First, the STATA files are imported using the *haven* library. Columns with only missing values are removed from the DHS dataset, encoded columns are converted to factor variables.

```
library(haven)
library(magrittr)
library(collapse)

# Uganda Demographic and Health Survey 2016: Birth Recode
DHSBR <- paste0(DHS_path, "/Data/UGBR7BDT - Births
Recode/UGBR7BFL.dta") %>%
         read_dta %>% get_vars(fNobs(.) > 0L) %>% as_factor

# Uganda National Household Survey 2016/17: Poverty Estimates
UNHSPOV <- paste0(UNHS_path, "/Household/pov16_rev1.dta") %>%
           read_dta %>% as_factor

# Uganda National Population and Housing Census 2014
CENS <- paste0(CENS_path, "/UBOS 2014 Census.dta") %>% read_dta
```

We start with aggregating the DHS dataset. This data has 786 variables, most of which are categorical:

```
fdim(DHSBR)
## [1] 57906   786

table(vclasses(DHSBR))
##
## character    factor   numeric
##         2       696        88
```

We can obtain a detailed statistical summary of the data using `descr`. The output prints nicely to the console, but can also be converted to a data.frame.

```
descr(DHSBR, table = FALSE) %>% as.data.frame %>% head(10)
##    Variable     Class                                          Label
N Ndist         Mean
## 1    caseid character                            case identification
57906 13745           NA
## 2      bidx   numeric                            birth column number
57906    18 3.486720e+00
## 3      v000 character                         country code and phase
57906     1           NA
## 4      v001   numeric                                 cluster number
57906   696 3.557185e+02
## 5      v002   numeric                               household number
57906   221 2.558897e+01
## 6      v003   numeric                        respondent's line number
57906    20 1.960799e+00
## 7      v004   numeric                             ultimate area unit
57906   696 3.557185e+02
## 8      v005   numeric women's individual sample weight (6 decimals)
```

```
57906   686 9.848528e+05
## 9      v006   numeric                                month of interview
57906     7 8.630176e+00
## 10     v007   numeric                                 year of interview
57906     1 2.016000e+03
##              SD   Min     Max        Skew      Kurt     1%     5%
25%    50%     75%     95%
## 1            NA    NA      NA          NA        NA     NA     NA
NA     NA      NA      NA
## 2  2.367381e+00     1      18  1.05848241  3.806124      1      1
2      3       5       8
## 3            NA    NA      NA          NA        NA     NA     NA
NA     NA      NA      NA
## 4  1.915351e+02     1     697  0.01173270  1.881827     13     53
195    356     519     664
## 5  2.926832e+01     1     545  3.89808066 31.759599      1      2
10     19      28      86
## 6  1.201193e+00     1      21  5.53129314 49.135251      1      1
1      2       2       3
## 7  1.915351e+02     1     697  0.01173270  1.881827     13     53
195    356     519     664
## 8  5.543562e+05 45069 5145429  1.78199379  9.540138 102618 227215
702216 896184 1186668 1973187
## 9  1.496144e+00     6      12 -0.01157971  2.034968      6      6
7      9      10      11
## 10 0.000000e+00  2016    2016         NaN       NaN   2016   2016
2016   2016    2016    2016
##         99%
## 1        NA
## 2        10
## 3        NA
## 4       691
## 5       140
## 6         8
## 7       691
## 8   3142092
## 9        11
## 10     2016
```

The DHS sample comprises 20,880 selected households and 18,506 women being interviewed. Of these women 13,745 had given birth and are recorded in this dataset. As the descriptive statistics above show, the first column gives the women-id (caseid), and the second column an integer id (bidx) for each of the born children.

The aggregation task for this dataset shall simply be to aggregate over the children for each women. A first step to decide how this aggregation is to be done is to examine which variables vary by women i.e. contain child characteristics.

```
# Tabulate child-variant variables
table(varying(DHSBR, ~ caseid))
##
## FALSE  TRUE
##   521   264
```

```
# Examine the numeric child-variant variables
DHSBR %>% fgroup_by(caseid) %>% num_vars %>%
  get_vars(varying(.)) %>% namlab
##    Variable                                                              Label
## 1      bidx                                                     birth column number
## 2      bord                                                      birth order number
## 3        b1                                                          month of birth
## 4        b2                                                           year of birth
## 5        b3                                                     date of birth (cmc)
## 6        b7                                           age at death (months, imputed)
## 7        b8                                                    current age of child
## 8       b11                                        preceding birth interval (months)
## 9       b12                                       succeeding birth interval (months)
## 10      b17                                                            day of birth
## 11      b18                                             century day code of birth (cdc)
## 12      b19 current age of child in months (months since birth for dead children)
## 13      b20                                                   duration of pregnancy
## 14     midx                                                   index to birth history
## 15     hidx                                                   index to birth history
## 16    hidxa                                                   index to birth history
## 17    hwidx                                                   index to birth history
## 18      hw1                                                    child's age in months
## 19    idxml                                                   index to birth history
## 20    idx94                                                   index to birth history
```

These are all variables that we would prefer to aggregate using the average, not the sum or extreme values. It is also noteworthy that the weights don’t vary by child, but only by women, so weighted aggregation is actually not necessary in this case.

```
# Renaming weights variable
```

```
setrename(DHSBR, v005 = weights)
# Confirm that it does not vary by child
varying(DHSBR, weights ~ caseid)
## weights
##   FALSE
```

Thus aggregation in this case is very simple using the `collap()` function, which by default aggregates numeric columns using the mean, and categorical columns using the statistical mode (i.e. the most frequent value):

```
# Aggregating, same as collap(DHSBR, ~ caseid, fmean, fmode), or
collapv(1)
DHSBR_agg <- collap(DHSBR, ~ caseid) %>% fdroplevels

head(DHSBR_agg)
## # A tibble: 6 x 786
##   caseid   bidx v000   v001  v002  v003  v004 weights  v006  v007
v008 v008a  v009  v010  v011  v012
##
## 1 "    0~   1.5 UG7       1     3     2     1 1099225     8  2016
1400 42613     7  1991  1099    25
## 2 "    0~   1.5 UG7       1     4     1     1 1099225     8  2016
1400 42609    12  1975   912    40
## 3 "    0~   1   UG7       1     4     2     1 1099225     8  2016
1400 42609     7  1995  1147    21
## 4 "    0~   1.5 UG7       1     4     6     1 1099225     8  2016
1400 42611     1  1993  1117    23
## 5 "    0~   1.5 UG7       1     4     7     1 1099225     8  2016
1400 42609     2  1986  1034    30
## 6 "    0~   1   UG7       1     4     8     1 1099225     8  2016
1400 42609     5  1989  1073    27
## # ... with 770 more variables: v013 , v014 , v015 , v016 , v017 ,
## #   v018 , v019 , v019a , v020 , v021 , v022 , v023 , v024 ,
## #   v025 , v027 , v028 , v030 , v034 , v040 , v042 , v044 ,
## #   v045a , v045b , v045c , v046 , v101 , v102 , v104 ,
## #   v105 , v106 , v107 , v113 , v115 , v116 , v119 , v120 ,
## #   v121 , v122 , v123 , v124 , v125 , v127 , v128 , v129 ,
## #   v130 , v131 , v133 , v135 , v136 , v137 , v138 , v139 ,
## #   v140 , v149 , v150 , v151 , v152 , v153 , awfactt ,
## #   awfactu , awfactr , awfacte , awfactw , v155 , v157 , v158 ,
## #   v159 , v160 , v161 , v167 , v168 , v169a , v169b ,
## #   v170 , v171a , v171b , v190 , v191 , v190a , v191a ,
## #   ml101 , v201 , v202 , v203 , v204 , v205 , v206 , v207 ,
## #   v208 , v209 , v210 , v211 , v212 , v213 , v214 , v215 ,
## #   v216 , v217 , v218 , v219 , ...

# Aggregating preserves column order and data types / classes +
attributes
identical(namlab(DHSBR_agg, class = TRUE),
          namlab(DHSBR, class = TRUE))
## [1] TRUE
```

Apart from the simplicity and speed of this solution, `collap()` by default preserves the original column order (argument `keep.col.order = TRUE`) and all attributes of columns and the data frame itself. So we can truly speak of an aggregated / collapsed version of this dataset. Calling `fdroplevels` on the result is a likewise highly optimized and non-destructive solution to dropping any redundant factor levels from any of the 696 aggregated factor variables.

Let us now consider the poverty estimates dataset:

```
fdim(UNHSPOV)
## [1] 15636    44

table(vclasses(UNHSPOV))
##
##  factor numeric
##      17      27

descr(UNHSPOV, table = FALSE) %>% as.data.frame %>% head(10)
##    Variable   Class                          Label     N Ndist
Mean           SD       Min
## 1      hhid numeric  Unique identifier  in 2016/17 15636 15636
89610.296943 50753.531112 201.00000
## 2  finalwgt numeric                                  15636  1731
540.811778   519.368731  10.65561
## 3  district  factor                  District Code 15636   112
NA           NA         NA
## 4        ea numeric               Enumeration area 15636    67
9.157265    10.810512   1.00000
## 5     urban  factor         Urban/Rural Identifier 15636     2
NA           NA         NA
## 6    subreg  factor         15       sub     region 15636    15
NA           NA         NA
## 7    region  factor Region of Residence in 2016/17 15636     4
NA           NA         NA
## 8    regurb  factor            RegionxRural/Urban 15636     8
NA           NA         NA
## 9     equiv numeric                    (sum) equiv 15636  9448
3.438747     1.897926   0.71000
## 10    hsize numeric                    (sum) hsize 15636    20
4.515285     2.548680   1.00000
##             Max         Skew      Kurt         1%            5%
25%          50%          75%
## 1  178010.00000 0.002337925  1.833309 2102.35000 9907.7500000
46178.250000 89401.500000 1.327083e+05
## 2    5156.81494 3.097397657 18.780390   34.65487   76.0465393
207.895950   399.305145 6.978978e+02
## 3            NA          NA        NA          NA            NA
NA           NA            NA
## 4      90.00000 3.683418249 21.263899    1.00000    1.0000000
3.000000     6.000000 1.100000e+01
## 5            NA          NA        NA          NA            NA
NA           NA            NA
## 6            NA          NA        NA          NA            NA
```

```
NA            NA            NA
## 7             NA           NA         NA          NA            NA
NA            NA            NA
## 8             NA           NA         NA          NA            NA
NA            NA            NA
## 9      17.28507 0.904448197  4.183096    0.77380    0.8743333
2.009667     3.146083 4.559833e+00
## 10     23.00000 0.734721072  3.761180    1.00000    1.0000000
3.000000     4.000000 6.000000e+00
##             95%          99%
## 1  1.695023e+05 176403.65000
## 2  1.444975e+03   2700.59717
## 3            NA           NA
## 4  2.800000e+01     60.00000
## 5            NA           NA
## 6            NA           NA
## 7            NA           NA
## 8            NA           NA
## 9  6.972708e+00      8.84461
## 10 9.000000e+00     12.00000
```

Using the `qsu()` function, we can also summarize the variation in two of the key variables between district averages and within districts, separated for rural and urban areas. This can give us an idea of the variation in poverty levels we are erasing by aggregating this data to the district level.

```
qsu(UNHSPOV, fexp30 + welfare ~ urban, ~ district, ~ finalwgt,
    vlabels = TRUE)[,"SD",,] # Showing only the standard deviation (SD)
## , , fexp30: Monthly food expenses
##
##            Overall     Between      Within
## Rural  168101.761  47831.6226  161254.386
## Urban   243424.17  56966.9794  240210.089
##
## , , welfare: Welfare based on usual members present
##
##            Overall     Between      Within
## Rural  99872.8917  35288.1075  95355.6836
## Urban  202069.239   64221.637  195061.104
```

The variance breakdown shows that apart from rural welfare, most of the variation in food expenditure and welfare levels is between district averages rather than within districts. We can again examine the numeric variables:

```
UNHSPOV %>% num_vars %>% namlab
##    Variable
Label
## 1      hhid
Unique identifier  in 2016/17
## 2  finalwgt
## 3        ea
Enumeration area
## 4     equiv
```

```
(sum) equiv
## 5     hsize
(sum) hsize
## 6    fexp30
Monthly food expenses
## 7    rexp30                     Monthly household expenditures after
adjusting for inflation
## 8   rrfxp30
## 9   rrexp30 Monthly household expenditures in real prices after
adjusting for regional price
## 10 nrrexp30 Monthly nominal household expenditures in market prices
& after regional price a
## 11  cpexp30 Monthly household expenditures in constant prices after
adjusting for regional p
## 12 fcpexp30 Monthly household food expenditures in constant prices
after adjusting for regio
## 13     mult
## 14    rmult
## 15  welfare                                              Welfare based
on usual members present
## 16 fwelfare
## 17    hmult
## 18   plinen
Poverty line in nominal terms
## 19  ctpline                                                     Poverty
line in constant prices
## 20   hpline                                         Food poverty line in
2009/10 constant prices
## 21   spline                                              Poverty line in
2009/10 constant prices
## 22 fpoor_16                                          food Poor in 2016
based on welfare variable
## 23   decile
Quantile group
## 24      pid
Individual indentifier
## 25    hhage
Age in completed years
## 26  hhedyrs                                                      Number
of school years completed
## 27  hhelder
```

These are also all variables that we would aggregate using a measure of central tendency. The categorical variables are mostly identifiers and also some categorical versions of welfare variables (welfare quintiles), which can all sensibly be aggregated using the statistical mode:

```
UNHSPOV %>% cat_vars %>% namlab
##        Variable
Label
## 1      district
District Code
## 2         urban                                                Urban/Rural
```

```
Identifier
## 3        subreg                                                15      sub
region
## 4        region                                     Region of Residence
in 2016/17
## 5        regurb
RegionxRural/Urban
## 6       poor_16
Poverty status
## 7        quints                        Quintiles based on the national
population
## 8        qurban                         Quintiles based on rural/urban
population
## 9       qregion                            Quintiles based on regional
population
## 10        hhrel  Relationship of household member  to the head of
the household
## 11        mstat                              Marital  status of
household member
## 12        hhsex                        RECODE of R02 (Sex of the
household member)
## 13      hhedlev
## 14 hhstatus_emp                            Activity status(employed,
subsistence)
## 15     hhstatus Activity status(employed, subsistence, unemployed,
not working)
## 16       hhindu
RECODE of B4b
## 17     hhmrtsex                                                Marital
by headship
```

Below we aggregate this dataset, applying the weighted median to numeric data and the weighted mode (default) to categorical data, this time using `collapg` which is a wrapper around `collap` operating on grouped data frames / tibbles.

```
# Weighted aggregation by district, after removing household id and
enumeration area
UNHSPOV %>%
  fselect(-hhid, -ea) %>%
  fgroup_by(district) %>%
  collapg(fmedian, w = finalwgt) %>%
  fdroplevels %>%
  head
## # A tibble: 6 x 42
##   district finalwgt urban subreg region regurb equiv hsize fexp30
rexp30 rrfxp30 rrexp30 nrrexp30
##
## 1 "KALANG~   12994. Rural Centr~ Centr~ Centr~  1.87     2 2.46e5
1.83e5 240877. 180432.  324962.
## 2 "KAMPAL~  460128. Urban Kampa~ Centr~ Centr~  2.30     3 2.89e5
4.17e5 267612. 402942.  662020.
## 3 "KIBOGA"   20524. Rural Centr~ Centr~ Centr~  3.16     4 1.81e5
```

```
2.45e5 171290. 233323.  418979.
## 4 "      L~  118868. Rural Centr~ Centr~ Centr~  2.77     4 2.11e5
2.26e5 199803. 220439.  386698.
## 5 "MASAKA"   92389. Urban Centr~ Centr~ Centr~  2.64     3 2.42e5
2.74e5 224339. 269409.  473376.
## 6 "MPIGI"    65521. Rural Centr~ Centr~ Centr~  2.81     4 2.29e5
2.49e5 222739. 240048.  428228.
## # ... with 29 more variables: cpexp30 , fcpexp30 , mult , rmult ,
welfare ,
## #   fwelfare , hmult , plinen , ctpline , hpline , spline ,
## #   poor_16 , fpoor_16 , quints , decile , qurban , qregion ,
## #   pid , hhrel , hhage , mstat , hhsex , hhedyrs , hhedlev ,
## #   hhstatus_emp , hhstatus , hhindu , hhelder , hhmrtsex
```

Note in the result above that the weighting variable is also aggregated. The default is `wFUN = fsum` so the weights in each group are summed.

At last let's consider the census dataset. On first sight it is a bit simpler than the other two, consisting of 5 character identifiers from the macro-region to the parish level, followed by 270 numeric variables.

```
fdim(CENS)
## [1] 7653  275

table(vclasses(CENS))
##
## character   numeric
##         5       270
```

The specialty of this data is however that some variables are recorded in population totals, and some in percentage terms.

```
descr(CENS, table = FALSE) %>% as.data.frame %>% head(15)
##          Variable     Class
Label    N Ndist
## 1          Region character
7653     4
## 2        District character
7653   122
## 3          County character
7653   199
## 4       Subcounty character
7653  1382
## 5          Parish character
7653  6438
## 6           POP_M   numeric
Population Size: Male 7557  3548
## 7           POP_F   numeric
Population Size: Female 7557  3664
## 8          POP_SR   numeric                                Population
Size: Sex Ratio 7557   609
## 9             POP   numeric
Population Size: Total 7557  4923
```

```
## 10       HHEAD_M   numeric       Headship of Households by Sex:
Male Headed: Number 7557  1736
## 11     HHEAD_M_P   numeric      Headship of Households by Sex: Male
Headed: Percent 7557   359
## 12       HHEAD_F   numeric     Headship of Households by Sex:
Female Headed: Number 7557   846
## 13     HHEAD_F_P   numeric    Headship of Households by Sex: Female
Headed: Percent 7557   359
## 14   HHEAD_10_17   numeric  Household Headship by specific age
groups: 10-17: Number 7557    70
## 15 HHEAD_10_17_P   numeric Household Headship by specific age
groups: 10-17: Percent 7556    40
##            Mean          SD  Min     Max       Skew       Kurt
1%      5%    25%    50%    75%
## 1           NA          NA   NA      NA         NA         NA
NA      NA     NA     NA     NA
## 2           NA          NA   NA      NA         NA         NA
NA      NA     NA     NA     NA
## 3           NA          NA   NA      NA         NA         NA
NA      NA     NA     NA     NA
## 4           NA          NA   NA      NA         NA         NA
NA      NA     NA     NA     NA
## 5           NA          NA   NA      NA         NA         NA
NA      NA     NA     NA     NA
## 6  2236.0525341 2060.3798193 39.0 45834.0  5.8878678  68.350438
335.000  549.00 1155.0 1782.0 2686.0
## 7  2347.0690750 2285.1063696 26.0 52061.0  6.3804915  77.223950
324.000  550.60 1193.0 1852.0 2831.0
## 8    97.1208813   10.7985572 35.0   365.2  5.2374120  86.423031
78.300   85.20   91.9   95.8  100.5
## 9  4583.1216091 4338.2687374 65.0 97895.0  6.1578818  73.263475
668.680 1101.60 2350.0 3634.0 5520.0
## 10  733.6140003  795.4130787  3.0 19855.0  7.5065928 101.724761
106.000  175.00  362.0  565.0  861.0
## 11   77.0265979    6.0370928 21.3    95.5 -0.5516445   5.158277
61.956   67.28   73.1   77.3   81.2
## 12  232.9163689  300.3926888  1.0  7018.0  7.3292989  91.895443
20.000   38.00  100.0  167.0  267.0
## 13   22.9735477    6.0371554  4.5    78.7  0.5516337   5.158115
10.600   13.70   18.8   22.7   26.9
## 14    4.7338891    7.3239515  0.0   148.0  5.0812704  49.771747
0.000    0.00    1.0    3.0    6.0
## 15    0.4547512    0.4600549  0.0     9.2  4.0165231  49.845440
0.000    0.00    0.2    0.4    0.6
##         95%       99%
## 1        NA        NA
## 2        NA        NA
## 3        NA        NA
## 4        NA        NA
## 5        NA        NA
## 6   5102.40 10264.160
## 7   5331.80 11562.360
```

```
## 8    112.40   133.732
## 9  10449.40 22273.800
## 10  1677.00  3929.520
## 11    86.30    89.400
## 12   568.00  1590.760
## 13    32.72    38.044
## 14    16.00    37.000
## 15     1.20     1.900
```

The population counts are easily aggregated by simply computing a sum, but variables providing percentages of the population need to be aggregated using a weighted mean, where the total population serves as the weighting variable. This shows the percentage change variables:

```
# gvr is a shorthand for get_vars(..., regex = TRUE)
gvr(CENS, "_P$") %>% namlab %>% head(10)
##          Variable
Label
## 1      HHEAD_M_P                      Headship of Households by Sex:
Male Headed: Percent
## 2      HHEAD_F_P                    Headship of Households by Sex:
Female Headed: Percent
## 3  HHEAD_10_17_P                Household Headship by specific age
groups: 10-17: Percent
## 4  HHEAD_18_30_P                Household Headship by specific age
groups: 18-30: Percent
## 5  HHEAD_M_A60_P                  Household Headship by specific age
groups: 60+: Percent
## 6    HPOP_0_17_P                        Household Population by
Broad Ages: 0-17: Percent
## 7   HPOP_18_30_P                       Household Population by Broad
Ages: 18-30: Percent
## 8   HPOP_31_59_P                       Household Population by Broad
Ages: 31-59: Percent
## 9     HPOP_A60_P                         Household Population by
Broad Ages: 60+: Percent
## 10      POP_L1_P Population Distribution by Special Age groups: Less
than 1 year: Percent

# Making sure all of these variables are indeed on a percentage scale
range(fmax(gvr(CENS, "_P$")))
## [1]   8.9 100.0
```

To aggregate this data with `collap`, we need to supply the names or indices of both percentage and non-percentage variables together with the corresponding aggregator functions in a list passed to the `custom` argument. Weights are passed to the `w` argument. A specialty here is that we are using `fsum_uw` instead of `fsum`. The postfix `_uw` prevents the weights from being passed to `fsum`, which would otherwise calculate a survey total (i.e. a weighted sum) instead of a simple summation.

```
perc_vars <- gvr(CENS, "_P$", return = "indices")
pop_vars <- setdiff(num_vars(CENS, "indices"), perc_vars)
```

```
collap(CENS, ~ Region + District, w = ~ POP,
       custom = list(fmean = perc_vars, fsum_uw = pop_vars),
       keep.w = FALSE) %>% head
## # A tibble: 6 x 272
##   Region District  POP_M  POP_F POP_SR    POP HHEAD_M HHEAD_M_P
HHEAD_F HHEAD_F_P HHEAD_10_17
##
## 1 Centr~ Buikwe   207324 215447  6807. 422771   71148      72.8
26685      27.2          691
## 2 Centr~ Bukoman~  75109  76304  2442. 151413   23426      68.3
10902      31.7          177
## 3 Centr~ Butamba~  50082  50758  2495. 100840   15128      69.8
6550      30.2          139
## 4 Centr~ Buvuma    48414  41476  4703.  89890   20289      81.3
4830      18.7          211
## 5 Centr~ Gomba     82167  77755  3923. 159922   25794      73.3
9446      26.7          207
## 6 Centr~ Kalanga~  31349  22944  2353   54293   15493      77.1
4548      22.9          123
## # ... with 261 more variables: HHEAD_10_17_P , HHEAD_18_30 ,
HHEAD_18_30_P ,
## #   HHEAD_M_A60 , HHEAD_M_A60_P , HHEAD , HPOP_0_17 , HPOP_0_17_P ,
## #   HPOP_18_30 , HPOP_18_30_P , HPOP_31_59 , HPOP_31_59_P , HPOP_A60
,
## #   HPOP_A60_P , HPOP , POP_L1 , POP_L1_P , POP_0_4 , POP_0_4_P ,
## #   POP_0_8 , POP_0_8_P , POP_2_8 , POP_2_8_P , POP_2_17 ,
## #   POP_2_17_P , POP_6_12 , POP_6_12_P , POP_6_15 , POP_6_15_P ,
## #   POP_10_15 , POP_10_15_P , POP_10_17 , POP_10_17_P , POP_15_24 ,
## #   POP_15_24_P , POP_16_24 , POP_16_24_P , POP_15_29 , POP_15_29_P
,
## #   POP_A2 , POP_A2_P , POP_A10 , POP_A10_P , POP_A15 , POP_A15_P ,
## #   POP_A18 , POP_A18_P , POP_A20 , POP_A20_P , POP_A65 , POP_A65_P
,
## #   EDU_6_12_NAS_M , EDU_6_12_NAS_M_P , EDU_6_12_NAS_F ,
EDU_6_12_NAS_F_P ,
## #   EDU_6_12_NAS , EDU_6_12_NAS_P , EDU_6_12_PRI_M ,
EDU_6_12_PRI_M_P ,
## #   EDU_6_12_PRI_F , EDU_6_12_PRI_F_P , EDU_6_12_PRI ,
EDU_6_12_PRI_P ,
## #   EDU_13_18_SEC_M , EDU_13_18_SEC_M_P , EDU_13_18_SEC_F ,
EDU_13_18_SEC_F_P ,
## #   EDU_13_18_SEC , EDU_13_18_SEC_P , EDU_A15_BS4_M ,
EDU_A15_BS4_M_P ,
## #   EDU_A15_BS4_F , EDU_A15_BS4_F_P , EDU_A15_BS4 , EDU_A15_BS4_P ,
## #   EDU_A18_HO_M , EDU_A18_HO_M_P , EDU_A18_HO_F , EDU_A18_HO_F_P ,
## #   EDU_A18_HO , EDU_A18_HO_P , EDU_A20_HA_M , EDU_A20_HA_M_P ,
## #   EDU_A20_HA_F , EDU_A20_HA_F_P , EDU_A20_HA , EDU_A20_HA_P ,
IL_A18_M ,
## #   IL_A18_M_P , IL_A18_F , IL_A18_F_P , IL_A18 , IL_A18_P ,
## #   IL_10_17 , IL_10_17_P , IL_18_30 , IL_18_30_P , IL_A60 ,
## #   IL_A60_P , BR_L1 , ...
```

Also with the custom argument, the columns are by default (`keep.col.order = TRUE`) rearranged into the order in which they occur. Here we additionally use `keep.w = FALSE`, because the variable `POP` is both used as the weighting variable but also contained in `pop_vars`, and we don't want to have it twice in the output.

Since we are only aggregating numeric data, we may compare the computation speed with a matching *data.table* expression[1]:

```
library(microbenchmark)
library(data.table)
setDT(CENS)

microbenchmark(
  data.table = cbind(CENS[, lapply(.SD, weighted.mean, POP), by =
.(Region, District), .SDcols = perc_vars],
                     CENS[, lapply(.SD, sum), by = .(Region, District),
.SDcols = pop_vars][, -(1:2)]),
  collapse = collap(CENS, ~ Region + District, w = ~ POP,
                     custom = list(fmean = perc_vars, fsum_uw =
pop_vars),
                     keep.w = FALSE)
)
## Unit: milliseconds
##        expr        min         lq       mean     median        uq
max neval cld
##  data.table 153.317076 169.257733 181.740319 175.767603 191.12234
346.31187   100   b
##    collapse   8.997704   9.260768   9.990888   9.837097  10.24251
14.12287   100  a
```