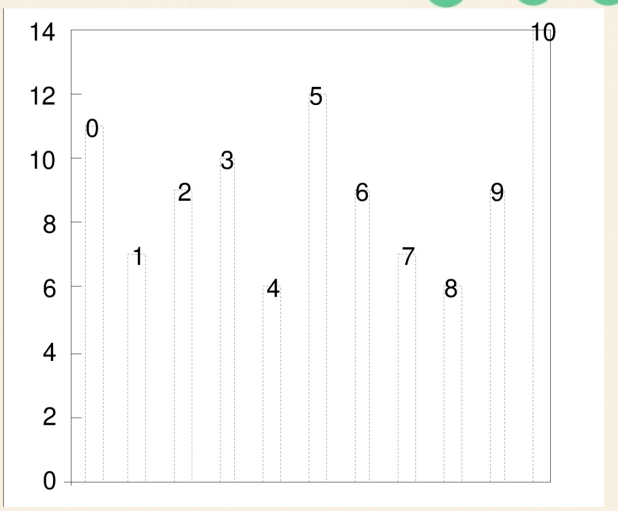
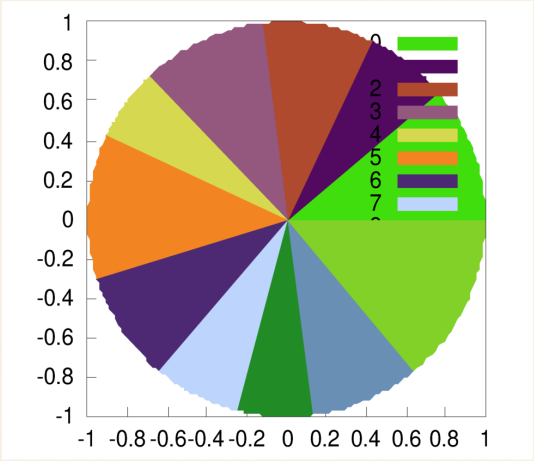
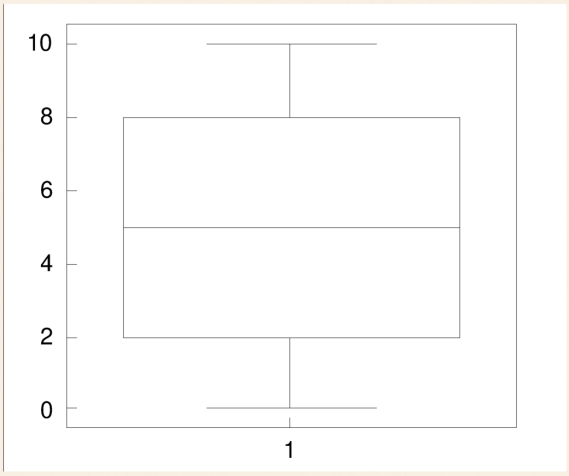
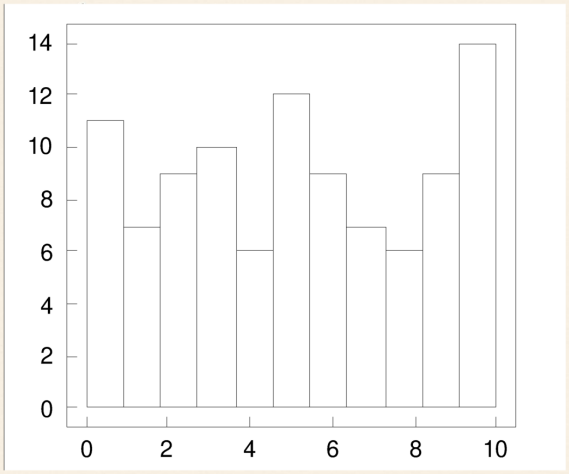
MAXIMA
PAR L'EXEMPLE

VOLUME 2
Guide des statistiques descriptives









Guide to Descriptive Statistics with Maxima

Table of Contents

1 Introduction

This guide is the second volume of the "Maxima by Example" collection. It demonstrates how to use Maxima to handle this field of mathematics. *This volume only covers single-variable series.*
This document, as well as the PDF and wxMaxima versions of this Guide to Descriptive Statistics with Maxima, are available on the website https://maxima-french-doc.fr.
Maxima features a package called `descriptive` which contains most of the procedures required for processing statistical data. It is essential to load this package using the following command:
`load(descriptive)`
before calling any of the functions included in this package.
This guide is based on Maxima version 5.45 and applies to all subsequent versions.

2 Data Series with Maxima

2.1 Randomly Generating Data with Maxima

2.1.1 The random(n) function

Command	*Comment*
`random(6)`	generates a random integer between 0 and 5
`random(6.0)`	generates a random decimal between 0 and strictly less than 6
`makelist(f(i),i,n1,n2)`	generates a list of $f(i)$ for i ranging from n1 to n2.
`donnees[i]`	extracts the i-th term of the list named donnees.

Maxima Command:
```
makelist(random(6)+1,i,1,10)
```
Comment: Generating 10 rolls of a die.

Result: $[3, 1, 3, 6, 5, 2, 6, 6, 1, 2]$

Maxima Command:
```
makelist(random(6.0),i,1,4)
```
Comment: Generating 4 random real numbers between 0 and strictly less than 6.

Result: $[2.493376336328897, 3.675443335522948, 5.050875932015527, 3.033536667862871]$

Maxima Command:
```
makelist(random(20)/(random(10)+1),i,1,15);
```
Comment: Generating a series of rational fractions.

Result: $\left[\frac{17}{7}, \frac{9}{10}, \frac{3}{10}, \frac{6}{7}, 4, 0, \frac{6}{5}, 1, 1, \frac{13}{4}, \frac{7}{9}, \frac{4}{7}, 15, \frac{17}{7}, 1\right]$

2.1.2 Generating Series of Positive or Negative Reals

We will define a new function `random2(n)` which generates both positive and negative numbers, supplementing `random(n)` which only generates positive reals.

Maxima Command:
```
random2(n):=(
    block([signe],
      signe:1,
      if random(2) = 1 then signe:(-1) else signe:1,
      return(signe*random(n)))
);
```

```
makelist(random2(6),i,1,15);
```
Comment: Generating integers between -5 and 5.

Result: $[-2, 5, 0, 0, 5, 2, -3, 4, 0, 0, -3, -4, -1, -2, -2]$

2.2 Resetting the Maxima Random Generator

To obtain non-reproducible series of random numbers, you can reset Maxima's random number generator. This initialization is done with the command: `set_random_state(make_random_state(12345))` Simply change the number in parentheses to initialize Maxima's random generator differently.

3 Importing and Exporting CSV Data

3.1 Exporting Data to CSV

It can be useful to export data generated by Maxima to process them in other software (spreadsheets, statistical analysis software, etc.).

Maxima Command:
```
maliste:makelist(x^2,x,1,9);
file: openw("my_list.csv");
for item in maliste do
    printf(file, "~a,", item);
close(file);
```
Comment: We generate a data list called maliste. We then open the file my_list.csv (which is created if it does not exist) and write the data separated by commas. The file must then be closed to be saved.

Result: The file my_list.csv is created in the current directory and contains: 1,4,9,16,25,36,49,64,81,

Maxima Command:
```
ma_liste: [1, 3, 5];
file: openw("my_list_column.csv");
for item in ma_liste do
    printf(file, "~a~%", item);
close(file);
```
Comment: These commands create a CSV file in the current directory, but the data is arranged in a column (rather than a row).

Result: The file my_list_column.csv is created in the current directory and contains:
1
3
5

You may also want to export a CSV file with a weighted statistical series, with values on the first line and frequencies (counts) on the second. Maxima can generate a list of lists, for example:

Maxima Command:
```
ma_liste: [[1, 2], [3, 4], [5, 6]];
file: openw("my_list.csv");
num_lines: length(ma_liste[1]);
for i: 1 thru num_lines do (
    for j: 1 thru length(ma_liste) do (
        printf(file, "~a", ma_liste[j][i]),
          if j < length(ma_liste) then printf(file, ",")
```

```
    ),
    printf(file, "~%")
);
close(file);
```
Comment: These commands create a CSV file in the current directory, but the data is arranged in two columns.

Result: The file my_list_column.csv is created in the current directory and contains:
1,3,5
2,4,6

3.2 Importing a CSV File

For a CSV file named my_list.csv containing:
1,2,3
4,5,6

Maxima Command:
```
maliste4:read_list("my_list.csv");
```
Comment: Reading data from the CSV file and generating a Maxima list.

Result: maliste4 gives: $[1\,,3\,,5\,,2\,,4\,,6]$

Maxima Command:
```
A:read_matrix("my_list.csv");
```
Comment: Reading data from the CSV file and generating a Maxima matrix.

Result: $A = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$

4 Study of a Single-Variable Statistical Series

The `descriptive` package contains many functions for studying single-variable statistical series. It must be loaded before the study begins using the command: `load(descriptive)`

4.1 Generating the Sample Series

We use as an example the series obtained from the grades of 100 students on an exam, scored out of 10 (integer grades). The generated series is called serieexemple (displayed across 10 lines).

serieexemple: $\begin{pmatrix} 2 & 10 & 3 & 5 & 10 & 7 & 1 & 10 & 10 & 10 \\ 9 & 5 & 1 & 9 & 2 & 1 & 6 & 0 & 10 & 1 \\ 0 & 6 & 6 & 3 & 9 & 4 & 7 & 5 & 5 & 0 \\ 0 & 4 & 10 & 9 & 1 & 7 & 5 & 3 & 0 & 9 \\ 3 & 5 & 10 & 7 & 0 & 9 & 10 & 7 & 0 & 6 \\ 6 & 0 & 8 & 2 & 5 & 4 & 8 & 1 & 5 & 3 \\ 8 & 8 & 2 & 3 & 6 & 6 & 2 & 10 & 2 & 3 \\ 1 & 10 & 4 & 8 & 3 & 5 & 3 & 2 & 7 & 6 \\ 0 & 10 & 4 & 9 & 7 & 5 & 10 & 2 & 9 & 5 \\ 5 & 6 & 10 & 2 & 4 & 0 & 8 & 3 & 9 & 0 \end{pmatrix}$

Note: It is possible to customize this document by manually modifying the data series in the TeX source file (or directly in the wxMaxima file). The modification is made at line 289 of this document. Just keep the name for the data series so that all calculations are performed automatically. If the number of data points exceeds 100, only the first 100 values are displayed above. The routine also only displays the first 20 terms of a series if the number of values is between 20 and 99, to avoid display issues during document compilation.

4.2 Frequencies and Relative Frequencies

4.2.1 Total Frequency (Sample Size)

Maxima Command:

```
total_frequency:length(serieexemple)
```

Comment: Calculating the total frequency (sample size) of a statistical series.

Result: total_frequency: 100

4.2.2 Absolute Frequencies of a 1-Variable Statistical Series

Maxima Command:

```
/* Function to calculate frequencies (counts) */
calculate_counts(data) := block(
    [counts, val, i],
    counts: [],
    /* Iterate through each unique value in the data */
    for val in setify(data) do (
        /* Count occurrences of the value */
        count: count_equal(data, val),
        /* Add the value and its count to the list */
        counts: append(counts, [val, count])
    ),
    return(counts)
)$

/* Function to count occurrences of a value in a list */
count_equal(lst, val) := block(
    [count, item],
    count: 0,
    for item in lst do (
        if item = val then count: count + 1
    ),
    return(count)
)$
counts: calculate_counts(serieexemple);
counts;
```

Comment: Calculating the absolute frequencies of a statistical series.

Result: Absolute frequencies: $[0, 11, 1, 7, 2, 9, 3, 10, 4, 6, 5, 12, 6, 9, 7, 7, 8, 6, 9, 9, 10, 14]$
For each value in the series, the command returns its value followed by its count.

Maxima Command:

```
/* Extract odd-indexed terms (values) */
odd_terms: makelist(counts[2*i-1], i, 1, length(counts)/2);
/* Extract even-indexed terms (counts) */
even_terms: makelist(counts[2*i], i, 1, length(counts)/2);
/* Build a table for results */
frequency_table: matrix(odd_terms, even_terms)$

frequency_table2: matrix(
  append(["values"], frequency_table[1]),
```

```
  append(["counts"], frequency_table[2])
);
/* Display results */
frequency_table2$
```
Comment: Presenting results in a table format.

Result: $\begin{pmatrix} \text{values} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ \text{counts} & 11 & 7 & 9 & 10 & 6 & 12 & 9 & 7 & 6 & 9 & 14 \end{pmatrix}$

Alternative solution: using the discrete_freq command from the descriptive package

Maxima Command:
```
load(descriptive)$
discrete_freq(serieexemple);
```
Comment: Command to obtain two lists: the first for values, the second for counts.

Result: $[[0,1,2,3,4,5,6,7,8,9,10],[11,7,9,10,6,12,9,7,6,9,14]]$

Maxima Command:
```
frequency_table_pkg:matrix(discrete_freq(serieexemple)[1],
discrete_freq(serieexemple)[2]);
frequency_table2_pkg: matrix(
  append(["values"], frequency_table_pkg[1]),
  append(["counts"], frequency_table_pkg[2])
);
```
Comment: Presentation in matrix form for better readability.

Result: $\begin{pmatrix} \text{values} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ \text{counts} & 11 & 7 & 9 & 10 & 6 & 12 & 9 & 7 & 6 & 9 & 14 \end{pmatrix}$

4.2.3 Relative Frequencies of a 1-Variable Statistical Series

Maxima Command:
```
even_terms2:even_terms/length(serieexemple)$
frequency_table3: matrix(odd_terms, even_terms2)$
frequency_table4: matrix(
  append(["values"], frequency_table3[1]),
  append(["frequencies"], frequency_table3[2])
)$
frequency_table4;
```
Comment: We divide the counts by the total frequency and generate a new matrix for the results.

Result: $\begin{pmatrix} \text{values} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ \text{frequencies} & \frac{11}{100} & \frac{7}{100} & \frac{9}{100} & \frac{1}{10} & \frac{3}{50} & \frac{3}{25} & \frac{9}{100} & \frac{7}{100} & \frac{3}{50} & \frac{9}{100} & \frac{7}{50} \end{pmatrix}$

Maxima Command:
```
ev(frequency_table4,numer);
```
Comment: Previous table with approximate frequency values.

Result:
$$\begin{pmatrix} \text{values} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ \text{frequencies} & 0.11 & 0.07 & 0.09 & 0.1 & 0.06 & 0.12 & 0.09 & 0.07 & 0.06 & 0.09 & 0.14 \end{pmatrix}$$

4.2.4 Cumulative Frequencies (Ascending)

Maxima Command:

```
even_terms3: []$
current_sum: 0$
for i: 1 thru length(even_terms) do (
    current_sum: current_sum + even_terms[i],
    even_terms3: append(even_terms3, [current_sum])
)$
frequency_table5: matrix(odd_terms, even_terms3)$
frequency_table6: matrix(
  append(["values"], frequency_table5[1]),
  append(["Cumul. Count (Asc)"], frequency_table5[2])
)$
frequency_table6;
```

Comment: The commands add each previous count into the list to obtain the ascending cumulative frequencies.

Result:
$$\begin{pmatrix} \text{values} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ \text{Cumul. Count (Asc)} & 11 & 18 & 27 & 37 & 43 & 55 & 64 & 71 & 77 & 86 & 100 \end{pmatrix}$$

4.2.5 Cumulative Frequencies (Descending)

Maxima Command:

```
even_termsr:reverse(even_terms)$
odd_termsr:reverse(odd_terms)$
even_terms4: []$
current_sum: 0$
for i: 1 thru length(even_termsr) do (
    current_sum: current_sum + even_termsr[i],
    even_terms4: append(even_terms4, [current_sum])
)$
frequency_table7: matrix(odd_termsr, even_terms4)$
frequency_table8: matrix(
  append(["values"], frequency_table7[1]),
  append(["Cumul. Count (Desc)"], frequency_table7[2])
)$
frequency_table8;
```

Comment: We follow the same principle as before, but we first reverse the lists of values and counts to obtain them in descending order.

Result:
$$\begin{pmatrix} \text{values} & 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ \text{Cumul. Count (Desc)} & 14 & 23 & 29 & 36 & 45 & 57 & 63 & 73 & 82 & 89 & 100 \end{pmatrix}$$

4.2.6 Relative Cumulative Frequencies (Ascending)

Maxima Command:

```
frequency_table9: matrix(odd_terms, even_terms3/
length(serieexemple))$
frequency_table10: matrix(
```

```
  append(["values"], frequency_table9[1]),
  append(["Rel. Cumul. (Asc)"], frequency_table9[2])
)$
frequency_table10;
```
Comment: We divide the ascending cumulative counts by the total series frequency.

Result: $\begin{pmatrix} \text{values} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ \text{Rel. Cumul. (Asc)} & \frac{11}{100} & \frac{9}{50} & \frac{27}{100} & \frac{37}{100} & \frac{43}{100} & \frac{11}{20} & \frac{16}{25} & \frac{71}{100} & \frac{77}{100} & \frac{43}{50} & 1 \end{pmatrix}$

Maxima Command:
```
ev(frequency_table10,numer);
```
Comment: Numerical values of the relative ascending cumulative frequencies.

Result: $\begin{pmatrix} \text{values} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ \text{Rel. Cumul. (Asc)} & 0.11 & 0.18 & 0.27 & 0.37 & 0.43 & 0.55 & 0.64 & 0.71 & 0.77 & 0.86 & 1 \end{pmatrix}$

4.2.7 Relative Cumulative Frequencies (Descending)

Maxima Command:
```
frequency_table11: matrix(odd_termsr, even_terms4/
length(serieexemple))$
frequency_table12: matrix(
  append(["values"], frequency_table11[1]),
  append(["Rel. Cumul. (Desc)"], frequency_table12[2])
)$
frequency_table12;
```
Comment: We divide the descending cumulative counts by the total series frequency.

Result: $\begin{pmatrix} \text{values} & 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ \text{Rel. Cumul. (Desc)} & \frac{7}{50} & \frac{23}{100} & \frac{29}{100} & \frac{9}{25} & \frac{9}{20} & \frac{57}{100} & \frac{63}{100} & \frac{73}{100} & \frac{41}{50} & \frac{89}{100} & 1 \end{pmatrix}$

Maxima Command:
```
ev(frequency_table12,numer);
```
Comment: Numerical values of the relative descending cumulative frequencies.

Result: $\begin{pmatrix} \text{values} & 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ \text{Rel. Cumul. (Desc)} & 0.14 & 0.23 & 0.29 & 0.36 & 0.45 & 0.57 & 0.63 & 0.73 & 0.82 & 0.89 & 1 \end{pmatrix}$

5 Graphical Representations of a 1-Variable Statistical Series

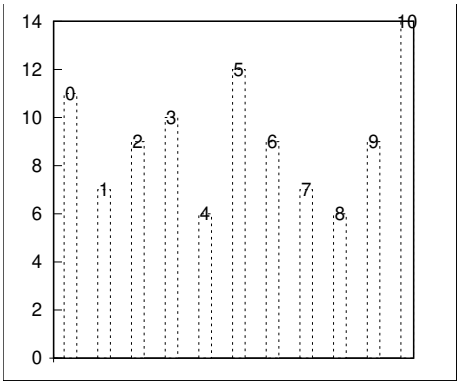
5.1 Bar Plot

Maxima Command:
```
wxbarsplot(serieexemple);
```
Comment: This command comes from the descriptive package and automatically draws a bar chart for the given series.

For reference, the absolute frequency table of the series is:

values	0	1	2	3	4	5	6	7	8	9	10
counts	11	7	9	10	6	12	9	7	6	9	14

Result:



5.2 Pie Chart Representation

Maxima Command:

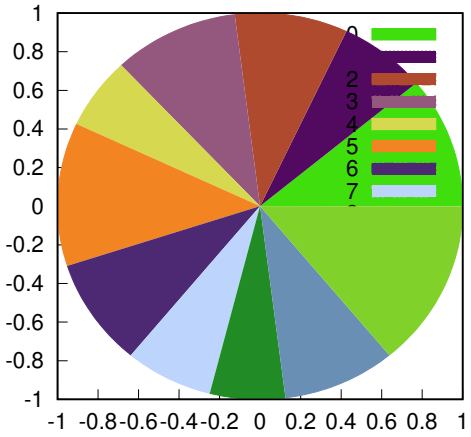
```
wxpiechart(serieexemple);
```

Comment: This command comes from the descriptive package and automatically draws a pie chart for the given series.

For reference, the absolute frequency table of the series is:

values	0	1	2	3	4	5	6	7	8	9	10
counts	11	7	9	10	6	12	9	7	6	9	14

Result:



5.3 Histogram Representation

Maxima Command:

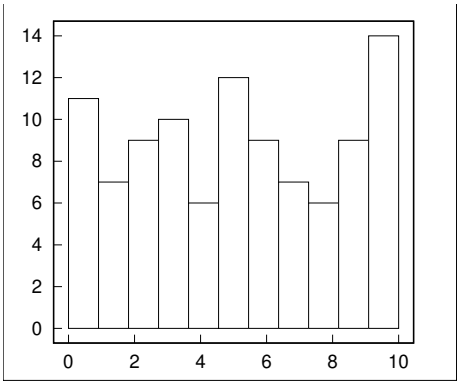
```
wxhistogram(serieexemple,nclasses=length(odd_terms));
```

Comment: This command comes from the descriptive package and automatically draws a histogram for the given series. We specify that the number of rectangles equals the number of series values using the nclasses parameter (otherwise, this parameter defaults to 10).

For reference, the absolute frequency table of the series is:

values	0	1	2	3	4	5	6	7	8	9	10
counts	11	7	9	10	6	12	9	7	6	9	14

Result:



6 Parameters of a 1-Variable Statistical Series

6.1 Mean

Maxima Command:

```
fpprintprec:5$
mean(serieexemple);
float(mean(serieexemple));
```

Comment: We set the display of decimals to 5 and use the mean command from the descriptive package.

Result: Mean $\frac{511}{100} \simeq 5.11$

6.2 Range

Maxima Command:

```
smin(serieexemple);
smax(serieexemple);
range(serieexemple);
```

Comment: The smin, smax, and range functions from the descriptive package provide the answers.

Result: Minimum value: 0 Maximum value: 10 Range: 10

6.3 Median

Maxima Command:

```
median(serieexemple)
```

Comment: Command from the descriptive package.

Result: Median $5 \simeq 5.0$

6.4 Quartiles

Maxima Command:

```
quantile(serieexemple,0.25);
quantile(serieexemple,0.75);
qrange(serieexemple):
```

Comment: These commands from the descriptive package give the first and third quartiles, and the interquartile range respectively.

Result: First quartile: 2.0
Third quartile: 8.0
Interquartile range: 6

6.5 Variance and Standard Deviation

6.5.1 For a Sample

The sample variance is calculated by dividing by $n-1$, where n is the total series frequency.

Maxima Command:

```
variance(data):=block(
[aux],
 moyenne:mean(data),
 aux:sum((data[i] - moyenne)^2, i, 1, length(data)) /
 (length(data) - 1),
    return(aux)
)$
variance(serieexemple);
sqrt(variance(serieexemple));
```

Comment: We define a Maxima function calculating this variance. The square root of the result gives the standard deviation.

Result: Variance $\frac{36593}{3300} \simeq 11.089$
Standard Deviation $\frac{\sqrt{36593}}{10\sqrt{33}} \simeq 3.33$

Maxima Command:

```
var1(serieexemple);
```

Comment: You can also directly use the var1 function from the descriptive package.

Result: Variance: $\frac{36593}{3300}$

6.5.2 For the Population

The population variance is calculated by dividing by n, the total series frequency.

Maxima Command:

```
variancep(data):=block(
[aux],
 moyenne:mean(data),
 aux:sum((data[i] - moyenne)^2, i, 1, length(data)) /
 (length(data)),
    return(aux)
)$
variancep(serieexemple);
sqrt(variancep(serieexemple));
```

Comment: We define a Maxima function calculating this variance. The square root of the result gives the standard deviation.

Result: Variance $\frac{109779}{10000} \simeq 10.978$
Standard Deviation $\frac{\sqrt{109779}}{100} \simeq 3.3133$

Maxima Command:

```
var(serieexemple);
```

Comment: You can also directly use the var function from the descriptive package.

Result: Variance: $\frac{109779}{10000}$

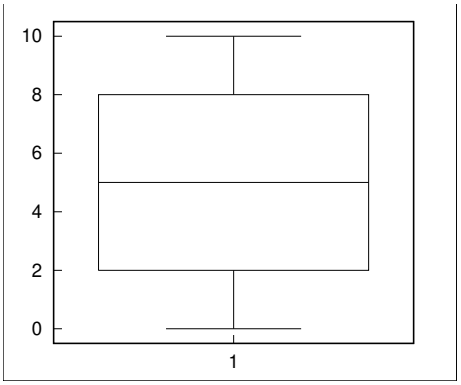
7 Box Plot

Maxima Command:

```
wxboxplot(serieexemple);
```

Comment: Command from the descriptive package.

Result:



8 Addenda

8.1 Study of a Weighted Series

When the series to be studied is given as a weighted series, it can be transformed into a full list of all data, which then allows for the application of all commands covered in this manual. Let's take the following series as an example:

Value	1	2	3	4	5	6	7
Frequency	11	20	9	7	29	32	15

Maxima Command:

```
test_series:[1,11,2,20,3,9,4,7,5,29,6,32,7,15];
```

Comment: We define a list of the type [value1, count1, value2, count2, ...]

Result: serietest: $[1, 11, 2, 20, 3, 9, 4, 7, 5, 29, 6, 32, 7, 15]$

Maxima Command:

```
new_series: []$
/* Generating the new series */
for i: 1 thru length(test_series) step 2 do (
    value: test_series[i],
    count: test_series[i + 1],
    for j: 1 thru count do (
        new_series: endcons(value, new_series)
    )
```

```
)$
```
Comment: This program generates new_series which lists all values of the series individually.

Result: $[1,1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3,3,4,4,4]$
$[4,4,4,4,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,6,6,6,6,6,6,6,6,6,6,6,6,6]$
$[6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7]$
This series can now be studied according to the model used for serieexemple.

8.2 Frequency by Classes

It is possible to calculate frequencies by class (using the initial serieexemple):

Maxima Command:

```
continuous_freq(serieexemple,5);
```

Comment: This command from the descriptive package calculates counts distributed over 5 classes. The second argument equals the number of classes (default is 10).

Result: $[[0,2,4,6,8,10]\,,[27,16,21,13,23]]$
this represents a frequency of 27 for the class [0,2], a frequency of 16 for the class [3,4], a frequency of 21 for the class [5,6], a frequency of 13 for the class [7,8] and a frequency of 23 for the class [9,10].

For any remarks or comments on this document, you can send an email to michel.gosse@free.fr.