o++o and EXCEL in Comparison

Klaus Benecke, Eicke Redweik, Stephan Schenkl

(Stand: 05.09.2021)

In the present work o++o and EXCEL are compared on the basis of a typical problem with a structured table regarding the programming effort. Furthermore it is considered, which expenditure results for a slight change of the schema of the structured table. It turns out that in the o++o program only the initial table has to be slightly extended, whereas the EXCEL application requires strong changes. One can even say that these are different EXCEL applications. From the user's point of view, the solution of this problem or class of problems is greatly simplified by o++o, since many EXCEL applications can be implemented with o++o in a similar but simpler way. If one compares the typical EXCEL user with an o++o user, one will find that both the formulation and the subsequent adaptation of an o++o solution take significantly less time than the EXCEL solution. The results of this work apply to non-structured tables as well as structured tables, as long as they do not contain an empty set and can thus be easily flattened without loss of information. Finally, both approaches are compared according to general aspects.

Table of Contents

1. Initial Problem

For the following table, the values of BMI (Body Mass Index) are to be determined according to the formula. Both the value for each person included and for each age (ALTER) listed for this person are to be calculated, as well as grouped by age the BMI values averaged over the persons of the age group, and the average value of all BMI values. Only persons for whom age values greater than 20 are available are to be included. We will see in a moment that the wording of the query in o++o is easier to grasp than the preceding description in English.

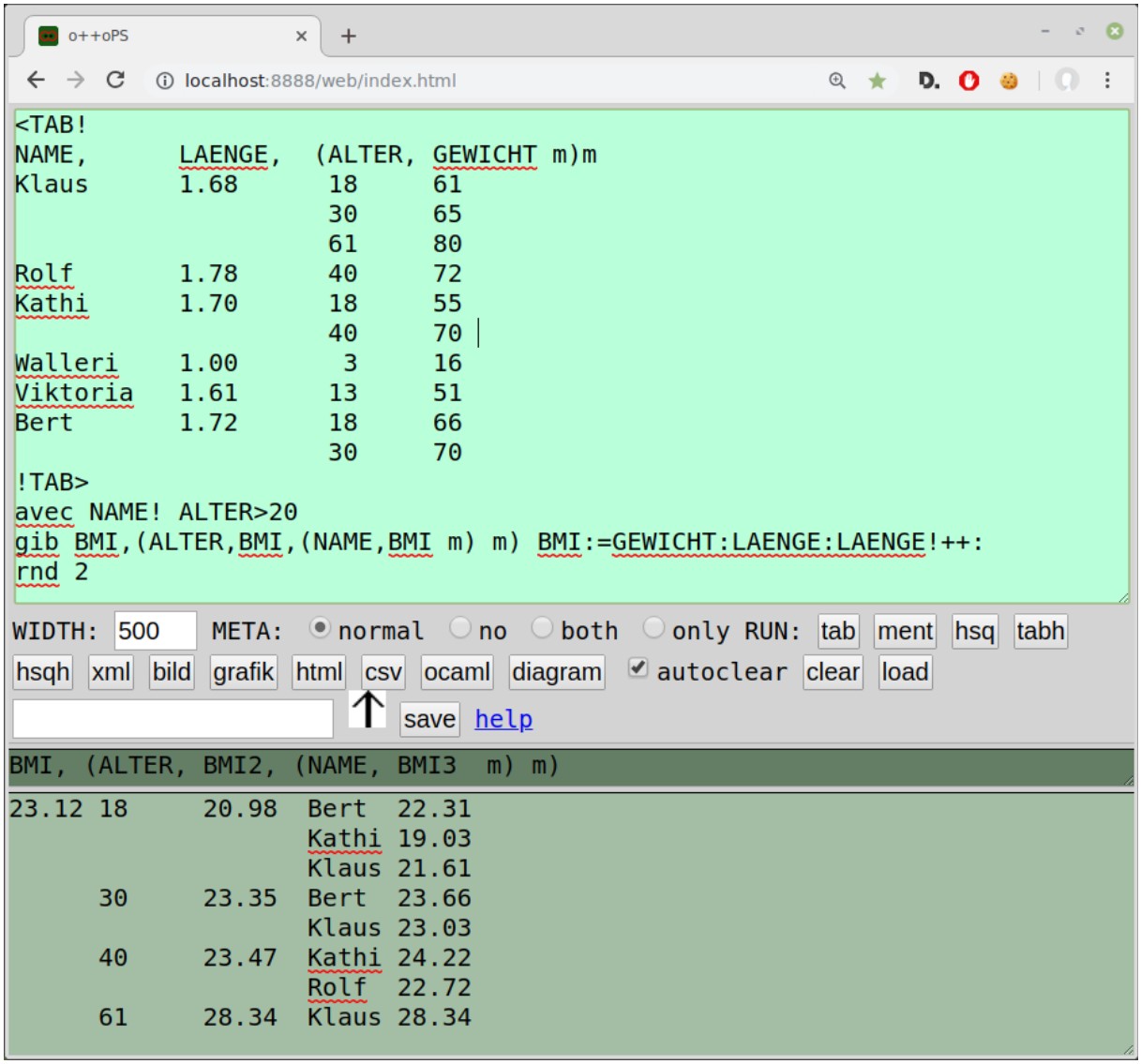
NAME	LAENGE	ALTER	GEWICHT
Klaus	1.68	18	61
		30	65
		56	80
Rolf	1.78	40	72
Kathi	1.70	18	55
		40	70
Walleri	1.00	3	16
Viktoria	1.61	13	51
Bert	1.72	18	66
		30	70

1.1. Solution with o++o

The given table can be saved as a file or entered into the program part (upper light green field) as below in TAB representation. The o++o program consists of the desired selection (avec-line) and a restructuring with aggregation (gib-line). By

```
rnd 2
```

all numbers are then rounded to 2 digits after the point . The result in TAB format can be found in the lower slightly darker green field and the table header in the middle field.



o++o: *Input data and processing code (top, green background), output tab schema (middle) and output tab content (bottom, darkgreen background).*

If, as already mentioned, you simply save the source data as a text file in the "tab" format under the name `persons.tab`,

```
NAME,       LAENGE,  (ALTER, GEWICHT m)m
Klaus       1.68      18     61
                      30     65
                      61     80
Rolf        1.78      40     72
Kathi       1.70      18     55
                      40     70
Walleri     1.00       3     16
Viktoria    1.61      13     51
Bert        1.72      18     66
```

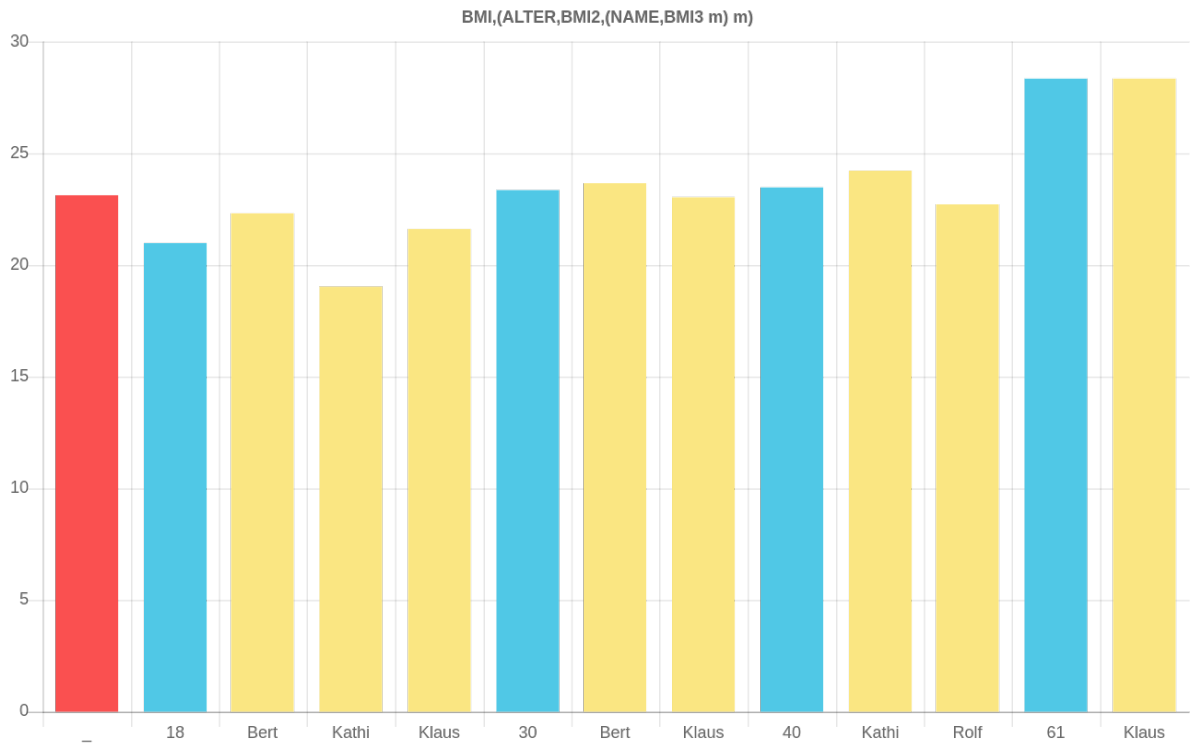
```
                    30     70
```

the above program takes the following form.

```
persons.tab
avec NAME! ALTER>20
gib BMI,(ALTER,BMI,(NAME,BMI m) m) BMI:=GEWICHT:LAENGE:LAENGE!++:
rnd 2
```

Detailed descriptions of o++o can be found at www.ottops.de.

By clicking on the diagram button a selection list for a desired structured diagram appears. After a further click the selected diagram is displayed. Here the values of the parent columns are highlighted in blue. The bar of the zero level is red.



o++o diagram: *Bar chart for the structured output table*

In o++o diagrams numerical values are interpreted as column lengths and non-numerical values as signatures. Since ALTER in the above program is numeric, for the generation of the above diagram still the program line

```
ALTER::=ALTER wort
```

has been attached.

1.2. Solution with EXCEL (manual)

When using EXCEL, filtering must first be performed manually according to the criterion "persons with age > 20", since the naive application filter condition ALTER>20 would eliminate relevant data rows after "flattening" the table.

The formula for the BMI is entered in column E2 of this table and copied to E9.

E2 =D2/B2^2

	A	B	C	D	E	F
1	NAME	LAENGE	ALTER	GEWICHT	BMI3	
2	Klaus	1,68	18	61	21,6128118	
3		1,68	30	65	23,0300454	
4		1,68	56	80	28,3446712	
5	Rolf	1,78	40	72	22,7244035	
6	Kathi	1,7	18	55	19,0311419	
7		1,7	40	70	24,2214533	
8	Bert	1,72	18	66	22,3093564	
9		1,72	30	70	23,6614386	
10						

BMI calculation in EXCEL: *Relevant data rows were manually filtered, the LENGTH duplicated, and the BMI per data row determined by formula and copied.*

Subsequently, sorting and grouping by age are performed. In the process, the names are also copied into the empty cells to create a flat table structure. The calculation of the BMI average per age group takes place in column F and the calculation of the BMI average for all data in column G.

G2 =MITTELWERT(E2:E21)

	A	B	C	D	E	F	G	H
1	NAME	LAENGE	ALTER	GEWICHT	BMI3	BMI2	BMI	
2	Klaus	1,68	18	61	21,6128118	20,9844367	23,1169153	
3	Kathi	1,7	18	55	19,0311419			
4	Bert	1,72	18	66	22,3093564			
5								
6								
7	Klaus	1,68	30	65	23,0300454	23,345742		
8	Bert	1,72	30	70	23,6614386			
9								
10								
11								
12								
13	Rolf	1,78	40	72	22,7244035	23,4729284		
14	Kathi	1,7	40	70	24,2214533			
15								
16								
17								
18								
19	Klaus	1,68	56	80	28,3446712	28,3446712		
20								

BMI calculation in EXCEL on grouped data: *The data rows of the flat tables were manually grouped by ALTER and average values for BMI were determined.*

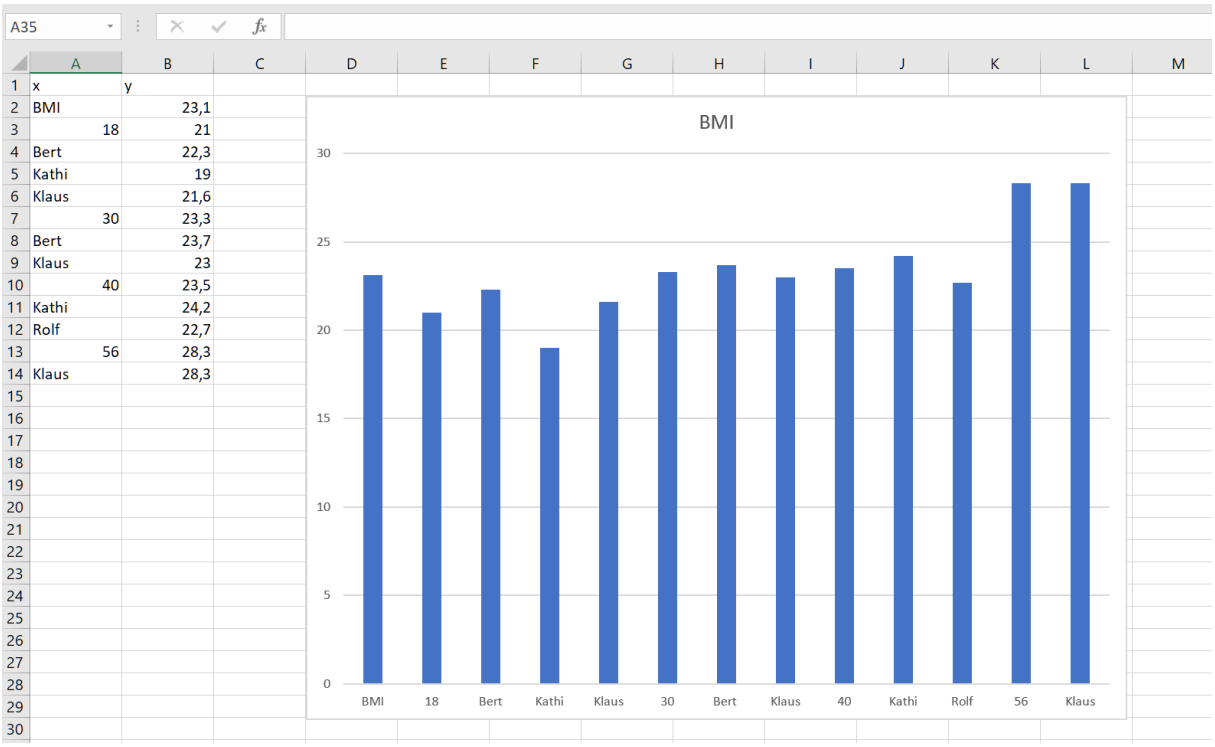
If rows or columns are added or deleted, the formula areas of the calculating cells must be corrected during manual editing. According to the desired arrangement, columns must be moved and columns not required in the results table must be hidden.

C2 =MITTELWERT(E2:E4)

	A	B	C	D	E	H
1	BMI	ALTER	BMI2	NAME	BMI3	
2	23,1169153	18	20,9844367	Klaus	21,6128118	
3		18		Kathi	19,0311419	
4		18		Bert	22,3093564	
5		30	23,345742	Klaus	23,0300454	
6		30		Bert	23,6614386	
7		40	23,4729284	Rolf	22,7244035	
8		40		Kathi	24,2214533	
9		56	28,3446712	Klaus	28,3446712	
10						

Manual column selection in EXCEL: *Columns manually arranged and hidden.*

This result table corresponds to the calculation result of o++o except for the rounding and the redundancy in the ALTER column. The graphical output requires further manual preparation of the data to be displayed, which also grows with increasing data volume.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	x	y											
2	BMI	23,1											
3	18	21											
4	Bert	22,3											
5	Kathi	19											
6	Klaus	21,6											
7	30	23,3											
8	Bert	23,7											
9	Klaus	23											
10	40	23,5											
11	Kathi	24,2											
12	Rolf	22,7											
13	56	28,3											
14	Klaus	28,3											

Graphical representation in EXCEL: *Data is arranged in a flat table to create a bar chart*

1.3. Solution with EXCEL (automated)

Preliminary remark: *EXCEL tends to interpret empty cells as null values or 0 values. To prevent the appearance of unwanted 0-values a query of the form IF(<cell>="";"";<formula>) is often used, which is not repeated explicitly each time.*

Furthermore, steps that work on a variable table size, for example row selection or column selection, can throw errors in cells outside the output table if null values are passed to them. In this case, these

cells are converted to empty cells by `IFERROR(<formula>;"")`. Again, this is not explicitly repeated at each point.

The processing pipeline is implemented by outputting each step on a new worksheet. The schema is not touched in many processing steps and is instead copied to the following worksheet unchanged. The cells of the schema are called schema cells, the other relevant cells are called data cells.

EXCEL formulas that process an entire worksheet can take a long time and a lot of memory. For this example, formulas are used that take up a maximum of the range A1:J50 on each worksheet (including the schema).

When using EXCEL, the data can either be entered manually or stored as a file in "csv" format. By using free cells, it is also possible to use a structured table as shown in the example for EXCEL. In this case, the first preprocessing step is to "flatten" the input table, because further processing in EXCEL requires a non-structured table.

This step is realized by the formula `IF(input.A2="";A1;input.A2)`. Since the first row is fully occupied, this application fills each empty cell with the contents of the cell above it.

	A	B	C	D
1	NAME	LAENGE	ALTER	GEWICHT
2	Klaus	1,68	18	61
3			30	65
4			56	80
5	Rolf	1,78	40	72
6	Kathi	1,7	18	55
7			40	70
8	Walleri	1	3	16
9	Viktoria	1,61	13	51
10	Bert	1,72	18	66
11			30	70

Worksheet Input: Structured table *within a spreadsheet*

	A	B	C	D
1	NAME	LAENGE	ALTER	GEWICHT
2	Klaus	1,68	18	61
3	Klaus	1,68	30	65
4	Klaus	1,68	56	80
5	Rolf	1,78	40	72
6	Kathi	1,7	18	55
7	Kathi	1,7	40	70
8	Walleri	1	3	16
9	Viktoria	1,61	13	51
10	Bert	1,72	18	66
11	Bert	1,72	30	70

Worksheet Input-Flat: *Flat table within a spreadsheet*

Now an automatic filtering can be done according to the criterion "persons with age > 20". The selection can best be done in three steps:

- The lines to be selected are determined with the formula `$'input-flat'.C2>=20`, where column C contains the ALTER values (worksheet 'avec-bool').
- You transfer the rows to be selected into a completely new worksheet with the formula `IF(COUNTIF('avec-bool'.$B$2:$B$50;$'input-flat'.$A2);$'input-flat'.A2;"")` (worksheet 'avec-holes'), leaving blank lines.
- Blank lines are eliminated in two steps:
  - Blank lines are first created with `NOT($'avec-holes'.A2="")` recognized and in this worksheet will be used with `COUNTIF($A$1:A2;1)` an index for (non-empty) data rows is created (worksheet 'hole-help').
  - With the formula `LOOKUP('hole-help'.$C2;$'hole-help'.$B1:$'hole-help'.$B50;'avec-holes'.A2:A50)` only the non-empty data rows are read in (worksheet 'avec').

A table is obtained which is equivalent to the table generated with o++o after this processing step.

	A	B	C	D
1				
2	FALSCH			
3	WAHR	Klaus		
4	WAHR	Klaus		
5	WAHR	Rolf		
6	FALSCH			
7	WAHR	Kathi		
8	FALSCH			
9	FALSCH			
10	FALSCH			
11	WAHR	Bert		

Worksheet Avec-Bool: *Truth values for the selection of the relevant keys*

	A	B	C	D
1	NAME	LAENGE	ALTER	GEWICHT
2	Klaus	1,68	18	61
3	Klaus	1,68	30	65
4	Klaus	1,68	56	80
5	Rolf	1,78	40	72
6	Kathi	1,7	18	55
7	Kathi	1,7	40	70
8				
9				
10	Bert	1,72	18	66
11	Bert	1,72	30	70

Worksheet Avec-Holes: *Flat table in spreadsheet after selection with blank rows*

	A	B	C	D
1		0		
2	WAHR	1	0	
3	WAHR	2	1	
4	WAHR	3	2	
5	WAHR	4	3	
6	WAHR	5	4	
7	WAHR	6	5	
8	FALSCH	6	6	
9	FALSCH	6	7	
10	WAHR	7	8	
11	WAHR	8	9	

Worksheet Hole-Help: *Truth values for selecting the blank lines*

	A	B	C	D
1	NAME	LAENGE	ALTER	GEWICHT
2	Klaus	1,68	18	61
3	Klaus	1,68	30	65
4	Klaus	1,68	56	80
5	Rolf	1,78	40	72
6	Kathi	1,7	18	55
7	Kathi	1,7	40	70
8	Bert	1,72	18	66
9	Bert	1,72	30	70
10				
11				

Worksheet Avec: *Flat table after selection without blank lines*

In the further it is now necessary to sort the data first according to age. For the sake of simplicity, the columns are first sorted according to the scheme ALTER, NAME, GEWICHT, LAENGE with LOOKUP (worksheet 'gib-help').

Sorting is only done for individual columns, so first sort by ALTER:

`SMALL(OFFSET('gib-help'.$A$2:$A$50;0;0;49-COUNTIF('gib-help'.$A$2:$A$50;""));ROW(F2)-1)`. Note in particular the determination of the number of rows depending on the maximum supported data rows (49 = 50 - 1, since one schema row is reserved).

For this sorted column, the still missing data cells are now determined row by row using `INDEX($'gib-help'.B:B;'gib-help-sort-help'.$F2+1;1)` added (worksheet 'gib-help-sort').

	A	B	C	D
1	ALTER	NAME	GEWICHT	LAENGE
2	18	Klaus	61	1,68
3	30	Klaus	65	1,68
4	56	Klaus	80	1,68
5	40	Rolf	72	1,78
6	18	Kathi	55	1,7
7	40	Kathi	70	1,7
8	18	Bert	66	1,72
9	30	Bert	70	1,72
10				
11				

Worksheet Gib-Help: *Table columns automatically re-sorted*

	A	B	C	D
1	ALTER	NAME	GEWICHT	LAENGE
2	18	Klaus	61	1,68
3	18	Kathi	55	1,7
4	18	Bert	66	1,72
5	30	Klaus	65	1,68
6	30	Bert	70	1,72
7	40	Rolf	72	1,78
8	40	Kathi	70	1,7
9	56	Klaus	80	1,68
10				
11				

Worksheet Gib-Help-Sort: *Sorted by first table column (age)*

This is the last worksheet that contains information about WEIGHT and LENGTH, so at the latest now the BMI must be calculated using `$'gib-help-sort'.C2/$'gib-help-sort'.D2/$'gib-help-sort'.D2` can be calculated (worksheet 'BMI').

	A	B	C	D	E
1	ALTER	NAME	GEWICHT	LAENGE	BMIh
2	18	Klaus	61	1,68	21,61281179
3	18	Kathi	55	1,7	19,03114187
4	18	Bert	66	1,72	22,30935641
5	30	Klaus	65	1,68	23,03004535
6	30	Bert	70	1,72	23,66143862
7	40	Rolf	72	1,78	22,72440348
8	40	Kathi	70	1,7	24,22145329
9	56	Klaus	80	1,68	28,3446712
10					
11					

Worksheet BMI: *A new column forms the basis of the BMI columns in the structured output table.*

The processing of `gib` is completed by first transferring the columns ALTER and NAME (formula as for column re-sorting): `HLOOKUP(B$1;'gib-help-sort'.$A$1:$J$50;ROW($K2))`. Three new columns are calculated from the secured BMI values:

- BMI `=AVERAGE($BMI.E2:E50)`
- BMI2 `=IF(B2=B1;"";AVERAGEIF(BMI.$A$2:$A$50;B2;BMI.$E$2:$E$50))`
- BMI3 `=$BMI.E2`

Afterwards the sorting and grouping by age takes place. For this, the names must also be copied into the empty cells. The calculation of the BMI average values per age group takes place in column F and the calculation of the BMI average value for all data in column G.

If rows or columns are added or deleted, the formula areas of the cells to be calculated must be corrected. According to the desired arrangement, columns must be moved and columns not required in the results table must be hidden (worksheet 'gib').

	A	B	C	D	E
1	BMI	ALTER	BMI	NAME	BMI
2	23,11691525	18	20,98443669	Klaus	21,61281179
3				Kathi	19,03114187
4				Bert	22,30935641
5		30	23,34574198	Klaus	23,03004535
6				Bert	23,66143862
7		40	23,47292839	Rolf	22,72440348
8				Kathi	24,22145329
9		56	28,3446712	Klaus	28,3446712
10					
11					

Worksheet Gib: The structured output-table with all grouped values fot the *BMI*

This result table corresponds to the calculation result of o++o except for the rounding. The rounding is done by means of `IFERROR(ROUND($gib.A1;2);$gib.A1)` (note: IFERROR is used here to ignore text cells).

	A	B	C	D	E
1	BMI	ALTER	BMI	NAME	BMI
2	23,12	18	20,98	Klaus	21,61
3				Kathi	19,03
4				Bert	22,31
5		30	23,35	Klaus	23,03
6				Bert	23,66
7		40	23,47	Rolf	22,72
8				Kathi	24,22
9		56	28,34	Klaus	28,34
10					
11					

Worksheet Output: *The final output table with rounded values*

2. Extended Problem

Extending the example by several values for weight per person and age results in the following calculation process. All values for WEIGHT are to be included in the calculation of the BMI average values per age and the total average value for BMI. If one first forms the average values for WEIGHT per person and age, a deviating result follows.

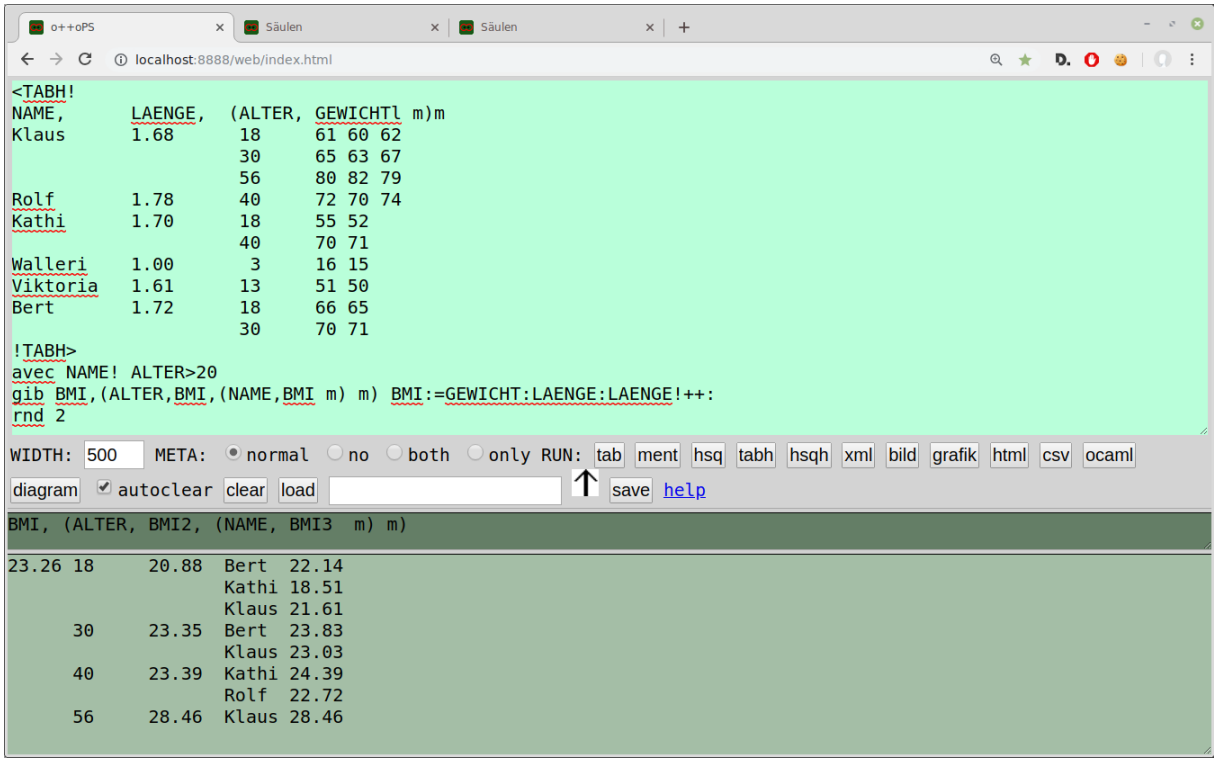
NAME	LAENGE	ALTER	GEWICHT
Klaus	1.68	18	61 60 62
		30	65 63 67
		56	80 82 79
Rolf	1.78	40	72 70 74
Kathi	1.70	18	55 52
		40	70 71
Walleri	1.00	3	16 15
Viktoria	1.61	13	51 50
Bert	1.72	18	66 65
		30	70 71

2.1. Solution with o++o

Input of the table as Tabment and the o++o program. We recognize that except for typing in the additional data, only an `l` for list after WEIGHT and an H at TAB must be inserted. The H stands for horizontal, since the weight data were represented now compactly horizontally.

```
<TABH!
NAME,       LAENGE,   (ALTER, GEWICHTl m)m
Klaus       1.68       18     61 60 62
                       30     65 63 67
                       56     80 82 79
Rolf        1.78       40     72 70 74
Kathi       1.70       18     55 52
                       40     70 71
Walleri     1.00        3     16 15
Viktoria    1.61       13     51 50
Bert        1.72       18     66 65
                       30     70 71
!TABH>
```

```
avec NAME! ALTER>20
gib BMI,(ALTER,BMI,(NAME,BMI m) m) BMI:=GEWICHT:LAENGE:LAENGE!++:
rnd 2
```



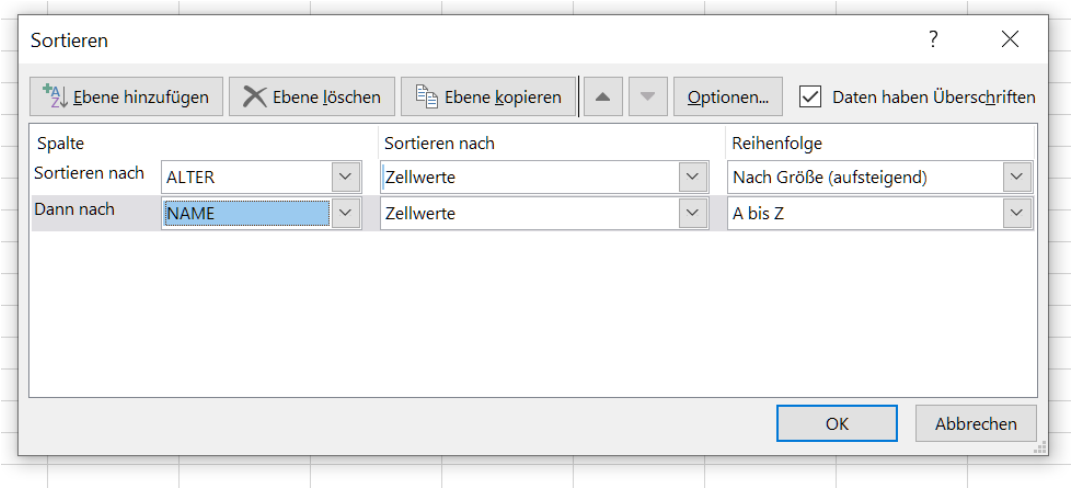
Extended problem with o++o: *Only the input table is modified and an output table with the corresponding schema is generated by the same code*

The desired diagram can be generated again by two clicks, if you transform again the type of ALTER into WORT. Since it is similar to the above diagram, we do not present a second one here.

2.2. Solution with Excel (manual)

After creating the required table columns, the values for NAME, LENGTH, AGE and WEIGHT are entered. This creates redundancy because each weight to be included must be in a row with the corresponding values for NAME and AGE. Filtering by the criterion "persons with age > 20" must again be done manually.

User-defined sorting is selected for sorting the table values by ALTER and NAME. This is semi-automatic, but must be triggered manually again after each modification of the input data.



Semi-automatic sorting in EXCEL: *A dialog allows manual selection of the columns to be sorted by*

H24

	A	B	C	D	E
1	NAME	LAENGE	ALTER	GEWICHT	
2	Bert	1,72	18	66	
3	Bert	1,72	18	65	
4	Kathi	1,7	18	55	
5	Kathi	1,7	18	52	
6	Klaus	1,68	18	61	
7	Klaus	1,68	18	60	
8	Klaus	1,68	18	62	
9	Bert	1,72	30	70	
10	Bert	1,72	30	71	
11	Klaus	1,68	30	65	
12	Klaus	1,68	30	63	
13	Klaus	1,68	30	67	
14	Kathi	1,7	40	70	
15	Kathi	1,7	40	71	
16	Rolf	1,78	40	72	
17	Rolf	1,78	40	70	
18	Rolf	1,78	40	74	
19	Klaus	1,68	61	80	
20	Klaus	1,68	61	82	
21	Klaus	1,68	61	79	
22					
23					

Flat sorted table in EXCEL: *The intermediate results include more data rows after the list of weights is resolved*

A new column BMIx is inserted in this table, in the cells of which the BMI values are calculated according to the formula =D2/B2^2. The formula from field E2 must be copied into all cells of column E. In doing so, the formulas are automatically adjusted according to the selected row.

Column BMI3 is calculated according to the formula =AVERAGE(E2:E3). It is necessary to pay attention to the correct selection of the ranges for the calculation of the average value.

Analogously, the values for BMI2 are calculated for the relevant ranges and BMI.

H2 =MITTELWERT(E2:E21)

	A	B	C	D	E	F	G	H
1	NAME	LAENGE	ALTER	GEWICHT	BMIx	BMI3	BMI2	BMI
2	Bert	1,72	18	66	22,3093564	22,1403461	20,8776213	23,2622421
3	Bert	1,72	18	65	21,9713359			
4	Kathi	1,7	18	55	19,0311419	18,5121107		
5	Kathi	1,7	18	52	17,9930796			
6	Klaus	1,68	18	61	21,6128118	21,6128118		
7	Klaus	1,68	18	60	21,2585034			
8	Klaus	1,68	18	62	21,9671202			
9	Bert	1,72	30	70	23,6614386	23,8304489	23,3502068	
10	Bert	1,72	30	71	23,9994592			
11	Klaus	1,68	30	65	23,0300454	23,0300454		
12	Klaus	1,68	30	63	22,3214286			
13	Klaus	1,68	30	67	23,7386621			
14	Kathi	1,7	40	70	24,2214533	24,3944637	23,3924276	
15	Kathi	1,7	40	71	24,567474			
16	Rolf	1,78	40	72	22,7244035	22,7244035		
17	Rolf	1,78	40	70	22,0931701			
18	Rolf	1,78	40	74	23,3556369			
19	Klaus	1,68	61	80	28,3446712	28,462774	28,462774	
20	Klaus	1,68	61	82	29,053288			
21	Klaus	1,68	61	79	27,9903628			
22								

Table with BMI values in EXCEL: *The columns BMIx, BMI3, BMI2 and BMI are added and determined in this order*

Now the number of decimal digits can be defined with "Format cells" and columns that are no longer required in the result display can be hidden. The calculation result then appears in EXCEL as follows:

	A	C	F	G	H
1	NAME	ALTER	BMI3	BMI2	BMI
2	Bert	18	22,14	20,88	23,26
4	Kathi		18,51		
6	Klaus		21,61		
9	Bert	30	23,83	23,35	
11	Klaus		23,03		
14	Kathi	40	24,39	23,39	
16	Rolf		22,72		
19	Klaus	61	28,46	28,46	
22					

Final output in EXCEL: *The calculation result is a formatted table*

2.3. Solution with Excel (automated)

The solution differs only with transition from 'gib-help-sort' to 'gib'. Another worksheet 'gib-avg-help' is used to mark rows with cells containing information about ALTER and NAME e.g.

`CONCATENATE(TEXT($BMI.A2;"##,##");$BMI.B2)`.

This results in new formulas for the BMI values:

- BMI `=AVERAGE($BMI.E2:E50)`
- BMI2 `=IF(B2=B1;"";AVERAGEIF(BMI.$A$2:$A$50;B2;BMI.$E$2:$E$50))`
- BMI3 `=IFERROR(IF(AND(B2=B1;D2=D1);"";AVERAGEIF('gib-avg-help'.$B$2:$B$50;CONCATENATE(TEXT(B2;"##,##");D2);BMI.$E$2:$E$50));"")`

The worksheet still contains redundant lines without BMI values (worksheet 'gib-holes').

These can be marked and selected in the same way as the empty rows in 'avec-holes' (worksheet 'gib'). Afterwards, rounding is performed as usual (worksheet 'rnd').

	A	B	C	D	E
1	BMI	ALTER	BMI2	NAME	BMI3
2	23,26	18	20,88	Klaus	21,61
3				Kathi	18,51
4				Bert	22,14
5		30	23,35	Klaus	23,03
6				Bert	23,83
7		40	23,39	Rolf	22,72
8				Kathi	24,39
9		56	28,46	Klaus	28,46
10					
11					

Worksheet BMI2 Output: *The final output table with rounded values*

3. Evaluation

Compared to working with spreadsheets, which requires a sequence of carefully formatted worksheets and intermediate results and a high number of user inputs, as shown in the example above, o++o can solve this task with a program consisting of 4 lines, which can be modified.

Spreadsheets are popular because content can be modified interactively and the result is immediately visible. Even simple applications like in the described example require a lot of work and precise knowledge of how the software works. With much less time and effort o++o generates an interactively modifiable program and provides immediately visible results. Furthermore, the possible modifications are much more far-reaching.

o++o allows a high flexibility of the calculations with low adaptation effort. For example, in the line

```
gib BMI,(ALTER,BMI,(NAME,BMI m) m) BMI:=GEWICHT:LAENGE:LAENGE!++:
```

only ALTER and NAME are swapped to realize a completely different request.

The **manual solution using EXCEL** for the extended example of calculating BMI values requires the following steps:

1. Creating the worksheet with table columns and entering the values for NAME, LENGTH, AGE and WEIGHT.
2. Manual filtering, because the menu function Filter is not expressive enough
3. Fill empty cells so that the Sort menu function can be applied with subsequent sorting
4. Enter the formula for BMI calculation into the top cell of the column BMIx
5. Copy this formula into all cells of this column (in the given example there are 20 cells)
6. Entering the formula for calculating the average BMI3 and copying this formula to 7 other cells
7. Adjusting the ranges in these cells
8. Enter the formulas for BMI2 and BMI and copy the formula for BMI2 into 3 more cells and additionally adjust the ranges
9. Application of the menu function "Format cells”
10. Hide 3 columns that are not needed to display the results.

Thus, the initial creation of this worksheet requires the application of 3 menu functions and the input of 4 formulas into cells to be calculated. These formulas must be copied into 29 additional cells. In addition, validity areas of the formulas must be adjusted in 12 cells.

If additional values are added to this example, the validity areas of the formulas must be adjusted in this worksheet.

The **automated solution with EXCEL** requires analogous steps over several worksheets, but does not require any subsequent adjustment of formulas when the input data changes.

The solution with o++o, on the other hand, only requires:

1. Input of the table
2. Input of an o++o program consisting of 2 or 3 lines.
3. The result can be called in different representation forms, including graphic output, with one (or two) click(s) each.

Our effort for solving this task with o++o was less than 20% of the time needed with EXCEL. Knowledge of the EXCEL and o++o systems was assumed for this assumption. However, for a more accurate and general estimate of the difference in effort, more extensive studies need to be done.

4. General Comparison

The comparison on the chosen example leaves many other aspects untouched. We present the following relevant differences between **EXCEL** and **o++o** independently of the above example:

1. EXCEL works with 2-dimensional tables, where both dimensions are equal. o++o tables always have a table header (for the horizontal dimension) with whose columns typically is calculated. The horizontal dimension is described by tuples (generalization of a pair), whereas the vertical dimension is specified by collections (list, set, bag ...).

2. Data entry in EXCEL can be faster than data entry in tab files of o++o, because the latter requires the entry of appropriate spaces. This is avoided with hsq files. However, these are somewhat more difficult to read for the beginner.

3. One must write each number in a separate cell in EXCEL. This quickly crowds the screen (of a smartphone). In o++o there is an analogous problem when entering in tab format. Beginner-friendly displays tend to show the table format directly and take up corresponding space.

4. Since EXCEL does not understand corresponding schemas, XML data are not directly processable.

5. One cannot make queries with the EXCEL concepts to databases, XML, ... with the EXCEL “concepts”. One must learn completely different languages (SQL, XQuery, ...) for it.

6. To understand EXCEL worksheets is very difficult, because one o++o formula usually corresponds to several EXCEL formulas and these EXCEL formulas are distributed over many cells and can be connected with references. Due to the low manageability, EXCEL worksheets are prone to errors and existing errors are difficult to correct.

7. o++o is based on mathematical concepts and set-wise operations, such as set, list, tuple, stroke-list-operation,.... The handling of EXCEL knows few concepts, but requires a lot of detailed knowledge (when and how are copied formulas adjusted; does an adjustment also take place when inserting or deleting rows or columns, etc.).

8. Because an o++o program contains few formulas, it is easier to read and modify.

9. Data and program are usually separated in o++o. Therefore data can be used by several programs without any problems.

10. o++o conditions, for example, select only data and not formulas. For example, for a simple condition "avec LENGTH>1.60" applied to our output table, there is no equivalent filter condition in EXCEL, since whole person records (several rows each) including child (AGE,WEIGHT) pairs are deleted. A condition "avec NAME! WEIGHT>70" cannot be mapped even after "flattening" the table, because it "contains" an existential quantifier (We are looking for all persons containing a weight entry greater than 70). Although o++o conditions are more expressive than EXCEL filter conditions, they are easier to oversee because there are no collisions with cell formulas.

11. For aggregations (sums, averages, maximum,...) per value must be presorted or grouped in EXCEL, not in o++o.

12. o++o can process the content aspects of structured tables directly. These are usually closer to the desired print image than flat tables. To create more specific print images, one can output

the data via XML and use a stylesheet language. In EXCEL, on the other hand, content issues and form issues are not separated.

13. o++o is based on an abstract tabment notion for data. A tabment (=TABle+docuMENT) can be represented in a variety of concrete ways (tab, xml, graph, diagram, ...) and compactly (hsq). EXCEL is based on **one** two-dimensional concrete view of a worksheet. This view includes data and formulas.

14. EXCEL formulas are more difficult to understand than o++o operations. A single formula may require more analysis than a full o++o program.