Phase 2 Dataset QA

Julia

2024-06-10

Quality Assurance Check

```
# Load the tidyverse (including ggplot2) and janitor
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.4     ✔ readr     2.1.5
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ ggplot2   3.4.4     ✔ tibble    3.2.1
## ✔ lubridate 1.9.3     ✔ tidyr     1.3.0
## ✔ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```
library(janitor)
```

```
##
## Attaching package: 'janitor'
##
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
# Upload data
phase2_dat <- read_tsv("Phase2data.txt") |>  janitor::clean_names()
```

```
## Rows: 89110 Columns: 31
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: "\t"
## chr  (13): Sample, Anon Student Id, Problem Hierarchy, Problem Name, Step Du...
## dbl  (13): Row, Problem View, Step Name, Incorrects, Hints, Corrects, Opport...
## lgl   (1): Predicted Error Rate (Unique-step)
## dttm  (4): Step Start Time, First Transaction Time, Correct Transaction Time...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
pretest_dat <- read_tsv("pretest_data.txt") |>  janitor::clean_names()
```

```
## Rows: 22945 Columns: 31
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: "\t"
## chr  (13): Sample, Anon Student Id, Problem Hierarchy, Problem Name, Step Du...
## dbl  (13): Row, Problem View, Step Name, Incorrects, Hints, Corrects, Opport...
## lgl   (1): Predicted Error Rate (Unique-step)
## dttm  (4): Step Start Time, First Transaction Time, Correct Transaction Time...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
practice_dat <- read_tsv("practice_data.txt") |>  janitor::clean_names()
```

```
## Rows: 44810 Columns: 31
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: "\t"
## chr  (13): Sample, Anon Student Id, Problem Hierarchy, Problem Name, Step Du...
## dbl  (13): Row, Problem View, Step Name, Incorrects, Hints, Corrects, Opport...
## lgl   (1): Predicted Error Rate (Unique-step)
## dttm  (4): Step Start Time, First Transaction Time, Correct Transaction Time...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
posttest_dat <- read_tsv("posttest_data.txt") |>  janitor::clean_names()
```

```
## Rows: 21355 Columns: 31
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: "\t"
## chr  (13): Sample, Anon Student Id, Problem Hierarchy, Problem Name, Step Du...
## dbl  (13): Row, Problem View, Step Name, Incorrects, Hints, Corrects, Opport...
## lgl   (1): Predicted Error Rate (Unique-step)
## dttm  (4): Step Start Time, First Transaction Time, Correct Transaction Time...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Goal of this QA:

Check that all conditions that were supposed to happen happened, that participants completed all conditions (and how many didn't), how many trials per condition per participant, that there is pre-post for all conditions for all participants (and how for how many there isn't), etc.

```
# check structure of the data
str(phase2_dat)
```

```
## spc_tbl_ [89,110 × 31] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ row                           : num [1:89110] 1 2 3 4 5 6 7 8 9 10 ...
##  $ sample                        : chr [1:89110] "All Data" "All Data" "All Data"
"All Data" ...
##  $ anon_student_id               : chr [1:89110] "23911" "23911" "23911" "23911"
...
##  $ problem_hierarchy             : chr [1:89110] "Topic 6.10B Rock Cycle" "Topic 6.
10B Rock Cycle" "Topic 6.10B Rock Cycle" "Topic 6.10B Rock Cycle" ...
##  $ problem_name                  : chr [1:89110] "What does the rock cycle includ
e?" "Metamorphic rocks can form without the application of heat and pressure." "How can
sedimentary rocks become igneous rocks?" "Magma originates from the melting of _____ and
sedimentary rocks." ...
##  $ problem_view                  : num [1:89110] 1 1 1 1 1 1 1 1 1 1 ...
##  $ step_name                     : num [1:89110] 103497 103508 103565 103578 103510
...
##  $ step_start_time               : POSIXct[1:89110], format: "2024-03-26 17:05:49"
"2024-03-26 17:06:38" ...
##  $ first_transaction_time        : POSIXct[1:89110], format: "2024-03-26 17:06:38"
"2024-03-26 17:06:59" ...
##  $ correct_transaction_time      : POSIXct[1:89110], format: "2024-03-26 17:06:38"
"2024-03-26 17:06:59" ...
##  $ step_end_time                 : POSIXct[1:89110], format: "2024-03-26 17:06:38"
"2024-03-26 17:06:59" ...
##  $ step_duration_sec             : chr [1:89110] "49" "21" "28" "197" ...
##  $ correct_step_duration_sec     : chr [1:89110] "49" "21" "28" "197" ...
##  $ error_step_duration_sec       : chr [1:89110] "." "." "." "." ...
##  $ first_attempt                 : chr [1:89110] "correct" "correct" "correct" "cor
rect" ...
##  $ incorrects                    : num [1:89110] 0 0 0 0 0 1 1 1 1 1 ...
##  $ hints                         : num [1:89110] 0 0 0 0 0 0 0 0 0 0 ...
##  $ corrects                      : num [1:89110] 1 1 1 1 1 0 0 0 0 0 ...
##  $ condition                     : chr [1:89110] "Low Learning objective spacing (L
earning objective spacing)~~Low Question variability (Question variability)~~H"| __trunc
ated__ "Low Learning objective spacing (Learning objective spacing)~~Low Question variab
ility (Question variability)~~H"| __truncated__ "Low Learning objective spacing (Learnin
g objective spacing)~~Low Question variability (Question variability)~~H"| __truncated__
"Low Learning objective spacing (Learning objective spacing)~~Low Question variability
(Question variability)~~H"| __truncated__ ...
##  $ kc_lo                         : chr [1:89110] "I can describe how a rock is form
ed through a cycle" "I can describe how a rock is formed through a cycle" "I can explain
how an igneous rock is transformed from a metamorphic and sedimentary rock" "I can expla
in how an igneous rock is transformed from a metamorphic and sedimentary rock" ...
##  $ opportunity_lo                : num [1:89110] 1 2 1 2 1 2 1 2 1 2 ...
##  $ predicted_error_rate_lo       : num [1:89110] 0.466 0.465 0.538 0.529 0.564 ...
##  $ kc_single_kc                  : chr [1:89110] "Single-KC" "Single-KC" "Single-K
C" "Single-KC" ...
##  $ opportunity_single_kc         : num [1:89110] 1 2 3 4 5 6 7 8 9 10 ...
##  $ predicted_error_rate_single_kc  : num [1:89110] 0.542 0.542 0.542 0.542 0.542 ...
##  $ kc_topic                      : chr [1:89110] "6.10B Rock Cycle" "6.10B Rock Cyc
le" "6.10B Rock Cycle" "6.10B Rock Cycle" ...
##  $ opportunity_topic             : num [1:89110] 1 2 3 4 5 6 7 8 9 10 ...
##  $ predicted_error_rate_topic    : num [1:89110] 0.508 0.508 0.508 0.508 0.508 ...
```

```
##  $ kc_unique_step                 : chr [1:89110] "KC2292" "KC370" "KC879" "KC1835"
...
##  $ opportunity_unique_step        : num [1:89110] 1 1 1 1 1 1 1 1 1 1 ...
##  $ predicted_error_rate_unique_step: logi [1:89110] NA NA NA NA NA NA ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Row = col_double(),
##   ..   Sample = col_character(),
##   ..   `Anon Student Id` = col_character(),
##   ..   `Problem Hierarchy` = col_character(),
##   ..   `Problem Name` = col_character(),
##   ..   `Problem View` = col_double(),
##   ..   `Step Name` = col_double(),
##   ..   `Step Start Time` = col_datetime(format = ""),
##   ..   `First Transaction Time` = col_datetime(format = ""),
##   ..   `Correct Transaction Time` = col_datetime(format = ""),
##   ..   `Step End Time` = col_datetime(format = ""),
##   ..   `Step Duration (sec)` = col_character(),
##   ..   `Correct Step Duration (sec)` = col_character(),
##   ..   `Error Step Duration (sec)` = col_character(),
##   ..   `First Attempt` = col_character(),
##   ..   Incorrects = col_double(),
##   ..   Hints = col_double(),
##   ..   Corrects = col_double(),
##   ..   Condition = col_character(),
##   ..   `KC (LO)` = col_character(),
##   ..   `Opportunity (LO)` = col_double(),
##   ..   `Predicted Error Rate (LO)` = col_double(),
##   ..   `KC (Single-KC)` = col_character(),
##   ..   `Opportunity (Single-KC)` = col_double(),
##   ..   `Predicted Error Rate (Single-KC)` = col_double(),
##   ..   `KC (Topic)` = col_character(),
##   ..   `Opportunity (Topic)` = col_double(),
##   ..   `Predicted Error Rate (Topic)` = col_double(),
##   ..   `KC (Unique-step)` = col_character(),
##   ..   `Opportunity (Unique-step)` = col_double(),
##   ..   `Predicted Error Rate (Unique-step)` = col_logical()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
# get summary of the data
summary(phase2_dat)
```

```
##       row           sample          anon_student_id    problem_hierarchy
##  Min.   :    1   Length:89110       Length:89110       Length:89110
##  1st Qu.:22278   Class :character   Class :character   Class :character
##  Median :44556   Mode  :character   Mode  :character   Mode  :character
##  Mean   :44556
##  3rd Qu.:66833
##  Max.   :89110
##
##  problem_name        problem_view     step_name
##  Length:89110       Min.   :1.000   Min.   :100485
##  Class :character   1st Qu.:1.000   1st Qu.:101378
##  Mode  :character   Median :1.000   Median :102239
##                     Mean   :1.009   Mean   :103887
##                     3rd Qu.:1.000   3rd Qu.:108820
##                     Max.   :6.000   Max.   :109624
##
##  step_start_time                 first_transaction_time
##  Min.   :2024-03-19 13:40:53.00   Min.   :2024-03-19 13:41:21.00
##  1st Qu.:2024-03-28 19:09:23.75   1st Qu.:2024-03-28 18:52:28.00
##  Median :2024-04-11 18:12:00.00   Median :2024-04-11 17:38:43.50
##  Mean   :2024-04-15 13:03:43.73   Mean   :2024-04-15 11:37:24.60
##  3rd Qu.:2024-05-02 14:01:45.25   3rd Qu.:2024-05-02 14:01:37.50
##  Max.   :2024-05-31 20:53:00.00   Max.   :2024-05-31 20:53:12.00
##  NA's   :738
##  correct_transaction_time         step_end_time
##  Min.   :2024-03-19 13:41:21.00   Min.   :2024-03-19 13:41:21.00
##  1st Qu.:2024-03-28 20:24:34.00   1st Qu.:2024-03-28 18:52:28.00
##  Median :2024-04-11 20:13:44.00   Median :2024-04-11 17:38:43.50
##  Mean   :2024-04-16 10:26:37.45   Mean   :2024-04-15 11:37:24.60
##  3rd Qu.:2024-05-02 15:40:31.00   3rd Qu.:2024-05-02 14:01:37.50
##  Max.   :2024-05-31 20:53:00.00   Max.   :2024-05-31 20:53:12.00
##  NA's   :55389
##  step_duration_sec  correct_step_duration_sec error_step_duration_sec
##  Length:89110       Length:89110              Length:89110
##  Class :character   Class :character          Class :character
##  Mode  :character   Mode  :character          Mode  :character
##
##
##
##
##  first_attempt        incorrects         hints      corrects
##  Length:89110       Min.   :0.0000   Min.   :0   Min.   :0.0000
##  Class :character   1st Qu.:0.0000   1st Qu.:0   1st Qu.:0.0000
##  Mode  :character   Median :1.0000   Median :0   Median :0.0000
##                     Mean   :0.6217   Mean   :0   Mean   :0.3784
##                     3rd Qu.:1.0000   3rd Qu.:0   3rd Qu.:1.0000
##                     Max.   :3.0000   Max.   :0   Max.   :1.0000
##
##   condition            kc_lo           opportunity_lo   predicted_error_rate_lo
##  Length:89110       Length:89110       Min.   : 1.000   Min.   :0.2219
##  Class :character   Class :character   1st Qu.: 2.000   1st Qu.:0.5320
##  Mode  :character   Mode  :character   Median : 4.000   Median :0.6248
```

```
##                                    Mean   : 4.412   Mean   :0.6216
##                                    3rd Qu.: 6.000   3rd Qu.:0.7142
##                                    Max.   :11.000   Max.   :0.9263
##
##  kc_single_kc       opportunity_single_kc predicted_error_rate_single_kc
##  Length:89110       Min.   :  1.00        Min.   :0.3605
##  Class :character   1st Qu.: 49.00        1st Qu.:0.5418
##  Mode  :character   Median : 97.00        Median :0.6202
##                     Mean   : 97.86        Mean   :0.6216
##                     3rd Qu.:146.00        3rd Qu.:0.7034
##                     Max.   :204.00        Max.   :0.8538
##
##    kc_topic         opportunity_topic predicted_error_rate_topic
##  Length:89110       Min.   : 1.00     Min.   :0.3206
##  Class :character   1st Qu.:10.00     1st Qu.:0.5381
##  Mode  :character   Median :20.00     Median :0.6234
##                     Mean   :19.99     Mean   :0.6216
##                     3rd Qu.:30.00     3rd Qu.:0.7087
##                     Max.   :42.00     Max.   :0.8871
##
##  kc_unique_step     opportunity_unique_step predicted_error_rate_unique_step
##  Length:89110       Min.   :1.000           Mode:logical
##  Class :character   1st Qu.:1.000           NA's:89110
##  Mode  :character   Median :1.000
##                     Mean   :1.001
##                     3rd Qu.:1.000
##                     Max.   :6.000
##                     NA's   :1084
```

It can be seen that the dataset presents 89,110 rows. However, there may be duplicate records that could inflate the participant count. So, a check for duplicates will be done below.

```
# check for duplicates in the participants
duplicate_check <- phase2_dat |>
  group_by(anon_student_id, condition) |>
  summarise(count = n()) |>
  filter(count > 1)
```

```
## `summarise()` has grouped output by 'anon_student_id'. You can override using
## the `.groups` argument.
```

```
nrow(duplicate_check)
```

```
## [1] 3712
```

```
head(duplicate_check, 12)
```

```
## # A tibble: 12 × 3
## # Groups:   anon_student_id [2]
##    anon_student_id condition                                              count
##    <chr>           <chr>                                                  <int>
##  1 23911           High Learning objective spacing (Learning objective sp…    16
##  2 23911           High Learning objective spacing (Learning objective sp…     8
##  3 23911           High Learning objective spacing (Learning objective sp…     8
##  4 23911           High Learning objective spacing (Learning objective sp…     8
##  5 23911           Low Learning objective spacing (Learning objective spa…    56
##  6 23911           Low Learning objective spacing (Learning objective spa…    24
##  7 23911           Low Learning objective spacing (Learning objective spa…    40
##  8 23911           Low Learning objective spacing (Learning objective spa…    40
##  9 23912           High Learning objective spacing (Learning objective sp…     8
## 10 23912           High Learning objective spacing (Learning objective sp…    16
## 11 23912           High Learning objective spacing (Learning objective sp…     8
## 12 23912           High Learning objective spacing (Learning objective sp…     8
```

There are 3,712 duplicates, but if you view the dataset, it can be seen that there are 8 conditions for each duplicated participant. So, there should be around 464 total participants who have completed all 8 conditions.***However, this does not align with our 89,110 total rows in the dataset. This suggests that there might be additional data not accounted for by the 465 participants completing 8 conditons each.***

```
# verify the total number of unique participants after checking duplicates before
total_unique_participants <- phase2_dat |>
  summarise(unique_participants = n_distinct(anon_student_id))
total_unique_participants
```

```
## # A tibble: 1 × 1
##   unique_participants
##                 <int>
## 1                 465
```

Since there is a mismatch in expected rows and total rows (465 participants, but 89,110 rows), let's find the distribution of entries for participants. The ideal outcome would include most participants with 8 entries. Let's see if there are any anomalies.

```
# created a dataframe named 'participant_entries' to show the number of unique "entries"
or rows for each participant
participant_entries <- phase2_dat |>
  group_by(anon_student_id) |>
  summarise(entries = n())

# Summarize the entries to see the distribution
entries_distribution <- participant_entries |>
  count(entries)

print(participant_entries)
```

```
## # A tibble: 465 × 2
##    anon_student_id entries
##    <chr>             <int>
##  1 23911               200
##  2 23912               200
##  3 23913               200
##  4 23914               200
##  5 23915               200
##  6 23916               175
##  7 23917               201
##  8 23918               200
##  9 23919               200
## 10 23920               180
## # ℹ 455 more rows
```

```
print(entries_distribution)
```

```
## # A tibble: 49 × 2
##    entries     n
##      <int> <int>
##  1      19     1
##  2      25     1
##  3      50     1
##  4      54     1
##  5      58     1
##  6      60     1
##  7      66     1
##  8      98     1
##  9     115     1
## 10     121     1
## # ℹ 39 more rows
```

Overall, it can be seen that there are many participants who have 100+ entries/rows (with the greatest number of participants having 200 entries) which accounts for the high row count of 89,110 in the dataset.Since there are so many entries per participant, we'll need to count the number of trials per participant, which can be shown a few sections below.

Check for missing values:

```
# check which columns (variables) have missing values by using 'colSums' function
colSums(is.na(phase2_dat))
```

```
##                              row                           sample
##                                0                                0
##                   anon_student_id                problem_hierarchy
##                                0                                0
##                      problem_name                     problem_view
##                                0                                0
##                         step_name                   step_start_time
##                                0                              738
##           first_transaction_time         correct_transaction_time
##                                0                            55389
##                     step_end_time                step_duration_sec
##                                0                                0
##        correct_step_duration_sec          error_step_duration_sec
##                                0                                0
##                     first_attempt                       incorrects
##                                0                                0
##                             hints                         corrects
##                                0                                0
##                         condition                            kc_lo
##                                0                                0
##                   opportunity_lo          predicted_error_rate_lo
##                                0                                0
##                      kc_single_kc            opportunity_single_kc
##                                0                                0
##   predicted_error_rate_single_kc                         kc_topic
##                                0                                0
##                opportunity_topic       predicted_error_rate_topic
##                                0                                0
##                   kc_unique_step          opportunity_unique_step
##                             1084                             1084
## predicted_error_rate_unique_step
##                            89110
```

From this function, it looks like 'correct_transaction_time', 'step_start_time', 'kc_unique_step', 'opportunity_unique_step', and 'predicted_error_rate_unique_step' have missing values (NAs).

Verify all conditions occurred:

```
# outputs all unique conditions present in the 'condition' column to verify that all con
ditions are present (all conditions that were supposed to happen happened.)
unique(phase2_dat$condition)
```

```
## [1] "Low Learning objective spacing (Learning objective spacing)~~Low Question variab
ility (Question variability)~~High Topic spacing (Topic spacing)"
## [2] "Low Learning objective spacing (Learning objective spacing)~~High Question varia
bility (Question variability)~~High Topic spacing (Topic spacing)"
## [3] "High Learning objective spacing (Learning objective spacing)~~High Question vari
ability (Question variability)~~High Topic spacing (Topic spacing)"
## [4] "High Learning objective spacing (Learning objective spacing)~~Low Question varia
bility (Question variability)~~High Topic spacing (Topic spacing)"
## [5] "High Learning objective spacing (Learning objective spacing)~~High Question vari
ability (Question variability)~~Low Topic spacing (Topic spacing)"
## [6] "Low Learning objective spacing (Learning objective spacing)~~Low Question variab
ility (Question variability)~~Low Topic spacing (Topic spacing)"
## [7] "Low Learning objective spacing (Learning objective spacing)~~High Question varia
bility (Question variability)~~Low Topic spacing (Topic spacing)"
## [8] "High Learning objective spacing (Learning objective spacing)~~Low Question varia
bility (Question variability)~~Low Topic spacing (Topic spacing)"
```

Verify if participants completed all conditions:

```
# check that participants completed all conditions
participant_conditions <- phase2_dat |>
  group_by(anon_student_id) |>
  summarise(conditions_completed = n_distinct(condition))

print(paste("Number of participants who completed all conditions:", nrow(participant_con
ditions))) # count number of participants that completed all conditions
```

```
## [1] "Number of participants who completed all conditions: 465"
```

```
# check participants who did not complete all conditions
incomplete_participant_cond <- participant_conditions |>
  filter(conditions_completed < 8)

incomplete_participant_cond
```

```
## # A tibble: 3 × 2
##   anon_student_id conditions_completed
##   <chr>                          <int>
## 1 24652                              6
## 2 29078                              7
## 3 29091                              7
```

A total of 3 participants did not complete all 8 conditions.

Count trials per condition per participant

```
trials_per_condition <- phase2_dat |>
  group_by(anon_student_id, condition) |>
  summarise(trials = n())
```

```
## `summarise()` has grouped output by 'anon_student_id'. You can override using
## the `.groups` argument.
```

```
head(trials_per_condition)
```

```
## # A tibble: 6 × 3
## # Groups:   anon_student_id [1]
##   anon_student_id condition                                              trials
##   <chr>           <chr>                                                   <int>
## 1 23911           High Learning objective spacing (Learning objective sp…    16
## 2 23911           High Learning objective spacing (Learning objective sp…     8
## 3 23911           High Learning objective spacing (Learning objective sp…     8
## 4 23911           High Learning objective spacing (Learning objective sp…     8
## 5 23911           Low Learning objective spacing (Learning objective spa…    56
## 6 23911           Low Learning objective spacing (Learning objective spa…    24
```

The first row shows that there were 16 trials for the first condition for the first participant. Since they completed all 8 conditions, and each condition has differing numbers of trials, this explains how this first participant has 200 entries/rows, as calculated above.

Check Pre-Post Data:

```
# Check for unique participants in each dataset to see if they match up with the Phase 2
dataset
unique_pretest <- n_distinct(pretest_dat$anon_student_id)
unique_posttest <- n_distinct(posttest_dat$anon_student_id)
unique_practice <- n_distinct(practice_dat$anon_student_id)

print(paste("Unique participants in pretest:", unique_pretest))
```

```
## [1] "Unique participants in pretest: 464"
```

```
print(paste("Unique participants in posttest:", unique_posttest))
```

```
## [1] "Unique participants in posttest: 432"
```

```
print(paste("Unique participants in practice:", unique_practice))
```

```
## [1] "Unique participants in practice: 464"
```

It seems like all 464 participants continued throughout the practice, but only 432 participants were able to take part in the post-test.

```
# merge pre-post-practice datasets
combined_prepost <- bind_rows(pretest_dat, posttest_dat,practice_dat)

# Check for pre-post data for all conditions for all participants
prepost_check <- combined_prepost |>
  group_by(anon_student_id, condition) |>
  summarise(present_test_types = n_distinct(sample)) |>
  filter(present_test_types > 2)
```

```
## `summarise()` has grouped output by 'anon_student_id'. You can override using
## the `.groups` argument.
```

```
unique_participants <- prepost_check |>
  summarise(unique_participants_with_prepost = n_distinct(anon_student_id))

# Check for participants missing pre-post data
missing_prepost <- combined_prepost |>
  group_by(anon_student_id, condition) |>
  summarise(present_test_types = n_distinct(sample)) |>
  filter(present_test_types < 3) |>
  summarise(unique_missing_prepost = n_distinct(anon_student_id))
```

```
## `summarise()` has grouped output by 'anon_student_id'. You can override using
## the `.groups` argument.
```

```
print(paste("Number of unique participants with pretest, posttest, or practice data for
any condition:", nrow(unique_participants)))
```

```
## [1] "Number of unique participants with pretest, posttest, or practice data for any c
ondition: 431"
```

```
print(paste("Number of unique participants missing pretest, posttest, or practice data f
or any condition:", nrow(missing_prepost)))
```

```
## [1] "Number of unique participants missing pretest, posttest, or practice data for an
y condition: 46"
```

- Number of unique participants ***with*** pretest, posttest, or practice data for any condition: 431
- Number of unique participants ***missing*** pretest, posttest, or practice data for any condition: 46

```
# check how many trials there were for each condition per student in the pretest, experi
ment, and posttest
trials_per_condition <- combined_prepost |>
  group_by(anon_student_id, condition, sample) |>
  summarise(trials = n(), .groups = 'drop')

head(trials_per_condition) # display only the first 6 rows
```

```
## # A tibble: 6 × 4
##   anon_student_id condition                                          sample trials
##   <chr>           <chr>                                              <chr>   <int>
## 1 23911           High Learning objective spacing (Learning objec… Postt…      4
## 2 23911           High Learning objective spacing (Learning objec… Pract…      8
## 3 23911           High Learning objective spacing (Learning objec… Prete…      4
## 4 23911           High Learning objective spacing (Learning objec… Postt…      2
## 5 23911           High Learning objective spacing (Learning objec… Pract…      4
## 6 23911           High Learning objective spacing (Learning objec… Prete…      2
```

Check how many times, across the entire sample of students who completed everything, each question was seen:

```
# first, group by student id and condition to see the distribution of test types and the
n filter by 'conditions_completed' to only include participants who have completed all c
onditions and tests
num_questions <- combined_prepost |>
  group_by(anon_student_id, condition) |>
  summarise(present_test_types = n_distinct(sample)) |>
  filter(present_test_types == 3) |>  # this filters to only show participants who have
completed all pre-post tests and practice
  group_by(anon_student_id) |>
  summarise(conditions_completed = n_distinct(condition)) |>
  filter(conditions_completed == 8)
```

```
## `summarise()` has grouped output by 'anon_student_id'. You can override using
## the `.groups` argument.
```

```
# Get the list of student ids who completed all conditions
students_completed_all <- num_questions$anon_student_id # create a vector containing the participants who have completed all 8 conditions

# Filter the combined data to include only students who completed all conditions
filtered_data <- combined_prepost |>
  filter(anon_student_id %in% students_completed_all)

# Count the number of times each question was seen
question_counts <- filtered_data |>
  group_by(problem_name) |>
  summarise(count = n()) |>
  arrange(desc(count)) # display count from high to low to see if there is any potential bias

question_counts
```

```
## # A tibble: 3,213 × 2
##    problem_name                                                    count
##    <chr>                                                           <int>
##  1 How do igneous rocks form?                                        104
##  2 What does weathering do to rocks?                                  92
##  3 What process leads to the formation of igneous rocks?              88
##  4 How do you calculate the total number of atoms in a molecule with a co…    84
##  5 What is the role of the brain in the Nervous System?               79
##  6 How many parents are involved in asexual reproduction?             77
##  7 Sexual reproduction requires _____ parents.                        77
##  8 How are sedimentary rocks formed?                                  75
##  9 To whom does the subscript outside the parentheses apply?          74
## 10 What is the primary function of the nervous system?                73
## # ℹ 3,203 more rows
```