Analysis and Graphical Representation of Health and Labour Force Participation among the Elderly in Europe

Humboldt-Universität zu Berlin
School of Business and Economics
Ladislaus von Bortkiewicz Chair of Statistics

Statistical Programming Languages
Winter 2017/18

Seminar Paper by

Claudia Günther, Phi Nguyen, Julian Winkel
576419, 526624, 562959

Berlin, 2018-03-15

List of Tables

1 Wald test on no joint impact of health on participation 11
2 Women's current and counterfactual labor share 16
3 Men's current and counterfactual (CF) labor share for different age groups 17

List of Figures

Abbreviations

XX	XXXXXXXXXXX

Contents

1 Introduction **2**

2 Data Cleaning and Manipulation **3**

2.1 Theory and Design 3

2.2 Implementation 3

2.3 Empirical Results 6

3 Multidisciplinary and crossnational summary statistics **7**

3.1 Theory and Design 7

3.2 Implementation 7

3.3 Empirical Results 7

4 Crosssectional probit regression **8**

4.1 Theory and Design 8

4.2 Implementation 8

4.3 Empirical Results 8

5 Wald Test **9**

5.1 Theory and Design 9

5.2 Implementation 9

5.3 Empirical Results 11

6 Counterfactual exercise **14**

6.1 Theory and Design 14

6.2 Implementation 14

6.3 Empirical Results 16

7 Graphical Representation **18**

7.1 Theory and Design 18

7.2 Implementation 18

7.3 Empirical Results 21

8 Conclusion **24**

9 References **25**

1 Introduction

- relevance of exploring relationship between health and labour force participation due to changing demographic in Europe
- cite relevant study about ageing
- few sentences about relevant papers exploring relationship -> use paper from DIW
- very short literature overview on relationship between health and labour force participation
- share data set as rich data set for this purpose: 2 sentences about it
- introduction of journal article
- our approach: replicate results an enrich analysis
- especially: introduce graphical visualization tools for descriptive statistics -> ease interpretation of variables
- our aim: write code in a way that allows the user to work with easySHARE data set, even when working on different question

2 Data Cleaning and Manipulation

The original SHARE data set used by Kalwij and Vermeulen (2005), released in spring 2005, contains data on the life circumstances of eligible members from approximately 18,000 households. In order to be eligible for participation in SHARE, at least one household member must have been born in or before 1954. The SHARE survey contains a variety of different health, social, and socioeconomic indicators such as labor force participation or household composition. SHARE contains data from eleven countries: Austria, Belgium, Denmark, France, Germany, Greece, Italy, the Netherlands, Spain, Sweden and Switzerland. This cross-national, multidisciplinary, and longitudinal nature of the SHARE data set provide an abundance of unique and useful opportunities for analysis.

2.1 Theory and Design

The easySHARE data set, released in spring 2017, is the data set for which we will be focusing on in this paper. It is a panel data set of 108 variables of more than 108 variables of more than 100,000 individuals covering data from six survey waves carried out between 2004 and 2015. As can be derived from its name, it is designed to be a simplified version of the broader SHARE data set. As a result, the data set includes slightly different observations and does not provide the same level of granularity in variables broader SHARE data set, which we shall show below. Additionally, since we are only concerned with a smaller subset of observations (from the initial wave of data), we will remove observations that are no longer relevant for our study.

The rest of this section describes our process of data transformation, aggregation, and cleaning , not only to match as closely as possible the initial SHARE data set, but also to convert our data sets into a more accessible state for future analyses.

2.2 Implementation

The `read.and.clean()` function is designed to take the raw easySHARE data set file, which is provided in an `.rda` file format, and convert that into a list of data frames. We split the data set into list items because future analyses involve either summary statistics on the entire data set or regressions conducted on a subset of the data set. The function additionally accepts a wave number as an argument. Although we are only concerned with the initial wave for our study, this additional parameter allows us to conduct supplementary panel studies if we choose to do so later. The function only runs specifically with the raw easySHARE data set (`easySHARE_rel6_0_0.rda`), which is the default `dataset` argument. The `dataset` argument may alternatively be specified to point to the directory where that data set is located. Otherwise it is assumed that it is located in the root directory.

We begin by initializing and downloading the packages (if necessary) required for this function. Much of the funciton utilizes the functions in the `tidyverse` family of functions developed by Hadley Wickham, namely `dplyr` and `magrittr`. The `dplyr` package is designed for accessible and uncomplicated data manipulation, and it works specifically with data frames. The package provides a consistent set of verbs that do a small thing very well, such as aggregation, selection, filtering, or creation of new variables. The `magrittr` package expands upon this further with the pipe operator `%>%`, which enables the chaining of smaller functions to create something more complex. Furthermore, the pipe operator dramatically improves readability by reading chains left to right as opposed to inside-out. This is especially useful when working with nested functions, which can be deconstructed into a series of chained commands.

```r
neededPackages = c("dplyr", "magrittr", "infuser", "countrycode")
allPackages    = c(neededPackages %in% installed.packages()[,"Package"])

    if (!all(allPackages)) {
        missingIDX = which(allPackages == FALSE)
        needed     = neededPackages[missingIDX]
        lapply(needed, install.packages)
    }

invisible(lapply(neededPackages, function(x) suppressPackageStartupMessages(
    library(x, character.only = TRUE))))
```

We continue with basic filtering to select only variables that are pertinent to our study. For the purposes of our work, we are only concerned with the initial wave and adults between the ages of 50 and 64. Then we encode missing values. Although the overall response rate in the SHARE are comparably high, the data set still has numerous missing values. The reason for this is due to the fact that the study was carried out on a cross-national scale, with some national survey institutions deciding not to participate in all survey modules. The reason for the missing values are documented well in the "Guide to easySHARE release 6.0.0" and specifically coded. For example, the numbers -13 and -14 refer to "not asked in this wave" and "not asked in this country". Since this coding scheme is not useful for the purpose of our analysis, we recode all of the missing values as "NA".

```r
# Encode missing values according to SHARE data set guidelines
a = c(-2, -3, -4, -7, -9, -12, -13, -14, -15, -16)
b = c("tocheck","implausible", "tocheck", "uncoded", "notApplicable",
      "dontKnow", "notAskedWave", "notAskedCountry", "noInformation",
      "noDropOff")
missing.value.codes = data.frame(a,b)

# Find NA locations and declare them as such
df.decl = apply(dat, 2, function(z) {
    na.loc    = which(z %in% a)
    z[na.loc] = NA
   return(z)
})
```

We finish converting the raw data set into a final data frame by adding readable country names and transforming raw variables into useful numeric or logical variables as described in the paper. We use the `countrycode` package to change the country codes into human-friendly country names. There are minor differences between the data set that Kalwij and Vermeulen (2005) use and the easySHARE data set, which means we must make certain assumptions in our data cleaning process. For example, certain questions in the survey are an aggregated total of a series of questions, such as the types of chronic diseases the subjects suffer from. Whereas the original authors have access to the specific diseases, we are only provided a count of the total number. Therefore we replace certain variables with the closest relative we have available. In other cases, multiple variables are provided in our data set but it is ambiguous which one the original authors use. An interesting example is employment status: our data set contains one variable for type of employment (full-time, part-time, or not working) and one for labor hours (the number of hours worked by the survey respondent). However, some respondents gave answers that ran counter (for example, saying they worked full-time but only working 20 hours a week). The differences in answers can be attributed to varying interpretations of survey responses.

A simplified example of the final script is shown below.

```
df.out       = df %>%
        dplyr::left_join(country_data, by = c("country_mod" = "iso3n")) %>%
        dplyr::filter(iso3c %in% country_list) %>%
        dplyr::mutate(country       = factor(country.name.en),
                      gender        = factor(ifelse(female, "FEMALE", "MALE")),
                      age           = factor(floor(age)),
                      edu_low       = isced1997_r %in% c(0, 1),
                      edu_high      = isced1997_r %in% c(5, 6),
                      h_chronic     = chronic_mod,
                      h_overweight  = bmi2 == 3,
                      labor_ft      = ep013_mod >= 32,
                      labor_hrs     = floor(ep013_mod)) %>%
        dplyr::select(country, gender,
                      starts_with("h_"),
                      starts_with("labor_")) %>%
        na.omit() %>%                                   # remove missing values
        set_rownames(NULL)                              # reset row numbering
```

Our last step involves splitting the data sets, standardizing any numeric variables, converting the data frames into a model matrix, then storing the resultant data frames into a list that is the final object returned by the function.

The data frames are split into separate country and gender splits, since we will create separate regressions for each of these splits. Then we apply a custom-built standardization function, which finds standardizes all numeric variables and ignores all other variables. Finally, we use a custom-built function that takes advantage of the base `model.matrix()` function to convert all split data frames into a model matrix. This useful function converts all logical vectors into dummies to replicate the process used in the paper. A corollary benefit is that the resultant data frames can be directly used in R's in-built regression functions.

```
standardize.df = function(df) {
    idx = sapply(df, is.numeric)
    idx = seq(1:length(idx))[idx]

    df.reg = df %>%
        mutate_at(.vars = vars(idx),
                  .funs = standardize) %>%
```

```r
        mutate(labor_participation = !labor_np) %>%
        dplyr::select(country, gender, age,
                      h_chronic, h_adla, h_obese, h_maxgrip,
                      edu_second, edu_high, children, couple,
                      labor_participation)

    return(df.reg)
}

splits    = split(df.out, f = list(df.out$country, df.out$gender),
                  drop = TRUE)
df.reg    = standardize.df(df.out)
df.splits = lapply(splits, standardize.df)

dummify = function(df) {
    df = df %>%
        dplyr::select(-country, -gender)
    model     = ~ 0 + .
    new.df    = model.matrix(model, df)
    new.df    = data.frame(new.df)
    return(new.df)
}

df.splits = lapply(df.splits, dummify)
```

2.3 Empirical Results

The validity of our `read.and.clean()` can be verified by comparing the overall results when running the `read.and.clean()` function with the ones presented in the original paper. As the function is called, messages are printed to the console to show the status of the data reading process. Messages additionally show the names of the items in the list so that users can access the specific data set of their choosing.

In the original paper, the authors retain a sample of 12,237 observations, which is slightly smaller than our retained observations of 12,689. However, at the individual country level, some peculiarities in number of observations emerge. Namely, we have almost twice as many observations in France, while having relative sparsity in Austria. This again is likely due to differences in data set and in the data gathering process. We will discuss more details about the summary statistics in future sections.

```r
source('Scripts/ReadAndClean.R')
datasets = read.and.clean(dataset = "easySHARE_rel6_0_0.rda", wav = 1)
```

```
## Loading data set...
## Selecting values only from Wave 1 and between ages 50 and 64.
## Removing missing values.
## Rows removed:   276047
## Rows remaining: 12689
##
## Final output is a list containing 3 data.frames:
## `df.out` is a data.frame containing variables for calculating summary statistics.
## `df.reg` is a data.frame containing standardized variables for regression.
## `df.splits` splits the regression data.frame into individual data.frames for each country/
##          gender split. 22 data.frames are contained in this list.
```

Lastly, when an out-of-bounds value for the wave is selected, an error message occurs. Note that only six waves or cross-sections are included in our data set.

```r
source('Scripts/ReadAndClean.R')
datasets = read.and.clean(dataset = "easySHARE_rel6_0_0.rda", wav = 99)
```

```
## Loading data set...

## Error in read.and.clean(dataset = "easySHARE_rel6_0_0.rda", wav = 99): Out of bounds.
##                               Select a value between 1 and 6.
```

3 Multidisciplinary and crossnational summary statistics

JULIAN

3.1 Theory and Design

3.2 Implementation

3.3 Empirical Results

4 Crosssectional probit regression

JULIAN

4.1 Theory and Design

4.2 Implementation

4.3 Empirical Results

5 Wald Test

In the context of probit regression, the Wald test can be used to test multiple hypothesis regarding the model specifications. For example, We can test whether the fit of the model is improved if a subset of regression coefficients are all set equal to zero, an exclusion restriction. Kalwij and Vermeulen (2005) conduct a Wald test to check the null hypothesis that none of the included health variables has an impact on labour participation in order to investigate the joint impact of health on participation.

5.1 Theory and Design

Depending on the estimation method and distributional assumptions, the Wald statistic can be formulated in different ways. The general form of the Wald statistic after MLE for testing hypothesis regarding our $k \times 1$ parameter vector θ is given by

$$W = c(\hat{\theta})'[\nabla_\theta c(\hat{\theta})\hat{V}\nabla_\theta c(\hat{\theta})]^{-1}c(\hat{\theta}) \qquad (***)$$

where $c(\hat{\theta})$ is a $m \times 1$ vector of linear or nonlinear restrictions, $\nabla_\theta c(\hat{\theta})$ is the $m \times k$ Jacobian of $c(\hat{\theta})$ evaluated at $\hat{\theta}$ and $\hat{V}$ is the estimated asymtotic covariance matrix (Wooldridge 2010, 463). Under H0, the Wald test statistic is asymptotically $\chi 2_m$ distributed, with m being the number of specified restrictions. In order to assure the test statistic W has the assumed limiting distribution, we need to impose some restrictions: Under H0, θ must lie within parameter space and R must be of rank m (Wooldridge 2010, 362). We limit our attention to testing a set of general linear restrictions since the Wald test is not invariant to the re-formulation of non-linear hypothesis. We thus formulate our nullhypothesis in accordance with the common linear restriction structure of H0: $R\hat{\beta} = r$ againsts the alternative H1: $R\hat{\beta} \neq r$ to facilitate the derivation of our test statistics where R is a $m \times k$ matrix of rank m (equivalent to the Jacobian), whereas the restriction function r is a $m \times 1$ vector. Given the conditions mentioned above, the Wald statistic can then be rewritten as

$$W = (R\hat{\beta} - r)'[R\hat{V}R']^{-1}(R\hat{\beta} - r) \qquad (***)$$

which facilitates our calculations (Greene 2012, 527–29; Wooldridge 2010, 362). When we are interested in the joint significance of a subset of s coefficients, the Wald test statistic $W \overset{a}{\sim} \chi 2_s$ can be even more simplified (Wooldridge 2010, 362):

$$W = \hat{\beta}_s'[R\hat{V}R']^{-1}\hat{\beta}_s = \hat{\beta}_s'[\hat{V}_s]^{-1}\hat{\beta}_s \qquad (***)$$

5.2 Implementation

In order to test linear hypothesis regarding the model specifications we design two Wald test versions. The function `joint.wald.test` can be used to test the joint significance of a subset of model coefficients, whereas `general.wald.test` allows checking any linear hypothesis. The two tests are designed in a way that they align well with the glm estimation output. The specific design of the joint significance test `joint.wald.test` is makes the function easy to use, understand and modify to special needs. The only required input is the model summary from glm estimation, whereas the significance level and test specifications are optional.

```
joint.wald.test = function(model.summary, confidence.level = 0.95, spec = NULL){

    joint.wald.test        = numeric(6)
    names(joint.wald.test) = c("Name","W","p-value", "df", "H0" , "Decision")
    beta                   = model.summary$coefficients[,1]
    Var_beta_est           = vcov(model.summary)
```

In order to set up default values for the hypothesis specification, we make use of a local if-else statement inside the `joint.wald.test` function. By declaring the model specification to be null in line XXX we set the foundation for the default specification in line XXX - XXX. Here, we re-asign a sequence vector to `spec` that includes the number of all model coefficients in increasing order, unless `spec` is specified differently.

```
   if(is.null(spec)){
    spec = 1:length(beta)
  } else if(is.logical(spec)){ stop("spec cannot be a logical vector, tranform to integer vector!")
  } else if(any(spec == 0)){ stop("spec cannot contain zero values!")
  } else if(!is.integer(spec)){
      spec.len = length(spec)
      spec = as.integer(spec, length = spec.len)
      warning("Converting spec to integer and proceeding")
  }
    joint.wald.test        = numeric(6)
    names(joint.wald.test) = c("Test","W","p-value", "df", "H0" , "Decision")
    beta                   = model.summary$coefficients[,1]
    Var_beta_est           = vcov(model.summary)

    W = t(beta[spec]) %*% solve(Var_beta_est[spec,spec]) %*% beta[spec]
```

This means `joint.wald.test` will conduct a joint significance test on all model coefficients by default. The Wald test statistic in line XXX is calculated via simple matrix algebra based on the provided input by the model summary. This formula is equivalent to equation YYY.

The proper format of the specification vector `spec` is crucial as it is used to extract the needed estimates from the model coeffictient vector and covariance matrix. In line XXX, the term `Var_beta_est[spec,spec]` extracts all covariance matrix elements corresponding to the joint significance hypothesis of the particular specification. To select the right covariance elements and degrees of freedom, `spec` must be a vector of integers of length $0 < m \leq k$. We therefore add several stops and warnings. Specifically, we make sure `spec` is neither logical nor does it contain any zero, since the function would then determine an incorrect number of degrees of freedom. Furthermore, we only allow for integers since we are testing the joint significance of coefficients. In line XXX-XXX we transform non-integer input to a vector integers and inform about this process by giving a warnings messsage.

The general Wald test is designed in a similar manner, except it allows for the general formulation of linear hypothesis of the form $R\hat{\beta} = r$. The test statistic in this case is giving by equation YYY. As in the `joint.wald.test` function, we use local local if-else statements to setup a default settings. If only the model summary is given as an input, the `general.wald.test` conducts a joint significance test at a 95% confidence level. The crucial elements to the proper usage of this function are the correctly specified Jacobian matrix R and restriction vector r, which must be of size $m \times k$ and $m \times 1$ respectively. We additionally need to assure that R is of rank m. To assure the correct input, we incorporate stops and warning messages.

```r
    if(class(model.summary) != "summary.glm") stop("model.summary must be a glm summary!")
    if(confidence.level > 1 | confidence.level < 0) stop("confidence.level out of bounds!")

    dim_R = dim(R)
        m = length(r)

    if(dim_R[1] != m | dim_R[2] != k) stop("R has wrong dimension!")
    if(rankMatrix(R)[1] != m) stop("R has wrong rank!")
```

We first check in line XXX whether the `model.summary` has the right class and whether the significance level lies within the expected interval. Otherwise, the function's executions will be stopped. In the second step, we make sure the matrix `R`and the restriction vector `r` have matching dimension. We first extract the matrix dimensions `dim_R` in line XXX and check them via indexing in line XXX-XXX.

5.3 Empirical Results

We use our designed wald test functions to carry out several hypothesis test on our model coefficients from probit estimation. Speficically, we check the null hypothesis that none of the four included health variables has an impact on labour participation for each country and for both men an women. The results for men can be seen in table XXX.

Table 1: Wald test on no joint impact of health on participation

	W statistic	p-value	Test decision
Austria	13.63	0.00859	Reject H0
Belgium	12.59	0.01346	Reject H0
Denmark	24.32	6.886e-05	Reject H0
France	5.318	0.2562	Cannot reject H0
Germany	39.42	5.701e-08	Reject H0
Greece	0.9592	0.9159	Cannot reject H0
Italy	8.334	0.08009	Cannot reject H0
Netherlands	19.95	0.0005113	Reject H0
Spain	34.96	4.738e-07	Reject H0
Sweden	25.42	4.141e-05	Reject H0
Switzerland	9.906	0.04205	Reject H0

Our results our in line with the findings of Kalwij and Vermeulen (2005). As expected, the null hypothesis of no impact of health on labour participation is rejected at a 95% confidence level for most countries. A particularity is the high p-value of France, leading to the inability to reject H0, which stands in contrast to the findings of Kalwij and Vermeulen (2005). We explain this with our differing sample, which is almost 50%

larger, and the lower overall employment rate of our sample. Regarding the Wald tests for women, our results are very similar to those Kalwij and Vermeulen (2005). As for men, The majority of null hypothesis can be rejected for women, with the null for German women being barely not rejected. The inability to reject the null for some countries is due to relatively low health-related probit coefficients (in absolute terms), which implies that these health indicators are not strongly related with the labor particpation decision. The results of our `joint.wald.test` function are comparable those of our packages, like the `wald.test` from the aod packages. To check our function this we can conduct both test on all subsets. For example, for German women the aod packages' and our `joint.wald.test` return similar output:

```r
wald.test(Sigma = vcov(allModels$Germany.FEMALE),
              b = allModels$Germany.FEMALE$coefficients, Terms = 16:19)

joint.wald.test(allSummaries$Germany.FEMALE, spec = 16:19)
```

```
## Wald test:
## ----------
##
## Chi-squared test:
## X2 = 9.2, df = 4, P(> X2) = 0.057
```

```
##                  W           p-value                 df
##           "9.175"         "0.05688"                "4"
##          Decision
## "Cannot reject H0"
```

A weakness of the aod package is, that its wald test cannot deal with empty classes. For the case of Switzerland, where our male sample does not include any 64-year-olds, the `wald.test` from the aod packages leads to the error message

```r
wald.test(Sigma = vcov(allModels$Switzerland.MALE),
              b = allModels$Switzerland.MALE$coefficients, Terms = 16:19 )
```

```
## Error in L %*% V: nicht passende Argumente
```

Our wald test function is robust against the presence of empty classes since we work with model summaries, meaning that the coefficients of empty classes have been dropped instead of being a missing value. Thus, our `joint.wald.test` function can conduct the test for Swiss men without errors:

```r
joint.wald.test(allSummaries$Switzerland.MALE, spec = 16:19)
```

```
      W     p-value          df    Decision
"9.906"   "0.04205"         "4" "Reject H0"
```

Both our wald test functions are sensitive to the improper formulation of tested model restrictions. For example, the `general.wald.test` function stop from execution if the Jacobian matrix `R` and the restriction vector `r` have non-matching dimensions.

```r
matrix.R     = diag(1, 23)
vector.wrong = rep(1,22)
vector.right = rep(1,23)
general.wald.test(allSummaries$Switzerland.MALE, R = matrix.R, r = vector.wrong)
general.wald.test(allSummaries$Switzerland.MALE, R = matrix.R, r = vector.right)
```

```
FALSE Error in general.wald.test(allSummaries$Switzerland.MALE, R = matrix.R, : R has wrong dimension!
```

```
          W     p-value          df    Decision 
   "1050"         "0"        "22" "Reject H0"
```

In this example, `R`is an identity matrix of size 22×22, whereas `vector.wrong` is 21×1. As soon as we add another element to the `r` vector, the function works properly due to dimension matching.

6 Counterfactual exercise

Counterfactual exercises are a common tool in economic policy analysis that allow assessing and quantifying possible effects of policies. In order to evaluate the quantitative importance of a healthy population on labour participation, Kalwij and Vermeulen (2005) conduct a simple counterfactual exercise. They compare the current national employment rate to the predicted counterfactual rate. This counterfactual employment rate is obtained by asuming that all individuals are in perfect health. According to their definition, a perfectly healthy individual has never had a severe or mild medical condition, suffers from no restrictions in activities of daily living, is not obese and has a grip strength of an average 50-51 year-old (fe)male individual.

6.1 Theory and Design

Counterfactual exercises based on regression methods that can be carried out in numerous ways. The simplest method to is to replace the existing covariates values X with counterfactual values X_{cf} (Chernozhukov, Fernández-Val, and Melly 2013). For the case of probit regression with a binary response variable y for employment, we can interpret the fitted values $\hat{y}$ and $\hat{y}_{cf}$ as current and counterfactual employment probabilities for each individual. Based on these calculated individual employment probabilities, we can calculate the current and counterfactual population employment rates (Π_{ac}, Π_{cf}) by taking the mean, making use of the Law of Large Numbers. Obtaining the counterfactual probabilities $\hat{y}_{cf}$ requires a two-step procedure. First, the regular probit regression of y on X is tun to obtain the estimated coefficients $\hat{\beta}$. Second, the counterfactual probabilities for each individuals Π_i are calculated by predicting y based on X_{cf} and the estimated coefficients.

$$\Pi_{icf} = Pr(Yi = 1|Xi = xi_{cf}) = \Phi(xi'_{cf}\beta) \tag{***}$$

We can assess counterfactual health-related effects by comparing the actual and counterfactual employment rates Π_{ac} and Π_{cf}. We can gain closer insights into the nonlinear relation between health effects and particpation by calculating Π_{ac} and Π_{cf} for different age groups. This also allows assessing the proportion of decline in participation that is due to decline in health condition according to our model estimates.

6.2 Implementation

Our counterfactual effects are based on the manipulation of explanatory variables according to some specified criteria. We design this counterfactual exercise in a way that allows us to directly use the output from our probit estimation. As a first step, we create a simple function `empl.rate` that takes an estimation model object as input and predicts the employment rate, i.e. the mean of fitted values, as an output. In order to calculate the both current and counterfactual employment rates Π_{ac} and Π_{cf} conveniently, we need to manipulate the original covariates contained in our model objects. For this purpose we design the function `X.cf`, whose only input the model object, allowing us to treat each country separately later on. As can be seen in line XXX, the function creates a duplicate version of orginal data `X`, whose values will be replaced later on.

In line XXX, we calculate the minimum value for each variable in `X`, which corresponds to the perfect health conditions of the variables `h_chronic`, `h_adlaTRUE`, and `h_obeseTRUE` since we are dealing with standarized varaibles. We obtain these values with making use of the apply function and store them in the vector `X_min`.

We also calculate the mean average grip strength of 50- to 51-years old in the sample `X`. As can be seen in line XXX, we derive this mean with the `tapply` command because it allows us to specify an age group index and returns a vector whose elements we can access.

```
    X                   = model$data
    X_cf                = X

    X_min               = data.frame(t(apply(X, 2, min)))

    names.vec           = c("h_chronic", "h_adlaTRUE", "h_obeseTRUE")
    X_cf[, names.vec]   = X_min[names.vec]

        age_50_51          = ifelse(X$age50 == 1 | X$age51 == 1, 1,0)

    X_cf[, c("h_maxgrip")] =  tapply(X$h_maxgrip, age_50_51 == 1, mean)[[2]]

              model$data   = X_cf
```

To obtain the counterfactual covariates, the values in `X_cf` are replaced with perfect health values. As a final step, we replace the original model data in `X_cf`. This allows us to access both the model estimates and counterfactual covariates in one object. By running our `empl.rate` function on the returned model, we directly optain the counterfactual employment rate for a given country.

Since we are also interested in the current and counterfactual employment rates of different age groups, we create a more advanced function `empl.rate.age`. This function displayed in line XXX requires the estimation model as mandatory argument and has two optional arguments (`group.size` and `group.low`) that allow to change the age group and group sizes. Given the specified arguments, the function then calculates the current and counterfactual employment rate for each age group.

```
empl.rate.age        = function(model, group.size = 5, group.low = c(50,55,60)){

    X                = model$data

    empl.probability = predict(object = model, newdata =  X, type = "response")
```

The proper value assignment of the numerical vector `group.low` is essential since the dynamic creation of age groups is based on it. As can be seen in line XXX, we use a small for loop to determine the upper age group bound and create a label of each age group. Furthermore, we assign all of the corresponding age values (e.g. `age50`, `age51`) to the vector `z`, which correspond to the age variable names in `X`.

```
        for (i in group.low){
                                k = i + group.size -1
                        vec.label = paste("names.vec.age", i, sep=".")
                                z = assign(vec.label, paste0("age",i:k))
```

We need to assign the age values to `z` in order to locate the right individuals in our data frame `X`. We find the individuals belonging to a given age group by summing over rows of `X` and storing this vector inside the list

`IND_row_vec` displayed in line XXX. We can take the sum because the age values assigned to `z` correspond to the column names of the data frame `X`. Thus, for a given individual, the row sum of `X[, z]` is either zero or one, dependong on wether the individual belongs to the age group or not. Following this step, we filter out only the relevant age groups via indexing.

```
              IND_row_vec[[i]] = apply(X[, z], 1, sum)
            IND_row_vec      = IND_row_vec[group.low]

empl.rate[k] = empl.probability %*% IND_row_vec[[k]] / sum(IND_row_vec[[k]])
```

Finally, the employment rate for each age group is calculate in line XXX. For this, we first sum the predicted employment probability for all individuals in age group k via vector multiplication. Next, we devide this sum by the overall number of individuals in that age group, corresponding to the vector length `IND_row_vec[[k]]`.

Running the `empl.rate.age` function on a given model object will return a numerical vector containing the employment rate of the specified age groups.

6.3 Empirical Results

Running our created counterfactual functions on all the created subsamples allows us to assess the labor participation potential of a healthy population for different countries and both genders. Table XXX display these results on an aggregate level. On average, the predicted participation rate is 4.5 percentage points given the perfectly healthy individuals (certeris paribus), although there are quite some outliers. For Dutch women, the estimated counterfactual participation rate is 8 percentage point higher, corresponding to a relative increase of 18%. This is due to the high coefficients of chronic disease and obeseness in absolute terms, combined with above average prevalence of chronic disease and obeseness among Dutch women. For men, the results are even more pronounced. In Spain, for example, the counterfactual participation rate is almost 12 percentage points higher, driven by high coefficients of chronic disease, medical conditions and obeseness in absolute terms. As expected, our results are in line with Kalwij and Vermeulen (2005) general, though the estimated participations rates differ for some countries, as a result of differing samples.

Table 2: Women's current and counterfactual labor share

	Employment Current	Employment Counterfactual.
Austria	0.33	0.37
Belgium	0.38	0.43
Denmark	0.64	0.69
France	0.54	0.57
Germany	0.56	0.60
Greece	0.30	0.31
Italy	0.25	0.27
Netherlands	0.45	0.52
Spain	0.36	0.38
Sweden	0.76	0.83
Switzerland	0.63	0.63

It must be stressed, however, that the counterfactual estimates presented by Kalwij and Vermeulen (2005) should be interpreted with caution. First of all, the transformation of the status quo populations with perfectly health individuals is articial and represents a state that could not realistically be reached with any policy. Second, some of the high counterfactual rates are driven by relatively high model coefficients that are not statistically significant, indicating non-robust results. Third, the aggregate country estimates do not properly display the variation of health effects among different age groups. In order to address this, we also analyze the results of the counterfactual exercise with help of our `empl.rate.age` function. The output of this for men is displayed in table XXX. As expected, both the current and counterfactual labor shares decrease at higher age. The difference between current and counterfactual shares is also increasing in age, though some exceptions (Austria, Italy and Greece) exist. This increasing difference points towards the negative health effects on labor associated with increasing age. Again, our results are in line with those obtained by Kalwij and Vermeulen (2005).

Table 3: Men's current and counterfactual (CF) labor share for different age groups

	Current 50-54	CF 50-54	Current 55-59	CF 55-59	Current 60-64	CF 60-64
Austria	0.82	0.85	0.57	0.64	0.12	0.16
Belgium	0.79	0.81	0.54	0.58	0.22	0.26
Denmark	0.83	0.87	0.75	0.83	0.56	0.65
France	0.86	0.87	0.57	0.62	0.17	0.20
Germany	0.81	0.86	0.75	0.85	0.37	0.51
Greece	0.84	0.84	0.70	0.70	0.45	0.46
Italy	0.85	0.87	0.52	0.56	0.28	0.30
Netherlands	0.87	0.90	0.76	0.81	0.30	0.36
Spain	0.81	0.90	0.74	0.85	0.40	0.55
Sweden	0.93	0.95	0.82	0.86	0.70	0.77
Switzerland	0.92	0.94	0.89	0.93	0.63	0.72

7 Graphical Representation

In this section, we want to introduce novel graphical elements not present in the paper to augment the existing summary statistic tables. We believe that the original paper's lack of any graphical details, either for summarizing valuable data points or for exploratory analysis, hampers a reader's ease of understanding. Tables, although useful for organizing and displaying many discrete data points, lack both detail and decipherability. Furthermore, tables offer no visibility into distributions inherent to the data. We believe that a necessary and important enhancement to the paper would be to include graphs that allow readers to view distributions, analyze trends, and compare data points across groups or geographies.

7.1 Theory and Design

An increasingly commonly used method for representing country-level data graphically is with choropleth maps. Choropleth maps shade or pattern areas within a defined geographic region (such as a country, state, or city) in relation to a data variable. A coloring progression or gradient is used to show value levels within each region. They are useful for identifying values that are associated with a given geographic location and quickly comparing those values across different regions. There are a multitude of great packages in R, along with open source shape files that contain the geographic coordinates to draw these maps, to aid in building choropleth maps.

However, we opt not to use choropleths, instead focusing on a novel approach called tile grid maps. Rather than using actual geographic borders, countries are reduced to a uniform shape and size (in our case, a simple square) in order to ensure equal visual weight. This is done to avoid visual imbalances inherent to choropleth maps: larger geographic regions appear to have more value, influencing a reader's perception on the relative importance of such a region. Additionally, sparsely populated areas may also have a larger visual emphasis. We use tile grid maps since relative population sizes are of little importance to us, but we want to focus rather on comparing country data on variables that are either percentages or have been normalized or fit on the same scale.

Tile grid maps are not without fault, however. Europe does not contain smooth borders like those found in the United States, hence it can be difficult to draw a comparable representation of Europe with grids alone. Nevertheless, the general shape of Europe can be quickly perceived by the reader, which will be shown in the empirical results section.

Additionally, we choose to look at distributions of certain variables, rather than just looking at the mean or median values. By looking at the overall spread of each numeric data variable, we are able to gather a more holistic picture of the nature of the data variable.

7.2 Implementation

We develop two new graphics-oriented functions to generate both graphs listed above. The first is the tile grid maps, defined in the function `health.gridmap()`. Instead of just generating a single tile grid map, we create facets of tile grid maps, where the facetting is determined by the user. Either they can choose to split by gender (in which two grid maps will be displayed side-by-side) or by age group (in which three graphs will be shown). This facetting method allows readers to distinguish differences between countries as well as within countries. For example, in the same view, we can see how the hours worked per week differ between Germany and Spain, while also seeing how labor hours generally decrease for all countries as groups age over time.

In order to draw the grid map, we must first draw the coordinates. This is done in a simple Excel spreadsheet to align each country with a specific point on a two-dimensional grid. Rather than sourcing this Excel file each time, we load the file once, then use the `dput` function to replicate the structure of the table (this is not shown in the script). This structure is then defined explicitly within the function.

We define a "base" plotting function that draws the initial outline of the tile grid, using the `ggplot2` package. Additional plotting parameters are added later based on the function parameters passed. This is an advantage of using `ggplot2`: we can define the initial plot, then `layer` on additional graphical elements as needed, such as dynamic titles or flexible scales.

```r
plot_function = function(df) {

    p = ggplot(data = df, aes(x = X, y = Y)) +
        geom_tile(aes(fill = get(xvar), color = ""),
                  color = "white", size = 0.6) +
        geom_text(aes(label = iso3c), color = "white") +
        geom_text(aes(label = round(get(xvar), 2)), vjust = 2,
                  color = "white", size = 3, na.rm = TRUE) +
        coord_fixed(ratio = 1) +
        theme_minimal() +
        theme(axis.line        = element_blank(),
                axis.text        = element_blank(),
                axis.title       = element_blank(),
                panel.background = element_blank(),
                panel.grid       = element_blank(),
                legend.position  = "bottom") +
        theme(plot.title    = element_text(size=16),
                plot.subtitle = element_text(size=10,
                                             color = "#7F7F7F"))
    return(p)
}
```

We opt for color-blind safe colors, using a sequential gradient color scheme, to enable maximal readability. The range of these color scales is always set to range from the minimum to maximum values between all facets of data. This is done to ensure that there are no values that fall out of bounds, as well as to apply the same color scale for all graphs in view.

```r
minval = min(df.f[[xvar]], df.m[[xvar]], na.rm = TRUE)
maxval = max(df.f[[xvar]], df.m[[xvar]], na.rm = TRUE)
```

If a logical variable is selected, we display the percentage of survey respondents for which that value is true. If a numeric variable is selected, we display the average value within survey respondents. The final output is stored as a `grob`, an element from the `gridExtra` package, which allows us to organize mutliple graphs side-by-side in a single window. We can either draw the final output using the `grid.draw()` function or we can save to disk using the `ggsave()` function.

```r
example5 = health.gridmap('labor_hrs', 'age')
grid.draw(example5)
ggsave("Output/gridmap_laborhrs_byage.png", plot=example5, width=12,
       height=8, units="in")
```

The second function creates a series of individual bar charts, split by a collection of selected genders and countries, for a selected numeric variable. A vector of countries and/or genders can be passed as arguments to the function, in the event a user only cares about a subset of the overall data set. In the event no vector of countries and/or genders are supplied, the entire data set is used.

In some cases, outliers in the data set may skew the view of the graphic. Therefore there is an additional default parameter to remove outliers.

```r
# Remove outliers (outside 1.5 IQR of 25% and 75% percentiles)
rm.outliers = function(x, na.rm = TRUE) {
    qnt = quantile(df.out[[x]], probs = c(0.25, 0.75), na.rm = na.rm)
    out = 1.5*IQR(df.out[[x]], na.rm = na.rm)

    new.out = df.out
    new.out[(df.out[[x]] < qnt[1] - out), ] = NA
    new.out[(df.out[[x]] > qnt[2] + out), ] = NA

    new.out = new.out[complete.cases(new.out), ]
    return(new.out)
}

if (remove.outliers) df.out = rm.outliers(x = xvar)
```

In addition, bars indicating the average over the entire data set are overlaid on top of each bar chart. This is to enable readers to see how similar or different one distribution is compared to the average. We do this by calculating the distribution of the entire data set, then creating a dummy data frame with all possible combinations of gender and country. We then use this dummy data frame as an additional geometric argument in `ggplot`.

```r
total.dist = table(df.out[[xvar]])/nrow(df.out)

total        = expand.grid(countries, gen, sort(unique(df.out[[xvar]])))
names(total) = c("country", "gender", xvar)
total        = arrange(total, country, gender, get(xvar))
total$value  = rep(total.dist, times = length(countries)*length(gen))
```

Finally, to prevent overcrowding in the view, we decide on the optimal way to arrange the graphs. If only a single gender is selected, we organize the graphs so that only four columns are shown. If all genders are selected, the graphs are organized so that gender is along the y-axis and countries are along the x-axis.

```r
if (length(gen) == 1) {
    plot.bar = plot.bar +
        labs(subtitle = paste0("Faceted by country. Gender selected is ", gen,
        ". Distribution of the entire data set shown is overlaid on each graph.")) +
        facet_wrap(~ country, ncol = 4)
} else {
    plot.bar = plot.bar +
        facet_grid(gender ~ country)
}
```

7.3 Empirical Results

We can test the validity of each of the graphics by generating a few sample outputs. For the tile grid maps, we include some stopping conditions to ensure correct facetting and/or to ensure that only a numeric or logical variable is supplied. A similar sequence of error messages is shown for the health distribution maps when the variable is not numeric, the countries supplied are not contained within the data set, or the correct genders are not supplied.

```r
source("Scripts/ReadAndClean.R")
source("Scripts/Graphics.R")
datasets = read.and.clean(dataset = "easySHARE_rel6_0_0.rda")

#Only keep relevant data sets
df.out = datasets$df.out

health.gridmap(xvar = 'country', facetting = 'gender')
```

```
## Error in health.gridmap(xvar = "country", facetting = "gender"): 'xvar' must be numeric
##          or logical
```

```r
health.gridmap(xvar = 'h_obese', facetting = 'country')
```

```
## Error in health.gridmap(xvar = "h_obese", facetting = "country"): 'facetting' variable
##          must be either 'gender' or 'age'
```

```r
health.distribution(xvar = 'age', gen = 'MALEEEEEEE')
```

```
## Error in health.distribution(xvar = "age", gen = "MALEEEEEEE"): 'gender' must be one of
##               'all', 'MALE', or 'FEMALE'
```

```r
health.distribution(xvar = 'labor_pt', gen = 'FEMALE', countries = 'all')
```

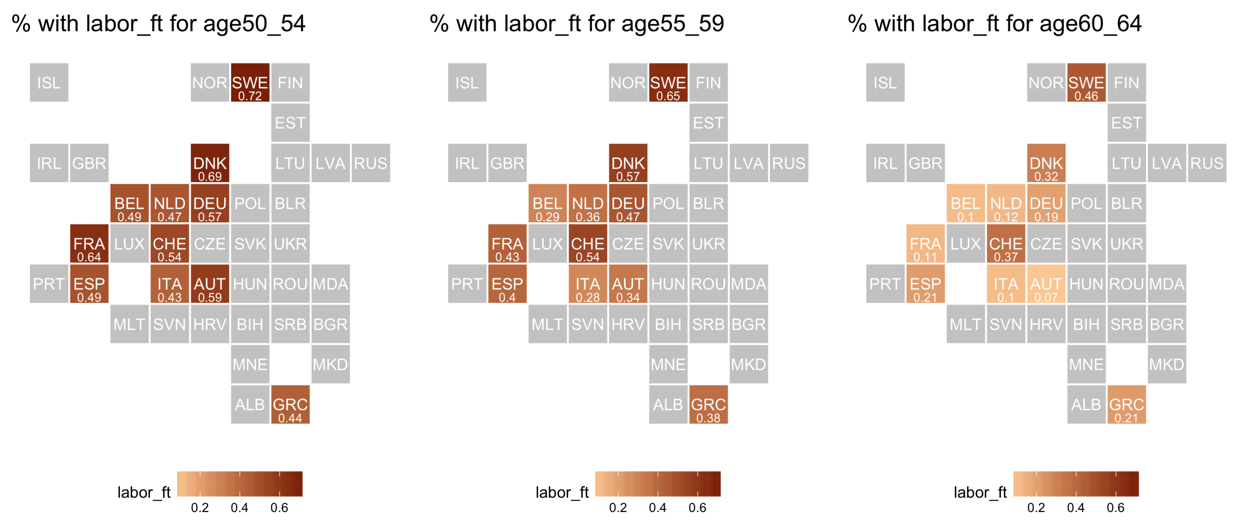
```
## Error in health.distribution(xvar = "labor_pt", gen = "FEMALE", countries = "all"): 'xvar'
##                                             must be numeric
```

```r
health.distribution(xvar = 'age', gen = 'FEMALE', countries = 'China')
```

```
## Error in health.distribution(xvar = "age", gen = "FEMALE", countries = "China"): 'countries'
##                      must be 'all' or a character vector containing countries
##                      in the `df.out` data frame
```

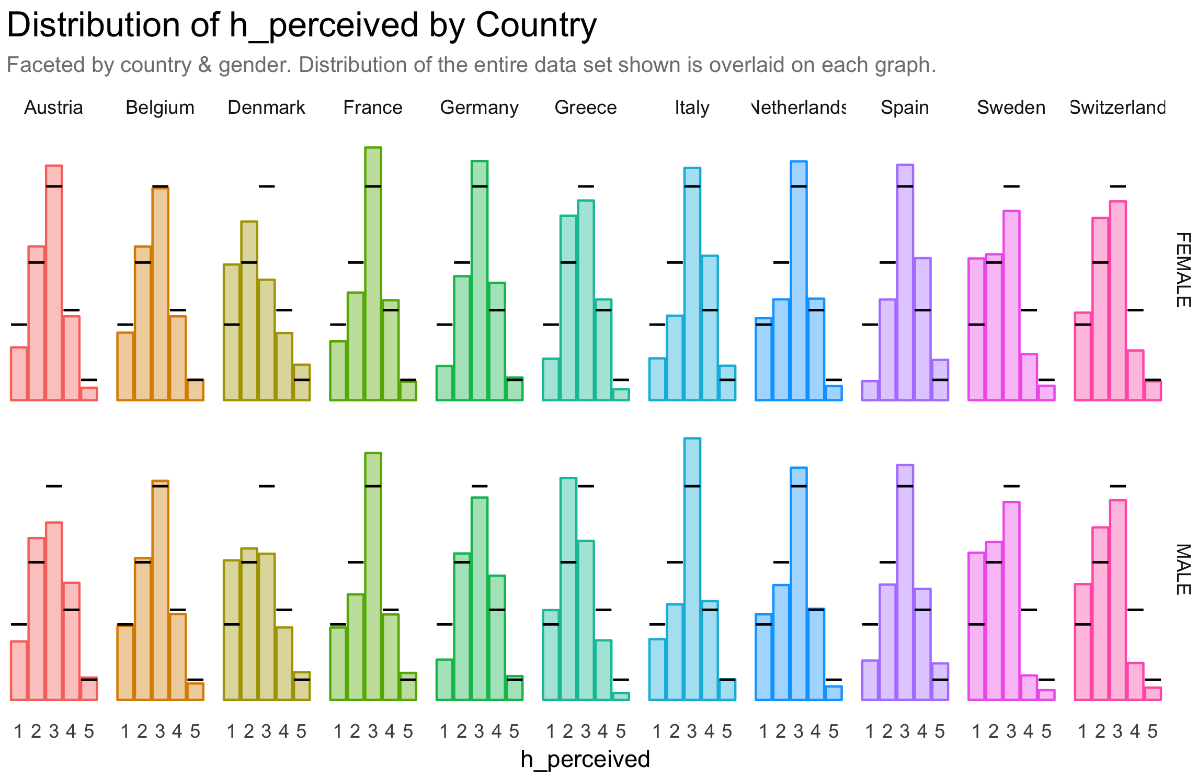
We show an examples of correct output below. Here, we want to examine average labor hours per week, facetted by age group. As we can see in this example, the percentage of the population that works full time generally decreases for all countries as age increases. However, in countries like Sweden and Switzerland, a higher percentage continue to work even in old age.

```r
health.gridmap(xvar = 'h_obese', facetting = 'country')
```



Finally, we display an example of the health distribution map. Here, we seek to evaluate the spread of self-perceived health among survey respondents after removing outliers. The black line indicates the overall distribution for the entire data set. As we can see, there are not too many differences between male and female distributions within countries. However, countries like Spain tend to believe they are healthier than average, whereas the opposite effect is true in Switzerland.

```r
health.distribution(xvar = 'h_perceived', remove.outliers = FALSE)
```


Distribution of h_perceived by Country
Faceted by country & gender. Distribution of the entire data set shown is overlaid on each graph.
Austria
Belgium
Denmark
France
Germany
Greece
Italy
Netherlands
Spain
Sweden
Switzerland
FEMALE
MALE
1 2 3 4 5
h_perceived

8 Conclusion

Quantlet 4

9 References

Chernozhukov, Victor, Iván Fernández-Val, and Blaise Melly. 2013. "Inference on Counterfactual Distributions." *Econometrica* 81 (6). Wiley Online Library: 2205–68.

Greene, William H. 2012. *Econometric Analysis.* Seventh Edition. Pearson Education.

Kalwij, Adriaan, and Frederic Vermeulen. 2005. "Labour Force Participation of the Elderly in Europe: The Importance of Being Healthy."

Wooldridge, Jeffrey M. 2010. *Econometric Analysis of Cross Section and Panel Data.* MIT press.

Declaration of Authorship

We hereby confirm that we have authored this Seminar paper independently and without use of others than the indicated sources. All passages which are literally or in general matter taken out of publications or other sources are marked as such.

Berlin, 2018-03-15

Claudia Günther, Phi Nguyen, Julian Winkel