# Appendix B: Exploratory LPA for Ocean Microplastics

Appendix B: 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.

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

### Examining the Data

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

Table 1

*Descriptive statistics*

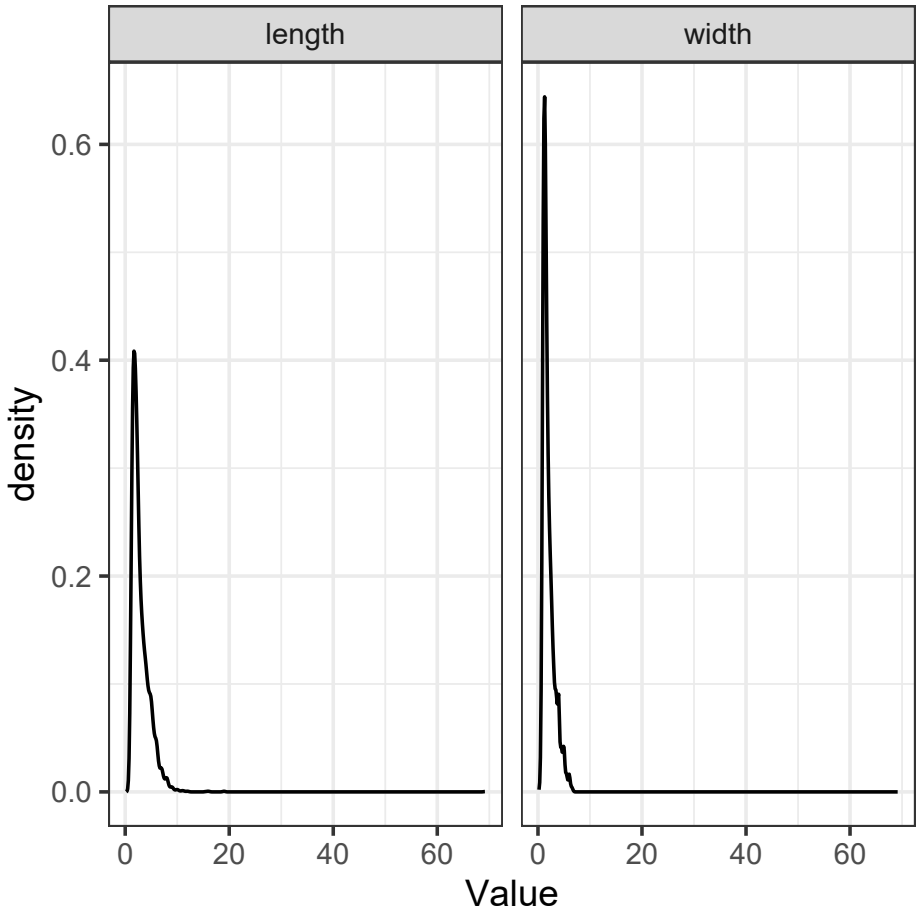
| name | type | n | missing | unique | mean | median | sd | min | max | skew_2se | kurt_2se |
|---|---|---|---|---|---|---|---|---|---|---|---|
| length | numeric | 5605 | 0.00 | 2086 | 2.94 | 2.37 | 1.89 | 1.00 | 69.16 | 137.38 | 2,116.43 |
| width | numeric | 5605 | 0.00 | 2079 | 2.00 | 1.62 | 1.11 | 0.20 | 6.76 | 22.23 | 37.08 |

categorical, we remove columns for continuous data to de-clutter the table:

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

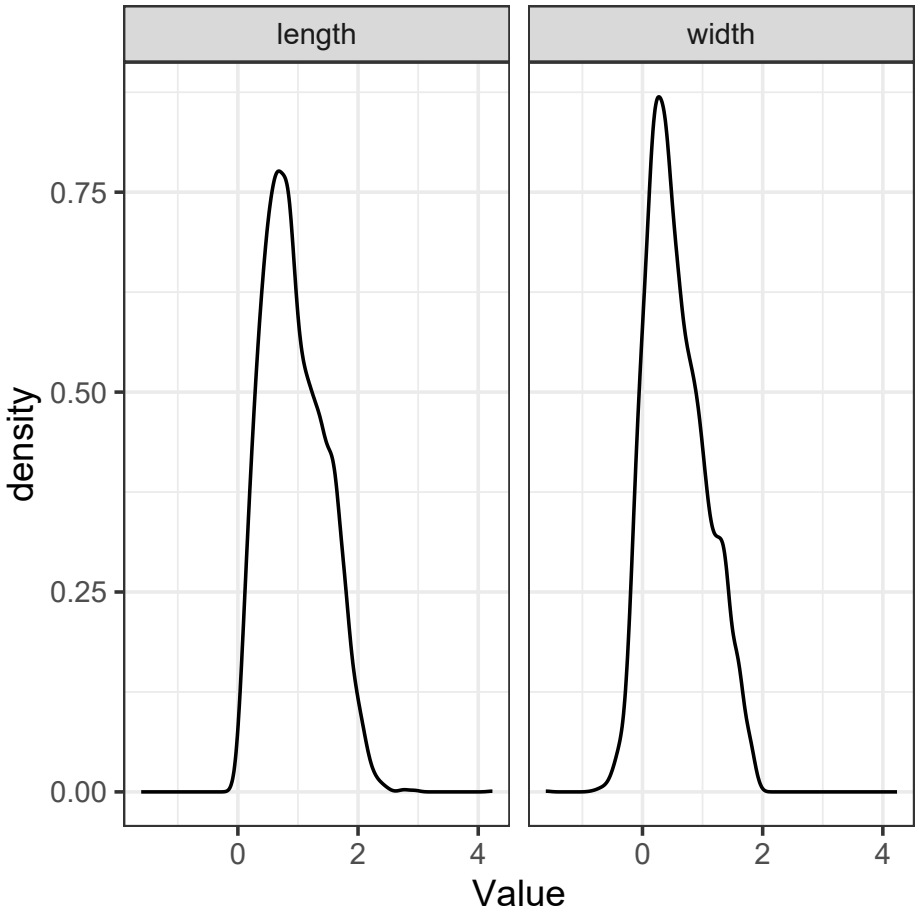
Additionally, we can plot the data. The `ggplot2` function `geom_density()` is useful to visualize continuous data:

```
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()
```



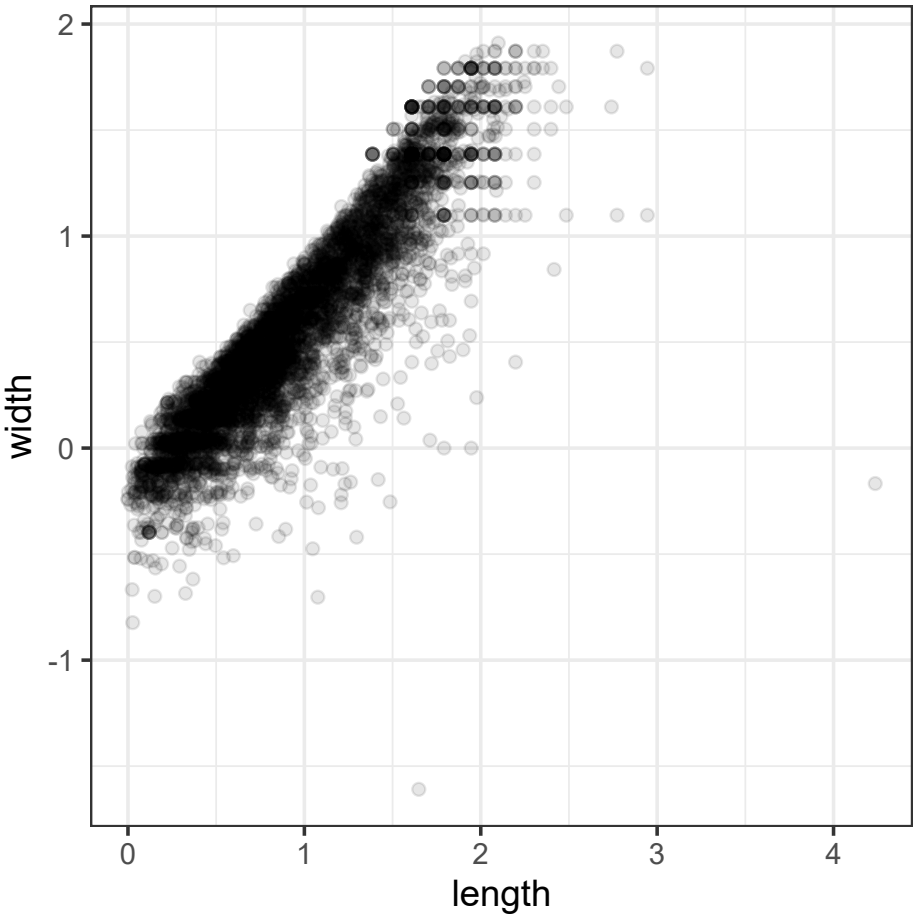

The data are correctly coded as `numeric`. There are no missing values; if any variables had missing values, we would report an MCAR test with `mice::mcar()`, and explain that missing data are accounted for using FIML. Both the table above and the density plot indicate that the data are extremely right-skewed 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 = .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.

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; depending on the results, we can always choose to increase it later. 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()`.

```
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 local identifiability. The sample size is `5605`. We can calculate the ratio of observations to parameters and append it to the fit table:

```
fit$par_ratio <- (5605*fit$n_min) / (fit$Parameters/fit$Classes)
fit[ , c("Name", "LL", "Parameters", "par_ratio",
         "BIC", "Entropy",
         "prob_min", "prob_max",
         "n_min", "n_max",
         "lmr_p")]
```

As can be seen from the fit table, the lowest ratio of observations to parameters is 18, which is no cause for concern.

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 set a lower bound for class size and only consider solutions that capture at least 10% of the sample.

An 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 < .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.

Table 2

*Model fit table. Some columns removed to fit page.*

| Name | LL | p | par_ratio | BIC | Ent. | p_min | n_min | lmr_p |
|---|---|---|---|---|---|---|---|---|
| equal var 1 | -8,107.31 | 4 | 1,401.25 | 16,249.14 | 1.00 | 1.00 | 1.00 | NA |
| equal var 2 | -5,211.25 | 7 | 564.00 | 10,482.91 | 0.87 | 0.94 | 0.35 | 0.00 |
| equal var 3 | -4,138.27 | 10 | 341.70 | 8,362.85 | 0.83 | 0.88 | 0.20 | 0.00 |
| equal var 4 | -3,500.42 | 13 | 246.15 | 7,113.04 | 0.83 | 0.89 | 0.14 | 0.00 |
| free var 1 | -8,107.31 | 4 | 1,401.25 | 16,249.14 | 1.00 | 1.00 | 1.00 | NA |
| free var 2 | -5,137.90 | 9 | 501.33 | 10,353.49 | 0.85 | 0.94 | 0.40 | 0.00 |
| free var 3 | -4,005.10 | 14 | 374.36 | 8,131.04 | 0.83 | 0.89 | 0.31 | 0.00 |
| free var 4 | -3,330.87 | 19 | 245.05 | 6,825.74 | 0.84 | 0.88 | 0.21 | 0.00 |
| equal var, equal cov 1 | -3,388.56 | 5 | 1,121.00 | 6,820.28 | 1.00 | 1.00 | 1.00 | NA |
| equal var, equal cov 2 | -3,082.31 | 8 | 432.25 | 6,233.66 | 0.72 | 0.86 | 0.31 | 0.00 |
| equal var, equal cov 3 | -3,030.10 | 11 | 256.36 | 6,155.14 | 0.67 | 0.71 | 0.17 | 0.00 |
| equal var, equal cov 4 | -3,020.67 | 14 | 220.29 | 6,162.19 | 0.61 | 0.68 | 0.14 | 0.00 |
| free var, equal cov 1 | -3,388.56 | 5 | 1,121.00 | 6,820.28 | 1.00 | 1.00 | 1.00 | NA |
| free var, equal cov 2 | -2,544.74 | 10 | 97.00 | 5,175.79 | 0.64 | 0.51 | 0.09 | 0.00 |
| free var, equal cov 3 | -2,256.95 | 15 | 71.80 | 4,643.36 | 0.68 | 0.54 | 0.06 | 0.00 |
| free var, equal cov 4 | -2,068.70 | 20 | 26.00 | 4,310.02 | 0.63 | 0.52 | 0.02 | 0.00 |
| equal var, free cov 1 | -3,388.56 | 5 | 1,121.00 | 6,820.28 | 1.00 | 1.00 | 1.00 | NA |
| equal var, free cov 2 | -2,551.86 | 9 | 107.78 | 5,181.40 | 0.65 | 0.52 | 0.09 | 0.00 |
| equal var, free cov 3 | -2,359.39 | 13 | 84.23 | 4,830.99 | 0.68 | 0.56 | 0.07 | 0.00 |
| equal var, free cov 4 | -2,173.85 | 17 | 27.53 | 4,494.44 | 0.63 | 0.60 | 0.02 | 0.00 |
| free var, free cov 1 | -3,388.56 | 5 | 1,121.00 | 6,820.28 | 1.00 | 1.00 | 1.00 | NA |
| free var, free cov 2 | -2,574.87 | 11 | 406.91 | 5,244.69 | 0.56 | 0.81 | 0.40 | 0.00 |
| free var, free cov 3 | -2,111.38 | 17 | 36.00 | 4,369.50 | 0.65 | 0.51 | 0.04 | 0.00 |
| free var, free cov 4 | -2,024.10 | 23 | 15.48 | 4,246.73 | 0.62 | 0.50 | 0.02 | 0.00 |

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

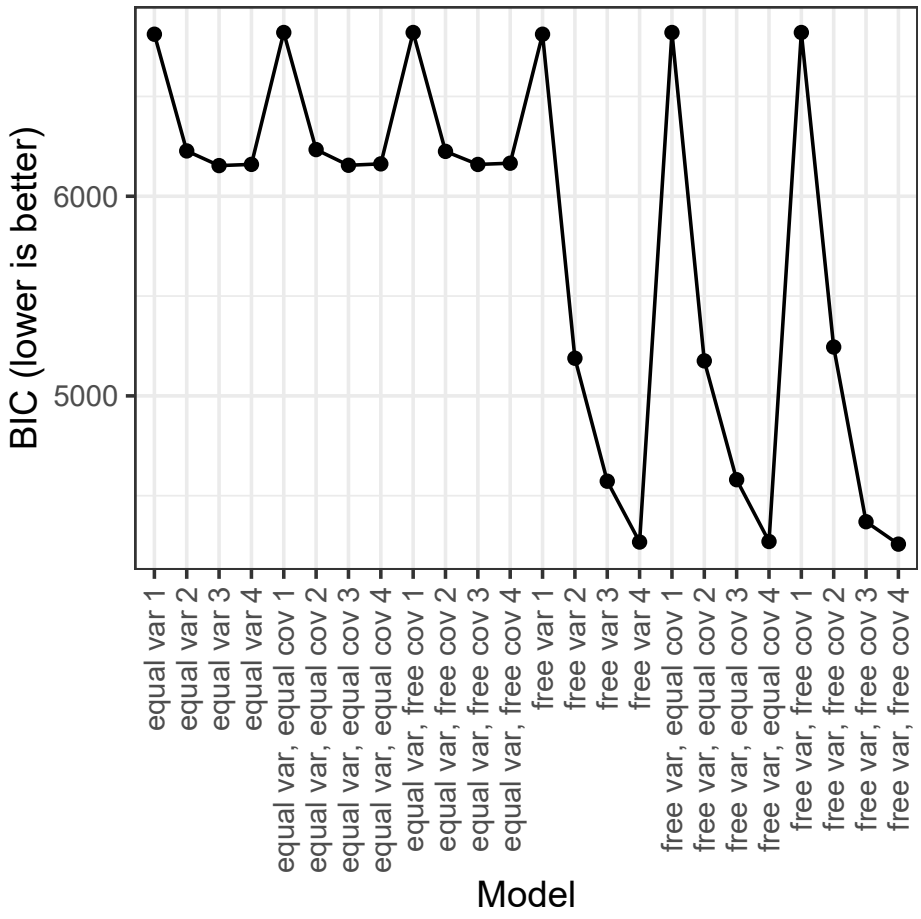



*Figure 1*. Bivariate profile plot

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. Similarly, all LMR tests are significant, indicating a preference for more complex models.

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.

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 |

### Interpreting the Final Class Solution

and extract the classification probabilities and model results. 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 there is little sense in requesting the standard errors and p-values: With a sample size of 5606, every parameter is significantly different from zero.

```
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)
```

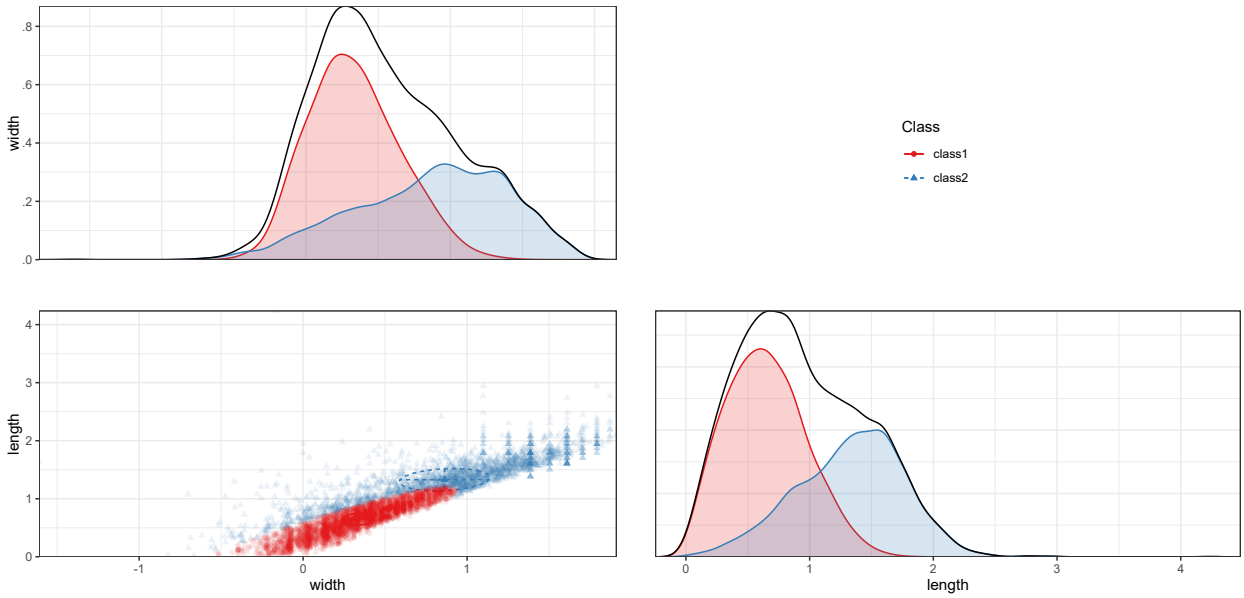



*Figure 2*. Bivariate profile plot

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 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 can simply 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:

```
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 4.1.3 (2022-03-10)
#> Platform: x86_64-w64-mingw32/x64 (64-bit)
#> Running under: Windows 10 x64 (build 19045)
#>
#> Matrix products: default
#>
#> locale:
#> [1] LC_COLLATE=English_United Kingdom.1252
#> [2] LC_CTYPE=English_United Kingdom.1252
#> [3] LC_MONETARY=English_United Kingdom.1252
#> [4] LC_NUMERIC=C
#> [5] LC_TIME=English_United Kingdom.1252
#>
#> attached base packages:
#> [1] stats     graphics  grDevices utils     datasets  methods   base
#>
#> other attached packages:
```

```
#> [1] ggplot2_3.4.0    tidySEM_0.2.4.6  OpenMx_2.21.1    scales_1.2.1
#> [5] yaml_2.3.6       papaja_0.1.1     tinylabels_0.2.3
#>
#> loaded via a namespace (and not attached):
#>   [1] TH.data_1.1-1         colorspace_2.1-0      estimability_1.4.1
#>   [4] parameters_0.20.1     rstudioapi_0.14       listenv_0.9.0
#>   [7] lavaan_0.6-13         rstan_2.26.13         fansi_1.0.4
#>  [10] mvtnorm_1.1-3         codetools_0.2-18      splines_4.1.3
#>  [13] mnormt_2.1.1          knitr_1.41            texreg_1.38.6
#>  [16] bayesplot_1.10.0      jsonlite_1.8.4        effectsize_0.8.2
#>  [19] compiler_4.1.3        httr_1.4.4            emmeans_1.8.4-1
#>  [22] backports_1.4.1       assertthat_0.2.1      Matrix_1.5-3
#>  [25] fastmap_1.1.0         cli_3.6.0             htmltools_0.5.4
#>  [28] prettyunits_1.1.1     tools_4.1.3           igraph_1.3.5
#>  [31] coda_0.19-4           gtable_0.3.1          glue_1.6.2
#>  [34] RANN_2.6.1            dplyr_1.0.10          V8_4.2.2
#>  [37] Rcpp_1.0.10           carData_3.0-5         vctrs_0.5.2
#>  [40] nlme_3.1-161          psych_2.2.9           insight_0.18.8
#>  [43] xfun_0.36             stringr_1.5.0         globals_0.16.2
#>  [46] ps_1.7.2              proto_1.0.0           CompQuadForm_1.4.3
#>  [49] lifecycle_1.0.3       future_1.30.0         MASS_7.3-58.2
#>  [52] zoo_1.8-11            parallel_4.1.3        sandwich_3.0-2
#>  [55] inline_0.3.19         curl_5.0.0            gridExtra_2.3
#>  [58] pander_0.6.5          blavaan_0.4-3         loo_2.5.1
#>  [61] StanHeaders_2.26.13   stringi_1.7.12        highr_0.10
#>  [64] bayestestR_0.13.0     fastDummies_1.6.3     checkmate_2.1.0
#>  [67] boot_1.3-28           pkgbuild_1.4.0        rlang_1.0.6
```

```
#>  [70] pkgconfig_2.0.3       matrixStats_0.63.0    evaluate_0.20
#>  [73] lattice_0.20-45       rstantools_2.2.0      processx_3.8.0
#>  [76] tidyselect_1.2.0      parallelly_1.34.0     plyr_1.8.8
#>  [79] magrittr_2.0.3        bookdown_0.32         R6_2.5.1
#>  [82] generics_0.1.3        multcomp_1.4-20       DBI_1.1.3
#>  [85] withr_2.5.0           gsubfn_0.7            pillar_1.8.1
#>  [88] survival_3.5-0        datawizard_0.6.5      abind_1.4-5
#>  [91] tibble_3.1.8          future.apply_1.10.0   crayon_1.5.2
#>  [94] car_3.1-1             nonnest2_0.5-5        utf8_1.2.2
#>  [97] tmvnsim_1.0-2         MplusAutomation_1.1.0 rmarkdown_2.20
#> [100] grid_4.1.3            data.table_1.14.6     pbivnorm_0.6.0
#> [103] bain_0.2.8            callr_3.7.3           digest_0.6.31
#> [106] xtable_1.8-4          dbscan_1.1-11         RcppParallel_5.1.6
#> [109] stats4_4.1.3          munsell_0.5.0
```