Handling missing data

Contents

Run this chunk first!

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------

## v ggplot2 3.3.1     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   0.8.5
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
source('Data_Munging/FACES_fns.R')
```

Multiple ways to load data:

Merged Cells

Value in merged cells filled accordingly: - Merged Cell gets "unmerged" - Value in original merged cell occupies the left most cell after unmerging

Identifying missing data

This chunk identifies the missing data by participant, Time, Group, and Survey.

```
(missing_sum <- faces %>%
  filter(is.na(Response)) %>%
  group_by(`Participant #`, Time, Group, Survey) %>%
  summarise(missing_data = sum(is.na(Response))) %>%
  arrange(missing_data) %>%
  ungroup())
```

```
## # A tibble: 20 x 5
##    `Participant #` Time  Group        Survey missing_data
##              <dbl> <chr> <chr>        <chr>         <int>
##  1               5 Pre   Experimental AKS               1
##  2               8 Post  Experimental AKS               1
##  3               8 Post  Experimental FES               1
##  4              14 Post  Control      FACES             1
##  5              19 Post  Control      FACES             1
##  6              18 Pre   Control      FES               2
##  7               8 Pre   Experimental SEAS              5
##  8               6 Post  Experimental SCS               7
##  9               6 Pre   Experimental SCS               7
## 10               8 Pre   Experimental SCS               7
## 11              18 Pre   Control      SCS               7
## 12              19 Post  Control      SCS               7
## 13               6 Post  Experimental SEAS              9
## 14               6 Pre   Experimental SEAS              9
## 15               6 Post  Experimental FPPS             18
## 16               6 Pre   Experimental FPPS             18
## 17               8 Post  Experimental FPPS             18
## 18               8 Pre   Experimental FPPS             18
## 19              18 Pre   Control      FPPS             18
## 20              19 Post  Control      FPPS             18
```

Splitting the missing data

- Rule: If a participant is missing 2 or less responses, then we keep the participant;
  - else: we remove the participant from the affected surveys.
- The `distinct` function removes instances where the participant is missing more than 2 responses for both a `pre` and `post` time survey response. One instance in either `Pre` or `Post` is enough to get the record removed.

```
keepers <- missing_sum %>%
  filter(missing_data <= 2)

throwers <- missing_sum %>%
  distinct(`Participant #`, Group, Survey, .keep_all = T) %>% # added to removed Pre and Post
  filter(missing_data > 2)
```

Removing `throwers` from the overall dataset

- Right join filters out records
  - Right join by `Participant #`, `Group`, and `Survey` allows the participant records from `Pre` and `Post` to be matched up.
- The resulting data set is a list of records that needs to be removed from the original data set

```
(data_to_remove <- faces %>%
  right_join(throwers, by = c('Participant #', 'Group', 'Survey'),
             suffix = c('', '_mis')))
```

```
## # A tibble: 240 x 8
##    `Participant #` Time  Group    Survey Response Question Time_mis missing_data
##              <dbl> <chr> <chr>    <chr>     <dbl>    <int> <chr>           <int>
##  1               8 Pre   Experim~ SEAS          4        1 Pre                 5
##  2               8 Pre   Experim~ SEAS          4        2 Pre                 5
##  3               8 Pre   Experim~ SEAS          4        3 Pre                 5
##  4               8 Pre   Experim~ SEAS          4        4 Pre                 5
##  5               8 Pre   Experim~ SEAS         NA        5 Pre                 5
##  6               8 Pre   Experim~ SEAS          5        6 Pre                 5
##  7               8 Pre   Experim~ SEAS         NA        7 Pre                 5
##  8               8 Pre   Experim~ SEAS         NA        8 Pre                 5
##  9               8 Pre   Experim~ SEAS         NA        9 Pre                 5
## 10               8 Pre   Experim~ SEAS         NA       10 Pre                 5
## # ... with 230 more rows
```

Records removed below:

```
(faces_clean <-  setdiff(x = faces, y = select(data_to_remove, -Time_mis, -missing_data)))
```

```
## # A tibble: 2,864 x 6
##    `Participant #` Time  Group        Survey Response Question
##              <dbl> <chr> <chr>        <chr>     <dbl>    <int>
##  1               1 Pre   Experimental FACES         2        1
##  2               1 Pre   Experimental FACES         5        2
##  3               1 Pre   Experimental FACES         5        3
##  4               1 Pre   Experimental FACES         5        4
##  5               1 Pre   Experimental FACES         5        5
##  6               1 Pre   Experimental FACES         2        6
##  7               1 Pre   Experimental FACES         5        7
##  8               1 Pre   Experimental AKS           3        1
##  9               1 Pre   Experimental AKS           1        2
## 10               1 Pre   Experimental AKS           1        3
## # ... with 2,854 more rows
```

```
# Check that we only took the rows we intended to
nrow(faces_clean) == nrow(faces) - nrow(data_to_remove)
```

Quick checks

```
## [1] TRUE
```

```
# Check for missing data
anyNA(faces_clean)
```

```
## [1] TRUE
```

We still have missing data. Those are from the responses from the missing data in the keepers dataset.

Imputing the data:

- Right join filters out records, again
  - Right join by `Participant #`, `Group`, `Survey`, and `Time` because we only want to impute for that particular missing value
- Filter down to missing responses, because all questions for each survey category was matched
- The resulting data set is a list of records that needs to be imputed

```
(data_to_impute <- faces %>%
  right_join(keepers, by = c('Participant #', 'Group', 'Survey', 'Time'),
             suffix = c('', '_kp')) %>%
  filter(is.na(Response)))
```

```
## # A tibble: 7 x 7
##   `Participant #` Time  Group        Survey Response Question missing_data
##             <dbl> <chr> <chr>        <chr>     <dbl>    <int>        <int>
## 1               5 Pre   Experimental AKS          NA       15            1
## 2               8 Post  Experimental AKS          NA        8            1
## 3               8 Post  Experimental FES          NA        9            1
## 4              14 Post  Control      FACES        NA        7            1
## 5              19 Post  Control      FACES        NA        2            1
## 6              18 Pre   Control      FES          NA       29            2
## 7              18 Pre   Control      FES          NA       30            2
```

```
faces_clean2 <- faces_clean %>%
  group_by(Time, Group, Survey, Question) %>%
  mutate(Response = ifelse(is.na(Response), mean(Response, na.rm = T), Response)) %>%
  ungroup()
```

```
# To visulize what we changed
data_to_impute %>%
  left_join(faces_clean2, by = c('Participant #', 'Time', 'Group', 'Survey', 'Question'),
            suffix = c('_mis', '_fill'))
```

Quick check, again

```
## # A tibble: 7 x 8
##   `Participant #` Time  Group Survey Response_mis Question missing_data
##             <dbl> <chr> <chr> <chr>         <dbl>    <int>        <int>
## 1               5 Pre   Expe~ AKS              NA       15            1
## 2               8 Post  Expe~ AKS              NA        8            1
## 3               8 Post  Expe~ FES              NA        9            1
## 4              14 Post  Cont~ FACES            NA        7            1
## 5              19 Post  Cont~ FACES            NA        2            1
## 6              18 Pre   Cont~ FES              NA       29            2
## 7              18 Pre   Cont~ FES              NA       30            2
## # ... with 1 more variable: Response_fill <dbl>
```

```r
# Should be true
nrow(faces_clean2) == nrow(faces_clean)
```

```
## [1] TRUE
```

```r
# Should be false
anyNA(faces_clean2)
```

```
## [1] FALSE
```

Data is clean beyond previous chunk!

Summing Responses

This section pivots the data back to wide after summing up response by Participant for each Time (Pre and Post), Group (Int and Control), and Survey.

```r
(faces_sum <- faces_clean2 %>%
   # First group manipulation
  group_by(`Participant #`, Survey, Time, Group) %>%
  summarise(score = sum(Response),
            n_questions = n()) %>%
  ungroup() %>%

   # Second group manipulation
  group_by(Survey, Time, Group) %>%
  summarise(mean_participant_score = mean(score),
            spl_size = length(unique(`Participant #`)),
            question_chk = length(unique(n_questions)) == 1) %>%
  ungroup())
```

```
## # A tibble: 24 x 6
##    Survey Time  Group        mean_participant_score spl_size question_chk
##    <chr>  <chr> <chr>                         <dbl>    <int> <lgl>
##  1 AKS    Post  Control                        57.3        6 TRUE
##  2 AKS    Post  Experimental                   57.6       10 TRUE
##  3 AKS    Pre   Control                        57.3        6 TRUE
##  4 AKS    Pre   Experimental                   54.5       10 TRUE
##  5 FACES  Post  Control                        25.6        6 TRUE
##  6 FACES  Post  Experimental                   28.8       10 TRUE
##  7 FACES  Pre   Control                        25.3        6 TRUE
##  8 FACES  Pre   Experimental                   25.7       10 TRUE
##  9 FES    Post  Control                       130.         6 TRUE
## 10 FES    Post  Experimental                  151.        10 TRUE
## # ... with 14 more rows
```

First group manipulation - `score` is a sum of their responses - `n_questions` is the number of questions per survey. - Should be the same across comparisons

Second group manipulation - `mean_participant_score` is self explanatory. Should match closely with excel pivot table Shantel put together - `question_chk` is a check to ensure that the number of questions answered by each participant for each survey is the same while averaging responses.

Comparison via Wilcoxon test...

Did not do that yet, this chunk needs updating.

```
faces_sum %>%
  pivot_wider(id_cols = c( 'Survey', 'Group'),
              names_from = contains(c('Time')),
              values_from = c(mean_participant_score, spl_size))
```

```
## # A tibble: 12 x 6
##    Survey Group  mean_participant_~ mean_participant~ spl_size_Post spl_size_Pre
##    <chr>  <chr>               <dbl>             <dbl>         <int>        <int>
##  1 AKS    Contr~               57.3              57.3             6            6
##  2 AKS    Exper~               57.6              54.5            10           10
##  3 FACES  Contr~               25.6              25.3             6            6
##  4 FACES  Exper~               28.8              25.7            10           10
##  5 FES    Contr~              130.               127.             6            6
##  6 FES    Exper~              151.               134.            10           10
##  7 FPPS   Contr~               61.5              57               4            4
##  8 FPPS   Exper~               80.2              77.1             8            8
##  9 SCS    Contr~               26.8              21.8             4            4
## 10 SCS    Exper~               28.8              25.5             8            8
## 11 SEAS   Contr~               36                36               6            6
## 12 SEAS   Exper~               43.1              36.5             8            8
```