Package 'genvar'

October 16, 2019

Title An Imperative Library for Data Manipulation

Version 0.0.2.0

Description
Implements tools for manipulating data sets and performing regressions in a way that is familiar to users of a popular, but proprietary, statistical package commonly used in the social sciences. Loads a single dataset into memory and implements a set of imperative commands to modify that data and perform regressions and other analysis on the dataset. Offers an alternative to standard R's function-based approach to data manipulation.

Depends R (>= 3.5.1.0)

Imports Formula, foreign, readstata13, sandwich, plm, clubSandwich,
rlang

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

BugReports `https://github.com/flynnzac/genvar`

NeedsCompilation no

Author Zach Flynn [aut, cre]

Maintainer Zach Flynn <`zlflynn@gmail.com`>

R topics documented:

`addobs` *add observations to the data set*

Description

Add observations to the data set, similar in functionality to Stata's append command

Usage

```
addobs(obs)
```

Arguments

`obs`	one of two possible input types: <ul><li>An R data frame with the same columns as the current dataset.</li><li>A comma-separated string in the following format: "var1=1,var2=2,var3=3" which inputs a single observation.</li></ul>

Value

returns NULL, invisibly

Examples

```
library(plm)
data(Produc)
use(Produc, clear=TRUE)
keepvar("state year emp unemp")
addobs("state='Puerto Rico',year=1990,emp=100,unemp=5")
listif()
df <- data.frame(state=rep("Puerto Rico", times=2), year=1991:1992,emp=c(102,104),unemp=c(4.9,5.1))
addobs(df)
listif()
```

`assert_loaded` *assert a dataset is loaded in genvar and error otherwise*

Description

assert a dataset is loaded in genvar and error otherwise

Usage

```
assert_loaded()
```

Value

returns NULL, invisibly capture(clear()) assert_loaded() use(cars) assert_loaded()

`builddata` *creates a dataset of a given number of observations*

Description

Creates a dataset of a given number of observations. Does so by creating a variable called "v1" with all missing values.

Usage

```
builddata(n, replace = FALSE)
```

Arguments

`n`	the number of observations to make the new dataset
`replace`	if TRUE, replace a dataset in memory, if FALSE, error if a dataset is already loaded

Value

returns NULL, invisibly

Examples

```
buildata(100, replace=TRUE)
listif()
```

capture *captures an expression, returning TRUE if there was an error and FALSE otherwise*

Description

captures an expression, returning TRUE if there was an error and FALSE otherwise

Usage

```
capture(expr, silent = FALSE)
```

Arguments

`expr`	an expression to be evaluated
`silent`	if TRUE, suppress error messages from printing (default: FALSE)

Value

FALSE if the expression successfully ran and TRUE otherwise

Examples

```
capture({log(1)})
capture({log(-1)})
```

clear *clears the dataset in memory*

Description

removes a dataset from memory, errors if no dataset is loaded

Usage

```
clear()
```

Value

returns NULL invisibly

Examples

```
use(cars, clear=TRUE)
listif()
clear()
listif()
```

`collapse` *collapses a data set by variables using arbitrary aggregation functions*

Description

collapse a data set to produce summary statistics possibly by a set of variables as in the Stata code: collapse (fun1) var1 (fun2) var2, by(byvar1 byvar2). But this function is more flexible than the Stata version because any arbitrary function can be used in collapse not just traditional aggregation functions.

Usage

```
collapse(values, byvar)
```

Arguments

`values`	an argument with the form `"fun1(var1) fun2(var2) fun3(var3,var4)"` describe the aggregations to be performed where fun1, fun2, fun3 are most likely aggregation functions like "sum", "mean", "max", "median", etc. But the function could be anything that returns a scalar.
`byvar`	a variable list giving the variables to collapse by. The resulting dataset will have as many rows as there are unique levels of the `byvar` variable list.

Value

returns NULL, invisibly

Examples

```
library(plm)
data(Produc)
use(Produc, clear=TRUE)
listif()
collapse("sum(emp)","year")
listif()
```

`count` *Counts how many observations (optionally, satisfying a condition)*

Description

Counts how many observations (optionally, satisfying a condition)

Usage

```
count(ifstmt)
```

Arguments

`ifstmt`	an optional argument which gives an condition that must be met for the observation to be counted

Value

returns the count

Examples

```
use(cars, clear=TRUE)
count()
count(speed <= 20)
```

---

describe *lists the names of the variables in the dataset*

Description

lists the names of the variables in the dataset

Usage

```
describe(pattern)
```

Arguments

`pattern`	an optional regular expression which only returns variable names that match the expression. Can be unquoted if it is just a variable name.

Value

A vector of names of variables with an attribute called "type" giving the types of the variables. The class of the object is "varlist".

Examples

```
use(cars, clear=TRUE)
describe()
describe("s*")
describe(speed)
```

---

destring *convert a variable with string type into a numeric value*

Description

convert a variable with string type into a numeric value

Usage

```
destring(varlist)
```

Arguments

`varlist`	variables to convert, in the form "var1 var2 var3" or, if a single variable, an unquoted variable will work as well (i.e. var1).

Value

returns NULL, invisibly

Examples

```
use(cars, clear=TRUE)
tostring(speed)
listif()
describe()
destring(speed)
listif()
describe()
```

`do` *Executes R code on the dataset*

Description

Executes an R expression using variables from the dataset, possibly separately for each level of a given varlist (like the `by` prefix in Stata).

Usage

```
do(expr, by)
```

Arguments

`expr`	an R expression which can use any of the variable names in the current dataset. It can be quoted or unquoted.
`by`	a variable list in "var1 var2 var3" format or, if a single variable, it can be unquoted (var1). The R expression will be applied separately for the data subsetted to each level of the variable list.

Value

returns whatever the expression `expr` returns. If `by` is specified, it will be a list of the result for applying the expression to each section of the data

Examples

```
use(cars, clear=TRUE)
do(coef(lm(speed~dist)))
```

`dropif` *drops rows from the dataset*

Description

drops rows from the dataset

Usage

```
dropif(x)
```

Arguments

`x`	a condition like `var1==2` describing the observations that should be removed from the data set.

Value

returns NULL, invisibly

Examples

```
use(cars, clear=TRUE)
listif()
dropif(speed <= 20)
listif()
```

`dropvar` *drops variables in varlist format from the dataset*

Description

drops variables in varlist format from the dataset

Usage

```
dropvar(x)
```

Arguments

`x`	a varlist in "var1 var2 var3" format or unquoted if a single variable

Value

returns NULL, invisibly

Examples

```
use(cars, clear=TRUE)
listif()
dropvar(speed)
listif()
```

`estimates_get` *display estimation results*

Description

display estimation results

Usage

```
estimates_get(name = NULL)
```

Arguments

`name`	name of estimates to be returned. If unspecified, return current estimates.

Value

returns a table of the estimated coefficients and standard errors

Examples

```
use(cars, clear=TRUE)
reg(speed,dist)
estimates_store("speed_dist")
reg(dist,speed)
estimates_store("dist_speed")
estimates_get("speed_dist")
estimates_get("dist_speed")
```

`estimates_restore` *restore* `genvar` *estimates*

Description

restore `genvar` estimates

Usage

```
estimates_restore(name)
```

Arguments

`name`	name of estimates to be restored

Value

returns NULL, invisibly

Examples

```
use(cars, clear=TRUE)
reg(speed, dist)
estimates_store("speed_dist")
reg(dist,speed)
estimates_get()
estimates_restore("speed_dist")
estimates_get()
```

`estimates_save` *save* genvar *estimates*

Description

save genvar estimates

Usage

```
estimates_save(estfile)
```

Arguments

`estfile`	file to save current estimates to.

Value

returns NULL, invisibly

Examples

```
use(cars, clear=TRUE)
reg(speed,dist)
fp <- file.path(tempdir(), "myest.rdata")
estimates_save(fp)
clear()
estimates_use(fp)
estimates_get()
```

`estimates_store` *store* genvar *estimates*

Description

store genvar estimates

Usage

```
estimates_store(name)
```

Arguments

`name`	name to use to store current estimates from a `genvar` estimation function like `reg`, `logit`, or `probit`.

Value

returns NULL, invisibly

Examples

```
use(cars, clear=TRUE)
reg(speed,dist)
estimates_store("speed_dist")
reg(dist,speed)
estimates_store("dist_speed")
estimates_get("speed_dist")
estimates_get("dist_speed")
```

`estimates_use` *loads* `genvar` *estimates from file*

Description

loads `genvar` estimates from file

Usage

```
estimates_use(file)
```

Arguments

`file`	file to load estimates from.

Value

returns NULL, invisibly

Examples

```
use(cars, clear=TRUE)
reg(speed,dist)
fp <- file.path(tempdir(), "myest.rdata")
estimates_save(fp)
clear()
estimates_use(fp)
estimates_get()
```

`fillin` *Fully rectangularize a dataset*

Description

Make the dataset have one observation for every possible interaction of a list of variables.

Usage

```
fillin(varlist)
```

Arguments

`varlist`	a variable list in "var1 var2 var3 x*" format where "*" matches zero or more of any character and "?" matches one of any character (an unquoted list will work as well with one variable). On exit, the data set will contain one observation for every possible interaction of variables with missing values filled in where appropriate.

Value

returns NULL, invisibly

Examples

```
library(plm)
data(Produc)
use(Produc, clear=TRUE)
keepvar("state year emp unemp")
addobs("state='Mars',year=1990,emp=100,unemp=4.0")
fillin("state year")
listif()
```

`forval` *Execute code in the datasets environment for all values of a vector, replacing a macro with the value in each iteration*

Description

Execute code in the datasets environment for all values of a vector, replacing a macro with the value in each iteration

Usage

```
forval(values, expr, macro = "%val")
```

Arguments

`values`	the vector of values to loop over. For example, specifying 1:5 would loop over integers from 1 to 5.
`expr`	a quoted expression (the experession must be enclosed in quotes) to evaluate in the loop which (presumably) uses the macro expression
`macro`	a word to replace in the quoted expression with the values we are looping over (default: "%val")

Value

returns NULL, invisibly

Examples

```
use(cars, clear=TRUE)
listif()
forval (2:4, "gen(speed%val, speed^%val)")
listif()
```

`forvar` *apply a function to each of a list of variables*

Description

apply a function to each of a list of variables

Usage

```
forvar(varlist, action, macro = "%var")
```

Arguments

`varlist`	a list of variables in the format "var1 var2 var3" (or, if a single variable, it may be unquoted).
`action`	a quoted expression (must be quoted) to apply to each variable where the variable is represented in the expression by `macro`.
`macro`	an expression that will be replaced in `action` for each variable, by default %var.

Value

returns NULL, invisibly

Examples

```
use(cars, clear=TRUE)
forvar("speed dist", "gen(%var2, %var^2)")
listif()
```

gen — *generates a new variable that is a transformation of existing variables in the dataset or replaces one*

Description

generates a new variable that is a transformation of existing variables in the dataset or replaces one

Usage

```
gen(var, value, byvar = NULL, subset = NULL, replace = FALSE)
```

Arguments

var	the name of the variable to be generated
value	the transformation of the dataset to replace the "newvar" in option `form` with. For example, value="sum(wage*female)" to get a variable which has total female wages. In Stata, the same command would be: "egen femalewage = total(wage*female)".
byvar	apply the value for each level of the by variables, specified either as a formula, like ~byvar1+byvar2+... or as a varlist "byvar1 byvar2 byvar3...".
subset	only generate values if the condition provided in subset is true. Make sure to enclose the expression in quotes, like so: subset="female==1 & highschool==1" to generate the values only for women who graduated from highschool. This option is used like the "if" in Stata.
replace	either TRUE or FALSE. If FALSE (default), the code refuses to alter the variable if the variable already exists. Otherwise, if replace=TRUE, then the values will be replaced.

Value

returns NULL, invisibly

Examples

```
use(cars, clear=TRUE)
gen(speeddist, speed*dist)
listif()
```

getdata — *exports data frame from genvar environment to R environment*

Description

Returns the data frame currently in the genvar environment. It is equivalent to calling `listif()`, but the name is not as intuitive to use for this purpose.

Usage

```
getdata()
```

Value

the data frame currently in the genvar environment

Examples

```
use(cars, clear=TRUE)
getdata()
all(getdata() == listif())
```

`gvmerge` *merge two datasets*

Description

Merges two datasets using either a left (keep all elements in current datasets and replace with missing if not present in new set), right (keep all elements in new dataset), outer (keep all observations in both datasets), or inner join (only keep elements in both datasets)

Usage

```
gvmerge(data, on, kind = "left", ...)
```

Arguments

`data`	dataset to merge in, either an R data frame, a csv file name, or a dta (Stata) file name.
`on`	a variable list to merge on with the form "var1 var2 var3" (or possibly unquoted if a single variable).
`kind`	one of "left", "right", "outer", or "inner" (default: "left")
`...`	extra options to pass to `read.csv` or `read.dta` (for old Stata files) or `read.dta13` (for newer ones).

Value

returns NULL, invisibly

Examples

```
library(plm)
data(Produc)
use(Produc, clear=TRUE)
collapse(mean(emp), year)
rename(mean(emp), avgemp)
gvmerge(Produc, on="year", kind="right")
listif()
```

gvplot — *convenience interface to R's* `plot` *command*

Description

Executes a plot command in genvar's environment so that gvplot(xvar,yvar) will plot a scatter plot of the variables xvar and yvar in the genvar enviroment.

Usage

```
gvplot(...)
```

Arguments

`...`	arguments to be passed to `R`'s plot command.

Value

returns NULL, invisibly

Examples

```
library(plm)
data(Produc)
use(Produc, clear=TRUE)
gen(laborforce, emp/(1-unemp/100))
empfrac = function (emp, laborforce) sum(emp)/sum(laborforce)
collapse("empfrac(emp,laborforce)", year)
rename(empfrac(emp, laborforce), empfrac)
destring(year)
gvplot(year, empfrac, type="b", main="Employment Percentage over Time",
xlab="Year", ylab="Employment Percentage", pch=19)
```

headdata — *get first few observations*

Description

get first few observations

Usage

```
headdata(num)
```

Arguments

`num`	how many of the first observations to get

Value

returns the first `num` rows of data

Examples

```
use(cars, clear=TRUE)
headdata(5)
```

`is_loaded` *a command to determine whether data is loaded*

Description

a command to determine whether data is loaded

Usage

```
is_loaded()
```

Value

returns TRUE if dataset is loaded in genvar and FALSE otherwise

Examples

```
capture(clear())
is_loaded()
use(cars)
is_loaded()
```

`keepif` *keeps some rows in the dataset and drops the rest*

Description

keeps some rows in the dataset and drops the rest

Usage

```
keepif(x)
```

Arguments

`x`	a condition like: `var1==2` in which case observations that satisfy the condition are kept and all others are removed.

Value

returns NULL, invisibly

Examples

```
use(cars, clear=TRUE)
keepif(speed <= 20)
listif()
```

keepvar *keeps some variables in the dataset and drops the others*

Description

keeps some variables in the dataset and drops the others

Usage

```
keepvar(x)
```

Arguments

x	a varlist either of the form "var1 var2 var3" or, if a single variable, it can be unquoted.

Value

returns NULL, invisibly

Examples

```
use(cars, clear=TRUE)
keepvar(speed)
listif()
```

L *a function to take lags and leads with panel data*

Description

a function to take lags and leads with panel data, mostly a wrapper for `plm`'s lag function.

Usage

```
L(x, k = 1, ...)
```

Arguments

x	variable to lag
k	how many lags to take? If a negative number, leads will be generated.
...	other options to pass to `plm::lag`, does not need to be specified

Value

returns lag of the variable as a data frame

Examples

```
library(plm)
data(Produc)
use(Produc, clear=TRUE)
xtset(year, state)
gen(Lemp, L(emp))
gen(L2emp, L(emp,2))
headdata(10)
```

`listif` *prints the part of the dataset that satisfies certain conditions*

Description

prints the part of the dataset that satisfies certain conditions

Usage

```
listif(cond, vars, ...)
```

Arguments

`cond`	a conditional expression; only observations that satisfy the condition will be returned.
`vars`	a variable list; only variables in the list will be returned.
`...`	other options, currently ignored

Value

the part of the dataset that satisfies the condition and contains the specified columns

Examples

```
use(cars, clear=TRUE)
listif()
listif(speed <= 20)
```

`logit` *estimate a logistic regression*

Description

estimate a logistic regression

Usage

```
logit(y, x, subset = NULL, weights = NULL, linkfunc = "logit", ...)
```

Arguments

`y`	name of the dependent variable
`x`	names of the independent variables in varlist format, either "x1 x2 x3" or if it is a single variable it does not need to be quoted.
`subset`	conditions to run the command only of a subset of the data (analogous to "if" statements in Stata)
`weights`	the name of a variable to use for weights in estimation
`linkfunc`	specify the linking function (logit, by default). Can set to "probit" to do probit estimation or use `probit` (which is equivalent).
`...`	other options to pass to `glm`

Value

b coefficient vector

V covariance matrix of coefficients

Examples

```
library(plm)
data(Produc)
use(Produc, clear=TRUE)
gen(empmedian, emp > median(emp))
r = logit(empmedian, unemp)
r
```

`pred` *gets fitted values from a genvar regression object*

Description

Gets fitted values from a genvar regression object. For panel models, this predicts the non-fixed effects part of the regression.

Usage

```
pred()
```

Details

Operates on the loaded estimation object, see `estimates_use`.

Value

returns predictions from model

Examples

```
use(cars, clear=TRUE)
listif()
reg(dist, speed)
gen(fit, pred())
listif()
```

`preserve` *preserve a data set before modification*

Description

preserve a data set before modification

Usage

```
preserve(data = NULL)
```

Arguments

`data`	a data set to preserve

Value

a value that can be passed to `restore` to restore the data set later

Examples

```
require(stats)
use(cars, clear=TRUE)
p <- preserve()
collapse("mean(dist)", "speed")
list()
restore(p, replace=TRUE)
list()
```

`probit` *estimate a probit regression*

Description

`probit(...)` is equivalent to `logit(..., linkfunc="probit")`.

Usage

```
probit(...)
```

Arguments

`...`	options passed to `logit`.

`reg`	*regress y on x with robust standard errors, clustered standard errors, HAC standard errors, panel fixed effects, etc*

Description

regress y on x with robust standard errors, clustered standard errors, HAC standard errors, panel fixed effects, etc.

Usage

```
reg(y, x, subset = NULL, effect = NULL, robust = TRUE, hac, cluster,
  rtype = 1)
```

Arguments

`y`	name of the dependent variable
`x`	names of the independent variables in "x1 x2 x3" format. To include a variable as a categorical variable (when you would use "i.state" to get state dummies in Stata), include it as "factor(state)".
`subset`	conditions to subset the data
`effect`	either "twoways", "individual", or "time" for fixed effects. Dataset must already have been `xtset`.
`robust`	whether to use robust standard errors
`hac`	which variable to order by to compute heteroskedastic and auto correlation standard errors (if unspecified, do not do HAC correction)
`cluster`	a variable list giving the names of the variables to cluster by in producing clustered standard errors
`rtype`	gives the type of heteroskedasticity correction to make. By default, it is "1" to implement HC1 which is the same as Stata's small sample corrected standard errors. rtype can be any integer from 0 to 3 with each value corresponding to a different heteroskedastic correction (HCx). See documention for `vcovHC` in package `sandwich`.

Value

b coefficient vector

V covariance matrix of coefficients

Examples

```
library(plm)
data(Produc)
use(Produc, clear=TRUE)
r = reg(emp, unemp)
r
xtset(year,state)
r = reg(emp, unemp, hac=year)
r
r = reg(emp, unemp, cluster=year)
r
```

`rename` *renames variables in the dataset*

Description

renames variables in the dataset

Usage

```
rename(var, newvar)
```

Arguments

`var`	the name of the variable to rename
`newvar`	the new name of the variable

Value

returns NULL, invisibly

Examples

```
use(cars, clear=TRUE)
listif()
rename("speed","velocity")
listif()
```

`restore` *restore a dataset from a previous preserve to be currently used*

Description

restore a dataset from a previous preserve to be currently used

Usage

```
restore(envir, replace = FALSE)
```

Arguments

`envir`	a previous preserve value.
`replace`	if TRUE, restore even if another dataset is in memory. If FALSE, do not.

Value

the preserved data set

Examples

```
require(stats)
use(cars, clear=TRUE)
p <- preserve()
collapse("mean(dist)","speed")
list()
restore(p, replace=TRUE)
list()
```

---

savedata *saves data to a CSV or RDS file*

Description

saves data to a CSV or RDS file

Usage

```
savedata(file, rds = FALSE)
```

Arguments

`file`	a file name to save the current data to
`rds`	whether to save the file to an RDS file (default: FALSE)

Value

returns NULL, invisibly

Examples

```
use(cars, clear=TRUE)
savedata(file.path(tempdir(), "cars.csv"))
savedata(file.path(tempdir(), "cars.rds"), rds=TRUE)
```

---

shape *reshapes a data set from wide to long or from long to wide formats*

Description

reshapes a data set from wide to long or from long to wide formats

Usage

```
shape(form, direction = "long")
```

Arguments

`form`	if direction="long", then the argument should have the form: id1+id2+..~newvar|stub where there are variables in the data set named "stubXXXX" and "newvar" is the name of the new variable that will be added to the data set which will contain the various values of "stubXXXX" on exit. The variable "stub" on exit will contain the value of "XXXX". Variables (id1,id2,...) will also be included in the dataset on exit. The command behaves like "reshape long stub, i(id1 id2 ...) j(newvar)" in Stata. If direction="wide", then the argument should have the form, id1+id2+...~values1+values2+...|byvar1+byvar2+... The variables (id1,id2,...,byvar1,byvar2,...) should uniqely identify observations in the data. On exit the dataset will contain (id1,id2,...) in addition to values1byvar1.byvar2, values2byvar1.byvar2, ... for each unique value of (byvar1,byvar2,...). The command behaves like "reshape wide values1 values2 ..., i(id1 id2 ...) j(byvar1...)
`direction`	either "long" or "wide" to indicate the direction to reorient the data set

Value

returns NULL, invisibly

Examples

```
library(plm)
data(Produc)
use(Produc, clear=TRUE)
listif()
shape(state~emp|year, direction="wide")
listif()
shape(state~year|emp, direction="long")
listif()
```

`structure_varlist` *creates a formula object from a varlist, mostly for internal use.*

Description

A varlist in `genvar` is a space-separated string potentially with wildcard characters, "var1 var2 var3 x*". This function converts a varlist to a formula or to a vector.

Usage

```
structure_varlist(x, type = "formula")
```

Arguments

`x`	the varlist to be converted in "var1 var2 var3" format. Can be specified using the *globbing* characters "*" (match zero or more of any character) or "?" (match any single character) like "var*" or "var?" for "var1 var2 var3" or using regular expressions if `regex=TRUE` ("var[0-9]+" = "var1 var2 var3").
`type`	if "formula", return a varlist in formula format; if "vector", return a varlist in character vector format.

Value

a formula object which can be passed to `model.frame` or a character vector giving the name of each variable

Examples

```
use(cars, clear=TRUE)
structure_varlist("speed dist", type="formula")
structure_varlist("speed dist", type="vector")
structure_varlist("*", type="vector")
```

`subset.varlist` *generate a varlist that is a subset of another*

Description

generate a varlist that is a subset of another

Usage

```
## S3 method for class 'varlist'
subset(x, vars, ...)
```

Arguments

`x`	a varlist
`vars`	a set of variable names
`...`	currently ignored

Value

returns the subset of variable names with an attibute "type" giving the types. The vector is of class "varlist"

summarize — *summarize a variable list, giving basic descriptive statistics*

Description

summarize a variable list, giving basic descriptive statistics

Usage

```
summarize(varlist, detail = FALSE)
```

Arguments

`varlist`	a variable list either in "var1 var2 x*" form or, optionally, unquoted.
`detail`	if TRUE, provide a more detailed output for each variable

Value

returns NULL, invisibly

Examples

```
use(cars, clear=TRUE)
summarize(speed)
summarize("speed dist")
```

taildata — *get last few observations*

Description

get last few observations

Usage

```
taildata(num)
```

Arguments

`num`	how many of the last few observations to get

Value

returns last `num` rows of data

Examples

```
use(cars, clear=TRUE)
taildata(5)
```

`tostring` *convert a variable of another type into a string variable*

Description

convert a variable of another type into a string variable

Usage

```
tostring(varlist)
```

Arguments

`varlist`	variables to convert, in the form "var1 var2 var3", or if a single variable, var1 (unquoted) will work as well

Value

returns NULL, invisibly

Examples

```
use(cars, clear=TRUE)
tostring(speed)
listif()
```

`use` *uses a dataset, marking it as the active dataset*

Description

uses a dataset, marking it as the active dataset

Usage

```
use(x, clear = FALSE, type = NULL, ...)
```

Arguments

`x`	usually either a data.frame or a csv/dta filename to be imported. An R function which returns a data.frame can also be specified.
`clear`	if TRUE, erase current data if it already exists (default: FALSE).
`type`	either "csv" or "dta" for loading csv or dta data set
`...`	other options to pass to `read.csv` in case x is a csv file or to `read.dta` or `read.dta13` depending on the type of file being loaded

Value

returns NULL invisibly

Examples

```
library(plm)
data(Produc)
use(Produc, clear=TRUE)
listif()
dropvar(".*")
```

`xtset` *prepares a panel dataset for lag operations*

Description

prepares a panel dataset for lag operations. The lag function in R is simply "lag(var,numlags)". After calling `xtset`, this lag function will work on the panel in the way you would expect.

Usage

```
xtset(timevar, obsvar)
```

Arguments

`timevar`	the name of the variable to for the time dimension
`obsvar`	the name of the variable to use for the observation dimension

Value

returns NULL, invisibly

Examples

```
library(plm)
data(Produc)
use(Produc, clear=TRUE)
xtset(year, state)
gen(Lemp, L(emp))
listif(vars="emp Lemp")
reg(emp, unemp, effect="twoway")
reg(emp, unemp, effect="individual")
reg(emp, unemp, effect="time")
```

Index