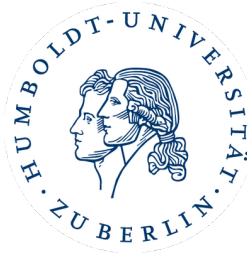# Analysis and Graphical Representation of Health and Labour Force Participation among the Elderly in Europe

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

List of Figures

Abbreviations

XX   XXXXXXXXXX

Contents

1 Introduction

The proportion of elderly people in Europe continues to rise and is expected to impose major social, fiscal and economic challenges in the future. According to official projections, half of the European population is expected to be 50 years or older by 2050 (Commission and others 2012). Analyzing the relationship between aging populations, health indicators and labor force participation is essential to come up with suitable policy responses to those demographic changes. Empirical results on the relation between health, wellbeing and employment are ambiguous. While some studies show that poor health contributes to early retirement (Cai and Kalb 2006; Karpansalo et al. 2004; Monden 2005), another branch of research suggests that unemployment has negative effects on wellbeing and health (Shields and Price 2005; Kassenboehmer and Haisken-DeNew 2008).

The Survey of Health, Ageing and Retirement in Europe (SHARE) launched in 2004 is an excellent research base for aging related research designs. This multidisciplinary and cross-national panel database of data on health, socio-economic status and social networks contains data of more than 120,000 individuals aged 50 or older (Borsch-Supan 2017). Based on the first SHARE data release, Kalwij and Vermeulen (2005) analyze the impact of different health indicator on individuals' participation decisions in 11 European countries. They find that health effects on labor force participation behavior of individuals aged 50-64 differ considerable between countries, age groups and gender.

In this paper, we present the developed code that can be utilised to replicate all the results from Kalwij and Vermeulen (2005) and furthermore provides a basis for further data exploration and visualization. For this purpose, we use the recently released easySHARE data set. We design the code in a way that allows others to explore the easySHARE data set, generate comprehensible and instructive summary statistic tables or graphics and conduct regression-based analysis.

Our code is devided into six quantlets, wich are meant to be executed chronologically. In `Quantlet 1`Q, we transform, clean and aggregate the variables in the raw data set and generate a list of data frames for subsequent data analysis. `Quantlet 2`Q generates a series of summary statistics tables that present health indicators, labor force particpation and labor supply choice by country, age group and gender. In `Quantlet 3`Q, we conduct probit regression for each country and both genders, calculate marginal effects and average employment probability. `Quantlet 4`Q lays the foundation for two Wald tests that can be used to test the joint significant of a subset of variables as well as any linear hypothesis. `Quantlet 5`Q presents counterfactual exercises that are based on the priorly carreid out probit estimations. By manipulating the population health indicators, we calculate current and counterfactual labor participation rates for different age groups. In `Quantlet 6`Q, we introduce new methods to graphically represent the summary statistics fo exploratory data analsis. Specifically, we provide tools to generate carefully selected graphics for a subset of desired variables and sample subsets.

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 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.

```
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))))
```

```
## Warning: package 'dplyr' was built under R version 3.3.3
```

```
## Warning: package 'infuser' was built under R version 3.3.3
```

```
## Warning: package 'countrycode' was built under R version 3.3.3
```

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".

```
# 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) %>%
        mutate(labor_participation = !labor_np) %>%
```

```r
        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.

## Warning: package 'bindrcpp' was built under R version 3.3.3
```

```
## 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

3.1 Theory and Design

Since we are working with a cross-national survey, it is useful to to present an overview on a country-by-country basis. For each country, survey participants will be classified according to their age group. Within each class, shares of gender-specific as well as overall labor participation rates and health indicators are calculated and discussed.

3.2 Implementation

An efficient way of seperating the easySHARE dataset into country, gender and age is explained in the following. In order to minimize hard-coding and possibly related errors, a grid is expanded between named variables. A function that calculates labor participation rates can subsequently be applied onto the defined grid.

`group.share` calculates the average or percentage for an input data frame and a column name by country. As it is applied on a data frame grouped by gender, one possible outcome is the average grip strength for Swedish women included in the survey. The particular share is calculated by finding the sum of occurences and dividing by their total, represented by the length of the vector. A related procedure that allows for multiple subsets and based upon `dplyr` is used in `labor`, which calculates labor participation rates for all combinations of country, gender and age.

```r
# Function for calculation group shares per country
group.share = function(df, y, agg.country = TRUE) {
    if (agg.country) {
        final = tapply(df[[y]], df$country, function(x) {
            val = sum(x)/length(x)
            return(val)
        })
    } else { # return total if country-level aggregation not specified
        final = sum(df[[y]])/length(df[[y]])
    }
    return(final)
}


# Function for calculating labor participation rate per country / gender / age
labor = function(df, .country, .gender, .age) {
    dataset = df %>%
        filter(country == .country & gender == .gender & get(.age) == TRUE)
    denom = nrow(dataset)
    numer = with(dataset, sum(labor_ft | labor_pt))
    perc  = round(numer/denom, 4)

    return(perc)
}
```

Within the code below, `args.list` contains three sub-lists divided by countriy (`c`), gender (`g`) and age
(`a`). The list is passed as an argument to `labor`, which are simulatenously iterated over the sub-lists using
`pmap`. This function has an advantage over a similar functions, since it allows multiple arguments and uses
parallelization by default.

```r
# use expand.grid to get all combinations of country / gender / age
agegroups = names(df.out[, 4:6])
args      = with(df.out,
                 expand.grid(levels(country), levels(gender), agegroups,
                             stringsAsFactors = FALSE))
args.list = list(c = as.vector(t(args[1])),
                 g = as.vector(t(args[2])),
                 a = as.vector(t(args[3])))


# Map over multiple arguments
output   = pmap(args.list, function(c, g, a) labor(df.out, c, g, a))
output.v = do.call("c", output)

labor.part.share    = cbind(args, output.v)
labor.part.share.df = spread(labor.part.share, Var3, output.v) %>%
```

```
    arrange(desc(Var2)) %>%
    set_colnames(c("Country", "Gender",
                    paste0("Age ", rep(c("50-54", "55-59", "60-64")))))
```

The resulting `output` list can be easily bound into a data frame and arranged using `dplyr`. We then use `tidyr::spread` in order to spread the variable `Var3`, being the three age groups, into separate columns. Using the pipe operator `%>%` keeps our code compact and avoids unnecessary variable assignment and inefficient memory requirements.

In the next code chunk, `vars` is assigned to be a name vector that specifies whether labor force participation is full time, part time or not prevalent. Fetching it using `grep` from the cleaned data set makes it robust against name changes in the underlying data frame and ensures consistency. Conveniently, `grep` does not necessarily require regular expressions as an input. Instead, it is able to perform matches on simple strings and, in this case, returns the column names that contain 'labor'. This is forwarded as a name index to the second argument in `group.share` (comment: name the first code chunk in this section). It is used to subset the target data frame `df` by the extracted variable names containing the string "labor". Since this task involves invoking a function upon multiple list elements, `lapply` is the best candidate in order to construct labor participation rates grouped by gender, (optionally) by country and classified into non participation, part time and full time work.

```
# Labor supply choice tables
df.female = df.out %>% filter(gender == "FEMALE")
vars      = names(df.out)[grep("labor", names(df.out))][3:1] # invert

labor.supply.f    = lapply(vars, function(x) group.share(df.female, x, 1))
labor.supply.f.df = do.call(cbind.data.frame, labor.supply.f) %>%
    set_colnames(c("Nonparticipation", "Half time", "Full time"))
```

3.3 Empirical Results

Before interpreting the data, we advise to note that some country's participants are generally older than other ones. As the evaluation of the probit model shows, we figure age to have a negative impact on health. Hence certain samples may appear less healthy than others due to the fact that they are older on average. Additionally, depending on the national pension schemes, workers may have a different incentive to extend or restrict working depending on their age (Kalwij and Vermeulen 2005).

As depicted in the table below, the mean amount of chronic disease is 75%. On the other hand, 79% out of the 12689 participants have a good self-perceived health. At first sight, one may deduce that a the majority considers themselves to be healthy despite the prevalence of a chronic disease. But the use of the arithmetic mean is sensitive towards outliers and the amount of chronic diseases is in fact counted numerically. In combination, these values indicate an uneven distribution. Surprisingly, the share of overweight people is only 42% compared to an EU-average which Eurostat (2014) reports reports to be 59.6% among 45 to 64 year old citizens. However, information on obesity grouped by age is not available to us.

Table 1: Summary Statistics over all groups

	TOTAL
Observations	12689.00
Age 50-54	0.36
Age 55-59	0.37
Age 60-64	0.27
Num chronic diseases (mean)	0.75
ADLs (in %)	0.04
Max grip strength (mean)	37.87
Overweight (in %)	0.42
Obese (in %)	0.18
Bad mental health (in %)	0.21
Good self-perceived health (in %)	0.79

Having obtained the share of labor force participation grouped by gender, country and age, striking values can be presented.

Among the selected countries, France, the Netherlands and Belgium have the fewest laborers among the oldest age group, which is ranging from 60 to 64 years. In the middle age group, ranging from 54-59, the Belgian and Italian population have sparse labor participation rates among men and women. Within the youngest age group including people from 50 to 54 years old, variance among men is generally low. This is not the case female workers, among which Greek, Italian and Spanish participation rates do not exceed 45.5%.

Concerning health statistics, Switzerland and the Scandinavian countries stand out positively. Together, they from a group that has an overall low percentage of obese people while achieving high shares of self-reported good health. Among all other countries, the Swiss have the least number of chronic diseases and fewest overweight. Only the Greek have lower share of ADL's.

Switzerland and the Nordic countries additionally present the least non-participation rates and the highest full-time working rates. Summary Statistics show that those countries, which strive in health topics are related to high labor force participation rates in all categories.

The overall age distribution is almost even in the younger age brackets (36% and 37%). The participants beween 54 and 59 years old make up 27%. Generally, labor force participation is higher for males than for females. This applies to part-time work, too.

The overall results of the descriptive statistics align well with Kalwij and Vermeulen (2005).

Table 2: Men: Labor Force Participation Rate by age

	Age 50-54	Age 55-59	Age 60-64
Austria	0.8200	0.5700	0.1250
Belgium	0.7879	0.5398	0.2139
Denmark	0.8333	0.7537	0.5625

	Age 50-54	Age 55-59	Age 60-64
France	0.8549	0.5746	0.1729
Germany	0.8025	0.7538	0.3704
Greece	0.8359	0.7040	0.4545
Italy	0.8443	0.5220	0.2738
Netherlands	0.8688	0.7585	0.2969
Spain	0.8031	0.7410	0.3966
Sweden	0.9254	0.8147	0.7005
Switzerland	0.9114	0.8947	0.6349

Table 3: Women: Labor Force Participation Rate by age

	Age 50-54	Age 55-59	Age 60-64
Austria	0.6395	0.3235	0.1129
Belgium	0.6200	0.3105	0.1261
Denmark	0.8467	0.7099	0.2768
France	0.7007	0.5815	0.1938
Germany	0.7738	0.5767	0.2350
Greece	0.3849	0.2929	0.1507
Italy	0.4337	0.2410	0.0762
Netherlands	0.5939	0.4836	0.1616
Spain	0.4550	0.3698	0.2063
Sweden	0.8378	0.8109	0.6000
Switzerland	0.7234	0.6234	0.4643

4 Crosssectional probit regression

4.1 Theory and Design

Probit models allow for the estimation of binary, random response variables (Schild 2017).
Probit and Logit models mitigate problems that would occur in case a (linear) regression would be used for a similar task. In this case, the range of the response variable would not be restricted to the interval (0,1). Consequently, as probabilities are commonly defined on this space, non-interpretable probability estimations would be delivered. While literature often refers to use maximum likelihood to estimate coefficients, the `glm` function in `R` uses iterative re-weighted least squares (R Core Team 2013). The estimated coefficients are subject to a Wald Test on joint significance.

Schild (2017) defines the model formula as

$$P(Y = 1|X = x) = \Phi(x'\beta)$$

It holds under the assumption that $\epsilon \sim N(0,1)$ (wikipedia - how to quote?)

Using this model, it is investigated how the probability of labor force participation is influenced by health and age variables.

Whereas a simple linear regression allows for direct interpretation of the estimated coefficients, this is not the case for probit models. As we are particularly interested in the effects of certain regressors on the probability of labor force participation, we are especially focussing on the calculation and evaluation of marginal effects. The marginal effect of x_j on $P(Y = 1|X = x)$ is defined as the effect of a ceteris paribus change of x_j on the conditional probability that the response variable Y is realized as one. In case of a probit model, the marginal effect of x_j on $P(Y = 1|X = x)$ is dependent on x.

Wooldridge (2012) specifies two possible interpretations of the mariginal effect.

The "Partial Effect at the Average" denotes the marginal effect of a regressor on an average person $(\frac{1}{n}\Sigma_{x_i})$. Problematic are discrete x_i because the resulting averages do not represent anybody in particular. Furthermore, in case of a continuous explanatory variable wrapped inside a non-linear function, it remains unclear whether to apply the average over the function or insert the average into named function.

According to Bauer (2014), the Partial Effect at the Average is defined as

$$PEA = \Phi(\beta_0 + \beta_1\Sigma_{i=1}^{n}x_i)\beta_1$$

The "Average Partial Effect"(or "Average Marginal Effect") describes the effect of a regressor on the probability of Y = 1. Bauer (2014) defines it as

$$APE = \frac{1}{n}\Sigma_{i=1}^{n}\Phi(\hat{\beta}_0 + x_i\hat{\beta}_1)\beta_1$$

4.2 Implementation

As stated in the 'Read and Clean' section, `df.splits` is obtained using the `split` function. A list is returned, that seperates the cleaned data frame by country and gender. This structure is helpful throughout the following sections, as it enables us to build a series of concatenated `lapply` functions. This function invokes a c-loop that iterates over each list element. As `lapply` takes a list as an input and returns a list, too; it preserves the initial structure. First, it is used to create a probit model for each subset within the list. Since we are estimating with an intercept, the age 50 dummy variable is perfectly correlated with the other age dummies. Hence it is excluded due to multicollinearity.

```
# Probit for each country and gender
allModels = lapply(df.splits, function(z){

  z     = z[-z$age50] # Multicollinearity
  model = glm(z$labor_participationTRUE ~., family = binomial(link = "probit"), data = z)

  return(model)
})


# Return summaries
allSummaries = lapply(allModels, summary)
```

The loop below is a robust way to apply the Wald Test for the dataset subsetted by country and gender and utilises only health-related variables. A list intended to log is created, which tracks failure of applying the Wald Test on joint significance of the selected variables. `Try` ensures continuation of the loop despite possible errors. In case an error occurs, the class of `testOutput` will be `try-error` and consequently logged into named list. Since it preserves the structure as specified before, results can be easily transformed and named. This setup has proven to be useful for debugging our test as well as finding out that some health variables are empty for certain subgroups, leading to selective failure of the test.

```
# Wald Test for all models
wald.log = list() # Save Wald Test Output

for(i in 1:length(allSummaries)){

  # Get Element
  SummaryElement = allSummaries[[i]]

  # Specify the of coefficients to be tested: only health variables
  health = c(16:19)

  # Test only the joint significance of health variables
  testOutput = try(joint.wald.test(allSummaries[[i]], 0.95, health))

  if(class(testOutput) == "try-error"){
```

```
    msg = paste0("Wald Test failed for Model Element ", i)
    warning(msg)
    wald.log[[i]] = "Error"
  } else{
    wald.log[[i]] = testOutput
  }
  rm(SummaryElement) # clean up
}


wald.bound = as.data.frame(wald.log)
colnames(wald.bound) = names(allModels)
```

Furthermore, the application of the Wald Test can easily be substitued. We use a different Wald Test to validate our results.

```
    testOutput = try(wald.test(b = coef(ModelElement),
                               Sigma = vcov(ModelElement), Terms = health)$result)
```

In line with the proposed concatenated `lapply` structure, the probability of employment can be predicted for each subset. Marginal effects are calculated using `probitmfx`, which firstly recalculates a probit model. As the boolean 'atmean' evaluates to `true`, the marginal effects are defined as the partial effects for the average observation (Fernihough 2014).

```
# Calculate marginal effects and standard errors

# Same structure as before, but must calculate model again.
mfx.Models = lapply(df.splits, function(z){

    z   = z[-z$age50] # Multicollinearity
    res = probitmfx(z$labor_participationTRUE ~.,atmean = TRUE,  data = z)

    return(res)
})
```

4.3   Empirical Results

By means of the presented model, we evaluate how labor force participation behavior of individuals aged 50-64 is influenced by different health indicators across 11 European countries.

Kalwij and Vermeulen (2005) summarize their own results as follows:

- The influence of health (as captured by the indicators) on labor force participation differs among individuals and across countries
- Health improvements are associated with higher labor force participation rates
- Decreasing health explains decreasing labor force participation of the elderly in Europe

These main results are validated by the analysis. As shown representatively on the example below, which is a summary of the probability of Austrian women participating in the labor market as explained by the probit model, age and the prevalence of chronic diseases are negatively related to employment. The overall negative influence and the statistical significance of age on the dependent variable is growing with increasing age. Conversely, high education and having second education positively predict labor force participation for the specified group, but statistical significance usually restricted to one of them. Additionally, we find maximum grip strength to be statistically significant in multiple models. Having children or a partner does not appear to make a difference. Especially the negative influence of age is extraordinarily stable across different models.

	Estimate	Std. Error	z value	Pr($>$|z|)
(Intercept)	0.6844	0.4351	1.57	0.1157
age51	-0.2492	0.4880	-0.51	0.6096
age52	-0.3909	0.4369	-0.89	0.3709
age53	-0.7001	0.4576	-1.53	0.1260
age54	-0.1578	0.5613	-0.28	0.7786
age55	-0.9350	0.4169	-2.24	0.0249
age56	-0.7092	0.4398	-1.61	0.1069
age57	-1.4925	0.4367	-3.42	0.0006
age58	-1.1910	0.4340	-2.74	0.0061
age59	-1.6635	0.5006	-3.32	0.0009
age60	-1.8610	0.4525	-4.11	0.0000
age61	-1.6772	0.4641	-3.61	0.0003
age62	-1.7351	0.4652	-3.73	0.0002
age63	-2.2090	0.4923	-4.49	0.0000
age64	-6.2803	203.4545	-0.03	0.9754
h_chronic	-0.1722	0.0908	-1.90	0.0578
h_adlaTRUE	0.2082	0.4569	0.46	0.6486
h_obeseTRUE	-0.1314	0.1912	-0.69	0.4920
h_maxgrip	0.0201	0.0814	0.25	0.8049
edu_secondTRUE	-0.0076	0.2424	-0.03	0.9751
edu_highTRUE	0.6778	0.2885	2.35	0.0188
children	-0.0162	0.0837	-0.19	0.8468
coupleTRUE	-0.1606	0.1683	-0.95	0.3397

Table 4: Marginal effects of health and age related variables on the probability of labor market participation of Austrian women

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.

```
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. Spefically, 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 5: 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:

```
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

```
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:

```
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.

```
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)
```

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.

```
45                  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 6: 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 7: 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

	Current 50-54	CF 50-54	Current 55-59	CF 55-59	Current 60-64	CF 60-64

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.

```
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.

```
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.

```
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.

```
# 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`.

```
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.

```
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.

```
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')


## Warning: package 'gridExtra' was built under R version 3.3.3

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

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


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

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


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

```
##                     'all', 'MALE', or 'FEMALE'

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

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

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.

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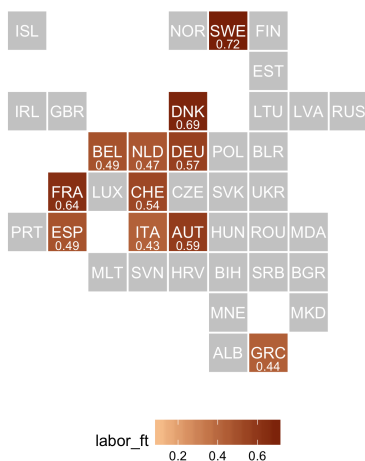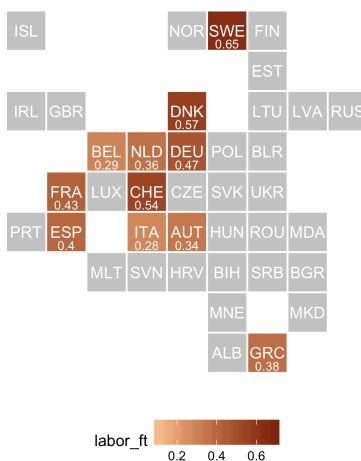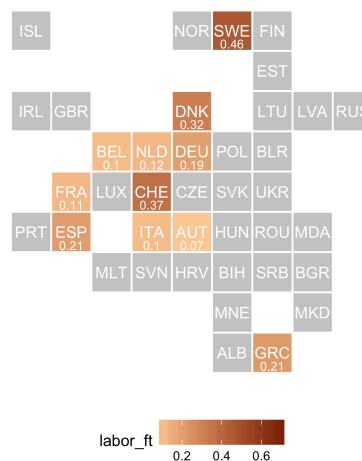

Finally, we display an example of the health distribution map. Here, we seek to evaluate the spread of self-perceived health among survey respondents after removing outliers. The black line indicates the overall distribution for the entire data set. As we can see, there are not too many differences between 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.

```
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.

h_perceived
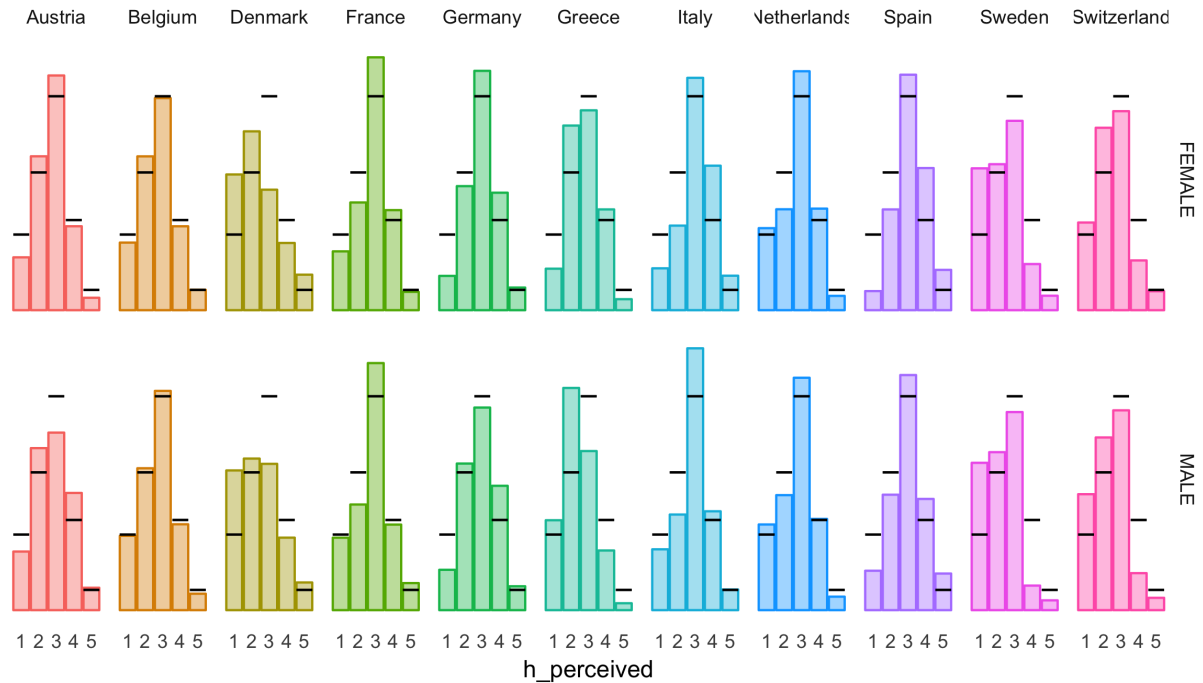
8 Conclusion

The purpose of this paper was to present the developed six quantlets that can be used to carry out statistical and graphical analysis based on the easySHARE data set. Our code is designed to easy the exploration of the easySHARE data set, generate comprehensible and instructive summary statistic tables or graphics and conduct regression-based analysis. The quantlets have been applied and tested on the easySHARE data set to replicate the results of Kalwij and Vermeulen (2005). The code as well as the empirical results have been saved in a workspace accessible on Github.

Our empirical results are in line with those obtained by Kalwij and Vermeulen (2005) in general. However, some minor difference emerge as a results of differing sample size. Based on probit regression, we also find that higher age and worse health conditions are associated with lower labour particpation, although there is great variation in the realtionship across countries. Our results on the joint impact of health variables on participation (Wald test) are also similar except that we cannot reject the null in a higher number of cases. The same pattern emerges in the counterfactual exercise, where our results are diverging for some countries. This indicates that the results obtained by Kalwij and Vermeulen (2005) are not entirely robust and should be validated.

Overall, our quantlets perform well on the tested data set. The output generated by our functions is as expected and can be validated with other packages. It works fine for the replication of the chosen paper and should perform well for other tasks performed on the easySHARE data set. Since the quantlet or constructed

with a modular design, they can be quickly adjusted for other research tasks. While some of our functions go beyond the functionality of other packages (e.g. `wald.test` from `aod` package), our main focus lies on the development of tools to comprehensively analyze the easySHARE data set. One notable source of weakness of our code is its specifictivity and limitation of scope. Since it has been developed for the special usage of the easySHARE data set, it might not perform well on other data. As a future task, our quantlets could therefore be modified to also be applicable to different panel data.

9 References

Bauer, Dietmar. 2014. *Einfuehrung in Die Mikrooekonometrie.* Universitaet Bielefeld.

Borsch-Supan, A. 2017. "Survey of Health, Ageing and Retirement in Europe (Share) Wave 2. Release Version: 6.0. 0. Share-Eric [Data Set]."

Cai, Lixin, and Guyonne Kalb. 2006. "Health Status and Labour Force Participation: Evidence from Australia." *Health Economics* 15 (3). Wiley Online Library: 241–61.

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

Commission, EuroStat European, and others. 2012. "Active Ageing and Solidarity Between Generations." *A Statistical Portrait of the European Union* 2012.

Eurostat. 2014. "Share of overweight population by sex and age." http://ec.europa.eu/eurostat/ statistics-explained/index.php/File:Share_of_overweight_population_by_sex_and_age,_2014.png/.

Fernihough, Alan. 2014. *Mfx: Marginal Effects, Odds Ratios and Incidence Rate Ratios for Glms.* https: //CRAN.R-project.org/package=mfx.

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."

Karpansalo, Minna, Pirjo Manninen, Jussi Kauhanen, Timo A Lakka, and Jukka T Salonen. 2004. "Perceived Health as a Predictor of Early Retirement." *Scandinavian Journal of Work, Environment & Health.* JSTOR, 287–92.

Kassenboehmer, Sonja C, and John P Haisken-DeNew. 2008. "You're Fired! The Causal Negative Effect of Unemployment on Life Satisfaction."

Monden, Christiaan WS. 2005. "Changing Social Variations in Self-Assessed Health in Times of Transition? The Baltic States 1994–1999." *The European Journal of Public Health* 15 (5). Oxford University Press: 498–503.

R Core Team. 2013. *R: A Language and Environment for Statistical Computing.* Vienna, Austria: R Foundation for Statistical Computing. http://www.R-project.org/.

Schild, K.-H. 2017. "Binaere Auswahlmodelle." *Abt. Statistik, Fb. Wirtschaftswissenschaften*, november. Philipps-Universitaet Marburg.

Shields, Michael A, and Stephen Wheatley Price. 2005. "Exploring the Economic and Social Determinants of Psychological Well-Being and Perceived Social Support in England." *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 168 (3). Wiley Online Library: 513–37.

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

———. 2012. *Introductory Econometrics.* Vol. 5. South Western College Pub - A Modern Approach.

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