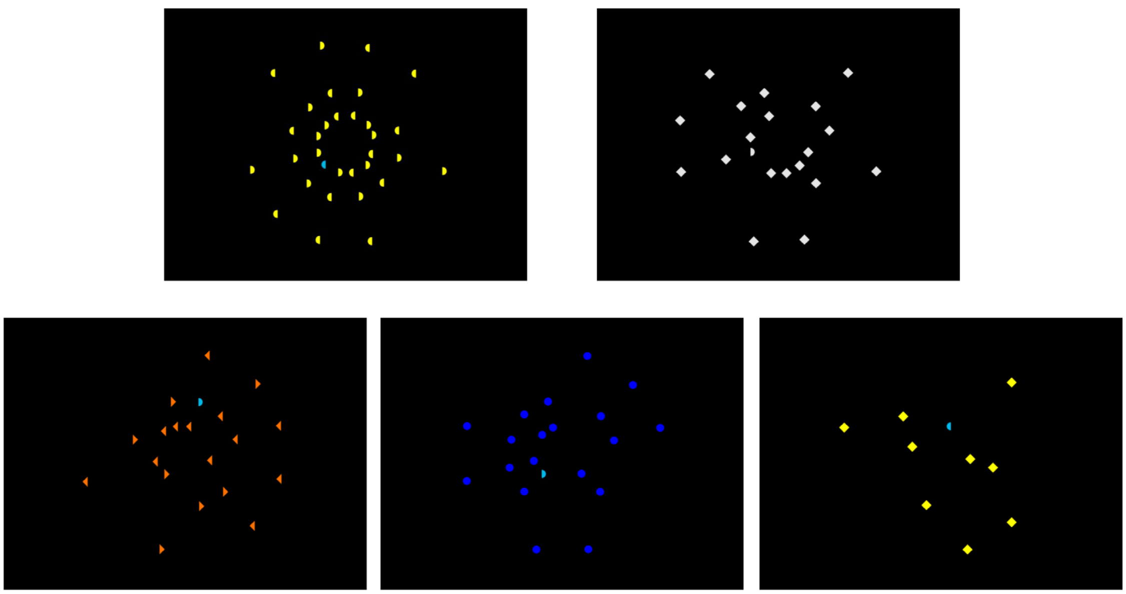
Predicting Visual Search Slopes

Dr Alasdair Clarke, Senior Lecturer in Psychology, University of Essex, UK & Dr Anna Hughes, Lecturer in Psychology, University of Essex, UK

The dataset

In this Research Scenario, we are using a real dataset from the journal article '*Bayesian multi-level modelling for predicting single and double feature visual search*'.

The study involved a visual search task where participants were trying to find a unique target in a scene of distractors (which all looked the same). Here are some examples of stimuli similar to those used in our experiment:



Top left: Here, the target is a blue semicircle within a set of homogeneous (yellow semicircle) distractors. Top right: The target is a grey semicircle in circular grey distractors. Bottom left: The target is a blue semicircle in orange diamond distractors. Bottom middle: The target is a blue semicircle in dark blue triangle distractors. Bottom right: The target is a blue semicircle in yellow circular distractors.

In our paper, we were interested in something called the *search slope*, which is how the difficulty of the search task (as measured by reaction time) changes as a function of the number of distractors present. Many studies have shown that the difficulty increases with the number of distractors, but the steepness of this slope varies depending on the properties of the distractors.

In this scenario, we will walk you through the key analyses that we did, which all asked the question: if we know the search slopes for trials where the distractors are different from the target in one feature dimension (i.e. colour), can we use these slopes to predict how people will behave in a different set of trials, where the distractors are different from the target in two feature dimensions (i.e. colour and shape together).

The analysis we will do is slightly simplified from the one used in the real paper, so the exact numbers you get at the end of this exercise will differ slightly. However, the stages of processing (including data cleaning, plotting, and model building) follow the same logic to those we used in our real research. Working through these activities will therefore help to build the skills you need to work with your own research data.

Loading in the dataset

Our dataset is called `accuracy_rt_data.txt` and it can be found in the folder for this research scenario in the OLC.

ACTIVITY 1: Load in the dataset.

What do we have in our dataset?

One of the first challenges we encounter with a new dataset is understanding what each column means! It's always worth spending a few minutes checking this carefully before going further. In this dataset we have the following:

- *imageFileName*: this column contains the file name for the specific image that was shown to the participant.
- *observer*: a number that identifies each participant in the experiment.
- *block*: this tells us which 'version' of the experiment the participants was doing on this trial. They are labelled things like '1A', '1B' etc. We will work out more about what this means a bit later on.
- *feature*: this tells us which type of distractor was present on this trial. At the moment these are just labelled with numbers (e.g. `1`, `2`, `3`). Again, we will look into making these labels more meaningful later on.
- *n*: this is the trial number. Notice how it resets at the beginning of each new participant. Note also that trial number starts at zero! (This is an interesting quirk of how some programming languages record numbers).
- *distractor_no*: this is the number of distractors that were present (in addition to the target) on this trial.
- *rt*: this is how long it took the participant to find the target (in seconds).
- *accuracy*: this tells us whether the participant found the target correctly (labelled with a `1`) or not (labelled with a `0`).

ACTIVITY 2: Use the `head` function to check you can see all these columns in the data.

ACTIVITY 3: Use the `summary` function to check how many trials each person completed, and how many participants were in the experiment. Did all participants complete all trials?

Data tidying

Before we begin any analysis, we will need to tidy up our data to make it easier to use. This is often the part of analysis that takes the longest! Real data is often messy, and we need to make decisions about how to deal with outliers and how best to label the data to make it as easy as possible to understand.

ACTIVITY 4: We won't need the `imageFileName` column (mentioned above) for the analyses we are going to complete. We can therefore remove it from our dataset at this point. Write some code to do this.

ACTIVITY 5: We mentioned above that the column called `n` contains trial number information, and that these trial numbers start from 0 because this is how many programming languages count. However, for humans, it is more useful if the trial numbers start from 1! Write some code to change the trial numbers to be more 'human-readable'.

ACTIVITY 6: Sometimes it is useful to rename the columns in our data to make them easier to understand. Write some code to rename the `n` column to `trial`, the `distractor_no` column to `nd` and the `accuracy` column to `correct`.

ACTIVITY 7: For some of our analyses, we will want to use the log of the number of distractors. Write some code to create a new variable in our dataset, `log_nd`. Hint: we can't take logs of zero (it's mathematically impossible!) One way of dealing with this is to add one to all the numbers before we take the log.

Checking accuracy

Before we go further, we should check that people could do the task: in this experiment, we expect accuracy rates to be quite high.

ACTIVITY 8: Calculate accuracy per person. How many of our participants had an accuracy above 90%?

ACTIVITY 9: Write some code to remove incorrect trials.

Checking reaction times

We also want to check our reaction time data: we will want to avoid analysing any implausibly short or long reaction times, as again, these are likely to not really be measuring what we are interested in.

ACTIVITY 10: In the original paper, we included participants if *their average response time was not smaller or larger than two standard deviations from the group average response time*. Write some code to work out which (if any) participants should be removed, and remove these participants from the dataset.

ACTIVITY 11: Finally, *for those participants we included, we then removed the top and bottom 1% of their data*. Write some code to carry out this pre-processing step.

Thinking about the experimental design

ND = 0 trials

ACTIVITY 12: Some trials in the dataset have no distractors i.e. `nd = 0`. Create a new data frame `d0` that consists of this subset of the data. What do you notice about the `feature` column for these trials?

ACTIVITY 13: For the analyses we are going to carry out, we need to duplicate this `d0` dataset three times, setting all the values of the `feature` column to 1 in the first duplication (`d01`), and then to 2 and 3 in the next duplications (`d02` and `d03`) respectively. This is because we will need to use these 'no distractor' trials as a baseline for all the different `feature` conditions. Write some code to create these duplications, and then create a final full dataset of these three baseline duplications plus all the other trials where `nd > 0`.

What's going on with the block and feature columns?

The next processing step needs us to understand more about the `block` and `feature` columns in our data.

In this experiment, *block 1a* involved participants searching for a red semicircle target among orange, pink or purple semicircular distractors i.e. they searched for a target that differed from the distractors by a *single feature* (in this case colour). *Block 1b* involved participants searching for a semicircular target within triangle, circle or diamond distractors - this was therefore another *single feature* block, but this time using shape.

In *blocks 2a, 2b and 2c* participants once again searched for a red semicircle, but this time the distractors differed in both shape and colour. These were called *double feature* conditions.

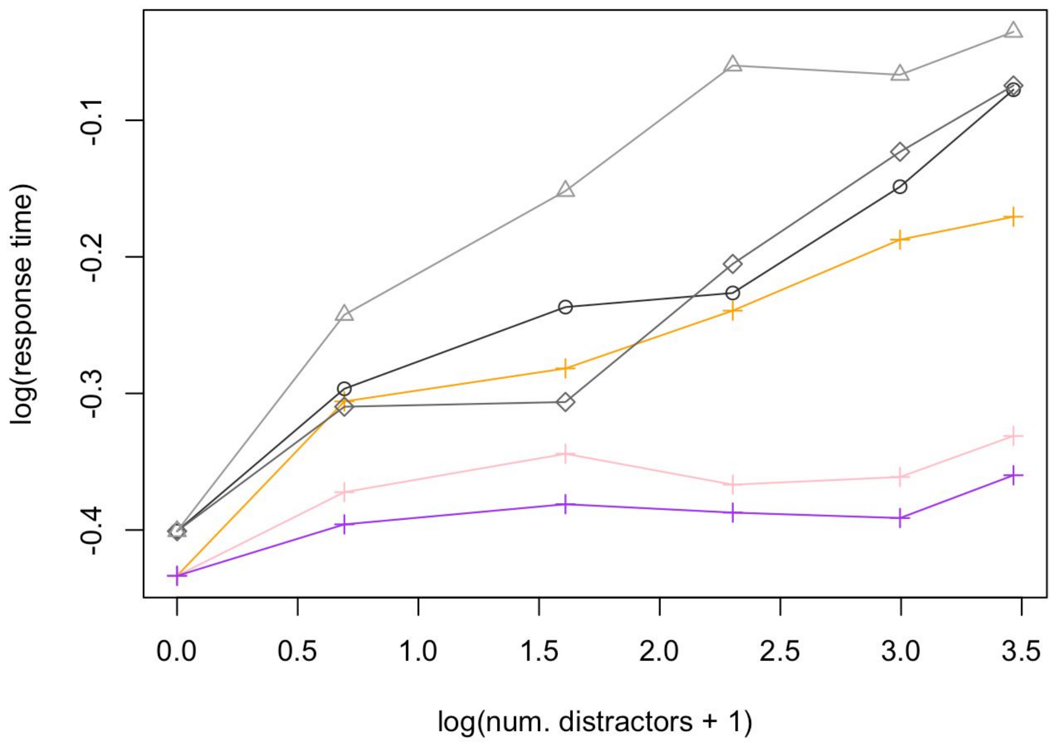
The table below shows what the different levels of `feature` mean in the different `blocks`:

Block	Feature 1	Feature 2	Feature 3
1a	orange	pink	purple
1b	circle	diamond	triangle
2a	pink circle	orange diamond	purple triangle
2b	orange circle	purple diamond	pink triangle
2c	purple circle	pink diamond	orange triangle

ACTIVITY 14: Create a new `feature` column made by combining the `block` and `feature` columns and re-label the data so that instead of the feature levels being 1, 2, or 3 they are instead the appropriate distractor types for the block.

Plotting Experiment 1

It is a good idea to visualise our data before we begin statistical analysis. We will start by making the following graph, to understand how reaction time varies with the number of distractors for the different types of distractor conditions in Experiment 1:



We can see that, as expected, participants' reaction times increase with an increasing number of distractors. However, the rate of increase does vary depending on the distractor type: for example, orange distractors lead to a higher slope compared to the pink and purple distractors, and generally the distractors in Experiment 1b (where the distractors differed in shape only) lead to higher slopes compared to the distractors in Experiment 1a (where the distractors differed in colour only).

ACTIVITY 15: Subset the data to create a new data frame, `d1`, which contains only Experiment 1 blocks.

ACTIVITY 16: Use the `aggregate` function to calculate the average log reaction time per person for each combination of distractor type and number, and create a new data frame `d1agg` to store these averages.

ACTIVITY 17: Now use `d1agg` to find the overall mean log reaction time for each combination of distractor type and number, and create a new data frame `d1agg2` to store these averages.

ACTIVITY 18: Use `d1agg2` to plot your graph of log number of distractors against average log reaction time. Try to make the colours and shapes of the lines/points match those of the distractors presented in that condition!

Modelling Experiment 1

We can now finally start building our statistical models for Experiment 1! We want to ask: how is log reaction time in this experiment affected by the log number of distractors, the type of distractors, and their interaction?

ACTIVITY 19: Write out the model formula (in linear model syntax and using the variable names in your dataset `d1`) that would answer this question.

ACTIVITY 20: This experiment is a repeated measures design, with each participant completing multiple trials. How could we account for this in an `lme4` framework? Write out the random effects structure that would allow each participant in the data to have their own random intercept. (Note: it would be better to allow for random slopes as well, but this is difficult in a frequentist framework. In the journal article, we used Bayesian methods to be able to achieve this).

ACTIVITY 21: Put the full model structure together and run the model, as well as the model summary.

ACTIVITY 22: In the model summary, the 'baseline' category is indicated by the 'Intercept' row. However, to help us interpret our results, it would be helpful for us to remove this intercept and instead just have one slope for each distractor type and their interaction with the log number of distractors. We can do this by adding a 0 to the right hand side of our model equation e.g. `dv ~ 0 + iv`. We will also need to adjust the model formula to include just the main effect of `feature` and the interaction of `feature` with `log_nd`. Rewrite your model in this way, run this new model, and check how it affects the output.

ACTIVITY 23: We are going to use the slopes from these models to allow us to make predictions about what we should expect to see in Experiment 2. We therefore need to extract these slopes (i.e. the `feature:log_nd` coefficients of the model) and save them to a new data frame (`slopes1`). Write some code to complete this task.

Modelling Experiment 2

The goal of the next section is to use the predicted values of the log slopes in the single feature conditions (which we calculated in the previous section) to predict the values of the log slopes in the double feature conditions i.e. Experiment 2.

First, we will fit a similar model to the one we fit in the previous section on the Experiment 2 data so that we know what the *true* log slope values are.

ACTIVITY 24: Subset the data to create a new data frame, `d2`, which contains only Experiment 2 blocks.

ACTIVITY 25: Fit the identical model to the final model we fit on Experiment 1, but using the Experiment 2 data (i.e. with the zero intercept and the random effects for participant).

ACTIVITY 26: As for Experiment 1, we now need to extract these slopes (i.e. the `feature:log_nd` coefficients of the model) and save them to a new data frame (`slopes2`). Write some code to complete this task.

ACTIVITY 27: translate this equation into code in order to add a new column to your `slopes2` dataset called `orthog_contrast`, which is the predicted slope for the double feature distractor type based on the orthogonal contrast model.

ACTIVITY 28: Finally, we want to plot a correlation between `orthog_contrast`, our predicted slopes in Experiment 2, and `b`, the real, experimentally observed slopes in Experiment 2. Compute the best fit line and add it to the graph. Is this correlation significant? What do these results tell us about how well our model is able to predict behaviour?

Concluding remarks

The modelling we have carried out shows that we do seem to be able to predict behaviour on the double feature visual search task based on behaviour on the single feature search task.

However, it is always good to think about possible limitations of your findings. For example, our correlation here is based on a relatively small sample of data points: perhaps it would be more convincing if we were able to increase this by running more experiments with different combinations of different colours and shapes. In addition, we might not just want to consider the correlation, but whether we can really predict *exactly* what the slopes should be i.e. is the slope of our best fit line close to 1?