

# Flexible experiments in the browser: A tutorial in jsPsych and Google app engine

Dani Navarro



# Motivation

- ***What is this?***

- Setting up a stand-alone experiment with jsPsych
- Deploying it to the web with Google app engine

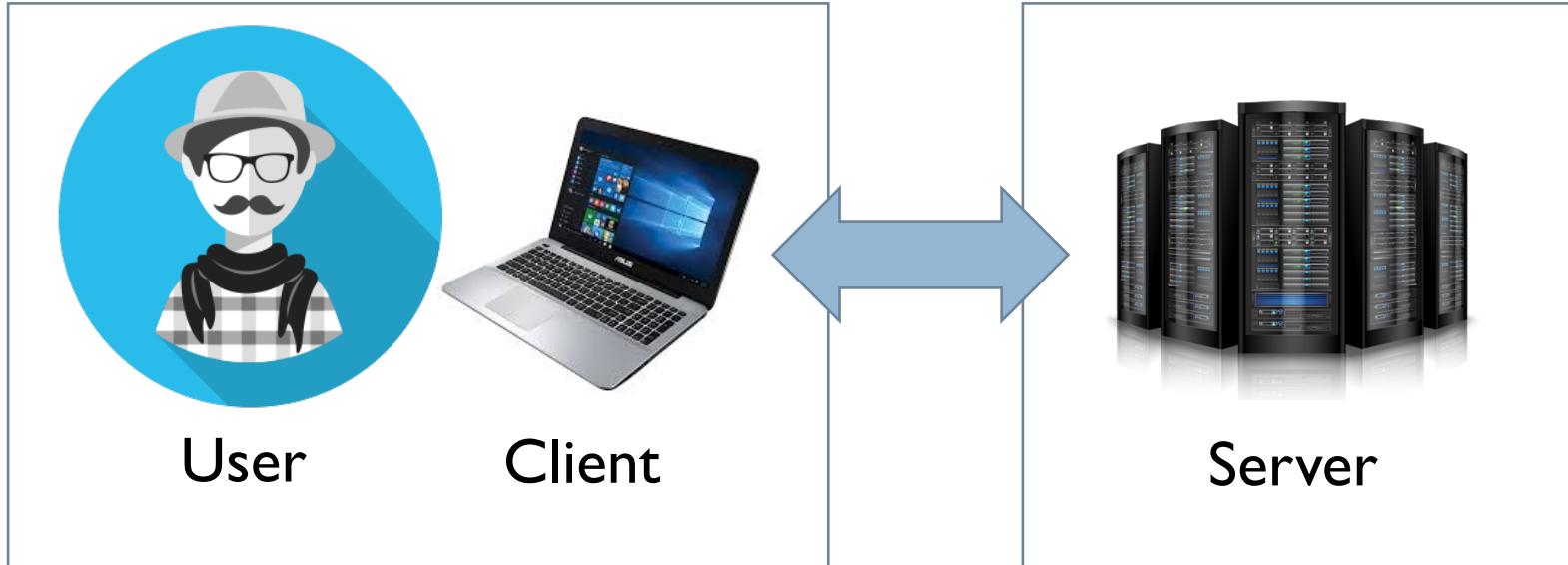
- ***What is this not?***

- Participant recruitment (via MTurk, Sona, etc)
- Detailed comparison to Qualtrics, Psychtoolbox, etc

- ***Why bother?***

- Freedom: your experiment runs anywhere, for free
- Flexibility: once you're comfortable with JS, you can make the experiment as flexible as you like.
- ... in this tutorial we cover some of this flexibility, but you can do a lot more than this!

# The simplest model

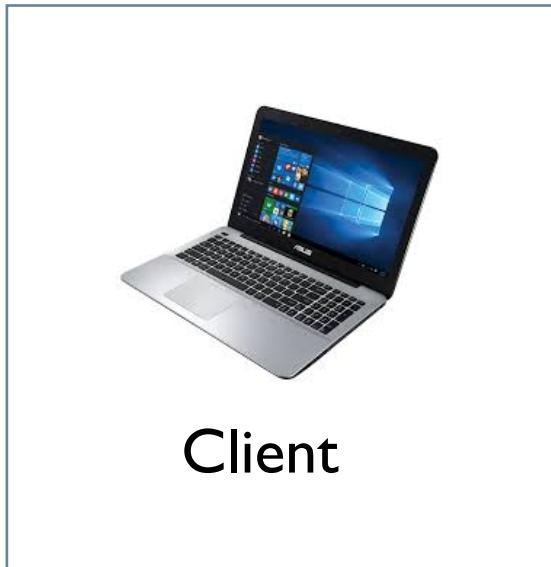


- The code for the experiment is executed on the client machine inside the browser
- When finished, the client sends the data to the server
- The code for the experiment is hosted on the server
- The data from the experiment is stored on the server



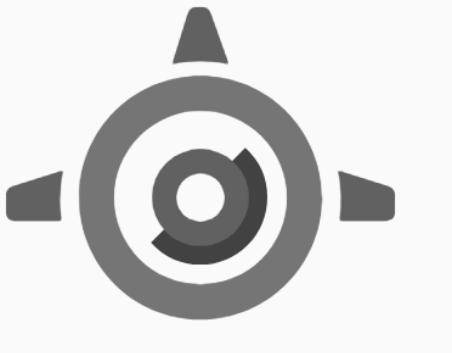
jspsych

# The jsPsych library



<http://www.jspsych.org/>

- **What is this?**
  - What the server sends to the client is a webpage, consisting of HTML, CSS and JavaScript (JS)
  - You could write your experiment in raw HTML/CSS/JS but that's tedious
- **Why use jsPsych?**
  - jsPsych takes care of the uglier side of JS, stores data in a nice format, and is build specifically for behavioural experiments
  - Also it's free



# Google app engine

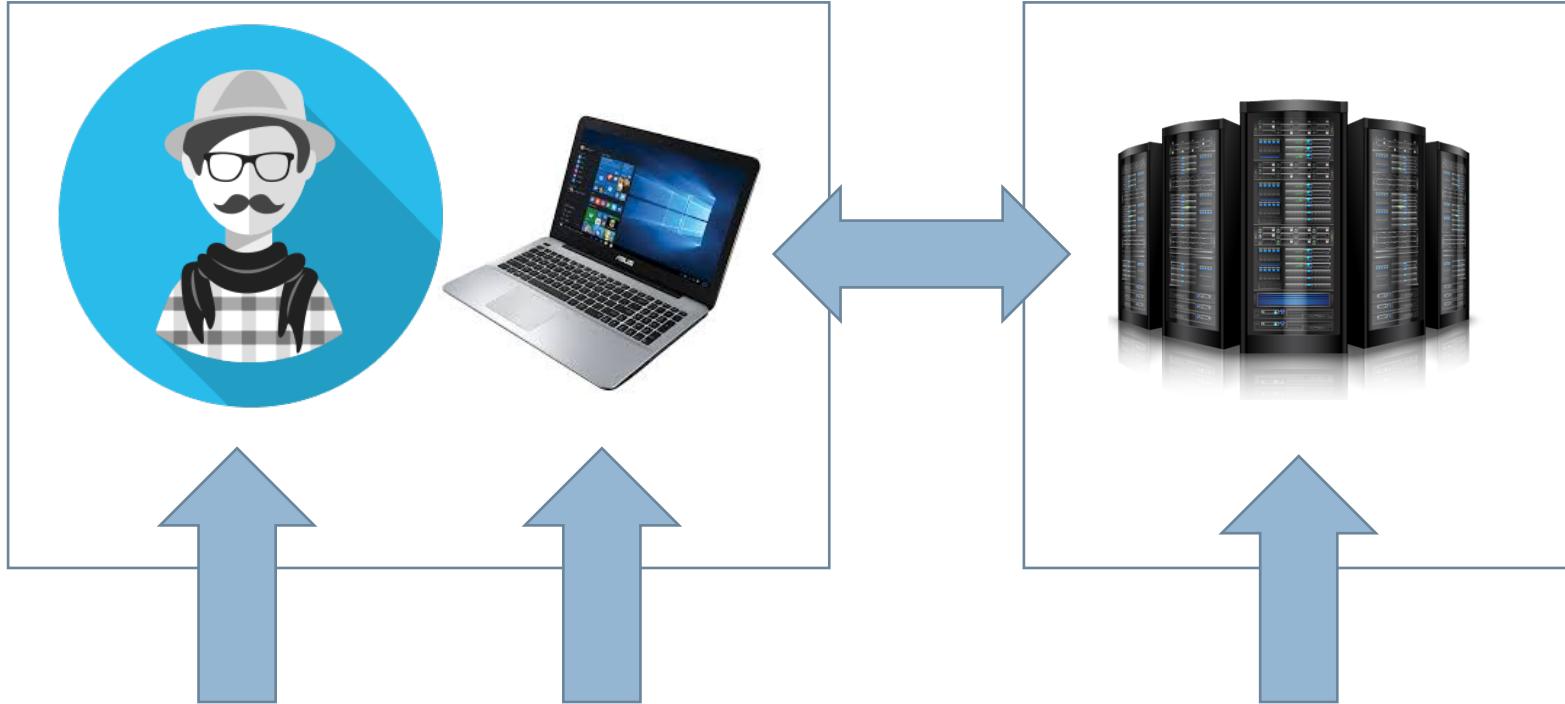
- **What is this?**
  - Google will let you host your site on their servers almost entirely for free
  - At the end of the tutorial I'll talk about how to use their service
- **Why use it?**
  - Flexibility: once you can do a basic GAE set up, you can use the server to manage many participants interacting simultaneously



Server

<https://cloud.google.com/appengine/>

# Division of responsibility!



amazon mechanical turk  
beta

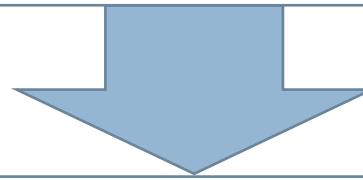


jspsych

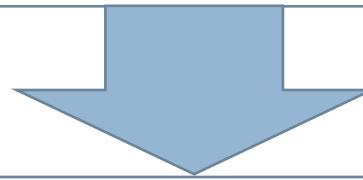




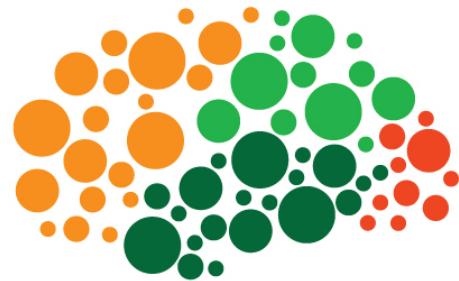
Build a simple experiment by  
following the jsPsych tutorial



Use the lab “blankex” template  
to add UNSW ethics, GAE  
hooks, MTurk code, etc



Create a GAE project for the  
experiment and push it online



# jspsych

<http://docs.jspsych.org/tutorials/hello-world/>

<http://docs.jspsych.org/tutorials/rt-task/>

(the tutorials are pretty comprehensive so I don't have anything to add)

# Brackets text editor

<http://brackets.io/>

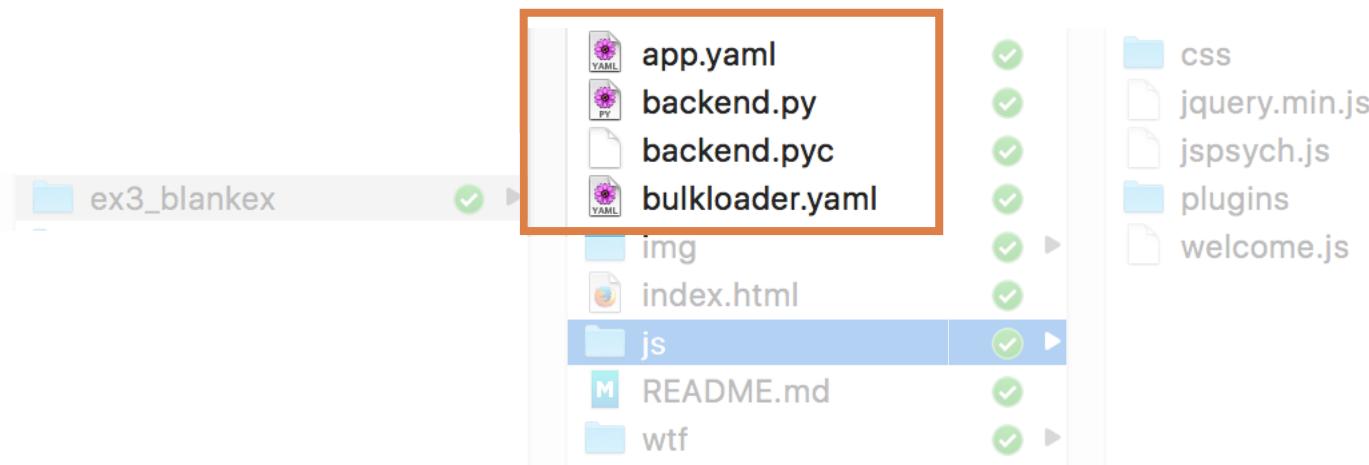
```
1  <!doctype html>
2 ▼ <html>
3 ▼   <head>
4     <title>My experiment</title>
5     <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>
6     <script src="jspsych-5.0.3/jspsych.js"></script>
7     <script src="jspsych-5.0.3/plugins/jspsych-text.js"></script>
8     <link href="jspsych-5.0.3/css/jspsych.css" rel="stylesheet" type="text/css"></link>
9   </head>
10  <body>
11  </body>
12  <script>
13
14 ▼  var hello_trial = {
15    type: 'text',
16    text: 'Hello world!'
17  }
18
19 ▼  jsPsych.init({
20   timeline: [ hello_trial ]
21 })
22
23  </script>
24 </html>|
```



<https://github.com/djnavarro/blankex>



These files are part of the Google app engine configuration, we'll ignore them for now





These files are our jsPsych  
(and jquery) libraries, we  
don't need to touch them

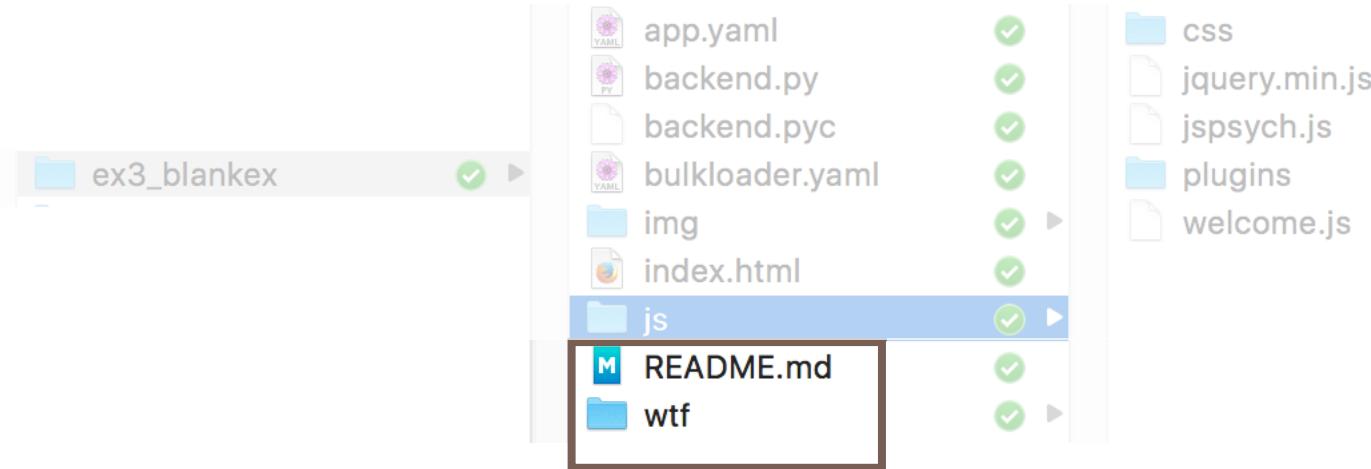
The screenshot shows a file explorer window with the following structure:

- app.yaml
- backend.py
- backend.pyc
- bulkloader.yaml
- img
- index.html
- js** (highlighted with a blue selection bar)
- README.md
- wtf

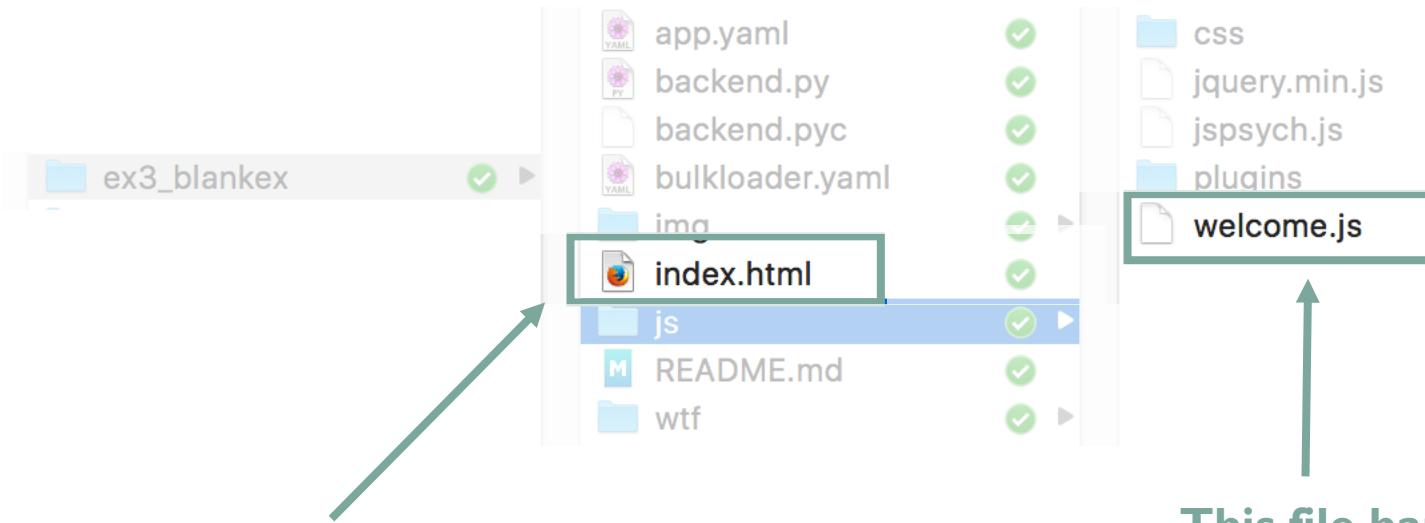
A callout box highlights the contents of the 'js' folder:

- css
- jquery.min.js
- jspsych.js
- plugins

Below the 'js' folder, there is a file named 'welcome.js'.



These files aren't very  
important, and we can  
ignore them for now



**This file handles the main experiment... so we want to insert the jsPsych code for our RT task into this**

**This file handles the instructions sheet, demographics, consent form, etc... so we'll need to edit a few bits to make it appropriate**

```
1  <!doctype html>
2 ▼ <html>
3
```

Make sure we load all the jsPsych plugins we need... in this case I added the “text” and “single-stim” plugins because the RT task needs them but the default blankex template doesn’t include them



```
4 ▼ <head>
5     <title>UNSW CCS</title>
6     <script src="./js/jquery.min.js"></script>
7     <script src="./js/jspsych.js"></script>
8     <script src="./js/plugins/jspsych-survey-multi-choice.js"></script>
9     <script src="./js/plugins/jspsych-button-response.js"></script>
10    <script src="./js/plugins/jspsych-text.js"></script>
11    <script src="./js/plugins/jspsych-single-stim.js"></script>
12    <script src="./js/welcome.js"></script>
13    <link href="./js/css/jspsych.css" rel="stylesheet" type="text/css"></link>
14 </head>
```

```
1  <!doctype html>
2 ▼ <html>
3
```

Note this line: it loads the welcome.js script, which is needed to run all the UNSW information sheet stuff, consent, demographic screen, etc

```
4 ▼ <head>
5     <title>UNSW CCS</title>
6     <script src=".js/jquery.min.js"></script>
7     <script src=".js/jspysch.js"></script>
8     <script src=".js/plugins/jspysch-survey-multi-choice.js"></script>
9     <script src=".js/plugins/jspysch-button-response.js"></script>
10    <script src=".js/plugins/jspysch-text.js"></script>
11    <script src=".js/plugins/jspysch-single-stim.js"></script>
12    <script src=".js/welcome.js"></script>
13    <link href=".js/css/jspysch.css" rel="stylesheet" type="text/css"></link>
14 </head>
```



```
15
16    <body>
17        <div id="welcome"></div>
18    </body>
19
20    <script>
21
22        /* initialise timeline*/
23        var timeline=[];
24        var introloop=[];
25        var turkcode = (Math.floor(Math.random() * 899999) + 100000).toString();
26    <var images = [
27        './img/blue.png',
28        './img/orange.png'
29    ];
30
```

List the image files so that jsPsych preloads them properly

```
31 /* function to start the jsPsych experiment */
32 ▼
33
34     // record the turkcode in the jsPsych data
35 ▼
36     jsPsych.data.addProperties({
37         turkcode: turkcode
38     });

```



Make sure jsPsych records the turkcode. Nothing to do here for this task, but often you'll want to add condition variables here

```
35 ▼          jsPsych.data.addProperties({
36              turkcode: turkcode
37          });

38
39 ▼      jsPsych.init({
40          timeline: timeline,
41          preload_images: images,
42          on_finish: function() {
43              endExperiment( jsPsych.data.dataAsCSV(), function() {
44                  document.write('<div id="endscreen" class="endscreen" style="width:1000px"><div class="endscreen" style="text-align:center; border:0px solid; padding:10px; font-size:120%; width:800px; float:right"><p><br><br><br>All done!<br><br>Your completion code is <span id="turkcode" style="font-weight:bold; font-size:130%">' + turkcode + '</span>. To receive payment for the HIT, return to the Amazon Mechanical Turk page and enter this code. Please contact us if something goes wrong and we\'ll fix it as quickly as possible.</p> </div></div>') })
45          }
46      });
47  }
```



At the end of the experiment, jsPsych will convert the data to a CSV format, and display a message on the screen displaying the MTurk completion code. Nothing to edit here right now

```
35
36         jsPsych.data.addProperties({
37             turkcode: turkcode
38         });
39
39 ▼     jsPsych.init({
40         timeline: timeline,
41         preload_images: images,
42         on_finish: function() {
43             endExperiment( jsPsych.data.dataAsCSV(), function() {
44                 document.write('<div id="endscreen" class="endscreen" style="width:1000px"><div class="endscreen" style="text-align:center; border:0px solid; padding:10px; font-size:120%; width:800px; float:right"><p><br><br><br>All done!<br><br>Your completion code is <span id="turkcode" style="font-weight:bold; font-size:130%">' + turkcode + '</span>. To receive payment for the HIT, return to the Amazon Mechanical Turk page and enter this code. Please contact us if something goes wrong and we\'ll fix it as quickly as possible.</p> </div></div>') })
45             }
46         });
47     }
```

This line handles the communication with Google app engine. For the moment it's commented out

```
48
49     /* save and finish */
50     function endExperiment(dataset,callback) {
51         // $.post('submit',[{"content": dataset}]); // uncomment to post data
52         setTimeout(callback,1000)
53     }
```

```
35
36         jsPsych.data.addProperties({
37             turkcode: turkcode
38         });
39
40         jsPsych.init({
41             timeline: timeline,
42             preload_images: images,
43             on_finish: function() {
44                 endExperiment( jsPsych.data.dataAsCSV(), function() {
45                     document.write('<div id="endscreen" class="endscreen" style="width:1000px"><div class="endscreen" style="text-align:center; border:0px solid; padding:10px; font-size:120%; width:800px; float:right"><p><br><br><br>All done!<br><br>Your completion code is <span id="turkcode" style="font-weight:bold; font-size:130%">' + turkcode + '</span>. To receive payment for the HIT, return to the Amazon Mechanical Turk page and enter this code. Please contact us if something goes wrong and we\'ll fix it as quickly as possible.</p> </div></div>') })
46             }
47         });
48     }
49 }
```

Notice that we've got this "wrapped" inside the `endExperiment` function to prevent the browser moving onto the completion code before the "post" request gets sent to Google. This is important

```
48     /* save and finish */
49     function endExperiment(dataset,callback) {
50         // $.post('submit',[{"content": dataset}]); // uncomment to post data
51         setTimeout(callback,1000)
52     }
```

```
54      /* change the display property of a set of objects */
55  ▼ function setDisplay(theClass, theValue) {
56      var i, classElements = document.getElementsByClassName(theClass);
57  ▼      for (i = 0; i < classElements.length; i = i + 1) {
58          classElements[i].style.display = theValue;
59      }
60 }
```



This is just a function I use a lot, but there's nothing for us to do here

```
62 ▼ /* A grossly typical way to run the instructions is to go through a series  
63 of small slides (implemented as trials in jsPsych) with a minimum reading time  
64 enforced for each slide, such that the "continue" button doesn't appear until  
65 the time elapses. At the end of the instructions, there is a short quiz, and if  
66 the participant gets them wrong they are sent back to the beginning */  
67  
68 /* define welcome message block */  
69 ▼ var welcome_block = {  
70   type: "text",  
71   text: "Welcome to the experiment. Press any key to begin."  
72 };  
73  
74 /* define instructions block */  
75 ▼ var instructions_block = {  
76   type: "text",  
77   text: "<p>In this experiment, a circle will appear in the center " +  
78     "of the screen.</p><p>If the circle is <strong>blue</strong>, " +  
79     "press the letter F on the keyboard as fast as you can.</p>" +  
80     "<p>If the circle is <strong>orange</strong>, do not press " +  
81     "any key.</p>" +  
82     "<div class='left center-content'><img src='img/blue.png'></img>" +  
83     "<p class='small'><strong>Press the F key</strong></p></div>" +  
84     "<div class='right center-content'><img src='img/orange.png'></img>" +  
85     "<p class='small'><strong>Do not press a key</strong></p></div>" +  
86     "<p>Press any key to begin.</p>",  
87   timing_post_trial: 2000  
88 };  
89
```

Now we have some editing to do: these two introductory trials are taken straight from our RT experiment

```
74  /* define instructions block */
75  var instructions_block = {
76    type: "text",
77    text: "<p>In this experiment, a circle will appear in the center " +
78          "of the screen.</p><p>If the circle is <strong>blue</strong>, " +
79          "press the letter F on the keyboard as fast as you can.</p>" +
80          "<p>If the circle is <strong>orange</strong>, do not press " +
81          "any key.</p>" +
82          "<div class='left center-content'><img src='img/blue.png'></img>" +
83          "<p class='small'><strong>Press the F key</strong></p></div>" +
84          "<div class='right center-content'><img src='img/orange.png'></img>" +
85          "<p class='small'><strong>Do not press a key</strong></p></div>" +
86          "<p>Press any key to begin.</p>",
87    timing_post_trial: 2000
88  };
89
```

Because these two trials are part of the “instruction loop” that you can’t escape until you get the instruction check trials correct, we push them to the “introloop”



```
90  /* instructions are pushed inside the loop node */
91  introloop.push(welcome_block);
92  introloop.push(instructions_block);
93
```

```
95 /* text defining the instruction check questions */
96 var Q0_text = "<b>Question 1:</b> What should you do when you see a blue circle?";
97 var Q0_answers = ["Press space bar", "Do nothing", "Press F"];
98 var Q1_text = "<b>Question 2:</b> What should you do when you see an orange circle?";
99 var Q1_answers = ["Press space bar", "Do nothing", "Press J"];
100
101 var correctstring = '{"Q0":"' + Q0_answers[2] + '", "Q1":"' + Q1_answers[1] + '"}';
102
```

Edit the text of the instruction check  
questions and the answers

```
103 /* define instruction check block */
104 var instructioncorrect = false;
105 var instruction_check = {
106   type: "survey-multi-choice",
107   preamble: ["<p align='center'><b>Check your knowledge before you begin!</b></p>"],
108   questions: [Q0_text, Q1_text],
109   options: [Q0_answers, Q1_answers],
110   on_finish: function(data) {
111     if( data.responses == correctstring) {
112       action = false;
113       instructioncorrect = true;
114     }
115   }
116 }
117 introloop.push(instruction_check)
118
```

```
95    /* text defining the instruction check questions */
96    var Q0_text = "<b>Question 1:</b> What should you do when you see a blue circle?";
97    var Q0_answers = ["Press space bar", "Do nothing", "Press F"];
98    var Q1_text = "<b>Question 2:</b> What should you do when you see an orange circle?";
99    var Q1_answers = ["Press space bar", "Do nothing", "Press J"];
100
101   var correctstring = '{"Q0":"' + Q0_answers[2] + '", "Q1":"' + Q1_answers[1] + '"}';
102
```



Make sure “correctstring” uses the correct options

(remember: JS indexes from 0, so Q0\_answers[2] refers to the third response option, not the second)

```
103    /* define instruction check block */
104    var instructioncorrect = false;
105    var instruction_check = {
106        type: "survey-multi-choice",
107        preamble: ["<p align='center'><b>Check your knowledge before you begin!</b></p>"],
108        questions: [Q0_text, Q1_text],
109        options: [Q0_answers, Q1_answers],
110        on_finish: function(data) {
111            if( data.responses == correctstring) {
112                action = false;
113                instructioncorrect = true;
114            }
115        }
116    }
117    introloop.push(instruction_check)
118
```

```
95 /* text defining the instruction check questions */
96 var Q0_text = "<b>Question 1:</b> What should you do when you see a blue circle?";
97 var Q0_answers = ["Press space bar", "Do nothing", "Press F"];
98 var Q1_text = "<b>Question 2:</b> What should you do when you see an orange circle?";
99 var Q1_answers = ["Press space bar", "Do nothing", "Press J"];
100
101 var correctstring = '{"Q0":"' + Q0_answers[2] + '", "Q1":"' + Q1_answers[1] + '"}';
102
```

The instruction check questions are implemented as a survey trial with multiple choice questions



```
103 /* define instruction check block */
104 var instructioncorrect = false;
105 var instruction_check = {
106   type: "survey-multi-choice",
107   preamble: ["<p align='center'><b>Check your knowledge before you begin!</b></p>"],
108   questions: [Q0_text, Q1_text],
109   options: [Q0_answers, Q1_answers],
110   on_finish: function(data) {
111     if( data.responses == correctstring) {
112       action = false;
113       instructioncorrect = true;
114     }
115   }
116 }
117 introloop.push(instruction_check)
118
```

```
95 /* text defining the instruction check questions */
96 var Q0_text = "<b>Question 1:</b> What should you do when you see a blue circle?";
97 var Q0_answers = ["Press space bar", "Do nothing", "Press F"];
98 var Q1_text = "<b>Question 2:</b> What should you do when you see an orange circle?";
99 var Q1_answers = ["Press space bar", "Do nothing", "Press J"];
100
101 var correctstring = '{"Q0":"' + Q0_answers[2] + '", "Q1":"' + Q1_answers[1] + '"}';
102
```

Keep track of whether the participant has made  
the correct responses (no editing required by us!)

```
103 /* define instruction check block */
104 var instructioncorrect = false;
105 var instruction_check = {
106   type: "survey-multi-choice",
107   preamble: ["<p align='center'><b>Check your knowledge before you begin!</b></p>"],
108   questions: [Q0_text, Q1_text],
109   options: [Q0_answers, Q1_answers],
110   on_finish: function(data) {
111     if( data.responses == correctstring) {
112       action = false;
113       instructioncorrect = true;
114     }
115   }
116 }
117 introloop.push(instruction_check)
118
```

```
119      /* define a page for the incorrect response */
120     var showsplash = true;
121     var splash_screen = {
122        type: 'button-response',
123        timing_post_trial: 0,
124        button_html: '<button class="jspsych-btn" style="display:none">%choice%</button>',
125        choices: ['Click here to read the instructions again'],
126        on_trial_start: function() { setTimeout(function() {setDisplay("jspsych-btn", "")}, 500)},
127        is_html: true,
128        stimulus: 'Unfortunately, at least one of your answers was incorrect.'
129     }
130
131    /* ...but push it to a conditional node that only shows it if the response was wrong */
132    var conditional_splash = {
133      timeline: [splash_screen],
134      conditional_function: function(data) {
135         return !instructioncorrect // skip if correct
136      }
137    }
138    introloop.push(conditional_splash)
139
```



If the participant gets it wrong, we're going to need a “splash” screen that informs them they've made a mistake and will be sent back to the beginning.

```
119     /* define a page for the incorrect response */
120     var showsplash = true;
121     var splash_screen = {
122         type: 'button-response',
123         timing_post_trial: 0,
124         button_html: '<button class="jspsych-btn" style="display:none">%choice%</button>',
125         choices: ['Click here to read the instructions again'],
126         on_trial_start: function() { setTimeout(function() {setDisplay("jspsych-btn","",500)},
127             is_html: true,
128             stimulus: 'Unfortunately, at least one of your answers was incorrect.'
129         }
130
131     /* ...but push it to a conditional node that only shows it if the response was wrong */
132     var conditional_splash = {
133         timeline: [splash_screen],
134         conditional_function: function(data) {
135             return !instructioncorrect // skip if correct
136         }
137     }
138     introloop.push(conditional_splash)
```



However, we only want them to see this if they got it wrong, so this trial is placed inside a “conditional” node, and we include this conditional node in the introloop. That way, this “failure” screen only appears if the participant got it wrong

```
140 /* finally, add the entirety of this introductory section to a loop node ... */
141 ▼ var loop_node = {
142     timeline: introloop,
143     ▼ loop_function: function(data) {
144         //var action = true;
145         return !instructioncorrect // stop looping if correct
146     }
147 }
148 timeline.push(loop_node) // ... and add that to the timeline
```



The whole of this introloop is folded into a single “loop node”, which we then push to the global timeline.

```
140 /* finally, add the entirety of this introductory section to a loop node ... */
141 ▼ var loop_node = {
142     timeline: introloop,
143     loop_function: function(data) {
144         //var action = true;
145         return !instructioncorrect // stop looping if correct
146     }
147 }
148 timeline.push(loop_node) // ... and add that to the timeline
```



The whole of this introloop is folded into a single “loop node”, which we then push to the global timeline.

Then define a splash screen saying “congrats for getting the questions right” and push it to the global timeline so that it displays as soon as the user escapes the introloop



```
150 /* success trial */
151 ▼ var successtrial = {
152     type: 'button-response',
153     timing_post_trial: 0,
154     button_html: '<button class="jpsych-btn" style="display:none">%choice%</button>',
155     choices: ['Click here to begin the experiment'],
156     on_trial_start: function() { setTimeout(function() {setDisplay("jpsych-btn", "")}, 500)},
157     is_html: true,
158     stimulus: 'Well done!'
159 };
160 timeline.push(successtrial);
```

```
162     /* now comes all the code from our RT experiment... */  
163  
164     /* define test block */  
165     var test_stimuli = [  
166         {  
167             stimulus: "img/blue.png",  
168             data: { response: 'go' }  
169         },  
170         {  
171             stimulus: "img/orange.png",  
172             data: { response: 'no-go' }  
173         }  
174     ];  
175     ...
```

Some more editing: cut and paste the code from the RT experiment here (i.e., the `test_block` and the `debrief_block`)

```
162     /* now comes all the code from our RT experiment... */  
163  
164     /* define test block */  
165     var test_stimuli = [  
166         {  
167             stimulus: "img/blue.png",  
168             data: { response: 'go' }  
169         },  
170         {  
171             stimulus: "img/orange.png",  
172             data: { response: 'no-go' }  
173         }  
174     ];
```

## Push the relevant trials to the timeline

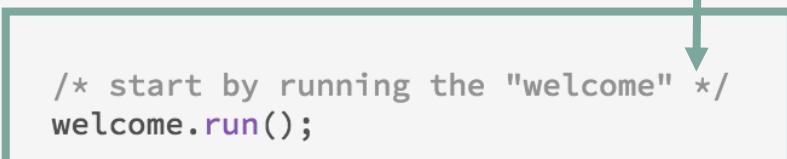


```
235     timeline.push(successtrial);  
236     timeline.push(test_block);  
237     timeline.push(debrief_block);  
238  
239     /* start by running the "welcome" */  
240     welcome.run();  
241  
242  
243 </script>  
244 </html>
```

```
162 /* now comes all the code from our RT experiment... */  
163  
164     /* define test block */  
165     var test_stimuli = [  
166         {  
167             stimulus: "img/blue.png",  
168             data: { response: 'go' }  
169         },  
170         {  
171             stimulus: "img/orange.png",  
172             data: { response: 'no-go' }  
173         }  
174     ];
```

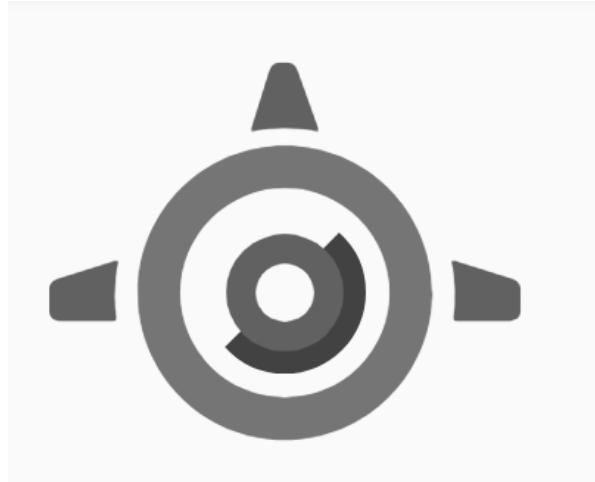
Start the experiment by calling welcome, which will ensure that the UNSW stuff runs first, and it will call the startExperiment() function when it's finished

```
235     timeline.push(successtrial);  
236     timeline.push(test_block);  
237     timeline.push(debrief_block);  
238  
239     /* start by running the "welcome" */  
240     welcome.run();  
241  
242  
243 </script>  
244 </html>
```



```
1 var welcome = {};
2
3 // ----- things that vary from task to task -----
4
5 welcome.task = {};
6 welcome.task.blurb = 'A short RT study is a short psychological study investigating
7 how people make decisions.';
8 welcome.task.time = '5 minutes';
9 welcome.task.pay = 'US$0.85';
10
11 // ----- things that vary between ethics approvals -----
12
13 welcome.ethics = {};
14 welcome.ethics.approval = 'XXXX';
15 welcome.ethics.name = 'The name of my study...';
16 welcome.ethics.selection = 'You are invited to participate in a study of how human reasoning
17 works. We hope to learn what information people find most useful in guiding their judgments
18 about the world. You were selected as a possible participant in this study because you
19 accepted our HIT on Amazon Mechanical Turk.';
20 welcome.ethics.description = 'If you decide to participate, we will present you with some
21 reasoning problems in which you need to use the (possibly incomplete) information to make
22 judgements (or guesses) about the truth of different propositions. Detailed instructions will
23 be provided once the task begins. The exact number of problems you need to solve depends on
24 which version of the task the computer assigns you to, but the on-screen display will inform
25 you of how much further you have to go. The task should take approximately ' +
26 welcome.task.time + ' to complete including reading time.';
27
```

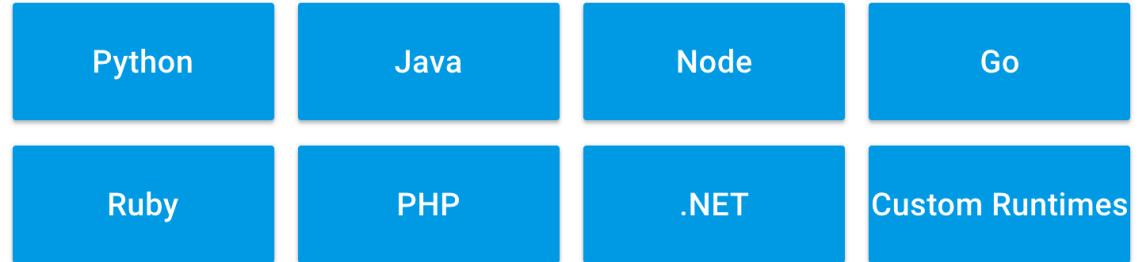
```
19 // ----- function to start the task -----
20 ▼ welcome.run = function() {
21     document.getElementById("welcome").innerHTML =
22         welcome.section.header +
23         welcome.section.start +
24         welcome.section.consent +
25         welcome.section.demographics;
26 }
27
28 // ----- actions to take at the end of each click -----
29 welcome.click = {};
30 ▼ welcome.click.start = function() {
31     welcome.helpers.setDisplay('start', 'none');
32     welcome.helpers.setDisplay('consent', '');
33     welcome.helpers.setHeader(' ');
34 }
35 ▼ welcome.click.consent = function() {
36     welcome.helpers.setDisplay('consent', 'none');
37     welcome.helpers.setDisplay('demographics', '');
38     welcome.helpers.setHeader(' ');
39 }
40 ▼ welcome.click.demographics = function() {
41     welcome.helpers.setDisplay('demographics', 'none');
42     welcome.helpers.setDisplay('header', 'none');
43 ▼ jsPsych.data.addProperties({ // record the condition assignment in the jsPsych data
44     gender: welcome.helpers.getRadioButton("gender"),
45     age: document.getElementById("age").value,
46     language: document.getElementById("language").value,
47     country: document.getElementById("country").value
48 });
49 startExperiment(); // start the jsPsych experiment
50 }
51
52
53 // ----- html for the various sections -----
54 welcome.section = {};
55 welcome.section.header =
56     '<!-- ##### Heading ##### -->' +
```



<https://cloud.google.com/appengine/>

---

## Flexible environment



---

## Standard environment



**Step 1: Select the SDK (software developer kit) for a standard Python environment**

<https://cloud.google.com/appengine/downloads>

Or, you can download the original App Engine SDK for Python.

To install the original local development server and the `appcfg` tooling, you can install the original App Engine SDK for Python.

## Optional: Download and install the original App Engine SDK for Python

LINUX

MAC OS X

WINDOWS

1. Download the App Engine SDK for Python:

 DOWNLOAD

Version:

1.9.60 - 2017-09-05

Size:

72.0 MB

SHA1 Checksum:

`7d4f3378eb793f334a24c281c514c77807d498c4`

Step 2: Ignore Google's preferred gcloud thing  
and get the original App Engine SDK

Yeah, but I like the point and click thing

2. To install the SDK on Mac OS X:

- a. In the Finder, click **Go > Applications** to open the Applications folder.
- b. Double click the `GoogleAppEngineLauncher-1.9.60.dmg` file that you downloaded to open it, then drag the **GoogleAppEngineLauncher** icon over to the Applications folder.
- c. Double-click **GoogleAppEngineLauncher** in the Application folder.
- d. When prompted to *Make command symlinks*, click **OK**. The symlinks allow you to run important SDK command-line tools in any terminal window.



**Important:** The GoogleAppEngineLauncher is a convenient UI-based tool for running and deploying App Engine apps, but it *does not* provide all the features you'll need. You can use the equivalent [gcloud](#) command-line tool, for many of the tasks that you'll want to perform.

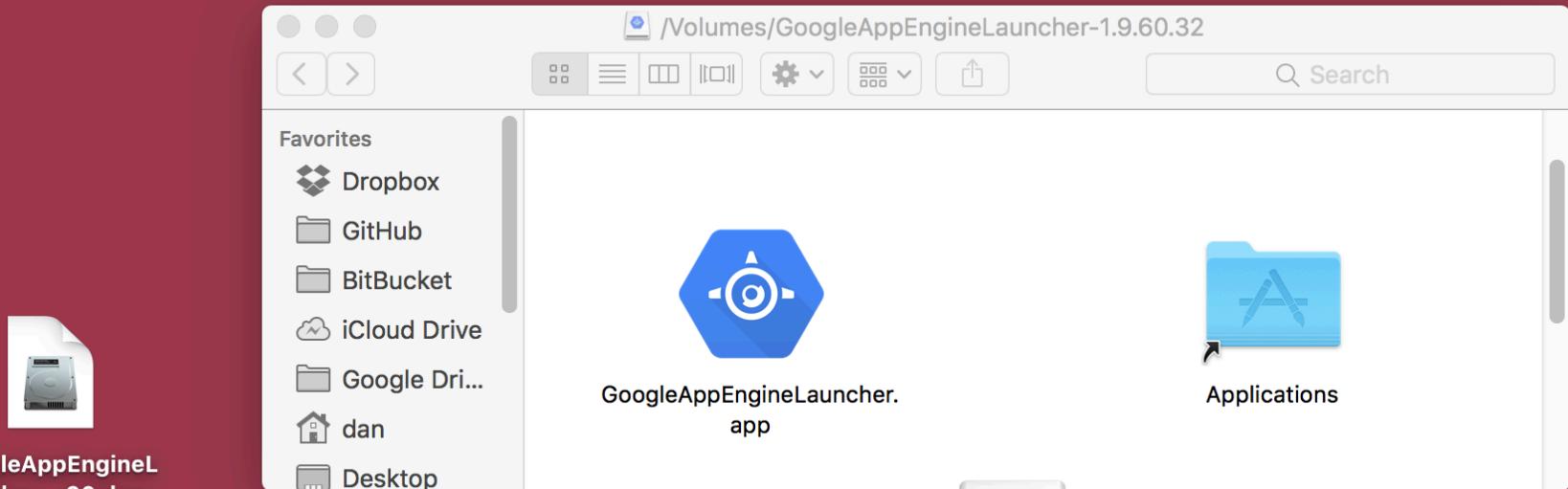
- e. Notice that the installation process above unpacks the contents of the App Engine SDK at the location:

```
/usr/local/google_appengine
```

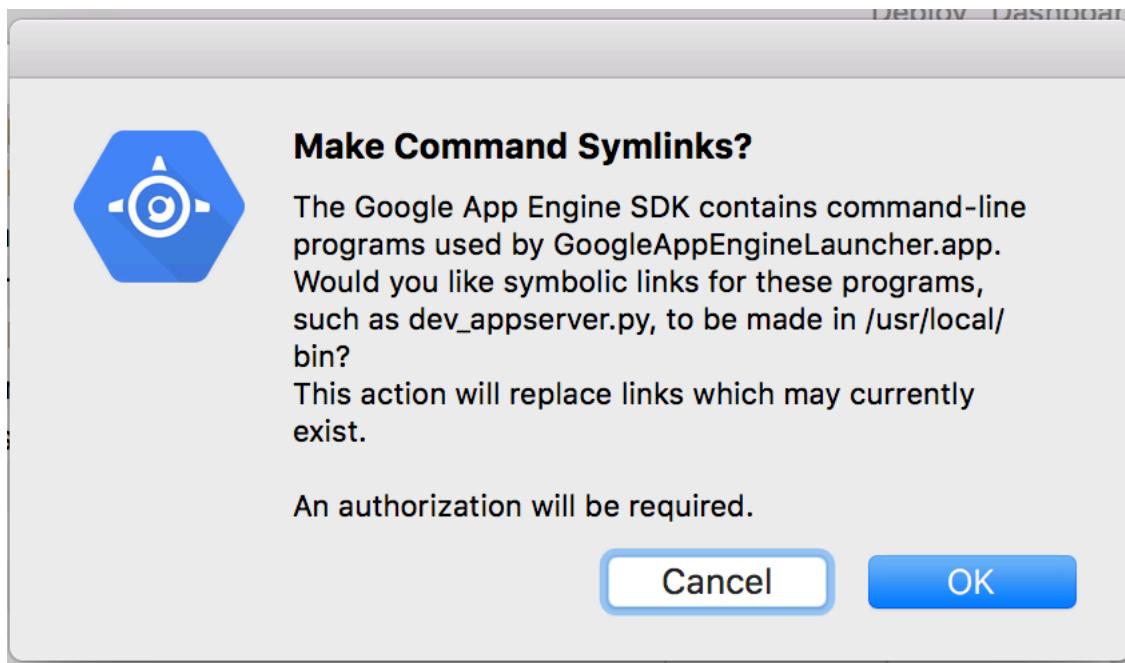
- f. The App Engine SDK requires Python 2.7, which is installed by default on Mac OS X 10.6 (Snow Leopard) or later. Verify your Mac's Python installation using the following command:

```
/usr/bin/env python -V
```

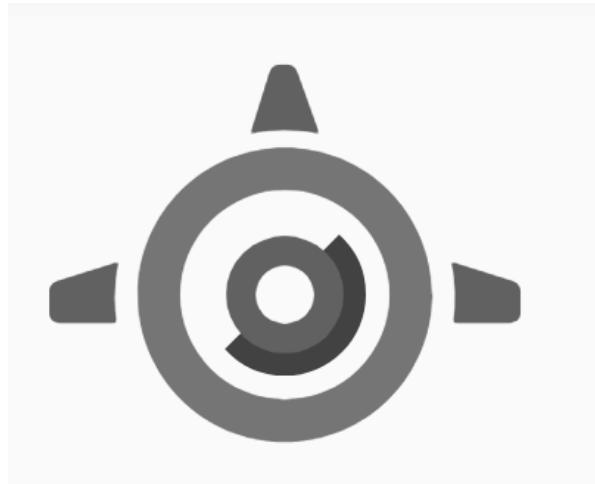
If the output looks like `Python 2.7.[NUMBER]` then you already have the correct Python version installed. Otherwise you can download and install Python 2.7 from the [Python web site](#).



### Step 3: Install the GAE Launcher



Step 4: Allow it to make the symbolic links!

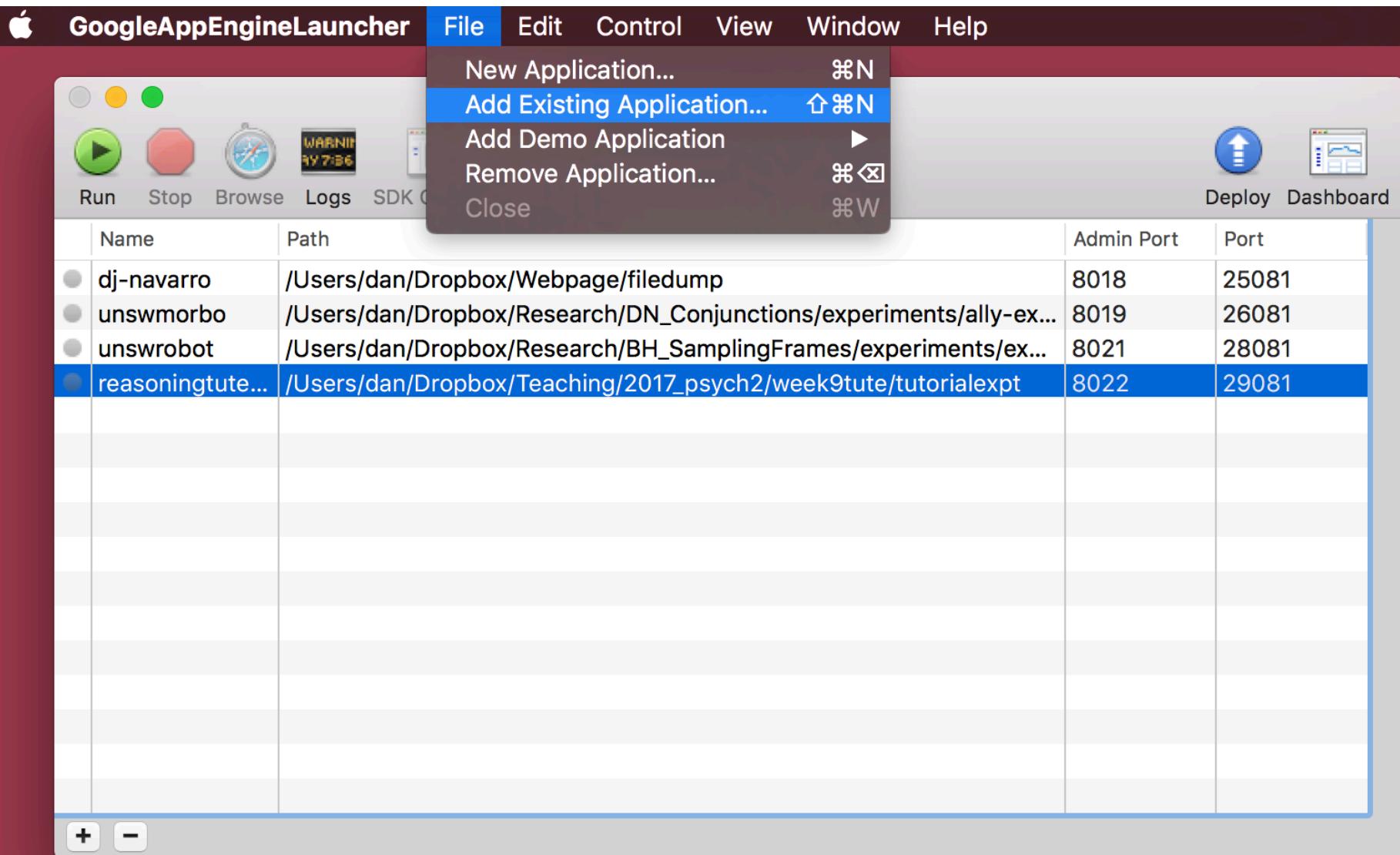


Creating the GAE project...

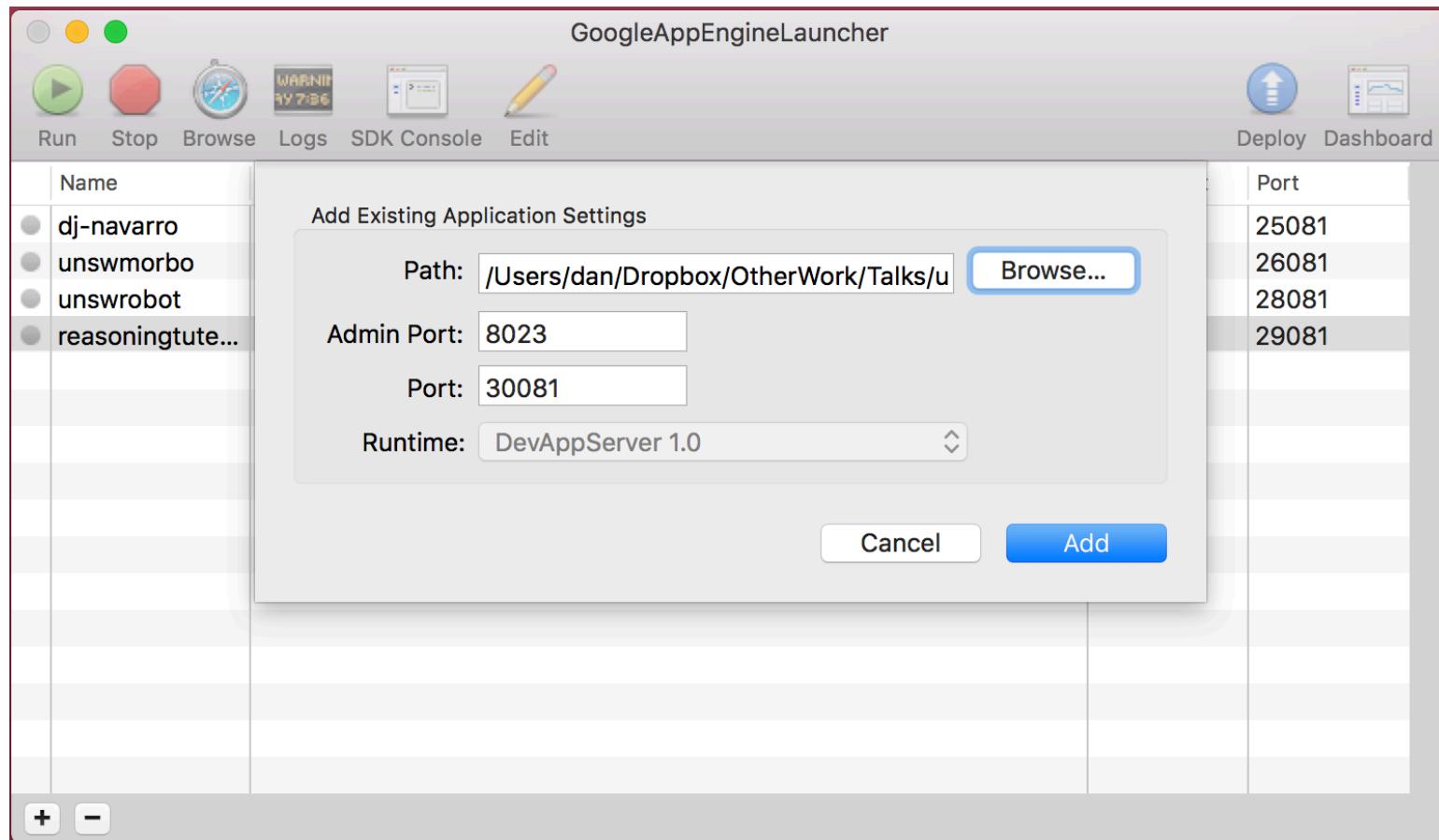
Step 1: Open the app.yaml file and give your project a unique name

```
1 application: unsw-dani-gaert
2 version: 1
3 runtime: python27
4 api_version: 1
5 threadsafe: true
6
7 handlers:
8 - url: /img
9   static_dir: img
10 - url: /js
11   static_dir: js
12
13 - url: /.*
14   script: backend.application
15
16 libraries:
17 - name: webapp2
18   version: latest
19 - name: jinja2
20   version: latest
21
22
23 builtins:
24 - remote_api: on
25
```

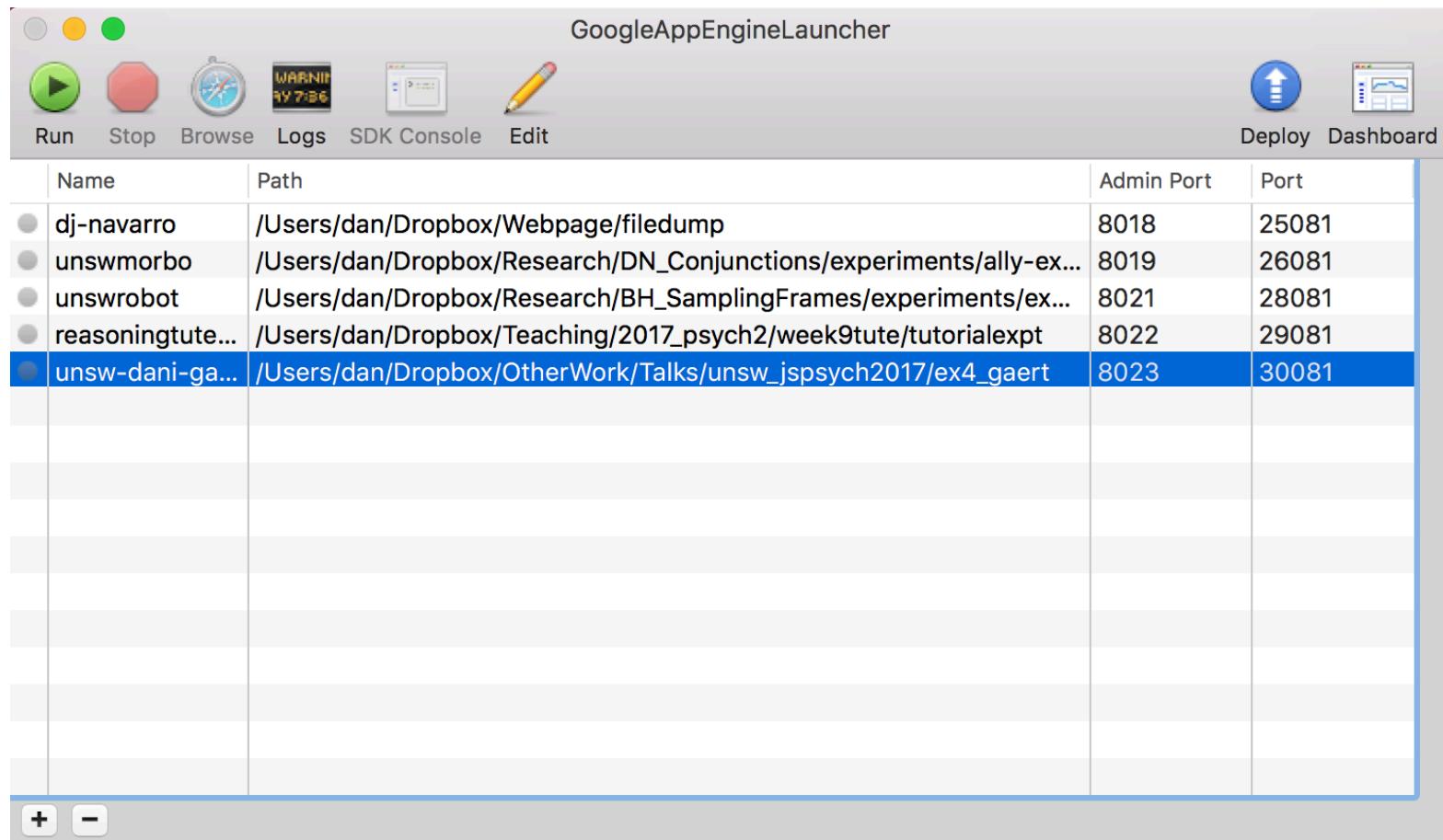
## Step 2: Within the GAE Launcher, “Add Existing Application”



## Step 2: Within the GAE Launcher, “Add Existing Application”



## Step 2: Within the GAE Launcher, “Add Existing Application”



**Step 3 (optional): Click on “run”, then “browse”, to see it running in a simulated version of the GAE environment**



Thanks for accepting the HIT. "**A short RT study**" is a short psychological study investigating how people make decisions. It involves the following steps:

1. We ask for demographic information (not connected to your Amazon ID)
2. Because this is a University research project, we ask for your informed consent. (The format of the consent form is a standard university document, so it sometimes looks a little weird on MTurk)
3. The study then explains how to do the task in detail. You will need to pass a short test to check that you understand how the study works.
4. Next comes the experiment itself.
5. At the end, we will give you the completion code you need to get paid for the HIT.

The total time taken should be about 5 minutes. Please do not use the "back" button on your browser or close the window until you reach the end and receive your completion code. This is very likely to break the experiment and may make it difficult for you to get paid. However, if something does go wrong, please contact us! When you are ready to begin, click on the "start" button below.

**Start!**

Step 4: Click on “dashboard” to take you to the (online) Google App Engine dashboard for this project, then click “select a project”....

The screenshot shows a web browser window for the Google Cloud Platform App Engine Dashboard. The URL in the address bar is <https://console.cloud.google.com/appengine?project=unsw-dani-gaert>. The dashboard itself displays a message: "You don't have permission to see App Engine resources in this project." This message is preceded by a yellow warning icon. Below this message, the text "Obviously not, because it doesn't exist yet!" is overlaid in a large, bold font. The left sidebar lists various App Engine management options: Dashboard, Services, Versions, Instances, Task queues, Security scans, Firewall rules, Quotas, and Blobstore. The "Dashboard" option is currently selected, as indicated by its blue background.

Step 5: Click on “+” to create a new project

The screenshot shows a Mac OS X desktop environment with a window titled "Select" from the Google Cloud Platform. The window lists several existing projects:

Name	ID
reasoningtute	reasoningtute-179409
compcogscisydney	compcogscisydney
unswrobot	unswrobot
dj-navarro	dj-navarro
unswmorbo	unswmorbo
unswstroop2	unswstroop2

The "Recent" tab is selected in the navigation bar. A search bar at the top allows for filtering projects and folders. A "+" button is visible in the top right corner of the dialog.

Step 6: Give your project the desired name, and then create it!

The screenshot shows a web browser window for the Google Cloud Platform. The title bar says "Google Cloud Platform". The address bar is secure (green lock) and shows the URL <https://console.cloud.google.com/projectcreate?>. The main header is "Google Cloud Platform" with a navigation menu icon on the left. Below the header, the title "New Project" is displayed. A message box contains the text: "You have 3 projects remaining in your quota. [Learn more.](#)". The "Project name" field is filled with "unsw-dani-gaert". Below the project name, a note states: "Your project ID will be unsw-dani-gaert" with a "Edit" link. At the bottom are two buttons: a blue "Create" button and a white "Cancel" button.

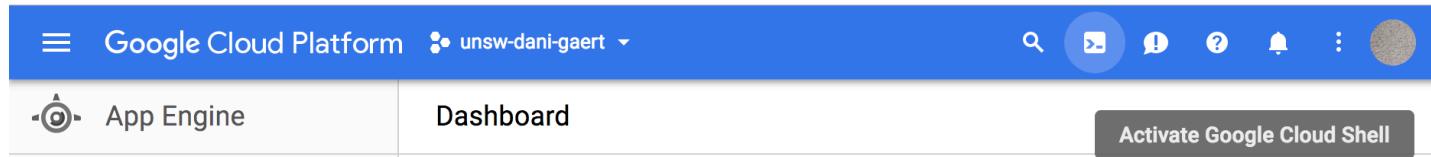
You have 3 projects remaining in your quota. [Learn more.](#)

Project name [?](#)

unsw-dani-gaert

Your project ID will be unsw-dani-gaert [?](#) [Edit](#)

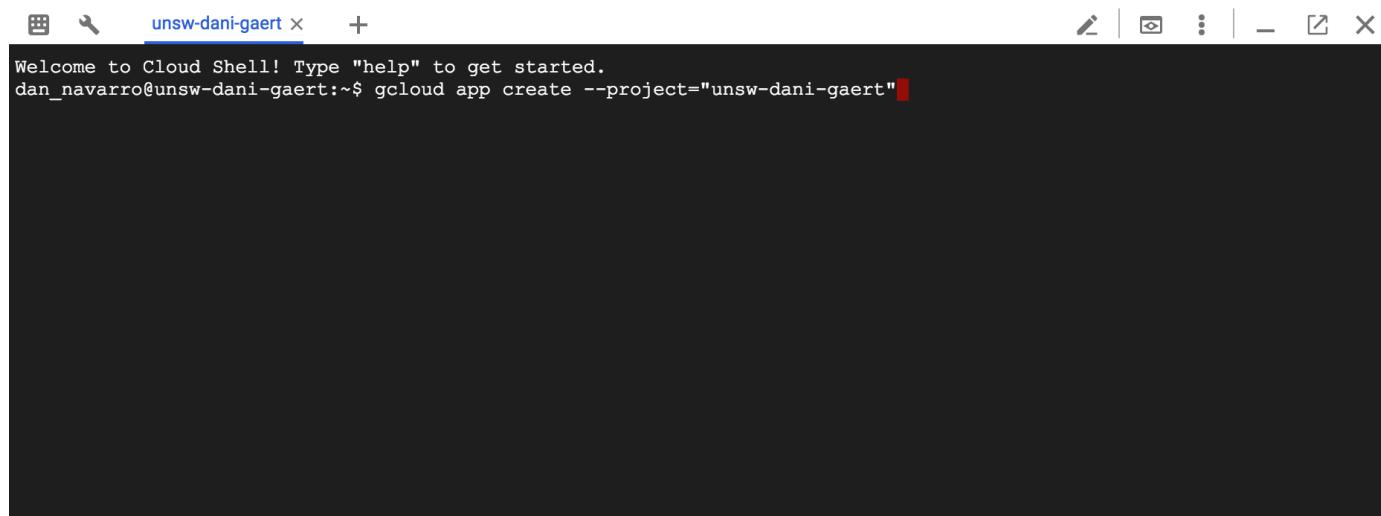
[Create](#) [Cancel](#)



The screenshot shows the Google Cloud Platform dashboard. At the top, there's a blue header bar with the "Google Cloud Platform" logo and a dropdown menu for the project "unsw-dani-gaert". To the right of the header are several icons: a magnifying glass for search, a blue circular icon with a white arrow, a speech bubble with an exclamation mark, a question mark, a bell, and three vertical dots. Below the header, there are two tabs: "App Engine" (selected) and "Dashboard". On the far right of the dashboard area is a button labeled "Activate Google Cloud Shell".

Step 7: Open up the shell and type

```
gcloud app create --project="my-project-name"
```



The screenshot shows a terminal window titled "unsw-dani-gaert". The window has standard OS X-style window controls at the top. The terminal itself is dark-themed and displays the following text:

```
Welcome to Cloud Shell! Type "help" to get started.  
dan_navarro@unsw-dani-gaert:~$ gcloud app create --project="unsw-dani-gaert"
```



```
dan_navarro@unsw-dani-gaert:~$ gcloud app create --project="unsw-dani-gaert"
You are creating an app for project [unsw-dani-gaert].
WARNING: Creating an App Engine application for a project is irreversible and the region
cannot be changed. More information about regions is at
https://cloud.google.com/appengine/docs/locations.
```

Please choose the region where you want your App Engine application located:

- [1] europe-west2 (supports standard and flexible)
- [2] us-east1 (supports standard and flexible)
- [3] us-east4 (supports standard and flexible)
- [4] asia-northeast1 (supports standard and flexible)
- [5] australia-southeast1 (supports standard and flexible)
- [6] southamerica-east1 (supports standard and flexible)
- [7] us-central (supports standard and flexible)
- [8] europe-west (supports standard and flexible)
- [9] europe-west3 (supports standard and flexible)
- [10] cancel

Please enter your numeric choice:

Step 8: Select the region (i.e., which of Google's server farms to use)



[https://cloud.google.com/appengine/docs/locations.](https://cloud.google.com/appengine/docs/locations)

Please choose the region where you want your App Engine application located:

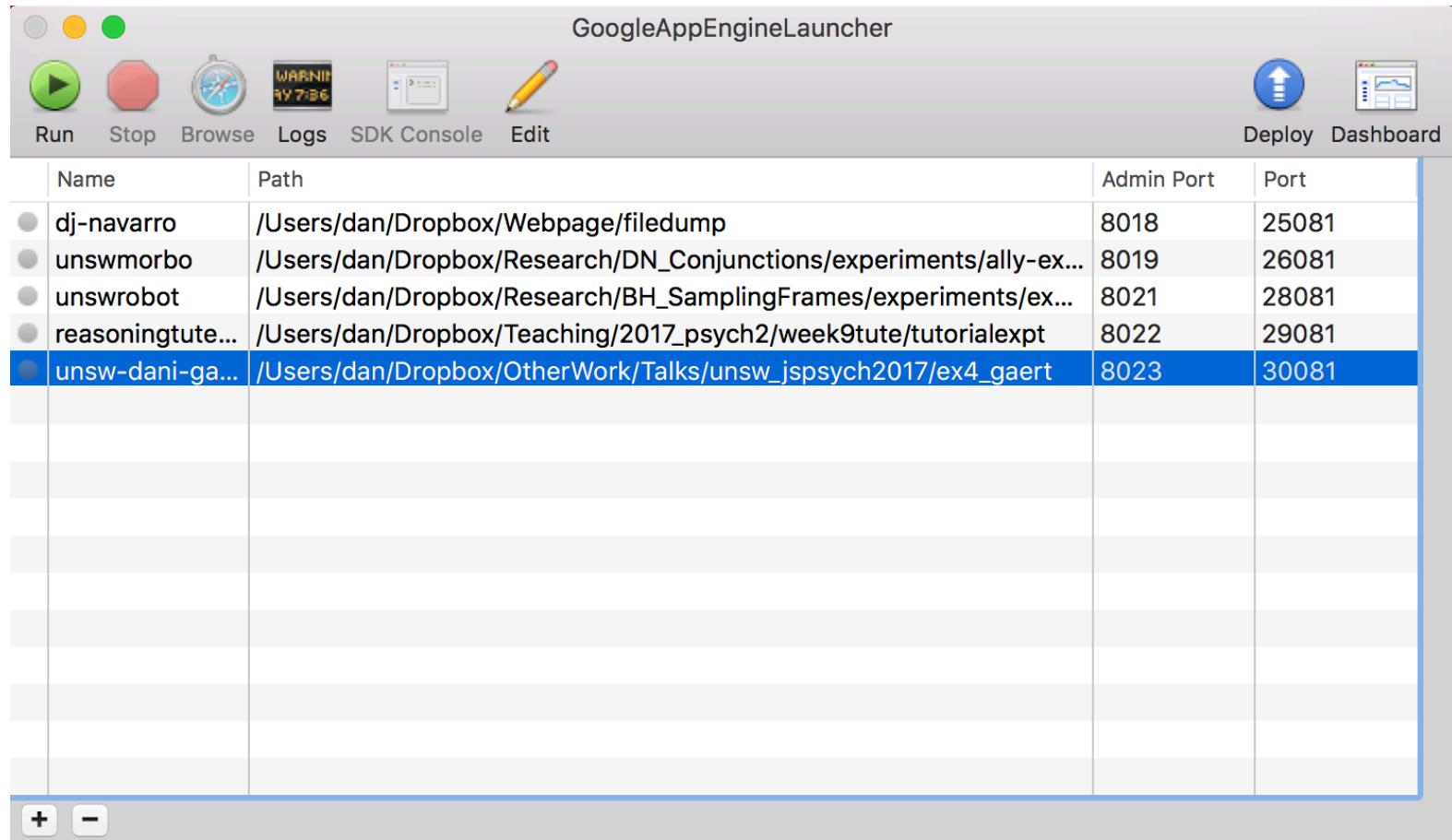
- [1] europe-west2 (supports standard and flexible)
- [2] us-east1 (supports standard and flexible)
- [3] us-east4 (supports standard and flexible)
- [4] asia-northeast1 (supports standard and flexible)
- [5] australia-southeast1 (supports standard and flexible)
- [6] southamerica-east1 (supports standard and flexible)
- [7] us-central (supports standard and flexible)
- [8] europe-west (supports standard and flexible)
- [9] europe-west3 (supports standard and flexible)
- [10] cancel

Please enter your numeric choice: 7

Creating App Engine application in project [unsw-dani-gaert] and region [us-central]....done.  
Success! The app is now created. Please use `gcloud app deploy` to deploy your first app.  
dan\_navarro@unsw-dani-gaert:~\$

Success! It's all ready to go, now we just get to...

**Step 9: Hit the deploy button!**



# UNSW Computational Cognitive Science

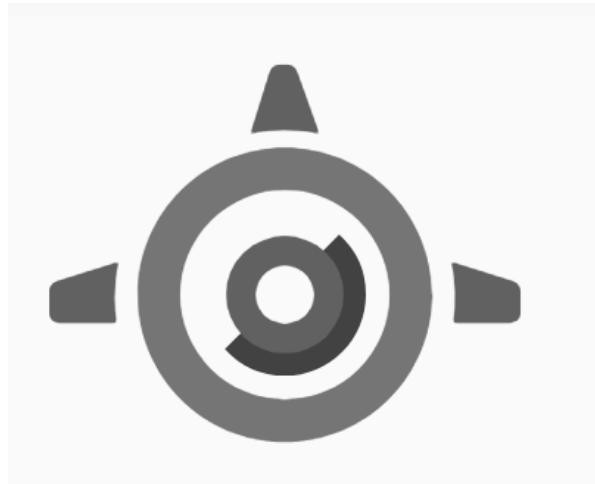
Thanks for accepting the HIT. "**A short RT study**" is a short psychological study investigating how people make decisions. It involves the following steps:

1. We ask for demographic information (not connected to your Amazon ID)
2. Because this is a University research project, we ask for your informed consent. (The format of the consent form is a standard university document, so it sometimes looks a little weird on MTurk)
3. The study then explains how to do the task in detail. You will need to pass a short test to check that you understand how the study works.
4. Next comes the experiment itself.
5. At the end, we will give you the completion code you need to get paid for the HIT.

The total time taken should be about 5 minutes. Please do not use the "back" button on your browser or close the window until you reach the end and receive your completion code. This is very likely to break the experiment and may make it difficult for you to get paid. However, if something does go wrong, please contact us! When you are ready to begin, click on the "start" button below.

Start!

Step 10: Go check out our shiny new website

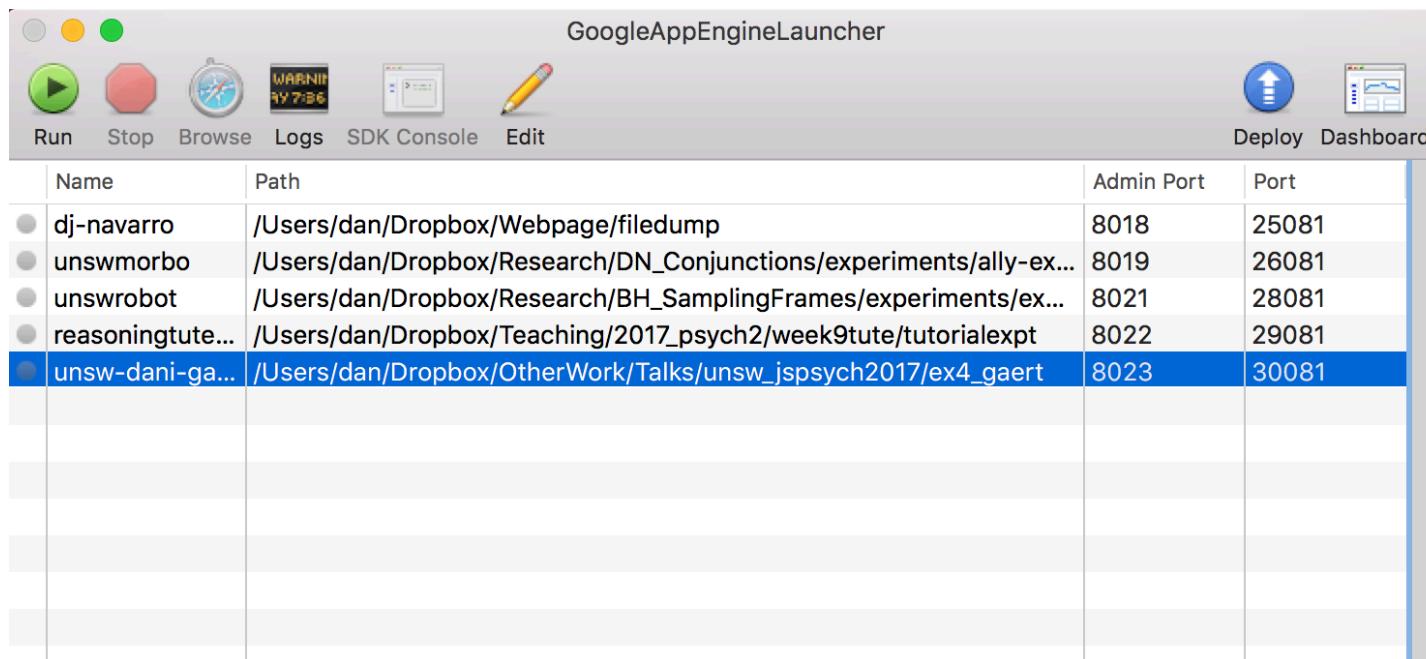


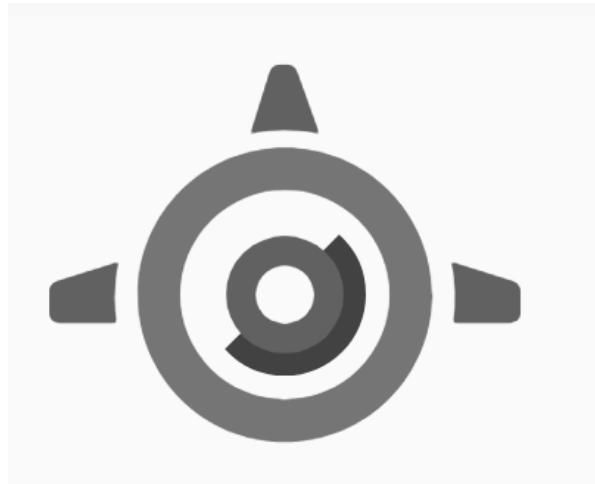
Updating the GAE project...

Step 1: Make whatever changes you want to on your local copy (e.g., uncomment the “post” line so that the application will save the data

```
48     /* save and finish */
49     function endExperiment(dataset,callback) {
50         $.post('submit',[{"content": dataset}]); // uncomment to post data
51         setTimeout(callback,1000)
52     }
```

Step 2: Hit the deploy button





Checking that your application is writing the data



Google Cloud Platform



unsw-dani-gaert ▾



Home



App Engine >



Datastore >

PRODUCTS ^

STORAGE



Entities

Dashboard

Indexes

Admin

Use the menu to navigate to the “Datastore”

☰ Google Cloud Platform unsw-dani-gaert 🔍

Datastore Entities + CREATE ENTITY ⏪ REFRESH 🗑 DELETE

Entities

Query by kind Query by GQL

Kind DataObject Filter entities Number of col

Name/ID content

id=5629499534213120 "rt","key\_press","trial\_type","trial\_index","time\_elapsed","internal\_node\_id","gend

Properties

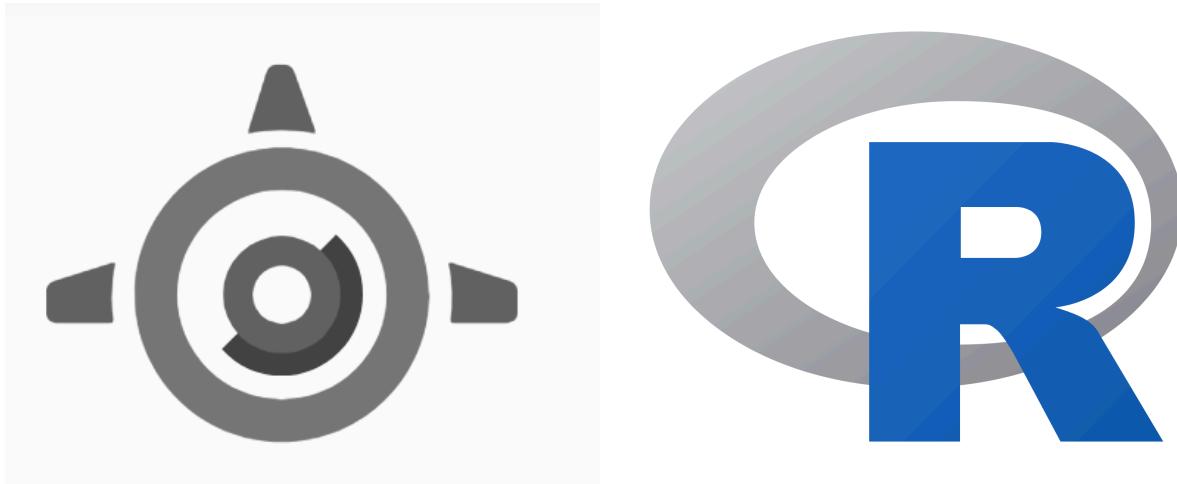
content

"rt","key\_press","trial\_type","trial\_index","time\_elapsed","internal\_node\_id","gender","age","language","country","turkcode","responses","stimulus","button\_pressed","response","correct","1513","32","text","0","1517","0.0-0.0-0.0","other","99","Klingon","Spain","529946","","","","","","2538","32","text","1","5059","0.0-0.0-1.0","other","99","Klingon","Spain","529946","","","","","7493","","survey-multi-choice","2","14572","0.0-0.0-2.0","other","99","Klingon","Spain","529946","{"Q0": "Pres ..."}

date  
2017-09-10 (13:12:32.612)

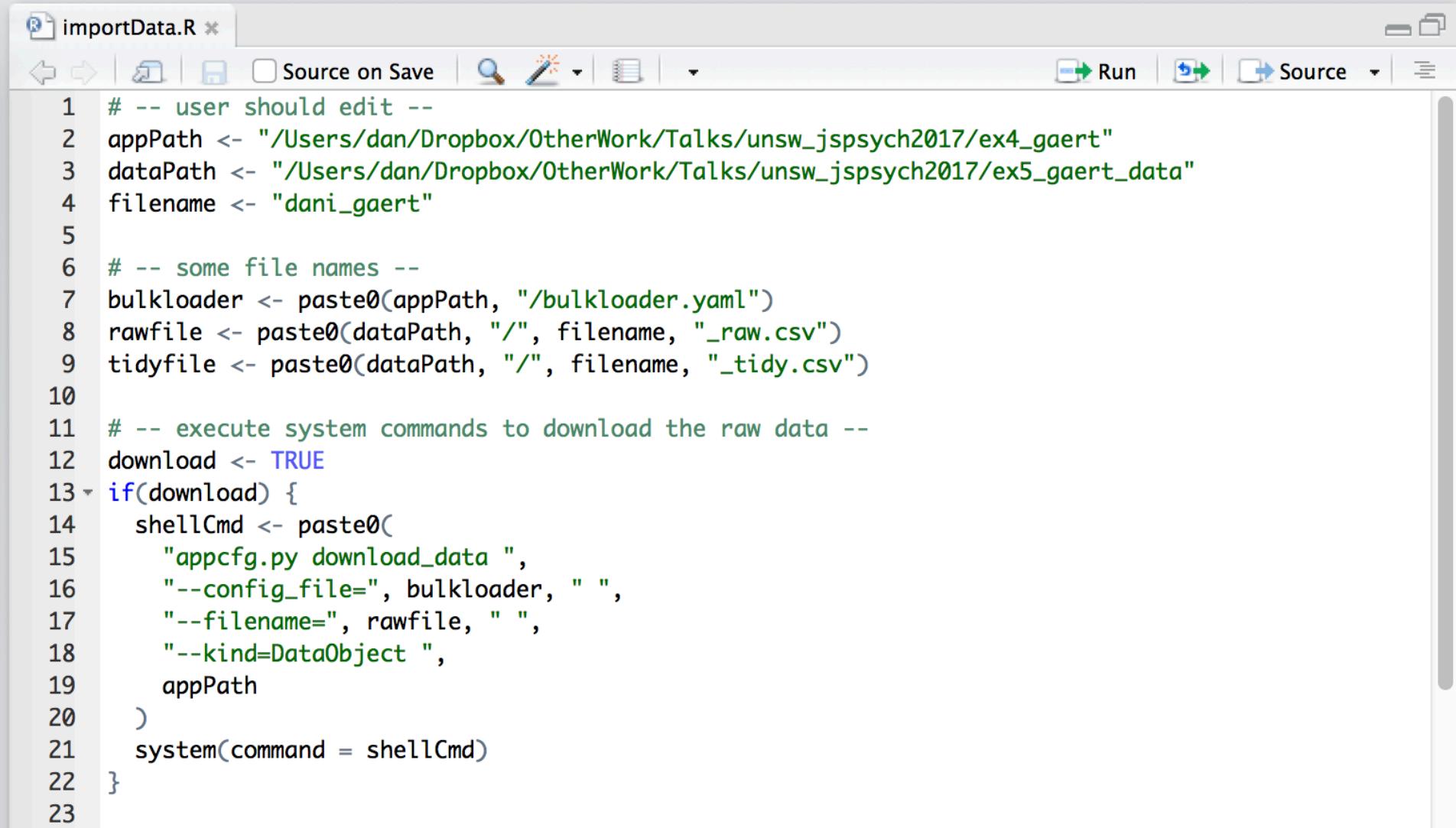
The screenshot shows the Google Cloud Datastore interface. On the left, there's a sidebar with icons for Entities, Dashboard, Indexes, and Admin. The main area has tabs for 'Entities' (selected), 'CREATE ENTITY', 'REFRESH', and 'DELETE'. Below these are sections for 'Query by kind' (set to 'DataObject') and 'Query by GQL'. A 'Filter entities' button is present. The main content area shows a table with two rows. The first row has columns 'Name/ID' (checkbox checked) and 'content'. The second row has 'id=5629499534213120' and its value, which is a JSON-like string. To the right, a large panel titled 'Entity details' shows the entity's key (DataObject id:5629499534213120), key literal (Key(DataObject, 5629499534213120)), and URL-safe key (ahFzfnVuc3ctZGFuaS1nYWVydHIXCxIKRGF0YU9iamVjdBiAgICAgICACgw). It also lists 'Properties' with their values, such as content, date (2017-09-10), and a long JSON string for the response.

Yep, there it is. We've only got one entry because only one person has completed the task, but you can see that it's all the responses for that person



Extracting and tidying the data from Google

“importData.R” is a convenient little R script. All you need to do is tell it where your experiment files are located (appPath), where you want the data saved (dataPath), and what name you would like to give the data...



The screenshot shows the RStudio interface with the file "importData.R" open. The code editor displays the following R script:

```
1 # -- user should edit --
2 appPath <- "/Users/dan/Dropbox/0therWork/Talks/unsw_jspysch2017/ex4_gaert"
3 dataPath <- "/Users/dan/Dropbox/0therWork/Talks/unsw_jspysch2017/ex5_gaert_data"
4 filename <- "dani_gaert"
5
6 # -- some file names --
7 bulkloader <- paste0(appPath, "/bulkloader.yaml")
8 rawfile <- paste0(dataPath, "/", filename, "_raw.csv")
9 tidyfile <- paste0(dataPath, "/", filename, "_tidy.csv")
10
11 # -- execute system commands to download the raw data --
12 download <- TRUE
13 if(download) {
14   shellCmd <- paste0(
15     "appcfg.py download_data ",
16     "--config_file=", bulkloader, " ",
17     "--filename=", rawfile, " ",
18     "--kind=DataObject ",
19     appPath
20   )
21   system(command = shellCmd)
22 }
```

The RStudio toolbar at the top includes icons for file operations, source control, and execution (Run, Source). The status bar at the bottom shows the current file path: "C:\Users\dan\Dropbox\0therWork\Talks\unsw\_jspysch2017\ex4\_gaert\importData.R".

The script uses the App Engine command line tools to download all data entries from the GAE project, and creates two files...

```
> source('~/Dropbox/OtherWork/Talks/unsw_jspsych2017/ex5_gaert_data/importData.R')
02:41 PM Application: unsw-dani-gaert
02:41 PM Downloading data records.
[INFO    ] Logging to bulkloader-log-20170910.144136
[INFO    ] Throttling transfers:
[INFO    ] Bandwidth: 250000 bytes/second
[INFO    ] HTTP connections: 8/second
[INFO    ] Entities inserted/fetched/modified: 20/second
[INFO    ] Batch Size: 10
[INFO    ] Opening database: bulkloader-progress-20170910.144136.sql3
[INFO    ] Opening database: bulkloader-results-20170910.144136.sql3
2017-09-10 14:41:37,015 INFO client.py:546 Attempting refresh to obtain initial access_token
2017-09-10 14:41:37,016 INFO client.py:804 Refreshing access_token
[INFO    ] Connecting to unsw-dani-gaert.appspot.com/_ah/remote_api
[INFO    ] Downloading kinds: ['DataObject']

[INFO    ] Have 2 entities, 0 previously transferred
[INFO    ] 2 entities (10335 bytes) transferred in 2.2 seconds
```

The “raw” file contains the data in the same format  
in which it was written to the server

```
content,date,key
"rt","key_press","trial_type","trial_index","time_elapsed","internal
_node_id","gender","age","language","country","turkcode","responses"
,"stimulus","button_pressed","response","correct"
1513,32,"text",0,1517,"0.0-0.0-0.0","other",99,"Kling
on","Spain",529946,,,"",
2538,32,"text",1,5059,"0.0-0.0-1.0","other",99,"Kling
on","Spain",529946,,,"",
7493,,,"survey-multi-choice",2,14572,"0.0-0.0-2.0","other",
99,"Klingon",Spain,529946,{Q0:Press
F,Q1:Do nothing},,
1532,,,"button-response",3,17110,"0.0-1.0","other",99,
"Klingon",Spain,529946,,,"Well done!",0,,,
1243,70,"single-stim",4,18355,"0.0-2.0-0.0","other",99
,Klingon",Spain,529946,,,"img/blue.png",,,"go",true
516,70,"single-stim",5,20154,"0.0-2.0-1.0","other",99,
"Klingon",Spain,529946,,,"img/blue.png",,,"go",true
-1,-1,"single-stim",6,22871,"0.0-2.0-2.0","other",99,
"Klingon",Spain,529946,,,"img/orange.png",,,"no-go",true
400,70,"single-stim",7,24874,"0.0-2.0-3.0","other",99,
"Klingon",Spain,529946,,,"img/blue.png",,,"go",true
-1,-1,"single-stim",8,28589,"0.0-2.0-4.0","other",99,
"Klingon",Spain,529946,,,"img/orange.png",,,"no-go",true
-1,-1,"single-stim",9,32275,"0.0-2.0-5.0","other",99,
"Klingon",Spain,529946,,,"img/orange.png",,,"no-go",true
-1,-1,"single-stim",10,34881,"0.0-2.0-6.0","other",99,
"Klingon",Spain,529946,,,"img/orange.png",,,"no-go",true
```

The “tidy” file is a CSV that contains one row for every trial in the experiment (including instructions and instruction check), with all participants concatenated

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	subj_id	timestamp	gender	age	language	country	turkcode	rt	key_press	trial_type	trial_index	time_elapsed	internal_node_responses	stimulus	button_press_response	correct	
2	1	2017-09-10T03:12:32	other	99	Klingon	Spain	529946	1513	32	text	0	1517	0.0-0.0-0.0		NA		
3	1	2017-09-10T03:12:32	other	99	Klingon	Spain	529946	2538	32	text	1	5059	0.0-0.0-1.0		NA		
4	1	2017-09-10T03:12:32	other	99	Klingon	Spain	529946	7493	NA	survey-mul	2	14572	0.0-0.0-2.0	{"Q0":"Press F","Q1":"Do nothing"}			
5	1	2017-09-10T03:12:32	other	99	Klingon	Spain	529946	1532	NA	button-resp	3	17110	0.0-1.0		Well done!	0	
6	1	2017-09-10T03:12:32	other	99	Klingon	Spain	529946	1243	70	single-stim	4	18355	0.0-2.0-0.0	img/blue.png	NA	go	TRUE
7	1	2017-09-10T03:12:32	other	99	Klingon	Spain	529946	516	70	single-stim	5	20154	0.0-2.0-1.0	img/blue.png	NA	go	TRUE
8	1	2017-09-10T03:12:32	other	99	Klingon	Spain	529946	-1	-1	single-stim	6	22871	0.0-2.0-2.0	img/orange.png	NA	no-go	TRUE
9	1	2017-09-10T03:12:32	other	99	Klingon	Spain	529946	400	70	single-stim	7	24874	0.0-2.0-3.0	img/blue.png	NA	go	TRUE
10	1	2017-09-10T03:12:32	other	99	Klingon	Spain	529946	-1	-1	single-stim	8	28589	0.0-2.0-4.0	img/orange.png	NA	no-go	TRUE
11	1	2017-09-10T03:12:32	other	99	Klingon	Spain	529946	-1	-1	single-stim	9	32275	0.0-2.0-5.0	img/orange.png	NA	no-go	TRUE
12	1	2017-09-10T03:12:32	other	99	Klingon	Spain	529946	-1	-1	single-stim	10	34881	0.0-2.0-6.0	img/orange.png	NA	no-go	TRUE
13	1	2017-09-10T03:12:32	other	99	Klingon	Spain	529946	450	70	single-stim	11	36777	0.0-2.0-7.0	img/blue.png	NA	go	TRUE
14	1	2017-09-10T03:12:32	other	99	Klingon	Spain	529946	-1	-1	single-stim	12	40488	0.0-2.0-8.0	img/orange.png	NA	no-go	TRUE
15	1	2017-09-10T03:12:32	other	99	Klingon	Spain	529946	450	70	single-stim	13	42565	0.0-2.0-9.0	img/blue.png	NA	go	TRUE
16	1	2017-09-10T03:12:32	other	99	Klingon	Spain	529946	380	70	single-stim	14	44845	0.0-2.0-10.0	img/blue.png	NA	go	TRUE
17	1	2017-09-10T03:12:32	other	99	Klingon	Spain	529946	441	70	single-stim	15	46781	0.0-2.0-11.0	img/blue.png	NA	go	TRUE
18	1	2017-09-10T03:12:32	other	99	Klingon	Spain	529946	362	70	single-stim	16	48884	0.0-2.0-12.0	img/blue.png	NA	go	TRUE
19	1	2017-09-10T03:12:32	other	99	Klingon	Spain	529946	-1	-1	single-stim	17	51420	0.0-2.0-13.0	img/orange.png	NA	no-go	TRUE
20	1	2017-09-10T03:12:32	other	99	Klingon	Spain	529946	-1	-1	single-stim	18	53869	0.0-2.0-14.0	img/orange.png	NA	no-go	TRUE
21	1	2017-09-10T03:12:32	other	99	Klingon	Spain	529946	-1	-1	single-stim	19	57426	0.0-2.0-15.0	img/orange.png	NA	no-go	TRUE
22	1	2017-09-10T03:12:32	other	99	Klingon	Spain	529946	-1	-1	single-stim	20	59824	0.0-2.0-16.0	img/orange.png	NA	no-go	TRUE
23	1	2017-09-10T03:12:32	other	99	Klingon	Spain	529946	378	70	single-stim	21	61491	0.0-2.0-17.0	img/blue.png	NA	go	TRUE
24	1	2017-09-10T03:12:32	other	99	Klingon	Spain	529946	-1	-1	single-stim	22	64142	0.0-2.0-18.0	img/orange.png	NA	no-go	TRUE
25	1	2017-09-10T03:12:32	other	99	Klingon	Spain	529946	414	70	single-stim	23	65840	0.0-2.0-19.0	img/blue.png	NA	go	TRUE
26	1	2017-09-10T03:12:32	other	99	Klingon	Spain	529946	3196	32	text	24	70651	0.0-3.0		NA		
27	2	2017-09-10T03:44:05	NA	NA	NA	United St	675907	1417	32	text	0	1422	0.0-0.0-0.0		NA		
28	2	2017-09-10T03:44:05	NA	NA	NA	United St	675907	1219	32	text	1	3643	0.0-0.0-1.0		NA		
29	2	2017-09-10T03:44:05	NA	NA	NA	United St	675907	5230	NA	survey-mul	2	10887	0.0-0.0-2.0	{"Q0":"Do nothing"}	NA		
30	2	2017-09-10T03:44:05	NA	NA	NA	United St	675907	2281	NA	button-resp	3	14174	0.0-0.0-3.0-0.0	Unfortunately, at	0		
31	2	2017-09-10T03:44:05	NA	NA	NA	United St	675907	3927	32	text	4	18102	0.0-0.1-0.1		NA		
32	2	2017-09-10T03:44:05	NA	NA	NA	United St	675907	685	32	text	5	19790	0.0-0.1-1.1		NA		
33	2	2017-09-10T03:44:05	NA	NA	NA	United St	675907	5908	NA	survey-mul	6	27706	0.0-0.1-2.1	{"Q0":"Press F","Q1":"Do nothing"}	NA		
34	2	2017-09-10T03:44:05	NA	NA	NA	United St	675907	2275	NA	button-resp	7	30988	0.0-1.0	Well done!	0		
35	2	2017-09-10T03:44:05	NA	NA	NA	United St	675907	-1	-1	single-stim	8	32491	0.0-2.0-0.0	img/orange.png	NA	no-go	TRUE
36	2	2017-09-10T03:44:05	NA	NA	NA	United St	675907	341	70	single-stim	9	34382	0.0-2.0-1.0	img/blue.png	NA	go	TRUE
37	2	2017-09-10T03:44:05	NA	NA	NA	United St	675907	-1	-1	single-stim	10	37500	0.0-2.0-2.0	img/orange.png	NA	no-go	TRUE

Done!

