Data Management and Analysis Workshop: Part 1

Dataset assembly

We will walk through some basic data cleaning, analysis, and visualization tasks using per pupil expenditures as a running example. The repository contains all the data and code we need, so you can just follow along! This notebook introduces data cleaning and dataset assembly.

Load libraries

The tidyverse is a popular collection of R libraries for data science. They provide functions and data structures that extend what is available in base R, i.e., out-of-the-box. Most data wrangling tasks use dplyr and tidyr packages, and most data visualization is with ggplot2. For details, features, and examples not covered in this notebook, please see the documentation here.

```
# Install required packages.
req <- c("tidyverse", "openxlsx")
new <- req[!(req %in% installed.packages()[, "Package"])]
if (length(new)) install.packages(new)

# Load required packages.
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(tidyr)
library(openxlsx)
```

Load data

Keep a simple inventory of the files that the program reads and writes at the top of the script. This will orient anyone who reads the code to its requirements and results. Everything the program needs should be contained in the repository.

Set your working directory to wherever you downloaded this repository. This tells R the folder in which to look for the data and save our results. Please see the R Studio documentation here for how to set the working directory.

```
# Identify inputs and outputs.
PWD <- getwd()
FIS <- file.path(PWD, "in", "sdf21_1a.txt")
MEM <- file.path(PWD, "in", "ccd_lea_052_2021_l_1a_080621.csv")
DTA <- file.path(PWD, "out", "data.rda")
XLS <- file.path(PWD, "out", "data.xlsx")
```

```
# Load and peek at the fiscal data.
# Need to use the function that corresponds to the file type, e.g., txt.
fis <- read.delim(FIS)
head(fis)
```

```
    LEAID       CENSUSID FIPST CONUM CSA  CBSA                    NAME  STNAME
1 0100002              N     1  1101 388 33860 Alabama Youth Services Alabama
2 0100005 01504840100000     1  1095   N 10700       Albertville City Alabama
3 0100006 01504800100000     1  1095   N 10700        Marshall County Alabama
4 0100007 01503740100000     1  1073 142 13820            Hoover City Alabama
5 0100008 01504530100000     1  1089 290 26620           Madison City Alabama
6 0100009              N     1  1121 142 45180 Al Inst Deaf And Blind Alabama
  STABBR SCHLEV AGCHRT YEAR CCDNF CENFILE GSLO GSHI   V33 MEMBERSCH  TOTALREV
1     AL      N      N   21     1       0   KG   12    -2        -2        -1
2     AL     03      3   21     1       1   PK   12  5842      5842  74420000
3     AL     03      3   21     1       1   PK   12  5758      5758  72199000
4     AL     03      3   21     1       1   PK   12 13640     13640 202260000
```

```
5     AL     03      3   21     1       1   PK   12 11804     11804 169202000
6     AL      N      3   21     1       0   KG   12    -2        -1        -2
   TFEDREV     C14      C15    C19    C22 C23    C26 C27    B11      C20      C25
1       -1      -1       -1     -1     -1  -1     -1  -1     -1       -1       -1
2 10327000 2502000 1255000  83000  28000   0      0   0 177000 2374000 3904000
3 13429000 3294000 1155000 203000 402000   0 191000   0  70000 5138000 2976000
4 12871000  929000 2659000 129000 315000   0      0   0 165000 3024000 5574000
5 16719000  718000 2133000 100000 206000   0      0   0  69000 7762000 4904000
6       -2      -2       -2     -2     -2  -2     -2  -2     -2       -2       -2
  C36    B10 B12 B14    B13   TSTREV      C01   C04   C05    C06    C07 C08 C09
1  -1     -1  -1  -1     -1       -1       -1    -1    -1     -1     -1  -1  -1
2   0      0   0   0   4000 47611000 32259000 73000     0 465000 691000   0   0
3   0      0   0   0      0 43954000 32405000 91000 10000 554000 308000   0   0
4   0      0   0   0  76000 84578000 66992000 93000     0 934000 331000   0   0
5   0 690000   0   0 137000 85046000 62305000 97000     0 692000 187000   0   0
6  -2     -2  -2  -2     -2       -2       -2    -2    -2     -2     -2  -2  -2
  C10      C11      C12       C13 C35 C38 C39   TLOCREV T02       T06 T09 T15 T40
1  -1       -1       -1        -1  -1  -1  -1        -1  -2        -2  -2  -2  -2
2   0 1640000 2466000 10017000   0   0   0  16482000  -2  6318000   0   0   0
3   0 1932000 3804000  4850000   0   0   0  14816000  -2  7704000   0   0   0
4   0 3440000 6564000  6224000   0   0   0 104811000  -2 84745000   0   0   0
5   0 3196000 4175000 14394000   0   0   0  67437000  -2 34879000   0   0   0
6  -2      -2       -2        -2  -2  -2  -2        -2  -2        -2  -2  -2  -2
     T99    D11       D23   A07 A08     A09  A11      A13 A15    A20    A40
1     -2     -1        -1    -1  -1      -1   -1       -1  -1     -1     -1
2 445000   3000  6476000 73000   0  107000    0 1492000   0 251000      0
3  10000 169000  2539000  1000   0   57000    0 1048000   0 191000      0
4   6000      0  9068000 19000   0 1106000 3000 4106000   0 858000 181000
5   6000      0 23200000 56000   0  101000    0 2639000   0 333000 634000
6     -2     -2       -2    -2  -2      -2   -2       -2  -2     -2     -2
      U11    U22    U30    U50     U97 C24  TOTALEXP  TCURELSC  TCURINST
1      -1     -1     -1     -1      -1  -1        -1        -1        -1
2   91000 196000   1000 135000  894000   0  59207000  54703000  32229000
3  309000  97000  14000 498000 2179000   0  68866000  64386000  35942000
4 1779000 881000  11000 404000 1644000   0 192421000 165287000 104288000
5 1629000 927000 140000 444000 2449000   0 184180000 125095000  74804000
6      -2     -2     -2     -2      -2  -2        -2        -2        -2
        E13    V91 V92 TCURSSVC      E17      E07     E08      E09      V40
1        -1     -1  -1       -1       -1       -1      -1       -1       -1
2  32229000      0   0 18679000  3696000  1603000 1317000  2780000  5483000
3  35942000      0   0 24244000  4258000  2267000 2287000  3944000  6261000
4 104288000  36000   0 59067000 12404000  6670000 1083000 10931000 15516000
5  74804000 129000   0 49055000  9954000 14884000 2248000  6206000  9241000
```

```
6        -2     -2  -2       -2       -2       -2      -2       -2       -2
      V45     V90 V85 TCUROTH     E11 V60 V65 TNONELSE     V70    V75     V80
1      -1      -1  -1      -1      -1  -1  -1       -1      -1     -1      -1
2 2610000 1190000   0 3795000 3795000   0   0   486000    3000   1000  482000
3 3909000 1318000   0 4200000 4200000   0   0   836000  191000 175000  470000
4 7721000 4742000   0 1932000 1932000   0   0  8516000 7068000 109000 1339000
5 4491000 2031000   0 1236000 1236000   0   0  5246000 4903000      0  343000
6      -2      -2  -2      -2      -2  -2  -2       -2      -2     -2      -2
   TCAPOUT      F12     G15    K09     K10 K11 L12 M12   Q11     I86       Z32
1       -1       -1      -1     -1      -1  -1  -1  -1    -1      -1        -1
2  2499000  1377000  585000  28000  509000   0   0   0 59000 1460000  28529000
3  3079000  1229000 1594000  48000  208000   0   0   0 10000  555000  34640000
4 12558000 10539000       0 104000 1915000   0   0   0 26000 5998000 103647000
5 46681000 46297000   10000 146000  228000   0   0   0 89000 6940000  72579000
6       -2       -2      -2     -2      -2  -2  -2  -2    -2      -2        -2
       Z33      Z35     Z36     Z37     Z38     V11     V13     V15     V17
1       -1       -1      -1      -1      -1      -1      -1      -1      -1
2 18798000 14280000  897000 1247000  582000 2264000  920000  616000 1968000
3 20894000 13819000 2027000 1315000 1410000 2657000 1273000  993000 2815000
4 68668000 52267000 5448000 1846000 1421000 7327000 4287000  543000 7921000
5 48430000 34352000 3515000 3091000 1755000 5991000 4269000 1428000 4466000
6       -2       -2      -2      -2      -2      -2      -2      -2      -2
      V21     V23     V37     V29      Z34      V10     V12     V14    V16
1      -1      -1      -1      -1       -1       -1      -1      -1     -1
2  873000 1385000  713000  990000 11016000  6887000  826000  329000 181000
3 1643000 2212000  545000 1410000 13492000  7798000  946000  402000 312000
4 5413000 4185000 1763000  557000 38788000 23716000 2221000 1431000 141000
5 2254000 2287000 1238000  131000 26722000 17502000 1891000 1489000 348000
6      -2      -2      -2      -2       -2       -2      -2      -2     -2
      V18     V22     V24     V38    V30 V32     V93     X_19H   X_21F   X_31F
1      -1      -1      -1      -1     -1  -1      -1        -1      -1      -1
2  714000  395000  904000  210000 571000   0  443000  25505000 1045000 3035000
3 1025000  706000 1259000  184000 807000   0  957000  24582000       0  766000
4 2668000 2429000 2485000 1999000 273000   0 2199000 158255000       0 6940000
5 1558000 1067000 1368000  436000  84000   0 1612000 140823000       0 5721000
6      -2      -2      -2      -2     -2  -2      -2        -2      -2      -2
      X_41F X_61V X_66V      W01      W31       W61     V95     V02 K14
1        -1    -1    -1       -1       -1        -1      -1      -1  -1
2  23516000     0     0 15440000  8566000  14950000 1081000 1829000   0
3  23816000     0     0  2248000  1721000   3616000 1416000 2592000   0
4 151315000     0     0        0  5318000 122652000 3973000 3318000   0
5 135102000     0     0 31491000 72517000  16152000 2714000 9047000   0
6        -2    -2    -2       -2       -2        -2      -2      -2  -2
```

```
        CE1       CE2 CE3       SE1       SE2      SE3    SE4      SE5    AR1
1        -1        -1  -1        -1        -1       -1     -1       -1     -1
2  42239000   9989000   0   2792000   2058000   397000      0   191000 891000
3  47566000  14331000   0   7073000   4278000  1189000  10000   682000 903000
4 150163000   9097000   0  19592000  13719000  4057000   5000  1671000 281000
5 103978000  11702000   0  11012000   7461000  1913000 410000   919000  87000
6        -2        -2  -2        -2        -2       -2     -2       -2     -2
     AR1A     AR1B     AR2 AR2A AR3     AR6 AR6A      AE1     AE2     AE3   AE4
1      -1       -1      -1   -1  -1      -1   -1       -1      -1      -1    -1
2  189000        0  110000    0   0  511000    0  1702000  418000 1264000     0
3 2405000        0  410000    0   0  765000    0  4492000 3373000 1070000 30000
4 1281000   272000  272000    0   0  582000    0  2493000 1479000 1008000     0
5 1173000  3225000  521000    0   0 1699000    0  6251000 1063000 5021000     0
6      -2       -2      -2   -2  -2      -2   -2       -2      -2      -2    -2
      AE5 AE6     AE7 AE8 WEIGHT FL_V33 FL_MEMBERSCH FL_C14 FL_C15 FL_C19
1      -1  -1      -1  -1      1      N            N      M      M      M
2    9000   0 1004000   0       1      R            R      R      R      R
3 1055000   0  414000   0       1      R            R      R      R      R
4  268000   0   16000   0       1      R            R      R      R      R
5 4628000   0   40000   0       1      R            R      R      R      R
6      -2  -2      -2  -2      1      N            M      N      N      N
  FL_C22 FL_C23 FL_C26 FL_C27 FL_B11 FL_C20 FL_C25 FL_C36 FL_B10 FL_B12 FL_B14
1      M      M      M      M      M      M      M      M      M      M      M
2      R      R      R      M      R      R      R      R      R      R      M
3      R      R      R      M      R      R      R      R      R      R      M
4      R      R      R      M      R      R      R      R      R      R      M
5      R      R      R      M      R      R      R      R      R      R      M
6      N      N      N      N      N      N      N      N      N      N      N
  FL_B13 FL_C01 FL_C04 FL_C05 FL_C06 FL_C07 FL_C08 FL_C09 FL_C10 FL_C11 FL_C12
1      M      M      M      M      M      M      M      M      M      M      M
2      R      R      R      R      R      R      M      M      M      R      R
3      R      R      R      R      R      R      M      M      M      R      R
4      R      R      R      R      R      R      M      M      M      R      R
5      R      R      R      R      R      R      M      M      M      R      R
6      N      N      N      N      N      N      N      N      N      N      N
  FL_C13 FL_C35 FL_C38 FL_C39 FL_T02 FL_T06 FL_T09 FL_T15 FL_T40 FL_T99 FL_D11
1      M      M      M      M      N      N      N      N      N      N      M
2      R      R      M      M      N      R      M      M      M      R      R
3      R      R      M      M      N      R      M      M      M      R      R
4      R      R      M      M      N      R      M      M      M      R      R
5      R      R      M      M      N      R      M      M      M      R      R
6      N      N      N      N      N      N      N      N      N      N      N
  FL_D23 FL_A07 FL_A08 FL_A09 FL_A11 FL_A13 FL_A15 FL_A20 FL_A40 FL_U11 FL_U22
```

1	M	M	M	M	M	M	M	M	M	M	M
2	R	R	M	R	R	R	R	R	R	R	R
3	R	R	M	R	R	R	R	R	R	R	R
4	R	R	M	R	R	R	R	R	R	R	R
5	R	R	M	R	R	R	R	R	R	R	R
6	N	N	N	N	N	N	N	N	N	N	N

	FL_U30	FL_U50	FL_U97	FL_C24	FL_E13	FL_V91	FL_V92	FL_E17	FL_E07	FL_E08	FL_E09
1	M	M	M	M	M	M	M	M	M	M	M
2	R	R	R	R	R	R	M	R	R	R	R
3	R	R	R	R	R	R	M	R	R	R	R
4	R	R	R	R	R	R	M	R	R	R	R
5	R	R	R	R	R	R	M	R	R	R	R
6	N	N	N	N	N	N	N	N	N	N	N

	FL_V40	FL_V45	FL_V90	FL_V85	FL_E11	FL_V60	FL_V65	FL_V70	FL_V75	FL_V80	FL_F12
1	M	M	M	M	M	M	M	M	M	M	M
2	R	R	R	R	R	M	M	R	R	R	R
3	R	R	R	R	R	M	M	R	R	R	R
4	R	R	R	R	R	M	M	R	R	R	R
5	R	R	R	R	R	M	M	R	R	R	R
6	N	N	N	N	N	N	N	N	N	N	N

	FL_G15	FL_K09	FL_K10	FL_K11	FL_L12	FL_M12	FL_Q11	FL_I86	FL_Z32	FL_Z33	FL_Z35
1	M	M	M	M	M	M	M	M	M	M	M
2	R	R	R	R	M	M	R	R	R	R	R
3	R	R	R	R	M	M	R	R	R	R	R
4	R	R	R	R	M	M	R	R	R	R	R
5	R	R	R	R	M	M	R	R	R	R	R
6	N	N	N	N	N	N	N	N	N	N	N

	FL_Z36	FL_Z37	FL_Z38	FL_V11	FL_V13	FL_V15	FL_V17	FL_V21	FL_V23	FL_V37	FL_V29
1	M	M	M	M	M	M	M	M	M	M	M
2	R	R	R	R	R	R	R	R	R	R	R
3	R	R	R	R	R	R	R	R	R	R	R
4	R	R	R	R	R	R	R	R	R	R	R
5	R	R	R	R	R	R	R	R	R	R	R
6	N	N	N	N	N	N	N	N	N	N	N

	FL_Z34	FL_V10	FL_V12	FL_V14	FL_V16	FL_V18	FL_V22	FL_V24	FL_V38	FL_V30	FL_V32
1	M	M	M	M	M	M	M	M	M	M	M
2	R	R	R	R	R	R	R	R	R	R	M
3	R	R	R	R	R	R	R	R	R	R	M
4	R	R	R	R	R	R	R	R	R	R	M
5	R	R	R	R	R	R	R	R	R	R	M
6	N	N	N	N	N	N	N	N	N	N	N

	FL_V93	FL_19H	FL_21F	FL_31F	FL_41F	FL_61V	FL_66V	FL_W01	FL_W31	FL_W61	FL_V95
1	M	M	M	M	M	M	M	M	M	M	M

```
2      R      A      R      R      I      M      M      R      R      R      R
3      R      A      R      R      I      M      M      R      R      R      R
4      R      A      R      R      I      M      M      R      R      R      R
5      R      A      R      R      I      M      M      R      R      R      R
6      N      N      N      N      N      N      N      N      N      N      N
  FL_V02 FL_K14 FL_CE1 FL_CE2 FL_CE3 FL_SE1 FL_SE2 FL_SE3 FL_SE4 FL_SE5 FL_AR1
1      M      M      M      M      M      M      M      M      M      M      M
2      R      R      R      R      M      R      R      R      R      R      R
3      R      R      R      R      M      R      R      R      R      R      R
4      R      R      R      R      M      R      R      R      R      R      R
5      R      R      R      R      M      R      R      R      R      R      R
6      N      N      N      N      N      N      N      N      N      N      N
  FL_AR1A FL_AR1B FL_AR2 FL_AR2A FL_AR3 FL_AR6 FL_AR6A FL_AE1 FL_AE2 FL_AE3
1       M       M      M       M      M      M       M      M      M      M
2       R       R      R       M      M      R       M      R      R      R
3       R       R      R       M      M      R       M      R      R      R
4       R       R      R       M      M      R       M      R      R      R
5       R       R      R       M      M      R       M      R      R      R
6       N       N      N       N      N      N       N      N      N      N
  FL_AE4 FL_AE5 FL_AE6 FL_AE7 FL_AE8
1      M      M      M      M      M
2      R      R      R      R      R
3      R      R      R      R      R
4      R      R      R      R      R
5      R      R      R      R      R
6      N      N      N      N      N
```

```
# These are some helpful functions for accessing information about the dataset.
print(paste("Number of rows:", nrow(fis)))
```

```
[1] "Number of rows: 19554"
```

```
print(paste("Number of columns:", ncol(fis)))
```

```
[1] "Number of columns: 310"
```

Check unique identifier

We think the fiscal data contain one observation for each school district. This is an important assumption that would compromise our analysis and break our code were it to fail. For

example, there may be a more complicated nesting structure or duplicates that we'd need to handle first. Let's test this.

```
# Ensure data are at the school district level.
leaid <- unique(fis$LEAID)
stopifnot(nrow(fis) == length(leaid))
```

Technical note. The code makes extensive use of the `stopifnot()` function as a defensive programming technique. It takes an expression that must evaluate to `TRUE`, and otherwise throws an error and stops the program. This helps document and verify assumptions of the data.

Handle missing data

The columns `MEMBERSCH` and `TOTALEXP` describe total number of students and total expenditures, respectively. We can see that they contain some negative values, which is impossible! Reading the documentation, we learn that these values mark different kinds of missingness. We need to recode these as `NA` values to correctly analyze the data.

```
# Summarize students and expenditures.
summary(fis[, c("MEMBERSCH", "TOTALEXP")])
```

```
   MEMBERSCH          TOTALEXP
 Min.   :    -9   Min.   :-2.000e+00
 1st Qu.:   199   1st Qu.: 3.676e+06
 Median :   613   Median : 1.029e+07
 Mean   :  2500   Mean   : 4.303e+07
 3rd Qu.:  1808   3rd Qu.: 3.171e+07
 Max.   :894493   Max.   : 3.420e+10
```

```
# Tabulate negative values in students.
table(fis$MEMBERSCH[fis$MEMBERSCH < 0])
```

```
  -9   -3   -2   -1
  65  465 1032   71
```

```
# Tabulate negative values in expenditures.
table(fis$TOTALEXP[fis$TOTALEXP < 0])
```

```
 -2  -1
897 491
```

```r
# Recode missing values to NA.
fis <- mutate(fis, across(c("MEMBERSCH", "TOTALEXP"), ~ replace(.x, .x < 0, NA)))
```

```r
# Summarize students and expenditures again.
summary(fis[, c("MEMBERSCH", "TOTALEXP")])
```

```
   MEMBERSCH          TOTALEXP
 Min.   :     0   Min.   :0.000e+00
 1st Qu.:   280   1st Qu.:4.694e+06
 Median :   716   Median :1.192e+07
 Mean   :  2728   Mean   :4.632e+07
 3rd Qu.:  2027   3rd Qu.:3.491e+07
 Max.   :894493   Max.   :3.420e+10
 NA's   :1633     NA's   :1388
```

Technical note. We used base R grammar to filter for missing values and tidyverse grammar to replace them. Both are correct, however sometimes one is more concise than the other. Like with any writing, aim for clear and simple code.

Generate new measures

Now that we have a numerator and denominator, we can calculate a rate. In statistics, we call this defining a variable. But in the tidyverse, we call this creating a column. Note that we have some school districts with zero students, so we will use a conditional statement to generate these values where we can.

```r
# Calculate per pupil expenditure.
fis <- mutate(
  fis,
  ppe = case_when(
    MEMBERSCH > 0 ~ TOTALEXP / MEMBERSCH,
    .default = NA
  )
)
```

```r
# Summarize per pupil expenditure.
summary(fis$ppe)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
      0   12503   15525   25246   21132 7236000    2239
```

Transform data

We're also interested in how per pupil expenditure varies by majority race, but that information is not in the fiscal data. For that, we'll need to get the membership data.

```r
# Load and peek at the membership data.
# Need to use the function that corresponds to the file type, e.g., csv.
mem <- read.csv(MEM)
head(mem)
```

```
  SCHOOL_YEAR FIPST STATENAME ST               LEA_NAME STATE_AGENCY_NO UNION
1   2020-2021     1   ALABAMA AL Alabama Youth Services               1    NA
2   2020-2021     1   ALABAMA AL Alabama Youth Services               1    NA
3   2020-2021     1   ALABAMA AL Alabama Youth Services               1    NA
4   2020-2021     1   ALABAMA AL Alabama Youth Services               1    NA
5   2020-2021     1   ALABAMA AL Alabama Youth Services               1    NA
6   2020-2021     1   ALABAMA AL Alabama Youth Services               1    NA
  ST_LEAID  LEAID   GRADE                   RACE_ETHNICITY    SEX STUDENT_COUNT
1   AL-210 100002 Grade 1 American Indian or Alaska Native Female            NA
2   AL-210 100002 Grade 1 American Indian or Alaska Native   Male            NA
3   AL-210 100002 Grade 1                            Asian Female            NA
4   AL-210 100002 Grade 1                            Asian   Male            NA
5   AL-210 100002 Grade 1        Black or African American Female            NA
6   AL-210 100002 Grade 1        Black or African American   Male            NA
                                 TOTAL_INDICATOR     DMS_FLAG
1 Category Set A - By Race/Ethnicity; Sex; Grade Not reported
2 Category Set A - By Race/Ethnicity; Sex; Grade Not reported
3 Category Set A - By Race/Ethnicity; Sex; Grade Not reported
4 Category Set A - By Race/Ethnicity; Sex; Grade Not reported
5 Category Set A - By Race/Ethnicity; Sex; Grade Not reported
6 Category Set A - By Race/Ethnicity; Sex; Grade Not reported
```

```r
# Note data are not at the school district level!
leaid <- unique(mem$LEAID)
stopifnot(nrow(mem) > length(leaid))
```

Inspecting the data and reading the documentation, we learn that the data contain multiple observations per school district to disaggregate membership by grade, race, and gender. We want to aggregate and reshape the data to the school district level with additional columns for total membership by race.

```
# Write a function to check uniqueness of identifiers.
# Use this in a chain of tidyverse functions to check your work.
isid <- function(.data, ...) {
  if(any(duplicated(dplyr::select(.data, ...)))) {
    stop("indexers do not uniquely identify the observations")
  }
  return(.data)
}
```

```
# Aggregate and reshape to the school district level.
lea <- mem |>
  # Select observations for the race-gender level.
  filter(TOTAL_INDICATOR == "Derived - Subtotal by Race/Ethnicity and Sex minus Adult Educati
  # Consolidate race/ethnicity categories with shorthand labels.
  mutate(race = case_match(
    RACE_ETHNICITY,
    "Asian"                                     ~ "asian",
    "Black or African American"                 ~ "black",
    "Hispanic/Latino"                           ~ "hisp",
    "White"                                     ~ "white",
    c("American Indian or Alaska Native",
      "Native Hawaiian or Other Pacific Islander",
      "Two or more races",
      "Not Specified")                          ~ "other"
  )) |>
  # Sum membership by race and overall within district.
  group_by(LEAID, race) |>
  summarize(n = sum(STUDENT_COUNT, na.rm = TRUE)) |>
  isid(LEAID, race) |>
  mutate(tot = sum(n)) |>
  ungroup() |>
  # Reshape to district level.
  pivot_wider(names_from = race, values_from = n) |>
  isid(LEAID) |>
  # Calculate group shares.
  rowwise() |>
  mutate(across(
    c(asian, black, hisp, white, other),
```

```
    ~ .x / tot * 100,
    .names = "pct_{.col}"
  )) |>
  # Create indicators on majority status.
  mutate(across(
    c(asian, black, hisp, white, other),
    ~ if_else(tot == 0, FALSE, .x / tot > 0.5),
    .names = "maj_{.col}"
    )
  ) |>
  ungroup() |>
  # Identify majority group.
  mutate(maj_group = case_when(
    maj_asian ~ "asian",
    maj_black ~ "black",
    maj_hisp  ~ "hisp",
    maj_white ~ "white",
    maj_other ~ "other",
    tot > 0   ~ "none"
  ))
```

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

Technical note. The last example uses the pipe operator `|>` to pass (a) the output of the previous function to (b) the input of the next function. This idiom allows chaining of discrete operations into an easily readable sequence of operations.

Merge data

With both datasets cleaned, we can bring them together to assemble our analysis file. Specifically, we will join the columns by matching their rows with the school district identifier. We can say that the dataframes have a one-to-one relationship because each row in one will match to at most one row in the other.

```
# Merge fiscal and membership data.
dta <- fis |>
  # Numericize school district identifier to enable merge.
  mutate(LEAID = as.numeric(LEAID)) |>
  # Merge datasets and keep matches.
  inner_join(lea, by = "LEAID", relationship = "one-to-one") |>
```

```
  # Keep non-missing values.
  filter(!is.na(ppe) & !is.na(maj_group))
```

```
Warning: There was 1 warning in `mutate()`.
i In argument: `LEAID = as.numeric(LEAID)`.
Caused by warning:
! NAs introduced by coercion
```

Note that the inner join only keeps school districts that exist in both dataframes. We also drop school districts with missing values. We need to check how much of the sample we lose due to these decisions.

```
# Check number of districts dropped in merge.
all_leaid <- unique(c(as.numeric(fis$LEAID), mem$LEAID))
```

```
Warning in unique(c(as.numeric(fis$LEAID), mem$LEAID)): NAs introduced by
coercion
```

```
pct_retain <- nrow(dta) / length(all_leaid) * 100
print(paste("Percent of districts retained:", sprintf("%3.2f", pct_retain)))
```

```
[1] "Percent of districts retained: 87.59"
```

This seems like a lot. In a real project, we would go back to see how these school districts are distinct from the sample retained and whether they belong. If they do, we would need to find a way to recover the missing data or understand how our analysis could be biased.

Save file

As a last step, keep just the data elements required and rename columns for readability.

```
# Clean up.
dta <- dta |>
  # Choose one column for total district enrollment.
  select(-tot) |>
  rename(tot = MEMBERSCH) |>
  # Rename columns.
  rename(exp = TOTALEXP) |>
  rename_with(tolower, everything()) |>
  # Keep and reorder required columns.
  select(leaid, name, stname, tot, exp, ppe, starts_with("pct_"), maj_group)
```

With data cleaning complete, save the analysis file for use in the analysis program. Come back to the cleaning program to implement major revisions that the analysis requires. Aim to keep the workflow simple so that anyone who looks through the repository can follow along.

```
# Save dataset to load into another R program.
# This puts the dataframe into the same namespace.
save(dta, file = DTA)

# Save to Excel workbook.
write.xlsx(dta, XLS)
```