

# 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 items 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)],
}

# 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, rev),  
                                compute_agreement_rate_vector(response_option_index, rev))  
  # 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 questionnaire
# 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$recode_func(pick(everything()), reverse = TRUE),
                                   scoring_params$recode_func(pick(everything()), reverse = FALSE)),
           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)
```

agent_id	bfi2_h1	bfi2_h2	bfi2_h3	bfi2_h4	bfi2_h5	bfi2_h6	bfi2_h7	bfi2_h8	bfi2_h9	bfi2_h10	bfi2_h11	bfi2_h12	bfi2_h13	bfi2_h14	bfi2_h15	bfi2_h16	bfi2_h17	bfi2_h18	bfi2_h19	bfi2_h20	bfi2_h21	bfi2_h22	bfi2_h23	bfi2_h24	bfi2_h25	bfi2_h26	bfi2_h27	bfi2_h28	bfi2_h29	bfi2_h30	bfi2_h31	bfi2_h32	bfi2_h33	bfi2_h34	bfi2_h35	bfi2_h36	bfi2_h37	bfi2_h38	bfi2_h39	bfi2_h40	bfi2_h41	bfi2_h42	bfi2_h43	bfi2_h44	bfi2_h45	bfi2_h46	bfi2_h47	bfi2_h48	bfi2_h49	bfi2_h50	bfi2_h51	bfi2_h52	bfi2_h53	bfi2_h54	bfi2_h55	bfi2_h56	bfi2_h57	bfi2_h58	bfi2_h59	bfi2_h60	bfi2_h61	bfi2_h62	bfi2_h63	bfi2_h64	bfi2_h65	bfi2_h66	bfi2_h67	bfi2_h68	bfi2_h69	bfi2_h70	bfi2_h71	bfi2_h72	bfi2_h73	bfi2_h74	bfi2_h75	bfi2_h76	bfi2_h77	bfi2_h78	bfi2_h79	bfi2_h80	bfi2_h81	bfi2_h82	bfi2_h83	bfi2_h84	bfi2_h85	bfi2_h86	bfi2_h87	bfi2_h88	bfi2_h89	bfi2_h90	bfi2_h91	bfi2_h92	bfi2_h93	bfi2_h94	bfi2_h95	bfi2_h96	bfi2_h97	bfi2_h98	bfi2_h99	bfi2_h100	bfi2_h101	bfi2_h102	bfi2_h103	bfi2_h104	bfi2_h105	bfi2_h106	bfi2_h107	bfi2_h108	bfi2_h109	bfi2_h110	bfi2_h111	bfi2_h112	bfi2_h113	bfi2_h114	bfi2_h115	bfi2_h116	bfi2_h117	bfi2_h118	bfi2_h119	bfi2_h120	bfi2_h121	bfi2_h122	bfi2_h123	bfi2_h124	bfi2_h125	bfi2_h126	bfi2_h127	bfi2_h128	bfi2_h129	bfi2_h130	bfi2_h131	bfi2_h132	bfi2_h133	bfi2_h134	bfi2_h135	bfi2_h136	bfi2_h137	bfi2_h138	bfi2_h139	bfi2_h140	bfi2_h141	bfi2_h142	bfi2_h143	bfi2_h144	bfi2_h145	bfi2_h146	bfi2_h147	bfi2_h148	bfi2_h149	bfi2_h150	bfi2_h151	bfi2_h152	bfi2_h153	bfi2_h154	bfi2_h155	bfi2_h156	bfi2_h157	bfi2_h158	bfi2_h159	bfi2_h160	bfi2_h161	bfi2_h162	bfi2_h163	bfi2_h164	bfi2_h165	bfi2_h166	bfi2_h167	bfi2_h168	bfi2_h169	bfi2_h170	bfi2_h171	bfi2_h172	bfi2_h173	bfi2_h174	bfi2_h175	bfi2_h176	bfi2_h177	bfi2_h178	bfi2_h179	bfi2_h180	bfi2_h181	bfi2_h182	bfi2_h183	bfi2_h184	bfi2_h185	bfi2_h186	bfi2_h187	bfi2_h188	bfi2_h189	bfi2_h190	bfi2_h191	bfi2_h192	bfi2_h193	bfi2_h194	bfi2_h195	bfi2_h196	bfi2_h197	bfi2_h198	bfi2_h199	bfi2_h200	bfi2_h201	bfi2_h202	bfi2_h203	bfi2_h204	bfi2_h205	bfi2_h206	bfi2_h207	bfi2_h208	bfi2_h209	bfi2_h210	bfi2_h211	bfi2_h212	bfi2_h213	bfi2_h214	bfi2_h215	bfi2_h216	bfi2_h217	bfi2_h218	bfi2_h219	bfi2_h220	bfi2_h221	bfi2_h222	bfi2_h223	bfi2_h224	bfi2_h225	bfi2_h226	bfi2_h227	bfi2_h228	bfi2_h229	bfi2_h230	bfi2_h231	bfi2_h232	bfi2_h233	bfi2_h234	bfi2_h235	bfi2_h236	bfi2_h237	bfi2_h238	bfi2_h239	bfi2_h240	bfi2_h241	bfi2_h242	bfi2_h243	bfi2_h244	bfi2_h245	bfi2_h246	bfi2_h247	bfi2_h248	bfi2_h249	bfi2_h250	bfi2_h251	bfi2_h252	bfi2_h253	bfi2_h254	bfi2_h255	bfi2_h256	bfi2_h257	bfi2_h258	bfi2_h259	bfi2_h260	bfi2_h261	bfi2_h262	bfi2_h263	bfi2_h264	bfi2_h265	bfi2_h266	bfi2_h267	bfi2_h268	bfi2_h269	bfi2_h270	bfi2_h271	bfi2_h272	bfi2_h273	bfi2_h274	bfi2_h275	bfi2_h276	bfi2_h277	bfi2_h278	bfi2_h279	bfi2_h280	bfi2_h281	bfi2_h282	bfi2_h283	bfi2_h284	bfi2_h285	bfi2_h286	bfi2_h287	bfi2_h288	bfi2_h289	bfi2_h290	bfi2_h291	bfi2_h292	bfi2_h293	bfi2_h294	bfi2_h295	bfi2_h296	bfi2_h297	bfi2_h298	bfi2_h299	bfi2_h300	bfi2_h301	bfi2_h302	bfi2_h303	bfi2_h304	bfi2_h305	bfi2_h306	bfi2_h307	bfi2_h308	bfi2_h309	bfi2_h310	bfi2_h311	bfi2_h312	bfi2_h313	bfi2_h314	bfi2_h315	bfi2_h316	bfi2_h317	bfi2_h318	bfi2_h319	bfi2_h320	bfi2_h321	bfi2_h322	bfi2_h323	bfi2_h324	bfi2_h325	bfi2_h326	bfi2_h327	bfi2_h328	bfi2_h329	bfi2_h330	bfi2_h331	bfi2_h332	bfi2_h333	bfi2_h334	bfi2_h335	bfi2_h336	bfi2_h337	bfi2_h338	bfi2_h339	bfi2_h340	bfi2_h341	bfi2_h342	bfi2_h343	bfi2_h344	bfi2_h345	bfi2_h346	bfi2_h347	bfi2_h348	bfi2_h349	bfi2_h350	bfi2_h351	bfi2_h352	bfi2_h353	bfi2_h354	bfi2_h355	bfi2_h356	bfi2_h357	bfi2_h358	bfi2_h359	bfi2_h360	bfi2_h361	bfi2_h362	bfi2_h363	bfi2_h364	bfi2_h365	bfi2_h366	bfi2_h367	bfi2_h368	bfi2_h369	bfi2_h370	bfi2_h371	bfi2_h372	bfi2_h373	bfi2_h374	bfi2_h375	bfi2_h376	bfi2_h377	bfi2_h378	bfi2_h379	bfi2_h380	bfi2_h381	bfi2_h382	bfi2_h383	bfi2_h384	bfi2_h385	bfi2_h386	bfi2_h387	bfi2_h388	bfi2_h389	bfi2_h390	bfi2_h391	bfi2_h392	bfi2_h393	bfi2_h394	bfi2_h395	bfi2_h396	bfi2_h397	bfi2_h398	bfi2_h399	bfi2_h400	bfi2_h401	bfi2_h402	bfi2_h403	bfi2_h404	bfi2_h405	bfi2_h406	bfi2_h407	bfi2_h408	bfi2_h409	bfi2_h410	bfi2_h411	bfi2_h412	bfi2_h413	bfi2_h414	bfi2_h415	bfi2_h416	bfi2_h417	bfi2_h418	bfi2_h419	bfi2_h420	bfi2_h421	bfi2_h422	bfi2_h423	bfi2_h424	bfi2_h425	bfi2_h426	bfi2_h427	bfi2_h428	bfi2_h429	bfi2_h430	bfi2_h431	bfi2_h432	bfi2_h433	bfi2_h434	bfi2_h435	bfi2_h436	bfi2_h437	bfi2_h438	bfi2_h439	bfi2_h440	bfi2_h441	bfi2_h442	bfi2_h443	bfi2_h444	bfi2_h445	bfi2_h446	bfi2_h447	bfi2_h448	bfi2_h449	bfi2_h450	bfi2_h451	bfi2_h452	bfi2_h453	bfi2_h454	bfi2_h455	bfi2_h456	bfi2_h457	bfi2_h458	bfi2_h459	bfi2_h460	bfi2_h461	bfi2_h462	bfi2_h463	bfi2_h464	bfi2_h465	bfi2_h466	bfi2_h467	bfi2_h468	bfi2_h469	bfi2_h470	bfi2_h471	bfi2_h472	bfi2_h473	bfi2_h474	bfi2_h475	bfi2_h476	bfi2_h477	bfi2_h478	bfi2_h479	bfi2_h480	bfi2_h481	bfi2_h482	bfi2_h483	bfi2_h484	bfi2_h485	bfi2_h486	bfi2_h487	bfi2_h488	bfi2_h489	bfi2_h490	bfi2_h491	bfi2_h492	bfi2_h493	bfi2_h494	bfi2_h495	bfi2_h496	bfi2_h497	bfi2_h498	bfi2_h499	bfi2_h500	bfi2_h501	bfi2_h502	bfi2_h503	bfi2_h504	bfi2_h505	bfi2_h506	bfi2_h507	bfi2_h508	bfi2_h509	bfi2_h510	bfi2_h511	bfi2_h512	bfi2_h513	bfi2_h514	bfi2_h515	bfi2_h516	bfi2_h517	bfi2_h518	bfi2_h519	bfi2_h520	bfi2_h521	bfi2_h522	bfi2_h523	bfi2_h524	bfi2_h525	bfi2_h526	bfi2_h527	bfi2_h528	bfi2_h529	bfi2_h530	bfi2_h531	bfi2_h532	bfi2_h533	bfi2_h534	bfi2_h535	bfi2_h536	bfi2_h537	bfi2_h538	bfi2_h539	bfi2_h540	bfi2_h541	bfi2_h542	bfi2_h543	bfi2_h544	bfi2_h545	bfi2_h546	bfi2_h547	bfi2_h548	bfi2_h549	bfi2_h550	bfi2_h551	bfi2_h552	bfi2_h553	bfi2_h554	bfi2_h555	bfi2_h556	bfi2_h557	bfi2_h558	bfi2_h559	bfi2_h560	bfi2_h561	bfi2_h562	bfi2_h563	bfi2_h564	bfi2_h565	bfi2_h566	bfi2_h567	bfi2_h568	bfi2_h569	bfi2_h570	bfi2_h571	bfi2_h572	bfi2_h573	bfi2_h574	bfi2_h575	bfi2_h576	bfi2_h577	bfi2_h578	bfi2_h579	bfi2_h580	bfi2_h581	bfi2_h582	bfi2_h583	bfi2_h584	bfi2_h585	bfi2_h586	bfi2_h587	bfi2_h588	bfi2_h589	bfi2_h590	bfi2_h591	bfi2_h592	bfi2_h593	bfi2_h594	bfi2_h595	bfi2_h596	bfi2_h597	bfi2_h598	bfi2_h599	bfi2_h600	bfi2_h601	bfi2_h602	bfi2_h603	bfi2_h604	bfi2_h605	bfi2_h606	bfi2_h607	bfi2_h608	bfi2_h609	bfi2_h610	bfi2_h611	bfi2_h612	bfi2_h613	bfi2_h614	bfi2_h615	bfi2_h616	bfi2_h617	bfi2_h618	bfi2_h619	bfi2_h620	bfi2_h621	bfi2_h622	bfi2_h623	bfi2_h624	bfi2_h625	bfi2_h626	bfi2_h627	bfi2_h628	bfi2_h629	bfi2_h630	bfi2_h631	bfi2_h632	bfi2_h633	bfi2_h634	bfi2_h635	bfi2_h636	bfi2_h637	bfi2_h638	bfi2_h639	bfi2_h640	bfi2_h641	bfi2_h642	bfi2_h643	bfi2_h644	bfi2_h645	bfi2_h646	bfi2_h647	bfi2_h648	bfi2_h649	bfi2_h650	bfi2_h651	bfi2_h652	bfi2_h653	bfi2_h654	bfi2_h655	bfi2_h656	bfi2_h657	bfi2_h658	bfi2_h659	bfi2_h660	bfi2_h661	bfi2_h662	bfi2_h663	bfi2_h664	bfi2_h665	bfi2_h666	bfi2_h667	bfi2_h668	bfi2_h669	bfi2_h670	bfi2_h671	bfi2_h672	bfi2_h673	bfi2_h674	bfi2_h675	bfi2_h676	bfi2_h677	bfi2_h678	bfi2_h679	bfi2_h680	bfi2_h681	bfi2_h682	bfi2_h683	bfi2_h684	bfi2_h685	bfi2_h686	bfi2_h687	bfi2_h688	bfi2_h689	bfi2_h690	bfi2_h691	bfi2_h692	bfi2_h693	bfi2_h694	bfi2_h695	bfi2_h696	bfi2_h697	bfi2_h698	bfi2_h699	bfi2_h700	bfi2_h701	bfi2_h702	bfi2_h703	bfi2_h704	bfi2_h705	bfi2_h706	bfi2_h707	bfi2_h708	bfi2_h709	bfi2_h710	bfi2_h711	bfi2_h712	bfi2_h713	bfi2_h714	bfi2_h715	bfi2_h716	bfi2_h717	bfi2_h718	bfi2_h719	bfi2_h720	bfi2_h721	bfi2_h722	bfi2_h723	bfi2_h724	bfi2_h725	bfi2_h726	bfi2_h727	bfi2_h728	bfi2_h729	bfi2_h730	bfi2_h731	bfi2_h732	bfi2_h733	bfi2_h734	bfi2_h735	bfi2_h736	bfi2_h737	bfi2_h738	bfi2_h739	bfi2_h740	bfi2_h741	bfi2_h742	bfi2_h743	bfi2_h744	bfi2_h745	bfi2_h746	bfi2_h747	bfi2_h748	bfi2_h749	bfi2_h750	bfi2_h751	bfi2_h752	bfi2_h753	bfi2_h754	bfi2_h755	bfi2_h756	bfi2_h757	bfi2_h758	bfi2_h759	bfi2_h760	bfi2_h761	bfi2_h762	bfi2_h763	bfi2_h764	bfi2_h765	bfi2_h766	bfi2_h767	bfi2_h768	bfi2_h769	bfi2_h770	bfi2_h771	bfi2_h772	bfi2_h773	bfi2_h774	bfi2_h775	bfi2_h776	bfi2_h777	bfi2_h778	bfi2_h779	bfi2_h780	bfi2_h781	bfi2_h782	bfi2_h783	bfi2_h784	bfi2_h785	bfi2_h786	bfi2_h787	bfi2_h788	bfi2_h789	bfi2_h790	bfi2_h791	bfi2_h792	bfi2_h793	bfi2_h794	bfi2_h795	bfi2_h796	bfi2_h797	bfi2_h798	bfi2_h799	bfi2_h800	bfi2_h801	bfi2_h802	bfi2_h803	bfi2_h804	bfi2_h805	bfi2_h806	bfi2_h807	bfi2_h808	bfi2_h809	bfi2_h810	bfi2_h811	bfi2_h812	bfi2_h813	bfi2_h814	bfi2_h815	bfi2_h816	bfi2_h817	bfi2_h818	bfi2_h819	bfi2_h820	bfi2_h821	bfi2_h822	bfi2_h823	bfi2_h824	bfi2_h825	bfi2_h826	bfi2_h827	bfi2_h828	bfi2_h829	bfi2_h830	bfi2_h831	bfi2_h832	bfi2_h833	bfi2_h834	bfi2_h835	bfi2_h836	bfi2_h837	bfi2_h838	bfi2_h839	bfi2_h840	bfi2_h841	bfi2_h842	bfi2_h843	bfi2_h844	bfi2_h845	bfi2_h846	bfi2_h847	bfi2_h848	bfi2_h849	bfi2_h850	bfi2_h851	bfi2_h852	bfi2_h853	bfi2_h854	bfi2_h855	bfi2_h856	bfi2_h857	bfi2_h858	bfi2_h859	bfi2_h860	bfi2_h861	bfi2_h862	bfi2_h863	bfi2_h864	bfi2_h865	bfi2_h866	bfi2_h867	bfi2_h868	bfi2_h869	bfi2_h870	bfi2_h871	bfi2_h872	bfi2_h873	bfi2_h874	bfi2_h875	bfi2_h876	bfi2_h877	bfi2_h878	bfi2_h879	bfi2_h880	bfi2_h881	bfi2_h882	bfi2_h883	bfi2_h884	bfi2_h885	bfi2_h886	bfi2_h887	bfi2_h888	bfi2_h889	bfi2_h890	bfi2_h891	bfi2_h892	bfi2_h893	bfi2_h894	bfi2_h895	bfi2_h896	bfi2_h897	bfi2_h898	bfi2_h899	bfi2_h900	bfi2_h901	bfi2_h902	bfi2_h903	bfi2_h904	bfi2_h905	bfi2_h906	bfi2_h907	bfi2_h908	bfi2_h909	bfi2_h910	bfi2_h911	bfi2_h912	bfi2_h913	bfi2_h914	bfi2_h915	bfi2_h916	bfi2_h917	bfi2_h918	bfi2_h919	bfi2_h920	bfi2_h921	bfi2_h922	bfi2_h923	bfi2_h924	bfi2_h925	bfi2_h926	bfi2_h927	bfi2_h928	bfi2_h929	bfi2_h930	bfi2_h931	bfi2_h932	bfi2_h933	bfi2_h934	bfi2_h935	bfi2_h936	bfi2_h937	bfi2_h938	bfi2_h939	bfi2_h940	bfi2_h941	bfi2_h942	bfi2_h943	bfi2_h944	bfi2_h945	bfi2_h946	bfi2_h947	bfi2_h948	bfi2_h949	bfi2_h950	bfi2_h951	bfi2_h952	bfi2_h953	bfi2_h954	bfi2_h955	bfi2_h956	bfi2_h957	bfi2_h958	bfi2_h959	bfi2_h960	bfi2_h961	bfi2_h962	bfi2_h963	bfi2_h964	bfi2_h965	bfi2_h966	bfi2_h967	bfi2_h968	bfi2_h969	bfi2_h970	bfi2_h971	bfi2_h972	bfi2_h973	bfi2_h974	bfi2_h975	bfi2_h976	bfi2_h977	bfi2_h978	bfi2_h979	bfi2_h980	bfi2_h981	bfi2_h982	bfi2_h983	bfi2_h984	bfi2_h985	bfi2_h986	bfi2_h987	bfi2_h988	bfi2_h989	bfi2_h990	bfi2_h991	bfi2_h992	bfi2_h993	bfi2_h994	bfi2_h995	bfi2_h996	bfi2_h997	bfi2_h998	bfi2_h999	bfi2_h1000	bfi2_h1001	bfi2_h1002	bfi2_h1003	bfi2_h1004	bfi2_h1005	bfi2_h1006	bfi2_h1007	bfi2_h1008	bfi2_h1009	bfi2_h1010	bfi2_h1011	bfi2_h1012	bfi2_h1013	bfi2_h1014	bfi2_h1015	bfi2_h1016	bfi2_h1017	bfi2_h1018	bfi2_h1019	bfi2_h1020	bfi2_h1021	bfi2_h1022	bfi2_h1023	bfi2_h1024	bfi2_h1025	bfi2_h1026	bfi2_h1027	bfi2_h1028	bfi2_h1029	bfi2_h1030	bfi2_h1031	bfi2_h1032	bfi2_h1033	bfi2_h1034	bfi2_h1035	bfi2_h1036	bfi2_h1037	bfi2_h1038	bfi2_h1039	bfi2_h1040	bfi2_h1041	bfi2_h1042	bfi2_h1043	bfi2_h1044	bfi2_h1045	bfi2_h1046	bfi2_h1047	bfi2_h1048	bfi2_h1049	bfi2_h1050	bfi2_h1051	bfi2_h1052	bfi2_h1053	bfi2_h1054	bfi2_h1055	bfi2_h1056	bfi2_h1057	bfi2_h1058	bfi2_h1059	bfi2_h1060	bfi2_h1061	bfi2_h1062	bfi2_h1063	bfi2_h1064	bfi2_h1065	bfi2_h1066	bfi2_h1067	bfi2_h1068	bfi2_h1069	bfi2_h1070	bfi2_h1071	bfi2_h1072	bfi2_h1073	bfi2_h1074	bfi2_h1075	bfi2_h1076	bfi2_h1077	bfi2_h1078	bfi2_h1079	bfi2_h1080	bfi2_h1081	bfi2_h1082	bfi2_h1083
----------	---------	---------	---------	---------	---------	---------	---------	---------	---------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------

```

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")

  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), "\n"))
    }
  } 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 week", "more than twice a week"))
  )-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 D_wide
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`
  ),
  psqi_latency_score_i2 = score_sleep_latency(
    psqi_q_2_response_numeric_i2,
    `frequency_trouble_sleep_because;psqi-past_q_6_response`
  )

```

```
)
```

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.