Allocations & Distributions

Allocations & Distributions with finaquant® protos

Included examples show how allocations and distributions can be calculated with finaquant protos, the non-commercal calculation engine library (.NET) based on table functions.

Written by Tunc Ali Kütükcüoglu ©Finaquant Analytics Ltd, 22. November 2012

Contents

Finaquant Analytics GmbH

Digitally signed by Finaquant Analytics GmbH
DN: cn=Finaquant Analytics GmbH, c=CH, o=Finaquant Analytics GmbH
Reason: I am the author of this document
Location: Herrliberg/Switzerland
Date: 2012.11.22 23:27:49 +01'00'

Getting and using the source code (C#)

All the source code in C# required for running the examples below including the simple distribution function can be downloaded at the product page of finaquant® protos.

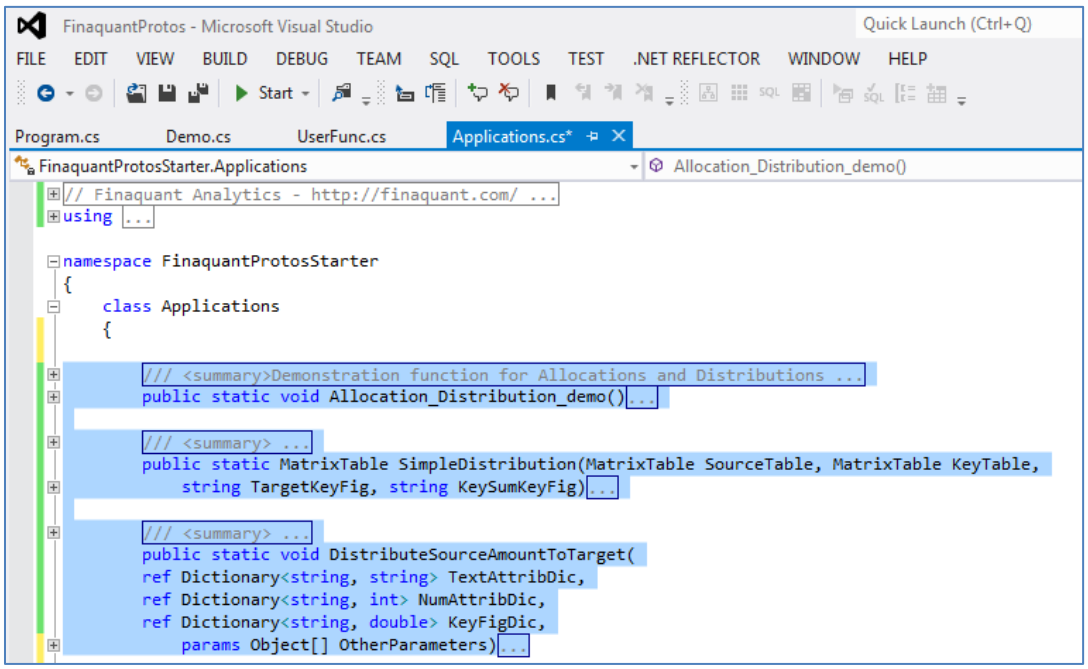
The downloadable zip package contains:

- This pdf document as User Manual
- Code file Applications.cs

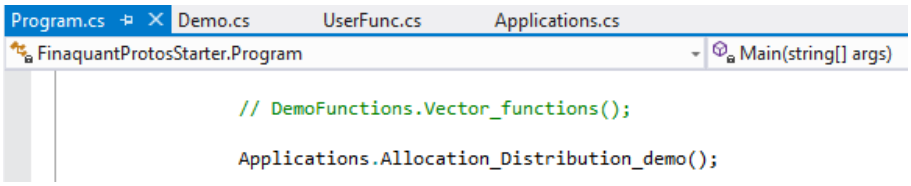
You will find three functions (methods in C# programming jargon) in the code file Applications.cs:

1. Allocation_Distribution_demo()
2. SimpleDistribution()
3. DistributeSourceAmountToTarget()

Copy all these three functions and paste them into the code file Applications.cs in your Visual Studio project FinaquantProtosStarter, into the area defined by the class Application without deleting any existing method or code already existing in this class.



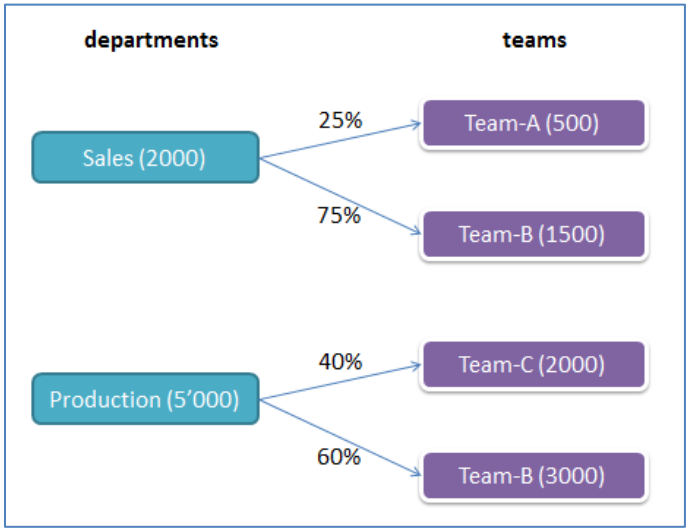
You can now run the demonstration function by first adding the statement `Applications.Allocation_Distribution_demo()` into the code file Program.cs then pressing the key F5.



For any problems and questions please refer to product's forum at:
http://finaquant.com/forum/finaquant-protos

Allocations and Distributions

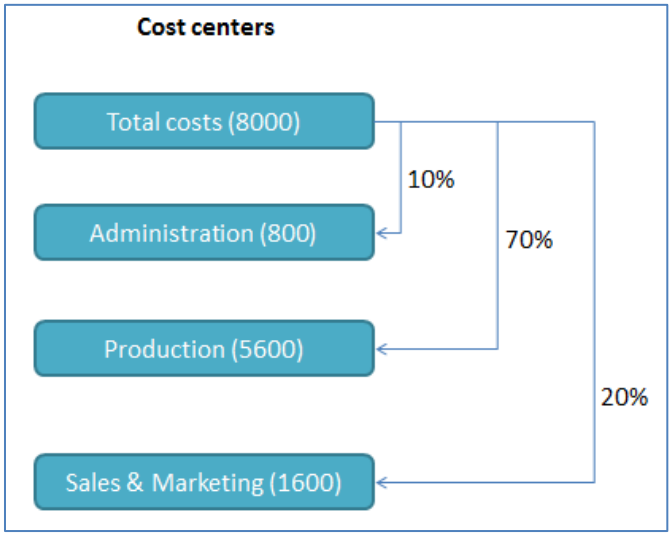
Though most commonly used for financial planning and accounting, allocations and distributions can be needed in any calculation where proportionate (pro-rata) distribution of some amounts is required.



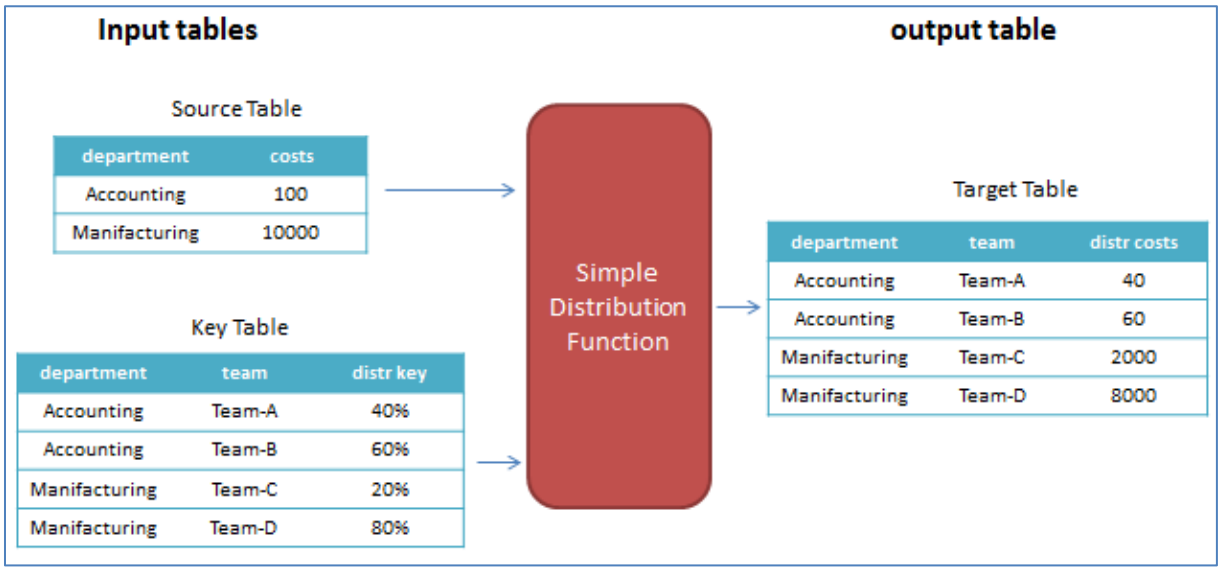
A classical distribution example is distributing costs obtained at company level further to departments and teams in proportion to given key amounts or ratios. Another example could be distributing bonus amounts calculated at department level further to teams and individuals.

Distribution is done from a set of entities to other entities. Let's take distribution of costs from the entity *department* to entity *team* as an example. The initial table `costs(department)` is transformed into a bigger table `costs(department, team)` by the distribution with the additional field *team*.

Allocation in turn means distributing some amounts within the same entity group. A typical example is distributing costs from some central cost centers to other de-central cost centers. Note that no additional fields are added into the initial table `costs(cost_center)` by the allocation; only new field values.



In the examples below, you will see how a distribution can be accomplished in four simple steps using the available table functions in finaquant® protos. These four steps will be than packed into a general distribution function with tables as input and output parameters as shown below:



The simple distribution function (which is a table function) has two input tables, and an output table, plus some additional detail parameters like the names of new key figures to be added:

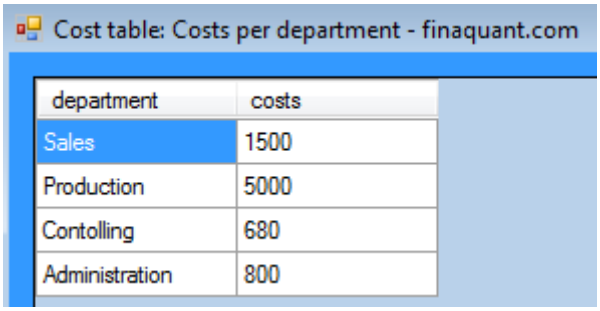
```
OutputTable = SimpleDistribution(SourceTable, KeyTable,
      … detail parameters)
```

Note that an allocation can be formulated as a special case of distribution, as you will see in the following examples.

Distribution in four steps: Distribution of costs

In this example, costs are distributed from department to teams and persons. Following cost and key tables are given as inputs:

SourceTable from which the amounts are distributed



KeyTable with distribution keys (or ratios)

Key table: Key table for distribution from department to teams and persons - finaquant.com

department	team	person	costs_key
Sales	Team A	Tom	2
Sales	Team A	Jim	6
Sales	Team B	Susan	3
Sales	Team B	Agnes	4
Production	Team C	Hakan	1
Production	Team C	Timur	2
Production	Team D	Jang	3
Production	Team D	Moris	4

The strategy for calculating the distributed costs is the following:

- Ensure that we have every attribute and key figure we need to calculate the distributed costs by combining the tables and adding the necessary key figures into the combined table.
- Calculate the distributed costs by row-by-row processing of combined table.

This is a simple and effective strategy which can be used for many other kinds of table calculations. First, ensure that you have all the required parameters as fields of the of table, than calculate the desired fields by row-by-row processing of the table.

Step 1: Combine cost and key tables

```
// combine tables
NumVector MatchedRowsTbl1, MatchedRowsTbl2;
var CombinedTbl1 = MatrixTable.CombineTables(CostKeyTable, CostTable,
      null, null, null, out MatchedRowsTbl1, out MatchedRowsTbl2);

// filter out unmatched rows
CombinedTbl1 = MatrixTable.PartitionRow(CombinedTbl1, MatchedRowsTbl1);

// view table
MatrixTable.View_MatrixTable(CombinedTbl1,
      "Step 1: Combined cost and key tables");
```

Step 1: Combined cost and key tables - finaquant.com

department	team	person	costs_key	costs
Sales	Team A	Tom	2	1500
Sales	Team A	Jim	6	1500
Sales	Team B	Susan	3	1500
Sales	Team B	Agnes	4	1500
Production	Team C	Hakan	1	5000
Production	Team C	Timur	2	5000
Production	Team D	Jang	3	5000
Production	Team D	Moris	4	5000

We have now two key figures *costs* and *cost_key* in the combined table as shown above. The aggregation key figure *key_sum* (sum of *cost_key* w.r.t. *department*, the common attribute of cost and key tables) is still missing. We need key_sum to calculate distributed costs:

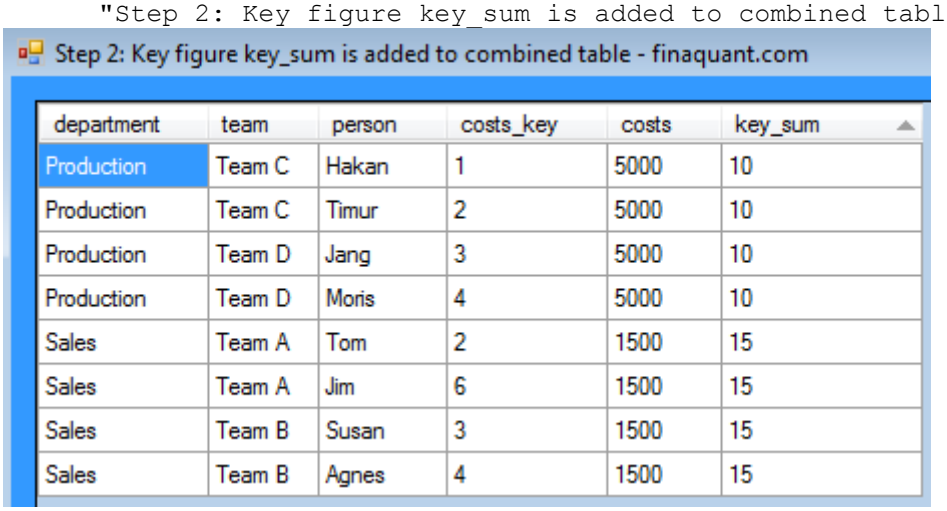
```
costs_distributed = costs x (costs_key / key_sum)
```

Step 2: Add key figure "key_sum" into the combined table

```
bool IfSuccess;
string Warnings;

// add aggregation key figure into table
CombinedTbl1 = MatrixTable.AggregateSelectedKeyFigure_B(CombinedTbl1,
      TextVector.CreateVectorWithElements("department"),
      "costs_key", "key_sum", AggregateOption.nSum,
      out IfSuccess, out Warnings);

// view table
MatrixTable.View_MatrixTable(CombinedTbl1,
      "Step 2: Key figure key_sum is added to combined table");
```



department	team	person	costs_key	costs	key_sum
Production	Team C	Hakan	1	5000	10
Production	Team C	Timur	2	5000	10
Production	Team D	Jang	3	5000	10
Production	Team D	Moris	4	5000	10
Sales	Team A	Tom	2	1500	15
Sales	Team A	Jim	6	1500	15
Sales	Team B	Susan	3	1500	15
Sales	Team B	Agnes	4	1500	15

Step 3: Insert new key figure "costs_distributed" into combined table

This key figure must be inserted into the table so that it can be calculated by row-by-row processing of table with the formula stated above.

```
// insert key figure "costs_distributed" into combined table
CombinedTbl1 = MatrixTable.InsertNewColumn(CombinedTbl1,
"costs_distributed", FillAllValue: 0.0);

// view table
MatrixTable.View_MatrixTable(CombinedTbl1,
      "Step 3: New key figure costs_distributed is inserted into to
combined table");
```

Step 3: New key figure costs_distributed is inserted into to combined table - finaquant.com

department	team	person	costs_key	costs	key_sum	costs_distributed
Sales	Team A	Tom	2	1500	15	0
Sales	Team A	Jim	6	1500	15	0
Sales	Team B	Susan	3	1500	15	0
Sales	Team B	Agnes	4	1500	15	0
Production	Team C	Hakan	1	5000	10	0
Production	Team C	Timur	2	5000	10	0
Production	Team D	Jang	3	5000	10	0
Production	Team D	Moris	4	5000	10	0

The initial value of the new key figure *costs_distributed* is zero for all rows of the combined table. The resultant values for this key figure will be calculated with the next and last step.

Step 4: Calculate distributed costs

```
// row-by-row processing with user-defined function
CombinedTbl1 = MatrixTable.TransformRowsDic(CombinedTbl1,
DistributeSourceAmountToTarget,
      "costs_distributed",     // target key figure
      "costs",                 // source key figure
      "costs_key",             // key
      "key_sum");              // sum

// view table
MatrixTable.View_MatrixTable(CombinedTbl1,
      "Step 4: Combined table after distribution");
```

Step 4: Combined table after distribution - finaquant.com

department	team	person	costs_key	costs	key_sum	costs_distributed
Sales	Team A	Tom	2	1500	15	200
Sales	Team A	Jim	6	1500	15	600
Sales	Team B	Susan	3	1500	15	300
Sales	Team B	Agnes	4	1500	15	400
Production	Team C	Hakan	1	5000	10	500
Production	Team C	Timur	2	5000	10	1000
Production	Team D	Jang	3	5000	10	1500
Production	Team D	Moris	4	5000	10	2000

User-defined function `DistributeSourceAmountToTarget` as input to the table transformation function above implements the formula for distributed costs:

```
public static void DistributeSourceAmountToTarget(
…
// Calculate distributed costs:
KeyFigDic[target_keyfig] =
KeyFigDic[source_keyfig] * KeyFigDic[key_keyfig] / KeyFigDic[sum_keyfig];
…
```

Simple Distribution Function

All the four steps for calculating the distributed amounts can be packed into a single general distribution function with the following input and output parameters:

```
OutputTable =
SimpleDistribution(SourceTable,  KeyTable, TargetKeyFig, KeySumKeyFig)
```

where:

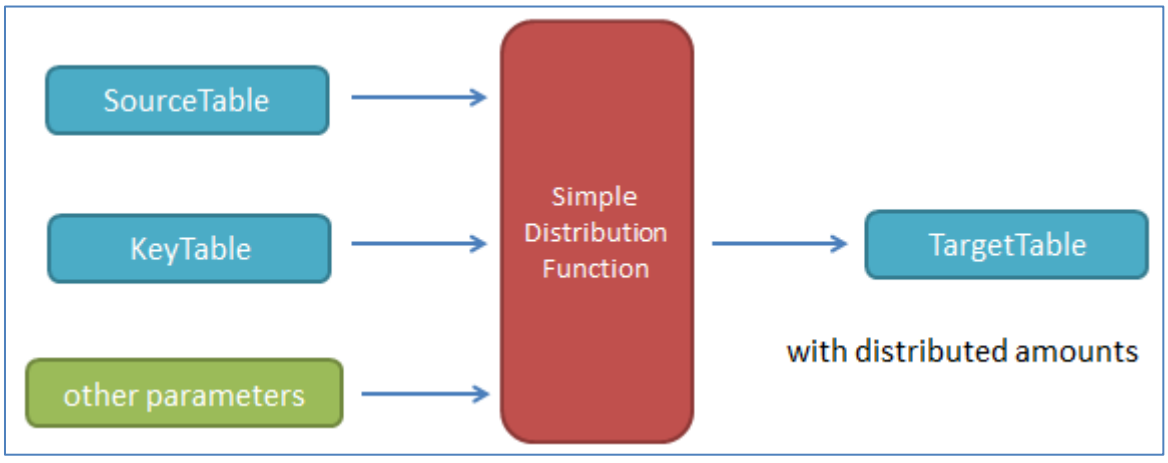
- *TargetKeyFig* is the name of the resultant key figure with distributed amounts, like costs_distributed
- *KeySumKeyFig* is the name of the aggregated key figure as sum of key values w.r.t. common attributes, like key_sum

Following conditions must be satisfied for input parameters of the distribution function:

- Both *SourceTable* and *KeyTable* must have exactly one key figure (representing source and key amounts respectively)
- *SourceTable* and *KeyTable* must have at least one common attribute (text or numeric)
- Both input tables must share the same MetaData object (i.e. common data universe)
- The key figures *TargetKeyFig* and *KeySumKeyFig* must be defined in MetaData
- Both input tables must not contain the key figures *TargetKeyFig* or *KeySumKeyFig*

The exact signature of the distribution function in C# is as follows:

```
public static MatrixTable SimpleDistribution(MatrixTable SourceTable,
MatrixTable KeyTable, string TargetKeyFig, string KeySumKeyFig)
```



Let's test this function with the same input tables (cost and cost key) introduced in the first example above (distribution in 4 steps):

```
MatrixTable CombinedTbl2 =
SimpleDistribution(SourceTable: CostTable, KeyTable: CostKeyTable,
      TargetKeyFig: "costs_distributed", KeySumKeyFig: "key_sum");

// view table
MatrixTable.View_MatrixTable(CombinedTbl2,
      "Result of SimpleDistributuion(): Output table with distributed
amounts");
```

Result of SimpleDistributuion(): Output table with distributed amounts - finaquant.com

department	team	person	costs_key	costs	key_sum	costs_distributed
Sales	Team A	Tom	2	1500	15	200
Sales	Team A	Jim	6	1500	15	600
Sales	Team B	Susan	3	1500	15	300
Sales	Team B	Agnes	4	1500	15	400
Production	Team C	Hakan	1	5000	10	500
Production	Team C	Timur	2	5000	10	1000
Production	Team D	Jang	3	5000	10	1500
Production	Team D	Moris	4	5000	10	2000

So we obtained the same distribution results. That is, the four distribution steps explained in the previous example are successfully packed into a function with generalized input and output parameters. Now you can execute the distribution function on any pair of input tables (source and key) provided that all the parameter conditions listed above are satisfied.

Note that the input tables can have any number of attributes, and they can also have multiple common attributes. In the previous example we had one common attribute: *department*. In the next example we have multiple common attributes, namely *department* and *country*. Source and key amounts are specified with this attribute pair.

Distributing bonus amounts to teams and years

In this example scenario, a company with headquarters in Paraguay has sales and production departments in countries Bolivia and Peru. The bonus amounts as performance incentives for employees are first calculated per department and country. These bonus amounts are then distributed to teams and years. Bonus amounts are distributed also to years, because the bonus payments will be done at the end of 2013 and 2014.

Source table: Bonus amounts per department and country

Bonus table - finaquant.com

department	country	bonus
Sales	Bolivia	1500
Sales	Peru	1200
Production	Bolivia	2200
Production	Peru	3300

KeyTable: Key amounts for the distribution

Bonus key table - finaquant.com

department	country	team	year	bonus_key
Sales	Bolivia	Team A	2013	3
Sales	Bolivia	Team B	2013	4
Sales	Bolivia	Team A	2014	1
Sales	Bolivia	Team B	2014	2
Sales	Peru	Team C	2013	4
Sales	Peru	Team D	2013	6
Sales	Peru	Team C	2014	1
Sales	Peru	Team D	2014	1
Production	Bolivia	Team A	2013	4
Production	Bolivia	Team B	2013	4
Production	Bolivia	Team A	2014	1
Production	Bolivia	Team B	2014	2
Production	Peru	Team C	2013	5
Production	Peru	Team D	2013	3
Production	Peru	Team C	2014	1
Production	Peru	Team D	2014	2

The simple distribution function will be tested again with these new input tables:

```
// Distribute
MatrixTable ResultTbl = SimpleDistribution(SourceTable: BonusTable,
KeyTable: BonusKeyTable,
      TargetKeyFig: "bonus_distributed", KeySumKeyFig: "key_sum");

// view table
MatrixTable.View_MatrixTable(ResultTbl,
      "Result of SimpleDistribution() with bonus and key tables as input:
Output table with distributed bonus amounts");
```

Result of SimpleDistribution() with bonus and key tables as input: Output table with distributed bonus amounts

department	country	team	year	bonus_key	bonus	key_sum	bonus_distributed
Sales	Bolivia	Team A	2013	3	1500	10	450
Sales	Bolivia	Team B	2013	4	1500	10	600
Sales	Bolivia	Team A	2014	1	1500	10	150
Sales	Bolivia	Team B	2014	2	1500	10	300
Sales	Peru	Team C	2013	4	1200	12	400
Sales	Peru	Team D	2013	6	1200	12	600
Sales	Peru	Team C	2014	1	1200	12	100
Sales	Peru	Team D	2014	1	1200	12	100
Production	Bolivia	Team A	2013	4	2200	11	800
Production	Bolivia	Team B	2013	4	2200	11	800
Production	Bolivia	Team A	2014	1	2200	11	200
Production	Bolivia	Team B	2014	2	2200	11	400
Production	Peru	Team C	2013	5	3300	11	1500
Production	Peru	Team D	2013	3	3300	11	900
Production	Peru	Team C	2014	1	3300	11	300
Production	Peru	Team D	2014	2	3300	11	600

Note that the sum of key amounts (`key_sum`) are obtained by aggregating `bonus_key` with respect to the attribute pair department and country, as common attributes of input tables.

Allocating profits to profit centers

This example will illustrate that allocations can be formulated as a special case of distribution.

In this example scenario, the profits accumulated at a main hub are distributed to other cost centers according to the given distribution keys.

Source table: Accumulated profits

Profit table - finaquant.com

profitcenter_source	profit
Profit Center Main Hub	12000

KeyTable: Key amounts for the allocation of profits to other cost centers

Profit key table - finaquant.com

profitcenter_source	profitcenter_target	profit_key
Profit Center Main Hub	Profit Center - Computers	2
Profit Center Main Hub	Profit Center - Office Materials	1
Profit Center Main Hub	Profit Center - Household goods	4
Profit Center Main Hub	Profit Center - Digital cameras	3

Just by separating the attribute *profitcenter* into two (source and target) we have transformed the allocation into a distribution problem. We can now apply the distribution function on these source and key tables:

```
// Distribute
ResultTbl = SimpleDistribution(SourceTable: ProfitTable, KeyTable:
ProfitKeyTable,
      TargetKeyFig: "profit_distributed", KeySumKeyFig: "key_sum");

// view table
MatrixTable.View_MatrixTable(ResultTbl,
      "Allocation results with profit and key tables as input: Output table
with distributed profits");
```

Allocation results with profit and key tables as input: Output table with distributed profits - finaquant.com

profitcenter_source	profitcenter_target	profit_key	profit	key_sum	profit_distributed
Profit Center Main Hub	Profit Center - Computers	2	12000	10	2400
Profit Center Main Hub	Profit Center - Office Materials	1	12000	10	1200
Profit Center Main Hub	Profit Center - Household goods	4	12000	10	4800
Profit Center Main Hub	Profit Center - Digital cameras	3	12000	10	3600

Conclusions

Simple proportionate distribution can be accomplished by simple table functions in four steps. All these steps can be packed into a general distribution function with two input tables representing the source (initial values to be distributed) and key (distribution ratios) amounts.

Once you have your distribution function, you can forget about the details of the function (four steps etc.) and simply apply this function on any input table pairs specifying the source and key amounts, provided that:

a. You have a simple proportionate (pro-rata) distribution case.
b. All the parameter conditions for the distribution function are satisfied.

There are indeed more complex conditional allocation and distribution cases that cannot be captured by the simple distribution function. Such distribution functions will be offered with the commercial version of finaquant® protos: finaquant® calcs.

Nevertheless, if you have sufficient analytical skills, you can calculate almost any kind of distribution with the table functions of finaquant® protos. The key functions will be table combinations, aggregations and transformations (especially row-by-row processing).

Allocations can be formulated as a distribution problem, as the last example above illustrates (profit allocation).