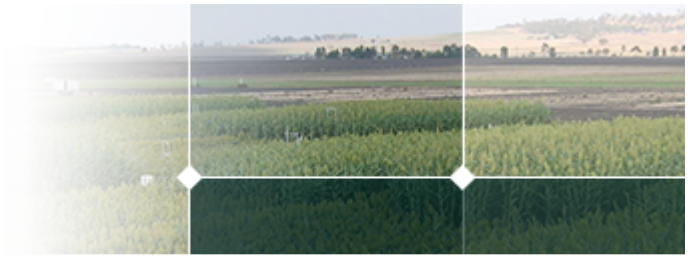


2020.3.25.4946

# 1 Report Model Documentation and Examples

*Dean Holzworth (CSIRO, Australia) and Val Snow (AgResearch, New Zealand)*

(Text last updated 24 October 2019)

**Report** is the model in APSIM that is used to generate columns of simulation outputs for further analysis.

The simulations in this example are to show examples of the different reporting capabilities and to provide a test that they continue to produce the correct results.

The intention is that this documentation is read alongside the simulation that produced it to see the detail of the examples. The simulation is to be found under the "Examples" button as Report.apsim.

# 2 Reporting Basics

Setting up outputs from a simulation requires describing **what** to output and **when** that should be done.

## 2.1 Properties - setting up what should be reported

Figure 1 shows a screen capture of a simple report. The upper box (Properties) holds descriptions of what should be reported. The general syntax is:

```
[Model].Variable
```

The first part, "[Model]" just above, gives the name of a model in the simulation. Models are almost any component in the simulation - e.g. Clock, Weather, Wheat, Irrigation - including Manager scripts (although these are a slightly special case and more on that below). The name of the model must be in square brackets. As with Manager scripts, Properties uses Intellisense to assist with constructing the output descriptors. For example, see row 1 of Figure 1. After typing "[Clock]" once a "." is typed a drop-down list of possible variables will appear as hints and in Figure 1 the choice from the list was "Today" which gives the date of the simulation output.

*Note that model and variable names are case-sensitive. Capitalisation in the right places is important.*

The component "ReportSimple" shows four output examples (see also below). The second row (see also below) shows an example of relabelling the output (the "as DairyRainfall" text) - this simply allows the user to give a more relevant label for the output in the output file.

```
        [Weather].Rain as DailyRainfall
```

The third row shows that it may be necessary to work through several layers to get to the output wanted. In each case a "." after the variable name gives a list with hints for outputs.

```
        [Wheat].Grain.Total.Wt
```

The fourth row (also below) shows the special case of reporting an output from a Manager script. The first part ("[SowingRule]") is the name of the Manager script (so the output text needs to say current with any name changes you might make). Outputs from Managers always need a ".Script" before Intellisense will show the possible outputs.

```
        [SowingRule].Script.SowingDepth
```

## 2.2 Reporting frequency - setting up when reports should happen

Also needed is a specification of *when* an output should be made. The instruction for when to report is in the form of:

```
        [Model].Event or Date(s)
```

### 2.2.1 Event

An Event is something that happens, like a stage of the day or a management action. The most-used example of a reporting frequency is:

```
        [Clock].EndOfDay
```

and this creates an output for every "EndOfDay" event - "EndOfDay" is an event created by Clock every day after all the models have done their calculations. Any (almost) event can be used to control the frequency of reporting such as:

```
[Wheat].Sowing
[Wheat].Flowering
[Wheat].Harvesting
[Wheat].PlantEnding
[Irrigation].Irrigated
[Fertiliser].Fertilised
```

where “Wheat” might be any crop. This also shows that multiple triggers for reporting can be used.

It is worth noting that when an event is used to trigger reporting, the output happens immediately rather than at the end of the simulation day. If you are getting strange outputs then consider this as a possible reason. To understand more about what order models do their calculations see https://apsimnextgeneration.netlify.com/development.

### 2.2.2 Dates

In addition to specifying events, you can also specify one or more dates in the frequency window. Dates can be specified one per line at be in either dd-mmm or dd-mmm-yyyy form e.g.

```
1-jul
1-jul-1980
```

Note that there can be a mixture of events and dates.

## 2.3 Reporting at irregular intervals or specific dates

If reporting is needed for particular days (e.g. to compare against measurements in an experiment) a combination of a Manager and a Report component will do the trick. An example of such a Manager (**ReportHelper**) and two Report components (**ReportOnSpecificDaysEveryYear** and **ReportOnSpecificDates**) are included in this example. Note that when using a Manager to control reporting, the Report frequency in the Report component should be left blank.

[Note that an Operations component could also be used to trigger irregular reporting dates.]

# 3 Dealing with outputs that have layers

Several of the outputs from APSIM are arrays - water content in the soil layers is a good example of this - and there are some features that make reporting arrays easier in APSIM.

To report all elements of an array, the syntax is the same as reporting a single variable. See for example in Row 2 of "ReportArrays the text

```
[Soil].SoilWater.SWmm
```

produces one column of output for each element (soil layer) in the array. When a particular layer is wanted then specify that in square brackets so the "[1]" is the top (closest to the soil surface) layer. For example,

```
[Soil].SoilWater.SWmm[1] as TopLayerWater_mm
```

More often some sort of aggregation is in the array is wanted and for this it is necessary to specify both which elements of the array are to be aggregated and what type of aggregation is wanted.

There are four options for specifying how the array elements should be aggregated - Sum, Mean, Min and Max - the meaning of these is self-evident. These are applied to the array as, for example Mean(x) where the x is the output to be averaged. Use round brackets, capitalise the aggregation type and no spaces.

For specifying how the aggregation is to work there are several options. Giving no layer information at all includes all of the array. A range is specified as, for example [3:6] for the third to sixth (inclusive, here meaning four layers) layers. Giving the colon but no numerical value means from the first (e.g. [:5]) or to the last (e.g. [2:]) elements. Report will not indicate the depth of the layers but the user can either get this information from the Soil input or can output the data using [Soil].Thickness which is an array giving the thickness of each layer in mm.

Some examples of possible array outputs to show the syntax are:

```
Sum([Soil].Thickness[1:3]) as DepthToBottomOfLayer3
Sum([Soil].SoilWater.SWmm) as TotalWaterStored
Sum([Soil].SoilWater.SWmm[1:2]) as WaterStoredTop2Layers
Sum([Soil].SoilWater.SWmm[1:3]) as WaterStoredTop3Layers
Sum([Soil].SoilWater.SWmm[4:]) as WaterStoredLayer4AndBelow
Sum([Soil].SoilWater.SWmm[:6]) as WaterStoredDowntoLayer6
Mean([Soil].Nutrient.NO3.ppm[1:4]) as MeanNO3ppmTop4Layers
Min([Soil].Nutrient.NO3.ppm[1:4]) as MinNO3ppmTop4Layers
Max([Soil].Nutrient.NO3.ppm[1:4]) as MaxNO3ppmTop4Layers
```

Simple, element-by-element, array operations can be included in the specification. For example

```
[Soil].Nutrient.Urea.kgha + [Soil].Nutrient.NH4.kgha + [Soil].Nutrient.NO3.kgha
```

will produce one column of data for each soil layer with the total of the amount of N as Urea, NH4 and NO3. Inserting

```
Sum([Soil].Nutrient.Urea.kgha[1:3] + [Soil].Nutrient.NH4.kgha[1:3] + [Soil].Nutrient.NO3.kgha[1:3]) as MineralN
```

will give a single column of data with the total of Urea-N, NH4-N and NO3-N dfrom the surface the the bottom of the third layer. These expressions can get more complex as in:

```
Sum([Soil].Nutrient.Urea.kgha[1:3] + [Soil].Nutrient.NH4.kgha[1:3] + [Soil].Nutrient.NO3.kgha[1:3]) * 1000 / Sum([Soil].Thick
```

which would give s single (somewhat nonsensical) output of the accumulation in mineral N with depth in the soil.

# 4 Reporting at Intervals Beyond Every Day

***Note well*** *– reporting at aggregations other than every day can become complex. It is recommended that the results be critically evaluated to ensure that the reporting specified is as intended.*

APSIM allows reporting at various intervals. Several of these methods have been described above. In addition to the usual [Clock].EndOfDay for daily reporting, [Clock] also has events of EndOfWeek, EndOfMonth, EndOfYear and EndOfSimulation. When the reporting is not every day then it is necessary to consider what the aggregation of the output should be. Some variables would usually be reported as their value on the day – many state variables (e.g. plant biomass) are like this. Others are almost always wanted to be summed over the interval since the last report – drainage and evaporation are good examples here.

APSIM provides several ways to construct the aggregations. Not all the aggregations will be sensible for all outputs and it is up to the user to ensure the sensibility of the instructions to Report. For example, it makes no physical sense to sum the biomass of a plant in a monthly aggregation. It would make sense to report the value at the end of the month or to report the increase (difference) in biomass from the start to the end and it might make sense to report an average biomass. These issues must be considered when constructing more complex Report specifications. The general syntax of aggregated reporting is:

```
AggregationType of [Model].Variable from Start to End as Label
```

The new elements here are *AggregationType*, *Start* and *End*.

## 4.1 AggregationType

AggregationType can be any of:

- Sum of
- Mean of
- Min of
- Max of
- First of
- Last of
- Diff of

Most of these are self-evident. First and Last are chronological values. Diff is the increase in the output variable over the reporting interval – if the variable decreases then it will have a negative value.

## 4.2 Start

The most useful form of *Start* is *[ReportName].DateOfLastOutput* where the first part is the name of the current Report component. *DateOfLastOutput* is pretty much as stated. Other *Start* constructs might be a general, e.g. *1-jan*, or specific, e.g. *1-jan-1982*, date. *Start* can also be an event, e.g. *[Wheat].sowing* or *[Clock].StartOfYear*.

When *Start* is a Clock or Report event the aggregation always starts at the start of the day. For events created by crop or other models the event can happen at any point of the day and it is not always clear without careful examination if the current-day calculations and updates will be included or excluded. This is user-beware.

## 4.3 End

*[Clock].Today* is the most useful *End* specification. This means that the end of the aggregation will be controlled by the Report frequency. *End* can also be an event, e.g. *[Wheat].Harvesting* or a general or specific date. As with *Start* and events, treat these with caution. Always consider the interaction between the aggregation interval and the report frequency. For example if the report frequency is *[Clock].EndOfMonth*:

```
Sum of [Weather].Rain from [Report].DateOfLastOutput to [Clock].Today
```

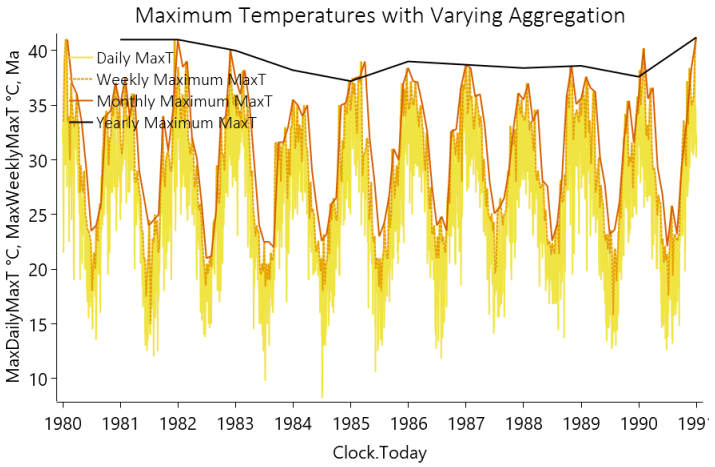
will produce a monthly rainfall total while

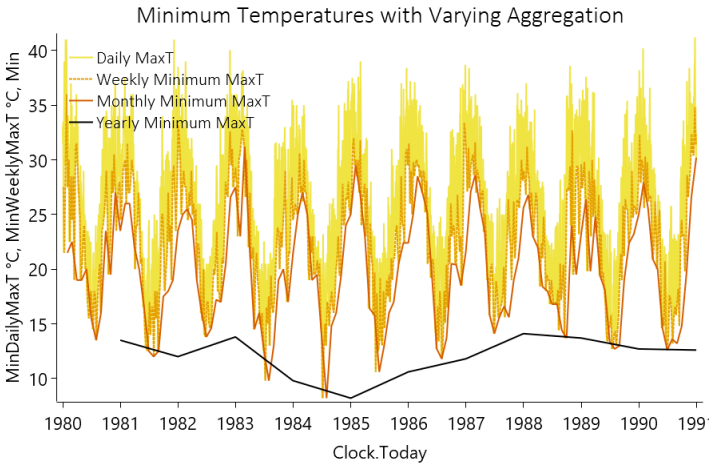
```
Sum of [Weather].Rain from 1-jan to [Clock].Today
```

will produce a cumulative rainfall as the months of the year progress and the accumulation will reset again on the next 1-Jan.

## 4.4 Some examples

The figures below show some aggregation examples of daily, weekly, monthly and annual aggregation reporting maximum temperature in various ways. Note that an aggregation to the end of the simulation is also possible. See the accompanying example *Report.apsimx* for details of these aggregations.



# 5 More Reporting Examples

## 5.1 Perennial Crop Example

The simulation in this section (titled "Annual Reporting In June") is a multi-year pasture cutting trial simulation. The Report components provide some examples of ways to get useful outputs from a simulation. See the generated documentation but also look at the output specifications in the Reports components.

Note that in the documentation, many of the output specifications are broken over two or more lines. This is only to show then text in the generated PDF. In a Report component, all the text would be on a single line.

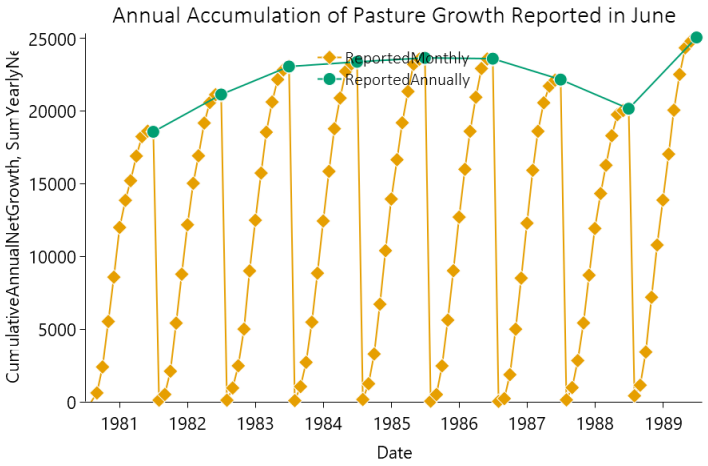
### 5.1.1 Getting Annual Patterns of Herbage Accumulation

When working with perennial crops that are harvested or grazed frequently, often an output that plots the accumulation of growth during a year is wanted. That output should be zeroed between years. If simulating a site in the Southern Hemisphere, usually the end of the year will be in June or July (winter). Here this is achieved as "CumulativeAnnualNetGrowth" using reporting at the end of the month (see in MonthlyReporting) and:

```
Sum of ([Ryegrass].NetGrowthWt + [WhiteClover].NetGrowthWt)
    from 1-Jul to [Clock].Today as CumulativeAnnualNetGrowth
```

with the output shown as the orange line below. Note that here it is necessary to calculate the output summing ryegrass and white clover. For comparison, the pattern of accumulation is plotted against the annual total (green line) and that was specified in AnnualReporting using:

```
Sum of ([Ryegrass].NetGrowthWt + [WhiteClover].NetGrowthWt)
    from [AnnualReporting].DayAfterLastOutput to [Clock].Today
    as SumYearlyNetGrowth
```



### 5.1.2 Working with Soil Carbon

When working with soil carbon (or organic nitrogen), gradual changes can hide systematic changes because the totals are so large and while the changes are small, they become important. In this example, using annual reporting only, there are three examples of reporting soil carbon.

The green line is the total soil carbon and is the most basic output. The specification is:

```
    Sum([Soil].SoilNitrogen.TotalC)/1000 as SoilCarbonToday_tonnesPerHa
```

where the "/1000" converts the standard output from kg /ha to tonnes /ha. The output is plotted against the right-hand axis and shows little change.

The orange line is the annual **change** in soil carbon (plotted against the left-hand axis) was created using:

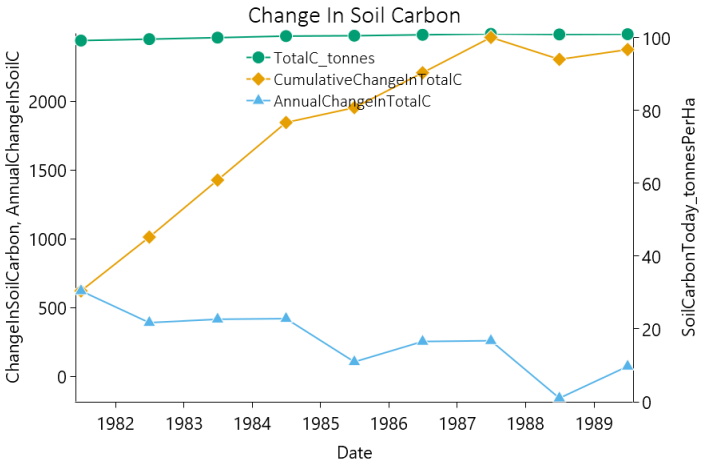
```
Diff of Sum([Soil].SoilNitrogen.TotalC) from [Clock].StartOfSimulation
           to [Clock].Today as ChangeInSoilCarbon
```

This clearly shows that soil carbon is increasing most years but that the increase is diminishing towards the end of the simulation. Note that for this simulation the line could also have been created using:

```
Diff of Sum([Soil].SoilNitrogen.TotalC)
    from 01-jul-1980 to [Clock].Today as ChangeInSoilCarbon
```

but the first version is more generic in that if the start date of the simulation is changed it will still be valid. The second version can be more useful where there is an initial spin-up period of several years before changes in soil carbon are of interest. Another form of soil carbon output that can be useful is the change within any year. That (the blue line) shows even more clearly the slow stabilisation of the total carbon and is specified using:

```
Diff of Sum([Soil].SoilNitrogen.TotalC)
    from 01-jul to [Clock].Today as AnnualChangeInSoilCarbon
```




### 5.1.3 Daily and Monthly Leaching

The example below shows daily, monthly, and cumulative annual leaching. The daily output is from "DailyReporting" and is simply:

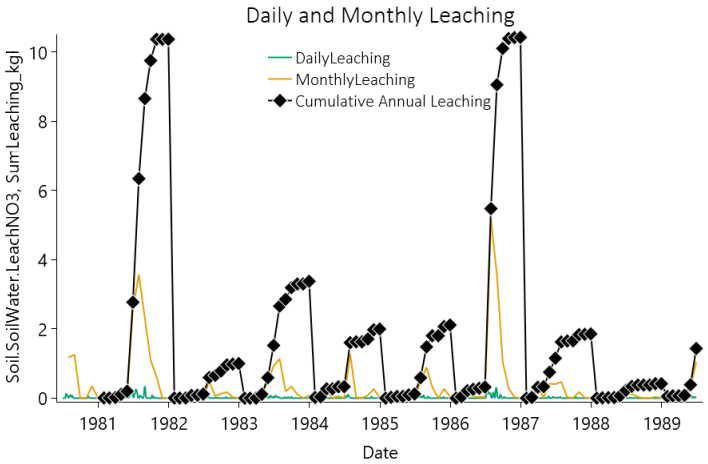
```
[Soil].SoilWater.LeachNO3
```

The other two outputs are both calculated in "MonthlyReporting" as:

```
Sum of [Soil].SoilWater.LeachNO3 from [MonthlyReporting].DayAfterLastOutput
    to [Clock].Today as SumLeaching_kgPerHa
```

and

```
Sum of [Soil].SoilWater.LeachNO3
    from 1-Jan to [Clock].Today as CumLeaching_kgPerHa
```




## 5.2 Annual Crop Example

In annual cropping simulations, users often want to know the values of outputs only during the period that the crop is in the ground with the outputs summarised over the interval between sowing and harvesting. The examples in this section show how to do this and also show some results that might be unexpected to look out for.

Note this example also shows and example of specifying dates in the report frequency e.g.

```
1-jan
1-feb
1-mar
2-jan-1980
```

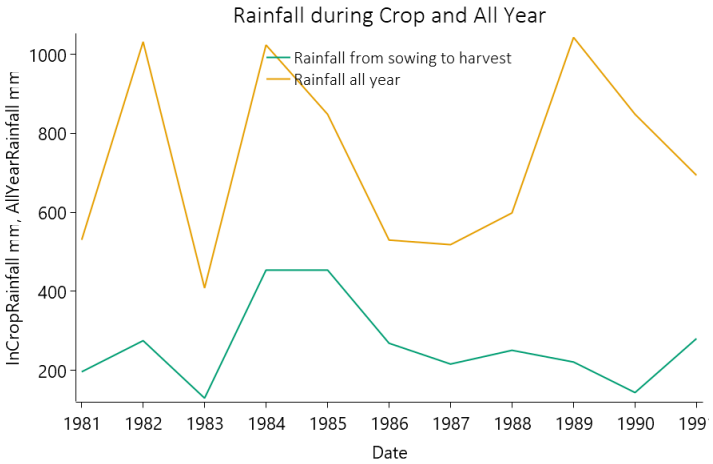
See ReportSpecificDates report model.

### 5.2.1 Comparing Rainfall during the Crop and All Year

The example below shows, for each calendar year, the amount of rain that fell when the crop was in the ground as compared to the annual rainfall. The Report frequency in "ReportInCropAnnually" is {Clock].EndOfYear and the output specifications are:

```
Sum of [Weather].Rain from [Wheat].Sowing
    to [Wheat].Harvesting as InCropRainfall

Sum of [Weather].Rain from 1-Jan to
    [Clock].Today as AllYearRainfall
```




### 5.2.2 Soil Water storage during the Cropping Phase

The next example also uses outputs from "ReportInCropAnnually" and shows how to get information about soil conditions during the crop. The output specification:

```
Mean of Sum([Soil].SoilWater.SWmm) from [Wheat].Sowing
    to [Wheat].Harvesting as InCropMeanSoilWater
```

gives the whole-soil profile soil water storage as a mean while the crop is in the ground. The output is reported at the end of the year but is for the period from sowing to harvesting only. If the interest is only in the water storage in the top three layers then the specification would be:

```
Mean of Sum([Soil].SoilWater.SWmm[:3]) from [Wheat].Sowing
    to [Wheat].Harvesting as InCropMeanSoilWaterTopThreeLayers
```

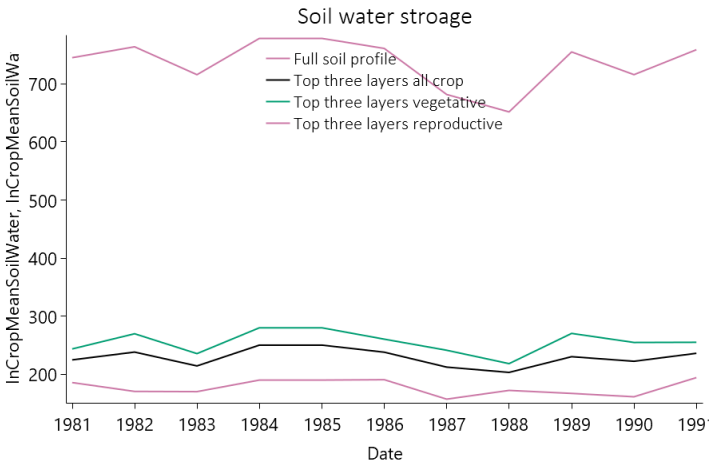
and that produces the black line in the figure below. If the interest was in the water storage during the vegetative stages compared to the reproductive stages then:

```
Mean of Sum([Soil].SoilWater.SWmm[:3]) from [Wheat].Sowing
    to [Wheat].Flowering as VegetativeMeanSoilWaterTopThreeLayers
```

and

```
Mean of Sum([Soil].SoilWater.SWmm[:3]) from [Wheat].Flowering
    to [Wheat].Harvesting as ReproductiveMeanSoilWaterTopThreeLayers
```

produce the green and red lines to show that there was less water storage during the reproductive stages. Of course other types of aggregation could be reported such as the difference durign the phases, min/max etc.




### 5.2.3 Reporting Yield and When Things Can Seem to Go Wrong

The Report model is a powerful and very useful component to get the information needed from the simulation - but there are some traps that should be noted.

The example below is primarily based on "ReportInCropAnnually" which has a reporting frequency of [Clock].EndOfYear. That Report uses an output specification of:

```
Max of ([Wheat].Grain.Wt * 10000 / 1000) from [Wheat].Sowing
    to [Wheat].Harvesting as FinalYield_kg_Ha
    // * 10000 / 1000 to convert from g/m2 to kg/ha
```

(note that the double slash, //, denotes a comment which is shown in green in the user interface). This produces the green line in the figure below. It is immediately noticeable that the yeild in late 1984 was exactly the same as the previous year - this should raise red flags. Alternative outputs that might be expected to give the same values are:

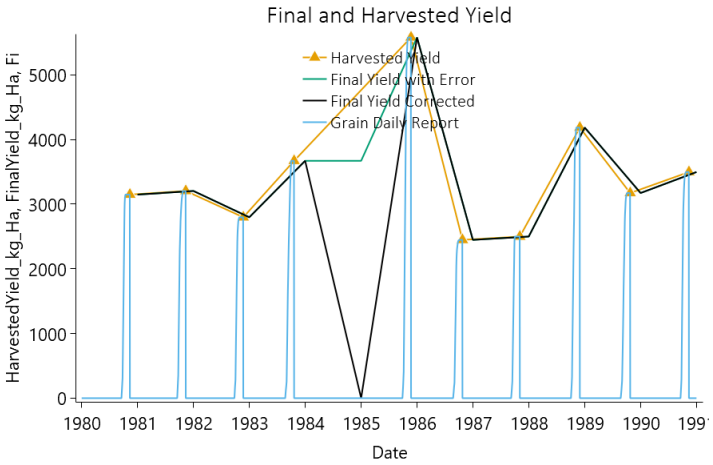
```
Max of ([Wheat].Grain.Wt * 10000 / 1000)
    from [ReportInCropAnnually].DayAfterLastOutput
    to [Wheat].Harvesting as FinalYield_kg_Ha_v1
```

also at a reporting frequency of [Clock].EndOfYear (the black line), or

```
([Wheat].Grain.Wt * 10000 / 1000) as HarvestedYield_kg_Ha
```

with a reporting frequency of [Wheat].Harvesting which produced the orange line (see ReportGrainOnHarvesting), or for more detail report the same output on a daily basis (see ReportGrainDaily) which is the blue line.

The additional reporting clearly shows that there was no grain yield in 1984 and further investigation (the summary file) shows that there was no crop sown because the sowing conditions were not met. The apparent yield in the green line is not an error in Report but is a result of the specification in combination with the possibility that there will not be a crop every year. When Report does its work at the end of 1984 it outputs the maximum grain yield from the last sowing event to the last harvesting event from the Wheat model - and these were the ones from 1983. This has also affected the green line in Section 5.2.1 and all the lines in 5.2.2 but they were not so immediately obvious. The salient message is to consider the output specifications within the context of the management conditions in the simulation.




# 6 Grouping

Report has the ability to produce temporally aggregated variables grouped by another variable. The example in this section has two contrasting reports that summerise variables seaasonally in 2 different ways.

1. The first report (SeasonalByYear) has multiple reporting frequencies, one for each end of season. This produces a year x season output table.
2. The second report (SeasonalOverall) temporally aggregates from start to end of simulation and has a reporting frequency of end of simulation. It also has a 'Group By' of [Weather].Season which, at the end of the simulation, will split each variable into values for each season resulting in multpile rows of output, one for each season (4 rows of output).

To use the 'Group By' capability, variables need to be aggregated like this:

```
from [Clock].StartOfSimulation to [Clock].EndOfSimulation
```

The reporting frequency must be:

```
[Clock].EndOfSimulation
```

The 'Group By' variable can be any APSIM variable e.g.

```
[Clock].Today.Month
[Weather].Season
```

## 6.1 Clock

The clock model is resonsible for controlling the daily timestep in APSIM. It keeps track of the simulation date and loops from the start date to the end date, publishing events that other models can subscribe to.

### 6.1.1 Pre-timestep events (in order)

| Events |
|---|
| DoInitialSummary |
| StartOfSimulation |
| CLEMInitialiseResource |
| CLEMInitialiseActivity |
| CLEMValidate |
| FinalInitialise |
| StartOfMonth |
| StartOfYear |

### 6.1.2 Timestep events (in order)

| Events |
|---|
| DoWeather |
| DoDailyInitialisation |
| StartOfDay |
| DoManagement |
| DoPestDiseaseDamage |

| Events |
| --- |
| DoEnergyArbitration |
| DoSoilWaterMovement |
| DoSoilTemperature |
| DoSoilOrganicMatter |
| DoSurfaceOrganicMatterDecomposition |
| DoUpdateWaterDemand |
| DoWaterArbitration |
| PrePhenology |
| DoPhenology |
| DoPotentialPlantGrowth |
| DoPotentialPlantPartioning |
| DoNutrientArbitration |
| DoActualPlantPartioning |
| DoActualPlantGrowth |
| DoStock |
| DoLifecycle |
| DoUpdate |
| DoManagementCalculations |
| DoReportCalculations |
| EndOfDay |
| DoReport |

### 6.1.3 Post-timestep events (in order)

| Events |
| --- |
| EndOfSimulation |

Reads in weather data and makes it available to other models.

The manager model

This model collects the simulation initial conditions and stores into the DataStore. It also provides an API for writing messages to the DataStore.

The APSIM farming systems model has a long history of use for simulating mixed or intercropped systems. Doing this requires methods for simulating the competition of above and below ground resources. Above ground competition for light has been calculated within APSIM assuming a mixed turbid medium using the Beer-Lambert analogue as described by Keating et al., 1993. The MicroClimate Snow et al., 2004 model now used within APSIM builds upon this by also calculating the impact of mutual shading on canopy conductance and partitions aerodynamic conductance to individual species in applying the Penman-Monteith model for calculating potential crop water use. The arbitration of below ground resources of water and nitrogen is calculated by this model.

Traditionally, below ground competition has been arbitrated using two approaches. Firstly, the early approaches Adiku et al., 1995; Carberry et al., 1996 used an alternating order of uptake calculation each day to ensure that different crops within a simulation did not benefit from precedence in daily orders of calculations. Soil water simulations using the SWIM3 model Huth et al., 2012 arbitrate individual crop uptakes as part of the simulataneous solutions of various soil water fluxes as part of its solution of the Richards' equation Richards, 1931.

The soil arbitrator operates via a simple integration of daily fluxes into crop root systems via a Runge-Kutta calculation.

If Y is any soil resource, such as water or N, and U is the uptake of that resource by one or more plant root systems, then

$Y_{t+1} = Y_t - U$

Because U will change through the time period in complex manners depending on the number and nature of demands for that resource, we use Runge-Kutta to integrate through that time period using

$Y_{t+1} = Y_t + 1/6 \times (U_1 + 2xU_2 + 2xU_3 + U_4)$

Where $U_1$,$U_2$,$U_3$ and $U_4$ are 4 estimates of the Uptake rates calculated by the crop models given a range of soil resource conditions, as follows:

$U_1 = f(Y_t)$,

$U_2 = f(Y_t - 0.5xU_1)$,

$U_3 = f(Y_t - 0.5xU_2)$,

$U_4 = f(Y_t - U_3)$.

So $U_1$ is the estimate based on the uptake rates at the beginning of the time interval, similar to a simple Euler method. $U_2$ and $U_3$ are estimates based on the rates somewhere near the midpoint of the time interval. $U_4$ is the estimate based on the rates toward the end of the time interval.

The iterative procedure allows crops to influence the uptake of other crops via various feedback mechanisms. For example, crops rapidly extracting water from near the surface will dry the soil in those layers, which will force deeper rooted crops to potentially extract water from lower layers. Uptakes can notionally be of either sign, and so trees providing hydraulic lift of water from water tables could potentially make this water available for uptake by mutplie understory species within the timestep. Crops are responsible for meeting resource demand by whatever means they prefer. And so, leguminous crops may start by taking up mineral N at the start of the day but rely on fixation later in a time period if N becomes limiting. This will reduce competition from others and change the balance dynamically throughout the integration period.

The design has been chosen to provide the following benefits:

1) The approach is numerically simple and pure.

2) The approach does not require the use of any particular uptake equation. The uptake equation is embodied within the crop model as designed by the crop model developer and tester.

3) The approach will allow any number of plant species to interact.

4) The approach will allow for arbitration between species in any zone, but also competition between species that may demand resources from multiple zones within the simulation.

5) The approach will automatically arbitrate supply of N between zones, layers, and types (nitrate vs ammonium) with the preferences of all derived by the plant model code.

## 6.2 MicroClimate

The module MICROMET, described here, has been developed to allow the calculation of potential transpiration for multiple competing canopies that can be either layered or intermingled.

The manager model

This class encapsulates an operations schedule.

This model controls irrigation events, which can be triggered using the Apply() method.

The fertiliser model

### 6.2.1 SurfaceOrganicMatter

The surface organic matter model.

Encapsulates a list of residue types for SurfaceOrganicMatter model

The soil class encapsulates a soil characterisation and 0 or more soil samples. the methods in this class that return double[] always return using the "Standard layer structure" i.e. the layer structure as defined by the Water child object. method. Mapping will occur to achieve this if necessary. To obtain the "raw", unmapped, values use the child classes e.g. SoilWater, Analysis and Sample.

A model for capturing physical soil parameters

A soil crop parameterization class.

A soil crop parameterization class.

The SoilWater module is a cascading water balance model that owes much to its precursors in CERES (Jones and Kiniry, 1986) and PERFECT(Littleboy et al, 1992). The algorithms for redistribution of water throughout the soil profile have been inherited from the CERES family of models.

The water characteristics of the soil are specified in terms of the lower limit (ll15), drained upper limit(dul) and saturated(sat) volumetric water contents. Water movement is described using separate algorithms for saturated or unsaturated flow. It is notable that redistribution of solutes, such as nitrate- and urea-N, is carried out in this module.

Modifications adopted from PERFECT include: * the effects of surface residues and crop cover on modifying runoff and reducing potential soil evaporation, * small rainfall events are lost as first stage evaporation rather than by the slower process of second stage evaporation, and * specification of the second stage evaporation coefficient(cona) as an input parameter, providing more flexibility for describing differences in long term soil drying due to soil texture and environmental effects.

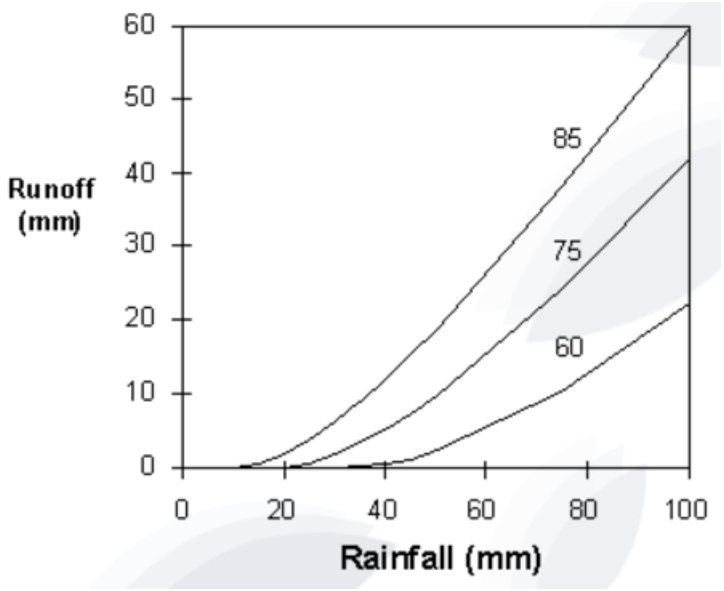
The module is interfaced with SurfaceOrganicMatter and crop modules so that simulation of the soil water balance responds to change in the status of surface residues and crop cover(via tillage, decomposition and crop growth).

Enhancements beyond CERES and PERFECT include: * the specification of swcon for each layer, being the proportion of soil water above dul that drains in one day * isolation from the code of the coefficients determining diffusivity as a function of soil water (used in calculating unsaturated flow).Choice of diffusivity coefficients more appropriate for soil type have been found to improve model performance. * unsaturated flow is permitted to move water between adjacent soil layers until some nominated gradient in soil water content is achieved, thereby accounting for the effect of gravity on the fully drained soil water profile.

SoilWater is called by APSIM on a daily basis, and typical of such models, the various processes are calculated consecutively. This contrasts with models such as SWIM that solve simultaneously a set of differential equations that describe the flow processes.

Runoff from rainfall is calculated using the USDA-Soil Conservation Service procedure known as the curve number technique. The procedure uses total precipitation from one or more storms occurring on a given day to estimate runoff. The relation excludes duration of rainfall as an explicit variable, and so rainfall intensity is ignored. When irrigation is applied it can optionally be included in the runoff calculation. This flag (willRunoff) can be set when applying irrigation.

Figure: Runoff response curves (ie runoff as a function of total daily rainfall) are specified by numbers from 0 (no runoff) to 100 (all runoff). Response curves for three runoff curve numbers for rainfall varying between 0 and 100 mm per day.




The user supplies a curve number for average antecedent rainfall conditions (CN2Bare). From this value the wet (high runoff potential) response curve and the dry (low runoff potential) response curve are calculated. The SoilWater module will then use the family of curves between these two extremes for calculation of runoff depending on the daily moisture status of the soil. The effect of soil moisture on runoff is confined to the effective hydraulic depth as specified in the module's ini file and is calculated to give extra weighting to layers closer to the soil surface. Figure: Runoff response curves (ie runoff as a function of total daily rainfall) are specified by numbers from 0 (no runoff) to 100 (all runoff).

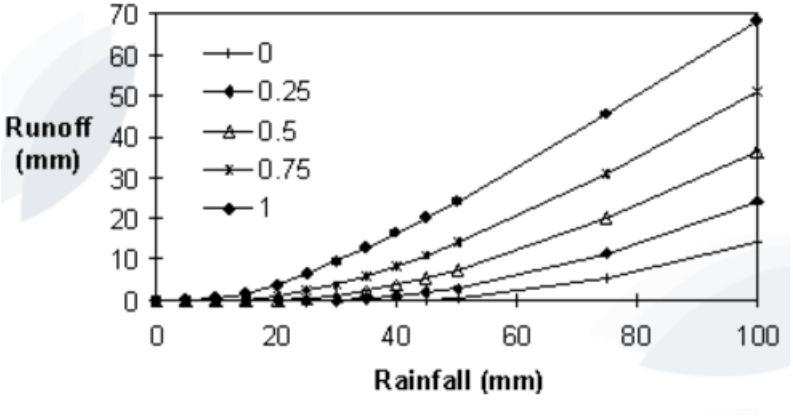
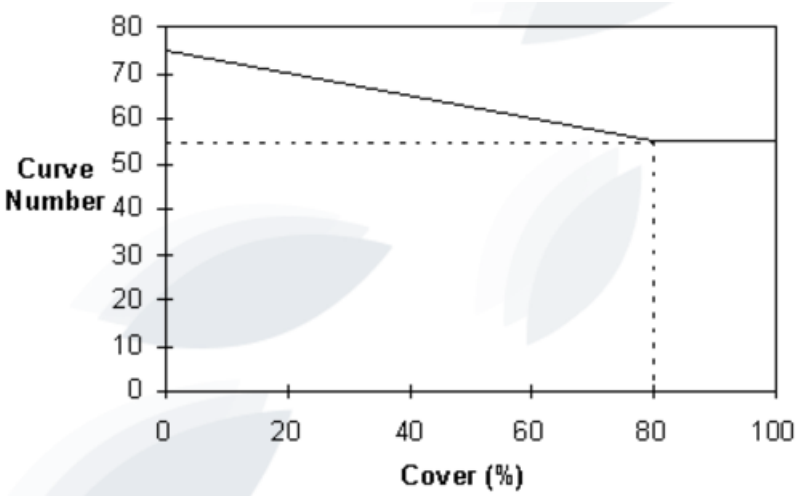



Figure: Residue cover effect on runoff curve number where bare soil curve number is 75 and total reduction in curve number is 20 at 80% cover.




Surface residues inhibit the transport of water across the soil surface during runoff events and so different families of response curves are used according to the amount of crop and residue cover.The extent of the effect on runoff is specified by a threshold surface cover (CNCov), above which there is no effect, and the corresponding curve number reduction (CNRed).

Tillage of the soil surface also reduces runoff potential, and a similar modification of Curve Number is used to represent this process. A tillage event is directed to the module, specifying cn_red, the CN reduction, and cn_rain, the rainfall amount required to remove the tillage roughness. CN2 is immediately reduced and increases linearly with cumulative rain, ie.roughness is smoothed out by rain.

Implements the curve number reduction caused by cover.

Implements the curve number reduction caused by tillage. Mark Littleboy's tillage effect on runoff (used in PERFECT v2.0) Littleboy, Cogle, Smith, Yule and Rao(1996). Soil management and production of alfisols in the SAT's I. Modelling the effects of soil management on runoff and erosion.Aust.J.Soil Res. 34: 91-102.

Soil evaporation is assumed to take place in two stages: the constant and the falling rate stages.

In the first stage the soil is sufficiently wet for water to be transported to the surface at a rate at least equal to the potential evaporation rate. Potential evapotranspiration is calculated using an equilibrium evaporation concept as modified by Priestly and Taylor(1972).

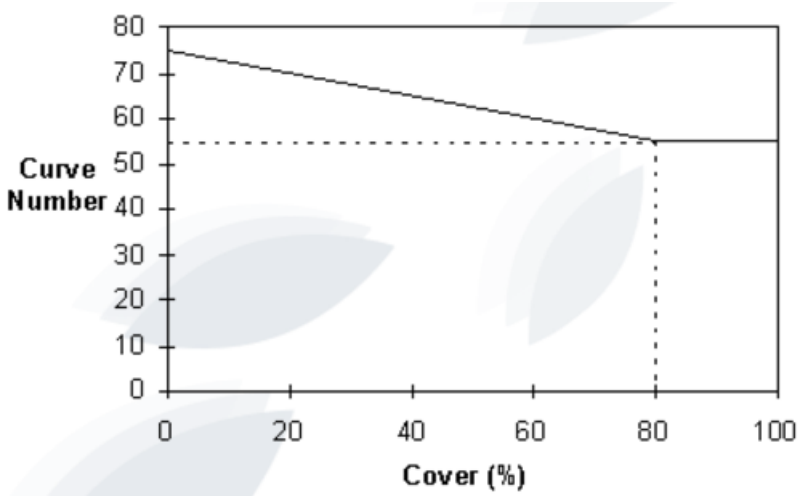
Once the water content of the soil has decreased below a threshold value the rate of supply from the soil will be less than potential evaporation (second stage evaporation). These behaviors are described in SoilWater through the use of two parameters: U and CONA.

The parameter U (as from CERES) represents the amount of cumulative evaporation before soil supply decreases below atmospheric demand. The rate of soil evaporation during the second stage is specified as a function of time since the end of first stage evaporation. The parameter CONA (from PERFECT) specifies the change in cumulative second stage evaporation against the square root of time.

i.e. $Es = CONA\ t^{1/2}$

Water lost by evaporation is removed from the surface layer of the soil profile thus this layer can dry below the wilting point or lower limit (LL) to a specified air-dry water content (air_dry).

Figure: Cumulative Soil Evaporation through time for U = 6 mm and CONA = 3.5.




For $t <= t_1$ Es = Eos For $t > t_1$ Es = U x t + CONA x Sqrt($t-t_1$)

Lateral movement of water is calculated from a user specified lateral inflow ('InFlow').

Lateral Outflow is the flow that occurs as a result of the soil water going above DUL and the soil being on a slope. So if there is no slope and the water goes above DUL there is no lateral outflow. KLAT is just the lateral resistance of the soil to this flow. It is a soil water conductivity.

The calculation of lateral outflow on a layer basis is now performed using the equation: Lateral flow for a layer = KLAT * d * s / $(1 + s^2)^{0.5}$ * L / A * unit conversions. Where: KLAT = lateral conductivity (mm/day) d = depth of saturation in the layer(mm) = Thickness * (SW - DUL) / (SAT - DUL) if SW > DUL. (Note this allows lateral flow in any "saturated" layer, not just those inside a water table.) s = slope(m / m) L = catchment discharge width. Basically, it's the width of the downslope boundary of the catchment. (m) A = catchment area. ($m^2$)

NB. with Lateral Inflow it is assumed that ALL the water goes straight into the layer. Irrespective of the layers ability to hold it. It is like an irrigation. KLAT has no effect and does not alter the amount of water coming into the layer. KLAT only alters the amount of water flowing out of the layer

When water content in any layer is below SAT but above DUL, a fraction of the water drains to the next deepest layer each day.

Flux = SWCON x (SW - DUL)

Infiltration or water movement into any layer that exceeds the saturation capacity of the layer automatically cascades to the next layer.

For water contents below DUL, movement depends upon the water content gradient between adjacent layers and the diffusivity, which is a function of the average water contents of the two layers.

Unsaturated flow may occur both towards the surface and downwards, but cannot move water out of the bottom of the deepest layer in the profile. Flow between adjacent layers ceases at a soil water gradient (gravity_gradient) specified in the SoilWater ini file.

The diffusivity is defined by two parameters set by the user (diffus_const, diffus_slope) in the SoilWater parameter set (Default values, from CERES, are 88 and 35.4, but 40 and 16 have been found to be more appropriate for describing water movement in cracking clay soils).

Diffusivity = diffus_const x exp(diffus_slope x thet_av)

where thet_av is the average of SW - LL15 across the two layers. Flow = Diffusivity x Volumetric Soil Water Gradient

Water table is the depth (in mm) below the ground surface of the first layer which is above saturation.

Computes the soil C and N processes

This class encapsulates a SoilNitrogen model NO3 solute.

This class encapsulates a SoilNitrogen model NH4 solute.

This class encapsulates a SoilNitrogen model urea solute.

This class encapsulates a SoilNitrogen model 'PlantAvailableNO3' solute.

This class encapsulates a SoilNitrogen model NH4 solute.

A model for capturing soil organic parameters

This class captures chemical soil data

Represents the simulation initial water status. There are multiple ways of specifying the starting water; 1) by a fraction of a full profile, 2) by depth of wet soil or 3) a single value of plant available water.

The class represents a soil sample.

Calculates the average soil temperature at the centre of each layer, based on the soil temperature model of EPIC (Williams et al 1984) This code was separated from old SoilN - tidied up but not updated (RCichota, sep/2012)

**6.2.2 AGPRyegrass**

Describes a pasture species.

Describes a generic above ground organ of a pasture species.

Describes a generic tissue of a pasture species.

Describes a generic tissue of a pasture species.

Describes a generic tissue of a pasture species.

Describes a generic tissue of a pasture species.

Describes a generic above ground organ of a pasture species.

Describes a generic tissue of a pasture species.

Describes a generic tissue of a pasture species.

Describes a generic tissue of a pasture species.

Describes a generic tissue of a pasture species.

Describes a generic above ground organ of a pasture species.

Describes a generic tissue of a pasture species.

Describes a generic tissue of a pasture species.

Describes a generic tissue of a pasture species.

Describes a generic tissue of a pasture species.

#### 6.2.3 AGPWhiteClover

Describes a pasture species.

Describes a generic above ground organ of a pasture species.

Describes a generic tissue of a pasture species.

Describes a generic tissue of a pasture species.

Describes a generic tissue of a pasture species.

Describes a generic tissue of a pasture species.

Describes a generic above ground organ of a pasture species.

Describes a generic tissue of a pasture species.

Describes a generic tissue of a pasture species.

Describes a generic tissue of a pasture species.

Describes a generic tissue of a pasture species.

Describes a generic above ground organ of a pasture species.

Describes a generic tissue of a pasture species.

Describes a generic tissue of a pasture species.

Describes a generic tissue of a pasture species.

Describes a generic tissue of a pasture species.

A report class for writing output to the data store.

A report class for writing output to the data store.

# 7 References


Adiku, S. G. K., Carberry, P. S., Rose, C. W., McCown, R. L., Braddock, R., 1995. A maize (zea-mays) - cowpea (vigna-unguiculata) intercrop model. Ecophysiology of Tropical Intercropping, Inst Natl Recherche Agronomique, Paris, 397-406.

Carberry, P.S., McCown, R. L., Muchow, R. C., Dimes, J. P., Probert, M. E., 1996. Simulation of a legume ley farming system in northern Australia using the Agricultural Production Systems Simulator. Australian Journal of Experimental Agriculture 36 (8), 1037-1048.

Huth, N.I., Bristow, K.L., Verburg, K., 2012. SWIM3: Model use, calibration, and validation. Transactions of the ASABE 55 (4), 1303-1313.

Keating, B. A., Carberry, P. S., 1993. Resource capture and use in inter cropping - solar radiation. Field Crops Research 34 (3-4), 273-301.

Richards, Lorenzo Adolph, 1931. Capillary conduction of liquids through porous mediums. Journal of Applied Physics 1 (5), 318-333.

Snow, V. O., Huth, N. I., 2004. The APSIM MICROMET Module. HortResearch Internal Report, HortResearch, Auckland.