

# Tidy questionnaire data: demo

```
library(tidyverse)
```

## About

The goal of this notebook is to compare various data analysis programming approaches to scoring questionnaire data—which is perhaps the most common operation to perform on questionnaire data.

One common approach is to have the data organized into a one-agent-per-row format so that responses to various questions are encoded in separate columns and by doing operations on those columns, new score columns can be computed.

When the data is organized into a one-response-per-row format (which we call the tidy questionnaire format), then several strategies are possible and we will explore maybe some of them.

## Sample dataset

```
bfi2 <- read_csv("data/x_bfi2.csv")
psqi <- read_csv("data/psqi.csv")
```

Here we will use a real, albeit small data set, as our goal is only to explore coding strategies that are nevertheless meant to apply to real use cases. This dataset contains data from 8 participants who completed at least one of two standard questionnaires, either once or twice. For simplicity, this dataset is offered as two distinct tabular csv files, one per questionnaire.

The first questionnaire in this dataset is a slightly modified version of the bfi-2 personality test which contains 60 questions. We picked this questionnaire for this example because it is well-known and widely used, contains many items and can be used to compute a large number of scores and subscores (called “domain scales” and “fact scales”). This dataset contains data for only three participants: two of these completed the questionnaire twice, and of them completed it only once.

The second questionnaire in this dataset is the “Pittsburgh Sleep Quality Index (PSQI)”. We chose this questionnaire because it uses a wide range of responses

types (e.g., time of day, number of minutes, hours per night, ordered categories representing a frequency, open text responses, ratings of quality) and uses more complex scoring rules—this questionnaire may therefore pose additional challenges for data organization and analysis. This dataset contains data from 5 participants who all completed that questionnaire twice. Furthermore, one of these participants (agent\_id “030”) is also included in the bfi-2 data.

For each question we recorded what response option participants chose, how long it took them to select that option once the question was displayed (each question was displayed one by one on the screen), and how long it took them to validate their response by clicking on the “Next” button (participants could change their responses, while a given question was displayed, but once they clicked “Next”, they were presented with the next question and could not navigate back to earlier questions.) These are just a subset of information that could be collected per response, which is nevertheless more than what seems to be typically collected in questionnaire data.

### Data formats: tidy/long vs untidy/wide

We stored the two data tables in the tidy questionnaire data format and will now create a version of these two data files that follows the “one-person-per-row” (aka “wide”) format. Furthermore, we will combine both pairs of files to form a combined tidy and a combined wide table respectively. This is to illustrate in particular how data from multiple questionnaires may be combined and how sparsity in each questionnaire data affects the size of the resulting combined tables.

```
# step 1: create wide data versions of the two tables
bfi2_wide <- bfi2 |> pivot_wider(id_cols = c(agent_id, instrument_id),
                                      names_from = c(stimulus_id, iteration),
                                      names_glue = "{stimulus_id}_{.value}_i{iteration}",
                                      values_from = c(trial_index,
                                                      stimulus_type,
                                                      stimulus_description,
                                                      response_option_index,
                                                      response_description,
                                                      response_numeric,
                                                      response_time,
                                                      response_validation_time))

psqi_wide <- psqi |> pivot_wider(id_cols = c(agent_id, instrument_id),
                                      names_from = c(stimulus_id, iteration),
                                      names_glue = "{stimulus_id}_{.value}_i{iteration}",
                                      values_from = c(trial_index,
                                                      stimulus_type,
                                                      stimulus_description,
                                                      response_option_index,
```

```

    response_description,
    response_numeric,
    response_time,
    response_validation_time))

# step 2: create a table that combines data from the two questionnaires
D_tidy <- rbind(bfi2, psqi)

D_wide <- full_join(bfi2_wide, psqi_wide, by = "agent_id")

# step 3: save the generated tables as csv files
write.csv(bfi2_wide, file = 'data/x_bfi2_wide.csv')
write.csv(psqi_wide, file = 'data/psqi_wide.csv')
write.csv(D_tidy, file = 'data/D_tidy.csv')
write.csv(D_wide, file = 'data/D_wide.csv')

```

Table 1: Dimensions and NA counts for each dataset, separately and combined, under the tidy and the wide tabular formats.

dataset	rows	columns	rows x cols	NAs	csv size
bfi2	305	12	3660	15	31.5 KB
psqi	190	12	2280	100	27.5 KB
D_tidy (combined)	495	12	5940	115	66.6 KB
	NA	NA	NA	NA	
bfi2_wide	3	978	2934	503	59.6 KB
psqi_wide	5	306	1530	100	36.2 KB
D_wide (combined)	7	1283	8981	5121	109.1 KB

There are several important points to note in the table above. The tidy data tables all have the same small number of columns; their number of rows will vary depending on the amount of data collected. Conversely, their wide-formatted counterparts have a number of rows and columns that both depend on the specific dataset and study design (e.g., more repetitions of the same questions would lead to more columns). Thus the shape of the tables when they are wide is less predictable. Furthermore, because columns represent the attributes of an observation, the wide format is much more difficult to process. For example, there are 978 columns in the wide version of the bfi2 data but only 12 in its tidy version. When looking at the number of missing values it is clear to see that in some cases (when data is sparse), the number of missing values increases substantially when transforming the data from tidy to wide (for bfi2, that number increased from 15 to 503). Another important observation is that when combining tidy questionnaire tables, the resulting table is easily predictable: the numbers of rows and the number of NAs are simply added, while the number

(and meaning) of the columns remain the same. In contrast, when combining wide data tables, the resulting table is unpredictable without knowledge about the content of the data. Note also the large increase in NAs for this combined table (5121) compared to the combined tidy data (115).

We also wanted to assess whether these data formats affect the size of the data. One simple but somewhat naive way of assessing size is to count the number of cells in each table. While this may in principle serve as a proxy, it is not that straightforward because the data in each cell may require more or less memory space depending for instance on how it is encoded (e.g., a boolean vs a string). The strategy we chose here was to save the various data tables as csv files and assess the size of those files. As can be seen in the table, the wide formats in our example are systematically larger than their tidy counterparts.

## Scoring questionnaires

The next step will be to score each of the questionnaires, using the tidy and wide data formats and compare their advantages and disadvantages. In addition to computing those scores we will also compute the median response time to answer the corresponding questions. While this is not classically done, the idea of including it here is to assess how computing additional metrics on more than one response attribute would affect the complexity of the code.

### BFI-2

#### The wide data format

We'll start with the wide data format as this is the most common and the "approach" to scoring such data appears at least conceptually the most straightforward: because the data is already organized in an one-row-per-person format, scoring a questionnaire simply implies doing specific operations on a subset of the columns of that table.

Let's start with a simple example: computing the *anxiety* score from the bfi2 data. According to the [documentation](#), this requires using the responses from the items Anxiety: 4R, 19, 34, 49R (where R indicates items that need to be reverse coded).

So our first challenge is to figure out how to find the responses to those times in our dataset.

```
# custom function to convert response option index into the response score as defined in doc
compute_agreement_rate_vector <- function(x, level_count = 7, reverse = FALSE){
  rate <- (x-1)/(level_count-1)
  if (reverse){
    rate <- 1-rate
  }
  rate
```

```

}

# custom function to compute the anxiety score
compute_anxiety_score <- function(df, iteration){
  # - Anxiety: 4R, 19, 34, 49R
  iteration_text <- paste0('_i', iteration)
  (compute_agreement_rate_vector(df[[paste0('bfi-2_q_4_response_option_index', iteration_text)]]) +
   compute_agreement_rate_vector(df[[paste0('bfi-2_q_19_response_option_index', iteration_text)]]) +
   compute_agreement_rate_vector(df[[paste0('bfi-2_q_34_response_option_index', iteration_text)]]) +
   compute_agreement_rate_vector(df[[paste0('bfi-2_q_49_response_option_index', iteration_text)]])) / 4
}

# computing the anxiety score and adding as a new column to the data
D_wide$anxiety_score_i1 <- compute_anxiety_score(D_wide, iteration = 1)
D_wide$anxiety_score_i2 <- compute_anxiety_score(D_wide, iteration = 2)

# we can do something similar to compute the average response times
compute_anxiety_response_times <- function(df, iteration){
  iteration_text <- paste0('_i', iteration)
  (df[[paste0('bfi-2_q_4_response_time', iteration_text)]]) +
  df[[paste0('bfi-2_q_19_response_time', iteration_text)]]) +
  df[[paste0('bfi-2_q_34_response_time', iteration_text)]]) +
  df[[paste0('bfi-2_q_49_response_time', iteration_text)]]) / 4
}

D_wide$anxiety_response_times_i1 <- compute_anxiety_response_times(D_wide, iteration = 1)
D_wide$anxiety_response_times_i2 <- compute_anxiety_response_times(D_wide, iteration = 2)

D_wide[c('agent_id', 'anxiety_score_i1', 'anxiety_score_i2', 'anxiety_response_times_i1', 'anxiety_response_times_i2)]
```

# A tibble: 7 x 5

	agent_id	anxiety_score_i1	anxiety_score_i2	anxiety_response_times_i1
	<chr>	<dbl>	<dbl>	<dbl>
1	002	0.542	0.708	1.95
2	019	0.458	0.625	2.48
3	030	0.625	NA	2.28
4	008	NA	NA	NA
5	012	NA	NA	NA
6	026	NA	NA	NA
7	031	NA	NA	NA

# i 1 more variable: anxiety\_response\_times\_i2 <dbl>

There are several points we want to highlight about the code above. Firstly,

this type of code tends to be quite verbose. We wrote functions here to aim to make that code compact; in practice it is rather common to not see any such functions but instead see long assignment statements.

Secondly, the code above is strongly tied, not only to information about the questionnaire (e.g., which items to pick)—which is unavoidable—, it is also strongly tied to the specific way in which the column names have been arbitrarily constructed. If for some reason we change how column names are constructed, we could have to replace all of the code above. Furthermore, if the study had not involved repetitions, the naming would have been different and thus we would have again to change the code.

Third, as in our example there are (up to) 2 repetitions of this questionnaire, we have to manually run the scoring code on each value of those iterations. Again, if there were a different number of iterations or different intentions to group data (e.g., completing surveys in morning vs evening), we would again have to change the code above.

Finally, when computing multiple aggregates on a particular response (in our example, how long they took to respond to those questions), we can see that it requires pretty much writing an independent chunk of code (the code for the anxiety response time and the code for anxiety scores are run independently from each other). This means that there are redundancies and the potential for mistakes. For example, the selection of items is the same in both cases (i.e., 4R, 19, 34, 49R), but this selection is specified in two different locations (both as part of the column names to be selected). As these are entered manually, and because for a given study there could potentially be a large number of such duplicated lists (e.g., there are 20 scores that one can compute out of this bfi2 questionnaire) which makes it hard to not make mistakes and to check that no mistakes were made. The above code takes about 25 lines to compute two metrics for one dimension; if we extrapolate, scoring the bfi2 questionnaire in this way would require 500 lines of code.

The names of the aggregated values need to be specified by the data analyst. For example, here we chose “anxiety\\_score\\_i1”. This is again an arbitrary choice (we could have picked instead “score\\_bfi\\_anx\\_1”, “anxiety\\_r1\\_bfi2”, or something else). This custom naming of the outputs requires extra work, is specific to the dataset (if there were more repetitions, we would need additional assignment statements); it will also cause the subsequent code to be tailor made to those specific column names.

It is of course possible to develop more custom code (i.e., more functions) to make the data analysis code more compact, less redundant and less error prone. But that code would require extra work, it would likely be highly specific to the way the column names were constructed for this particular dataset, and in practice, this does not seem to be what data analysts do.

## The tidy data format

We will now do the same type of analysis starting with the tidy data format.

```
score_bfi2_anxiety <- function(df){

  df |>
    # "select" the relevant items for this scale
    filter(stimulus_id %in% paste0("bfi-2_q_", c(4, 19, 34, 49))) |>
    # recode responses into 0-1
    mutate(response_score = ifelse(stimulus_id %in% paste0("bfi-2_q_", c(4, 49)),
                                    compute_agreement_rate_vector(response_option_index, reverse = TRUE),
                                    compute_agreement_rate_vector(response_option_index, reverse = FALSE)))
    # compute score for this scale
    summarize(
      dimension = "bfi2_extraversion",
      score = mean(response_score),
      response_time_mean = mean(response_time),
      .groups = "drop")

}

# compute the scores for our dataset
bfi2_anxiety <- D_tidy |>
  group_by(agent_id, iteration) |>
  score_bfi2_anxiety()

knitr::kable(bfi2_anxiety)
```

agent_id	iteration	dimension	score	response_time_mean
002	1	bfi2_extraversion	0.5416667	1.9545
002	2	bfi2_extraversion	0.7083333	2.0165
019	1	bfi2_extraversion	0.4583333	2.4805
019	2	bfi2_extraversion	0.6250000	14.0690
030	1	bfi2_extraversion	0.6250000	2.2760

There are several points worth noting about the code above. Firstly, the code in `score_bfi2_anxiety()` only contains information about the IDs of the items that need to be used and how they are to be used. It does not contain any information related to the specifics of the study or data analysis intention. For instance, there is no notion of “repetition” in this code. Second, the selection of relevant items is done only once with the filter statement; this makes the code more compact and less error prone. Thirdly, the way that aggregates are computed on the data—which is done via the `summarize()`—makes it very easy

to add additional metrics. For instance, if one wanted to also compute the minimum, maximum of and standard deviation of the response times, it would only require one additional statement per metric within the summarise function (e.g., `response_time_min = min(response_time)`); this is in stark contrast with how we analyzed the wide data.

Forth, if we look at how those functions are being applied to the data, we can see that by using the `group_by()` function (which is part of the `dplyr` package, included in the `tidyverse`) we can apply the scoring functions for any type of subset of our data. Here we specify that we want to have separate scoring for each agent and for each iteration. We don't need to know or specify in advance how many iterations are in the dataset. The number of iterations does not impact the amount of code one needs to write. This code is also easy to update in the future; if there was a “`time_of_day`” column, that column could be used to compute different scores for whatever levels that variable has by simply changing the contents of that `group_by()` function. Importantly, this coding approach cleanly separates the scoring function which is tied only to the questionnaire used from the specific study design and data analysis strategies (which in our example is embedded in the `group_by()` function call).

It is also important to note that the output of the scoring “anxiety” following this approach is again a tidy table where no extra effort was required to name things. The shape of this table will depend on the specific analysis that was computed (e.g., if there are more iterations, there will be more rows), but the columns of that table will reflect the various metrics computed in the summary call.

Of course at some point one may want to display the table as a one-row-per-person table; this can be done quite easily.

### Scoring the bfi2 questionnaire as a whole

As stated earlier we won't use the strategy we employed for the wide data to score all of the dimensions as this would take us quite some time to do and would not be very interesting. But doing this exercise would certainly allow the reader to experience how painful that could be.

Here we will only use the tidy data as a starting point. We will abstract the code above a bit more to show how compact questionnaire data analysis may be.

```
# function to "recode" response values
compute_agreement_rate <- function(df, col = "response_option_index", level_count = 7, reverse = FALSE) {
  rate <- (df[[col]] - 1) / (level_count - 1)
  if (reverse){
    rate <- 1 - rate
  }
  rate
```

```

}

# function takes as input a data frame (df) and a list containing the parameters of a specific
# recode and aggregate
score_questionnaire <- function(df, scoring_params){

  # if reverse_ids is not defined, copy stimulus_ids
  if (all(is.na(scoring_params$reverse_ids))){
    scoring_params$reverse_ids <- "___none__"
  }

  df |>
    filter(stimulus_id %in% scoring_params$stimulus_ids) |>
    mutate(response_score = ifelse(stimulus_id %in% scoring_params$reverse_ids,
                                    scoring_params$recode_func(pick(everything())), reverse =
                                    scoring_params$recode_func(pick(everything())), reverse =
    summarize(dimension = scoring_params$dimension,
              score = mean(response_score, na.rm = TRUE),
              response_time = mean(response_time, na.rm = TRUE),
              n = n(),
              na_n = sum(is.na(response_time)),
              .groups = "drop"
            )
}

```

The code above is quite generic and applies to the common case where response option indices are converted to numbers using a particular set of rules and then aggregated. Because this code is generic and could apply to a wide range of questionnaires, it could be part of a package and reused broadly, preventing the need to write custom code.

We wrote the code in a way that the `score_questionnaire()` takes as input the questionnaire data and a list of parameters that specify how to score a particular dimension. This type of code that separates the specifics of a score (tied to a particular instrument) from the mechanics of computing the score is preferable over hard coding the specifics in the analysis code directly (as we did earlier) as it makes it easier to review if the specification is correct or not.

To compute a specific score, we will then need to do the following:

```

# we can now encode the scoring rules for the questionnaire in a compact way:
bfi2_extraversion <- list(
  dimension = "bfi2_extraversion",
  recode_func = compute_agreement_rate,
  stimulus_ids = paste0("bfi-2_q_", c(1, 6, 11, 16, 21, 26, 31, 36, 41, 46, 51, 56)),

```

```

    reverse_ids = paste0("bfi-2_q_", c(11, 16, 26, 31, 36, 51))
)

# and use the generic code to compute the results for that dimension:
D_tidy |>
  group_by(agent_id, iteration) |>
  score_questionnaire(bfi2_extraversion) |>
  knitr::kable()

```

agent_id	iteration	dimension	score	response_time	n	na_n
002	1	bfi2_extraversion	0.6111111	11.823333	12	0
002	2	bfi2_extraversion	0.5972222	2.045917	12	0
019	1	bfi2_extraversion	0.7500000	2.151667	12	0
019	2	bfi2_extraversion	0.6805556	13.529833	12	0
030	1	bfi2_extraversion	0.6388889	1.762917	12	0

To compute all of the 20 scores from the bfi2 questionnaires, we first need to specify the parameters for each score. This can be done all in one, easy to review, place:

```

# Storing all the scoring rules for the bfi-2 questionnaire in lists:

## Extraversion: 1, 6, 11R, 16R, 21, 26R, 31R, 36R, 41, 46, 51R, 56
bfi2_extraversion <- list(
  dimension = "bfi2_extraversion",
  recode_func = compute_agreement_rate,
  stimulus_ids = paste0("bfi-2_q_", c(1, 6, 11, 16, 21, 26, 31, 36, 41, 46, 51, 56)),
  reverse_ids = paste0("bfi-2_q_", c(11, 16, 26, 31, 36, 51))
)

## Agreeableness: 2, 7, 12R, 17R, 22R, 27, 32, 37R, 42R, 47R, 52, 57
bfi2_agreeableness <- list(
  dimension = "bfi2_agreeableness",
  recode_func = compute_agreement_rate,
  stimulus_ids = paste0("bfi-2_q_", c(2, 7, 12, 17, 22, 27, 32, 37, 42, 47, 52, 57)),
  reverse_ids = paste0("bfi-2_q_", c(12, 17, 22, 37, 42, 47))
)

## Conscientiousness: 3R, 8R, 13, 18, 23R, 28R, 33, 38, 43, 48R, 53, 58R
bfi2_conscientiousness <- list(
  dimension = "bfi2_conscientiousness",
  recode_func = compute_agreement_rate,

```

```

stimulus_ids = paste0("bfi-2_q_", c(3, 8, 13, 18, 23, 28, 33, 38, 43, 48, 53, 58)),
reverse_ids = paste0("bfi-2_q_", c(3, 8, 23, 28, 48, 58))
)

## Negative Emotionality: 4R, 9R, 14, 19, 24R, 29R, 34, 39, 44R, 49R, 54, 59
bfi2_negative <- list(
  dimension = "bfi2_negative",
  recode_func = compute_agreement_rate,
  stimulus_ids = paste0("bfi-2_q_", c(4, 9, 14, 19, 24, 29, 34, 39, 44, 49, 54, 59)),
  reverse_ids = paste0("bfi-2_q_", c(4, 9, 24, 29, 44, 49))
)

## Open-Mindedness: 5R, 10, 15, 20, 25R, 30R, 35, 40, 45R, 50R, 55R, 60
bfi2_open <- list(
  dimension = "bfi2_open",
  recode_func = compute_agreement_rate,
  stimulus_ids = paste0("bfi-2_q_", c(5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60)),
  reverse_ids = paste0("bfi-2_q_", c(5, 25, 30, 45, 50, 55))
)

## Sociability: 1, 16R, 31R, 46
bfi2_sociability <- list(
  dimension = "bfi2_sociability",
  recode_func = compute_agreement_rate,
  stimulus_ids = paste0("bfi-2_q_", c(1, 16, 31, 46)),
  reverse_ids = paste0("bfi-2_q_", c(16, 31))
)

## Assertiveness: 6, 21, 36R, 51R
bfi2_assertiveness <- list(
  dimension = "bfi2_assertiveness",
  recode_func = compute_agreement_rate,
  stimulus_ids = paste0("bfi-2_q_", c(6, 21, 36, 51)),
  reverse_ids = paste0("bfi-2_q_", c(36, 51))
)

## Energy Level: 11R, 26R, 41, 56
bfi2_energy <- list(
  dimension = "bfi2_energy",
  recode_func = compute_agreement_rate,
  stimulus_ids = paste0("bfi-2_q_", c(11, 26, 41, 56)),
  reverse_ids = paste0("bfi-2_q_", c(11, 26))
)

## Compassion: 2, 17R, 32, 47R

```

```

bfi2_compassion <- list(
  dimension = "bfi2_compassion",
  recode_func = compute_agreement_rate,
  stimulus_ids = paste0("bfi-2_q_", c(2, 17, 32, 47)),
  reverse_ids = paste0("bfi-2_q_", c(17, 47))
)

## Respectfulness: 7, 22R, 37R, 52
bfi2_respectfulness <- list(
  dimension = "bfi2_respectfulness",
  recode_func = compute_agreement_rate,
  stimulus_ids = paste0("bfi-2_q_", c(7, 22, 37, 52)),
  reverse_ids = paste0("bfi-2_q_", c(22, 37))
)

## Trust: 12R, 27, 42R, 57
bfi2_trust <- list(
  dimension = "bfi2_trust",
  recode_func = compute_agreement_rate,
  stimulus_ids = paste0("bfi-2_q_", c(12, 27, 42, 57)),
  reverse_ids = paste0("bfi-2_q_", c(27, 57))
)

## Organization: 3R, 18, 33, 48R
bfi2_organization <- list(
  dimension = "bfi2_organization",
  recode_func = compute_agreement_rate,
  stimulus_ids = paste0("bfi-2_q_", c(3, 18, 33, 48)),
  reverse_ids = paste0("bfi-2_q_", c(3, 48))
)

## Productiveness: 8R, 23R, 38, 53
bfi2_productiveness <- list(
  dimension = "bfi2_productiveness",
  recode_func = compute_agreement_rate,
  stimulus_ids = paste0("bfi-2_q_", c(8, 23, 38, 53)),
  reverse_ids = paste0("bfi-2_q_", c(8, 23))
)

## Responsibility: 13, 28R, 43, 58R
bfi2_responsibility <- list(
  dimension = "bfi2_responsibility",
  recode_func = compute_agreement_rate,
  stimulus_ids = paste0("bfi-2_q_", c(13, 28, 43, 58)),
  reverse_ids = paste0("bfi-2_q_", c(28, 58))
)

```

```

)

## Anxiety: 4R, 19, 34, 49R
bfi2_anxiety <- list(
  dimension = "bfi2_anxiety",
  recode_func = compute_agreement_rate,
  stimulus_ids = paste0("bfi-2_q_", c(4, 19, 34, 49)),
  reverse_ids = paste0("bfi-2_q_", c(4, 49))
)

## Depression: 9R, 24R, 39, 54
bfi2_depression <- list(
  dimension = "bfi2_depression",
  recode_func = compute_agreement_rate,
  stimulus_ids = paste0("bfi-2_q_", c(9, 24, 39, 54)),
  reverse_ids = paste0("bfi-2_q_", c(9, 24))
)

## Emotional Volatility: 14, 29R, 44R, 59
bfi2_volatility <- list(
  dimension = "bfi2_volatility",
  recode_func = compute_agreement_rate,
  stimulus_ids = paste0("bfi-2_q_", c(14, 29, 44, 59)),
  reverse_ids = paste0("bfi-2_q_", c(29, 44))
)

## Intellectual Curiosity: 10, 25R, 40, 55R
bfi2_curiosity <- list(
  dimension = "bfi2_curiosity",
  recode_func = compute_agreement_rate,
  stimulus_ids = paste0("bfi-2_q_", c(10, 25, 40, 55)),
  reverse_ids = paste0("bfi-2_q_", c(25, 55))
)

## Aesthetic Sensitivity: 5R, 20, 35, 50R
bfi2_aesthetic <- list(
  dimension = "bfi2_aesthetic",
  recode_func = compute_agreement_rate,
  stimulus_ids = paste0("bfi-2_q_", c(5, 20, 35, 50)),
  reverse_ids = paste0("bfi-2_q_", c(5, 50))
)

## Creative Imagination: 15, 30R, 45R, 60
bfi2_imagination <- list(
  dimension = "bfi2_imagination",

```

```

    recode_func = compute_agreement_rate,
    stimulus_ids = paste0("bfi-2_q_", c(15, 30, 45, 60)),
    reverse_ids = paste0("bfi-2_q_", c(30, 45))
)

```

Note that the above is a sort of documentation of how to score the bfi2 questionnaire; this content should be an integral part of the questionnaire (and questionnaire data collection system) itself rather than being manually entered as we did here.

Once we have defined the list of scores we want to compute on our data, we can create a list of those parameter lists:

```

## We can group all of those scale parameter lists into an instrument specific list
bfi2_params <- list(
  # 5 domain scales:
  bfi2_extraversion,
  bfi2_agreeableness,
  bfi2_conscientiousness,
  bfi2_negative,
  bfi2_open,
  # 15 facets:
  bfi2_sociability,
  bfi2_assertiveness,
  bfi2_energy,
  bfi2_compassion,
  bfi2_respectfulness,
  bfi2_trust,
  bfi2_organization,
  bfi2_productiveness,
  bfi2_responsibility,
  bfi2_anxiety,
  bfi2_depression,
  bfi2_volatility,
  bfi2_curiosity,
  bfi2_aesthetic,
  bfi2_imagination)

```

The next step is then to execute the scoring code on each of them:

```

# we use purrr instead of for loops
# we also first create a wrapper to simplify code
f <- function(df, param_list){
  df |>
    group_by(agent_id, iteration) |>
    score_questionnaire(param_list)
}

```

```
bfi2_scores <- map_dfr(bfi2_params, ~ f(D_tidy, .x))
```

Thus if we exclude the generic code and the specification of how to score the bfi2 questionnaire, the code to score the questionnaire only takes a handful of lines of code, which are the same irrespective of the number of scores to extract from the questionnaire or from the number of iterations that participants completed.

Some people might want at this stage to have a table showing for each person their score on the various dimensions of the questionnaire. We could simply use pivot\_wider() here to create a one-person-per-row table. However, it is unlikely that we would want to display all of the data in bfi2\_scores. As an example, we will only display the scores (i.e., ignoring response\_time, n and na\_n) and average the scores across repetitions when there are multiple.

```
bfi2_scores_wide <- bfi2_scores |>  
  # averaging scores across repetitions for each dimension  
  group_by(agent_id, dimension) |>  
  summarize(score = mean(score, na.rm=TRUE), .groups = "drop") |>  
  # reshaping the table  
  pivot_wider(id_cols = agent_id, names_from = dimension, values_from = score)  
  
knitr::kable(bfi2_scores_wide)
```

age	bfi2_h2n	bfi2_h2r	bfi2_h2l	bfi2_f2n	bfi2_f2r	bfi2_f2l	bfi2_g2n	bfi2_g2r	bfi2_g2l	bfi2_p2n	bfi2_p2r	bfi2_p2l	bfi2_l2n	bfi2_l2r	bfi2_l2l	bfi2_s2n	bfi2_s2r	bfi2_s2l	bfi2_t2n	bfi2_t2r	bfi2_t2l	bfi2_u2n	bfi2_u2r	bfi2_u2l	bfi2_v2n	bfi2_v2r	bfi2_v2l	bfi2_w2n	bfi2_w2r	bfi2_w2l	bfi2_x2n	bfi2_x2r	bfi2_x2l	bfi2_y2n	bfi2_y2r	bfi2_y2l	bfi2_z2n	bfi2_z2r	bfi2_z2l	bfi2_h3n	bfi2_h3r	bfi2_h3l	bfi2_f3n	bfi2_f3r	bfi2_f3l	bfi2_g3n	bfi2_g3r	bfi2_g3l	bfi2_p3n	bfi2_p3r	bfi2_p3l	bfi2_l3n	bfi2_l3r	bfi2_l3l	bfi2_s3n	bfi2_s3r	bfi2_s3l	bfi2_t3n	bfi2_t3r	bfi2_t3l	bfi2_u3n	bfi2_u3r	bfi2_u3l	bfi2_v3n	bfi2_v3r	bfi2_v3l	bfi2_w3n	bfi2_w3r	bfi2_w3l	bfi2_x3n	bfi2_x3r	bfi2_x3l	bfi2_y3n	bfi2_y3r	bfi2_y3l	bfi2_z3n	bfi2_z3r	bfi2_z3l	bfi2_h4n	bfi2_h4r	bfi2_h4l	bfi2_f4n	bfi2_f4r	bfi2_f4l	bfi2_g4n	bfi2_g4r	bfi2_g4l	bfi2_p4n	bfi2_p4r	bfi2_p4l	bfi2_l4n	bfi2_l4r	bfi2_l4l	bfi2_s4n	bfi2_s4r	bfi2_s4l	bfi2_t4n	bfi2_t4r	bfi2_t4l	bfi2_u4n	bfi2_u4r	bfi2_u4l	bfi2_v4n	bfi2_v4r	bfi2_v4l	bfi2_w4n	bfi2_w4r	bfi2_w4l	bfi2_x4n	bfi2_x4r	bfi2_x4l	bfi2_y4n	bfi2_y4r	bfi2_y4l	bfi2_z4n	bfi2_z4r	bfi2_z4l	bfi2_h5n	bfi2_h5r	bfi2_h5l	bfi2_f5n	bfi2_f5r	bfi2_f5l	bfi2_g5n	bfi2_g5r	bfi2_g5l	bfi2_p5n	bfi2_p5r	bfi2_p5l	bfi2_l5n	bfi2_l5r	bfi2_l5l	bfi2_s5n	bfi2_s5r	bfi2_s5l	bfi2_t5n	bfi2_t5r	bfi2_t5l	bfi2_u5n	bfi2_u5r	bfi2_u5l	bfi2_v5n	bfi2_v5r	bfi2_v5l	bfi2_w5n	bfi2_w5r	bfi2_w5l	bfi2_x5n	bfi2_x5r	bfi2_x5l	bfi2_y5n	bfi2_y5r	bfi2_y5l	bfi2_z5n	bfi2_z5r	bfi2_z5l	bfi2_h6n	bfi2_h6r	bfi2_h6l	bfi2_f6n	bfi2_f6r	bfi2_f6l	bfi2_g6n	bfi2_g6r	bfi2_g6l	bfi2_p6n	bfi2_p6r	bfi2_p6l	bfi2_l6n	bfi2_l6r	bfi2_l6l	bfi2_s6n	bfi2_s6r	bfi2_s6l	bfi2_t6n	bfi2_t6r	bfi2_t6l	bfi2_u6n	bfi2_u6r	bfi2_u6l	bfi2_v6n	bfi2_v6r	bfi2_v6l	bfi2_w6n	bfi2_w6r	bfi2_w6l	bfi2_x6n	bfi2_x6r	bfi2_x6l	bfi2_y6n	bfi2_y6r	bfi2_y6l	bfi2_z6n	bfi2_z6r	bfi2_z6l	bfi2_h7n	bfi2_h7r	bfi2_h7l	bfi2_f7n	bfi2_f7r	bfi2_f7l	bfi2_g7n	bfi2_g7r	bfi2_g7l	bfi2_p7n	bfi2_p7r	bfi2_p7l	bfi2_l7n	bfi2_l7r	bfi2_l7l	bfi2_s7n	bfi2_s7r	bfi2_s7l	bfi2_t7n	bfi2_t7r	bfi2_t7l	bfi2_u7n	bfi2_u7r	bfi2_u7l	bfi2_v7n	bfi2_v7r	bfi2_v7l	bfi2_w7n	bfi2_w7r	bfi2_w7l	bfi2_x7n	bfi2_x7r	bfi2_x7l	bfi2_y7n	bfi2_y7r	bfi2_y7l	bfi2_z7n	bfi2_z7r	bfi2_z7l	bfi2_h8n	bfi2_h8r	bfi2_h8l	bfi2_f8n	bfi2_f8r	bfi2_f8l	bfi2_g8n	bfi2_g8r	bfi2_g8l	bfi2_p8n	bfi2_p8r	bfi2_p8l	bfi2_l8n	bfi2_l8r	bfi2_l8l	bfi2_s8n	bfi2_s8r	bfi2_s8l	bfi2_t8n	bfi2_t8r	bfi2_t8l	bfi2_u8n	bfi2_u8r	bfi2_u8l	bfi2_v8n	bfi2_v8r	bfi2_v8l	bfi2_w8n	bfi2_w8r	bfi2_w8l	bfi2_x8n	bfi2_x8r	bfi2_x8l	bfi2_y8n	bfi2_y8r	bfi2_y8l	bfi2_z8n	bfi2_z8r	bfi2_z8l	bfi2_h9n	bfi2_h9r	bfi2_h9l	bfi2_f9n	bfi2_f9r	bfi2_f9l	bfi2_g9n	bfi2_g9r	bfi2_g9l	bfi2_p9n	bfi2_p9r	bfi2_p9l	bfi2_l9n	bfi2_l9r	bfi2_l9l	bfi2_s9n	bfi2_s9r	bfi2_s9l	bfi2_t9n	bfi2_t9r	bfi2_t9l	bfi2_u9n	bfi2_u9r	bfi2_u9l	bfi2_v9n	bfi2_v9r	bfi2_v9l	bfi2_w9n	bfi2_w9r	bfi2_w9l	bfi2_x9n	bfi2_x9r	bfi2_x9l	bfi2_y9n	bfi2_y9r	bfi2_y9l	bfi2_z9n	bfi2_z9r	bfi2_z9l	bfi2_h10n	bfi2_h10r	bfi2_h10l	bfi2_f10n	bfi2_f10r	bfi2_f10l	bfi2_g10n	bfi2_g10r	bfi2_g10l	bfi2_p10n	bfi2_p10r	bfi2_p10l	bfi2_l10n	bfi2_l10r	bfi2_l10l	bfi2_s10n	bfi2_s10r	bfi2_s10l	bfi2_t10n	bfi2_t10r	bfi2_t10l	bfi2_u10n	bfi2_u10r	bfi2_u10l	bfi2_v10n	bfi2_v10r	bfi2_v10l	bfi2_w10n	bfi2_w10r	bfi2_w10l	bfi2_x10n	bfi2_x10r	bfi2_x10l	bfi2_y10n	bfi2_y10r	bfi2_y10l	bfi2_z10n	bfi2_z10r	bfi2_z10l	bfi2_h11n	bfi2_h11r	bfi2_h11l	bfi2_f11n	bfi2_f11r	bfi2_f11l	bfi2_g11n	bfi2_g11r	bfi2_g11l	bfi2_p11n	bfi2_p11r	bfi2_p11l	bfi2_l11n	bfi2_l11r	bfi2_l11l	bfi2_s11n	bfi2_s11r	bfi2_s11l	bfi2_t11n	bfi2_t11r	bfi2_t11l	bfi2_u11n	bfi2_u11r	bfi2_u11l	bfi2_v11n	bfi2_v11r	bfi2_v11l	bfi2_w11n	bfi2_w11r	bfi2_w11l	bfi2_x11n	bfi2_x11r	bfi2_x11l	bfi2_y11n	bfi2_y11r	bfi2_y11l	bfi2_z11n	bfi2_z11r	bfi2_z11l	bfi2_h12n	bfi2_h12r	bfi2_h12l	bfi2_f12n	bfi2_f12r	bfi2_f12l	bfi2_g12n	bfi2_g12r	bfi2_g12l	bfi2_p12n	bfi2_p12r	bfi2_p12l	bfi2_l12n	bfi2_l12r	bfi2_l12l	bfi2_s12n	bfi2_s12r	bfi2_s12l	bfi2_t12n	bfi2_t12r	bfi2_t12l	bfi2_u12n	bfi2_u12r	bfi2_u12l	bfi2_v12n	bfi2_v12r	bfi2_v12l	bfi2_w12n	bfi2_w12r	bfi2_w12l	bfi2_x12n	bfi2_x12r	bfi2_x12l	bfi2_y12n	bfi2_y12r	bfi2_y12l	bfi2_z12n	bfi2_z12r	bfi2_z12l	bfi2_h13n	bfi2_h13r	bfi2_h13l	bfi2_f13n	bfi2_f13r	bfi2_f13l	bfi2_g13n	bfi2_g13r	bfi2_g13l	bfi2_p13n	bfi2_p13r	bfi2_p13l	bfi2_l13n	bfi2_l13r	bfi2_l13l	bfi2_s13n	bfi2_s13r	bfi2_s13l	bfi2_t13n	bfi2_t13r	bfi2_t13l	bfi2_u13n	bfi2_u13r	bfi2_u13l	bfi2_v13n	bfi2_v13r	bfi2_v13l	bfi2_w13n	bfi2_w13r	bfi2_w13l	bfi2_x13n	bfi2_x13r	bfi2_x13l	bfi2_y13n	bfi2_y13r	bfi2_y13l	bfi2_z13n	bfi2_z13r	bfi2_z13l	bfi2_h14n	bfi2_h14r	bfi2_h14l	bfi2_f14n	bfi2_f14r	bfi2_f14l	bfi2_g14n	bfi2_g14r	bfi2_g14l	bfi2_p14n	bfi2_p14r	bfi2_p14l	bfi2_l14n	bfi2_l14r	bfi2_l14l	bfi2_s14n	bfi2_s14r	bfi2_s14l	bfi2_t14n	bfi2_t14r	bfi2_t14l	bfi2_u14n	bfi2_u14r	bfi2_u14l	bfi2_v14n	bfi2_v14r	bfi2_v14l	bfi2_w14n	bfi2_w14r	bfi2_w14l	bfi2_x14n	bfi2_x14r	bfi2_x14l	bfi2_y14n	bfi2_y14r	bfi2_y14l	bfi2_z14n	bfi2_z14r	bfi2_z14l	bfi2_h15n	bfi2_h15r	bfi2_h15l	bfi2_f15n	bfi2_f15r	bfi2_f15l	bfi2_g15n	bfi2_g15r	bfi2_g15l	bfi2_p15n	bfi2_p15r	bfi2_p15l	bfi2_l15n	bfi2_l15r	bfi2_l15l	bfi2_s15n	bfi2_s15r	bfi2_s15l	bfi2_t15n	bfi2_t15r	bfi2_t15l	bfi2_u15n	bfi2_u15r	bfi2_u15l	bfi2_v15n	bfi2_v15r	bfi2_v15l	bfi2_w15n	bfi2_w15r	bfi2_w15l	bfi2_x15n	bfi2_x15r	bfi2_x15l	bfi2_y15n	bfi2_y15r	bfi2_y15l	bfi2_z15n	bfi2_z15r	bfi2_z15l	bfi2_h16n	bfi2_h16r	bfi2_h16l	bfi2_f16n	bfi2_f16r	bfi2_f16l	bfi2_g16n	bfi2_g16r	bfi2_g16l	bfi2_p16n	bfi2_p16r	bfi2_p16l	bfi2_l16n	bfi2_l16r	bfi2_l16l	bfi2_s16n	bfi2_s16r	bfi2_s16l	bfi2_t16n	bfi2_t16r	bfi2_t16l	bfi2_u16n	bfi2_u16r	bfi2_u16l	bfi2_v16n	bfi2_v16r	bfi2_v16l	bfi2_w16n	bfi2_w16r	bfi2_w16l	bfi2_x16n	bfi2_x16r	bfi2_x16l	bfi2_y16n	bfi2_y16r	bfi2_y16l	bfi2_z16n	bfi2_z16r	bfi2_z16l	bfi2_h17n	bfi2_h17r	bfi2_h17l	bfi2_f17n	bfi2_f17r	bfi2_f17l	bfi2_g17n	bfi2_g17r	bfi2_g17l	bfi2_p17n	bfi2_p17r	bfi2_p17l	bfi2_l17n	bfi2_l17r	bfi2_l17l	bfi2_s17n	bfi2_s17r	bfi2_s17l	bfi2_t17n	bfi2_t17r	bfi2_t17l	bfi2_u17n	bfi2_u17r	bfi2_u17l	bfi2_v17n	bfi2_v17r	bfi2_v17l	bfi2_w17n	bfi2_w17r	bfi2_w17l	bfi2_x17n	bfi2_x17r	bfi2_x17l	bfi2_y17n	bfi2_y17r	bfi2_y17l	bfi2_z17n	bfi2_z17r	bfi2_z17l	bfi2_h18n	bfi2_h18r	bfi2_h18l	bfi2_f18n	bfi2_f18r	bfi2_f18l	bfi2_g18n	bfi2_g18r	bfi2_g18l	bfi2_p18n	bfi2_p18r	bfi2_p18l	bfi2_l18n	bfi2_l18r	bfi2_l18l	bfi2_s18n	bfi2_s18r	bfi2_s18l	bfi2_t18n	bfi2_t18r	bfi2_t18l	bfi2_u18n	bfi2_u18r	bfi2_u18l	bfi2_v18n	bfi2_v18r	bfi2_v18l	bfi2_w18n	bfi2_w18r	bfi2_w18l	bfi2_x18n	bfi2_x18r	bfi2_x18l	bfi2_y18n	bfi2_y18r	bfi2_y18l	bfi2_z18n	bfi2_z18r	bfi2_z18l	bfi2_h19n	bfi2_h19r	bfi2_h19l	bfi2_f19n	bfi2_f19r	bfi2_f19l	bfi2_g19n	bfi2_g19r	bfi2_g19l	bfi2_p19n	bfi2_p19r	bfi2_p19l	bfi2_l19n	bfi2_l19r	bfi2_l19l	bfi2_s19n	bfi2_s19r	bfi2_s19l	bfi2_t19n	bfi2_t19r	bfi2_t19l	bfi2_u19n	bfi2_u19r	bfi2_u19l	bfi2_v19n	bfi2_v19r	bfi2_v19l	bfi2_w19n	bfi2_w19r	bfi2_w19l	bfi2_x19n	bfi2_x19r	bfi2_x19l	bfi2_y19n	bfi2_y19r	bfi2_y19l	bfi2_z19n	bfi2_z19r	bfi2_z19l	bfi2_h20n	bfi2_h20r	bfi2_h20l	bfi2_f20n	bfi2_f20r	bfi2_f20l	bfi2_g20n	bfi2_g20r	bfi2_g20l	bfi2_p20n	bfi2_p20r	bfi2_p20l	bfi2_l20n	bfi2_l20r	bfi2_l20l	bfi2_s20n	bfi2_s20r	bfi2_s20l	bfi2_t20n	bfi2_t20r	bfi2_t20l	bfi2_u20n	bfi2_u20r	bfi2_u20l	bfi2_v20n	bfi2_v20r	bfi2_v20l	bfi2_w20n	bfi2_w20r	bfi2_w20l	bfi2_x20n	bfi2_x20r	bfi2_x20l	bfi2_y20n	bfi2_y20r	bfi2_y20l	bfi2_z20n	bfi2_z20r	bfi2_z20l	bfi2_h21n	bfi2_h21r	bfi2_h21l	bfi2_f21n	bfi2_f21r	bfi2_f21l	bfi2_g21n	bfi2_g21r	bfi2_g21l	bfi2_p21n	bfi2_p21r	bfi2_p21l	bfi2_l21n	bfi2_l21r	bfi2_l21l	bfi2_s21n	bfi2_s21r	bfi2_s21l	bfi2_t21n	bfi2_t21r	bfi2_t21l	bfi2_u21n	bfi2_u21r	bfi2_u21l	bfi2_v21n	bfi2_v21r	bfi2_v21l	bfi2_w21n	bfi2_w21r	bfi2_w21l	bfi2_x21n	bfi2_x21r	bfi2_x21l	bfi2_y21n	bfi2_y21r	bfi2_y21l	bfi2_z21n	bfi2_z21r	bfi2_z21l	bfi2_h22n	bfi2_h22r	bfi2_h22l	bfi2_f22n	bfi2_f22r	bfi2_f22l	bfi2_g22n	bfi2_g22r	bfi2_g22l	bfi2_p22n	bfi2_p22r	bfi2_p22l	bfi2_l22n	bfi2_l22r	bfi2_l22l	bfi2_s22n	bfi2_s22r	bfi2_s22l	bfi2_t22n	bfi2_t22r	bfi2_t22l	bfi2_u22n	bfi2_u22r	bfi2_u22l	bfi2_v22n	bfi2_v22r	bfi2_v22l	bfi2_w22n	bfi2_w22r	bfi2_w22l	bfi2_x22n	bfi2_x22r	bfi2_x22l	bfi2_y22n	bfi2_y22r	bfi2_y22l	bfi2_z22n	bfi2_z22r	bfi2_z22l	bfi2_h23n	bfi2_h23r	bfi2_h23l	bfi2_f23n	bfi2_f23r	bfi2_f23l	bfi2_g23n	bfi2_g23r	bfi2_g23l	bfi2_p23n	bfi2_p23r	bfi2_p23l	bfi2_l23n	bfi2_l23r	bfi2_l23l	bfi2_s23n	bfi2_s23r	bfi2_s23l	bfi2_t23n	bfi2_t23r	bfi2_t23l	bfi2_u23n	bfi2_u23r	bfi2_u23l	bfi2_v23n	bfi2_v23r	bfi2_v23l	bfi2_w23n	bfi2_w23r	bfi2_w23l	bfi2_x23n	bfi2_x23r	bfi2_x23l	bfi2_y23n	bfi2_y23r	bfi2_y23l	bfi2_z23n	bfi2_z23r	bfi2_z23l	bfi2_h24n	bfi2_h24r	bfi2_h24l	bfi2_f24n	bfi2_f24r	bfi2_f24l	bfi2_g24n	bfi2_g24r	bfi2_g24l	bfi2_p24n	bfi2_p24r	bfi2_p24l	bfi2_l24n	bfi2_l24r	bfi2_l24l	bfi2_s24n	bfi2_s24r	bfi2_s24l	bfi2_t24n	bfi2_t24r	bfi2_t24l	bfi2_u24n	bfi2_u24r	bfi2_u24l	bfi2_v24n	bfi2_v24r	bfi2_v24l	bfi2_w24n	bfi2_w24r	bfi2_w24l	bfi2_x24n	bfi2_x24r	bfi2_x24l	bfi2_y24n	bfi2_y24r	bfi2_y24l	bfi2_z24n	bfi2_z24r	bfi2_z24l	bfi2_h25n	bfi2_h25r	bfi2_h25l	bfi2_f25n	bfi2_f25r	bfi2_f25l	bfi2_g25n	bfi2_g25r	bfi2_g25l	bfi2_p25n	bfi2_p25r	bfi2_p25l	bfi2_l25n	bfi2_l25r	bfi2_l25l	bfi2_s25n	bfi2_s25r	bfi2_s25l	bfi2_t25n	bfi2_t25r	bfi2_t25l	bfi2_u25n	bfi2_u25r	bfi2_u25l	bfi2_v25n	bfi2_v25r	bfi2_v25l	bfi2_w25n	bfi2_w25r	bfi2_w25l	bfi2_x25n	bfi2_x25r	bfi2_x25l	bfi2_y25n	bfi2_y25r	bfi2_y25l	bfi2_z25n	bfi2_z25r	bfi2_z25l	bfi2_h26n	bfi2_h26r	bfi2_h26l	bfi2_f26n	bfi2_f26r	bfi2_f26l	bfi2_g26n	bfi2_g26r	bfi2_g26l	bfi2_p26n	bfi2_p26r	bfi2_p26l	bfi2_l26n	bfi2_l26r	bfi2_l26l	bfi2_s26n	bfi2_s26r	bfi2_s26l	bfi2_t26n	bfi2_t26r	bfi2_t26l	bfi2_u26n	bfi2_u26r	bfi2_u26l	bfi2_v26n	bfi2_v26r	bfi2_v26l	bfi2_w26n	bfi2_w26r	bfi2_w26l	bfi2_x26n	bfi2_x26r	bfi2_x26l	bfi2_y26n	bfi2_y26r	bfi2_y26l	bfi2_z26n	bfi2_z26r	bfi2_z26l	bfi2_h27n	bfi2_h27r	bfi2_h27l	bfi2_f27n	bfi2_f27r	bfi2_f27l	bfi2_g27n	bfi2_g27r	bfi2_g27l	bfi2_p27n	bfi2_p27r	bfi2_p27l	bfi2_l27n	bfi2_l27r	bfi2_l27l	bfi2_s27n	bfi2_s27r	bfi2_s27l	bfi2_t27n	bfi2_t27r	bfi2_t27l	bfi2_u27n	bfi2_u27r	bfi2_u27l	bfi2_v27n	bfi2_v27r	bfi2_v27l	bfi2_w27n	bfi2_w27r	bfi2_w27l	bfi2_x27n	bfi2_x27r	bfi2_x27l	bfi2_y27n	bfi2_y27r	bfi2_y27l	bfi2_z27n	bfi2_z27r	bfi2_z27l	bfi2_h28n	bfi2_h28r	bfi2_h28l	bfi2_f28n	bfi2_f28r	bfi2_f28l	bfi2_g28n	bfi2_g28r
-----	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------

```

validate_psqi_question_2_input <- function(x){
  if (!is.numeric(x)) {
    stop("Input must be numeric")
  } else if (any(x < 0, na.rm = TRUE)){
    stop("Input must be larger than 0")
  }
  return(x)
}

# validate_psqi_question_2_input(c(1, 1, 2, NA))

score_psqi_question_2 <- function(x){
  validate_psqi_question_2_input(x)
  as.numeric(cut(x, breaks = c(0,15, 30, 60, 24*60))) - 1
}

# score question 5:
validate_psqi_question_5_input <- function(x){

  q5_levels = c("not during the past month", "less than once a week", "once or twice a week",
               "more than twice a week", "almost daily", "daily")

  if (is.factor(x)) { # if x is encoded as a factor variable
    if (! all(levels(x) == q5_levels)){
      stop(paste0("Input does not have the required levels: \nInput levels: ", levels(x), "\nRequired levels: ", q5_levels))
    }
  } else if (!is.character(x)){
    stop("Input must be either factor or string")
  } else {
    # check inputs are
    for (element in x){
      if (is.na(element)) next
      if (!element %in% q5_levels){
        stop(paste0("Input contains unexpected values: ", element) )
      }
    }
    return(x)
  }
}

```

```

score_psqi_question_5 <- function(x){
  validate_psqi_question_5_input(x)
  as.numeric(
    factor(x, levels = c("not during the past month", "less than once a week", "once or twice a month", "more than once a month"))
  )-1
}

# combine the subcomponent scores
score_sleep_latency_components <- function(x,y){
  as.numeric(
    cut(x + y, breaks = c(-1, 0, 2, 4, 6))
  )-1
}

# wrapper to do all the previous steps in one function call
score_sleep_latency <- function(q2,q5){
  x <- score_psqi_question_2(q2)
  y <- score_psqi_question_5(q5)
  score_sleep_latency_components(x,y)
}

```

The code above shows that one can write simple functions to process each question separately, a function to aggregate those results and a wrapper function to keep the data analysis code cleaner. It is important to note here that this code expects very specific input values: in the first case a number of minutes (i.e., a positive number between 0 and 1440), in the second, one out of 4 possible response categories. In the above code we show how the data analysis code can make sure that the input is as expected (as opposed to relying on the data being correctly formatted).

Having created those functions, we can now analyze our specific dataset:

```

# use the above code to compute the psqi sleep latency score for each iteration and append to the dataset
D_wide <- D_wide |>
  mutate(psqi_latency_score_i1 = score_sleep_latency(
    psqi_q_2_response_numeric_i1,
    `frequency_trouble_sleep_because;psqi-past_q_6_response`_i1
  ),
  psqi_latency_score_i2 = score_sleep_latency(
    psqi_q_2_response_numeric_i2,
    `frequency_trouble_sleep_because;psqi-past_q_6_response`_i2
  )
)

```

```
)
```

The above code is rather compact thanks to the use of functions. However, as in with the bfi2\_wide example, we end up having to write code that is specific to our study: we need to compute the scores separately for each iteration, which leads to duplicated code and the potential to make mistakes (forgetting to change the i1 to i2 when copy-pasting code).

The second observation: we add new columns to the main table; we are now mixing in the table things that are data (responses to questions) and things that we compute using that data (the psqi sleep latency score). We could of course save the resulting columns in a separate table.

```
D_wide |>  
  select(agent_id, psqi_latency_score_i1, psqi_latency_score_i2) |>  
  knitr::kable()
```

agent_id	psqi_latency_score_i1	psqi_latency_score_i2
002	NA	NA
019	NA	NA
030	1	1
008	2	3
012	2	1
026	3	2
031	1	0

### The tidy data format

We could use an approach similar to what we did for BFI-2: define a list of parameters to compute a specific score and a generic function to run the analysis of data. This would work for calculating the subscores, but then a second step would be required to compute the aggregate score from those two subscores.

The alternative is to create a function that takes as input a data frame and returns the scores

```
score_sleep_latency_components_df <- function(x){  
  
  if (length(x) != 2){  
    stop("Data does not have the expected number of items")  
  } else {  
    score_sleep_latency_components(x[1], x[2])  
  }  
  
}
```

```

score_psqi_latency <- function(df){

  # because rowwise will override any existing groups, we need to save and restore them when
  group_vars <- dplyr::group_vars(df)

  df |>
    filter(stimulus_id %in% c("psqi_q_2", "frequency_trouble_sleep_because;psqi-past_q_6"))
    # process each question separately
    rowwise() |>
    mutate(score_tmp = ifelse(stimulus_id=="psqi_q_2",
                               score_psqi_question_2(response_numeric),
                               NA)) |>
    mutate(score_tmp = ifelse(stimulus_id=="frequency_trouble_sleep_because;psqi-past_q_6",
                               score_psqi_question_5(response_description),
                               score_tmp)) |>
    ungroup() |>
    group_by(across(all_of(group_vars))) |>

    # aggregate scores across all selected responses
    summarize(dimension = "psqi_latency",
              score = score_sleep_latency_components_df(score_tmp),
              response_time = mean(response_time, na.rm = TRUE),
              n = n(),
              na_n = sum(is.na(response_time)),
              .groups = "drop"
              )

}

D_tidy |>
  group_by(agent_id, iteration) |>
  score_psqi_latency()

# A tibble: 10 x 7
  agent_id iteration dimension     score response_time     n   na_n
  <chr>      <dbl> <chr>     <dbl>        <dbl> <int> <int>
1 008          1 psqi_latency     2       5.73      2     0
2 008          2 psqi_latency     3       7.95      2     0
3 012          1 psqi_latency     2      13.2       2     0
4 012          2 psqi_latency     1       5.34      2     0
5 026          1 psqi_latency     3      10.5       2     0
6 026          2 psqi_latency     2       5.38      2     0
7 030          1 psqi_latency     1       3.69      2     0
8 030          2 psqi_latency     1      10.0       2     0

```

9	031	1	psqi_latency	1	23.5	2	0
10	031	2	psqi_latency	0	15.8	2	0

The code in this case is quite a bit more complex; the main advantages are however that a) we can compute more metrics on the selected items without much effort (e.g., response\_time) compared to the strategy we used on the wide data format and b) we can have studies that involve any number of iterations (or other types of grouping variables) and reuse the exact same code; in contrast, the code used on the wide data format is strongly tied to the specifics of the study, be it through the study parameters encoded in the column names or the number of steps required in the analysis.

## Conclusion

This notebook presents a more realistic, although still simplified, example of the analysis of questionnaire data using the wide and tidy data formats respectively. Overall, the tidy data format seems preferable for several reasons. However, we also note that in some cases handling the tidy data format may require code that is more complex than the wide data format.