Demographic Methods - Practical 9 (Cohort Fertility and Parity Progression)

2025-11-17

The heading of the R script

The R script begins by clearing the workspace. It leverages the ggplot2 package for plotting and the data.table package for high-performance data manipulation; please ensure these packages are installed.

```r
rm(list = ls())
#install.packages("ggplot2")
#install.packages("data.table")
library(ggplot2)
library(data.table)
```

Reading the data (cohort fertility)

In this exercise, we are provided with age-period-cohort estimates of exposure PY; and the corresponding number of births of the Sweden female population from 1891 to 2024, denoted as B (either observed or estimated). The data for this exercise have been compiled from the *Human Fertility Database* (HFD). For ease of access, an excerpt has been temporarily stored in a GitHub repository and can be retrieved using the following lines of code. However, the data can also be downloaded directly from the HFD (including those for other countries) by running the R script "practical_9_data.R", and providing your username and password.

```r
GitHub                  = "https://raw.githubusercontent.com/Romero-Prieto/teaching/main/Demographic%20Met
data                    = read.csv(GitHub)
```

To simplify data preparation, we can define the fertility records as a table, using the function "as.data.table(*object*)". Before visualisation, we can sort the records by year, cohort, and age (in that order), using the function "setorder(*object*, var_1, var_2, ..., var_n)". The negative symbol indicates that we want the cohort to be sorted in decreasing order. Finally, we can display 5 selected records using the function "print(*object*[Range])".

```r
data                    = as.data.table(data)
setorder(data, "Year", -"Cohort", "x")
print(data[10:14, ])
```

```
##      Year Cohort     x      B       PY
##     <int>  <int> <int>  <num>    <num>
## 1:   1891   1875    15  14.44 22521.77
## 2:   1891   1875    16  37.98 22712.78
## 3:   1891   1874    16  54.99 21502.92
## 4:   1891   1874    17 114.78 21496.67
## 5:   1891   1873    17 157.21 21238.35
```

Note that women born in 1875 turned 16 years old in 1891, as shown in the first two lines. Also note that for this specific cohort and year, 14.44 births were given by mothers aged 15 (i.e., before their birthday), and 37.98 births by mothers aged 16. This age-period-cohort data represent triangle shapes in a Lexis diagram. As a result of combining two adjacent triangles, these data could be used to analyse either period-age (rectangles), cohort-age (horizontal parallelograms), or cohort-period (vertical parallelograms) patterns or interactions.

Some syntax examples for data preparation

To select all elements of the "Cohort" column in the "data" table.

```
subset               = data[["Cohort"]]
```

To select the "sEL" elements of the "Cohort" column in the "data" table. "sEL" can be either a list of elements or a logical vector with the same length as the rows of "data".

```
sEL                  = data[["Cohort"]] > 1990
subset               = data[["Cohort"]][sEL]
```

To extract a subsidiary table from the "data", including all observations but restricted to the columns "Cohort" and "Year".

```
subset               = data[, c("Cohort", "Year")]
```

To extract a subsidiary table from the "data", restricted to the "sEL" observations and the columns "Age" and "Cohort". "sEL" can be either a list of elements or a logical vector with the same length as the rows of "data".

```
subset               = data[sEL, c("Cohort", "Year")]
```

To collapse the table "data" for a selection of cohorts (i.e., from 1950 to 1969), adding the number of births ("B") and the number of person-years ("PY") by each combination of "Year" and "x" (i.e., age) and ignoring NA values. The resulting table is ordered by "Year" and "x".

```
subset               = data[Cohort >= 1950 & Cohort < 1970,
                            list(B = sum(B, na.rm = TRUE), PY = sum(PY, na.rm = TRUE)),
                            by = list(Year, x)][order(Year, x)]
```

Data preparation (cohort fertility)

Fertility data can be arranged for cohort analysis by aggregating the number of births (B) and the exposure to the risk of giving birth (PY) by age and cohort. As indicated by the following line of code, we can create a new table named "cohort" which is adding a set of listed columns (i.e, births and exposure) by a given list of variables (i.e, cohort and age, the latter denoted by x). The syntax of the sum operator "B = sum(B, na.rm = TRUE)", implies that a variable B is created as the sum of all values in B, excluding any NA values (i.e., TRUE). In this exercise B and PY records are free of any NA values, but it is a good practice to specify how missing values should be handled.

```
cohort               = data[, list(B = sum(B, na.rm = TRUE),
                                   PY = sum(PY, na.rm = TRUE)), by = list(Cohort, x)][order(Cohort, x)]
print(cohort[1:5, ])
```

```
##    Cohort     x     B       PY
##     <int> <int> <num>    <num>
## 1:   1835    55     0 11884.45
## 2:   1836    54     0 11657.39
## 3:   1836    55     0 23043.90
## 4:   1837    53     0 11263.26
## 5:   1837    54     0 22275.60
```

However, this is not the case for the Cohort variable, which is missing for some observations (i.e., when age is recorded using an open age interval, the actual cohort is not identified). We can remove these observations, knowing that we are not representing a large number of births (e.g., less than 12 or more than 55 years at the time of fertility). To do so, we can select the records by negating the "is.na(object)" function—using an exclamation mark (!)—applied to the Cohort variable in the cohort table, as shown.

```
cohort               = cohort[!is.na(cohort[["Cohort"]])]
```

Since the reporting of data began in 1891, older cohorts have incomplete fertility schedules. For example, the earliest fertility years are not observed for women born before 1875. Incomplete fertility schedules are also the case of cohorts who are younger than 49 years old by the final year of data reporting. To prepare the data for analysis, we identify and exclude all cohorts with incomplete fertility schedules through a four-step process—one per code line. First, we create a dichotomous variable “rEP”, to indicate if an observation is part of the reproductive ages. Next, we collapse this variable for each cohort to summarise the data and count the number of reproductive ages, creating a table named “sEL”. We then select cohorts with 35 consecutive years of fertility data beginning at age 15 and omit all other columns. Finally, we retain the fertility schedules of all cohorts that meet this criterion merging the two tables, as detailed below.

```
cohort[["rEP"]]      = cohort[["x"]] >= 15 & cohort[["x"]] < 50
sEL                  = cohort[, list(rEP = sum(rEP, na.rm = TRUE)), by = list(Cohort)]
sEL                  = sEL[,"Cohort"][sEL[["rEP"]] == 35]
cohort               = cohort[sEL, on = "Cohort"]
```

Following the same reasoning and similar steps, we can prepare a table for period analysis using these lines of code:

```
period               = data[, list(B = sum(B, na.rm = TRUE),
                                   PY = sum(PY, na.rm = TRUE)), by = list(Year, x)][order(Year, x)]
period[["rEP"]]      = period[["x"]] >= 15 & period[["x"]] < 50
sEL                  = period[, list(rEP = sum(rEP, na.rm = TRUE)), by = list(Year)]
sEL                  = sEL[,"Year"][sEL[["rEP"]] == 35]
period               = period[sEL, on = "Year"]
```

Example questions

Calculate the following values for the 1970 cohort:

a. Total Fertility Rate of the Cohort (TFRc).

Hint: A new column “NnFx” can be generated by multiplying the age-specific fertility rates by the corresponding length of the age intervals. Then, these values should be aggregated for the selected cohort and across the defined range of reproductive ages (i.e., “cohort[[”rEP”]] == 1”).

```
cohort[["NnFx"]]     = cohort[["B"]]/cohort[["PY"]]*1
print(data.frame(cohort[rEP == 1 & Cohort == 1970, list(TFR = sum(NnFx, na.rm = TRUE)),
                        by = list(Cohort)]), row.names = FALSE)
```

```
##  Cohort     TFR
##    1970 1.99852
```

b. Mean Age of the Fertility Schedule of the Cohort (μ_{cf}).

Hint: A new column can be generated by multiplying the mid-point age (i.e., $x + 0.5$) by “NnFx”. Then, these values should be aggregated for the selected cohort and across the defined range of reproductive ages, and finally divided by the “TFRc”.

```
cohort[["mu"]]       = cohort[["NnFx"]]*(cohort[["x"]] + .5)
TFRc                 = cohort[rEP == 1 & Cohort == 1970,
                              list(TFR = sum(NnFx, na.rm = TRUE), mu = sum(mu, na.rm = TRUE)),
                              by = list(Cohort)]
TFRc[["mu"]]         = TFRc[["mu"]]/TFRc[["TFR"]]
print(data.frame(TFRc), row.names = FALSE)
```

```
##  Cohort     TFR      mu
##    1970 1.99852 29.7234
```

Calculate the following values for the year 2000:

c. Total Fertility Rate (TFR) and Mean Age of the Fertility Schedule (μ_f).
Hint: The same approach outlined in questions **a** and **b** can be applied; however, in this case, a specific year should be selected from the period table.

```
period[["NnFx"]]     = period[["B"]]/period[["PY"]]*1
period[["mu"]]       = period[["NnFx"]]*(period[["x"]] + .5)
TFR                  = period[rEP == 1 & Year == 2000,
                              list(TFR = sum(NnFx, na.rm = TRUE), mu = sum(mu, na.rm = TRUE)),
                              by = list(Year)]
TFR[["mu"]]          = TFR[["mu"]]/TFR[["TFR"]]
print(data.frame(TFR), row.names = FALSE)
```

```
##  Year      TFR       mu
##  2000 1.558514 29.85089
```

Exercise 1: Cohort Fertility (Compulsory)

a. Calculate the Total Fertility Rate (TFRc) and the Mean Age of the Fertility (μ_{cf}) for all cohorts.
Hint: The same approach described in the example questions can be applied. Since the columns "NnFx" and "mu" were generated as part of the example, the only remaining step is to collapse the cohort table. However, in this case, it is not necessary to select a specific cohort.

```
TFRc                 = cohort[rEP == 1,
                              list(TFR = sum(NnFx, na.rm = TRUE), mu = sum(mu, na.rm = TRUE)),
                              by = list(Cohort)]
TFRc[["mu"]]         = TFRc[["mu"]]/TFRc[["TFR"]]
TFRc
```

```
##      Cohort      TFR       mu
##       <int>    <num>    <num>
##    1:  1875 3.505663 31.13993
##    2:  1876 3.468691 31.03758
##    3:  1877 3.405288 30.92296
##    4:  1878 3.344133 30.79724
##    5:  1879 3.274155 30.68062
##  ---
##  97:   1971 1.976314 30.00525
##  98:   1972 1.975296 30.17949
##  99:   1973 1.959574 30.45740
## 100:   1974 1.964988 30.61002
## 101:   1975 1.954961 30.66186
```

b. Calculate the Total Fertility Rate (TFR) and the Mean Age of the Fertility (μ_f) for all years.
Hint: The approach followed in the previous questions should be applied, with the additional step of collapsing the period table to include all available years.

```
TFR                  = period[rEP == 1,
                              list(TFR = sum(NnFx, na.rm = TRUE), mu = sum(mu, na.rm = TRUE)),
                              by = list(Year)]
TFR[["mu"]]          = TFR[["mu"]]/TFR[["TFR"]]
TFR
```

```
##       Year       TFR        mu
##       <int>    <num>    <num>
##   1:  1891 4.141176 32.43192
##   2:  1892 3.965773 32.35430
##   3:  1893 4.038212 32.37759
##   4:  1894 4.018654 32.37592
##   5:  1895 4.082778 32.23590
##  ---
## 130:  2020 1.668500 31.34318
## 131:  2021 1.672789 31.49426
## 132:  2022 1.524565 31.56595
## 133:  2023 1.453109 31.64502
## 134:  2024 1.432871 31.77252
```

Exercise 2: Cohort Fertility (Compulsory)

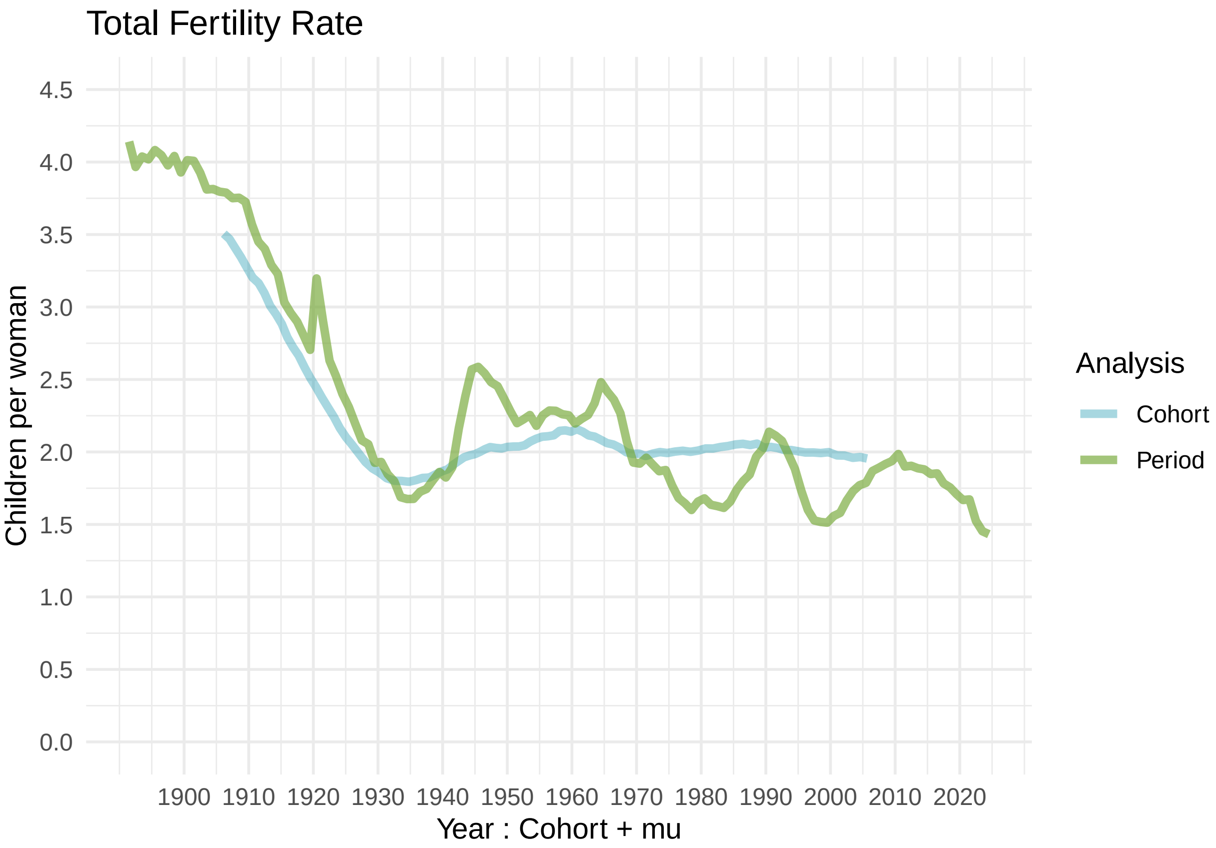
As a contrast, plot the TFR and the TFRc in the same figure. Discuss your results.
Hint: To make these two to measures time-comparable, plot the period TFR at the mid-year (i.e., $Year + 0.5$) and the TFRc at the year of birth plus the Mean Age of the Fertility Schedule (i.e., $Cohort + \mu_{cf}$).

```
TFRc[["Year"]]        = TFRc[["Cohort"]] + TFRc[["mu"]]
TFRc[["Analysis"]]    = "Cohort"

TFR[["Year"]]         = TFR[["Year"]] + 0.5
TFR[["Analysis"]]     = "Period"
Fertility             = data.frame(rbind(TFRc[, c("Year", "TFR", "mu", "Analysis")], TFR))

coloUR                = c("Cohort" = rgb(0.00,0.55,0.65,0.35),
                          "Period" = rgb(0.45,0.65,0.20,0.65))

ggplot(Fertility, aes(x = Year, y = TFR, color = Analysis, group = Analysis)) +
  geom_line(linewidth = 1.5) +
  coord_cartesian(ylim = c(0, 4.5)) +
  scale_color_manual(values = coloUR) +
  scale_x_continuous(breaks = seq(1900, 2020, by = 10)) +
  scale_y_continuous(breaks = seq(0, 4.5, by = 0.5)) +
  labs(title = "Total Fertility Rate", x = "Year : Cohort + mu", y = "Children per woman") +
  theme_minimal()
```



Exercise 3: Parity Progression

In this exercise, we are provided with an excerpt of individual-level data from the UNICEF *Multiple Indicator Cluster Survey* (MICS). For a group of women aged 45-49 interviewed in Malawi in 2019-20 (i.e., those with almost completed fertility), the file includes the *Parity* (i.e., total number of children ever born); the place of residence *UR* (urban = 1, rural = 2); the level of *Education* (less than complete primary = 1, incomplete secondary = 2, complete secondary or higher = 3); and a relative weight W (i.e., the number of women each participant represents). For ease of access, the file has been temporarily stored in a GitHub repository and can be retrieved using the following lines of code. No additional data preparation is required.

```
GitHub               = "https://raw.githubusercontent.com/Romero-Prieto/teaching/main/Demographic%20Met
data                 = read.csv(GitHub)
data                 = as.data.table(data)
data
```

```
##       Parity    UR Education         W
##        <int> <int>     <int>     <num>
##    1:      4     2         1 0.2711989
##    2:      8     2         1 0.0427552
##    3:      5     2         1 1.2126919
##    4:      7     2         1 0.9613293
##    5:      5     2         1 0.5923323
##   ---
## 1665:      4     2         2 0.2687332
## 1666:      6     2         1 1.9555703
## 1667:      8     2         1 0.5240250
## 1668:      9     2         1 0.5445887
## 1669:      4     2         1 1.6466823
```

Example questions

a. Calculate the Parity Progression Ratios and the TFR for this cohort.
Hint: A collapsed and sorted table can be generated by parity. In some cases, one or more parities might lack empirical support (i.e., the maximum parity is 15, but no woman reported having 13 children). Hence, to avoid missing parities and potential errors, it is good practice to create a data frame that includes all parities from 0 up to the maximum number of children ever born, and then allocate the empirical data to that frame, as described by the following three lines of code:

```
summary              = data[, list(Wi = sum(W, na.rm = TRUE)), by = list(Parity)][order(Parity)]
PPR                  = data.table(Parity = 0:max(data[["Parity"]]),
                                  Wi = rep(0,max(data[["Parity"]]) + 1))
setDT(PPR)[summary, on = "Parity", c("Wi") := list(i.Wi)]
```

Using the number of women by parity, we can then quantify the number of women with parity less than or equal to i (Pi); the proportion of women progressing from parity i — 1 to i (PPRi); and the proportion of women progressing from parity 0 to i (PPRoi), following the next three lines of code. Note that we use the functions head(*object*, *index*) and tail(*object*, *index*). When *index* is positive, head() returns the first *index* elements (i.e., from position 1 to *index*), and tail() returns the last *index* elements. When *index* is negative, head() returns all elements except the last *index* elements, and tail() returns all elements except the first *index* elements.

```
PPR[["Pi"]]          = rev(cumsum(rev(PPR[["Wi"]])))
PPR[["PPRi"]]        = c(NA,tail(PPR[["Pi"]],-1)/head(PPR[["Pi"]],-1))
PPR[["PPRoi"]]       = c(NA,cumprod(tail(PPR[["PPRi"]],-1)))
```

Finally, we can calculate the TFRc and display both the Parity Progression Table and the resulting Total Fertility Rate, as shown:

```
TFRc                 = sum(tail(PPR[["PPRoi"]],-1))
print(data.frame(PPR), row.names = FALSE)
```

```
##  Parity         Wi          Pi      PPRi        PPRoi
##       0  26.487010 1669.000000        NA           NA
##       1  44.240165 1642.512990 0.9841300 0.9841300123
##       2  71.342818 1598.272825 0.9730656 0.9576230228
##       3 104.014071 1526.930006 0.9553626 0.9148771759
##       4 191.738699 1422.915936 0.9318803 0.8525559830
##       5 257.125766 1231.177236 0.8652495 0.7376735989
##       6 302.826536  974.051471 0.7911545 0.5836138232
##       7 259.433408  671.224935 0.6891062 0.4021719206
##       8 190.341742  411.791527 0.6134926 0.2467294950
##       9 113.147087  221.449785 0.5377716 0.1326841131
##      10  73.281497  108.302697 0.4890621 0.0648907713
##      11  16.689788   35.021200 0.3233641 0.0209833435
##      12  10.613651   18.331412 0.5234376 0.0109834705
##      13   6.176381    7.717762 0.4210129 0.0046241831
##      14   0.000000    1.541380 0.1997186 0.0009235352
##      15   1.541380    1.541380 1.0000000 0.0009235352
```

```
print(data.frame(TFRc), row.names = FALSE)
```

```
##      TFRc
##  5.915388
```

b. Calculate the probability that a woman of this cohort would have 3 children.
Hint: Using the column Wi, calculate the proportion of women with exactly three children.

```
PPR[["Wi"]][PPR[["Parity"]] == 3]/PPR[["Pi"]][PPR[["Parity"]] == 0]
```

```
## [1] 0.06232119
```

c. Calculate the probability that a woman of this cohort would have at least 3 children.
Hint: Report the PPR from 0 to 3.

```
PPR[["PPRoi"]][PPR[["Parity"]] == 3]
```

```
## [1] 0.9148772
```

d. Calculate the probability that a woman of this cohort would have at least 3 children if she already have one child (progression form 1 to 3).
Hint: Divide the PPR from 0 to 3 by the PPR from 0 to 1.

```
PPR[["PPRoi"]][PPR[["Parity"]] == 3]/PPR[["PPRoi"]][PPR[["Parity"]] == 1]
```

```
## [1] 0.9296304
```

Compulsory questions

a. Using Parity Progression tables, calculate the Urban-Rural TFR gap for this cohort and comment on your results.
Hint: Using the code from the example question, a temporary function named **fertility** can be defined to return the **TFRc** for a given input **data**, in the following form: fertility = function(data) { *list of operations* }. Next, evaluate the function on a subset of the data representing either urban or rural residents (e.g., “result = fertility(data[data[[”UR”]] == 1])”). Finally, calculate the gap as the difference between the TFRc values for rural and urban residents.

```
fertility = function(data){
  summary                = data[, list(Wi = sum(W, na.rm = TRUE)), by = list(Parity)][order(Parity)]
  PPR                    = data.table(Parity = 0:max(data[["Parity"]]),
                                      Wi = rep(0,max(data[["Parity"]]) + 1))
  setDT(PPR)[summary, on = "Parity", c("Wi") := list(i.Wi)]
  PPR[["Pi"]]            = rev(cumsum(rev(PPR[["Wi"]])))
  PPR[["PPRi"]]          = c(NA,tail(PPR[["Pi"]], -1)/head(PPR[["Pi"]], -1))
  PPR[["PPRoi"]]         = c(NA,cumprod(tail(PPR[["PPRi"]], -1)))

  TFRc                   = sum(tail(PPR[["PPRoi"]], -1))
  return(list(PPR = PPR, TFRc = TFRc))
  }

Residence              = c("Urban: ", "Rural: ", "Gap: ")
TFRc                   = rep(NA, 3)
for (i in 1:2){
  result                 = fertility(data[data[["UR"]] == i])
  TFRc[i]                = result[["TFRc"]]
}
TFRc[3]                = TFRc[2] - TFRc[1]
print(data.frame(Residence, TFRc), row.names = FALSE)
```

```
##  Residence     TFRc
##    Urban:  4.986598
##    Rural:  6.067671
##      Gap:  1.081072
```

b. Using Parity Progression tables, calculate the TFR for each level of education of this cohort and comment on your results.

Hint: Use the **fertility** function to evaluate the subset of data for women with different levels of education (e.g., “result = fertility(data[data[[”Education”]] == 1])”).

```
Education            = c("Less than complete primary: ",
                         "Incomplete secondary: ",
                         "Complete secondary or higher: ")
TFR                  = rep(NA, 2)
for (i in 1:3){
  result               = fertility(data[data[["Education"]] == i])
  TFRc[i]              = result[["TFRc"]]
}
print(data.frame(Education, TFRc), row.names = FALSE)
```

```
##                        Education     TFRc
##    Less than complete primary:  6.258818
##          Incomplete secondary:  5.170427
##  Complete secondary or higher:  2.808793
```