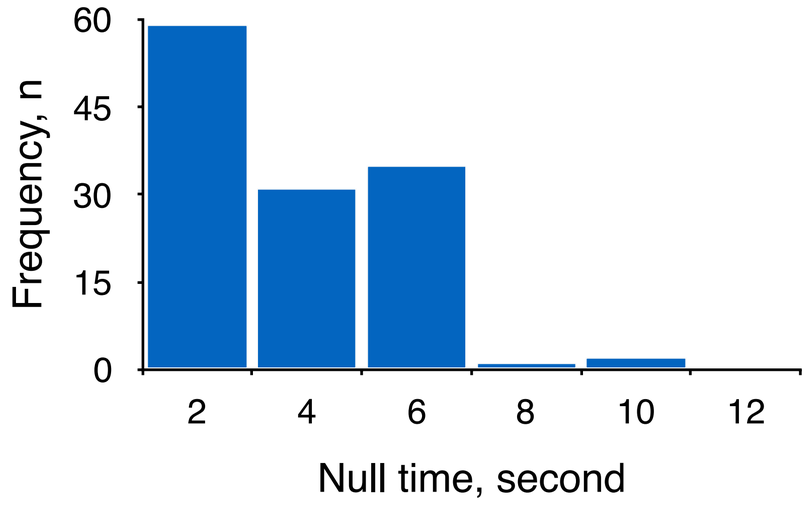
optseq2 - example

Create a schedule with three non-null event types for a run with 240 time points and TR = 2 sec. Event type 1 will have a duration of 2 sec with 40 repetitions. Event type 2 will have a duration of 4 sec with 20 repetitions, and event type 3 will have a duration of 6 sec with 30 repetitions. Set the PSD window to 20 sec to assure that the response to event type 3 can be captured; explicitly set the temporal resolution (ie, the dPSD) to one TR.

```
optseq2 --ntp 240 --tr 2 \
        --psdwin 0 20 2 \
        --ev evt1 2 40 \
        --ev evt2 4 20 \
        --ev evt3 6 30 \
        --nkeep 3 \
        --o ex2 \
        --nsearch 10000
```

```
  0.0000    1    2.000   1.0000          evt1
  2.0000    0    4.000   1.0000          NULL
  6.0000    1    2.000   1.0000          evt1
  8.0000    3    6.000   1.0000          evt3
 14.0000    1    2.000   1.0000          evt1
 16.0000    3    6.000   1.0000          evt3
 22.0000    1    2.000   1.0000          evt1
 24.0000    0    2.000   1.0000          NULL
 26.0000    1    2.000   1.0000          evt1
 28.0000    0    6.000   1.0000          NULL
 34.0000    1    2.000   1.0000          evt1
 36.0000    1    2.000   1.0000          evt1
 38.0000    3    6.000   1.0000          evt3
 44.0000    2    4.000   1.0000          evt2
 48.0000    0    2.000   1.0000          NULL
```

extract
null time

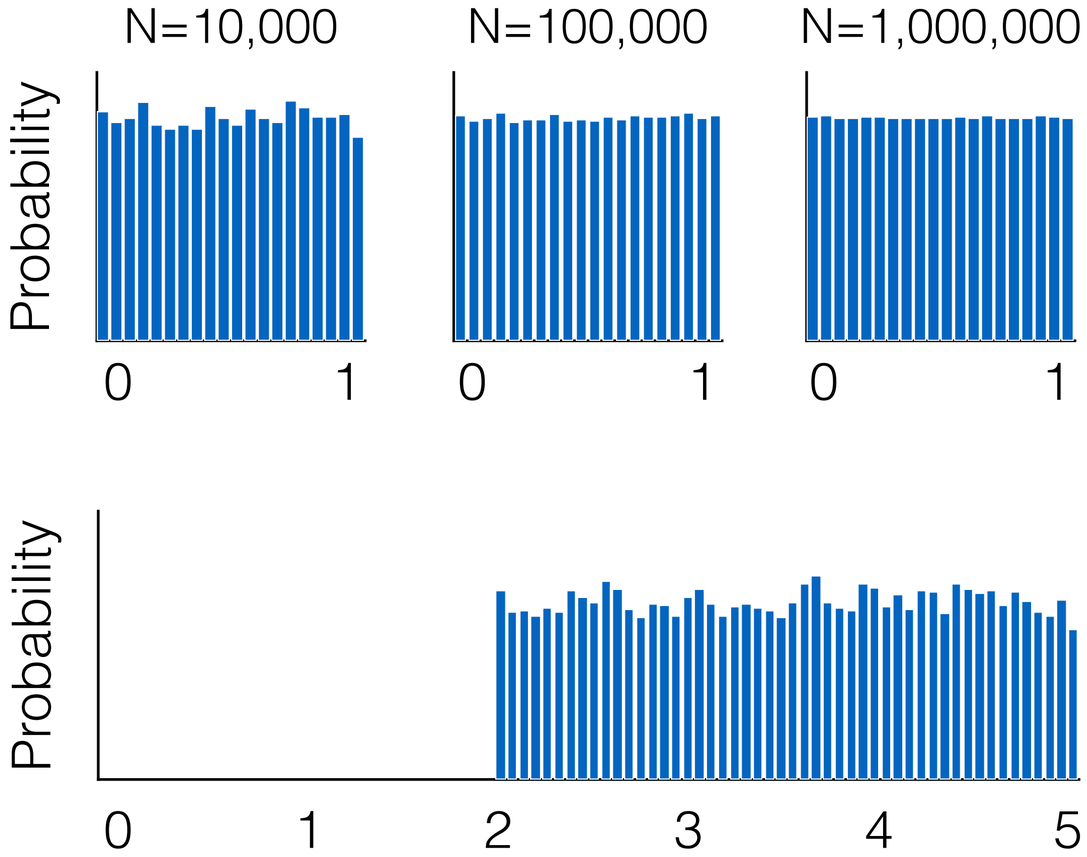


Random number generation in Matlab

Uniform distribution

uniform_dist.m

```
>> N = 10000;
>>
>> % Uniform distribution (0,1)
>> dist1 = rand(N,1);
>> figure;
>> hist(dist1,20);
>>
>>
>> % Uniform distribution (2,5)
>> dist2 = 3*dist1+2;
>> figure;
>> hist(dist2,50);
```

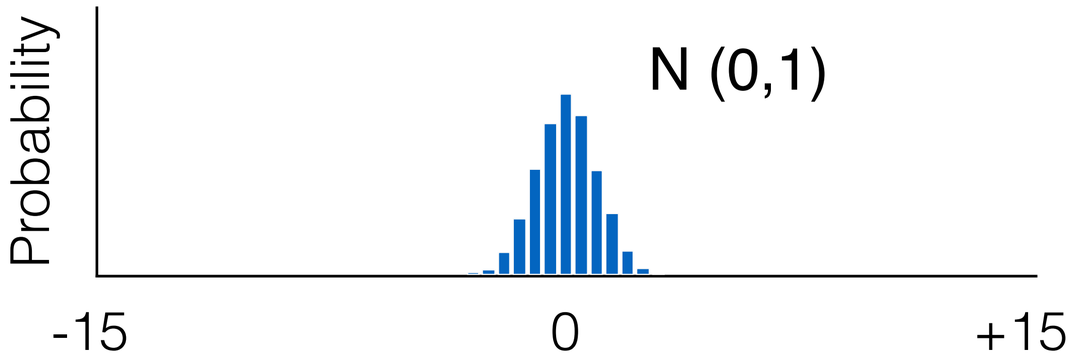


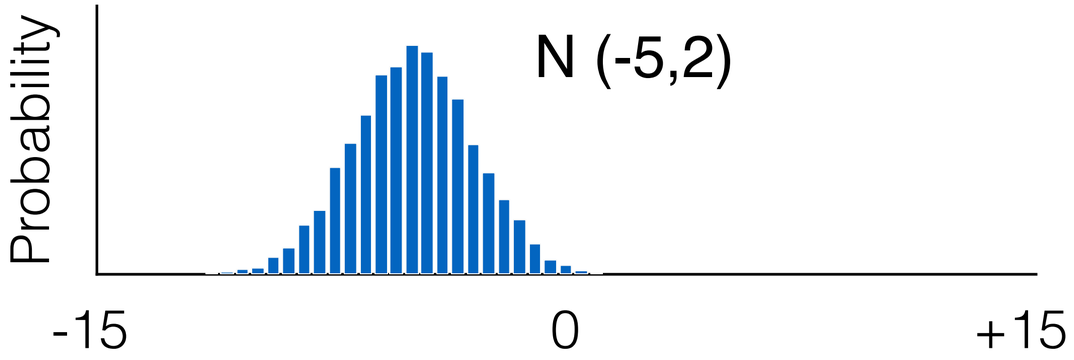
Gaussian distribution

gaussian_dist.m

= normal distribution, bell-shape distribution

```
>> N = 10000;
>>
>> % Gaussian distribution N(0,1)
>> dist1 = randn(N,1);
>> mu = mean(dist1);
>> SD = std(dist1);
>> fprintf('mean=%.1f, SD=%.2f\n',mu,SD);
>> figure;
>> hist(dist1,20);
>>
>>
>> % Gaussian distribution N(-5,2)
>> dist2 = 2*dist1-5;
>> mu = mean(dist2);
>> SD = std(dist2);
>> fprintf('mean=%.1f, SD=%.2f\n',mu,SD);
>> figure;
>> hist(dist2,20);
```

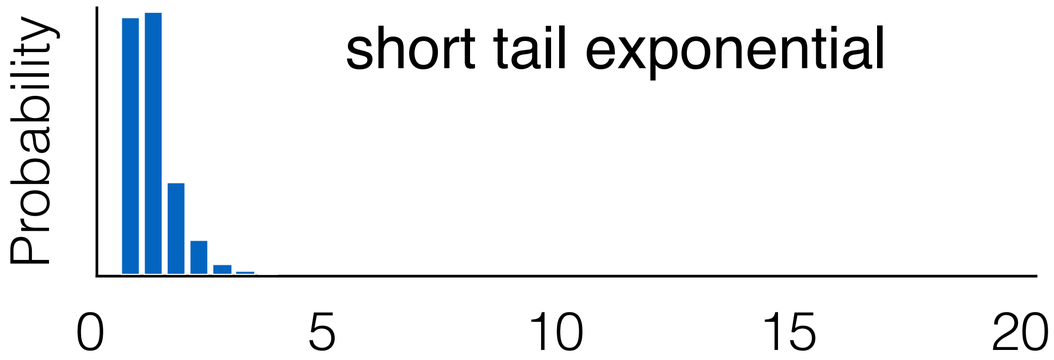


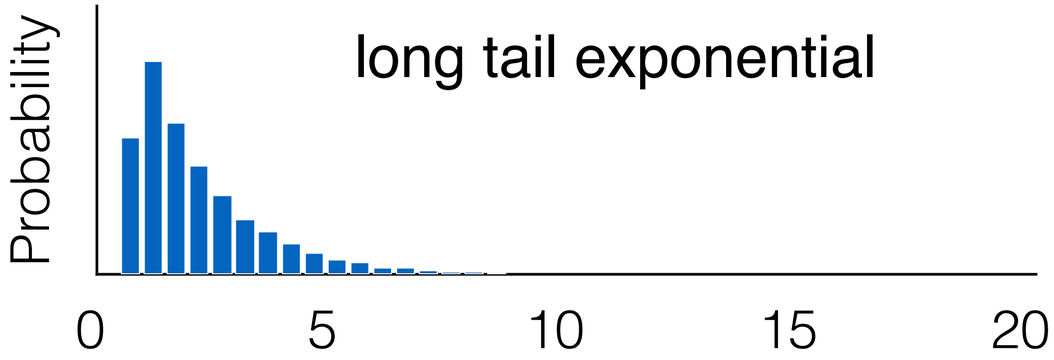


Exponential distribution

exponential_dist.m

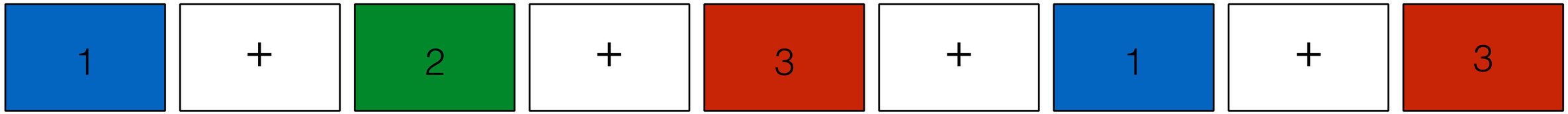
```
>> N = 10000;
>>
>> % Exponential distribution (short-tail)
>> mu = 1;
>> truncate_at = 0.5;
>> dist1 = exprnd(mu-truncate_at,N,1)+truncate_at;
>> mu = mean(dist1);
>> SD = std(dist1);
>> fprintf('mean=%.1f, SD=%.2f\n',mu,SD);
>> figure;
>> k=0:0.5:20;
>> hist(dist1,k);
>>
>>
>> % Exponential distribution (long-tail)
>> mu = 2;
>> truncate_at = 0.5;
>> dist2 = exprnd(mu-truncate_at,N,1)+truncate_at;
>> mu = mean(dist2);
>> SD = std(dist2);
>> fprintf('mean=%.1f, SD=%.2f\n',mu,SD);
>> figure;
>> hist(dist2,k);
```





Generate random sequence

How can we randomize the presentation orders (or sequences) of the blue, green, and red signs?

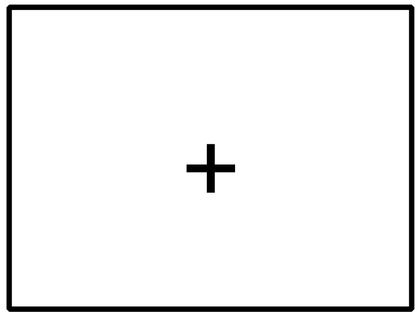


2 1 3 | 2 3 1 | 1 2 3 | 1 3 2 ← random seq → 1 2 2 | 1 3 2 | 3 3 3 | 1 1 2

```
>> mylist  = [1 2 3];
>> N_repeat = 4;
>>
>> full_list = [];
>> for i=1:N_repeat,
>>    idx = randperm(length(mylist));
>>    full_list = [full_list; mylist(idx)'];
>> end
```

```
>> mylist  = [1 2 3];
>> full_list = repmat(mylist,1,4);
>> idx = randperm(length(full_list));
>> full_list = full_list(idx);
```

Total Null Time



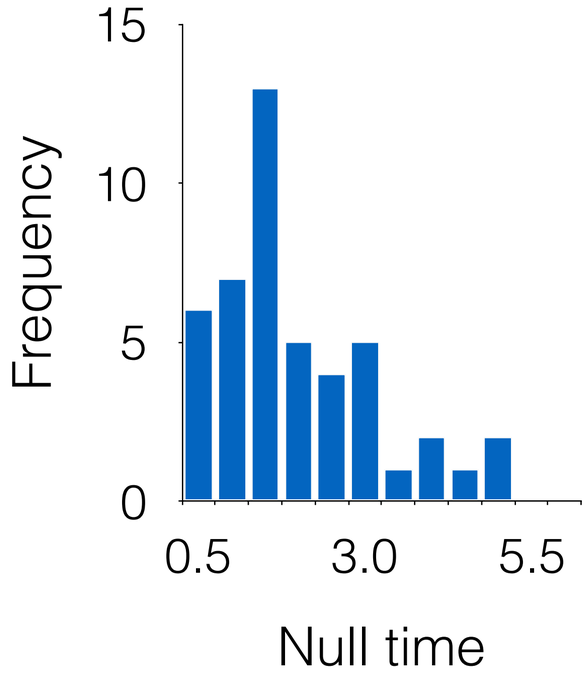
ITI
~2,000 ms

x 50 trials ≈ 100 sec

Generate Null Time (long tail)

in optseq2/demo.m

```
>> muNull   = 2;          % mean null time
>> nNull    = 50;         % the number of null events
>> Nkeep    = 4;          % the number of sequences
>> psdwin   = [0.5, 6];   % minimum and maximum null time
>> outflag = 'ex2'        % prefix of the output file name
>>
>> generate_null(muNull,nNull,Nkeep,psdwin,outflag);
```



Psychtoolbox Examples

http://psychtoolbox.org/

Psychtoolbox-3

Psychophysics Toolbox Version 3 (PTB-3) is a free set of Matlab and GNU Octave functions for vision and neuroscience research. It makes it easy to synthesize and show accurately controlled visual and auditory stimuli and interact with the observer.

Getting started

- Overview
- Versions
- System Requirements
- Download
- Donations please

Getting help

- Tutorial ← a lot of examples
- Function Reference
- FAQ
- Forum ← academic discussion
- Old Psychtoolbox website

Diving in

- Contribute to the project
- Wiki
- Bugs & Feature requests
- Git repository

Click to download

Support **Forum** | Automatic **Installer** | Download PTB **ZIP File** | View On **GitHub**

edit and fork this page on GitHub
Last updated on November 01, 2016

Overview

Psychtoolbox interfaces between Matlab or Octave and the computer hardware. The PTB core routines provide access to the display frame buffer and color lookup table, reliably synchronize with the vertical screen retrace, support sub-millisecond timing, expose raw OpenGL commands, support video playback and capture as well as low-latency audio, and facilitate the collection of observer responses. Ancillary routines support common needs like color space transformations and the QUEST threshold seeking algorithm.

Psychtoolbox has many active users, an active forum, and is widely cited. PTB-3 is based on the deprecated Psychophysics Toolbox Version 2 with its Matlab C extensions rewritten to be more modular and to use OpenGL on the back end. The current version supports Matlab R2012a+ and Octave 4.0.x on Linux, MacOSX, and Windows.

Psychtoolbox News

Psychtoolbox beta updated

kleinerm September 10, 2016

Psychtoolbox 3.0.13 "Boldly go where no one has gone before!" was released at 10th September 2016. As usual, the complete development history can be found in our GitHub repository. The release tag is "PTB_Beta-2016-09-10_V3.0.13", with the full tree and commit logs under the URL:

Release: https://github.com/Psychtoolbox-3/Psychtoolbox-3/tree/PTB_Beta-2016-09-10_V3.0.13

New features and improvements:

in Tutorial menu

(1) The bsics

Totally Minimal Demo: This demo aims to be as minimal as possible. It opens a screen and colours it grey. Thats it. But it demonstrates pretty much the steps you will always use when starting a program i.e. opening a on-screen window.

Totally Minimal Demo #2: This is the same as "TotallyMinimalDemo", except after opening the window we query numerous things about it. This demos how to get pretty much all the basic info you will need about the screen.

Accurate Timing Demo: This demo shows three basic ways in which to present a stimulus which would change on each frame. I demonstrate varying degrees of potential accuracy. *Please read the extensive comments in the demo closely, they are important and should not be ignored.* For presenting a stimulus which is unchanging, or changes less frequently then each frame, see the next demo.

Wait Frames Demo: Here we demonstrate how to update an image on the screen at a rate different to that of the monitors refresh rate. For example, you might want to update the stimulus every second, rather then at the frame rate of your monitor.

NOTE: For the purposes of subsequent demos we rarely specify timing precisely. This is *for demo purposes only*, so as to ensure that the demos will run on systems not capable of precise timing. When presenting visual stimuli in a real experiment you should use accurate timing. Overall timing on systems can vary greatly, both between different systems, and with different tasks on the same system. If timing is important to you then take the time to measure it. Don't leave things to chance.

Screen Coordinates Demo: In this demo you can move the mouse cursor around the screen and we write text on the screen letting you know what the X and Y pixel coordinates of the cursor are (rounded to the nearest pixel). This allows you to picture how screen coordinates are defined as well as how to query the position of the mouse.

(10) Full Experiments

In this section we will use the full knowledge we have gained to code some simple experiments. This will allow you to see how to construct an experimental script and how to use standard psychophysical techniques such as the method of constant stimuli.

Stroop Task Demo: This page will talk you through designing a simple experiment with Psychtoolbox. We will use a standard example: the Stroop Task. Code will be provided and you will hopefully learn about how to structure a piece of experimental code.

Orientation Threshold Demo: This demo will show you how to get an orientation threshold using a two alternative forced choice experiment, with the method of constant stimuli.

Change Blindness Demo: This demo shows you a basic change blindness experiment where two pictures alternate with one another, separated by a blank screen. Either an object in the image changes colour, or disappears and reappears. Press the space bar when you see the change. The length of time it has taken you to see the change is recorded. The code file needs to be placed in the same directory as the folder of images, which can be downloaded here. This demo is a modified version of one by Krista Ehinger.

Change Screen Color

lecture2/psychtoolbox/tutorials/changeScreenColor.m

```
>> % add path
>> lecturepath = '/Volumes/JetDrive/workshops/Matlab/lecture2/psychtoolbox/';
>> addpath(fullfile(lecturepath,'utils'));
>> addpath(fullfile(lecturepath,'init'));
>>
>> % Clear the workspace and the screen
>> sca;
>> close all;
>> clearvars;
>>
>>
>> % Initialize and default setting
>> screenSize = [800,600];     % Set screen size
>> load_standard_colors;       % Load standard colors
>> PsychDefaultSetup(2);       % default settings for Psychtoolbox
>> init_screen_setting;        % Set screen setting
>> init_keyresponse_setting;   % Set keyresponse setting
>>
>>
>> % Text for instructuon screen
>> Screen(windowPtr,'TextSize',20);
>> DrawFormattedText(windowPtr,'Example for changing screen color.\n\n Press
   Enter key to continue.','center','center',black);
>> Screen('Flip',windowPtr);
>> KbWait([], 2);
>>
>>
>> % Show a loading screen while creating graphics
>> DrawFormattedText(windowPtr,'Press r, g, or b to change screen
   color','center','center',black);
>> vbl =  Screen('Flip',windowPtr);
```

```
>> % Run until a key is pressed
>> cnt = 1;
>> color = [1 1 1];
>> % ListenChar(2);
>> keypressed = '';
>> while cnt<10
>>
>>     % Check the keyboard to see if a button has been pressed
>>     KbWait([], 2);
>>     [isKeyDown, secs, keyCode] = KbCheck;
>>     if strcmp(KbName(keyCode),'ESCAPE')
>>         break;
>>     elseif strcmp(KbName(keyCode),'r'),
>>         color = red;
>>     elseif strcmp(KbName(keyCode),'g')
>>         color = green;
>>     elseif strcmp(KbName(keyCode),'b')
>>         color = blue;
>>     else
>>         continue
>>     end
>>
>>     % Color the screen a specific color
>>     keypressed{cnt} = KbName(keyCode);
>>     Screen('FillRect', windowPtr, color);
>>
>>     % Flip to the screen
>>     Screen('Flip', windowPtr);
>>     cnt = cnt+1;
>> end
>> % ListenChar(0);
>> sca % Closes Screen (or sca)
```

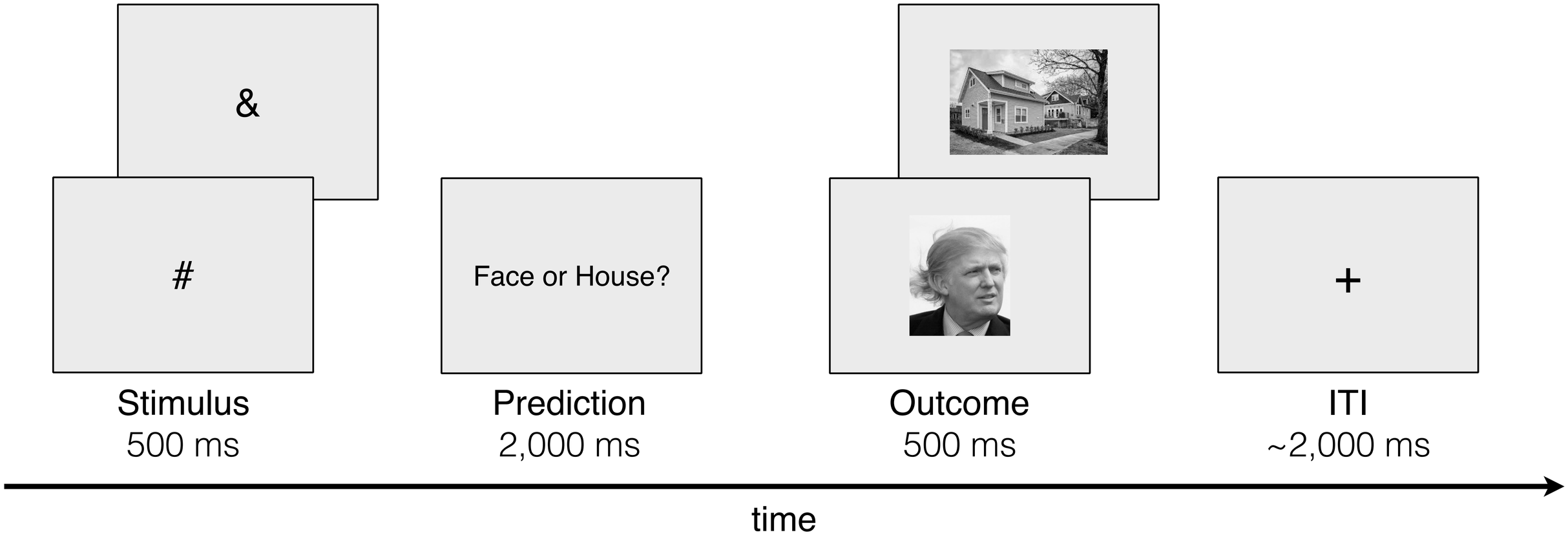
Response Time

lecture2/psychtoolbox/tutorials/ResponseTime.m

```
>> % addpath
>> lecturepath = '/Volumes/JetDrive/workshops/Matlab/lecture2/psychtoolbox/';
>> addpath(fullfile(lecturepath,'utils'));
>> addpath(fullfile(lecturepath,'init'));
>>
>> % Clear the workspace and the screen
>> sca;
>> close all;
>> clearvars;
>>
>>
>> % Initialize and default setting
>> screenSize = [800,600];     % Set screen size
>> load_standard_colors;       % Load standard colors
>> PsychDefaultSetup(2);       % default settings for Psychtoolbox
>> init_screen_setting;        % Set screen setting
>> init_keyresponse_setting;   % Set keyresponse setting
>>
>>
>>
>> % Text for instructuon screen
>> Screen(windowPtr,'TextSize',20);
>> DrawFormattedText(windowPtr,'Example for getting keyboard response time.
   \n\n Press Enter key to continue.','center','center',black);
>> Screen('Flip',windowPtr);
>> KbWait([], 2);
>>
>> %Show a loading screen while creating graphics
>> DrawFormattedText(windowPtr,'Start!','center','center',black);
>> Screen('Flip',windowPtr);
```

```
>> % Run until a key is pressed
>> cnt = 1;
>> color = [1 1 1];
>>
>> RT_list = nan(10,1);
>> key_list = '';
>> ListenChar(2);
>> while cnt<5
>>
>>     % Check the keyboard to see if a button has been pressed
>>     rndImg      = rand(100,100,3)*255; % creates random colored image
>>     rndImagePtr = Screen('MakeTexture', windowPtr, rndImg); % makes texture
>>     Screen('DrawTexture', windowPtr, rndImagePtr, []);   % draws to backbuffer
>>
>>     % Jitters
>>     WaitSecs(rand+.5);  % jitters pre-stim interval between .5 and 1.5 seconds
>>
>>     % Calculate response time
>>     starttime = Screen('Flip',windowPtr);      % swaps backbuffer to frontbuffer
>>     endtime = KbWait([], 2);
>>     [isKeyDown, t, keyCode] = KbCheck;
>>     RT_list(cnt)   = (endtime-starttime)*1000; % in ms unit
>>     key_list{cnt}  = KbName(keyCode);
>>
>>     % makes feedback string
>>     RTtext = sprintf('Response Time = %1.2f msecs',(endtime-starttime)*1000);
>>     DrawFormattedText(windowPtr,RTtext, 'center' ,'center', [255 0 255]); % shows
   RT
>>     vbl = Screen('Flip',windowPtr); % swaps backbuffer to frontbuffer
>>     Screen('Flip',windowPtr,vbl+1); % erases feedback after 1 second
>>     cnt = cnt+1;
>> end
>> ListenChar(0);
>> sca;   % Closes Screen
```

Example visual stimuli



Change Images on Screen (1/2)

lecture2/psychtoolbox/tutorials/changeImage.m

```
>> % addpath
>> lecturepath = '/Volumes/JetDrive/workshops/Matlab/lecture2/psychtoolbox/';
>> addpath(fullfile(lecturepath,'utils'));
>> addpath(fullfile(lecturepath,'init'));
>>
>> % Clear the workspace and the screen
>> sca;
>> close all;
>> clearvars;
>>
>>
>> % Initialize and default setting
>> screenSize = [800,600];     % Set screen size
>> load_standard_colors;       % Load standard colors
>> PsychDefaultSetup(2);       % default settings for Psychtoolbox
>> init_screen_setting;        % Set screen setting
>> init_keyresponse_setting;   % Set keyresponse setting
>>
>>
>> % Image path
>> imagePath = '/Volumes/JetDrive/workshops/Matlab/lecture2/psychtoolbox/tutorials/images';
>>
>>
>>
>> % Text for instructuon screen
>> Screen(windowPtr,'TextSize',20);
>> DrawFormattedText(windowPtr,'Example for changing images.\n\n Press any key to continue.','center','center',black);
>> Screen('Flip',windowPtr);
>> KbWait([], 2);
>>
>>
>> % Show a loading screen while creating graphics
>> DrawFormattedText(windowPtr,'Press [a] for Real and [s] for Unreal image.\n\n Press any key to start.','center','center',black);
>> Screen('Flip',windowPtr);
>> KbWait([], 2);
>>
>>
>> % Run until a key is pressed
>> cnt = 1;
>> ListenChar(2);
>> keypressed = cell(0);
>> list_RT = [];
```

Initialization of the stimulus!

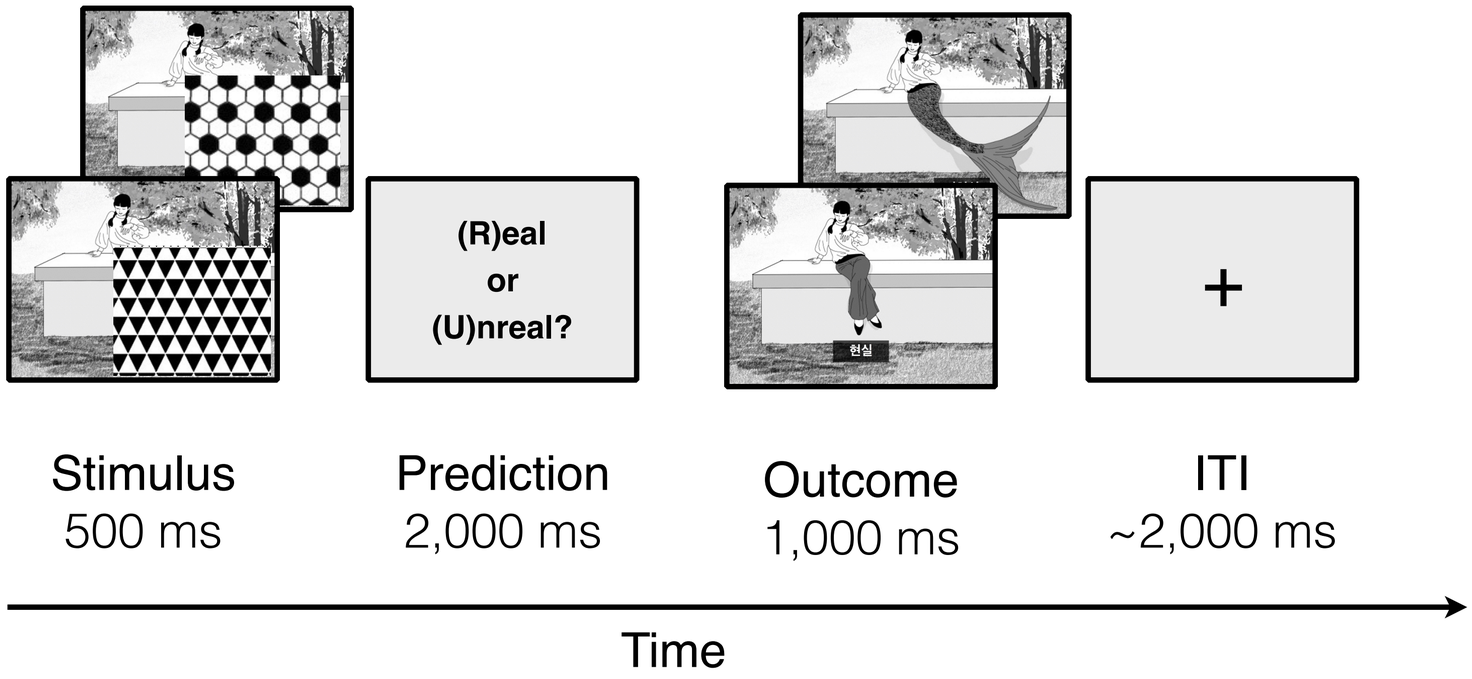
Change Images on Screen (1/2)

```
>> % Show a loading screen while creating graphics
>> DrawFormattedText(windowPtr,'Start!','center','center',black);
>> t0=Screen('Flip',windowPtr);
>> Screen(windowPtr,'TextSize',40);
>> while cnt<10
>>
>>     if cnt==1, t1 = t0; end
>>
>>     %--------------------------------------------------------------------
>>     % Show cue image
>>     %--------------------------------------------------------------------
>>     % Showing crosshairs
>>     id = randperm(2,1);
>>     if id==1,
>>         DrawFormattedText(windowPtr,'#','center','center',black);
>>         imgname = 'face';
>>     else
>>         DrawFormattedText(windowPtr,'&','center','center',black);
>>         imgname = 'house';
>>     end
>>     Screen('Flip',windowPtr);
>>     WaitSecs('UntilTime',t1 + 1);
>>     t1 = t1 + 1;  % Update time stamp
>>
>>
>>     %--------------------------------------------------------------------
>>     % Get keyboard responses
>>     %--------------------------------------------------------------------
>>     % Showing crosshairs
>>     DrawFormattedText(windowPtr,'Face vs. House?\n\n(a)           (s)
   ','center','center',black);
>>     Screen('Flip',windowPtr);
>>
>>     % Check the keyboard to see if a button has been pressed
>>     keyresponse_time = 2;
>>     endtime = KbWait([], 2, t1 + keyresponse_time );
>>     [isKeyDown, t, keyCode] = KbCheck;
>>
>>     if isKeyDown,
>>         RT = (endtime-t1);     % in ms unit
>>         response = KbName(keyCode); % convert to keyboard layout
>>         WaitSecs('UntilTime', t1 + keyresponse_time-RT);
>>     else
>>         RT = nan;
>>         response = nan;
>>     end
>>     % Save keyboard responses
>>     keypressed{cnt} = response;
>>     list_RT(cnt) = RT*1000;
>>     t1 = t1 + keyresponse_time;  % Update time stamp
>>
>>
>>     %--------------------------------------------------------------------
>>     % Show Image
>>     %--------------------------------------------------------------------
>>     % Load image
>>     imgNo = randperm(4,1);
>>     fn_image = fullfile(imagePath,sprintf('%s-%02d.jpg',imgname,imgNo));
>>     imageData = imread(fn_image);
>>
>>     % Make the image into a texture
>>     imageTexture = Screen('MakeTexture', windowPtr, imageData);
>>
>>     % Draw image into buffer
>>     Screen('DrawTexture', windowPtr, imageTexture, [], [], 0);
>>
>>     % Show image on screen
>>     Screen('Flip', windowPtr);
>>     WaitSecs('UntilTime',t1 + 1);
>>     t1 = t1 + 1;  % Update time stamp
>>
>>
>>     %--------------------------------------------------------------------
>>     % ITI
>>     %--------------------------------------------------------------------
>>     % Showing crosshairs
>>     DrawFormattedText(windowPtr,'+','center','center',black);
>>
>>     Screen('Flip',windowPtr);
>>     ITI = rand(1,1)*2+2;   % uniform ITI
>>     WaitSecs('UntilTime',t1 + ITI);
>>     t1 = t1 + ITI;  % Update time stamp
>>
>>     cnt = cnt+1;
>> end
>> ListenChar(0);
>> sca % Closes Screen (or sca)
>>
```

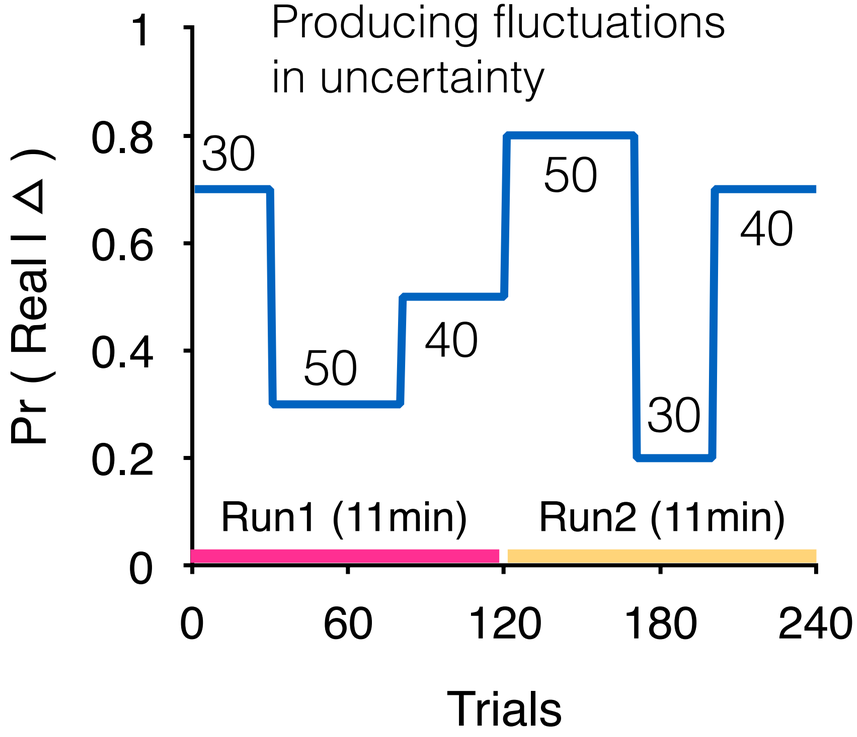
Prediction error

Experimental design

a) Example trial



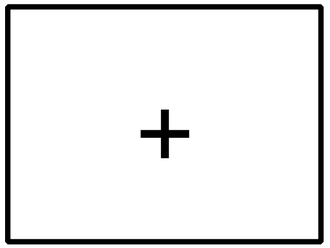
b) Probability curve



Total Null Time

lecture2/psychtoolbox/prediction_error/generate_null_time.m

```
>> addpath('/Volumes/JetDrive/workshops/Matlab/lecture2/optseq2');
>>
>> % Setting for null time
>> muNull  = 2;
>> nNull   = 30;
>> Nkeep   = 2;
>> psdwin  = [0.5 6];
>> outflag = 'null-n30';
>> generate_null(muNull,nNull,Nkeep,psdwin,outflag);
>>
>>
>> % Setting for null time
>> muNull  = 2;
>> nNull   = 40;
>> Nkeep   = 2;
>> psdwin  = [0.5 6];
>> outflag = 'null-n40';
>> generate_null(muNull,nNull,Nkeep,psdwin,outflag);
>>
>>
>> % Setting for null time
>> muNull  = 2;
>> nNull   = 50;
>> Nkeep   = 2;
>> psdwin  = [0.5 6];
>> outflag = 'null-n50';
>> generate_null(muNull,nNull,Nkeep,psdwin,outflag);
```



ITI
~2,000 ms

x 30 trials ≈ 60 sec

x 40 trials ≈ 80 sec

x 50 trials ≈ 100 sec

Generate Sequences

lecture2/psychtoolbox/prediction_error/generate_sequences.m

```
>> % Write sequences
>> nullpath = '/Volumes/JetDrive/workshops/Matlab/lecture2/psychtoolbox/prediction_error/optseq2';
>> outpath = '/Volumes/JetDrive/workshops/Matlab/lecture2/psychtoolbox/prediction_error';
>>
>>
>> % Create probability curve
>> prob = [0.7 0.3 0.5 0.8 0.2 0.7];
>> trials_in_block = [30 50 40 50 30 40];
>> nblock = length(trials_in_block);
>>
>> % Inizialize
>> prob_ts = [];
>> block = struct();
>>
>> for i=1:nblock;
>>
>>     % number of trials within a block
>>     ntrials = trials_in_block(i);
>>
>>     % load ITI-data
>>     if i<=3,
>>         fn_null = fullfile(nullpath,sprintf('null-n%d-%03d.csv',ntrials,1));
>>     else
>>         fn_null = fullfile(nullpath,sprintf('null-n%d-%03d.csv',ntrials,2));
>>     end
>>     ITI = dlmread(fn_null);
>>
>>     % Fluctuating uncertainty across blocks
>>     Pr_in_block = prob(i)*ones(ntrials,1);
>>     prob_ts = [prob_ts; Pr_in_block];
>>
>>     % Real / Unreal under cueT
>>     cueT_nReal = round(ntrials*prob(i)/2);
>>     cueT_nUnreal = ntrials/2 - cueT_nReal;
>>     seqs = [ones(cueT_nReal,1); zeros(cueT_nUnreal,1)];
>>     stim = cell(0);
>>     outcome = cell(0);
>>     for j=1:length(seqs),
>>         stim{j} = 'cueT';
>>         if seqs(j) == 1,
>>             outcome{j} = 'real';
>>         else
>>             outcome{j} = 'unreal';
>>         end
>>     end
>>
>>     % Real / Unreal under cueH
>>     cueH_nReal = round(ntrials*(1-prob(i))/2);
>>     cueH_nUnreal = ntrials/2 - cueH_nReal;
>>     seqs = [ones(cueH_nReal,1); zeros(cueH_nUnreal,1)];
>>     jj = j;
>>     for j=1:length(seqs),
>>         stim{jj+j} = 'cueH';
>>         if seqs(j) == 1,
>>             outcome{jj+j} = 'real';
>>         else
>>             outcome{jj+j} = 'unreal';
>>         end
>>     end
>>
>>
>>     % Shaffling sequence using permutation
>>     idx = randperm(ntrials);
>>     outcomeOrder = cell(0);
>>     stimOrder = cell(0);
>>     for j=1:ntrials,
>>         stimOrder{j} = stim{idx(j)};
>>         outcomeOrder{j} = outcome{idx(j)};
>>     end
>>
>>
>>     % Shaffling order of stimulus image
>>     try
>>         imageOrder = randperm(40,ntrials);
>>     catch
>>         imageOrder1 = randperm(40);
>>         imageOrder2 = randperm(40,ntrials-40);
>>         imageOrder = [imageOrder1,imageOrder2];
>>     end
>>
>>
>>     block(i).prob = prob(i);
>>     block(i).ntrials = ntrials;
>>     block(i).stimOrder = stimOrder;
>>     block(i).outcomeOrder = outcomeOrder;
>>     block(i).imageOrder = imageOrder;
>>     block(i).ITI = ITI;
>> end
>>
>> fn_out = fullfile(outpath,'sequence.mat');
>> save(fn_out,'block');
```

Check output sequences

>> **load** lecture2/psychtoolbox/prediction_error/sequence.mat

```
>> block

block =

1x6 struct array with fields:

    prob
    ntrials
    stimOrder
    outcomeOrder
    imageOrder
    ITI
```

```
>> block(1).stimOrder

ans =

  Columns 1 through 15

    'cueH'    'cueH'    'cueH'    'cueH'    'cueH'    'cueT'    'cueT'    'cueH'    'cueT'    'cueT'    'cueT'    'cueT'    'cueH'    'cueH'    'cueT'

  Columns 16 through 30

    'cueT'    'cueH'    'cueH'    'cueT'    'cueT'    'cueT'    'cueH'    'cueT'    'cueT'    'cueH'    'cueH'    'cueT'    'cueT'    'cueH'    'cueH'
```

```
>> block(1)

ans =

            prob: 0.7000
         ntrials: 30
       stimOrder: {1x30 cell}
    outcomeOrder: {1x30 cell}
      imageOrder: [1x30 double]
             ITI: [30x1 double]
```

```
>> block(1).outcomeOrder

ans =

  Columns 1 through 15

    'unreal'    'real'    'real'    'real'    'real'    'unreal'    'real'    'unreal'    'real'    'real'    'real'    'real'    'unreal'    'unreal'    'real'

  Columns 16 through 30

    'real'    'unreal'    'unreal'    'real'    'unreal'    'real'    'unreal'    'unreal'    'real'    'unreal'    'unreal'    'real'    'unreal'    'unreal'    'unreal'
```

Prediction Error (1/3)

lecture2/psychtoolbox/prediction_error/prediction_error.m

```
%--------------------------------------------------------------------------
% Clear the workspace and the screen
%--------------------------------------------------------------------------
sca;
close all;
clearvars;


%--------------------------------------------------------------------------
% Create experimental meta data
%--------------------------------------------------------------------------
subj = struct();
subj.name = input('Enter subject''s ID: ','s');
if isempty(subj.name)
    subj.name = 'test';
end



%--------------------------------------------------------------------------
% Set Path
%--------------------------------------------------------------------------
lecturepath = '/Volumes/JetDrive/workshops/Matlab/lecture2/psychtoolbox/';
PEpath      = fullfile(lecturepath,'prediction_error');
addpath(fullfile(lecturepath,'utils'));
addpath(fullfile(lecturepath,'init'));



%--------------------------------------------------------------------------
% Initialize and default setting
%--------------------------------------------------------------------------
% screenSize = 'full';
screenSize = [1024,768];   % Set screen size
load_standard_colors;      % Load standard colors
PsychDefaultSetup(2);      % default settings for Psychtoolbox
init_screen_setting;       % Set screen setting
init_keyresponse_setting;  % Set keyresponse setting
```

```
%--------------------------------------------------------------------------
% Create directories to save resulting files
%--------------------------------------------------------------------------
RESPpath = fullfile(PEpath,'RESP'); mkdir(RESPpath);

%--------------------------------------------------------------------------
% Load sequence information
%--------------------------------------------------------------------------
fn_sequence = fullfile(PEpath,'sequence1.mat');  % location of sequence1.mat
load(fn_sequence);
nBlocks = length(block);

stimulusDuration = 0.5;
outcomeDuration  = 1.0;
maxResponseTime  = 2.0;


%--------------------------------------------------------------------------
% Show instruction
%--------------------------------------------------------------------------
str_messages = 'Are you Ready? \nPress any key to start!';
show_message_on_screen(str_messages, windowRect, windowPtr, [], [], black, gray);
KbWait([], 2);


%--------------------------------------------------------------------------
% Show a loading screen while waiting a Trigger signal from MRI
%--------------------------------------------------------------------------
show_message_on_screen('Initializing...', windowRect, windowPtr, [], 2, black, gray);


%--------------------------------------------------------------------------
% Additional setting before showing eventInfos
%--------------------------------------------------------------------------
Priority(topPriorityLevel);
ListenChar(2);
```

Prediction Error (2/3)

lecture2/psychtoolbox/prediction_error/prediction_error.m

```
%--------------------------------------------------------------------------
% Run #1 (1-3 blocks)
%--------------------------------------------------------------------------
ST = clock;  % Get time stamp
Screen(windowPtr,'TextSize',60);
show_message_on_screen('+', windowRect, windowPtr, [], 2, black, gray);
% Variable initialize
eventInfo = struct();
cnt = 1;
tic;
timePtr = GetSecs;
for i=1:3,

    % Get sequence for each block
    prob         = block(i).prob;
    ntrials      = block(i).ntrials;
    stimOrder    = block(i).stimOrder;
    outcomeOrder = block(i).outcomeOrder;
    imageOrder   = block(i).imageOrder;
    ITI          = block(i).ITI;

    for j=1:ntrials,

        % Stimulus (duration = 0.5 sec)
        imgName = sprintf('A%02d-%s.png',imageOrder(j),stimOrder{j});
        fn_stim = fullfile(PEpath,'images',imgName);
        eventInfo(cnt).trial = cnt;
        eventInfo(cnt).prob = prob;
        eventInfo(cnt).blockid = i;
        eventInfo(cnt).ntrials = ntrials;
        eventInfo(cnt).stimOnset = toc;
        eventInfo(cnt).stimType = stimOrder{j};
        timePtr = show_image_on_screen(fn_stim, windowPtr, timePtr, stimulusDuration, gray);


        % Prediction (duration = 2 sec)
        eventInfo(cnt).prediction = toc;
        Screen(windowPtr,'TextSize',30);
        show_message_on_screen('(R)eal or (U)nreal ?', windowRect, windowPtr, [], [], black, gray);
        [rtime, resp, timePtr] = get_keyboard_response(timePtr, maxResponseTime);
        eventInfo(cnt).respTime = rtime;
        eventInfo(cnt).keyPressed = resp;

        % Outcome (duration = 1 sec)
        imgName = sprintf('A%02d-%s.png',imageOrder(j),outcomeOrder{j});
        fn_stim = fullfile(PEpath,'images',imgName);
        eventInfo(cnt).outcomeOnset = toc;
        eventInfo(cnt).outcomeType = outcomeOrder{j};
        timePtr = show_image_on_screen(fn_stim, windowPtr, timePtr, outcomeDuration, gray);


        % Inter-trial interval (duration ~ 2 sec)
        eventInfo(cnt).ITI = toc;
        Screen(windowPtr,'TextSize',60);
        timePtr = show_message_on_screen('+',windowRect, windowPtr, timePtr, ITI(j), black,
gray);

        % Count the number of trials
        cnt = cnt+1;
    end
end

% Write eventInfo and Responses
subj.time = sprintf('%04d-%02d-%02d %02dj%02dm%02ds',fix(clock));
fn_experiment = fullfile(RESPpath,sprintf('%s-run1-%s.csv', subj.name, subj.time));
write_to_csvfile(fn_experiment,eventInfo);


%--------------------------------------------------------------------------
% Inter-session interval
%--------------------------------------------------------------------------
str_messages = '-End of Run #1-';
show_message_on_screen(str_messages, windowRect, windowPtr, [], 2, black, gray);
str_messages = 'You did a good job! \nPress any key to proceed the second round';
show_message_on_screen(str_messages, windowRect, windowPtr, [], [], black, gray);
KbWait([], 2);
```

Prediction Error (3/3)

lecture2/psychtoolbox/prediction_error/prediction_error.m

```
%--------------------------------------------------------------------------
% Run #2 (4-6 blocks)
%--------------------------------------------------------------------------
Screen(windowPtr,'TextSize',60);
show_message_on_screen('+', windowRect, windowPtr, [], 2, black, gray);
% Variable initialize
eventInfo = struct();
RESP = struct();
cnt = 1;
tic;
timePtr = GetSecs;
for i=4:6,

    % Get sequence for each block
    prob         = block(i).prob;
    ntrials      = block(i).ntrials;
    stimOrder    = block(i).stimOrder;
    outcomeOrder = block(i).outcomeOrder;
    imageOrder   = block(i).imageOrder;
    ITI          = block(i).ITI;


    for j=1:ntrials,

        % Stimulus (duration = 0.5 sec)
        imgName = sprintf('A%02d-%s.png',imageOrder(j),stimOrder{j});
        fn_stim = fullfile(PEpath,'images',imgName);
        eventInfo(cnt).trial = cnt;
        eventInfo(cnt).prob = prob;
        eventInfo(cnt).blockid = i;
        eventInfo(cnt).ntrials = ntrials;
        eventInfo(cnt).stimOnset = toc;
        eventInfo(cnt).stimType = stimOrder{j};
        timePtr = show_image_on_screen(fn_stim, windowPtr, timePtr, stimulusDuration, gray);


        % Prediction (duration = 2 sec)
        eventInfo(cnt).prediction = toc;
        Screen(windowPtr,'TextSize',30);
        show_message_on_screen('(R)eal or (U)nreal ?', windowRect, windowPtr, [], [], black, gray);
        [rtime, resp, timePtr] = get_keyboard_response(timePtr, maxResponseTime);
        eventInfo(cnt).respTime = rtime;
        eventInfo(cnt).keyPressed = resp;

        % Outcome (duration = 1 sec)
        imgName = sprintf('A%02d-%s.png',imageOrder(j),outcomeOrder{j});
        fn_stim = fullfile(PEpath,'images',imgName);
        eventInfo(cnt).outcomeOnset = toc;
        eventInfo(cnt).outcomeType = outcomeOrder{j};
        timePtr = show_image_on_screen(fn_stim, windowPtr, timePtr, outcomeDuration, gray);


        % Inter-trial interval (duration ~ 2 sec)
        eventInfo(cnt).ITI = toc;
        Screen(windowPtr,'TextSize',60);
        timePtr = show_message_on_screen('+',windowRect, windowPtr, timePtr, ITI(j), black, gray);

        % Count the number of trials
        cnt = cnt+1;
    end
end

%--------------------------------------------------------------------------
%  Write eventInfo and Responses
%--------------------------------------------------------------------------
fn_experiment = fullfile(RESPpath,sprintf('%s-run2-%s.csv', subj.name, subj.time));
write_to_csvfile(fn_experiment,eventInfo);



%--------------------------------------------------------------------------
% End-of-experiment
%--------------------------------------------------------------------------
str_messages = '-End of Run #2-';
show_message_on_screen(str_messages, windowRect, windowPtr, [], 2, black, gray);
str_messages = 'Thank you for your participantion.';
Screen(windowPtr,'TextSize',30);
show_message_on_screen(str_messages, windowRect, windowPtr, [], 2, black, gray);


Priority(0);
ListenChar(0);

% Clear the screen
sca;


elapsedtime = toc;
ET = clock;
fprintf('==========================================================================\n');
fprintf('    Started Time : %g-%g-%g  %g:%g:%d \n', round(ST));
fprintf('        End Time : %g-%g-%g  %g:%g:%d \n', round(ET));
fprintf('    Elapsed Time : %g min. (%g sec.)\n',elapsedtime/60, elapsedtime);
fprintf('==========================================================================\n\n');
```

Homework!

- Perform prediction_error experiment at your home
- and send your results to skyeong@yuhs.ac
- your results are located at lecture2/psychtoolbox/prediction_error/RESP/*.csv