Getting Started with Tabular

# Introduction

This document is a brief walkthrough of how to use the Tabular add-in for Excel to perform inference on relational data.

We present Tabular’s model notation and illustrate inference on a practical example: player ranking using a simplified TrueSkill model.

# Requirements

Excel 2013, 32 or 64 bit edition. .NET Framework version 4.5.

# Disclaimer

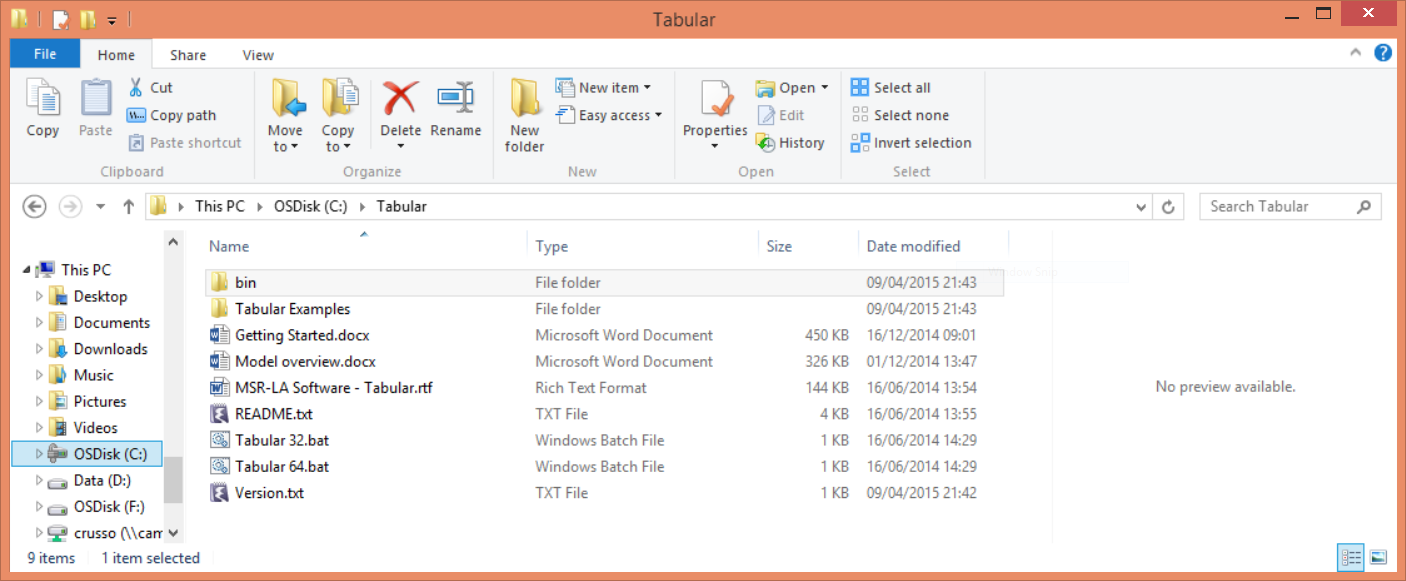
Note this is alpha-release, research software. When loaded, the add-in will sometimes crash Excel. However, it should not corrupt your Excel installation. We strongly recommend you save any other open Excel workbooks before using the add-in.

# Feedback

We welcome feedback and bug reports. Feedback may be sent to the Tabular Distribution Group [tabular-discussions](mailto:tabular-discussions@microsoft.com), and bug reports to <tabular-bugs@microsoft.com>.

# Installation

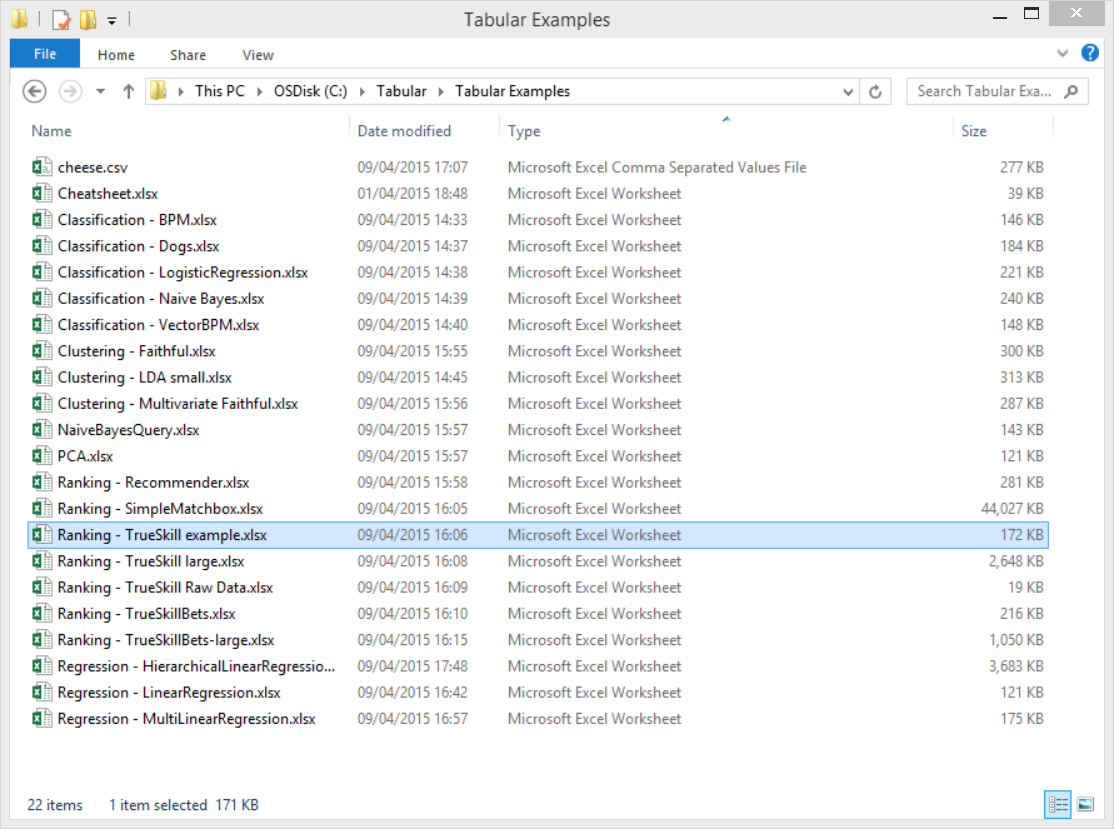
Tabular is delivered as an Excel add-in stored in a zip file. The unzipped folder contains the following:



The Tabular folder contains two .bat files, Tabular 32.bat and Tabular 64.bat. Double-click the appropriate .bat file to launch your 32- or 64-bit version of Excel with the add-in installed.

(The Tabular*XX*.bat file simply starts Excel with bin\TabularTaskPaneDNA-AddIn*XX*.xll. The .xll file installs the add-in for the duration of your Excel session. No windows installer is involved and the changes to Excel are temporary.)

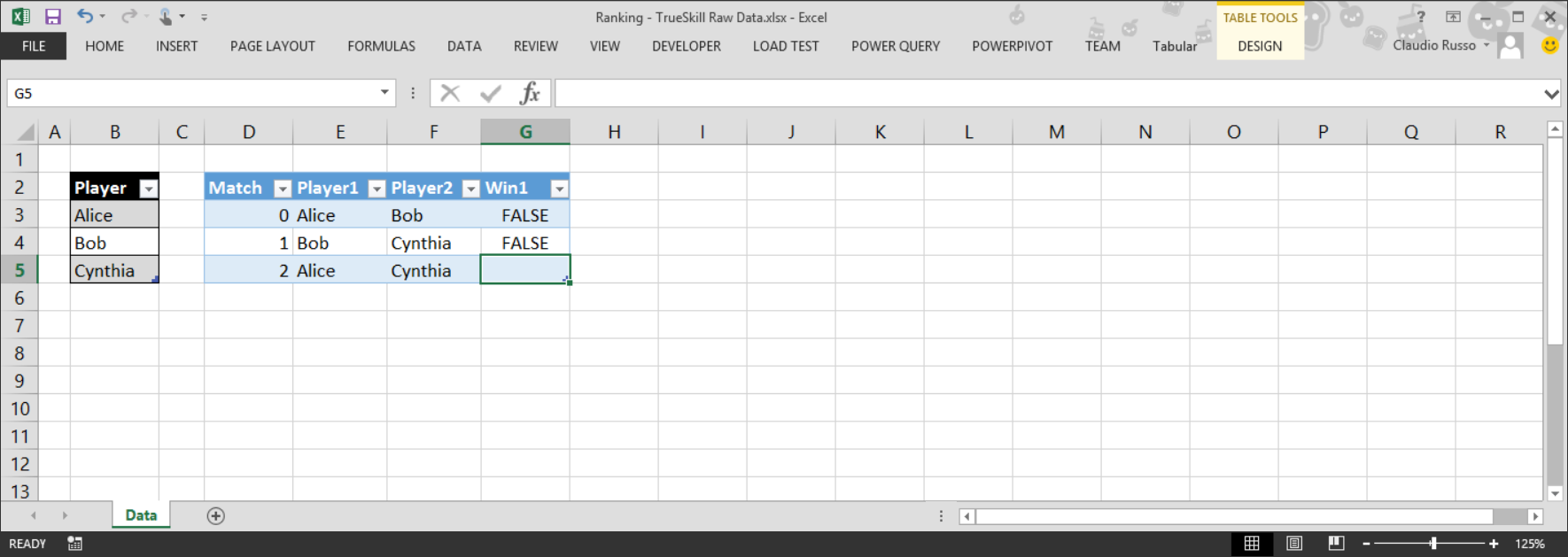
Folder Tabular Examples contains some sample Excel workbooks that you can open and explore after installing the add-in. (Cheatsheet.xlsx is a spreadsheet that summarizes the syntax of Tabular by example – it doesn’t contain any models.)



# Walkthrough: TrueSkill player ranking

Our example is a reconstruction of the TrueSkill player ranking model. The model can be used to assess the relative skills of players in a two-player game (e.g. chess), based on the known outcomes of a collection of matches. The model can also be used to predict the outcomes of tentative matches between players.

Our starting point is some raw data in the workbook Tabular Examples\ Ranking - TrueSkill Raw Data.xlsx. Open the workbook. Sheet Data contains two Excel tables, **Players** and **Matches**.



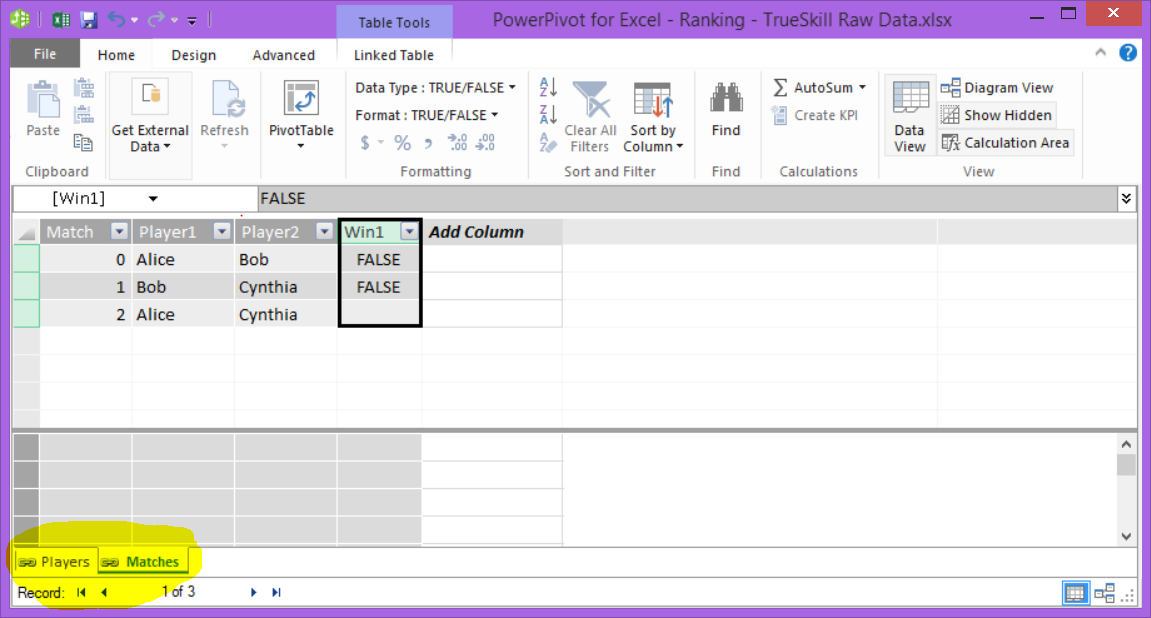
The **Players** table stores unique **Player** identifiers (column **Player**) together with potentially other attributes like age, sex, etc... The **Matches** table records the outcome of individual matches, identified by column **Match**, between pairs of players, **Player1** and **Player2**. **Player1** and **Player1** are indices into the **Players** table. The boolean **Win1** column records whether **Player1** has won the match (otherwise **Player2** has won – there are no ties).

(The data set is deliberately small to ease explanation – a much larger data set can be found in Tabular Examples\ Ranking - TrueSkill large.xlsx, which has 100 players and 20000 matches.)

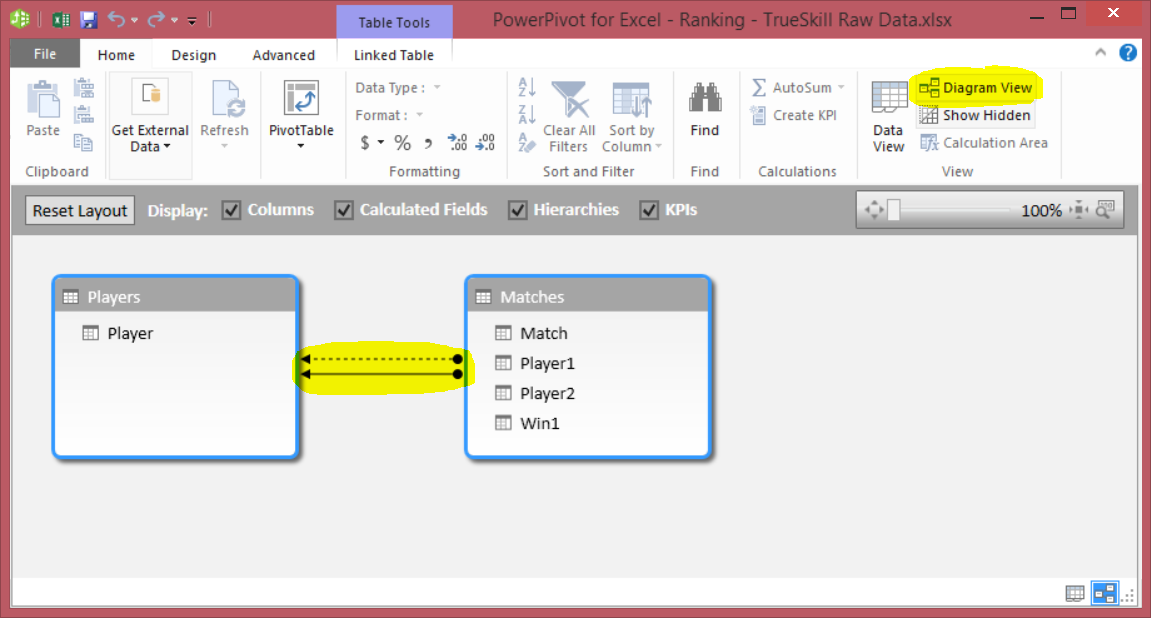
We’d like to use the match data to (1) infer the relative ranking of players and (2) to predict the outcome of matches with missing **Win1** values. Notice that match 2 is missing a value in column **Win1**. Tabular will interpret the missing data as a query to infer a distribution of values for that cell.

Our first step it to use Excel’s *PowerPivot* feature to construct a relational data model for our raw tables. Tabular relies on the Excel PowerPivot data model to obtain the types of table columns (which must be uniform within a column) and to interpret raw integers as links to other tables (such as links between the **Matches**.**Player1** column and the **Players**.**Player** column).

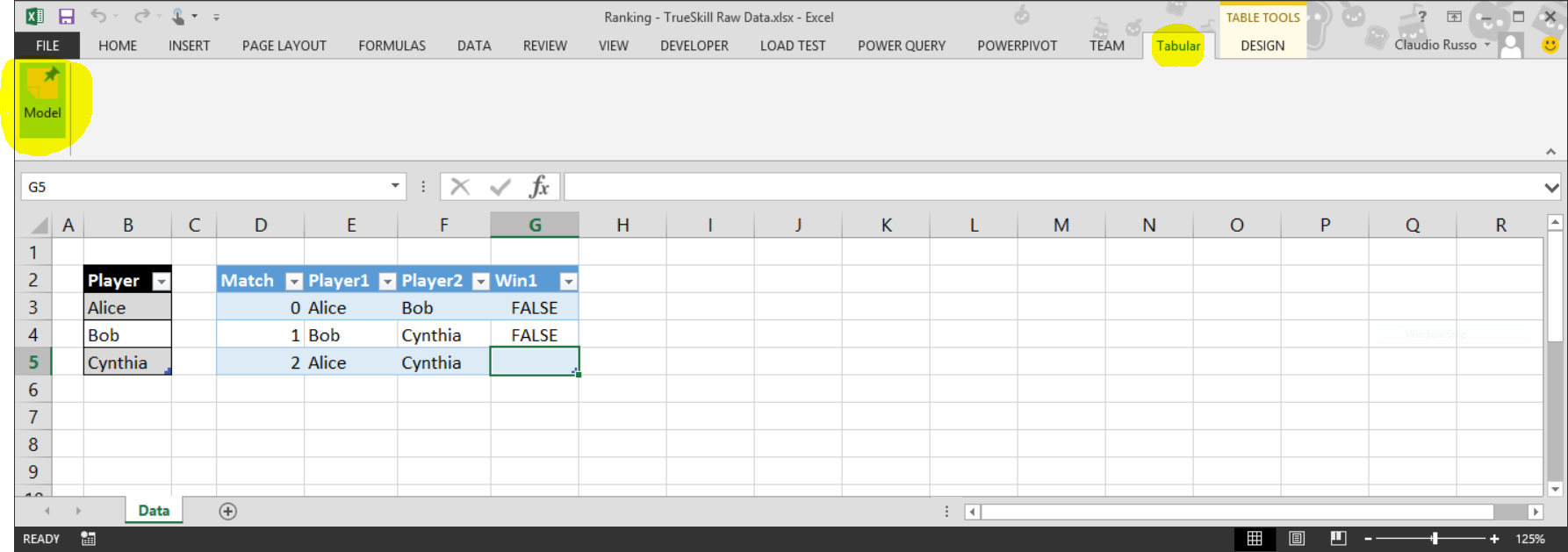
Select each table, go to the PowerPivot pane and add the table to the data model. Within PowerPivot, rename the tables from their defaults (typically Table1 and Table2) to **Players** and **Matches**. Theses PowerPivot table names will be referenced from our Tabular model.



From the PowerPivot Diagram View, draw two links from columns **Matches**.**Player1** and **Matches**.**Player2** to the **Players.Player** column to declare the foreign key relationships (e.g. **Player1** is a foreign key into **Players**).



Now we are ready to use the Tabular add-in. First, click on the Tabular menu. This displays the Tabular ribbon with a single button called Model.

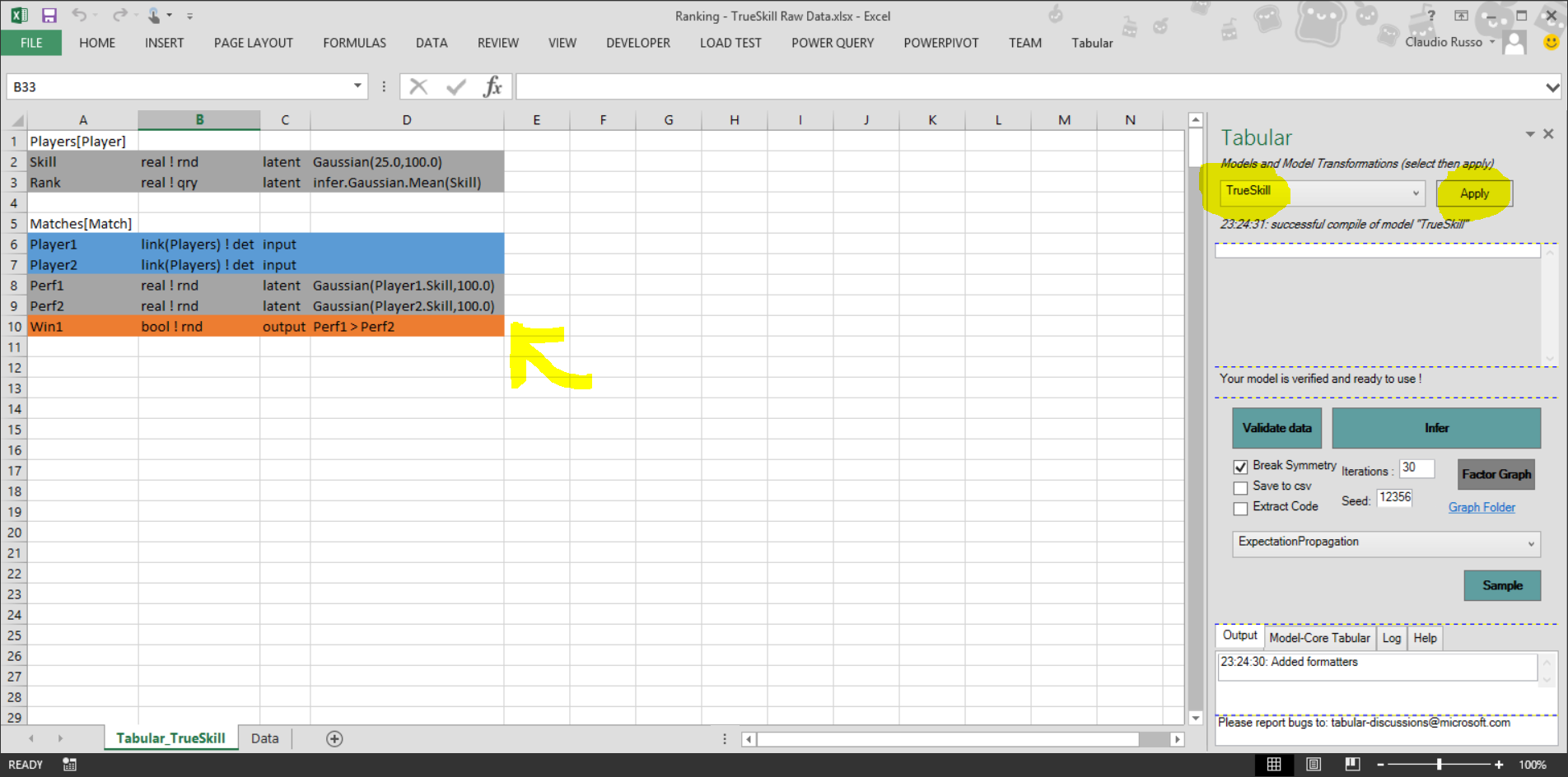


Clicking on Model brings up the Tabular task pane (on the right in the next screen shot).

The task pane is used to report various diagnostics and to perform inference once the user has authored a model for the data.

Once we’ve added our tables to the Excel data model, we can start using Tabular to construct a Tabular model for the data. The Tabular model is a simple program, authored in the top-left corner of a separate Excel sheet. Tabular relies on the name of the sheet to determine whether it contains a model. Any sheet whose name begins with “Tabular\_” is recognized as containing a model.

In the task pane, use the drop down menu to select the default TrueSkill model, which is built into Tabular for convenience only. This creates a new sheet called Tabular\_TrueSkill, with a pre-supplied model in the top-left corner of the sheet. The sheet can also be created manually, provided it is named Tabular\_*ModelName* and authored in the top-left corner of the sheet. The colouring is automatic.



Note that the task pane always operates on the active sheet and it is possible to have several model worksheets in the same workbook, though only one will be active at a time.

Let’s zoom in on the model:

|  |  |  |  |
| --- | --- | --- | --- |
| Players[Player] |  |  |  |
| Skill | real ! rnd | latent | Gaussian(25.0,100.0) |
| Rank | real ! qry | latent | infer.Gaussian.Mean(Skill) |
|  |  |  |  |
| Matches[Match] |  |  |  |
| Player1 | link(Players) ! det | input |  |
| Player2 | link(Players) ! det | input |  |
| Perf1 | real ! rnd | latent | Gaussian(Player1.Skill,100.0) |
| Perf2 | real ! rnd | latent | Gaussian(Player2.Skill,100.0) |
| Win1 | bool ! rnd | output | Perf1 > Perf2 |

The model declares each table in dependency order using the PowerPivot table names. Each table is defined by a set of column declarations (displayed horizontally). Each declaration declares the column’s name (e.g. **Skill**), type (e.g. **bool**, **real** or **link**(**Players**)), modifier (**input**, **output**, **latent** etc.) and, for non-input columns, an expression giving its probabilistic definition in terms of previously declared columns of the same table. Types are further refined by the space their value inhabit. The space is invariable deterministic (**det**), random (**rnd**) or query (**qry**) corresponding, roughly, to value computed from data, distributions inferred by inference and deterministic properties/parameters of distributions obtained by querying the distributions obtained through inference.

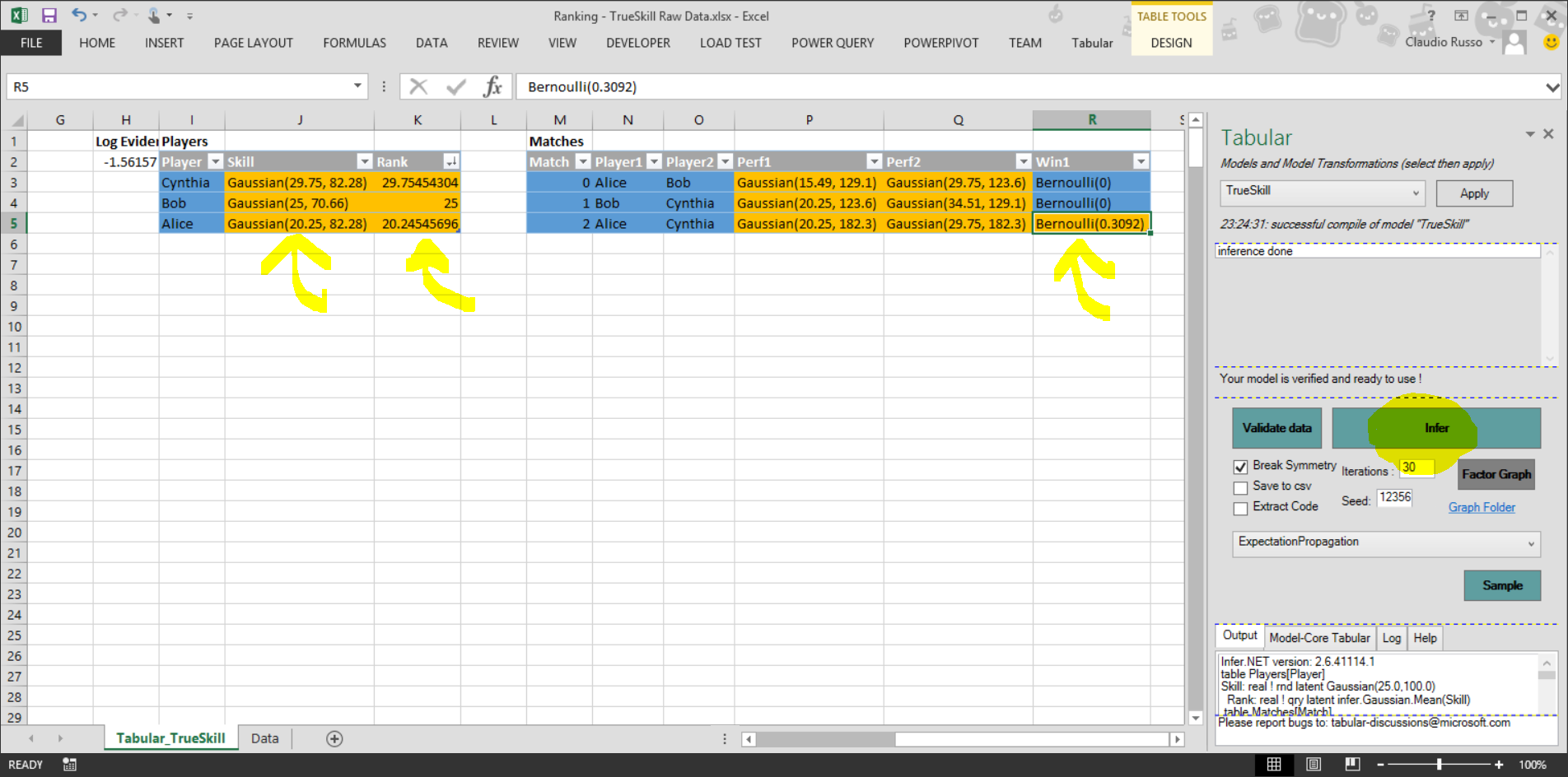
Columns of other tables are accessed through *links* (e.g. **Player1**.**Skill**). Columns marked **input** (blue) and **output** (orange) must be present in the table being modelled (i.e. the concrete data). Columns marked **latent** (grey) are fictional columns that are introduced only as part of the model. The idea is that each table of the model defines a “storyline” of how the output and latent columns of the table are generated from the inputs and, through any links, the models of preceding tables.

For instance, the model for the **Players** table says that each row has a latent **Skill** drawn from a Gaussian (25.0,100.0), i.e. a normal distribution with mean 25 and variance 100. The **Rank** of a player is obtained by inferring (i.e. querying) the **Mean** of their (Gaussian) Skill.

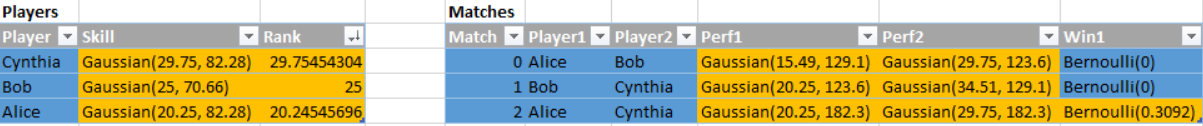
The model for the **Matches** tables says that each row takes the foreign keys **Player1** and **Player2** as input, generates the performances **Perf1** and **Perf2** by adding some Gaussian noise to the **Skill**s of the players (**Player1**.**Skill** and **Player2**.**Skill**), and makes **Win1** TRUE whenever **Perf1** > **Perf2**.

Tabular automatically checks the model for type and space correctness, in particular ensuring that query space values only depend on random space values that in turn may only depend on deterministic input. If the model is sensible, the **Infer** button is enabled in the Tabular task pane. Clicking on **Infer** causes Tabular to perform Bayesian inference on the model. Inference can be cancelled by clicking **Stop** during inference.

The results of inference are displayed next to the model, as tables with the original output columns and inferred latent columns. Missing values from the original table are displayed as uncertain values (i.e. distributions) in the result table:



Let’s zoom in on the inferred data:



Here, Tabular has inferred the relative skills of the **Players** in the latent **Skill** column, as Gaussian distributions, so we can see that Alice probably ranks below Bob and Cynthia because the Mean of her inferred **Skill** is lowest (20.25) (with Variance (82.28)). We can also see that Alice is likely to lose the match against Cynthia – her chance of winning is just a Bernoulli with P(True)=0.3092. Notice that match 2 has an uncertain outcome – because the value of **Win1** was missing for match 2 in the concrete data (the **Players** table on the Data sheet). However, matches 0 and 1 have certain outcomes (P(True) = 0) since the values were both present and FALSE in the data.

Tabular also reports the *log evidence* for the model – this measures the likelihood of the model given the data and can be used to compare different models of the same data.

We can now play with the model, adjusting parameters or introducing new features to the model to better account for the data and improve predictive power, like age, training, etc.

That’s it. For now, please consult the other examples and Tabular Examples\Cheatsheet.xlsx to discover more of the syntax of Tabular.

# Keys and Primary ID’s of tables

Tabular needs to associate a key or primary ID with each table. The key is a column whose values uniquely identify each row of the table (such as **Player** in the **Players** table). They can be integers or strings and must to be unique to each row.

In order to identify which column contains the key of a table, Tabular uses the following heuristics, choosing the first one that applies:

1. **A column name specified in the model, following the table name and surrounded by square brackets (e.g. Players[Player]);**
2. a column named “ID”;
3. a column whose name is the singularized version of the table’s name (e.g. Factories / Factory);
4. a column used by any other tables as a foreign key;
5. a column whose name is the singularized version of the table followed by “ID”;
6. otherwise the ID is the position or index of the row in the table, in the order in which it was returned from PowerPivot (which is, unfortunately, arbitrary).

(The heuristics are needed because PowerPivot does not, as far as we are aware, let one define the primary key of a table)

We strongly recommend sticking to option 1.

# Saving settings

The additional settings related to the inference, namely **Iterations** and **Algorithm**, of a model can optionally be saved in a pseudo-table named **Settings**.

In order to do so, write a row containing the name of each settings to be saved in the first column, followed in the second column by the value to give that setting. The Clustering – Faithful examples uses this to select VariationalMessagePassing (VMP) as its inference algorithm, overriding the default choice of ExpectationPropagation (EP).

# Other Functionality

### Help!

The **Help** tab of the Tabular task pane contains a concise description of the full Tabular syntax.

## Space Inference.

Spaces on types (but not the types themselves) may be omitted and will be inferred by the Tabular type checker. For example, this abbreviated TrueSkill model type checks fine and elaborates to the built-in one.

|  |  |  |  |
| --- | --- | --- | --- |
| Players[Player] |  |  |  |
| Skill | real | latent | Gaussian(25.0,100.0) |
| Rank | real | latent | infer.Gaussian.Mean(Skill) |
|  |  |  |  |
| Matches[Match] |  |  |  |
| Player1 | link(Players) | input |  |
| Player2 | link(Players) | input |  |
| Perf1 | real | latent | Gaussian(Player1.Skill,100.0) |
| Perf2 | real | latent | Gaussian(Player2.Skill,100.0) |
| Win1 | bool | output | Perf1 > Perf2 |

## Sampling

The **Sample** button can be used to sample from an (unconditioned) model and is useful tool for generating synthetic data on which to experiment with inference. Sampling is pseudo-random and depends on the value of the editable (random) **Seed.**

## Default, Typed and Core Models

The drop-down list used to select the built-in TrueSkill model has an option to construct a **Default Model** from the PowerPivot schema, a useful starting point for any subsequent modelling once the data tables and foreign key relations between them have been added to PowerPivot.

The drop-down list also has an option to display the **Typed Model** of the current model (with full standard function prelude and fully inferred spaces on type declarations) as well as the **Core Model** of the current model, in which all functions and other higher-level constructs have been in-lined or otherwise reduced to Core form (corresponding to a first-order graphical model). The models are produced in new sheets and are executable.

## CSV export

The results of inference can be saved to csv files by checking the **Save to csv** box. The folder containing thecsv files is accessible form a hyperlink output alongside the data after inference.

## Symmetry Breaking

Automatic (but sometimes undesirable) **Symmetry Breaking** may be disabled by **un**checking the Symmetry Breaking tick-box.

## Code Extraction

Models in Excel may also be extracted as Infer.NET modelling code (a C# program) by checking the **Extract Code** box. The extracted code is accessible from a hyperlink output alongside the data after inference.

### Producing Factor Graphs (requires third-party tool Graphviz).

Tabular is able to render models as factor graphs using the graphviz toolset.

For example, here’s the factor graph for TrueSkill:

To achieve this, Tabular must be able to locate the graphviz executable dot.exe.

Dot.exe is located by consulting the following places :

* The environment variable graphvizdot. This should define the full path to the program (e.g. “C:\dot.exe”)
* Any folders mentioned in your path
* In %ProgramFiles(x86)%\Graphviz2.34\bin\dot.exe

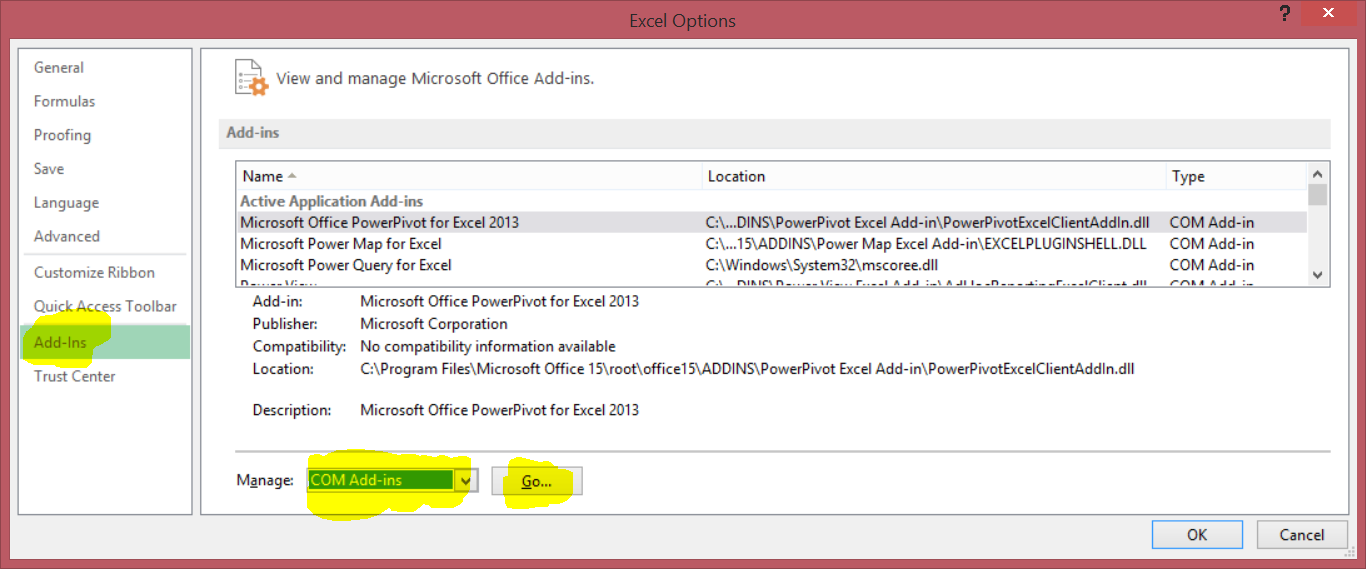
If dot.exe is available, the **Factor Graph** button is enabled to allow for the transformation of the Tabular model to a dot file which is rendered as the image of a factor graph in Excel.

### Additional Excel Functions (Deprecated)

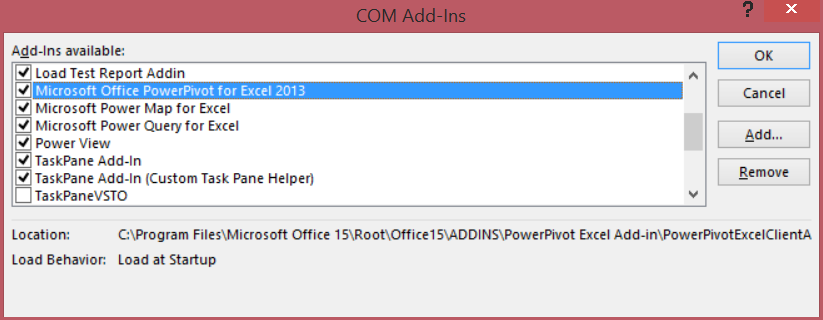
Tabular’s distributions are rendered as simple strings. The Tabular plugin also provides a small set of User Defined Functions (UDFs) that makes it possible to extract properties of the distribution strings (such as Mean(d) and Variance(d)). The UDFs can then be used in ordinary Excel formulas to calculate derived values for use in evaluation and charting. A small set of Tabular UDFs are declared under Excel function category “Tabular”. This functionality is largely subsumed by the availability of qry-space computations but may still be useful to some users.

## Troubleshooting

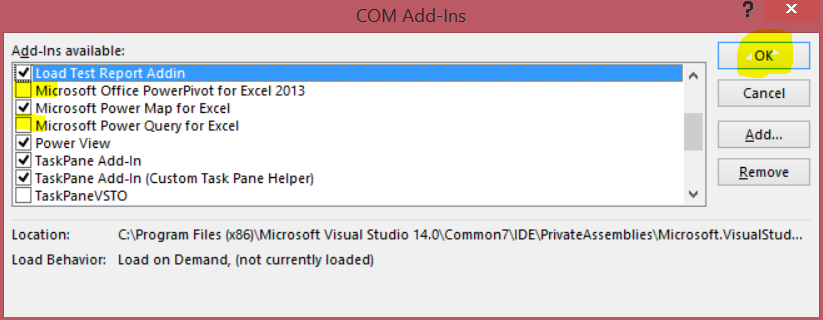
Though fairly stable now, Tabular does occasionally crash. If you find that the PowerPivot and or PowerQuery tabs have disappeared from the Excel Ribbon, do not despair. They can be restored by first disabling then re-enabling them via the Excel Options, COM Add-Ins dialog box:



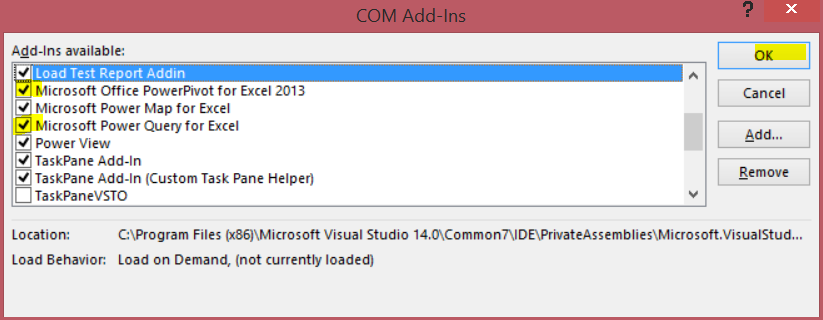
First clear the PowerPivot and Power Query check boxes and hit OK.



First clear the PowerPivot and Power Query check boxes and hit OK:



Now repeat and select the PowerPivot and Power Query check boxes and hit OK.



The Ribbon should now contain the previously missing PowerPivot and Power Query tabs.