Appendix C: Latent Class Growth Analysis

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

Appendix C: 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()
```

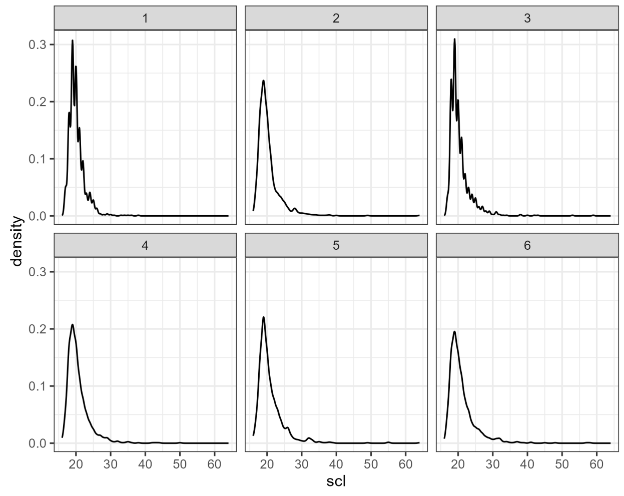

Figure 1

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.

```
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
```

```r
df_scores$sqrt <- scales::rescale(sqrt(df_scores$scl), to = c(0, 1))
# Cube root transform
df_scores$qrt <- scales::rescale(df_scores$scl^.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:

```r
# 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()
```

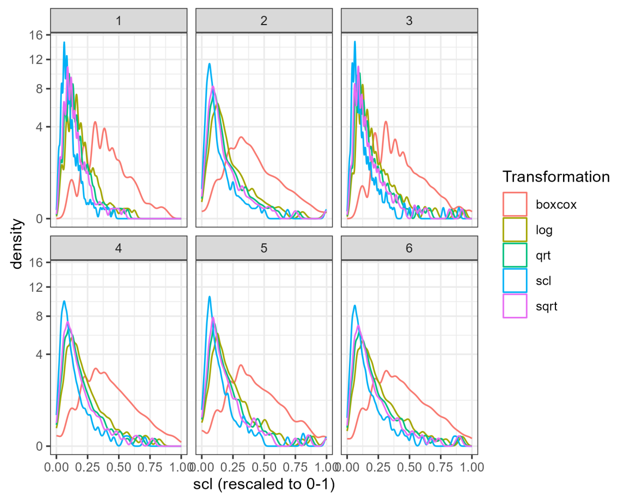

Figure 2

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 = "i
names(dat) <- gsub("boxcox.", "scl", names(dat))
```

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 not 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.

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

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

```r
# Get fit table fit
tab_fit <- table_fit(res_step)
# Select columns
tab_fit <- tab_fit[, c("Name", "Classes", "LL", "Parameters", "BIC", "Entropy", "prob_mi
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.

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"))
```

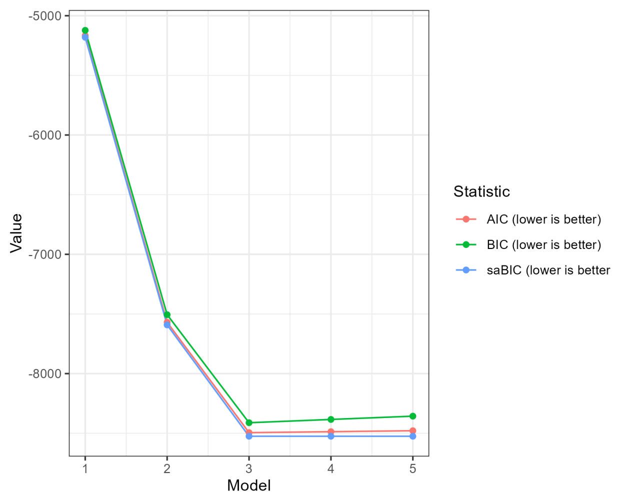

Figure 3

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.

```r
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", "
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]
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]

Table 3 continued

	Category	lhs	est	se	pval	confint	name
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]" "vscl1"
#>  [4] "vscl2"             "vscl3"             "vscl4"
#>  [7] "vscl5"             "vscl6"             "class1.M[1,7]"
#> [10] "class1.M[1,8]"     "class1.M[1,9]"     "class2.M[1,7]"
#> [13] "class2.M[1,8]"     "class2.M[1,9]"     "class3.M[1,7]"
#> [16] "class3.M[1,8]"     "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:

`"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

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

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, .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("S
p
```

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

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)
#>
```

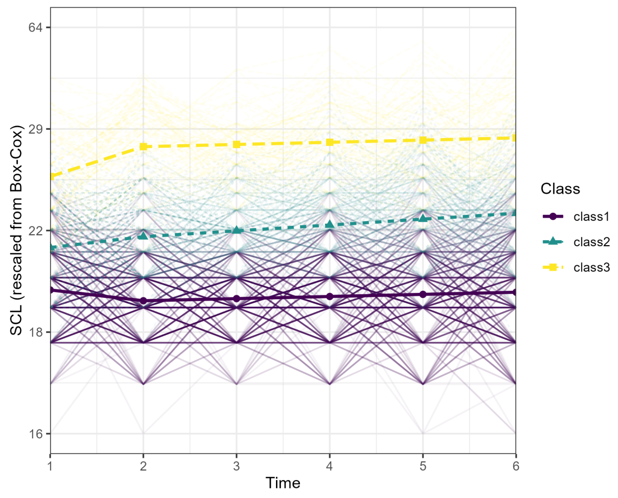

Figure 4

```
#> 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] MASS_7.3-58.2    ggplot2_3.4.0    tidySEM_0.2.4.6  OpenMx_2.21.1
#> [5] scales_1.2.1     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        zoo_1.8-11
#> [52] parallel_4.1.3       sandwich_3.0-2       inline_0.3.19
#> [55] curl_5.0.0           gridExtra_2.3        pander_0.6.5
#> [58] blavaan_0.4-3        loo_2.5.1            StanHeaders_2.26.13
#> [61] stringi_1.7.12       bayestestR_0.13.0    fastDummies_1.6.3
#> [64] checkmate_2.1.0      boot_1.3-28          pkgbuild_1.4.0
#> [67] rlang_1.0.6          pkgconfig_2.0.3      matrixStats_0.63.0
#> [70] evaluate_0.20        lattice_0.20-45      rstantools_2.2.0
#> [73] processx_3.8.0       tidyselect_1.2.0     parallelly_1.34.0
#> [76] plyr_1.8.8           magrittr_2.0.3       bookdown_0.32
#> [79] R6_2.5.1             generics_0.1.3       multcomp_1.4-20
#>  [82] DBI_1.1.3            withr_2.5.0           gsubfn_0.7
#>  [85] pillar_1.8.1         survival_3.5-0        datawizard_0.6.5
#>  [88] abind_1.4-5          tibble_3.1.8          future.apply_1.10.0
#>  [91] crayon_1.5.2         car_3.1-1             nonnest2_0.5-5
#>  [94] utf8_1.2.2           tmvnsim_1.0-2         MplusAutomation_1.1.0
#>  [97] rmarkdown_2.20       grid_4.1.3            data.table_1.14.6
#> [100] pbivnorm_0.6.0       bain_0.2.8            callr_3.7.3
#> [103] digest_0.6.31        xtable_1.8-4          dbscan_1.1-11
#> [106] RcppParallel_5.1.6   stats4_4.1.3          munsell_0.5.0
```