Package 'intsvy'

November 23, 2012

Type Package

Title Data Manager of International Assessment Studies of Student Performance

Version 1.0

Date 2012-11-23

Author Daniel Caro

Maintainer Daniel Caro <daniel.caro@education.ox.ac.uk>

Description Provides tools for importing and analyzing data from international student assessments

License GPL-2

Depends memisc, foreign, plyr

LazyData yes

Repository CRAN

Date/Publication 2012-11-23 16:54:14

R topics documented:

intsvy-package 2
country.ug 3
pirls.ben.pv 3
pirls.mean 5
pirls.mean.pv 6
pirls.reg 7
pirls.reg.pv 9
pirls.rho 10
pirls.rho.pv 12
pirls.select.merge 14
pirls.table 16
pirls.var.label 17
pisa.mean 18

pisa.mean.pv 20
pisa.reg.pv 21
pisa.table 23
timss.ben.pv 24
timss.mean 26
timss.mean.pv 27
timss.reg 28
timss.reg.pv 30
timss.rho 31
timss.rho.pv 33
timss.select.merge 35
timss.var.label 37

Index **39**

`intsvy-package` *Analysis of international large-scale assessment datasets (PIRLS, TIMSS, PISA)*

Description

The package provide useRs with tools for reading and analyzing datasets from three international assessments (i.e., PIRLS 2006, TIMSS 2007, and PISA 2009). To read the data, it includes functions that view labels, allow the useR to select variables and countries, merge student and school files, and finally import the data for analysis into R. For the analysis, it includes functions for producing frequency tables and calculating correlations, means, and regressions that handling plausible values and the complex sample design (i.e., replicate weights)

Details

Package:	intsvy
Type:	Package
Version:	1.0
Date:	2012-11-23
License:	GPL-2

Author(s)

Daniel Caro <daniel.caro@education.ox.ac.uk>

References

Technical reports of the corresponding surveys

country.ug *Country abbreviations and codes*

Description

The dataset contains the names, abbreviations, and ISO codes for the countries participating in PIRLS and TIMSS. These labels are not included in the original datasets are necessary for selecting countries and printing outputs.

Usage

```
country.ug
```

Format

A data frame

Source

The PIRLS/TIMSS User Guides

pirls.ben.pv *Calculates percentage of students at or above a benchmark*

Description

pirls.ben.pv calculates the percentage of students performing at or above the levels of achievement given by the useR. It produces the results for the international PIRLS/TIMSS benchmarks by default

Usage

```
pirls.ben.pv(pvlabel, cutoff = c(400, 475, 550, 625), by, data)
```

Arguments

pvlabel	The label corresponding to the achievement variable, e.g., "ASRREA" for overall reading performance.
cutoff	The cut-off points or benchmarks for calculating the percentage of students. The default are the PIRLS/TIMSS international benchmarks: cutoff = c(400, 475, 550, 625)
by	The variable label defining the grouping, usually the countries (by="IDCNTRY"), but could be any other categorical variable.
data	An R object, normally a data frame, containing the data from PIRLS.

Value

pirls.ben.pv returns a data frame with the percentage of students at or above the benchmark and the corresponding standard error.

Author(s)

Daniel Caro

See Also

timss.ben.pv

Examples

```
## Not run:
## Benchmarks in international report 2006
pirls.ben.pv(pvlabel ="ASRREA", cutoff = c(400, 475, 550, 625), data=pirls.data)

## End(Not run)

## The function is currently defined as
function (pvlabel, cutoff = c(400, 475, 550, 625), by, data)
{
    pv.ben.input <- function(pvlabel, cutoff, data) {
        pvname <- paste(pvlabel, "0", 1:5, sep = "")
        tabpv1 <- sapply(1:length(cutoff), function(z) sapply(1:75,
            function(x) percent(data[[pvname[1]]] >= cutoff[z],
                weights = ifelse(data[["JKZONE"]] == x, 2 * data[["TOTWGT"]] *
                  data[["JKREP"]], data[["TOTWGT"]]), na.rm = TRUE,
                total = F)))
        tabpv <- sapply(1:length(cutoff), function(z) sapply(pvname,
            function(x) percent(data[[x]] >= cutoff[z], weights = data[["TOTWGT"]],
                na.rm = TRUE, total = F)))
        tabpvw <- sapply(1:length(cutoff), function(y) sum(sapply(1:75,
            function(x) (tabpv1[x, y] - tabpv[1, y])^2)))
        tabpvb <- (1 + 1/5) * apply(tabpv, 2, var)
        tabse <- round((tabpvw + tabpvb)^(1/2), 2)
        tabtot <- round(apply(tabpv, 2, mean), 2)
        result <- data.frame(Benchmark = paste(rep("At or above",
            length(cutoff)), cutoff), Percentage = tabtot, `Std. err.` = tabse,
            check.names = F)
        return(result)
    }
    if (missing(by)) {
        return(pv.ben.input(pvlabel = pvlabel, cutoff = cutoff,
            data = data))
    }
    else {
        return(ddply(data, by, function(x) pv.ben.input(pvlabel = pvlabel,
            cutoff = cutoff, data = x)))
    }
  }
```

`pirls.mean` *Calculates the mean*

Description

Calculates the mean for an observed variable (NOT one with plausible values)

Usage

```
pirls.mean(variable, by, data)
```

Arguments

`variable`	A variable label. The mean is calculated for this variable (e.g., variable="ASDAGE")
`by`	The variable label defining the grouping, usually the countries (by="IDCNTRY"), but could be any other categorical variable.
`data`	An R object, normally a data frame, containing the data from PIRLS.

Value

pirls.mean returns a data frame including the means and standard errors.

Author(s)

Daniel Caro

See Also

timss.mean, pisa.mean

Examples

```
## Not run:
## Replicating Exhibit 5.10 in p.162 of User Guide 2006

# Number of hours spent reading stories or articles in books or magazines outside of school
pirls.data$TIME[as.numeric(pirls.data$ASBGTSP4) ==1] <-5.5
pirls.data$TIME[as.numeric(pirls.data$ASBGTSP4) ==2] <-4
pirls.data$TIME[as.numeric(pirls.data$ASBGTSP4) ==3] <-2
pirls.data$TIME[as.numeric(pirls.data$ASBGTSP4) ==4] <-0.5
pirls.data$TIME[as.numeric(pirls.data$ASBGTSP4) ==5] <-0

table(pirls.data$ASBGTSP4, pirls.data$TIME)

pirls.mean(variable="TIME", by="IDCNTRYL", data=pirls.data)
## End(Not run)


## The function is currently defined as
```

```
function (variable, by, data)
{
    mean.input <- function(variable, data) {
        meanrp <- sapply(1:75, function(x) weighted.mean(as.numeric(data[[variable]]),
            ifelse(data[["JKZONE"]] == x, 2 * data[["TOTWGT"]] *
                data[["JKREP"]], data[["TOTWGT"]]), na.rm = TRUE))
        meantot <- weighted.mean(as.numeric(data[[variable]]),
            data[["TOTWGT"]], na.rm = TRUE)
        meanse <- (sum((meanrp - meantot)^2))^(1/2)
        result <- data.frame(Mean = meantot, Std.err. = meanse)
        return(round(result, 2))
    }
    if (missing(by)) {
        return(mean.input(variable = variable, data = data))
    }
    else {
        return(ddply(data, by, function(x) mean.input(data = x,
            variable = variable)))
    }
  }
```

`pirls.mean.pv` *Calculates mean performance*

Description

pirls.mean.pv uses five plausible values to calculate mean performance and its standard error

Usage

```
pirls.mean.pv(pvlabel, by, data)
```

Arguments

`pvlabel`	The label corresponding to the achievement variable, e.g., "ASRREA" for overall reading performance.
`by`	The variable label defining the grouping, usually the countries (by="IDCNTRY"), but could be any other categorical variable.
`data`	An R object, normally a data frame, containing the data from PIRLS.

Value

pirls.mean.pv returns a data frame with the mean and standard errors.

Author(s)

Daniel Caro

See Also

timss.mean.pv, pisa.mean.pv

Examples

```
## Not run:
## Replicating Exhibit 3.10 in User Guide 2006, p.64
table(as.numeric(pirls.data$ASBGTOC5), pirls.data$ASBGTOC5)
pirls.data$NEWTOCS[as.numeric(pirls.data$ASBGTOC5) ==1 ] <- "EVERY DAY OR ALMOST EVERY DAY"
pirls.data$NEWTOCS[as.numeric(pirls.data$ASBGTOC5) ==2 ] <- "ONCE OR TWICE A WEEK"
pirls.data$NEWTOCS[as.numeric(pirls.data$ASBGTOC5) ==3 | as.numeric(pirls.data$ASBGTOC5) ==4 ] <- "TWICE A MON

table(pirls.data$NEWTOCS)

pirls.mean.pv(pvlabel="ASRREA", by=c("IDCNTRYL", "NEWTOCS"), data=pirls.data)
## End(Not run)


## The function is currently defined as
function (pvlabel, by, data)
{
    pv.input <- function(pvlabel = "ASRREA", data) {
        pvnames <- paste(pvlabel, "0", 1:5, sep = "")
        read1m <- sapply(1:75, function(x) weighted.mean(data[[pvnames[[1]]]],
            ifelse(data[["JKZONE"]] == x, 2 * data[["TOTWGT"]] *
                data[["JKREP"]], data[["TOTWGT"]]), na.rm = TRUE))
        readm <- sapply(pvnames, function(x) weighted.mean(data[[x]],
            data[["TOTWGT"]], na.rm = TRUE))
        varw <- sum((read1m - readm[1])^2)
        varb <- (1 + 1/5) * var(readm)
        readse <- (varw + varb)^(1/2)
        result <- data.frame(Mean = mean(readm), Std.err. = readse)
        return(round(result, 2))
    }
    if (missing(by)) {
        return(pv.input(pvlabel = pvlabel, data = data))
    }
    else {
        return(ddply(data, by, function(x) pv.input(data = x,
            pvlabel = pvlabel)))
    }
  }
```

---

pirls.reg *Regression analysis*

Description

pirls.mean performs standard linear regression analysis (OLS) for an observed depedent variable (NOT for plausible values).

Usage

```
pirls.reg(regform, by, data)
```

Arguments

regform	A regression formula (e.g., "ASBGBOOK ~ ITSEX").
by	The variable label defining the grouping, usually the countries (by="IDCNTRY"), but could be any other categorical variable.
data	An R object, normally a data frame, containing the data from PIRLS.

Value

pirls.reg returns a data frame with coefficients, standard errors and t-values. If "by" is specified, results are reported in a list.

Author(s)

Daniel Caro

See Also

timss.reg

Examples

```
## Not run:
# Replicating Exhibit 3.13 in User Guide 2006, p. 68

# Recode ASBGBOOK
table(as.numeric(pirls.data$ASBGBOOK), pirls.data$ASBGBOOK)
pirls.data$BOOK[as.numeric(pirls.data$ASBGBOOK)==1] <- 5
pirls.data$BOOK[as.numeric(pirls.data$ASBGBOOK)==2] <- 18
pirls.data$BOOK[as.numeric(pirls.data$ASBGBOOK)==3] <- 63
pirls.data$BOOK[as.numeric(pirls.data$ASBGBOOK)==4] <- 151
pirls.data$BOOK[as.numeric(pirls.data$ASBGBOOK)==5] <- 251
table(pirls.data$BOOK)

# Exhibit 3.13 p.68 User Guide
pirls.reg(regform= "BOOK ~ ITSEX", by="IDCNTRYL", data=pirls.data)

## End(Not run)



## The function is currently defined as
function (regform, by, data)
{
    reg.input <- function(regform, data) {
        Coefrp <- sapply(1:75, function(i) coefficients(lm(formula = as.formula(regform),
            data = data, weights = ifelse(data[["JKZONE"]] ==
                i, 2 * data[["TOTWGT"]] * data[["JKREP"]], data[["TOTWGT"]]))))
```

```
        Reg <- summary(lm(formula = as.formula(regform), data = data,
            weights = data[["TOTWGT"]]))
        Coeftot <- Reg$coefficients[, 1]
        R2 <- Reg$r.squared
        CoefSE <- apply((Coefrp - Coeftot)^2, 1, sum)^(1/2)
        CoefT <- Coeftot/CoefSE
        RegTab <- round(data.frame(Estimate = Coeftot, `Std. Error` = CoefSE,
            `t value` = CoefT, check.names = F), 2)
        return(RegTab)
    }
    if (missing(by)) {
        return(reg.input(regform = regform, data = data))
    }
    else {
        return(lapply(split(data, data[by]), function(x) reg.input(regform = regform,
            data = x)))
    }
}
```

`pirls.reg.pv` *Regression analysis for plausible values*

Description

pirls.reg.pv performs standard linear regression analysis (OLS) while handling plausible values and replicate weights.

Usage

```
pirls.reg.pv(x, pvlabel = "ASRREA", by, data)
```

Arguments

`x`	Data labels of independent variables (e.g., x = c("ASDHEHLA", "ITSEX")).
`pvlabel`	The label corresponding to the achievement variable, e.g., "ASRREA" for overall reading performance.
`by`	The variable label defining the grouping, usually the countries (by="IDCNTRY"), but could be any other categorical variable.
`data`	An R object, normally a data frame, containing the data from PIRLS.

Value

pirls.reg.pv returns a data frame with coefficients, standard errors and t-values. If "by" is specified, results are reported in a list.

Author(s)

Daniel Caro

See Also

timss.reg.pv, pisa.reg.pv

Examples

```
## Not run:
## Replicating Exhibit 3.15 in User Guide 2006, p.70
pirls.reg.pv(x=c("ITSEX"), pvlabel = "ASRREA", by="IDCNTRYL", data=pirls.data)
## End(Not run)


## The function is currently defined as
function (x, pvlabel = "ASRREA", by, data)
{
    reg.pv.input <- function(x, pvlabel = pvlabel, data) {
        pvnames <- paste(pvlabel, "0", 1:5, sep = "")
        regform <- lapply(pvnames, function(i) paste(i, "~",
            paste(x, collapse = "+")))
        Coefrpv1 <- sapply(1:75, function(i) coefficients(lm(formula = as.formula(regform[[1]]),
            data = data, weights = ifelse(data[["JKZONE"]] ==
                i, 2 * data[["TOTWGT"]] * data[["JKREP"]], data[["TOTWGT"]])))
        Regpv <- lapply(regform, function(i) summary(lm(formula = i,
            data = data, weights = data[["TOTWGT"]])))
        Coefpv <- sapply(Regpv, function(i) i[["coefficients"]][,
            1])
        R2 <- mean(sapply(Regpv, function(i) i[["r.squared"]]))
        Coeftot <- apply(Coefpv, 1, mean)
        Varw <- apply((Coefrpv1 - Coefpv[, 1])^2, 1, sum)
        Varb <- (1 + 1/5) * apply(Coefpv, 1, var)
        CoefSE <- (Varw + Varb)^(1/2)
        CoefT <- Coeftot/CoefSE
        RegTab <- round(data.frame(Estimate = Coeftot, `Std. Error` = CoefSE,
            `t value` = CoefT, check.names = F), 2)
        return(RegTab)
    }
    if (missing(by)) {
        return(reg.pv.input(x = x, pvlabel = pvlabel, data = data))
    }
    else {
        return(lapply(split(data, data[by]), function(i) reg.pv.input(x = x,
            pvlabel = pvlabel, data = i)))
    }
  }
```

`pirls.rho` *Correlation matrix*

Description

pirls.rho produces a correlations matrix for observed variables (NOT for plausible values)

Usage

```
pirls.rho(variables, by, data)
```

Arguments

variables	Data labels for the variables in the correlation matrix (e.g., variables=c("ASRREA01", "ASDAGE"))
by	The variable label defining the grouping, usually the countries (by="IDCNTRY"), but could be any other categorical variable.
data	An R object, normally a data frame, containing the data from PIRLS.

Value

pirls.rho returns a matrix including correlation and standard error values

Author(s)

Daniel Caro

See Also

timss.rho

Examples

```
## Not run:
pirls.rho(variables=c("ASRREA01", "ASDAGE"), by="IDCNTRYL", data=pirls.data)

## End(Not run)


## The function is currently defined as
function (variables, by, data)
{
    rho.input <- function(variables, data) {
        data <- na.omit(data[c(variables, "TOTWGT", "JKREP",
            "JKZONE")])
        rhorp <- lapply(1:75, function(i) cov.wt(data[variables],
            wt = ifelse(data[["JKZONE"]] == i, 2 * data[["TOTWGT"]] *
                data[["JKREP"]], data[["TOTWGT"]]), cor = T)[[5]])
        rhotot <- cov.wt(data[variables], wt = data[["TOTWGT"]],
            cor = T)[[5]]
        rhose <- Reduce("+", lapply(rhorp, function(x) (x - rhotot)^2))^(1/2)
        rhomat <- round(do.call(cbind, lapply(1:ncol(rhotot),
            function(x) t(rbind(rhotot[, x], rhose[, x])))),
            3)
        colnames(rhomat) <- unlist(lapply(1:length(variables),
            function(x) c(paste(variables, "Rho", sep = " ")[x],
                paste(variables, "SE", sep = " ")[x])))
        return(round(rhomat, 6))
    }
```

```
  if (missing(by)) {
      return(rho.input(variables = variables, data = data))
  }
  else {
      return(lapply(split(data, data[by]), function(x) rho.input(variables = variables,
          data = x)))
  }
}
```

---

pirls.rho.pv *Two-way weighted correlation*

Description

pirls.rho.pv calculates the correlation and standard error among two achievement variables each based on 5 plausible values or one achievement variable and an observed variable (i.e., with observed scores rather than plausible values).

Usage

```
pirls.rho.pv(variable, pvlabels, by, data)
```

Arguments

variable	A data label for the observed variable (e.g., variable="ASDAGE")
pvlabels	One or two labels describing the achievement variables (e.g., pvlabels = c("ASRLIT", "ASRINF"))
by	The variable label defining the grouping, usually the countries (by="IDCNTRY"), but could be any other categorical variable.
data	An R object, normally a data frame, containing the data from PIRLS.

Value

pirls.rho.pv returns a matrix with correlations and standard errors.

Author(s)

Daniel Caro

See Also

timss.rho.pv

Examples

```
## Not run:
# Replicating Exhibit A.15 p.309 in International report 2006
pirls.rho.pv(pvlabels=c("ASRLIT", "ASRINF"), by="IDCNTRYL", data=pirls.data)

## End(Not run)


## The function is currently defined as
function (variable, pvlabels, by, data)
{
    rho.pv.input <- function(variable, pvlabels, data) {
        if (length(pvlabels) == 2 & missing(variable)) {
            pvnames <- lapply(pvlabels, function(x) paste(x,
                "0", 1:5, sep = ""))
            data <- na.omit(data[c(unlist(pvnames), "TOTWGT",
                "JKREP", "JKZONE")])
            rhopvrp <- lapply(1:75, function(i) cov.wt(x = data[c(pvnames[[1]][1],
                pvnames[[2]][1])], cor = T, wt = ifelse(data[["JKZONE"]] ==
                i, 2 * data[["TOTWGT"]] * data[["JKREP"]], data[["TOTWGT"]]))[[5]])
            rhopvtot <- lapply(1:5, function(i) cov.wt(x = data[c(pvnames[[1]][i],
                pvnames[[2]][i])], cor = T, wt = data[["TOTWGT"]])[[5]])
            varw <- Reduce("+", lapply(rhopvrp, function(x) (x -
                rhopvtot[[1]])^2))
            varb <- (1 + 1/5) * apply(simplify2array(rhopvtot),
                c(1, 2), var)
            rhose <- (varw + varb)^(1/2)
            rhotot <- Reduce("+", rhopvtot)/length(rhopvtot)
            rhomat <- round(do.call(cbind, lapply(1:ncol(rhotot),
                function(x) t(rbind(rhotot[, x], rhose[, x])))),
                5)
            colnames(rhomat) <- unlist(lapply(1:2, function(i) c(paste(pvlabels,
                "Rho", sep = " ")[i], paste(pvlabels, "SE", sep = " ")[i])))
            return(round(rhomat, 3))
        }
        else {
            pvnames <- paste(pvlabels, "0", 1:5, sep = "")
            data <- na.omit(data[c(variable, pvnames, "TOTWGT",
                "JKREP", "JKZONE")])
            rhopvrp <- lapply(1:75, function(i) cov.wt(x = data[c(variable,
                pvnames[1])], cor = T, wt = ifelse(data[["JKZONE"]] ==
                i, 2 * data[["TOTWGT"]] * data[["JKREP"]], data[["TOTWGT"]]))[[5]])
            rhopvtot <- lapply(pvnames, function(i) cov.wt(x = data[c(variable,
                i)], cor = T, wt = data[["TOTWGT"]])[[5]])
            varw <- Reduce("+", lapply(rhopvrp, function(x) (x -
                rhopvtot[[1]])^2))
            varb <- (1 + 1/5) * apply(simplify2array(rhopvtot),
                c(1, 2), var)
            rhose <- (varw + varb)^(1/2)
            rhotot <- Reduce("+", rhopvtot)/length(rhopvtot)
            rhomat <- round(do.call(cbind, lapply(1:ncol(rhotot),
                function(x) t(rbind(rhotot[, x], rhose[, x])))),
```

```
                5)
            colnames(rhomat) <- unlist(lapply(1:2, function(i) c(paste(c(variable,
                pvlabels), "Rho", sep = " ")[i], paste(c(variable,
                pvlabels), "SE", sep = " ")[i])))
            return(round(rhomat, 3))
        }
    }
    if (missing(by)) {
        return(rho.pv.input(variable = variable, pvlabels = pvlabels,
            data = data))
    }
    else {
        if (length(pvlabels) == 2 & missing(variable)) {
            return(lapply(split(data, data[by]), function(x) rho.pv.input(pvlabels = pvlabels,
                data = x)))
        }
        else {
            return(lapply(split(data, data[by]), function(x) rho.pv.input(variable = variable,
                pvlabels = pvlabels, data = x)))
        }
    }
}
```

`pirls.select.merge` *Select and merge data*

Description

pirls.select.merge selects and merges data from PIRLS 2006 downloaded from the IEA data repository

Usage

```
pirls.select.merge(folder = getwd(), countries, student, home, school)
```

Arguments

`folder`	Directory path where the data are located. The data could be organized within folders but it should not be duplicated.
`countries`	The selected countries, supplied with the abbreviation (e.g., countries=c("AUT", "BGR") or codes (countries=c(40, 100)). If no countries are selected, all are selected.
`student`	The data labels for the selected student variables.
`home`	The data labels for the selected home background variables.
`school`	The data labels for the selected school variables.

Value

pirls.select.merge returns a data frame with the selected data from PIRLS 2006.

Author(s)

Daniel Caro

See Also

timss.select.merge

Examples

```
## Not run:

## Using country abbreviations
pirls.data <- pirls.select.merge(folder= "/home/eldani/Work/R work/PIRLS/Data",
              countries= c("AUT", "BGR", "TWN", "ZAF", "ESP", "SWE", "TTO", "MKD", "USA"),
              student= c("ITSEX", "ITLANG", "ASBGBOOK", "ASDAGE", "ASDHHER", "ASBGTOC5", "ASDGTHC2", "ASBGTSP4
              home= c("ASDHEHLA", "ASDHEDUP", "ASDHPEMP", "ASDHPATR", "ASDHOCCP"),
              school= c("ACDGASR", "ACDGCMP", "ACDGPPSS"))

# Using country ISO codes
pirls.data <- pirls.select.merge(folder= "/home/eldani/Work/R work/PIRLS/Data",
              countries= c(276, 504, 554, 643, 752),
              student= c("ITSEX", "ITLANG", "IDBOOK", "ASDAGE", "ASDHHER", "ASDGTHC2"),
              home= c("ASDHEHLA", "ASDHEDUP", "ASDHPEMP", "ASDHPATR", "ASDHOCCP"),
              school= c("ACDGASR", "ACDGCMP", "ACDGPPSS"))


## End(Not run)


## The function is currently defined as
function (folder = getwd(), countries, student, home, school)
{
    files.all <- lapply(c("asg", "ash", "acg"), function(x) list.files(folder,
        full.names = T, pattern = paste("^", x, ".*.sav$", sep = ""),
        recursive = T))
    cntlab <- toupper(unique(unlist(lapply(files.all, function(x) substr(x,
        nchar(x) - 8, nchar(x) - 6)))))
   country.list <- country.ug[country.ug$ISO %in% intersect(country.ug$ISO,
        cntlab), ]
    if (missing(countries)) {
        countries = cntlab
    }
    if (is.numeric(countries)) {
        countries = country.list[country.list$Code %in% countries,
            "ISO"]
    }
    files.select <- lapply(files.all, function(y) sapply(countries,
        function(x) y[substr(y, nchar(y) - 8, nchar(y) - 6) ==
            tolower(x)]))
    student.data <- do.call("rbind", lapply(lapply(files.select[[1]],
        function(y) spss.system.file(y)), function(x) as.data.frame(x[,
        c("IDCNTRY", "IDSCHOOL", "IDCLASS", "IDSTUD", "JKREP",
```

```
            "JKZONE", "HOUWGT", "SENWGT", "TOTWGT", "ASRREA01",
            "ASRREA02", "ASRREA03", "ASRREA04", "ASRREA05", "ASRINF01",
            "ASRINF02", "ASRINF03", "ASRINF04", "ASRINF05", "ASRLIT01",
            "ASRLIT02", "ASRLIT03", "ASRLIT04", "ASRLIT05", student)])))
  student.data$IDCNTRYL <- factor(student.data$IDCNTRY, levels = country.list[country.list$Code %in%
      unique(student.data$IDCNTRY), "Code"], labels = country.list[country.list$Code %in%
      unique(student.data$IDCNTRY), "Country"])
  home.data <- do.call("rbind", lapply(lapply(files.select[[2]],
      function(y) spss.system.file(y)), function(x) as.data.frame(x[,
      c("IDCNTRY", "IDSTUD", home)])))
  school.data <- do.call("rbind", lapply(lapply(files.select[[3]],
      function(y) spss.system.file(y)), function(x) as.data.frame(x[,
      c("IDCNTRY", "IDSCHOOL", "SCHWGT", school)])))
  pirls.student <- merge(student.data, home.data, by = c("IDCNTRY",
      "IDSTUD"), suffixes = c(".st", ".hm"))
  pirls.all <- merge(pirls.student, school.data, by = c("IDCNTRY",
      "IDSCHOOL"), suffixes = c(".st", ".sc"))
  return(pirls.all)
}
```

`pirls.table` *Frequency table*

Description

pirls.table produces a frequency table for a categorical variable printing percentages and standard errors.

Usage

```
pirls.table(variable, by, data)
```

Arguments

`variable`	The data label with the variable to be analyzed.
`by`	The variable label defining the grouping, usually the countries (by="IDCNTRY"), but could be any other categorical variable.
`data`	An R object, normally a data frame, containing the data from PIRLS.

Value

pirls.table returns a data frame with percentages and standard errors.

Author(s)

Daniel Caro

See Also

timss.table, pisa.table

Examples

```
## Not run:
pirls.table(variable="ITSEX", by=c("IDCNTRYL", "ASDHPATR"), data=pirls.data)

## End(Not run)


## The function is currently defined as
function (variable, by, data)
{
    table.input <- function(variable, data) {
        tabrp <- sapply(1:75, function(x) percent(data[[variable]],
            total = F, weights = ifelse(data[["JKZONE"]] == x,
                2 * data[["TOTWGT"]] * data[["JKREP"]], data[["TOTWGT"]]),
            na.rm = TRUE))
        tabtot <- round(percent(data[[variable]], weights = data[["TOTWGT"]],
            na.rm = TRUE, total = F), 3)
        tabse <- round(sapply(1:length(table(data[[variable]])),
            function(x) sum(sapply(1:75, function(y) (tabrp[x,
                y] - tabtot[[x]])^2))^(1/2)), 3)
        result <- data.frame(table(data[[variable]]), Percentage = as.numeric(tabtot),
            Std.err. = tabse)
        names(result)[1] <- variable
        return(result)
    }
    if (missing(by)) {
        return(table.input(variable = variable, data = data))
    }
    else {
        return(ddply(data, by, function(x) table.input(data = x,
            variable = variable)))
    }
  }
```

`pirls.var.label` *Data labels*

Description

pirls.var.labels prints and saves variable labels in a text file

Usage

```
pirls.var.label(folder = getwd())
```

Arguments

folder	Directory path where the data are located. The data could be organized within folders but it should not be duplicated.

Value

pirls.var.label returns a list with variable labels for the student, home, and school data. The output is also printed on a text sheet located in "folder".

Author(s)

Daniel Caro

See Also

timss.var.label

Examples

```
## Not run:
pirls.var.label(folder= getwd())

## End(Not run)


## The function is currently defined as
function (folder = getwd())
{
    files.all <- lapply(c("asg", "ash", "acg"), function(x) list.files(folder,
        full.names = T, pattern = paste("^", x, ".*.sav$", sep = ""),
        recursive = T))
    var.label <- list(‘Student background and achievement‘ = description(spss.system.file(files.all[[1]][1])),
        ‘Home background‘ = description(spss.system.file(files.all[[2]][1])),
        ‘School variables‘ = description(spss.system.file(files.all[[3]][1])))
    print(var.label)
    capture.output(var.label, file = file.path(folder, "Variable labels.txt"))
    cat("The file 'Variable labels.txt' in the directory", folder,
        "contains the variable labels of the complete dataset",
        sep = " ")
  }
```

---

pisa.mean *Calculates the mean*

Description

Calculates the mean for an observed variable (NOT one with plausible values).

Usage

```
pisa.mean(variable, by, data)
```

Arguments

variable	A variable label. The mean is calculated for this variable (e.g., variable="ESCS")
by	The variable label defining the grouping, usually the countries (by="CNT"), but could be any other categorical variable.
data	An R object, normally a data frame, containing the data from PISA.

Value

pisa.mean returns a data frame including the means and standard errors

Author(s)

Daniel Caro

See Also

pirls.mean, timss.mean

Examples

```
## Not run:
## Replicating Table II.1.1, p. 152 in International Report 2009
pisa.mean(variable="ESCS", by="CNT", data=student.data)
## End(Not run)


## The function is currently defined as
function (variable, by, data)
{
    mean.input <- function(variable, data) {
        meanrp <- achmrp <- sapply(1:80, function(i) weighted.mean(as.numeric(data[[variable]]),
            data[[paste("W_FSTR", i, sep = "")]], na.rm = TRUE))
        meantot <- weighted.mean(as.numeric(data[[variable]]),
            data[["W_FSTUWT"]], na.rm = TRUE)
        meanse <- (0.05 * sum((meanrp - meantot)^2))^(1/2)
        result <- data.frame(Mean = meantot, Std.err. = meanse)
        return(round(result, 3))
    }
    if (missing(by)) {
        return(mean.input(variable = variable, data = data))
    }
    else {
        for (i in by) {
            data[[c(i)]] <- as.character(data[[c(i)]])
        }
        return(ddply(data, by, function(x) mean.input(data = x,
            variable = variable)))
```

```
  }
}
```

`pisa.mean.pv` *Mean for plausible values*

Description

pisa.mean.pv uses five plausible values to calculate mean performance and its standard erro

Usage

```
pisa.mean.pv(pvlabel, by, data)
```

Arguments

`pvlabel`	The label corresponding to the achievement variable, e.g., "READ" for overall reading performance.
`by`	The variable label defining the grouping, usually the countries (by="CNT"), but could be any other categorical variable.
`data`	An R object, normally a data frame, containing the data from PISA.

Value

pisa.mean.pv returns a data frame with the mean and standard errors.

Author(s)

Daniel Caro

See Also

timss.mean.pv

Examples

```
## Not run:
## Replicating Figure I.2.16 p. 56 International Report 2009
pisa.mean.pv(pvlabel = "READ", by = "CNT", data = student.data)
## End(Not run)


## The function is currently defined as
function (pvlabel, by, data)
{
    pv.input <- function(pvlabel = "READ", data) {
        pvnames <- paste("PV", 1:5, pvlabel, sep = "")
        achmrp <- sapply(pvnames, function(k) sapply(1:80, function(i) weighted.mean(data[[k]],
            data[[paste("W_FSTR", i, sep = "")]], na.rm = TRUE)))
```

```
        achmtot <- sapply(pvnames, function(x) weighted.mean(data[[x]],
            data[["W_FSTUWT"]], na.rm = TRUE))
        achtot <- mean(achmtot)
        varw <- mean(sapply(1:5, function(i) 0.05 * sum((achmrp[,
            i] - achmtot[i])^2)))
        varb <- (1/(5 - 1)) * sum(sapply(1:5, function(i) (achmtot[i] -
            achtot)^2))
        achse <- (varw + (1 + 1/5) * varb)^(1/2)
        result <- data.frame(Mean = achtot, Std.err. = achse)
        return(round(result, 3))
    }
    if (missing(by)) {
        return(pv.input(pvlabel = pvlabel, data = data))
    }
    else {
        for (i in by) {
            data[[c(i)]] <- as.character(data[[c(i)]])
        }
        return(ddply(data, by, function(x) pv.input(data = x,
            pvlabel = pvlabel)))
    }
}
```

`pisa.reg.pv` *Regression analysis for plausible values.*

Description

pisa.reg.pv performs standard linear regression analysis (OLS) while handling plausible values and replicate weights.

Usage

```
pisa.reg.pv(x, pvlabel = "READ", by, data)
```

Arguments

`x`	Independent variables.
`pvlabel`	Data label for plausible value.
`by`	The variable label defining the grouping, usually the countries (by="CNT"), but could be any other categorical variable.
`data`	An R object, normally a data frame, containing the data from PISA.

Value

pisa.reg.pv returns a data frame with coefficients, standard errors and t-values. If "by" is specified, results are reported in a list.

Author(s)

Daniel Caro

See Also

pirls.reg.pv, timss.reg.pv

Examples

```
## Not run:
pisa.reg.pv(pvlabel="READ", x="ST04Q01", data=student.data)

## End(Not run)



## The function is currently defined as
function (x, pvlabel = "READ", by, data)
{
    reg.pv.input <- function(x, pvlabel = pvlabel, data) {
        pvnames <- paste("PV", 1:5, pvlabel, sep = "")
        regform <- lapply(pvnames, function(i) paste(i, "~",
            paste(x, collapse = "+")))
        Coefrpv <- lapply(regform, function(k) sapply(1:80, function(i) coefficients(lm(formula = as.formula(k
            data = data, weights = data[[paste("W_FSTR", i, sep = "")]])))))
        Regpv <- lapply(regform, function(i) summary(lm(formula = i,
            data = data, weights = data[["W_FSTUWT"]])))
        Coefpv <- sapply(Regpv, function(i) i[["coefficients"]][,
            1])
        R2 <- mean(sapply(Regpv, function(i) i[["r.squared"]]))
        Coeftot <- apply(Coefpv, 1, mean)
        Varw <- apply(0.05 * sapply(1:5, function(i) apply((Coefrpv[[i]] -
            Coefpv[, i])^2, 1, sum)), 1, mean)
        Varb <- (1/4) * apply(sapply(1:5, function(i) (Coefpv[,
            i] - Coeftot)^2), 1, sum)
        CoefSE <- (Varw + (1 + 1/5) * Varb)^(1/2)
        CoefT <- Coeftot/CoefSE
        RegTab <- round(data.frame(Estimate = Coeftot, `Std. Error` = CoefSE,
            `t value` = CoefT, check.names = F), 3)
        return(RegTab)
    }
    if (missing(by)) {
        return(reg.pv.input(x = x, pvlabel = pvlabel, data = data))
    }
    else {
        return(lapply(split(data, data[by]), function(i) reg.pv.input(x = x,
            pvlabel = pvlabel, data = i)))
    }
  }
```

pisa.table *Frequency table*

Description

pirls.table produces a frequency table for a categorical variable printing percentages and standard errors.

Usage

```
pisa.table(variable, by, data)
```

Arguments

variable	The data label with the variable to be analyzed.
by	The variable label defining the grouping, usually the countries (by="CNT"), but could be any other categorical variable.
data	An R object, normally a data frame, containing the data from PISA.

Value

pisa.table returns a data frame with percentages and standard errors.

Author(s)

Daniel Caro

See Also

pirls.table, timss.table

Examples

```
## Not run:
## Replicating Table A2.4b in International Report 2009
pisa.table(variable="ST01Q01", by=c("CNT", "ST04Q01"), data=student.data)

## End(Not run)


## The function is currently defined as
function (variable, by, data)
{
    table.input <- function(variable, data) {
        tabrp <- sapply(1:80, function(i) percent(as.factor(as.numeric(data[[variable]])),
            total = F, weights = data[[paste("W_FSTR", i, sep = "")]],
            na.rm = T))
        tabtot <- round(percent(as.factor(as.numeric(data[[variable]])),
```

```
            weights = data[["W_FSTUWT"]], na.rm = TRUE, total = F),
            3)
        tabse <- round(sapply(1:length(table(as.numeric(data[[variable]]))),
            function(x) (0.05 * sum(sapply(1:80, function(y) (tabrp[x,
                y] - tabtot[[x]])^2)))^(1/2)), 3)
        result <- data.frame(table(as.numeric(data[[variable]])),
            Percentage = as.numeric(tabtot), Std.err. = tabse)
        names(result)[1] <- variable
        return(result)
    }
    if (missing(by)) {
        return(table.input(variable = variable, data = data))
    }
    else {
        for (i in by) {
            data[[c(i)]] <- as.character(data[[c(i)]])
        }
        return(ddply(data, by, function(x) table.input(data = x,
            variable = variable)))
    }
}
```

`timss.ben.pv` *Calculates percentage of students at or above a benchmark*

Description

timss.ben.pv calculates the percentage of students performing at or above the levels of achievement given by the useR. It produces the results for the international PIRLS/TIMSS benchmarks by default

Usage

```
timss.ben.pv(pvlabel, cutoff = c(400, 475, 550, 625), by, data)
```

Arguments

`pvlabel`	The label corresponding to the achievement variable, e.g., "BSMMAT" for overall reading performance.
`cutoff`	The cut-off points or benchmarks for calculating the percentage of students. The default are the PIRLS/TIMSS international benchmarks: cutoff = c(400, 475, 550, 625)
`by`	The variable label defining the grouping, usually the countries (by="IDCNTRY"), but could be any other categorical variable.
`data`	An R object, normally a data frame, containing the data from TIMSS.

Value

timss.ben.pv returns a data frame with the percentage of students at or above the benchmark and the corresponding standard error.

Author(s)

Daniel Caro

See Also

pirls.ben.pv

Examples

```
## Not run:
# Exhibit 2.16, p31 User Guide
timss.ben.pv(pvlabel="BSMMAT", by="IDCNTRYL", data=timss.data)
## End(Not run)



## The function is currently defined as
function (pvlabel, cutoff = c(400, 475, 550, 625), by, data)
{
    pv.ben.input <- function(pvlabel, cutoff, data) {
        pvname <- paste(pvlabel, "0", 1:5, sep = "")
        tabpv1 <- sapply(1:length(cutoff), function(z) sapply(1:75,
            function(x) percent(data[[pvname[1]]] >= cutoff[z],
                weights = ifelse(data[["JKZONE"]] == x, 2 * data[["TOTWGT"]] *
                  data[["JKREP"]], data[["TOTWGT"]]), na.rm = TRUE,
                total = F)))
        tabpv <- sapply(1:length(cutoff), function(z) sapply(pvname,
            function(x) percent(data[[x]] >= cutoff[z], weights = data[["TOTWGT"]],
                na.rm = TRUE, total = F)))
        tabpvw <- sapply(1:length(cutoff), function(y) sum(sapply(1:75,
            function(x) (tabpv1[x, y] - tabpv[1, y])^2)))
        tabpvb <- (1 + 1/5) * apply(tabpv, 2, var)
        tabse <- round((tabpvw + tabpvb)^(1/2), 2)
        tabtot <- round(apply(tabpv, 2, mean), 2)
        result <- data.frame(Benchmark = paste(rep("At or above",
            length(cutoff)), cutoff), Percentage = tabtot, `Std. err.` = tabse,
            check.names = F)
        return(result)
    }
    if (missing(by)) {
        return(pv.ben.input(pvlabel = pvlabel, cutoff = cutoff,
            data = data))
    }
    else {
        return(ddply(data, by, function(x) pv.ben.input(pvlabel = pvlabel,
            cutoff = cutoff, data = x)))
    }
  }
```

`timss.mean` *Calculates the mean*

Description

Calculates the mean for an observed variable (NOT one with plausible values).

Usage

```
timss.mean(variable, by, data)
```

Arguments

`variable`	A variable label. The mean is calculated for this variable.
`by`	The variable label defining the grouping, usually the countries (by="IDCNTRY"), but could be any other categorical variable.
`data`	An R object, normally a data frame, containing the data from TIMSS.

Value

timss.mean returns a data frame including the means and standard errors.

Author(s)

Daniel Caro

See Also

pirls.mean

Examples

```
## Not run:
## Exhibit 2.5, p.19 User Guide 2007
timss.mean(variable="BSDAGE", by="IDCNTRYL", data=timss.data)
## End(Not run)



## The function is currently defined as
function (variable, by, data)
{
    mean.input <- function(variable, data) {
        meanrp <- sapply(1:75, function(x) weighted.mean(as.numeric(data[[variable]]),
            ifelse(data[["JKZONE"]] == x, 2 * data[["TOTWGT"]] *
                data[["JKREP"]], data[["TOTWGT"]]), na.rm = TRUE))
        meantot <- weighted.mean(as.numeric(data[[variable]]),
            data[["TOTWGT"]], na.rm = TRUE)
        meanse <- (sum((meanrp - meantot)^2))^(1/2)
```

```
        result <- data.frame(Mean = meantot, Std.err. = meanse)
        return(round(result, 2))
    }
    if (missing(by)) {
        return(mean.input(variable = variable, data = data))
    }
    else {
        return(ddply(data, by, function(x) mean.input(data = x,
            variable = variable)))
    }
}
```

`timss.mean.pv` *Calculates mean performance*

Description

timss.mean.pv uses five plausible values to calculate mean performance and its standard error

Usage

```
timss.mean.pv(pvlabel, by, data)
```

Arguments

`pvlabel`	The label corresponding to the achievement variable, e.g., "BSMMAT" for overall reading performance.
`by`	The variable label defining the grouping, usually the countries (by="IDCNTRY"), but could be any other categorical variable.
`data`	An R object, normally a data frame, containing the data from PIRLS.

Value

timss.mean.pv returns a data frame with the mean and standard errors.

Author(s)

Daniel Caro

See Also

pirls.mean.pv

Examples

```
## Not run:
## Exhibit 2.8, p.22 User Guide 2007
timss.mean.pv(pvlabel="BSMMAT", by=c("IDCNTRYL", "ITSEX"), data=timss.data)
## End(Not run)




## The function is currently defined as
function (pvlabel, by, data)
{
    pv.input <- function(pvlabel = "ASRREA", data) {
        pvnames <- paste(pvlabel, "0", 1:5, sep = "")
        read1m <- sapply(1:75, function(x) weighted.mean(data[[pvnames[[1]]]],
            ifelse(data[["JKZONE"]] == x, 2 * data[["TOTWGT"]] *
                data[["JKREP"]], data[["TOTWGT"]]), na.rm = TRUE))
        readm <- sapply(pvnames, function(x) weighted.mean(data[[x]],
            data[["TOTWGT"]], na.rm = TRUE))
        varw <- sum((read1m - readm[1])^2)
        varb <- (1 + 1/5) * var(readm)
        readse <- (varw + varb)^(1/2)
        result <- data.frame(Mean = mean(readm), Std.err. = readse)
        return(round(result, 2))
    }
    if (missing(by)) {
        return(pv.input(pvlabel = pvlabel, data = data))
    }
    else {
        return(ddply(data, by, function(x) pv.input(data = x,
            pvlabel = pvlabel)))
    }
  }
```

`timss.reg` *Regression analysis*

Description

pisa.mean performs standard linear regression analysis (OLS) for an observed depedent variable (NOT for plausible values).

Usage

```
timss.reg(regform, by, data)
```

Arguments

regform	A regression formula (e.g., "ASBGBOOK ~ ITSEX").
by	The variable label defining the grouping, usually the countries (by="IDCNTRY"), but could be any other categorical variable.
data	An R object, normally a data frame, containing the data from TIMSS.

Value

timss.reg returns a data frame with coefficients, standard errors and t-values. If "by" is specified, results are reported in a list.

Author(s)

Daniel Caro

See Also

pirls.reg

Examples

```
## Not run:
# Exhibit 2.11
timss.data$SEX[timss.data$ITSEX==2]=1
timss.data$SEX[timss.data$ITSEX==1]=0

timss.reg(regform="BSDAGE ~ SEX", by=c("IDCNTRYL"), data=timss.data)
timss.mean(variable="BSDAGE", by=c("IDCNTRYL", "ITSEX"), data=timss.data)

## End(Not run)




## The function is currently defined as
function (regform, by, data)
{
    reg.input <- function(regform, data) {
        Coefrp <- sapply(1:75, function(i) coefficients(lm(formula = as.formula(regform),
            data = data, weights = ifelse(data[["JKZONE"]] ==
                i, 2 * data[["TOTWGT"]] * data[["JKREP"]], data[["TOTWGT"]])))
        Reg <- summary(lm(formula = as.formula(regform), data = data,
            weights = data[["TOTWGT"]]))
        Coeftot <- Reg$coefficients[, 1]
        R2 <- Reg$r.squared
        CoefSE <- apply((Coefrp - Coeftot)^2, 1, sum)^(1/2)
        CoefT <- Coeftot/CoefSE
        RegTab <- round(data.frame(Estimate = Coeftot, `Std. Error` = CoefSE,
            `t value` = CoefT, check.names = F), 2)
        return(RegTab)
    }
```

```
  if (missing(by)) {
      return(reg.input(regform = regform, data = data))
  }
  else {
      return(lapply(split(data, data[by]), function(x) reg.input(regform = regform,
          data = x)))
  }
}
```

`timss.reg.pv` *Regression analysis for plausible values*

Description

pisa.reg.pv performs standard linear regression analysis (OLS) while handling plausible values and replicate weights.

Usage

```
timss.reg.pv(x, pvlabel = "ASRREA", by, data)
```

Arguments

`x`	Data labels of independent variables.
`pvlabel`	The label corresponding to the achievement variable.
`by`	The variable label defining the grouping, usually the countries (by="IDCNTRY"), but could be any other categorical variable.
`data`	An R object, normally a data frame, containing the data from TIMSS.

Value

timss.reg.pv returns a data frame with coefficients, standard errors and t-values. If "by" is specified, results are reported in a list.

Author(s)

Daniel Caro

See Also

pirls.reg.pv

Examples

```
## Not run:
## Exhibit 2.13, p.27
timss.reg.pv(pvlabel="BSMMAT", by=c("IDCNTRYL"), x="SEX", data=timss.data)
## End(Not run)


## The function is currently defined as
function (x, pvlabel = "ASRREA", by, data)
{
    reg.pv.input <- function(x, pvlabel = pvlabel, data) {
        pvnames <- paste(pvlabel, "0", 1:5, sep = "")
        regform <- lapply(pvnames, function(i) paste(i, "~",
            paste(x, collapse = "+")))
        Coefrpv1 <- sapply(1:75, function(i) coefficients(lm(formula = as.formula(regform[[1]]),
            data = data, weights = ifelse(data["JKZONE"] ==
                i, 2 * data["TOTWGT"] * data["JKREP"], data["TOTWGT"])))))
        Regpv <- lapply(regform, function(i) summary(lm(formula = i,
            data = data, weights = data["TOTWGT"]))))
        Coefpv <- sapply(Regpv, function(i) i["coefficients"][,
            1])
        R2 <- mean(sapply(Regpv, function(i) i["r.squared"]))
        Coeftot <- apply(Coefpv, 1, mean)
        Varw <- apply((Coefrpv1 - Coefpv[, 1])^2, 1, sum)
        Varb <- (1 + 1/5) * apply(Coefpv, 1, var)
        CoefSE <- (Varw + Varb)^(1/2)
        CoefT <- Coeftot/CoefSE
        RegTab <- round(data.frame(Estimate = Coeftot, `Std. Error` = CoefSE,
            `t value` = CoefT, check.names = F), 2)
        return(RegTab)
    }
    if (missing(by)) {
        return(reg.pv.input(x = x, pvlabel = pvlabel, data = data))
    }
    else {
        return(lapply(split(data, data[by]), function(i) reg.pv.input(x = x,
            pvlabel = pvlabel, data = i)))
    }
  }
```

`timss.rho` *Correlation matrix*

Description

timss.rho produces a correlations matrix for observed variables (NOT for plausible values)

Usage

```
timss.rho(variables, by, data)
```

Arguments

variables	Data labels for the variables in the correlation matrix (e.g., variables=c("ASRREA01", "ASDAGE"))
by	The variable label defining the grouping, usually the countries (by="IDCNTRY"), but could be any other categorical variable.
data	An R object, normally a data frame, containing the data from TIMSS.

Value

pirls.rho returns a matrix including correlation and standard error values

Author(s)

Daniel Caro

See Also

pirls.rho

Examples

```
## Not run:
timss.rho(variables=c("BSMMAT01", "BSDGEDUP"), data=timss.data)

## End(Not run)


## The function is currently defined as
function (variables, by, data)
{
    rho.input <- function(variables, data) {
        data <- na.omit(data[c(variables, "TOTWGT", "JKREP",
            "JKZONE")])
        rhorp <- lapply(1:75, function(i) cov.wt(data[variables],
            wt = ifelse(data[["JKZONE"]] == i, 2 * data[["TOTWGT"]] *
                data[["JKREP"]], data[["TOTWGT"]]), cor = T)[[5]])
        rhotot <- cov.wt(data[variables], wt = data[["TOTWGT"]],
            cor = T)[[5]]
        rhose <- Reduce("+", lapply(rhorp, function(x) (x - rhotot)^2))^(1/2)
        rhomat <- round(do.call(cbind, lapply(1:ncol(rhotot),
            function(x) t(rbind(rhotot[, x], rhose[, x])))),
            3)
        colnames(rhomat) <- unlist(lapply(1:length(variables),
            function(x) c(paste(variables, "Rho", sep = " ")[x],
                paste(variables, "SE", sep = " ")[x])))
        return(round(rhomat, 6))
    }
    if (missing(by)) {
        return(rho.input(variables = variables, data = data))
    }
    else {
```

```
        return(lapply(split(data, data[by]), function(x) rho.input(variables = variables,
            data = x)))
    }
  }
```

`timss.rho.pv` *Two-way weighted correlation*

Description

pirls.rho.pv calculates the correlation and standard error among two achievement variables each based on 5 plausible values or one achievement variable and an observed variable (i.e., with observed scores rather than plausible values).

Usage

```
timss.rho.pv(variable, pvlabels, by, data)
```

Arguments

`variable`	A data label for the observed variable (e.g., variable="ASDAGE")
`pvlabels`	One or two labels describing the achievement variables (e.g., pvlabels = c("ASRLIT", "ASRINF"))
`by`	The variable label defining the grouping, usually the countries (by="IDCNTRY"), but could be any other categorical variable.
`data`	An R object, normally a data frame, containing the data from TIMSS.

Value

timss.rho.pv returns a matrix with correlations and standard errors.

Author(s)

Daniel Caro

See Also

pirls.rho.pv

Examples

```
## Not run:
timss.rho.pv(variable="BSDGEDUP", pvlabel="BSMMAT", by="IDCNTRYL", data=timss.data)

## End(Not run)


## The function is currently defined as
```

```
function (variable, pvlabels, by, data)
{
    rho.pv.input <- function(variable, pvlabels, data) {
        if (length(pvlabels) == 2 & missing(variable)) {
            pvnames <- lapply(pvlabels, function(x) paste(x,
                "0", 1:5, sep = ""))
            data <- na.omit(data[c(unlist(pvnames), "TOTWGT",
                "JKREP", "JKZONE")])
            rhopvrp <- lapply(1:75, function(i) cov.wt(x = data[c(pvnames[[1]][1],
                pvnames[[2]][1])], cor = T, wt = ifelse(data[["JKZONE"]] ==
                i, 2 * data[["TOTWGT"]] * data[["JKREP"]], data[["TOTWGT"]]))[[5]])
            rhopvtot <- lapply(1:5, function(i) cov.wt(x = data[c(pvnames[[1]][i],
                pvnames[[2]][i])], cor = T, wt = data[["TOTWGT"]])[[5]])
            varw <- Reduce("+", lapply(rhopvrp, function(x) (x -
                rhopvtot[[1]])^2))
            varb <- (1 + 1/5) * apply(simplify2array(rhopvtot),
                c(1, 2), var)
            rhose <- (varw + varb)^(1/2)
            rhotot <- Reduce("+", rhopvtot)/length(rhopvtot)
            rhomat <- round(do.call(cbind, lapply(1:ncol(rhotot),
                function(x) t(rbind(rhotot[, x], rhose[, x])))),
                5)
            colnames(rhomat) <- unlist(lapply(1:2, function(i) c(paste(pvlabels,
                "Rho", sep = " ")[i], paste(pvlabels, "SE", sep = " ")[i])))
            return(round(rhomat, 3))
        }
        else {
            pvnames <- paste(pvlabels, "0", 1:5, sep = "")
            data <- na.omit(data[c(variable, pvnames, "TOTWGT",
                "JKREP", "JKZONE")])
            rhopvrp <- lapply(1:75, function(i) cov.wt(x = data[c(variable,
                pvnames[1])], cor = T, wt = ifelse(data[["JKZONE"]] ==
                i, 2 * data[["TOTWGT"]] * data[["JKREP"]], data[["TOTWGT"]]))[[5]])
            rhopvtot <- lapply(pvnames, function(i) cov.wt(x = data[c(variable,
                i)], cor = T, wt = data[["TOTWGT"]])[[5]])
            varw <- Reduce("+", lapply(rhopvrp, function(x) (x -
                rhopvtot[[1]])^2))
            varb <- (1 + 1/5) * apply(simplify2array(rhopvtot),
                c(1, 2), var)
            rhose <- (varw + varb)^(1/2)
            rhotot <- Reduce("+", rhopvtot)/length(rhopvtot)
            rhomat <- round(do.call(cbind, lapply(1:ncol(rhotot),
                function(x) t(rbind(rhotot[, x], rhose[, x])))),
                5)
            colnames(rhomat) <- unlist(lapply(1:2, function(i) c(paste(c(variable,
                pvlabels), "Rho", sep = " ")[i], paste(c(variable,
                pvlabels), "SE", sep = " ")[i])))
            return(round(rhomat, 3))
        }
    }
    if (missing(by)) {
        return(rho.pv.input(variable = variable, pvlabels = pvlabels,
            data = data))
```

```
    }
    else {
        if (length(pvlabels) == 2 & missing(variable)) {
            return(lapply(split(data, data[by]), function(x) rho.pv.input(pvlabels = pvlabels,
                data = x)))
        }
        else {
            return(lapply(split(data, data[by]), function(x) rho.pv.input(variable = variable,
                pvlabels = pvlabels, data = x)))
        }
    }
}
```

`timss.select.merge` *Select and merge data*

Description

timss.select.merge selects and merges data from TIMSS 2007 downloaded from the IEA data repository

Usage

```
timss.select.merge(folder = getwd(), countries, student, school, math.teacher, science.teacher)
```

Arguments

`folder`	Directory path where the data are located. The data could be organized within folders but it should not be duplicated.
`countries`	The selected countries, supplied with the abbreviation (e.g., countries=c("AUT", "BGR") or codes (countries=c(40, 100)). If no countries are selected, all are selected.
`student`	The data labels for the selected student variables.
`school`	The data labels for the selected school variables.
`math.teacher`	The data labels for the math teacher data.
`science.teacher`	The data labels for the science teacher data.

Value

timss.select.merge returns a data frame with the selected data from PIRLS 2006.

Author(s)

Daniel Caro

See Also

pirls.select.merge

Examples

```
## Not run:

timss.data <- timss.select.merge(folder="/home/eldani/Work/R work/TIMSS/Data",
              student =c("BSDGEDUP", "ITSEX", "BSDAGE"), countries=c("DZA", "ARM", "AUS", "BHR"),
              school =c("BCDSRMI", "BCDGPPSC"))

## End(Not run)



## The function is currently defined as
function (folder = getwd(), countries, student, school, math.teacher,
    science.teacher)
{
    files.all <- lapply(c("bsg", "bcg", "btm", "bts"), function(x) list.files(folder,
        full.names = T, pattern = paste("^", x, ".*.sav$", sep = ""),
        recursive = T))
    cntlab <- toupper(unique(unlist(lapply(files.all, function(x) substr(x,
        nchar(x) - 8, nchar(x) - 6)))))
    country.list <- country.ug[country.ug$ISO %in% intersect(country.ug$ISO,
        cntlab), ]
    if (missing(countries)) {
        countries = cntlab
    }
    if (is.numeric(countries)) {
        countries = country.list[country.list$Code %in% countries,
            "ISO"]
    }
    files.select <- lapply(files.all, function(y) sapply(countries,
        function(x) y[substr(y, nchar(y) - 8, nchar(y) - 6) ==
            tolower(x)]))
    student.data <- do.call("rbind", lapply(files.select[[1]],
        function(y) read.spss(y, use.value.labels = F, to.data.frame = T)[c("IDCNTRY",
            "IDSCHOOL", "IDCLASS", "IDSTUD", "JKREP", "JKZONE",
            "HOUWGT", "SENWGT", "TOTWGT", "BSMMAT01", "BSMMAT02",
            "BSMMAT03", "BSMMAT04", "BSMMAT05", "BSSSCI01", "BSSSCI02",
            "BSSSCI03", "BSSSCI04", "BSSSCI05", student)]))
    student.data$IDCNTRYL <- factor(student.data$IDCNTRY, levels = country.list[country.list$Code %in%
        unique(student.data$IDCNTRY), "Code"], labels = country.list[country.list$Code %in%
        unique(student.data$IDCNTRY), "Country"])
    school.data <- do.call("rbind", lapply(files.select[[2]],
        function(y) read.spss(y, use.value.labels = F, to.data.frame = T)[c("IDCNTRY",
            "IDSCHOOL", "SCHWGT", school)]))
    timss.all <- merge(student.data, school.data, by = c("IDCNTRY",
        "IDSCHOOL"), suffixes = c(".st", ".sc"))
    return(timss.all)
  }
```

timss.var.label *Data labels*

Description

timss.var.labels prints and saves variable labels in a text file

Usage

```
timss.var.label(folder = getwd())
```

Arguments

folder	Directory path where the data are located. The data could be organized within folders but it should not be duplicated.

Value

timss.var.label returns a list with variable labels for the student, home, and school data. The output is also printed on a text sheet located in "folder".

Author(s)

Daniel Caro

See Also

pirls.var.label

Examples

```
## Not run:
timss.var.label(folder= getwd())

## End(Not run)

## The function is currently defined as
function (folder = getwd())
{
    files.all <- lapply(c("bsg", "bcg", "btm", "bts"), function(x) list.files(folder,
        full.names = T, pattern = paste("^", x, ".*.sav$", sep = ""),
        recursive = T))
    var.label <- list(`Student background and achievement` = description(spss.system.file(files.all[[1]][1])),
        `School data` = description(spss.system.file(files.all[[2]][1])),
        `Math teacher` = description(spss.system.file(files.all[[3]][1])),
        `Science teacher` = description(spss.system.file(files.all[[4]][1])))
    print(var.label)
    capture.output(var.label, file = file.path(folder, "Variable labels.txt"))
    cat("The file 'Variable labels.txt' in the directory", folder,
```

```
      "contains the variable labels of the complete dataset",
      sep = " ")
}
```

Index

∗Topic **datasets**
country.ug, 3

country.ug, 3

intsvy *(*intsvy-package*)*, 2
intsvy-package, 2

pirls.ben.pv, 3
pirls.mean, 5
pirls.mean.pv, 6
pirls.reg, 7
pirls.reg.pv, 9
pirls.rho, 10
pirls.rho.pv, 12
pirls.select.merge, 14
pirls.table, 16
pirls.var.label, 17
pisa.mean, 18
pisa.mean.pv, 20
pisa.reg.pv, 21
pisa.table, 23

timss.ben.pv, 24
timss.mean, 26
timss.mean.pv, 27
timss.reg, 28
timss.reg.pv, 30
timss.rho, 31
timss.rho.pv, 33
timss.select.merge, 35
timss.var.label, 37