

```

1 # SPM-2: derived normalized T-scores for each case
  in standardization sample;
2 # create raw-to-T lookup tables.
3
4 suppressMessages(library(here)) # BEST WAY TO SPEC
  IFY FILE PATHS
5 suppressMessages(suppressWarnings(library(tidyvers
  e)))
6 library(bestNormalize) # NORMALIZATION METHODS
7 suppressMessages(library(psych)) # DESCRIPTIVE TAB
  LES
8 suppressMessages(library(data.table))
9
10 # READ FINALIZED STAND SAMPLE -----
    -----
11
12 Child_512_Home <-
13   suppressMessages(as_tibble(read_csv(
14     here("INPUT-FILES/CHILD/ALLDATA-DESAMP-NORMS-I
  NPUT/Child-512-Home-allData-desamp.csv")
15   )))
16
17 score_names <- c("TOT", "SOC", "VIS", "HEA", "TO
  U", "TS", "BOD", "BAL", "PLA")
18
19 # Check for any dupIDs (anyDuplicated() returns ro
  w number of FIRST dup ID encountered)
20 anyDuplicated(Child_512_Home$IDNumber)
21
22 # Check for any NAs on IDNumber, returns TRUE if N
  A exist
23 any(is.na(Child_512_Home$IDNumber))
24
25 # extract cases with Dup ID numbers or NA on IDNum
  ber, write out for investigation
26 Child_512_Home_dupMissIDs <- Child_512_Home %>%
27   mutate(dup = duplicated(IDNumber)) %>%
28   filter(dup == TRUE | is.na(IDNumber)) %>%
29   select(-dup) %>%
30   write_csv(
31     here(
32       paste0(
33         'OUTPUT-FILES/CHILD/DUP-IDS/Child-512-Home
  -dupIDs-missingIDs-',
34         format(Sys.Date(), "%Y-%m-%d"),
35         '.csv'
36       )
37     ),
38     na = 'missing'
39   )
40
41 # DETERMINE BEST NORMALIZATION MODEL -----
    -----
42
43 # (NOTE: THIS SECTION SHOULD BE TOGGLED OFF AFTER
  SELECTION OF NORMALIZATION
44 # MODEL)
45
46 # # # create a bestNormalize object to lock down t
  he normalizing function that will be used on repea
  ted runs of the norms.
47 # TOT_nz_obj <- bestNormalize(Child_512_Home$TOT_r

```

```

1 # SPM-2: derived normalized T-scores for each case
  in standardization sample;
2 # create raw-to-T lookup tables.
3
4 suppressMessages(library(here)) # BEST WAY TO SPEC
  IFY FILE PATHS
5 suppressMessages(suppressWarnings(library(tidyvers
  e)))
6 library(bestNormalize) # NORMALIZATION METHODS
7 suppressMessages(library(psych)) # DESCRIPTIVE TAB
  LES
8 suppressMessages(library(data.table))
9
10 # READ FINALIZED STAND SAMPLE -----
    -----
11
12 Child_512_Home <-
13   suppressMessages(as_tibble(read_csv(
14     here("INPUT-FILES/CHILD/ALLDATA-DESAMP-NORMS-I
  NPUT/Child-512-Home-allData-desamp.csv")
15   )))
16
17 score_names <- c("TOT", "SOC", "VIS", "HEA", "TO
  U", "TS", "BOD", "BAL", "PLA")
18
19 # Check for any dupIDs (anyDuplicated() returns ro
  w number of FIRST dup ID encountered)
20 anyDuplicated(Child_512_Home$IDNumber)
21
22 # Check for any NAs on IDNumber, returns TRUE if N
  A exist
23 any(is.na(Child_512_Home$IDNumber))
24
25 # extract cases with Dup ID numbers or NA on IDNum
  ber, write out for investigation
26 Child_512_Home_dupMissIDs <- Child_512_Home %>%
27   mutate(dup = duplicated(IDNumber)) %>%
28   filter(dup == TRUE | is.na(IDNumber)) %>%
29   select(-dup) %>%
30   write_csv(
31     here(
32       paste0(
33         'OUTPUT-FILES/CHILD/DUP-IDS/Child-512-Home
  -dupIDs-missingIDs-',
34         format(Sys.Date(), "%Y-%m-%d"),
35         '.csv'
36       )
37     ),
38     na = 'missing'
39   )
40
41 # DETERMINE BEST NORMALIZATION MODEL -----
    -----
42
43 # (NOTE: THIS SECTION SHOULD BE TOGGLED OFF AFTER
  SELECTION OF NORMALIZATION
44 # MODEL)
45
46 # # # create a bestNormalize object to lock down t
  he normalizing function that will be used on repea
  ted runs of the norms.
47 # TOT_nz_obj <- bestNormalize(Child_512_Home$TOT_r

```

```

aw)
48 #
49 # # print transformation
50 # TOT_nz_obj$chosen_transform
51 #
52 # # Extract transformation type
53 # chosen_transform <- class(TOT_nz_obj$chosen_trans
  form)[1]
54 #
55 # # apply the chosen method to create normalized z
  -scores for each case.
56 # TOT_nz_transform <- eval(as.name(chosen_transfor
  m))(Child_512_Home$TOT_raw)
57
58
59 # APPLY SELECTED NORMALIZATION MODEL TO CREATE NOR
  MALIZED Z-SCORES -----
60
61 # Apply a static, repeatable transformation to cre
  ate normalized z-scores for
62 # each case.
63
64 # create char vec with names for the nine score tr
  ansformations
65 nz_transform_names <- c(paste0(score_names, '_nz_t
  ransform'))
66
67 # pull nine raw score columns into a list
68 raw_score_cols_list <- map(score_names, ~ Child_51
  2_Home %>%
69     pull(
70         !!as.name(paste0(.x, '_raw'))
71     )
72 )
73
74 # create the nine named objects that contain the n
  ormalization for each score
75 # distribution. In this call of `purrr::walk2()`,
  the .f calls assign(), because
76 # the central purpose of this code is to use assig
  n to create a series of named
77 # objects in the global environment. The `walk` fu
  nctions are used when the
78 # output of interest is a side effect. The names f
  or these objects are contained
79 # in the .x argument (a char vec). The data to be
  normalized is in the list of
80 # nine raw score columns `raw_score_cols_list`, wh
  ich as assigned to the .y
81 # argument of walk2(), using the dot . shorthand.
  Within assign(), the value
82 # argument allows the selected normalization trans
  formation to be applied to the
83 # .y data.
84
85 # NOTE: MUST SUBSITUTE NAMED TRANSFORMATION FROM P
  REVIOUS STEP IN THIS LINE:
86 # value = orderNorm(.y), e.g., value = [SELECTED T
  RANSFORMATION](.y),
87
88 raw_score_cols_list %>%
89   walk2(
90     .x = c(nz_transform_names),      # names to
  assign
91     .y = .,                          # object to be assigned
92     .f = ~ assign(x = .x,
93                   value = orderNorm(.y),

```

```

aw)
48 #
49 # # print transformation
50 # TOT_nz_obj$chosen_transform
51 #
52 # # Extract transformation type
53 # chosen_transform <- class(TOT_nz_obj$chosen_tran
  sform)[1]
54 #
55 # # apply the chosen method to create normalized z
  -scores for each case.
56 # TOT_nz_transform <- eval(as.name(chosen_transfor
  m))(Child_512_Home$TOT_raw)
57
58
59 # APPLY SELECTED NORMALIZATION MODEL TO CREATE NOR
  MALIZED Z-SCORES -----
60
61 # Apply a static, repeatable transformation to cre
  ate normalized z-scores for
62 # each case.
63
64 # create char vec with names for the nine score tr
  ansformations
65 nz_transform_names <- c(paste0(score_names, '_nz_t
  ransform'))
66
67 # pull nine raw score columns into a list
68 raw_score_cols_list <- map(score_names, ~ Child_51
  2_Home %>%
69     pull(
70         !!as.name(paste0(.x, '_raw'))
71     )
72 )
73
74 # create the nine named objects that contain the n
  ormalization for each score
75 # distribution. In this call of `purrr::walk2()`,
  the .f calls assign(), because
76 # the central purpose of this code is to use assig
  n to create a series of named
77 # objects in the global environment. The `walk` fu
  nctions are used when the
78 # output of interest is a side effect. The names f
  or these objects are contained
79 # in the .x argument (a char vec). The data to be
  normalized is in the list of
80 # nine raw score columns `raw_score_cols_list`, wh
  ich as assigned to the .y
81 # argument of walk2(), using the dot . shorthand.
  Within assign(), the value
82 # argument allows the selected normalization trans
  formation to be applied to the
83 # .y data.
84
85 # NOTE: MUST SUBSITUTE NAMED TRANSFORMATION FROM P
  REVIOUS STEP IN THIS LINE:
86 # value = orderNorm(.y), e.g., value = [SELECTED T
  RANSFORMATION](.y),
87
88 raw_score_cols_list %>%
89   walk2(
90     .x = c(nz_transform_names),      # names to
  assign
91     .y = .,                          # object to be assigned
92     .f = ~ assign(x = .x,
93                   value = orderNorm(.y),

```

```

94         envir = .GlobalEnv)
95     )
96
97 # Each of the named objects (normalization for each score) created in the
98 # previous snippet is a list. Use base::mget to put these named objects into a
99 # 'list of lists'. Here, mget takes a single argument, a char vec holding the
100 # names of the lists that are to be put into the new list `nz_transform_list`
101
102 nz_transform_list <- mget(nz_transform_names)
103
104 # Create a char vec containing the names of the output objects for the next
105 # step. These output objects are single-column named dfs containing the
106 # normalized z scores corresponding to the raw score for each case. The objects and the single
107 # columns within them have the same names
108 nz_names <- c(paste0(score_names, '_nz'))
109
110 # Create nine single-column named dataframes, each containing the normalized z
111 # scores for each case. The input is the list `nz_transform_list` containing the
112 # nine normalization objects (each itself a list). That input is assigned to the
113 # .y argument of `walk2()`, while the names of the output objects `nz_names` are
114 # assigned to the .x argument. Within assign(), the value argument has as its
115 # innermost function `purrr::pluck()`, which extracts an element of a list in
116 # the .y input. In this case, what's being extracted is the `x.t`, the vector of
117 # normalized z scores. That vector is wrapped in `data.frame`, to coerce it into
118 # a data frame, which is then wrapped in `setNames`, which names the column of
119 # the resulting data frame using the variable names contained in the .x
120 # argument.
121 nz_transform_list %>%
122   walk2(
123     .x = c(nz_names),
124     .y = .,
125     .f = ~ assign(x = .x,
126                   value = setNames(data.frame(pluck(.y, 'x.t')), c(.x)),
127                   envir = .GlobalEnv)
128   )
129
130 # remove the normalization objects, which are no longer needed
131 rm(list = ls(nz_transform_list))
132
133
134 # DERIVE NORMALIZED T-SCORES FOR EACH CASE -----
135
136 # put the nine single column normalized z-score data frames into a list
137 nz_col_list <- mget(nz_names)
138
139 # Next snippet replaces the normalized z-score with

```

```

94         envir = .GlobalEnv)
95     )
96
97 # Each of the named objects (normalization for each score) created in the
98 # previous snippet is a list. Use base::mget to put these named objects into a
99 # 'list of lists'. Here, mget takes a single argument, a char vec holding the
100 # names of the lists that are to be put into the new list `nz_transform_list`
101
102 nz_transform_list <- mget(nz_transform_names)
103
104 # Create a char vec containing the names of the output objects for the next
105 # step. These output objects are single-column named dfs containing the
106 # normalized z scores corresponding to the raw score for each case. The objects and the single
107 # columns within them have the same names
108 nz_names <- c(paste0(score_names, '_nz'))
109
110 # Create nine single-column named dataframes, each containing the normalized z
111 # scores for each case. The input is the list `nz_transform_list` containing the
112 # nine normalization objects (each itself a list). That input is assigned to the
113 # .y argument of `walk2()`, while the names of the output objects `nz_names` are
114 # assigned to the .x argument. Within assign(), the value argument has as its
115 # innermost function `purrr::pluck()`, which extracts an element of a list in
116 # the .y input. In this case, what's being extracted is the `x.t`, the vector of
117 # normalized z scores. That vector is wrapped in `data.frame`, to coerce it into
118 # a data frame, which is then wrapped in `setNames`, which names the column of
119 # the resulting data frame using the variable names contained in the .x
120 # argument.
121 nz_transform_list %>%
122   walk2(
123     .x = c(nz_names),
124     .y = .,
125     .f = ~ assign(x = .x,
126                   value = setNames(data.frame(pluck(.y, 'x.t')), c(.x)),
127                   envir = .GlobalEnv)
128   )
129
130 # remove the normalization objects, which are no longer needed
131 rm(list = ls(nz_transform_list))
132
133
134 # DERIVE NORMALIZED T-SCORES FOR EACH CASE -----
135
136 # put the nine single column normalized z-score data frames into a list
137 nz_col_list <- mget(nz_names)
138
139 # Next snippet replaces the normalized z-score with

```

```

      h a normalized T-score (and
140 # truncates the T-score distribution).
141
142 # map2_dfc takes a list of data frames as input, a
      nd outputs a single data
143 # frame, binding the transformed output columns to
      gether. In map2_dfc, the input
144 # list is assigned to the .x argument, and the vec
      tor of score_names is assigned
145 # to the .y argument. Note the use of unquoting
      `!!`, `as.name`, and the
146 # specialized equals sign `:=` for NSE (non-standa
      rd evaluation)
147 NT_cols <- map2_dfc(nz_col_list, score_names, ~
148 .x %>% mutate(
149   !!as.name(paste0(.y, '_NT')) := round(!!as.name
      (paste0(.y, '_nz'))*10)+50)
150 ) %>% mutate_at(
151   vars(paste0(.y, '_NT')), ~ case_when(
152     .x < 25 ~ 25,
153     .x > 75 ~ 75,
154     TRUE ~ .x
155   )
156 ) %>%
157   select(
158     paste0(.y, '_NT')
159   )
160 ) %>%
161   mutate_if(is.numeric, as.integer)
162
163 # Bind the normalized T-score columns to the table
      containing raw scores for
164 # each case.
165 Child_512_Home <- Child_512_Home %>% bind_cols(NT_
      cols) %>%
166   mutate(clin_status = 'typ',
167     clin_dx = NA) %>%
168   select(IDNumber, Age, age_range, Gender:Region,
      data, clin_status, clin_dx, everything())
169
170 # write T-scores per case table to .csv
171 write_csv(Child_512_Home, here(
172   'OUTPUT-FILES/CHILD/T-SCORES-PER-CASE/Child-512-Ho
      me-T-Scores-per-case.csv')
173   # paste0(
174   #   'OUTPUT-FILES/CHILD/T-SCORES-PER-CASE/Child-
      512-Home-T-Scores-per-case-',
175   #   format(Sys.Date(), "%Y-%m-%d"),
176   #   '.csv'
177   # )
178 ),
179 na = ''
180 )
181
182 # clean up environment
183 rm(list = ls(pattern='.*_nz'))
184
185 # histogram to check normality
186 # MASS::truehist(Child_512_Home$TOT_NT, h = 1)
187 # hist_plot <- ggplot(data = Child_512_Home, aes(T
      OT_NT)) +
188 #   geom_histogram(
189 #     binwidth = .2,
190 #     col = "red"
191 #   ) +
192

```

```

      h a normalized T-score (and
140 # truncates the T-score distribution).
141
142 # map2_dfc takes a list of data frames as input, a
      nd outputs a single data
143 # frame, binding the transformed output columns to
      gether. In map2_dfc, the input
144 # list is assigned to the .x argument, and the vec
      tor of score_names is assigned
145 # to the .y argument. Note the use of unquoting
      `!!`, `as.name`, and the
146 # specialized equals sign `:=` for NSE (non-standa
      rd evaluation)
147 NT_cols <- map2_dfc(nz_col_list, score_names, ~
148 .x %>% mutate(
149   !!as.name(paste0(.y, '_NT')) := round(!!as.name
      (paste0(.y, '_nz'))*10)+50)
150 ) %>% mutate_at(
151   vars(paste0(.y, '_NT')), ~ case_when(
152     .x < 40 ~ 40,
153     .x > 80 ~ 80,
154     TRUE ~ .x
155   )
156 ) %>%
157   select(
158     paste0(.y, '_NT')
159   )
160 ) %>%
161   mutate_if(is.numeric, as.integer)
162
163 # Bind the normalized T-score columns to the table
      containing raw scores for
164 # each case.
165 Child_512_Home <- Child_512_Home %>% bind_cols(NT_
      cols) %>%
166   mutate(clin_status = 'typ',
167     clin_dx = NA) %>%
168   select(IDNumber, Age, age_range, Gender:Region,
      data, clin_status, clin_dx, everything())
169
170 # write T-scores per case table to .csv
171 write_csv(Child_512_Home, here(
172   'OUTPUT-FILES/NORMS-OUTPUT-4080T/Child-512-Home-T-
      Scores-per-case-4080T.csv')
173   # paste0(
174   #   'OUTPUT-FILES/CHILD/T-SCORES-PER-CASE/Child-
      512-Home-T-Scores-per-case-',
175   #   format(Sys.Date(), "%Y-%m-%d"),
176   #   '.csv'
177   # )
178 ),
179 na = ''
180 )
181
182 # clean up environment
183 rm(list = ls(pattern='.*_nz'))
184
185 # histogram to check normality
186 # MASS::truehist(Child_512_Home$TOT_NT, h = 1)
187 # hist_plot <- ggplot(data = Child_512_Home, aes(T
      OT_NT)) +
188 #   geom_histogram(
189 #     binwidth = .2,
190 #     col = "red"
191 #   ) +
192

```

```

#   scale_y_continuous(breaks = seq(0, 250, 25)) +
193 #   labs(title = "TOT_NT")
194 # print(hist_plot)
195
196 # GENERATE RAW-TO-T LOOKUP TABLES -----
-----
197
198 # Generate raw-to-T lookup columns. Handle TOT and
  subscale scores separately,
199 # because each type has different raw score range.
  Start with TOT. Input is
200 # stand sample with raw scores and normalized T scores
  for each case. Group
201 # cases by raw score, relationship between raw and
  T is many-to-one.
202 TOT_lookup <- Child_512_Home %>% group_by(
203   TOT_raw
204 ) %>%
205   # Because raw-to-T is many to one, all values of
  T are identical for each raw,
206   # and summarizing by the min value of T per raw
  yields the ONLY value of T per
207   # raw. But we need the raw column to contain all
  possible values of raw, and
208   # not all possible values of raw are represented
  in the stand sample. Thus
209   # current data object jumps possible raw values
  (e.g, raw = 62 and raw = 65
210   # might be adjacent rows in this table)
211   summarise(
212     TOT_NT = min(TOT_NT)
213   ) %>%
214   # complete expands the table vertically, filling
  in missing values of raw
215   # within the range given. This leaves NA cells for
  or T for those rows that
216   # didn't have raw values in the input object.
217   complete(
218     TOT_raw = 10:240
219   ) %>%
220   # fill replaces NA in T going down the table, with
  values from the last
221   # preceding (lagging) cell that was not NA.
222   fill(
223     TOT_NT
224   ) %>%
225   # A second call of fill is needed to handle inputs
  where the first cell(s) of
226   # T are NA. 2nd fill call is uses direction up to
  fill those first NA cells
227   # with the value from the first subsequent (leading)
  cell that is not NA.
228   fill(
229     TOT_NT,
230     .direction = "up"
231   ) %>%
232   rename(
233     raw = TOT_raw
234   ) %>%
235   mutate_at(
236     vars(TOT_NT), ~ case_when(
237       raw < 60 ~ NA_integer_,
238       TRUE ~ .x
239     )
240   )
241

```

```

#   scale_y_continuous(breaks = seq(0, 250, 25)) +
193 #   labs(title = "TOT_NT")
194 # print(hist_plot)
195
196 # GENERATE RAW-TO-T LOOKUP TABLES -----
-----
197
198 # Generate raw-to-T lookup columns. Handle TOT and
  subscale scores separately,
199 # because each type has different raw score range.
  Start with TOT. Input is
200 # stand sample with raw scores and normalized T scores
  for each case. Group
201 # cases by raw score, relationship between raw and
  T is many-to-one.
202 TOT_lookup <- Child_512_Home %>% group_by(
203   TOT_raw
204 ) %>%
205   # Because raw-to-T is many to one, all values of
  T are identical for each raw,
206   # and summarizing by the min value of T per raw
  yields the ONLY value of T per
207   # raw. But we need the raw column to contain all
  possible values of raw, and
208   # not all possible values of raw are represented
  in the stand sample. Thus
209   # current data object jumps possible raw values
  (e.g, raw = 62 and raw = 65
210   # might be adjacent rows in this table)
211   summarise(
212     TOT_NT = min(TOT_NT)
213   ) %>%
214   # complete expands the table vertically, filling
  in missing values of raw
215   # within the range given. This leaves NA cells for
  or T for those rows that
216   # didn't have raw values in the input object.
217   complete(
218     TOT_raw = 10:240
219   ) %>%
220   # fill replaces NA in T going down the table, with
  values from the last
221   # preceding (lagging) cell that was not NA.
222   fill(
223     TOT_NT
224   ) %>%
225   # A second call of fill is needed to handle inputs
  where the first cell(s) of
226   # T are NA. 2nd fill call is uses direction up to
  fill those first NA cells
227   # with the value from the first subsequent (leading)
  cell that is not NA.
228   fill(
229     TOT_NT,
230     .direction = "up"
231   ) %>%
232   rename(
233     raw = TOT_raw
234   ) %>%
235   mutate_at(
236     vars(TOT_NT), ~ case_when(
237       raw < 60 ~ NA_integer_,
238       TRUE ~ .x
239     )
240   )
241

```

```

242 # Repeat above for subscale raw-to-T columns.
243 subscale_names <- score_names[2:9]
244
245 subscale_lookup <- map(
246   subscale_names,
247   ~ Child_512_Home %>% group_by(
248     !!as.name(paste0(.x, '_raw'))
249   ) %>%
250     summarise(
251       !!as.name(paste0(.x, '_NT')) := min(!as.nam
e(paste0(.x, '_NT')))
252     ) %>%
253     complete(
254       !!as.name(paste0(.x, '_raw')) := 10:240
255     ) %>%
256     fill(
257       paste0(.x, '_NT')
258     ) %>%
259     fill(
260       paste0(.x, '_NT'),
261       .direction = "up"
262     ) %>%
263     rename(
264       raw = !!as.name(paste0(.x, '_raw'))
265     ) %>%
266     mutate_at(
267       vars(!as.name(paste0(.x, '_NT'))), ~ case_w
hen(
268         raw > 40 ~ NA_integer_,
269         TRUE ~ .x
270       )
271     )
272 ) %>%
273 reduce(
274   left_join,
275   by = 'raw'
276 )
277
278 # join TOT and subscale columns
279 all_lookup <- full_join(TOT_lookup, subscale_looku
p, by = 'raw')
280
281 all_lookup_col_names <- c(paste0(score_names, '_ra
w'))
282
283 # write final raw-to-T lookup table to .csv
284 write_csv(all_lookup, here(
285   'OUTPUT-FILES/CHILD/RAW-T-LOOKUP-TABLES/Child-512-
Home-raw-T-lookup.csv'
286   # paste0(
287   #   'OUTPUT-FILES/CHILD/RAW-T-LOOKUP-TABLES/Chil
d-512-Home-raw-T-lookup-',
288   #   format(Sys.Date(), "%Y-%m-%d"),
289   #   '.csv'
290   # )
291 ),
292 na = ''
293 )
294
295
296 # generate print pub format raw-to-T table
297 all_lookup_pub <- all_lookup %>%
298   # gather collapses wide table into three-column
tall table with key-value
299   # pairs: rawscore, scale(key var, many rows for
each scale), T(value

```

```

242 # Repeat above for subscale raw-to-T columns.
243 subscale_names <- score_names[2:9]
244
245 subscale_lookup <- map(
246   subscale_names,
247   ~ Child_512_Home %>% group_by(
248     !!as.name(paste0(.x, '_raw'))
249   ) %>%
250     summarise(
251       !!as.name(paste0(.x, '_NT')) := min(!as.nam
e(paste0(.x, '_NT')))
252     ) %>%
253     complete(
254       !!as.name(paste0(.x, '_raw')) := 10:240
255     ) %>%
256     fill(
257       paste0(.x, '_NT')
258     ) %>%
259     fill(
260       paste0(.x, '_NT'),
261       .direction = "up"
262     ) %>%
263     rename(
264       raw = !!as.name(paste0(.x, '_raw'))
265     ) %>%
266     mutate_at(
267       vars(!as.name(paste0(.x, '_NT'))), ~ case_w
hen(
268         raw > 40 ~ NA_integer_,
269         TRUE ~ .x
270       )
271     )
272 ) %>%
273 reduce(
274   left_join,
275   by = 'raw'
276 )
277
278 # join TOT and subscale columns
279 all_lookup <- full_join(TOT_lookup, subscale_looku
p, by = 'raw')
280
281 all_lookup_col_names <- c(paste0(score_names, '_ra
w'))
282
283 # write final raw-to-T lookup table to .csv
284 write_csv(all_lookup, here(
285   'OUTPUT-FILES/NORMS-OUTPUT-4080T/Child-512-Home-ra
w-T-lookup-4080T.csv'
286   # paste0(
287   #   'OUTPUT-FILES/CHILD/RAW-T-LOOKUP-TABLES/Chil
d-512-Home-raw-T-lookup-',
288   #   format(Sys.Date(), "%Y-%m-%d"),
289   #   '.csv'
290   # )
291 ),
292 na = ''
293 )
294
295
296 # generate print pub format raw-to-T table
297 all_lookup_pub <- all_lookup %>%
298   # gather collapses wide table into three-column
tall table with key-value
299   # pairs: rawscore, scale(key var, many rows for
each scale), T(value

```

```

300 # var, one row for each value of T within each s
    cale)
301 gather(scale, T,-raw) %>%
302 group_by(scale) %>%
303 # expand the table vertically, adding new rows,
    so there's a row for every possible T value
304 complete(T = 25:75) %>%
305 ungroup() %>%
306 # regroup table by two levels
307 group_by(scale, T) %>%
308 # filter step retains all 1-row groups, and the
    first and last rows of any
309 # multi-row groups. n() == 1 returns 1-row group
    s; n() > 1 & row_number()
310 # %in% c(1, n()) returns rows of multi-row group
    s with the row number of
311 # either 1 (first row), or n() which is the numb
    er of rows and also the
312 # number of the last row. The first and last row
    s hold the min and max
313 # values of raw for that value of T (the groupin
    g variable)
314 filter(n() == 1 | n() > 1 & row_number() %in% c
    (1, n())) %>%
315 # Summarise creates a table with one row per gro
    up (one row per
316 # possible value of T). For the 1-row groups, st
    r_c simply passes the
317 # value of raw as a string; for the multi-row gr
    oups, str_c joins the min
318 # and max values of raw with the '--' separator.
319 summarise(raw = str_c(raw, collapse = '--')) %>%
320 # recode missing values of raw to '-'
321 mutate_at(vars(raw), ~ case_when(is.na(.x) ~ '-'
    ', TRUE ~ .x)) %>%
322 # sort on two levels
323 arrange(scale, desc(T)) %>%
324 # spread table back to wide, all values of T (on
    e row for each), scale
325 # columns filled with values of rawscore
326 spread(scale, raw) %>%
327 # sort descending on T
328 arrange(desc(T)) %>%
329 # rename with desired final column names
330 rename_at(vars(ends_with('_NT')), ~ gsub("_NT",
    "_raw", .)) %>%
331 # order columns left-to-right
332 select(T, all_lookup_col_names)
333
334 # write final print format raw-to-T lookup table t
    o .csv
335 write_csv(all_lookup_pub, here(
336 'OUTPUT-FILES/CHILD/PRINT-FORMAT-NORMS-TABLES/Chil
    d-512-Home-print-raw-T-lookup.csv')
337 # paste0(
338 # 'OUTPUT-FILES/CHILD/PRINT-FORMAT-NORMS-TABLE
    S/Child-512-Home-print-raw-T-lookup-',
339 # format(Sys.Date(), "%Y-%m-%d"),
340 # '.csv'
341 # )
342 ),
343 na = ''
344

```

```

300 # var, one row for each value of T within each s
    cale)
301 gather(scale, T,-raw) %>%
302 group_by(scale) %>%
303 # expand the table vertically, adding new rows,
    so there's a row for every possible T value
304 complete(T = 40:80) %>%
305 ungroup() %>%
306 # regroup table by two levels
307 group_by(scale, T) %>%
308 # filter step retains all 1-row groups, and the
    first and last rows of any
309 # multi-row groups. n() == 1 returns 1-row group
    s; n() > 1 & row_number()
310 # %in% c(1, n()) returns rows of multi-row group
    s with the row number of
311 # either 1 (first row), or n() which is the numb
    er of rows and also the
312 # number of the last row. The first and last row
    s hold the min and max
313 # values of raw for that value of T (the groupin
    g variable)
314 filter(n() == 1 | n() > 1 & row_number() %in% c
    (1, n())) %>%
315 # Summarise creates a table with one row per gro
    up (one row per
316 # possible value of T). For the 1-row groups, st
    r_c simply passes the
317 # value of raw as a string; for the multi-row gr
    oups, str_c joins the min
318 # and max values of raw with the '--' separator.
319 summarise(raw = str_c(raw, collapse = '--')) %>%
320 # recode missing values of raw to '-'
321 mutate_at(vars(raw), ~ case_when(is.na(.x) ~ '-'
    ', TRUE ~ .x)) %>%
322 # sort on two levels
323 arrange(scale, desc(T)) %>%
324 # spread table back to wide, all values of T (on
    e row for each), scale
325 # columns filled with values of rawscore
326 spread(scale, raw) %>%
327 # sort descending on T
328 arrange(desc(T)) %>%
329 # rename with desired final column names
330 rename_at(vars(ends_with('_NT')), ~ gsub("_NT",
    "_raw", .)) %>%
331 # order columns left-to-right
332 select(T, all_lookup_col_names) %>%
333 # drop row where T == NA
334 filter(!is.na(T))
335
336 # write final print format raw-to-T lookup table t
    o .csv
337 write_csv(all_lookup_pub, here(
338 'OUTPUT-FILES/NORMS-OUTPUT-4080T/Child-512-Home-pr
    int-raw-T-lookup-4080T.csv')
339 # paste0(
340 # 'OUTPUT-FILES/CHILD/PRINT-FORMAT-NORMS-TABLE
    S/Child-512-Home-print-raw-T-lookup-',
341 # format(Sys.Date(), "%Y-%m-%d"),
342 # '.csv'
343 # )
344 ),
345 na = ''
346

```

```

)
345
346 # write raw score descriptives for all scales (usi
ng psych::describe)
347 Child_512_Home_raw_desc <-
348   Child_512_Home %>%
349   select(contains('raw')) %>%
350   describe(fast = T) %>%
351   rownames_to_column() %>%
352   rename(scale = rowname) %>%
353   select(scale, n, mean, sd) %>%
354   mutate_at(vars(mean, sd), ~(round(., 2)))
355
356 write_csv(Child_512_Home_raw_desc, here(
357   'OUTPUT-FILES/CHILD/DESCRIPTIVES/Child-512-Home-
raw-desc.csv'
358   # paste0(
359   #   'OUTPUT-FILES/CHILD/DESCRIPTIVES/Child-512-H
ome-row-desc-',
360   #   format(Sys.Date(), "%Y-%m-%d"),
361   #   '.csv'
362   # )
363 ),
364 na = ''
365 )
366
367 # write table of demographic counts
368
369 var_order <- c("data", "age_range", "Age", "Gende
r", "ParentHighestEducation", "HighestEducation",
370               "Ethnicity", "Region")
371
372 cat_order <- c(
373   # data
374   NA, "SM", "Qual", "Sp", "Daycare", "In-house-En
g", "In-house-Sp", "In-house-Alt",
375   # age_range
376   NA, "3.5 to 6 mo", "03.5 to 10 mo", "7 to 10.5 m
o", "09.5 to 20 mo", "11 to 31.5 mo",
377   "21 to 31.5 mo", "5 to 8 years", "9 to 12 year
s", "12 to 13 years", "14 to 15 years",
378   "16 to 17 years", "18 to 21 years", "21.00 to 3
0.99 years", "31.00 to 40.99 years",
379   "41.00 to 50.99 years", "51.00 to 64.99 years",
"65.00 to 99.99 years",
380   # Age
381   "2", "3", "4", "5",
382   # Gender
383   NA, "Male", "Female",
384   # ParentHighestEducation & HighestEducation
385   NA, "Did not complete high school (no diploma)",
"High school graduate (including GED)",
386   "Some college or associate degree", "Bachelor's
degree or higher",
387   # Ethnicity
388   NA, "Hispanic", "Asian", "Black", "White", "Amer
icanIndAlaskanNat",
389   "NativeHawPacIsl", "MultiRacial", "Other",
390   # Region
391   NA, "northeast", "midwest", "south", "west")
392
393
394 Child_512_Home_demo_counts <- Child_512_Home %>%
395   select(data, age_range, ParentHighestEducation,
Gender, Ethnicity, Region) %>%
396   gather("Variable", "Category") %>%
397   group_by(Variable, Category) %>%

```

```

)
347
348 # write raw score descriptives for all scales (usi
ng psych::describe)
349 Child_512_Home_raw_desc <-
350   Child_512_Home %>%
351   select(contains('raw')) %>%
352   describe(fast = T) %>%
353   rownames_to_column() %>%
354   rename(scale = rowname) %>%
355   select(scale, n, mean, sd) %>%
356   mutate_at(vars(mean, sd), ~(round(., 2)))
357
358 write_csv(Child_512_Home_raw_desc, here(
359   'OUTPUT-FILES/CHILD/DESCRIPTIVES/Child-512-Home-
raw-desc.csv'
360   # paste0(
361   #   'OUTPUT-FILES/CHILD/DESCRIPTIVES/Child-512-H
ome-row-desc-',
362   #   format(Sys.Date(), "%Y-%m-%d"),
363   #   '.csv'
364   # )
365 ),
366 na = ''
367 )
368
369 # write table of demographic counts
370
371 var_order <- c("data", "age_range", "Age", "Gende
r", "ParentHighestEducation", "HighestEducation",
372               "Ethnicity", "Region")
373
374 cat_order <- c(
375   # data
376   NA, "SM", "Qual", "Sp", "Daycare", "In-house-En
g", "In-house-Sp", "In-house-Alt",
377   # age_range
378   NA, "3.5 to 6 mo", "03.5 to 10 mo", "7 to 10.5 m
o", "09.5 to 20 mo", "11 to 31.5 mo",
379   "21 to 31.5 mo", "5 to 8 years", "9 to 12 year
s", "12 to 13 years", "14 to 15 years",
380   "16 to 17 years", "18 to 21 years", "21.00 to 3
0.99 years", "31.00 to 40.99 years",
381   "41.00 to 50.99 years", "51.00 to 64.99 years",
"65.00 to 99.99 years",
382   # Age
383   "2", "3", "4", "5",
384   # Gender
385   NA, "Male", "Female",
386   # ParentHighestEducation & HighestEducation
387   NA, "Did not complete high school (no diploma)",
"High school graduate (including GED)",
388   "Some college or associate degree", "Bachelor's
degree or higher",
389   # Ethnicity
390   NA, "Hispanic", "Asian", "Black", "White", "Amer
icanIndAlaskanNat",
391   "NativeHawPacIsl", "MultiRacial", "Other",
392   # Region
393   NA, "northeast", "midwest", "south", "west")
394
395
396 Child_512_Home_demo_counts <- Child_512_Home %>%
397   select(data, age_range, ParentHighestEducation,
Gender, Ethnicity, Region) %>%
398   gather("Variable", "Category") %>%
399   group_by(Variable, Category) %>%

```



```

398   count(Variable, Category) %>%
399   arrange(match(Variable, var_order), match(Category, cat_order)) %>%
400   ungroup() %>%
401   mutate(Variable = case_when(
402     lag(Variable) == "data" & Variable == "data" ~
403     "",
404     lag(Variable) == "age_range" & Variable == "age_range" ~ "",
405     lag(Variable) == "Gender" & Variable == "Gender" ~ "",
406     lag(Variable) == "ParentHighestEducation" & Variable == "ParentHighestEducation" ~ "",
407     lag(Variable) == "Ethnicity" & Variable == "Ethnicity" ~ "",
408     lag(Variable) == "Region" & Variable == "Region" ~ "",
409     TRUE ~ Variable
410   ))
411 write_csv(Child_512_Home_demo_counts, here(
412   'OUTPUT-FILES/CHILD/DESCRIPTIVES/Child-512-Home-demo-counts.csv'
413   # paste0(
414   #   'OUTPUT-FILES/CHILD/DESCRIPTIVES/Child-512-Home-demo-counts-',
415   #   format(Sys.Date(), "%Y-%m-%d"),
416   #   '.csv'
417   # )
418 ),
419 na = '(missing)'
420 )
421
422
423
424
425

```

```

400   count(Variable, Category) %>%
401   arrange(match(Variable, var_order), match(Category, cat_order)) %>%
402   ungroup() %>%
403   mutate(Variable = case_when(
404     lag(Variable) == "data" & Variable == "data" ~
405     "",
406     lag(Variable) == "age_range" & Variable == "age_range" ~ "",
407     lag(Variable) == "Gender" & Variable == "Gender" ~ "",
408     lag(Variable) == "ParentHighestEducation" & Variable == "ParentHighestEducation" ~ "",
409     lag(Variable) == "Ethnicity" & Variable == "Ethnicity" ~ "",
410     lag(Variable) == "Region" & Variable == "Region" ~ "",
411     TRUE ~ Variable
412   ))
413 write_csv(Child_512_Home_demo_counts, here(
414   'OUTPUT-FILES/CHILD/DESCRIPTIVES/Child-512-Home-demo-counts.csv'
415   # paste0(
416   #   'OUTPUT-FILES/CHILD/DESCRIPTIVES/Child-512-Home-demo-counts-',
417   #   format(Sys.Date(), "%Y-%m-%d"),
418   #   '.csv'
419   # )
420 ),
421 na = '(missing)'
422 )
423
424
425
426
427

```