Online Supplementary Materials A: Exploratory LPA for Ocean Microplastics

Online Supplementary Materials A: Exploratory LPA for Ocean Microplastics

This is an example of exploratory latent class analysis (LCA) with continuous indicators, otherwise known as latent profile analysis (LPA) or finite Gaussian mixture modeling, using `tidySEM`. The present example uses data collected by Alkema as part of a study on ocean microplastics. The purpose of this study was to provide a more nuanced model for the distribution of different sizes of ocean microplastics than the commonly used normal distribution. To this end, a mixture of normals was used. Since there is no theoretical reason to expect a certain number of classes, this is an exploratory LCA. To view its documentation, run the command `?tidySEM::alkema_microplastics` in the R console. The original analyses are available at https://github.com/cjvanlissa/lise_microplastics; in this vignette, we take a different approach to the analysis to showcase other possibilities.

Loading the Data

To load the data, simply attach the `tidySEM` package. For convenience, we assign the variables used for analysis to an object called `df`. As explained in the paper, the classes are quite different for lines, films, and fragments. For this reason, we here only use data from fragments. The indicators are fragments' length and width in millimeters. The sample size was not planned.

```
# Load required packages
library(tidySEM)
library(ggplot2)
# Load data
df_analyze <- alkema_microplastics[alkema_microplastics$category ==
    "Fragment", ]
df <- df_analyze[, c("length", "width")]
```

Table 1

Descriptive statistics

name	type	n	unique	mean	median	sd	min	max	skew_2se	kurt_2se
length	numeric	5605	2086	2.94	2.37	1.89	1.00	69.16	137.38	2,116.43
width	numeric	5605	2079	2.00	1.62	1.11	0.20	6.76	22.23	37.08

Descriptive statistics

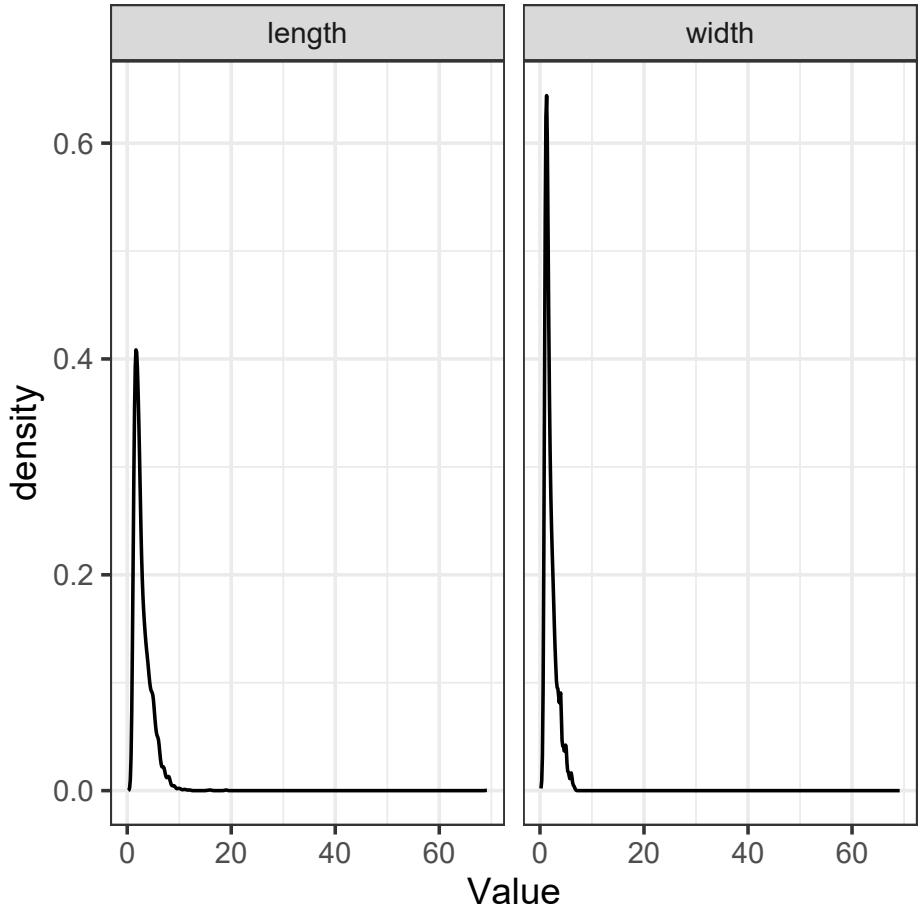
As per the best practices, the first step in LCA is examining the observed data. We use `tidySEM::descriptives()` to describe the data numerically. Because all items are continuous, we remove columns for categorical data to de-clutter the table:

```
desc <- tidySEM::descriptives(df)
desc <- desc[, c("name", "type", "n", "unique", "mean", "median",
    "sd", "min", "max", "skew_2se", "kurt_2se")]
papaja::apa_table(desc, caption = "Descriptive statistics")
```

The data are correctly coded as `numeric` and the distributional characteristics match the intended measurement level. The variable scales are comparable (both in millimeters and no large discrepancies between variances). There are no missing values; if any variables had missing values, we would report an MCAR test with `mice::mcar()`, and explain that missing data are accounted for using FIML. Additionally, we can plot the data. The `ggplot2` function `geom_density()` is useful to visualize continuous data:

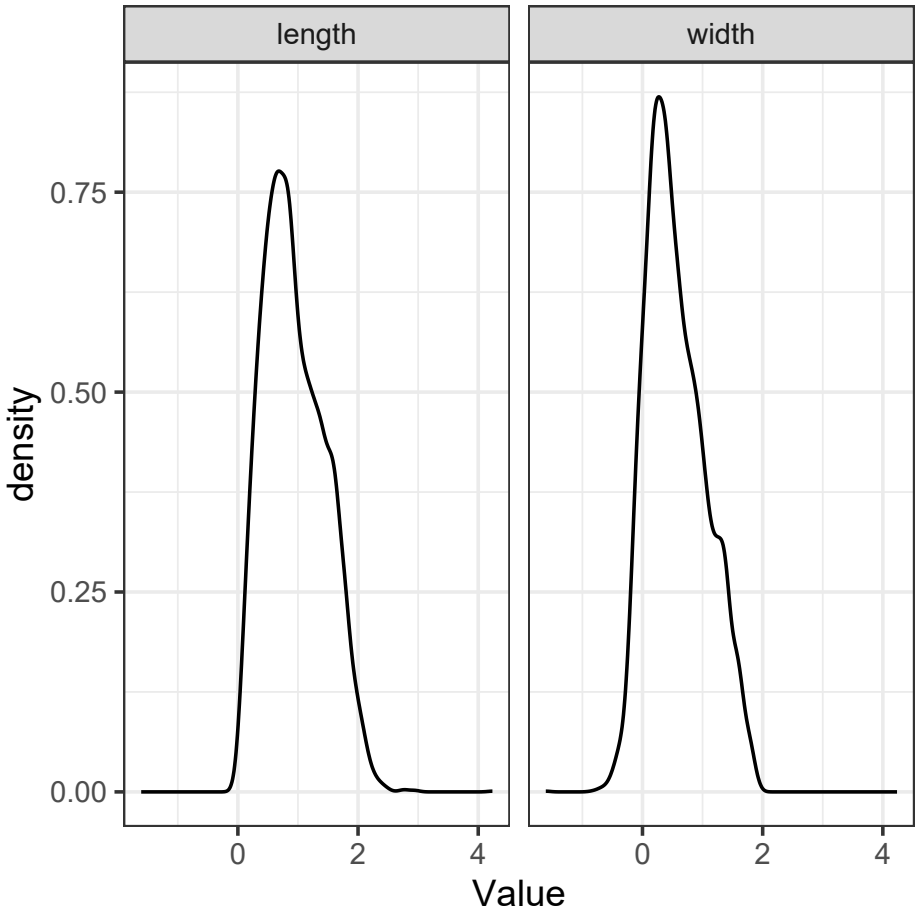
```
df_plot <- df
names(df_plot) <- paste0("Value.", names(df_plot))
df_plot <- reshape(df_plot, varying = names(df_plot), direction = "long",
```

```
    timevar = "Variable")
ggplot(df_plot, aes(x = Value)) + geom_density() + facet_wrap(~Variable) +
    theme_bw()
```



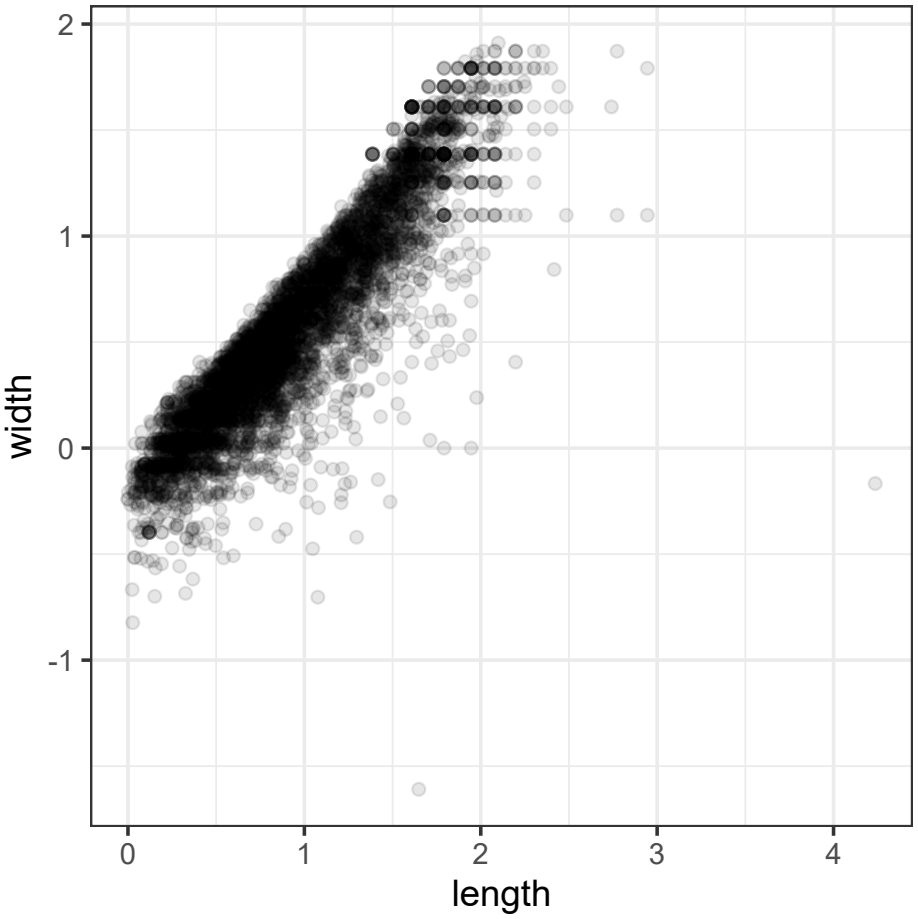
Both the table above and the density plot indicate that the data are extremely right-skewed and kurtotic. With this in mind, it can be useful to transform and rescale the data. We will use a log transformation.

```
df_plot$Value <- log(df_plot$Value)
ggplot(df_plot, aes(x = Value)) + geom_density() + facet_wrap(~Variable) +
    theme_bw()
```



The log transformation addresses the aforementioned concerns regarding skew and kurtosis. To confirm this, reshape the data to wide format and examine a scatterplot:

```
df <- reshape(df_plot, direction = "wide", v.names = "Value")[,
    -1]
names(df) <- gsub("Value.", "", names(df), fixed = TRUE)
ggplot(df, aes(x = length, y = width)) + geom_point(alpha = 0.1) +
    theme_bw()
```



Conducting Latent Profile Analysis

As all variables are continuous, we can use the convenience function `tidySEM::mx_profiles()`, which is a wrapper for the generic function `mx_mixture()` optimized for continuous indicators. Its default settings are appropriate for LPA, assuming fixed variances across classes and zero covariances. Its arguments are `data` and number of `classes`. All variables in `data` are included in the analysis, which is why we first selected the indicator variables. The models are estimated using simulated annealing, with start values determined via initial K-means clustering.

As this is an exploratory LCA, we will conduct a rather extensive search across model specifications and number of classes. We will set the maximum number of classes K to four

to limit computational demands. We set a seed to ensure replicable results.

As the analysis takes a long time to compute, it is prudent to save the results to disk immediately, so as not to lose them. For this, we use the function `saveRDS()`. We can later use `res <- readRDS("res_gmm.RData")` to load the analysis from the file.

```
set.seed(123)
res <- mx_profiles(data = df, classes = 1:4, variances = c("equal",
    "varying"), covariances = c("zero", "equal", "varying"),
    expand_grid = TRUE)
saveRDS(res, "res_gmm.RData")
```

Class Enumeration

To compare the fit of the estimated models, we create a model fit table using `table_fit()`. We will use the BIC for class enumeration.

```
fit <- table_fit(res)
```

First, we determine whether any models can be disqualified. There were no indications of convergence problems during estimation, so this is not a reason to disqualify solutions. Next, we check for global and local identifiability. The global ratio of observations per parametrer is large, as the minimum `np_ratio` is 244. The smallest ratio of class size to class-specific parameters is 18 (see `np_local`), which is no cause for concern.

```
tbl <- fit[, c("Name", "LL", "Parameters", "BIC", "Entropy",
    "prob_min", "n_min", "np_ratio", "np_local")]
names(tbl) <- c("Name", "LL", "p", "BIC", "Ent.", "p_min", "n_min",
    "np_ratio", "np_local")
papaja::apa_table(tbl, caption = "Model fit table.")
```

Table 2

Model fit table.

Name	LL	p	BIC	Ent.	p_min	n_min	np_ratio	np_local
equal var 1	-8,107.31	4	16,249.14	1.00	1.00	1.00	1,401.25	1,401.25
equal var 2	-5,211.25	7	10,482.91	0.87	0.94	0.35	800.71	658.00
equal var 3	-4,138.27	10	8,362.85	0.83	0.88	0.20	560.50	427.12
equal var 4	-3,500.42	13	7,113.04	0.83	0.89	0.14	431.15	320.00
free var 1	-8,107.31	4	16,249.14	1.00	1.00	1.00	1,401.25	1,401.25
free var 2	-5,137.90	9	10,353.49	0.85	0.94	0.40	622.78	564.00
free var 3	-4,005.10	14	8,131.04	0.83	0.89	0.31	400.36	436.75
free var 4	-3,330.87	19	6,825.74	0.84	0.88	0.21	295.00	291.00
equal var, equal cov 1	-3,388.56	5	6,820.28	1.00	1.00	1.00	1,121.00	1,121.00
equal var, equal cov 2	-3,082.31	8	6,233.66	0.72	0.86	0.31	700.62	494.00
equal var, equal cov 3	-3,030.10	11	6,155.14	0.67	0.71	0.17	509.55	313.33
equal var, equal cov 4	-3,020.67	14	6,162.19	0.61	0.68	0.14	400.36	280.36
free var, equal cov 1	-3,388.56	5	6,820.28	1.00	1.00	1.00	1,121.00	1,121.00
free var, equal cov 2	-2,544.74	10	5,175.79	0.64	0.51	0.09	560.50	107.78
free var, equal cov 3	-2,256.95	15	4,643.36	0.68	0.54	0.06	373.67	82.85
free var, equal cov 4	-2,068.70	20	4,310.02	0.63	0.52	0.02	280.25	30.59
equal var, free cov 1	-3,388.56	5	6,820.28	1.00	1.00	1.00	1,121.00	1,121.00
equal var, free cov 2	-2,551.86	9	5,181.40	0.65	0.52	0.09	622.78	121.25
equal var, free cov 3	-2,359.39	13	4,830.99	0.68	0.56	0.07	431.15	99.55
equal var, free cov 4	-2,173.85	17	4,494.44	0.63	0.60	0.02	329.71	33.43
free var, free cov 1	-3,388.56	5	6,820.28	1.00	1.00	1.00	1,121.00	1,121.00
free var, free cov 2	-2,574.87	11	5,244.69	0.56	0.81	0.40	509.55	447.60
free var, free cov 3	-2,111.38	17	4,369.50	0.65	0.51	0.04	329.71	40.80
free var, free cov 4	-2,024.10	23	4,246.73	0.62	0.50	0.02	243.70	17.80

However, note that we have a very large sample, and for many models, the smallest class comprises only a very small percentage of the total sample. Since the purpose of this analysis is to better represent the distribution of ocean microplastics, we can wonder whether it makes sense to allow for classes that only describe a small percentage of the cases. We therefore only consider solutions that capture at least 10% of the sample.

Another interesting characteristic of this data is that the BIC and the entropy are strongly correlated. The raw correlation between these two metrics is .66, `cor(fit$BIC, fit$Entropy)`. If we omit the 1-class models, for which entropy is technically not defined, the correlation is even as high as .85, `cor(fit$BIC[!fit$Classes == 1], fit$Entropy[!fit$Classes == 1])`.

This strong correlation indicates that an increase in fit comes with a decrease in class separability. This illustrates why entropy should not be treated as a model fit criterion. It also illustrates that criteria for class enumeration should be explicit, because we will likely come to a different decision depending on which criteria are used.

As mentioned before, we drop models with $< 10\%$ of cases in the smallest class:

```
fit <- fit[!fit$n_min < 0.1, ]
```

If our strategy is to optimize fit, we can examine the fit table above, or plot a scree plot for the BIC by calling `plot(fit)`. Note that, due to the large sample size, all ICs give identical conclusions.

```
plot(fit) + theme(axis.text.x = element_text(angle = 90, vjust = 0.5,
    hjust = 1))
```

Looking at the blocks of 1-4 class models for each model specification, it appears that the BIC keeps decreasing with the addition of more classes. Across the blocks, the BIC keeps decreasing with increasingly complex model specifications.

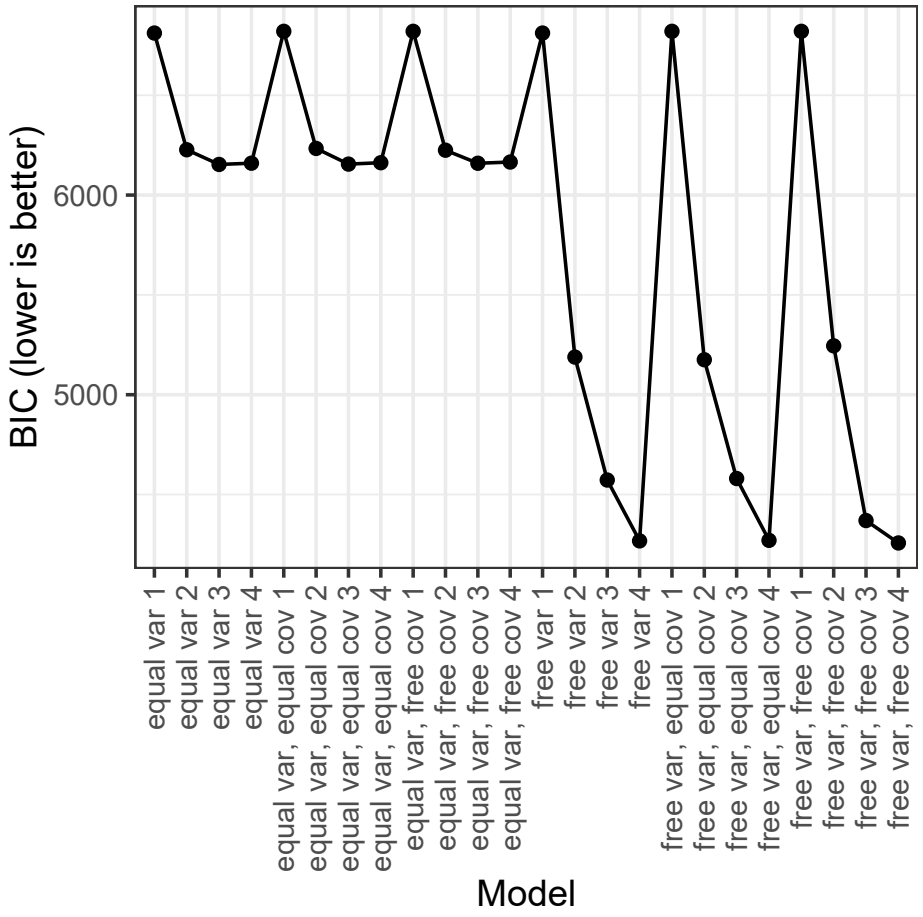


Figure 1. Bivariate profile plot

Out of the 16 models that remain after removing those with $< 10\%$ of cases in the smallest class, one model stands out: The 2-class model with free (co)variances. We thus select this as our final model.

Interpreting the Final Class Solution

We here request the estimates (`est`) and standardized estimates `std_est`, because the latter allows us to interpret the correlations between length and width. Note that standard errors and p-values are relatively uninformative: With a sample size of 5606, every parameter is significantly different from zero.

Table 3

Results of a 2-class model with free (co)variances

label	est	std_est
Means.length	0.65	2.04
Means.width	0.34	1.07
Variances.length	0.10	1.00
Covariances.length.WITH.width	0.09	0.92
Variances.width	0.10	1.00
Means.length	1.33	3.03
Means.width	0.86	1.64
Variances.length	0.19	1.00
Covariances.length.WITH.width	0.20	0.85
Variances.width	0.28	1.00
mix2.weights[1,2]	0.75	NA

```
res_bic <- res[["free var, free cov 2"]]
cp <- class_prob(res_bic)
results <- table_results(res_bic, columns = c("label", "est",
    "std_est"))
results
```

Interpreting the results is facilitated by examining a plot of the model and data. Relevant plot functions are `plot_bivariate()`, `plot_density()`, and `plot_profiles()`. However, we omit the density plots, because `plot_bivariate()` also includes them.

```
plot_bivariate(res_bic)
```

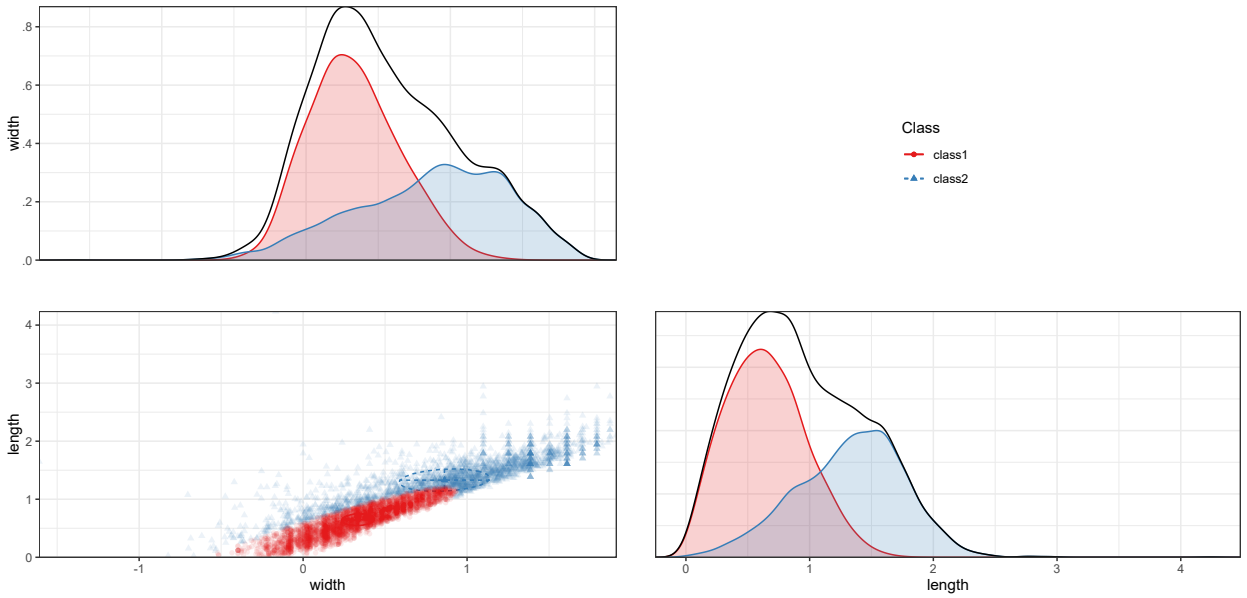


Figure 2. Bivariate profile plot

On the diagonal of the bivariate plot are weighted density plots: normal approximations of the density function of observed data, weighed by class probability. On the off-diagonal are plots for each pair of indicators, with the class means indicated by a point, class standard deviations indicated by lines, and covariances indicated by circles.

The bivariate and marginal plots show that the classes are not clearly separable, as also evident from the low entropy. At the same time however, it is clear that the observed distributions are non-normal, and the second class accounts for some of this non-normality (there is a smaller 'bump' to the right of the mode, which could be the mean of a second normal distribution). The first class (57%) accounts for smaller fragments, and the second class (43%) accounts for some of the right-skew in fragments' length and width. We label class 1 as *small fragments*, and class 2 as *larger fragments*.

It also appears that the correlation between length and width is stronger for small fragments than for large fragments. To test the difference, use `wald_test(res_bic,`

`hypothesis = "c11 = c21"`). Results indicate that the correlation is indeed significantly larger for small fragments ($r = .92$) than for larger fragments ($r = .85$), $\chi^2(1) = 11.56, p < .001$. Thus, small fragments are more coextensive than large fragments.

There are, however, concerns about the interpretability of this solutions: the entropy is `.56` and the minimum classification probability is `.81`. This is because of substantial overlap in the distributions of the two classes.

Auxiliary Analyses

Finally, we may want to compare the different classes on auxiliary variables or models. The `BCH()` function applies three-step analysis, which compares the classes using a multi-group model, controlling for classification error. For example, we can test whether polymer type differs between the two classes. Because polymer type is a nominal variable, we must convert it to dummies and estimate a threshold for each dummy:

```
df_pt <- mx_dummies(df_analyze$poly_type)
aux_pt <- BCH(res_bic, model = "poly_typeOther | t1
                                poly_typePE | t1
                                poly_typePP | t1",
    data = df_pt)
aux_pt <- mxTryHardOrdinal(aux_pt)
```

To obtain an omnibus likelihood ratio test of the significance of the differences in polymer type across classes, use `lr_test(aux_pt)`. The results indicate that there are significant differences in polymer types across classes, $\Delta LL(3) = 17.14, p < .001$. The results can be reported in probability scale using `table_prob(aux_pt)`. To test differences for specific polymer types, we can use Wald tests:

```
wald_test(aux_pt, "class1.Thresholds[1,1] = class2.Thresholds[1,1];
          class1.Thresholds[1,2] = class2.Thresholds[1,2];
          class1.Thresholds[1,3] = class2.Thresholds[1,3]")
```

The results indicate that there is no significant difference in the prevalence of "Other" polymer types across classes. However, PE is significantly more prevalent in class 1, and PP is significantly more prevalent in class 2.

R session

```
sessionInfo()
#> R version 4.2.2 (2022-10-31 ucrt)
#> Platform: x86_64-w64-mingw32/x64 (64-bit)
#> Running under: Windows 10 x64 (build 19045)
#>
#> Matrix products: default
#>
#> locale:
#> [1] LC_COLLATE=English_United States.utf8
#> [2] LC_CTYPE=C
#> [3] LC_MONETARY=English_United States.utf8
#> [4] LC_NUMERIC=C
#> [5] LC_TIME=English_United States.utf8
#>
#> attached base packages:
#> [1] stats     graphics  grDevices utils     datasets
#> [6] methods   base
```

```
#>
#> other attached packages:
#>  [1] qpdf_1.3.0       MASS_7.3-58.1
#>  [3] scales_1.2.1     yaml_2.3.7
#>  [5] papaja_0.1.1     tinylabels_0.2.3
#>  [7] progressr_0.13.0 future_1.32.0
#>  [9] ggplot2_3.4.2    tidySEM_0.2.4.8
#> [11] OpenMx_2.21.8
#>
#> loaded via a namespace (and not attached):
#>   [1] backports_1.4.1       systemfonts_1.0.4
#>   [3] plyr_1.8.8            igraph_1.4.2
#>   [5] MplusAutomation_1.1.0 listenv_0.9.0
#>   [7] usethis_2.1.6         rstantools_2.3.1
#>   [9] inline_0.3.19         digest_0.6.31
#>  [11] htmltools_0.5.4       rsconnect_0.8.29
#>  [13] fansi_1.0.4           magrittr_2.0.3
#>  [15] checkmate_2.1.0       gert_1.9.2
#>  [17] credentials_1.3.2     globals_0.16.2
#>  [19] RcppParallel_5.1.7    matrixStats_0.63.0
#>  [21] rmdfiltr_0.1.3        sandwich_3.0-2
#>  [23] svglite_2.1.1         askpass_1.1
#>  [25] prettyunits_1.1.1     colorspace_2.1-0
#>  [27] textshaping_0.3.6     xfun_0.37
#>  [29] dplyr_1.1.1           callr_3.7.3
#>  [31] crayon_1.5.2          jsonlite_1.8.4
#>  [33] zoo_1.8-12            glue_1.6.2
```

```
#>  [35] prereg_0.6.0          gtable_0.3.3
#>  [37] V8_4.2.2              car_3.1-2
#>  [39] pkgbuild_1.4.0        rstan_2.26.16
#>  [41] future.apply_1.10.0   abind_1.4-5
#>  [43] bain_0.2.8            mvtnorm_1.1-3
#>  [45] rstatix_0.7.2         Rcpp_1.0.10
#>  [47] xtable_1.8-4          progress_1.2.2
#>  [49] tmvnsim_1.0-2         stats4_4.2.2
#>  [51] StanHeaders_2.26.16   worcs_0.1.10
#>  [53] httr_1.4.5            RColorBrewer_1.1-3
#>  [55] lavaan_0.6-15         ellipsis_0.3.2
#>  [57] pkgconfig_2.0.3       loo_2.6.0
#>  [59] farver_2.1.1          sass_0.4.5
#>  [61] utf8_1.2.3            tidyselect_1.2.0
#>  [63] labeling_0.4.2        rlang_1.1.0
#>  [65] munsell_0.5.0         tools_4.2.2
#>  [67] cachem_1.0.6          cli_3.6.0
#>  [69] dbscan_1.1-11         gsubfn_0.7
#>  [71] generics_0.1.3        ranger_0.14.1
#>  [73] broom_1.0.4           evaluate_0.20
#>  [75] fastmap_1.1.0         ragg_1.2.5
#>  [77] sys_3.4.1             rticles_0.24
#>  [79] blavaan_0.4-7         fs_1.6.1
#>  [81] processx_3.8.0        knitr_1.42
#>  [83] pander_0.6.5          purrr_1.0.1
#>  [85] RANN_2.6.1            gh_1.3.1
#>  [87] nlme_3.1-160          formatR_1.14
```

```
#>  [89] nonnest2_0.5-5        compiler_4.2.2
#>  [91] bayesplot_1.10.0      rstudioapi_0.14
#>  [93] curl_5.0.0            ggsignif_0.6.4
#>  [95] tibble_3.2.1          bslib_0.4.2
#>  [97] pbivnorm_0.6.0        highr_0.10
#>  [99] ps_1.7.2              lattice_0.20-45
#> [101] texreg_1.38.6         Matrix_1.5-1
#> [103] psych_2.3.3           vctrs_0.6.2
#> [105] CompQuadForm_1.4.3    pillar_1.9.0
#> [107] lifecycle_1.0.3       jquerylib_0.1.4
#> [109] data.table_1.14.8     R6_2.5.1
#> [111] bookdown_0.32         renv_0.16.0
#> [113] gridExtra_2.3         parallelly_1.35.0
#> [115] codetools_0.2-18      boot_1.3-28
#> [117] fastDummies_1.6.3     assertthat_0.2.1
#> [119] proto_1.0.0           openssl_2.0.6
#> [121] withr_2.5.0           mnormt_2.1.1
#> [123] parallel_4.2.2        hms_1.1.2
#> [125] quadprog_1.5-8        grid_4.2.2
#> [127] tidyr_1.3.0           coda_0.19-4
#> [129] rmarkdown_2.20        carData_3.0-5
#> [131] ggpubr_0.6.0          tinytex_0.44
```

Online Supplementary Materials B: Confirmatory LPA for the Caregiver Compass

Online Supplementary Materials B: Confirmatory LPA for the Caregiver Compass

This is an example of confirmatory LPA using `tidySEM`. The simulated data are based on work by Zegwaard and colleagues, who sought to establish a typology of caregivers who support a close other receiving outpatient psychological care. Qualitative research among experts resulted in a theory postulating the existence of four types of caregivers (translated from the original Dutch):

Balanced

The balanced caregiver experiences relative balance between the costs and benefits of caring for a close other.

Imbalanced

The imbalanced caregiver experiences a precarious balance between the costs and benefits of caring for a close other.

Lonely

The lonely caregiver experiences a strong sense of isolation.

Entrapped

The entrapped caregiver strongly feels a sense of being entangled in responsibilities which are difficult to fulfill.

The goal of this confirmatory study was to validate this hypothesized class solution in a sample of caregivers. A convenience sample was used, with no prior sample size justification. To view the data documentation, run the command `?tidySEM::zegwaard_carecompass` in the R console.

Loading the Data

To load the data, simply attach the `tidySEM` package. For convenience, we assign the variables used for analysis to an object called `df`. We first only use the four scales: `c("burdened", "trapped", "negaffect", "loneliness")`.

```
# Load required packages
library(tidySEM)
library(ggplot2)
# Load data
df <- zegwaard_carecompass[, c("burdened", "trapped", "negaffect",
    "loneliness")]
```

Descriptive statistics

We use `tidySEM::descriptives()` to describe the data numerically. Because all scales are continuous, we select only columns for continuous data to de-clutter the table:

```
desc <- tidySEM::descriptives(df)
desc <- desc[, c("name", "n", "missing", "unique", "mean", "median",
    "sd", "min", "max", "skew_2se", "kurt_2se")]
desc
```

The table indicates two potential causes for concern: there is a small percentage of missingness, and all variables have relatively high kurtosis. Since there are some missing values, we can conduct an MCAR test using `mice::mcar(df)`. According to Hawkins' test, there is no evidence to reject the assumptions of multivariate normality and MCAR, $\tilde{\chi^2}(6) = 3.78, \tilde{p} = 0.71$. Missing data will be accounted for using FIML.

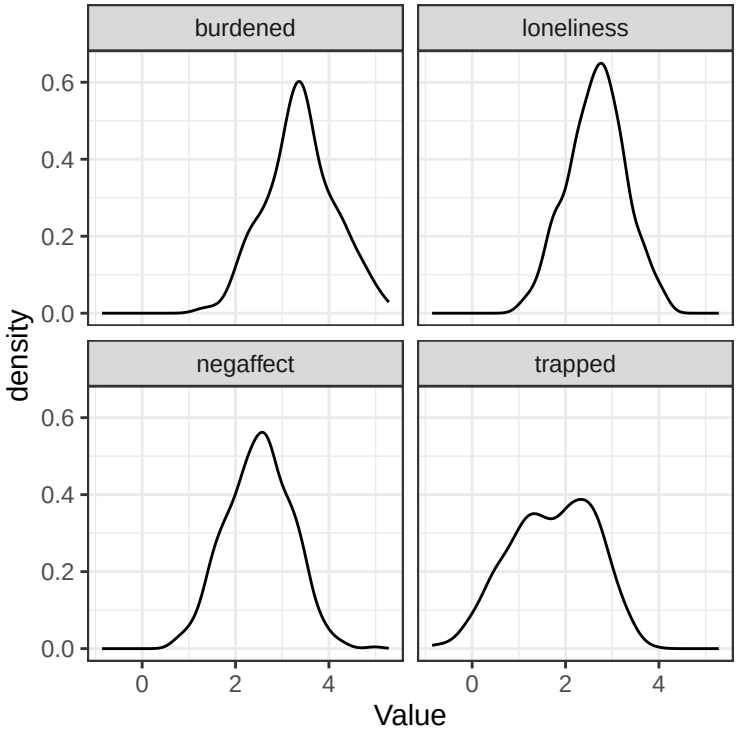
Table 1

Descriptive statistics

name	n	missing	unique	mean	median	sd	min	max	skew_2se	kurt_2se
burdened	509	0.01	509	3.38	3.39	0.75	1.20	5.28	0.17	6.51
trapped	505	0.02	505	1.73	1.79	0.90	-0.86	3.81	-1.03	5.39
negaffect	506	0.01	506	2.50	2.52	0.69	0.71	4.98	0.08	6.52
loneliness	510	0.01	510	2.66	2.69	0.62	0.98	4.22	-0.33	6.31

Additionally, we can plot the data. The `ggplot2` function `geom_density()` is useful for continuous data. Visual inspection confirms the conclusions from the `descriptives()` table: the data are kurtotic (peaked).

```
df_plot <- df
names(df_plot) <- paste0("Value.", names(df_plot))
df_plot <- reshape(df_plot, varying = names(df_plot), direction = "long",
    timevar = "Variable")
ggplot(df_plot, aes(x = Value)) + geom_density() + facet_wrap(~Variable) +
    theme_bw()
```



Conducting Latent Profile Analysis

As all variables are continuous, we can use the convenience function `tidySEM::mx_profiles()`, which is a wrapper for the generic function `mx_mixture()` optimized for continuous indicators. Its default settings are appropriate for LPA, assuming fixed variances across classes and zero covariances. Its arguments are `data` and number of `classes`. All variables in `data` are included in the analysis, which is why we first selected the indicator variables. As this is a confirmatory LCA, we do not follow a strictly data-driven class enumeration procedure. We will set the maximum number of classes K to one more than the theoretically expected number. We set a seed to ensure replicable results.

```
set.seed(123)
res <- mx_profiles(data = df, classes = 1:5)
```

This analysis should produce some messages about cluster initialization. These relate

to the selection of starting values, which relies on the K-means algorithm and is not robust to missing data. The algorithm automatically switches to hierarchical clustering, no further action is required.

Class Enumeration

To compare the fit of the theoretical model against other models, we create a model fit table using `table_fit()` and retain relevant columns. We also determine whether any models can be disqualified.

In this example, all models converge without issues. If, for example, the two-class solution had not converged, we could use the function `res[[2]] <- mxTryHard(res[[2]])` to aid convergence.

Next, we check for local identifiability. The sample size is consistently reported as 513, which means that partially missing cases were indeed included via FIML. The smallest class size occurs in the 5-class model, where the smallest class is assigned 7% of cases, or 38 cases. This model has 28 parameters, approximately 6 per class. We thus have at least five observations per parameter in every class, and do not disqualify the 5-class model.

There are concerns about theoretical interpretability of all solutions, as the entropies and minimum classification probabilities are all low. However, in this confirmatory use case, we address this when interpreting the results.

Using ICs. the 4-class solution has the lowest BIC, which means it is preferred over all other solutions including a 1-class solution and a solution with more classes. Note that a scree plot for the BIC can be plotted by calling `plot(fit)`. Following the elbow criterion, a three-class solution would also be defensible. The function `ic_weights(tab_compare)` allows us to compute IC weights; it indicates that, conditional on the set of models, the 4-class model has a posterior model probability of nearly 100%.

Table 2

Model fit table

Name	LL	p	n	BIC	Entropy	p_min	p_max	n_min	n_max
equal var 1	-2,241.98	8	513	4,533.89	1.00	1.00	1.00	1.00	1.00
equal var 2	-2,031.37	13	513	4,143.85	0.74	0.91	0.93	0.42	0.58
equal var 3	-1,951.35	18	513	4,015.02	0.78	0.89	0.91	0.19	0.54
equal var 4	-1,916.25	23	513	3,976.03	0.75	0.81	0.92	0.16	0.34
equal var 5	-1,912.93	28	513	4,000.59	0.70	0.39	0.92	0.07	0.31

Using LMR tests. If we conduct LMR tests, we find that the tests are significant for all pairwise model comparisons, except for the 5-class model:

```
lr_lmr(fit)
```

Using BLRT tests. We can also use the BLRT test. As it is very computationally expensive, we will use a low number of replications here. In practice, one might use a much higher number (1000+) for published research. Keep in mind that the p-value of the BLRT is subject to Monte Carlo error; if it fluctuates when analyses are replicated or its value is very close to the critical threshold, consider increasing the number of replications.

To accelerate computations, we can use the `future` package for parallel computing (see `?plan` to select the appropriate back-end for your system). To track the function's progress, we use the `progressr` ecosystem, which allows users to choose how they want to be informed. The example below uses a progress bar:

Table 3

LMR test table

Null	Alt	lr	lmr_lr	df	lmr_p
	equal var 1	NA	NA	NA	NA
equal var 1	equal var 2	421.23	399.87	5.00	0.00
equal var 2	equal var 3	160.03	151.92	5.00	0.00
equal var 3	equal var 4	70.19	66.63	5.00	0.00
equal var 4	equal var 5	6.64	6.31	5.00	0.28

```
library(future)
library(progressr)
plan(multisession)  # Parallel processing for Windows
handlers("progress")  # Progress bar
set.seed(1)
res_blrt <- BLRT(res, replications = 20)
```

In sum, across all class enumeration criteria, there is strong support for a 4-class solution.

Optional: Alternative Model Specifications

In the case of confirmatory LCA, the theory would be refuted by strong evidence against the hypothesized model and number of classes. In the preceding, we only compared the theoretical model against models with different number of classes. Imagine, however, that a Reviewer argues that variance ought to be freely estimated across classes. We could compare our theoretical model against their competing model as follows. Note that we can put two models into a list to compare them.

Table 4

BLRT test table

null	alt	lr	df	blrt_p	samples
NA	NA	NA	NA	NA	NA
mix1	mix2	421.23	5	0.00	20
mix2	mix3	160.03	5	0.00	20
mix3	mix4	70.19	5	0.00	20
mix4	mix5	6.64	5	0.65	20

```
res_alt <- mx_profiles(df, classes = 4, variances = "varying")
compare <- list(res[[4]], res_alt)
table_fit(compare)
```

The alternative model incurs 12 additional parameters for the free variances. Yet, it has a higher BIC, which indicates that this additional complexity does not outweigh the increase in fit.

Interpreting the Final Class Solution

To interpret the final class solution, we first reorder the 4-class model by class size. This helps prevent label switching.

```
res_final <- mx_switch_labels(res[[4]])
```

The 4-class model yielded classes of reasonable size; using `class_pro`the largest class comprised 33%, and the smallest comprised 16% of cases. However, the entropy was low, $S = .75$, indicating poor class separability. Furthermore, the posterior classification

Table 5

Comparing competing theoretical models

Name	LL	Parameters	BIC	Entropy	prob_min	prob_max	n_min	n_max
1	-1,916.25	23	3,976.03	0.75	0.81	0.92	0.16	0.34
2	-1,909.23	35	4,036.87	0.78	0.84	0.92	0.16	0.32

probability ranged from [.81, .92], which means that at least some classes had a high classification error. We produce a table of the results below.

```
table_results(res_final, columns = c("label", "est", "se", "confint",
    "class"))
```

The results are best interpreted by examining a plot of the model and data, however. Relevant plot functions are `plot_bivariate()`, `plot_density()`, and `plot_profiles()`. However, we omit the density plots, because `plot_bivariate()` also includes them.

```
plot_bivariate(res_final)
```

On the diagonal of the bivariate plot are weighted density plots: normal approximations of the density function of observed data, weighed by class probability. On the off-diagonal are plots for each pair of indicators, with the class means indicated by a point, class standard deviations indicated by lines, and covariances indicated by circles. As this model has zero covariances, all circles are round (albeit warped by the different scales of the X and Y axes)

The marginal density plots show that trappedness distinguishes classes rather well. For all other indicators, groups are not always clearly separated in terms of marginal

Table 6

Four-class model results

label	est	se	confint	class
mix4.weights[1,2]	0.86	0.15	[0.56, 1.15]	NA
mix4.weights[1,3]	0.66	0.11	[0.44, 0.88]	NA
mix4.weights[1,4]	0.47	0.08	[0.32, 0.63]	NA
Variances.burdened	0.23	0.02	[0.19, 0.27]	class1
Variances.trapped	0.17	0.02	[0.14, 0.20]	class1
Variances.negaffect	0.31	0.02	[0.27, 0.36]	class1
Variances.loneliness	0.24	0.02	[0.20, 0.28]	class1
Means.burdened	3.27	0.04	[3.18, 3.36]	class1
Means.trapped	1.28	0.05	[1.18, 1.38]	class1
Means.negaffect	2.31	0.06	[2.20, 2.42]	class1
Means.loneliness	2.73	0.04	[2.64, 2.82]	class1
Means.burdened	3.40	0.06	[3.28, 3.52]	class2
Means.trapped	2.27	0.06	[2.15, 2.38]	class2
Means.negaffect	2.81	0.06	[2.70, 2.93]	class2
Means.loneliness	2.79	0.06	[2.66, 2.91]	class2
Means.burdened	4.25	0.07	[4.12, 4.38]	class3
Means.trapped	2.67	0.05	[2.58, 2.77]	class3
Means.negaffect	2.92	0.06	[2.80, 3.03]	class3
Means.loneliness	2.01	0.06	[1.89, 2.14]	class3
Means.burdened	2.38	0.06	[2.26, 2.50]	class4
Means.trapped	0.38	0.05	[0.28, 0.49]	class4
Means.negaffect	1.78	0.07	[1.65, 1.91]	class4
Means.loneliness	3.18	0.06	[3.07, 3.30]	class4

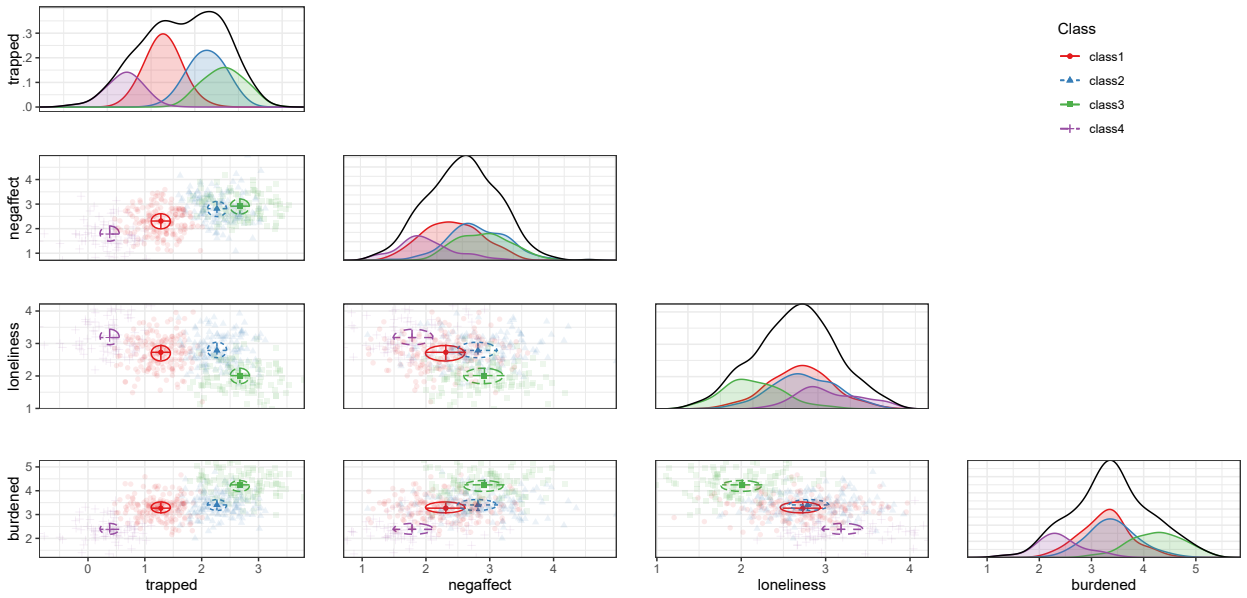


Figure 1. Bivariate profile plot

density: class 2 and 3 coalesce on negative affect, 1 and 2 coalesce on loneliness, and 1 and 2 coalesce on burden. Nevertheless, the off-diagonal scatterplots show reasonable bivariate separation for all classes.

We can obtain a more classic profile plot using `plot_profiles(res_final)`. This plot conveys less information than the bivariate plot, but is readily interpretable. Below is a comparison between the most common type of visualization for LPA, and the best-practices visualization provided by `tidySEM`. Note that the best practices plot includes class means and error bars, standard deviations, and a ribbon plot of raw data weighted by class probability to indicate how well the classes describe the observed distribution. The overlap between the classes is clearly visible in this figure; this is why the entropy and classification probabilities are relatively low.

Based on the bivariate plot, we can label class 1 as the *balanced* type (33%), class 2 as the *imbalanced* type (29%), class 3 as the *entrapped* type (22%), and class 4 as the *lonely* type (16%). Note however that the observed classes do not match the hypothesized pattern of class parameters exactly.

```
plot_profiles(res_final)
```

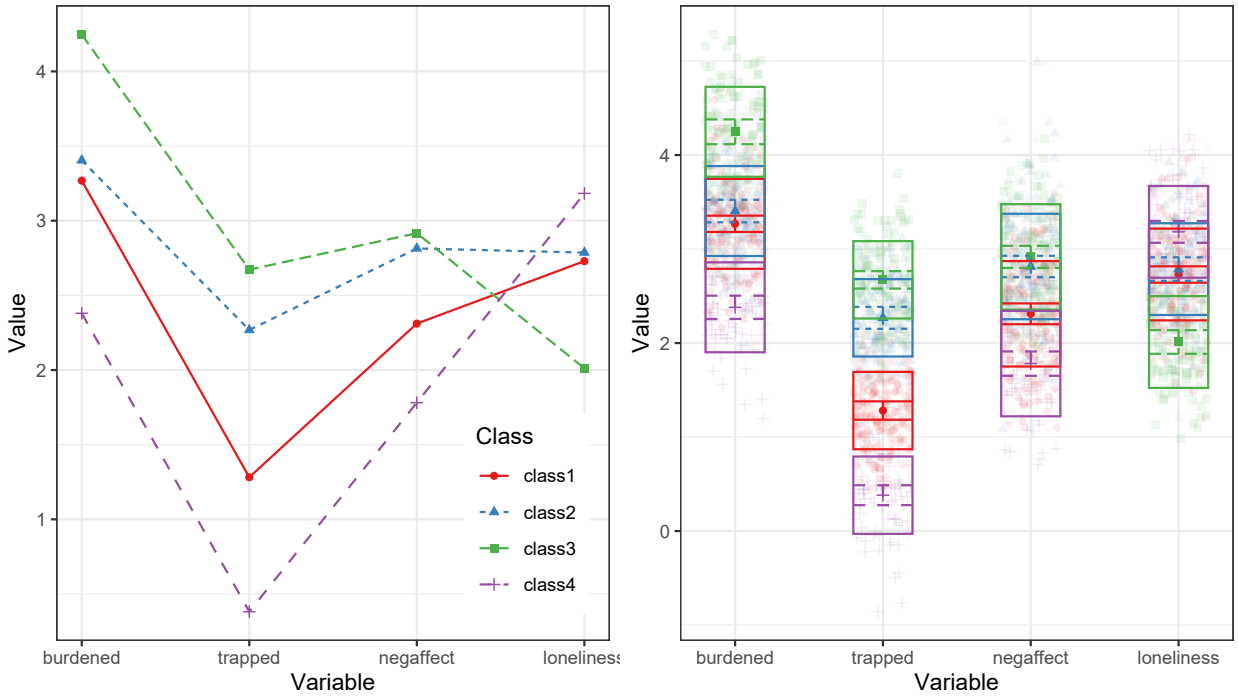


Figure 2. Bivariate profile plot

Auxiliary Analyses

We may want to compare the different classes on auxiliary variables or models. The `BCH()` function applies three-step analysis, which compares the classes using a multi-group model, controlling for classification error. We consider two examples: a single variable, and an auxiliary model.

Comparing Means or Proportions Across Classes. For a single (continuous or ordinal) variable, we can call the BCH function and simply supply the auxiliary variable to the `data` argument, omitting the `model` argument. Below, we estimate an auxiliary model to compare the sex of patients between classes:

```
aux_sex <- BCH(res_final, data = zegwaard_carecompass$sexpatient)
```

To obtain an omnibus likelihood ratio test of the significance of these sex differences across classes, as well as pairwise comparisons between classes, use `lr_test(aux_sex)`. The results indicate that there are significant sex differences across classes, $\Delta LL(1) = 8.7, p = .003$. Pairwise comparisons indicate that class 3 differs significantly from classes 1 and 2. The results can be reported in probability scale using `table_prob(aux_sex)`. It appears that the entrapped class disproportionately cares for female patients.

Comparing Auxiliary Models Across Classes. We can also compare a simple model between classes. Specifically, we will examine whether the distance predicts the frequency of visits differently across classes (treated as continuous).

```
df_aux <- zegwaard_carecompass[, c("freqvisit", "distance")]
df_aux$freqvisit <- as.numeric(df_aux$freqvisit)
aux_model <- BCH(res_final, model = "freqvisit ~ distance", data = df_aux)
```

To obtain an omnibus likelihood ratio test of the difference in regression coefficients across classes and pairwise comparisons between classes, use `lr_test(aux_model, compare = "A")`. The results indicate that there are no significant sex differences across classes, $\Delta LL(3) = 0.98, p = .81$. The results can be reported using `table_results(aux_model)`.

Predicting class membership

This LCA model was developed to help classify care providers in a clinical context, so that mental healthcare professionals can provide tailored support to those who take care of their clients. In `tidySEM`, it is possible to predict class membership for new data. Imagine

that we administer the care compass questionnaire to a new individual. We can assign their scale scores to a `data.frame`, and supply it to the `predict()` function via the `newdata` argument. The result includes the individual's most likely class, as well as posterior probabilities for all classes.

```
df_new <- data.frame(burdened = 2, trapped = 0.5, negaffect = 1.5,
    loneliness = 4)
predict(res_final, newdata = df_new)
```

```
#>       class1  class2 class3  class4 predicted
#> [1,] 0.00081 1.4e-15      1 4.6e-08         3
```

R session

```
sessionInfo()
#> R version 4.2.2 (2022-10-31 ucrt)
#> Platform: x86_64-w64-mingw32/x64 (64-bit)
#> Running under: Windows 10 x64 (build 19045)
#>
#> Matrix products: default
#>
#> locale:
#> [1] LC_COLLATE=English_United States.utf8
#> [2] LC_CTYPE=C
#> [3] LC_MONETARY=English_United States.utf8
#> [4] LC_NUMERIC=C
#> [5] LC_TIME=English_United States.utf8
#>
```

```
#> attached base packages:
#> [1] stats     graphics  grDevices utils     datasets
#> [6] methods   base
#>
#> other attached packages:
#>  [1] qpdf_1.3.0       MASS_7.3-58.1
#>  [3] scales_1.2.1     yaml_2.3.7
#>  [5] papaja_0.1.1     tinylabels_0.2.3
#>  [7] progressr_0.13.0 future_1.32.0
#>  [9] ggplot2_3.4.2    tidySEM_0.2.4.8
#> [11] OpenMx_2.21.8
#>
#> loaded via a namespace (and not attached):
#>   [1] backports_1.4.1       systemfonts_1.0.4
#>   [3] plyr_1.8.8            igraph_1.4.2
#>   [5] MplusAutomation_1.1.0 listenv_0.9.0
#>   [7] usethis_2.1.6         rstantools_2.3.1
#>   [9] inline_0.3.19         digest_0.6.31
#>  [11] htmltools_0.5.4       rsconnect_0.8.29
#>  [13] fansi_1.0.4           magrittr_2.0.3
#>  [15] checkmate_2.1.0       gert_1.9.2
#>  [17] credentials_1.3.2     globals_0.16.2
#>  [19] RcppParallel_5.1.7    matrixStats_0.63.0
#>  [21] rmdfiltr_0.1.3        sandwich_3.0-2
#>  [23] svglite_2.1.1         askpass_1.1
#>  [25] prettyunits_1.1.1     colorspace_2.1-0
#>  [27] textshaping_0.3.6     xfun_0.37
```

```
#>  [29] dplyr_1.1.1           callr_3.7.3
#>  [31] crayon_1.5.2          jsonlite_1.8.4
#>  [33] zoo_1.8-12            glue_1.6.2
#>  [35] prereg_0.6.0          gtable_0.3.3
#>  [37] V8_4.2.2              car_3.1-2
#>  [39] pkgbuild_1.4.0        rstan_2.26.16
#>  [41] future.apply_1.10.0   abind_1.4-5
#>  [43] bain_0.2.8            mvtnorm_1.1-3
#>  [45] rstatix_0.7.2         Rcpp_1.0.10
#>  [47] xtable_1.8-4          progress_1.2.2
#>  [49] tmvnsim_1.0-2         stats4_4.2.2
#>  [51] StanHeaders_2.26.16   worcs_0.1.10
#>  [53] httr_1.4.5            RColorBrewer_1.1-3
#>  [55] lavaan_0.6-15         ellipsis_0.3.2
#>  [57] pkgconfig_2.0.3       loo_2.6.0
#>  [59] farver_2.1.1          sass_0.4.5
#>  [61] utf8_1.2.3            tidyselect_1.2.0
#>  [63] labeling_0.4.2        rlang_1.1.0
#>  [65] munsell_0.5.0         tools_4.2.2
#>  [67] cachem_1.0.6          cli_3.6.0
#>  [69] dbscan_1.1-11         gsubfn_0.7
#>  [71] generics_0.1.3        ranger_0.14.1
#>  [73] broom_1.0.4           evaluate_0.20
#>  [75] fastmap_1.1.0         ragg_1.2.5
#>  [77] sys_3.4.1             rticles_0.24
#>  [79] blavaan_0.4-7         fs_1.6.1
#>  [81] processx_3.8.0        knitr_1.42
```

```
#>  [83] pander_0.6.5          purrr_1.0.1
#>  [85] RANN_2.6.1            gh_1.3.1
#>  [87] nlme_3.1-160          formatR_1.14
#>  [89] nonnest2_0.5-5        compiler_4.2.2
#>  [91] bayesplot_1.10.0      rstudioapi_0.14
#>  [93] curl_5.0.0            ggsignif_0.6.4
#>  [95] tibble_3.2.1          bslib_0.4.2
#>  [97] pbivnorm_0.6.0        highr_0.10
#>  [99] ps_1.7.2              lattice_0.20-45
#> [101] texreg_1.38.6         Matrix_1.5-1
#> [103] psych_2.3.3           vctrs_0.6.2
#> [105] CompQuadForm_1.4.3    pillar_1.9.0
#> [107] lifecycle_1.0.3       jquerylib_0.1.4
#> [109] data.table_1.14.8     R6_2.5.1
#> [111] bookdown_0.32         renv_0.16.0
#> [113] gridExtra_2.3         parallelly_1.35.0
#> [115] codetools_0.2-18      boot_1.3-28
#> [117] fastDummies_1.6.3     assertthat_0.2.1
#> [119] proto_1.0.0           openssl_2.0.6
#> [121] withr_2.5.0           mnormt_2.1.1
#> [123] parallel_4.2.2        hms_1.1.2
#> [125] quadprog_1.5-8        grid_4.2.2
#> [127] tidyr_1.3.0           coda_0.19-4
#> [129] rmarkdown_2.20        carData_3.0-5
#> [131] ggpubr_0.6.0          tinytex_0.44
```

Online Supplementary Materials C: Latent Class Analysis for Ordinal Indicators

Online Supplementary Materials C: Latent Class Analysis for Ordinal Indicators

This is an example of exploratory LCA with ordinal indicators using `tidySEM`. The present example uses synthetic data based on a study by Maene and colleagues. In a convenience sample of Flemish (Belgian) high-school students with a migration background, this study set out to identify distinct classes based on ordinal indicators of National, Regional, and Heritage identity. Sample size was not justified.

The approach to class enumeration was semi-theory driven: The researchers expected to find profiles that were distinct on all three types of identity (national, regional, and heritage) - but the exact number of classes was not pre-specified (hypothesis 1).

Hypothesis 2 stated that adolescents who are nationally integrated would have lower depressive feelings than students from students with other combinations of identifications (hypothesis 2). Hypothesis 3 was that, for assimilated and separated adolescents, there would not be a significant effect of perceived teacher discrimination on depressive symptoms.

Use the command `?tidySEM::maene_identity` to view the data documentation.

Loading the Data

To load the data, simply attach the `tidySEM` package. For convenience, we assign the indicator data to an object called `df`:

```
# Load required packages
library(tidySEM)
library(ggplot2)
# Load data
df <- maene_identity[1:5]
```

Table 1

Descriptive statistics for ordinal items

name	type	n	missing	unique	mode	mode_value	v
Ethnic_1	ordered, factor	439	0	6	269	5	0.53
Ethnic_2	ordered, factor	439	0	6	272	5	0.52
Ethnic_3	ordered, factor	439	0	6	262	5	0.54
Belgian	ordered, factor	439	0	11	88	7	0.86
Flemish	ordered, factor	439	0	11	90	7	0.87

Examining the Data

We use `tidySEM::descriptives()` to describe the data numerically. Because all items are categorical, we remove columns for continuous data to de-clutter the table:

```
desc <- tidySEM::descriptives(df)
desc <- desc[, c("name", "type", "n", "missing", "unique", "mode",
    "mode_value", "v")]
desc
```

Additionally, we can plot the data. The `ggplot2` function `geom_bar()` is useful for ordinal data:

```
df_plot <- df
names(df_plot) <- paste0("Value.", names(df_plot))
df_plot <- reshape(df_plot, varying = names(df_plot), direction = "long")
ggplot(df_plot, aes(x = Value)) + geom_bar() + facet_wrap(~time,
    scales = "free") + theme_bw()
```

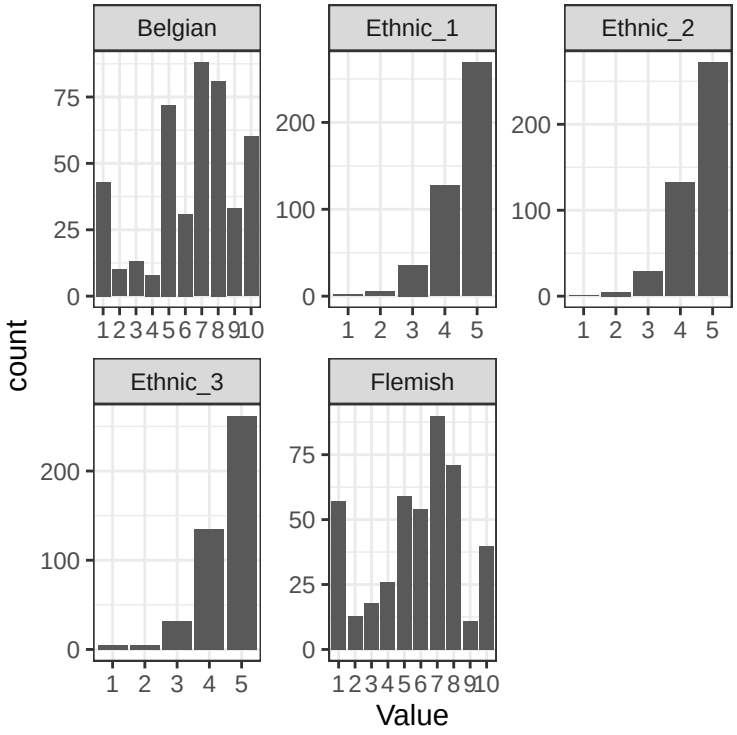


Figure 1. Bar charts for ordinal indicators

As we can see, the `descriptives()` table provides invaluable information about the measurement level of the indicators, which is used to specify the model correctly. If these data had not been coded as ordinal factors, these descriptive statistics would have revealed that each variable has only 5-10 unique values. The proportion of missing values is reported in the `"missing"` column. If any variables had missing values, we would report an MCAR test with `mice::mcar()` and explain that missing data are accounted for using FIML. In our example, we see that there are no missing values, hence we proceed with our analysis. Note that the ethic identification variables are very right-skewed and some response categories have near-zero frequencies; this can cause problems in model specification and convergence. One potential solution would be to merge small adjacent categories. We will not do this here, however.

Conducting Latent Class Analysis

Before we fit a series of LCA models, we set a random seed using `set.seed()`. This is important because there is some inherent randomness in the estimation procedure, and using a seed ensures that we (and others) can exactly reproduce the results.

Next, we fit the LCA models. As all variables are ordinal, we can use the convenience function `tidySEM::mx_lca()`, which is a wrapper for the generic function `mx_mixture()` optimized for LCA with ordinal data. Any mixture model can be specified through `mx_mixture()`. At the time of writing, there are two other wrapper functions for special cases: `mx_profiles()`, for latent profile analysis, and `mx_growth_mixture()`, for latent growth analysis and growth mixture models. All of these functions have arguments `data` and number of `classes`. All variables in `data` are included in the analysis, so relevant variables must be selected first.

We here consider 1-6 class models, but note that this may be overfit, as some of the indicators have only 5 response categories.

```
set.seed(123)
res <- mx_lca(data = df, classes = 1:6)
```

Class Enumeration

In class enumeration, we want to compare a sequence of LCA models fitted to the data. First, note that all models converged without issues. If this had not been the case, it is possible to aid convergence using `mxTryHardOrdinal()`, which expands the search for optimal parameter values for models with ordinal indicators. It is part of the family of functions based on `mxTryHard()`.

Next, we create a model fit table using `table_fit()` and retain relevant columns. We also determine whether any models can be disqualified.

Table 2

Model fit table

Name	LL	n	Parameters	BIC	Entropy	prob_min	n_min	np_ratio	np_local
1	-3,075.53	439	30	6,333.60	1.00	1.00	1.00	14.63	14.63
2	-2,844.76	439	61	6,060.67	0.87	0.97	0.45	7.20	6.63
3	-2,767.83	439	92	6,095.43	0.91	0.95	0.21	4.77	3.13
4	-2,684.12	439	123	6,116.64	0.91	0.94	0.11	3.57	1.63
5	-2,621.06	439	154	6,179.13	0.95	0.95	0.10	2.85	1.50

```
fit <- table_fit(res)
fit[, c("Name", "LL", "n", "Parameters", "BIC", "Entropy", "prob_min",
    "n_min", "np_ratio", "np_local")]
```

Note that both the global and local ratio of cases to parameters is low; for models of 3 or more classes, there are just a few observations per parameter in the smallest class (see `np_local`). This is a good reason to disqualify those classes. We will eliminate classes 4-6 on those criteria, but in real data applications, the 3-class solution might also be disqualified.

In terms of classification diagnostics, note that the entropy and the minimum classification probabilities are high for all models. This indicates that all classes are distinct.

Class Enumeration

Using BIC. Based on the BIC, we would prefer the 2-class model. This decision is also validated by a scree plot of the BIC, obtained by running `plot(fit)`.

Theoretical considerations. Despite the BIC indicating that the 2-class model is best, upon examining the 2-class and 3-class solution, it was noted that theoretically crucial distinctions between ethnic, regional (Flemish), and national (Belgian) identity were not captured by the 2-class solution but were captured by the 3-class solution. Based on this theoretical consideration, the analysis proceeded with the 3-class solution.

```
res_final <- res[[3]]
```

Interpreting the Final Class Solution

The 3-class model yielded classes of reasonable size; the largest class comprised 33%, and the smallest comprised 16% of cases. The entropy was high, $S = .93$, indicating good class separability. Furthermore, the posterior classification probability ranged from $[.94, .99]$, which means that all classes had low classification error. We can produce a table of results using `table_results(res_final)`. However, the results are thresholds, indicating quantiles of a standardized normal distribution. These may be difficult to interpret. Therefore, we ask for the results in probability scale:

```
tab <- table_prob(res_final)
reshape(tab, direction = "wide", v.names = "Probability", timevar = "group",
    idvar = c("Variable", "Category"))
```

The results can also be interpreted by plotting the response probabilities:

```
plot_prob(res_final, bw = TRUE)
```

Note that the first class (33%) has relatively high identification with the ethnic indicators and relatively low identification with Belgian and Flemish identity. The second class (16%) has moderate identification with Belgian and Flemish identity, and relatively

Table 3

Three-class model results in probability scale

Variable	Category	Probability.class1	Probability.class2	Probability.class3
Ethnic_1	1	0.00	0.01	0.01
Ethnic_1	2	0.00	0.02	0.01
Ethnic_1	3	0.02	0.20	0.00
Ethnic_1	4	0.07	0.52	0.18
Ethnic_1	5	0.90	0.26	0.80
Ethnic_2	1	0.00	0.00	0.01
Ethnic_2	2	0.00	0.03	0.00
Ethnic_2	3	0.00	0.16	0.01
Ethnic_2	4	0.00	0.66	0.11
Ethnic_2	5	1.00	0.15	0.88
Ethnic_3	1	0.00	0.03	0.00
Ethnic_3	2	0.00	0.03	0.00
Ethnic_3	3	0.01	0.18	0.00
Ethnic_3	4	0.05	0.71	0.05
Ethnic_3	5	0.94	0.05	0.95
Belgian	1	0.33	0.05	0.03
Belgian	2	0.11	0.00	0.00
Belgian	3	0.14	0.00	0.00
Belgian	4	0.06	0.02	0.00
Belgian	5	0.36	0.16	0.07
Belgian	6	0.00	0.11	0.07
Belgian	7	0.00	0.27	0.23
Belgian	8	0.00	0.21	0.26
Belgian	9	0.00	0.04	0.15
Belgian	10	0.00	0.15	0.20
Flemish	1	0.36	0.08	0.06
Flemish	2	0.07	0.01	0.03

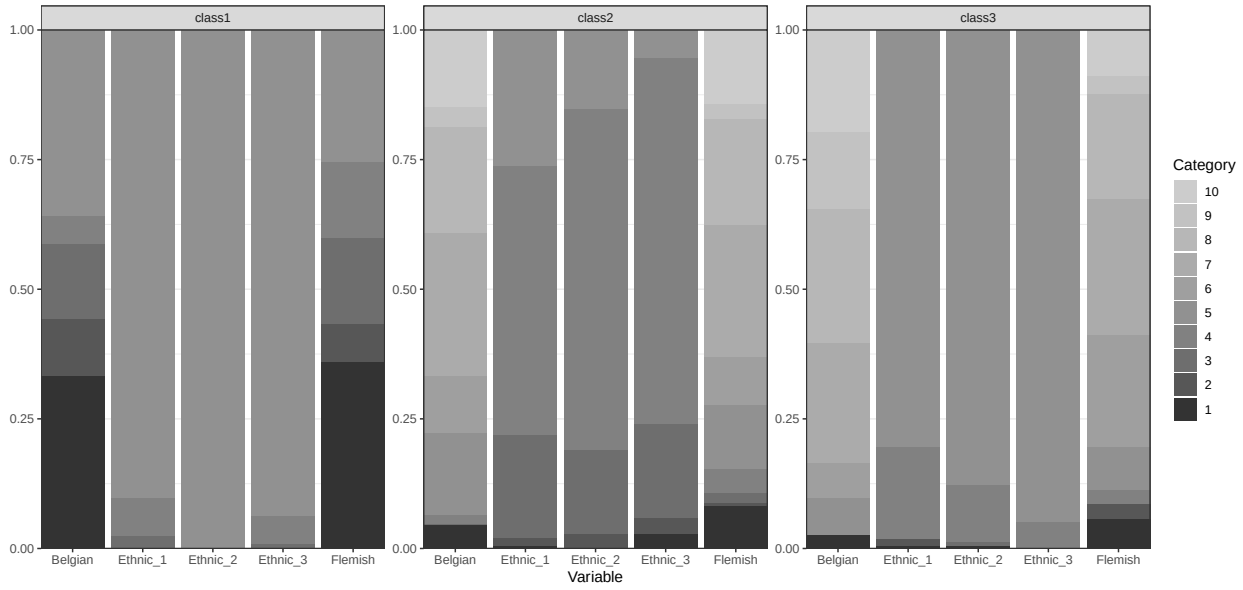


Figure 2. Probability plot

low identification with ethnic identity. Finally, the third class (50%) has high identification with all identities.

Based on the probability plot, we can label class 1 as *ethic identifiers*, class 2 as *low identifiers*, and class 3 as *high identifiers*.

Auxiliary Analyses

To address the remaining two hypotheses, we will perform auxiliary analyses. Hypothesis 2 stated that adolescents who are nationally integrated would have lower depressive feelings than students from students with other combinations of identifications (hypothesis 2).

To test this hypothesis, we can call the BCH function and supply the auxiliary variable depression to the `data` argument, omitting the `model` argument. Below, we estimate an auxiliary model to compare depressive symptoms across classes:

```
aux_dep <- BCH(res_final, data = maene_identity$depression)
```

To obtain an omnibus likelihood ratio test of the significance of depression differences across classes, as well as pairwise comparisons between classes, use `lr_test(aux_dep)`. The results indicate that there are no significant differences in depression across classes, $\Delta LL(5) = 4.32, p = .50$.

Hypothesis 3 was that, for assimilated and separated adolescents, there would not be a significant effect of perceived teacher discrimination on depressive symptoms. To test this hypothesis, we will compare the regression coefficient of discrimination on depressive symptoms across classes.

```
df_aux <- maene_identity[, c("vict_teacher", "depression")]
# Dummy-code vict_teacher
df_aux$vict_teacher <- (as.integer(df_aux$vict_teacher) - 1)
aux_model <- BCH(res_final, model = "depression ~ vict_teacher",
    data = df_aux)
```

To view the coefficients of this model, we can use either `coef(aux_model)` or `table_results(aux_model, columns = NULL)`. As evident from the results table, the coefficients labeled `class1.A[1,2]` are the regression coefficients.

There are two ways to test the difference in regression coefficients across classes: using `lr_test(aux_model, compare = "A")`, to compare the 'A matrix' (regression coefficients) across classes, or `wald_test(aux_model, "class1.A[1,2]=class2.A[1,2]&class1.A[1,2]=class3.A[1,2]")`. The results indicate that there are no significant differences in the regression coefficients across classes, $\chi^2(2) = 1.16, p = .56$.

R session

```
sessionInfo()
#> R version 4.2.2 (2022-10-31 ucrt)
#> Platform: x86_64-w64-mingw32/x64 (64-bit)
#> Running under: Windows 10 x64 (build 19045)
#>
#> Matrix products: default
#>
#> locale:
#> [1] LC_COLLATE=English_United States.utf8
#> [2] LC_CTYPE=C
#> [3] LC_MONETARY=English_United States.utf8
#> [4] LC_NUMERIC=C
#> [5] LC_TIME=English_United States.utf8
#>
#> attached base packages:
#> [1] stats     graphics  grDevices utils     datasets
#> [6] methods   base
#>
#> other attached packages:
#>  [1] qpdf_1.3.0       MASS_7.3-58.1
#>  [3] scales_1.2.1     yaml_2.3.7
#>  [5] papaja_0.1.1     tinylabels_0.2.3
#>  [7] progressr_0.13.0 future_1.32.0
#>  [9] ggplot2_3.4.2    tidySEM_0.2.4.8
#> [11] OpenMx_2.21.8
```

```
#>
#> loaded via a namespace (and not attached):
#>   [1] backports_1.4.1       systemfonts_1.0.4
#>   [3] plyr_1.8.8            igraph_1.4.2
#>   [5] MplusAutomation_1.1.0 listenv_0.9.0
#>   [7] usethis_2.1.6         rstantools_2.3.1
#>   [9] inline_0.3.19         digest_0.6.31
#>  [11] htmltools_0.5.4       rsconnect_0.8.29
#>  [13] fansi_1.0.4           magrittr_2.0.3
#>  [15] checkmate_2.1.0       gert_1.9.2
#>  [17] credentials_1.3.2     globals_0.16.2
#>  [19] RcppParallel_5.1.7    matrixStats_0.63.0
#>  [21] rmdfiltr_0.1.3        sandwich_3.0-2
#>  [23] svglite_2.1.1         askpass_1.1
#>  [25] prettyunits_1.1.1     colorspace_2.1-0
#>  [27] textshaping_0.3.6     xfun_0.37
#>  [29] dplyr_1.1.1           callr_3.7.3
#>  [31] crayon_1.5.2          jsonlite_1.8.4
#>  [33] zoo_1.8-12            glue_1.6.2
#>  [35] prereg_0.6.0          gtable_0.3.3
#>  [37] V8_4.2.2              car_3.1-2
#>  [39] pkgbuild_1.4.0        rstan_2.26.16
#>  [41] future.apply_1.10.0   abind_1.4-5
#>  [43] bain_0.2.8            mvtnorm_1.1-3
#>  [45] rstatix_0.7.2         Rcpp_1.0.10
#>  [47] xtable_1.8-4          progress_1.2.2
#>  [49] tmvnsim_1.0-2         stats4_4.2.2
```

```
#>  [51] StanHeaders_2.26.16   worcs_0.1.10
#>  [53] httr_1.4.5            RColorBrewer_1.1-3
#>  [55] lavaan_0.6-15         ellipsis_0.3.2
#>  [57] pkgconfig_2.0.3       loo_2.6.0
#>  [59] farver_2.1.1          sass_0.4.5
#>  [61] utf8_1.2.3            tidyselect_1.2.0
#>  [63] labeling_0.4.2        rlang_1.1.0
#>  [65] munsell_0.5.0         tools_4.2.2
#>  [67] cachem_1.0.6          cli_3.6.0
#>  [69] dbscan_1.1-11         gsubfn_0.7
#>  [71] generics_0.1.3        ranger_0.14.1
#>  [73] broom_1.0.4           evaluate_0.20
#>  [75] fastmap_1.1.0         ragg_1.2.5
#>  [77] sys_3.4.1             rticles_0.24
#>  [79] blavaan_0.4-7         fs_1.6.1
#>  [81] processx_3.8.0        knitr_1.42
#>  [83] pander_0.6.5          purrr_1.0.1
#>  [85] RANN_2.6.1            gh_1.3.1
#>  [87] nlme_3.1-160          formatR_1.14
#>  [89] nonnest2_0.5-5        compiler_4.2.2
#>  [91] bayesplot_1.10.0      rstudioapi_0.14
#>  [93] curl_5.0.0            ggsignif_0.6.4
#>  [95] tibble_3.2.1          bslib_0.4.2
#>  [97] pbivnorm_0.6.0        highr_0.10
#>  [99] ps_1.7.2              lattice_0.20-45
#> [101] texreg_1.38.6         Matrix_1.5-1
#> [103] psych_2.3.3           vctrs_0.6.2
```

```
#> [105] CompQuadForm_1.4.3    pillar_1.9.0
#> [107] lifecycle_1.0.3       jquerylib_0.1.4
#> [109] data.table_1.14.8     R6_2.5.1
#> [111] bookdown_0.32         renv_0.16.0
#> [113] gridExtra_2.3         parallelly_1.35.0
#> [115] codetools_0.2-18      boot_1.3-28
#> [117] fastDummies_1.6.3     assertthat_0.2.1
#> [119] proto_1.0.0           openssl_2.0.6
#> [121] withr_2.5.0           mnormt_2.1.1
#> [123] parallel_4.2.2        hms_1.1.2
#> [125] quadprog_1.5-8        grid_4.2.2
#> [127] tidyr_1.3.0           coda_0.19-4
#> [129] rmarkdown_2.20        carData_3.0-5
#> [131] ggpubr_0.6.0          tinytex_0.44
```

Online Supplementary Materials D: Latent Class Growth Analysis

Online Supplementary Materials D: Latent Class Growth Analysis

This vignette illustrated `tidySEM`'s ability to perform latent class growth analysis, or growth mixture modeling. The simulated data used for this example are inspired by work in progress by Plas and colleagues, on heterogeneity in depression trajectories among Dutch military personnel who were deployed to Afghanistan. The original data were collected as part of the *Prospection in Stress-related Military Research (PRISMO)* study, which examined of psychological problems after deployment in more than 1,000 Dutch military personnel from 2005-2019.

First, we load all required packages:

```
library(tidySEM)
library(ggplot2)
library(MASS)
```

Data preprocessing

We first examined the descriptive statistics for the sum score scales:

Note that all variables were extremely right-skewed due to censoring at the lower end of the scale.

We can examine these distributions visually as well:

```
df_plot <- reshape(df, direction = "long", varying = names(df))
ggplot(df_plot, aes(x = scl)) + geom_density() + facet_wrap(~time) +
    theme_bw()
```

As this type of skew can result in convergence problems in LCGA, we compared several transformations to reduce skew: The square and cube root, log, inverse, and Box-Cox transformations.

Table 1

Item descriptives

name	mean	median	sd	min	max	skew_2se	kurt_2se
scl.1	20.20	20.00	2.36	17.00	38.00	14.52	40.39
scl.2	20.42	19.00	3.51	16.00	64.00	25.90	111.96
scl.3	20.49	20.00	3.41	17.00	59.00	26.41	107.36
scl.4	20.61	20.00	3.36	16.00	50.00	18.23	54.25
scl.5	20.93	20.00	4.06	16.00	64.00	24.54	93.40
scl.6	21.07	20.00	4.10	16.00	58.00	20.44	65.76

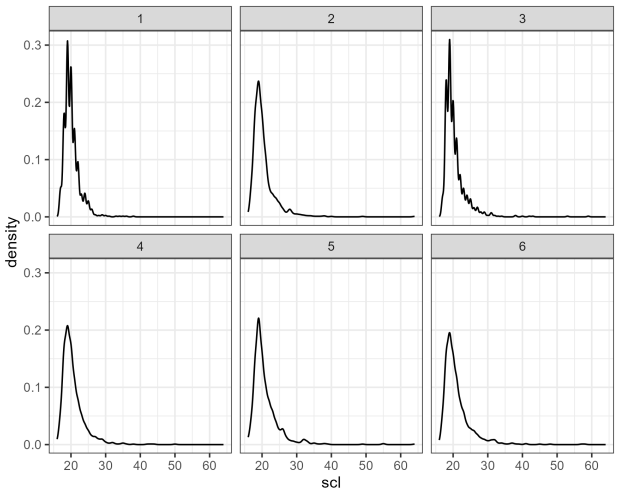


Figure 1

```r
df_scores <- df_plot

# Store original range of SCL

rng_scl <- range(df_scores$scl)

# Log-transform

df_scores$log <- scales::rescale(log(df_scores$scl), to = c(0,
    1))

# Square root transform

df_scores$sqrt <- scales::rescale(sqrt(df_scores$scl), to = c(0,
    1))

# Cube root transform

df_scores$qrt <- scales::rescale(df_scores$scl^0.33, to = c(0,
    1))

# Reciprocal transform

df_scores$reciprocal <- scales::rescale(1/df_scores$scl, to = c(0,
    1))

# Define function for Box-Cox transformation

bc <- function(x, lambda) {
    (((x^lambda) - 1)/lambda)
}

# Inverse Box-Cox transformation

invbc <- function(x, lambda) {
    ((x * lambda) + 1)^(1/lambda)
}

# Box-Cox transform

b <- MASS::boxcox(lm(df_scores$scl ~ 1), plotit = FALSE)

lambda <- b$x[which.max(b$y)]

df_scores$boxcox <- bc(df_scores$scl, lambda)
```

```
# Store range of Box-Cox transformed data
rng_bc <- range(df_scores$boxcox)
df_scores$boxcox <- scales::rescale(df_scores$boxcox, to = c(0,
    1))
# Rescale SCL
df_scores$scl <- scales::rescale(df_scores$scl, to = c(0, 1))
```

We can plot these transformations:

```
# Make plot data
df_plot <- do.call(rbind, lapply(c("scl", "log", "sqrt", "qrt",
    "boxcox"), function(n) {
    data.frame(df_scores[c("time", "id")], Value = df_scores[[n]],
        Transformation = n)
}))
# Plot
ggplot(df_plot, aes(x = Value, colour = Transformation)) + geom_density() +
    facet_wrap(~time) + scale_y_sqrt() + xlab("scl (rescaled to 0-1)") +
    theme_bw()
```

Evidently, the Box-Cox transformation reduced skew the most. Consequently, we proceeded with the Box-Cox transformed scores for analysis.

```
dat <- df_scores[, c("id", "time", "boxcox")]
dat <- reshape(dat, direction = "wide", v.names = "boxcox", timevar = "time",
    idvar = "id")
names(dat) <- gsub("boxcox.", "scl", names(dat))
```

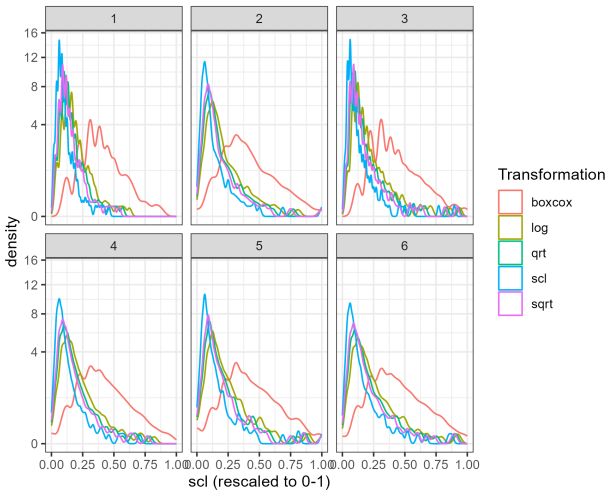


Figure 2

Latent Class Growth Analysis

Next, we estimated a latent class growth model for SCL. The model included an overall intercept, centered at T1, `i`. To model the potential effect of deployment on depresion, we also included a dummy variable that was zero before deployment, and 1 after deployment, `step`. Finally, to model potential change (or recovery) in depression post-deployment, we included a linear slope from T2-T6, `s`. All variances of growth parameters were fixed to zero due to the sparse nature of the data. In this vignette, we do not consider more than 5 classes, because the analyses are computationally very intensive and the data were simulated from a 3-class model.

It is important to highlight that in LCGA, the subgroups will be limited by the specify growth structure, meaning that LCA will identify distinctive growth patterns within the intercept, step, and slope growth. For example, if there is a subgroup that follows a quadratic growth pattern this models will not be able to identify it.

NOTE: The time scales in this model are not correct; it currently assumes that all measurements are equidistant. Feel free to experiment with adjusting this.

```
set.seed(27796)
dat[["id"]] <- NULL
res_step <- mx_growth_mixture(model = "
  i =~ 1*scl1 + 1*scl2 + 1*scl3 +1*scl4 +1*scl5 +1*scl6
  step =~ 0*scl1 + 1*scl2 + 1*scl3 +1*scl4 +1*scl5 +1*scl6
  s =~ 0*scl1 + 0*scl2 + 1*scl3 +2*scl4 +3*scl5 +4*scl6
  scl1 ~~ vscl1*scl1
  scl2 ~~ vscl2*scl2
  scl3 ~~ vscl3*scl3
  scl4 ~~ vscl4*scl4
  scl5 ~~ vscl5*scl5
  scl6 ~~ vscl6*scl6
  i ~~ 0*i
  step ~~ 0*step
  s ~~ 0*s
  i ~~ 0*s
  i ~~ 0*step
  s ~~ 0*step",
    classes = 1:5, data = dat)
# Additional iterations because of convergence problems for
# model 1:
res_step[[1]] <- mxTryHardWideSearch(res_step[[1]], extraTries = 50)
```

Note that the first model showed convergence problems, throwing the error: *The model does not satisfy the first-order optimality conditions to the required accuracy, and no improved point for the merit function could be found during the final linesearch.* To address this problem, we performed additional iterations to

find a better solution, using `OpenMx::mxTryHardWideSearch()`. This also illustrates that `tidySEM` mixture models inherit from `OpenMx`'s `MxModel`, and thus, different `OpenMx` functions can be used to act on models specified via `tidySEM`.

The fifth model also evidenced convergence problems, but this (as we will see) is because the solution is overfitted.

Class enumeration

To determine the correct number of classes, we considered the following criteria:

1. We do not consider classes with, on average, fewer than 5 participants per parameter in a class due to potential local underidentification
2. Lower values for information criteria (AIC, BIC, saBIC) indicate better fit
3. Significant Lo-Mendell-Rubin LRT test indicates better fit for k vs $k-1$ classes
4. We do not consider solutions with entropy $< .90$ because poor class separability compromises interpretability of the results
5. We do not consider solutions with minimum posterior classification probability $< .90$ because poor class separability compromises interpretability of the results

```
# Get fit table fit
tab_fit <- table_fit(res_step)
# Select columns
tab_fit <- tab_fit[, c("Name", "Classes", "LL", "Parameters",
    "BIC", "Entropy", "prob_min", "n_min", "warning", "lmr_p")]
tab_fit
```

According to the Table, increasing the number of classes keeps increasing model fit according to all ICs except the BIC, which increased after 3 classes.

Table 2

Fit of LCGA models

Name	Classes	LL	p	BIC	Ent.	p_min	n_min	warn	lmr_p
1	1.00	2,592.32	9	-5,122.67	1.00	1.00	1.00	NA	NA
2	2.00	3,797.78	13	-7,506.04	0.92	0.97	0.31	NA	0.00
3	3.00	4,264.43	17	-8,411.80	0.95	0.96	0.10	NA	0.00
4	4.00	4,264.45	21	-8,384.30	0.72	0.00	0.00	NA	1.00
5	5.00	4,264.45	25	-8,356.75	0.55	0.00	0.00	TRUE	1.00

The first two LMR tests are significant, indicating that a 2- and 3-class solution were a significant improvement over a 1- and 2-class solution, respectively. However, solutions with >3 classes had entropy and minimum posterior classification probability below the pre-specified thresholds. Models with >3 solutions also had fewer than five observations per parameter. This suggests that the preferred model should be selected from 1-3 classes.

Scree plot. A scree plot indicates that the largest decrease in ICs occurs from 1-2 classes, and the inflection point for all ICs is at 3 classes. Moreover, the BIC increased after 3 classes. A three-class solution thus appears to be the most parsimonious solution with good fit.

```
plot(tab_fit, statistics = c("AIC", "BIC", "saBIC"))
```

Based on the aforementioned criteria, we selected a 3-class model for further analyses. First, to prevent label switching, we re-order these classes by the value of the intercept `i`. Then, we report the estimated parameters.

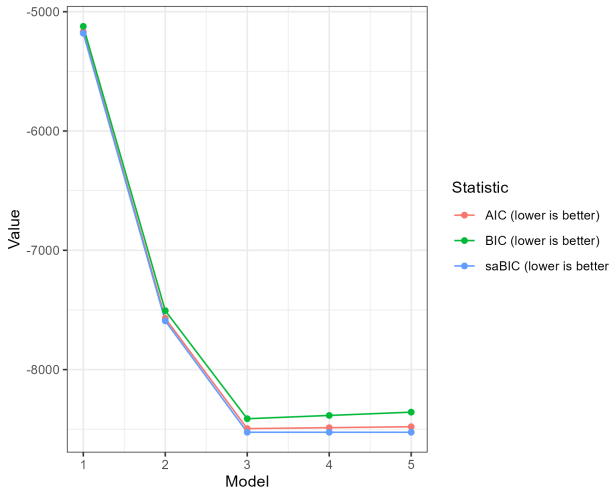


Figure 3

```
res_final <- mx_switch_labels(res_step[[3]], param = "M[1,7]",
    decreasing = FALSE)
tab_res <- table_results(res_final, columns = NULL)
# Select rows and columns
tab_res <- tab_res[tab_res$Category %in% c("Means", "Variances"),
    c("Category", "lhs", "est", "se", "pval", "confint", "name")]
tab_res
```

Table 3

Results from 3-class LCGA model

	Category	lhs	est	se	pval	confint	name
16	Means	i	0.35	0.00	0.00	[0.35, 0.36]	class1.M[1,7]
17	Means	step	-0.03	0.01	0.00	[-0.04, -0.02]	class1.M[1,8]
18	Means	s	0.01	0.00	0.00	[0.00, 0.01]	class1.M[1,9]
19	Variances	scl1	0.01	0.00	0.00	[0.01, 0.01]	class1.S[1,1]
20	Variances	scl2	0.01	0.00	0.00	[0.01, 0.01]	class1.S[2,2]

Table 3 continued

	Category	lhs	est	se	pval	confint	name
21	Variances	scl3	0.01	0.00	0.00	[0.01, 0.01]	class1.S[3,3]
22	Variances	scl4	0.01	0.00	0.00	[0.01, 0.01]	class1.S[4,4]
23	Variances	scl5	0.01	0.00	0.00	[0.01, 0.01]	class1.S[5,5]
24	Variances	scl6	0.01	0.00	0.00	[0.01, 0.02]	class1.S[6,6]
40	Means	i	0.46	0.01	0.00	[0.44, 0.47]	class2.M[1,7]
41	Means	step	0.03	0.01	0.00	[0.01, 0.04]	class2.M[1,8]
42	Means	s	0.01	0.00	0.00	[0.01, 0.02]	class2.M[1,9]
43	Variances	scl1	0.01	0.00	0.00	[0.01, 0.01]	class2.S[1,1]
44	Variances	scl2	0.01	0.00	0.00	[0.01, 0.01]	class2.S[2,2]
45	Variances	scl3	0.01	0.00	0.00	[0.01, 0.01]	class2.S[3,3]
46	Variances	scl4	0.01	0.00	0.00	[0.01, 0.01]	class2.S[4,4]
47	Variances	scl5	0.01	0.00	0.00	[0.01, 0.01]	class2.S[5,5]
48	Variances	scl6	0.01	0.00	0.00	[0.01, 0.02]	class2.S[6,6]
64	Means	i	0.63	0.01	0.00	[0.61, 0.65]	class3.M[1,7]
65	Means	step	0.07	0.01	0.00	[0.05, 0.10]	class3.M[1,8]
66	Means	s	0.01	0.00	0.13	[-0.00, 0.01]	class3.M[1,9]
67	Variances	scl1	0.01	0.00	0.00	[0.01, 0.01]	class3.S[1,1]
68	Variances	scl2	0.01	0.00	0.00	[0.01, 0.01]	class3.S[2,2]
69	Variances	scl3	0.01	0.00	0.00	[0.01, 0.01]	class3.S[3,3]
70	Variances	scl4	0.01	0.00	0.00	[0.01, 0.01]	class3.S[4,4]
71	Variances	scl5	0.01	0.00	0.00	[0.01, 0.01]	class3.S[5,5]
72	Variances	scl6	0.01	0.00	0.00	[0.01, 0.02]	class3.S[6,6]

As evident from these results, Class 1 started at a relatively lower level of depressive symptoms, experienced a decrease after deployment, followed by increase over time. Class 2 started at a moderate level of depressive symptoms, experienced an increase after deployment, followed by significant increase over time from T2-T6. Class 3 started at a relatively higher level, experienced an increase after deployment, followed by stability.

Wald tests

To test whether parameters are significantly different between classes, we can use Wald tests. Wald tests can be specified for all parameters in the model, using the hypothesis syntax from the `bain` package for informative hypothesis testing.

To identify the names of parameters in the model, we can use the `name` column of the results table above. Alternatively, to see all parameters in the model, run:

```
names(coef(res_final))
```

```
#>  [1] "mix3.weights[1,2]" "mix3.weights[1,3]"
#>  [3] "vscl1"             "vscl2"
#>  [5] "vscl3"             "vscl4"
#>  [7] "vscl5"             "vscl6"
#>  [9] "class1.M[1,7]"     "class1.M[1,8]"
#> [11] "class1.M[1,9]"     "class2.M[1,7]"
#> [13] "class2.M[1,8]"     "class2.M[1,9]"
#> [15] "class3.M[1,7]"     "class3.M[1,8]"
#> [17] "class3.M[1,9]"
```

Next, specify equality constrained hypotheses. For example, a hypothesis that states that the mean intercept is equal across groups is specified as follows:

Table 4

Wald tests

Hypothesis	df	chisq	p
Mean i	2	628.38	0.00
Mean step	2	65.32	0.00
Mean slope	2	13.03	0.00

```
"class1.M[1,7] = class2.M[1,7] & class1.M[1,7] = class3.M[1,7]
```

It is also possible to consider comparisons between two classes at a time. When conducting many significance tests, consider correcting for multiple comparisons however.

```
wald_tests <- wald_test(res_final, "
                    class1.M[1,7] = class2.M[1,7]&
                    class1.M[1,7] = class3.M[1,7];
                    class1.M[1,8] = class2.M[1,8]&
                    class1.M[1,8] = class3.M[1,8];
                    class1.M[1,9] = class2.M[1,9]&
                    class1.M[1,9] = class3.M[1,9]")
# Rename the hypothesis
wald_tests$Hypothesis <- c("Mean i", "Mean step", "Mean slope")
papaja::apa_table(wald_tests, digits = 2, caption = "Wald tests")
```

All Wald tests are significant, indicating that there are significant differences between the intercepts, step function, and slopes of the three classes.

Trajectory plot

Finally, we can plot the growth trajectories. This can help interpret the results better, as well as the residual heterogeneity around class trajectories.

```
p <- plot_growth(res_step[[3]], rawdata = TRUE, alpha_range = c(0,
    0.05))
# Add Y-axis breaks in original scale
brks <- seq(0, 1, length.out = 5)
labs <- round(invbc(scales::rescale(brks, from = c(0, 1), to = rng_bc),
    lambda))
p <- p + scale_y_continuous(breaks = seq(0, 1, length.out = 5),
    labels = labs) + ylab("SCL (rescaled from Box-Cox)")
p
```

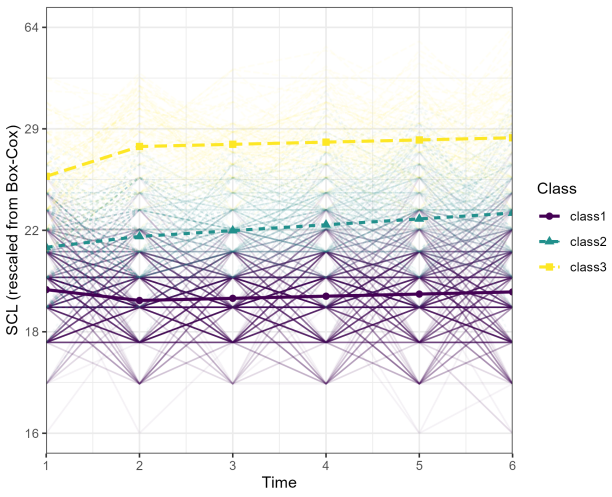


Figure 4

Note that the observed individual trajectories show very high variability within classes.

R session

```
sessionInfo()
#> R version 4.2.2 (2022-10-31 ucrt)
#> Platform: x86_64-w64-mingw32/x64 (64-bit)
#> Running under: Windows 10 x64 (build 19045)
#>
#> Matrix products: default
#>
#> locale:
#> [1] LC_COLLATE=English_United States.utf8
#> [2] LC_CTYPE=C
#> [3] LC_MONETARY=English_United States.utf8
#> [4] LC_NUMERIC=C
#> [5] LC_TIME=English_United States.utf8
#>
#> attached base packages:
#> [1] stats     graphics  grDevices utils     datasets
#> [6] methods   base
#>
#> other attached packages:
#>  [1] qpdf_1.3.0       MASS_7.3-58.1
#>  [3] scales_1.2.1     yaml_2.3.7
#>  [5] papaja_0.1.1     tinylabels_0.2.3
#>  [7] progressr_0.13.0 future_1.32.0
#>  [9] ggplot2_3.4.2    tidySEM_0.2.4.8
#> [11] OpenMx_2.21.8
```

```
#>
#> loaded via a namespace (and not attached):
#>   [1] backports_1.4.1       systemfonts_1.0.4
#>   [3] plyr_1.8.8            igraph_1.4.2
#>   [5] MplusAutomation_1.1.0 listenv_0.9.0
#>   [7] usethis_2.1.6         rstantools_2.3.1
#>   [9] inline_0.3.19         digest_0.6.31
#>  [11] htmltools_0.5.4       rsconnect_0.8.29
#>  [13] fansi_1.0.4           magrittr_2.0.3
#>  [15] checkmate_2.1.0       gert_1.9.2
#>  [17] credentials_1.3.2     globals_0.16.2
#>  [19] RcppParallel_5.1.7    matrixStats_0.63.0
#>  [21] rmdfiltr_0.1.3        sandwich_3.0-2
#>  [23] svglite_2.1.1         askpass_1.1
#>  [25] prettyunits_1.1.1     colorspace_2.1-0
#>  [27] textshaping_0.3.6     xfun_0.37
#>  [29] dplyr_1.1.1           callr_3.7.3
#>  [31] crayon_1.5.2          jsonlite_1.8.4
#>  [33] zoo_1.8-12            glue_1.6.2
#>  [35] prereg_0.6.0          gtable_0.3.3
#>  [37] V8_4.2.2              car_3.1-2
#>  [39] pkgbuild_1.4.0        rstan_2.26.16
#>  [41] future.apply_1.10.0   abind_1.4-5
#>  [43] bain_0.2.8            mvtnorm_1.1-3
#>  [45] rstatix_0.7.2         Rcpp_1.0.10
#>  [47] xtable_1.8-4          progress_1.2.2
#>  [49] tmvnsim_1.0-2         stats4_4.2.2
```

```
#>  [51] StanHeaders_2.26.16   worcs_0.1.10
#>  [53] httr_1.4.5            RColorBrewer_1.1-3
#>  [55] lavaan_0.6-15         ellipsis_0.3.2
#>  [57] pkgconfig_2.0.3       loo_2.6.0
#>  [59] farver_2.1.1          sass_0.4.5
#>  [61] utf8_1.2.3            tidyselect_1.2.0
#>  [63] labeling_0.4.2        rlang_1.1.0
#>  [65] munsell_0.5.0         tools_4.2.2
#>  [67] cachem_1.0.6          cli_3.6.0
#>  [69] dbscan_1.1-11         gsubfn_0.7
#>  [71] generics_0.1.3        ranger_0.14.1
#>  [73] broom_1.0.4           evaluate_0.20
#>  [75] fastmap_1.1.0         ragg_1.2.5
#>  [77] sys_3.4.1             rticles_0.24
#>  [79] blavaan_0.4-7         fs_1.6.1
#>  [81] processx_3.8.0        knitr_1.42
#>  [83] pander_0.6.5          purrr_1.0.1
#>  [85] RANN_2.6.1            gh_1.3.1
#>  [87] nlme_3.1-160          formatR_1.14
#>  [89] nonnest2_0.5-5        compiler_4.2.2
#>  [91] bayesplot_1.10.0      rstudioapi_0.14
#>  [93] curl_5.0.0            ggsignif_0.6.4
#>  [95] tibble_3.2.1          bslib_0.4.2
#>  [97] pbivnorm_0.6.0        highr_0.10
#>  [99] ps_1.7.2              lattice_0.20-45
#> [101] texreg_1.38.6         Matrix_1.5-1
#> [103] psych_2.3.3           vctrs_0.6.2
```

```
#> [105] CompQuadForm_1.4.3    pillar_1.9.0
#> [107] lifecycle_1.0.3       jquerylib_0.1.4
#> [109] data.table_1.14.8     R6_2.5.1
#> [111] bookdown_0.32         renv_0.16.0
#> [113] gridExtra_2.3         parallelly_1.35.0
#> [115] codetools_0.2-18      boot_1.3-28
#> [117] fastDummies_1.6.3     assertthat_0.2.1
#> [119] proto_1.0.0           openssl_2.0.6
#> [121] withr_2.5.0           mnormt_2.1.1
#> [123] parallel_4.2.2        hms_1.1.2
#> [125] quadprog_1.5-8        grid_4.2.2
#> [127] tidyr_1.3.0           coda_0.19-4
#> [129] rmarkdown_2.20        carData_3.0-5
#> [131] ggpubr_0.6.0          tinytex_0.44
```

Online Supplementary Materials E: SMART-LCA Checklist

Online Supplementary Materials E: SMART-LCA Checklist

This Appendix describes the SMART-LCA Checklist: Standards for More Accuracy in Reporting of different Types of Latent Class Analysis. The paper discusses the best practices on which the workflow is based; this vignette describes specific points to check when writing, reviewing, or reading a paper in greater detail. Note that, although the steps are numbered for reference purposes, we acknowledge the process of conducting and reporting research is not always linear.

1. Pre-Analysis
    - Define a research question and study design suitable for LCA.
    - Determine whether the application of LCA is confirmatory or exploratory.
    - Choose valid and reliable indicators with an appropriate data type for LCA.
    - Justify sample size.
    - Optionally: pre-register the analysis plan.
    - Optionally: use Preregistration-As-Code.
2. Examining Observed Data
    - Assess indicator distributions and scales.
    - Does the observed distribution match the intended measurement level? E.g., continuous variables have many unique values, categorical variables have a well-balaced response distribution.
    - Are all indicators on similar scales?
    - Are there distributional idiosyncrasies that need to be accounted for in model specification or by transforming the data (e.g., extreme skew)?
3. Data Preprocessing
    - Make sure continuous variables have type `numeric` or `integer` and an interval measurement level.

- Convert ordered and binary variables to `mxFactor`.
- Convert nominal variables to binary dummy variables using `mx_dummies()`.
- Rescale indicators that are on very different scales to prevent convergence issues, e.g. using the `scale()` function.
- Optionally, transform data to account for distributional idiosyncracies that could cause the model to fit poorly.

4. Missing data

    - Examine and report proportion of missingness per variable
    - Report MAR test using `mice::mcar()`.
    - Select and report appropriate method to handle missingness. In `tidySEM`, this is full information maximum likelihood (FIML), which is valid regardless of the outcome of the MAR test.

5. Model specification

    - For exploratory LCA, list all sensible alternative specifications
    - In confirmatory LCA, justify alternative model specifications on theoretical grounds
    - Clearly report the model specification, so it is clear what all parameters are.
    - Optionally, verify that the model can capture distributional idiosyncracies (see Data Preprocessing).

6. Number of classes

    - Justify the maximum number of classes k.
    - In exploratory LCA, estimate 1:k classes.
    - In confirmatory LCA, estimate the theoretical number of classes and choose other models to benchmark it against. This should include a 1-class model, optionally competing theoretical models, and optionally models with a different numbers of classes.

7. Justify the criteria used for class enumeration, which can include:
    - Information criteria.
    - LR tests.
    - Theoretically expected class solution.
8. Justify the criteria used to eliminate models from consideration, including:
    - Model convergence.
    - Classification diagnostics.
    - Local identifiability (acceptable number of observations per parameter in each class).
    - Theoretical interpretability of the results.
9. Transparency and Reproducibility
    - Use free open-source software to enable reproducibility.
    - Comprehensively cite literature, software, and data sources.
    - Share analysis code.
    - If possible, share data. Otherwise, share a synthetic dataset created via `worcs::synthetic()` or `mxGenerateData()`.
    - Share all digital research output in line with the FAIR principles.
    - Optionally, use the Workflow for Open Reproducible Code in Science and the `worcs` R-package automate creating a reproducible research archive.
10. Estimation and Convergence
    - Report the method of estimation. In `tidySEM`, this is simulated annealing with informative start values, determined via K-means clustering for complete data (Hartigan & Wong, 1979) and hierarchical clustering when there are missing values.
    - Assess convergence of all models before interpreting them.

- Optionally, if there is non-convergence, use functions like `mxTryHard()` to aid convergence.

11. Reporting Results

- Report the fit of all models under consideration in a table.
- Report classification diagnostics for all models under consideration.
- Report the minimal class proportions for all models under consideration.
- Report class proportions for the final model, obtained via class_prob(res, "sum.posterior").
- Report point estimates, standard errors and confidence intervals for model parameters, obtained via `table_results()`.
- Optionally, convert thresholds to conditional item probabilities using `table_prob()`.
- Assign informative class names while clarifying that these names are just shorthand.

12. Inference

- Standard p-values in `table_results()` test the hypothesis that parameters are equal to zero. Consider whether these tests are meaningful and relevant.
- Control for, or reflect on, the study-wide Type I error level.
- Optionally, use the standard errors to test other null hypotheses.
- Optionally, use `wald_test()` to test informative hypotheses.
- Optionally, to test a confirmatory LCA model, compare parameter estimates to hypothesized values.

13. Visualization

- Visualize raw data to assess class separability and model fit.
- Show point estimates of model parameters.
- If possible, show confidence bounds.

- Optionally, show multivariate distributions for multivariate models.

14. Follow-up analyses.

    - Account for classification inaccuracy in follow-up analyses, using `BCH()` or an equivalent method.

Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *28*(1), 100–108. Retrieved from http://www.jstor.org/stable/2346830