Introduction to the LCAplotter package

```r
library(LCAplotter)
```

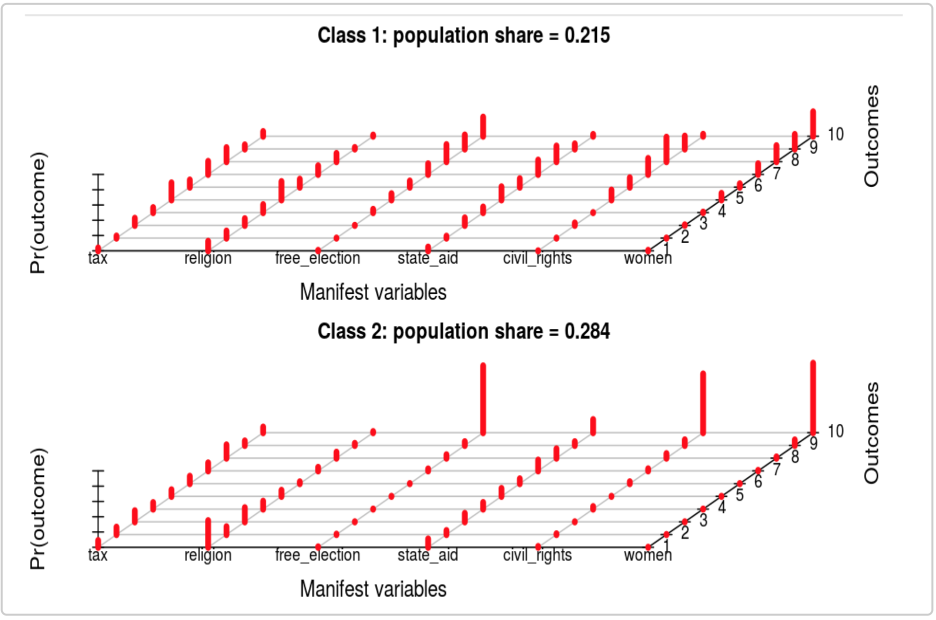
Motivation and introduction

This is a package built on top of utilities provided by the poLCA package.

Latent class analysis (LCA) or Latent profile analysis (LPA), which uses a parametric model to place respondents into classes (or clusters) based on their response patterns. In LPA, the number of classes is determined by the expectation-maximization algorithm which involves an iterative process until the model converges on a best fit for the data. This involves the notion that there should be shared variance within the clusters, and that clusters should be empirically distinct from each other. For more detailed introduction to the method, please refer here

LCA has been extensively used in social science however the visualization component is poorly implemented in R compared with its counterparts Mplus or STATA.

The poLCA package only offers the following 3D plots, which is explicit in itself, however, may not be propoer for more rigrous research settings:



This package contributes to solving this plot by adding visualization capasities to facilitate broader usage of LCA.

Termenology explanation

Item response variable: In orde to conduct LCA modeling, you would usually need to have a set of item response variables, and such item response variables usually taps into certain specific aspects of (a) broader theoretical construct(s). For example, to measure democracy, there are 6 or 7 item response variable with each of them covering one particular parf of democracy. For example, one item response variable is related to the taxation while another related to women’s rights.

formula: This is the required piece for the poLCA package to fit the LCA model. A formula expression is of the form response ~ predictors. LCAplotter package adopts the way formula parameter is defined from the poLCA package.

Data

LCAplotter has encapsulated an example dataset called democracy. The view of the dataset is as follows:

```
knitr::kable(democracy[1:10,])
```

country	tax	religion	free_election	state_aid	civil_rights	women
United States	5	1	5	5	8	8
United States	1	1	10	2	10	10
United States	5	10	10	10	8	10
United States	6	3	9	9	9	7
United States	10	10	10	10	10	5
United States	5	1	8	5	8	8
United States	6	2	9	9	5	9
United States	5	5	5	5	5	10
United States	9	6	10	10	9	10
United States	4	3	9	8	8	8

Note: This data format is important because both poLCA and LCAplotter depend on this format in order to function properly. **Thus, please make sure in your data, you have item response variables as columns.**

After loading the LCAplotter package via

```
library(LCAplotter)
# or
require(LCAplotter)
```

you could directly access the data by calling its name

```
democracy
```

This dataset is an extracted from the World Value Survey wave 5 and subsetted from the US sample. All the item response variables are measureing the 'essential characteristics of democracy' and there are in total 6 of them.

Individuals are asked to read the following prompt:

> Many things may be desirable, but not all of them are essential characteristics of democracy. Please tell me for each of the following things how essential you think it is as a characteristic of democracy. Use this scale where 1 means "not at all an essential characteristic of democracy" and 10 means it definitely is "an essential characteristic of democracy."

The 6 item statements are:

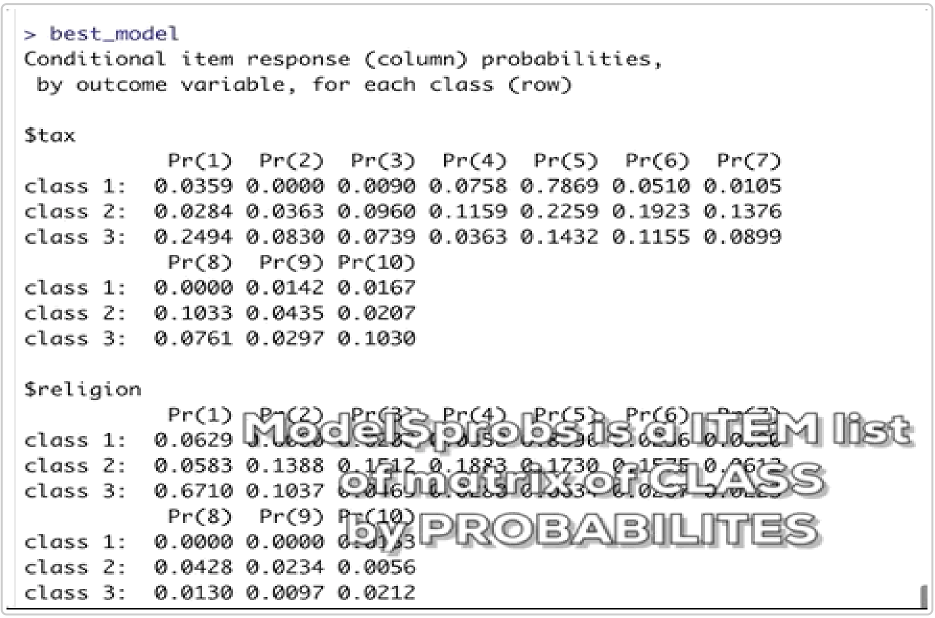
1. Government taxes rich to subsidize poor;
2. Religious authorities interpret laws;
3. People choose leaders in free elections;
4. People receive state aid for employment;
5. Civil rights protect people's liberties from state oppression;
6. Women have the same rights as men.

We would like to conduct LDA to find how many hidden classes (latent clusters) are there among our sample data (N = 1182). Each hidden class may have different understanding or answering patterns for those 6 item response variables.

poLCA model object

Because LCAplotter is built on top of poLCA, we are inheriting the model object from poLCA. The poLCA model object is a list of diverse outputs. The most related output is the **probs** list, which is a 3 dimentional tensor with dimentionality as $item\ response \times class \times probability$.

A more explicit view could be seen below:



```
> best_model
Conditional item response (column) probabilities,
 by outcome variable, for each class (row)

$tax
           Pr(1)  Pr(2)  Pr(3)  Pr(4)  Pr(5)  Pr(6)  Pr(7)
class 1:  0.0359 0.0000 0.0090 0.0758 0.7869 0.0510 0.0105
class 2:  0.0284 0.0363 0.0960 0.1159 0.2259 0.1923 0.1376
class 3:  0.2494 0.0830 0.0739 0.0363 0.1432 0.1155 0.0899
           Pr(8)  Pr(9) Pr(10)
class 1:  0.0000 0.0142 0.0167
class 2:  0.1033 0.0435 0.0207
class 3:  0.0761 0.0297 0.1030

$religion
           Pr(1)  Pr(2)  Pr(3)  Pr(4)  Pr(5)  Pr(6)  Pr(7)
class 1:  0.0629 [illegible] [illegible] [illegible] 0.8396 0.0256 [illegible]
class 2:  0.0583 0.1388 0.1512 0.1883 0.1730 0.1575 0.0612
class 3:  0.6710 0.1037 0.0469 [illegible] [illegible] 0.0207 [illegible]
           Pr(8)  Pr(9) Pr(10)
class 1:  0.0000 0.0000 [illegible]
class 2:  0.0428 0.0234 0.0056
class 3:  0.0130 0.0097 0.0212
```

This model object is important because we need to feed it into the visualization functions.

Visualization

Finding the most optimal model

Before visualizing the final model, we need to find out the optimal model according to the fit indices. This could be achieved by running the **find_best_fit** function.

```
formula = cbind(tax, religion, free_election, state_aid, civil_rights, women) ~ 1
best_model = find_best_fit(democracy, formula)
```

```
print(best_model)
#> Conditional item response (column) probabilities,
#>  by outcome variable, for each class (row)
#> 
#> $tax
#>            Pr(1)  Pr(2)  Pr(3)  Pr(4)  Pr(5)  Pr(6)  Pr(7)  Pr(8)  Pr(9) Pr(10)
#> class 1:  0.0359 0.0000 0.0090 0.0758 0.7870 0.0509 0.0104 0.0000 0.0142 0.0167
#> class 2:  0.2494 0.0830 0.0739 0.0363 0.1433 0.1155 0.0900 0.0761 0.0297 0.1030
#> class 3:  0.0284 0.0362 0.0960 0.1159 0.2259 0.1924 0.1376 0.1033 0.0435 0.0207
#> 
#> $religion
#>            Pr(1)  Pr(2)  Pr(3)  Pr(4)  Pr(5)  Pr(6)  Pr(7)  Pr(8)  Pr(9) Pr(10)
#> class 1:  0.0630 0.0000 0.0208 0.0358 0.8397 0.0255 0.0000 0.0000 0.0000 0.0153
#> class 2:  0.6710 0.1037 0.0469 0.0280 0.0634 0.0207 0.0223 0.0130 0.0097 0.0212
#> class 3:  0.0583 0.1388 0.1512 0.1883 0.1730 0.1575 0.0612 0.0428 0.0234 0.0056
#> 
#> $free_election
#>            Pr(1)  Pr(2)  Pr(3)  Pr(4)  Pr(5)  Pr(6)  Pr(7)  Pr(8)  Pr(9) Pr(10)
#> class 1:  0.0070 0.0000 0.0169 0.0166 0.8321 0.0877 0.0000 0.0085 0.0061 0.0252
#> class 2:  0.0423 0.0076 0.0024 0.0000 0.0079 0.0053 0.0045 0.0132 0.0361 0.8807
#> class 3:  0.0082 0.0046 0.0269 0.0466 0.0294 0.1094 0.1217 0.2493 0.1993 0.2045
#> 
#> $state_aid
#>            Pr(1)  Pr(2)  Pr(3)  Pr(4)  Pr(5)  Pr(6)  Pr(7)  Pr(8)  Pr(9) Pr(10)
#> class 1:  0.0155 0.0000 0.0366 0.0127 0.8265 0.0964 0.0000 0.0000 0.0000 0.0124
#> class 2:  0.1340 0.0721 0.0581 0.0750 0.1178 0.1038 0.1121 0.0806 0.0474 0.1991
#> class 3:  0.0000 0.0271 0.0667 0.1301 0.1251 0.1831 0.2029 0.1551 0.0962 0.0137
#> 
#> $civil_rights
#>            Pr(1)  Pr(2)  Pr(3)  Pr(4)  Pr(5)  Pr(6)  Pr(7)  Pr(8)  Pr(9) Pr(10)
#> class 1:  0.0247 0.0216 0.0250 0.0479 0.6887 0.0553 0.0526 0.0333 0.0000 0.0509
#> class 2:  0.0362 0.0064 0.0090 0.0114 0.0234 0.0137 0.0306 0.0784 0.0930 0.6979
#> class 3:  0.0087 0.0111 0.0248 0.0263 0.1208 0.1383 0.2322 0.1771 0.2136 0.0471
#> 
#> $women
#>            Pr(1)  Pr(2)  Pr(3)  Pr(4)  Pr(5)  Pr(6)  Pr(7)  Pr(8)  Pr(9) Pr(10)
#> class 1:  0.0000 0.0083 0.0484 0.0733 0.6487 0.0371 0.0294 0.0367 0.0147 0.1033
#> class 2:  0.0241 0.0060 0.0031 0.0000 0.0134 0.0110 0.0170 0.0277 0.0442 0.8536
#> class 3:  0.0079 0.0166 0.0201 0.0206 0.0714 0.0921 0.1144 0.1828 0.2640 0.2101
#> 
#> Estimated class population shares 
#>  0.1029 0.6069 0.2902 
#>  
#> Predicted class memberships (by modal posterior prob.) 
#>  0.1049 0.6142 0.2809 
#>  
#> ========================================================= 
#> Fit for 3 latent classes: 
#> ========================================================= 
#> number of observations: 1182 
#> number of estimated parameters: 164 
#> residual degrees of freedom: 1018 
#> maximum log-likelihood: -11447.52 
#>  
#> AIC(3): 23223.05
#> BIC(3): 24055.34
#> G^2(3): 7902.974 (Likelihood ratio/deviance statistic) 
#> X^2(3): 10758371 (Chi-square goodness of fit) 
#> 
```

The default criterion used to find the optimal model fit is BIC. This could be easily changed by speficifying other fitting indices such as “aic” or “Gsq”, speficially, you could do as follows:

```
best_model = find_best_fit(democracy, formula, criterion = 'aic')

# or

best_model = find_best_fit(democracy, formula, criterion = 'Gsq')
```

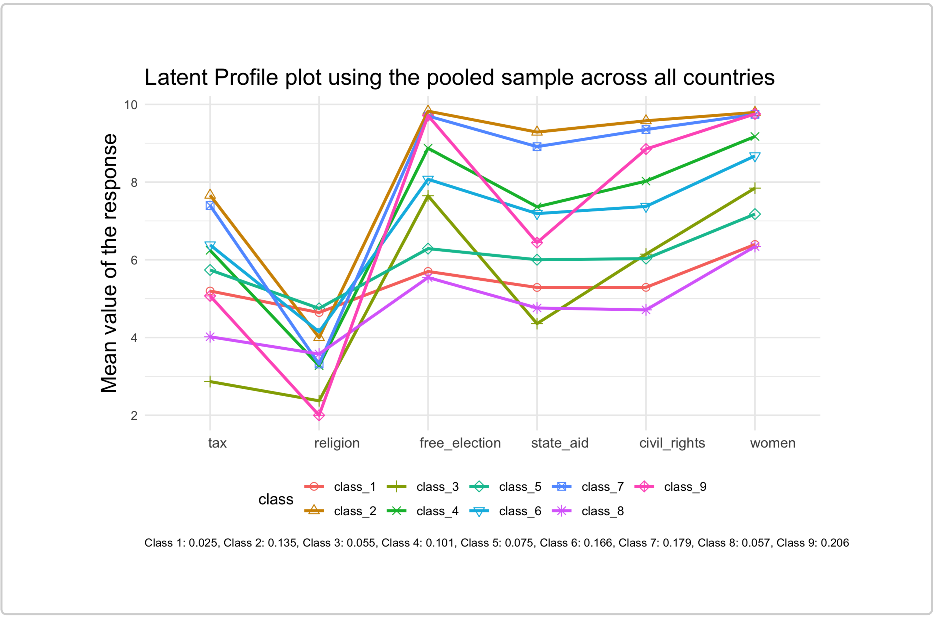
Further, we could change the search space for the searching of the most optimal model. The default search space is from 1 - 7 classes (clusters). If you want to enlarge the search space to 9, for example, you could change the parameter as:

```
best_model = find_best_fit(democracy, formula, criterion = 'Gsq', maximum_num_class = 9)
```

The find_best_fit function will return an poLCA model object, which affords us to move forward to visualizations.

profile_plot

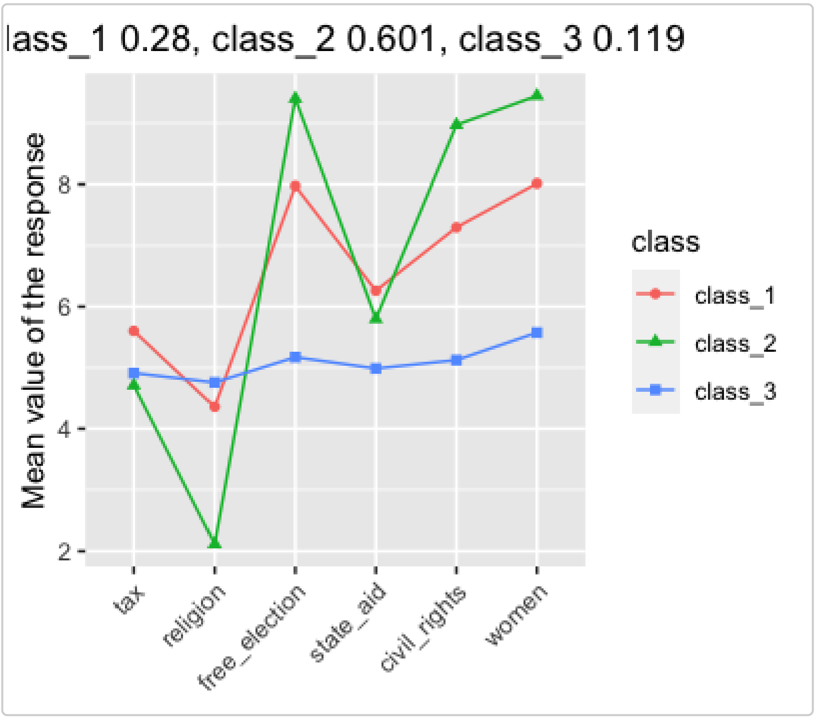
The profile plot looks like the following:



To construc profile plot, you could run the profile_plot function:

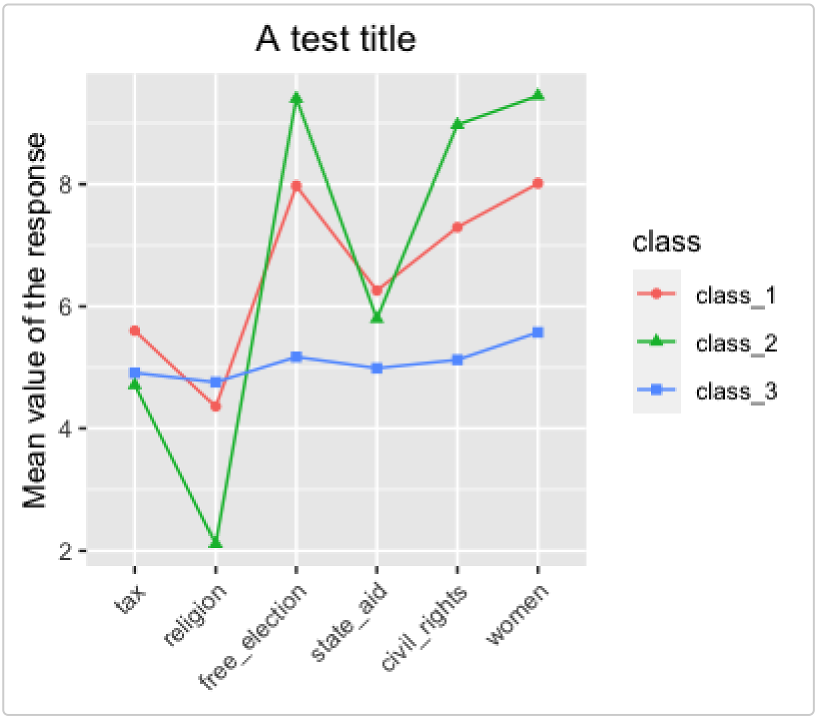
```
plot = profile_plot(democracy, num_var = 6, model = NULL, form = formula, maximum_num_class = 3)
```

```
print(plot)
```



The plot object has been returned and it could be further customized by adding more ggplot syntax. For example:

```
print(plot +
        ggtitle("A test title"))
```



profile_plot function also wraps in the parameters for the poLCA package. Thus if you want to customized the model fitting parameters of poLCA funtion such as calc.se = FALSE, you could directly pass it in the parameters of the profile plot.

stacked_bar plots

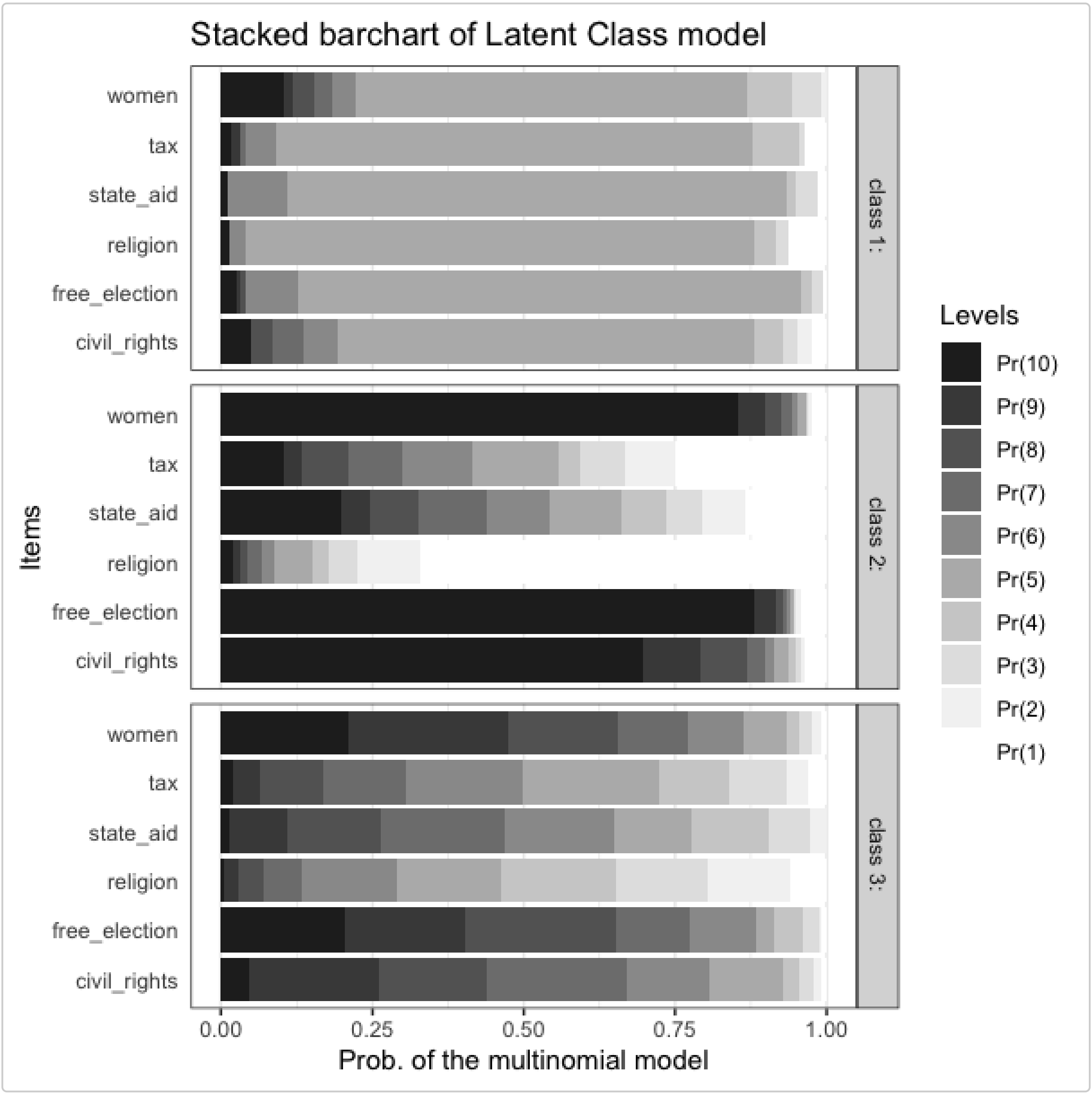
There are two operationalization for the stacked_bar plots

1. stacked_bar_by_class
2. stacked_bar_by_item

stacked_bar_by_class plot

This is the stacked bar plot facetted by the different classes. Each class would have its own 'window' and within each 'window' you could view its correponding probabilities for each item response. An example would look like this:

```
p = stacked_bar_by_class(best_model)
print(p)
```



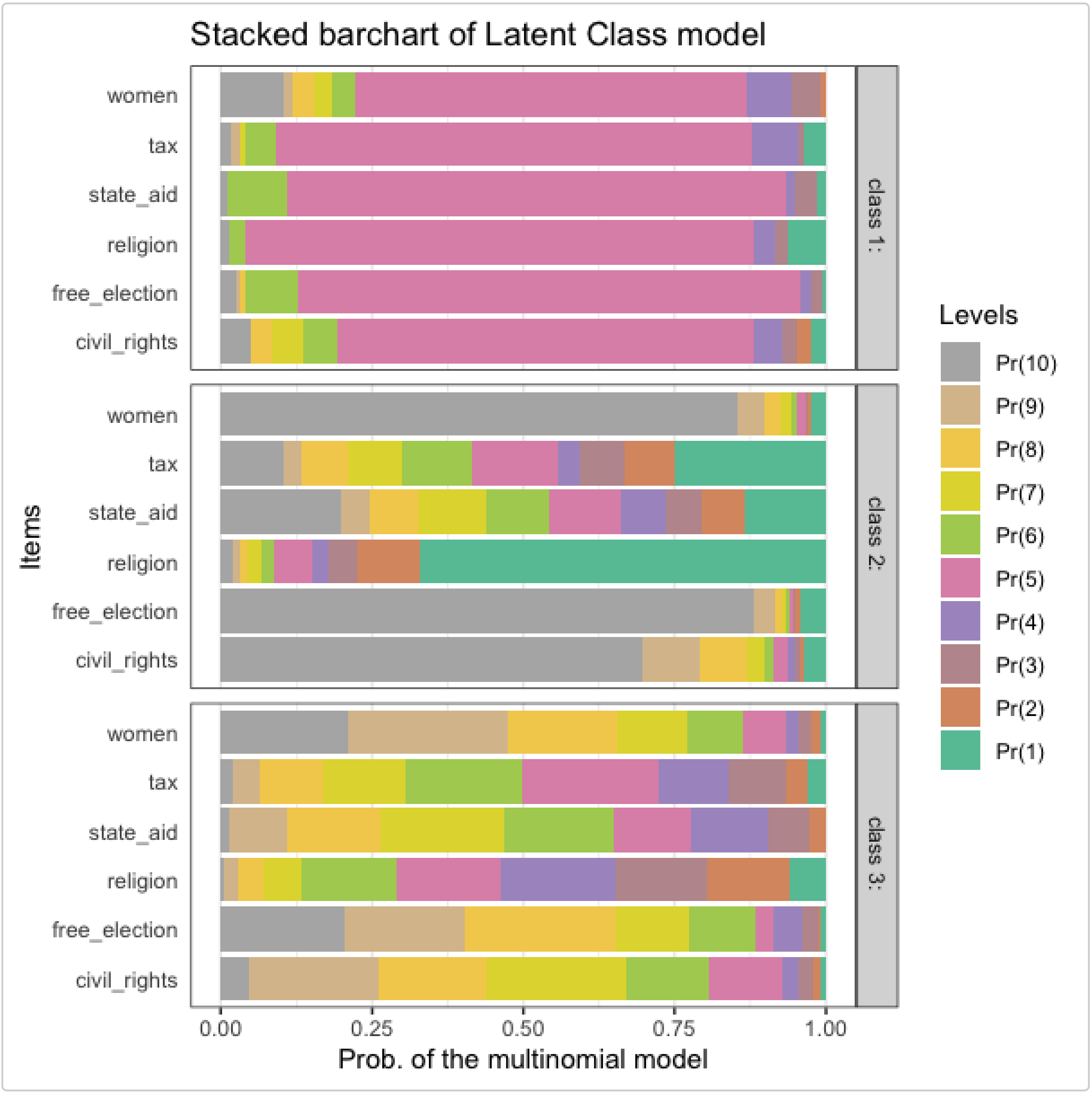
This plot offers us a direct comparison of answer patterns for each item response variable WITHIN a certain class. For example, in class 1, the 'women', 'free election', and 'civil rights' are generally more agreed than 'taxation', 'staet aid', and 'religion.' However, the class 3 shows a huge contrasts with class 1 in that ALL of the item response variables are less agreeed upon than class 1.

The return object is a ggplot object which supports further custmization of ggplot syntax.

Further, this function supports the color palatte from the **RColorBrewer** package. The default palatte used is called "Greys". You could customize the color palette simply by:

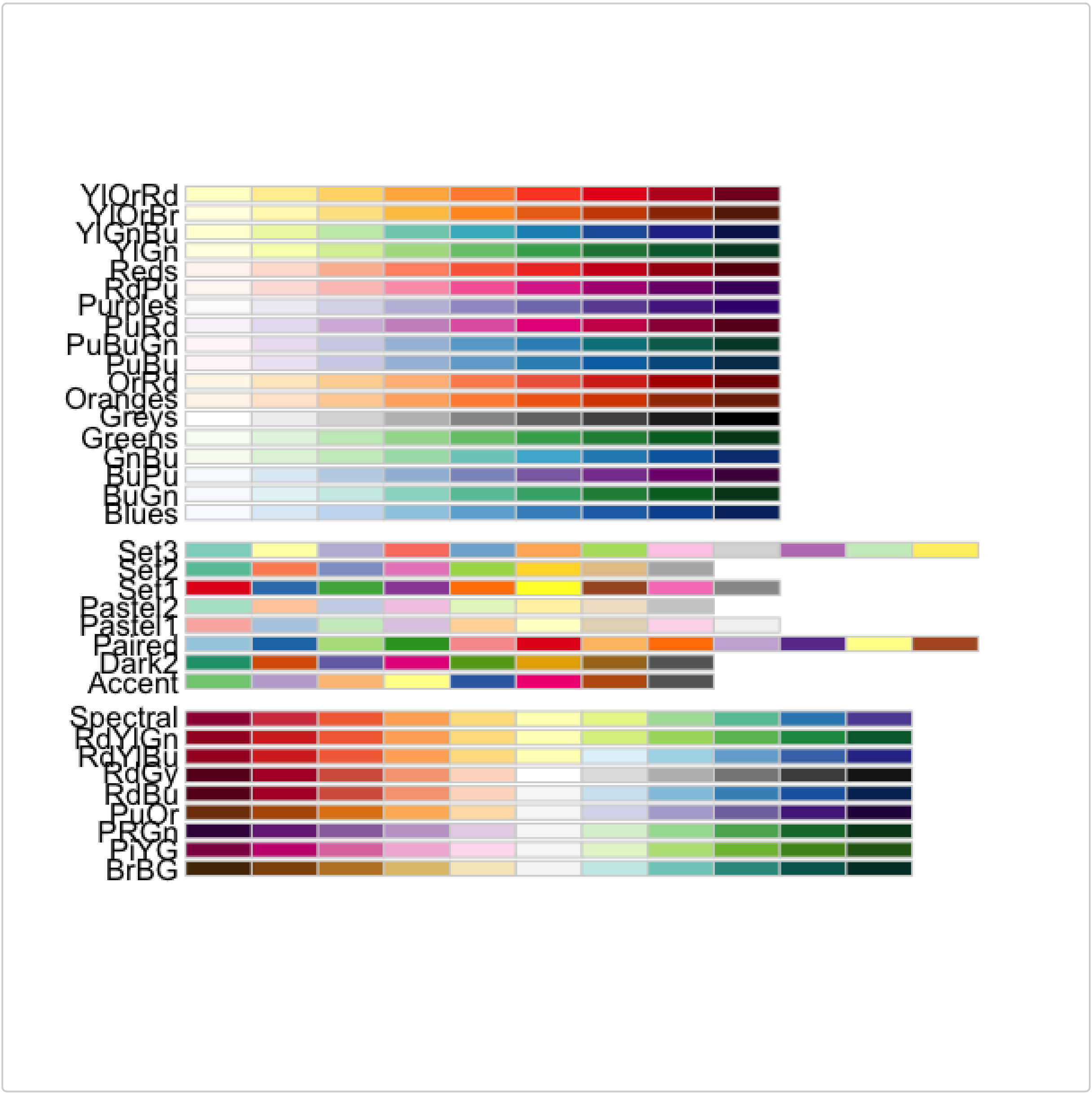
```
p = stacked_bar_by_class(best_model, color_palette = 'Set2')

print(p)
```



All of the color palettes could be view from below:

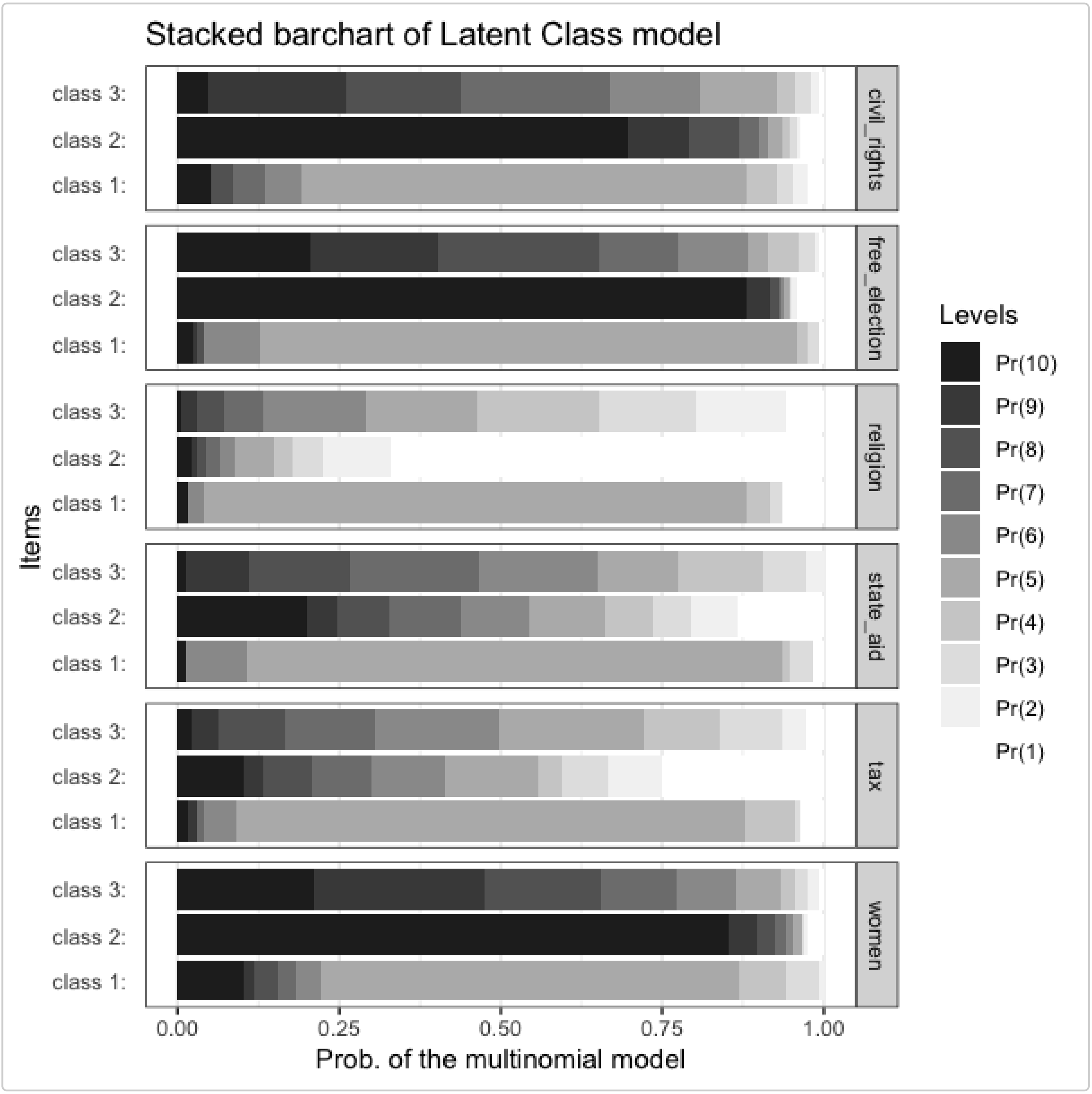
```
RColorBrewer::display.brewer.all()
```



stacked_bar_by_item plot

This is the stacked bar plot facetted by the different items. Each item would have its own 'window' (facets) and within each facets you could view its correponding probabilities for each class (clusters). An example would look like this:

```
p = stacked_bar_by_item(best_model)
#> Loading required package: RColorBrewer
print(p)
```



This plot offers us direct comparison of the differences between classes on each item response. For example, we could view directly that for the women rights variable, the class 1 is way higher compared with the other two classes.

You could change the color palettes as well by specifying the color_palette paramter.