The general linear model
Lecture 6 – The general*ised* linear model

Jan Vanhove

https://janhove.github.io

Ghent, 14–16 July 2025

Up till now, we've only dealt with outcome variables that we considered to be more or less continuous.[1] In language research, however, we often deal with binary outcome data such as accuracy (correct/incorrect), presence vs. absence, etc. It is possible to analyse such binary data in a general linear model by coding one level of the variable as 1 and the other as 0. The predicted values can then be interpreted as the estimated success probabilities (e.g., probability of a correct answer, of the presence of a given phoneme, etc.). This is referred to as the **linear probability model**, and it may be an appropriate tool when analysing the data from experiments with categorical predictors (see Hellevik, 2009; Huang, 2019). While the residuals of such models are bound to violate the assumptions of equality of distributions and normality, bootstrapping techniques can be used to double-check the inferences drawn from such models under different sets of assumptions.

That said, researchers should be aware of some possible problems with the linear probability model (see Jaeger, 2008). An especially glaring problem is that linear probability models with continuous predictors can generate impossible predictions, namely probabilities smaller than 0 or greater than 1. Even for categorical predictors, such models can yield confidence intervals that lie partially outside of the $[0, 1]$ interval.

While the latter problem can be addressed in linear probability models by constructing the confidence intervals using bootstrap techniques, another option is to adjust the general linear model so that it can accommodate binary (and other) outcome types. This gives rise to the **generalised linear model** (I didn't come up with these names…). In our field, the most common type of generalised linear model beside the general linear model is the logistic regression model for dealing with binary outcome data, which we'll introduce in this lecture.

[1] In none of the real-life examples we encountered were the outcomes literally continuous, though. Scores on a 204-item test can only be integers between 0 and 204, for instance. Hence 'considered'.

Table 6.1: Number of choices for Programme A and Programme B depending on how the programmes were presented (Tversky & Kahneman, 1981).

	Programme A	Programme B
Gain framing	109	43
Loss framing	34	121

1 Categorical predictors

Tversky & Kahneman (1981) presented their participants with a hypothetical scenario and asked them to choose between one of two actions:

> "Imagine that the U.S. is preparing for the outbreak of a an unusual Asian disease, which is expected to kill 600 people. Two alternative programs to combat the disease have been proposed. Assume that the exact scientific estimate of the consequences of the programs are as follows:"

For about half of the participants, the consequences of these programmes were framed as gains (*gain framing*):

> "If Program A is adopted, 200 people will be saved.
>
> If Program B is adopted, there is a 1/3 probability that 600 people will be saved, and a 2/3 probability that no people will be saved."

For the other participants, the consequences of these programmes were framed as losses (*loss framing*):

> "If Program A is adopted, 400 people will die.
>
> If Program B is adopted, there is a 1/3 probability that nobody will die, and a 2/3 probability that 600 people will die."

Note that the outcome of Programme A is the same regardless of how its consequences are framed, and the same goes for Programme B. Note further that both programmes have the same expected value in terms of lives saved, namely 200. The only difference is that this is a certainty in Programme A and a probabilistic expectation in Programme B. Table 6.1 shows how the participants' preferences varied between the framing conditions.

Let's reconstruct the dataset:

```
library(tidyverse)
theme_set(theme_bw())
d <- tibble(
  framing = c(rep("gain", 109 + 43),
```

```
              rep("loss", 34 + 121)),
  choice = c(rep("A", 109), rep("B", 43),
             rep("A", 34), rep("B", 121))
)
# inspect:
slice_sample(d, n = 8)
```

```
# A tibble: 8 x 2
  framing choice
  <chr>   <chr>
1 loss    B
2 loss    B
3 gain    A
4 loss    B
5 gain    A
6 gain    A
7 gain    B
# i 1 more row
```

```
xtabs(~ framing + choice, d)
```

```
       choice
framing   A   B
   gain 109  43
   loss  34 121
```

We can compute the proportion of choices for Programme A (the sure option) like so:

```
d |>
  group_by(framing) |>
  summarise(proportion_sure = mean(choice == "A"))
```

```
# A tibble: 2 x 2
  framing proportion_sure
  <chr>             <dbl>
1 gain              0.717
2 loss              0.219
```

Using the linear probability model, the difference in the proportions can be retrieved, and uncertainty estimates for this difference can be computed:

```
d <- d |>
  mutate(sure_choice = ifelse(choice == "A", 1, 0),
```

```
        n.loss = ifelse(framing == "loss", 1, 0))  # treatment coding
tversky.lm <- lm(sure_choice ~ n.loss, data = d)
summary(tversky.lm)$coefficients
```

```
            Estimate Std. Error t value   Pr(>|t|)
(Intercept)  0.71711   0.035180  20.384 6.9944e-59
n.loss      -0.49775   0.049511 -10.053 1.0034e-20
```

```
confint(tversky.lm)
```

```
               2.5 %   97.5 %
(Intercept)  0.64788  0.78633
n.loss      -0.59518 -0.40032
```

That is, the percentage of sure choices is just shy of 50 ± 5 percentage points lower in the loss framing condition than in the gain framing condition. The 95% confidence interval spans from -60 to -40 percentage points. You can double-check these confidence intervals using a semi-parametric bootstrap that does not assume equality of distributions, but we won't do so here.

As explained in the introduction, a possible problem with the linear probability model is that the model generates predictions (and confidence intervals) that go outside the permissible range for probabilities. While this isn't the case here, we still need to work out a solution should we ever need it. One such solution is to not work directly with proportions or probabilities but to work with so-called log-odds instead. To understand these, we first need to understand what odds and odds ratios are.

Instead of saying that the proportion of sure choices under gain framing was 0.7171, we could also say that for each unsure choice, there are about 2.53 sure choices:

$$\frac{109}{43} = \frac{109 \Big/ (109+43)}{43 \Big/ (109+43)} = \frac{0.7171}{1-0.7171} \approx 2.53.$$

The **odds** of a sure choice in the gain framing condition, then, are 2.53 to 1. Similarly, the odds of a sure choice in the loss framing condition are

$$\frac{34}{121} = \frac{0.2194}{1-0.2194} \approx 0.28,$$

that is, for each unsure choice there were 0.28 sure choices.

To convert a proportion p to odds φ, then, we can use the formula

$$\varphi = \frac{p}{1-p}.$$

Given some odds φ, we can backtransform these to the proportion p using the formula

$$p = \frac{\varphi}{1+\varphi}.$$

In order to express the preference difference between the gain and loss framing conditions, we can compute an **odds ratio**. This shows that the odds of a sure choice in the loss framing condition are

$$\frac{34\Big/121}{109\Big/43} = \frac{0.28}{2.53} = 0.11$$

as large as in the gain framing condition. Equivalently, the odds of a sure choice are about 9 times as large in the gain framing condition as in the loss framing condition.

Odds and odds ratios cannot be negative. But by taking their (natural) logarithm, we can further transform them so that their theoretical range is the entire real axis.[2] The natural logarithm of odds or odds ratios are called **log-odds**. In our example, the log-odds of a sure choice in the gain framing condition are

$$\ln 2.53 \approx 0.93,$$

compared to

$$\ln 0.28 \approx -1.27$$

in the loss framing condition. Due to the properties of logarithms ($\log a/b = \log a - \log b$), the difference between these log-odds is the logarithm of the odds ratio:

$$\ln 0.28 - \ln 2.53 = \ln \frac{0.28}{2.53} = \ln 0.11 \approx -2.21.$$

As these computations show, a proportion or probability p can be converted to log-odds η using the following formula:

$$\eta = \ln \varphi = \ln \frac{p}{1-p}.$$

This function, which maps proportions to log-odds, is known as the **logit function**. To com-

[2]For $a > 1, c > 0$, if $a^b = c$, then $\log_a c = b$. The natural logarithm, written ln or simply log, uses Euler's number, $e \approx 2.718$ for a. Note that since $a^0 = 1$, $\log_a 1 = \ln 1 = 0$. Note further that

$$a^{\log_a c} = c.$$

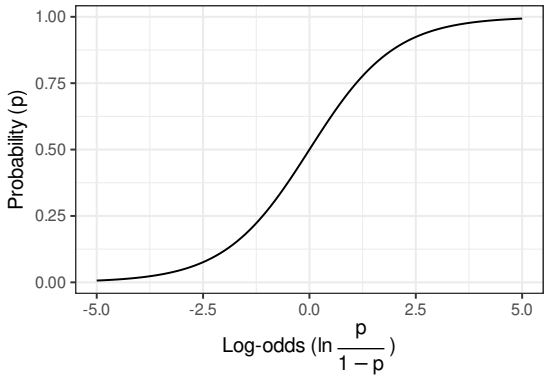

Figure 6.1: The logistic function converts log-odds into probabilities.

pute proportions or probabilities from log-odds, we need to find its inverse:

$$
\begin{aligned}
& & \eta &= \ln \frac{p}{1-p} \\
\Rightarrow & & \exp(\eta) &= \frac{p}{1-p} \\
\Rightarrow & & \exp(\eta) - p\exp(\eta) &= p \\
\Rightarrow & & \exp(\eta) &= p(1+\exp(\eta)) \\
\Rightarrow & & p &= \frac{\exp(\eta)}{1+\exp(\eta)} = \frac{1}{1/(\exp(\eta))+1} = \frac{1}{1+\exp(-\eta)}.
\end{aligned}
$$

For instance, given the log-odds of 0.93, we can compute the original proportion like so:

```
1 / (1 + exp(-0.93))
```

```
[1] 0.71708
```

The function for converting log-odds into probabilities is called the **logistic function**. It is implemented in R as `plogis()`:

```
plogis(0.93)
```

```
[1] 0.71708
```

Figure 6.1 shows that the logistic function looks like.

For our present example, we now let p_i denote the probability that the i-th participant prefers the sure programme, $i = 1, \ldots, 307$. Then we can conceive of each participant's decision as a **Bernoulli experiment** with success probability p_i. More formally, the choice of the i-th

participant is modelled as a Bernoulli random variable $X_i \sim \text{Bernoulli}(p_i)$, which means that

$$\mathbb{P}(X_i = x) = p_i^x (1 - p_i)^{1-x} = \begin{cases} 1 - p_i, & \text{if } x = 0, \\ p_i, & \text{if } x = 1. \end{cases}$$

(For $x \notin \{0, 1\}$, $\mathbb{P}(X_i = x) = 0$.) Our goal is to estimate the p_is. We do so by modelling them as log-odds in a linear model and then backtransforming these into probabilities:

$$\begin{aligned} X_i &\sim \text{Bernoulli}(p_i), \\ p_i &= \frac{1}{1 + \exp(-\eta_i)}, \\ \eta_i &= \beta_0 + \beta_1 x_i, \end{aligned}$$

where $x_i = 1$ if the i-th participant was presented the scenario with loss framing and $x_i = 0$ otherwise and where $X_1, \ldots X_{307}$ are independent.

The model parameters are estimated using maximum likelihood estimation, which was mentioned at the end of Lecture 1. In brief, we pick as our estimates $\widehat{\beta}_0, \widehat{\beta}_1$ the β_0 and β_1 given which the data we have observed would be most likely to occur:

$$\begin{aligned} (\widehat{\beta}_0, \widehat{\beta}_1) &= \underset{(\beta_0, \beta_1)}{\arg\max} \prod_{i=1}^{307} p_i^{X_i} (1 - p_i)^{1 - X_i} \\ &= \underset{(\beta_0, \beta_1)}{\arg\max} \sum_{i=1}^{307} \left(X_i \log(p_i) + (1 - X_i) \log(1 - p_i) \right), \end{aligned}$$

where

$$p_i = \frac{1}{1 + \exp(-(\beta_0 + \beta_1 x_i))}.$$

Figure 6.2 shows that a $\widehat{\beta}_0$ value around 1 and a $\widehat{\beta}_1$ value around -2 maximise these data's likelihood.

The same result can be obtained more quickly using the `glm()` function. It works similarly to the `lm()` we're already familiar with, except that we need to specify a distribution family and a so-called link function. As mentioned above, we consider the participants' choices to be Bernoulli distributed. The Bernoulli distribution is a more specific instance of the binomial distribution, which we won't go into. The choice of the logit link function is due to our working with log-odds.

```
tversky.glm <- glm(sure_choice ~ n.loss, data = d,
                   family = binomial(link = "logit"))
```

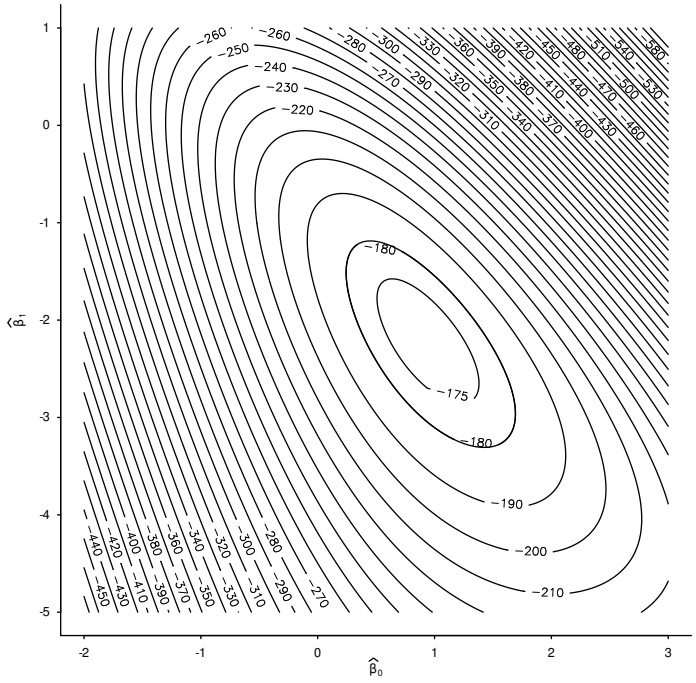

Figure 6.2: The log-likelihood of different choices for $\widehat{\beta}_0$ and $\widehat{\beta}_1$. The log-likelihood is maximised for $\widehat{\beta}_0$ values near 1 and $\widehat{\beta}_1$ values around -2.

```
summary(tversky.glm)$coefficients
```

```
            Estimate Std. Error z value   Pr(>|z|)
(Intercept)  0.93015    0.18008  5.1651 2.4032e-07
n.loss      -2.19958    0.26478 -8.3073 9.7888e-17
```

The interpretation of the estimated model parameters is as follows:

- The log-odds of a sure choice when the `n.loss` value is 0 are about 0.93. This corresponds to odds of about 2.53 and to a probability of about 0.72:

```
exp(0.93)
```

```
[1] 2.5345
```

```
plogis(0.93)
```

```
[1] 0.71708
```

- The difference between the log-odds of a sure choice when the `n.loss` value is 1 compared to when it is 0 is -2.2. This corresponds to an odds ratio of 0.11:

```
exp(-2.2)
```

```
[1] 0.1108
```

- In order to obtain the estimated probability of a sure choice in the loss framing condition, we need to compute the log-odds of this first and then convert them into probabilities using the logistic function:

```
plogis(0.93 - 2.2)
```

```
[1] 0.21926
```

We had already computed all these numbers by hand. The added value of the model is that it produces standard errors for the estimated β parameters as well as approximate confidence intervals. Note that these are expressed in log-odds, too!

```
confint(tversky.glm)
```

```
               2.5 %  97.5 %
(Intercept)  0.58527  1.2931
n.loss      -2.73084 -1.6913
```

We can express these confidence intervals on the odds scale by exponentiation:

```
exp(confint(tversky.glm))
```

```
               2.5 %  97.5 %
(Intercept) 1.795472 3.64420
n.loss      0.065164 0.18428
```

That is, the 95% interval for the estimated odds ratio of 0.11 is about $[0.065, 0.184]$. While *odds* can be converted into probabilities, odds *ratios* cannot. If you want to estimate the effect of framing in percentage points and obtain a confidence interval that is also expressed in percentages, consider fitting a linear probability model.

Remark 6.1 (Inference in generalised linear models)**.** Under certain assumptions, the confidence intervals and p-values reported in the `lm()` output are exact. This is not the case for models fitted using `glm()`: For generalised linear models, only approximate confidence procedures (based on asymptotic results) are available. That said, it is possible to apply (parametric) bootstrap and permutation procedures that may be more reliable for smaller samples in particular; see Remarks 6.6 and 6.7. ♢

Remark 6.2 (Linear probability model done better)**.** Having introduced the `glm()` function, we

can now fit the linear probability model in a slightly better way. Specifically, we assume that

$$\begin{aligned} X_i &\sim \text{Bernoulli}(p_i), \\ p_i &= \eta_i, \\ \eta_i &= \beta_0 + \beta_1 x_i, \end{aligned}$$

$i = 1, \ldots, n$. Compared to the logistic model, the only difference is that the link function is the identity (i.e., $\eta_i = p_i$) instead of the logit function. In contrast to the general linear models considered earlier, there aren't any error terms in this model! So instead of using `lm()`, which estimates error variance, we can use `glm()` with a suitable link function:

```
lpm.glm <- glm(sure_choice ~ n.loss, data = d,
               family = binomial(link = "identity"))
summary(lpm.glm)$coefficients
```

```
            Estimate Std. Error z value   Pr(>|z|)
(Intercept)  0.71711   0.036533  19.629 8.7213e-86
n.loss      -0.49775   0.049390 -10.078 6.9193e-24
```

Compare the output of this model to the output of the model `tversky.lm`. ♢

Remark 6.3 (Significance tests for 2×2 tables)**.** If a full-fledged model isn't required, it is possible to calculate exact p-values for 2×2 contingency tables. See my blogpost *Exact significance tests for* 2×2 *tables* for details. ♢

Generalised linear models can accommodate multiple predictors, as in the following exercise.

Exercise 6.4 (Fitting interactions)**.** Keysar et al. (2012) extended the experiment by Tversky & Kahneman (1981) by having some participants complete the experiment in their native language (English) and others in a foreign language they were learning (Japanese). They were interested in finding out whether the framing effect that Tversky & Kahneman (1981) had observed was as pronounced when the participants deal with the dilemma in a foreign language as it was in their native language. That is, they were interested in the interaction between framing and language.

We read in the data and summarise it numerically:

```
library(here)
d <- read_csv(here("data", "Keysar2012.csv"))
d |>
  group_by(language, framing) |>
  summarise(
    sure_choices = sum(sure_choice == 1),
```

```
    unsure_choices = sum(sure_choice == 0)
  )

# A tibble: 4 x 4
# Groups:   language [2]
  language framing sure_choices unsure_choices
  <chr>    <chr>          <int>          <int>
1 English  gain              24              7
2 English  loss              14             16
3 Japanese gain              13             17
4 Japanese loss              12             18
```

1. (With R.) Create a dummy variable for `language` that has the value 0 for English and 1 for Japanese. Also create a dummy variable for `framing` that has the value 0 for gain framing and 1 for loss framing. (That is, use treatment coding.) Fit a logistic model with these dummy variables and their pointwise product (interaction) as predictors.

2. (Paper and pencil.) You should have obtained the following estimated model parameters:

```
            Estimate Std. Error
(Intercept)   1.2321    0.42956
n.language   -1.5004    0.56592
n.framing    -1.3657    0.56432
interaction   1.2285    0.77012
```

   (a) Explain what each estimated model parameter means literally.

   (b) Based on these coefficients, compute the estimated probability that a participant who read the dilemma in the loss-framing condition and in English picks the sure option. Verify your answer using the numerical summary above.

   (c) Based on these coefficients, compute the estimated probability that a participant who read the dilemma in the loss-framing condition and in Japanese picks the sure option. Verify your answer using the numerical summary above.

3. (With R.) Compute a 95% confidence interval for the interaction parameter and convert it to an odds ratio.

4. (Paper and pencil.) How would you answer Keysar et al.'s research question? ♢

Exercise 6.5 (Sum coding)**.** We refit Keysar et al.'s data using sum-coded predictors:

```
d <- d |>
  mutate(n.language = ifelse(d$language == "Japanese", yes = 0.5, no = -0.5),
         n.framing = ifelse(d$framing == "loss", yes = 0.5, no = -0.5),
         interaction = n.language*n.framing)
keysar.glm <- glm(sure_choice ~ n.language + n.framing + interaction,
                  data = d,
                  family = binomial(link = "logit"))
summary(keysar.glm)$coefficients[, 1:2]
```

```
            Estimate Std. Error
(Intercept)  0.10622    0.19253
n.language  -0.88617    0.38506
n.framing   -0.75144    0.38506
interaction  1.22847    0.77012
```

The estimate for the interaction parameter is the same as before but the estimates for the other parameters have changed. Explain what the parameter estimates for `(Intercept)`, `n.language` and `n.framing` mean. ♢

Remark 6.6 (Confidence intervals using parametric bootstrapping)**.** The confidence intervals for the model parameters rely on large-sample approximations. One option to verify them is to apply a parametric bootstrap. To this end, we generate new outcome data from the fitted model (using `simulate()`) and then refit the model on these new outcome data and store the parameter estimates we're interested in. The following code snippet constructs a 95% confidence interval for the interaction parameter using the parametric bootstrap and the percentile method.

```
B <- 20000
interaction_bs <- vector(length = B)
for (i in 1:B) {
  choice_y <- unlist(simulate(keysar.glm))
  keysar_bs <- glm(choice_y ~ n.language + n.framing + interaction,
                   data = d, family = binomial(link = "logit"))
  interaction_bs[[i]] <- coef(keysar_bs)[[4]]
}
# hist(interaction_bs)
quantile(interaction_bs, probs = c(0.025, 0.975))
```

```
    2.5%    97.5%
-0.26717  2.93981
```

The upper end of this confidence interval is a fair bit higher than the approximate one you can

compute using `confint()`. ♢

Remark 6.7 (p-values in the generalised linear model)**.** We haven't discussed p-values much in this lecture series. But I'd like to really drive home the point that statistical inference in the generalised linear model relies on large-sample approximations and that different methods for computing p-values may give different results.

The first large-sample approximation method is the **Wald method**, which can be obtained using `summary()`. In the present example, the Wald p-value for the interaction term is $p = 0.111$ ($z = 1.60$).

```
summary(keysar.glm)$coefficients
```

```
            Estimate Std. Error  z value Pr(>|z|)
(Intercept)  0.10622    0.19253  0.55171  0.58115
n.language  -0.88617    0.38506 -2.30138  0.02137
n.framing   -0.75144    0.38506 -1.95148  0.05100
interaction  1.22847    0.77012  1.59517  0.11067
```

The second large-sample approximation method is the **likelihood ratio test**. It tends to be more accurate than Wald's method and requires you to fit a null model. In the present example, this would be a model with main effects for language and framing but no interaction effect. A model comparison using the `anova()` function yields $p = 0.108$ ($\chi^2_1 = 2.59$). Note though that this analysis is not an analysis of variance that you may have encountered elsewhere but a so-called analysis of deviance. (We won't go into the concept of deviance, but it is closely related to the ratio of the likelihoods of the models being compared.)

```
null_model <- glm(sure_choice ~ n.language + n.framing,
                  data = d, family = binomial(link = "logit"))
anova(null_model, keysar.glm, test = "LRT")
```

```
Analysis of Deviance Table

Model 1: sure_choice ~ n.language + n.framing
Model 2: sure_choice ~ n.language + n.framing + interaction
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1       118        159
2       117        156  1     2.59     0.11
```

The third large-sample approximation method is the **score test**, which yields $p = 0.109$ ($\chi^2_1 = 2.57$):

```
anova(null_model, keysar.glm, test = "Rao")
```

```
Analysis of Deviance Table

Model 1: sure_choice ~ n.language + n.framing
Model 2: sure_choice ~ n.language + n.framing + interaction
  Resid. Df Resid. Dev Df Deviance  Rao Pr(>Chi)
1       118        159
2       117        156  1     2.59 2.57     0.11
```

Of these three large-sample approximations, the likelihood-ratio test is to be preferred. That said, it relies on the differences in the deviances following a χ^2 distribution under the null hypothesis, which may not be a reasonable assumption for small datasets.

Alternatively, we can use a technique similar to the parametric bootstrap: We generate new outcome data *under the fitted null model* and then refit the full model. We then count in how many cases the estimated interaction term in the models fitted on the null data was at least as extreme as the interaction term that we estimated for the actual data. This proportion serves as a p-value – in our case, $p = 0.118$.

```
B <- 20000
interaction_bs <- vector(length = B)
for (i in 1:B) {
  choice_y <- unlist(simulate(null_model)) # simulate under null!
  keysar_bs <- glm(choice_y ~ n.language + n.framing + interaction,
                   data = d, family = binomial(link = "logit"))
  interaction_bs[[i]] <- coef(keysar_bs)[[4]]
}
# hist(interaction_bs)
# abline(v = coef(keysar.glm)[[4]], col = "blue", lty = "dashed")
mean(abs(interaction_bs) >= abs(coef(keysar.glm)[[4]]))
```

```
[1] 0.11775
```

We could also carry out a bootstrap version of the likelihood ratio test, which yields a p-value of 0.114.

```
B <- 20000
lrt_bs <- vector(length = B)
for (i in 1:B) {
  choice_y <- unlist(simulate(null_model)) # simulate under null!
  # fit both null and full model to data
```

```
  null_bs <- glm(choice_y ~ n.language + n.framing,
                 data = d, family = binomial(link = "logit"))
  keysar_bs <- glm(choice_y ~ n.language + n.framing + interaction,
                 data = d, family = binomial(link = "logit"))
  # compare fitted models
  lrt_bs[[i]] <- anova(null_bs, keysar_bs, test = "LRT")[2, 4]
}
observed_lrt <- anova(null_model, keysar.glm, test = "LRT")[2, 4]
mean(lrt_bs >= observed_lrt)
```

```
[1] 0.1136
```

Note that with the bootstrap procedure, the results will vary slightly from run to run due to the randomness involved in simulating the new observations. ♢

2 Continuous predictors

You can also fit continuous predictors in the generalised linear model. To illustrate the procedure, we'll take a look at how well one of the participants in Vanhove & Berthele (2013) translated 181 words from languages she did not know depending on the orthographic overlap (expressed on a scale from 0 to 10) between these words and their translations.[3]

Let's read in the data and draw a scatterplot including a scatterplot smoother (Figure 6.3). Even though this smoother doesn't take into account that the data are binary (notice how it extends beyond the $[0, 1]$ interval), it highlights a monotonic association between orthographic overlap and the participant's translation success.

```
library(here)
d <- read_csv(here("data", "VanhoveBerthele2013_oneparticipant.csv"))

ggplot(data = d,
       aes(x = Overlap,
           y = Correct)) +
  geom_point(shape = 1,
             # vertical jittering to better show the data points
             position = position_jitter(width = 0, height = 0.02)) +
  # scatterplot smoother without confidence band
  geom_smooth(se = FALSE)
```

[3] If we wanted to model the data of all participants, we would need logistic mixed-effects models, which are well outside the scope of this lecture.

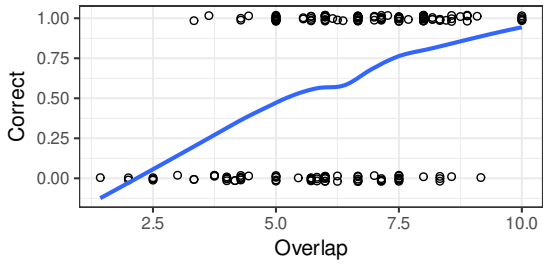

Figure 6.3: There seems to be a monotonous relationship between the degree of overlap and whether the participant correctly translated the word. Note, though, that scatterplot smoother extends beyond the $[0, 1]$ interval on the y axis.

We can fit these data in a logistic model:

```
translation.glm <- glm(Correct ~ Overlap, data = d,
                       family = binomial(link = "logit"))
summary(translation.glm)$coefficients
```

```
            Estimate Std. Error z value   Pr(>|z|)
(Intercept) -3.31321    0.73401 -4.5138 6.3664e-06
Overlap      0.59443    0.11436  5.1980 2.0143e-07
```

Other than that the estimates are in log-odds, there is nothing new here:

- The `(Intercept)` estimates the log-odds of a correct translation if the degree of orthographic `Overlap` is 0. Expressed in odds, there is a 0.037–to–1 chance of a correct translation for words with this little overlap; or about a probability of 0.035.

  ```
  exp(-3.31)
  ```

  ```
  [1] 0.036516
  ```

  ```
  plogis(-3.31)
  ```

  ```
  [1] 0.03523
  ```

  Note that there are no words with 0 overlap in this dataset. If we wanted to, we could centre the overlap variable so that the intercept estimate becomes more informative.

- The odds ratio of a correct translation for words with an orthographic overlap of 1 vs. for words with an orthographic overlap of 0 is about 1.8. That is, the odds of correctly translating a word with an orthographic overlap of 1 are estimated to be about 1.8 as large as those of correctly translating a word with an orthographic overlap of 0.

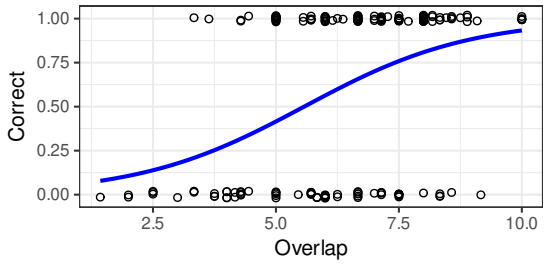

Figure 6.4: The same data, but this time the trend line is based on our logistic model.

```
exp(0.59)

[1] 1.804
```

- To obtain the estimated probability of correctly translating a word given its overlap, first compute the relevant log-odds and then convert them to the probability scale. For instance, the estimated probability of correctly translating a word with an orthographic overlap of 7.23 is about 73%.

```
plogis(-3.313 + 0.594*7.23)

[1] 0.72743
```

Let's draw a plot that shows how, according to the model, the success probability varies with the degree of orthographic overlap (Figure 6.4). Note that the plotted curve isn't straight. This is because we modelled the data linearly in log-odds space; by converting the result back to probability space via the logistic function, the trend line becomes nonlinear. That is, we could not *not* have found a nonlinear trend line. A full-fledged tutorial on drawing trend lines based on models is available from `https://janhove.github.io/visualise_uncertainty/`.

```
# quick function for computing the probabilities
model_to_prob <- function(x, model) {
  plogis(coef(model)[[1]] + coef(model)[[2]]*x)
}

ggplot(data = d,
       aes(x = Overlap,
           y = Correct)) +
  geom_point(shape = 1,
             position = position_jitter(width = 0, height = 0.02)) +
  stat_function(fun = model_to_prob, args = list(model = translation.glm),
                colour = "blue", linewidth = 1)
```

For further guidance on visualising logistic regression models, see `https://janhove.github.io/visualise_uncertainty/`.

Recommended reading

We've only scratched the surface. While we've covered the logit and the identity link function, other link functions exist (e.g., probit, complementary-log-log). We haven't mentioned overdispersion at all, either. Further, some useful extensions of logistic (binary) regression include multinomial regression and different kinds of ordinal regression. Agresti (2002) is the go-to reference for categorical data analysis.

References

Agresti, Alan. 2002. *Categorical data analysis*. Hoboken, NJ: Wiley 2nd edn.

Hellevik, Ottar. 2009. Linear versus logistic regression when the dependent variable is a dichotomy. *Quality & Quantity* 43. 59–74. doi:10.1007/s11135-007-9077-3.

Huang, Francis L. 2019. Alternatives to logistic regression models in experimental studies. *Journal of Experimental Education* doi:10.1080/00220973.2019.1699769.

Jaeger, T. Florian. 2008. Categorical data analysis: Away from ANOVAs (transformation or not) and towards logit mixed models. *Journal of Memory and Language* 59(4). 434–446. doi:10.1016/j.jml.2007.11.007.

Keysar, Boas, Sayuri L. Hayakawa & Sun Gyu An. 2012. The foreign-language effect: Thinking in a foreign tongue reduces decision biases. *Psychological Science* 23(6). 661–668. doi:10.1177/0956797611432178.

Tversky, Amos & Daniel Kahneman. 1981. The framing of decisions and the psychology of choice. *Science* 211(4481). 453–458. doi:10.1126/science.7455683.

Vanhove, Jan & Raphael Berthele. 2013. Factoren bij het herkennen van cognaten in onbekende talen: algemeen of taalspecifiek? *Taal & Tongval* 65. 171–210. doi:10.5117/TET2013.2.VANH.