Statistisk sentralbyrå
Statistics Norway

LÆRERMOD 2.0

Technical documentation of Statistics Norway's teacher projection model

Trude Gunnes and Pål Knudsen

TALL SOM FORTELLER
NOTATER / DOCUMENTS
2024/33

In the series Documents, documentation, method descriptions, model descriptions and standards are published.

© Statistics Norway

Published: 30 August 2024

ISBN 978-82-587-1013-1 (electronic)
ISSN 2535-7271 (electronic)

Symbols in tables	Symbol
Category not applicable Figures do not exist at this time, because the category was not in use when the figures were collected.	.
Not available Figures have not been entered into our databases or are too unreliable to be published.	..
Confidential Figures are not published to avoid identifying persons or companies.	:
Decimal punctuation mark	.

Preface

This documentation, funded by the Norwegian Ministry of Education and Research, presents the new version of LÆRERMOD – a teacher projection model.

Statistics Norway, August 2024

Linda Nøstbakken

Abstract

This documentation highlights the new version of LÆRERMOD - assumptions, equations, and implementation in Python. LÆRERMOD is a teacher projection model that calculates the annual supply and demand separately for seven groups of teacher educations.

Contents

1. Introduction

LÆRERMOD calculates future supply and demand for teachers at the national level. The model aims to provide information to ensure a balanced number of teacher graduates in the coming years and prevent increased teacher shortages.

As before, we define a teacher by education, not occupation. In other words, a teacher in LÆRERMOD has a teacher education. Previously, the model projected five education groups: (i) kindergarten teachers, (ii) compulsory education teachers, (iii) vocational teachers, subject teachers, and lecturers, teachers with (iv) practical pedagogical education (PPE), and (v) practical pedagogical education in vocational subjects (PPE-V). Now, we split group (iii). It has not happened before because splitting vocational and subject-teacher educations using educational classification codes is not straightforward. Additionally, lecturer education programs are new, so the population of lecturers is relatively small. LÆRERMOD 2.0 projects separate demand and supply for seven groups of teacher educations:

- Kindergarten / pre-school teacher education
- General teacher education
- Lecturer / subject teacher education
- Practical pedagogical education (PPE)
- Teacher education in practical and aesthetic subjects
- Vocational teacher education
- PPE - vocational

The disaggregation enables us to grasp the supply and demand for vocational teachers. Additionally, we can aggregate the supply and demand for vocational teachers and those with practical pedagogical education in vocational subjects to get a comprehensive picture of the future situation for qualified vocational subject teachers. Furthermore, we can aggregate supply and demand for people with lecturer education and practical pedagogical education. They teach one to two subjects, mainly in secondary and upper-secondary schools.

Compared to Gunnes and Knudsen (2016), we have also made these changes in the model:

- Teacher vacancies in the starting year are no longer included. Supply is equal to demand in the starting year in our reference scenario. We correct for any teacher shortage in the starting year in an alternative scenario based on Statistics Norway's statistics on employees in kindergartens and schools, where we include the number of full-time equivalents in teaching positions without teacher education. Teacher shortage is thus no longer unfilled teaching positions but the number of unqualified teachers in teaching positions.
- Previously, we used the KOSTRA files for the initial population of teacher graduates. These contain labor market and educational information published in year t with information from year t-2. We now use the a-register from year t-1 and link education information ourselves. The starting year (i.e., statistical year) is, therefore, year t-1, not year t-2.
- Those with other pedagogical educations are not included in the initial population of teachers. See discussion in Gunnes et al. (2018).
- The definition of enrolled first-year teacher students has changed. See discussion in Gunnes et al. (2023). In addition, we now include only students under 50 years old (18-49).
- Although completion rates for PPE and PPE-Y teachers are in StatBank Norway, we calculate these ourselves as these are one-year programs that are easier to complete than other

teacher education programs. Since completion rates are from previous cohorts, we often use alternative probabilities in an alternative scenario.
- People in upper secondary education are now in the age group 15-49, not 15-99. The number of people in upper secondary education varies little whether we use 15-49 or 15-99, but given the ageing of the population, the growth rate will be somewhat higher for those in the group 15-99 than 15-49. We have, therefore, changed this even though it has little significance for the results.
- The projection period is extended (to 2060) to highlight the dynamics of teacher retirement (supply side) and the development in the number of children and young people (demand side).

To enable all these changes in the model, we have migrated it to Python.

This technical note describes the characteristics and assumptions of LÆRERMOD, not new projection results. We also present the equations and document the implementation of LÆRERMOD in Python.

We include three appendices:

- The input data of LÆRERMOD (Appendix A).
- How we correct for teacher shortages in the initial year and interpret a corrected versus uncorrected surplus/deficit of teachers (Appendix B).
- Implementation of the model in R (Appendix C).

We have used artificial intelligence (AI) in chapter 3 and Appendix C in this technical note (ChatGPT). AI will further become convenient for creating figures showing variables calculated in the model, e.g., the retirement behavior of teacher graduates. AI can also help develop a new version of the model where we include new components (e.g., regional aspects). By feeding ChatGPT with aggregated data on new variables (e.g., regional information), ChatGPT generates Python codes that incorporate new variables based on these data. This way, we can easily extend the model. The job is to make adequate assumptions so that the projections become better and suitable for policymaking.

2. Assumptions

The model answers one essential question: Will the teacher shortage – i.e., individuals without teacher education in teaching positions – increase or decrease in the future? In other words, will there be fewer or more qualified teachers in teaching positions in the long run?

The starting point for the calculations is the current population of teacher-educated individuals. The sample includes all aged 18-74 with teacher education in the starting year.

Next, we analyze employment rates based on education, age, and gender. To account for varying work hours, we calculate average full-time equivalents for each type of teacher education, broken down by one-year age groups and gender. As a result, we have comprehensive information on age, gender, and teacher education for the starting population. (See Appendix A for data sources.)

LÆRERMOD is a partial model. It includes people with teacher education and no other educational groups. The model highlights the continuity of the teacher population over time and consists of several building blocks:

(i) a constant annual candidate production/inflow of new teachers
(ii) ageing and future retirement among the teacher-educated
(iii) teacher densities in the baseline year (in and outside the education sector)
(iv) the development in the number of future children, students, and other users of teachers

LÆRERMOD calculates whether today's production of teacher candidates is sustainable in the long term: Can we maintain today's teacher densities given the development in the future number of children and students and the retirement among teachers? Or, must the teacher candidate production augment to counteract the demographics and retirement due to age?

We assume that supply equals demand in the starting year. Therefore, the number of teacher full-time equivalents on the supply and demand side is balanced in the starting year. However, a teacher shortage may exist – the demand for teachers is higher than the supply. We correct for any teacher shortage in the starting year in an alternative scenario (see Appendix B). That is, in the reference scenario, a future teacher surplus is uncorrected – we disregard the initial teacher shortage. Hence, a surplus does not necessarily mean we produce too many teachers. It means we can reduce the initial teacher shortages. A future surplus may, in other words, be an improvement in the form of increased teacher density (i.e., the number of teacher graduates per user) in the long term.

We project supply and demand for each of the seven education groups included in the model.

Supply and demand are calculated separately. The supply side, influenced by the inflow of new teacher candidates and the outflow of retired teachers, is assumed to be independent of the demand side, which depends on the demographic component.

LÆRERMOD is an annual model projecting supply and demand for several years (until 2060) at the national level. The model does not consider:

- The need for teachers across different geographic regions and individual schools
- The need for teachers in different subject areas

2.2. Supply side

The starting population changes on the supply side during the projection period due to two factors:

- The inflow of new teacher candidates
- The outflow due to ageing and retirement among the starting population of teachers and new teacher candidates

Employed teachers in the starting year grow older in the projection period. Due to age, they leave the working population and are no longer part of the supply side. In addition to age, retirement also varies between genders and teacher educations. In other words, retirement varies yearly depending on the age- and gender profiles of the seven starting populations.

As mentioned, we have age, gender, and type of teacher education for all individuals aged 18-74 in the starting population. For each one-year age, gender, and type of teacher education, average full-time equivalents are calculated based on observed retirement among the starting population, which decreases with age and is zero from age 75. During the projection period, the population ages each year, and individuals are assigned the average employment rate for their age, gender, and type of teacher education.

The inflow of teacher candidates is constant in each projection year based on the number of enrolled first-year students in the starting year and a completion rate based on previous cohorts. It estimates how many will complete a teacher education and thus contribute to the labor market as qualified teachers.

For each enrolled first-year student in teacher education programs, we have information about gender, age, and study length (to calculate the age when they enter the labor market), so we can assign the candidates the same employment pattern and retirement behavior as those in the starting population of teachers. Employment rate and average full-time equivalents are, in other words, constant in the projection period and equal to the base year for a given age, gender, and type of teacher education. It simplifies the model. At the same time, these have been relatively constant over time, so empirically, the assumption is reasonable.

The annual inflow of candidates remains constant in LÆRERMOD, not because we believe it will stay unchanged, but because the model illustrates what will happen in the long run if this standard is maintained. We can also use the model to consider scenarios where the number of students is lower or higher than today. A lower (higher) number of students will result in a lower (higher) projected future supply than in the baseline calculations. LÆRERMOD relies on assumptions, and understanding these assumptions is crucial for interpreting the results. The model does not aim at generating the most likely forecasts to accurately project the number of teacher full-time equivalents in the future. Instead, LÆRERMOD addresses one fundamental question: Will the future teacher population be sufficient if today's candidate production is maintained, or should policy measures be implemented to increase candidate production?

The supply of teachers increases during the projection period if the number of candidates exceeds the number of teachers that retire. Previous projections show that the supply is upward-sloping throughout the projection period.

2.3. Demand side

The demand for teachers is demographically driven – it is the number of future children and young people who will need teachers. The total fertility rate, i.e., how many children women in the population will give birth to on average, thus affects the demand for teachers. Since different age

groups demand different types of teachers and we expect varying growth within the different age groups, the demand side is sector-distributed. Therefore, we allocate full-time equivalents based on sectors in the starting year. These sectors include:

- Kindergarten
- Compulsory education (primary education and middle school)
- Upper secondary education (high school)
- Higher education
- Others in the education sector (adult education, higher vocational education, etc.)
- Outside the education sector

LÆRERMOD acknowledges that not everyone with teacher education works as a teacher – some work outside the sector. The model also takes into account that compulsory school teachers, for instance, may work in various education sectors, not solely within compulsory education. Thus, the model focuses on where teachers work rather than where they are qualified.

The sector share of teacher full-time equivalents in the starting year remains constant in the projection period. In other words, the proportional distribution of teacher full-time equivalents across the six sectors does not change. If 80% of those with compulsory school teacher education work in compulsory education in the starting year, 80% will also work here in the final year of the calculations. This assumption is reasonable if we believe that the relative wage ratio between sectors 1-6 does not change in the projection period.

In addition to the number of employed full-time equivalents, we also know the number of users in each sector in the baseline year. LÆRERMOD considers that not all children of kindergarten age attend kindergarten, for example. And not everyone in the given age group is in secondary or higher education. Furthermore, the model considers the stay duration for children in kindergarten because it varies and affects the demand. The user coverage rates in the sectors, i.e., the number of users relative to the total number of people in an age group, are constant in the projection period. For example, if 90% of children aged 0-5 attend kindergarten in the starting year, the proportion is the same in the final year of the calculations. For compulsory education, the coverage rate is equal to 1 since it is mandatory. The assumption of constant coverage rates simplifies the model, and it is reasonable to assume they are so over the projection period.

Furthermore, the model calculates teacher density (for each teacher education group) in each sector. It is the number of full-time equivalents divided by the number of users per sector. These densities (7 teacher educations *x* 6 sectors) are constant in each projection year. Then, the demand is calculated based on how many full-time equivalents are needed to maintain these densities for each education group of teachers each year, given the development in the future number of children, students, and other users. We consider the age distribution for children in kindergartens because the teacher density is higher among younger than older children.

For the growth in users per sector (the demographic component), we use:

- Kindergarten: Growth in the number of children 0-5 years.
- Compulsory education: Growth in individuals aged 6-15.
- Secondary education: Growth in individuals aged 15-49.
- Higher education: Growth in individuals aged 18-49.
- Other in the sector (e.g., adult education): Growth in the entire population (0-99).
- Outside the sector: Growth in the entire population (0-99).

User growth varies from sector to sector and year to year and is based on Statistics Norway's national population projections. They are updated every two years. There are also regional

projections so that each municipality can compare its growth in the number of users in each sector with the national development.

Fewer (more) teacher full-time equivalents are in demand if the number of children/students/users decreases (increases). Previous projections show increasing demand over time. In the most recent projections, the demand has flattened out over time due to the demographic component (low fertility).

Similarly, to a constant candidate production on the supply side, constant teacher densities (users per teacher) are assumed on the demand side. LÆRERMOD calculates whether today's candidate production is sustainable and whether we can maintain today's teacher densities in the long term, given retirement on the supply side and demographic development on the demand side.

2.4. Difference between supply and demand

We must weigh the supply against the demand to calculate whether the teacher shortage will increase or decrease in the long term. If the supply exceeds the demand, a so-called surplus of teachers is estimated, conversely, a deficit. We compare surpluses and deficits to the teacher shortage in the initial year of the calculations. If there is a teacher shortage in the initial year, an estimated future surplus or deficit is an improvement or worsening compared to today's situation. A future surplus (or deficit) must be corrected by the number of full-time equivalents in teaching positions without teacher education in the initial year because we have set supply equal to demand in the starting year of the calculations. Hence, a surplus does not mean we produce too many teachers, but we can reduce the initial teacher shortage. LÆRERMOD thus calculates whether we can maintain today's standards for teacher shortage in the long term - or not.

The surpluses or deficits we compute are sector-specific. Because the sector-specific distribution of teacher full-time equivalents is constant in the projection period, we can deploy the share in the initial year to proportionally distribute a surplus or deficit across sectors 1-6.

LÆRERMOD considers several variables. The model demonstrates that not only the number of new teacher students matters. Other variables also play a role:

- Completion probability for each teacher education.
- Future retirement – the age and gender distribution of the initial population and new graduates, which varies between teacher educations.
- The teacher density in each sector in the baseline year that we aim to maintain.
- Out-of-sector mobility – the proportion of teachers working outside the sector.
- Demographics – the number of users in each sector in each projection year.

Fewer teacher students (from one projection to another) does not necessarily increase the teacher shortage since other variables in the model can offset it. For example, if fertility decreases, the demand will decrease because fewer teachers are needed to maintain a given teacher density.

Previous projections quantify the different components of the model (see an overview on page 13).[1] In short, the empirical landscape is as follows: On the supply side, we see signs that fewer young people are applying to teacher education programs (student numbers have declined since 2019). At the same time, on the demand side, birth rates have been lower in the last ten years. Completion probabilities, teacher densities, and out-of-sector mobility are relatively constant. Retirement is also less burdensome given relatively young cohorts in the initial teacher populations. Together, this

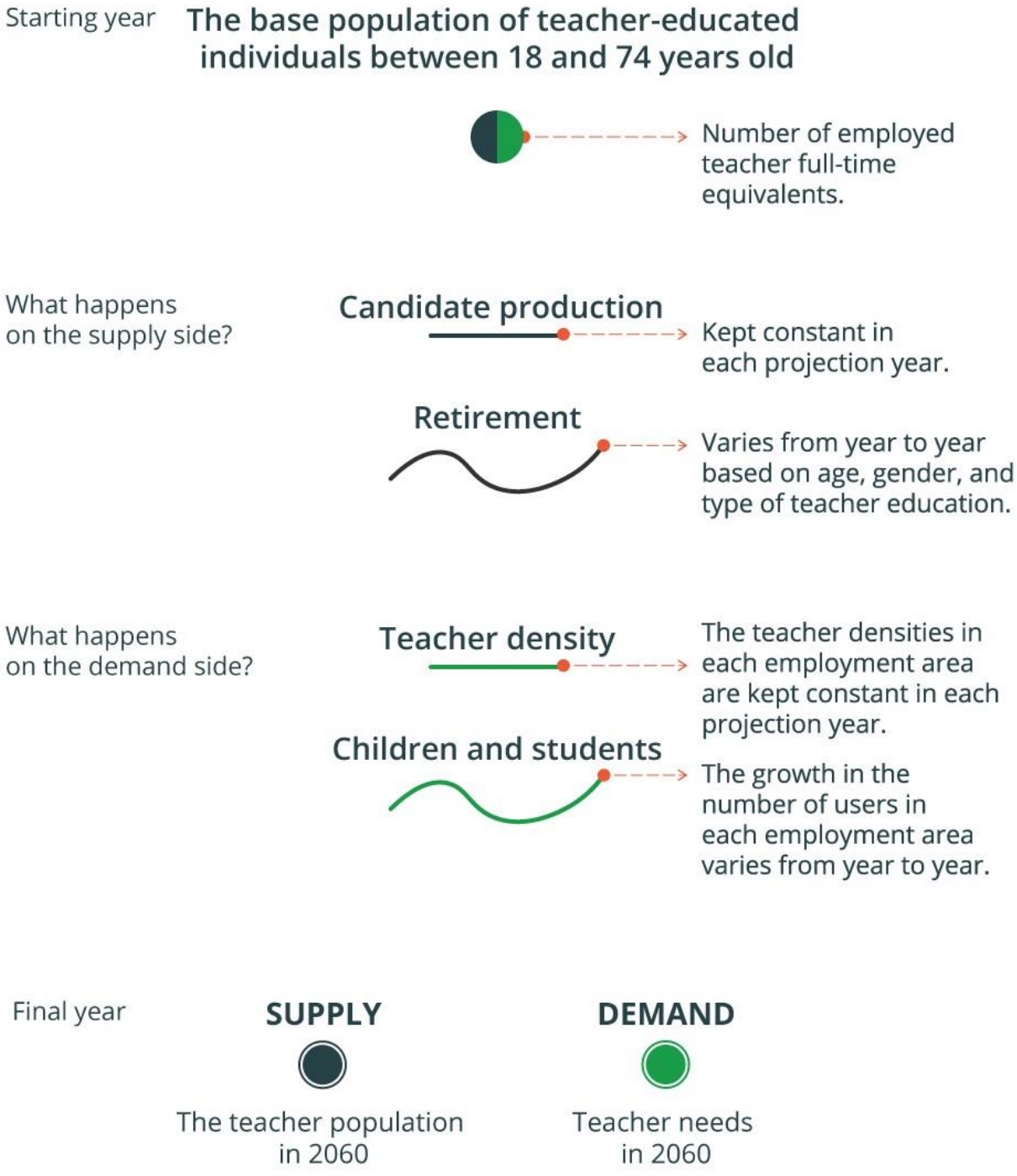
[1] Teachermod is the direct translation of LÆRERMOD.

suggests that the teacher shortages are not likely to increase significantly soon, even though student numbers are reduced.

To reduce teacher shortages, one can think of policy measures directed towards:

- Increasing applicants to teacher education programs (nudge those with preferences to become teachers).
- Increasing study completion rates (improve institutions or let more students through).
- Reducing out-of-sector mobility (better work environments and career paths to maintain teachers in the profession).

Overview of the components in the projection (TEACHERMOD)



3. Equations

The equations are slightly changed compared to previous versions of the model described in Gunnes and Knudsen (2016). For example, as mentioned in the Introduction (chapter 1), those with other pedagogical educations and special needs education are no longer included in the initial teacher population, which reduces the number of equations. Additionally, structural changes have resulted in a new order. Previously, we calculated separately the demographic components and the initial population of teachers. We now consider the supply, demand, and the difference between supply and demand. Note that parts of the discussion of the equations in this chapter are analyses conducted with ChatGPT but are assured by the authors.

LÆRERMOD is an annual model. For each year, the model calculates supply, demand, and the difference between supply and demand as teacher full-time equivalents. The starting point is the population of educated teachers in the baseline year.

Symbols:

- U = teacher education (7 separate groups)
- a = age (one-year age intervals ranging from 18 to 74 for teacher-educated individuals and 18 to 49 for teacher students)
- k = gender (male or female)
- i = individual
- S = employment sector (6 separate groups)
- g = age groups of children in kindergarten (0, 1-2, 3, 4-5 years) and all the five remaining age groups linked to the remaining five employment sectors
- o = stay duration categories in kindergartens (with a maximum stay duration of 42,5 hours per week)
- t = year (baseline year, projection years, and final year)

3.1. Supply side

Employment among educated teachers in the baseline year

Equation (1) defines the ratio of employed teachers among all educated teachers for each teacher education (U), age (a), and gender (k). This ratio is crucial for assessing the proportion of qualified individuals employed as teachers within each education, age, and gender.

(1) $$(Employmentrate)_U^{a,k} = \frac{(Employed\ teachers)_U^{a,k}}{(Total\ numbers\ of\ educated\ teachers)_U^{a,k}}$$

Equation (2) gives the average full-time equivalents, providing a more accurate picture of the total labor available from each education, age, and gender.

(2) $$(Average\ fulltime\ equivalent)_U^{a,k} = \frac{(Fulltime\ equivalent\ teachers)_U^{a,k}}{(Employed\ teachers)_U^{a,k}}$$

Candidate production of new teachers

Equation (3) calculates the total number of individuals who have started a teacher education (U) for a given gender (k) by summing all within the age group 18-49 years. The equation provides a picture of the first-year recruitment to teacher education programs, which is fundamental for understanding the future supply of teachers in the labor market.

(3) $$(First\ year\ teacher\ students)_U^k = \sum_{a=18}^{a=49} (First\ year\ teacher\ students)_U^{a,k}$$

Equation (4) calculates the share of first-year students with a specific combination of age (a) and gender (k) among all first-year students within an education group U. This distribution is critical for understanding the demographic composition of new teacher students and suggests the future composition of teachers in the labor market.

(4) $$(Share\ of\ first\ year\ teacher\ \ students)_U^{a,k} = \frac{(First\ year\ teacher\ students)_U^{a,k}}{(First\ year\ \ teacher\ students)_U^{k=male} + (First\ year\ \ teacher\ students)_U^{k=female}}$$

Equation (5) calculates the number of teacher graduates by multiplying the number of new students by an expected completion rate. It estimates how many teacher students will complete their education and thus contribute to the labor market as qualified teachers.

(5) $$Teacher\ graduates_U = (First\ year\ \ teacher\ students)_U * (Completionrate)_U$$

Equation (6) calculates the expected age at graduation for each specific combination of age (a), gender (k), and education (U) based on the age of first-year students and the length of the study program. The age at graduation is assigned to the new candidates when they enter the labor market.

(6) $$(Age\ at\ graduation)_U^{a,k} = (Age\ as\ first\ year\ \ teacher\ student)_U^{a,k} + (Studylength)_U$$

Equation (7) calculates the number of teacher candidates distributed by type of teacher education, gender, and age, where the number of candidates is from (5), age from (6), and the proportion of students by age and gender from equation (4) - providing an accurate distribution of newly graduated individuals into the workforce during the projection period.

(7) $$Graduates_U^{a,k} = Graduates_U * (Share\ of\ first\ year\ \ teacher\ students)_U^{a,k}$$

Note that candidate production is constant every year during the projection period. In other words, we do not calculate a trend in the number of new teachers over time. These are projections, not forecasts. The model answers the question: Is today's production of teacher candidates sustainable in the long term?

Projection years
Equation (8) is central to the model for understanding how the population of teachers develops over time.

(8) $$Population_U^{t+1,a+1,k} = Population_U^{t,a,k}$$

$$base\ year + 1 \leq t < end\ year,\ \ a < 75$$

Equation (8) updates the population of teachers for each projection year, i.e., the base year + 1 and onwards to the final projection year. It increments the age by one year for each projection year *t*.

Processes:

- Ageing of the population. For each year t in the projection period, the age of each teacher is incremented by one year by moving the population from age group *a* in year *t* to age group *a+1* in year *t+1*.
- Retirement. The oldest in the population retire from the labor market when the age is beyond the upper age limit for the workforce (at age 75).
- Continuity in the population. By incrementing age, it ensures that the population of teachers is updated through natural ageing while also allowing for departures through retirement and new candidates through the education system (see equation (9)).

Equation (8) is crucial for projecting the supply of teachers over time. Without the ageing and retirement processes described above, it would be impossible to make projections for the supply of teachers. Using this simple method, the model ensures that the demographic composition of the teacher workforce changes over time, which is key to understanding current and future challenges for the teaching profession. Future supply is crucial for a good understanding of the teacher workforce.

Equation (9) incorporates newly graduated candidates from (7) to the existing population of teachers in equation (8) based on age and gender. The inflow of new labor is constant every year.

(9) $$Population\ with\ new\ teacher\ graduates_U^{t,a,k} = Population_U^{t,a,k} + Graduates_U^{a,k}$$

$$base\ year + 1 \leq t \leq end\ year$$

Supply per year for each age and gender
Equation (10) calculates the supply of teachers by multiplying the updated population in (9) with employment rates and average full-time equivalents distributed by education, age, and gender. It provides the labor supply for a specific year.

(10) $$Supply_U^{t,a,k} = (Population\ with\ new\ graduats)_U^{t,a,k} * (Employmentrate)_U^{a,k} * (Average\ fulltime\ equvivalent)_U^{a,k}$$

$$base\ year \leq t \leq end\ year$$

Note that the employment rate from (1) and the average full-time equivalents from (2) are constant and equal to the baseline year over the projection period. It simplifies the model, and at the same time, they have been relatively constant over time. Most individuals work until the formal retirement age, but the supply decreases with age. The model considers variation between genders, ages, and educations from the baseline year.

3.2. Demand side

Teacher full-time equivalents in the baseline year by sector
Equation (11) defines the demand in the base year for each teacher education (U) and employment sector (S) as the sum of the number of employed men and women multiplied by their respective full-time equivalents.

(11) $$Demand_{U,S}^{t=base\ year} = Employed_{U,S}^{k=male,\ t=base\ year} * (Average\ fulltime\ equivalent)_{U,S}^{k=male,\ t=base\ year} + Employed_{U,S}^{k=female,\ t=base\ year} * (Average\ fulltime\ equivalent)_{U,S}^{k=female,\ t=base\ year}$$

This equation provides the demand for teacher labor by employment sector based on the number of full-time equivalents in the baseline year. It is the starting point for assessing the future balance between supply and demand in the education sector and planning for future needs.

Recall supply equals demand in the base year in the reference scenario.

Number of children, stay duration, and user index in kindergartens in the base year
Equation (12) calculates the total number of children in each of the four age groups (g) across categories of stay duration (o) in the kindergarten sector (S=1) by summing the number of children in the various age groups over the stay duration categories, *o*.

(12) $$Users_{S=1,g} = \sum_i Users_{S=1,i}$$

The equation provides the number of children in kindergartens by age groups across stay duration categories to better plan for the necessary capacity and resources in kindergartens.

Equation (13) calculates the total number of users in each age group (g) and stay duration category (o) in kindergartens (S=1) by summing the number of children in g in each stay duration category (o).

(13) $$Users_{S=1,g,o} = \sum_i Users_{S=1,i}$$

By summing over the stay duration categories, equation (13) provides information on the total number of kindergarten children within each age group and stay duration category.

Equation (14) determines the average hours for each stay duration category (o). It is the average minimum and maximum hours for each stay duration category and estimates the time a child spends in kindergarten.

(14) $$Hours_o = Hours_{Min,o} + \frac{Hours_{Max,o} - Hours_{Min,o}}{2}$$

The equation is essential for understanding and planning staffing needs and resource allocation in the kindergarten sector. By knowing the average number of hours for children in different stay duration categories, we can better estimate the need for teachers.

Equation (15) calculates the user index for each of the age groups (g) by applying a multiplier (α) that reflects the need for more staff per child the younger the children are. It is done by multiplying the multiplier with the number of children in each stay duration category (from equation (13)) with the average number of hours (from equation (14)) and then dividing the result by the product of the number of children in g (from equation (12)) and the maximum stay duration (42,5 hours).

(15) $$(User\ index)_{S=1,g} = \frac{\alpha * Users_{S=1,g,o} * Hours_o}{Users_{S=1,g} * 42{,}5} \qquad \alpha = \begin{cases} 2 & if\ g \leq 2 \\ 1{,}5 & if\ g = 3 \\ 1 & if\ g = 4 \end{cases}$$

Equation (15) is a factor that is important for understanding the relationship between the number of children in kindergarten and the need for staff, especially considering how many hours they spend in kindergarten, as well as the age and higher teacher density among younger than older children.

Equation (16) adjusts the number of children in the base year for each age group by multiplying the number of children in kindergarten in (12) with the user index in (15). It measures the number of users in each age group adjusted for factors such as stay duration, o, and the multiplier, α.

(16) $$(User\ adjusted)_{S=1,g} = Users_{S=1,g} * (User\ index)_{S=1,g}$$

This equation provides a more nuanced understanding of the demand for teachers in the kindergarten sector by considering the number of users adjusted for how specific characteristics (o and α) affect the need for teachers. It is crucial for effective planning and allocation of resources in the kindergarten sector.

Students and other users of teachers in the base year

Equation (17) calculates the number of users in the primary schools (S=2) in the baseline year. It is done by summing all individuals in the population between 6 and 15 years.

(17) $$Users_{S=2} = \sum_{a=6}^{a=15} Population$$

This equation shows the extent of the demand for teachers in primary schools based on the current population structure. By knowing the number of children of primary school age, the education authorities and school planners can anticipate the need for teacher resources and other necessary facilities to meet this demand.

Equation (18) calculates the number of users in the base year within “other education” (S=5), which includes adult education and upper vocational education. This group encompasses all individuals from 0 to 99 years old and reflects educational services outside the traditional school age.

(18) $$Users_{S=5} = \sum_{a=0}^{a=99} Population$$

The equation is essential for highlighting the need for teachers in education services aimed at adults and other groups outside the traditional school age.

Equation (19) calculates the number of users of teachers outside the sector (S=6) in the base year, which includes all individuals in the population between 0 and 99 years.

(19) $$Users_{S=6} = \sum_{a=0}^{a=99} Population$$

This equation is crucial for identifying the demand for teacher full-time equivalents outside the sector. It is a measure of out-of-sector mobility among teachers. Equation (18) highlights that the model considers that not everyone with teacher education works as a teacher.

The coverage rate in sectors 2, 5, and 6 (equations (17) - (19)) is 1. Here, there is only one user group. For sectors 3 (secondary education) and 4 (higher education), the number of users in the base year is based on numbers from StatBank Norway. Here, as well as for the kindergarten sector, the coverage rates are less than one since not everyone in the various age groups linked to these sectors is in kindergarten, secondary education, or higher education.

Demographic development

Equation (20) calculates the population (total number of individuals) for each of the age-defined user groups (g) in the six employment sectors from the baseline year and throughout the projection period. We sum the population based on Statistics Norway's population projections within the age group of each user group in the sectors.

(20) $$Population_g^t = \sum_i Population_i^t \qquad base\ year \leq t \leq end\ year$$

By calculating the population for each user group over time, equation (20) indicates how the population structure and size change and what this means for the demand for teachers in the different sectors. It is crucial for planning according to society's needs.

Equation (21) calculates the number of users in each projection year for each age group in the six employment sectors by adjusting the number of users from the previous year with the change in population for the respective user groups.

(21) $$Users_g^t = Users_g^{t-1} * \frac{Population_g^t}{Population_g^{t-1}} \qquad base\ year + 1 \leq t \leq end\ year$$

The equation makes it possible to track changes in demand over time by demographic changes. It ensures that the supply aligns with future needs and population trends.

Equation (22) sums over individuals in each age group to find the total number of users in each employment sector for each projection year. It measures the total number of users - adjusted for group-specific characteristics reflected in the user index (only for the kindergarten sector).

(22) $$Users_S^t = \sum_i Users_i^t \qquad base\ year \leq t \leq end\ year$$

By summing the number of users, equation (22) provides an overall picture of demand in each sector. It measures how demand changes over time in different parts of the population.

Equation (23) calculates the demographic component for each employment sector (S) for each projection year (t), which measures the change in demand based on population development.

(23) $$(Demografic\ component)_S^t = \frac{Users_S^t}{Users_S^{base\ year}} \qquad base\ year + 1 \leq t \leq end\ year$$

The demographic component reflects how the demand in each sector changes over time due to population changes. Compared with the baseline year, it provides insight into how demographic trends affect the need for teachers (for a given teacher density) in the different sectors.

Initial teacher shortage, standard changes, and demand per sector

Equation (24) determines the demand in the baseline year for each employment sector (S) and education (U). If necessary, in alternative scenarios, as opposed to the reference scenario, it corrects for the teacher shortage in the base year and a sectorial standard change (e.g., changed teacher density). Therefore, the demand in the baseline year is the sum of the demand found in (11) and any addition based on the initial teacher shortage multiplied by a possibly specified standard change:

(24) $$Demand\ adjusted_{U,S}^{base\ year} = \left(Demand_{U,S}^{base\ year} + Teacher\ shortage_{U,S}^{base\ year}\right) * (Standard\ \ change)_S^{base\ year}$$

Equation (24) indicates the need for teacher labor in the different sectors in the baseline year and how this need can change due to the initial teacher shortage and standard changes. It is the basis for making precise projections of future demand for teachers.

Equation (25) calculates the demand in the different employment sectors for each projection year based on the demand in the baseline year adjusted for demographic changes and any standard changes. The demand in the projection years is thus the demand in the base year found in (24) multiplied by the demographic component for each sector and year from (23) and possibly a specified future standard change:

(25) $$Demand\ adjusted_{U,S}^{t} = Demand_{U,S}^{base\ year} * (Demografic\ \ component)_{S}^{t} * (Standard\ change)_{S}^{t}$$
$$base\ year + 1 \leq t \leq end\ year$$

Equation (25) shows how the population changes and changes in policy, technology, or the economy can affect the need for teachers over time. It ensures that the education system can meet society's needs for qualified teachers.

3.3. Difference between supply and demand

Equation (26) sums the supply calculated in equation (10) over age (a) and gender (k), providing a measure of the supply of teachers from each education group in each projection year.

(26) $$Supply_{U}^{t} = \sum_{a=18}^{a=74} \sum_{k} Supply_{U}^{t,a,k} \qquad base\ year \leq t \leq end\ year$$

Supply is critical for assessing whether enough qualified teachers are available to meet the demand for teachers. By understanding the total supply, education planners and authorities can take necessary measures to balance supply and demand and contribute to an education sector with enough qualified teachers.

Equation (27) sums the demand in equations (24) and (25) over the six employment sectors (S), providing a comprehensive picture of the demand for teachers from each education group in each year going forward. Demand can be adjusted for initial teacher shortage and standard change.

(27) $$Demand_{U}^{t} = \sum_{S=1}^{S=6} Demand_{U,S}^{t} \qquad base\ year \leq t \leq end\ year$$

By calculating the total demand, the equation provides insight into the overall need for teachers in society, which is crucial for ensuring that kindergartens and schools have enough qualified personnel to meet future needs or to identify any imbalances between supply and demand, addressed through policy or other measures.

Equation (28) calculates the difference between supply and demand for each of the seven educations (U) for each year in the projection period (t), providing insight into potential imbalances between the availability of teachers and the need for such resources in the long run.

(28) $$Difference_{U}^{t} = Supply_{U}^{t} - Demand_{U}^{t} \qquad base\ year \leq t \leq end\ year$$

A positive difference indicates a surplus, while a negative difference indicates a deficit. This calculation is crucial for strategic planning within the education sector to correct imbalances and ensure that the supply of teachers meets society's needs.

A surplus or deficit means improvement or worsening if not correct for the initial teacher shortage in equation (24). If one does not consider the teacher shortage in the baseline year, the model assumes that supply equals demand, even if this is not necessarily the case. See Appendix B.

4. The implementation of LÆRERMOD in Python

The key technical change in LÆRERMOD 2.0 is migrating the model to Python. The content of this chapter is exclusively produced manually, without the use of AI. This Python version aligns with the equations in chapter 3.

We use the Pandas library to handle data structures flexibly. The model also utilizes source code control in Git for robustness.

```
import pandas as pd
from functools import reduce
pd.options.display.multi_sparse = False

WelcomeMessage = """
Welcome to the Python version of LÆRERMOD!

+-------------------------------------------------------------+
|    LÆRERMOD calculates supply and demand for the following 7  |
|     groups of teachers:                                     |
+-------------------------------------------------------------+
| 1. Kindergarten teachers                                    |
| 2. Primary- and middle school teachers                      |
| 3. Lecturers                                                |
| 4. PPE (Practical Pedagogical Education)                    |
| 5. Teacher education in practical and aesthetic subjects    |
| 6. Vocational teachers                                      |
| 7. PPE Vocational                                           |
+-------------------------------------------------------------+
"""

print(WelcomeMessage)

# ************************************************************************************************ #
# Start and end year for the projection.                                                           #
# ************************************************************************************************ #

BaseYear = 2024
EndYear = 2060

# ************************************************************************************************ #
# Reading input files. See Appendix 1 for source data.                                             #
# ************************************************************************************************ #

AgeDistributed = pd.read_fwf('inputdata/agedistributed.txt')

AgeDistributedStudents = pd.read_fwf('inputdata/agedistributedstudents.txt')
CandidateProduction = pd.read_fwf('inputdata/candidateproduction.txt')

SectorDistributed = pd.read_fwf('inputdata/sectordistributed.txt')

People = pd.read_fwf('inputdata/mmmm.txt')

DemographicsGroup1 = pd.read_fwf('inputdata/number_children_kindergartens.txt')
DemographicsGroup3 = pd.read_fwf('inputdata/number_students_secondary.txt')
DemographicsGroup4 = pd.read_fwf('inputdata/number_students_highereducation.txt')

TeacherShortage = pd.read_fwf('inputdata/teachershortage.txt')

StandardChange = pd.read_fwf('inputdata/change_standard.txt')

# ************************************************************************************************ #
# Creating row labels on existing columns for later use.                                           #
# ************************************************************************************************ #

AgeDistributed.set_index(['Education'], inplace=True)
AgeDistributedStudents.set_index(['Education'], inplace=True)
CandidateProduction.set_index(['Education'], inplace=True)
SectorDistributed.set_index(['Education', 'Sector'], inplace=True)
People.set_index(['Age', 'Gender'], inplace=True)

# ************************************************************************************************ #
# Creating a constant with abbreviations for the educations included in the model.                 #
# ************************************************************************************************ #
```

```
Educations = ['ba', 'gr', 'lu', 'ph', 'pe', 'yr', 'py']

# ********************************************************************************************** #
# Creating dictionaries for later use.                                                           #
# ********************************************************************************************** #

PeopleSector = {}
UserGroup = {}
DemographicsSector = {}
DemographicsGroup = {}
SumDemographicsGroup = {}
Users = {}
```

4.1. Supply side

```
# ********************************************************************************************** #
# Initial population of teachers.                                                                #
# ********************************************************************************************** #

# ********************************************************************************************** #
# Calculating employment rate.                                                                   #
# This is Equation 1 in the model.                                                               #
# ********************************************************************************************** #

AgeDistributed['EmploymentRate'] = AgeDistributed.apply(lambda row: row['Employed'] /
                                                                    row['Count']
                                                    if row['Count'] > 0 else 0, axis=1)

# ********************************************************************************************** #
# Copying this into a table and removing columns that are now redundant.                         #
# ********************************************************************************************** #

Population = AgeDistributed.copy()
AgeDistributed.drop(['Count', 'Employed'], axis=1, inplace=True)

# ********************************************************************************************** #
# Finding average full-time equivalents.                                                         #
# This is Equation 2 in the model.                                                               #
# ********************************************************************************************** #

Population['AverageFullTimeEquivalents'] = Population.FullTimeEquivalents / Population.Employed

# ********************************************************************************************** #
# Indicating that this is the population in the base year and removing redundant columns.        #
# ********************************************************************************************** #

Population['Year'] = BaseYear
Population.drop(['Employed', 'EmploymentRate', 'AverageFullTimeEquivalents'],
                axis=1, inplace=True)

# ********************************************************************************************** #
# Projection of the initial population. Year 2 to end year. Based on statistics from             #
# the base year.                                                                                 #
# ********************************************************************************************** #

# ********************************************************************************************** #
# Candidate production: Calculating the number of first-year students for each education.        #
# This is Equation 3 in the model.                                                               #
# ********************************************************************************************** #

NumberOfFirstYearStudents = AgeDistributedStudents.groupby(
                                ['Education']).sum().rename(columns={'All': 'Total'})

# ********************************************************************************************** #
# Copying the total number of students for the relevant education into a new column in           #
# the AgeDistributedStudents table. Adding a variable for gender.                                #
# ********************************************************************************************** #

AgeDistributedStudents = AgeDistributedStudents.merge(NumberOfFirstYearStudents['Total'],
                                                      how='inner', on='Education')
NewStudents = pd.concat([AgeDistributedStudents, AgeDistributedStudents],
                        keys=[1, 2], names=['Gender']).reset_index()

# ********************************************************************************************** #
# Calculating the proportion of students for each age and gender.                                #
# This is Equation 4 in the model.                                                               #
# ********************************************************************************************** #
```

```
NewStudents['ProportionStudentsByAge'] = NewStudents.apply(lambda row: row['Men'] /
                                                           row['Total'] if row['Gender']==1
                                                           else row['Women'] /
                                                           row['Total'], axis=1)

# ********************************************************************************************* #
# Indicating that the number of students is constant in each projection year.                   #
# ********************************************************************************************* #

CandidateProduction = CandidateProduction.merge(
    pd.concat([pd.DataFrame({'Year': list(range(BaseYear, EndYear+1))})] * 7,
              keys=Educations, names=['Education']), how='inner', on='Education')

# ********************************************************************************************* #
# Calculating the number of annual candidates using new students and completion percentages.    #
# This is Equation 5 in the model.                                                              #
# ********************************************************************************************* #

CandidateProduction['Candidates'] = (CandidateProduction.NumberOfNewStudents *
                                     CandidateProduction.CompletionPercentage)

# ********************************************************************************************* #
# Indicating that the number of candidates should be constant during the projection period.     #
# ********************************************************************************************* #

Candidates = NewStudents.merge(CandidateProduction, how='inner', on=['Education'])

# ********************************************************************************************* #
# Calculating the age at graduation. Ensuring that the graduation age is named the same as in   #
# the table to which the rows will be added later.                                              #
# This is Equations 6 and 7 in the model.                                                       #
# ********************************************************************************************* #

Candidates['Age'] = (Student.Age +
                     StudyLength)
Candidates['GraduatesByAge'] = (Candidates.Candidates *
                                Candidates.ProportionStudentsByAge)

# ********************************************************************************************* #
# Copying the population in the base year, calculated in equation 2, into a new table which     #
# will be the starting point for the calculations.                                              #
# ********************************************************************************************* #

PopulationCurrentYear = Population.copy()

# ********************************************************************************************* #
# For each projection year, the population will get one year older and new candidates added.    #
# ********************************************************************************************* #

for t in range(BaseYear + 1, EndYear + 1):

    # ***************************************************************************************** #
    # Retirement (for the initial population and candidates).                                   #
    # ***************************************************************************************** #

    # ***************************************************************************************** #
    # For each year, the age is incremented.                                                    #
    # This is Equation 8 in the model.                                                          #
    # ***************************************************************************************** #

    PopulationCurrentYear.Age += 1

    # ***************************************************************************************** #
    # Candidates by age and gender found in equations 6 and 7 are added to the table.           #
    # ***************************************************************************************** #

    PopulationCurrentYear = PopulationCurrentYear.merge(
                                Candidates[Candidates['Year'] == t].copy(),
                                           how='outer',
                                           on=['Education', 'Gender', 'Age'])

    # ***************************************************************************************** #
    # Graduates by age and gender from Equation 7 are added to the population.                  #
    # This is Equation 9 in the model.                                                          #
    # ***************************************************************************************** #

    PopulationCurrentYear.Count = (PopulationCurrentYear.Count.fillna(0) +
                                   PopulationCurrentYear.GraduatesByAge.fillna(0))
```

```
    # ********************************************************************************************* #
    # Indicating that this should be the population in the projection year.                         #
    # ********************************************************************************************* #

    PopulationCurrentYear['Year'] = t

    # ********************************************************************************************* #
    # The population in the projection year is added as a new cohort.                               #
    # ********************************************************************************************* #

    Columns = ['Education', 'Gender', 'Age', 'Count', 'Year']
    Population = pd.concat([Population, PopulationCurrentYear[Columns]])

    # ********************************************************************************************* #
    # Copying the population in the projection year to the table for the next projection year.      #
    # ********************************************************************************************* #

    PopulationCurrentYear = Population[Population['Year']==t].copy()

# ************************************************************************************************* #
# Retrieving EmploymentRate and AverageFullTimeEquivalents calculated for                           #
# the initial population in Equations 6 and 7. Indicating that this should be the supply table.     #
# ************************************************************************************************* #

Supply = Population.merge(AgeDistributed, how='left', on=['Education', 'Gender', 'Age'])

# ************************************************************************************************* #
# Calculating supply.                                                                               #
# This is Equation 10 in the model.                                                                 #
# ************************************************************************************************* #

Supply['Supply'] = Supply.Count * Supply.EmploymentRate * Supply.AverageFullTimeEquivalents
```

4.2. Demand side

```
# ************************************************************************************************* #
# Initial population of teachers.                                                                   #
# ************************************************************************************************* #

# ************************************************************************************************* #
# Calculating employed in the base year, i.e., supply. The demand in the base year is set           #
# equal to this.                                                                                    #
# This is Equation 11 in the model.                                                                 #
# ************************************************************************************************* #

SectorDistributed = pd.DataFrame({'Demand': ((SectorDistributed.EmployedMen *
                                              SectorDistributed.AverageFullTimeEquivalentMen) +
                                             (SectorDistributed.EmployedWomen *
                                             SectorDistributed.AverageFullTimeEquivalentWomen)),
                                  'Year': BaseYear})

# ************************************************************************************************* #
# Creating an empty table for demand where each of the 7 educations is included.                    #
# ************************************************************************************************* #

Demand = pd.DataFrame({'Education': Educations, 'Demand': 0})

# ************************************************************************************************* #
# For each of the 7 educations and each of the 6 sectors, the values found in                       #
# Equation 11 are copied into the table with demand. This transposes the table.                     #
# ************************************************************************************************* #

for S in range(1, 7):
    Demand[f'DemandSector{S}'] = SectorDistributed.Demand[
        SectorDistributed.Demand.index.get_level_values('Sector') == S].reset_index(drop=True)

# ************************************************************************************************* #
# Projection years. Finding the number of users in the base year to calculate coverage rates        #
# and densities. Growth in the number of users forward based on SSB's national                      #
# population projections.                                                                           #
# ************************************************************************************************* #

# ************************************************************************************************* #
# Creating 6 empty tables to be filled with the number of users in each sector.                     #
# ************************************************************************************************* #
```

```
UserGroup[1] = pd.DataFrame({'ToAge': [0, 2, 2, 3, 5, 5],
                             'Age': range(0, 6)})
UserGroup[2] = pd.DataFrame({'ToAge': [15] * 10,
                             'Age': range(6, 16)})
UserGroup[3] = pd.DataFrame({'ToAge': [15] * 16 + list(range(16, 25)) + [49] * 25,
                             'Age': range(0, 50)})
UserGroup[4] = pd.DataFrame({'ToAge': list(range(19, 30)) + [34] * 5 + [39] * 5 +
                                      [44] * 5 + [49] * 5,
                             'Age': range(19, 50)})
UserGroup[5] = pd.DataFrame({'ToAge': 99,
                             'Age': range(0, 100)})
UserGroup[6] = pd.DataFrame({'ToAge': 99,
                             'Age': range(0, 100)})

# ********************************************************************************************* #
# Summarizing the number of children in kindergartens in each user group by average stay.      #
# This is Equations 12 and 13 in the model.                                                     #
# ********************************************************************************************* #

ChildrenGroup1 = pd.DataFrame({'Users': DemographicsGroup1.Age0,
                               'Hours': DemographicsGroup1.HoursMin + (
                                        (DemographicsGroup1.HoursMax -
                                         DemographicsGroup1.HoursMin) / 2)})
ChildrenGroup2 = pd.DataFrame({'Users': DemographicsGroup1.Age1 + DemographicsGroup1.Age2,
                               'Hours': DemographicsGroup1.HoursMin + (
                                        (DemographicsGroup1.HoursMax -
                                         DemographicsGroup1.HoursMin) / 2)})
ChildrenGroup3 = pd.DataFrame({'Users': DemographicsGroup1.Age3,
                               'Hours': DemographicsGroup1.HoursMin + (
                                        (DemographicsGroup1.HoursMax -
                                         DemographicsGroup1.HoursMin) / 2)})
ChildrenGroup4 = pd.DataFrame({'Users': DemographicsGroup1.Age4 + DemographicsGroup1.Age5,
                               'Hours': DemographicsGroup1.HoursMin + (
                                        (DemographicsGroup1.HoursMax -
                                         DemographicsGroup1.HoursMin) / 2)})

# ********************************************************************************************* #
# Creating an empty table to be filled with the number of users in the kindergarten sector.     #
# ********************************************************************************************* #

DemographicsGroup[1] = pd.DataFrame(columns=['FromAge', 'ToAge', 'Users', 'UserIndex'])

# ********************************************************************************************* #
# Calculating users of kindergarten in each of the 4 user groups and user indices for these.    #
# This is Equation 14 and Equation 15 in the model.                                             #
# ********************************************************************************************* #

DemographicsGroup[1].loc[len(DemographicsGroup[1].index)] = [0, 0, ChildrenGroup1.Users.sum(),
                                                             (2 * ChildrenGroup1.Users.
                                                              mul(ChildrenGroup1.Hours.values).sum()) /
                                                             (ChildrenGroup1.Users.sum() * 42.5)]
DemographicsGroup[1].loc[len(DemographicsGroup[1].index)] = [1, 2, ChildrenGroup2.Users.sum(),
                                                             (2 * ChildrenGroup2.Users.
                                                              mul(ChildrenGroup2.Hours.values).sum()) /
                                                             (ChildrenGroup2.Users.sum() * 42.5)]
DemographicsGroup[1].loc[len(DemographicsGroup[1].index)] = [3, 3, ChildrenGroup3.Users.sum(),
                                                             (1.5 * ChildrenGroup3.Users.
                                                              mul(ChildrenGroup3.Hours.values).sum()) /
                                                             (ChildrenGroup3.Users.sum() * 42.5)]
DemographicsGroup[1].loc[len(DemographicsGroup[1].index)] = [4, 5, ChildrenGroup4.Users.sum(),
                                                             (1 * ChildrenGroup4.Users.
                                                              mul(ChildrenGroup4.Hours.values).sum()) /
                                                             (ChildrenGroup4.Users.sum() * 42.5)]

# ********************************************************************************************* #
# Updating the numbers for the number of kindergarten users in each of the 4 user groups when   #
# accounting for user indices.                                                                  #
# This is Equation 16 in the model.                                                             #
# ********************************************************************************************* #

DemographicsGroup[1].loc[0] = [0, 0, DemographicsGroup[1].loc[0].Users *
                                     DemographicsGroup[1].loc[0].UserIndex,
                               DemographicsGroup[1].loc[0].UserIndex]
DemographicsGroup[1].loc[1] = [1, 2, DemographicsGroup[1].loc[1].Users *
                                     DemographicsGroup[1].loc[1].UserIndex,
                               DemographicsGroup[1].loc[1].UserIndex]
DemographicsGroup[1].loc[2] = [3, 3, DemographicsGroup[1].loc[2].Users *
                                     DemographicsGroup[1].loc[2].UserIndex,
```

```
                                DemographicsGroup[1].loc[2].UserIndex]
DemographicsGroup[1].loc[3] = [4, 5, DemographicsGroup[1].loc[3].Users *
                                      DemographicsGroup[1].loc[3].UserIndex,
                                DemographicsGroup[1].loc[3].UserIndex]

# ********************************************************************************************* #
# Calculating students in primary school.                                                       #
# This is Equation 17 in the model.                                                             #
# ********************************************************************************************* #

DemographicsGroup[2] = pd.DataFrame({'FromAge': 6,
                                     'ToAge': 15,
                                     'Users': People.query('Age>=6 and Age<=15')
                                     [str(BaseYear)].sum(), 'UserIndex': 1.0}, index=[0])

# ********************************************************************************************* #
# Copying the users in Sector 3 and Sector 4 that were read earlier.                            #
# ********************************************************************************************* #

DemographicsGroup[3] = DemographicsGroup3.copy()
DemographicsGroup[4] = DemographicsGroup4.copy()

# ********************************************************************************************* #
# Calculating users of others in the sector (adult education, vocational schools, etc.).        #
# This is Equation 18 in the model.                                                             #
# ********************************************************************************************* #

DemographicsGroup[5] = pd.DataFrame({'FromAge': 0,
                                     'ToAge': 99,
                                     'Users': People[str(BaseYear)].sum(),
                                     'UserIndex': 1.0}, index=[0])

# ********************************************************************************************* #
# Calculating users outside the sector.                                                         #
# This is Equation 19 in the model.                                                             #
# ********************************************************************************************* #

DemographicsGroup[6] = pd.DataFrame
DemographicsGroup[6] = pd.DataFrame({'FromAge': 0,
                                     'ToAge': 99,
                                     'Users': People[str(BaseYear)].sum(),
                                     'UserIndex': 1.0}, index=[0])

# ********************************************************************************************* #
# Calculating the demographic development in each employment sector.                            #
# ********************************************************************************************* #

for S in range(1, 7):

    # ***************************************************************************************** #
    # Creating an empty table for the population in the current sector.                         #
    # ***************************************************************************************** #

    PeopleSector[S] = pd.DataFrame()

    # ***************************************************************************************** #
    # Finding the population from the population projections for the user groups in the group.  #
    # This is Equation 20 in the model.                                                         #
    # ***************************************************************************************** #

    PeopleSector[S] = UserGroup[S].merge(People, how='inner',
                                         on='Age').groupby(['ToAge']).sum()

    # ***************************************************************************************** #
    # Indicating a row label for the maximum age of the user group.                             #
    # ***************************************************************************************** #

    DemographicsGroup[S] = DemographicsGroup[S].set_index(['ToAge'])

    # ***************************************************************************************** #
    # Indicating that the number of read users should be users in the base year.                #
    # ***************************************************************************************** #

    DemographicsGroup[S]['Users' + str(BaseYear)] = DemographicsGroup[S].Users
```

```
    # ********************************************************************************************* #
    # Calculating the number of users in each projection year.                                      #
    # This is Equation 21 in the model.                                                             #
    # ********************************************************************************************* #

    for t in range(BaseYear + 1, EndYear + 1):
        DemographicsGroup[S][f'Users{t}'] = \
        DemographicsGroup[S][f'Users{t-1}'] * (PeopleSector[S][str(t)] /
                                               PeopleSector[S][str(t-1)])

    # ********************************************************************************************* #
    # Create an empty table for users in each projection year.                                      #
    # ********************************************************************************************* #

    SumDemographicsGroup[S] = pd.DataFrame()

    # ********************************************************************************************* #
    # Calculating the sum of users in each projection year.                                         #
    # This is Equation 22 in the model.                                                             #
    # ********************************************************************************************* #

    for t in range(BaseYear, EndYear + 1):
        SumDemographicsGroup[S][f'SumUsers{t}'] = [DemographicsGroup[S][f'Users{t}'].sum()]

    # ********************************************************************************************* #
    # Creating an empty table to contain the demographic development in the sector.                 #
    # ********************************************************************************************* #

    DemographicsSector[S] = pd.DataFrame({'Year': [BaseYear], f'DemographicComponent{S}': [1]})

    # ********************************************************************************************* #
    # Calculating the demographic development for each projection year for each user group.         #
    # This is Equation 23 in the model.                                                             #
    # ********************************************************************************************* #

    for t in range(BaseYear + 1, EndYear + 1):
        NextCohort = pd.DataFrame({
            'Year': t,
            f'DemographicComponent{S}': (SumDemographicsGroup[S][f'SumUsers{t}'] /
                                         SumDemographicsGroup[S][f'SumUsers{BaseYear}'])})

        # ***************************************************************************************** #
        # The demographic development in the projection year is added as a new cohort in the        #
        # table with the demographic development in the sector.                                     #
        # ***************************************************************************************** #

        DemographicsSector[S] = pd.concat([DemographicsSector[S], NextCohort],
                                          ignore_index=True)

# ************************************************************************************************* #
# Copying the tables with the demographic development in each sector together with                  #
# any standard change into the same table (alternative path).                                       #
# ************************************************************************************************* #

DemographicIndex = StandardChange.copy()
for Sector in range(1, 7):
    DemographicIndex = pd.merge(DemographicIndex, DemographicsSector[Sector])

# ************************************************************************************************* #
# Adding the constant indicating the 7 educations in the model to the table.                        #
# ************************************************************************************************* #

DemographicIndex = pd.concat([DemographicIndex] * 7, keys=Educations, names=['Education'])

# ************************************************************************************************* #
# Teacher densities based on the base year. Held constant.                                          #
# ************************************************************************************************* #

# ************************************************************************************************* #
# Copying the table with the demographic development in each sector, the transposed table           #
# with demand found in equation 11, and any specified teacher shortage into the same table.        #
# ************************************************************************************************* #

Demand = reduce(lambda left, right: pd.merge(left, right, on=['Education'], how='outer'),
                [DemographicIndex, Demand, TeacherShortage]).set_index(['Education', 'Year'])
```

```
# ********************************************************************************************** #
# Calculating demand.                                                                            #
# This is Equation 24 and Equation 25 in the model.                                              #
# ********************************************************************************************** #

for S in range(1, 7):
    Demand['Demand'] = (Demand['Demand'] +
                        (Demand[f'DemandSector{S}'] +
                         Demand[f'TeacherShortageSector{S}']) *
                         Demand[f'DemographicComponent{S}'] *
                         Demand[f'StandardChange{S}'])
```

4.3. Difference between supply and demand

```
# ********************************************************************************************** #
# Combining supply and demand.                                                                   #
# This is Equation 26 and Equation 27 in the model.                                              #
# ********************************************************************************************** #

SupplyDemand = pd.concat([pd.DataFrame({'Supply': SectorDistributed.Demand,
                                        'Year': BaseYear}).groupby(['Education', 'Year'],
                                                                   as_index=True).sum(),
                          Supply.groupby(['Education', 'Year'], as_index=True).sum().
                          query('Year > @BaseYear')]).merge(Demand, how='outer',
                                                            on=['Education', 'Year'])

# ********************************************************************************************** #
# Calculating the difference.                                                                    #
# This is Equation 28 in the model.                                                              #
# ********************************************************************************************** #

SupplyDemand['Difference'] = SupplyDemand.Supply - SupplyDemand.Demand

# ********************************************************************************************** #
# Printing the results.                                                                          #
# ********************************************************************************************** #

Order = {'ba': 1, 'gr': 2, 'lu': 3, 'ph': 4, 'pe': 5, 'yr': 6, 'py': 7}

SupplyDemand = SupplyDemand[['Supply', 'Demand', 'Difference']]
SupplyDemand = SupplyDemand.sort_values(by=['Education', 'Year'],
                                        key=lambda x: x.map(Order))
SupplyDemand.rename(index={'ba': 'Kindergarten teachers',
                           'gr': 'Primary- and middle school teachers',
                           'lu': 'Lecturers',
                           'ph': 'PPE (Practical Pedagogical Education)',
                           'pe': 'Teacher education in practical and aesthic subjects',
                           'vr': 'Vocational Teachers',
                           'py': 'PPE Vocational'}, inplace=True)

SupplyDemand.round(0).astype(int).to_csv('results/Laerermod.csv')
SupplyDemand.round(0).astype(int).to_excel('results/Laerermod.xlsx')
print(SupplyDemand.round(0).astype(int).to_string())

print('\nLærermod is now complete, welcome back.\n')
```

References

Gunnes, T., Ekren, R. og Arnesen, H. S. (2023). LÆRERMOD 2020-2040: Tilbud og etterspørsel for fem grupper av lærerutdanninger. SSB, Rapporter 13.

Gunnes, T., Ekren, R. og Steffensen, K. (2018). LÆRERMOD 2016-2040: Fremtidig tilbud og etterspørsel for fem typer lærere. SSB, Rapporter 35.

Gunnes, T. og Knudsen, P. (2016). LÆRERMOD: Forutsetninger og likninger. SSB, Notat 25.

Appendix A: Data in LÆRERMOD

LÆRERMOD uses aggregated data based on individual-level register data from Statistics Norway and information from StatBank Norway. We use register data and statistics from the baseline year. This section covers the data for both the reference and alternative scenarios separately for the supply and demand sides.

Supply Side

Register Data:

- Education and Labor Market Statistics: We identify the initial population, which consists of everyone with teacher education, and then find the employed among them in the age group 18-74 for each age, gender, and teacher education.
- Student Statistics: Newly enrolled first-year students in teacher education programs, including their age and gender.

From StatBank Norway:

- Completion rates for each teacher education.

Demand Side

Register Data:

Based on Statistics Norway's labor market statistics, we distribute employed teachers in the initial population in the baseline year across six employment areas.

From StatBank Norway:

To calculate coverage rates and teacher densities in the baseline year in each sector, we use the following:

- Number of children in kindergarten by age groups (0, 1-2, 3, and 4-5) and intervals of attendance hours per week, as well as the total number of children aged 0-5.
- Number of students in compulsory education, covering all individuals aged 6-15.
- Number of students in upper secondary education aged 15-49, as well as the total number of individuals aged 15-49.
- Number of students in higher education aged 18-49, as well as the total number of individuals aged 18-49.

We use the middle alternative from Statistics Norway’s population projections in the reference scenario to calculate the growth in the number of users within each sector during the projection period.

Appendix B: Correction for initial teacher shortage in the baseline year

In the reference scenario, the teacher supply equals the teacher demand in the baseline year, such that initial teacher shortage is assumed away. However, we correct the estimated future surplus or deficit in an alternative scenario by considering the initial teacher shortage.

We use Statistics Norway's statistics on employees in kindergarten and schools to account for the initial teacher shortage. It provides information on the number of workers without teacher education in teaching positions at the start year of the calculations. This statistic defines a teacher based on occupation codes but distinguishes between those with and without teacher education. In contrast, LÆRERMOD only includes those with teacher education.

For kindergartens, compulsory schools, and secondary education, the number of teachers needed to fill all teaching positions with trained teachers, i.e., teacher shortage, is specified. We correct for this shortage in the start year on the demand side of the model in alternative scenarios for kindergarten teachers, compulsory school teachers, and various teacher educations qualified to work in secondary schools (e.g., lecturer education, PPE).

Appendix C: The implementation of LÆRERMOD in R

Because many use R instead of Python, we also document LÆRERMOD in R. The implementation of LÆRERMOD in Python, reported in chapter 4, has been uploaded to ChatGPT. Then, we instructed ChatGPT to translate this into R. Although some manual adjustments were necessary, the content of this chapter is thus mainly generated by ChatGPT. The R version of LÆRERMOD produces the same results as the Python version and uses identical input files.

```
#*********************************************************************************************#
# Importing R libraries and printing a welcoming message.                                     #
# ********************************************************************************************#

library(readr)
library(dplyr)
library(tidyr)
library(purrr)
library(tidyverse)
library(openxlsx)
library(writexl)

options(dplyr.summarise.inform = FALSE)

cat("
Welcome to the R version of LÆRERMOD!

+-------------------------------------------------------------+
|    The model LÆRERERMOD calculates supply and               |
|    demand for the following 7 groups of teachers:           |
+-------------------------------------------------------------+
| 1. Kindergarten teachers                                    |
| 2. Primary- and middle school teachers                      |
| 3. Lecturers                                                |
| 4. Practical Pedagogical Education (PPE)                    |
| 5. Teacher education in practical and aesthetic Subjects    |
| 6. Vocational teachers                                      |
| 7. PPE Vocational                                           |
+-------------------------------------------------------------+
\n")

# ********************************************************************************************* #
# Start and end year for the projection.                                                        #
# ********************************************************************************************* #

BaseYear <- 2024
EndYear  <- 2060

# ********************************************************************************************* #
# Reading input files. See Appendix 1 for source data.                                          #
# ********************************************************************************************* #

AgeDistributed <- read.table("inputdata/agedistributed.txt", header = TRUE)

AgeDistributedStudents <- read.table('inputdata/agedistributedstudents.txt', header = TRUE)
CandidateProduction <- read.table('inputdata/candidateproduction.txt', header = TRUE)

SectorDistributed <- read.table('inputdata/sectordistributed.txt', header = TRUE)

People <- read.table('inputdata/mmmm.txt', header = TRUE)

DemographyGroup1 <- read.table('inputdata/number_children_kindergartens.txt', header = TRUE)
DemographyGroup3 <- read.table('inputdata/number_students_secondary.txt', header = TRUE)
DemographyGroup4 <- read.table('inputdata/number_students_highereducation.txt', header = TRUE)

TeacherShortage <- read.table('inputdata/teachershortage.txt', header = TRUE)

StandardChange <- read.table('inputdata/change_standard.txt', header = TRUE)
StandardChange$Year <- paste0("X", as.character(StandardChange$Year))

WorkHourChange <- read.table('inputdata/change_workhour.txt', header = TRUE)

# ********************************************************************************************* #
# Creating row labels on existing columns so they can later be used for linking.                #
# ********************************************************************************************* #
```

```
AgeDistributed <- AgeDistributed %>% mutate(Education = factor(Education))
AgeDistributedStudents <- AgeDistributedStudents %>% mutate(Education = factor(Education))
CandidateProduction <- CandidateProduction %>% mutate(Education = factor(Education))
SectorDistributed <- SectorDistributed %>% mutate(Education = factor(Education),
                                                  Sector = factor(Sector))
People <- People %>% mutate(Age = factor(Age), Gender = factor(Gender))

# ********************************************************************************************** #
# Creating a constant with the abbreviations for each education included in the model.           #
# ********************************************************************************************** #

Educations <- c('ba', 'gr', 'lu', 'ph', 'pe', 'yr', 'py')

# ********************************************************************************************** #
# Creating lists for later filling.                                                              #
# ********************************************************************************************** #

PeopleSector <- list()
UserGroup <- list()
DemographySector <- list()
DemographyGroup <- list()
SumDemographyGroup <- list()
Users <- list()

# ********************************************************************************************** #
# Initial teacher population.                                                                    #
# ********************************************************************************************** #

# ********************************************************************************************** #
# Calculating employment rate.                                                                   #
# This is Equation 1 in the model.                                                               #
# ********************************************************************************************** #

AgeDistributed$EmploymentRate <- ifelse(AgeDistributed$Count > 0,
                                        AgeDistributed$Employed / AgeDistributed$Count,
                                        0)

# ********************************************************************************************** #
# Copying this into a table and removing columns that are now redundant.                         #
# ********************************************************************************************** #

Population <- AgeDistributed
AgeDistributed <- AgeDistributed %>% select(-Count, -Employed)

# ********************************************************************************************** #
# Finding the average full-time equivalents.                                                     #
# This is Equation 2 in the model.                                                               #
# ********************************************************************************************** #

Population$AverageFullTimeEquivalent <- Population$FTEs / Population$Employed

# ********************************************************************************************** #
# Indicates that this is the population in the base year and removes redundant columns.          #
# ********************************************************************************************** #

Population$Year <- BaseYear
Population <- Population %>% select(-Employed, -EmploymentRate, -AverageFullTimeEquivalent)

# ********************************************************************************************** #
# Projecting the initial population. Year 2 to end year. Statistics from the base year.          #
# ********************************************************************************************** #

# ********************************************************************************************** #
# Candidate production:                                                                          #
# Calculate the total number of first-year students for each education.                          #
# This is Equation 3 in the model.                                                               #
# ********************************************************************************************** #

TotalFirstYearStudents <- AgeDistributedStudents %>%
  group_by(Education) %>%
  summarise(Total = sum(All)) %>%
  ungroup()

# ********************************************************************************************** #
# Copies the total number of students for the relevant education into a new column in            #
# the table AgeDistributedStudents. Adds a variable for gender.                                  #
# ********************************************************************************************** #
```

```
AgeDistributedStudents <- AgeDistributedStudents %>%
  inner_join(TotalFirstYearStudents, by = "Education")

NewStudents <- AgeDistributedStudents %>%
  mutate(Gender = 1) %>%
  bind_rows(AgeDistributedStudents %>% mutate(Gender = 2))

# ************************************************************************************************ #
# Calculates the proportion of students by age and gender.                                         #
# This is Equation 4 in the model.                                                                 #
# ************************************************************************************************ #

NewStudents <- NewStudents %>%
  mutate(StudentShareByAge = if_else(Gender == 1,
                                     Men / Total,
                                     Women / Total))

# ************************************************************************************************ #
# Indicates that the number of students is constant in each projection year.                       #
# ************************************************************************************************ #

Years <- data.frame(Year = BaseYear:EndYear)
EducationYears <- expand.grid(Education = Educations, Year = Years$Year)

CandidateProduction <- CandidateProduction %>% inner_join(EducationYears, by = "Education")

# ************************************************************************************************ #
# Calculates the number of annual graduates using new students and completion percentages.         #
# This is Equation 5 in the model.                                                                 #
# ************************************************************************************************ #

CandidateProduction <- CandidateProduction %>%
  mutate(Graduates = NumberOfNewStudents * CompletionPercentage)

# ************************************************************************************************ #
# Indicates that the number of graduates should be constant in the projection period.              #
# ************************************************************************************************ #

Graduates <- inner_join(NewStudents, CandidateProduction,
                        by = "Education", relationship = "many-to-many")

# ************************************************************************************************ #
# Calculates graduation age. Ensures the graduation age is named the same as in the table. The     #
# rows will be added later. This is Equation 6 and Equation 7 in the model.                        #
# ************************************************************************************************ #

Graduates <- Graduates %>%
  mutate(Age = Age + StudyLength,
         GraduatesByAge = Graduates * StudentShareByAge)

# ************************************************************************************************ #
# Copies the population in the base year, calculated in Equation 2, into two new tables that       #
# will be the basis for the calculations.                                                          #
# ************************************************************************************************ #

CurrentYearPopulation <- Population

# ************************************************************************************************ #
# In each projection year, the population ages and new graduates are added.                        #
# ************************************************************************************************ #

for (t in (BaseYear + 1):EndYear) {

    # ******************************************************************************************** #
    # Retirement (for the initial population and graduates).                                       #
    # ******************************************************************************************** #

    # ******************************************************************************************** #
    # For each year, the age of the population is incremented.                                     #
    # This is Equation 8 in the model.                                                             #
    # ******************************************************************************************** #

    CurrentYearPopulation$Age <- CurrentYearPopulation$Age + 1
```

```
    # ************************************************************************************** #
    # Graduates by age and gender from Equations 6 are added to the table.                   #
    # ************************************************************************************** #

    CurrentYearPopulation <- merge(x = CurrentYearPopulation,
                                   y = Graduates[Graduates$Year == t, ],
                                   by = c("Education", "Gender", "Age"),
                                   all = TRUE)

    # ************************************************************************************** #
    # Graduates by age and gender in Equation 7 are added to the population.                 #
    # This is Equation 9 in the model.                                                       #
    # ************************************************************************************** #

    CurrentYearPopulation$Count <- ifelse(is.na(CurrentYearPopulation$Count), 0,
                                          CurrentYearPopulation$Count) +
                                   ifelse(is.na(CurrentYearPopulation$GraduatesByAge), 0,
                                          CurrentYearPopulation$GraduatesByAge)

    # ************************************************************************************** #
    # Indicates that this should be the population in the projection year.                   #
    # ************************************************************************************** #

    CurrentYearPopulation$Year <- t

    # ************************************************************************************** #
    # The population in the projection year is added to the initial population as a new cohort.#
    # ************************************************************************************** #

    Population <- rbind(Population, CurrentYearPopulation[, c("Education", "Gender", "Age",
                                                              "Count", "Year")])

    # ************************************************************************************** #
    # Copies the population in the projection year to the table for the next projection year.  #
    # ************************************************************************************** #

    CurrentYearPopulation <- Population[Population$Year == t, ]
}

# ************************************************************************************** #
# Incorporates the Employment Rate and Average FTEs calculated for the initial population in   #
# Equations 6 and 7. Specifies that this should become the table for the supply.         #
# ************************************************************************************** #

Supply <- merge(x = Population,
                y = AgeDistributed,
                by = c("Education", "Gender", "Age"),
                all.x = TRUE)

# ************************************************************************************** #
# Calculates the supply.                                                                 #
# This is Equation 10 in the model.                                                      #
# ************************************************************************************** #

Supply$Supply <- Supply$Count * Supply$EmploymentRate * Supply$AverageFullTimeEquivalent

# ************************************************************************************** #
# Initial teacher population.                                                            #
# ************************************************************************************** #

# ************************************************************************************** #
# Calculates employed in the base year. Demand in the base year is set equal to supply.  #
# This is Equation 11 in the model.                                                      #
# ************************************************************************************** #

SectorDistributed$Demand <- (SectorDistributed$EmployedMen *
                             SectorDistributed$AverageFullTimeEquivalentMen) +
                            (SectorDistributed$EmployedWomen *
                             SectorDistributed$AverageFullTimeEquivalentWomen)
SectorDistributed$Year <- BaseYear

# ************************************************************************************** #
# Creates an empty table for the demand where each of the 7 educations is included.      #
# ************************************************************************************** #

Demand <- data.frame(Education = Educations, Demand = rep(0, length(Educations)))
```

```
# ********************************************************************************************** #
# For each of the 7 educations and each of the 6 sectors, the values found in Equation 11 are  #
# copied into the table with the demand. This transposes the table.                            #
# ********************************************************************************************** #

for (S in 1:6) {
  Demand[paste0('DemandSector', S)] <- SectorDistributed$Demand[SectorDistributed$
                                                                 Sector == S]
}

# ********************************************************************************************** #
# Projection years. Calculates the number of users in the base year to estimate coverage       #
# rates and densities. The growth forward in the number of users based on SSB's national       #
# population projections.                                                                      #
# ********************************************************************************************** #

# ********************************************************************************************** #
# Creates 6 empty tables that will be filled with the number of users in each sector.          #
# ********************************************************************************************** #

UserGroup[[1]] <- data.frame(ToAge = c(0, 2, 2, 3, 5, 5),
                             Age = 0:5)

UserGroup[[2]] <- data.frame(ToAge = rep(15, 10),
                             Age = 6:15)

UserGroup[[3]] <- data.frame(ToAge = c(rep(15, 16), 16:24, rep(49, 25)),
                             Age = 0:49)

UserGroup[[4]] <- data.frame(ToAge = c(19:29, rep(34, 5), rep(39, 5), rep(44, 5),
                                       rep(49, 5)),
                             Age = 19:49)

UserGroup[[5]] <- data.frame(ToAge = rep(99, 100),
                             Age = 0:99)

UserGroup[[6]] <- data.frame(ToAge = rep(99, 100),
                             Age = 0:99)

# ********************************************************************************************** #
# Sums the number of children in kindergartens in each user group by average stay duration.    #
# This is Equation 12 and Equation 13 in the model.                                            #
# ********************************************************************************************** #

ChildrenGroup1 <- data.frame(Users = DemographyGroup1$Age0,
                             Hours = DemographyGroup1$HoursMin +
                                     ((DemographyGroup1$HoursMax -
                                       DemographyGroup1$HoursMin) / 2))

ChildrenGroup2 <- data.frame(Users = DemographyGroup1$Age1 + DemographyGroup1$Age2,
                             Hours = DemographyGroup1$HoursMin +
                                     ((DemographyGroup1$HoursMax -
                                       DemographyGroup1$HoursMin) / 2))

ChildrenGroup3 <- data.frame(Users = DemographyGroup1$Age3,
                             Hours = DemographyGroup1$HoursMin +
                                     ((DemographyGroup1$HoursMax -
                                       DemographyGroup1$HoursMin) / 2))

ChildrenGroup4 <- data.frame(Users = DemographyGroup1$Age4 + DemographyGroup1$Age5,
                             Hours = DemographyGroup1$HoursMin +
                                     ((DemographyGroup1$HoursMax -
                                       DemographyGroup1$HoursMin) / 2))

# ********************************************************************************************** #
# Creates an empty table to be filled with the number of users in the kindergarten sector.     #
# ********************************************************************************************** #

DemographyGroup[[1]] <- data.frame(FromAge = integer(), ToAge = integer(),
                                   Users = integer(), UserIndex = numeric())

# ********************************************************************************************** #
# Calculates kindergarten users in each of the 4 user groups.                                  #
# This is Equation 14 and Equation 15 in the model.                                            #
# ********************************************************************************************** #

DemographyGroup[[1]] <- rbind(DemographyGroup[[1]],
                             data.frame(FromAge = 0, ToAge = 0,
```

```
                                    Users = sum(ChildrenGroup1$Users),
                                    UserIndex = (2 * sum(ChildrenGroup1$Users *
                                                         ChildrenGroup1$Hours)) /
                                                (sum(ChildrenGroup1$Users) * 42.5)))

DemographyGroup[[1]] <- rbind(DemographyGroup[[1]],
                              data.frame(FromAge = 1, ToAge = 2,
                                    Users = sum(ChildrenGroup2$Users),
                                    UserIndex = (2 * sum(ChildrenGroup2$Users *
                                                         ChildrenGroup2$Hours)) /
                                                (sum(ChildrenGroup2$Users) * 42.5)))

DemographyGroup[[1]] <- rbind(DemographyGroup[[1]],
                              data.frame(FromAge = 3, ToAge = 3,
                                    Users = sum(ChildrenGroup3$Users),
                                    UserIndex = (1.5 * sum(ChildrenGroup3$Users *
                                                           ChildrenGroup3$Hours)) /
                                                (sum(ChildrenGroup3$Users) * 42.5)))

DemographyGroup[[1]] <- rbind(DemographyGroup[[1]],
                              data.frame(FromAge = 4, ToAge = 5,
                                    Users = sum(ChildrenGroup4$Users),
                                    UserIndex = (1 * sum(ChildrenGroup4$Users *
                                                         ChildrenGroup4$Hours)) /
                                                (sum(ChildrenGroup4$Users) * 42.5)))

# ************************************************************************************************ #
# Updating the figures for the number of kindergarten users in each of the 4 user groups when  #
# taking the user indices into account.                                                        #
# This is Equation 16 in the model.                                                            #
# ************************************************************************************************ #

DemographyGroup[[1]][1, ] <- c(0, 0, DemographyGroup[[1]]$Users[1] *
                                     DemographyGroup[[1]]$UserIndex[1],
                                DemographyGroup[[1]]$UserIndex[1])
DemographyGroup[[1]][2, ] <- c(1, 2, DemographyGroup[[1]]$Users[2] *
                                     DemographyGroup[[1]]$UserIndex[2],
                                DemographyGroup[[1]]$UserIndex[2])
DemographyGroup[[1]][3, ] <- c(3, 3, DemographyGroup[[1]]$Users[3] *
                                     DemographyGroup[[1]]$UserIndex[3],
                                DemographyGroup[[1]]$UserIndex[3])
DemographyGroup[[1]][4, ] <- c(4, 5, DemographyGroup[[1]]$Users[4] *
                                     DemographyGroup[[1]]$UserIndex[4],
                                DemographyGroup[[1]]$UserIndex[4])

# ************************************************************************************************ #
# Ensures the Age column is numeric.                                                           #
# ************************************************************************************************ #

People$Age <- as.numeric(as.character(People$Age))

# ************************************************************************************************ #
# Calculates students in primary school.                                                       #
# This is Equation 17 in the model.                                                            #
# ************************************************************************************************ #

DemographyGroup[[2]] <- data.frame(FromAge = 6,
                                   ToAge = 15,
                                   Users = sum(People[People$Age >= 6 & People$Age <= 15,
                                                      paste0("X", as.character(BaseYear))]),
                                   UserIndex = 1.0)

# ************************************************************************************************ #
# Copies the users in Sectors 3 and 4 that were read earlier.                                  #
# ************************************************************************************************ #

DemographyGroup[[3]] <- DemographyGroup3
DemographyGroup[[4]] <- DemographyGroup4

# ************************************************************************************************ #
# Calculates other users in the sector (adult education, vocational schools, etc.).            #
# This is Equation 18 in the model.                                                            #
# ************************************************************************************************ #

DemographyGroup[[5]] <- data.frame(FromAge = 0,
                                   ToAge = 99,
                                   Users = sum(People[, paste0("X", as.character(BaseYear))]),
                                   UserIndex = 1.0)
```

```
# ************************************************************************************************ #
# Calculates users outside the sector.                                                             #
# This is Equation 19 in the model.                                                                #
# ************************************************************************************************ #

DemographyGroup[[6]] <- data.frame(FromAge = 0,
                                   ToAge = 99,
                                   Users = sum(People[, paste0("X", as.character(BaseYear))]),
                                   UserIndex = 1.0)

# ************************************************************************************************ #
# Calculates the demographic development in each employment sector.                                #
# ************************************************************************************************ #

for (S in 1:6) {

    # ******************************************************************************************** #
    # Finds the population from the population projections for the user groups.                     #
    # This is Equation 20 in the model.                                                            #
    # ******************************************************************************************** #

    PeopleSector[[S]] <- merge(UserGroup[[S]], People, by = "Age") %>% group_by(ToAge) %>%
                             summarize(across(c(paste0("X", as.character(BaseYear))
                                                :paste0("X", as.character(EndYear))),
                                              \(x) sum(x, na.rm = TRUE)))

    # ******************************************************************************************** #
    # Sets a row label for the maximum age of the user group.                                      #
    # ******************************************************************************************** #

    rownames(DemographyGroup[[S]]) <- DemographyGroup[[S]]$ToAge

    # ******************************************************************************************** #
    # Indicates that the number of recorded users should be the users in the base year.            #
    # ******************************************************************************************** #

    DemographyGroup[[S]][paste0("Users", paste0("X", as.character(BaseYear)))] <-
    DemographyGroup[[S]]$Users

    # ******************************************************************************************** #
    # Calculates the number of users in each projection year.                                      #
    # This is Equation 21 in the model.                                                            #
    # ******************************************************************************************** #

    for (t in (BaseYear + 1):EndYear) {
        DemographyGroup[[S]][[paste0("Users", paste0("X", as.character(t)))]] <-
        DemographyGroup[[S]][[paste0("Users", paste0("X", as.character(t-1)))]] *
        (PeopleSector[[S]][[paste0("X", as.character(t))]] /
         PeopleSector[[S]][[paste0("X", as.character(t-1))]])
    }

    # ******************************************************************************************** #
    # Creates an empty table for the summation of users in each projection year.                   #
    # ******************************************************************************************** #

    SumDemographyGroup[[S]] <- data.frame(t(BaseYear:EndYear))

    # ******************************************************************************************** #
    # Calculates the sum of the users in each projection year.                                     #
    # This is Equation 22 in the model.                                                            #
    # ******************************************************************************************** #

    for (t in BaseYear:EndYear) {
        SumDemographyGroup[[S]][[paste0("SumUsers",
                                        paste0("X", as.character(t)))]] <-
        sum(DemographyGroup[[S]][[paste0("Users",
                                         paste0("X", as.character(t)))]], na.rm = TRUE)
    }

    # ******************************************************************************************** #
    # Creates an empty table to contain the demographic development in the sector.                 #
    # ******************************************************************************************** #

    KN <- paste0("DemographyComponent", S)
    DemographySector[[S]] <- data.frame(Year = paste0("X", as.character(BaseYear)),
                                        TempColumn = 1)
    names(DemographySector[[S]])[names(DemographySector[[S]]) == "TempColumn"] <- KN
```

```
    # ********************************************************************************************* #
    # Calculates the demographic development for each projection year for each user group.      #
    # This is Equation 23 in the model.                                                         #
    # ********************************************************************************************* #

    for (t in (BaseYear + 1):EndYear) {
        NextCohort <- data.frame(Year = paste0("X", as.character(t)))
        NextCohort[KN] <- SumDemographyGroup[[S]][[paste0("SumUsers",
                                                  paste0("X", as.character(t)))]] /
                          SumDemographyGroup[[S]][[paste0("SumUsers",
                                                  paste0("X", as.character(BaseYear)))]]

        # ********************************************************************************** #
        # The demographic development in the projection year is added as a new cohort in the   #
        # table with the demographic development in the sector.                              #
        # ********************************************************************************** #

        DemographySector[[S]] <- rbind(DemographySector[[S]], NextCohort)
    }
}

# ********************************************************************************************* #
# Copies the tables with the demographic development in each sector together with the          #
# standard change into the same table (alternative path).                                     #
# ********************************************************************************************* #

DemographyIndex <- StandardChange

for (Sector in 1:6) {
    DemographyIndex <- merge(DemographyIndex, DemographySector[[Sector]],
                             by = "Year", all = TRUE)
}

# ********************************************************************************************* #
# Adds the constant indicating the 7 educations in the model to the table.                    #
# ********************************************************************************************* #

DemographyIndex <- DemographyIndex %>%
                   expand_grid(Education = Educations) %>%
                   arrange(Education, Year)

# ********************************************************************************************* #
# Teacher densities based on the base year. Kept constant.                                    #
# ********************************************************************************************* #

# ********************************************************************************************* #
# Copies the table with the demographic development in each sector, the transposed table with  #
# the demand in Equation 11, and any specified teacher shortage into the same table.          #
# ********************************************************************************************* #

Demand <- merge(DemographyIndex, Demand, by = c("Education"), all = TRUE)
Demand <- merge(Demand, TeacherShortage, by = c("Education"), all = TRUE)

# ********************************************************************************************* #
# Calculates the demand.                                                                      #
# This is Equation 24 and Equation 25 in the model.                                           #
# ********************************************************************************************* #

for(S in 1:6) {
    Demand <- Demand %>%
    mutate(!!sym(paste0("Demand")) := !!sym(paste0("Demand")) +
                                      (!!sym(paste0("DemandSector", S)) +
                                       !!sym(paste0("TeacherShortageSector", S))) *
                                      !!sym(paste0("DemographyComponent", S)) *
                                      !!sym(paste0("StandardChange", S)))
}

# ********************************************************************************************* #
# Combines supply and demand.                                                                 #
# This is Equation 26 and Equation 27 in the model.                                           #
# ********************************************************************************************* #

Supply$Year <- paste0("X", as.character(Supply$Year))

FirstAggregate <- aggregate(Demand ~ Education + Year, data = SectorDistributed, FUN = sum)
FirstAggregate$Year <- paste0("X", as.character(BaseYear))
names(FirstAggregate)[names(FirstAggregate) == "Demand"] <- "Supply"
```

```
SecondAggregate <- aggregate(Supply ~ Education + Year,
                             data = Supply,
                             FUN = sum,
                             subset = Year > paste0("X", as.character(BaseYear)))

SupplyDemand <- merge(rbind(FirstAggregate, SecondAggregate),
                      Demand,
                      by = c("Education", "Year"),
                      all = TRUE)

SupplyDemand$Year <- sub("X", "", SupplyDemand$Year, fixed = TRUE)

# ********************************************************************************************* #
# Calculates the difference.                                                                    #
# This is Equation 28 in the model.                                                             #
# ********************************************************************************************* #

SupplyDemand$Difference <- with(SupplyDemand, Supply - Demand)

# ********************************************************************************************* #
# Prints the results and a friendly farewell message.                                           #
# ********************************************************************************************* #

SupplyDemand <- data.frame(lapply(SupplyDemand[c("Education",
                                                 "Year",
                                                 "Supply",
                                                 "Demand",
                                                 "Difference")],
                                  function(x) {if(is.numeric(x)) {as.integer(round(x))}
                                               else {x}}))

Order <- c(ba = 1, gr = 2, lu = 3, ph = 4, pe = 5, yr = 6, py = 7)

SupplyDemand$EducationOrdered <- SupplyDemand$Education
SupplyDemand$EducationOrdered <- with(SupplyDemand, names(Order)
                                      [match(Education, names(Order))])
SupplyDemand$EducationOrdered <- factor(SupplyDemand$EducationOrdered,
                                        levels = names(Order))

SupplyDemand <- SupplyDemand %>% arrange(EducationOrdered, Year)

SupplyDemand$EducationOrdered <- NULL
SupplyDemand$Education <- factor(SupplyDemand$Education,
                                 levels = c("ba", "gr", "lu", "ph", "pe", "yr", "py"),
                                 labels = c("Kindergarten teachers",
                                            "Primary- and middle school teachers”,
                                            "Lecturers",
                                            "PPE (Practical and pedagogical education",
                                            "Teacher education in practical and aesthetic
                                             subjects",
                                            "Vocational Teachers",
                                            "PPE Vocational"))

write_csv(SupplyDemand, 'results/Laerermod.csv')
write_xlsx(SupplyDemand, 'results/Laerermod.xlsx')

cat(sprintf("Education                                          Year Supply Demand
Difference\n"))
invisible(apply(SupplyDemand, 1, function(x) {
    cat(sprintf("%-47s     %4s %5d  %5s     %6s",
                x[["Education"]],
                x[["Year"]],
                round(as.numeric(x[["Supply"]])),
                round(as.numeric(x[["Demand"]])),
                round(as.numeric(x[["Difference"]]))),
        "\n")
}))

cat("\nLærermod has now completed, welcome back.\n")
```