

The Filter module allows data in an output file to be filtered according to a user specified expression. Only those rows that evaluate as true will be passed to the parent component.

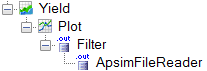
# Graph example:

Suppose we have an output file that contains information on harvests for the years 1940-1980, but we only want to display data between 1950 and 1960.

We already have a graph component in our simulation. When fully expanded it looks like this:



To use the Filter component, we drop it onto the Plot node then move the ApsimFileReader node onto the newly created Filter Node. The output file will now pass through the filter expression before being given to the Plot.



In the filter component we then type in:

Date >= #1/1/1950# AND Date < #1/1/1960#

Note the # symbol when evaluating dates.

# Filter Expression Examples

Some basic filter expressions are shown below. Expressions can be combined with Boolean AND, OR and NOT keywords. Arithmetic expressions are also allowed. Full documentation can be found below.

|  |  |
| --- | --- |
| To show: | Expression |
| Yields greater than 1000 t/ha | yield > 1000 |
| Yields greater than 1000 t/ha occurring after 1970 | yield > 1000 AND Date > #1/1/1970# |
| Yields that were less than 30% of the total biomass | yield / biomass < 0.3 |
| Yields greater than the average yield | yield > AVG(yield) |
| The maximum yield | yield = MAX(yield) |

Note that all expressions must evaluate to either true or false which is why the last example is

“yield = MAX(yield)” instead of just “MAX(yield)”.

# Filter Expression Documentation

The Filter component uses a subset of the C# DataColumn.Expression Property to evaluate expressions. The relevant subset of the MSDN documentation is reproduced below. The entire document can be found [here](http://msdn.microsoft.com/en-us/library/system.data.datacolumn.expression.aspx).

## Expression Syntax

When you create an expression, use the ColumnName property to refer to columns. For example, if the ColumnName for one column is "UnitPrice", and another "Quantity", the expression would be as follows:

"UnitPrice \* Quantity"

If a column is used in an expression, then the expression is said to have a dependency on that column. If a dependent column is renamed or removed, no exception is thrown. An exception will be thrown when the now-broken expression column is accessed.

When you create an expression for a filter, enclose strings with single quotation marks:

"LastName = 'Jones'"

If a column name contains any non-alphanumeric characters or starts with a digit or matches (case-insensitively) any of the following reserved words, it requires special handling, as described in the following paragraphs.

And Between Child

False In Is

Like Not Null

Or Parent True

If a column name satisfies one of the above conditions, it must be wrapped in either square brackets or the "`" (grave accent) quotes. For example, to use a column named "Column#" in an expression, you would write either "[Column#]":

Total \* [Column#]

or "`Column#`":

Total \* `Column#`

If the column name is enclosed in square brackets then any ']' and '\' characters (but not any other characters) in it must be escaped by prepending them with the backslash ("\") character. If the column name is enclosed in grave accent characters then it must not contain any grave accent characters in it. For example, a column named "Column[]\" would be written:

Total \* [Column[\]\\]

or

Total \* `Column[]\`

Date values should be enclosed within pound signs (#) or single quotes (') based on the data provider. Decimals and scientific notation are permissible for numeric values. For example:

FirstName = 'John'

Price <= 50.00

Birthdate < #1/31/82#

String literals are identified when there are single quotes surrounding the value. For example:

'John'

Boolean literals are true and false; they are not quoted in expressions.

## Operators

Concatenation is allowed using Boolean AND, OR, and NOT operators. You can use parentheses to group clauses and force precedence. The AND operator has precedence over other operators. For example:

(LastName = 'Smith' OR LastName = 'Jones') AND FirstName = 'John'

When you create comparison expressions, the following operators are allowed:

< >

<= >=

<> =

IN LIKE

The following arithmetic operators are also supported in expressions:

+ (addition) - (subtraction)

\* (multiplication) / (division)

% (modulus)

## Wildcard Characters

Both the \* and % can be used interchangeably for wildcard characters in a LIKE comparison. If the string in a LIKE clause contains a \* or %, those characters should be enclosed in brackets ([]). If a bracket is in the clause, each bracket character should be enclosed in brackets (for example [[] or []]). A wildcard is allowed at the start and end of a pattern, or at the end of a pattern, or at the start of a pattern. For example:

"ItemName LIKE '\*product\*'"

"ItemName LIKE '\*product'"

"ItemName LIKE 'product\*'"

Wildcard characters are not allowed in the middle of a string. For example, 'te\*xt' is not allowed.

## Aggregates

The following aggregate types are supported:

Sum (Sum) Avg (Average)

Min (Minimum) Max (Maximum)

Count (Count) StDev (Statistical standard deviation)

Var (Statistical variance)