Program „EFECTAS“

April 21, 2020

Version 1.1

Table of Contents

1. Description

This program is prepared on the basis of three open source R program packages BIFIEsurvey, EdSurvey and intsvy.

This program is designed to work with PISA 2015 data. EFECTAS opportunities:

- Download PISA data from the OECD website
- Data adaptation for BIFIEsurvey, EdSurvey and intsvy packages
- Presentation of descriptive statistics for categorical and continuous variables.

- Pearson and Spearman correlations
- Linear and logostic regressions
- Multilevel analysis
- Data visualization

2. Data downloading

This function is taken from EdSurvey package. Uses an Internet connection to download PISA data to a computer. Data come from the OECD website.

Function usage

`downloadPISA(root,years = c(2000, 2003, 2006, 2009, 2012, 2015))`

`root`	a character string indicating the directory where the PISA data should be stored. Files are placed in a folder named PISA/[year]. For Windows, the path is written "C:/Users/ ". For Mac, the path is written "/Users/".
`years`	an integer vector of the assessment years to download. Valid years are 2000, 2003, 2006, 2009, 2012, and 2015. Program EFECTAS is designed for 2015 data analysis.

Code example

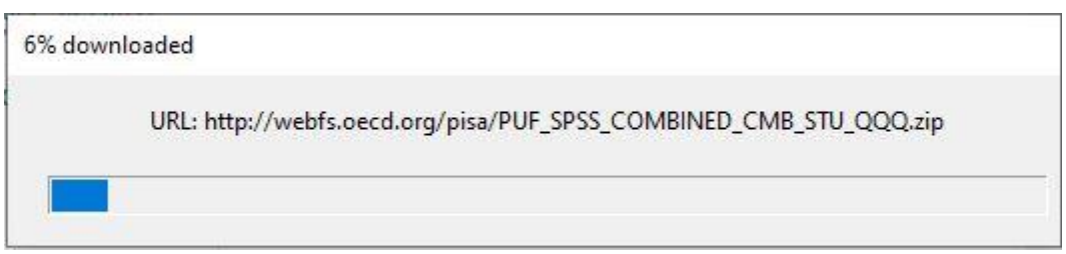
```
# download PISA 2015 data (International Database only)
myroot <- "C:/Users/User/ " #write your own path
year <- 2015
downloadPISA(myroot, year)
```

Function result

Myroot and year are values given to `downloadPISA` function. The `downloadPISA` function will output the message in the console window.

```
Processing PISA data for year 2015
Database INT
trying URL 'http://webfs.oecd.org/pisa/PUF_SPSS_COMBINED_CMB_STU_QQQ.zip'
Content type 'application/x-zip-compressed' length 440232149 bytes (419.8 MB)
```

Also a data download table will appear with a note of the download progress.



The archived data is downloaded to the computer. The data is extracted when it is downloaded.

The result of the function is the 2015 PISA data in the PISA / 2015 directory.

3. Data adaptation

In the downloaded data, the student and school databases are separate. Databases are interconnected before statistical analysis. It is not recommended to use all the data in the database for statistical analysis. It is recommended to select the analysed countries and variables that are needed for the specific analysis. This function is used to select and prepare the required data for BIFIEsurvey, EdSurvey and intsvy packages.

Function usage

`form_data (path_root, mycountry, myvariables)`

`path_root`	a character string indicating the directory where the PISA data is stored. For Windows, the path is written "C:/Users/ ". For Mac, the path is written "/Users/".
`mycountry`	a character vector of the country/countries to include using the three-digit ISO country code. A list of country codes can be found in the PISA codebook or https://en.wikipedia.org/wiki/ISO_3166-1#Current_codes. If you want to use all countires write "all", but it is not recommended.
`myvariables`	a character vector of the variables to be included in the data. The names of the variables are written in lower case. Country code, student ID, school ID, weights and replicate weight are default in the data. There is no need to write all plausible values names (e.g. "pv1math", "pv2math"), it is enough to write a common name (e.g. "math") and all plausible values will be assigned to the data. Common names for plausible values in PISA 2015 data: "math", "read", "scie", "scep", "sced", "scid", "skco", "skpe", "ssph", "ssli", "sses", "flit", "clps".

Code example

```
myroot <- "C:/Users/User/ " #write your own path
mycountries <- c("LTU")
myvariables <- c("math", "read", "scie","escs", "st011q12ta", "st004d01t")
mydata <- form_data(myroot, mycountries, myvariables)
```

Function result

The data is provided in the data.frame.

The data column names are displayed with the function `colnames(mydata)`.

Brief information about each column (min, max, median, mean, 1st and 3rd Quantiles and NA's) are displayed with the function `summary(mydata)` .

4. Descriptive statistic

There are four functions in descriptive statistic: `frequency_table`, `descriptive`, `tile` and `ben_marks`.

4.1. Frequency table

This function displays frequency table, number of NA and missing values, percentage of categorical variables. Also this function can display number of unique entries of all variables. The function can calculate frequencies for several variables at the same time, except the number of missing values calculate for one variable.

Function usage

```
frequency_table(mydata, myvariable, variables, group = NULL,
missing_values = FALSE, unikalus = FALSE)
```

mydata	Data.frame formed with form_data function
myvariable	a character vector of the variables to be included in the data. The names of the variables are written in lower case. Country code, student ID, school ID, weights and replicate weight are default in the data. There is no need to write all plausible values names (e.g. “pv1math”, “pv2math”), it is enough to write a common name (e.g. “math”) and all plausible values will be assigned to the data. Common names for plausible values in PISA 2015 data: "math", "read", "scie", "scep", "sced", "scid", "skco", "skpe", "ssph", "ssli", "sses", "flit", "clps".
variables	a character vector of the variables for which the frequency table is calculating
group	Optional grouping variable(s). Default is NULL
missing_values	It is system missing data. Default value FALSE
uniq	Logical expression. Default FALSE. When a TRUE value is obtained, the function outputs a frequency table that shows how many unique records the variable has.

1. Code example – frequency table for several variables.

```
myvariables <- c("math", "read", "scie","escs", "st011q12ta", "st004d01t",
"st034q02ta", "sc012q01ta")
dsc_variables1 <-  c("st011q12ta", "st004d01t", "st034q02ta", "sc012q01ta")
table1 <- frequency_table(mydata, myvariables, dsc_variables1)
table1
```

Function result

The result gives a table with calculated frequencies of variables. The name of a single variable in the table is repeated as many times as it has unique records. For example, variable st011q12ta (In your home: A dictionary) has three unique records (1 answer Yes, 2 – No and 9 – NA). The unique record value show in varval column. Ncases column is frequency of records. Nweiht column is sum of weight. Perc column is percentage.

	var	varval	Ncases	Nweight	perc
1	st011q12ta	1	5575	25403.9518	86.6153995
2	st011q12ta	2	679	3282.9456	11.1932838
3	st011q12ta	9	136	642.7045	2.1913167
4	st004d01t	1	3201	14733.0953	49.2505258
5	st004d01t	2	3324	15181.4996	50.7494742
6	st034q02ta	1	1628	7987.3580	27.2499860
7	st034q02ta	2	2463	10507.9828	35.8494492
8	st034q02ta	3	1305	5874.5862	20.0419703
9	st034q02ta	4	866	4314.7507	14.7203741
10	st034q02ta	9	125	626.7428	2.1382204
11	sc012q01ta	1	2862	11942.0085	39.9203418
12	sc012q01ta	2	2065	9756.2490	32.6136759
13	sc012q01ta	3	1566	8058.5540	26.9385363
14	sc012q01ta	9	32	157.7833	0.5274459

2. Code example – frequency table for several grouped variables

```
myvariables <- c("math", "read", "scie","escs", "st011q12ta", "st004d01t",
"st034q02ta", "sc012q01ta")
dsc_variables1 <-  c("st011q12ta", "st034q02ta")
dsc_group <- c("st004d01t")
table2 <- frequency_table(mydata, myvariables, dsc_variables1, mygroup =
dsc_group)
table2
```

Function result

The result gives a table with calculated frequencies of grouped variables. In the table, the name of a single variable is repeated several times due to the number of unique entries and grouping. Varval column is unique record value. Groupvar column is grouping variable name. Groupval is unique group record value. Ncases column is frequency of records. Nweiht column is sum of weight. Perc column is percentage.

	var	varval	groupvar	groupval	Ncases	Nweight	perc
1	st011q12ta	1	st004d01t	1	2832	12969.9671	89.787449
2	st011q12ta	2	st004d01t	1	265	1295.1025	8.965632
3	st011q12ta	9	st004d01t	1	35	180.1198	1.246919
4	st011q12ta	1	st004d01t	2	2743	12433.9847	83.536954
5	st011q12ta	2	st004d01t	2	414	1987.8431	13.355200
6	st011q12ta	9	st004d01t	2	101	462.5847	3.107846
7	st034q02ta	1	st004d01t	1	714	3546.3988	24.581665
8	st034q02ta	2	st004d01t	1	1328	5776.2533	40.037777
9	st034q02ta	3	st004d01t	1	698	3160.3058	21.905483
10	st034q02ta	4	st004d01t	1	361	1787.6136	12.390744
11	st034q02ta	9	st004d01t	1	28	156.4366	1.084332
12	st034q02ta	1	st004d01t	2	914	4440.9592	29.836308
13	st034q02ta	2	st004d01t	2	1135	4731.7295	31.789831
14	st034q02ta	3	st004d01t	2	607	2714.2804	18.235724
15	st034q02ta	4	st004d01t	2	505	2527.1372	16.978414
16	st034q02ta	9	st004d01t	2	97	470.3061	3.159723

3. Code example – unique entries of variables

```
myvariables <- c("math", "read", "scie","escs", "st011q12ta", "st004d01t",
"st034q02ta", "sc012q01ta")
```

```
dsc_variables1 <-  c("st011q12ta", "st034q02ta", " escs")
dsc_group <- c("st004d01t")
unique = TRUE
table3 <- frequency_table(mydata, myvariables, dsc_variables1, mygroup =
dsc_group, uniq = unique)
table3
```

Function result

The result gives a table with numbers of unique records of grouped variables. In the table, the name of a single variable is repeated several times due to the grouping. parm column is variable name. Groupvar column is grouping variable name. Groupval is unique group record value. Ncases column is frequency of records. Nweiht column is sum of weight. Est column is number of unique value.

	parm	groupvar	groupval	Ncases	Nweight	est	fmi	VarMI
1	st011q12ta	st004d01t	1	3201	14733.1	4	0	0
2	st034q02ta	st004d01t	1	3201	14733.1	6	0	0
3	escs	st004d01t	1	3201	14733.1	2946	0	0
4	st011q12ta	st004d01t	2	3324	15181.5	4	0	0
5	st034q02ta	st004d01t	2	3324	15181.5	6	0	0
6	escs	st004d01t	2	3324	15181.5	3059	0	0

4. Code example – frequescy of missing values

```
#  only for one variable!!!
myvariables <- c("math", "read", "scie","escs", "st011q12ta", "st004d01t",
"st034q02ta", "sc012q01ta")
dsc_variables1 <-  c("st011q12ta")
missing_values = TRUE
table4 <- frequency_table(mydata, myvariables, dsc_variables1,
missing_values = missing_values)
table4
```

Function result

The result gives a frequency table. The name of variable is in the first column name. N is number of records. The sum of weight is in the third column. Percent column is the value of percentage.

Estimates are weighted using weight variable 'w_fstuwt'

	st011q12ta	N	Weighted N	Weighted Percent	Weighted Percent SE
1	(Missing)	135	584.9930	1.955544	0.2177754
2	YES	5575	25403.9518	84.921597	0.6022620
3	NO	679	3282.9456	10.974394	0.5129865
4	NO RESPONSE	136	642.7045	2.148465	0.2078285

4.2. Descriptive

This function displays minimal and maximum values, average, standard deviation, median, 1st and 3rd quantile for continuous variables. The function can calculate descriptive statistic for several variables at the same time. The total descriptive statistic for all plausible values are calculated by giving the common name of the plausible values for function. Common names for plausible values in PISA 2015 data: "math", "read", "scie", "scep", "sced", "scid", "skco", "skpe", "ssph", "ssli", "sses", "flit", "clps".

Function usage

`descriptive(variables)`

`variables`	a character vector of the variables for which the frequency table is calculating

Code example

```
dsc_variables2 <- c("escs", "pv1scie", "pv2scie", "pv3scie", "pv4scie",
"pv5scie", "pv6scie", "pv7scie", "pv8scie",
                       "pv9scie", "pv10scie", "scie")
table5 <- descriptive(dsc_variables2)
table5
```

Function result

The result gives a descriptive statistic for continuous variables. Variable column is name of the variable the row regards. N column is total number of cases (both valid and invalid cases). Weighted N column is the sum of weights. Min. column is smallest value of the variable. 1st Qu. column is first quantile of the variable. Median column is median value of the variable. Mean column is mean of the variable. 3rd Qu. column is third quantile of the variable. Max. column is largest value of the variable. SD column is standard deviation or weighted standard deviation. NA’s column is number ofNAin variable and in weight variables. Zero-weights column is number of zero-weight cases if users choose to produce weighted statistics. (Bailey et al., 2019)

Estimates are weighted using weight variable ‘w_fstuwt’

	Variable	N	Weighted N	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	SD	NA's	Zero-weights
1	escs	6525	29914.59	-4.0459	-0.7808629	0.02421431	-0.06461623	0.6699	3.3762	0.8679327	191	0
2	pv1scie	6525	29914.59	201.9440	410.8310476	473.32192831	475.72596098	539.4137	781.4020	90.1687247	0	0
3	pv2scie	6525	29914.59	154.0540	410.6011133	473.94889000	475.77189091	541.3776	772.9270	91.4804085	0	0
4	pv3scie	6525	29914.59	122.7140	409.8895578	473.44563925	475.71041635	539.9696	755.2230	90.7686350	0	0
5	pv4scie	6525	29914.59	174.7500	412.5065545	473.26889474	476.01198712	539.8733	791.2340	90.1361860	0	0
6	pv5scie	6525	29914.59	159.2560	411.1250294	473.83107098	475.67308325	538.8728	744.1510	90.7502637	0	0
7	pv6scie	6525	29914.59	147.3920	410.2115856	473.19108925	474.99724337	540.7224	758.8670	91.0029834	0	0
8	pv7scie	6525	29914.59	124.6800	408.4588370	472.29833559	473.93314134	539.0653	786.8990	91.3785024	0	0
9	pv8scie	6525	29914.59	156.0520	409.5562078	472.25031280	475.37765897	540.6257	787.4750	91.3030057	0	0
10	pv9scie	6525	29914.59	103.8590	408.7914974	473.39220705	475.10878158	540.1935	768.2880	90.8651762	0	0
11	pv10scie	6525	29914.59	187.9950	410.7171742	473.58724834	475.77930169	540.7203	744.4320	91.4098430	0	0
12	scie	6525	29914.59	153.2696	410.2688605	473.25356163	475.40894656	540.0834	769.0898	90.9275556	0	0

4.3. Percentile

Calculates the percentiles of a numeric variable. The percentiles can be calculated only for one variable at the same time.

Function usage

`tile(variables, percent)`

`variables`	the character name of the variable to percentiles computed, typically a subjectscale or subscale
`percent`	a numeric vector of percentiles in the range of 0 to 100 (inclusive)

Code example

```
prc_variables <- "escs"
percent <- c(5, 25, 50, 75, 95)
percentiles <- tile(prc_variables, percent)
percentiles
```

Function result

The result gives a table. percentile column is the percentile of this row. estimate column is the estimated value of the percentile. ee column is the jackknife standard error of the estimated percentile. df column is degrees of freedom. confInt.ci_lower column is the lower bound of the confidence interval. confInt.ci_upper column is the upper bound of the confidence interval. nsmall column is the number of units with more extreme results, averaged across plausible values. (Bailey et al., 2019)

```
Percentile
Call: percentile(variable = variables, percentiles = percent, data = Pisa.data,
    weightVar = "w_fstuwt")
full data n: 6525
n used: 6334

 percentile    estimate         se       df confInt.ci_lower confInt.ci_upper nsmall
          5 -1.43910118 0.02301012 21.08789       -1.7188636       -1.3142000    327
         25 -0.78086288 0.03868594 21.15925       -1.0358000       -0.5099690   1600
         50  0.02421431 0.04347068 31.09775       -0.3308272        0.3452626   3105
         75  0.66990000 0.01804455 24.00991        0.4492029        0.8688038   1556
         95  1.14122770 0.02074189 32.36995        1.0071749        1.4130047    317
```

4.4. Benchmarks

Calculates percentage of students at each proficiency level defined by PISA. Or at proficiency levels provided by the useR (Caro and Biecek, 2019). The benchmarks can be calculated only for one variable. The variable is common plausible values name. Common names for plausible values in PISA 2015 data: "math", "read", "scie", "scep", "sced", "scid", "skco", "skpe", "ssph", "ssli", "sses", "flit", "clps". For variables that do not have plausible values, benchmarks is not available.

Function usage

```
ben_marks(mydata, variable, bench)
```

`mydata`	Data.frame formed with form_data function
`variable`	A common plausible value name. Common names for plausible values in PISA 2015 data: "math", "read", "scie", "scep", "sced", "scid", "skco", "skpe", "ssph", "ssli", "sses", "flit", "clps".
`bench`	The cut-off points for the assessment benchmarks (e.g., cutoff= c(357.77, 420.07,482.38, 544.68, 606.99, 669.30)).

Code example

```
bn_variable <- "scie"
bench <- c(357.77, 420.07,482.38, 544.68, 606.99, 669.30)
marks <- ben_marks(mydata, bn_variable, bench)
marks
```

Function result

The result gives a table. CNT column is name of country. Benchmarks column is cut-off points specify by you. Percentage column is percentage of students at each proficiency level. Std. err. column is value of standard error.

```
        CNT       Benchmarks Percentage Std. err.
1 LITHUANIA         <= 357.77      10.10      0.76
2 LITHUANIA (357.77, 420.07]      18.52      0.73
3 LITHUANIA (420.07, 482.38]      25.06      0.80
4 LITHUANIA (482.38, 544.68]      22.92      0.70
5 LITHUANIA (544.68, 606.99]      15.32      0.73
6 LITHUANIA  (606.99, 669.3]       6.74      0.57
7 LITHUANIA          > 669.3       1.35      0.29
```

5. Correlation

Calculate Pearson and Spearman correlation coefficients. Pearson correlation can be calculated between several continuous variables and for all plausible values named with common name. Common names for plausible values in PISA 2015 data: "math", "read", "scie", "scep", "sced", "scid", "skco", "skpe", "ssph", "ssli", "sses", "flit", "clps". Spearman correlation can be calculated between two continuous variables. Cannot be calculated for all plausible values.

Function usage

```
correlation(mydata, myvariables, variables, group = NULL, method =
c("Pearson",”Spearman”)
```

`mydata`	Data.frame formed with form_data function
`myvariables`	a character vector of the variables to be included in the data. The names of the variables are written in lower case. Country code, student ID, school ID, weights and replicate weight are default in the data. There is no need to write all plausible values names (e.g. “pv1math”, “pv2math”), it is enough to write a common name (e.g. “math”) and all plausible values will be assigned to the data. Common names for plausible values in PISA 2015 data: "math", "read", "scie", "scep", "sced", "scid", "skco", "skpe", "ssph", "ssli", "sses", "flit", "clps".
`variables`	a character vector of the variables for which the correlation is calculating
`group`	Optional grouping variable(s). Default is NULL
`method`	a character string indicating which correlation coefficient (or covariance) is to be computed. One of Pearson (default) or Spearman.

1. Code example – Pearson correlation between several variables

```
myvariables <- c("math", "read", "scie","escs", "st011q12ta", "st004d01t",
"st034q02ta", "sc012q01ta")
cor_variables <- c("escs","pv1scie", "pv2scie")
cor_coef <- correlation(mydata, myvariables, cor_variables)
cor_coef
```

Function result

Outputs four tables: correlation statistic, correlation matrix, covariance statistic and covariance matrix.

```
$`Correlation statistic`
     var1    var2 Ncases  Nweight       cor      cor_SE      t  df            p cor_fmi cor_VarMI   cor_VarRep
2    escs pv1scie   6334 29037.19 0.3423767 0.018891263  18.12 Inf 2.215950e-73       0         0 3.568798e-04
3    escs pv2scie   6334 29037.19 0.3470641 0.018183736  19.09 Inf 3.057694e-81       0         0 3.306482e-04
5 pv1scie pv2scie   6334 29037.19 0.9136037 0.002796806 326.66 Inf 0.000000e+00       0         0 7.822124e-06

$`Correlation matrix`
$`Correlation matrix`$one1
             escs   pv1scie   pv2scie
escs    1.0000000 0.3423767 0.3470641
pv1scie 0.3423767 1.0000000 0.9136037
pv2scie 0.3470641 0.9136037 1.0000000


$`Covariance statistic`
     var1    var2 Ncases  Nweight          cov      cov_SE cov_df   cov_VarRep
1    escs    escs   6334 29037.19    0.7532142   0.0149807    Inf 2.244215e-04
2    escs pv1scie   6334 29037.19   26.8599562   1.7640406    Inf 3.111839e+00
3    escs pv2scie   6334 29037.19   27.6094959   1.7501336    Inf 3.062968e+00
4 pv1scie pv1scie   6334 29037.19 8171.1609019 240.1109341    Inf 5.765326e+04
5 pv1scie pv2scie   6334 29037.19 7569.8851817 242.8940267    Inf 5.899751e+04
6 pv2scie pv2scie   6334 29037.19 8401.9312483 258.1732486    Inf 6.665343e+04

$`Covariance matrix`
$`Covariance matrix`$one1
              escs    pv1scie   pv2scie
escs     0.7532142   26.85996   27.6095
pv1scie 26.8599562 8171.16090 7569.8852
pv2scie 27.6094959 7569.88518 8401.9312
```

2. Code example – Pearson correlation between several variables for all plausible values

```
myvariables <- c("math", "read", "scie","escs", "hisei", "st011q12ta",
"st004d01t", "st034q02ta", "sc012q01ta")
cor_variables <- c("escs","hisei", "scie")
cor_coef <- correlation(mydata, myvariables, cor_variables)
cor_coef
```

Function result

Outputs four tables: correlation statistic, correlation matrix, covariance statistic and covariance matrix.

```
$`Correlation statistic`
   var1  var2 Ncases  Nweight       cor      cor_SE      t  df            p       cor_fmi     cor_VarMI   cor_VarRep
2  escs hisei   5690 26303.67 0.8696389 0.004870733 178.54 Inf 0.000000e+00 -9.151489e-12 -1.973730e-16 2.372404e-05
3  escs  scie   5690 26303.67 0.3485954 0.018973487  18.37 Inf 2.284118e-75  7.940431e-02  2.598638e-05 3.314082e-04
5 hisei  scie   5690 26303.67 0.3237368 0.018500122  17.50 Inf 1.432692e-68  5.808960e-02  1.807402e-05 3.223731e-04

$`Correlation matrix`
$`Correlation matrix`$one1
           escs     hisei      scie
escs  1.0000000 0.8696389 0.3485954
hisei 0.8696389 1.0000000 0.3237368
scie  0.3485954 0.3237368 1.0000000


$`Covariance statistic`
   var1  var2 Ncases  Nweight          cov       cov_SE cov_df       cov_fmi     cov_VarMI   cov_VarRep
1  escs  escs   5690 26303.67    0.7250491   0.01509054    Inf -9.533907e-13 -1.973730e-16 2.277243e-04
2  escs hisei   5690 26303.67   16.5281365   0.24592329    Inf -2.757035e-12 -1.515825e-13 6.047826e-02
3  escs  scie   5690 26303.67   26.6453300   1.75526708    Inf  7.917952e-02  2.217719e-01 2.837013e+00
4 hisei hisei   5690 26303.67  498.1984755   5.33465983    Inf -1.999893e-12 -5.174014e-11 2.845860e+01
5 hisei  scie   5690 26303.67  648.6335644  42.54771593    Inf  5.635520e-02  9.274571e+01 1.708288e+03
6  scie  scie   5690 26303.67 8057.6721170 239.16538365 870.24  1.016954e-01  5.288166e+03 5.138310e+04

$`Covariance matrix`
$`Covariance matrix`$one1
            escs     hisei       scie
escs   0.7250491  16.52814   26.64533
hisei 16.5281365 498.19848  648.63356
scie  26.6453300 648.63356 8057.67212
```

3. Code example – Pearson correlation between several variables for all plausible values grouped by gender.

```
myvariables <- c("math", "read", "scie","escs", "hisei", "st011q12ta",
"st004d01t", "st034q02ta", "sc012q01ta")
cor_variables <- c("escs","hisei", "scie")
group <- c("st004d01t")
(cor_coef <- correlation(mydata, myvariables, cor_variables, group =
group))
```

Function result

Outputs six tables: correlation statistic, correlation matrix for female, correlation matrix for male, covariance statistic, covariance matrix for female and covariance matrix for male.

```
$`Correlation statistic`
    var1  var2  groupvar groupval Ncases  Nweight       cor       cor_SE      t     df            p      cor_fmi    cor_VarMI   cor_VarRep
3   escs hisei st004d01t        1   2852 13239.26 0.8830391 0.004674764 188.89    Inf 0.000000e+00 9.934842e-12 1.973730e-16 2.185342e-05
4   escs hisei st004d01t        2   2838 13064.41 0.8563069 0.008073197 106.07    Inf 0.000000e+00 3.331112e-12 1.973730e-16 6.517652e-05
5   escs  scie st004d01t        1   2852 13239.26 0.3748305 0.021186472  17.69    Inf 5.007542e-70 7.122519e-02 2.906419e-05 4.168960e-04
6   escs  scie st004d01t        2   2838 13064.41 0.3262172 0.022779241  14.32 673.05 8.549230e-41 1.156373e-01 5.454863e-05 4.588903e-04
9  hisei  scie st004d01t        1   2852 13239.26 0.3318393 0.021795077  15.23    Inf 2.235915e-52 2.128305e-02 9.190898e-06 4.649154e-04
10 hisei  scie st004d01t        2   2838 13064.41 0.3174350 0.023374860  13.58 882.01 2.777863e-38 1.010148e-01 5.017533e-05 4.911912e-04

$`Correlation matrix`
$`Correlation matrix`$st004d01t1
           escs     hisei      scie
escs  1.0000000 0.8830391 0.3748305
hisei 0.8830391 1.0000000 0.3318393
scie  0.3748305 0.3318393 1.0000000

$`Correlation matrix`$st004d01t2
           escs     hisei      scie
escs  1.0000000 0.8563069 0.3262172
hisei 0.8563069 1.0000000 0.3174350
scie  0.3262172 0.3174350 1.0000000


$`Covariance statistic`
    var1  var2  groupvar groupval Ncases  Nweight          cov       cov_SE cov_df       cov_fmi     cov_VarMI   cov_VarRep
1   escs  escs st004d01t        1   2852 13239.26    0.7279614   0.01614682    Inf  0.000000e+00  0.000000e+00 2.607199e-04
2   escs  escs st004d01t        2   2838 13064.41    0.7203024   0.02217563    Inf -4.414979e-13 -1.973730e-16 4.917583e-04
3   escs hisei st004d01t        1   2852 13239.26   16.7365807   0.31225507    Inf  0.000000e+00  0.000000e+00 9.750323e-02
4   escs hisei st004d01t        2   2838 13064.41   16.2950808   0.33082859    Inf  0.000000e+00  0.000000e+00 1.094476e-01
5   escs  scie st004d01t        1   2852 13239.26   27.9052049   1.97773501    Inf  7.859586e-02  2.794751e-01 3.604013e+00
6   escs  scie st004d01t        2   2838 13064.41   25.5201847   2.09430701 739.98  1.102838e-01  4.397439e-01 3.902404e+00
7  hisei hisei st004d01t        1   2852 13239.26  493.4751499   6.98181401    Inf -3.502717e-12 -1.552204e-10 4.874573e+01
8  hisei hisei st004d01t        2   2838 13064.41  502.7352080   7.98650307    Inf  2.676876e-12  1.552204e-10 6.378423e+01
9  hisei  scie st004d01t        1   2852 13239.26  643.2034551  49.78296112    Inf  3.549611e-02  7.997413e+01 2.390372e+03
10 hisei  scie st004d01t        2   2838 13064.41  656.0513369  54.52956941 897.90  1.001169e-01  2.706317e+02 2.675779e+03
11  scie  scie st004d01t        1   2852 13239.26 7613.1830229 267.21016534 383.66  1.531610e-01  9.941721e+03 6.046538e+04
12  scie  scie st004d01t        2   2838 13064.41 8495.9205216 318.94009942 478.97  1.370776e-01  1.267628e+04 8.777887e+04

$`Covariance matrix`
$`Covariance matrix`$st004d01t1
            escs     hisei      scie
escs   0.7279614  16.73658   27.9052
hisei 16.7365807 493.47515  643.2035
scie  27.9052049 643.20346 7613.1830

$`Covariance matrix`$st004d01t2
            escs     hisei       scie
escs   0.7203024  16.29508   25.52018
hisei 16.2950808 502.73521  656.05134
scie  25.5201847 656.05134 8495.92052
```

4. Code example – Spearman correlation

```
cor_variables <- c("escs", "pv1scie")
method <- "Spearman"
cor_coef <- correlation(mydata, myvariables, cor_variables, method =
method)
cor_coef
```

Function result

Outputs only correlation coefficient.

```
Method: Spearman
full data n: 6525
n used: 6334

Correlation: 0.3465872
```

6. Regression

Calculate linear and logistic regression.

6.1. Linear regression

Regression is calculated for one dependent variable and for several independent variables. Linear regression can be calculated with three packages: BIFIEsurvey, EdSurvey and intsvy. Regression can be calculated for single plausible value and for all plausible values named with common names. Common names for plausible values in PISA 2015 data: "math", "read", "scie", "scep", "sced", "scid", "skco", "skpe", "ssph", "ssli", "sses", "flit", "clps".

Function usage

`line_regression(mydata, myvariables, depended, independed, num_pack)`

`mydata`	Data.frame formed with form_data function
`myvariables`	a character vector of the variables to be included in the data. The names of the variables are written in lower case. Country code, student ID, school ID, weights and replicate weight are default in the data. There is no need to write all plausible values names (e.g. “pv1math”, “pv2math”), it is enough to write a common name (e.g. “math”) and all plausible values will be assigned to the data. Common names for plausible values in PISA 2015 data: "math", "read", "scie", "scep", "sced", "scid", "skco", "skpe", "ssph", "ssli", "sses", "flit", "clps".
`depended`	string for the dependent variable in the regression model
`independed`	a character vector of the independed variables
`num_pack`	the package number with which the regression is to be calculated

1. Code example – linear regression with BIFIEsurvey package

```
myvariables <- c("math", "read", "scie","escs", "st011q12ta", "st004d01t",
"st034q02ta", "sc012q01ta")
depended <- "scie"
independed <- c("st011q12ta", "st004d01t")
package <- 1 #1 - BIFIEsurvey, 2 - EdSurvey, 3 - intsvy
reg_equation <- line_regression(mydata, myvariables, depended, independed,
package)
summary(reg_equation)
```

Function result

```
Multiply imputed dataset

Number of persons = 6525
Number of imputed datasets = 10
Number of Jackknife zones per dataset = 0
Fay factor = 0.05

Statistical Inference for Linear Regression

  parameter         var groupvar groupval Ncases Nweight       est fmi  VarMI
1         b (Intercept)      one        1   6390 29329.6 498.4104   1 4.4289
2         b  st011q12ta      one        1   6390 29329.6 -11.6983   1 0.1506
3         b   st004d01t      one        1   6390 29329.6  -5.1523   1 1.8741
4     sigma          NA      one        1   6390 29329.6  90.0638   1 0.1842
5       R^2          NA      one        1   6390 29329.6   0.0251   1 0.0000
6      beta (Intercept)      one        1   6390 29329.6   0.0000   0 0.0000
7      beta  st011q12ta      one        1   6390 29329.6  -0.1535   1 0.0000
8      beta   st004d01t      one        1   6390 29329.6  -0.0282   1 0.0001
```

2. Code example – linear regression with EdSurvey package

```
myvariables <- c("math", "read", "scie","escs", "st011q12ta", "st004d01t",
"st034q02ta", "sc012q01ta")
depended <- "scie"
independed <- c("st011q12ta", "st004d01t")
package <- 2 #1 - BIFIEsurvey, 2 - EdSurvey, 3 - intsvy
reg_equation <- line_regression(mydata, myvariables, depended, independed,
package)
summary(reg_equation)
```

Function result

```
Formula: scie ~ st011q12ta + st004d01t

Weight variable: 'w_fstuwt'
Variance method: jackknife
JK replicates: 80
Plausible values: 10
jrrIMax: 1
full data n: 6525
n used: 6254

Coefficients:
                  coef      se        t    dof  Pr(>|t|)
(Intercept)   482.6121   2.9923 161.2830 65.385 < 2.2e-16 ***
st011q12taNO  -22.4961   5.3158  -4.2319 69.994 6.911e-05 ***
st004d01tMALE  -5.0634   3.0436  -1.6636 64.623     0.101
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Multiple R-squared: 0.0074
```

3. Code example – linear regression with intsvy package

```
myvariables <- c("math", "read", "scie","escs", "st011q12ta", "st004d01t",
"st034q02ta", "sc012q01ta")
depended <- "scie"
independed <- c("st011q12ta", "st004d01t")
package <- 3 #1 - BIFIEsurvey, 2 - EdSurvey, 3 - intsvy
reg_equation <- line_regression(mydata, myvariables, depended, independed,
package)
summary(reg_equation)
```

Function result

```
$LITHUANIA
                         Estimate  Std. Error    t value
(Intercept)           482.46850464 2.924700173 164.963407
ST011Q12TANO          -22.53018685 5.313184231  -4.240430
ST011Q12TANO RESPONSE -88.58246919 9.442062580  -9.381686
ST004D01TMALE          -4.76994562 2.950509753  -1.616651
R-squared               0.02661929 0.005736723   4.640155
```

6.2. Multiple linear regression

Regression is calculated for several dependent variable and for several the same independent variables. Regression can be calculated for single plausible value and for all plausible values named with common names. Common names for plausible values in PISA 2015 data: "math", "read", "scie", "scep", "sced", "scid", "skco", "skpe", "ssph", "ssli", "sses", "flit", "clps".

Function usage

```
multi_regression (mydata, depended, independed)
```

`mydata`	Data.frame formed with form_data function
`depended`	string for the dependent variable in the regression model
`independed`	a character vector of the independed variables

Code example

```
depended <- c("scie","read")
independed <- c("st004d01t")
reg_equation_multi <- multi_regression(mydata, depended, independed)
summary(reg_equation_multi)
```

Function result

```
Formula: scie | read ~ st004d01t

jrrIMax:
Weight variable: 'w_fstuwt'
Variance method:
JK replicates: 80
full data n: 6525
n used: 6525

Coefficients:

scie
                    coef       se         t    dof Pr(>|t|)
(Intercept)   479.1618   2.8465 168.3336 58.371  < 2e-16 ***
st004d01tMALE  -7.3949   3.0571  -2.4189 66.086  0.01833 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

read
                    coef       se         t    dof  Pr(>|t|)
(Intercept)   492.2423   3.0107 163.4963 52.528 < 2.2e-16 ***
st004d01tMALE -39.0857   3.1260 -12.5033 73.161 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual correlation matrix:

      scie  read
scie 1.000 0.879
read 0.879 1.000

Multiple R-squared by dependent variable:

  scie   read
0.0017 0.0429
```

6.3. Logistic regression

Regression is calculated for one dependent variable and for several independent variables. Regression can be calculated for single plausible value and for all plausible values named with common names. Common names for plausible values in PISA 2015 data: "math", "read", "scie", "scep", "sced", "scid", "skco", "skpe", "ssph", "ssli", "sses", "flit", "clps".

Function usage

`log_regression(mydata, myvariables, depended, independed)`

`mydata`	Data.frame formed with form_data function
`myvariables`	a character vector of the variables to be included in the data. The names of the variables are written in lower case. Country code, student ID, school ID, weights and replicate weight are default in the data. There is no need to write all plausible values names (e.g. "pv1math", "pv2math"), it is enough to write a common name (e.g. "math") and all plausible values will be assigned to the data. Common names for plausible values in PISA 2015 data: "math", "read",

	"scie", "scep", "sced", "scid", "skco", "skpe", "ssph", "ssli", "sses", "flit", "clps".
`depended`	string for the dependent variable in the regression model
`independed`	a character vector of the independed variables

Code example

```
myvariables <- c("math", "read", "scie","escs", "st011q12ta", "st004d01t",
"st034q02ta", "sc012q01ta")
depended <- "scie"
independed <- c("st011q12ta", "st004d01t")
reg_log_equation <- log_regression(mydata, myvariables, depended,
independed)
reg_log_equation
```

Function result

	parameter	var	groupvar	groupval	Ncases	Nweight	est	fmi	VarMI
1	b	(Intercept)	one	1	6390	29329.6	0.0000000	0.0000000	0.000000e+00
2	b	st011q12ta	one	1	6390	29329.6	0.0000000	0.0000000	0.000000e+00
3	b	st004d01t	one	1	6390	29329.6	0.0000000	0.0000000	0.000000e+00
4	R2	NA	one	1	6390	29329.6	0.9989879	0.9999973	4.879036e-09

7. Multilevel

This is two-level modelling. About two-level modeling you can read in OECD manual in section “Multilevel analyses” (OECD, 2009). This function provides model estimates and the total error for all replicate weights for one plausible value or calculates the mean of the estimates and the total error for all replicate weights for several plausible values.

Function usage

```
multilevel(mydata, depended, independed, random, class)
```

`mydata`	Data.frame formed with form_data function
`depended`	dependent variable in the two-level model
`independed`	a character vector of independed variables in the two-level model
`random`	a character vector of random variables in the two-level model
`class`	the same grouping variable in the two-level model

1. Code example – empty model (one plausible value)

```
depended <- "pv1scie"
class <- "cntschid"
modmy1a <- multilevel(mydata = mydata1, depended = depended, class = class)
summary.EFECTAS(modmy1a)
```

Function result

Output three table: random effects (estimates and total error for all replicate weights), fixed effects (estimates and total error for all replicate weights) and intraclass correlation (estimate count from random effect intercept and residual, and total error for all replicate weights).

```
Linear mixed model fit by maximum likelihood.
Formula: pv1scie ~ 1 + (1 | cntschid)
Standard errors were calculated for all replicate weights.

Random effects:
   Groups        Name Variance Std. Dev.
 cntschid (Intercept)  2574.89     85.89
 Residual              5351.81    121.04

Number of object:  6092, groups:  cntschid,  310

Fixed effects:
             Estimate Std. Error
 (Intercept)   462.67       0.74

Intraclass correlation:
 Estimate Std. Error
     0.32       0.01

All models results you can find in file analysis_output.txt
This output you can find in file result_output.txt
```

2. Code example – empty model (all plausible value)

```
depended <- "scie"
class <- "cntschid"
modmy1b <- multilevel(mydata = mydata1, depended = depended, class = class)
summary.EFECTAS(modmy1b)
```

Function result

Output three table: random effects (the mean of the estimates and the total error for all replicate weights), fixed effects (the mean of the estimates and the total error for all replicate weights) and intraclass correlation (the mean of the estimates count from random effect intercept and residual, and total error for all replicate weights).

```
Linear mixed model fit by maximum likelihood.
Formula: scie ~ 1 + (1 | cntschid)
Computation of final estimates and their respective standard errors.

Random effects:
   Groups        Name Variance Std. Dev.
 cntschid (Intercept)  2604.34    104.56
 Residual              5416.42    150.49

Number of object:  6092, groups:  cntschid,  310

Fixed effects:
             Estimate Std. Error
 (Intercept)   461.73       1.05

Intraclass correlation:
 Estimate Std. Error
     0.32       0.01

All models results you can find in file analysis_output.txt
This output you can find in file result_output.txt
```

3. Code example – model with fixed slopes (one plausible value)

```
depended <- "pv1scie"
independed <- c("escs", "gender")
class <- "cntschid"
modmy2a <- multilevel(mydata = mydata1, depended = depended,
independed = independed, class = class)
summary.EFECTAS(modmy2a)
```

Function result

Output three table: random effects (estimates and total error for all replicate weights), fixed effects (estimates and total error for all replicate weights) and intraclass correlation (estimate count from random effect intercept and residual, and total error for all replicate weights).

```
Linear mixed model fit by maximum likelihood.
Formula: pv1scie ~ 1 + (1 | cntschid)
Standard errors were calculated for all replicate weights.

Random effects:
   Groups        Name Variance Std. Dev.
 cntschid (Intercept)  2010.28     96.94
 Residual              5220.52    118.18

Number of object:  6092, groups:  cntschid,  310

Fixed effects:
             Estimate Std. Error
 (Intercept)   467.37       1.40
        escs    18.21       1.57
      gender    -1.52       2.24

Intraclass correlation:
 Estimate Std. Error
     0.28       0.01

All models results you can find in file analysis_output.txt
This output you can find in file result_output.txt
```

4. Code example – model with fixed slopes (all plausible value)

```
depended <- "scie"
independed <- c("escs", "gender")
class <- "cntschid"
modmy2b <- multilevel(mydata = mydata1, depended = depended,
independed = independed, class = class)
summary.EFECTAS(modmy2b)
```

Function result

Output three table: random effects (the mean of the estimates and the total error for all replicate weights), fixed effects (the mean of the estimates and the total error for all replicate weights) and intraclass correlation (the mean of the estimates count from random effect intercept and residual, and total error for all replicate weights).

```
Linear mixed model fit by maximum likelihood.
Formula: scie ~ escs + gender + (1 | cntschid)
Computation of final estimates and their respective standard errors.

Random effects:
   Groups        Name Variance Std. Dev.
 cntschid (Intercept)  2045.49    108.66
 Residual              5282.49    147.03

Number of object:  6092, groups:  cntschid,  310

Fixed effects:
             Estimate Std. Error
 (Intercept)   466.70       1.77
        escs    18.22       1.71
      gender    -2.03       2.62

Intraclass correlation:
 Estimate Std. Error
     0.28       0.01

All models results you can find in file analysis_output.txt
This output you can find in file result_output.txt
```

5. Code example – model with random slopes (one plausible value)

```
depended <- "pv1scie"
independed <- c("escs", "gender", "type", "mu_escs", "escs*type",
"escs*mu_escs", "gender*type", "gender*mu_escs")
random <- c("escs", "gender")
class <- "cntschid"
modmy3a <- multilevel(mydata = mydata1, depended = depended,
independed = independed, random = random,
class = class)
summary.EFECTAS(modmy3a)
```

Function result

Output three table: random effects (estimates and total error for all replicate weights), fixed effects (estimates and total error for all replicate weights) and intraclass correlation (estimate count from random effect intercept and residual, and total error for all replicate weights).

```
Linear mixed model fit by maximum likelihood.
Formula: pv1scie ~ 1 + (1 | cntschid)
Standard errors were calculated for all replicate weights.

Random effects:
     Groups        Name Variance Std. Dev.
   cntschid (Intercept)   993.66     68.76
 cntschid.1        escs    84.14     89.28
 cntschid.2      gender   164.59    239.07
   Residual              5117.44    185.46

Number of object:  6092, groups:  cntschid,  310

Fixed effects:
                Estimate Std. Error
    (Intercept)   479.27       1.27
           escs    16.50       1.25
         gender    -2.26       2.55
           type  -102.76       5.53
        mu_escs    63.25       1.86
      escs:type   -21.33       3.97
   escs:mu_escs     5.39      13.65
    gender:type    13.08       3.66
 gender:mu_escs    -2.65       8.69

Intraclass correlation:
 Estimate Std. Error
     0.16       0.01

All models results you can find in file analysis_output.txt
This output you can find in file result_output.txt
```

6. Code example – model with random slopes (all plausible value)

```
depended <- "scie"
independed <- c("escs", "gender", "type", "mu_escs", "escs*type",
"escs*mu_escs", "gender*type", "gender*mu_escs")
random <- c("escs", "gender")
class <- "cntschid"
modmy3b <- multilevel(mydata = mydata1, depended = depended,
independed = independed, random = random,
class = class)
summary.EFECTAS(modmy3b)
```

Function result

Output three table: random effects (the mean of the estimates and the total error for all replicate weights), fixed effects (the mean of the estimates and the total error for all replicate weights) and intraclass correlation (the mean of the estimates count from random effect intercept and residual, and total error for all replicate weights).

```
Linear mixed model fit by maximum likelihood.
Formula: scie ~ escs + gender + type + mu_escs + escs * type + escs * mu_escs + gender * type + gender * mu_escs + (escs + gender || cntschid)
Computation of final estimates and their respective standard errors.

Random effects:
     Groups        Name Variance Std. Dev.
   cntschid (Intercept)  1038.73     83.22
 cntschid.1        escs    80.30     92.41
 cntschid.2      gender   152.84    241.20
   Residual              5182.96    208.12

Number of object:  6092, groups:  cntschid,  310

Fixed effects:
               Estimate Std. Error
    (Intercept)  478.62       1.73
           escs   16.63       1.45
         gender   -2.88       2.96
           type  -98.44       8.77
        mu_escs   64.95       2.59
      escs:type  -20.23       7.13
   escs:mu_escs    5.75      12.36
    gender:type    3.93      14.28
 gender:mu_escs   -6.69       8.90

Intraclass correlation:
 Estimate Std. Error
     0.17       0.01

All models results you can find in file analysis_output.txt
This output you can find in file result_output.txt
```

7.1. Data preparation according to OECD manual

Data must be cleared of NA values or incorrect values before analysis. NA values can be removed. NA values can be removed by importing data with getData function from EdSurvey package. replace_value function helps to replace numeric and text data values with other numeric values.

Function usage

```
replace_value(mydata, oldname, newname, change)
```

`mydata`	Data.frame formed with form_data function
`oldname`	column name of the replaceable values
`newname`	new column name of the replaceable values. Can be empty than the values will be rewritten.
`change`	vector describing the change of values. Odd variables shows old values, and even variables shows new values. For example: 1) gender vector consists of values 1 (female) and 2 (male). You want to change to 0 (male) and 1 (female), than change vector will be c(2,0) 2) gender vector consists of values “Female” and “Male”. You want to change to 0 (male) and 1 (female), than change vector will be c(“Male”, 0, “Female”, 1)

1. Code example – all values change to different values

```
change <- c("FEMALE", 1, "MALE", 0)
mydata1 <- replace_value(mydata = mydata1, oldname = "st004d01t", newname =
"gender", change = change)
```

Function result

A new column “gender” will appear in mydata1 data. Column “gender” values will be 0 and 1. The old column “st004d01t” with values “FEMALE” and “MALE” will remain.

2. Code example – some values change to the same values

```
change <- c(111, 0, 121, 0, 112, 0, 122, 1, 222, 1)
mydata1 <- replace_value(mydata = mydata1, oldname = "immig", change =
change)
```

Function result

Column "immig" values 111, 121, 112, 122 and 222 will be replaced with values 0 and 1.

3. Code example – data preparation according to OECD manual

```
change <- c("FEMALE", 1, "MALE", 0)
mydata1 <- replace_value(mydata2 = mydata1, oldname = "st004d01t",
      newname = "gender", change = change)
change <- c("NO RESPONSE", 9, "OTHER COUNTRY", 2, "COUNTRY OF TEST",
             1)
mydata1 <- replace_value(mydata2 = mydata1, oldname = "st019aq01t",
                           change = change)
mydata1 <- replace_value(mydata2 = mydata1, oldname = "st019bq01t",
                           change = change)
mydata1 <- replace_value(mydata2 = mydata1, oldname = "st019cq01t",
                           change = change)
mydata1$immig <- (100*mydata1$st019aq01t)+(10*mydata1$st019bq01t)+
  (mydata1$st019cq01t)
change <- c(111, 0, 121, 0, 112, 0, 122, 1, 222, 1)
mydata1 <- replace_value(mydata2 = mydata1, oldname = "immig",
change = change)
mydata1$st019aq01t <- NULL
mydata1$st019bq01t <- NULL
mydata1$st019cq01t <- NULL
change <- c("GENERAL", 0, "PRE-VOCATIONAL", 1, "VOCATIONAL", 1,
             "MODULAR", 1)
mydata1 <- replace_value(mydata2 = mydata1, oldname = "iscedo",
                           newname = "vocation", change = change)
mydata1 <- na.omit(mydata1)
```

7.2. Weight normalization acordint to OECD manual

The sum of the weights is equal to the number of students in the dataset.

Function usage

```
normalization_weight(mydata)
```

mydata	Data.frame formed with form_data function

Code example

```
mydata1 <- normalization_weight(mydata1)
```

Literature

Bailey, P., C'deBaca, R., Emad, A., Huo, H., Lee, M., Liao, Y., . . . Zhang, T. (2019). Edsurvey: Analysis of nces education survey and assessment data [Computer software manual]. Retrieved from https://www.air.org/project/nces-data-r-project-edsurvey (R package version 2.3.2)

Caro, D., Biecek, P. (2019). Intsvy: International Assessment Data Manager [Computer software manual]. Retrieved from https://cran.r-project.org/web/packages/intsvy/intsvy.pdf (R package version 2.4)

OECD (2009), PISA Data Analysis Manual: SAS, Second Edition, PISA, OECD Publishing, Paris, https://doi.org/10.1787/9789264056251-en.