Examples of probabilistic analysis of ordinal survey data

An example of analyzing ordinal data with a probabilistic model

```
# Load libraries needed for the analysis
library(tidyverse)
library(brms)
library(tidybayes)
library(haven)
library(broom.mixed)
library(kableExtra)
```

```
# Loading the data
census_data_all <- read_sav("../../input/data_processed/combined_census_data.sav") %>%
  filter(exclude == 0) %>%
  mutate(Q2 = haven::as_factor(Q2),
         Age_group = case_when(
          Age < 18 ~ "<18",
          Age > 17 & Age < 30 ~ "18-29",
          Age > 29 & Age < 50 ~ "30-49",
          Age > 49 & Age < 65 ~ "50-64",
          Age > 64 ~ ">64"
        ))
```

This document provides an overview how ordinal variables can be analyzed using probabilistic models.

Look at the data

The live music audience census has several ordinal variables. For the examples in this document, we will focus on the question "How would you rate the live music scene in your city?", which is coded as Q11 in the data. To begin let's focus only on the Helsinki data.

```
# Seperating the Helsinki data
helsinki_data <- census_data_all %>%
  filter(City == "Helsinki")
```

Let's take a look how people in Helsinki answered to the question.

```
helsinki_data %>%
  count(Q11) %>%
  na.omit()
```

```
# A tibble: 4 x 2
  Q11                n
  <dbl+lbl>      <int>
1 2 [Bad]            1
2 3 [Average]       27
3 4 [Good]         108
4 5 [Very good]     20
```

We can see that no one responded with option one, and only one person responded with option two. Using frequentist model here can lead to problems with providing estimates for these two categories. Probabilistic models provide better estimates in this scenario.

Fitting an ordinal regression model

There are several regression models that can be used to analyze ordinal data, but the most common is cumulative logistic ordinal regression model. This is the model that is used here. The probabilistic model can be fitted using `brm` function from `brms` package. This function allows fitting several different regression models in the probabilistic framework. Let's start off by fitting a model with gender, coded as Q2, and age group as explanatory variables.

```
# Changing the data type to work with the model
helsinki_data <- helsinki_data %>%
  mutate(Q11 = as.numeric(Q11))
# Fitting the model
```

```r
fit1 <- brm(Q11 ~ Q2 + Age_group, data = helsinki_data,
            family = cumulative("logit"), seed = 2025,
            prior = prior(normal(-1.39,1), class = "Intercept", coef = "1") +
                        prior(normal(-0.41,1), class = "Intercept", coef = "2") +
                        prior(normal(0.41,1), class = "Intercept", coef = "3") +
                        prior(normal(1.39,1), class = "Intercept", coef = "4") +
                        prior(normal(0,2), class = "b"), backend = "cmdstanr",
            control = list(adapt_delta = 0.95), silent = 0, refresh = 0)
```

In the code above the family command is used to specify the model family, cumulative regression model with logistic link function.

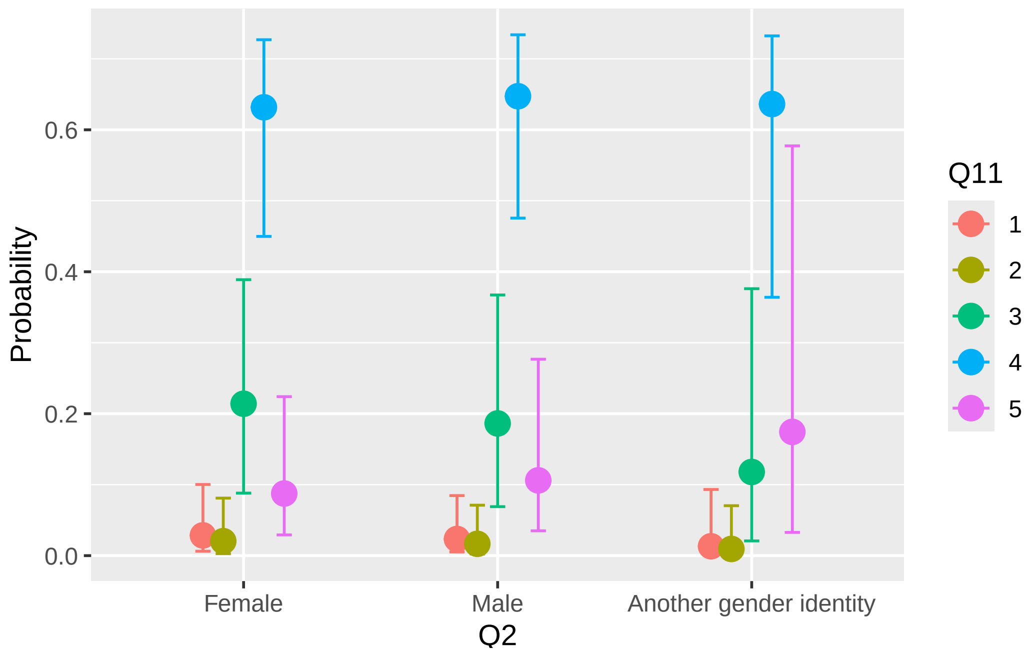
```r
# Looking at the results of the model fit
tidy(fit1) %>%
  select(-c(effect, component, group)) %>%
  kable()
```

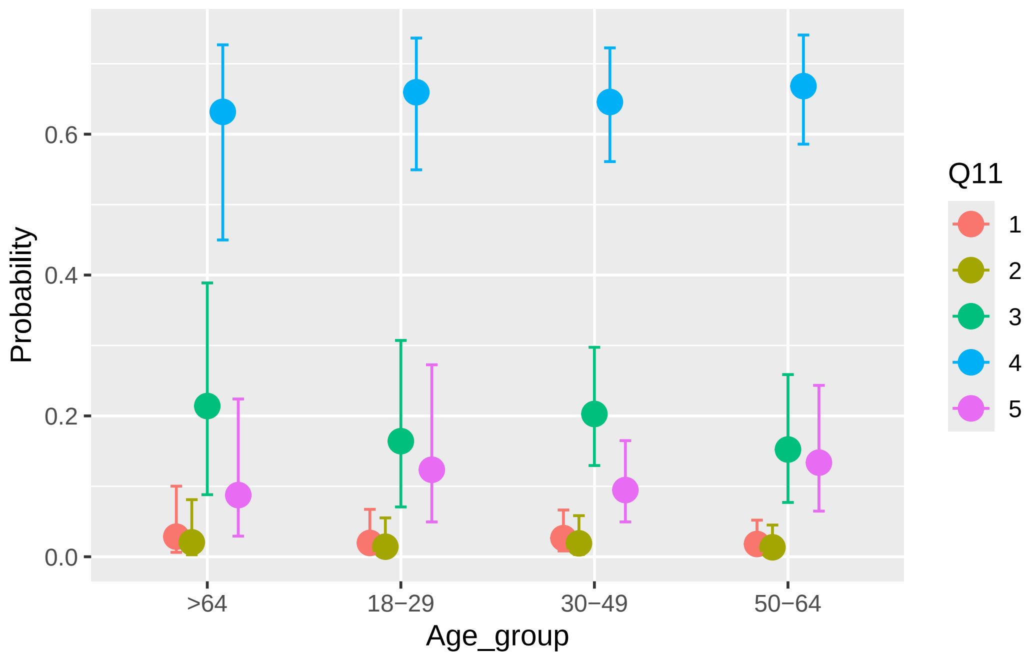
term	estimate	std.error	conf.low	conf.high
(Intercept)[1]	-3.5414761	0.7236487	-5.0693625	-2.1946315
(Intercept)[2]	-2.8979859	0.6265104	-4.1662688	-1.6868680
(Intercept)[3]	-0.9860508	0.5422130	-2.1125982	0.0616879
(Intercept)[4]	2.3592383	0.5724670	1.2424162	3.5023658
Q2Male	0.2173960	0.3418935	-0.4354241	0.8843705
Q2Anothergenderidentity	0.7966127	0.7896695	-0.6961592	2.3837458
Age_group18M29	0.3910522	0.6798590	-0.9165632	1.7245062
Age_group30M49	0.0929566	0.5704468	-1.0514322	1.2428862
Age_group50M64	0.4824873	0.6083028	-0.7557084	1.6716872

The interesting values here are the estimates for the covariate effects, that is `Q2Male`, `Age_group18M29`, etc. These tell how much these groups differ from the reference category for that categorical variable, female for gender and over 64 for age group. All the estimates are close to zero and the 95% intervals include zero, so the explanatory variables don't seem to have a strong effect on the answer.

The differences between the different levels of the explanatory variables can also be assessed visually. `brms` includes a useful function called `conditional_effects()` for this purpose. It is important to specify `categorical = TRUE` when working with ordinal models.

```r
# Plotting predictions at the different levels of explanatory variables
conditional_effects(fit1, categorical = TRUE)
```

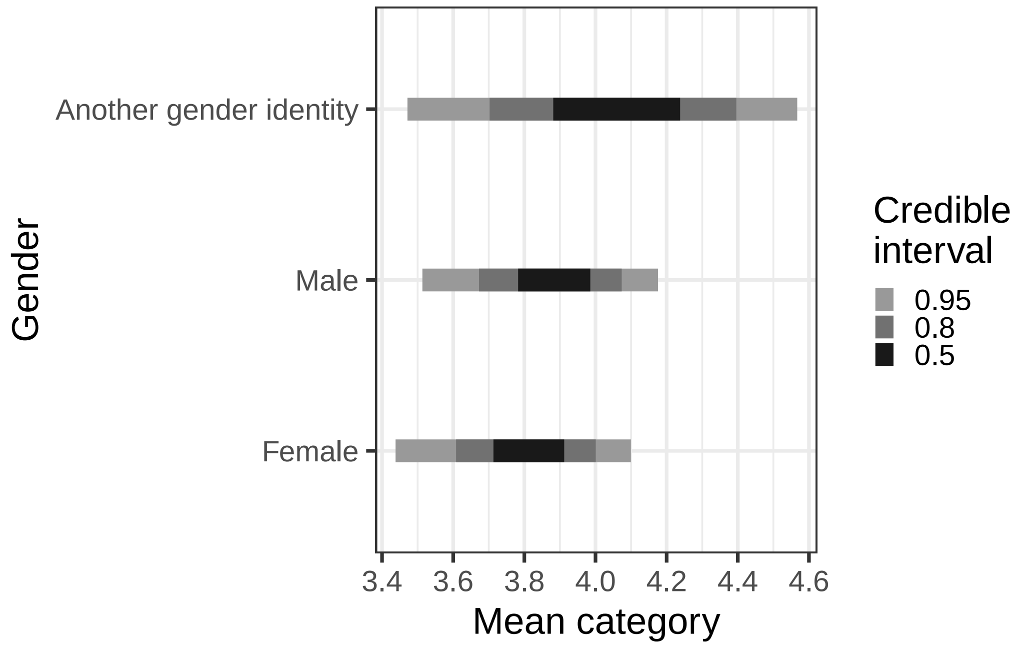




There does not seem to be huge differences in the answering patters between different genders or age groups.

The explanatory variables can also be compared using predicted mean category.

```
# Mean category
expand.grid(Q2 = c("Female", "Male", "Another gender identity"),
                         Age_group = c("30-49", "18-29", "50-64", ">64")) %>%
  add_epred_draws(object = fit1) %>%
  mutate(mean_cat = as.numeric(.category) * .epred) %>%
  group_by(.draw, .row, Q2) %>%
  summarise(mean_cat = sum(mean_cat)) %>%
  ggplot(aes(x = mean_cat, y = Q2)) +
  stat_interval() +
  scale_color_grey(end = 0.1, start = 0.6, name = "Credible\ninterval") +
  theme_bw(base_size = 14) +
  theme(legend.key.size = unit(0.3, "cm")) +
  labs(x = "Mean category", y = "Gender")
```



Priors used

In the model fitting above there was no comment on setting the priors. Priors are an important part of probabilistic models. A typical regression model has one intercept term, but the cumulative model has $J-1$ intercept terms, where J is the number of answer categories. These intercepts can be set a common prior distribution, but better option is to have a separate one for them. One way to set the prior distributions is to assume that all the categories have an equal probability, i.e. 20% when there are five categories. The hyperparameter values can be obtained sing logit function and cumulative probabilities.

```
# Calculating priors hyperparameter valeus
tibble(prob = cumsum(c(0.2, 0.2, 0.2, 0.2, 0.2))) %>%
  mutate(hyperparameter_value = log(prob/(1-prob)))
```

```
# A tibble: 5 x 2
   prob hyperparameter_value
  <dbl>                <dbl>
1   0.2               -1.39
2   0.4               -0.405
3   0.6                0.405
4   0.8                1.39
5   1                Inf
```

From above we get the values for the hyperparameters.

The prior used for the coefficient effects is a normal distribution with expected values of zero and standard derivation of 2. This implies that the effects are distributed around zero, but non-zero effects are also likely.

Model comparison

Probabilistic models offer an easy way to do model comparison. This is done with leave-it-one-out cross-validation method, which is implemented in `loo()` function.

Let's start by fitting a model only gender as a explanatory variable.

```
# Model without age group as explanatory variable
fit2 <- brm(Q11 ~ Q2, data = helsinki_data,
            family = cumulative("logit"), seed = 2025,
            prior = prior(normal(-1.39,1), class = "Intercept", coef = "1") +
                        prior(normal(-0.41,1), class = "Intercept", coef = "2") +
                        prior(normal(0.41,1), class = "Intercept", coef = "3") +
                        prior(normal(1.39,1), class = "Intercept", coef = "4") +
                        prior(normal(0,2), class = "b"), backend = "cmdstanr",
            control = list(adapt_delta = 0.99), silent = 0, refresh = 0)
```

```
# Model output
tidy(fit2) %>%
  select(-c(effect, component, group)) %>%
  kable()
```

term	estimate	std.error	conf.low	conf.high
(Intercept)[1]	-3.7109548	0.5015618	-4.7891662	-2.8018975
(Intercept)[2]	-3.0850907	0.3757582	-3.8406152	-2.3887400
(Intercept)[3]	-1.1803572	0.2234786	-1.6341595	-0.7468280
(Intercept)[4]	2.1128708	0.2774380	1.5862583	2.6775615
Q2Male	0.2448034	0.3244363	-0.3892906	0.8815063
Q2Anothergenderidentity	0.7079855	0.7701200	-0.8282752	2.1877748

From the output we see that the gender is now the only explanatory variable. We can now use `loo()` to compare these two models to each other.

```
# Comparing the two models to each other
loo(fit1, fit2)$diffs
```

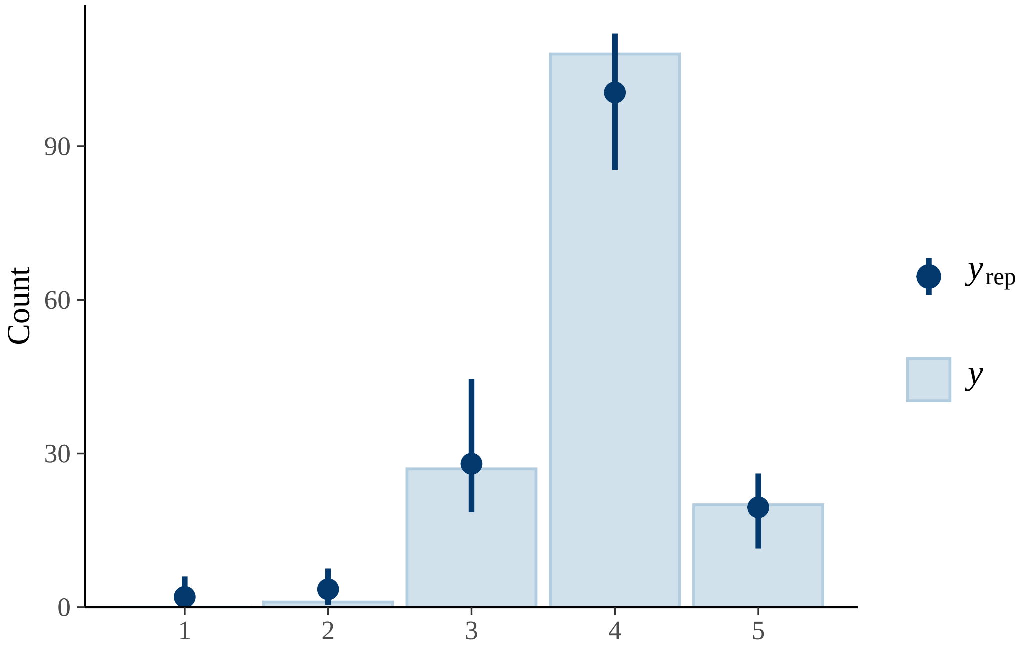
```
     elpd_diff se_diff
fit2  0.0       0.0   
fit1 -1.6       1.1   
```

From the output we see that the second model, without age group, seems to perform slightly better. Larger elpd value indicates better fit.

Posterior predictive checks

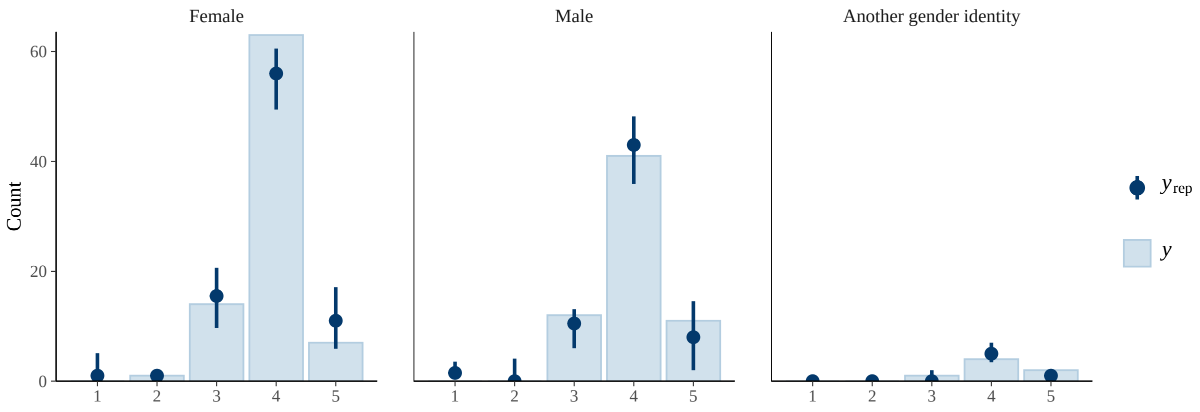
Posterior predictive checks can be used to check that the values given by probabilistic model are in line with the data used to fit the model. For the ordinal cumulative model plotting the predicted category counts against the observed counts is a common posterior check. This can be done using function `pp_check()` from `brms` with argument `type = "bars"`.

```
# Model posterior predictive check
pp_check(fit1, type = "bars")
```



Based on the figure, the model output is inline with the observed data. This checking can also be done at different levels of categorical explanatory variable, for example gender. The function used is the same, but now `type = "bars_grouped"` is used instead and additional function parameter is needed `group = "Q2"`.

```
# Model posterior predictive check at different levels of gender
pp_check(fit1, type = "bars_grouped", group = "Q2")
```



Again, the model output seems to agree with the observed data.

Testing proportional odds assumption

Proportional odds assumption, which means that the effect of explanatory variables is same for all answer categories, is an important part of the cumulative logistic model. This assumption should always be tested to be sure that the model isn't mispecified. In the probabilistic framework this can be done using the `loo` function introduced before.

The assumption is tested by fitting a model that allows category-specific effects that is then compared to the proportional odds model. This alternative model cannot be fit using the cumulative family as it can lead to problems, which is why an alternative model family is used. We will use the adjacent-category model for this purpose. The category specific effects are specified by putting the explanatory variable inside function `cs()`.

```r
# Testing proportional odds assumption
# Fitting model from the
fit1_adj <- brm(Q11 ~ Q2 + Age_group, data = helsinki_data,
            family = acat("logit"), seed = 2025,
            prior = prior(normal(-1.39,1), class = "Intercept", coef = "1") +
                        prior(normal(-0.41,1), class = "Intercept", coef = "2") +
                        prior(normal(0.41,1), class = "Intercept", coef = "3") +
                        prior(normal(1.39,1), class = "Intercept", coef = "4") +
                        prior(normal(0,2), class = "b"), backend = "cmdstanr",
            control = list(adapt_delta = 0.95), silent = 0, refresh = 0)
# Fitting model with category effects for gender
fit1_cs_Q2 <- brm(Q11 ~ cs(Q2) + Age_group, data = helsinki_data,
            family = acat("logit"), seed = 2025,
            prior = prior(normal(-1.39,1), class = "Intercept", coef = "1") +
                        prior(normal(-0.41,1), class = "Intercept", coef = "2") +
                        prior(normal(0.41,1), class = "Intercept", coef = "3") +
                        prior(normal(1.39,1), class = "Intercept", coef = "4") +
                        prior(normal(0,2), class = "b"), backend = "cmdstanr",
            control = list(adapt_delta = 0.95), silent = 0, refresh = 0)
# Fitting model with category effects for age group
fit1_cs_age <- brm(Q11 ~ Q2 + cs(Age_group), data = helsinki_data,
            family = acat("logit"), seed = 2025,
            prior = prior(normal(-1.39,1), class = "Intercept", coef = "1") +
                        prior(normal(-0.41,1), class = "Intercept", coef = "2") +
                        prior(normal(0.41,1), class = "Intercept", coef = "3") +
                        prior(normal(1.39,1), class = "Intercept", coef = "4") +
                        prior(normal(0,2), class = "b"), backend = "cmdstanr",
            control = list(adapt_delta = 0.95), silent = 0, refresh = 0)

fit1_cs <- brm(Q11 ~ cs(Q2) + cs(Age_group), data = helsinki_data,
            family = acat("logit"), seed = 2025,
            prior = prior(normal(-1.39,1), class = "Intercept", coef = "1") +
                        prior(normal(-0.41,1), class = "Intercept", coef = "2") +
                        prior(normal(0.41,1), class = "Intercept", coef = "3") +
                        prior(normal(1.39,1), class = "Intercept", coef = "4") +
                        prior(normal(0,2), class = "b"), backend = "cmdstanr",
            control = list(adapt_delta = 0.95), silent = 0, refresh = 0)
```

```r
# Comparing the models using loo
loo(fit1, fit1_adj, fit1_cs, fit1_cs_age, fit1_cs_Q2)$diffs
```

```
            elpd_diff se_diff
fit1_adj     0.0       0.0
fit1_cs_Q2  -0.7       1.2
fit1_cs_age -1.1       2.7
fit1_cs     -1.7       2.9
fit1        -4.1       0.4
```

To differentiate the effect of the modeling family from the effect of the explanatory variables, an adjacent-category model is fit without the category-specific effects. We can see from the output that the proportional odds assumption seems to hold. Interestingly the adjacent-category family seems to give a little better of a fit, but the non-proportional versions perform worse.

Hierarchical example

If we want to compare the differences between the cities, we can do so by fitting a hierarchical model. The hierarchical model can also be fit with the `brm()` function. This is done by adding the term `(1 | City)` to the model formula.

```r
# Fitting a hieararchical model
census_data_all <- census_data_all %>%
  mutate(Q11 = as.integer(Q11))
fith <- brm(Q11 ~ Q2 + Age_group + (1 | City), data = census_data_all,
            family = cumulative("logit"), seed = 2025,
            prior = prior(normal(-1.39,1), class = "Intercept", coef = "1") +
                        prior(normal(-0.41,1), class = "Intercept", coef = "2") +
                        prior(normal(0.41,1), class = "Intercept", coef = "3") +
                        prior(normal(1.39,1), class = "Intercept", coef = "4") +
                        prior(normal(0,2), class = "b") +
                        prior(normal(0,1), class = "sd"), backend = "cmdstanr",
            control = list(adapt_delta = 0.99), silent = 0, refresh = 0)
```

Now the model fitting time is much longer when compared to the two previous models. Let's look at the output.

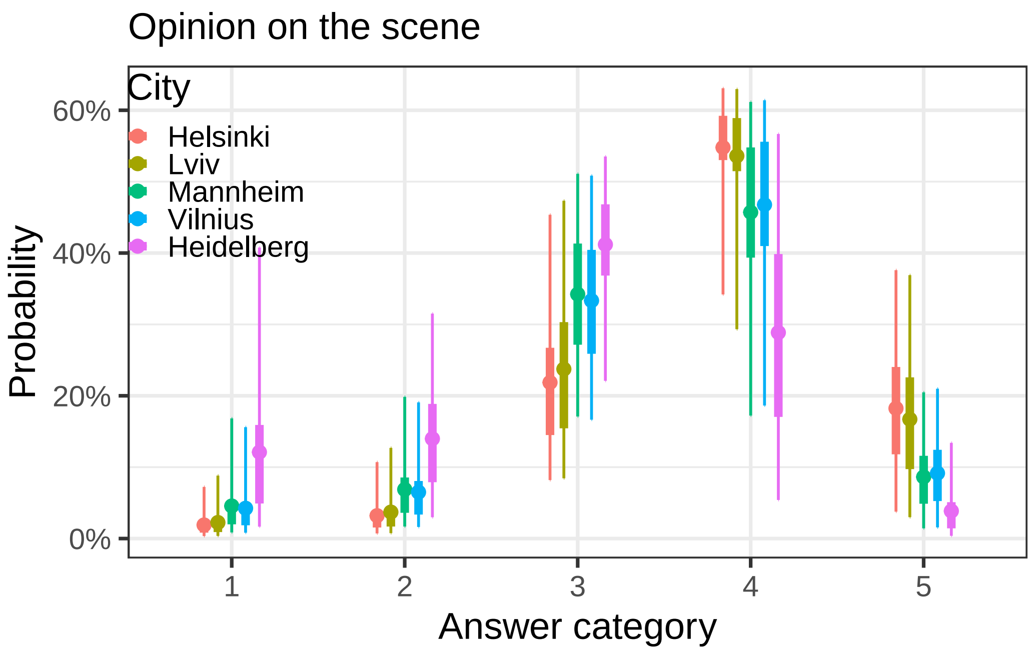
```
# Model ouptut
tidy(fith) %>%
  select(-c(effect, component, group)) %>%
  kable()
```

term	estimate	std.error	conf.low	conf.high
(Intercept)[1]	-3.4519286	0.6301435	-4.6827783	-2.2063638
(Intercept)[2]	-2.3698116	0.5939884	-3.5296775	-1.1856767
(Intercept)[3]	-0.2912972	0.5913751	-1.4375187	0.8894460
(Intercept)[4]	2.4451410	0.6042419	1.3047595	3.6467925
Q2Male	0.0640173	0.2057689	-0.3312283	0.4563819
Q2Anothergenderidentity	-0.9680104	0.5165799	-1.9811082	0.0486212
Age_group18M29	0.1726377	0.4620488	-0.7017289	1.0927885
Age_group30M49	-0.5594507	0.4448805	-1.4287382	0.3264278
Age_group50M64	-0.3976363	0.4838820	-1.3456015	0.5542994
sd___(Intercept)	1.0424966	0.4226234	0.3994085	2.0289673

Now there is an additional term in the output, `sd(Intercetp)`, which is estimated standard deviation for the city-specific random effects. Let's plot the city-specific estimates and their intervals. To do this we have to first retrieve expected posterior values from the model. This can be done using function `add_epred_draws` function from `tidybayes` package.

```
# Obtaining posterior estimates
fith_draws <- expand.grid(Q2 = c("Female", "Male", "Another gender identity"),
                          Age_group = c("30-49", "18-29", "50-64", ">64"),
                          City = c("Helsinki", "Lviv",
                                   "Mannheim", "Vilnius", "Heidelberg")) %>%
  add_epred_draws(object = fith)
```

```
# Plotting the city comparisons
fith_draws %>%
  group_by(.category, City) %>%
  summarise(est = mean(.epred), lwr95 = quantile(.epred, 0.025),
            upr95 = quantile(.epred, 0.975), lwr50 = quantile(.epred, 0.25),
            upr50 = quantile(.epred, 0.75)) %>%
  ggplot(aes(x = .category, y = est, color = City)) +
  geom_point(position = position_dodge(width = 0.4), size = 2) +
  geom_errorbar(aes(ymin = lwr95, ymax = upr95), width = 0,
                position = position_dodge(width = 0.4), size = 0.5)  +
  geom_errorbar(aes(ymin = lwr50, ymax = upr50), width = 0,
                position = position_dodge(width = 0.4), size = 1.5) +
  theme_bw(base_size = 14) +
  theme(legend.position = c(0.1,0.8),
        legend.background = element_rect(fill = "transparent"),
        legend.key = element_rect(fill = "transparent"),
        legend.key.size = unit(0.3, "cm")) +
  labs(x = "Answer category", y = "Probability",
       color = "City", subtitle = "Opinion on the scene") +
  scale_y_continuous(label = scales::percent)
```



We can see that there are some differences between the cities with people in Helsinki and Lviv seem more likely to answer the positive categories. It is important to note that the intervals for Heidelberg are large, because the number of answers from the city was low.

Conclusion

In this document we gave an overview of how to do probabilistic analysis of ordinal survey data. We covered how to fit the model and how to do model checking and comparison in the probabilistic framework. Additionally, we provided an example how to conduct hierarchical analysis of the data. If you want to learn more about the `brms` package, you can check out its website.

For more advanced forms of analysis you can check out the file `advanced_prob_examples.qmd` for those.