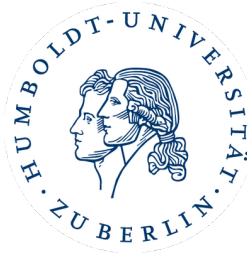


Analysis and Graphical Representation of Health and Labor Force Participation among the Elderly in Europe

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, 2018-03-16

List of Tables

1	Summary Statistics over all groups	9
2	Men: Labor Force Participation Rate by age	10
3	Women: Labor Force Participation Rate by age	10
4	Marginal effects for probability of labor market participation of Austrian women	14
5	Wald test on no joint impact of health on participation for men	17
6	Women's current and counterfactual labor share	22
7	Men's current and counterfactual (CF) labor share for different age groups	23

List of Figures

1	Full-time labor employment tile grid maps by age group	27
2	Distribution of self-perceived health by country and gender	28

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 Cross-National Summary Statistics	7
3.1	Theory and Design	7
3.2	Implementation	7
3.3	Empirical Results	9
4	Cross-Sectional Probit Regression	11
4.1	Theory and Design	11
4.2	Implementation	11
4.3	Empirical Results	13
5	Wald Test	15
5.1	Theory and Design	15
5.2	Implementation	15
5.3	Empirical Results	17
6	Counterfactual Exercise	20
6.1	Theory and Design	20
6.2	Implementation	20
6.3	Empirical Results	22
7	Graphical Representation	24
7.1	Theory and Design	24
7.2	Implementation	24
7.3	Empirical Results	26
8	Conclusion	29
9	References	30
10	Appendix	32
10.1	Quantlet Code	32
10.1.1	Quantlet 1	32
10.1.2	Quantlet 2	36
10.1.3	Quantlet 3	36
10.1.4	Quantlet 4	36
10.1.5	Quantlet 5	40
10.1.6	Quantlet 6	44

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, well-being 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 the opposite effect: Unemployment has also been found to have negative effects on well-being 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 utilized 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 divided into six quantlets, which are meant to be executed chronologically. In **Quantlet 1** , 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** , generates a series of summary statistics tables that present health indicators, labor force participation and labor supply choice by country, age group and gender. In **Quantlet 3** , we conduct probit regression for each country and both genders, calculate marginal effects and average employment probability. **Quantlet 4** , 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** , presents counterfactual exercises that are based on the previously carried out probit estimations. By manipulating the population health indicators, we calculate current and counterfactual labor participation rates for different age groups. In **Quantlet 6** , we introduce new methods to graphically represent the summary statistics for exploratory data analysis. Specifically, we provide tools to generate carefully selected graphics for a subset of desired variables and sample subsets.

2 Data Cleaning and Manipulation

The SHARE survey is a panel data set covering data from six survey waves carried out between 2004 and 2015. It consists of a variety of different health, social, and socioeconomic indicators from 27 European countries. The analysis conducted by Kalwij and Vermeulen (2005) is based on the first wave of the SHARE data set. Their sample contains observations from approximately 18,000 households from eleven European countries that participated in the first wave between 2004 and 2005.

2.1 Theory and Design

The easySHARE data set, released in spring 2017, is the basis of the following analysis. 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 less observations and does not provide the same level of granularity in variables compared to broader SHARE data set. Our main aspiration regarding the data cleaning process is to make our code readily accessible and comprehensible for other users of the easySHARE data set. The rest of this section describes our process of data transformation, aggregation, and cleaning, not only to match as closely as possible to the initial easySHARE 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. 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 may allow us to conduct supplementary panel studies. 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 function utilizes the functions in the `tidyverse` family of functions developed by Hadley Wickham, namely `dplyr` and `magrittr`. The `dplyr` package is designed for uncomplicated data manipulation, and it works specifically with data frames. The package provides a consistent set of verbs that perform small tasks 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.

```
[16] neededPackages = c("dplyr", "magrittr", "infuser", "countrycode")
[17] allPackages     = c(neededPackages %in% installed.packages()[, "Package"])
[18] if (!all(allPackages)) {
[19]   missingIDX = which(allPackages == FALSE)
[20]   needed     = neededPackages[missingIDX]
[21]   lapply(needed, install.packages)
[22] }
```

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

We continue with basic filtering to select only variables that are pertinent to our study. Then we encode missing values. The specific 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”.

```
[53] # Encode missing values according to SHARE data set guidelines  
[54] a = c(-2, -3, -4, -7, -9, -12, -13, -14, -15, -16)  
[55] b = c("tocheck", "implausible", "tocheck", "uncoded", "notApplicable",  
[56]     "dontKnow", "notAskedWave", "notAskedCountry", "noInformation",  
[57]     "noDropOff")  
[58] missing.value.codes = data.frame(a,b)  
  
[62] # Find NA locations and declare them as such  
[63] df.decl = apply(dat, 2, function(z) {  
[64]     na.loc = which(z %in% a)  
[65]     z[na.loc] = NA  
[66]     return(z)  
[67] })
```

We use the `countrycode` package to change the country codes into human-friendly country names. Then we make minor transformations to the variables. 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 (line 91 - 126).

```
[91] df.out = df %>%  
[92]     dplyr::left_join(country_data, by = c("country_mod" = "iso3n")) %>%  
[93]     dplyr::filter(iso3c %in% country_list) %>%  
[94]     dplyr::mutate(country = factor(country.name.en),  
[95]                 gender = factor(ifelse(female, "FEMALE", "MALE")),  
[99]                 age = factor(floor(age)),  
[100]                 edu_low = isced1997_r %in% c(0, 1),  
[102]                 edu_high = isced1997_r %in% c(5, 6),
```

```

[105]             h_chronic      = chronic_mod,
[108]             h_overweight   = bmi2 == 3,
[115]             labor_ft       = ep013_mod >= 32,
[118]             labor_hrs      = floor(ep013_mod)) %>%
[119]     dplyr::select(country, gender,
[121]                   starts_with("h_"),
[124]                   starts_with("labor_")) %>%
[125]     na.omit() %>%                                # remove missing values
[126]     set_rownames(NULL)                            # reset row numbering

```

Our last step involves splitting the data sets, standardizing all numeric variables, converting the data frames into a model matrix and 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 the custom-built `standardize.df()` function, which finds and 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 `dummify()` 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.

```

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

[154]         df.reg = df %>%
[155]             mutate_at(.vars = vars(idx),
[156]                       .funs = standardize) %>%
[157]             mutate(labor_participation = !labor_np) %>%
[158]             dplyr::select(country, gender, age,
[159]                           h_chronic, h_adla, h_obese, h_maxgrip,
[160]                           edu_second, edu_high, children, couple,
[161]                           labor_participation)
[162]
[163]         return(df.reg)
[164]     }

[167]     splits      = split(df.out, f = list(df.out$country, df.out$gender),
[168]                         drop = TRUE)
[169]     df.reg      = standardize.df(df.out)
[170]     df.splits   = lapply(splits, standardize.df)

[173]     dummify = function(df) {
[174]         df = df %>%

```



```

[175]         dplyr::select(-country, -gender)
[176]     model      = ~ 0 + .
[177]     new.df     = model.matrix(model, df)
[178]     new.df     = data.frame(new.df)
[179]     return(new.df)
[180] }
[181]
[182] 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 and the data frames contained within the outputted list.

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.

```

source('Quantlets/SPL_ELP_ReadandClean/SPL_ELP_ReadandClean.R')
datasets = read.and.clean(dataset = "easySHARE_re16_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.

```

3 Multidisciplinary and Cross-National Summary Statistics

3.1 Theory and Design

Since we are working with a cross-national survey, it is useful 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 separating the easySHARE data set into country, gender, and age is explained in the following. In order to minimize hard-coding and possibly related errors, we use the `expand.grid()` function to expand a grid between named variables in the data set. This allows us to calculate the necessary summary statistics for all combinations of country, gender, and age.

We create the function `labor()`, which calculates labor participation rates for the corresponding country, gender, and age cell within the grid. The `group.share()` function 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 occurrences and dividing by their total, represented by the length of the vector.

```
[52] # Function for calculation group shares per country
[53] group.share = function(df, y, agg.country = TRUE) {
[54]   if (agg.country) {
[55]     final = tapply(df[[y]], df$country, function(x) {
[56]       val = sum(x)/length(x)
[57]       return(val)
[58]     })
[59]   } else { # return total if country-level aggregation not specified
[60]     final = sum(df[[y]])/length(df[[y]])
[61]   }
[62]   return(final)
[63] }
[64]
[65] # Function for calculating labor participation rate per country / gender / age
[66] labor = function(df, .country, .gender, .age) {
[67]   dataset = df %>%
[68]     filter(country == .country & gender == .gender & get(.age) == TRUE)
[69]   denom = nrow(dataset)
[70]   numer = with(dataset, sum(labor_ft | labor_pt))
[71]   perc = round(numer/denom, 4)
[72]
```

```
[73]     return(perc)
[74] }
```

Within the code below, `args.list` (line 81-83) contains three sub-lists divided by country (`c`), gender (`g`) and age (`a`). The list is passed as an argument to `labor`, which are simultaneously 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.

```
[76] # use expand.grid to get all combinations of country / gender / age
[77] agegroups = names(df.out[, 4:6])
[78] args      = with(df.out,
[79]               expand.grid(levels(country), levels(gender), agegroups,
[80]                           stringsAsFactors = FALSE))
[81] args.list = list(c = as.vector(t(args[1])),
[82]                 g = as.vector(t(args[2])),
[83]                 a = as.vector(t(args[3])))
[84]
[85] # Map over multiple arguments
[86] output    = pmap(args.list, function(c, g, a) labor(df.out, c, g, a))
[87] output.v  = do.call("c", output)
[88]
[89] labor.part.share    = cbind(args, output.v)
[90] labor.part.share.df = spread(labor.part.share, Var3, output.v) %>%
[91]   arrange(desc(Var2)) %>%
[92]   set_colnames(c("Country", "Gender",
[93]                 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 call 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.

```
[105] # Labor supply choice tables
[106] df.female = df.out %>% filter(gender == "FEMALE")
```

```

[107] vars      = names(df.out)[grep("labor", names(df.out))][3:1] # invert
[108]
[109] labor.supply.f = lapply(vars, function(x) group.share(df.female, x, 1))
[110] labor.supply.f.df = do.call(cbind.data.frame, labor.supply.f) %>%
[111]     set_colnames(c("Nonparticipation", "Half time", "Full time"))

```

3.3 Empirical Results

We can verify the validity of our results by comparing them directly with the findings in the original paper. 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, 79% out of the 12689 participants have a good self-perceived health. The number of chronic diseases is 0.75 on average. Interpretation of this arithmetic mean may be misleading as it is sensitive towards outliers and since the number of chronic diseases is distributed quite unevenly between individuals. Surprisingly, the share of overweight people is only 42% compared to an EU-average which Eurostat (2014) reports to be 59.6% among 45 to 64 year old citizens.

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 labor share 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. The opposite is true for 50 to 54 year-old female workers: Participation rate in southern countries do not exceed 45.5%, while they reach values close to 85% in northern countries.

Concerning health statistics, Switzerland and the Scandinavian countries stand out positively. Together, they form 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 lowest mean of chronic diseases and fewest overweight rate. 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 between 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
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 Cross-Sectional Probit Regression

4.1 Theory and Design

Probit models allow for the estimation of binary, random response variables (Schild 2017). One advantage of probit models over linear regression models is that the range of the response variable is restricted to the (0,1) interval. This allows us to interpret estimates derived from probit models as probabilities. Probit models are usually estimated based on Maximum Likelihood. The `glm` function in R uses iterative re-weighted least squares to find the maximum likelihood estimates for the probit coefficients (R Core Team 2013). Schild (2017) defines the probit model formula as

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

The error term is assumed to be independent and identically distributed (Wooldridge (2012)). Using this model, it is investigated how the probability of labor force participation is affected 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 where we deal with non-linear effects. Instead of interpreting the model coefficients, we focus on 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 marginal effect. The “Partial Effect at the Average” (PEA) denotes the marginal effect of a regressor on an average person ($\frac{1}{n}\sum x_i$). 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. We therefore focus on the “Average Partial/Marginal Effect” (APE), which describes the average effect of a regressor on the probability that $Y = 1$ of all individuals. Bauer (2014) defines it as

$$APE = \frac{1}{n} \sum_{i=1}^n \Phi(\hat{\beta}_0 + x_i \hat{\beta}_1) \beta_1 \quad (2)$$

where β is the vector of coefficients and x_i is the vector of observations at index i. Φ represents the cumulative standard normal distribution function.

4.2 Implementation

As stated in the ‘Read and Clean’ section, `df.splits` is obtained using the `split` function. A list is returned that separates the cleaned data frame by country and gender.

This structure is helpful throughout the following sections because 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 to avoid multicollinearity.

```

[53] # Probit for each country and gender
[54] allModels = lapply(df.splits, function(z){
[55]
[56]     z      = z[-z$age50] # Multicollinearity
[57]     model = glm(z$labor_participationTRUE ~., family = binomial(link = "probit"), data = z)
[58]
[59]     return(model)
[60]
[61] })
[62]
[63] # Return summaries
[64] allSummaries = lapply(allModels, summary)

```

The loop below is a robust way to apply the Wald Test on health-related variables for the dataset subsetted by country and gender. 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 at the index `i`. 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.

```

[70] # Wald Test for all models
[71] wald.log = list() # Save Wald Test Output
[72]
[73] for(i in 1:length(allSummaries)){
[74]
[75]     # Get Element
[76]     SummaryElement = allSummaries[[i]]
[77]
[78]     # Specify the of coefficients to be tested: only health variables
[79]     health = c(16:19)
[80]
[81]     # Test only the joint significance of health variables
[82]     testOutput = try(joint.wald.test(allSummaries[[i]], 0.95, health))
[83]
[84]     if(class(testOutput) == "try-error"){
[85]         msg = paste0("Wald Test failed for Model Element ", i)
[86]         warning(msg)
[87]         wald.log[[i]] = "Error"
[88]     } else{
[89]         wald.log[[i]] = testOutput
[90]     }
[91]     rm(SummaryElement) # clean up
[92] }

```

```

[93]
[94] wald.bound = as.data.frame(wald.log)
[95] colnames(wald.bound) = names(allModels)

```

Furthermore, the application of the Wald Test can easily be substituted. We use a different Wald Test from the `aod` package to validate our results.

```

[82] testOutput = try(wald.test(b = coef(ModelElement),
[83]                      Sigma = vcov(ModelElement), Terms = health)$result)

```

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).

```

[117] # Calculate marginal effects and standard errors
[118]
[119] # Same structure as before, but must calculate model again.
[120] mfx.Models = lapply(df.splits, function(z){
[121]
[122]     z = z[-z$age50] # Multicollinearity
[123]     res = probitmfx(z$labor_participationTRUE ~.,atmean = TRUE, data = z)
[124]
[125]     return(res)
[126] })

```

4.3 Empirical Results

By means of the presented model, we evaluate how labor force participation behavior of individuals aged 50-64 is affected by age and demographic as well as health indicators across 11 European countries. Kalwij and Vermeulen (2005) summarize their own results as follows: In general, good health indicators are associated with higher labor force participation rates. Illness-related covariates have a negative effect on labor force participation rates. The influence of health (as captured by the indicators) on labor force participation differs between age groups, gender and across countries.

These main results are validated by our analysis. Table 4 shows a representative example of the estimation summary for Austrian women. Increasing age, obesity 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.

Table 4: Marginal effects for probability of labor market participation of Austrian women

	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

5 Wald Test

In the context of probit regression, the Wald test can be used to test multiple hypotheses regarding the model specifications. Kalwij and Vermeulen (2005) conduct a Wald test to check the null hypothesis that none of the included health variables has an impact on labor 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

$$W = c(\hat{\theta})' [\nabla_{\theta} c(\hat{\theta})' \hat{V} \nabla_{\theta} c(\hat{\theta})]^{-1} c(\hat{\theta}) \quad (3)$$

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 asymptotic covariance matrix (Wooldridge 2010, 463). Under H_0 , the Wald test statistic is asymptotically χ^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 H_0 , 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 null hypothesis in accordance with the common linear restriction structure of $H_0: R\hat{\beta} = r$ against the alternative $H_1: R\hat{\beta} \neq r$ 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

$$W = (R\hat{\beta} - r)' [R\hat{V}R']^{-1} (R\hat{\beta} - r) \quad (4)$$

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 $W \stackrel{a}{\sim} \chi^2_s$ can be simplified further (Wooldridge 2010, 362):

$$W = \hat{\beta}'_s [R\hat{V}R']^{-1} \hat{\beta}_s = \hat{\beta}'_s [\hat{V}_s]^{-1} \hat{\beta}_s \quad (5)$$

5.2 Implementation

To test the linear hypothesis regarding the model specifications, we design two Wald test versions. The function `joint.wald.test` tests the joint significance of a subset of model coefficients, whereas `general.wald.test` allows us to check any linear hypothesis. The two tests are designed in a way that they align well with the

glm estimation output. The only required input is the model summary from glm estimation; the significance level and test specifications are optional.

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

[46]     joint.wald.test      = numeric(4)
[47]     names(joint.wald.test) = c("W", "p-value", "df", "Decision")
[48]     beta                 = model.summary$coefficients[,1]
[49]     Var_beta_est         = vcov(model.summary)
```

To set up default values for the hypothesis specification, we make use of a local if-else statement inside the `joint.wald.test` function. In line 26 we reassign a sequence vector to `spec` that includes the number of all model coefficients in increasing order, unless `spec` is specified differently. This means `joint.wald.test` will conduct a joint significance test on all model coefficients by default. The Wald test statistic is calculated via simple matrix algebra based on the model summary input. This formula is equivalent to equation 5.

```
[25] if (is.null(spec)) {
[26]     spec = 1:length(beta) # default joint is significance test
[27] } else if (is.logical(spec)) {
[28]     stop("spec cannot be a logical vector, tranform to integer vector!")
[29] } else if (any(spec == 0)) {
[30]     stop("spec cannot contain zero values!")
[31] } else if (!is.integer(spec)) {
[32]     spec.len = length(spec)
[33]     spec     = as.integer(spec, length = spec.len)
[34]     warning("Converting spec to integer and proceeding")
[35] }

[51] # Wald test statistic
[52] W = t(beta[spec]) %*% solve(Var_beta_est[spec,spec]) %*% beta[spec]
```

The proper format of the specification vector `spec` is crucial as it is used to extract the needed estimates from the model coefficient vector and covariance matrix. To select the right covariance elements and degrees of freedom, `spec` must be a vector of integers of length $0 < m \leq k$. Specifically, we make sure `spec` is neither logical nor contains any zero in line 27-30. Furthermore, we only allow for integers since we are testing the joint significance of coefficients.

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 is given by equation 4. As in the `joint.wald.test` function, we use if-else statements to setup 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 for 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 . These conditions are tested in line 123-131.

```

[100]   if (class(model.summary) != "summary.glm") {
[101]       stop("model.summary must be a glm summary!")
[102]   }
[103]   if (confidence.level > 1 | confidence.level < 0) {
[104]       stop("confidence.level out of bounds!")
[105]   }

[123]   dim_R = dim(R) # Testing R
[124]   k      = length(beta) # Columns
[125]   m      = length(r) # Rows; general hypothesis test, therefore m = length(r)

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

```

We furthermore check whether the `model.summary` has the right class and whether the significance level lies within the expected interval. Otherwise, the execution of the function will be stopped. In the second step, we make sure the matrix `R` and the restriction vector `r` have matching dimensions.

5.3 Empirical Results

We use our designed Wald test functions to check the null hypothesis that none of the four included health variables has an impact on labor participation for each country and for both men and women. The results for men can be seen in table 5.

Table 5: Wald test on no joint impact of health on participation for men

	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 are in line with the findings of Kalwij and Vermeulen (2005). As expected, the null hypothesis of no impact of health on labor participation is rejected at a 95% confidence level for most countries. One difference is the inability to reject the null for France. 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 participation decision. The results of our `joint.wald.test` function are comparable those of our packages, like the `wald.test` from the `aod` package. For example, for German women, the `aod` package 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 handle empty classes. For the case of Switzerland, where our male sample does not include any 64-year-olds, the `wald.test` from the `aod` package leads to an error message.

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

```
## Error in L %*% V: non-conformable arguments
```

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.

```
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 executing 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 labor 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 assuming 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 are 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 run 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.

$$\Pi_{icf} = Pr(Y_i = 1 | X_i = x_{icf}) = \Phi(x_{icf}'\beta) \quad (6)$$

We can assess counterfactual health-related effects by comparing the actual and counterfactual employment rates Π_{ac} and Π_{cf} . We gain closer insights into the non-linear relation between health effects and participation 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 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 54, the function creates a duplicate version of original data X , whose values will be replaced later on.

In line 60, 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 standardized

variables. 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 64, 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.

```
[50] X.cf = function(model) {
[52]     X                = model$data
[54]     X_cf             = X
[57]     X_min            = data.frame(t(apply(X, 2, min)))
[59]     names.vec        = c("h_chronic", "h_adlaTRUE", "h_obeseTRUE")
[60]     X_cf[, names.vec] = X_min[names.vec]
[62]     age_50_51        = ifelse(X$age50 == 1 | X$age51 == 1, 1, 0)
[64]     X_cf[, c("h_maxgrip")] = tapply(X$h_maxgrip, age_50_51 == 1, mean)[[2]]
[65]     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 obtain 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 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.

```
[88] empl.rate.age = function(model, group.size = 5, group.low = c(50,55,60)) {
[94]     X                = model$data
[96]     # Predict probability of being employed of all individuals
[97]     empl.probability = predict(object = model, newdata = X, type = "response")
```

The proper value assignment of the numerical vector `group.low` is essential since we base the dynamic creation of age groups on it. As can be seen in line 104, we use a small `for` loop to determine the age group upper 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`.

```
[104] for (i in group.low) {
[106]     k          = i + group.size - 1
[108]     vec.label = paste("names.vec.age", i, sep = ".")
[110]     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 118. 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, depending on whether the individual belongs to the age group or not. Following this step, we filter out only the relevant age groups via indexing.


```

[118] IND_row_vec[[i]] = apply(X[, z], 1, sum)

[123] IND_row_vec = IND_row_vec[group.low]

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

```

Finally, we calculate the employment rate for each age group in line 128. For this, we first sum the predicted employment probability for all individuals in age group k via vector multiplication. Next, we divide 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 6 displays these results on an aggregate level. On average, the predicted participation rate is 4.5 percentage points given the perfectly healthy individuals (*ceteris paribus*), although there are quite some outliers. For Dutch women, the estimated counterfactual participation rate is 8 percentage points higher, corresponding to a relative increase of 18%. This is due to the high coefficients of chronic disease and obesity in absolute terms, combined with the above average prevalence of chronic disease and obesity 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 obesity in absolute terms. As expected, our results are in line with Kalwij and Vermeulen (2005) generally, though the estimated participation 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 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 artificial 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 7. 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. 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

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. Tables, although useful for organizing and displaying many discrete data points, lack both detail and readability. 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. They are useful for identifying values that are associated with a given geographic location and quickly comparing those values across different regions.

However, we opt not to use choropleths, instead focusing on an 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 the perception for a reader on the relative importance of such a region.

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.

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 define the initial plot, then ‘layer’ on additional graphical elements as needed.

```
[100] plot_function = function(df) {  
[101]  
[102]   p = ggplot(data = df, aes(x = X, y = Y)) +  
[103]     geom_tile(aes(fill = get(xvar), color = ""),  
[104]              color = "white", size = 0.6) +  
[105]     geom_text(aes(label = iso3c), color = "white") +
```

```

[106]     geom_text(aes(label = round(get(xvar), 2)), vjust = 2,
[107]               color = "white", size = 3, na.rm = TRUE) +
[108]     coord_fixed(ratio = 1) +
[109]     theme_minimal() +
[110]     theme(axis.line       = element_blank(),
[111]           axis.text       = element_blank(),
[112]           axis.title      = element_blank(),
[113]           panel.background = element_blank(),
[114]           panel.grid      = element_blank(),
[115]           legend.position  = "bottom") +
[116]     theme(plot.title      = element_text(size = 16),
[117]           plot.subtitle   = element_text(size = 10,
[118]                                           color = "#7F7F7F"))
[119]     return(p)
[120]   }

```

The range of the 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.

```

[174]   minval = min(df.f[[xvar]], df.m[[xvar]], na.rm = TRUE)
[175]   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 multiple 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.

```

example1 = health.gridmap('labor_hrs', 'age')
grid.draw(example1)
ggsave("Output/gridmap_laborhrs_byage.png", plot=example1, 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. An optional parameter allows for the removal of outliers.

```

# Remove outliers (outside 1.5 IQR of 25% and 75% percentiles)
[370]   rm.outliers = function(x, na.rm = TRUE) {
[371]     qnt = quantile(df.out[[x]], probs = c(0.25, 0.75), na.rm = na.rm)
[372]     out = 1.5*IQR(df.out[[x]], na.rm = na.rm)
[373]
[374]     new.out = df.out

```

```

[375]     new.out[(df.out[[x]] < qnt[1] - out), ] = NA
[376]     new.out[(df.out[[x]] > qnt[2] + out), ] = NA
[377]
[378]     new.out = new.out[complete.cases(new.out), ]
[379]     return(new.out)
[380] }
[381]
[382] 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.

```

[385]     total.dist = table(df.out[[xvar]])/nrow(df.out)
[386]
[387]     total      = expand.grid(countries, gen, sort(unique(df.out[[xvar]])))
[388]     names(total) = c("country", "gender", xvar)
[389]     total      = arrange(total, country, gender, get(xvar))
[390]     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.

```

[418] if (length(gen) == 1) {
[419]     plot.bar = plot.bar +
[420]         labs(subtitle = paste0("Faceted by country. Gender selected is ", gen,
[421]             ". Distribution of the entire data set shown is overlaid on each graph. ")) +
[422]         facet_wrap(~ country, ncol = 4)
[423] } else {
[424]     plot.bar = plot.bar +
[425]         facet_grid(gender ~ country)
[426] }

```

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('Quantlets/SPL_ELP_Graphics/SPL_ELP_Graphics.R')
load('easySHARE_clean.RData')

#Only keep relevant data sets

```

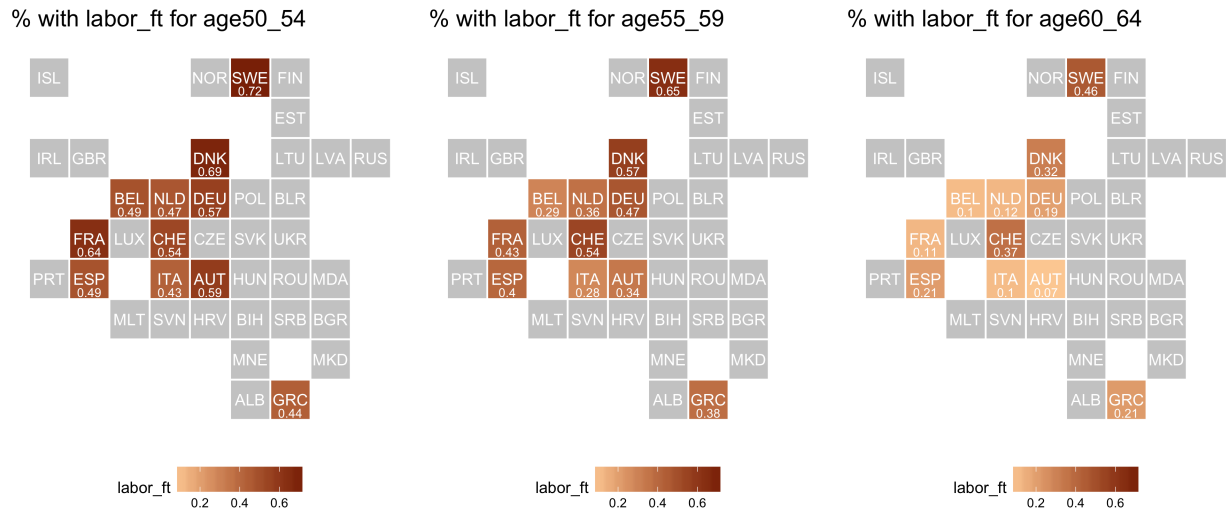


Figure 1: Full-time labor employment tile grid maps by age group

```
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
```

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

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

We show an examples of correct output below. Here, we want to full-time employment percentage, faceted by age group. As we can see in Figure 1, 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. As can be seen in Figure 2, we seek to evaluate the spread of self-perceived health among survey respondents after removing outliers. The black line indicates

Distribution of h_perceived by Country

Faceted by country & gender. Distribution of the entire data set shown is overlaid on each graph.

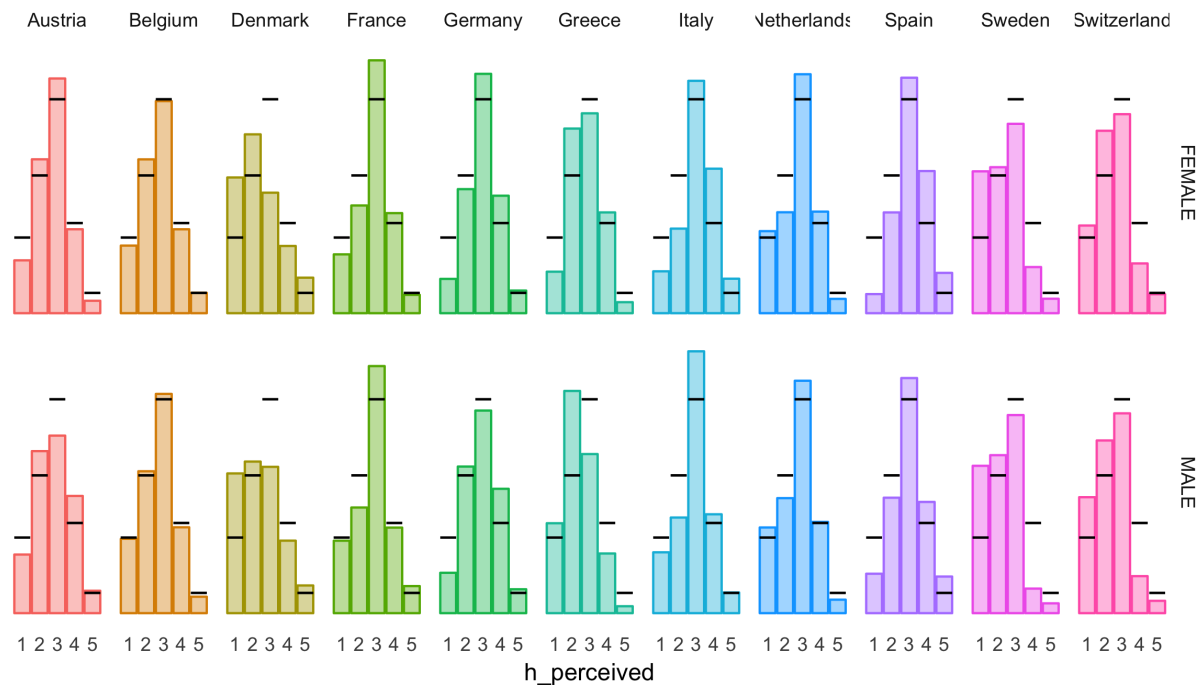



Figure 2: Distribution of self-perceived health by country and gender

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)
```

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 simplify the exploration of the easySHARE data set, generate comprehensible and instructive summary statistic tables and graphics, and conduct regression-based analyses. 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](#) .

In general, our empirical results are in line with those obtained by Kalwij and Vermeulen (2005). However, some minor differences emerge as a result of differing sample sizes by country and differing availability in preciseness of survey responses. Based on using a probit regression, we also find that higher age and worse health conditions are associated with lower labor participation, although there is great variation in the relationship across countries. Our results on the joint impact of health variables on participation (Wald test) are also similar except that we are unable to 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 is 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 specificity 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 sets.

9 References

- Arel-Bundock, Vincent. 2017. *Countrycode: Convert Country Names and Country Codes*. <https://CRAN.R-project.org/package=countrycode>.
- Auguie, Baptiste. 2016. *GridExtra: Miscellaneous Functions for “Grid” Graphics*. <https://CRAN.R-project.org/package=gridExtra>.
- Bache, Stefan Milton, and Hadley Wickham. 2014. *Magrittr: A Forward-Pipe Operator for R*. <https://CRAN.R-project.org/package=magrittr>.
- Bates, Douglas, and Martin Maechler. 2017. *Matrix: Sparse and Dense Matrix Classes and Methods*. <https://CRAN.R-project.org/package=Matrix>.
- Bauer, Dietmar. 2014. *Einfuehrung in Die Mikroekonometrie*. 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.
- Chang, Winston. 2017. *Webshot: Take Screenshots of Web Pages*. <https://CRAN.R-project.org/package=webshot>.
- 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.
- Henry, Lionel, and Hadley Wickham. 2017. *Purrr: Functional Programming Tools*. <https://CRAN.R-project.org/package=purrr>.
- 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.”

Leeper, Thomas J. 2017. *Margins: Marginal Effects for Model Objects*.

Lesnoff, M., Lancelot, and R. 2012. *Aod: Analysis of Overdispersed Data*. <http://cran.r-project.org/package=aod>.

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/>.

Ren, Kun, and Kenton Russell. 2016. *Formattable: Create 'Formattable' Data Structures*. <https://CRAN.R-project.org/package=formattable>.

RStudio, and Inc. 2017. *Htmltools: Tools for Html*. <https://CRAN.R-project.org/package=htmltools>.

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.

Smeets, Bart. 2017. *Infuser: A Very Basic Templating Engine*. <https://CRAN.R-project.org/package=infuser>.

Wickham, Hadley. 2009. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. <http://ggplot2.org>.

Wickham, Hadley, and Winston Chang. 2017. *Devtools: Tools to Make Developing R Packages Easier*. <https://CRAN.R-project.org/package=devtools>.

Wickham, Hadley, and Lionel Henry. 2017. *Tidyr: Easily Tidy Data with 'Spread()' and 'Gather()' Functions*. <https://CRAN.R-project.org/package=tidyr>.

Wickham, Hadley, Romain Francois, Lionel Henry, and Kirill Müller. 2017. *Dplyr: A Grammar of Data Manipulation*. <https://CRAN.R-project.org/package=dplyr>.

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.

10 Appendix

10.1 Quantlet Code

10.1.1 Quantlet 1



```
#####  
# SPL_ELP_ReadandClean.R  
#  
#####  
# Description:  
# Function loads easySHARE data set and outputs relevant data frames for  
# analysis and regression. Additionally, print error messages if data set is  
# not the correct easySHARE data set and relevant statistics printed (e.g. num  
# rows dropped, variables retained, data type)  
#  
#####  
  
read.and.clean <- function(dataset = "easySHARE_rel6_0_0.rda", wav = 1) {  
  
  # LOAD NECESSARY PACKAGES & DATA  
  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))))  
  
  # Load dataset  
  cat("Loading data set...", sep = "\n")  
  load(dataset)  
  
  dat.input = easySHARE_rel6_0_0  
  rm(easySHARE_rel6_0_0)  
  
  rows.dat.input = nrow(dat.input)  
  
  if (!(wav %in% 1:6)) stop('Out of bounds.  
    Select a value between 1 and 6.')
```

```
#####
# ENCODE MISSING VALUES

# Organize data.frame by selecting relevant variables
cat(infuse("Selecting values only from Wave {{wav}} and between ages 50 and 64.", wav = wav), sep =

dat = dat.input %>%
  dplyr::filter(wave == wav & (age <= 64 & age >= 50)) %>%
  dplyr::select(wave, country_mod,                # dataset details
    female, age, isced1997_r, ch001_, mar_stat, # demo variables
    chronic_mod, maxgrip, adla, bmi, bmi2, eurod, sphus, # health
    ep013_mod, ep005_)                               # labor (outcome var)

rm(dat.input)

# Encode missing values according to SHARE dataset 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)
# This data frame can be used to verify NA codes easily.
# But for the encoding only the numeric vector a is necessary.

# 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)
})

df = data.frame(df.decl)

# If working hours is NA and job status is not employed, this means
# individuals don't work
# If working hours is NA and job status is employed, we assume part time
# work
df$ep013_mod[is.na(df$ep013_mod) & df$ep005_ != 2] = 0
df$ep013_mod[is.na(df$ep013_mod) & df$ep005_ == 2] = 1

#####
# CREATE DATA FRAMES FOR ANALYSIS AND ESTIMATION
```

```

# Get country information from "countrycode" package
country_list = c("BEL", "NLD", "FRA", "SWE", "DEU", "GRC", "ITA", "ESP",
                 "DNK", "AUT", "CHE")

country_data = with(countrycode_data,
                    data.frame(iso3c, iso3n, country.name.en))

# Variables are cleaned, converted into human-readable naming conventions,
# and converted to dummies as described in the paper
cat("Removing missing values.", sep = "\n")

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")),
                age50_54    = age < 55,
                age55_59    = age >= 55 & age < 60,
                age60_64    = age >= 60,
                age         = factor(floor(age)),
                edu_low     = isced1997_r %in% c(0, 1),
                edu_second  = isced1997_r %in% 2:4,
                edu_high    = isced1997_r %in% c(5, 6),
                children    = ch001_,
                couple      = mar_stat %in% 1:3,
                h_chronic   = chronic_mod,
                h_maxgrip    = maxgrip,
                h_adla      = adla > 0,
                h_overweight = bmi2 == 3,
                h_obese     = bmi2 == 4,
                h_badmental  = eurod > 3,
                h_goodsp     = sphus < 4,
                h_bmi        = floor(bmi),
                h_depression = eurod,
                h_perceived  = sphus,
                labor_ft     = ep013_mod >= 32,
                labor_pt     = ep013_mod < 32 & ep013_mod > 0,
                labor_np     = ep013_mod == 0,
                labor_hrs    = floor(ep013_mod)) %>%
  dplyr::select(country, gender,          # country and gender
                starts_with("age"),       # age dummy
                starts_with("h_"),        # health indicators
                starts_with("edu_"),       # education dummies

```

```

        children, couple,                # demographic details
        starts_with("labor_")) %>%      # labor supply outcomes
na.omit() %>%                            # remove missing values
set_rownames(NULL)                      # reset row numbering

rows.df.out    = nrow(df.out)
rows.remove    = rows.dat.input - rows.df.out

cat(infuse("Rows removed:  {{rows.remove}}", rows.remove = rows.remove),
    sep = "\n")
cat(infuse("Rows remaining: {{rows.dat}}", rows.dat = rows.df.out),
    sep = "\n")
cat("", sep = "\n")

# Create standardized variables for numeric data
standardize = function(x) {
  mean = sum(x)/length(x)
  std  = sd(x)
  val  = (x - mean) / std
  return(val)
}

# Standardize and select correct variables
standardize.df = function(df) {
  # Gives a vector of integer column positions of numeric variables
  idx = sapply(df, is.numeric)
  idx = seq(1:length(idx))[idx]

  # Creating separate data set with standardized numeric variables for
  # regression, then reselect variables as described in paper (e.g. self-
  # reported health is removed)
  df.reg = df %>%
    mutate_at(.vars = vars(idx),
              .funs = standardize) %>%
    mutate(labor_participation = !labor_np) %>% # invert to get labor_pt
    dplyr::select(country, gender, age,
                  h_chronic, h_adla, h_obese, h_maxgrip,
                  edu_second, edu_high, children, couple,
                  labor_participation)

  return(df.reg)
}

```

```

# Split data frames into country/gender splits, then standardize numeric
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)

# Create necessary dummy variables for regression
dummify = function(df) {
  df = df %>%
    dplyr::select(-country, -gender)      # remove country/gender
  model = ~ 0 + .                        # needed to remove intercept
  new.df = model.matrix(model, df) # create dummies
  new.df = data.frame(new.df)
  return(new.df)
}

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

# df.out is for analysis, df.reg and df.splits are for estimation
all.output = list(df.out = df.out, df.reg = df.reg, df.splits = df.splits)

cat("Final output is a list containing 3 data.frames:", sep = "\n")
cat("`df.out` is a data.frame containing variables for calculating summary statistics.", sep = "\n")
cat("`df.reg` is a data.frame containing standardized variables for regression.", sep = "\n")
cat("`df.splits` splits the regression data.frame into individual data.frames for each country/
    gender split. 22 data.frames are contained in this list.", sep = "\n")

return(all.output)
}

```

10.1.2 Quantlet 2



10.1.3 Quantlet 3



10.1.4 Quantlet 4



```
#####
# SPL_ELP_LoadWald.R
#
#####
# Description:
# Contains a series of Wald tests
#
#####

#####
# Wald Test for Joint Significance

# Note: This a Wald test for the joint signifance of a subset of model
# coefficients
# m = number of restrictions
# Beta is vector of coefficients of size k x 1
# Confidence.level is the desired confidence level (between 0 and 1).
# Default: 0.95
# Spec is vector of integers of length 0 < m <= k specifying the subset of
# coefficients to be jointly tested

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

  # List all packages needed for session
  neededPackages = c("Matrix")
  allPackages     = c(neededPackages %in% installed.packages()[, "Package"])

  # Install packages (if not already installed)
  if (!all(allPackages)) {
    missingIDX = which(allPackages == FALSE)
    needed     = neededPackages[missingIDX]
    lapply(needed, install.packages)
  }

  # Load all defined packages
  invisible(lapply(neededPackages, function(x) suppressPackageStartupMessages(
    library(x, character.only = TRUE))))

  # Set up test restrictions
  if (is.null(spec)) {
    spec = 1:length(beta) # default joint is significance test
  } 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")
}

# Check Input
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!")
}

# Define test elements
joint.wald.test      = numeric(4)
names(joint.wald.test) = c("W", "p-value", "df", "Decision")
beta                 = model.summary$coefficients[,1]
Var_beta_est         = vcov(model.summary)

# Wald test statistic
W = t(beta[spec]) %*% solve(Var_beta_est[spec,spec]) %*% beta[spec]

# Set up test output
pval      = 1-pchisq(W,length(spec))
joint.wald.test[1] = format(W, digits = 4)
joint.wald.test[2] = format(pval, digits = 4)
joint.wald.test[3] = length(spec)
joint.wald.test[4] = ifelse(pval <= 1- confidence.level, "Reject H0",
                           "Cannot reject H0")

return(joint.wald.test)
}

#####
## Wald Test for Linear Hypothesis

# Note: This a general Wald test for linear hypothesis of model beta
# coefficients
# m = number of restrictions
# Beta is vector of coefficients of size k x 1
# Confidence level is the desired confidence level (between 0 and 1).

```

```

# Default: 0.95
# R is Jacobian matrix of size m x k
# r is restriction function size m x 1
# Example: H0: b1 + b2 = 1 and 2*b1 = 0 (2 restrictions, m = 2)
# R = 1 1 0 0 ... 0      and r = 1
#      2 0 0 0 ... 0      0

general.wald.test = function(model.summary,
                             confidence.level = 0.95,
                             R = NULL,
                             r = NULL) {

  # Check input
  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!")
  }

  # Define test elements
  general.wald.test = numeric(4)
  names(general.wald.test) = c("W", "p-value", "df", "Decision")
  beta = model.summary$coefficients[, 1]
  Var_beta_est = vcov(model.summary)

  # Set up restriction matrix/vector for linear hypothesis
  # default option is joint significance of all coefficients
  if (is.null(R)) {
    R = diag(1, length(beta)) # default of R is identity matrix
  }
  if (is.null(r)) {
    r = rep(0, length(beta)) # default for r is null vector
  }

  # Now R and r definitely exist, check dimensions
  dim_R = dim(R) # Testing R
  k = length(beta) # Columns
  m = length(r) # Rows; general hypothesis test, therefore m = length(r)

  ## Apply test
  if (dim_R[1] != m | dim_R[2] != k) stop("R has wrong dimension!")

```

```

# Check rank of R
if (rankMatrix(R)[1] != m) stop("R has wrong rank!")

# Wald test statistic
W = t(R%*%beta - r) %*% solve(R%*% Var_beta_est %*% t(R)) %*% (R%*%beta - r)

# Set up test output
pval = 1-pchisq(W,length(r))
general.wald.test[1] = format(W, digits = 4)
general.wald.test[2] = format(pval, digits = 4)
general.wald.test[3] = length(r)
general.wald.test[4] = ifelse(pval <= 1- confidence.level, "Reject H0", "Cannot reject H0")

return(general.wald.test)
}

```

10.1.5 Quantlet 5



```

#####
# Counterfactual
#
#####
# Description:
#
# Calculates current and counterfactual labor participation rates for different
# age groups, countries, and genders
#
#####

#####
# SOURCE DATA
load('easySHARE_clean.RData')

#Only keep relevant data sets
df.splits = datasets$df.splits
rm(datasets)

#####

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

```

```

    model = glm(z$labor_participationTRUE ~ . -age50,
                family = binomial(link = "probit"),
                data = z)
    return(model)
})

# Return summaries
allSummaries = lapply(allModels, summary)

#####
# Counterfactual Exercise

# Calculate employment rate based on whole population with mean probability

# Estimate current employment rate
empl.rate = function(model) {
  # Select data from model
  X = model$data
  # Predict probability of being employed of all individuals
  empl.probability = predict(object = model, newdata = X, type = "response")
  # Calculate predicted current employment rate
  empl.rate = sum(empl.probability) / length(empl.probability)
  return(empl.rate)
}

# Function for defining models with modified dataset (perfect health)
X.cf = function(model) {
  # Select data from model
  X = model$data
  # Create counterfactual data set with ideal health condition
  X_cf = X
  # Find ideal health conditions by finding the minimum value for each
  # variable
  X_min = data.frame(t(apply(X, 2, min)))
  # Replace mean values by values representing no health issues
  names.vec = c("h_chronic", "h_adlaTRUE", "h_obeseTRUE")
  X_cf[, names.vec] = X_min[names.vec]
  # Create index vector for 50 and 51 year olds
  age_50_51 = ifelse(X$age50 == 1 | X$age51 == 1, 1, 0)
  # Calculate max grip mean of individuals age 50-51 and replace
  X_cf[, c("h_maxgrip")] = tapply(X$h_maxgrip, age_50_51 == 1, mean)[[2]]
  model$data = X_cf
  return(model)
}

```

```

}

# Returns a list with all models with modified datasets
allModels.cf = lapply(allModels, X.cf)

# Calculate current and counterfactual employment rates
empl.current      = lapply(allModels, empl.rate)
empl.counterfact  = lapply(allModels.cf, empl.rate)

# Store results and display them
employment        = data.frame(cbind(unlist(empl.current),
                                     unlist(empl.counterfact)))
colnames(employment) = c("Employment Current" , "Employment Counterf.")

#####
# Counterfactual exercise for each age group

### Function for current and counterfactual employment rate among age groups
# Default arguments: group size and lower bound of age groups (group.low)

# Estimate current employment rate for age groups
empl.rate.age = function(model, group.size = 5, group.low = c(50,55,60)) {

  #Initialize storage of results
  empl.rate      = numeric(3)

  # Select data from model
  X              = model$data

  # Predict probability of being employed of all individuals
  empl.probability = predict(object = model, newdata = X, type = "response")

  # Create age group Index vectors
  # Initialize result list containing age group index vectors
  IND_row_vec = list()

  # Define vector of relevant age groups by youngest age
  for (i in group.low) {
    # Define upper bound k of age group
    k      = i + group.size -1
    # Create labels and contents of each age group
    vec.label = paste("names.vec.age", i, sep=".")
    # Assign ages to the age groups

```

```

z      = assign(vec.label, paste0("age",i:k))
# For age groups of size 1, use simple indexing (cannot use apply)
if (length(z) == 1) {
  IND_row_vec[[i]] = X[, z]

  # For all age groups greater than size 1
} else {
  # Sum over rows to find individuals belonging to age group i
  IND_row_vec[[i]] = apply(X[, z], 1, sum)
}
}

# Extract list of relevant age groups, which are each stored in a list
IND_row_vec = IND_row_vec[group.low]

# Calculate predicted current employment rate for age groups
# loop over age group list
for (k in 1: length(IND_row_vec)) {
  empl.rate[k] = empl.probability %*% IND_row_vec[[k]] /
    sum(IND_row_vec[[k]])
}
return(empl.rate)
}

# Calculate current and counterfactual employment rates for age groups
empl.current.age = t(data.frame(lapply(allModels, empl.rate.age)))
empl.counterfact.age = t(data.frame(lapply(allModels.cf, empl.rate.age)))

employment.age = data.frame(cbind(empl.current.age,
                                   empl.counterfact.age))
colnames(employment.age) = paste0(c(rep("empl.current", 3),
                                     rep("empl.counterfact", 3)),
                                   c("50.54", "55.59", "60.64"))

# Test for other age groups: Example of employment for each age group
empl.rate.age(allModels$Germany.MALE, group.size = 1, group.low = c(50:63))
empl.rate.age(allModels.cf$Germany.MALE, group.size = 1, group.low = c(50:63))

#####

# Decline in participation due to decline in health condition
health.decline = function(X) {
  # Differences between counterf. and current participation rates

```

```

dif.60.64 = X[, "empl.counterfact60.64"] - X[, "empl.current60.64"]
dif.50.54 = X[, "empl.counterfact50.54"] - X[, "empl.current50.54"]

# Absolute difference in current participation rate
dif.abs = X[, "empl.current50.54"] - X[, "empl.current60.64"]
measure = (dif.60.64 - dif.50.54) / dif.abs
return(measure)
}

participation.health = data.frame(health.decline(employment.age))

#####
# Some Remarks

# Define outout formate of counterfactual estimates: list
empl.each.age = list()

# Calculate employment for each age for all countries
for (i in 1:length(allModels)) {
  k = allModels[[i]]
  empl.each.age[[i]] = empl.rate.age(k, group.size = 1, group.low = c(50:64))
}

names(empl.each.age) = names(allModels)

```

10.1.6 Quantlet 6



```

#####
# SPL_ELP_Graphics.R
#
#####
# Description:
# Functions designed to create useful graphics (quantlet 6)
# Note that these functions are designed to specifically run with the 'df.out'
# data frame in the datasets lists found in `easySHARE_clean.RData`
#
# health.gridmap(xvar, facetting): generate a tile grid map by a variable in
# dataset, faceted by either gender or age. if numeric than average is
# calculated. if logical than % is calculated. both xvar and facetting should be
# character vectors. returns a grid object, which can be printed to device with # grid.draw() or saved u
#

```

```

# health.distribution(xvar, gen, countries, remove.outliers): show distribution
# of numeric variable in dataset, facетted by gender and country.
# Select a numeric variable in xvar (as a string), a vector of genders, and a
# vector of countries. By default, outliers are removed.
#
#####

##### GEOGRAPHIC VISUALIZATION OF NUMERIC DATA
# Tile grid map of numeric variables by geographic location

# accept metric and facетting as input. xvar is a character vector
# facетting can take either 'gender' or 'age' as inputs
# return error if neither 'gender' or 'age' is inputted

# output side by side grid maps
health.gridmap = function(xvar, facетting) {

  # List all packages needed for session
  neededPackages = c("dplyr", "ggplot2", "countrycode",
                     "gridExtra", "grid", "infuser")
  allPackages     = c(neededPackages %in% installed.packages()[ , "Package"])

  # Install packages (if not already installed)
  if(!all(allPackages)) {
    missingIDX = which(allPackages == FALSE)
    needed     = neededPackages[missingIDX]
    lapply(needed, install.packages)
  }

  # Load all defined packages (silently)
  invisible(lapply(neededPackages, library, character.only = TRUE))

  # first convert age to numeric, not factor variable
  if (is.factor(df.out$age)) {
    df.out$age = as.numeric(levels(df.out$age)[df.out$age])
  }

  # STOPPING CONDITIONS
  if (!is.numeric(df.out[[xvar]]) & !is.logical(df.out[[xvar]])) {
    stop("'xvar' must be numeric
        or logical")
  }
  if (!(facетting %in% c('gender', 'age'))){

```



```

    stop("'facetting' variable
    must be either 'gender' or 'age'")
}
# ADD TILE GRID LAYOUT
coordinates = structure(list(
  Country = c("Albania", "Austria", "Belarus", "Belgium",
    "Bosnia and Herzegovina", "Bulgaria", "Croatia",
    "Czech Republic", "Denmark", "Estonia", "Finland", "France",
    "Germany", "Greece", "Hungary", "Iceland", "Ireland",
    "Italy", "Latvia", "Lithuania", "Luxembourg", "Malta",
    "Montenegro", "Netherlands", "Norway", "Poland", "Portugal",
    "Republic of Moldova", "Romania", "Russian Federation",
    "Serbia", "Slovakia", "Slovenia", "Spain", "Sweden",
    "Switzerland", "The former Yugoslav Republic of Macedonia",
    "Ukraine",
    "United Kingdom of Great Britain and Northern Ireland"
  ),
  X = c(6, 5, 7, 3, 6, 8, 5, 5, 5, 7, 7, 2, 5, 7, 6, 1, 1, 4, 8, 7, 3, 3,
    6, 4, 5, 6, 1, 8, 7, 9, 7, 6, 4, 2, 6, 4, 8, 7, 2
  ),
  Y = c(1, 4, 6, 6, 3, 3, 3, 5, 7, 8, 9, 5, 6, 1, 4, 9, 7, 4, 7, 7, 5, 3,
    2, 6, 9, 6, 4, 4, 4, 7, 3, 5, 3, 4, 9, 5, 2, 5, 7
  ),
  InSet = c(FALSE, TRUE, FALSE, TRUE, FALSE, FALSE, FALSE, FALSE, TRUE,
    FALSE, FALSE, TRUE, TRUE, TRUE, FALSE, FALSE, FALSE, TRUE,
    FALSE, FALSE, FALSE, FALSE, FALSE, TRUE, FALSE, FALSE, FALSE,
    FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, TRUE, TRUE, TRUE,
    FALSE, FALSE, FALSE
  )
),
.Names = c("Country", "X", "Y", "InSet"),
row.names = c(NA, -39L),
class = c("tbl_df", "tbl", "data.frame")
)

# Get correct country information
codes      = countrycode_data
df.codes   = left_join(coordinates, codes,
                        by = c("Country" = "country.name.en")) %>%
  dplyr::select(Country, iso3c, X, Y)

# plot function to be used later
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)
}

# IF GENDER SELECTED:
if (facetting == 'gender') {

  # if logical than calculate percentage
  if (is.logical(df.out[[xvar]])) {

    var.type = '% with'

    df.graph = df.out %>%
      dplyr::select(country, gender, `xvar`) %>%
      group_by(country, gender) %>%
      summarize(value = sum(get(xvar))/length(get(xvar)))
    names(df.graph)[3] = xvar

    # split by gender
    df.m = df.graph %>%
      filter(gender == "MALE")
    df.f = df.graph %>%
      filter(gender == "FEMALE")
    df.m = suppressWarnings(
      left_join(df.codes, df.m, c('Country' = 'country')) %>%
      dplyr::select(iso3c, X, Y, `xvar`)

```

```

df.f = suppressWarnings(
  left_join(df.codes, df.f, c('Country' = 'country')) %>%
  dplyr::select(iso3c, X, Y, `xvar`)
}

# if numeric then calculate average
if (is.numeric(df.out[[xvar]])) {

  var.type = 'Avg.'

  df.graph = df.out %>%
    dplyr::select(country, gender, `xvar`) %>%
    group_by(country, gender) %>%
    summarize(value = mean(get(xvar)))
  names(df.graph)[3] = xvar

  # split by gender
  df.m = df.graph %>%
    filter(gender == "MALE")
  df.f = df.graph %>%
    filter(gender == "FEMALE")
  df.m = suppressWarnings(
    left_join(df.codes, df.m, c('Country' = 'country')) %>%
    dplyr::select(iso3c, X, Y, `xvar`)
  df.f = suppressWarnings(
    left_join(df.codes, df.f, c('Country' = 'country')) %>%
    dplyr::select(iso3c, X, Y, `xvar`)
}

# set equal range for color scales
minval = min(df.f[[xvar]], df.m[[xvar]], na.rm = TRUE)
maxval = max(df.f[[xvar]], df.m[[xvar]], na.rm = TRUE)

# PLOT
output = lapply(list(df.m, df.f), plot_function)
output[[1]] = output[[1]] +
  labs(title = infuse("#{intro} {metric} for {split}",
    intro = var.type, metric = xvar,
    split = 'male'),
    fill = xvar) +
  scale_fill_continuous(low = "#fdd0a2", high = "#8c2d04",
    na.value = "#CCCCCC",
    limits = c(minval, maxval))

```

```

output[[2]] = output[[2]] +
  labs(title = infuse("{{intro}} {{metric}} for {{split}}",
    intro = var.type, metric = xvar,
    split = 'female'),
    fill = xvar) +
  scale_fill_continuous(low = "#fdd0a2", high = "#8c2d04",
    na.value = "#CCCCCC",
    limits = c(minval, maxval))
return(arrangeGrob(output[[1]], output[[2]], ncol = 2))
}

# IF AGE SELECTED:
if (facetting == 'age') {

  # if logical than calculate percentage
  if (is.logical(df.out[[xvar]])) {

    var.type = '% with'

    # split by age group
    df50 = df.out %>%
      filter(age50_54 == TRUE) %>%
      group_by(country) %>%
      summarize(value = sum(get(xvar))/length(get(xvar)))
    df50 = suppressWarnings(
      right_join(df50, df.codes, c('country' = 'Country'))) %>%
      dplyr::select(iso3c, X, Y, value)
    names(df50)[4] = xvar

    df55 = df.out %>%
      filter(age55_59 == TRUE) %>%
      group_by(country) %>%
      summarize(value = sum(get(xvar))/length(get(xvar)))
    df55 = suppressWarnings(
      right_join(df55, df.codes, c('country' = 'Country'))) %>%
      dplyr::select(iso3c, X, Y, value)
    names(df55)[4] = xvar

    df60 = df.out %>%
      filter(age60_64 == TRUE) %>%
      group_by(country) %>%
      summarize(value = sum(get(xvar))/length(get(xvar)))
    df60 = suppressWarnings(

```

```

    right_join(df60, df.codes, c('country' = 'Country')) %>%
    dplyr::select(iso3c, X, Y, value)
    names(df60)[4] = xvar
  }

  # if numeric than calculate average
  if (is.numeric(df.out[[xvar]])) {

    var.type = 'Avg.'

    # split by age group
    df50 = df.out %>%
      filter(age50_54 == TRUE) %>%
      group_by(country) %>%
      summarize(value = mean(get(xvar)))
    df50 = suppressWarnings(
      right_join(df50, df.codes, c('country' = 'Country')) %>%
      dplyr::select(iso3c, X, Y, value)
    names(df50)[4] = xvar

    df55 = df.out %>%
      filter(age55_59 == TRUE) %>%
      group_by(country) %>%
      summarize(value = mean(get(xvar)))
    df55 = suppressWarnings(
      right_join(df55, df.codes, c('country' = 'Country')) %>%
      dplyr::select(iso3c, X, Y, value)
    names(df55)[4] = xvar

    df60 = df.out %>%
      filter(age60_64 == TRUE) %>%
      group_by(country) %>%
      summarize(value = mean(get(xvar)))
    df60 = suppressWarnings(
      right_join(df60, df.codes, c('country' = 'Country')) %>%
      dplyr::select(iso3c, X, Y, value)
    names(df60)[4] = xvar
  }

  # set equal range for color scales
  minval = min(df50[[xvar]], df55[[xvar]], df60[[xvar]], na.rm = TRUE)
  maxval = max(df50[[xvar]], df55[[xvar]], df60[[xvar]], na.rm = TRUE)

```

```

# PLOT
output = lapply(list(df50, df55, df60), plot_function)
output[[1]] = output[[1]] +
  labs(title = infuse("{{intro}} {{metric}} for {{split}}",
    intro = var.type, metric = xvar,
    split = 'age50_54'),
    fill = xvar) +
  scale_fill_continuous(low = "#fdd0a2", high = "#8c2d04",
    na.value = "#CCCCCC",
    limits = c(minval, maxval))
output[[2]] = output[[2]] +
  labs(title = infuse("{{intro}} {{metric}} for {{split}}",
    intro = var.type, metric = xvar,
    split = 'age55_59'),
    fill = xvar) +
  scale_fill_continuous(low = "#fdd0a2", high = "#8c2d04",
    na.value = "#CCCCCC",
    limits = c(minval, maxval))
output[[3]] = output[[3]] +
  labs(title = infuse("{{intro}} {{metric}} for {{split}}",
    intro = var.type, metric = xvar,
    split = 'age60_64'),
    fill = xvar) +
  scale_fill_continuous(low = "#fdd0a2", high = "#8c2d04",
    na.value = "#CCCCCC",
    limits = c(minval, maxval))

return(arrangeGrob(output[[1]], output[[2]], output[[3]], ncol = 3))

}
}

##### VIEW DISTRIBUTION OF NUMERIC VARIABLES
# Numeric variable: shows distribution for each country, then overlays average of entire data set

# slicing by dummy variables is not included. that can be done before the function is called and then t

# xvar is a character vector
# by default, gender is "all". can either be "all", "MALE" or "FEMALE"
# by default, countries is "all". can also pass a character vector of countries

health.distribution = function(xvar,
                                gen = "all",

```

```

        countries      = "all",
        remove.outliers = TRUE) {

  # List all packages needed for session
  neededPackages = c("dplyr", "ggplot2", "scales")
  allPackages    = c(neededPackages %in% installed.packages()[ , "Package"])

  # Install packages (if not already installed)
  if(!all(allPackages)) {
    missingIDX = which(allPackages == FALSE)
    needed     = neededPackages[missingIDX]
    lapply(needed, install.packages)
  }

  # Load all defined packages (silently)
  suppressMessages(lapply(neededPackages, library, character.only = TRUE))

  # first convert age to numeric, not factor variable
  if (is.factor(df.out$age)) {
    df.out$age = as.numeric(levels(df.out$age)[df.out$age])
  }

  # STOPPING CONDITIONS
  if (!is.numeric(df.out[[xvar]])) stop("'xvar'
                                     must be numeric")
  if (length(gen) > 1 | !all(gen %in% c("all", "MALE", "FEMALE"))) {
    stop("'gender' must be one of
         'all', 'MALE', or 'FEMALE'")
  }
  if (length(countries) > 1 & !all(countries %in% levels(df.out$country))) {
    stop("'countries'
         must be 'all' or a character vector containing countries
         in the `df.out` data frame")
  }
  if (length(countries) == 1) {
    if (!(countries %in% c('all', levels(df.out$country)))) {
      stop("'countries'
           must be 'all' or a character vector containing countries
           in the `df.out` data frame")
    }
  }

  # Set faceting variables

```

```

if (gen == "all") gen = c("MALE", "FEMALE")
if (length(countries) == 1) {
  if (countries == "all") countries = levels(df.out$country)
}

# Filter countries/genders as selected
df.out = df.out %>%
  filter(country %in% countries, gender %in% gen)

# 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)

# Overlay lines for average over entire data set
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))

plot.bar = ggplot(data = df.out, aes(x = get(xvar))) +

  geom_bar(aes(fill = country,
               color = country,
               y      = ..prop..),
           alpha = 0.4) +

  geom_errorbar(data = total, aes(x      = get(xvar),
                                ymin = value,
                                ymax = value)) +

```



```

theme_minimal() +
theme(legend.position = "none") +
theme(panel.grid.minor = element_blank()) +
theme(panel.grid.major = element_blank()) +
theme(axis.text.y = element_blank()) +

labs(title = paste0("Distribution of ", xvar, " by Country")) +
labs(subtitle = "Faceted by country & gender. Distribution of the entire data set shown is overlaid") +
labs(x = xvar) +
labs(y = "") +

theme(plot.title = element_text(size=16)) +
theme(plot.subtitle = element_text(size=10, color = "#7F7F7F"))

# show correct faceting and titles
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")) +
    facet_wrap(~ country, ncol = 4)
} else {
  plot.bar = plot.bar +
    facet_grid(gender ~ country)
}

return(plot.bar)
}

```

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, 2018-03-16

Claudia Günther, Phi Nguyen, Julian Winkel