

# Package ‘CompositeRegressionEstimation’

April 27, 2020

**Type** Package

**Title** X

**Version** 1.0

**Date** 2020-04-23

**Author** D. Bonnery

**Maintainer** D. Bonnery <dbonnery@umd.edu>

**Imports** ggplot2,  
abind,  
optimx,  
Matrix,  
Hmisc,  
MASS,  
filehash

**Suggests**

**Description** Data

**Remotes** DanielBonnery/arrayproduct

**Depends** arrayproduct,  
sampling,  
abind,  
optimx,  
Hmisc,  
MASS,  
filehash

**License** GPL (>= 2)

**LazyLoad** yes

**LazyData** true

**RoxygenNote** 7.0.2

## R topics documented:

add.rg . . . . . 2

add.rg3	3
AK	4
AK_est	6
CoeffGM	8
CoeffGM.array	8
CoeffGM.matrix	9
CoeffS2	10
composite	10
CPS_AK	12
CPS_AK_coeff.array.fl	13
CPS_AK_est	13
CPS_A_e	14
CPS_A_u	14
CPS_K_e	15
CPS_K_u	15
CPS_Xplus_array	15
CPS_Xplus_matrix	16
CPS_X_array	17
CPS_X_matrix	17
douuble	18
empirical.var	19
factorisedf	19
MR	20
varAK3	21
varAK3diff	22
varAK3diffrat	22
varAK3rat	23
var_lin	24
W.ak	24
W.multi.ak	25
W.rec	26
WS	27
WSrg	28
WSrg2	29
<b>Index</b>	<b>30</b>

---

add.rg

*Add a rotation group indicator to all tables of a list when missing.*

---

## Description

Add a rotation group indicator to all tables of a list when missing.

## Usage

```
add.rg(list.tables, id, rg.name)
```

**Arguments**

<code>list.tables</code>	a list of data.frames (order matter)
<code>id</code>	a vector of character strings indicating the variable names for the sample unit primary key.
<code>rg.name</code>	a character string

**Value**

a list of data.frames with a new variable named `rg.name`

---

<code>add.rg3</code>	<i>Add a rotation group indicator to a table indicating wheter a unit is present in the previous and next samples.</i>
----------------------	--

---

**Description**

Add a rotation group indicator to a table indicating wheter a unit is present in the previous and next samples.

**Usage**

```
add.rg3(df_1, df0, df1, id, rg.name = "rg")
```

**Arguments**

<code>df_1</code>	a data frame, the previous table
<code>df0</code>	a data frame, the current table
<code>df1</code>	a data frame, the next table
<code>id</code>	a vector of character strings indicating the variable names for the sample unit primary key.
<code>rg.name</code>	a character string

**Details**

creates a variable named `rg.name` that takes values 4 for elements present in the current and next tables only, 3 for elements present in the current table only, 2 for elements present in the previous, current and next tables, 1 for elements present in the previous and current tables only.

depends on `dplyr`, `tidyr`

**Value**

a list of data.frames with a new variable named `rg.name`

### Examples

```
df <- expand.grid(x= 1:10, y = 1:10)
df_1 <- df[sample(100,25),]
df0 <- df[sample(100,25),]
df1 <- df[sample(100,25),]
id=c("x", "y")
add.rg3(df_1, df0, df1, c("x", "y"))
```

AK

AK Estimator (recursive version)

### Description

Consider a sequence of monthly samples  $(S_m)_{m \in \{1, \dots, M\}}$ . In the CPS, a sample  $S_m$  is the union of 8 rotation groups:  $S_m = S_{m,1} \cup S_{m,2} \cup S_{m,3} \cup S_{m,4} \cup S_{m,5} \cup S_{m,6} \cup S_{m,7} \cup S_{m,8}$ , where two consecutive samples are always such that  $S_{m,2} = S_{m-1,1}$ ,  $S_{m,3} = S_{m-1,2}$ ,  $S_{m,4} = S_{m-1,3}$ ,  $S_{m,6} = S_{m-1,5}$ ,  $S_{m,7} = S_{m-1,6}$ ,  $S_{m,8} = S_{m-1,7}$ , and one year appart samples are always such that  $S_{m,5} = S_{m-12,1}$ ,  $S_{m,6} = S_{m-12,2}$ ,  $S_{m,7} = S_{m-12,3}$ ,  $S_{m,8} = S_{m-12,4}$ .

The subsamples  $S_{m,g}$  are called rotation groups, and rotation patterns different than the CPS rotation pattern are possible.

For each individual  $k$  of the sample  $m$ , one observes the employment status  $Y_{k,m}$  (A binary variable) of individual  $k$  at time  $m$ , and the survey weight  $w_{k,m}$ , as well as its "rotation group".

The AK composite estimator is defined in "CPS Technical Paper (2006), [section 10-11]":

For  $m = 1$ ,  $\hat{t}_{Y,1} = \sum_{k \in S_1} w_{k,1} Y_{k,1}$ .

For  $m \geq 2$ ,

$$\hat{t}_{Y,m} = (1 - K) \times \left( \sum_{k \in S_m} w_{k,m} Y_{k,m} \right) + K \times (\hat{t}_{Y,m-1} + \Delta_m) + A \times \hat{\beta}_m$$

where

$$\Delta_m = \eta_0 \times \sum_{k \in S_m \cap S_{m-1}} (w_{k,m} Y_{k,m} - w_{k,m-1} Y_{k,m-1})$$

and

$$\hat{\beta}_m = \left( \sum_{k \notin S_m \cap S_{m-1}} w_{k,m} Y_{k,m} \right) - \eta_1 \times \left( \sum_{k \in S_m \cap S_{m-1}} w_{k,m} Y_{k,m} \right)$$

For the CPS,  $\eta_0$  is the ratio between the number of rotation groups in the sample and the number of overlapping rotation groups between two month, which is a constant  $\eta_0 = 4/3$ ;  $\eta_1$  is the ratio between the number of non overlapping rotation groups the number of overlapping rotation groups between two month, which is a constant of  $1/3$ .

In the case of the CPS, the rotation group one sample unit belongs to in a particular month is a function of the number of times it has been selected before, including this month, and so the rotation group of an individual in a particular month is called the "month in sample" variable.

For the CPS, in month  $m$  the overlap  $S_{m-1} \cap S_m$  correspond to the individuals in the sample  $S_m$  with a value of month in sample equal to 2,3,4, 6,7 or 8. The overlap  $S_{m-1} \cap S_m$  correspond to the individuals in the sample  $S_m$  with a value of month in sample equal to 2,3,4, 6,7 or 8. as well as individuals in the sample  $S_{m-1}$  with a value of month in sample equal to 1,2,3, 5,6 or 7. When parametrising the function, the choice would be `group_1=c(1:3,5:7)` and `group0=c(2:4,6:8)`.

Computing the estimators recursively is not very efficient. At the end, we get a linear combinaison of month in sample estimates The functions AK3, and WSrg computes the linear combination directly and more efficiently.

### Usage

```
AK(
  list.tables,
  w,
  list.y,
  id = NULL,
  groupvar = NULL,
  groups_1 = NULL,
  groups0 = NULL,
  A = 0,
  K = 0,
  dft0.y = NULL,
  eta0 = 0,
  eta1 = 0
)
```

### Arguments

<code>list.tables</code>	a list of tables
<code>w</code>	a character string: name of the weights variable (should be the same in all tables)
<code>list.y</code>	a vector of variable names
<code>id</code>	a character string: name of the identifier variable (should be the same in all tables)
<code>groupvar</code>	a character string: name of the rotation group variable (should be the same in all tables)
<code>groups_1</code>	a character string:
<code>groups0</code>	if <code>groupvar</code> is not null, a vector of possible values for <code>L[[groupvar]]</code>
<code>eta0</code>	a numeric value
<code>eta1</code>	a numeric value

### Details

the function is based on the more general function `CompositeRegressionEstimation::composite`

## References

“CPS Technical Paper (2006). Design and Methodology of the Current Population Survey. Technical Report 66, U.S. Census Bureau.”, “Gurney, M. and Daly, J. F. (1965). A multivariate approach to estimation in periodic sample surveys. In Proceedings of the Social Statistics Section, American Statistical Association, volume 242, page 257.”

## See Also

CompositeRegressionEstimation::composite

## Examples

```
library(dataCPS)
data(cps200501,cps200502,cps200503,cps200504,
     cps200505,package="dataCPS")
list.tables<-list(cps200501,cps200502,cps200503,cps200504,
                  cps200505)
w="pwsswgt";id=c("hrhhid","pulineno");groupvar=NULL;list.y="pemlr";dft0.y=NULL;
groups_1=NULL;
groups0=NULL;
Coef=c(alpha_1=0,alpha0=1,beta_1=0,beta0=0,gamma_1=0)
AK(list.tables,w=w,list.y="pemlr",id=id,groupvar=groupvar)

## With the default choice of parameters for A,K,eta0,eta1
## the composite is equal to the direct estimator: we check
WS(list.tables = list.tables,weight = w,list.y = list.y)

## Example of use of a group variable.
w="pwsswgt";id=NULL;groupvar="hrmis";list.y="pemlr";dft0.y=NULL;
groups_1=c(1:3,5:7);
groups0=c(2:4,6:8);
Coef=c(alpha0=1,alpha_1=0,beta_1=0,beta0=0,gamma_1=0)
AK(list.tables,w=w,list.y="pemlr",id=id,groupvar="hrmis")
```

---

AK\_est

*AK estimation on array of month in sample estimates*

---

## Description

AK estimation on array of month in sample estimates

## Usage

```
AK_est(
  Y,
  month = names(dimnames(Y))[1],
  group = names(dimnames(Y))[2],
  variable = names(dimnames(Y))[3],
  S,
```

```

    S_1 = S - 1,
    a,
    k,
    groups = dimnames(Y)[[group]],
    eta0 = length(groups)/length(S),
    eta1 = eta0 - 1
  )

```

### Arguments

Y	an array of named dimensions with 3 dimensions: 1 for the month, 1 for the month in sample, 1 for the variable name
month	: name of the month dimension (by default the name of the first dimension of Y names(dimnames(dim(Y)))[1])
group	: name of the group dimension of Y (by default the name of the second dimension of Y names(dimnames(dim(Y)))[2])
S	a vector of integers, subvector of 1:ngroup, to be passed to W.ak, indicating the rotation group numbers this month that were present the previous months (for CPS, c(2:4,6:8))
a	a numeric value
k	a numeric value
eta0	a numeric value to be passed to W.ak
eta1	a numeric value to be passed to W.ak

### Value

an array

### Examples

```

library(dataCPS)
period=200501:200512
list.tables<-lapply(data(list=paste0("cps",period),package="dataCPS"),get);
names(list.tables)<-period
Y<-WSrg(list.tables,weight="pwsswgt",list.y="pemlr",rg="hrmis")
dimnames(Y);
month="m";
group="mis";
variable="y";
A=W.ak(months = dimnames(Y)[[month]],
      groups = dimnames(Y)[[group]],
      S=c(2:4,6:8),
      a=.5,
      k=.5,
      eta0=4/3,
      eta1=1/3)
ngroup=dim(Y)[group];
eta1=eta0-1;
eta0=ngroup/length(S)

```

```
AK_est(Y=Y,
      month="m",
      group="mis",
      S=c(2:4,6:8),
      a=.5,
      k=.6,
      eta0=eta0,
      eta1=eta0-1)
```

---

CoeffGM	<i>Compute Gauss Markov coefficient for CPS, matrix version</i>
---------	---

---

**Description**

Compute Gauss Markov coefficient for CPS, matrix version

**Usage**

CoeffGM(Sigma, nmonth = dim(Sigma)[[1]])

**Arguments**

Sigma                    a Variance covariance array

**Value**

a matrix.

**Examples**

CoeffGM(var())

---

CoeffGM.array	<i>Compute the Gauss Markov coefficients for Multivariate Blue</i>
---------------	--

---

**Description**

Compute the Gauss Markov coefficients for Multivariate Blue

**Usage**

CoeffGM.array(Sigma, X, Xplus = NULL)

**Arguments**

Sigma                    a (p\_1x...x p\_P) x (p\_1x...x p\_P) array  
X                        an (p\_1x...x p\_P) x (n\_1 x ...x n\_N) array  
Xplus:                   a general inverse of X



**Value**

the coefficients matrix  $W$  such that  $WY$  is the best unbiased linear estimator of  $\beta$  where  $E[Y] = X\beta$

**Examples**

```
beta= matrix(rchisq(12,1),4,3)
dimnames(beta)<-list(m=paste(200501:200504),y=c("e","u","n"))
X<-CPS_X_array(months=list(m=paste(200501:200504)),
               vars=list(y=c("e","u","n")),
               rgs=list(hrmis=paste(1:8)))
Xplus<-CPS_Xplus_array(months=list(m=paste(200501:200504)),
                      vars=list(y=c("e","u","n")),
                      rgs=list(hrmis=paste(1:8)),1/2)
EY<-arrayproduct::"%.%"(X,beta,I_A=list(c=integer(0),n=c("m","y","hrmis"),p=c("m2","y2")),I_B=list(c=integer(0),n=c("m2","y2")))
set.seed(1)
Sigma=rWishart(1,length(EY),diag(length(EY)))
Y<-array(mvrnorm(n = 100,mu = c(EY),Sigma = Sigma[, ,1]),c(100,dim(EY)))
dimnames(Y)<-c(list(rep=1:100),dimnames(EY))
Sigma.A<-array(Sigma,c(dim(EY),dim(EY)))
dimnames(Sigma.A)<-rep(dimnames(EY),2);names(dimnames(Sigma.A))[4:6]<-paste0(names(dimnames(Sigma.A))[4:6],"2")
W<-CoeffGM.array(Sigma.A,X,Xplus)
WY<-arrayproduct::"%.%"(W,Y,I_A=list(c=integer(0),n=c("y2","m2"),p=c("m","y","hrmis")),I_B=list(c=integer(0),n=c("m2","y2")))
DY<-arrayproduct::"%.%"(Xplus,Y,I_A=list(c=integer(0),n=c("y2","m2"),p=c("m","y","hrmis")),I_B=list(c=integer(0),n=c("m2","y2")))
plot(c(beta),c(apply(DY,1:2,var)),col="red")
plot(c(beta),c(apply(WY,1:2,var)))
```

CoeffGM.matrix

*Compute the Gauss Markov coefficients for Multivariate Blue for arrays*

**Description**

Compute the Gauss Markov coefficients for Multivariate Blue for arrays

**Usage**

```
CoeffGM.matrix(Sigma, X, Xplus = MASS::ginv(X))
```

**Arguments**

Sigma	a $p \times p$ matrix
X	an $n \times p$ matrix
Xplus:	a general inverse of X array

**Value**

the coefficients matrix  $W$  such that  $WY$  is the best unbiased linear estimator of  $\beta$  where  $E[Y] = X\beta$

Examples

```
A=array(rnorm(prod(2:5)),2:5);M=a2m(A,2);dim(A);dim(M);dim(a2m(A))
```

---

CoeffS2	<i>Compute the coefficients for Direct</i>
---------	--

---

Description

Compute the coefficients for Direct

Usage

```
CoeffS2(nmonth)
```

Arguments

- Sigma            a p x p matrix
- X                an n x p matrix
- Xplus:           a general inverse of X

Value

the coefficients matrix  $W$  such that  $WY$  is the best unbiased linear estimator of  $\beta$  where  $E[Y]=X\beta$

Examples

```
A=array(rnorm(prod(2:5)),2:5);M=a2m(A,2);dim(A);dim(M);dim(a2m(A))
```

---

composite	<i>Linear Composite Estimator from overlap and non overlapping consecutive subsamples direct totals</i>
-----------	---

---

Description

Consider a sequence of monthly samples  $(S_m)_{m \in \{1, \dots, M\}}$ . For each individual  $k$  of the sample  $m$ , one observes the employment status  $Y_{k,m}$  (A binary variable) of individual  $k$  at time  $m$ , and the survey weight  $w_{k,m}$ . The following program allows to compute recursively for  $m = 1, \dots, M$  the Census composite estimator of the total of  $Y_{.,m}$  with coefficients defined recursively as follows:  
For  $m = 1$ ,  $\hat{t}_{Y.,1} = \sum_{k \in S_1} w_{k,1} Y_{k,1}$ .

For  $m \geq 2$ ,

$$\hat{t}_{Y.,m} = \begin{bmatrix} \hat{t}_{Y.,m-1} \\ \sum_{k \in S_m} w_{k,m} Y_{k,m} \\ \sum_{k \in S_{m-1} \cap S_m} w_{k,m-1} Y_{k,m-1} \\ \sum_{k \in S_{m-1} \cap S_m} w_{k,m} Y_{k,m} \\ \sum_{k \in S_m \setminus S_{m-1}} w_{k,m} Y_{k,m} \end{bmatrix}^T \times \begin{bmatrix} \alpha_{(-1)} \\ \alpha_0 \\ \beta_{(-1)} \\ \beta_0 \\ \gamma_0 \end{bmatrix}$$

This function computes the estimators for given values of  $\alpha, \beta, \gamma$ .

An example of use of such estimate is the Census Bureau AK estimator: it is a special case of this estimator, with the values of  $\alpha, \beta, \gamma$  that are given as a function of two parameters A and K:

$$\begin{bmatrix} \alpha_{(-1)} \\ \alpha_0 \\ \beta_{(-1)} \\ \beta_0 \\ \gamma_0 \end{bmatrix} = \begin{bmatrix} K \\ 1 - K \\ -4K/3 \\ (4K - A)/3 \\ A \end{bmatrix}$$

for more references, please refer to the function `CompositeRegressionEstimation::AK`.

See ‘‘CPS Technical Paper (2006). Design and Methodology of the Current Population Survey. Technical Report 66, U.S. Census Bureau.’’

$$\begin{aligned} \hat{t}_{Y.,m} = & \quad K & \times \hat{t}_{Y.,m-1} \\ & + (1 - K) & \times \sum_{k \in S_m} w_{k,m} Y_{k,m} \\ & + (-4K/3) & \times \sum_{k \in S_{m-1} \cap S_m} w_{k,m-1} Y_{k,m-1} \\ & + (4K - A)/3 & \times \sum_{k \in S_{m-1} \cap S_m} w_{k,m} Y_{k,m} \\ & + A & \times \sum_{k \in S_m \setminus S_{m-1}} w_{k,m} Y_{k,m} \end{aligned}$$

Computing the estimators recursively is not very efficient. At the end, we get a linear combinaison of month in sample estimates. The functions `AK3`, and `WSrg` computes the linear combination directly and more efficiently.

For the CPS, in month  $m$  the overlap  $S_{m-1} \cap S_m$  correspond to the individuals in the sample  $S_m$  with a value of month in sample equal to 2,3,4, 6,7 or 8. The overlap  $S_{m-1} \cap S_m$  correspond to the individuals in the sample  $S_m$  with a value of month in sample equal to 2,3,4, 6,7 or 8. as well as individuals in the sample  $S_{m-1}$  with a value of month in sample equal to 1,2,3, 5,6 or 7. When parametrising the function, the choice would be `group_1=c(1:3,5:7)` and `group0=c(2:4,6:8)`.

## Usage

```
composite(
  list.tables,
  w,
  list.y,
  id = NULL,
  groupvar = NULL,
  groups_1 = NULL,
  groups0 = NULL,
  Coef = c(alpha_1 = 0, alpha0 = 1, beta_1 = 0, beta0 = 0, gamma0 = 0),
  dft0.y = NULL
)
```

**Arguments**

<code>list.tables</code>	a list of tables
<code>w</code>	a character string: name of the weights variable (should be the same in all tables)
<code>list.y</code>	a vector of variable names
<code>id</code>	a character string: name of the identifier variable (should be the same in all tables)
<code>groupvar</code>	a character string: name of the rotation group variable (should be the same in all tables)
<code>groups_1</code>	a character string:
<code>groups0</code>	if <code>groupvar</code> is not null, a vector of possible values for <code>L[[groupvar]]</code>

**See Also**

`CompositeRegressionEstimation::AK`

**Examples**

```
library(dataCPS)
data(cps200501, cps200502, cps200503, cps200504,
     cps200505, package="dataCPS")
list.tables<-list(cps200501, cps200502, cps200503, cps200504,
                  cps200505)
w="pwsswgt";id=c("hrhhid", "pulineno");groupvar=NULL;list.y="pemlr";dft0.y=NULL;
groups_1=NULL;groups0=NULL;Coef=c(alpha_1=0, alpha0=1, beta_1=0, beta0=0, gamma0=0)
composite(list.tables, w=w, list.y="pemlr", id=id, groupvar=groupvar)
##With the default choice of parameters for \code{Coef}, the composite is equal to the direct estimator: we check
WS(list.tables = list.tables, weight = w, list.y = list.y)
## Example of use of a group variable.
w="pwsswgt";id=NULL;groupvar="hrmis";list.y="pemlr";dft0.y=NULL;
groups_1=c(1:3, 5:7);groups0=c(2:4, 6:8);Coef=c(alpha0=1, alpha_1=0, beta_1=0, beta0=0, gamma0=0)
composite(list.tables, w=w, list.y="pemlr", id=id, groupvar=groupvar)
```

---

CPS\_AK

*Gives A,K coefficient for unemployed used by the Census*

---

**Description**

Gives A,K coefficient for unemployed used by the Census

**Usage**

`CPS_AK()`

**Value**

The vector `c(a1=CPS_A_u(), a2=CPS_A_e(), a3=0, k1=CPS_K_u(), k2=CPS_K_e(), k3=0)`

---

CPS\_AK\_coeff.array.fl    *Empirical variance of a collection of arrays.*

---

### Description

Empirical variance of a collection of arrays.

### Usage

```
CPS_AK_coeff.array.fl(
  nmonth,
  ak = list(c(a_1 = 0, a_2 = 0, a_3 = 0, k_1 = 0, k_2 = 0, k_3 = 0)),
  simplify = TRUE,
  statuslabel = c("0", "1", "_1")
)
```

### Arguments

nmonth	a strictly positive integer
ak,	a list of numeric vectors of length 6.
simplify	a boolean
statuslabel	: a character vector of dimension 3 indicating the label for unemployed, employed, not in the labor force.

### Examples

```
CPS_AK_coeff.array.fl()
```

---

CPS_AK_est	<i>Gives the variance of the AK estimators from the A,K coefficients and the variance covariance matrix of the month in sample estimates</i>
------------	--

---

### Description

Gives the variance of the AK estimators from the A,K coefficients and the variance covariance matrix of the month in sample estimates

### Usage

```
CPS_AK_est(
  mistotals,
  coeff = CPS_AK_coeff.array.fl(dim(mistotals)[1], ak, simplify = FALSE),
  ak = CPS_AK()
)
```

Arguments

- mistotals      An array of dimension nmonth x 8 x 3. mistotals[i,j,k] is the month in sample direct estimate for month i, month in sample j rotation group, and variable k.
- coeff      An array of coefficients W[ak,y2,m2,y1,mis1,m1] such that AK estimate for coefficients ak, month m2 and employment status y2 is sum(W[ak,y2,m2,,])\*Y[,,] where mistotals[y1,mis1,m1] is direct estimate on mis mis1 for emp stat y1 at month m1.
- ak:      an ak coefficients vector or a list of ak coefficients.

Value

The variance of the AK estimators from the A,K coefficients and the variance covariance matrix .

Examples

```
varAK3(ak=c(a1=.3,a2=.4,a3=0,k1=.4,k2=.7,k3=0), Sigma=array(drop(stats::rWishart(1,df=3*10*8,diag(3*10*8))),rep
```

---

CPS_A_e	<i>Gives K coefficient for unemployed used by the Census</i>
---------	--

---

Description

Gives K coefficient for unemployed used by the Census

Usage

```
CPS_A_e()
```

Value

.4

---

CPS_A_u	<i>Gives K coefficient for unemployed used by the Census</i>
---------	--

---

Description

Gives K coefficient for unemployed used by the Census

Usage

```
CPS_A_u()
```

Value

.3

---

CPS\_K\_e

*Gives K coefficient for unemployed used by the Census*


---

**Description**

Gives K coefficient for unemployed used by the Census

**Usage**

CPS\_K\_e()

**Value**

.7

---

CPS\_K\_u

*Gives K coefficient for unemployed used by the Census*


---

**Description**

Gives K coefficient for unemployed used by the Census

**Usage**

CPS\_K\_u()

**Value**

.4

---

CPS\_Xplus\_array

*Compute the Moore penrose general inverse of a the Yansaneh Fuller X matrix for CPS, array version*


---

**Description**

Compute the Moore penrose general inverse of a the Yansaneh Fuller X matrix for CPS, array version

**Usage**

CPS\_Xplus\_array(months, vars, rgs, alpha = 1/length(rgs[[1]]))

**Arguments**

months	a named list with one element, this element being a character string vector
vars	a named list with one element, this element being a character string vector
rgs	a named list with one element, this element being a character string vector
alpha	a numeric value

**Value**

an array.

**Examples**

```
X<-CPS_X_array(months=list(m=paste(200501:200504)),
               vars=list(y=c("e","u","n")),
               rgs=list(hrmis=paste(1:8)),1/2)
Xplus<-CPS_Xplus_array(months=list(m=paste(200501:200504)),
                      vars=list(y=c("e","u","n")),
                      rgs=list(hrmis=paste(1:8)),1/2)
arrayproduct::"%.%"(Xplus,X,
  I_A=list(c=integer(0),n=c("y2","m2"),p=c("y","hrmis","m")),
  I_B=list(c=integer(0),p=c("y","hrmis","m"),q=c("y2","m2")))
```

---

CPS_Xplus_matrix	<i>Compute the Moore penrose general inverse of a the Yansaneh Fuller X matrix for CPS</i>
------------------	--

---

**Description**

Compute the Moore penrose general inverse of a the Yansaneh Fuller X matrix for CPS

**Usage**

```
CPS_Xplus_matrix(X)
```

**Arguments**

nmonth	an integer, the number of months
nvar	an integer, the number of variables
nrg	an integer, the number of rotation groups
alpha	a coefficient

**Value**

a matrix.

**Examples**

```
CPS_Xplus_matrix(10)
```



---

CPS_X_array	<i>Compute X matrix for CPS, array version</i>
-------------	--

---

**Description**

Compute X matrix for CPS, array version

**Usage**

```
CPS_X_array(months, vars, rgs, alpha = 1/length(rgs[[1]]))
```

**Arguments**

months	a named list with one element, this element being a character string vector
vars	a named list with one element, this element being a character string vector
rgs	a named list with one element, this element being a character string vector
alpha	(default 1/length(rgs[[1]])) a numeric value

**Value**

an array.

**Examples**

```
X<-CPS_X_array(months=list(m=paste(200501:200504)),
               vars=list(y=c("e","u","n")),
               rgs=list(hrmis=paste(1:8)))
dimnames(X)
```

---

CPS_X_matrix	<i>X matrix for the simple month in sample model</i>
--------------	--

---

**Description**

X matrix for the simple month in sample model

**Usage**

```
CPS_X_matrix(nmonth, nvar, nrg, alpha = 1)
```

**Arguments**

nmonth	an integer, the number of months
nvar	an integer, the number of variables
nrg	an integer, the number of rotation groups
alpha=1/nrg	a coefficient

**Value**

a matrix.

**Examples**

```
CPS_X_matrix(10,3,8,1/8)
```

---

douuble	<i>Compute weighted sums</i>
---------	------------------------------

---

**Description**

Compute weighted sums

**Usage**

```
douuble(list.tables, w, id, y)
```

**Arguments**

- list.tables      A list of dataframes, order matters.
- w                either a real number or a character string indicating the name of the weight variable.
- id                primary key of the tables, used to merge tables together.
- y                a string indicating the name of a factor variable common to all tables of list.tables.

**Value**

a list of three arrays.

**Examples**

```
douuble(list.tables=lapply(1:10,function(x){cbind(id=1:nrow(Orange),Orange)[sample(nrow(Orange),30),]}),w="circumference",id="id",y="circumference")
```

---

empirical.var	<i>Empirical variance of a collection of arrays.</i>
---------------	--

---

**Description**

Empirical variance of a collection of arrays.

**Usage**

```
empirical.var(A, MARGIN, n)
```

**Arguments**

A	An array of dimension d_1 x ... d_p
MARGIN	a vector of integers
n	the array of dimension a_1 x ... x a_n $Y[i_1,...,i_n]=\text{sum}(W[i_1,...,i_n,...])$

**Examples**

```
empirical.var()
```

---

factorisedf	<i>Convert variables to numeric in dataframe.</i>
-------------	---

---

**Description**

Convert variables to numeric in dataframe.

**Usage**

```
factorisedf(dfr, list.y)
```

**Arguments**

dfr	A dataframe
list.y	character vector containing the names of the variables to be converted.

**Value**

a dataframe

**Examples**

```
factorisedf(Orange,names(Orange))
```

MR

*Regression Composite estimation***Description**

Regression Composite estimation

**Usage**

```
MR(
  list.tables,
  w,
  id,
  list.xMR = NULL,
  list.x1 = NULL,
  list.x2 = NULL,
  list.y = NULL,
  calibmethod = "linear",
  Alpha = 0.75,
  theta = 3/4,
  list.dft.x2 = NULL,
  dft0.xMR = NULL,
  mu0 = NULL,
  Singh = TRUE,
  dispweight = FALSE,
  analyse = FALSE
)
```

**Arguments**

<code>list.tables</code>	A list of dataframes
<code>w</code>	either a real number or a character string indicating the name of the weight variable.
<code>id</code>	an identifier
<code>list.xMR</code>	list of variables used to compute proxy composite regression variable
<code>list.x1</code>	list of auxiliary variables used in the calibration, whose calibrated weighted total has to be equal to initially weighted total
<code>list.x2</code>	id list of auxiliary variables used in the calibration, whose calibrated weighted total has to be equal to values provided by <code>list.dft.x2</code>
<code>Alpha</code>	a vector of alpha values. if <code>alpha="01"</code> , this will compute MR3
<code>theta</code>	a numerical value
<code>list.dft.x2</code>	id list of auxiliary variables used in the calibration, whose calibrated weighted total has to be equal to initially weighted total
<code>mu0</code>	a numerical value

Singh	a boolean
dispweight	a boolean
analyse	a boolean
list.y:	list of variables whose weighted sum needs to be computed. It can be factor or character variables.

Value

a dataframe.

Examples

```
MR(list.tables<-
plyr::dply(CRE_data,.variables=~time),w="Sampling.weight",list.xMR="Status",id="Identifier",list.y=c("Hobby",
```

---

varAK3	<i>Gives the variance of the AK estimators from the A,K coefficients and the variance covariance matrix of the month in sample estimates</i>
--------	--

---

Description

Gives the variance of the AK estimators from the A,K coefficients and the variance covariance matrix of the month in sample estimates

Usage

```
varAK3(ak, Sigma)
```

Arguments

ak	A set of 3 A, K coefficients, of the form c(a1=.3,a2=.4,a3=0,k1=.4,k2=.7,k3=0).
Sigma	An array of dimension 3 x 8 (number of rotation groups) x number of months x 3 x 8 (number of rotation groups) x number of months.

Value

The variance of the AK estimators from the A,K coefficients and the variance covariance matrix .

Examples

```
varAK3(ak=c(a1=.3,a2=.4,a3=0,k1=.4,k2=.7,k3=0), Sigma=array(drop(stats::rWishart(1,df=3*10*8,diag(3*10*8))),rep
```

---

varAK3diff	<i>Gives the variance of the consecutive differences of AK estimators from the A,K coefficients and the variance covariance matrix of the month in sample estimates</i>
------------	---

---

### Description

Gives the variance of the consecutive differences of AK estimators from the A,K coefficients and the variance covariance matrix of the month in sample estimates

### Usage

```
varAK3diff(ak, Sigma)
```

### Arguments

ak	A set of 3 A, K coefficients, of the form c(a1=.3,a2=.4,a3=0,k1=.4,k2=.7,k3=0).
Sigma	An array of dimension 3 x 8 (number of rotation groups) x number of months x 3 x 8 (number of rotation groups) x number of months.

### Value

The variance of the consecutive differences of the AK estimators from the A,K coefficients and the variance covariance matrix .

### Examples

```
varAK3diff(ak=c(a1=.3,a2=.4,a3=0,k1=.4,k2=.7,k3=0), Sigma=array(drop(stats::rWishart(1,df=3*10*8,diag(3*10*8)))+
add(10, 1)
```

---

varAK3diff	<i>Gives the variance of the unemployment rate estimates derived from AK estimators from the A,K coefficients and the variance covariance matrix of the month in sample estimates</i>
------------	---

---

### Description

Gives the variance of the unemployment rate estimates derived from AK estimators from the A,K coefficients and the variance covariance matrix of the month in sample estimates

### Usage

```
varAK3diff(ak, Sigma, Scompop, what = c(unemployed = "0", employed = "1"))
```

**Arguments**

ak	A set of 3 A, K coefficients, of the form $c(a1=.3, a2=.4, a3=0, k1=.4, k2=.7, k3=0)$ .
Sigma	An array of dimension 3 x 8 (number of rotation groups) x number of months x 3 x 8 (number of rotation groups) x number of months.
Scomppop	An array of dimension number of months x 3.

**Value**

The variance of the the unemployment rate estimates derived from the AK estimators from the A,K coefficients and the variance covariance matrix .

**Examples**

```
varAK3diffstat(ak=c(a1=.3,a2=.4,a3=0,k1=.4,k2=.7,k3=0), Sigma=array(drop(stats::rWishart(1,df=3*10*8,diag(3*10*8)))))
```

---

varAK3rat	<i>Gives the variance of the unemployment rate estimates derived from AK estimators from the A,K coefficients and the variance covariance matrix of the month in sample estimates</i>
-----------	---

---

**Description**

Gives the variance of the unemployment rate estimates derived from AK estimators from the A,K coefficients and the variance covariance matrix of the month in sample estimates

**Usage**

```
varAK3rat(ak, Sigma, Scomppop, what = c(unemployed = "0", employed = "1"))
```

**Arguments**

ak	A set of 3 A, K coefficients, of the form $c(a1=.3, a2=.4, a3=0, k1=.4, k2=.7, k3=0)$ .
Sigma	An array of dimension 3 x 8 (number of rotation groups) x number of months x 3 x 8 (number of rotation groups) x number of months.
Scomppop	An array of dimension number of months x 3.

**Value**

The variance of the the unemployment rate estimates derived from the AK estimators from the A,K coefficients and the variance covariance matrix .

**Examples**

```
varAK3rat(ak=c(a1=.3,a2=.4,a3=0,k1=.4,k2=.7,k3=0), Sigma=array(drop(stats::rWishart(1,df=3*10*8,diag(3*10*8)))))
```

---

var_lin	<i>Gives the variance of an array Y that is a linear transformation AX of an array X from the coefficients of A and Sigma=Var[X]</i>
---------	--

---

**Description**

Gives the variance of an array Y that is a linear transformation AX of an array X from the coefficients of A and Sigma=Var[X]

**Usage**

```
var_lin(A, Sigma)
```

**Arguments**

Sigma	An array of dimension $b_1 \times \dots \times b_p \times b_1 \times \dots \times b_p$
coeff	An array of dimension $a_1 \times \dots \times a_n \times b_1 \times \dots \times b_p$

**Value**

The variance of the AK estimators from the A,K coefficients and the variance covariance matrix .

**Examples**

```
a=c(2,4);b=c(3,10,8);A<-array(rnorm(prod(a)*prod(b)),c(a,b));
dimnames(A)[1:2]<-lapply(a,function(x){letters[1:x]});names(dimnames(A))[1:2]<-c("d1","d2");
Sigma=array(drop(stats::rWishart(1,df=prod(b),diag(prod(b)))),rep(b,2));
var_lin(A,Sigma)
```

---

W.ak	<i>general AK weights as a function of a and k parameters.</i>
------	--

---

**Description**

general AK weights as a function of a and k parameters.

**Usage**

```
W.ak(
  months,
  groups,
  S = c(2:4, 6:8),
  S_1 = S - 1,
  a,
  k,
  eta0 = length(groups)/length(S),
```



```

    eta1 = eta0 - 1,
    rescaled = F
  )

```

### Arguments

months	an integer, indicating number of months
groups	a vector of character strings or numeric string
S	a vector of integers indicating the indices of the rotation group in the sample that overlap with the previous sample: groups[S] are the overlapping rotation groups
S_1	a vector of integers indicating the indices of the corresponding rotation group of S in the previous month
a	a numeric value
k	a numeric value
rescaled	a boolean (default FALSE) indicating whether these AK coefficient are to be applied to rescaled or not rescaled month in sample weighted sums
nmonth	an integer, indicating number of months
ngroup	a vector of character strings or numeric string

### Value

an array of AK coefficients  $W[m2, m1, mis1]$  such that  $A_k$  estimate for month  $m2$  is  $\sum(W[y2,,]) * Y$  where  $Y[m1, mis1]$  is direct estimate on mis  $mis1$  for emp stat  $y1$  at month  $m1$ .

### Examples

```

W<-W.ak(months=1:3,groups=1:8,a=.2,k=.5);dimnames(W)
W<-W.ak(months=2:4,groups=letters[1:8],a=.2,k=.5);dimnames(W);
Y<-WSrg(list.tables,weight="pwsswgt",list.y="pemlr",rg="hrmis")
dimnames(Y);month="m";group="mis";variable="y";
A=W.ak(months = dimnames(Y)[[month]],groups = dimnames(Y)[[group]],S=c(2:4,6:8),a=.5,k=.5)

```

---

W.multi.ak

*general AK weights as a function of a and k parameters.*

---

### Description

general AK weights as a function of a and k parameters.

Usage

```
W.multi.ak(  
  months,  
  groups,  
  S,  
  S_1 = S - 1,  
  ak,  
  eta0 = length(groups)/length(S),  
  eta1 = eta0 - 1,  
  rescaled = F  
)
```

Arguments

- S                    a vector of integers indicating the indices of the rotation group in the sample
- ak                   a list of 2-dimension vectors
- nmonth              an integer, indicating number of months
- ngroups             : number of groups

Value

an array of AK coefficients  $W[m2,m1,mis1]$  such that  $A_k$  estimate for month  $m2$  is  $\sum(W[y2,,]) * Y$  where  $Y[m1,mis1]$  is direct estimate on mis  $mis1$  for emp stat  $y1$  at month  $m1$ .

Examples

```
W.multi.ak(months=1:3,groups=1:8,S=c(2:4,6:8),ak=list(c(a=.2,k=.5),c(a=.2,k=.4)))
```

---

W.rec	<i>general month in sample estimates weights for recursive linear combinaison of mis estimates</i>
-------	--

---

Description

general month in sample estimates weights for recursive linear combinaison of mis estimates

Usage

```
W.rec(  
  months,  
  groups,  
  S = c(2:4, 6:8),  
  S_1 = S - 1,  
  Coef = c(alpha_1 = 0, alpha0 = 1, beta_1 = 0, beta0 = 0, gamma0 = 0)  
)
```

**Arguments**

months	an integer, indicating number of months
groups	a vector of character strings or numeric string
S	a vector of integers indicating the indices of the rotation group in the sample that overlap with the previous sample: groups[S] are the overlapping rotation groups
S_1	a vector of integers indicating the indices of the corresponding rotation group of S in the previous month
Coef	a named vector of 5 numeric value
nmonth	an integer, indicating number of months
ngroup	a vector of character strings or numeric string

**Value**

an array of AK coefficients  $W[m2, m1, mis1]$  such that  $A_k$  estimate for month  $m2$  is  $\sum(W[y2,,]) * Y$  where  $Y[m1, mis1]$  is direct estimate on mis  $mis1$  for emp stat  $y1$  at month  $m1$ .

**Examples**

```
alpha0=runif(1);
W<-W.rec(months=1:3,
          groups=1:8,
          Coef=c(alpha_1=1-alpha0,alpha0=alpha0,beta0=runif(1),beta_1=runif(1),gamma0=runif(1)))
dimnames(W)
W<-W.ak(months=2:4,groups=letters[1:8],a=.2,k=.5);dimnames(W);
Y<-WSrg(list.tables,weight="pwsswgt",list.y="pemlr",rg="hrmis")
dimnames(Y);month="m";group="mis";variable="y";
A=W.rec(months = dimnames(Y)[[month]],groups = dimnames(Y)[[group]],S=c(2:4,6:8),a=.5,k=.5)
```

---

WS	<i>Compute weighted sums</i>
----	------------------------------

---

**Description**

Compute weighted sums

**Usage**

```
WS(list.tables, weight = 1, list.y = NULL, sep = "_n", dimname1 = "m")
```

**Arguments**

list.tables	A list of dataframes
weight	either a real number or a character string indicating the name of the weight variable.
list.y:	list of variables whose weighted sum needs to be computed. It can be factor or character variables.

**Value**

a dataframe.

**Examples**

```
WS(plyr::dply(CRE_data,.variables=~time),"Sampling.weight",c("Hobby","Status","State"));
WS(plyr::dply(CRE_data,.variables=~time),"Sampling.weight",character(0));
```

---

WSrg	<i>Weighted sums by rotation groups</i>
------	---

---

**Description**

Weighted sums by rotation groups

**Usage**

```
WSrg(
  list.tables,
  weight = 1,
  list.y = NULL,
  rg = "hrmis",
  rescale = F,
  dimname1 = "m"
)
```

**Arguments**

- list.tables      a named list of data frames
- weight           a character string indicating the variable name or a numerical value
- list.y           a vector of character strings indicating the study variables
- rg                a character string indicating the name of the rotation group.

**Value**

an array

**Examples**

```
library(dataCPS)
period<-200501:200512
list.tables<-lapply(data(list=paste0("cps",period),package="dataCPS"),get);
names(list.tables)<-period
Y<-WSrg(list.tables,"pwsswgt",list.y="pemlr",rg="hrmis")
dimnames(Y);dim(Y)
Y<-plyr::daply(plyr::ldply(list.tables,function(L){L[c("pemlr","pwsswgt","hrmis")]},.id="m"),
~m+pemlr+hrmis,function(d){data.frame(y=sum(d$pwsswgt))}[names(list.tables),,]
```

```
dimnames(Y);dim(Y)
system.time(plyr::daply(plyr::ldply(list.tables,,function(L){L[c("pemlr","pwsswgt","hrmis")]}) ,
~.id+pemlr+hrmis,function(d){data.frame(y=sum(d$pwsswgt))}))
system.time(WSrg(list.tables,weight="pwsswgt",list.y="pemlr",rg="hrmis"))
```

---

WSrg2

*Weighted sums by rotation groups*


---

## Description

Weighted sums by rotation groups

## Usage

```
WSrg2(list.tables, weight, y, rg = "hrmis", rescale = F, dimname1 = "m")
```

## Arguments

<code>list.tables</code>	a named list of data frames
<code>weight</code>	a character string indicating the variable name or a numerical value
<code>y</code>	a character strings indicating one study variable
<code>rg</code>	a character string indicating the name of the rotation group.

## Value

an array

## Examples

```
library(dataCPS)
period<-200501:200512
list.tables<-lapply(data(list=paste0("cps",period),package="dataCPS"),get);
names(list.tables)<-period
Y<-WSrg2(list.tables,"pwsswgt",list.y=c("pemlr","pwsswgt"),rg="hrmis")
Y<-WSrg2(list.tables,"pwsswgt",list.y=c("pemlr"),rg="hrmis")
```

# Index

`add.rg`, [2](#)  
`add.rg3`, [3](#)  
`AK`, [4](#)  
`AK_est`, [6](#)  
  
`CoeffGM`, [8](#)  
`CoeffGM.array`, [8](#)  
`CoeffGM.matrix`, [9](#)  
`CoeffS2`, [10](#)  
`composite`, [10](#)  
`CPS_A_e`, [14](#)  
`CPS_A_u`, [14](#)  
`CPS_AK`, [12](#)  
`CPS_AK_coeff.array.fl`, [13](#)  
`CPS_AK_est`, [13](#)  
`CPS_K_e`, [15](#)  
`CPS_K_u`, [15](#)  
`CPS_X_array`, [17](#)  
`CPS_X_matrix`, [17](#)  
`CPS_Xplus_array`, [15](#)  
`CPS_Xplus_matrix`, [16](#)  
  
`double`, [18](#)  
  
`empirical.var`, [19](#)  
  
`factorisedf`, [19](#)  
  
`MR`, [20](#)  
  
`var_lin`, [24](#)  
`varAK3`, [21](#)  
`varAK3diff`, [22](#)  
`varAK3diffrat`, [22](#)  
`varAK3rat`, [23](#)  
  
`W.ak`, [24](#)  
`W.multi.ak`, [25](#)  
`W.rec`, [26](#)  
`WS`, [27](#)  
`WSrg`, [28](#)  
`WSrg2`, [29](#)