STATS 102A Homework 2

Winter 2022

Takao Oba
205615894

Due Janurary 30, 2022

Sourcing the Functions

```
source("205615894_stats102a_hw2.R")
```

1 Teacher's Gradebook

The formatting of this question takes 15% of your HW 2 grade; the efficiency, accuracy, and robustness of your function account for another 60%. If your function violates any instructions of this assignment, it will not be graded.

Requirements for `messy_impute()` and `tidy_impute()`

1. Draw flowcharts (or algorithms) of the main functions sufficiently complete, clear, and concise enough to enable a person to accurately implement the function in any programming languages they are adept with using.
2. Write the function(s) which accurately implements the algorithm(s) as described or requested.
3. Include the error-handling to ensure your function(s) work properly.

One scenario which naturally creates non-tidy data is a teacher's gradebook. Table 1 shows an example with five homework and five quizzes.

Table 1: An example of a teacher's gradebook.

UID	Homework_1	Homework_2	...	Homework_5	Quiz_1	...	Quiz_5
123456787	70	90	...	80	76	...	70
123456788	91	85	...	73	90	...	80
123456789	60	71	...	78	88	...	73

1(a) Dataset Simulation

Create a simulated dataset in R [1] called `gradebook` that represents a possible gradebook in the basic format as Table 1:

[1] Please check how to use `runif()`.

- Each row of the gradebook should contain all measurements for a student.
- Each column should contain scores for one assignment.
- The last column should be "Quiz_5."

The simulated gradebook should contain the grades for 100 students and scores (out of 100) for 5 homework and 5 quizzes. **Set the seed for simulating your data with your UID**.

1(b) Dataset Randomization

Write R code in RMarkdown file to randomly replace 10% of Homework_5 and Quiz_5 by NA respectively, and then use `is.na()` in conjunction with `sum()` to show your results.

```
set.seed(205615894)
index <- sample(1:n, n * 0.1)
index
```

```
##  [1] 48 36 65 77 78 87 76  3 88 20
```

```
gradebook[index, ]$hw5 = NA
sum(is.na(gradebook$hw5))
```

```
## [1] 10
```

```
set.seed(205615894 + 1)
index2 <- sample(1:n, n * 0.1)
index2
```

```
##  [1] 97 25 76 68 55 91 49 28 17 74
```

```
gradebook[index2, ]$quiz5 = NA
sum(is.na(gradebook$quiz5))
```

```
## [1] 10
```

1(c) `messy_impute()`

Write a function `messy_impute()` that imputes missing values in the gradebook. Please also present your algorithm or flowchart to answer this question in the RMarkdown file.

```
knitr::include_graphics("flowchart_messy_impute.png")
```

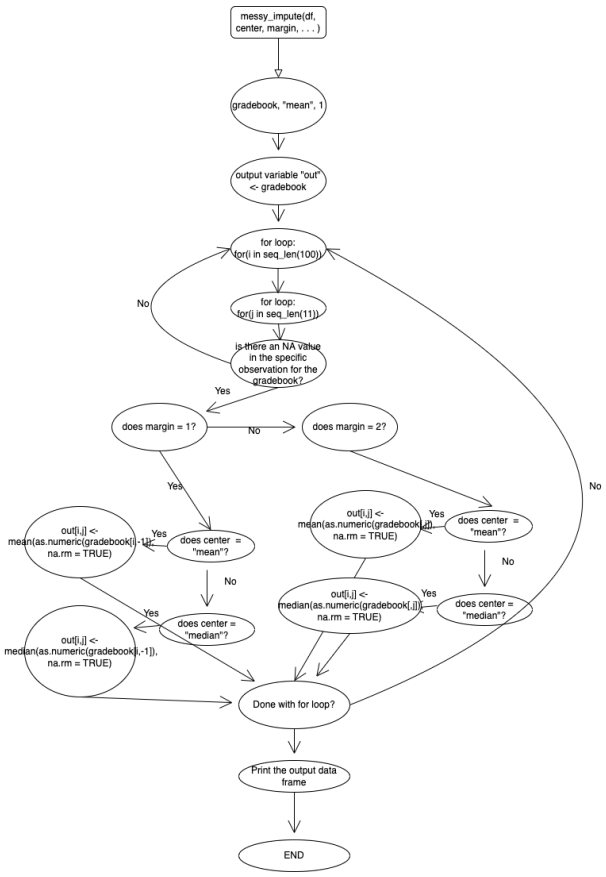

Figure 1: General stucture of a flowchart.

Note:

- *Imputation* is the process of replacing missing values by estimated values. The simplest (far from preferred) method to impute values is to replace missing values by the most typical value, say the mean.
- Assume the format of the gradebook is fixed (five homework and five quizzes), but NA values may occur in any assignments except for UID.
- The `messy_impute()` function should have at least three arguments and `...`:
  - A data frame contains the gradebook as specified in the example. – A center argument should be a character object indicating the impute functions (mean or median). – A margin argument should be an integer (1 or 2) indicating either the function imputes the missing values by row/student (1) or by column/assignment (2). If choosing by row, the function should process homework and quizzes separately.
- The function should return the imputed data frame (or tibble).

1(d) Test Cases

Select two students missing Homework_5 and two students missing Quiz_5. Please use these cases to demonstrate your function in the R markdown file. Here are some suggested cases but not limited to while grading.

```
# We will select student # 3 and student # 33 for those who had missing Homework_5
# We will select student # 17 and student # 27 for those who had missing Quiz_5

head(gradebook)
```

```
##         UID hw1 hw2 hw3 hw4 hw5 quiz1 quiz2 quiz3 quiz4 quiz5
## 1 205615894  47  63  99  82  74    48     6    70    99    32
## 2 205615895  35  57  37  74  84    63    83    16    31     3
## 3 205615896  64  53   5  14  NA    69    88    60    35    38
## 4 205615897  76  81  59  89  36    17    46    76    13    26
## 5 205615898  77  70  66  44  80    72     8     2    31    78
## 6 205615899  86  70  33  17  93    72    50    79    10    93
```

```
summary(gradebook[, c(6, 11)])
```

```
##       hw5             quiz5
##  Min.   :  2.00   Min.   : 0.00
##  1st Qu.: 31.00   1st Qu.:25.25
##  Median : 54.50   Median :49.00
##  Mean   : 53.01   Mean   :50.47
##  3rd Qu.: 77.75   3rd Qu.:74.00
##  Max.   :100.00   Max.   :99.00
##  NA's   :10       NA's   :10
```

```
# Test Case 1
messy_impute(gradebook, "mean", 1)
```

```
##           UID hw1 hw2 hw3 hw4       hw5 quiz1 quiz2 quiz3 quiz4    quiz5
## 1   205615894  47  63  99  82  74.00000    48     6    70    99 32.00000
## 2   205615895  35  57  37  74  84.00000    63    83    16    31  3.00000
## 3   205615896  64  53   5  14  47.33333    69    88    60    35 38.00000
## 4   205615897  76  81  59  89  36.00000    17    46    76    13 26.00000
## 5   205615898  77  70  66  44  80.00000    72     8     2    31 78.00000
## 6   205615899  86  70  33  17  93.00000    72    50    79    10 93.00000
## 7   205615900  75  88  68  84  83.00000    53    37    90    85 83.00000
## 8   205615901   2  63   1  46  86.00000    59    19    38    78 86.00000
## 9   205615902  87  91  11   8  43.00000    57    72     7    95 74.00000
## 10  205615903  19   5  81  90  31.00000    76    50     1     9  0.00000
## 11  205615904  55  78  94  91  63.00000    34    40    32     4 98.00000
## 12  205615905  25  20  85  59  56.00000     4    24    29    39 24.00000
## 13  205615906  62  69  42  92  17.00000    19    81    11    73  9.00000
## 14  205615907  50  66  54   8  16.00000    26    24    55    73 22.00000
## 15  205615908  12  93  41  88  53.00000    38    34    10    93 25.00000
## 16  205615909  23  19  85  29  45.00000     9    70    85    45 88.00000
## 17  205615910  75  76  94  16  77.00000    81    91    12    84 67.33333
## 18  205615911  77  11  24  69  37.00000    18    19     4   100 25.00000
## 19  205615912   4  59   1  35  86.00000    82    76    14    59 80.00000
## 20  205615913  14  92  72   4  50.33333    34    44    53    66 74.00000
## 21  205615914  25  28  72  62   4.00000    26    38    90    78 21.00000
## 22  205615915  70  45  93  22  52.00000    38    95    57     1 34.00000
## 23  205615916  50  83  21  75 100.00000    53    62    72     3 38.00000
## 24  205615917  96  65  48  66  68.00000     2    11     8    95 92.00000
```

```
## 25  205615918   8  14  68  52  22.00000    51    14    56    48 37.00000
## 26  205615919  69  43  28  15  52.00000    84    29    67    12 49.00000
## 27  205615920 100  91   7  64  40.00000     3    25    98    68 37.00000
## 28  205615921  93  38  23  51  66.00000     7    17    47    41 42.55556
## 29  205615922  70  71  12  67   5.00000    65    31    97    60 48.00000
## 30  205615923  35   4  10  74  61.00000    14    83    70    18 58.00000
## 31  205615924  22  57  13  88  50.00000    79    52    62    61 61.00000
## 32  205615925  73  28  83  86  20.00000    20    34    81    57 99.00000
## 33  205615926  76  29  80  71  54.00000    47    94    71    86 73.00000
## 34  205615927  59  29  39  35  60.00000    25    43    66     8 58.00000
## 35  205615928  31  81  10  37  59.00000    56    73    98    97 63.00000
## 36  205615929  86  59  51  18  68.00000    56    66    96    87 93.00000
## 37  205615930   2  11  78  59  87.00000    42    72    38    29  1.00000
## 38  205615931  20  46  84  15  71.00000    98    24     6    65 84.00000
## 39  205615932  25  98  21  88  32.00000    26    64    18     7 88.00000
## 40  205615933  74  36  83  37  16.00000    78    85    13    33 92.00000
## 41  205615934  82  95  34  82   4.00000     0    87    47    67 27.00000
## 42  205615935  64  25   2  27  21.00000    41    41    63     0 78.00000
## 43  205615936  25  31  92  24  31.00000    62    18    91    83 90.00000
## 44  205615937  35  12 100  71  24.00000    92    32    37    90 42.00000
## 45  205615938  88  58  20  70  65.00000    73    75    31    52 82.00000
## 46  205615939   5  78  82  10  11.00000    31    42    71    54 53.00000
## 47  205615940  96  62  35  47  87.00000    21    65    40    35 67.00000
## 48  205615941  57  13  76  29  41.77778    35    81    46    26 13.00000
## 49  205615942  56  36  94  41  64.00000    92     2    66    11 51.33333
## 50  205615943  25  38   0  73  85.00000    37    72    34    58 47.00000
## 51  205615944   9  40  46 100  67.00000    45    83    23     1 90.00000
## 52  205615945  60  40  18  77  84.00000    82     4    67    87 97.00000
## 53  205615946   1   9  18  55  73.00000    63    69    54    33 34.00000
## 54  205615947  86  76  60  17  48.00000    13     4    14    14 75.00000
## 55  205615948  56  47  72  48  47.00000    64    18    16    90 50.88889
## 56  205615949  26  22 100  98  79.00000    34    58    20    33 35.00000
## 57  205615950  73  79  14  96  83.00000    83    86    52    52 15.00000
## 58  205615951  25   7  31  70  72.00000    41     2     4    61 60.00000
## 59  205615952  99  35   5  88  46.00000    86    94    25     4 14.00000
## 60  205615953  88  33   1  75  75.00000    55    97    88    33 50.00000
## 61  205615954  93  72  26  11   3.00000    99    69    40    88 69.00000
## 62  205615955  45  84  17  21  13.00000    28    88    76    12 43.00000
## 63  205615956  75  42  44  52  72.00000    61    60    91    58 65.00000
## 64  205615957  93  85  64  54  54.00000    87    49    85    81 25.00000
## 65  205615958  87  52  94  11  53.22222    54     9    55    90 27.00000
## 66  205615959   6  57  31  87  20.00000     4    10    24    21 55.00000
## 67  205615960  74   4  86  96  77.00000    25    60     3    64 41.00000
## 68  205615961  57  56   4  58  55.00000    32    29    73    80 49.33333
## 69  205615962  54  21  28  33  38.00000    51    34    48    52  9.00000
## 70  205615963   7  70  59  53  49.00000     9    57    44    95 74.00000
## 71  205615964  63  97  43  24   5.00000    57    77    53    18 13.00000
## 72  205615965  55   7  35   4  94.00000    79    49    63    64 56.00000
## 73  205615966  24  61  59  12  48.00000    15     4    69    53 48.00000
## 74  205615967  14  43  85   0  25.00000    52    58   100    12 43.22222
## 75  205615968  13  85  45  80   2.00000    40    49     7    77 62.00000
## 76  205615969   5  95  24 100  55.87500     2    70    99    52 55.87500
## 77  205615970  18  83   6  20  44.44444    25     8    96    95 49.00000
## 78  205615971  28  58  98  85  67.88889    88    73    83    54 44.00000
```

```
## 79  205615972  51  78  37  99  42.00000    26    75     9    12  5.00000
## 80  205615973  28  15  37  34  34.00000    10    83    74    49 72.00000
## 81  205615974   0  14  27  88  78.00000     8    60    18    62  9.00000
## 82  205615975  30  26   9   1  31.00000     4    67    74    10 10.00000
## 83  205615976  96  30  56   4  13.00000    71    31    19    57 52.00000
## 84  205615977  69  71  93  31 100.00000    94    50    26    15 39.00000
## 85  205615978  48  74  13  25   3.00000    18    94    24    54 43.00000
## 86  205615979  29  54  60  37  88.00000    21    57    96    28 28.00000
## 87  205615980  93  36  90  89  70.44444    41    84    44    86 71.00000
## 88  205615981  99   7  10  68  52.22222    53    95    67    49 22.00000
## 89  205615982  10  17  25  35  84.00000    65    51    41    91 21.00000
## 90  205615983  80   8   6  29  80.00000    59    76     1    63 25.00000
## 91  205615984  21  52  37  57   8.00000    21    79    33    35 38.11111
## 92  205615985  67  65  48  76  71.00000    98    29    46    28 23.00000
## 93  205615986   8   5  15   8  21.00000    50    18    81    51 24.00000
## 94  205615987  92  21  51  38  90.00000    29   100    23     6 72.00000
## 95  205615988  25  46  12  98  92.00000     7    39    70    62 88.00000
## 96  205615989  95  94  49  44  57.00000    99    65    23    86 88.00000
## 97  205615990  60   2  59  49  60.00000    69    15    87    45 49.55556
## 98  205615991  33  31  91  65  81.00000    64    59    47    43 52.00000
## 99  205615992  41  74  27  95  52.00000    67    26    74    92 73.00000
## 100 205615993  19  14  44  90  86.00000    13    17    17    81 29.00000
```

```
summary(messy_impute(gradebook, "mean", 1)[,c(6, 11)])
```

```
##       hw5              quiz5
##  Min.   :  2.00   Min.   : 0.00
##  1st Qu.: 33.50   1st Qu.:27.00
##  Median : 54.00   Median :49.17
##  Mean   : 53.23   Mean   :50.27
##  3rd Qu.: 75.50   3rd Qu.:73.00
##  Max.   :100.00   Max.   :99.00
```

```
# Test Case 2
messy_impute(gradebook, "median", 2)
```

```
##           UID hw1 hw2 hw3 hw4   hw5 quiz1 quiz2 quiz3 quiz4 quiz5
## 1   205615894  47  63  99  82  74.0    48     6    70    99    32
## 2   205615895  35  57  37  74  84.0    63    83    16    31     3
## 3   205615896  64  53   5  14  54.5    69    88    60    35    38
## 4   205615897  76  81  59  89  36.0    17    46    76    13    26
## 5   205615898  77  70  66  44  80.0    72     8     2    31    78
## 6   205615899  86  70  33  17  93.0    72    50    79    10    93
## 7   205615900  75  88  68  84  83.0    53    37    90    85    83
## 8   205615901   2  63   1  46  86.0    59    19    38    78    86
## 9   205615902  87  91  11   8  43.0    57    72     7    95    74
## 10  205615903  19   5  81  90  31.0    76    50     1     9     0
## 11  205615904  55  78  94  91  63.0    34    40    32     4    98
## 12  205615905  25  20  85  59  56.0     4    24    29    39    24
## 13  205615906  62  69  42  92  17.0    19    81    11    73     9
## 14  205615907  50  66  54   8  16.0    26    24    55    73    22
## 15  205615908  12  93  41  88  53.0    38    34    10    93    25
## 16  205615909  23  19  85  29  45.0     9    70    85    45    88
```

```
## 17  205615910  75  76  94  16  77.0    81    91    12    84    49
## 18  205615911  77  11  24  69  37.0    18    19     4   100    25
## 19  205615912   4  59   1  35  86.0    82    76    14    59    80
## 20  205615913  14  92  72   4  54.5    34    44    53    66    74
## 21  205615914  25  28  72  62   4.0    26    38    90    78    21
## 22  205615915  70  45  93  22  52.0    38    95    57     1    34
## 23  205615916  50  83  21  75 100.0    53    62    72     3    38
## 24  205615917  96  65  48  66  68.0     2    11     8    95    92
## 25  205615918   8  14  68  52  22.0    51    14    56    48    49
## 26  205615919  69  43  28  15  52.0    84    29    67    12    49
## 27  205615920 100  91   7  64  40.0     3    25    98    68    37
## 28  205615921  93  38  23  51  66.0     7    17    47    41    49
## 29  205615922  70  71  12  67   5.0    65    31    97    60    48
## 30  205615923  35   4  10  74  61.0    14    83    70    18    58
## 31  205615924  22  57  13  88  50.0    79    52    62    61    61
## 32  205615925  73  28  83  86  20.0    20    34    81    57    99
## 33  205615926  76  29  80  71  54.0    47    94    71    86    73
## 34  205615927  59  29  39  35  60.0    25    43    66     8    58
## 35  205615928  31  81  10  37  59.0    56    73    98    97    63
## 36  205615929  86  59  51  18  54.5    56    66    96    87    93
## 37  205615930   2  11  78  59  87.0    42    72    38    29     1
## 38  205615931  20  46  84  15  71.0    98    24     6    65    84
## 39  205615932  25  98  21  88  32.0    26    64    18     7    88
## 40  205615933  74  36  83  37  16.0    78    85    13    33    92
## 41  205615934  82  95  34  82   4.0     0    87    47    67    27
## 42  205615935  64  25   2  27  21.0    41    41    63     0    78
## 43  205615936  25  31  92  24  31.0    62    18    91    83    90
## 44  205615937  35  12 100  71  24.0    92    32    37    90    42
## 45  205615938  88  58  20  70  65.0    73    75    31    52    82
## 46  205615939   5  78  82  10  11.0    31    42    71    54    53
## 47  205615940  96  62  35  47  87.0    21    65    40    35    67
## 48  205615941  57  13  76  29  54.5    35    81    46    26    13
## 49  205615942  56  36  94  41  64.0    92     2    66    11    49
## 50  205615943  25  38   0  73  85.0    37    72    34    58    47
## 51  205615944   9  40  46 100  67.0    45    83    23     1    90
## 52  205615945  60  40  18  77  84.0    82     4    67    87    97
## 53  205615946   1   9  18  55  73.0    63    69    54    33    34
## 54  205615947  86  76  60  17  48.0    13     4    14    14    75
## 55  205615948  56  47  72  48  47.0    64    18    16    90    49
## 56  205615949  26  22 100  98  79.0    34    58    20    33    35
## 57  205615950  73  79  14  96  83.0    83    86    52    52    15
## 58  205615951  25   7  31  70  72.0    41     2     4    61    60
## 59  205615952  99  35   5  88  46.0    86    94    25     4    14
## 60  205615953  88  33   1  75  75.0    55    97    88    33    50
## 61  205615954  93  72  26  11   3.0    99    69    40    88    69
## 62  205615955  45  84  17  21  13.0    28    88    76    12    43
## 63  205615956  75  42  44  52  72.0    61    60    91    58    65
## 64  205615957  93  85  64  54  54.0    87    49    85    81    25
## 65  205615958  87  52  94  11  54.5    54     9    55    90    27
## 66  205615959   6  57  31  87  20.0     4    10    24    21    55
## 67  205615960  74   4  86  96  77.0    25    60     3    64    41
## 68  205615961  57  56   4  58  55.0    32    29    73    80    49
## 69  205615962  54  21  28  33  38.0    51    34    48    52     9
## 70  205615963   7  70  59  53  49.0     9    57    44    95    74
```

```
## 71  205615964  63  97  43  24   5.0    57    77    53    18    13
## 72  205615965  55   7  35   4  94.0    79    49    63    64    56
## 73  205615966  24  61  59  12  48.0    15     4    69    53    48
## 74  205615967  14  43  85   0  25.0    52    58   100    12    49
## 75  205615968  13  85  45  80   2.0    40    49     7    77    62
## 76  205615969   5  95  24 100  54.5     2    70    99    52    49
## 77  205615970  18  83   6  20  54.5    25     8    96    95    49
## 78  205615971  28  58  98  85  54.5    88    73    83    54    44
## 79  205615972  51  78  37  99  42.0    26    75     9    12     5
## 80  205615973  28  15  37  34  34.0    10    83    74    49    72
## 81  205615974   0  14  27  88  78.0     8    60    18    62     9
## 82  205615975  30  26   9   1  31.0     4    67    74    10    10
## 83  205615976  96  30  56   4  13.0    71    31    19    57    52
## 84  205615977  69  71  93  31 100.0    94    50    26    15    39
## 85  205615978  48  74  13  25   3.0    18    94    24    54    43
## 86  205615979  29  54  60  37  88.0    21    57    96    28    28
## 87  205615980  93  36  90  89  54.5    41    84    44    86    71
## 88  205615981  99   7  10  68  54.5    53    95    67    49    22
## 89  205615982  10  17  25  35  84.0    65    51    41    91    21
## 90  205615983  80   8   6  29  80.0    59    76     1    63    25
## 91  205615984  21  52  37  57   8.0    21    79    33    35    49
## 92  205615985  67  65  48  76  71.0    98    29    46    28    23
## 93  205615986   8   5  15   8  21.0    50    18    81    51    24
## 94  205615987  92  21  51  38  90.0    29   100    23     6    72
## 95  205615988  25  46  12  98  92.0     7    39    70    62    88
## 96  205615989  95  94  49  44  57.0    99    65    23    86    88
## 97  205615990  60   2  59  49  60.0    69    15    87    45    49
## 98  205615991  33  31  91  65  81.0    64    59    47    43    52
## 99  205615992  41  74  27  95  52.0    67    26    74    92    73
## 100 205615993  19  14  44  90  86.0    13    17    17    81    29
```

```
summary(messy_impute(gradebook, "median", 2)[,c(6, 11)])
```

```
##       hw5             quiz5
##  Min.   :  2.00   Min.   : 0.00
##  1st Qu.: 33.50   1st Qu.:27.00
##  Median : 54.50   Median :49.00
##  Mean   : 53.16   Mean   :50.32
##  3rd Qu.: 75.50   3rd Qu.:73.00
##  Max.   :100.00   Max.   :99.00
```

```
messy_impute(gradebook, "mean", 1, trim = 0.25)
```

```
##           UID hw1 hw2 hw3 hw4       hw5 quiz1 quiz2 quiz3 quiz4    quiz5
## 1   205615894  47  63  99  82  74.00000    48     6    70    99 32.00000
## 2   205615895  35  57  37  74  84.00000    63    83    16    31  3.00000
## 3   205615896  64  53   5  14  47.33333    69    88    60    35 38.00000
## 4   205615897  76  81  59  89  36.00000    17    46    76    13 26.00000
## 5   205615898  77  70  66  44  80.00000    72     8     2    31 78.00000
## 6   205615899  86  70  33  17  93.00000    72    50    79    10 93.00000
## 7   205615900  75  88  68  84  83.00000    53    37    90    85 83.00000
## 8   205615901   2  63   1  46  86.00000    59    19    38    78 86.00000
## 9   205615902  87  91  11   8  43.00000    57    72     7    95 74.00000
```

```
## 10  205615903  19   5  81  90  31.00000    76    50     1     9  0.00000
## 11  205615904  55  78  94  91  63.00000    34    40    32     4 98.00000
## 12  205615905  25  20  85  59  56.00000     4    24    29    39 24.00000
## 13  205615906  62  69  42  92  17.00000    19    81    11    73  9.00000
## 14  205615907  50  66  54   8  16.00000    26    24    55    73 22.00000
## 15  205615908  12  93  41  88  53.00000    38    34    10    93 25.00000
## 16  205615909  23  19  85  29  45.00000     9    70    85    45 88.00000
## 17  205615910  75  76  94  16  77.00000    81    91    12    84 67.33333
## 18  205615911  77  11  24  69  37.00000    18    19     4   100 25.00000
## 19  205615912   4  59   1  35  86.00000    82    76    14    59 80.00000
## 20  205615913  14  92  72   4  50.33333    34    44    53    66 74.00000
## 21  205615914  25  28  72  62   4.00000    26    38    90    78 21.00000
## 22  205615915  70  45  93  22  52.00000    38    95    57     1 34.00000
## 23  205615916  50  83  21  75 100.00000    53    62    72     3 38.00000
## 24  205615917  96  65  48  66  68.00000     2    11     8    95 92.00000
## 25  205615918   8  14  68  52  22.00000    51    14    56    48 37.00000
## 26  205615919  69  43  28  15  52.00000    84    29    67    12 49.00000
## 27  205615920 100  91   7  64  40.00000     3    25    98    68 37.00000
## 28  205615921  93  38  23  51  66.00000     7    17    47    41 42.55556
## 29  205615922  70  71  12  67   5.00000    65    31    97    60 48.00000
## 30  205615923  35   4  10  74  61.00000    14    83    70    18 58.00000
## 31  205615924  22  57  13  88  50.00000    79    52    62    61 61.00000
## 32  205615925  73  28  83  86  20.00000    20    34    81    57 99.00000
## 33  205615926  76  29  80  71  54.00000    47    94    71    86 73.00000
## 34  205615927  59  29  39  35  60.00000    25    43    66     8 58.00000
## 35  205615928  31  81  10  37  59.00000    56    73    98    97 63.00000
## 36  205615929  86  59  51  18  68.00000    56    66    96    87 93.00000
## 37  205615930   2  11  78  59  87.00000    42    72    38    29  1.00000
## 38  205615931  20  46  84  15  71.00000    98    24     6    65 84.00000
## 39  205615932  25  98  21  88  32.00000    26    64    18     7 88.00000
## 40  205615933  74  36  83  37  16.00000    78    85    13    33 92.00000
## 41  205615934  82  95  34  82   4.00000     0    87    47    67 27.00000
## 42  205615935  64  25   2  27  21.00000    41    41    63     0 78.00000
## 43  205615936  25  31  92  24  31.00000    62    18    91    83 90.00000
## 44  205615937  35  12 100  71  24.00000    92    32    37    90 42.00000
## 45  205615938  88  58  20  70  65.00000    73    75    31    52 82.00000
## 46  205615939   5  78  82  10  11.00000    31    42    71    54 53.00000
## 47  205615940  96  62  35  47  87.00000    21    65    40    35 67.00000
## 48  205615941  57  13  76  29  41.77778    35    81    46    26 13.00000
## 49  205615942  56  36  94  41  64.00000    92     2    66    11 51.33333
## 50  205615943  25  38   0  73  85.00000    37    72    34    58 47.00000
## 51  205615944   9  40  46 100  67.00000    45    83    23     1 90.00000
## 52  205615945  60  40  18  77  84.00000    82     4    67    87 97.00000
## 53  205615946   1   9  18  55  73.00000    63    69    54    33 34.00000
## 54  205615947  86  76  60  17  48.00000    13     4    14    14 75.00000
## 55  205615948  56  47  72  48  47.00000    64    18    16    90 50.88889
## 56  205615949  26  22 100  98  79.00000    34    58    20    33 35.00000
## 57  205615950  73  79  14  96  83.00000    83    86    52    52 15.00000
## 58  205615951  25   7  31  70  72.00000    41     2     4    61 60.00000
## 59  205615952  99  35   5  88  46.00000    86    94    25     4 14.00000
## 60  205615953  88  33   1  75  75.00000    55    97    88    33 50.00000
## 61  205615954  93  72  26  11   3.00000    99    69    40    88 69.00000
## 62  205615955  45  84  17  21  13.00000    28    88    76    12 43.00000
## 63  205615956  75  42  44  52  72.00000    61    60    91    58 65.00000
```

```
## 64  205615957  93  85  64  54  54.00000    87    49    85    81 25.00000
## 65  205615958  87  52  94  11  53.22222    54     9    55    90 27.00000
## 66  205615959   6  57  31  87  20.00000     4    10    24    21 55.00000
## 67  205615960  74   4  86  96  77.00000    25    60     3    64 41.00000
## 68  205615961  57  56   4  58  55.00000    32    29    73    80 49.33333
## 69  205615962  54  21  28  33  38.00000    51    34    48    52  9.00000
## 70  205615963   7  70  59  53  49.00000     9    57    44    95 74.00000
## 71  205615964  63  97  43  24   5.00000    57    77    53    18 13.00000
## 72  205615965  55   7  35   4  94.00000    79    49    63    64 56.00000
## 73  205615966  24  61  59  12  48.00000    15     4    69    53 48.00000
## 74  205615967  14  43  85   0  25.00000    52    58   100    12 43.22222
## 75  205615968  13  85  45  80   2.00000    40    49     7    77 62.00000
## 76  205615969   5  95  24 100  55.87500     2    70    99    52 55.87500
## 77  205615970  18  83   6  20  44.44444    25     8    96    95 49.00000
## 78  205615971  28  58  98  85  67.88889    88    73    83    54 44.00000
## 79  205615972  51  78  37  99  42.00000    26    75     9    12  5.00000
## 80  205615973  28  15  37  34  34.00000    10    83    74    49 72.00000
## 81  205615974   0  14  27  88  78.00000     8    60    18    62  9.00000
## 82  205615975  30  26   9   1  31.00000     4    67    74    10 10.00000
## 83  205615976  96  30  56   4  13.00000    71    31    19    57 52.00000
## 84  205615977  69  71  93  31 100.00000    94    50    26    15 39.00000
## 85  205615978  48  74  13  25   3.00000    18    94    24    54 43.00000
## 86  205615979  29  54  60  37  88.00000    21    57    96    28 28.00000
## 87  205615980  93  36  90  89  70.44444    41    84    44    86 71.00000
## 88  205615981  99   7  10  68  52.22222    53    95    67    49 22.00000
## 89  205615982  10  17  25  35  84.00000    65    51    41    91 21.00000
## 90  205615983  80   8   6  29  80.00000    59    76     1    63 25.00000
## 91  205615984  21  52  37  57   8.00000    21    79    33    35 38.11111
## 92  205615985  67  65  48  76  71.00000    98    29    46    28 23.00000
## 93  205615986   8   5  15   8  21.00000    50    18    81    51 24.00000
## 94  205615987  92  21  51  38  90.00000    29   100    23     6 72.00000
## 95  205615988  25  46  12  98  92.00000     7    39    70    62 88.00000
## 96  205615989  95  94  49  44  57.00000    99    65    23    86 88.00000
## 97  205615990  60   2  59  49  60.00000    69    15    87    45 49.55556
## 98  205615991  33  31  91  65  81.00000    64    59    47    43 52.00000
## 99  205615992  41  74  27  95  52.00000    67    26    74    92 73.00000
## 100 205615993  19  14  44  90  86.00000    13    17    17    81 29.00000
```

```
summary(messy_impute(gradebook, "mean", 1, trim = 0.25)[,c(6, 11)])
```

```
##       hw5              quiz5
##  Min.   :  2.00   Min.   : 0.00
##  1st Qu.: 33.50   1st Qu.:27.00
##  Median : 54.00   Median :49.17
##  Mean   : 53.23   Mean   :50.27
##  3rd Qu.: 75.50   3rd Qu.:73.00
##  Max.   :100.00   Max.   :99.00
```

```
#Notice how the number of NAs for the three cases became 0 compared to the number of NAs of 10 seen in
```

1(e) Data Format Conversion

Write R code in the RMarkdown to convert the gradebook into the tidy format. Name this object `gradebook_tidy`.

```
library(tidyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1

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

```
# Your code here
gradebook_tidy <- gradebook %>%
  pivot_longer(hw1:quiz5, names_to = "Item", values_to = "Grade") %>%
arrange(Item, Grade) %>% select(UID, Item, Grade)
gradebook_tidy
```

```
## # A tibble: 1,000 x 3
##          UID Item  Grade
##        <int> <chr> <int>
##  1 205615974 hw1       0
##  2 205615946 hw1       1
##  3 205615901 hw1       2
##  4 205615930 hw1       2
##  5 205615912 hw1       4
##  6 205615939 hw1       5
##  7 205615969 hw1       5
##  8 205615959 hw1       6
##  9 205615963 hw1       7
## 10 205615918 hw1       8
## # ... with 990 more rows
```

1(f) `tidy_impute()`

Write a function `tidy_impute()` that imputes missing values in gradebook_tidy object. The `tidy_impute()` function should have the same arguments as in the `messy_impute()` function. You should return an imputed `gradebook_tidy` object. Please also present your algorithm or flowchart to answer this question in the R markdown file.

```
knitr::include_graphics("flowchart_tidy_impute.png")
```

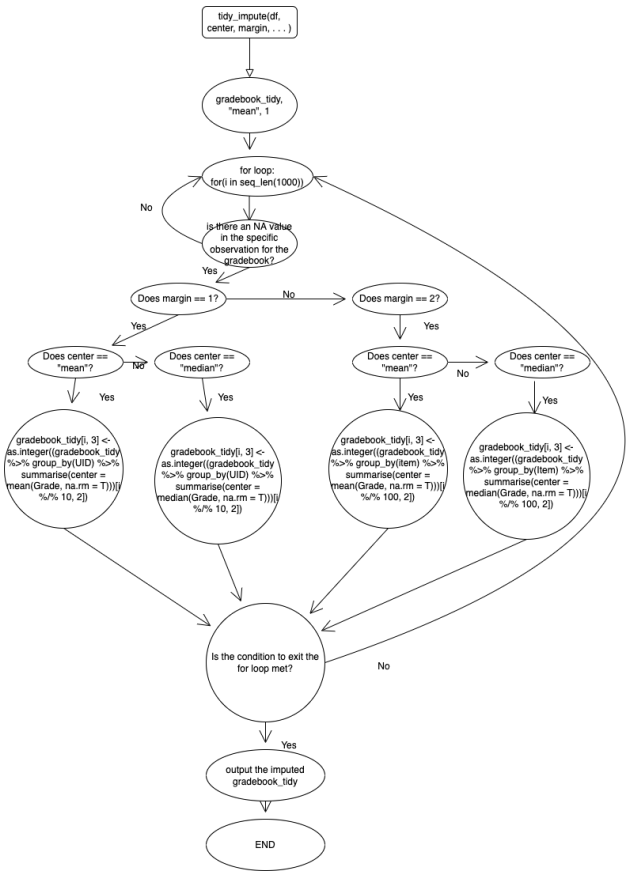

Figure 2: General stucture of a flowchart.

Note: Don't convert `gradebook_tidy` object to be a messy format or reuse your `messy_impute()` in any steps of your `tidy_impute()`.

1(g) Test Cases

Use the cases you select from (d) to demonstrate your function in the RMarkdown file.

```
(tidy_impute(gradebook_tidy, "mean", 1) %>% group_by(UID) %>%
    summarise(na = sum(is.na(Grade))))[1:5, ]
```

```
## # A tibble: 5 x 2
##         UID    na
```

```
##       <int> <int>
## 1 205615894     0
## 2 205615895     0
## 3 205615896     0
## 4 205615897     0
## 5 205615898     0
```

```
tidy_impute(gradebook_tidy, "mean", 1) %>% group_by(Item) %>%
summarise(na = sum(is.na(Grade)))
```

```
## # A tibble: 10 x 2
##    Item     na
##    <chr> <int>
##  1 hw1       0
##  2 hw2       0
##  3 hw3       0
##  4 hw4       0
##  5 hw5       0
##  6 quiz1     0
##  7 quiz2     0
##  8 quiz3     0
##  9 quiz4     0
## 10 quiz5     0
```

```
tidy_impute(gradebook_tidy, "mean", 1)
```

```
## # A tibble: 1,000 x 3
##          UID Item  Grade
##        <int> <chr> <int>
##  1 205615974 hw1       0
##  2 205615946 hw1       1
##  3 205615901 hw1       2
##  4 205615930 hw1       2
##  5 205615912 hw1       4
##  6 205615939 hw1       5
##  7 205615969 hw1       5
##  8 205615959 hw1       6
##  9 205615963 hw1       7
## 10 205615918 hw1       8
## # ... with 990 more rows
```

```
tidy_impute(gradebook_tidy, "median", 2)
```

```
## # A tibble: 1,000 x 3
##          UID Item  Grade
##        <int> <chr> <int>
##  1 205615974 hw1       0
##  2 205615946 hw1       1
##  3 205615901 hw1       2
##  4 205615930 hw1       2
##  5 205615912 hw1       4
##  6 205615939 hw1       5
```

```
##  7 205615969 hw1       5
##  8 205615959 hw1       6
##  9 205615963 hw1       7
## 10 205615918 hw1       8
## # ... with 990 more rows
```

```
tidy_impute(gradebook_tidy, "mean", 1, trim = 0.25)
```

```
## # A tibble: 1,000 x 3
##          UID Item  Grade
##        <int> <chr> <int>
##  1 205615974 hw1       0
##  2 205615946 hw1       1
##  3 205615901 hw1       2
##  4 205615930 hw1       2
##  5 205615912 hw1       4
##  6 205615939 hw1       5
##  7 205615969 hw1       5
##  8 205615959 hw1       6
##  9 205615963 hw1       7
## 10 205615918 hw1       8
## # ... with 990 more rows
```

2 Short Answer

The formatting takes 10% of your homework grade. The accuracy takes 15% of your grade.

2().1 (a)

Please find three examples in which data might naturally be presented in a messy (non-tidy) way. For each example, include the context, observations, and variables that would be recorded. Showcase a small sample, say 10 observations, for each example. You may search online for context, but you **MUST** cite your sources. Please don't make up data on your own.

```
library(datasets)
```

Answer:

Data set 1: USJudgeRatings

```
head(USJudgeRatings, 10)
```

```
##                CONT INTG DMNR DILG CFMG DECI PREP FAMI ORAL WRIT PHYS RTEN
## AARONSON,L.H.   5.7  7.9  7.7  7.3  7.1  7.4  7.1  7.1  7.1  7.0  8.3  7.8
## ALEXANDER,J.M.  6.8  8.9  8.8  8.5  7.8  8.1  8.0  8.0  7.8  7.9  8.5  8.7
## ARMENTANO,A.J.  7.2  8.1  7.8  7.8  7.5  7.6  7.5  7.5  7.3  7.4  7.9  7.8
## BERDON,R.I.     6.8  8.8  8.5  8.8  8.3  8.5  8.7  8.7  8.4  8.5  8.8  8.7
## BRACKEN,J.J.    7.3  6.4  4.3  6.5  6.0  6.2  5.7  5.7  5.1  5.3  5.5  4.8
## BURNS,E.B.      6.2  8.8  8.7  8.5  7.9  8.0  8.1  8.0  8.0  8.0  8.6  8.6
## CALLAHAN,R.J.  10.6  9.0  8.9  8.7  8.5  8.5  8.5  8.5  8.6  8.4  9.1  9.0
```

```
## COHEN,S.S.      7.0  5.9  4.9  5.1  5.4  5.9  4.8  5.1  4.7  4.9  6.8  5.0
## DALY,J.J.       7.3  8.9  8.9  8.7  8.6  8.5  8.4  8.4  8.4  8.5  8.8  8.8
## DANNEHY,J.F.    8.2  7.9  6.7  8.1  7.9  8.0  7.9  8.1  7.7  7.8  8.5  7.9
```

```
?USJudgeRatings
```

This data set provides the Lawyers' ratings of state judges who are in the US Superior Court in 1977. The variable that would be recorded in this data set is the Name, Rating type and the scores.
This data set is non-tidy or messy because the variables are not in its own column. This data set has its column associated with the specific rating types such as CONT which indicates the number of contracts of lawyer with judge, INTG which indicates the judicial integrity, and ETC. Source : New Haven Register, 14 January, 1977 (from John Hartigan).

Data set 2: USArrests

```
head(USArrests, 10)
```

```
##             Murder Assault UrbanPop Rape
## Alabama       13.2     236       58 21.2
## Alaska        10.0     263       48 44.5
## Arizona        8.1     294       80 31.0
## Arkansas       8.8     190       50 19.5
## California     9.0     276       91 40.6
## Colorado       7.9     204       78 38.7
## Connecticut    3.3     110       77 11.1
## Delaware       5.9     238       72 15.8
## Florida       15.4     335       80 31.9
## Georgia       17.4     211       60 25.8
```

```
?USArrests
```

This data set illustrates the statistics for the in arrests per 100,000 residents in the 50 states of United States of America in year 1973. The crimes that are listed in the data set are assault, murder, and rape. The variable that would be recorded in this data set are states, population per state, the type of crime, and the arrests per 100,000 people. This data set is non-tidy or messy because the variables are not in its own column. As we can see above, the column names include the type of crimes such as murder, assault, and rape instead of the actual variable which is the crime type. Source: World Almanac and Book of facts 1975. (Crime rates).

Statistical Abstracts of the United States 1975, p.20, (Urban rates), possibly available as https://books.google.ch/books?id=zl9qAAAAMAAJ&pg=PA20.

Data set 3: relig_income

```
head(relig_income, 10)
```

```
## # A tibble: 10 x 11
##    religion `<$10k` `$10-20k` `$20-30k` `$30-40k` `$40-50k` `$50-75k` `$75-100k`
##    <chr>      <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>      <dbl>
##  1 Agnostic      27        34        60        81        76       137        122
##  2 Atheist       12        27        37        52        35        70         73
##  3 Buddhist      27        21        30        34        33        58         62
##  4 Catholic     418       617       732       670       638      1116        949
```

```
##  5 Don't k~      15        14        15        11        10        35        21
##  6 Evangel~     575       869      1064       982       881      1486       949
##  7 Hindu          1         9         7         9        11        34        47
##  8 Histori~     228       244       236       238       197       223       131
##  9 Jehovah~      20        27        24        24        21        30        15
## 10 Jewish        19        19        25        25        30        95        69
## # ... with 3 more variables: $100-150k <dbl>, >150k <dbl>,
## #   Don't know/refused <dbl>
```

```
?relig_income
```

This data set was generated from the pew religion and income survey which asked the participants their income and religion. The variables in this data set is the religion, income, and the frequency for the given religion and income. This data-set is a messy (non-tidy) data set because the variables are not in its own column. In this given data set, the frequency is the inner cells and the income intervals are the column names, but in reality, these two should be switched. Source: New Haven Register, 14 January, 1977 (from John Hartigan).

2().2 (b)

For each of the three examples in (a), describe how the data might be better presented in a tidy way. Please use tidyverse functions to reorganize the small sample datasets in (a) into tidy format.

Answer:

Data set 1: USJudgeRatings

As we discussed in part a, the variables are not in its own columns. To convert the data set into a more tidy way, we must fix this.

```
USJudgeRatings_tidy <- USJudgeRatings %>% mutate(Name = rownames(USJudgeRatings)) %>% gather(`CONT`, `I
head(USJudgeRatings_tidy)
```

```
##             Name rating score
## 1  AARONSON,L.H.   CONT   5.7
## 2 ALEXANDER,J.M.   CONT   6.8
## 3 ARMENTANO,A.J.   CONT   7.2
## 4    BERDON,R.I.   CONT   6.8
## 5   BRACKEN,J.J.   CONT   7.3
## 6     BURNS,E.B.   CONT   6.2
```

```
head(USJudgeRatings)
```

```
##                CONT INTG DMNR DILG CFMG DECI PREP FAMI ORAL WRIT PHYS RTEN
## AARONSON,L.H.   5.7  7.9  7.7  7.3  7.1  7.4  7.1  7.1  7.1  7.0  8.3  7.8
## ALEXANDER,J.M.  6.8  8.9  8.8  8.5  7.8  8.1  8.0  8.0  7.8  7.9  8.5  8.7
## ARMENTANO,A.J.  7.2  8.1  7.8  7.8  7.5  7.6  7.5  7.5  7.3  7.4  7.9  7.8
## BERDON,R.I.     6.8  8.8  8.5  8.8  8.3  8.5  8.7  8.7  8.4  8.5  8.8  8.7
## BRACKEN,J.J.    7.3  6.4  4.3  6.5  6.0  6.2  5.7  5.7  5.1  5.3  5.5  4.8
## BURNS,E.B.      6.2  8.8  8.7  8.5  7.9  8.0  8.1  8.0  8.0  8.0  8.6  8.6
```

Now in this newly modified data set called USJudgeRatings_tidy, we can see that all of the columns are unique, distinct variables. We used the mutate function to first add a column for the names of the state

judges since in this data set, the names were not built in. Then we used the gather function to gather together the variable of the ratings which was spread across the columns in the USJudgeRatings dataset.

Data set 2: USArrests

As we discussed in part a, the variables are not in its own columns. To convert the data set into a more tidy way, we must fix this.

```
USArrests_tidy <- USArrests %>% mutate(State = rownames(USArrests)) %>% gather(`Murder`, `Assault`, `Ra

USArrests_tidy <- USArrests_tidy[, c(2, 1, 3, 4)]
head(USArrests_tidy)
```

```
##        State UrbanPop Crime Type Frequency (in 100,000 ppl)
## 1    Alabama       58     Murder                       13.2
## 2     Alaska       48     Murder                       10.0
## 3    Arizona       80     Murder                        8.1
## 4   Arkansas       50     Murder                        8.8
## 5 California       91     Murder                        9.0
## 6   Colorado       78     Murder                        7.9
```

Again, the newly modified data set called USArrests_tidy showcases the states, population for the corresponding state, four different crime types, and the frequency for the different crime types in a tidy manner. I utilized the mutate function again because the state name were not listed as a variable in the original data frame. Next, I used the gather functions since the crime type was spread along the columns of the data frame.

Data set 3: relig_income

As we discussed in part a, the variables are not in its own columns. To convert the data set into a more tidy way, we must fix this.

```
relig_income_tidy <- relig_income %>% pivot_longer(`<$10k`:`Don't know/refused`, names_to = "income", va
head(relig_income)
```

```
## # A tibble: 6 x 11
##   religion  `<$10k` `$10-20k` `$20-30k` `$30-40k` `$40-50k` `$50-75k` `$75-100k`
##   <chr>       <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>      <dbl>
## 1 Agnostic       27        34        60        81        76       137        122
## 2 Atheist        12        27        37        52        35        70         73
## 3 Buddhist       27        21        30        34        33        58         62
## 4 Catholic      418       617       732       670       638      1116        949
## 5 Don't kn~      15        14        15        11        10        35         21
## 6 Evangeli~     575       869      1064       982       881      1486        949
## # ... with 3 more variables: $100-150k <dbl>, >150k <dbl>,
## #   Don't know/refused <dbl>
```

```
head(relig_income_tidy)
```

```
## # A tibble: 6 x 3
##   religion income  frequency
##   <chr>    <chr>       <dbl>
## 1 Agnostic <$10k          27
## 2 Agnostic $10-20k        34
```

```
## 3 Agnostic $20-30k        60
## 4 Agnostic $30-40k        81
## 5 Agnostic $40-50k        76
## 6 Agnostic $50-75k       137
```

Again, the newly modified data set called relig_income_tidy provides a tidy visualization of the variables: religion, income, and frequency. Unlike the old data set in which the intervals for the income was categorized in the column names, in this new data set, we can all see it in each cells of the data frame as an observation. I used the pivot_longer function, which is similar to the gather function, but in this case the name of the column were complex such as <$10k, so instead of writing out all of the column names into the gather function, I gathered together all of the categories of the variable income into one column using pivot_longer function.