Modelling

- *Activations.Rmd*

In this file I plot the activation over the entire experiment for one participant. This file DOES NOT contain the plots for the test phase.
First I simulate one participant doing the entire experiment (in the order 050, 000, 100). The second block then plots the allergic reaction minus the not allergic one. It colours the one from the first block, while greying out the rest. The block after, the same is plotted but then every cue has a different colour.
The same is then done for the activation to JUST one outcome, first to allergic then to not allergic
After that there are some plots that show the same but are not pretty.

- *ActivationsOutExperiment1.Rmd*

This file does the same as *Activations.Rmd* does, but then with the data for our experiment 1. As this file is super similar to *Activations.Rmd* I mostly used it to check if the results were indeed the same, so it's not as polished as the plots in the former file are.
As I've only simulated the test phase here, the last two plots show what would happen if there was a test phase (only phase 1). These plots have the labels "cabbage, bran, yoghurt" instead of the new stimuli, as I've used this plot for a presentation and it was clearer to use the same labels everywhere.

- *AdjustedRuns.Rmd*

This file shows what happens if there were more trials or more stimuli in the original Van Hamme and Wasserman paper. This was to check if we wanted to test something like that, but the results weren't different enough to do that.

- *Experiment1ModelPlots.Rmd*

This file is the *Plots.Rmd* file, but then for the simulation of our Experiment 1. The test stuff and things plotted in *ActivationsOurExperiment1.Rmd* are also plotted here, but in a very rough and not very neat fashion.

- *FirstSetupVHWExp.Rmd*

My first attempt at modelling the VHW experiment, just ignore this file. The same info is now stored in *OriginalExperimentRun.R* and *VanHammeWassermanFunctions.R*

- *FixingVHW.Rmd*

Same as above, just ignore this file. All info is stored in *OriginalExperimentRun.R* and *VanHammeWassermanFunctions.R*

- *MoreFoodCond.R & MoreStimuli.R & MoreTrials.R*

Both of these files simulate the VHW experiment with either more food types or more stimuli. The results of these are plotted in *AdjustedRuns.Rmd*

- *OriginalExperimentRun.R*

This file simulates the original experiment run. It creates the different food conditions with cues, then the different outcome conditions, which it then runs with ParticipantRun (from *VanHammeWassermanFunctions.R)* for participants as shown in table 2 of the original paper.

- *OurExperiment1Run.R*

Same as *OriginalExperimentRun.R* but then for our experiment 1.

- *OurExperiment1RunTest.R*

Ignore this file, same as the above file, but I tested out how I needed to adjust the functions used in the file to make sure the test phase was handled well.

- *Plots.Rmd*

This file contains the original plots of the VHW paper (recreated in R by just kind of estimating the original values) compared to the plots created by the models of the VHW experiment.

- *RandomFreq.R & RandomFreqMax2.R*

In these two files I wanted to test what happened if we randomized the cue (instead of having the order always be AX, BX, BX, AX etc). However I didn't complete these files as I didn't quite know how to implement it well (make sure that stuff doesn't show up more than twice in a row etc).

- *VanHammeWassermanFunctions.R*

The functions used to simulate the VHW experiment. These are the adjusted version of updateWeights, RWLearning (now EDLearning), a function to fill in the cues from the string, a function to recreate what happens in the run of one participant, and a function to recognize which word type was asked. This file is called in *OriginalExperimentRun.R*

- *VanHammeWassermanFunctionsOurExp.R*

Same as *VanHammeWassermanFunctions.R* but then adjusted to be able to handle our experiment 1.

Experiment files, Python

All of these files can be run by opening then in Open Sesame, and then pressing either of these buttons: 

The big green one runs the experiment in full screen (if you want to quit press esc, and then Q), the two green ones run the experiment in a smaller screen (esc still works, but now you can also press the red "X" button next to these arrows). For both of these arrows you'll have to put in a certain subject number. If you just quickly want to check something that is not dependent on the subject number then you can press the blue arrows, which run a "Quickrun".

It's important to know that in the Python implementation you can work with something called a "Form". In this form you can determine how many rows and columns something has and how big all of those are when compared to each other. This means that I don't have to directly say "Hey this goes on coordinate X and Y" but I can just say "Put this in this [column][row] place", which automatically spaces things.

Something that's VERY IMPORTANT for both the Python and the Javascript files, if you create something yourself, ALL the loggers should be linked copies of each other. Otherwise you have many separate log files, instead of one big one.

- *Experiment1.osexp*

  This file combines *TestingPhase.osexp* and *TestingPhase.osexp* (together with an added practice phase) into a run of the first experiment. I've commented the separate inline python codes pretty well I'd say, but I'll explain the general idea of the file here.
  A participant gets sorted into one of the 6 groups, and is assigned stimuli randomly. They then see some explanation and get a quick practice run where they can practice with the ratings. (yes the text is below the input boxes, that's not a bug sadly).
  Once they're done practicing they go into the training phase, and see the three training phases in a certain order (depending on the group they're in).
  They see the stimuli for 2 seconds, outcome and stimuli together for 1 and then they have 12 seconds to rate. For the rating the python inline script I added a validator that checks if all three boxes are filled in, otherwise they don't continue.
  When they're done with a block they get some text and then move on the the next block. If the full training phase is done they continue to the test phase (with some explanation first). Here basically the same thing happens, but now they don't see the outcome appear, they just have to rate how likely they thing that a certain outcome *will* appear.

- *Experiment1Eduard.osexp*

  This file is the file I received from Eduard from the OpenSesame forum, he introduced lots of improvements on the code. I mainly keep this to remind myself where some of the short hand code comes from.

- *RatingSystemTest.osexp*

  I couldn't get the rating system to work, so this file is created to mess around with different styles of rating. In this case sliders.

- *TestingPhase.osexp & TrainingPhase.osexp*

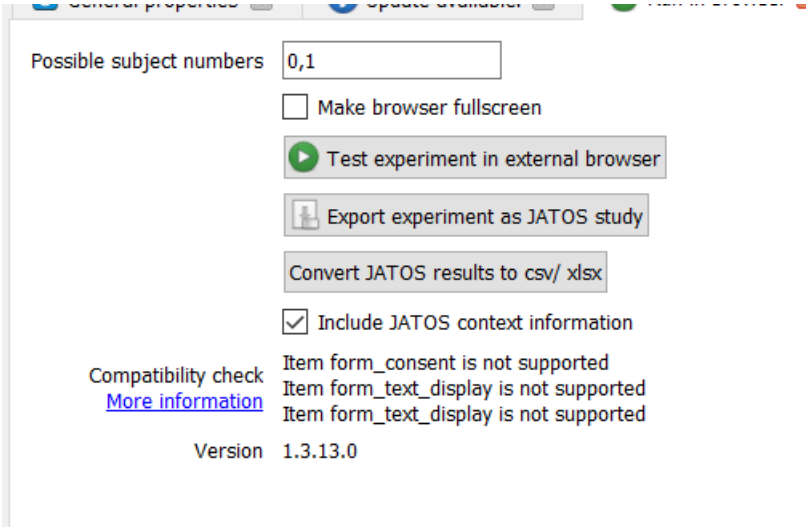
  These files contain the testing phase and training phase that are combined in *Experiment1.osexp*. Really useful for debugging or just testing them separately/seeing how long they take etc.

- *WidgetTest.osexp*

  Ignore this file. It's an early test.

Experiment files, JavaScript

To run the experiment online I had to revert the Python inline scripts to JavaScript inline scripts. This time instead of running the files with the arrows seen in the Python part, you will need this  button. It will run the experiment in a browser screen. Although the "Quickrun" can be used for debugging, this one uses Python as a base, so there are things that will work in the "Quickrun" but not in the browser and the other way around (for example Num pad implementation works in the Browser, not in the Quickrun).
If you press on the green button you will see the following screen: (at least if you have enough things in your file, if you do it when there is no file or a very simple one it will just run right away)



You can fill in the possible subject numbers, run the experiment in the browser (so you have to press the button first and then the grey line before it runs). Once you want to run the OsWeb file as a Jatos experiment you can export it to a .zip with the second button and the third converts the output of the Jatos results to a csv.
It also states that things such as "form_consent" are not supported, however it still works so I chose to ignore that. The version is the version of OsWeb that you have running.

OsWeb does not support things such as "Form" which I used to create the layout in Python. What I did is I calculated what points would make up the beginning and end of the columns and rows (minus the margins and stuff) and put those in as global variables. It's a bit weird, but for me that was easier than to figure it out myself.

If you want to run any of the full experiment in the browser you will have to comment out "var prolific_id = jatos.url…" and "vars.prolific_id =…" in the *Initialization* file, and in the full experiments

of 1.5 and 2 you will also need to comment out everything in the *Survey* file. When you want to run the experiments on JATOS you need to make sure that these lines are UNCOMMENTED again.

- *Debugfile.osexp*

  A file in which I was trying to figure out how to randomize the location of the stimuli in JavaScript.

- *BasisClickerExperiment.osexp*

  A file that I send to the Forum to see what was going wrong. In this file I still had the OLD version of the rating system, which would only go to the next file if a person did nothing for 12 seconds. As soon as a button was pressed the loop would reset. This is fixed and while the correct versions are of course seen in the full experiment files, it's seen well in *JavaScriptBasis.osexp*

- *TyringToGetOSToWork.osexp*

  This is a demo file of my experiment, where I first got the rating system to work properly. This is also the first file that I ran on JATOS to see how that worked.

- *JavaScriptBasis.osexp*

  A basis working file, if something went wrong I could revert to what was in here. This only has the training phase.
  It works as follows: the keyboard_loop loop loops (wow) every time that a character is pressed, the only time it breaks out of this loop is by the two things mentioned in the *Repeat_cycle*. This file indicates when the cycle *should* be repeated. So as long as these things are still true, the cycle repeats.
  The first is if the [check] == "no", where check is a variable that checks if all 3 ratings are filled in. So as long as a person only has 2 answers they cannot continue. However we also have a timing going on in the background. This one is first set in *SetTimer* outside of the *Keyboard_loop*. Within the loop the amount of elapsed time is subtracted from the total amount of time that a person has. If this is 0 or lower, then a timeout occurs. So as long as a timeout has not occurred, and not all 3 ratings are filled in, the *Keyboard_loop* will keep repeating and accepting input.

- *TrainingPhase.osexp & TestPhase.osexp & PracticeTest.osexp*

  Each respective file holds the training, test and practice phase

- *FullExperiment1OLD.osexp*

  As I adjusted FullExperiment 1 to experiment 1.5, but in the same file, this file contains the ORIGINAL experiment 1 file, while FullExperiment1, is actually experiment 1.5!
  In this experiment people first saw a consent form that they had to accept, after which they got some explanation which led to a (very barebones) practice phase, where they could test out the rating.
  Then based on which group they were in they saw a specific order of training phases. Each phase showed the cues for 2 seconds, the cues and outcomes for 1, then participants could rate for 12 seconds. They get a bit of text when they are done with a block. Once they are done with the training, they get some explanation and move on to the test phase. This phase has 3 parts, but the participant does not notice that, as they are just shown right after each other. At the end they are thanked for their participation and the experiment ends.

- *WorkingBasis.osexp*

This file holds the working basis for experiment 2. It only has one condition, so it's more of a debug file. This file also contains the link to the Survey in the last inline script.

- *FullExperiment1.osexp*

This file is actually the file for experiment 1.5. *FullExperiment1OLD.osexp* contains the original experiment 1. The differences between these two files are:
Experiment 1.5 has different text, more to the point and clearer. It also has a different test phase, where it's more similar to the rest of the experiment, the stimuli are also updated as the three outcomes are now more visually distinct. There is also a survey at the end of the experiment (which is a connection to a Qualtrics survey).

- *WorkingBasisExp2.osexp*

Same as *WorkingBasis.osexp* but now for experiment 2.

- *FullExperiment2.osexp*

The file for experiment 2. This time instead of pictures that are shown to the participant, they see words. We tried to stick as closely to the original experiment as possible, asking different foods and seeing whether they results in an allergic reaction or not. The explanations are also much more similar to the original one, being exactly the same where possible.
This file thus needs updating as now we show text instead of pictures, and the stimuli are no longer shown randomly. The time out system still works the same, with the exception that now everything is just seen in one go. This means that I set Stim and Outcome to a duration of 0 (aka I don't have to delete them but they are not shown anymore).
We also added a fourth test phase, where people just see one stimulus, and have to rate for two outcomes.
This experiment still has the survey at the end.

Experiment results files

Experiment 1, 1.5 and 2 all have their own result R files, however, they contain the same type of code so I'll walk you through the general steps.

First we load in the csv we got from Jatos. As this CSV contains all participants, we will have to manually filter out the ones that Returned their submission or where rejected.
There is some pre-processing of the data, after which I look at the amount of 1's people answered. After that the plots of the current experiment are compared to the plots of the original experiment, and the individual runs of a couple of participants are plotted.
We look at whether the no outcome trials are scored lower than the outcome trials, after which we continue on to analysing the test phase. The test phase is analysed per phase, and in each phase we basically look at the same thing. Does the score people gave to the old outcome in the test phase differ from that given in the training phase, do these scores differ between phases, do the scores to the old and new outcome differ.

Jatos, how does it work?

First you will have to ask Remco to create an account for you (or whoever the CIT person is at the uni). At the RUG we use the url: https://turing14.housing.rug.nl/jatos.

Once you are logged in you will be able to click on “Import Study”, this will allow you to upload the .zip file that you have. The experiment will then appear on the left with a generic name. You can change that name by clicking on it, and then going to “Properties” in the TOP of the screen. The rest of this page is not super interesting except for the field “End Redirect URL”. We need that later if we want to link this to Prolific.
The “Properties” on the right of the name allow you to change which subject numbers are allowed and allow you to change the name of the file within the experiment.

Now we uploaded the experiment and we can do an example run if we want by pressing “Run”. However how do we get a link for other people? You can do that by going to the green “Worker and Batch Manger” button.
Here you can either press the “Get Worker Links” or the arrow in front of default. Either way you will get to choose which types of workers you want to create links for. A Jatos worker is just for debugging, Personal Single is a small targeted group, where the experiment is created when you create the link, a personal multiple is the same but now a same person can do the experiment multiple times, a general single which is the same as personal single but now the experiment is created whenever someone clicks a link (so you don’t have to say how many people will test), general multiple same as general single but now someone can do the same experiment multiple times and Mturk for when you run with amazon turk. For more exact info see: https://www.jatos.org/Worker-Types.html.
I used a general single. This DID mean however that if someone refreshed the page, they could not take the experiment again (unless they got rid of all cookies).

Whichever type you choose, you can “Get Link” and then send the link to the people you want to take the experiment.

Prolific, how does it work?

If you want to run an experiment on Prolific you will have to create a researcher account, this has different abilities from a participant account.
To run a study you can press “New study” (no worries nothing runs until you say to Prolific 30 times that you actually want to run, so if you just want to mess around and see what’s possible, you can do it). The layout has changed 2 times in the 2 months that I ran an experiment on here, so I’m going to explain this in the broadest terms possible, as the layout might look different for you.

First you can add the title of your study, which others can see, and if you want an internal study name. Then a description. It’s useful to read the tips that the Prolific forum gives you, as they state what should and should not be in there.
You can also check which devices can be used to do your experiment (in this current one, only desktop), and whether people need a microphone, camera or something else.

Under URL of my study we can paste the worker URL that we got earlier. However, as we also want to log the prolific_id, as can be seen in the OpenSesame code in *Initialization:*

```
to directly be put in and I can trace it back to pay them
var prolific_id = jatos.urlQueryParameters.prolific_pid;
vars.prolific_id = prolific_id
```

We want to not only paste the URL of our study, but then add to the back of it "&prolific_pid={{%PROLIFIC_PID%}}&STUDY_ID={{%STUDY_ID%}}&SESSION_ID={{%SESSION_ID%}}" This is case sensitive. If you press the button "I'll use URL parameters" in prolific, it will automatically add the ending to your url. HOWEVER in the current code we log prolific_pid not PROLIFIC_PID, so be careful of that!!

As for the study completion, Prolific will redirect people and give them a completion code. This is useful later on to check if people actually completed or not (note, in the current experiment versions with the questionnaire, there will be no completion code, this is normal).
This is where we need to start the JATOS back up and go to the "End Redirect URL" in the properties tab up top. Here we can copy the URL that we get from Prolific (e.g. https://app.prolific.co/submissions/complete?cc=6EEFF865). This now means that Jatos will automatically handle the redirection.

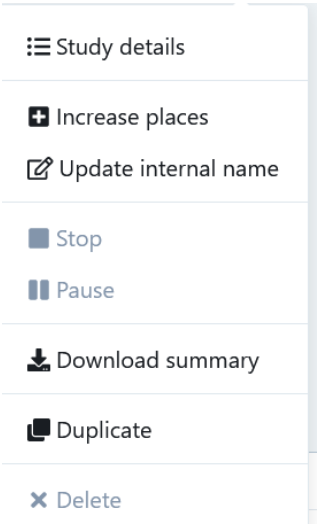
After all of this you can select specific people if you want. For example, participants who have not done your other studies, or participants from a specific country/language/background etc. This can be done in the "I want to apply custom prescreening" tab. The "Representative sample" just lets you choose a certain country.

Prolific wants you to upload the appropriate amount of money before you can run the experiment. Luckily they have a calculator in the "New studies" tab as well. Here you can state the amount of participants, how long you expect them to take, and how much you want to pay them (however that changes automatically and I'd recommend keeping the amount that Prolific suggests).
It will then show you the total costs.

If you have never run an experiment before, Prolific will give you 1 free test subject to test your experiment. Otherwise it's smart to press "Preview" under which there is a button with "Open study link in new window".
Do note though that this result will be logged in Jatos, so it's smart to remove this Jatos results before running the experiment, to avoid accidentally including your own test results.

Before I explain how running the experiment and paying participants goes, I want to quickly explain how to upload money to prolific.
You can either do this by bank transfer, or by credit card. I personally did it by bank transfer, and as they are a British company, there will be extra costs connected to that (as you transfer to outside of the EU). You can also request an invoice, such that you have a receipt to show when you want to get the money back from University.

Running the experiment. As soon as you've pressed "Publish" and stated that you read through everything and everything is correct and such, your experiment will be put online. The participants will start to come in quite quickly. You can find the study under "Active" and if you press the button "Action" in the top right corner when you selected an experiment you will see this:



You can see the study details, increase the amount of participants if you want, pause the experiment (no new participants can join) or full stop it. Download summary will give you information about age, country, language and working status of the participants.

The screen where the participants come in will look something like this on Prolific:

PARTICIPANT PROLIFIC ID	STARTED	TIME TAKEN	COMPLETION CODE	STATUS
5f5176476494183c40718758	28 Apr 2021, 11:12	00:15:56	NOCODE	APPROVED
5c879642f8e57d00168a286e	28 Apr 2021, 11:12	00:20:57	NOCODE	APPROVED
5f9ee672868a694c8b64bfbc	28 Apr 2021, 11:13	00:30:32	NOCODE	APPROVED
5ebd009d6a8f8f154b919e3a	28 Apr 2021, 11:13	00:23:55	NOCODE	APPROVED
6089161351d37ca59516f57d	28 Apr 2021, 11:14	N/A		RETURNED
602e3452f1c6b793faa7bcdd	28 Apr 2021, 11:13	00:20:34	NOCODE	APPROVED
6030191b2458093270559505	28 Apr 2021, 11:13	00:01:05	8561C38A	RETURNED
60815ba971b96b9204d01141	28 Apr 2021, 11:14	00:25:03	NOCODE	APPROVED
5f74f3c9e457b814d078623d	28 Apr 2021, 11:13	N/A		RETURNED
602fc5844525b3d343303a2a	28 Apr 2021, 11:13	00:18:06	NOCODE	APPROVED
5e8c4f7b7980cb03567f42d7	28 Apr 2021, 11:13	N/A		RETURNED
5ef6d6808c92d83c9a8d24a5	28 Apr 2021, 11:19	00:26:04	NOCODE	APPROVED
5eff169018a0e6046db7c405	28 Apr 2021, 11:13	00:26:49	NOCODE	APPROVED

Participants can have different statuses:
If they are still busy it will show "In Progress", if they are done "Awaiting Approval" and if they decided to stop at any point (aka just opt out) it shows "Return".
You can accept and reject submissions by clicking on either the checkmark (paying) or the cross (rejecting). However rejecting a participant will cause them to get a strike on their account, and it's a three strikes and you're out system. Therefore it's better/politer to first send them a message. (e.g. "Hey you answered 1 for everything, I do not believe that you answered the honestly, please return your submission", instead of rejecting them right away.
https://researcher-help.prolific.co/hc/en-gb/articles/360009092394-Reviewing-submissions-How-do-I-decide-who-to-accept-reject- This link explains more about when to reject or accept people, but the basics are that you should almost never reject someone, and that if they did do a substantial amount of work you pretty much always have to pay them.

So how to check if someone did a good job and if you can pay them or not? For that we will have to look at the results in Jatos. However going to Jatos Results you will see this screen:

		Result ID	Start Time	Last Seen	Duration	Batch	Worker ID	Worker Type	MTurk Worker ID (Confirmation Code)	State	Message
1	-	1158	2021/04/30 07:03:46	2021/04/30 07:21:49	00:20:01	Default	956	General Single	none (none)	FINISHED	none
1	-	1157	2021/04/30 06:55:49	2021/04/30 06:57:58	00:04:27	Default	955	General Single	none (none)	FAIL	It's not allowed to reload this component (ID: 264). Study (ID: 264) is finished.
1	-	1156	2021/04/28 10:53:13	2021/04/28 11:03:14	00:10:19	Default	954	General Single	none (none)	FINISHED	none
1	-	1155	2021/04/28 10:31:38	2021/04/28 10:47:40	00:17:25	Default	953	General Single	none (none)	FINISHED	none
1	-	1154	2021/04/28 10:30:16	2021/04/28 10:46:19	00:16:49	Default	952	General Single	none (none)	FINISHED	none
1	-	1153	2021/04/28 09:53:52	2021/04/28 10:29:55	00:36:03 (not finished yet)	Default	951	General Single	none (none)	DATA_RETRIEVED	none
1	-	1152	2021/04/28 09:48:54	2021/04/28 09:53:01	00:04:06 (not finished yet)	Default	950	General Single	none (none)	DATA_RETRIEVED	none
1	-	1151	2021/04/28 09:43:26	2021/04/28 10:05:28	00:22:50	Default	949	General Single	none (none)	FAIL	It's not allowed to reload this component (ID: 264). Study (ID: 264) is finished.
1	-	1150	2021/04/28 09:41:55	2021/04/28 09:45:56	00:04:10	Default	948	General Single	none (none)	FAIL	It's not allowed to reload this component (ID: 264). Study (ID: 264) is finished.
1	-	1149	2021/04/28 09:20:08	2021/04/28 09:38:31	00:20:19	Default	947	General Single	none (none)	FINISHED	none

This does not clearly show which participant is equal to which one on Prolific (as you can see in the MTWorker ID tab they do have that functionality for MTurk). So what you will have to do is press “Export results”. This results in a .txt file which you can convert to a csv/excel file in OpenSesame (as discussed in Experiment files, JavaScript).
Therefore it’s easier to wait until most participants are done, otherwise you’ll have to go through the download/convert/open/check phase a lot.
For this reason it’s very important to log the prolific ID in your OpenSesame file, this is the only way you can link these two. Once you think a person did a good job, you can press the green check in Prolific and they are then automatically payed!

In the tab “Messages” in Prolific, which is located next to “Studies” you will get messages from participants where something went wrong.