STATS:

a NetLogo extension for statistical procedures

Version 2.1.1

The "**stats**" extension provides basic statistical functions for data generated within NetLogo models. **stats** maintains an internal data table of observations on an arbitrary number of variables and allows the user to find such statistics for the data as the means, medians, quantiles, percentiles, standard deviations, and variance-covariance and correlation matrices; to regress one or more variables on another; and to forecast the value of any variable into the future based on past observations. Regression statistics include R^2, Adjusted R^2, F and the probability of F, and for each of the coefficients, the standard error, T statistic and probability of the T statistic. **stats** also provides the areas under the normal, student, binomial, Chi-Square, gamma and beta distributions, and their inverses.

Although **stats** can be used to analyze the overall results of a run of a model, it is intended mostly for the use of "smart" agents who gather and analyze data in order to make decisions. Agents can maintain their private data tables, or use tables shared as globals, or both.

The stats extension makes use internally of matrix algebra, and therefore uses the same Jama matrix package as is used by the **matrix** extension. **Jama-1.0.3.jar** (distributed with this extension or copied from the matrix subdirectory of the NetLogo extensions folder) must be placed in the same subdirectory as **stats.jar** itself. However, one need not include the **matrix** extension in one's NetLogo model to use the **stats** extension. The stats extension also uses **colt.jar**, version 1.2.0, from the Colt Project (http://acs.lbl.gov/software/colt/). Colt provides a set of Open Source Libraries for High Performance Scientific and Technical Computing in Java.[1] **colt.jar** is distributed with this extension and is also available from the Colt Project itself. It must be placed in the same subdirectory as **stats.jar**.

[1]The colt.jar distribution includes the following license information.

1. Extension procedures

stats contains the following procedures.

1.1 Data definition and manipulation

stats:newtable

a reporter that creates a new, empty data table and reports a pointer to it. For example:

```
let tbl stats:newtable
```

Once can create as many different data tables as one wants.

Stats:add *stats-table list*

a reporter that adds the data in *list* to the data table pointed to by *stats-table*. For example:

```
stats:add tbl (list nturtles temperature food-supply ntrees ….)
```

where the list of values to be added to the data table is given by the current values of nturtles, temperature, etc. Note that once the first entry into the data table has been made, all subsequent entries must contain the same number of variables in the same order as the first. Each new entry to the data table forms a new row of the table.

stats:set-names *stats-table list*

a command that allows a name to be assigned to each variable in the data table. *list* is a list of strings. The names of the variables should be listed in the same order as they appear in the data table. For example:

```
stats:set-names tbl ["nturtles" "temperature" "food-supply" "ntrees"…]
```

The variable names may be used in specifying the variables to be used in the various **stats** procedures, and they are used to label rows and columns in **stats:print-data**, **stats:print-correlation**, and **stats:print-covariance**.

stats:get-names *stats-table*

a reporter that returns the list of variable names for this data table as they were specified in **stats:set-names**. For example:

```
let variable-names stats:get-names tbl
```

stats:newtable-from-row-list *list*

a reporter that creates a new data table from the data in the nested list, *list*, and returns a pointer to the table. The data in *list* must be in the form of a list of lists, with each sublist being an observation on the n variables to be included in the table. Each sublist is therefore a row of the data table. For example:

```
let tbl stats:newtable-from-rowlist [[1 2 3] [4 5 6] [7 8 9]]
```

will create a new data table with the first row (observation) being [1 2 3], the second being [4 5 6], etc.

stats:get-data-as-list *stats-table*

a reporter that retrieves the data table, row by row, as a nested list. This list may be used as an input to create a new data table using **stats:newtable-from-list,** to retrieve a particular observation, or to save a data table before an export-world[2]. For example:

```
let data-list stats:get-data-as-list tbl
```

In data-list, the number of observations is

length data-list,

the number of variables is

length item 0 data-list,

and the value of the jth variable in the ith observation is

item j item i data-list.

stats:use-most-recent *stats-table number*

a command that instructs the operations that report statistics on the data to use only the most recent *number* observations. For example:

```
stats:use-most-recent tbl 20
```

will result in all subsequent procedures involving regressions, forecasts, correlations, etc., using only the most recent 20 observations in the data table. (If there are fewer than *number* observations in the data table, all are used.) By default, all observations are used and the default may be reset by specifying 0 as *number*.

stats:get-observations *stats-table variable*

a reporter that returns as a simple list the observations for the given variable in the data table. *variable* may be expressed as a variable number or a variable name. (Variables in the data table are numbered from zero.) All the observations are included unless the number has been limited by a prior **stats:use-most-recent**. For example:

```
let temp stats:get-observations tbl "temperature"
let temp stats:get-observations tbl 1
```

stats:get-nobs *stats-table*

a reporter that returns the total number of observations in the data table. For example:

```
let n stats:get-nobs tbl
```

[2] See the section below in Export/Import World.

stats:get-nobs-used *stats-table*

a reporter that returns the number of observations set for this table in the most recent **stats:use-most-recent**. Note that if the number returned is zero, all observations are used. For example:

```
let nused stats:get-nobs-used tbl
```

stats:trim-data *stats-table number*

a command that reduces the size of the data table to the most recent *number* observations. For example:

```
stats:trim-data tbl 100
```

purges all but the most recent 100 observations from the data table.

Note that it is seldom necessary to delete a data table. However, if there is a need to delete a table during a run, perhaps to save memory when it is no longer needed or after saving its contents before an export-world, simply set its pointer to zero. E.g.,

```
set tbl 0
```

This will delete the reference to the table and it will be swept up in the Java garbage collector in due course.

1.2 Data description

stats:means *stats-table*

stats:medians *stats-table*

stats:stddevs *stats-table*

reporters that return a list of (respectively) the means, medians and standard deviations of all the variables in the data table. Note that in keeping with the equivalent NetLogo reporter, the standard deviation is by default calculated using Bessel's correction for a the standard deviation of a sample. This may be changed with the **stats:useBessel?** command. The number of observations used may be limited by **stats:use-most-recent**. For example:

```
let medians stats:medians tbl
```

yields a list of the medians of each variable in the data table. If the mean, median or standard deviation of only one variable (say temperature) is needed in one operation rather calculating it for all the variables and then than extracting it from the list, one can use the corresponding NetLogo reporter on the observations for that variable. E.g.,

```
let med median stats:get-observations tbl "temperature".
```

stats:quantile *stats-table variable pcnt*

a reporter that returns the value of the observation for the specified variable that is greater than *pcnt* percent of the observations, where *pcnt* is a real number between 0 and 100. (The value returned may be an interpolation between two actual observations and therefore itself not equal to an actual observation.) *variable* may be expressed as a variable number or a variable name. For example:

```
let quant stats:quantile tbl "nturtles" 50
```

returns the median of the set of observations on the variable nturtles. The number of observations used may be limited by **stats:use-most-recent**.

stats:quantiles *stats-table variable ngroups*

a reporter that divides the range of observation on the specified variable into *ngroups* groups and returns a list of length n+1 such that $\frac{0*n}{(n+1)}$ of the observations are less than the first item of the list, $\frac{1*n}{(n+1)}$ of the observations are below the second item, $\frac{2*n}{(n+1)}$ of the observations are below the third item, etc. The first item of the list is therefore the smallest observation and the n+1 item of the list is the largest. (Other than the first and last items of the list, the values returned may be interpolations between two actual observations and therefore not equal to actual observations.) *variable* may be expressed as a number or a variable name. For example:

```
let quant stats:quantiles tbl "nturtles" 4
```

returns the quartiles of the observations on the variable nturtles. The first item of the list returned is the minimum value, the second the first quartile, the third the second quatile (median), the fourth the third quartile, and the fifth the maximum value. (**stats:quantiles tbl "nturtles" 1** returns a two-item list with the minimum and the maximum observations as items 0 and 1 respectively.) The number of observations used may be limited by **stats:use-most-recent**.

stats:percentile *stats-table variable value*

a reporter that returns the percentage of the observations for the specified variable that are less than or equal to *value*, where the percentage returned is a real number between 0 and 100. (The percentage returned may be an interpolation if the value given lies between two actual observations.) *variable* may be expressed as a number or a variable name. For example:

```
let pcnt stats:percentile tbl "nturtles" 20
```

returns the percentage of observations on the variable nturtles that lie below 20. The number of observations used may be limited by **stats:use-most-recent**.

stats:covariance *stats-table*

a reporter that returns the variance-covariance matrix for all the variables in the data table as a row-by-row nested list. Note that in keeping with the NetLogo reporter for the standard deviation, the variance-covariance matrix is, by default, calculated using Bessel's correction for a the variance-covariance of a sample. This may be changed with the **stats:useBessel?** command. The number of observations used may be limited by **stats:use-most-recent**. For example:

```
let covar stats:covariance tbl
```

The correlation covariance between the 2nd and 5th variable in the data table would be

```
item 1 item 4 covar
```
(or **item 4 item 1 covar**)

Note that the variance-covariance matrix is square and thus symmetric around the diagonal. Remember too that NetLogo counts from zero so the 2nd variable (row) would be item1 in the nested list and the 5th variable (column) would be item 4 of that row.

stats:useBessel? *stats-table boolean*

a command that sets or unsets the use of Bessel's correction. By default, variances, covariances and standard deviations are calculated using Bessel's correction. If instead they should be calculated without Bessel's correction, i.e., for a population rather than a sample, this may indicated with the command:

```
stats:useBessel? tbl false
```

The use of the Bessel correction is turned back on with

```
stats:useBessel? tbl true
```

stats:correlation *stats-table*

a reporter that returns the correlation matrix for all the variables in the data table as a row-by-row nested list. The number of observations used may be limited by **stats:use-most-recent**. For example:

```
let correl stats:correlation tbl
```

The correlation coefficient between the 2nd and 5th variable in the data table would be

```
item 1 item 4 correl
```
(or **item 4 item 1 correl**)

Note that the correlation matrix is square and thus symmetric around the diagonal. Remember too that NetLogo counts from zero so the 2nd variable (row) would be item1 in the nested list and the 5th variable (column) would be item 4 of that row.

1.3 Regression and forecasting

stats:regress-all *stats-table*

a reporter that returns a list of regression coefficients. The dependent variable in the regression is the first variable in the data table and all the other variables are included as

independent variables. Therefore, if there are n variables in the table, the list of coefficients contains n entries: the regression constant and the coefficients on the n-1 independent variables in the same order as they are contained in the data table. The number of observations used may be limited by **stats:use-most-recent**. For example:

let coeffs stats:regress-all tbl

Once the regression has been run, a number of regression statistics may be obtained. See **stats:get-rstats** and **stats:get-rcstats**, below.

stats:regress-on *stats-table list*

a reporter that also returns a list of regression coefficients. In this case *list* contains a list of the variables in the data table to use in the regression. The first variable in the list is the dependent variable and the others are the independent variables. The list of variables may be either a list of variable names (assuming **stats:set-names** has previously been used to enter the names of the variables in the data table), or a list of variable numbers, where the first variable in the table is 0, the second is 1, etc. The list of coefficients is in the same order as the variables in *list*, with the regression constant first. The number of observations used may be limited by **stats:use-most-recent**. For example:

let coeffs stats:regress-on tbl ["temperature" "ntrees" "nturtles"]
let coeffs stats:regress-on tbl [1 3 0]

Both run a regression with the temperature variable in the data table being the dependent variable (since the prior **stats:set-name** had given that variable the name "temperature" and the temperature variable in the table is in position 1 counting from 0), and using "ntrees" and "nturtles" (in positions 3 and 0 in the data table) as independent variables. Note that the variable references in *list* must be either all names or all position numbers. Once the regression has been run, a number of regression statistics may be obtained. See **stats:get-rstats** and **stats:get-rcstats**, below.

stats:get-rstats *stats-table*

a reporter that returns a list of statistics concerning the most recent regression. The list contains 11 items, the regression R^2 (item 0) and adjusted R^2 (item 1), the regression F statistic (item 2), the probability of F (item 3), the standard error of estimate (item 4), the total degrees of freedom (item 5), the regression degrees of freedom (item 6), the error degrees of freedom (item 7), the total sum of squares (item 8), the regression sum of squares (item 9), and the error sum of squares (item 10). For example:

let rstats stats:get-rstats tbl

gets the list of regression statistics, and

let F item 2 rstats and **let PofF item 3 rstats**

pulls out the regression F statistic and its associated probability.

stats:get-rcstats *stats-table*

a reporter that returns a nested list of statistics concerning the coefficients of the most recent regression. The first sublist contains the P values for the T statistics of each of the regression coefficients. The second sublist contains the T statistics themselves. The third sublist contains the standard error of each coefficient. Each sublist begins with the P, T or se value for the constant term, followed by those for the independent variables in the same order that they entered the regression. Therefore, if **stats:regress-on** was used, the order of the variables in the variable list sets the order of the P, T and se values in each sublist. For example:

```
let rcstats stats:get-rcstats tbl
```

gets the nested list of P, T and se values, and

```
item 3 item 0 rcstats
```

extracts the P value for the coefficient of the 3rd independent variable in the regression.

stats:forecast-linear-growth-at *stats-table variable T*
stats:forecast-compound-growth-at *stats-table variable T*
stats:forecast-continuous-growth-at *stats-table variable T*

Each of these reporters returns a forecasted value of the given variable T periods after the last observation on that variable. The variable is specified in the second argument either with a variable name or a variable number. T is specified in the third argument. It will often be 1, to indicate the period immediately after the last observation, or greater to get a forecast further in the future, but it may also be negative to get a forecast for a prior period. For example:

```
let fcst stats:forecast-linear-growth-at tbl "nturtles" 6
let fcst stats:forecast-linear-growth-at tbl 0 6
```

Both return a forecast of the first variable in the data table six periods after the last observation, using a linear forecast. The number of observations used may be limited by **stats:use-most-recent**.

The linear forecast uses the model $X_{t+T} = c + s(t + T)$ where t is the period of the last observation on the variable X, c is a constant and s is the slope. The compound forecast uses the model $X_{t+T} = c(1 + r)^{(t+T)}$ where c is a constant and r is the compound rate of growth. The continuous forecast uses the model $X_{t+T} = ce^{r(t+T)}$ where c is a constant and r is the continuous rate of growth. Whereas compound growth assumes discrete time with X growing by a given proportion (1 + r) each finite period of time (e.g., a day, a month or a year), continuous growth assumes that X is compounded continuously (e.g., each second or fraction of a second). In all models, the slope, s, or the growth rate, r, may be negative to indicate a falling trend in X. Because of the way in which the compound and continuous forecasts are calculated, negative values of X are not allowed when using those models.

NOTE that if there is only one observation in the data table, or if **stats:use-most-recent** has specified only one observation, then that single observation is returned as the forecast.

These forecast reporters assume that each observation is taken at a fixed interval. If this is not the case, then the best approach it to make the tick number or some other measure of the time at which the observation is taken a variable in the data table. Then standard regression techniques can be used to calculate the forecast using time as the independent variable. (If a compound or continuous growth is desired, remember to add the natural log of the variable to be forecast to the data gathered in the data table.) The forecast reporters do not calculate the various regression statistics available in **stats:regress-all** and **stats:regress-on**. If they are needed, again the standard regression procedures should be used.

stats:get-fparameters *stats-table*

a reporter that returns a list with the parameters of the most recent forecast. The list has two items, the constant and either the slope or the rate of growth for the model of the forecast. For example:

```
let fcst stats:forecast-linear-growth-at tbl "nturtles" 6
let fparams stats:get-fparameters tbl
```

results in fparams being a list consisting of the constant, c, and the slope, s, in the linear forecast equation, $X_t = c + s(t)$. Had we done a continuous forecast, the first and second items in the list would have been the parameters c and r in $X_t = ce^{r(t)}$. As noted above, if only one observation is used, the forecast simply returns that observation. In that case **stats:get-fparameters** will return a constant equal to the value of the variable and a slope or growth rate (s or r) equal to zero.

1.4 Data display

stats:print-data *stats-table*

stats:print-correlation *stats-table*

stats:print-covariance *stats-table*

These reporters return a string containing a nicely formatted version of the data table, the correlation matrix or the variance-covariance matrix, respectively, suitable for printing. If names for the variables have been specified, they are included. For example:

```
output-print stats:print-correlation tbl
```

sends a string to **output-print** that results in a nicely formatted copy of the correlation matrix being displayed in the NetLogo output area.

1.5 Distributions

stats:normal *x mean stddev*
stats:normal-left *x mean stddev*
stats:normal-inverse *area mean stddev*

These reporters return statistics from a normal distribution with the given mean and standard deviation. **stats:normal** returns the probability density at x. **stats:normal-left** returns the area under the normal distribution from minus infinity to x (that is to the left of x). **stats:normal-inverse** returns the x to the left of which is the given area (that is the area from minus infinity to x is the given area). x can be any real number. area must strictly greater than 0.0 and strictly less than 1.0.

stats:lognormal *x location scale*
stats:lognormal-left *x location scale*
stats:lognormal-inverse *area location scale*

These reporters return statistics from a log-normal distribution with the given location and size parameters. **stats:lognormal** returns the probability density at x. **stats:lognormal-left** returns the area under the log-normal distribution from zero to x (that is to the left of x). **stats:lognormal-inverse** returns the x to the left of which is the given area (that is the area from zero to x is the given area). If x is log-normally distributed with location and scale parameters m and s, then $\frac{\ln(x)-m}{s}$ is normally distributed with mean 0.0 and standard deviation 1.0. x can be any positive real number. area must be strictly less than 1.0. If area is less than or equal to zero, **stats:lognormal-inverse** returns 0.0.

stats:student-left *x df*
stats:student-inverse *area df*

These reporters return statistics from the students distribution with the given degrees of freedom. **stats:student-left** returns the area under the student distribution from minus infinity to x (that is to the left of x). **stats:student-inverse** returns the x to the left of which is the given area (that is the area from minus infinity to x is the given area). x can be any real number. area must strictly greater than 0.0 and strictly less than 1.0.

stats:binomial-coeff n k

This reporter returns the binomial coefficient, $\binom{n}{k}$. This is also the number of distinct subsets of k elements in a population of size n. The reporter uses the Γ distribution to

evaluate $\binom{n}{k}$, and thus works for fairly large values of n. Since NetLogo treats all numbers as real, the value returned may not take on an exact integer value.

stats:binomial-probability *n k p*
stats:binomial-sum-to *n k p*
stats:binomial-sum-above *n k p*

These reporters return statistics from the binomial distribution. n is the number of trials, k the number of successes, and p the probability of success in each trial. **stats:binomial-probability** returns the probability of exactly k successes in n trials. **stats:binomial-sum-through** returns the sum of the probabilities of 0 through k successes in n trials. **stats:binomial-complemented** returns the sum of the probabilities of k+1 through n successes in n trials. The probability, p, is expressed as a fraction in the range 0 through 1.

stats:chi-square-left *x df*
stats:chi-square-right x *df*

These reporters return statistics from the Chi-Square distribution with the given degrees of freedom. **stats:chi-squaret-left** returns the area under the Chi-Square distribution from zero to x (that is to the left of x). **stats:chi-square-right** returns the area under the Chi-Square distribution from x to infinity (that is to the right of x).

stats:gamma *a*
stats:logGamma *a*
stats:incompleteGamma *a x*
stats:incompleteGammaComplement *a x*

The first of these reporters returns the value of the gamma function with parameter *a*, while the second returns the natural log of the gamma function with parameter *a*. The third and fourth of these reporters return the regularized incomplete gamma function and regularized complemented incomplete gamma function, respectively, with parameter *a*. In the third reporter *x* is the integration end point and in the fourth it is the integration start point. (The regularized incomplete gamma functions are the values of the respective integrals divided by the value of gamma.)

stats:beta *a b*
stats:bigBeta *a b*
stats:incompleteBeta *a b x*

The first of these reporters returns the value of the beta function with the parameters *a* and *b*. Because the Colt beta function returns zero when $a + b \geq 172$, **stats:bigBeta** is

provided for cases where the arguments to beta function sum to more than 171. It makes use of the relation that

$$\text{beta}(a, b) = \frac{\text{gamma}(a) * \text{gamma}(b)}{\text{gamma}(a + b)}$$

Thus

$$\text{beta}(a, b) = \exp[(\ln(\text{gamma}(a)) - \ln(\text{gamma}(b))) - \ln(\text{gamma}(a + b)]$$

Although stats:logGamma could be used to compute this in NetLogo, the **stats:bigBeta** primitive is likely to be faster. The third of these reporters returns the regularized integral from zero to x of the beta function with parameters a and b. (The regularized integral is the integral divided by the value of beta.)

2. Export/Import World

NetLogo does not currently support the export of the nested data structure used by the **stats** extension. Therefore, **export-world** and **import-world** will not export and re-import the data tables and other statistics created by the **stats** extension. There is a partial work-around, however. One can save the data table and variable names as NetLogo lists before an export-world. After a subsequent **import-world**, the data and name lists may be used to recreate the original data table. The various statistics, regressions and forecasts, however, will have to be recalculated. For example:

```
let data stats:get-data-as-list tbl
let names stats:get-names tbl
export-world filename
.....
import-world filename
let tbl stats:newtable-from-row-list data
stats:set-names tbl names
```

Hopefully this will be fixed in a future version of the extension.

3. Comments, acknowledgements, etc.

Several of the procedures in this extension build upon procedures in the **matrix** extension bundled with NetLogo. Rather than put them there, it seemed best to break them out to a new extension focused on statistics. The simpler version of the **stats** procedures for regression and forecasts are still in the **matrix** extension. At some point in the future, the **stats** and **matrix** extensions may be designed to work together.

As is always the case, I need to acknowledge the help of Seth Tisue, Forrest Stonedahl and the many contributors to the NetLogo forums for ideas and advice. Obviously, any errors are entirely mine.

It would be interesting to add more statistical procedures to the ones contained in this extension. If anyone has any ideas for what might be useful, please let me know. As noted earlier, the **stats** extension makes use of the Jama matrix package (http://math.nist.gov/javanumerics/jama/). It does not, however, use matrices as arguments to or output from the **stats** procedures. This would not be hard to do, and may happen in a future version, particularly if it seems to be a felt need. Let me know.

Please also let me know if you find any errors.

Charles Staelin (cstaelin@smith.edu)
Department of Economics
Smith College
Northampton, MA 01063
U.S.A.

May 2019