The general linear model
Lecture 3 – Group differences

Jan Vanhove

https://janhove.github.io

Ghent, 14–16 July 2025

The general linear model can be used to model differences between two or more groups with respect to some continuous outcome variable. We'll first take a look at how differences between two groups can be modelled. Then we'll deal with differences between three or more groups. Finally, we'll discuss how data from factorial designs can be analysed.

1 Differences between two groups

Our first example in this lecture is from the field of social psychology.[1] Caruso et al. (2013) reported that American participants agree more strongly with statements that justify the US social system if they are reminded of money. Their participants responded to eight statements such as *Everyone has a fair shot at wealth and happiness* on a 7-point Likert scale. The eight responses per participant were averaged and submitted to analysis. For half of the participants, a faded but still visible dollar note was shown on the computer screen as they filled out the questionnaire; for the other half, this image was completely blurred. The authors reported that the mean system justification scores tended to be higher when a dollar note was visible than when it was blurred. Klein et al. (2014) attempted to replicate this finding in a 36 new samples. We'll work with the replication data from one such new sample:

```
library(tidyverse)
library(here)
theme_set(theme_bw())
d <- read_csv(here("data", "Klein2014_money_abington.csv"))
```

A sensible default choice when plotting a group comparison on a continuous outcome is to

[1] Studies with group comparisons in language-related fields either seem to be more complex than *just* a group comparison or don't share their data. But if you know of a suitable dataset from the language sciences, please let me know!

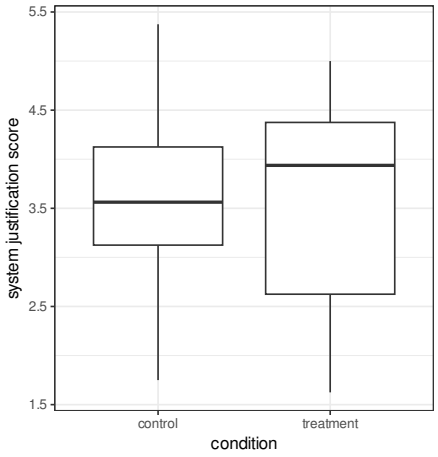

Figure 1: Comparison of the system justification scores in both conditions in Klein et al.'s (2014) replication of Caruso et al. (2013, Experiment 1). Data from the Abington sample.

use **boxplots**. The standard implementation is shown in Figure 1, but I prefer to overlay the individual data points, too. Boxplots can sometimes deceive, and adding the individual data points helps both us and our readers gauge what the data actually look like; see Figure 2; also see Weissgerber et al. (2015).

```
# standard boxplots
p_boxplot <- ggplot(data = d,
                    aes(x = MoneyGroup,
                        y = Sysjust)) +
  geom_boxplot() +
  xlab("condition") +
  ylab("system justification score")
p_boxplot
```

```
# boxplots with individual data points added
p_boxplotdeluxe <- ggplot(data = d,
                          aes(x = MoneyGroup,
                              y = Sysjust)) +
  # don't plot outliers twice
  geom_boxplot(outlier.shape = NA) +
  # add some random noise to the x position of the points to reduce overlap
  geom_point(shape = 1,
             position = position_jitter(width = 0.2, height = 0)) +
  xlab("bank note") +
```

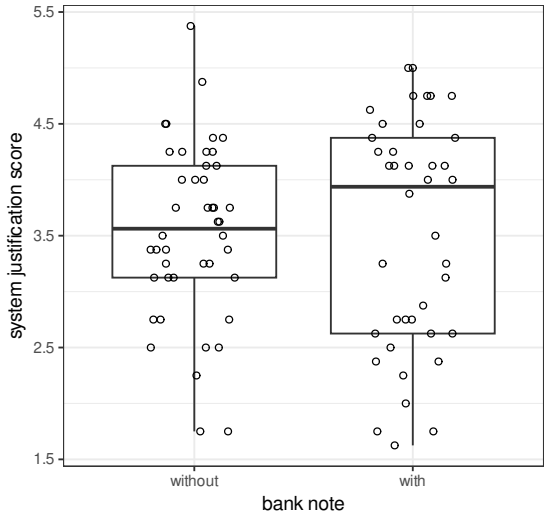

Figure 2: The same data but with the individual data points overlaid.

```
  scale_x_discrete(labels = c("without", "with")) +
  ylab("system justification score")
p_boxplotdeluxe
```

Note the setting `outlier.shape = NA` in the `geom_boxplot()` command. This suppresses the drawing of potential outliers in the boxplot. If we didn't do this, we would be plotting such potential outliers twice: once as part of the boxplot, and once as individual data points. (There are no such outliers in this dataset, though.) Moreover, the data points are jittered horizontally so that they don't overlap as much.

Using the functions provided by the tidyverse packages, we can construct a table with the basic descriptive statistics for both of the experiment's conditions.

```
d |>
  group_by(MoneyGroup) |>
  summarise(n = n(),
            mean = mean(Sysjust),
            median = median(Sysjust),
            stdev = sd(Sysjust))

# A tibble: 2 x 5
  MoneyGroup     n  mean median stdev
  <chr>      <int> <dbl>  <dbl> <dbl>
1 control       44  3.53   3.56 0.781
2 treatment     40  3.53   3.94 1.02
```

Let's turn to the matter of modelling these data in a general linear model. The basic equation is the same as in Lecture 2:

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \tag{1}$$

for $i = 1, 2, \ldots, n$. This time, x_i is a **dummy variable** that encodes whether the i-th participant is part of one group or the other. There are two common methods for encoding group membership using dummy variables when there are just two groups: treatment coding and sum coding.

In **treatment coding**, we designate one group as the 'treatment group' and the other as the 'control group'. Then, we set $x_i = 1$ if the i-th participant is part of the treatment group and $x_i = 0$ if the i-th participant is part of the control group. In our current example, the treatment group has already been identified, and we just need to add another variable to the tibble that encodes this information numerically:

```
d$n.MoneyGroup <- ifelse(d$MoneyGroup == "treatment", yes = 1, no = 0)
```

Now, we can use the newly created dummy variable like the more or less continuous predictor variable in Lecture 2:

```
money.lm <- lm(Sysjust ~ n.MoneyGroup, data = d)
coef(money.lm)
```

```
 (Intercept) n.MoneyGroup
   3.5340909   -0.0059659
```

Uncertainty estimates can be obtained just like in Lecture 2. Here, we'll skip straight to the estimated standard errors and confidence intervals based on the assumption that the ε_i were sampled i.i.d. from a normal distribution. But there's an exercise towards the end of this lecture where you verify the results obtained using a bootstrap that doesn't assume identical, normal error distributions.

```
summary(money.lm)$coefficients
```

```
               Estimate Std. Error   t value   Pr(>|t|)
(Intercept)   3.5340909    0.13633 25.923055 2.8981e-41
n.MoneyGroup -0.0059659    0.19756 -0.030198 9.7598e-01
```

```
confint(money.lm)
```

```
                2.5 %  97.5 %
(Intercept)   3.26289 3.80529
n.MoneyGroup -0.39898 0.38705
```

What do these parameter estimates actually refer to? Inspecting Equation 1, we notice that if

$x_i = 0$, then

$$y_i = \widehat{\beta}_0 + \widehat{\varepsilon}_i,$$

or, put differently,

$$\widehat{y}_i = \widehat{\beta}_0.$$

That is, the estimated intercept term shows us the estimated conditional mean for the group coded as 0, which should come as no surprise given what we've seen in Lecture 2. From Lecture 2, we also know that $\widehat{\beta}_0 + \widehat{\beta}_1$ gives us the estimated conditional mean for the group coded as 1. The estimated β_1 parameter, then, tells us how much higher the estimated mean for the 1-group is compared to the 0-group. In our example,

$$y_i = 3.53 - 0.006x_i + \widehat{\varepsilon}_i.$$

So from the output above, we can glean that the mean of the control group is 3.53 ± 0.14, whereas the estimated mean difference between the treatment and the control groups is -0.006 ± 0.20. It is usually the latter estimate that is relevant.

The second commonly used method for coding dummy variables is **sum coding**. In sum coding, the value of the dummy variable is $+0.5$ for the participants in one condition and -0.5 for the participants in the other conditions. (Alternatively, ± 1 is used.)

```
d$n.MoneyGroup2 <- ifelse(d$MoneyGroup == "treatment", yes = 0.5, no = -0.5)
money.lm2 <- lm(Sysjust ~ n.MoneyGroup2, data = d)
summary(money.lm2)$coefficients
```

```
                Estimate Std. Error   t value   Pr(>|t|)
(Intercept)    3.5311080    0.09878 35.747017 8.9152e-52
n.MoneyGroup2 -0.0059659    0.19756 -0.030198 9.7598e-01
```

```
confint(money.lm2)
```

```
                  2.5 %  97.5 %
(Intercept)    3.33460 3.72761
n.MoneyGroup2 -0.39898 0.38705
```

We can use these estimates to compute the conditional means for the treatment and control groups. The estimated regression equation now is given by

$$y_i = 3.53 - 0.006x_i + \widehat{\varepsilon}_i.$$

Due to rounding, this is the same equation as before. Note, however, that the actual estimate for the intercept is slightly different between the two models. To obtain the estimate for the conditional mean for the treatment group, we use $x_i = 0.5$; to obtain the estimated conditional

mean for the control group, we use $x_i = -0.5$:

$$\widehat{y}_{\text{treatment}} = 3.53 - 0.006 \cdot 0.5$$

and

$$\widehat{y}_{\text{control}} = 3.53 + 0.006 \cdot 0.5.$$

The difference between these estimates is exactly the parameter estimate for the slope:

$$\widehat{y}_{\text{treatment}} - \widehat{y}_{\text{control}} = (3.53 - 0.006 \cdot 0.5) - (3.53 + 0.006 \cdot 0.5) = 0.006.$$

As for the interpretation of the intercept estimate, consider the following:

$$\begin{aligned}
\widehat{\beta}_0 &= \frac{\widehat{\beta}_0 + \widehat{\beta}_0}{2} \\
&= \frac{(\widehat{\beta}_0 + \widehat{\beta}_1 \cdot 0.5) + (\widehat{\beta}_0 - \widehat{\beta}_1 \cdot 0.5)}{2} \\
&= \frac{\widehat{y}_{\text{treatment}} + \widehat{y}_{\text{control}}}{2}.
\end{aligned}$$

That is, when using sum-coding, the estimated intercept corresponds to the average of the conditional means for both groups, that is, to the **grand mean**.

Tip 3.1. If you want to use treatment coding, you don't actually have to code the dummy variables yourself – R can take care of this for you. However, by default, R codes such variables in alphabetical order. For instance, if your two groups are called 'English' and 'Dutch', the Dutch group would be coded as 0 and would be incorporated into the intercept estimate. But if your two groups are called 'Engels' and 'Nederlands', the English group would be coded as 0. I recommend you code your dummy variables by hand, so you know what's going on. ♢

Exercise 3.2. Refer back to the `money.lm` model on page 4. How would the parameter estimates change if we coded the treatment group as 0 and the control group as 1? Try to answer this question without using the computer. ♢

Exercise 3.3. Refer back to the `money.lm2` model on page 5. How would the parameter estimates change if we coded the treatment group as 1 and the control group as -1? How could you interpret the slope estimate? Try to answer these questions without using the computer. ♢

Exercise 3.4. One of the other findings that Klein et al. (2014) sought to replicate was the gambler's fallacy as studied by Oppenheimer et al. (2009). Klein et al. (2014) summarise this experiment as follows:

> "Oppenheimer & Monin (2009) investigated whether the rarity of an independent, chance observation influenced beliefs about what occurred before that event. Participants imagined that they saw a man rolling dice in a casino. In one condition, participants imagined witnessing three dice being rolled and all came up 6's. In a second condition two came up 6's and one came up 3. In a third condition, two dice were rolled and both came up 6's. All participants then estimated, in an open-ended format, how many times the man had rolled the dice before they entered the room to watch him. Participants estimated that the man rolled dice more times when they had seen him roll three 6's than when they had seen him roll two 6's or two 6's and a 3. For the replication, the condition in which the man rolls two 6's was removed leaving two conditions."

You can find the data for this replication study in `Klein2014_gambler.csv`. Analyse these data in light of the research question, but only consider the data from the `ufl` sample. Summarise your findings in two to three sentences.

To get you started, here's how you can read in the data from just the `ufl` sample:

```
gambler <- read_csv(here("data", "Klein2014_gambler.csv")) |>
  filter(Sample == "ufl")
```

Inspect the dataset by typing its name at the R prompt, or use `View(gambler)`. ♢

2 Differences between more than two groups

In Vanhove (2017), I investigated how German gender assignment by Belgian speakers of Dutch is affected by their native dialect. For example, I wanted to find out if speakers of Belgian Dutch dialects would more often pick the German masculine article *der* for *Knie* ('knee') if, in their own Dutch dialect, *knie* was masculine rather than feminine. (Belgian Dutch dialects differ somewhat in terms of gender assignment, though speakers don't seem to be really aware of this. German *Knie* is neuter, by the way.) It turned out that the German gender assignments of the informants hardy betrayed any influence of their own dialect at all. In a follow-up study (Vanhove, 2019), I tested my hunch that speakers of a Belgian Dutch dialect don't rely on their own dialect when assigning gender to German nouns because they lack metalinguistic knowledge about grammatical gender in their own dialect. To this end, I devised a small experiment with three conditions:

- Strategy: In the first condition, participants were furnished with a simple strategy for figuring out the grammatical gender of nouns in their own dialect.
- Information: In the second condition, participants were told that their own dialect, like German but unlike Standard Dutch, makes a three-way adnominal gender distinction.

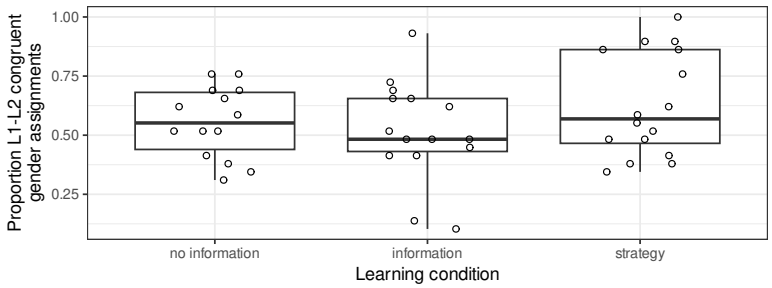

Figure 3

They weren't told how they could identify the grammatical gender of nouns in their own dialect, however.

- No information: Participants in this condition were provided with accurate information about some grammatical aspect of their dialect that was irrelevant to the task at hand.

Then, the participants assigned German gender to 29 German nouns with Dutch cognates, and I tallied how often their German gender assignments were congruent with the nouns' gender in the participants' native dialect. I expected that participants in the strategy condition would outperform those in the other two conditions.

Let's read in the data and draw some boxplots (Figure 3). Note the use of the `limits` parameter in the `scale_x_discrete()` call to change the order of the boxplots to something more meaningful than the default alphabetical one.

```
d <- read_csv(here("data", "Vanhove2018.csv"))
ggplot(d,
       aes(x = Condition,
           y = ProportionCongruent)) +
  geom_boxplot(outlier.shape = NA) +
  geom_point(shape = 1,
             position = position_jitter(width = 0.2, height = 0)) +
  scale_x_discrete(limits = c("no information", "information", "strategy")) +
  xlab("Learning condition") +
  ylab("Proportion L1-L2 congruent\ngender assignments")
```

A descriptive numerical summary is obtained easily enough:

```
d |>
  group_by(Condition) |>
  summarise(N = n(),
            Mean = mean(ProportionCongruent),
```

```
            Median = median(ProportionCongruent),
            StDev = sd(ProportionCongruent))

# A tibble: 3 x 5
  Condition          N  Mean Median StDev
  <chr>          <int> <dbl>  <dbl> <dbl>
1 information       15 0.517  0.483 0.213
2 no information    14 0.554  0.552 0.150
3 strategy          16 0.627  0.569 0.219
```

In the previous section, we recoded a variable with two levels (treatment and control) as a binary dummy variable. When we have a variable with k levels, we need $k-1$ binary dummy variables to code all conditions unambiguously. In our present example, we will use treatment coding to identify both whether participants were assigned to the strategy condition and whether they were assigned to the information condition; if both dummy variables read 0, then the participant was assigned to the no information condition:

```
d$Strategy <- ifelse(d$Condition == "strategy", 1, 0)
d$Information <- ifelse(d$Condition == "information", 1, 0)

# quick check
xtabs(~ Strategy + Condition, data = d)

        Condition
Strategy information no information strategy
       0          15             14        0
       1           0              0       16

xtabs(~ Information + Condition, data = d)

           Condition
Information information no information strategy
          0           0             14       16
          1          15              0        0
```

The general linear model can deal with several predictors: The principles from the previous lectures carry over, just with $d = 3$ β parameters rather than $d \in \{1, 2\}$. So we can add both dummy variables to the model. The resulting model equation looks like this:

$$y_i = \beta_0 + \beta_1 \cdot x_{1,i} + \beta_2 \cdot x_{2,i} + \varepsilon_i.$$

$x_{1,i}$ represents the value of the i-th participant on the first dummy variable; $x_{2,i}$ represents their value on the second dummy variable.

```
mod.lm <- lm(ProportionCongruent ~ Information + Strategy, data = d)
mod.lm


Call:
lm(formula = ProportionCongruent ~ Information + Strategy, data = d)

Coefficients:
(Intercept)  Information     Strategy
     0.5542      -0.0369       0.0730
```

The interpretation of these parameter estimates is analogous to what we discussed in the previous section. The intercept estimate is the conditional mean for the group that was coded as 0 on both dummy variables, i.e., the no information condition. The estimate for information is the difference between the estimated conditional means for the information and the no information conditions; the estimate for strategy is the difference between the estimated conditional means for the strategy and the no information conditions. You can verify this by reconstructing the condition means in the descriptive statistics using the model output.

We can estimate the uncertainty about these parameter estimates using bootstrap methods (see the exercise below). If we're comfortabe assuming that the residuals were all drawn from the same normal distribution, we can use `summary()` as before to obtain the estimated standard errors:

```
summary(mod.lm)$coefficients

             Estimate Std. Error  t value   Pr(>|t|)
(Intercept)  0.554187   0.053006 10.45526 2.9207e-13
Information -0.036946   0.073701 -0.50129 6.1878e-01
Strategy     0.072968   0.072581  1.00533 3.2049e-01
```

Confidence intervals can be computed using `confint()`. The confidence interval about the intercept estimate isn't of too much interest here, but it is computed by default. That doesn't mean you have to report it in your papers, though. (Relevance!)

```
confint(mod.lm, level = 0.90)

                 5 %     95 %
(Intercept)  0.46503 0.643340
Information -0.16091 0.087016
Strategy    -0.04911 0.195046
```

While we used treatment coding here, other coding systems exist, and these may be more

directly useful for you, depending on what you want to find out. For instance, we could code the dummy variables in such a way that $\widehat{\beta}_2$ estimates the difference between the conditional means for the third and second condition rather than between those for the third and first condition. Schad et al. (2020) explain how this can be accomplished; also see the entry on my blog *Tutorial: Obtaining directly interpretable regression coefficients by recoding categorical predictors*. This technique is extremely useful as it often enables you to read off the answers to your research questions directly from the model output, as opposed to your having to piece together an answer based on several parameter estimates.

Tip 3.5. Know what your parameter estimates refer to. It is vital that you know what the numbers in the output of your model literally mean before you try to interpret them in terms of your subject matter. ♢

Exercise 3.6 (Semi-parametric bootstrap without homoskedasticity assumption)**.** In the analyses in this lecture, we estimated standard errors and constructed confidence intervals based on the assumption that the residuals were i.i.d. normal. We can check whether we obtain similar results when we make different assumptions. To that end, we can run a semi-parametric bootstrap in which the bootstrapped residuals in a given condition are only sampled from that distribution. This way, we don't assume that the residuals are drawn from a normal distribution, and we acknowledge the possibility that the residuals in different conditions are drawn from different distributions.

The code below is similar to that of the previous semi-parametric bootstraps; the only thing that was added is the `group_by(Condition)` call in the *for*-loop. This splits up the dataset into groups by `Condition` so that the resampling of the `Residual` values in line 20 only happens within each condition. In other words, a residual value in the information condition can only be reassigned to another observation in the information condition, and similary for the other conditions.

Copy this code (ideally by typing rather than copy-pasting it; you'll learn more this way) and run it. How would the conclusions you draw from this analysis differ from the ones you would draw from the analysis in the lecture?

```
# Read in data, fit model
d <- read_csv(here("data", "Vanhove2018.csv"))
d$Strategy <- ifelse(d$Condition == "strategy", 1, 0)
d$Information <- ifelse(d$Condition == "information", 1, 0)
mod.lm <- lm(ProportionCongruent ~ Information + Strategy, data = d)

# Extract predictions, residuals
d$Prediction <- predict(mod.lm)
```

```r
d$Residual <- resid(mod.lm)

# Semi-parametric bootstrap w/o homoskedasticity assumption
n_bootstrap <- 20000
bs_b <- matrix(nrow = n_bootstrap, ncol = 3)

for (i in 1:n_bootstrap) {
  d <- d |>
    group_by(Condition) |>
    mutate(bs_outcome = Prediction + sample(Residual, replace = TRUE))
  bs_mod <- lm(bs_outcome ~ Information + Strategy, data = d)
  bs_b[i, ] <- coef(bs_mod)
}

apply(bs_b, 2, sd)
apply(bs_b, 2, quantile, prob = c(0.025, 0.975))
```

Adapt this code in order to double-check the confidence intervals from the money priming example. ♢

References

Caruso, Eugene M., Kathleen D Vohs, Brittani Baxter & Adam Waytz. 2013. Mere exposure to money increases endorsement of free-market systems and social inequality. *Journal of Experimental Psychology: General* 142(2). 301–306. doi:10.1037/a0029288.

Klein, Richard A., Kate A. Ratliff, Michelangelo Vianello, Reginald B. Adams Jr., tpán Bahník, Michael J. Bernstein, Konrad Bocian et al. 2014. Investigating variation in replicability: A many labs replication project. *Social Psychology* 45(3). 142–152. doi:10.1027/1864-9335/a000178.

Oppenheimer, Daniel M., Tom Meyvis & Nicolas Davidenko. 2009. Instructional manipulation checks: Detecting satisficing to increase statistical power. *Journal of Experimental Social Psychology* 45. 867–872. doi:10.1016/j.jesp.2009.03.009.

Oppenheimer, Daniel M & Benoît Monin. 2009. The retrospective gambler's fallacy: Unlikely events, constructing the past, and multiple universes. *Judgment and Decision Making* 4(5). 326–334.

Schad, Daniel J., Shravan Vasishth, Sven Hohenstein & Reinhold Kliegl. 2020. How to capitalize

on a priori contrasts in linear (mixed) models: A tutorial. *Journal of Memory and Language* 110. doi:10.1016/j.jml.2019.104038.

Vanhove, Jan. 2017. The influence of standard and substandard Dutch on gender assignment in second language German. *Language Learning* 67(2). 431–460. doi:10.1111/lang.12230.

Vanhove, Jan. 2019. Metalinguistic knowledge about the native language and language transfer in gender assignment. *Studies in Second Language Learning and Teaching* 9(2). 397–419. doi:10.14746/ssllt.2019.9.2.7.

Weissgerber, Tracey L., Natasa M. Milic, Stacey J. Winham & Vesna D. Garovic. 2015. Beyond bar and line graphs: Time for a new data presentation paradigm. *PLOS Biology* 13(4). e1002128. doi:10.1371/journal.pbio.1002128.