Coding predictors for regression analysis

An applied tutorial on how to prepare predictors for LMs, GLMs, and GLMMs, how to interpret results depending on these steps, and how to report results

T. Florian Jaeger

September 23, 2020

Contents

1 Reading and assignments in *preparation* of this class **2**

2 Quick recap: The linear model (LM) **2**
2.1 Assumptions of the LM . . . 2
2.2 Using an LM (fit) for your analysis . . . 3
2.3 Writing up results . . . 3

3 The data for this document **3**
3.1 Background . . . 4
3.2 Goals . . . 4
3.3 Methods . . . 4
3.4 Overview . . . 4

4 Coding categorical (and other) predictors **5**

5 Coding a binary categorical predictor **5**
5.1 Treatment/dummy-coding . . . 7
5.1.1 Prepare for class . . . 9
5.1.2 Questions for discussion in class . . . 9
5.2 Effect/deviation/sum/anova-coding . . . 9
5.2.1 Prepare for class . . . 11
5.2.2 Discussion questions for class . . . 11
5.3 Writing up results . . . 11

6 Combining factors and continuous predictors **12**
6.1 Do both the diffusion constant and the crowdedness condition affect threshold performance? . . . 12
6.1.1 Write-up . . . 12
6.1.2 Prepare for class . . . 13
6.1.3 Discussion questions for class . . . 13

7 Intermezzo: centering continuous predictors **14**

8 Interactions between factors and continuous predictors **15**
8.1 Do the effects of condition and diffucusion constant interact? . . . 16
8.1.1 Prepare for class . . . 17
8.1.2 Discussion questions for class . . . 17
8.2 Simple effects . . . 18
8.3 Writing up the analysis results . . . 20

9 Additional topics we can discuss in class **20**
9.1 Coding of continuous predictors . . . 20

9.2 Coding of factors with more than two levels 20

10 Session info **20**

1 Reading and assignments in *preparation* of this class

Please make sure you have read *all* of Gelman & Hill (2007, Ch 3 up to and incl. 4.5) as well as James et al. (2013, Ch 3.3 up to but *not* incl. 3.3.3). Then read and *work through* this document. We will use class to go through the important concepts and to address any questions that you have about the readings or the problem sets in this document. **Please note that this document is providing R code only.** Some of the steps described here might have less direct solutions in Matlab, so it is recommended that you start early. I have, however, tried to describe each step in a way that does not depend on R or Matlab.

In this document, you will find sections labeled "Prepare for class". Please work through those examples and write up your answers. If there are a few questions, you cannot answer, elicit help on the slack channel. In addition, there are sections called "Discussion questions for class". Please think about these questions, but don't worry if you get stuck. These are some questions we can go through during class.

2 Quick recap: The linear model (LM)

A linear model (or linear regression) describes an outcome y as the weighted sum of predictors x_1, ..., x_k, plus some error ϵ that is assumed to be normally distributed. This can be written in a variety of ways, e.g., for the specific ith of 1 to n outcomes, y_i, or for the entire n-element vector of outcomes $\mathbf{y}$ with the elements y_1, ..., y_n, or in terms of the expected value of $E(\mathbf{y})$:

$$y_i = \beta_0 + \beta_1 x_{1,i} + ... + \beta_k x_{k,i} + \epsilon_i,\ \epsilon_i \sim \text{Normal}(0, \sigma_{resid}) \quad \Leftrightarrow \quad (1)$$

$$\mathbf{y} = \beta_0 + \beta_1 \mathbf{x}_1 + ... + \beta_k \mathbf{x}_k + \epsilon,\ \epsilon_i \sim \text{Normal}(0, \sigma_{resid}) \quad \Leftrightarrow \text{ [cf. James et al., 2013, p. 63]} \quad (2)$$

$$\mathbf{y} = X\beta + \epsilon,\ \epsilon_i \sim \text{Normal}(0, \sigma_{resid}) \quad \Leftrightarrow \quad (3)$$

$$E(\mathbf{y}) = X\beta \quad \Leftrightarrow \quad (4)$$

$$\mathbf{y} = \text{Normal}(X\beta, \sigma_{resid}) \quad \text{[cf. Gelman \& Hill, 2007, p. 38]} \quad (5)$$

where X is a matrix with $k + 1$ columns and n rows (the first column of which is a vector of n 1s). X is also sometimes called the *model (or design) matrix*.

2.1 Assumptions of the LM

As a theoretical model the LM makes several assumptions (cf. Gelman & Hill, 2007, p. 45). The first two assumptions are shared with many/most other statistical models we use (to be precise, some models don't strictly make these assumptions, but they do correct for their violations):

- Independence of observations/errors
- Validity/exhaustivity

The other assumptions are more specific to the LM, though the last two of these are shared with GLMs and GLMMs (but not GAMMs, for example):

- Normality of errors
- Equal variance of errors
- Additivity of effects
- Linearity of each effects

Many, if not most, of these assumptions are actually violated when we apply the model. To some extent linear regression is robust to such violations, depending on the specific assumption. For example, instead of equality of variance it is often sufficient that variances are similar (homogeneity of variance) or that the variance-covariance matrix does not exhibit heteroscedasticity (from Ancient Greek hetero "different" and skedasis "dispersion", https://en.wikipedia.org/wiki/Heteroscedasticity; this extends the notion of homogeneity of variance to the *co*variance between variables). **But violations of the independence, normality, or homoscedasticity assumptions can make a model invalid, and the statistical inferences based on it invalid.**

2.2 Using an LM (fit) for your analysis

The linear model is a theoretical model. If we employ it during data analysis, we are making the assumption that this model describes the relation between the predictors and the outcome in the population we seek to study. This also means that all of our conclusions are contingent on these assumptions, including in particular the assumption that we have considered all relevant variables (or have otherwise ruled out that other effects can confound our analysis).

The coefficients of the LM (the βs) are parameters of this theoretical model (the coefficients), and we do not know their true value. Neither do we know the true values of the outcome, but we assume that our observations of the outcome variable are (potentially noisy) independent samples drawn from the outcome. Additionally, we often don't really know the predictors we believe to have causal effects on the outcome, but rather we have observable *measures* of these predictors. The linear model does *not* take this into account (but there are models that extend the LM that do).

When we use an LM to analyze our data, we fit a *specific* LM—i.e., an LM with a specific set of predictors to our data (the specific outcome). We specify this model using regression formula syntax. For example in R, the following would describe the formula for an LM that regresses an outcome variable Threshold against the intercept (1) and a predictor called Condition:

$$Threshold \sim 1 + Condition$$

The algorithm implemented in whatever function we use for that (e.g., *fitlm* in Matlab or *lm* in R) then determines the best-fitting estimates of the *coefficients*. In other words, we tell the LM-fitting function the outcome and predictor—the knowns—and the function then determines the coefficients—the unknowns. The resulting combination of the specific LM (the predictors and outcome variables) and the best-fitting coefficient estimates together constitute the **fit** or **fitted (linear) model**. The statistical inferences we draw, and write-up, are always based on such best-fitting models. Here we won't go into detail about how these "best-fitting" estimates and their standard errors are obtained, but we will return to that later in the semester.

2.3 Writing up results

We should **use $\hat{\beta}$ when we describe coefficient estimates and write up results**. This emphasizes that these are *estimates* of the assumed population parameters (β). It is worth noting that the $\hat{\beta}$s are not the only outputs we obtain from a LM fit. This fit also contains, for example, predicted values for each outcome based on the predictors for that observation. But the null hypothesis significance testing (NHST) that we conduct focuses on the best-fitting coefficient estimates and their standard errors. When we write that a predictor did or did not have a statistically significant effect, or when we talk about the size of the effect, those statements refer to the coefficient estimate, its standard error, and measures derived from them.

3 The data for this document

For this tutorial we are continuing to use Ashley Clark's data from her experiment on visual crowding effects on foveal processing. Here's a copy of her description of the data. One difference to the data from her study that you've seen so far is that we're including a third condition that Clark and colleagues collected as a control.

3.1 Background

Crowding is a visual phenomenon that has puzzled scientists for decades; an object in isolation can be perceived without problems, yet surrounding it with similar objects makes it harder to see. While crowding has been studied extensively in the visual periphery, humans normally orient objects in their center of gaze, the high-acuity region of the retina called the foveola. While the foveola is less than 1.5mm wide (or ~1 visual deg2), it contains more cones than rest of the retina combined. Human's ability to actively perceive the visual world relies heavily on not only the foveola itself, but also how and where the eye is positioned. The few studies that have examined crowding within foveal vision have produced contradictory results. Some reasons for these discrepancies include using relatively large stimuli, a small number of participants, abnormal stimuli presentation, and having indefinite stimulus presentation times with no eye tracking. More recent research, however, has highlighted the importance of precise eye tracking due to the eye's constant movement during even fixation. These small and constant movements of the eye, called fixational eye movements (FEMs), are beneficial or both high acuity vision, as well as daily tasks such as reading and face recognition. FEM's have never been examined in the context of crowding, and it remains unknown how individuals in previous crowding studies directed their foveola over stimuli, or even maintained fixation.

3.2 Goals

The goals of this research are to (1) investigate the effect of crowding within the foveola, and (2) examine if and how fixational eye movements influence crowding at this scale. Based on previous research, we hypothesize that crowding will be detrimental to foveal vision, as it is in peripheral vision, but on a finer scale. Further, based on the recent findings that FEMs are beneficial for high-acuity vision, I expect a relationship between FEMs and the strength of crowding within the foveola, with larger and less precise FEMs increasing the negative effects visual crowding.

3.3 Methods

Studying FEMs during visual crowding within the foveola requires high-precision eye tracking and accuracy in localizing the center of gaze. Current video eye trackers do not have the required spatial precision, as the error of gaze localization is as large as the foveola itself. However, by using a custom built state-of-the-art eye tracking system with arcminute precision, we will be able to examine exactly how FEMs contribute to crowding within the foveola. Stimuli consist of a number-font designed specifically for studying crowding in the fovea, as it allows for recognition even when numbers are closer together than traditional optotypes used in other crowding studies.6 Two conditions will be examined, the uncrowded (where a single number is presented), and the crowded (where the same size number is presented, but with four surrounding numbers). The size of the number and spacing between the numbers in the crowded condition change throughout the experiment based on the subject's performance using an adaptive procedure. The stimuli presented will vary in size, ranging from 0.5 arcminutes to 4 arcminutes in width. To determine the number-width threshold, we use a standard psychophysics procedure measuring the width of the stimulus at which a subject performs above chance level.

3.4 Overview

```
# A tibble: 20 x 8
   Subject Condition Threshold DiffusionConstant  Span  Area Curvature Speed
   <fct>   <fct>         <dbl>             <dbl> <dbl> <dbl>     <dbl> <dbl>
 1 1       Uncrowded      1.39              11.1  3.13 137.      12.0   40.6
 2 2       Uncrowded      1.19               4.89 2.13  57.2     15.3   38.0
 3 3       Uncrowded      1.62              18.5  4.19 127.      12.7   37.6
 4 4       Uncrowded      1.43              15.6  3.92 105.       9.82  45.7
 5 5       Uncrowded      1.27               7.07 2.52  94.9     14.3   37.1
 6 6       Uncrowded      1.80              23.5  3.79 131.      10.6   45.9
 7 7       Uncrowded      1.62              10.8  3.09 113.      13.0   45.6
 8 8       Uncrowded      1.51              18.5  3.89 101.       9.51  47.4
 9 9       Uncrowded      1.75              14.5  3.21 132.      11.3   48.7
10 10      Uncrowded      1.67              14.2  4.27 281.      10.3   46.3
11 1       Crowded        2.11              15.6  3.61 137.      11.1   43.6
12 2       Crowded        1.74               5.64 2.33  44.0     13.4   43.8
```

```
13 3       Crowded        3.17            39.4   6.27 177.      10.1   44.9
14 4       Crowded        1.92            14.4   3.69 104.      10.8   48.6
15 5       Crowded        1.52             6.25  2.31  67.2     15.8   34.8
16 6       Crowded        2.41            30.6   4.50 140.       9.68  46.1
17 7       Crowded        2.05            25.1   3.76 104.      12.3   44.4
18 8       Crowded        1.91            15.0   3.57  87.6     10.5   47.7
19 9       Crowded        2.31            11.3   3.05 127.      11.4   54.1
20 10      Crowded        1.89            17.3   4.38 141.       9.97  54.6
```

4 Coding categorical (and other) predictors

If we want to include categorical predictors (e.g., "Condition A" vs. "Condition B") in an LM, we thus need to translate these predictors into numerical variables. This process is sometimes called predictor *coding*. Some people use this term narrowly to only refer to the coding of categorical predictors (or, as they are sometimes called, *factors*; hence the name factor coding). Others use it to refer to any changes or transformation we make to our predictors, including continuous predictors (e.g. centering or scaling). In this tutorial, we talk about both.

If you have been using analysis of variance (ANOVA), coding is a process that you might already be familiar with: in ANOVA-based result reporting, we regularly follow the initial ANOVA significance tests with so called "planned" or "post-hoc" comparisons. These comparisons assess hypotheses about the means of the different experimental conditions.

You might have heard terms like (forward/backward) Helmert coding, (forward/backward) sliding difference coding, polynomial coding, sum/deviation/ANOVA coding, or treatment/dummy coding, etc. These are all names for different ways to code categorical factors, and they do exactly the same job in an LM as they do for the planned or unplanned post-hoc analyses after an ANOVA. In fact, the post-hoc tests reported for ANOVA-based approaches are typically run as an LM, but that's another topic. (Note that these post-hoc tests are the only way to obtain information about the *direction* of an effect in the ANOVA approach, since the ANOVA itself only assesses the significance of a predictor).

In an LM, we need to make these coding decision *prior* to fitting the model. **Most statistics programs have default coding choices built-in that they apply to any LM or other regression you fit. Be aware of these defaults, as they will affect what the output of an LM means if you forgot to explicitly define the factor coding for the LM.** In R, for example, all factors for which we have not explicitly defined coding will be treatment coded, with the reference level set to the first level of the factor (we'll get to what all of this means). And, if you haven't defined how the levels of the factor order, R will assume that they order alphanumerically.

Such defaults can be handy in that they mean you can just run a model like:

$$Threshold \sim 1 + Condition$$

where Threshold is the outcome, $\sim$ refers to "regress against", 1 refers to the intercept (which by default would be included anyway, but I'm being explicit here), and Condition is a factor with two levels "crowded" and "uncrowded". But defaults can also wreak havoc. In this example, R would automatically code "crowded" as the reference level (0), because it comes alphanumerically first, and "non-crowded" as the treatment (1). The coefficient for Condition would thus—perhaps counter-intuitively—reflect the increase in the threshold in the *un*crowded condition relative to the crowded condition. In short:

- **don't rely on default**;
- **code your factors explicitly**;
- **use variable names and level names that are transparent and avoid confusion**

5 Coding a binary categorical predictor

Let's start with the simplest case: coding a binary (two-level) factor. This will allow us to build intuitions both about *how* we can code variables, and *what the consequences of those codes are*. We'll stick with the example from above.

Factor coding essentially translates the categorical predictor into a numerical predictor. This is achieved through so called *contrasts*. Specifically, for a factor with k distinct levels we can use $k-1$ contrasts to create $k-1$ numerical predictors. These numerical predictors capture all the information of the k levels of the original factor. Each contrast is a vector with k numeric elements, describing the mapping from factor levels to numerical values.

For a binary factor, we thus have one contrast that is a vector with two values. This contrast defines one numerical predictor. E.g., by default in R, we'd get the following for Ashley's data set on visual crowding:

```
          Uncrowded
Crowded           0
Uncrowded         1
```

And, each time a regression function is called, this allows R to *implicitly* create a new column in our data that is the numerical translation of the Condition variable. Thus a model like

$$Threshold \sim 1 + Condition$$

is actually running the model

$$Threshold \sim 1 + ConditionUncrowded$$

where $ConditionUncrowded$ is a numerical variable that has been silently added to our data:

```
# A tibble: 20 x 9
   Subject Condition Threshold DiffusionConstant  Span  Area Curvature Speed ConditionUncrowded
   <fct>   <fct>         <dbl>             <dbl> <dbl> <dbl>     <dbl> <dbl>               <dbl>
 1 1       Uncrowded      1.39              11.1  3.13 137.      12.0   40.6                   1
 2 2       Uncrowded      1.19               4.89 2.13  57.2     15.3   38.0                   1
 3 3       Uncrowded      1.62              18.5  4.19 127.      12.7   37.6                   1
 4 4       Uncrowded      1.43              15.6  3.92 105.       9.82  45.7                   1
 5 5       Uncrowded      1.27               7.07 2.52  94.9     14.3   37.1                   1
 6 6       Uncrowded      1.80              23.5  3.79 131.      10.6   45.9                   1
 7 7       Uncrowded      1.62              10.8  3.09 113.      13.0   45.6                   1
 8 8       Uncrowded      1.51              18.5  3.89 101.       9.51  47.4                   1
 9 9       Uncrowded      1.75              14.5  3.21 132.      11.3   48.7                   1
10 10      Uncrowded      1.67              14.2  4.27 281.      10.3   46.3                   1
11 1       Crowded        2.11              15.6  3.61 137.      11.1   43.6                   0
12 2       Crowded        1.74               5.64 2.33  44.0     13.4   43.8                   0
13 3       Crowded        3.17              39.4  6.27 177.      10.1   44.9                   0
14 4       Crowded        1.92              14.4  3.69 104.      10.8   48.6                   0
15 5       Crowded        1.52               6.25 2.31  67.2     15.8   34.8                   0
16 6       Crowded        2.41              30.6  4.50 140.       9.68  46.1                   0
17 7       Crowded        2.05              25.1  3.76 104.      12.3   44.4                   0
18 8       Crowded        1.91              15.0  3.57  87.6     10.5   47.7                   0
19 9       Crowded        2.31              11.3  3.05 127.      11.4   54.1                   0
20 10      Crowded        1.89              17.3  4.38 141.       9.97  54.6                   0
```

Remember the *model matrix* X from above? In R (and probably Matlab), specifying an LM formula implicitly creates a model matrix X from the data, and the function lm is using that model matrix (and the column of outcomes, $\mathbf{y}$) to fit the linear model. Looking at the model matrix can be a powerful tool in understanding factor coding. For example, prior to specifying any factor coding, the model matrix for the above model would be:

```
model.matrix( ~ 1 + Condition, data = d)
```

```
   (Intercept) ConditionUncrowded
1            1                  1
2            1                  1
3            1                  1
4            1                  1
5            1                  1
6            1                  1
7            1                  1
8            1                  1
9            1                  1
10           1                  1
11           1                  0
12           1                  0
13           1                  0
```

```
14           1                  0
15           1                  0
16           1                  0
17           1                  0
18           1                  0
19           1                  0
20           1                  0
attr(,"assign")
[1] 0 1
attr(,"contrasts")
attr(,"contrasts")$Condition
[1] "contr.treatment"
```

Notice the implicitly created column of 1s for the intercept 'predictor' (i.e., x_0 is the constant 1). We also see that the first 10 rows of the data are coded as 1 for Condition and the second 10 rows as 0. That's because the first 10 rows are from the uncrowded condition (see above). Below, I'll show the model matrix for each of the coding examples. This should also help you translate the examples into another statistic program: if push comes to shove, you can always manually create the variables shown in the model matrix and then hand that manually created data to the model-fitting function of your program.

5.1 Treatment/dummy-coding

The default in many statistics programs is treatment-coding, like in the example we just went through. Under this coding scheme, we contrast all other condition against a baseline (or "reference") condition. Since the default is in this case somewhat counter-intuitive—it's more intuitive to think of visual *crowding* as the treatment—we set define our own treatment coding:

```
          Crowded
Crowded         1
Uncrowded       0
```

With this change in contrast, the model matrix also changes. Now the first 10 rows of the data are coded as 0 for the condition predictor, and the second rows are coded as 1.

```
model.matrix( ~ 1 + Condition, data = d)
```

```
   (Intercept) ConditionCrowded
1            1                 0
2            1                 0
3            1                 0
4            1                 0
5            1                 0
6            1                 0
7            1                 0
8            1                 0
9            1                 0
10           1                 0
11           1                 1
12           1                 1
13           1                 1
14           1                 1
15           1                 1
16           1                 1
17           1                 1
18           1                 1
19           1                 1
20           1                 1
attr(,"assign")
[1] 0 1
attr(,"contrasts")
attr(,"contrasts")$Condition
          Crowded
Crowded         1
Uncrowded       0
```

Now we can fit the LM and summarize that *fit*:

```
Call:
lm(formula = Threshold ~ 1 + Condition, data = d)

Residuals:
     Min       1Q   Median       3Q      Max
-0.58347 -0.19997 -0.03302  0.16157  1.06575

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)        1.5252     0.1115  13.676 5.99e-11 ***
ConditionCrowded   0.5767     0.1577   3.657   0.0018 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3527 on 18 degrees of freedom
Multiple R-squared:  0.4262,    Adjusted R-squared:  0.3943
F-statistic: 13.37 on 1 and 18 DF,  p-value: 0.001805
```

In this treatment-coded model:

- the intercept estimate give us the model's prediction for the mean outcome *of the reference level* of Condition (because the intercept *always* gives us the model's prediction when all other terms of the model are 0—e.g., when all other predictors are 0). In this case, this is the mean threshold for the uncrowded condition.
- the ConditionCrowded estimate tells us how much larger (or smaller) the threshold is for the treatment condition. In this case, this is the crowded condition.
- the sum of the intercept and the estimate for the coefficient for ConditionCrowded give us the model's prediction for the mean outcome of the treatment level.

We can convince ourselves that the above interpretation is correct by looking at the predictions of the model for each observation, and comparing them to the estimates for the intercept and the coefficient for ConditionCrowded. First, let's have a look at the model's predictions, and attach. It's just a vector with as many elements as there are outcomes in the data we fit with the linear regression.

```
       1        2        3        4        5        6        7        8        9       10       11       12       13       14
1.525174 1.525174 1.525174 1.525174 1.525174 1.525174 1.525174 1.525174 1.525174 1.525174 2.101896 2.101896 2.101896 2.101896 2.101
```

Note that the model only predicts two different values—one for the crowded condition and one for the uncrowded condition. This becomes apparent when we attach the fitted/predicted values to the data.frame. Here I'm only showing the Subject, Condition, Threshold, and fitted values:

```
# A tibble: 20 x 4
   Subject Condition Threshold fitted
   <fct>   <fct>         <dbl>  <dbl>
 1 1       Uncrowded      1.39   1.53
 2 2       Uncrowded      1.19   1.53
 3 3       Uncrowded      1.62   1.53
 4 4       Uncrowded      1.43   1.53
 5 5       Uncrowded      1.27   1.53
 6 6       Uncrowded      1.80   1.53
 7 7       Uncrowded      1.62   1.53
 8 8       Uncrowded      1.51   1.53
 9 9       Uncrowded      1.75   1.53
10 10      Uncrowded      1.67   1.53
11 1       Crowded        2.11   2.10
12 2       Crowded        1.74   2.10
13 3       Crowded        3.17   2.10
14 4       Crowded        1.92   2.10
15 5       Crowded        1.52   2.10
16 6       Crowded        2.41   2.10
17 7       Crowded        2.05   2.10
18 8       Crowded        1.91   2.10
19 9       Crowded        2.31   2.10
20 10      Crowded        1.89   2.10
```

For the uncrowded condition, the fitted value is 1.525 … the exact same as the intercept estimate. For the crowded condition, the fitted value is 2.101, which is 1.525 (the intercept estimate) + .576 (the estimate for the coefficient of ConditionCrowded). In other words, for all conditions and for all data points, the predicted/fitted value of the LM $\widehat{y_i} = intercept + slope * Condition_i = \beta_0 + \beta_{condition} x_{condition}$, where $x_{condition}$ is 1 for the crowded condition

and 0 for the uncrowded condition. Noice!

The following figure shows the data we entered into the LM, and the predictions of the LM.

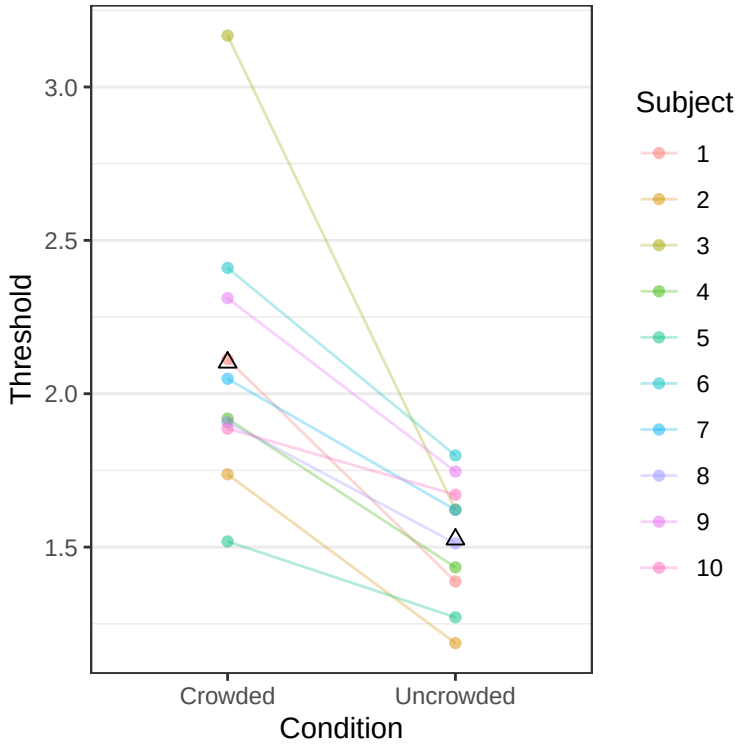

Figure 1: Size (in arcminutes) at which threshold performance (62.5% correct) was reached by crowding condition. Each color shows a separate subject. Black triangles show the predicted outcome of a linear model fit to the data.

5.1.1 Prepare for class

1. Why does the model predict only two different values for the outcome variable?
2. How would you calculate the epsilon/residual for each observation from this data? (Try it)
3. How could you calculate the residual sum of squares for this model from this data? (Try it)
4. We obtain the straightforward interpretation of the ConditionCrowded estimate only because we coded the treatment condition as 1, rather than, e.g., as 2. What would happen if we coded the treatment condition as 2 (and the reference condition still as 0)? Would our estimate for the coefficient for ConditionCrowded change? What about the estimate of the intercept? Why? (Much of this can be answered by trying to change the coding)
5. Would the predicted/fitted values of the model change if we change the coding to 2 vs. 0? Why?

5.1.2 Questions for discussion in class

6. How could you calculate the R^2 value shown as part of the model output from just the predicted/fitted values and the original data? (cf. James et al., 2013, p. 70-71)

5.2 Effect/deviation/sum/anova-coding

While treatment-coding is the default in the regression world, it is actually rarely used in experimental psychology, brain imaging, or related fields that use factorial/balanced designs. Rather, we typically code our data in a different way, because it changes the interpretation of the intercept in ways that researchers that are used to ANOVA tend to find more intuitive. This alternative is called effect/deviation/sum/anova-coding:

```
          Crowded.vs.Uncrowded
Crowded                    0.5
Uncrowded                 -0.5
```

With this change in contrast, the model matrix also changes:

```
model.matrix( ~ 1 + Condition, data = d)
```

```
   (Intercept) ConditionCrowded.vs.Uncrowded
1            1                         -0.5
2            1                         -0.5
3            1                         -0.5
4            1                         -0.5
5            1                         -0.5
6            1                         -0.5
7            1                         -0.5
8            1                         -0.5
9            1                         -0.5
10           1                         -0.5
11           1                          0.5
12           1                          0.5
13           1                          0.5
14           1                          0.5
15           1                          0.5
16           1                          0.5
17           1                          0.5
18           1                          0.5
19           1                          0.5
20           1                          0.5
attr(,"assign")
[1] 0 1
attr(,"contrasts")
attr(,"contrasts")$Condition
          Crowded.vs.Uncrowded
Crowded                    0.5
Uncrowded                 -0.5
```

Let's refit the model with this newly coded factor:

```

Call:
lm(formula = Threshold ~ 1 + Condition, data = d)

Residuals:
     Min       1Q   Median       3Q      Max
-0.58347 -0.19997 -0.03302  0.16157  1.06575

Coefficients:
                            Estimate Std. Error t value Pr(>|t|)
(Intercept)                  1.81354    0.07886  22.997 8.54e-15 ***
ConditionCrowded.vs.Uncrowded  0.57672    0.15772   3.657   0.0018 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3527 on 18 degrees of freedom
Multiple R-squared:  0.4262,    Adjusted R-squared:  0.3943
F-statistic: 13.37 on 1 and 18 DF,  p-value: 0.001805
```

In this deviation-coded model:

- the intercept estimate is now the predicted *overall mean* of the outcome.
- the estimate for the coefficient of Condition is the predicted difference between the two conditions.
- the two conditions means are described as the sum of the intercept estimate +/- .5-times the estimate for the coefficient of Condition.

Why do researchers used to ANOVA find this more intuitive? You'll sometimes here that it's nice that the intercept now corresponds to the mean, but the true convenience of this coding scheme will become apparent once we consider interactions below. It is under deviation-coding, that we can talk about *main effects* and *interactions* in the same sense as in an ANOVA. That's presumably also why some people call this coding scheme anova-coding. We'll get to that. But first some questions.

5.2.1 Prepare for class

1. What happens if we double the values we assign to each of the two deviation-coded conditions, i.e., if we use 1 vs. -1 instead of .5 vs. -.5? What if we set them to -1000 vs. 1000? What changes and what doesn't? Why? (Try it)

```
contrasts(d$Condition) = cbind("Crowded vs. Uncrowded" = c(1,-1))
```

2. Are the predicted/fitted values of any of these deviation-coded models different from each other? (Try it)
3. Are the predicted/fitted values of any of these deviation-coded models different from the treatment-coded model presented in the previous section? Why?

5.2.2 Discussion questions for class

4. We obtain the intuitive interpretation of the intercept (as the predicted grand mean of the data) only because the data is … what? (recall that the intercept is *always* the predicted value for the case when all other terms add up to zero). Conveniently, most data sets obtained from psychological experiments have the same property we are looking for here—either exactly or at least approximately (after exclusions).

5.3 Writing up results

How would we write up, for example, the deviation-coded model. There are, of course, about as many difference preferences as they are researchers, and so you will find conflicting advice. But generally, it will be helpful if you are clear about all of the following:

- What model you used
- What predictors you considered (incl. those that you did not include in the final model, cf. debate about *researchers' degrees of freedom*)
- How you coded the outcome (e.g., what unit is the outcome in; without this information, readers cannot determine the size of effects or interpret them on the original scale)
- How you coded predictors (without this information, readers cannot even determine in which *direction* the effect is!)
- What steps (if any) were taken to ascertain the validity of the model (see, e.g., model evaluation in Gelman & Hill, 2007, Section 3.7). This is arguably less important if you study a well-known phenomenon that has been analyzed with this method many times, and for which you have data that is a) balanced with regard to the predictors in the model, and b) has many observations relative to the number of predictors in the model.

When reporting results, I recommend you provide both the relevant statistics (coefficient estimates, p-values, etc.), and a descriptive interpretation of that result in non-technical language—but we careful to avoid language that is wrong (see, e.g., section on interactions for some examples). Finally, for any non-trivial model, I highly recommend the use of a summary table of the model and visualization of the *empirical* distribution of the data (potentially, while *also* showing the model's predictions). The latter helps readers not familiar with the analysis approach to understand your results.

Here is a rather detailed write-up for the model with the deviation-coded condition variable. In many scenarios (e.g., if the units of the outcome variable are not necessarily informative), you might decide to provide less detail:

We analyzed the data with a linear regression, using the function `lm` from the `base` package (citation with version) of the software `R` (citation with version). We calculated each subject's mean thresholds for the crowded and uncrowded condition. These 20 threshold values were regressed against condition (deviation-coded: .5 = *crowded* vs. -.5 = *uncrowded*). We found a statistically significant main effect of condition ($\hat{\beta} = .576, t = 3.657, p < .01$), so that subjects reached threshold performance at a size that was about half an arcminute larger in the crowded condition (mean = 2.102), compared to the uncrowded condition (mean = 1.525).

6 Combining factors and continuous predictors

Now that we know how to code factors (or at least binary factor), we can combine continuous and categorical predictors in our model. We first show a simple 'additive' model, in which we assume that the effects of the categorical and continuous predictors are additive. Then we consider a model that also contains an interaction, allowing the two effects to be more or less than additive. **The models presented here merely serve the purpose of illustrating how we can fit, analyze, and interpret models with continuous and categorical predictors. The one degree of freedom we have in our model so far (Condition) already puts as at the maximum of the recommended degrees of freedom for 20 data points.** Including additional parameters, as we do below, increases the risk of overfitting the model to the data.

6.1 Do both the diffusion constant and the crowdedness condition affect threshold performance?

For example, let's test the effects of both Condition and DiffusionConstant. For simplicity's sake, we continue to use deviation coding for Condition:

$$Threshold \sim 1 + Condition + DiffusionConstant$$

What happens when we include both of these predictors in the LM?

```
Call:
lm(formula = Threshold ~ 1 + Condition + DiffusionConstant, data = d)

Residuals:
     Min       1Q   Median       3Q      Max
-0.29652 -0.13259 -0.04913  0.10420  0.44403

Coefficients:
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                   1.260450   0.097333  12.950 3.11e-10 ***
ConditionCrowded vs. Uncrowded 0.431381   0.091055   4.738  0.00019 ***
DiffusionConstant             0.034632   0.005434   6.373 6.94e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1971 on 17 degrees of freedom
Multiple R-squared:  0.8307,    Adjusted R-squared:  0.8108
F-statistic: 41.71 on 2 and 17 DF,  p-value: 2.775e-07
```

Right away, we can see that the diffusion constant seems to account for a *lot* of additional variability in the model: the R^2 of the model is almost twice as large as the one we obtained when only considering condition. The output of the regression also tell us that both of the predictors have statistically significant effects on subjects' threshold performance. We can visualize the data and the model's predictions together:

6.1.1 Write-up

Here's a somewhat less detailed write-up for our two-predictor model, building on the example provided above:

We analyzed the data with a linear regression, using the function `lm` from the `base` package (citation with version) of the software `R` (citation with version). We calculated each subject's mean thresholds for the crowded and uncrowded condition. These 20 threshold values were regressed against condition (deviation-coded: .5 = *crowded* vs. -.5 = *uncrowded*) **and the diffusion constant**. We found a statistically significant effect of condition ($\hat{\beta} = .431, t = 4.738, p < .01$), so that subjects reach threshold performance at larger sizes in the crowded condition, compared to the uncrowded condition. **We also found a statistically significant effect of the diffusion constant ($\hat{\beta} = .035, t = 6.373, p < .01$), so that larger diffusion constants required larger sizes to achieve threshold performance.**

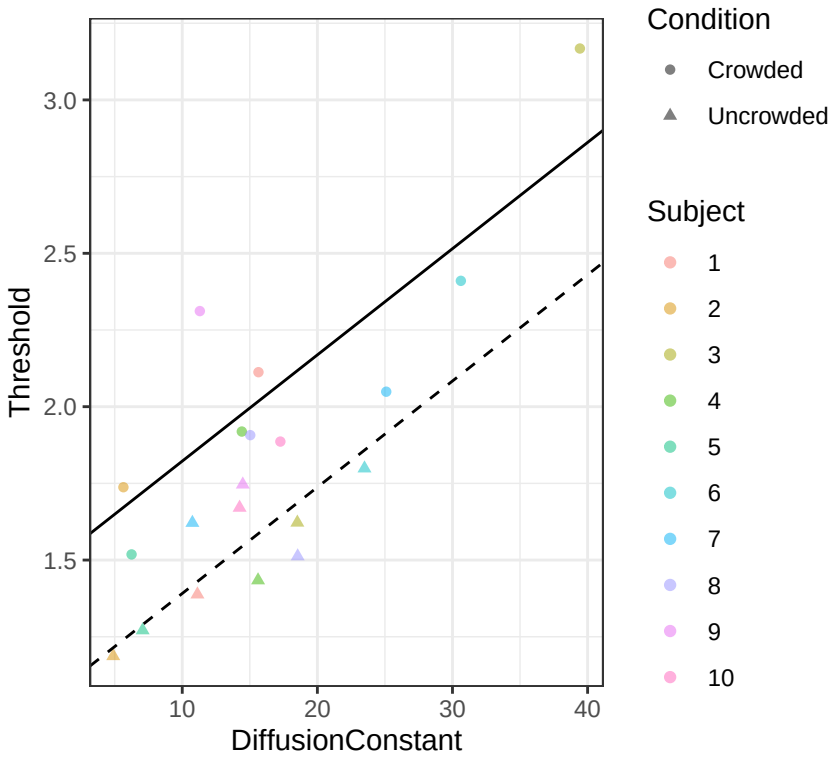

Figure 2: Size (in arcminutes) at which threshold performance (62.5% correct) was reached by diffusion constant and crowding condition. Each color shows a separate subject. Black lines show the predicted outcome of a linear model fit to the data.

6.1.2 Prepare for class

1. Notice how that the intercept estimate in this model is not the same as in the model that only contains Condition as a predictor. That also means that the intercept no longer represents the prediction for the grand mean of the outcome. Was that inevitable? Can you think of a scenario in which the inclusion of the additional predictor (DiffusionConstant) would *not* change the intercept estimate? (Recall that the intercept always is the model's prediction when all other terms of the model add to 0.) If you get stuck on this question, it will get resolved in the next section, but think about it before you read on.
2. The coefficient estimate for Condition has changed. Specifically, it is now somewhat smaller (.431 vs. .576). What do you make out this? Does it tell you something about the relation between the two *predictors* (Condition and DiffusionConstant)?
3. What is an intuitive geometric interpretation of this model? The effect of a single continuous predictor (DiffusionConstant) is described by the intercept and slope. So what does the present model result in? (If you get stuck in thinking about this, re-read Gelman & Hill, 2007, p. 31-33).

6.1.3 Discussion questions for class

4. Can we conclude from this model, and the fact that the two predictors are both significant, that the two effects are additive? Why or why not?
5. Can we conclude that DiffusionConstant has an effect in both crowdedness conditions? Why or why not?
6. Can we conclude that DiffusionConstant had an independent effect beyond condition? Why or why not?

7 Intermezzo: centering continuous predictors

Recall that the intercept estimate changed once we added the DiffusionConstant as a predictor to the LM. As we've already covered, the intercept always gives us the prediction when all other terms in the model add up to 0 (because, if $\beta_1 x_1 + ... + beta_k x_k = 0$ then the LM predicts that $E(y) = \beta_0$). Partly for this reason, it is often recommended that all predictors in the model are *centered*. Since the mean of a centered predictor x_i is 0 (i.e., $E(x_i) = 0$), the average of the term $\beta_i x_i$ is also 0. So if *all* predictors in a model are centered, then $\forall i : \beta_i x_i = 0 \Rightarrow \Sigma_{i=1}^{k} \beta_i x_i = 0 \Rightarrow E(y) = \beta_0$.

For data that is balanced with regard to a factor (e.g., Condition), deviation-coding results in a centered predictor. For example, for the deviation-coded model introduced above, we coded the crowded condition as .5 and the uncrowded condition as -.5. If both conditions appear equally often in the data (as they do), then the average of the implicitly created numerical predictor that results from this coding is 0.

But what about the continuous predictor in our model. DiffusionConstant does not have a mean of zero:

```
round(mean(d$DiffusionConstant), 5)
```

```
[1] 15.97013
```

In the combined model from the previous section, the intercept estimate thus corresponds to the threshold value that is expected when the diffusion constant is 0, which it never is because all values of the diffusion constant are positive:

```
range(d$DiffusionConstant)
```

```
[1]  4.893119 39.431606
```

Once we center diffusion constant by subtracting its mean from each of its values, the new mean of the centered diffusion constant (DiffusionConstant_c) is 0:

```
d %<>%
  mutate(across(where(is.numeric), list("c" = function(x) x - mean(x))))
round(mean(d$DiffusionConstant_c), 5)
```

```
[1] 0
```

When we re-fit the combined LM from the previous section with the new centered diffusion constant, the **intercept estimate now again predicts the overall mean threshold**:

```
Call:
lm(formula = Threshold ~ 1 + Condition + DiffusionConstant_c,
    data = d)

Residuals:
     Min       1Q   Median       3Q      Max
-0.29652 -0.13259 -0.04913  0.10420  0.44403

Coefficients:
                             Estimate Std. Error t value Pr(>|t|)
(Intercept)                  1.813535   0.044077  41.145  < 2e-16 ***
ConditionCrowded vs. Uncrowded 0.431381   0.091055   4.738  0.00019 ***
DiffusionConstant_c          0.034632   0.005434   6.373 6.94e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1971 on 17 degrees of freedom
Multiple R-squared:  0.8307,    Adjusted R-squared:  0.8108
F-statistic: 41.71 on 2 and 17 DF,  p-value: 2.775e-07
```

This—that the intercept estimates is the predicted grand mean of the outcome—will always be the case, no matter how many predictors we have in the model.

Note further that centering does *not* affect the slope estimates for the other effects in the model. Neither does it affect the standard error estimates of those slopes (or t statistics or p-value). Removing the mean from a predictor is a linear transformation does not affect the best-fitting slope of that predictor in predicting the outcome of an LM. Neither does centering affect the predicted/fitted values of the LM (you can compare the 20 predicted/fitted values for the combined model before and after centering DiffusionConstant). The same holds, of course, for the residuals, the R^2, the adusted R^2 and similar measures.

This will *not* always be the case. In the next section, for example, we will see that centering can change the standard error (and thus t statistic and p-value) for an interaction. More generally, centering can affect the the standard error when the removal of the mean from each variable changes the correlations between predictors (we we will return to the issue of correlations between predictors in future meetings).

8 Interactions between factors and continuous predictors

Next, we expand the analysis further. We remove the additivity assumption. Or rather, the addidivity assumption still holds but we expand the model in a way that we are not assuming the our two predictors are additive. This is done by including an interaction between the two predictors. Here I show this for the case of one continuous predictor (DiffusionConstant) and one binary categorical predictor (Condition), but the same logic extends to interactions between multiple continuous or multiple categorical predictors, as well as interactions between interactions (e.g., three-way interactions, etc.). Like for the remainder of this tutorial, we focus on the type of planned analysis typical for (or at least the idea of) experimental psychology. For that reason, we do not discuss considerations about when we should even consider an interaction (but see, for example, Harrell, 2001 for insightful discussion, as well as Gelman & Hill, 2007, p. 36).[1]

For the case of one continuous and one categorical predictor, the geometric interpretation of the model is that we now allow the two lines (corresponding to the continuous predictor's effect at the two levels of the categorical predictor) to not be parallel. And our question of whether the interaction is *significant* becomes the question whether the difference in the slope of the two lines is statistically different from zero.

We can ask this question by adding a new predictor to the model that is the *product* of the two predictors (cf. Gelman & Hill, 2007, p. 34-36; James et al., 2013, p. 87-90). We then ask whether this new predictor has a non-zero effect, i.e., we ask whether the coefficient for this new predictor is different from zero. In R, the regression formula for this model is written as (where the colon is the interaction operator):

$$Threshold \sim 1 + Condition + DiffusionConstant + Condition : DiffusionConstant$$

which essentially runs the following model (where I is the identity operator, which return x for x):

$$Threshold \sim 1 + Condition + DiffusionConstant + I(Condition * DiffusionConstant)$$

or shorter (where X1 * X2 is a shorthand for the full-factorial combination of X1 and X2):

$$Threshold \sim 1 + Condition * DiffusionConstant$$

Just like, e.g., factor coding implicit creates an additional numerical variable that encodes the factor's information, the interaction operator implicitly creates a new variable in our data that is the product if the (numerically coded) factor Condition and the continuous predictor DiffusionConstant. To see this, we again look at the model matrix:

With this change in contrast, the model matrix also changes:

```
model.matrix( ~ 1 + Condition * DiffusionConstant, data = d)
```

```
   (Intercept) ConditionCrowded vs. Uncrowded DiffusionConstant ConditionCrowded vs. Uncrowded:DiffusionConstant
1            1                          -0.5         11.123343                                        -5.561671
2            1                          -0.5          4.893119                                        -2.446560
3            1                          -0.5         18.518145                                        -9.259072
4            1                          -0.5         15.609822                                        -7.804911
5            1                          -0.5          7.069902                                        -3.534951
6            1                          -0.5         23.487701                                       -11.743851
7            1                          -0.5         10.750472                                        -5.375236
```

[1] The short of those considerations is that introducing additional complexity (degrees of freedom) increases the risk of overfitting to the data, and with it the risk of spurious/unreliable conclusions. It is thus advisable to include additional complexity in the model only if is is motivated theoretically, by previous work, or other a priori considerations. Another specific recommendation you'll see often is to only include interactions if all their components are also included in the model, unless there is a good theoretical reason to do otherwise. These recommendations are not specific to the LM.

```
8           1                         -0.5        18.541809                                -9.270905
9           1                         -0.5        14.479893                                -7.239946
10          1                         -0.5        14.243763                                -7.121881
11          1                          0.5        15.638986                                 7.819493
12          1                          0.5         5.644884                                 2.822442
13          1                          0.5        39.431606                                19.715803
14          1                          0.5        14.408659                                 7.204329
15          1                          0.5         6.249123                                 3.124562
16          1                          0.5        30.619534                                15.309767
17          1                          0.5        25.096704                                12.548352
18          1                          0.5        15.031910                                 7.515955
19          1                          0.5        11.302197                                 5.651099
20          1                          0.5        17.261093                                 8.630547
attr(,"assign")
[1] 0 1 2 3
attr(,"contrasts")
attr(,"contrasts")$Condition
          Crowded vs. Uncrowded
Crowded                     0.5
Uncrowded                  -0.5
```

8.1 Do the effects of condition and diffucusion constant interact?

Now we can fit the new LM with the interaction, and look at the summary of results. We will use the centered DiffusionConstant for this model—i.e., we'll fit the model $ Threshold 1 + Condition*DiffusionConstant_c$. For this model, we can refer to the effect of condition as the "*main* effect" of condition (roughyl speaking, the overall, or mean, effect of condition). If we didn't center the diffusion constant—and more generally, if there were other uncentered variables in the model—the effect of of condition would not be conventionally referred to as main effect.

```
Call:
lm(formula = Threshold ~ 1 + Condition * DiffusionConstant_c,
    data = d)

Residuals:
     Min       1Q   Median       3Q      Max
-0.30964 -0.14039 -0.05505  0.10824  0.45666

Coefficients:
                                                  Estimate Std. Error t value Pr(>|t|)
(Intercept)                                       1.804444   0.047022  38.374  < 2e-16 ***
ConditionCrowded vs. Uncrowded                    0.441727   0.094044   4.697 0.000242 ***
DiffusionConstant_c                               0.032167   0.006725   4.783 0.000203 ***
ConditionCrowded vs. Uncrowded:DiffusionConstant_c 0.008665   0.013449   0.644 0.528534
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2006 on 16 degrees of freedom
Multiple R-squared:  0.835, Adjusted R-squared:  0.8041
F-statistic: 26.99 on 3 and 16 DF,  p-value: 1.695e-06
```

The first thing we can notice is that the R^2 of the model has barely increased. In other words, adding the interaction doesn't explain much more variance in the outcome. Indeed, the adjusted R^2 (which is the R^2 corrected for the complexity of the model) has *de*creased. With this in mind, it is not surprising that the interaction does not have a significant effect. In short, we cannot reject the null hypothesis that the effect of the diffusion constant on the threshold is identical for crowded and uncrowded condition. This is also in line with a visualization of the data and the model:

Note also that the intercept estimate in the interaction model is not identical to the one from the combined model. This seems to contradict what we learned above, that the intercept estimate is the model's prediction for the grand mean if all predictors are centered. Note, however, that we did not center the interaction and, indeed, the mean of the product of condition and DiffusionConstant is *not* 0, but rather 1.049. There's thus no contradiction to the generalization that the intercept is the model's predicted overall mean outcome when all variables are centered.[2]

[2]Since the diffusion constant is not a variable that we controlled directly by our design, it can have different means for the two conditions even after centering it, and indeed it does (centering only guarantees that the *overall* mean of the diffusion constant is 0):

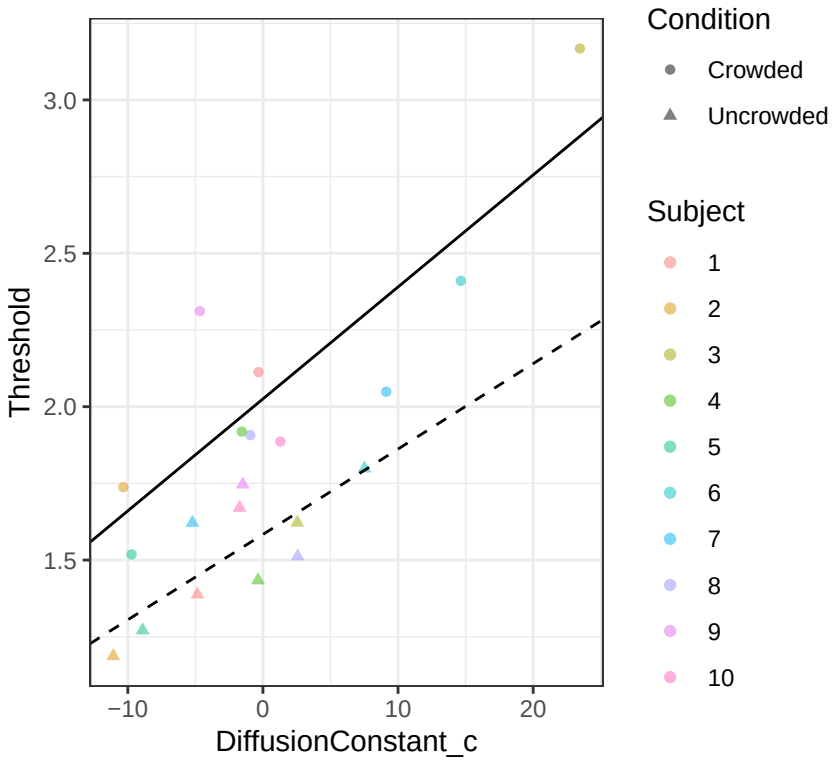

Figure 3: Size (in arcminutes) at which threshold performance (62.5% correct) was reached by diffusion constant (centered) and crowding condition. Each color shows a separate subject. Black lines show the predicted outcome of a linear model fit to the data.

8.1.1 Prepare for class

1. On Figure 3, draw line segment that correspond to the intercept.
2. On Figure 3, draw line segment that correspond to the effect of condition.
3. Calculate the slope for the crowded condition from the model output shown above.
4. What's a geometric interpretation of an interaction between two *continuous* predictors (x1 and x2)? For this it might be helpful to first think about the geometric interpretation of two additive continuous predictors (a plane over x1 and x2, the height of which is given along the third axis, y)
5. True or false? Since the interaction does not have a significant effect, we can safely remove it from the model.

8.1.2 Discussion questions for class

6. True or false? Now that we know that the slope of diffusion constant does not significantly differ between the two crowdedness conditions, and given that the effect of diffusion constant was significant, we can conclude that both conditions exhibit a significant effect of the diffusion constant.
7. True or false? If we had found a significant interaction between condition and diffusion constant, we could have concluded that the effect of conclusion constant goes in opposite directions for the two conditions?
8. When we fit the same interaction model with the uncentered diffusion constant instead, we get a seemingly conflicting result, where condition does not longer have a significant effect. Why is that the case? Is this result really conflicting? (The figure might help. Hint: draw the line segments corresponding to the effect of condition in this new interaction model.)

```
Call:
lm(formula = Threshold ~ 1 + Condition * DiffusionConstant, data = d)
```

In other words, the diffusion constant is itself affected by the condition and, as a consequence, condition and DiffusionConstant are correlated. Here we don't explore this further, but we will return to questions about collinearity later in the semester.

```
Residuals:
     Min       1Q   Median       3Q      Max
-0.30964 -0.14039 -0.05505  0.10824  0.45666

Coefficients:
                                                Estimate Std. Error t value Pr(>|t|)
(Intercept)                                     1.290728   0.109636  11.773 2.71e-09 ***
ConditionCrowded vs. Uncrowded                  0.303347   0.219271   1.383 0.185531
DiffusionConstant                               0.032167   0.006725   4.783 0.000203 ***
ConditionCrowded vs. Uncrowded:DiffusionConstant 0.008665   0.013449   0.644 0.528534
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2006 on 16 degrees of freedom
Multiple R-squared:  0.835, Adjusted R-squared:  0.8041
F-statistic: 26.99 on 3 and 16 DF,  p-value: 1.695e-06
```

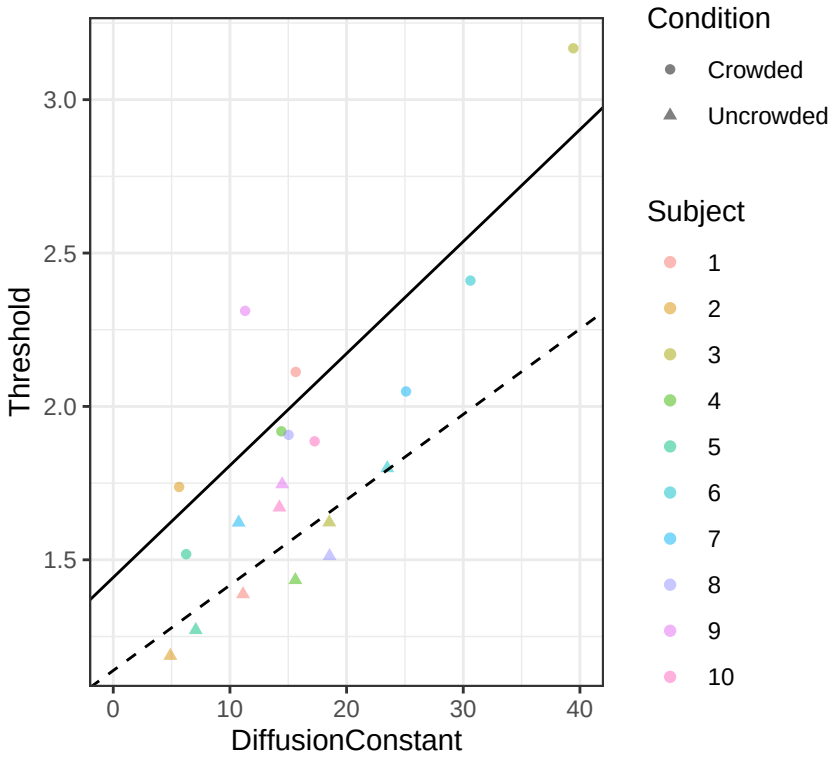

Figure 4: Size (in arcminutes) at which threshold performance (62.5% correct) was reached by diffusion constant (not-centered) and crowding condition. Each color shows a separate subject. Black lines show the predicted outcome of a linear model fit to the data.

1. Are the predicted/fitted values from this model different from the first interaction model we fit above?

8.2 Simple effects

When we have interactions in our model, we might also want to report the *simple effects*. For example, for the two-way interaction between condition and diffusion constant, we might want to report the simple effect of the diffusion constant for each condition. That's necessary because a significant interaction between two predictors x_1 and x_2 only tells us that the effect of x_1 on y differs depending on x_2 (or vice versa, that the effect of x_2 on y depends on the value of x_1). For example, for the present case a significant interaction would have indicated that the slope of diffusion constant differs between the two crowdedness conditions.

However, a signficant interaction does *not* tell us *how* the interacting effects depends on each other. For example, for the present case a significant interaction could indicate that there is a significant positive relation between the diffusion constant and threshold in one condition and a significant negative relation in the other

condition; but it could also indicate that there is a significant positive relation in one condition and a non-significant relation in the other; or even that the relation is significant and positive in both conditions but the size of the effect of diffusion constant differs between condition (and so on). To address which of those scenarios holds, it is necessary to conduct additional analyses. The standard approach to that is called simple effect analyses.

In this approach, we do not split the data into the two conditions and fit separate models ($Threshold \sim 1 + DiffusionConstant$) to it. Rather, we use all of the data we have and simple re-parameterize/recode the same LM we have already fit to read out the simple effects of DiffusionConstant at each level of condition. In R, this can be done with the convenient embedding operator /. Here's the model matrix resulting from this operator does for the present case:

```
model.matrix( ~ 1 + Condition / DiffusionConstant_c, data = d)
```

```
   (Intercept) ConditionCrowded vs. Uncrowded ConditionCrowded:DiffusionConstant_c ConditionUncrowded:DiffusionConstant_c
1            1                          -0.5                                0.0000000                                -4.846791
2            1                          -0.5                                0.0000000                               -11.077014
3            1                          -0.5                                0.0000000                                 2.548011
4            1                          -0.5                                0.0000000                                -0.360311
5            1                          -0.5                                0.0000000                                -8.900231
6            1                          -0.5                                0.0000000                                 7.517568
7            1                          -0.5                                0.0000000                                -5.219661
8            1                          -0.5                                0.0000000                                 2.571676
9            1                          -0.5                                0.0000000                                -1.490241
10           1                          -0.5                                0.0000000                                -1.726370
11           1                           0.5                               -0.3311476                                 0.000000
12           1                           0.5                              -10.3252492                                 0.000000
13           1                           0.5                               23.4614730                                 0.000000
14           1                           0.5                               -1.5614743                                 0.000000
15           1                           0.5                               -9.7210102                                 0.000000
16           1                           0.5                               14.6494003                                 0.000000
17           1                           0.5                                9.1265712                                 0.000000
18           1                           0.5                               -0.9382233                                 0.000000
19           1                           0.5                               -4.6679358                                 0.000000
20           1                           0.5                                1.2909599                                 0.000000
attr(,"assign")
[1] 0 1 2 2
attr(,"contrasts")
attr(,"contrasts")$Condition
          Crowded vs. Uncrowded
Crowded                     0.5
Uncrowded                  -0.5
```

and here is the resulting LM:

```
Call:
lm(formula = Threshold ~ 1 + Condition/DiffusionConstant_c, data = d)

Residuals:
     Min       1Q   Median       3Q      Max
-0.30964 -0.14039 -0.05505  0.10824  0.45666

Coefficients:
                                    Estimate Std. Error t value Pr(>|t|)
(Intercept)                         1.804444   0.047022  38.374  < 2e-16 ***
ConditionCrowded vs. Uncrowded      0.441727   0.094044   4.697 0.000242 ***
ConditionCrowded:DiffusionConstant_c   0.036500   0.006243   5.846 2.48e-05 ***
ConditionUncrowded:DiffusionConstant_c 0.027835   0.011912   2.337 0.032790 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2006 on 16 degrees of freedom
Multiple R-squared:  0.835, Adjusted R-squared:  0.8041
F-statistic: 26.99 on 3 and 16 DF,  p-value: 1.695e-06
```

This re-parameterization does not change the predicted/fitted responses of the model (not shown here) or the model fit (R^2, F-statistic, etc.; shown). But we now are provided with the effect of the diffusion constant at each level of the condition variable. In this case, we see that the effect of diffusion constant on the threshold is signficant and positive in both conditions.

8.3 Writing up the analysis results

We analyzed the data with a linear regression, using the function `lm` from the `base` package (citation with version) of the software `R` (citation with version). We calculated each subject's mean thresholds for the crowded and uncrowded condition. These 20 threshold values were regressed against condition (deviation-coded: .5 = *crowded* vs. -.5 = *uncrowded*), the diffusion constant **(centered), and their interaction**. We found a statistically significant **main** effect of condition ($\hat{\beta} = .442, t = 4.697, p < .01$), so that subjects reach threshold performance at larger sizes in the crowded condition, compared to the uncrowded condition. We also found a statistically significant effect of the diffusion constant ($\hat{\beta} = .037, t = 5.846, p < .01$), so that larger diffusion constants required larger sizes to achieve threshold performance. **The interaction was not significant ($\hat{\beta} = 0.009, p > .5$).**

9 Additional topics we can discuss in class

9.1 Coding of continuous predictors

- scaling (dividing continuous predictors through the standard deviation). Why is this done? What are the pros and cons?
- scaling by dividing through two-times the predictors standard deviation (Gelman, 2008). Why is this done?
- non-linear transforms of continuous predictors (log-transforming, polynomials)? Why is this done? In what sense does the LM still make a linearity assumption? (see also James et al., 2013, p. 90-92)

9.2 Coding of factors with more than two levels

- Testing hypotheses about the order of levels (see also James et al., 2013, p. 85; a nice summary of various coding schemes and how to make your own for R can be found at https://stats.idre.ucla.edu/r/library/r-library-contrast-coding-systems-for-categorical-variables/#:~:text=2%20Simple%20Coding&text=The%20difference%20in%20the
  - Helmert coding (or reverse Helmert coding)
  - simple coding
  - sliding difference coding (forward/backward)
  - polynomial coding (or orthogonal polynomial coding)
- How do we add and interpret interactions with such multi-level factors? A nice detailed introduction that goes over the interpretation for the different coefficients can be found at https://stats.idre.ucla.edu/other/mult-pkg/faq/general/faq-how-do-i-interpret-the-coefficients-of-an-effect-coded-variable-involved-in-an-interaction-in-a-regression-model/. R provides many packages that help with the interpretation of higher-order interactions. One is phia (https://cran.r-project.org/web/packages/phia/index.html).

10 Session info

```
devtools::session_info()
```

```
- Session info ---------------------------------------------------------------------------------------------------------------------------
 setting  value
 version  R version 3.6.0 (2019-04-26)
 os       macOS High Sierra 10.13.6
 system   x86_64, darwin15.6.0
 ui       X11
 language (EN)
 collate  en_US.UTF-8
 ctype    en_US.UTF-8
 tz       America/New_York
 date     2020-09-23

- Packages -------------------------------------------------------------------------------------------------------------------------------
 package     * version date       lib source
 assertthat    0.2.1   2019-03-21 [1] CRAN (R 3.6.0)
```

```
backports     1.1.7   2020-05-13 [1] CRAN (R 3.6.2)
broom       * 0.5.5   2020-02-29 [1] CRAN (R 3.6.0)
callr         3.4.3   2020-03-28 [1] CRAN (R 3.6.2)
cellranger    1.1.0   2016-07-27 [1] CRAN (R 3.6.0)
cli           2.0.2   2020-02-28 [1] CRAN (R 3.6.0)
colorspace    1.4-1   2019-03-18 [1] CRAN (R 3.6.0)
crayon        1.3.4   2017-09-16 [1] CRAN (R 3.6.0)
DBI           1.1.0   2019-12-15 [1] CRAN (R 3.6.0)
dbplyr        1.4.2   2019-06-17 [1] CRAN (R 3.6.0)
desc          1.2.0   2018-05-01 [1] CRAN (R 3.6.0)
devtools      2.2.2   2020-02-17 [1] CRAN (R 3.6.0)
digest        0.6.25  2020-02-23 [1] CRAN (R 3.6.0)
dplyr       * 1.0.2   2020-08-18 [1] CRAN (R 3.6.2)
ellipsis      0.3.1   2020-05-15 [1] CRAN (R 3.6.2)
evaluate      0.14    2019-05-28 [1] CRAN (R 3.6.0)
fansi         0.4.1   2020-01-08 [1] CRAN (R 3.6.0)
farver        2.0.3   2020-01-16 [1] CRAN (R 3.6.0)
forcats     * 0.5.0   2020-03-01 [1] CRAN (R 3.6.0)
fs            1.3.2   2020-03-05 [1] CRAN (R 3.6.0)
generics      0.0.2   2018-11-29 [1] CRAN (R 3.6.0)
ggplot2     * 3.3.0   2020-03-05 [1] CRAN (R 3.6.0)
glue          1.4.1   2020-05-13 [1] CRAN (R 3.6.0)
gtable        0.3.0   2019-03-25 [1] CRAN (R 3.6.0)
haven         2.2.0   2019-11-08 [1] CRAN (R 3.6.0)
hms           0.5.3   2020-01-08 [1] CRAN (R 3.6.0)
htmltools     0.4.0   2019-10-04 [1] CRAN (R 3.6.0)
httr          1.4.1   2019-08-05 [1] CRAN (R 3.6.0)
jsonlite      1.6.1   2020-02-02 [1] CRAN (R 3.6.0)
knitr       * 1.28    2020-02-06 [1] CRAN (R 3.6.0)
labeling      0.3     2014-08-23 [1] CRAN (R 3.6.0)
lattice       0.20-40 2020-02-19 [1] CRAN (R 3.6.0)
lifecycle     0.2.0   2020-03-06 [1] CRAN (R 3.6.0)
lubridate     1.7.4   2018-04-11 [1] CRAN (R 3.6.0)
magrittr    * 1.5     2014-11-22 [1] CRAN (R 3.6.0)
memoise       1.1.0   2017-04-21 [1] CRAN (R 3.6.0)
modelr        0.1.6   2020-02-22 [1] CRAN (R 3.6.0)
munsell       0.5.0   2018-06-12 [1] CRAN (R 3.6.0)
nlme          3.1-145 2020-03-04 [1] CRAN (R 3.6.0)
pillar        1.4.4   2020-05-05 [1] CRAN (R 3.6.2)
pkgbuild      1.0.8   2020-05-07 [1] CRAN (R 3.6.2)
pkgconfig     2.0.3   2019-09-22 [1] CRAN (R 3.6.0)
pkgload       1.0.2   2018-10-29 [1] CRAN (R 3.6.0)
prettyunits   1.1.1   2020-01-24 [1] CRAN (R 3.6.0)
processx      3.4.2   2020-02-09 [1] CRAN (R 3.6.0)
ps            1.3.3   2020-05-08 [1] CRAN (R 3.6.2)
purrr       * 0.3.3   2019-10-18 [1] CRAN (R 3.6.0)
R6            2.4.1   2019-11-12 [1] CRAN (R 3.6.0)
Rcpp          1.0.4.6 2020-04-09 [1] CRAN (R 3.6.0)
readr       * 1.3.1   2018-12-21 [1] CRAN (R 3.6.0)
readxl        1.3.1   2019-03-13 [1] CRAN (R 3.6.0)
remotes       2.1.1   2020-02-15 [1] CRAN (R 3.6.0)
reprex        0.3.0   2019-05-16 [1] CRAN (R 3.6.0)
rlang         0.4.7   2020-07-09 [1] CRAN (R 3.6.2)
rmarkdown     2.1     2020-01-20 [1] CRAN (R 3.6.0)
rprojroot     1.3-2   2018-01-03 [1] CRAN (R 3.6.0)
rstudioapi    0.11    2020-02-07 [1] CRAN (R 3.6.0)
rvest         0.3.5   2019-11-08 [1] CRAN (R 3.6.0)
scales        1.1.1   2020-05-11 [1] CRAN (R 3.6.2)
sessioninfo   1.1.1   2018-11-05 [1] CRAN (R 3.6.0)
stringi       1.4.6   2020-02-17 [1] CRAN (R 3.6.0)
stringr     * 1.4.0   2019-02-10 [1] CRAN (R 3.6.0)
testthat      2.3.2   2020-03-02 [1] CRAN (R 3.6.0)
tibble      * 3.0.1   2020-04-20 [1] CRAN (R 3.6.2)
tidyr       * 1.0.2   2020-01-24 [1] CRAN (R 3.6.0)
tidyselect    1.1.0   2020-05-11 [1] CRAN (R 3.6.2)
tidyverse   * 1.3.0   2019-11-21 [1] CRAN (R 3.6.0)
usethis       1.5.1   2019-07-04 [1] CRAN (R 3.6.0)
utf8          1.1.4   2018-05-24 [1] CRAN (R 3.6.0)
vctrs         0.3.4   2020-08-29 [1] CRAN (R 3.6.2)
withr         2.2.0   2020-04-20 [1] CRAN (R 3.6.2)
xfun          0.12    2020-01-13 [1] CRAN (R 3.6.0)
```

```
 xml2          1.2.2   2019-08-09 [1] CRAN (R 3.6.0)
 yaml          2.2.1   2020-02-01 [1] CRAN (R 3.6.0)

[1] /Library/Frameworks/R.framework/Versions/3.6/Resources/library
```