Descriptive Statistics

Documentation Version: March 22, 2002

Part Number: 000027

Contents

Chapter 1

Installation

1.1 UNIX

If you are unfamiliar with UNIX, see your system administrator or system documentation for information on the system commands referred to below. The device names given are probably correct for your system.

1.1.1 Download

1. Copy the `.tar.gz` file to `/tmp`.
2. Unzip the file.

   `gunzip` *appxxx*`.tar.gz`
3. `cd` to the **GAUSS** or **GAUSS Engine** installation directory. We are assuming `/usr/local/gauss` in this case.

   ```
   cd /usr/local/gauss
   ```
4. Untar the file.

   `tar xvf /tmp/`*appxxx*`.tar`

1.1.2 Floppy

1. Make a temporary directory.

   ```
   mkdir /tmp/workdir
   ```

2. `cd` to the temporary directory.

   ```
   cd /tmp/workdir
   ```

3. Use `tar` to extract the files.

   `tar xvf` *device_name*

   If this software came on diskettes, repeat the `tar` command for each diskette.

4. Read the README file.

   ```
   more README
   ```

5. Run the `install.sh` script in the work directory.

   ```
   ./install.sh
   ```

   The directory the files are install to should be the same as the install directory of **GAUSS** or the **GAUSS Engine**.

6. Remove the temporary directory (optional).

The following device names are suggestions. See your system administrator. If you are using Solaris 2.x, see Section 1.1.3.

Operating System	3.5-inch diskette	1/4-inch tape	DAT tape
Solaris 1.x SPARC	`/dev/rfd0`	`/dev/rst8`	
Solaris 2.x SPARC	`/dev/rfd0a` (vol. mgt. off)	`/dev/rst12`	`/dev/rmt/1l`
Solaris 2.x SPARC	`/vol/dev/aliases/floppy0`	`/dev/rst12`	`/dev/rmt/1l`
Solaris 2.x x86	`/dev/rfd0c` (vol. mgt. off)		`/dev/rmt/1l`
Solaris 2.x x86	`/vol/dev/aliases/floppy0`		`/dev/rmt/1l`
HP-UX	`/dev/rfloppy/c20Ad1s0`		`/dev/rmt/0m`
IBM AIX	`/dev/rfd0`	`/dev/rmt.0`	
SGI IRIX	`/dev/rdsk/fds0d2.3.5hi`		

1.1.3 Solaris 2.x Volume Management

If Solaris 2.x volume management is running, insert the floppy disk and type

```
volcheck
```

to signal the system to mount the floppy.

The floppy device names for Solaris 2.x change when the volume manager is turned off and on. To turn off volume management, become the superuser and type

```
/etc/init.d/volmgt off
```

To turn on volume management, become the superuser and type

```
/etc/init.d/volmgt on
```

1.2 Windows/NT/2000

1.2.1 Download

Unzip the `.zip` file into the **GAUSS** or **GAUSS Engine** installation directory.

1.2.2 Floppy

1. Place the diskette in a floppy drive.
2. Call up a DOS window
3. In the DOS window log onto the root directory of the diskette drive. For example:

```
A:<enter>
cd\<enter>
```

4. Type: **ginstall** *source_drive target_path*

source_drive	Drive containing files to install with colon included For example: **A:**
target_path	Main drive and subdirectory to install to without a final \ For example: **C:\GAUSS**

A directory structure will be created if it does not already exist and the files will be copied over.

*target_path***src**	source code files
*target_path***lib**	library files
*target_path***examples**	example files

1.3 Differences Between the UNIX and Windows/NT/2000 Versions

- If the functions can be controlled during execution by entering keystrokes from the keyboard, it may be necessary to press *Enter* after the keystroke in the UNIX version.

- On the Intel math coprocessors used by the Windows/NT/2000 machines, intermediate calculations have 80-bit precision, while on the current UNIX machines, all calculations are in 64-bit precision. For this reason, **GAUSS** programs executed under UNIX may produce slightly different results, due to differences in roundoff, from those executed under Windows/NT/2000.

Chapter 2

Descriptive Statistics

The *DESCRIPTIVE STATISTICS* module is a set of procedures which generates basic sample statistics of the variables in GAUSS data sets. These statistics describe the numerical characteristics of the random variables, and provide information for further statistical analysis.

2.1 Getting Started

GAUSS 3.1.0+ is required to use these routines.

2.1.1 README Files

The file `README.ds` contains any last minute information on this module. Please read it before using the procedures in this module.

2.2 Setup

In order to use the procedures in the *DESCRIPTIVE STATISTICS* module, the DSTAT library must be active. This is done by including `dstat` in the **LIBRARY** statement at the top of your program:

```
library dstat,quantal,pgraph;
```

This enables **GAUSS** to find the *DESCRIPTIVE STATISTICS* procedures. If you plan to make any right-hand references to the global variables, you also need the statement:

```
#include dstat.ext;
```

To reset global variables in succeeding executions of the program the following instruction can be used:

```
dstatset;
```

This could be included with the above statements without harm and would insure the proper definition of the global variables for all executions of the program.

2.3 Data Sets

A **GAUSS** data set is a binary disk file. Under DOS, each data set has two disk files associated with it: the first file, containing the data, has a `.dat` extension; the second file, containing the names of the variables associated with each column of the data set, is called the "header file" and has a `.dht` extension. For example, the files `mydata.dat` and `mydata.dht` are the two files associated with the **GAUSS** data set named "mydata". Under UNIX, everything is combined into one file with a `.dat` extension. For the data set "mydata", then, there would only be the file `mydata.dat`.

2.3.1 Data Transformations

It is assumed that the data set for analysis is ready before you call the procedures. If you need to modify your data, use **DATALOOP**. The data loop allows selection of observations, transformation of variables, selection of variables, deletion of missing values, etc. For more details on **DATALOOP**, please consult the **GAUSS** manual.

2.3.2 Creating Data Sets

There are three ways to create a **GAUSS** data set.

1. If you have an ASCII format data file, use the **ATOG** utility to convert it into a **GAUSS** data set. For details, see **ATOG** in the *UTILITIES* section of the **GAUSS** manual.

2. If you have a matrix in memory, use the command **CREATE** or **SAVED** to create a data set. See the *COMMAND REFERENCE* section of the **GAUSS** manual.

3. If you already have a **GAUSS** data set and want to create a new **GAUSS** data set from the existing one, use a data loop. See the *DATA TRANSFORMATIONS* section of the **GAUSS** manual.

To look at a **GAUSS** data set, use the keyword **DATALIST**. The syntax is:

```
DATALIST filename [variables];
```

For details, see **DATALIST** in the **GAUSS** manual.

2.3.3 The Uppercase/Lowercase Convention for Distinguishing Character and Numeric Data

To distinguish numeric variables from character variables in **GAUSS** data sets, **GAUSS** recognizes an "uppercase/lowercase" convention: if the variable name is uppercase, the variable is assumed to be numeric; if it is lowercase, the variable is assumed to be character. **ATOG** implements this convention automatically when you use the $ and # operators to toggle between character and numeric variable names listed in the **INVAR** statement.

When creating a data set using the **SAVED** command, this convention can be established as follows:

```
data = { M 32  21500,
         F 27  36000,
         F 28  19500,
         M 25  32000 };
dataset = "MYDATA";
vnames = { "sex" AGE PAY };
call saved(data,dataset,vnames);
```

It is necessary to put "sex" into quotes in order to prevent it from being forced to uppercase.

The procedure **GETNAME** can be used to retrieve the variable names:

```
names = getname("mydata");
print $names;
```

The names are:

```
sex
AGE
PAY
```

When you are selecting data using **DATALOOP**, the selection is *case-insensitive*. That is:

```
keep  AGE, PAY, SEX;

keep age PAY sex;
```

perform the same selection. Only when you are writing or creating a data set (as the above example using **SAVED** does) is the case of the variable names important.

If you have data sets which do not conform to the uppercase/lowercase convention, set the global variable **__vtype** to specify which of your variables are character and which are numeric.

2.4 Error Codes

If problems are encountered in data being analyzed, the procedures attempt to trap the errors. Errors are handled with the low order bit of the trap flag. Depending on the value of the trap flag, the procedure either sends an error message indicating the nature of the problem and terminates the program, or returns an error code without termination.

TRAP 0 terminate with error message

TRAP 1 return scalar error code

Error codes are particularly helpful if you are running a large program and need to obtain values to pass to other programs.

2.4.1 Tests for Error Codes

If an error is encountered and a procedure returns an error code, it will appear as a missing value. Use the **SCALERR** procedure to return the value of the error code. For example:

```
{ t, n } = crosstab(dataset,varnm);
errcode = scalerr(t);
if errcode /= 0;
    print "Error " errcode " was encountered.";
    end;
endif;
```

The error code returned by **SCALERR** is an integer.

2.4.2 Error Code Values

The following error codes are common to all of the procedures in the *DESCRIPTIVE STATISTICS* module:

1	the data file was not found.
2	undefined variables in input argument.
32	too many categories in frequency or crosstable. If this happens, the user may change the value of the global variable **__maxvec** and try again.
33	cannot create crosstable with only one variable.
41	the global variable **__miss** = 0, but missing values are found.
77	no observations left after deleting missing values.

2.5 Using the On-Line Help System

All of the procedures are automatically accessible through **GAUSS**'s on-line help system. If the DSTAT library is active, pressing Alt-H, then "H" again, then entering the name of a procedure listed in the library displays information about syntax, arguments, and globals used by that procedure.

The help system uses the same search path that **GAUSS** uses when it is attempting to compile your programs. That is, if the help system can find the procedure you request information on, then **GAUSS** can too. This feature can be particularly useful if you are getting "Undefined Symbol" errors, or if it appears that **GAUSS** is finding the wrong definition of a procedure being called.

If, when you attempt to locate the procedure through the help system, nothing appears on the screen or you are returned to your edit file or command mode, then **GAUSS** is not finding the procedure you requested. Check your SRC_PATH and check to see that the library file (with `.lcg` extension on the `lib` subdirectory) is active. If a file is found, check the top of the help screen for the name and location of the file.

2.6 Compatibility with Previous Versions

This new version of the *DESCRIPTIVE STATISTICS* module requires **GAUSS 3.1.0+**. Any programs that you had running under the previous modules may require minor changes before they run successfully under this new version.

If you used **DTRAN** for data transformations, you need to use **DATALOOP** instead.

A new global variable **__range** is now used. This global variable enables the user to specify the range of rows in the data set for analysis. The default is that the whole data set is used. If you need to sample part of your data set, you should set the global variable **__range** before you call the procedures.

Chapter 3

Descriptive Statistics Reference

A summary table listing the main procedures is displayed below.

dstatset

Purpose

Resets *DESCRIPTIVE STATISTICS* global variables to default values.

Library

`dstat`

Format

dstatset;

Remarks

It is generally good practice to put this instruction at the top of all programs that invoke procedures in the *DESCRIPTIVE STATISTICS* module. This prevents globals from being inappropriately defined when a program is run either several times or after another program that also calls *DESCRIPTIVE STATISTICS* procedures.

dstatset calls **GAUSSET**.

Source

`dstatset.src`

corr

Purpose

Computes the correlations for variables in a **GAUSS** data set.

Library

`dstat`

Format

{ *cor*,*vc*,*n*,*nms*,*des* } = **corr(***dataset*,*vars***);**

Input

dataset	string, name of data file.
vars	K×1 character vector, names of variables. – or – K×1 numeric vector, indices of variables. If 0, all variables are included.

Output

cor	K×K matrix, correlations in the order of *vars*.
vc	K×K matrix, covariances in the order of *vars*.
n	scalar, number of observations for listwise correlations. – or – K×K matrix of number of observations for each correlation. A matrix is returned if and only if pairwise correlations are selected.
nms	K×1 character vector, variable names in the order of *vars*.
des	K×7 matrix, descriptive statistics: *des*[**.,1**] Means *des*[**.,2**] Standard deviations *des*[**.,3**] Variances

des[**.,4**] Minimum

des[**.,5**] Maximum

des[**.,6**] Number of valid cases

des[**.,7**] Number of missing cases

Error handling is controlled by the low order bit of the trap flag.

TRAP 0 terminate with error message.

TRAP 1 return scalar error code in all return arguments.

The function **SCALERR** can be used to return the value of the error code. The error codes returned are:

1 data file not found

2 undefined variables in input argument

41 **__miss = 0** but missing values were encountered

77 no cases left after deleting missing observations

Globals

_dscor scalar, if 1, print correlation matrix. Default = 1.

_dstat scalar, if 1, print univariate descriptive statistics. Default = 1.

_dsmmt scalar, if 1, print moment matrix. Default = 0.

_dst scalar, if 1, print t-tests of hypothesis $H_0 : r = 0$. Based on the formula:

$$\sqrt{n-1}\frac{r}{\sqrt{1-r^2}} \sim t(n-2)$$

Default = 1.

_dsvc scalar, if 1, print covariance matrix. Default = 0.

__header string, specifies the format for the output header. **__header** can contain zero or more of the following characters:

t	print title (see **__title**)
l	bracket title with lines
d	print date and time
v	print procedure name and version number
f	print file name being analyzed

Example:

```
__header = "tld";
```

If **__header** == "", no header is printed. Default = "tldvf".

__miss scalar, determines how missing data is handled.

0 Missing values are not checked for, and so the data set must not have any missing observations. This is the fastest option.

1 Listwise deletion. Removes from computation any observation containing a missing value for any variable included in the analysis.

2 Pairwise deletion. **corr** does not require a complete set of data for each observation. This procedure deals separately with each pair of variables in the matrix, computing the covariance and correlation between that pair on the basis of all cases for which there is data. With pairwise deletion, any pair of variables containing missing values is excluded from the computation of their covariance.

Default = 0.

__output scalar, determines printing of intermediate results.

0 nothing is written.

1 serial ASCII output format suitable for disk files or printers.

2 (NOTE: DOS version only) output is suitable for screen only. ANSI.SYS must be active.

Under UNIX, default = 1; under DOS, default = 2.

__range 2×1 vector, the range of records in the data set used for analysis. The first element is the starting row index, the second element is the ending row index. Default is **__range** = { 0, 0 }, the whole data set. For example, if one wants the range of data from row 100 to the end of data, then **__range** should be set as:

```
__range = { 100, 0 };
```

__row scalar, specifies how many rows of the data set are read per iteration of the read loop. If **__row** = 0, the number of rows to be read is calculated by **corr**. Default = 0.

__rowfac scalar, "row factor". If a *DESCRIPTIVE STATISTICS* procedure fails due to insufficient memory while attempting to read a **GAUSS** data set, then **__rowfac** may be set to some value between 0 and 1 to read a *proportion* of the original number of rows of the **GAUSS** data set. For example, setting

```
__rowfac = 0.8;
```

causes **GAUSS** to read in only 80% of the rows that were originally calculated.

This global only has an effect when **__row** = 0.

Default = 1.

__title string, this is the title used by **__header**. Default = "".

__weight string or scalar, name or index of weight variable. By default, unweighted correlations are calculated.

Example

```
library dstat;
#include dstat.ext;
dstatset;
var = { pub1, pub3, pub6, job, enrol };
__weight = "PUB1";
__miss = 2;
__header = "tl";
__title =  "corr.e:  WEIGHTED PAIRWISE - ALL OPTIONS";
output file = corr.out reset;
call corr("scigau",var);
output off;
```

Source

destat.src

crosstab

Purpose

Creates contingency tables from raw or weighted data contained in a **GAUSS** data set.

Library

`dstat`

Format

`{ t,n }` = **crosstab(***dataset*,*vars*);

Input

dataset	string, name of data set.
vars	K×1 character vector of variable names for the table. – or – K×1 numeric vector of column indices of variables for the table. K must be at least 2. The first variable in *vars* is the row variable, the second variable is the column variable, the remaining variables are levels of the control variables.

Output

t	N×K matrix, table indices.
n	N×1 vector, table counts.

Error handling is controlled by the low order bit of the trap flag.

TRAP 0 terminate with error message.

TRAP 1 return scalar error code in all return arguments.

The function **SCALERR** can be used to return the value of the error code. The error codes returned are:

1 data file not found

2 undefined variables in input argument

32 too many cells in crosstable

33 cannot create crosstable with only one variable

Globals

_dscase scalar, if 1, case sensitivity turned on for character variables. Default = 0.

_dscol scalar, number of columns to print per section of a table. If a table is large, this allows printing it in sections. Default = 6.

_dscolp scalar, if 1, list column percentages. Default = 0.

_dspause scalar, if 1, pause between tables. Default = 1.

_dsrow scalar, number of rows to print per section of a table. If a table is large, this allows printing it in sections. Default = 3.

_dsrowp scalar, if 1, list row percentages. Default = 0.

_dstat scalar, if 1, print statistics. See documentation of **tblstat** for details on statistics printed. Default = 1.

_dstotp scalar, if 1, list total percentages. Default = 0.

__miss scalar, determines how missing data are handled.

0 Missing values are included in the table as a separate category if there are missing observations in the data.

1 Listwise deletion. Removes from computation any observation with a missing value for any variable included in the analysis.

Default = 0.

__output scalar, determines printing of intermediate results.

0 nothing is written.

1 serial ASCII output format suitable for disk files or printers.

2 (NOTE: DOS version only) output is suitable for screen only. ANSI.SYS must be active.

Under UNIX, default = 1; under DOS, default = 2.

__range 2×1 vector, the range of records in the data set used for analysis. The first element is the starting row index, the second element is the ending row index. Default is **__range** = { 0, 0 }, the whole data set. For example, if one wants the range of data from row 100 to the end of data, then **__range** should be set as:

```
__range = { 100, 0 };
```

__row scalar, specifies how many rows of the data set are read per iteration of the read loop. If **__row** = 0, the number of rows to be read is calculated by **crosstab**. Default = 0.

__rowfac scalar, "row factor". If **crosstab** fails due to insufficient memory while attempting to read a **GAUSS** data set, then **__rowfac** may be set to some value between 0 and 1 to read a *proportion* of the original number of rows of the **GAUSS** data set. For example, setting

```
__rowfac = 0.8;
```

causes **GAUSS** to read in only 80% of the rows that were originally calculated.

This global only has an effect when **__row** = 0.

Default = 1.

__vtype scalar or vector, indicates the types of the variables included in the analysis. Set **__vtype** only if you are NOT following the uppercase/lowercase convention.

If you have:

all character data	set **__vtype** = 0.
all numeric data	set **__vtype** = 1.
mixed data	set **__vtype** to a vector of 0's and 1's, 0 for character variables, 1 for numeric.

If you have mixed data, **__vtype** should be a K×1 or a (K+1)×1 vector, depending on whether or not a weight variable is specified (see **__weight** below). Set the elements of **__vtype** as follows:

[1:K]	types of *vars* variables
[K+1]	type of **__weight** variable (if specified)

By default, **__vtype** = −1. That is, data type is determined by looking at the case of each variable name.

See section 2.3.3 for a discussion of the uppercase/lowercase convention.

__weight scalar or string, name or index of weight variable. By default, no weighting is used.

Remarks

This procedure handles both character and numeric data. To indicate the type of each variable to be included in the analysis, you may either follow the uppercase/lowercase convention (see Section 2.3.3), or set the global variable **__vtype** as described above under **Globals**.

crosstab constructs a K-way table from variables $V_i(i = 1, K)$ in *vars*, with variable V_i having K_i categories. The resulting table contains $N = K_1 \times K_2 \times \ldots K_K$ cells. The observed variables for this table are contained in the N×1 vector n. The N cell indices associated with n are contained in the K×N matrix t, where element t_{ij} is the category level of variable V_j for n_i.

The matrices t and n that are returned by **crosstab** are in the appropriate form for input to the programs in the *LOGLINEAR ANALYSIS* module.

A table can contain at most **MAXVEC** cells.

Example

Print a two-way table.

```
library dstat;
#include dstat.ext;
dstatset;
dataset = "mydata";
call crosstab(dataset,2|5); /* crosstab 2nd variable by 5th */
```

Source

`crosstab.src`

See also

tblstat, **freq**

freq

Purpose

Computes frequency distributions for variables contained in a **GAUSS** data set.

Library

`dstat`

Format

`{` *cats*,*ncats*,*freqn* `}` = **freq(***dataset*,*vars***);**

Input

dataset	string, name of data set.
vars	K×1 character vector, names of variables – or – K×1 numeric vector, indices of variables for which frequency distributions are requested. If $vars = 0$, all the variables in the data set are used.

Output

cats	L×1 vector of categories for each variable, where $L = \sum_{i=1}^{K} cat_i$, and cat_i is the number of categories of the i^{th} variable.
ncats	K×1 vector of $(cat_1, cat_2, \ldots, cat_K)$, where each element is the number of categories for the corresponding variable.
freqn	L×1 vector of frequencies for each variable.

Error handling is controlled by the low order bit of the trap flag.

TRAP 0 terminate with error message.

TRAP 1 return scalar error code in all return arguments.

The function **SCALERR** can be used to return the value of the error code. The error codes returned are:

1 data file not found

2 undefined variables in input argument

32 too many cells in frequency

77 no cases left after deleting missing observations

Globals

_dscase scalar, if 1, case sensitivity turned on for character variables. Default = 0.

_dspause scalar, if 1, pause between tables. Default = 1.

_dstat scalar, if 1, print descriptive statistics. Default = 1.

___miss scalar, determines how missing data are handled.

0 Missing values are included in the table as a separate category if there are missings in the data.

1 Listwise deletion. Removes from computation any observation with a missing value for any variable included in the analysis.

Default = 0.

___output scalar, determines printing of intermediate results.

0 nothing is written.

1 serial ASCII output format suitable for disk files or printers.

2 (NOTE: DOS version only) output is suitable for screen only. ANSI.SYS must be active.

Under UNIX, default = 1; under DOS, default = 2.

___range 2×1 vector, the range of records in data set used for analysis. The first element is the starting row index, the second element is the ending row index. Default is **___range** = { 0, 0 }, the whole data set. For example, if one wants the range of data from row 100 to the end of data, then **___range** should be set as:

```
__range = { 100, 0 };
```

___row scalar, specifies how many rows of the data set are read per iteration of the read loop. If **___row** = 0, the number of rows to be read is calculated by **freq**. Default = 0.

___rowfac scalar, "row factor". If **freq** fails due to insufficient memory while attempting to read a **GAUSS** data set, then **___rowfac** may be set to some value between 0 and 1 to read a *proportion* of the original number of rows of the **GAUSS** data set. For example, setting

```
__rowfac = 0.8;
```

causes **GAUSS** to read in only 80% of the rows that were originally calculated.

This global only has an effect when **__row** = 0.

Default = 1.

__sort scalar, if 1, output is sorted by the names of the variables in *vars*. Default = 0.

__vtype scalar or vector, indicates the types of the variables included in the analysis. Set **__vtype** only if you are NOT following the uppercase/lowercase convention.

If you have:

all character data	set **__vtype** = 0.
all numeric data	set **__vtype** = 1.
mixed data	set **__vtype** to a vector of 0's and 1's, 0 for character variables, 1 for numeric.

If you have mixed data, **__vtype** should be a K×1 or a (K+1)×1 vector, depending on whether or not a weight variable is specified (see **__weight** below). Set the elements of **__vtype** as follows:

[1:K]	types of *vars* variables
[K+1]	type of **__weight** variable (if specified)

By default, **__vtype** = −1. That is, data type is determined by looking at the case of each variable name.

See section 2.3.3 for a discussion of the uppercase/lowercase convention.

__weight scalar or string, the name or index of the weight variable. By default, no weighting is used in calculations.

Remarks

This procedure handles both character and numeric data. To indicate the type of each variable to be included in the analysis, you may either follow the uppercase/lowercase convention (see section 2.3.3), or set the global variable **__vtype** as described above under **Globals**.

If there are missing data for a variable, counts excluding the missing category are also made.

For each variable in *vars*, **freq** returns a sorted list of all categories of that variable, the number of categories for that variable, and the number of cases in each category.

The procedure **getfreq** gets the categories and frequencies for a specified variable.

The total number of cells in the frequency distributions for all variables in *vars* cannot exceed **MAXVEC**.

Example

Using **freq**:

```
library dstat;
#include dstat.ext;
dstatset;
dataset = "mydata";
call freq(dataset,2|5);  /* <=== Frequencies for the   */
                         /*      2nd and 5th variables */
```

Using **freq**, **getfreq** and **HISTF**:

```
library dstat,pgraph;
#include dstat.ext;
graphset;
dstatset;
print "FR4.E:   Using getfreq and HISTF";
print;
dataset = "freq";
output file = fr1.out reset;
__miss = 1;      /* get rid of missing category */
{ cats,ncats,freqs } = freq(dataset,1|2|3);
output off;
/* get frequencies for var 1 */
histf(getfreq(1,cats,ncats,freqs));
```

Source

`freq.src`

See also

getfreq, **freqstat**

freqstat

Purpose

Prints percentages, descriptive statistics and simple histogram given category values and number of cases in each category.

Library

`dstat`

Format

freqstat(*nm*,*freq*,*val*,*c***);**

Input

nm	string, name of variable.
freq	L×1 vector, number of observations in each category.
val	L×1 character vector, label of each category. – or – L×1 numeric vector, value of each category.
c	scalar, 1 if numeric variable, 0 if character variable.

Output

None.

Globals

_dsfreq	scalar, if 1, print frequencies. Default = 1.
_dstat	scalar, if 1, print descriptive statistics. Default = 1.

Remarks

freqstat prints frequencies, percentages, descriptive statistics and a histogram.

freqstat works with matrices in memory; if your data is in a GAUSS data set, you should use the procedure **freq**, which works with data sets.

Example

```
library dstat;
dstatset;

output file = frst.out reset;
freq = { 10, 3, 4 };
cats = { 1, 2, 3 };
freqstat("VAR", freq, cats, 1);
output off;
```

Source

freqstat.src

See also

freq

getfreq

Purpose

Gets frequencies and categories for a particular variable from the results returned by **freq**.

Library

`dstat`

Format

{ *vfreq*,*vcat* } = **getfreq(***var*,*cats*,*ncats*,*freqs***);**

Input

var	scalar, the index of the variable for which frequencies are desired.
cats	K×1 character vector, labels – or – K×1 numeric vector, category values for all counts in the *freqs* vector.
ncats	L×1 vector, number of categories associated with each variable in the *freqs* vector.
freqs	M×1 vector, frequencies associated with *cats* and *ncats*.

Output

vfreq	Q×1 vector, frequencies associated with specified variable.
vcat	P×1 vector, categories associated with specified variable.

Remarks

The variables *cats*, *ncats* and *freqs* are the results returned from **freq**. **freq** returns vectors for all variables specified. **getfreq** returns the frequencies for a specified variable.

Example

```
library dstat,pgraph;
#include dstat.ext;
dstatset;
{ cats,ncats,freqs } =  freq(dataset,1|2|3);

{ f,c } =
   getfreq(1,cats,ncats,freqs);   /* <=== Get frequencies */
                                  /*      for VAR 1       */

histf(f,c);   /* <=== Plot histogram for VAR 1 */
```

Source

getfreq.src

See also

freq

means

Purpose

Computes the descriptive statistics for variables in a **GAUSS** data set.

Library

`dstat`

Format

`{` *nms*,*mn*,*std*,*min*,*max*,*valid*,*missing* `}` **= means(***dataset*,*vars***);**

Input

dataset	string, name of data file.
vars	K×1 character vector, names of variables. – or – K×1 numeric vector, indices of variables. If *vars* = 0, all variables are included.

Output

nms	K×1 character vector of variable names.
mn	K×1 vector of means.
std	K×1 vector of standard deviations.
min	K×1 vector of minimum values.
max	K×1 vector of maximum values.
valid	K×1 vector of the number of valid cases for each selected variable.
missing	K×1 vector of the number of missing cases in each selected variable.

Error handling is controlled by the low order bit of the trap flag.

TRAP 0 terminate with error message.

TRAP 1 return scalar error code in all return arguments.

The function **SCALERR** can be used to return the value of the error code. The error codes returned are:

1 data file not found

2 undefined variables in input argument

77 no cases left after deleting missing observations

Globals

___miss scalar, determines how missing data are handled.

0 Missing values are not checked for, so the data set must not have any missing observations. This is the fastest option.

1 Listwise deletion. Removes from computation any observation with a missing value for any variable included in the analysis.

2 Pairwise deletion. **means** does not require a complete set of data for each observation. This procedure deals separately with each variable in the matrix, computing the mean of that variable on the basis of all cases for which there is data. With pairwise deletion, missing values are excluded from computation, so the number of cases used in calculating the mean of each variable differs from variable to variable.

Default = 0.

___output scalar, determines printing of intermediate results.

0 nothing is written.

1 serial ASCII output format suitable for disk files or printers.

2 (NOTE: DOS version only) output is suitable for screen only. ANSI.SYS must be active.

Under UNIX, default = 1; under DOS, default = 2.

___range 2×1 vector, the range of records in data set used for analysis. The first element is the starting row index, the second element is the ending row index. Default is **___range** = { 0, 0 }, the whole data set. For example, if one wants the range of data from row 100 to the end of data, then **___range** should be set as:

```
__range = { 100, 0 };
```

___row scalar, specifies how many rows of the data set are read per iteration of the read loop. If **___row** = 0, the number of rows to be read is calculated by **means**. Default = 0.

__rowfac scalar, "row factor". If **means** fails due to insufficient memory while attempting to read a **GAUSS** data set, then **__rowfac** may be set to some value between 0 and 1 to read a *proportion* of the original number of rows of the **GAUSS** data set. For example, setting

```
__rowfac = 0.8;
```

causes **GAUSS** to read in only 80% of the rows that were originally calculated.

This global only has an effect when **__row** = 0.

Default = 1.

__sort scalar, if 1, output is sorted by the names of the variables in *vars*. Default = 0.

__vtype scalar or vector, indicates the types of the variables included in the analysis. Set **__vtype** only if you are NOT following the uppercase/lowercase convention.

If you have:

all character data	set **__vtype** = 0.
all numeric data	set **__vtype** = 1.
mixed data	set **__vtype** to a vector of 0's and 1's, 0 for character variables, 1 for numeric.

If you have mixed data, **__vtype** should be a K×1 or a (K+1)×1 vector, depending on whether or not a weight variable is specified (see **__weight** below). Set the elements of **__vtype** as follows:

[1:K]	types of *vars* variables
[K+1]	type of **__weight** variable (if specified)

By default, **__vtype** = −1. That is, data type is determined by looking at the case of each variable name.

See section 2.3.3 for a discussion of the uppercase/lowercase convention.

__weight string or scalar, name or index of weight variable. By default, calculations are unweighted.

Remarks

This procedure handles both character and numeric data. To indicate the type of each variable to be included in the analysis, you may either follow the uppercase/lowercase convention (see section 2.3.3), or set the global variable **__vtype** as described above under **Globals**.

Example

```
library dstat;
#include dstat.ext;
dstatset;
dsn = "cook";
output file = means.exp reset;
{ nms,mn,std,min,max,valid,missing } = means(dsn,0);
output off;
```

Source

destat.src

tblstat

Purpose

Computes statistics and measures of association for an I×J contingency table.

Library

`dstat`

Format

tblstat(x**);**

Input

x	I×J matrix of cell frequencies.

Output

Measures of fit and association for table are sent to the output device.

Remarks

The following statistics are computed and printed.

Statistic	Reference
Pearson's Chi Square	BFH, 124
Likelihood Chi Square	BFH, 124
Yate's Corrected Chi Square (2×2)	BFH, 124
McNemar's Symmetry Chi-Square	
Phi	Agresti, 175
Cramer's V (not for 2×2)	BFH, 386
Contingency (Pearson's P)	BFH, 385
Spearman's Rho (Correlation)	BFH, 381
Cohen's Kappa (symmetric tables)	BFH, 395
Yule's Q (2×2)	BFH, 378
Yule's Y (2×2)	BFH, 378
Goodman-Kruskal Gamma	Agresti, 159
Kendall's Tau-B	Agresti, 161
Stuart's Tau-C	Agresti, 177
Somer's D	Agresti, 161
Lambda	BFH, 388
Uncertainty	

Agresti: Agresti, Alan. 1984. *Analysis of Ordinal Categorical Data*. New York: John Wiley and Sons.

BFH: Bishop, Yvonne, Stephen Fienberg and Paul Holland 1975. *Discrete Multivariate Analysis: Theory and Practice*. Cambridge, Mass.: MIT Press.

Example

```
library dstat;
dstatset;
x = { 10 23,
      34 47 };
tblstat(x);
```

Source

tblstat.src

See also

crosstab

ttest

Purpose

Tests the differences of means between two groups.

Library

`dstat`

Format

{ *varl*,*descstat*,*mntest*,*vartest* } = **ttest(***dataset*,*grpvar*,*varnm***);**

Input

dataset string, name of data file.

grpvar string, name of the group variable.
– or –
scalar, index of the group variable.

grpvar may be the name of a variable that contains character data.

varnm K×1 character vector, names of variables to be tested.
– or –
K×1 numeric vector, indices of variables to be tested.
– or –
scalar 0, all variables but *grpvar* are tested.

Output

varl (K+1)×1 character vector, variable names.

descstat L×6 matrix of descriptive statistics, where

descstat[**.,1**] means of group 0
descstat[**.,2**] means of group 1
descstat[**.,3**] standard deviation of group 0
descstat[**.,4**] standard deviation of group 1
descstat[**.,5**] number of valid cases

descstat[**.,6**] number of missing cases

mntest L×6 matrix, results of test of the hypothesis that the true means are the same. Columns are:

mntest[**.,1**] t value for assumption that the two groups have equal variances

mntest[**.,2**] degrees of freedom for equal variances

mntest[**.,3**] probability for equal variances

mntest[**.,4**] t value for unequal variances

mntest[**.,5**] degrees of freedom for unequal variances

mntest[**.,6**] probability for unequal variances

vartest L×4 matrix of results from test of variances. Columns are:

vartest[**.,1**] F value for test of differences

vartest[**.,2**] degrees of freedom for equal variances

vartest[**.,3**] degrees of freedom for unequal variances

vartest[**.,4**] probability of the F statistic

Error handling is controlled by the low order bit of the trap flag.

TRAP 0 terminate with error message.

TRAP 1 return scalar error code in all return arguments.

The function **SCALERR** can be used to return the value of the error code. The error codes returned are:

1 data file not found

2 undefined variables in input argument

41 **__miss = 0** but missing values were encountered

77 no cases left after deleting missing observations

Globals

_dsgrpnm 2×1 character vector of names of groups. By default, the names "Group 0" and "Group 1" are used.

_dscut scalar or 2×1 vector.

The groups are defined by **_dscut** and the conditioning variable *grpvar* as follows:

scalar **_dscut**, numeric *grpvar*:

first group	$grpvar <$ **_dscut**
second group	$grpvar \geq$ **_dscut**

scalar **_dscut**, character *grpvar*:

first group	$grpvar =$ **_dscut**
second group	$grpvar \neq$ **_dscut**

vector (2×1) **_dscut**, character or numeric *grpvar*:

first group	$grpvar =$ **_dscut[1]**
second group	$grpvar =$ **_dscut[2]**

Default = 0.

__miss scalar, determines how missing data are handled.

0 Missing values are not checked for, and so the data set must not have any missing observations. This is the fastest option.

1 Listwise deletion. Removes from computation any observation containing a missing value for any variable included in the analysis.

2 Pairwise deletion. **ttest** does not require a complete set of data for each observation. This procedure deals separately with each pair of variables in the matrix, computing the covariance and correlation between that pair on the basis of all cases for which there is data. With pairwise deletion, any pair of variables containing missing values is excluded from the computation of their covariance.

Default = 0.

__output scalar, determines printing of intermediate results.

0 nothing is written.

1 serial ASCII output format suitable for disk files or printers.

2 (NOTE: DOS version only) output is suitable for screen only. ANSI.SYS must be active.

Under UNIX, default = 1; under DOS, default = 2.

__range 2×1 vector, the range of records in data set used for analysis. The first element is the starting row index, the second element is the ending row index. Default is **__range** = { 0, 0 }, the whole data set. For example, if one wants the range of data from row 100 to the end of data, then **__range** should be set as:

```
__range = { 100, 0 };
```

__row scalar, specifies how many rows of the data set are read per iteration of the read loop. If **__row** = 0, the number of rows to be read is calculated by **ttest**. Default = 0.

__rowfac scalar, "row factor". If **ttest** fails due to insufficient memory while attempting to read a **GAUSS** data set, then **__rowfac** may be set to some value between 0 and 1 to read a *proportion* of the original number of rows of the **GAUSS** data set. For example, setting

```
__rowfac = 0.8;
```

causes **GAUSS** to read in only 80% of the rows that were originally calculated.

This global only has an effect when **__row** = 0.

Default = 1.

__vtype scalar or vector, indicates the types of the variables included in the analysis. Set **__vtype** only if you are NOT following the uppercase/lowercase convention.

If you have:

all character data	set **__vtype** = 0.
all numeric data	set **__vtype** = 1.
mixed data	set **__vtype** to a vector of 0's and 1's, 0 for character variables, 1 for numeric.

If you have mixed data, **__vtype** should be a (K+1)×1 vector. Set the elements of **__vtype** as follows:

[1]	type of *grpvar* variable
[2:K+1]	types of *varnm* variables

By default, **__vtype** = −1. That is, data type is determined by looking at the case of each variable name.

See section 2.3.3 for a discussion of the uppercase/lowercase convention.

Remarks

This procedure handles both character and numeric data. To indicate the type of each variable to be included in the analysis, you may either follow the uppercase/lowercase convention (see section 2.3.3), or set the global variable **__vtype** as described above under **Globals**.

Descriptive statistics for each group and tests of differences of means are sent to the output device.

Example

```
library dstat;
dstatset;
dataset = "scigau";
vars = { cit1, pub1 };
```

```
grpvar = { job };
_dsgrpnm = { Lo_Job, Hi_Job };
__miss = 2;
_dscut = 2;
output file = ttest.out reset;
{ desc,mtest,vtest } = ttest(dataset,grpvar,vars);
output off;
```

Source

ttest.src

Index

S

T

U

V

W