Appendix A: Confirmatory LPA for the Caregiver Compass

Appendix A: 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.

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:

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

`c("burdened", "trapped", "negaffect", "loneliness").` ‘.

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

Examining the Data

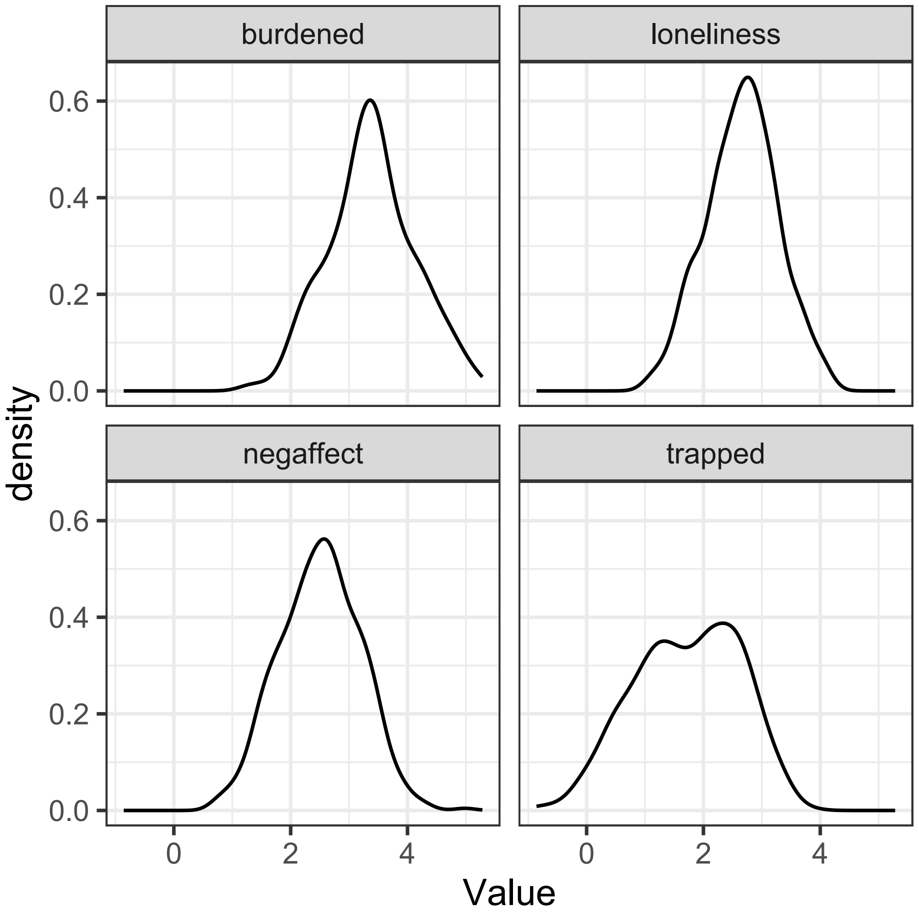
As per the best practices, the first step in LCA is examining the observed data. We use `tidySEM::descriptives()` to describe the data numerically. Because all scales are continuous, we select only columns for continuous data to de-clutter the table:

```r
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.

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.

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

This analysis should produce some warnings 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.

Class enumeration unambiguously indicates a preference for the 4-class solution: the

Table 2

Model fit table

Name	LL	p	n	BIC	Entropy	p_min	p_max	n_min	n_max	lmr_p
equal var 1	-2,241.98	8	513	4,533.89	1.00	1.00	1.00	1.00	1.00	NA
equal var 2	-2,031.37	13	513	4,143.85	0.74	0.91	0.93	0.42	0.58	0.00
equal var 3	-1,951.35	18	513	4,015.02	0.78	0.89	0.91	0.19	0.54	0.00
equal var 4	-1,916.25	23	513	3,976.03	0.75	0.81	0.92	0.16	0.34	0.00
equal var 5	-1,912.93	28	513	4,000.59	0.70	0.39	0.92	0.07	0.31	0.28

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. The p-values of the LMR LR test are significant for all pairwise model comparisons, except for the 5-class model. 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%.

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 3

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

```r
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.

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

The 4-class model yielded classes of reasonable size; 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 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)
```

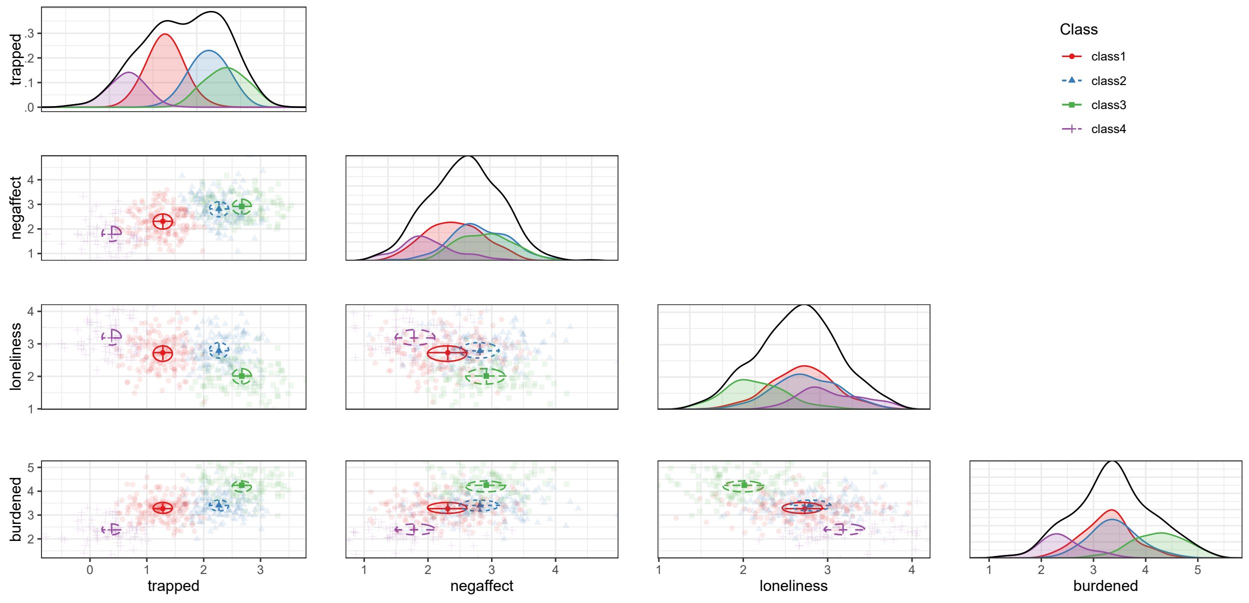


Figure 1. Bivariate profile plot

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. 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 density: class 2 and 3 coalesce on negative affect, 1 and 2 coalesce on loneliness, and 1 and 2 coalesce

Table 4

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

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%).

```
plot_profiles(res_final)
```

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. 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:

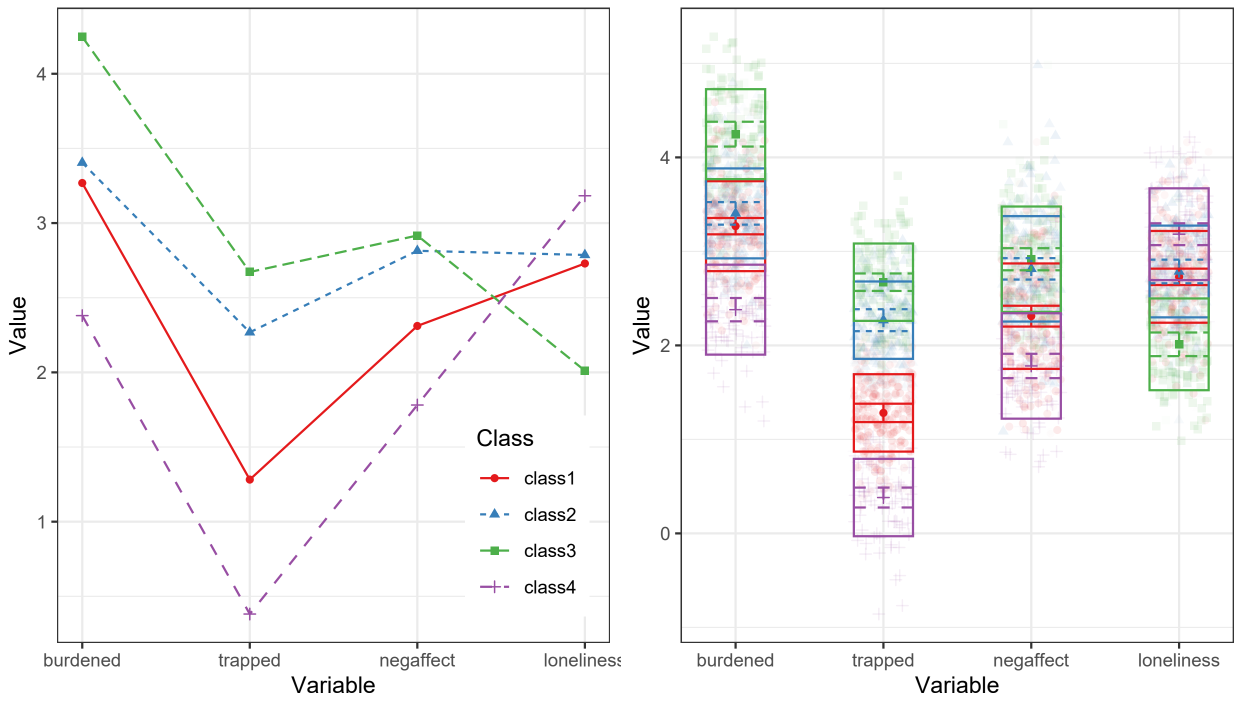


Figure 2. Bivariate profile plot

```
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. Finally, we 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)`.

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
```