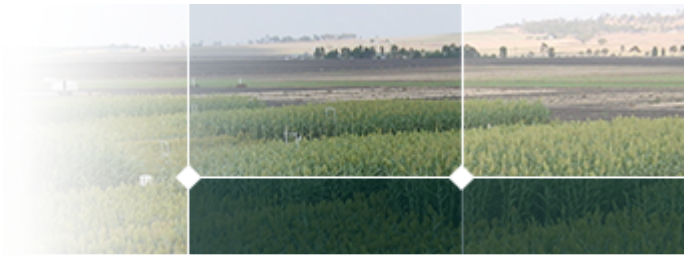


0.0.0.0

1 The APSIM Grapevine Model

Junqi Zhu^1, Amber Parker^2, Fang Gou^3, Rob Agnew^1, Linlin Yang^1, Marc Greven^4, Victoria Raw^1, Sue Neal1, Damian Martin^1, Mike C.T. Trought^1, Hamish Edward Brown^5

1. The New Zealand Institute for Plant & Food Research Limited (PFR) Marlborough Research Centre, PO Box 845 Blenheim 7240, New Zealand
2. Department of Wine, Food and Molecular Biosciences, Faculty of Agriculture and Life Sciences, Lincoln University, PO Box 85084, Lincoln 7647, Christchurch, New Zealand
3. Bragato Research Institute, PO Box 845, Blenheim 7240
4. Bordeaux Sciences Agro, 1, cours du Général de Gaulle, CS 40201 33175 Gradignan Cedex, France
5. The New Zealand Institute for Plant and Food Research Limited Lincoln Research Centre, Private Bag 4704, Christchurch 8140, New Zealand

Correspondance: junqi.zhu@plantandfood.co.nz;Hamish.brown@plantandfood.co.nz

1.1 Introduction

Grapevine is one of the most economically important fruit crops worldwide (Alston and Sambucci, 2019), and its use in wine production has played an important cultural role in many parts of the world. Grapevines are now cultivated in more than 90 countries for wine, distilled liquors, juice, table grapes, and raisin production (Fereres and FAO OIV, 2016). Because of its global economic importance, the climate diversity of the producing regions, and a large amount of studies (from genomics to production practices), grapevine has emerged as a model perennial fruit crop species.

1.2 Objective

The model presented here was built to simulate the phenology,light interception, carbon allocation and storage, and yield formation of grapevine (Vitis vinifera) as a model perennial crop.

1.3 Model construction

The APSIM Grapevine model was developed using the Plant Modelling Framework (PMF) of Brown et al., 2014. This new framework provides a library of plant organ and process submodels that can be coupled, at runtime, to construct a model in much the same way that models can be coupled to construct a simulation. This means that dynamic composition of lower level process and organ classes (e.g. photosynthesis, leaf) into larger constructions (e.g. maize, wheat, sorghum) can be achieved by the model developer without additional coding.

A series of development was taken to adapt the source code to simulate the perennial woody plants. The current model including the following features:

• A row strip configuration to represent the vineyard setup.

• An adopted strip light interception model to capture the canopy heterogeneous and to calculate the light interception by grapevine and inter-row cover crops

• A new carbon allocation method to allocate the carbohydrate simultaneously based on sink strength as well as sink priority. The new method capture the dynamics in carbon storage in perennial organ as well.

• An adapted phenology model which starts from endodormancy of the bud instead of emergence and reset to endodormancy by a management method (pruning) in winter to capture the perennial behaviour

• Using bud number per vine as input in the “sowing” method (model initialization) and calculating the number of primary shoots per vine based on fraction of budburst as a function of bud number

• An adopted strip water competition method to calculate water competition between grapevine and inter-row cover crops

The carbon assimilation and carbon allocation needs some further fine calibrations. so the effects of carbon stress has not been added to the yield formation processes, e.g. bunch number and berry number determination. Bunch number, berry number and potential berry fresh weight were determined by weather conditions at critical periods around flowerings of the previous and current season, see details at Zhu et al., 2020 OENO one. Berry dry mass accumulation following the source-sink carbon allocation rules. For getting a reliable estimation of yield and phenology, simulations should start one season before the season in question.

2 APSIM Description

The Agricultural Production Systems sIMulator (APSIM) is a farming systems modelling framework that is being actively developed by the APSIM Initiative.

It is comprised of

1. a set of biophysical models that capture the science and management of the system being modelled,
2. a software framework that allows these models to be coupled together to facilitate data exchange between the models,
3. a set of input models that capture soil characteristics, climate variables, genotype information, field management etc,
4. a community of developers and users who work together, to share ideas, data and source code,
5. a data platform to enable this sharing and
6. a user interface to make it accessible to a broad range of users.

The literature contains numerous papers outlining the many uses of APSIM applied to diverse problem domains. In particular, Holzworth et al., 2014; Keating et al., 2003; McCown et al., 1996; McCown et al., 1995 have described earlier versions of APSIM in detail, outlining the key APSIM crop and soil process models and presented some examples of the capabilities of APSIM.

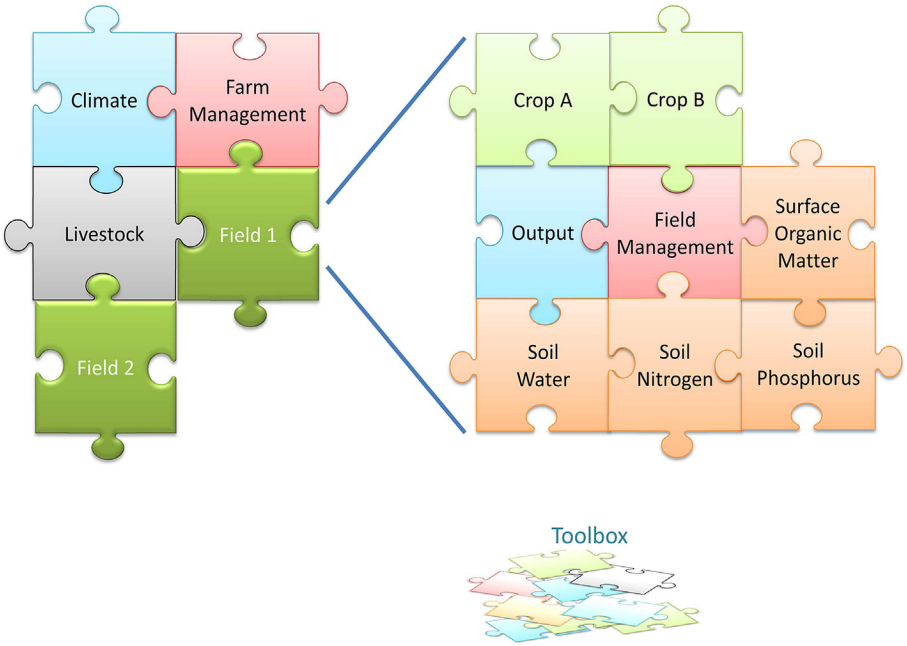


Figure 2: This conceptual representation of an APSIM simulation shows a “top level” farm (with climate, farm management and livestock) and two fields. The farm and each field are built from a combination of models found in the toolbox. The APSIM infrastructure connects all selected model pieces together to form a coherent simulation.*

The APSIM Initiative has begun developing a next generation of APSIM (APSIM Next Generation) that is written from scratch and designed to run natively on Windows, LINUX and MAC OSX. The new framework incorporates the best of the APSIM 7.x framework with an improved supporting framework. The Plant Modelling Framework (a generic collection of plant building blocks) was ported from the existing APSIM to bring a rapid development pathway for plant models. The user interface paradigm has been kept the same as the existing APSIM version,

but completely rewritten to support new application domains and the newer Plant Modelling Framework. The ability to describe experiments has been added which can also be used for rapidly building factorials of simulations. The ability to write C# scripts to control farm and paddock management has been retained. Finally, all simulation outputs are written to an SQLite database to make it easier and quicker to query, filter and graph outputs.

The model described in this documentation is for APSIM Next Generation.

APSIM is freely available for non-commercial purposes. Non-commercial use of APSIM means public-good research & development and educational activities. It includes the support of policy development and/or implementation by, or on behalf of, government bodies and industry-good work where the research outcomes are to be made publicly available. For more information visit the licensing page on the APSIM web site

3 Model description

The Grapevine model is constructed from the following list of software components. Details of the implementation and model parameterisation are provided in the following sections.

List of Plant Model Components.

Component Name	Component Type
Phenology	Models.PMF.Phen.Phenology
Structure	Models.PMF.Struct.Structure
Leaf	Models.PMF.Organs.Leaf
Shoot	Models.PMF.Organs.GenericOrgan
Cordon	Models.PMF.Organs.GenericOrgan
Trunk	Models.PMF.Organs.GenericOrgan
StructuralRoot	Models.PMF.Organs.GenericOrgan
Berry	Models.PMF.Organs.ReproductiveOrgan
Root	Models.PMF.Organs.Root
MortalityRate	Models.Functions.Constant
Arbitrator	Models.PMF.OrganArbitrator

3.1 Phenology

Grapevine's phenological development is simulated as the progression through a series of developmental phases, each bound by distinct growth *stages*.

3.1.1 ThermalTime

ThermalTime is the Average of sub-daily values from a Models.Functions.WangEngelTempFunction.

Firstly hourly estimates of air temperature (Ta) are interpolated from Tmax, Tmin and daylength (d) usig the method of Goudriaan et al., 1994. During sunlight hours Ta is calculated each hour using a sinusoidal curve fitted to Tmin and Tmax . After sunset Ta is calculated as an exponential decline from Ta at sunset to the Tmin at sunrise the next day. The hour (Th) of sunrise is calculated as Th = 12 - d/2 and Ta is assumed to equal Tmin at this time. Tmax is reached when Th equals 13.5.

Each of the interpolated air temperatures are then passed into the following Response and the Average taken to give daily ThermalTime

Response is calculated using a Wang and Engel beta function which has a value of zero below 3.9 increasing to a maximum value at 29 and decreasing to zero again at 37.9 Wang et al., 1998.

3.2 Phases

List of stages and phases used in the simulation of crop phenological development

Phase Number	Phase Name	Initial Stage	Final Stage
1	EndoDormancy	Endodormancy	Ecodormancy
2	Budding	Ecodormancy	BudBurst
3	Flowering	BudBurst	Flowering
4	FruitSet	Flowering	FruitSet
5	BerryDevelopment	FruitSet	Veraison
6	CanopySenescence	Veraison	LeafFall
7	Dormancy	LeafFall	EndoDormancy

3.2.1 EndoDormancy Phase

The *EndoDormancy* phase goes from the *Endodormancy* stage to the *Ecodormancy* stage.

The *Target* for completion is calculated as

Target = BaseTarget - [Phenology].AccChillBefPrune

Where:

BaseTarget = 52.6 (oD)

AccChillBefPrune = [Phenology].AccChillBefPrune

Progression through the *EndoDormancy* phase is calculated daily and accumulated until the *Target* is reached.

Progression is the Average of sub-daily values from a Models.Functions.SigmoidFunction.

Firstly hourly estimates of air temperature (Ta) are interpolated from Tmax, Tmin and daylength (d) usig the method of Goudriaan et al., 1994. During sunlight hours Ta is calculated each hour using a sinusoidal curve fitted to Tmin and Tmax . After sunset Ta is calculated as an exponential decline from Ta at sunset to the Tmin at sunrise the next day. The hour (Th) of sunrise is calculated as Th = 12 - d/2 and Ta is assumed to equal Tmin at this time. Tmax is reached when Th equals 13.5.

Each of the interpolated air temperatures are then passed into the following Response and the Average taken to give daily Progression

Response is calcualted using a sigmoid function of the form y = Xmax * 1 / 1 + $e^{-(Xvalue - Xo) / b}$. \n Ymax is calculated as

Ymax = 1 ()

Xo is calculated as

Xo = 12.42 ()

b is calculated as

b = -3.865 ()

Xvalue is calculated as

Unknown child name: XValue

3.2.2 Budding Phase

The *Budding* phase goes from the *Ecodormancy* stage to the *BudBurst* stage.

The *Target* for completion is calculated as

Target = 1011 (oD)

Progression through the *Budding* phase is calculated daily and accumulated until the *Target* is reached.

Progression is the Average of sub-daily values from a Models.Functions.XYPairs.

Firstly hourly estimates of air temperature (Ta) are interpolated from Tmax, Tmin and daylength (d) usig the method of Goudriaan et al., 1994. During sunlight hours Ta is calculated each hour using a sinusoidal curve fitted to Tmin and Tmax . After sunset Ta is calculated as an exponential decline from Ta at sunset to the Tmin at sunrise the next day. The hour (Th) of sunrise is calculated as Th = 12 - d/2 and Ta is assumed to equal Tmin at this time. Tmax is reached when Th equals 13.5.

Each of the interpolated air temperatures are then passed into the following Response and the Average taken to give daily Progression

Response is calculated from an XY matrix (graphed below) which returns a value for Y interpolated from the Xvalue provided.

Graph

3.2.3 Flowering Phase

The *Flowering* phase goes from the *BudBurst* stage to the *Flowering* stage.

The *Target* for completion is calculated as

Target = 21.1 (oD)

Progression through the *Flowering* phase is calculated daily and accumulated until the *Target* is reached.

Progression = [Phenology].ThermalTime

3.2.4 FruitSet Phase

The *FruitSet* phase goes from the *Flowering* stage to the *FruitSet* stage.

The *Target* for completion is calculated as

Target = 10 (oD)

Progression through the *FruitSet* phase is calculated daily and accumulated until the *Target* is reached.

Progression = [Phenology].ThermalTime

Fruit set (or the proportion of flowers that are retained as berries) represents a change-over from the static condition of the fully developed flower to the rapidly growing condition of the young fruit (Coombe, 1962).

In practice the criteria to define this stage is a bit blur. In current, we set the duration from 50% flowering to fruit set is 10 thermal days based on our field experience and the method for determining berry number by meteorology factors. Fruit set phase is used as the time when the berry number is determined in this model.

3.2.5 BerryDevelopment Phase

The *BerryDevelopment* phase goes from the *FruitSet* stage to the *Veraison* stage.

The *Target* for completion is calculated as

Target = 23 (oD)

Progression through the *BerryDevelopment* phase is calculated daily and accumulated until the *Target* is reached.

Progression = [Phenology].ThermalTime

3.2.6 CanopySenescence Phase

The *CanopySenescence* phase goes from the *Veraison* stage to the *LeafFall stage* which occurs when all leaves have fully senesced.

ThermalTime = [Phenology].ThermalTime

3.2.7 Dormancy

When LeafFall is reached phenology is rewound to EndoDormancy

Phenology is also rewound to the EndoDormant stage by a prun event

3.2.8 AccChillBefPrune

Accumulates *ChillBeforePrune* between [Start] and [End]

3.2.8.1 ChillBeforePrune

IF [Weather].DaysSinceWinterSolstice < PruningTime THEN

3.2.8.2 ChillBeforeAutumn

IF autumn < [Weather].DaysSinceWinterSolstice THEN

3.2.8.3 ChillBelowCriticalPhotoperiod

IF PhotoperiodFunction < CriticalPhotoPeriod THEN

ChillingUnit = [Phenology].EndoDormancy.Progression

ELSE

Zero = 0 ()

ELSE

Zero = 0 ()

ELSE

Zero = 0 ()

3.3 Structure

The structure model simulates morphological development of the plant to inform the Leaf class when and how many leaves and branches appear and provides an estimate of height.

3.3.1 Plant and Main-Stem Population

The *Plant.Population* is set at sowing with information sent from a manager script in the Sow method. The *PrimaryBudNumber* is also sent with the Sow method and the main-stem population (*MainStemPopn*) for Grapevine is calculated as:

MainStemPopn = *Plant.Population* x *PrimaryBudNumber*

Primary bud number is > 1 for crops like potato and grape vine where there are more than one main-stem per plant

3.3.2 Main-Stem leaf appearance

Each day the number of main-stem leaf tips appeared (*LeafTipsAppeared*) is calculated as:

LeafTipsAppeared += *DeltaTips*

Where *DeltaTips* is calculated as:

DeltaTips = *ThermalTime/Phyllochron*

Where *Phyllochron* is the thermal time duration between the appearance of leaf tips given by:

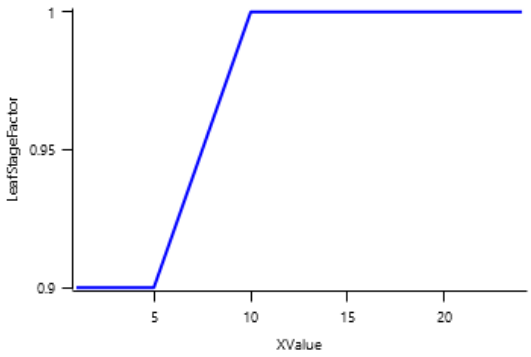
Phyllochron has a non-zero value between BudBurst and Veraison calcualted as:

Phyllochron = *MaximumPhyllochron x LeafStageFactor*

Where:

MaximumPhyllochron = 1.823 ()

LeafStageFactor is calculated using linear interpolation.



X	Y
1	0.9
5	0.9
10	1
24	1

XValue = *[Leaf].AppearedCohortNo*

ThermalTime is given by:

ThermalTime = *[Phenology].ThermalTime*

LeafTipsAppeared continues to increase until *FinalLeafNumber* is reached where *FinalLeafNumber* is calculated as:

FinalLeafNumber = 25 ()

3.3.3 Branching and Branch Mortality

The total population of stems (*TotalStemPopn*) is calculated as:

TotalStemPopn = *MainStemPopn* + *NewBranches* - *NewlyDeadBranches*

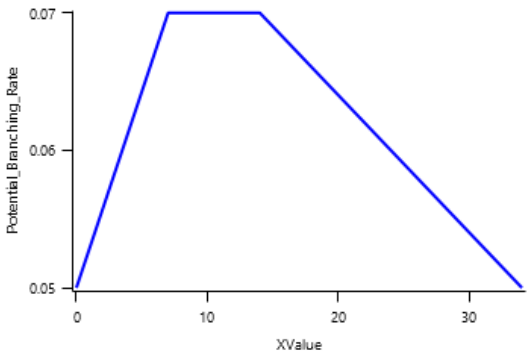
Where *NewBranches* = *MainStemPopn* x *BranchingRate*

and *BranchingRate* is given by:

BranchingRate = *Potential_Branching_Rate x DensityEffect x CanopySizeEffect*

Where:

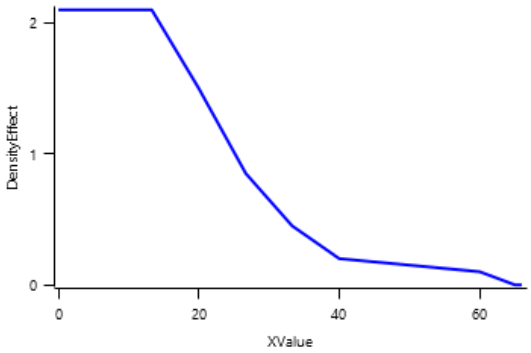
Potential_Branching_Rate is calculated using linear interpolation.



X	Y
0	0.05
7	0.07
8	0.07
13	0.07
14	0.07
34	0.05

XValue = [Structure].LeafTipsAppeared

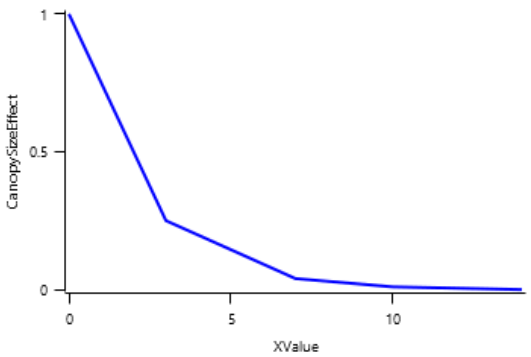
DensityEffect is calculated using linear interpolation.



X	Y
0	2.1
13.3	2.1
20	1.5
26.7	0.85
33.3	0.45
40	0.2
60	0.1
65	0
66	0

XValue = [Structure].BudsPerMeterRow.FixedValue

CanopySizeEffect is calculated using linear interpolation.



X	Y
0	1
3	0.25
7	0.04
10	0.01
14	0

XValue = [Leaf].LAI

NewlyDeadBranches is calcualted as:

NewlyDeadBranches = (*TotalStemPopn - MainStemPopn*) x *BranchMortality*

where *BranchMortality* is given by:

BranchMortality = 0 ()

3.3.4 Height

The Height of the crop is calculated by the *HeightModel*:

HeightModel = BaseHeight p ShootHeight

Where:

BaseHeight = 1100 ()

3.3.4.1 ShootHeight

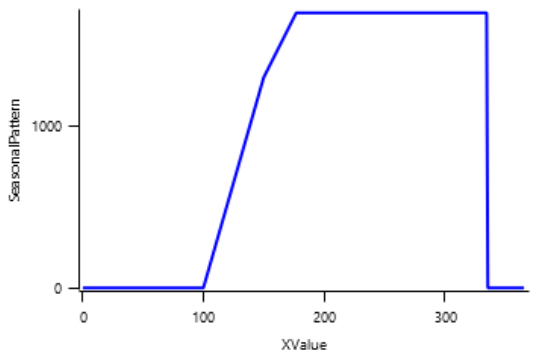
ShootHeight is calculated as the minimum of *ShootLength* and *TrimShootHeight*

Where:

ShootLength = SeasonalPattern x DensityEffect

Where:

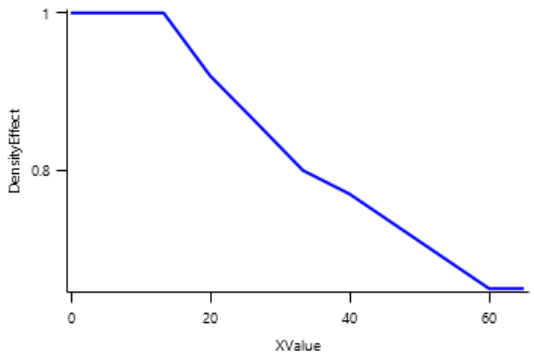
SeasonalPattern is calculated using linear interpolation.



X	Y
0	0
100	0
150	1300
177	1700
335	1700
336	0
366	0

XValue = [Weather].DaysSinceWinterSolstice

DensityEffect is calculated using linear interpolation.



X	Y
0	1
13.3	1
20	0.92
26.7	0.86
33.3	0.8
40	0.77
60	0.65
65	0.65

XValue = [Structure].BudsPerMeterRow.FixedValue

TrimShootHeight = 1100 ()

The structure module defines the bud number per vine, rate of main stem primordia initiation rate, phyllochron (thermal day interval between two successive leaf appearance, 1.823 td in this model), branching rate and branching mortality, final leaf number and canopy height. Phyllochron was defined by a phase look up function in combination with a linear interpolation function. The phase look up function defines when new leaf appearance will happen, which is currently defined between budburst and véraison. The linear interpolation function defines how phyllochron changes with leaf rank based on the data presented in (Greer and Weston, 2010). Branching rate was calculated as the product of three components: potential branching rate, the effect of leaf area per vine, and the effects of retained bud number per meter row

3.3.5 BudNumber

Each time BudBurst occurs bud number on each main-stem is set to

*FractionOfBudBurst * SowingData.BudNumber* (from manager at establishment)

FractionOfBudBurst = BudNumberEffect x CordonDiameterEffect

Where:

BudNumberEffect is calcualted using a sigmoid function of the form y = Xmax * 1 / 1 + $e^{-(Xvalue - Xo) / b}$. \n
Ymax is calculated as

Ymax = 1.4 ()

Xo is calculated as

Xo = 100 ()

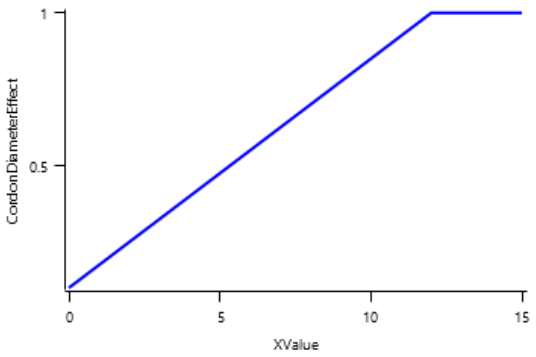
b is calculated as

b = -54.8977 ()

Xvalue is calculated as

XValue = [Structure].PrimaryBudNo

CordonDiameterEffect is calculated using linear interpolation.



X	Y
0	0.1
12	1
15	1

XValue = [Grapevine].Cordon.CordonDiameter

BudsPerMeterRow = 25 ()

3.4 Leaf

The leaves are modelled as a set of leaf cohorts and the properties of each of these cohorts are summed to give overall values for the leaf organ. A cohort represents all the leaves of a given main- stem node position including all of the branch leaves appearing at the same time as the given main-stem leaf (Lawless et al., 2005). The number of leaves in each cohort is the product of the number of plants per m^2 and the number of branches per plant. The *Structure* class models the appearance of main-stem leaves and branches. Once cohorts are initiated the *Leaf* class models the area and biomass dynamics of each. It is assumed all the leaves in each cohort have the same size and biomass properties. The modelling of the status and function of individual cohorts is delegated to *LeafCohort* classes.

3.4.1 Dry Matter Fixation

The most important DM supply from leaf is the photosynthetic fixation supply. Radiation interception is calculated from LAI using an extinction coefficient of:

ExtinctionCoeff = 0.6 ()

3.4.1.1 Photosynthesis

Biomass fixation is modelled as the product of intercepted radiation and its conversion efficiency, the radiation use efficiency (RUE) (Monteith et al., 1977). This approach simulates net photosynthesis rather than providing separate estimates of growth and respiration. The potential photosynthesis calculated using RUE is then adjusted according to stress factors, these account for plant nutrition (FN), air temperature (FT), vapour pressure deficit (FVPD), water supply (FW) and atmospheric CO_2 concentration (FCO2). NOTE: RUE in this model is expressed as g/MJ for a whole plant basis, including both above and below ground growth.

3.4.1.1.1 RUE

RUE is calculated using specific values or functions for various growth phases. The function will use a value of zero for phases not specified below.

Vegetative has a non-zero value between BudBurst and Flowering calcualted as:

Constant = 0.7 (g/MJ)

Reproductive has a non-zero value between Flowering and LeafFall calcualted as:

Constant = 0.8 (g/MJ)

3.4.1.1.2 FCO2

This model calculates the CO_2 impact on RUE using the approach of Reyenga et al., 1999.

For C3 plants,

$F_{CO2} = (CO_2 - CP) \times (350 + 2 \times CP)/(CO_2 + 2 \times CP) \times (350 - CP)$

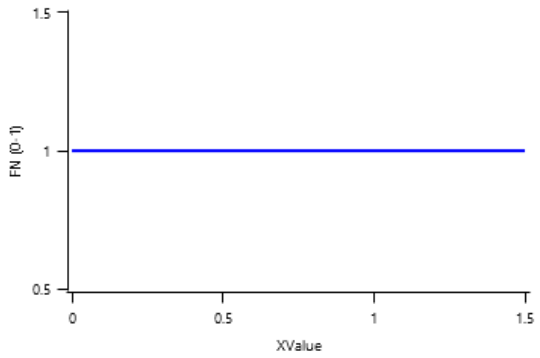
where CP, is the compensation point calculated from daily average temperature (T) as

$CP = (163.0 - T) / (5.0 - 0.1 * T)$

For C4 plants,

$F_{CO2} = 0.000143 * CO_2 + 0.95$

FN is calculated using linear interpolation.



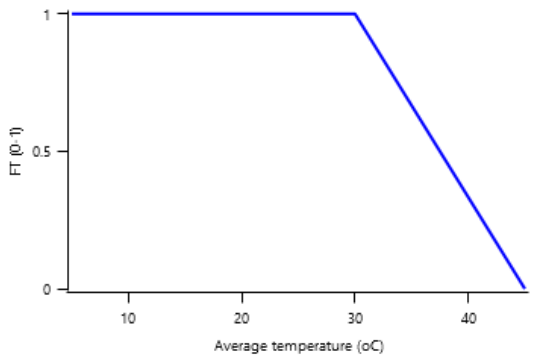
X	Y
0	1
1	1
1.5	1

XValue = [Leaf].Fn

3.4.1.1.3 FT

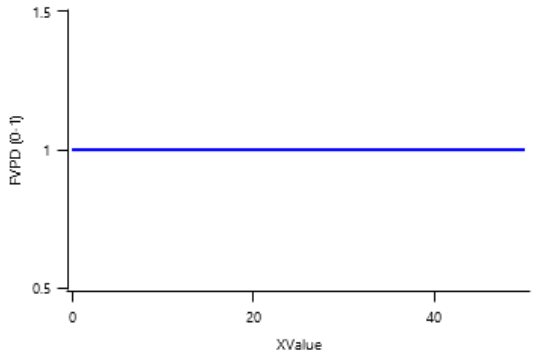
FT is calculated as a function of daily min and max temperatures, these are weighted toward max temperature according to the specified MaximumTemperatureWeighting factor. A value equal to 1.0 means it will use max temperature, a value of 0.5 means average temperature.

MaximumTemperatureWeighting = 0.75



X	Y
5	1
20	1
30	1
45	0

FVPD is calculated using linear interpolation.



X	Y
0	1
10	1
50	1

XValue = [Leaf].Photosynthesis.VPD

RadnInt = [Leaf].RadiationIntercepted

FW = 1 (0-1)

The grapevine model used the phytomer-based Leaf organ class. It predicted the appearance, expansion and senescence of cohorts of leaves at each position on the primary shoot and estimated how many leaves were present in each cohort based on branching rates. Branch leaves were treated the same as main-stem leaves appeared at the same time.

The maximum area of each cohort was currently set as a function of relative node position in respect to the final leaf number. The growth duration, lag duration, senescence duration and specific minimum and maximum leaf area were parameterised based on our field experiment, see Datasets used for Model calibration and validation. For correctly modelling leaf senescence and the dynamics of leaf area, a photoperiod acceleration effect was added. The photoperiod acceleration effect included and reflected the effects of the time of harvest on leaf senescence as well, as we observed that leaves would stay green much longer if the fruit were not harvested, See Supplementary Method 1 leaf module. Other leaf properties were parameterised to ensure no nitrogen or water stress on leaf expansion in the current grapevine model as the water and nitrogen dynamic part require further calibration.

The potential total DM demand of a leaf was calculated based upon the delta leaf area increase per day (constrained by stress) times the mean of maximum and minimum specific leaf area given in UI. The total DM demand was then divided into structural DM demand and Metabolic DM demand based on the fraction of structural DM in total leaf weight. The priority factor q for leaf structural and metabolic demand was set to a high value (1.6) as it is the immediate source of photosynthetic and has the highest priority early in the season (Buwalda, 1991; Lakso et al., 2008). Non-structural reserves were not considered in leaf current model. At leaf senescence, 50% of the total leaf DM was reallocated to other organs following the carbon allocation method, the other half was lost due to respiration to generate the energy required for the degradation and export metabolism in the face of declining photosynthesis (Keller, 2015).

The dynamics of seasonal canopy width and canopy depth were defined in the leaf class based on our field observation.

ThermalTime = [Structure].ThermalTime

Area = 1000

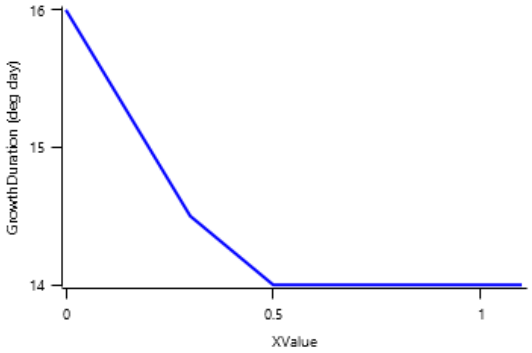
Area = 0

Area = 0

3.4.2 Potential Leaf Area index

Leaf area index is calculated as the sum of the area of each cohort of leaves The appearance of a new cohort of leaves occurs each time Structure.LeafTipsAppeared increases by one. From tip appearance the area of each cohort will increase for a certian number of degree days defined by the *GrowthDuration*

GrowthDuration is calculated using linear interpolation.



X	Y
0	16
0.3	14.5
0.5	14
1	14
1.1	14

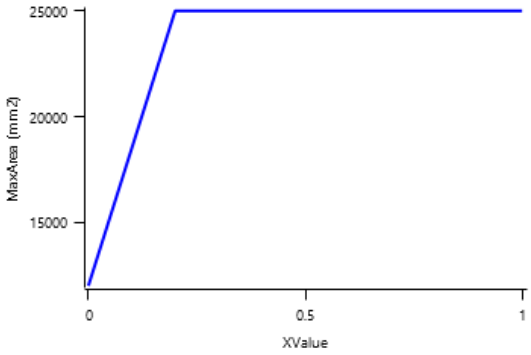
XValue = [Structure].RelativeNodeApperance

If no stress occurs the leaves will reach a Maximum area (*MaxArea*) at the end of the *GrowthDuration*. The *MaxArea* is defined by:

MaxArea = MaxArea x Constant

Where:

MaxArea is calculated using linear interpolation.



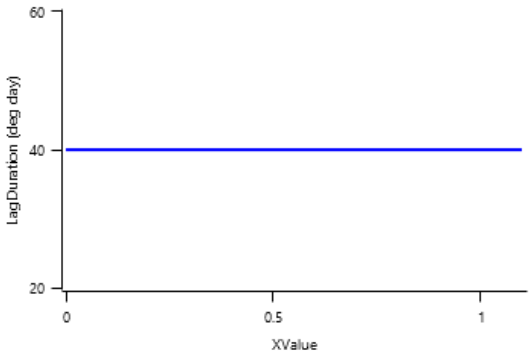
X	Y
0	12000
0.2	25000
1	25000

XValue = [Structure].RelativeNodeApperance

Constant = 0.5 (mm2)

In the absence of stress the leaf will remain at *MaxArea* for a number of degree days set by the *LagDuration* and then area will senesce to zero at the end of the *SenescenceDuration*

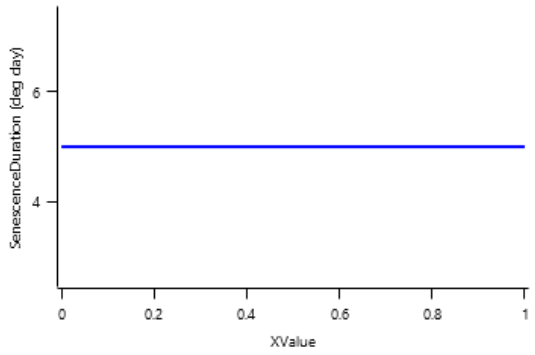
LagDuration is calculated using linear interpolation.



X	Y
0	40
0.15	40
0.3	40
0.4	40
0.73	40
1	40
1.1	40

XValue = [Structure].RelativeNodeApperance

SenescenceDuration is calculated using linear interpolation.

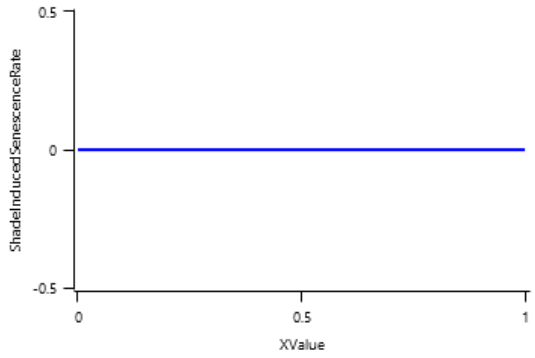


X	Y
0	5
0.2	5
1	5

XValue = [Structure].RelativeNodeApperance

Mutual shading can cause premature senescence of cohorts if the leaf area above them becomes too great. Each cohort models the proportion of its area that is lost to shade induced senescence each day as:

ShadeInducedSenescenceRate is calculated using linear interpolation.



X	Y
0	0
0.5	0
0.8	0
1	0

XValue = [Leaf].CohortCurrentRankCoverAbove

3.4.3 Stress effects on Leaf Area Index

Stress reduces leaf area in a number of ways. Firstly, stress occuring prior to the appearance of the cohort can reduce cell division, so reducing the maximum leaf size. Leaf captures this by multiplying the *MaxSize* of each cohort by a *CellDivisionStress* factor which is calculated as:

CellDivisionStress = 1 ()

Leaf.FN quantifys the N stress status of the plant and represents the concentration of metabolic N relative the maximum potentil metabolic N content of the leaf calculated as (*Leaf.NConc - MinimumNConc*)/(*CriticalNConc - MinimumNConc*).

Leaf.FW quantifies water stress and is calculated as *Leaf.Transpiration*/*Leaf.WaterDemand*, where *Leaf.Transpiration* is the minimum of *Leaf.WaterDemand* and *Root.WaterUptake*

Stress during the *GrowthDuration* of the cohort reduces the size increase of the cohort by multiplying the potential increase by a *ExpansionStress* factor:

ExpansionStress = 1 ()

Stresses can also acellerate the onset and rate of senescence in a number of ways. Nitrogen shortage will cause N to be retranslocated out of lower order leaves to support the expansion of higher order leaves and other organs When this happens the lower order cohorts will have their area reduced in proportion to the amount of N that is remobilised out of them.

Water stress hastens senescence by increasing the rate of thermal time accumulation in the lag and senescence phases. This is done by multiplying thermal time accumulation by *DroughtInducedLagAcceleration* and *DroughtInducedSenescenceAcceleration* factors, respectively:

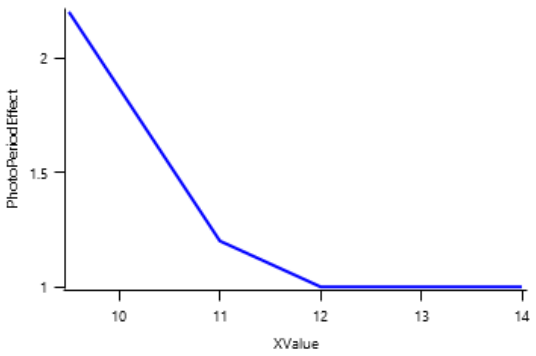
3.4.3.1 DroughtInducedLagAcceleration

IF [Weather].DaysSinceWinterSolstice < Autumn THEN

One = 1 ()

ELSE

PhotoPeriodEffect is calculated using linear interpolation.



X	Y
9.5	2.2
11	1.2
12	1
14	1

3.4.3.1.1 XValue

Returns the duration of the day, or photoperiod, in hours. This is calculated using the specified latitude (given in the weather file) and twilight sun angle threshold. If a variable called ClimateControl.PhotoPeriod is found in the simulation, it will be used instead.

Twilight = 0 (degrees)

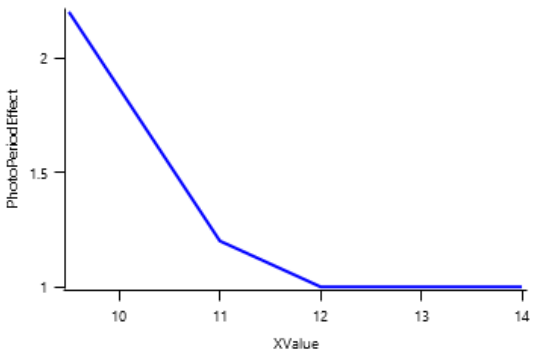
3.4.3.2 DroughtInducedSenAcceleration

IF [Weather].DaysSinceWinterSolstice < Autumn THEN

One = 1 ()

ELSE

PhotoPeriodEffect is calculated using linear interpolation.



X	Y
9.5	2.2
11	1.2
12	1
14	1

3.4.3.2.1 XValue

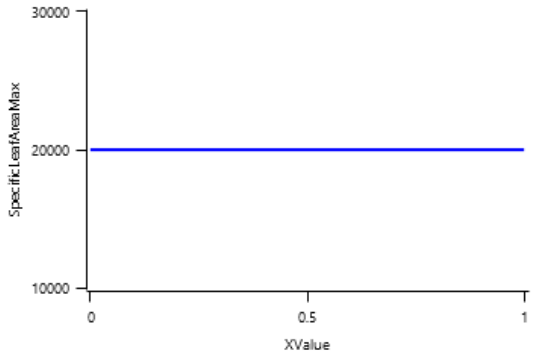
Returns the duration of the day, or photoperiod, in hours. This is calculated using the specified latitude (given in the weather file) and twilight sun angle threshold. If a variable called ClimateControl.PhotoPeriod is found in the simulation, it will be used instead.

Twilight = 0 (degrees)

3.4.4 Dry matter Demand

Leaf calculates the DM demand from each cohort as a function of the potential size increment (DeltaPotentialArea) an specific leaf area bounds. Under non stressed conditions the demand for non-storage DM is calculated as *DeltaPotentialArea* divided by the mean of *SpecificLeafAreaMax* and *SpecificLeafAreaMin*. Under stressed conditions it is calculated as *DeltaWaterConstrainedArea* divided by *SpecificLeafAreaMin*.

SpecificLeafAreaMax is calculated using linear interpolation.



X	Y
0	20000
1	20000

XValue = [Structure].RelativeNodeApperance

SpecificLeafAreaMin = 15000 ()

Non-storage DM Demand is then seperated into structural and metabolic DM demands using the *StructuralFraction*:

StructuralFraction = 0.8 ()

The storage DM demand is calculated from the sum of metabolic and structural DM (including todays demands) multiplied by a *NonStructuralFraction*:

Unknown child name: NonStructuralFraction

3.4.5 Nitrogen Demand

Leaf calculates the N demand from each cohort as a function of the potential DM increment and N concentration bounds. Structural N demand = *PotentialStructuralDMAllocation* * *MinimumNConc* where:

MinimumNConc = 0.0001 ()

Metabolic N demand is calculated as *PotentialMetabolicDMAllocation* * (*CriticalNConc* - *MinimumNConc*) where:

CriticalNConc = 0.0001 ()

Storage N demand is calculated as the sum of metabolic and structural wt (including todays demands) multiplied by *LuxaryNconc* (*MaximumNConc* - *CriticalNConc*) less the amount of storage N already present. *MaximumNConc* is given by:

MaximumNConc = 0.05 ()

3.4.6 Drymatter supply

In additon to photosynthesis, the leaf can also supply DM by reallocation of senescing DM and retranslocation of storgage DM: Reallocation supply is a proportion of the metabolic and non-structural DM that would be senesced each day where the proportion is set by:

DMReallocationFactor = 0.5 ()

Retranslocation supply is calculated as a proportion of the amount of storage DM in each cohort where the proportion is set by :

DMRetranslocationFactor = 1 ()

3.4.7 Nitrogen supply

Nitrogen supply from the leaf comes from the reallocation of metabolic and storage N in senescing material and the retranslocation of metabolic and storage N. Reallocation supply is a proportion of the Metabolic and Storage DM that would be senesced each day where the proportion is set by:

NReallocationFactor = 0 ()

Retranslocation supply is calculated as a proportion of the amount of storage and metabolic N in each cohort where the proportion is set by :

NRetranslocationFactor = 0 ()

DetachmentLagDuration = 1 (deg day)

DetachmentDuration = 1 (deg day)

InitialNConc = 0.05 ()

StorageFraction = 0 ()

SenessingLeafRelativeSize = 1 ()

LeafSizeShapeParameter = 0.01 ()

3.4.7.1 LagDurationAgeMultiplier

LagDurationAgeMultiplier = 1 1 1

3.4.7.2 SenescenceDurationAgeMultiplier

SenescenceDurationAgeMultiplier = 1 1 1

3.4.7.3 LeafSizeAgeMultiplier

LeafSizeAgeMultiplier = 1 1 1 1 1 1 1 1 1 1 1

RemobilisationCost = 0 ()

CarbonConcentration = 0.4 ()

MaintenanceRespirationFunction = Qm x ExponentialFunction

Where:

Qm = 0.03 ()

3.4.7.4 ExponentialFunction

An exponential function

SubtractFunction = [Weather].MeanT - RefT

Where:

RefT = 20 ()

Xvalue = [Weather].MeanT

3.4.8 FRGRFunction

FRGRFunction is calculated as the minimum of *RUE_FT* and *Others*

Where:

RUE_FT = [Leaf].Photosynthesis.FT

3.4.8.1 Others

Others is calculated as the minimum of *RUE_FN* and *RUE_FVPD*

Where:

RUE_FN = [Leaf].Photosynthesis.FN

RUE_FVPD = [Leaf].Photosynthesis.FVPD

3.4.9 Total Biomass

This is a composite biomass class, representing the sum of 1 or more biomass objects.

Total summarises the following biomass objects:

- [Leaf].Live
- [Leaf].Dead

3.4.10 BiomassRemovalDefaults

This organ will respond to certain management actions by either removing some of its biomass from the system or transferring some of its biomass to the soil surface residues. The following table describes the default proportions of live and dead biomass that are transferred out of the simulation using "Removed" or to soil surface residue using "To Residue" for a range of management actions. The total percentage removed for live or dead must not exceed 100%. The difference between the total and 100% gives the biomass remaining on the plant. These can be changed during a simulation using a manager script.

Method	% Live Removed	% Dead Removed	% Live To Residue	% Dead To Residue
Harvest	0	0	0	0
Cut	0	0	0	0
Prune	0	0	0	0
Graze	0	0	0	0

DMConversionEfficiency = 0.83 ()

RemobilisationCost = 0 ()

CarbonConcentration = 0.4 ()

StructuralFraction = 0.8 ()

StomatalConductanceCO2Modifier = 1 ()

3.4.11 DMDemandPriorityFactors

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

Structural = 1.6 (g/m2)

Metabolic = 1.6 (g/m2)

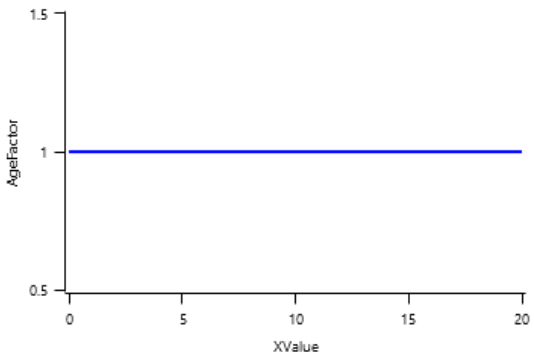
Storage = 0.01 (g/m2)

FrostFraction = 0 ()

WidthFunction = AgeFactor x SeasonalWidthPattern

Where:

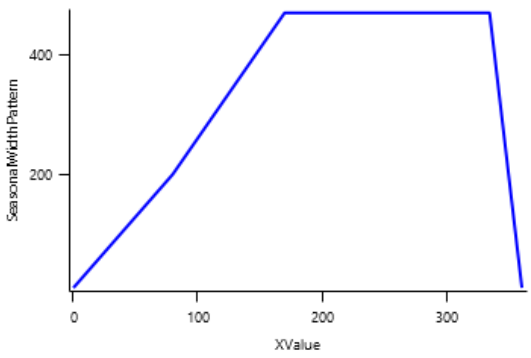
AgeFactor is calculated using linear interpolation.



X	Y
0	1
2	1
10	1
20	1

XValue = [Phenology].Age.Years

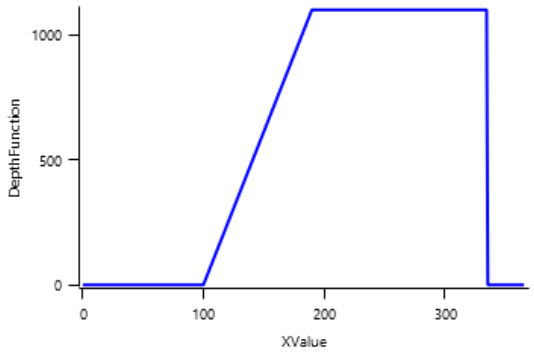
SeasonalWidthPattern is calculated using linear interpolation.



X	Y
0	10
80	200
170	470
335	470
361	10

XValue = [Weather].DaysSinceWinterSolstice

DepthFunction is calculated using linear interpolation.



X	Y
0	1
100	1
190	1100
335	1100
336	1
366	1

XValue = [Weather].DaysSinceWinterSolstice

3.5 Shoot

This organ is simulated using a GenericOrgan type. It is parameterised to calculate the growth, senescence, and detachment of any organ that does not have specific functions.

Shoot was represented by a Generic Organ class.

The structural biomass demand of the shoot module was calculated by a delta function that takes the difference of yesterday and today's value of a fitted biomass growth curve. The structural biomass growth curve was represented by a beta growth function (Yin et al., 2003) fitted on individual shoot weight over time and times the shoot population per square meter. Shoot population per square meter was calculated internally based on shoot number per vine and vine density. DMs(t)= SP*DM _max (1+ (t_e-t)/(t_e-t_m)) (t/t_e)^(t_e/(t_e-t_m)) 0=t_m<t_e (Eq. 8) DMs(t) is the DM of all shoots (g m-2) at time t from budburst. SP the shoot population per square meter. DMmax is the fitted maximum DM per shoot under four-cane pruned vines (g). t_e is the time when DMmax was reached (79.2 td), t_m is the time at maximum growth rate (43 td).

Note the shoot dry matter including both the primary and secondary shoots. For capturing the reduction of shoot biomass under high retained bud number per meter row caused both by smaller primary shoot and less lateral shoots (Greven et al., 2014), the priority factor for shoot DM demand was set to a very low value (4e-3).

Parameters for shoot non-structural DM demand, e.g. priority factors were set as half of the trunk and root.

3.5.1 Dry Matter Demand

The dry matter demand for the organ is calculated as defined in DMDemands, based on the DMDemandFunction and partition fractions for each biomass pool.

3.5.1.1 DMDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

Structural is the daily differential of

Integral = [Structure].MainStemPopn x BetaGrowthFunction

Where:

3.5.1.1.1 BetaGrowthFunction

a beta growth function of the form y = Ymax * (1 + (te - t)/(te-tm))* (t/te)^(te/(te-tm))

Ymax = 30 (g/m2)

XValue = [Shoot].ThermalTimeAfterBudBurst

ShootPopulation = [Structure].MainStemPopn

Metabolic = 0 (g/m2)

Storage is calculated as the minimum of *Storage* and *Zero*

Where:

Storage = SubtractFunction x DailySynthesis

Where:

SubtractFunction = StructuralWt - [Grapevine].Shoot.Live.StorageWt

Where:

StructuralWt = [Grapevine].Shoot.Live.Wt x MaxConcentration

Where:

MaxConcentration = 0.26 (g/m2)

StructuralWt = [Grapevine].Shoot.Live.Wt

StorageWt = [Grapevine].Shoot.Live.StorageWt

3.5.1.1.2 DailySynthesis

a beta growth function of the form y = Ymax * (1 + (te - t)/(te-tm))* (t/te)^(te/(te-tm))

Ymax = 0.03 (g/m2)

XValue = [Shoot].ThermalTimeAfterBudBurst

Zero = 0 (g/m2)

3.5.2 Nitrogen Demand

The N demand is calculated as defined in NDemands, based on DM demand the N concentration of each biomass pool.

3.5.2.1 NDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

Structural = [Shoot].minimumNconc x [Shoot].potentialDMAllocation.Structural

MinNconc = [Shoot].minimumNconc

PotentialDMAllocation = [Shoot].potentialDMAllocation.Structural

Metabolic = MetabolicNconc x [Shoot].potentialDMAllocation.Structural

Where:

MetabolicNconc = [Shoot].criticalNConc - [Shoot].minimumNconc

CritNconc = [Shoot].criticalNConc

MinNconc = [Shoot].minimumNconc

PotentialDMAllocation = [Shoot].potentialDMAllocation.Structural

3.5.2.1.1 Storage

The partitioning of daily N supply to storage N attempts to bring the organ's N content to the maximum concentration.

Storage = [Shoot].maximumNconc × ([Shoot].Live.Wt + potentialAllocationWt) - [Shoot].Live.N

The demand for storage N is further reduced by a factor specified by the [Shoot].NitrogenDemandSwitch.

MinimumNConc = 0 (g/g)

CriticalNConc = [Shoot].MinimumNConc

MaximumNConc = 0.0001 (g/g)

The demand for N is reduced by a factor specified by the NitrogenDemandSwitch.

NitrogenDemandSwitch has a non-zero value between BudBurst and LeafFall calcualted as:

Constant = 1 ()

3.5.3 Dry Matter Supply

Shoot will reallocate 100% of DM that senesces each day.

Shoot will retranslocate 4% of non-structural DM each day.

3.5.4 Nitrogen Supply

Shoot can reallocate up to 100% of N that senesces each day if required by the plant arbitrator to meet N demands.

Shoot can retranslocate up to 10% of non-structural N each day if required by the plant arbitrator to meet N demands.

3.5.5 Senescence and Detachment

Shoot has senescence parameterised to zero so all biomass in this organ will remain alive.

Shoot has detachment parameterised to zero so all biomass in this organ will remain with the plant until a defoliation or harvest event occurs.

3.5.6 BiomassRemovalDefaults

This organ will respond to certain management actions by either removing some of its biomass from the system or transferring some of its biomass to the soil surface residues. The following table describes the default proportions of live and dead biomass that are transferred out of the simulation using "Removed" or to soil surface residue using "To Residue" for a range of management actions. The total percentage removed for live or dead must not exceed 100%. The difference between the total and 100% gives the biomass remaining on the plant. These can be changed during a simulation using a manager script.

Method	% Live Removed	% Dead Removed	% Live To Residue	% Dead To Residue
Harvest	0	0	0	0
Cut	0	0	0	0
Prune	0	0	0	0
Graze	0	0	0	0

3.6 Cordon

This organ is simulated using a GenericOrgan type. It is parameterised to calculate the growth, senescence, and detachment of any organ that does not have specific functions.

Cordon, Trunk and Structural root module were all represented by the Generic Organ class, which has properties of biomass status and daily biomass demand and supply. The structural DM demand of cordon and trunk (Eq. 6) were calculated by their radius (r, unit m) derived from their DM and length, daily growth rate of the radius (dr/dt, m d-1), length (l, m) and wood density (, g m-3) (Cieslak et al., 2011). dStructuralDM _i/dt=q_(str,i) 2prl dr/dt Eq. 6 where i was organ type. Qstr,i was the priority factor for structural DM demand for a certain organ, determined as 0.7 for all three organs see model calibration methods. Daily growth rate of the radius were calculated based on the trunk circumference measurements on the same eight hundreds vines with 14 years' interval. The structural carbon demand of the structural root was calculated by the structural demand of the trunk times the structural root/trunk ratio (set to one in this model). The structural root/trunk ratio may vary between vineyards and training systems.

The non-structural DM demand was modelled as an active competing sink (Cieslak et al., 2011), and the parameters were kept the same for those three organs. The rate of non-structural DM synthesis depended on organ size and limited by overloading. dNonStructuralDM _i/dt=q_(NSC,i) *k_syn* (C_(max,NSC)* StructuralDM _i- NonStructuralDM _i) Eq. 7 The equation limited carbon storage owing to non-structural DM overloading and the organ storage capacity, which was proportional to the current organ structural biomass, Cmax,NSC was the maximum non-structural DM per unit of structural DM, set as 0.26 based on our field trunk and root samples (Greven et al., 2016). qNSC was set to 0.61 throughout the growing season. ksyn was daily non-structural DM synthesis rate, which was represented by a beta growth function Eq. 8 (Yin et al., 2003) for capturing the fast recovery of carbon reserves after flowering (Greven et al., 2016; Seleznyova et al., 2018).

Parameters were optimized based on the measured dynamics of non-structural carbon in trunk and root by Greven et al. (2016), see model calibration methods. k_syn=k_max (1+ (t_e-t)/(t_e-t_m)) (t/t_e)^(t_e/(t_e-t_m)) 0=t_m<t_e (Eq. 8) k_max was the maximum non-structural DM synthesis rate, 0.03 g g-1. t_e was the thermal day after budburst when kmax was reached, 54 td (around véraison). t_m was the time when maximum increase rate occurred, 34.4 td. For potential carbon retranslocation by those storage organs, a value of 3.7% each day (DMRetranslocationFactor) was obtained through optimization. Actual retranslocation depends on the daily carbon supply and demand differences. The biomass of retained cane for following season was reset by the pruning events based on input cane diameter in the UI each year for the cane-pruned vines. The biomass of trunk and fibrous root keep growing each year.

3.6.1 Dry Matter Demand

The dry matter demand for the organ is calculated as defined in DMDemands, based on the DMDemandFunction and partition fractions for each biomass pool.

3.6.1.1 DMDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

3.6.1.1.1 Metabolic

Metabolic is calculated using specific values or functions for various growth phases. The function will use a value of zero for phases not specified below.

GrowingSeason has a non-zero value between BudBurst and LeafFall calcualted as:

Constant = 0 (g/m2)

3.6.1.1.2 Structural

Structural is calculated using specific values or functions for various growth phases. The function will use a value of zero for phases not specified below.

GrowingSeason has a non-zero value between BudBurst and LeafFall calcualted as:

Structural = 2 x PI x Radius x Density x RadiusGR x [Cordon].CordonLength x [Grapevine].Population x [Cordon].CordonNumber

Where:

2 = 2 (g/m2)

PI = 3.1415926 (g/m2)

3.6.1.2 Radius

Raises the value of the child to the power of the exponent specified

RadiusSquare = [Cordon].Live.Wt d [Grapevine].Population d [Cordon].CordonNumber d Density d [Cordon].CordonLength d PI

Where:

Density = 500000 (g/m2)

PI = 3.1415926 (g/m2)

Wt = [Cordon].Live.Wt

population = [Grapevine].Population

CordonNumber = [Cordon].CordonNumber

Length = [Cordon].CordonLength

Density = 400000 (g/m2)

RadiusGR = 9E-06 (g/m2)

Length = [Cordon].CordonLength

Population = [Grapevine].Population

CordonNumber = [Cordon].CordonNumber

3.6.1.2.1 Storage

Storage is calculated using specific values or functions for various growth phases. The function will use a value of zero for phases not specified below.

GrowingSeason has a non-zero value between BudBurst and LeafFall calcualted as:

Storage = SubtractFunction x DailySynthesis

Where:

SubtractFunction = StructuralWt - [Grapevine].Cordon.Live.StorageWt

Where:

StructuralWt = [Grapevine].Cordon.Live.StructuralWt x MaxConcentration

Where:

MaxConcentration = 0.26 (g/m2)

StructuralWt = [Grapevine].Cordon.Live.StructuralWt

StorageWt = [Grapevine].Cordon.Live.StorageWt

3.6.1.2.1.1 DailySynthesis

a beta growth function of the form y = Ymax * (1 + (te - t)/(te-tm))* (t/te)^(te/(te-tm))

Ymax = 0.03 (g/m2)

XValue = [Shoot].ThermalTimeAfterBudBurst

3.6.2 Nitrogen Demand

The N demand is calculated as defined in NDemands, based on DM demand the N concentration of each biomass pool.

3.6.2.1 NDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

Structural = [Cordon].minimumNconc x [Cordon].potentialDMAllocation.Structural

MinNconc = [Cordon].minimumNconc

PotentialDMAllocation = [Cordon].potentialDMAllocation.Structural

Metabolic = MetabolicNconc x [Cordon].potentialDMAllocation.Structural

Where:

MetabolicNconc = [Cordon].criticalNConc - [Cordon].minimumNconc

CritNconc = [Cordon].criticalNConc

MinNconc = [Cordon].minimumNconc

PotentialDMAllocation = [Cordon].potentialDMAllocation.Structural

3.6.2.1.1 Storage

The partitioning of daily N supply to storage N attempts to bring the organ's N content to the maximum concentration.

Storage = [Cordon].maximumNconc × ([Cordon].Live.Wt + potentialAllocationWt) - [Cordon].Live.N

The demand for storage N is further reduced by a factor specified by the [Cordon].NitrogenDemandSwitch.

MinimumNConc = 0 (g/g)

CriticalNConc = [Cordon].MinimumNConc

MaximumNConc = 0.0001 (g/g)

The demand for N is reduced by a factor specified by the NitrogenDemandSwitch.

NitrogenDemandSwitch has a non-zero value between BudBurst and LeafFall calcualted as:

Constant = 1 ()

3.6.3 Dry Matter Supply

Cordon will reallocate 100% of DM that senesces each day.

Cordon will retranslocate 3.7% of non-structural DM each day.

3.6.4 Nitrogen Supply

Cordon can reallocate up to 100% of N that senesces each day if required by the plant arbitrator to meet N demands.

Cordon can retranslocate up to 4% of non-structural N each day if required by the plant arbitrator to meet N demands.

3.6.5 Senescence and Detachment

Cordon has senescence parameterised to zero so all biomass in this organ will remain alive.

Cordon has detachment parameterised to zero so all biomass in this organ will remain with the plant until a defoliation or harvest event occurs.

3.6.6 BiomassRemovalDefaults

This organ will respond to certain management actions by either removing some of its biomass from the system or transferring some of its biomass to the soil surface residues. The following table describes the default proportions of live and dead biomass that are transferred out of the simulation using "Removed" or to soil surface residue using "To Residue" for a range of management actions. The total percentage removed for live or dead must not exceed 100%. The difference between the total and 100% gives the biomass remaining on the plant. These can be changed during a simulation using a manager script.

Method	% Live Removed	% Dead Removed	% Live To Residue	% Dead To Residue
Harvest	0	0	0	0

Method	% Live Removed	% Dead Removed	% Live To Residue	% Dead To Residue
Cut	0	0	0	0
Prune	0	0	0	0
Graze	0	0	0	0

3.7 Trunk

This organ is simulated using a GenericOrgan type. It is parameterised to calculate the growth, senescence, and detachment of any organ that does not have specific functions.

3.7.1 Dry Matter Demand

The dry matter demand for the organ is calculated as defined in DMDemands, based on the DMDemandFunction and partition fractions for each biomass pool.

3.7.1.1 DMDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

3.7.1.1.1 Metabolic

Metabolic is calculated using specific values or functions for various growth phases. The function will use a value of zero for phases not specified below.

GrowingSeason has a non-zero value between BudBurst and LeafFall calcualted as:

Constant = 0 (g/m2)

3.7.1.1.2 Structural

Structural is calculated using specific values or functions for various growth phases. The function will use a value of zero for phases not specified below.

GrowingSeason has a non-zero value between BudBurst and LeafFall calcualted as:

Structural = 2 x PI x Radius x [Trunk].TrunkLength x Density x RadiusGR x [Grapevine].Population x Age

Where:

2 = 2 (g/m2)

PI = 3.1415926 (g/m2)

3.7.1.2 Radius

Raises the value of the child to the power of the exponent specified

RadiusSquare = [Trunk].Live.Wt d [Grapevine].Population d PI d [Trunk].TrunkLength d Density

Where:

PI = 3.1415926 (g/m2)

Density = 500000 (g/m2)

Wt = [Trunk].Live.Wt

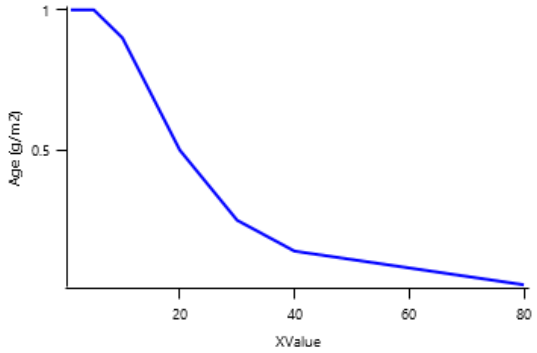
population = [Grapevine].Population

Length = [Trunk].TrunkLength

Density = 500000 (g/m2)

RadiusGR = 8E-06 (g/m2)

Age is calculated using linear interpolation.



X	Y
1	1
5	1
10	0.9
20	0.5
30	0.25
40	0.14
80	0.02

XValue = [Phenology].Age.Years

Length = [Trunk].TrunkLength

population = [Grapevine].Population

3.7.1.2.1 Storage

Storage is calculated using specific values or functions for various growth phases. The function will use a value of zero for phases not specified below.

GrowingSeason has a non-zero value between BudBurst and LeafFall calcualted as:

Storage is calculated as the minimum of *Storage* and *Zero*

Where:

Storage = SubtractFunction x DailySynthesis

Where:

SubtractFunction = StructuralWt - [Grapevine].Trunk.Live.StorageWt

Where:

StructuralWt = [Grapevine].Trunk.Live.StructuralWt x MaxConcentration

Where:

MaxConcentration = 0.26 (g/m2)

StructuralWt = [Grapevine].Trunk.Live.StructuralWt

StorageWt = [Grapevine].Trunk.Live.StorageWt

3.7.1.2.1.1 DailySynthesis

a beta growth function of the form y = Ymax * (1 + (te - t)/(te-tm))* (t/te)^(te/(te-tm))

Ymax = 0.03 (g/m2)

XValue = [Shoot].ThermalTimeAfterBudBurst

Zero = 0 (g/m2)

3.7.2 Nitrogen Demand

The N demand is calculated as defined in NDemands, based on DM demand the N concentration of each biomass pool.

3.7.2.1 NDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

Structural = [Trunk].minimumNconc x [Trunk].potentialDMAllocation.Structural

MinNconc = [Trunk].minimumNconc

PotentialDMAllocation = [Trunk].potentialDMAllocation.Structural

Metabolic = MetabolicNconc x [Trunk].potentialDMAllocation.Structural

Where:

MetabolicNconc = [Trunk].criticalNConc - [Trunk].minimumNconc

CritNconc = [Trunk].criticalNConc

MinNconc = [Trunk].minimumNconc

PotentialDMAllocation = [Trunk].potentialDMAllocation.Structural

3.7.2.1.1 Storage

The partitioning of daily N supply to storage N attempts to bring the organ's N content to the maximum concentration.

Storage = [Trunk].maximumNconc × ([Trunk].Live.Wt + potentialAllocationWt) - [Trunk].Live.N

The demand for storage N is further reduced by a factor specified by the [Trunk].NitrogenDemandSwitch.

MinimumNConc = 0 (g/g)

CriticalNConc = [Trunk].MinimumNConc

MaximumNConc = 0.0001 (g/g)

The demand for N is reduced by a factor specified by the NitrogenDemandSwitch.

NitrogenDemandSwitch has a non-zero value between BudBurst and LeafFall calcualted as:

Constant = 1 ()

3.7.3 Dry Matter Supply

Trunk will reallocate 100% of DM that senesces each day.

Trunk will retranslocate 3.7% of non-structural DM each day.

3.7.4 Nitrogen Supply

Trunk can reallocate up to 100% of N that senesces each day if required by the plant arbitrator to meet N demands.

Trunk can retranslocate up to 10% of non-structural N each day if required by the plant arbitrator to meet N demands.

3.7.5 Senescence and Detachment

Trunk has senescence parameterised to zero so all biomass in this organ will remain alive.

Trunk has detachment parameterised to zero so all biomass in this organ will remain with the plant until a defoliation or harvest event occurs.

3.7.6 BiomassRemovalDefaults

This organ will respond to certain management actions by either removing some of its biomass from the system or transferring some of its biomass to the soil surface residues. The following table describes the default proportions of live and dead biomass that are transferred out of the simulation using "Removed" or to soil surface residue using "To Residue" for a range of management actions. The total percentage removed for live or dead must not exceed 100%. The difference between the total and 100% gives the biomass remaining on the plant. These can be changed during a simulation using a manager script.

Method	% Live Removed	% Dead Removed	% Live To Residue	% Dead To Residue
Harvest	0	0	0	0

Method	% Live Removed	% Dead Removed	% Live To Residue	% Dead To Residue
Cut	0	0	0	0
Prune	0	0	0	0
Graze	0	0	0	0

3.8 StructuralRoot

This organ is simulated using a GenericOrgan type. It is parameterised to calculate the growth, senescence, and detachment of any organ that does not have specific functions.

Note this represents the structural roots of all the plant in the population. The structural root are primarily considered as storage organs, its researves can be made available to boost plant growth in spring and/or following a defoliation.

The structural root is separated from the mian root class because the current root class can not handle the biomass retranslocation while a generic organ can. The biomass of the structural root is expressed as gram per square meter, while the initial dry mass for the root class is expressed at per plant level.

structural roots in grapevine are perennial. It can grow in diamention and biomass. Its structural growth rate was calculated based on the the trunk demand times structural root and trunk ratio in this model. Its storage biomass typicall decrease in spring and refill after veraison.

3.8.1 Dry Matter Demand

The dry matter demand for the organ is calculated as defined in DMDemands, based on the DMDemandFunction and partition fractions for each biomass pool.

3.8.1.1 DMDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

3.8.1.1.1 Metabolic

Metabolic is calculated using specific values or functions for various growth phases. The function will use a value of zero for phases not specified below.

GrowingSeason has a non-zero value between BudBurst and LeafFall calcualted as:

Constant = 0 (g/m2)

3.8.1.1.2 Structural

Structural is calculated using specific values or functions for various growth phases. The function will use a value of zero for phases not specified below.

GrowingSeason has a non-zero value between BudBurst and LeafFall calcualted as:

Structural = 2 x PI x Radius x [Trunk].TrunkLength x Density x RadiusGR x [Grapevine].Population x RootTrunkRatio x Age

Where:

2 = 2 (g/m2)

PI = 3.1415926 (g/m2)

3.8.1.2 Radius

Raises the value of the child to the power of the exponent specified

RadiusSquare = [Trunk].Live.Wt d [Grapevine].Population d PI d [Trunk].TrunkLength d Density

Where:

PI = 3.1415926 (g/m2)

Density = 500000 (g/m2)

Wt = [Trunk].Live.Wt

population = [Grapevine].Population

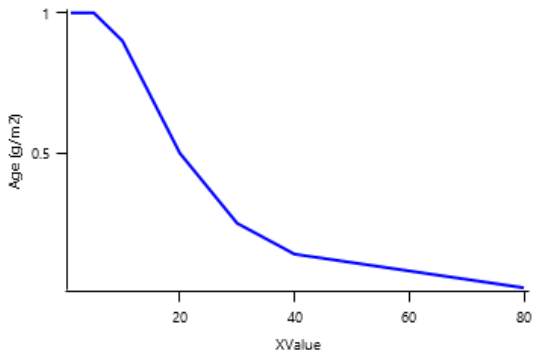
Length = [Trunk].TrunkLength

Density = 500000 (g/m2)

RadiusGR = 8E-06 (g/m2)

RootTrunkRatio = 1 (g/m2)

Age is calculated using linear interpolation.



X	Y
1	1
5	1
10	0.9
20	0.5
30	0.25
40	0.14
80	0.02

XValue = [Phenology].Age.Years

Length = [Trunk].TrunkLength

population = [Grapevine].Population

3.8.1.2.1 Storage

Storage is calculated using specific values or functions for various growth phases. The function will use a value of zero for phases not specified below.

GrowingSeason has a non-zero value between BudBurst and LeafFall calcualted as:

Storage is calculated as the minimum of *Storage* and *Zero*

Where:

Storage = SubtractFunction x DailySynthesis

Where:

SubtractFunction = StructuralWt - [Grapevine].StructuralRoot.Live.StorageWt

Where:

StructuralWt = [Grapevine].StructuralRoot.Live.StructuralWt x MaxConcentration

Where:

MaxConcentration = 0.26 (g/m2)

StructuralWt = [Grapevine].StructuralRoot.Live.StructuralWt

StorageWt = [Grapevine].StructuralRoot.Live.StorageWt

3.8.1.2.1.1 DailySynthesis

a beta growth function of the form y = Ymax * (1 + (te - t)/(te-tm))* (t/te)^(te/(te-tm))

Ymax = 0.03 (g/m2)

XValue = [Shoot].ThermalTimeAfterBudBurst

Zero = 0 (g/m2)

3.8.2 Nitrogen Demand

The N demand is calculated as defined in NDemands, based on DM demand the N concentration of each biomass pool.

3.8.2.1 NDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

Structural = [Trunk].minimumNconc x [Trunk].potentialDMAllocation.Structural

MinNconc = [Trunk].minimumNconc

PotentialDMAllocation = [Trunk].potentialDMAllocation.Structural

Metabolic = MetabolicNconc x [Trunk].potentialDMAllocation.Structural

Where:

MetabolicNconc = [Trunk].criticalNConc - [Trunk].minimumNconc

CritNconc = [Trunk].criticalNConc

MinNconc = [Trunk].minimumNconc

PotentialDMAllocation = [Trunk].potentialDMAllocation.Structural

3.8.2.1.1 Storage

The partitioning of daily N supply to storage N attempts to bring the organ's N content to the maximum concentration.

Storage = [StructuralRoot].maximumNconc × ([StructuralRoot].Live.Wt + potentialAllocationWt) - [StructuralRoot].Live.N

The demand for storage N is further reduced by a factor specified by the [StructuralRoot].NitrogenDemandSwitch.

MinimumNConc = 0 (g/g)

CriticalNConc = [Trunk].MinimumNConc

MaximumNConc = 0.0001 (g/g)

The demand for N is reduced by a factor specified by the NitrogenDemandSwitch.

NitrogenDemandSwitch has a non-zero value between BudBurst and LeafFall calcualted as:

Constant = 1 ()

3.8.3 Dry Matter Supply

StructuralRoot will reallocate 100% of DM that senesces each day.

StructuralRoot will retranslocate 3.7% of non-structural DM each day.

3.8.4 Nitrogen Supply

StructuralRoot can reallocate up to 100% of N that senesces each day if required by the plant arbitrator to meet N demands.

StructuralRoot can retranslocate up to 10% of non-structural N each day if required by the plant arbitrator to meet N demands.

3.8.5 Senescence and Detachment

StructuralRoot has senescence parameterised to zero so all biomass in this organ will remain alive.

StructuralRoot has detachment parameterised to zero so all biomass in this organ will remain with the plant until a defoliation or harvest event occurs.

3.8.6 BiomassRemovalDefaults

This organ will respond to certain management actions by either removing some of its biomass from the system or transferring some of its biomass to the soil surface residues. The following table describes the default proportions of live and dead biomass that are transferred out of the simulation using "Removed" or to soil surface residue using "To Residue" for a range of management actions. The total percentage removed for live or dead must not exceed 100%. The difference between the total and 100% gives the biomass remaining on the plant. These can be changed during a simulation using a manager script.

Method	% Live Removed	% Dead Removed	% Live To Residue	% Dead To Residue
Harvest	0	0	0	0
Cut	0	0	0	0
Prune	0	0	0	0
Graze	0	0	0	0

3.9 Berry

This organ uses a generic model for plant reproductive components. Yield is calculated from its components in terms of organ number and size (for example, grain number and grain size).

The final yield was calculated by bunches per shoot, shoots per vine, berries per bunch,and berry fresh or dry weight.

The effects of carbon status on yield compoent has not been included as it requires further calibration. Bunch number, berry number and potential berry fresh weight were determined by weather conditions at critical periods around flowerings of the previous and current season, see details at Zhu et al., 2020 OENO one. Furthermore, in the current model, carbon effects represented by total carbon supply and demand were added in the calculation of bunch number, berry number and potential berry fresh weight.

Berry dry mass accumulation following the source-sink carbon allocation rules. Brix was calculated based on the ratio of berry dry weight to fresh weight. Total titratable acid was simulated based on thermaltime accumulation after veraison follwoing a negative exponetial curve.

A long-term phenology and yield monitoring trial using both two-cane and four-cane trained vertically shoot positioned (VSP) Sauvignon blanc vines was established in four vineyards in Marlborough, New Zealand in 2004, and was used for calibrating the berry module. Phenology, bunch number, berry mass, yield and meteorology records were collated. A multivariable mixed linear model was used to assess the relationship between various yield components and weather conditions. The critical periods for each yield component and weather factor were optimised based on the maximum likelihood returned from the mixed linear model. The optimised critical periods of temperature for all yield components occurred mainly before 50 % flowering either in the previous season (during inflorescence initiation) and the current season, indicating the importance of the pre-flowering period on yield formation. Out of all weather factors, maximum daily temperature had the largest effect on bunch number and overall yield and strongly influenced berry number and bunch mass. Rainfall near flowering time had a negative effect on berry mass and bunch mass, but post-flowering rainfall had a strong positive effect.

3.9.1 MetFactors

The final yield was calculated by vines per hectare, shoots per vine, bunches per shoot, flower number per bunch, berries per bunch,and mean berry weight. The effects of temperature and carbon status were or will be considered on bunches per shoot, flower number per bunch, berries per bunch,and mean berry weight.

BUN_TmaxIni = TmaxIni d Time

Where:

A function that accumulates values from child functions

Tmax = [Weather].MaxT

A function that accumulates values from child functions

Days = 1 ()

BUN_RadIni = RadIni d Time

Where:

A function that accumulates values from child functions

Rad = [Weather].Radn

A function that accumulates values from child functions

Days = 1 ()

BUN_CarbonIni = FDMIni d Time

Where:

A function that accumulates values from child functions

FDM = [Grapevine].Arbitrator.FDM

A function that accumulates values from child functions

Days = 1 ()

BEN_TmaxFlow = TmaxFlow d Time

Where:

A function that accumulates values from child functions

Tmax = [Weather].MaxT

A function that accumulates values from child functions

Days = 1 ()

BEN_TminFlow = TminFlow d Time

Where:

A function that accumulates values from child functions

Tmin = [Weather].MinT

A function that accumulates values from child functions

Days = 1 ()

BEN_TmaxIni = TmaxIni d Time

Where:

A function that accumulates values from child functions

Tmax = [Weather].MaxT

A function that accumulates values from child functions

Days = 1 ()

BEN_RainTotFlow = RainTotFlow d Constant

Where:

A function that accumulates values from child functions

RainTot = [Weather].Rain

Constant = 1 ()

BEN_CarbonFlow = FDMFlow d Time

Where:

A function that accumulates values from child functions

FDM = [Grapevine].Arbitrator.FDM

A function that accumulates values from child functions

Days = 1 ()

BM_TmaxFlow = TmaxFlow d Time

Where:

A function that accumulates values from child functions

Tmax = [Weather].MaxT

A function that accumulates values from child functions

Days = 1 ()

BM_RainTotFlow = RainTotFlow d Constant

Where:

A function that accumulates values from child functions

RainTot = [Weather].Rain

Constant = 1 ()

BM_RadFlow = RadFlow d Time

Where:

A function that accumulates values from child functions

Rad = [Weather].Radn

A function that accumulates values from child functions

Days = 1 ()

BM_RainTotVer = RainTotVer d Constant

Where:

A function that accumulates values from child functions

RainTot = [Weather].Rain

Constant = 1 ()

BM_CarbonFlow = FDMFlow d Time

Where:

A function that accumulates values from child functions

FDM = [Grapevine].Arbitrator.FDM

A function that accumulates values from child functions

Days = 1 ()

3.9.1.1 BUN_TmaxIni1

Before Ecodormancy

PreEventValue = 21 ()

On Ecodormancy the value is set to:

PostEventValue = [Berry].MetFactors.BUN_TmaxIni

3.9.1.2 BUN_RadIni1

Before Ecodormancy

PreEventValue = 21 ()

On Ecodormancy the value is set to:

PostEventValue = [Berry].MetFactors.BUN_RadIni

3.9.1.3 BUN_CarbonIni1

Before Ecodormancy

PreEventValue = 1 ()

On Ecodormancy the value is set to:

PostEventValue = [Berry].MetFactors.BUN_CarbonIni

3.9.1.4 BEN_TmaxIni1

Before Veraison

PreEventValue = 21 ()

On Veraison the value is set to:

PostEventValue = [Berry].MetFactors.BEN_TmaxIni

BEN_CarbonIni = FDMIni d Time

Where:

A function that accumulates values from child functions

FDM = [Grapevine].Arbitrator.FDM

A function that accumulates values from child functions

Days = 1 ()

3.9.1.5 BEN_CarbonIni1

Before Ecodormancy

PreEventValue = 1 ()

On Ecodormancy the value is set to:

PostEventValue = [Berry].MetFactors.BEN_CarbonIni

3.9.2 YieldComponent

3.9.2.1 BunchesPerShoot

here we use a trigger function for bunches per shoot as the value of mean bunch initiation temperature in the previous season

Numbers were guessed based on Eltom 2013 Fig. 7.7. y=0.0981*Ta + 0.35

Bunches per shoot is affected by the temperature and CHO during initiation, it is further affected by the CHO during bud burst. The later decided whether the potential initiated inflorescence can grow out successfully or been realized.

For simiplicity, right now, the CHO effect during bud burst is expressed as a function of number of buds retained.

Before BudBurst

PreEventValue = 1.7 ()

On BudBurst the value is set to:

3.9.2.2 PostEventValue

PostEventValue is calculated as the minimum of *MaximumFunc* and *Max*

Where:

MaximumFunc is calculated as the minimum of *MultiplyFunction* and *one*

Where:

Try to ignor the first year, which give TmaxIni 0

MultiplyFunction = MetFun x CarbonIni

Where:

MetFun = Constant p TmaxIni p RadIni

Where:

Constant = -0.7965 ()

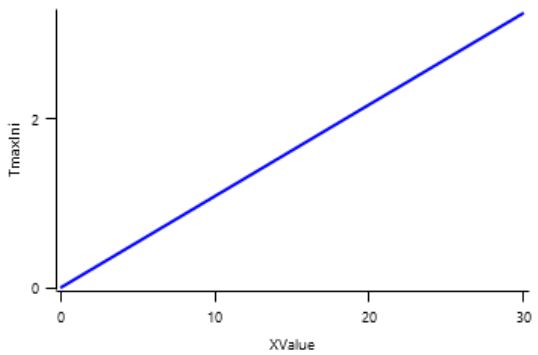
3.9.2.2.1 TmaxIni

IF [Berry].MetFactors.BUN_TmaxIni1 < Constant THEN

PreValue = 1.6 ()

ELSE

TmaxIni is calculated using linear interpolation.



X	Y
0	0
1	0.1084
10	1.084
20	2.168
30	3.252

XValue = [Berry].MetFactors.BUN_TmaxIni1

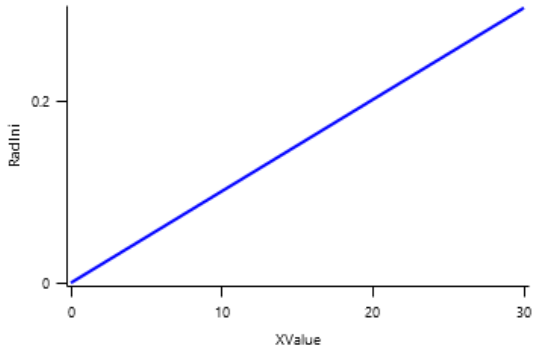
3.9.2.2.2 RadIni

IF [Berry].MetFactors.BUN_RadIni1 < Constant THEN

PreValue = 0.65 ()

ELSE

RadIni is calculated using linear interpolation.



X	Y
0	0
1	0.0101
10	0.101
20	0.202
30	0.303

XValue = [Berry].MetFactors.BUN_RadIni1

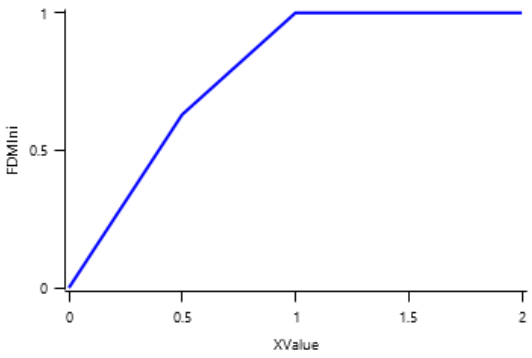
3.9.2.2.2.1 CarbonIni

IF [Berry].MetFactors.BUN_CarbonIni1 < Constant THEN

PreValue = 1 ()

ELSE

FDMIni is calculated using linear interpolation.



X	Y
0	0
0.5	0.63
1	1
2	1

XValue = [Berry].MetFactors.BUN_CarbonIni1

one = 1 ()

Max = 2.1 ()

3.9.2.3 BunchesPerVine

Trigger the calculation after the determination of bunches per shoot

Before Flowering

PreEventValue = 50 ()

On Flowering the value is set to:

PostEventValue is calculated as the minimum of *zero* and *NumberFunction*

Where:

zero = 10 ()

NumberFunction = [Berry].YieldComponent.BunchesPerShoot x ShootsNum

Where:

ShootsNum = [Grapevine].Structure.MainStemPopn d [Grapevine].Population

StemPopulation = [Grapevine].Structure.MainStemPopn

Population = [Grapevine].Population

BunchesPerShoot = [Berry].YieldComponent.BunchesPerShoot

3.9.2.4 BerryNum

here we use a trigger function for bunches per shoot as the value of mean flowering temperature Y0 is updating itself during the growing season, which would cause inconvience if we directly correlates the bunches per shoot and mean flowering temperature Y0.

Numbers were guessed based on Eltom 2013 Fig. 7.7. y=0.0981*Ta + 0.35

Bunches per shoot is affected by the temperature and CHO during initiation, it is further affected by the CHO during bud burst. The later decided whether the potential initiated inflorescence can grow out successfully or been realized.

For simiplicity, right now, the CHO effect during bud burst is expressed as a function of number of buds retained.

Before FruitSet

PreEventValue = 65 ()

On FruitSet the value is set to:

3.9.2.5 PostEventValue

PostEventValue is calculated as the minimum of *BerryNumFun* and *UpperLimit*

Where:

BerryNumFun is calculated as the minimum of *MultiplyFunction* and *zero*

Where:

MultiplyFunction = MetFun x FDMFlow x CarbonIni

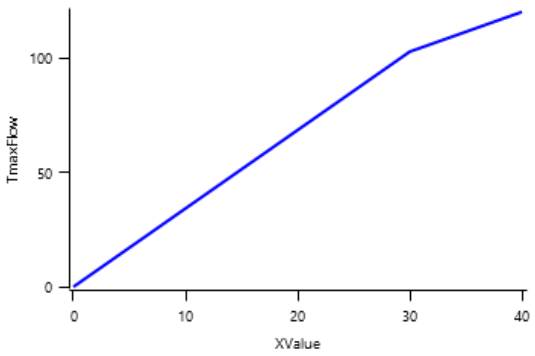
Where:

MetFun = Constant p TmaxFlow p TminFlow p TmaxIni

Where:

Constant = -67.4 ()

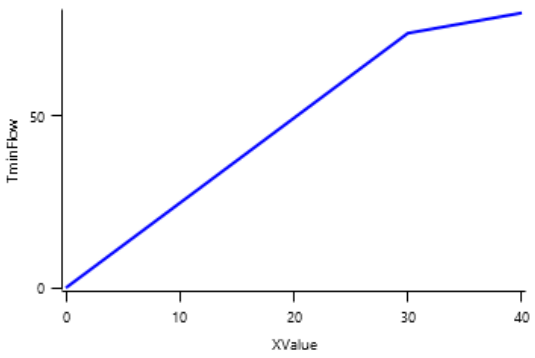
TmaxFlow is calculated using linear interpolation.



X	Y
0	0
1	3.4217
10	34.217
20	68.434
30	102.651
40	120

XValue = [Berry].MetFactors.BEN_TmaxFlow

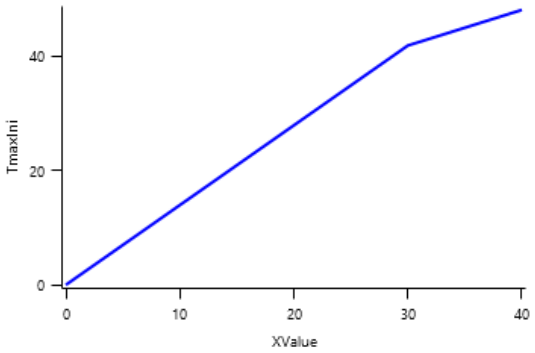
TminFlow is calculated using linear interpolation.



X	Y
0	0
1	2.4697
10	24.697
20	49.394
30	74.091
40	80

XValue = [Berry].MetFactors.BEN_TminFlow

TmaxIni is calculated using linear interpolation.



X	Y
0	0
1	1.3927
10	13.927
20	27.854
30	41.781
40	48

3.9.2.5.1 XValue

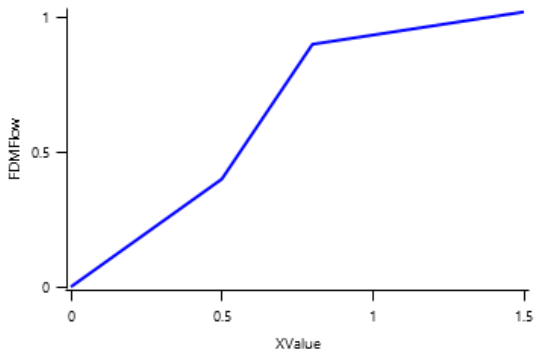
IF [Berry].MetFactors.BEN_TmaxIni1 < Constant THEN

GuessValue = 20 ()

ELSE

TmaxIni1 = [Berry].MetFactors.BEN_TmaxIni1

FDMFlow is calculated using linear interpolation.



X	Y
0	0
0.5	0.4
0.8	0.9
1.5	1.02

XValue = [Berry].MetFactors.BEN_CarbonFlow

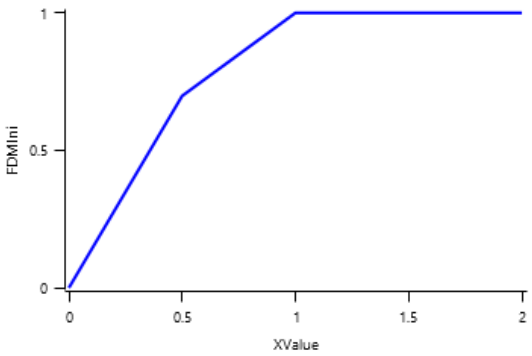
3.9.2.5.1.1 CarbonIni

IF [Berry].MetFactors.BEN_CarbonIni1 < Constant THEN

PreValue = 1 ()

ELSE

FDMIni is calculated using linear interpolation.



X	Y
0	0
0.5	0.698
1	1
2	1

XValue = [Berry].MetFactors.BEN_CarbonIni1

zero = 30 ()

UpperLimit = 150 ()

3.9.2.6 BerryMass

here we use a trigger function for bunches per shoot as the value of mean flowering temperature Y0 is updating itself during the growing season, which would cause inconvience if we directly correlates the bunches per shoot and mean flowering temperature Y0.

Numbers were guessed based on Eltom 2013 Fig. 7.7. y=0.0981*Ta + 0.35

Bunches per shoot is affected by the temperature and CHO during initiation, it is further affected by the CHO during bud burst. The later decided whether the potential initiated inflorescence can grow out successfully or been realized.

For simiplicity, right now, the CHO effect during bud burst is expressed as a function of number of buds retained.

Before Veraison

PreEventValue = 2.1 ()

On Veraison the value is set to:

3.9.2.7 PostEventValue

PostEventValue is calculated as the minimum of *BerryMassFun* and *UpperLimit*

Where:

BerryMassFun is calculated as the minimum of *LowerLimit* and *MultiplyFunction*

Where:

LowerLimit = 1 ()

MultiplyFunction = MetFun x FDMFlow

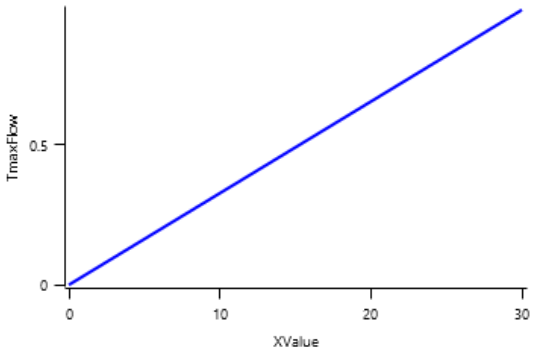
Where:

MetFun = Constant p TmaxFlow p RadFlow p RainTotFlow p RainTotVer

Where:

Constant = 0.92369 ()

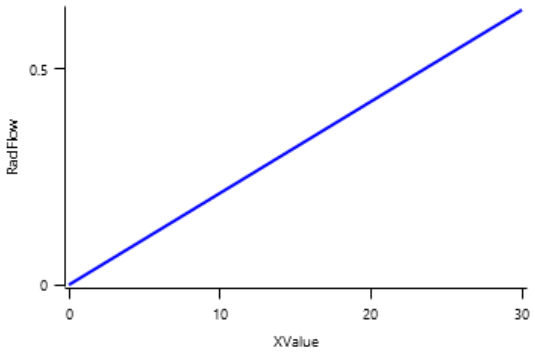
TmaxFlow is calculated using linear interpolation.



X	Y
0	0
1	0.0327134
10	0.327134
30	0.981402

XValue = [Berry].MetFactors.BM_TmaxFlow

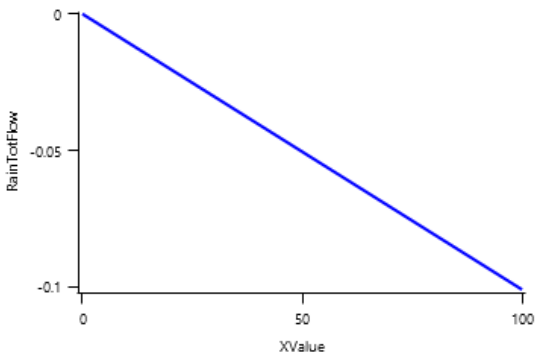
RadFlow is calculated using linear interpolation.



X	Y
0	0
1	0.0212212
10	0.212212
30	0.636636

XValue = [Berry].MetFactors.BM_RadFlow

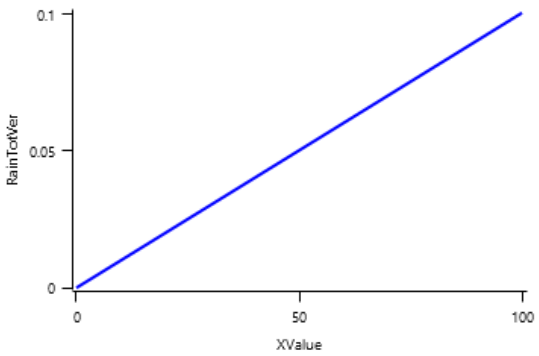
RainTotFlow is calculated using linear interpolation.



X	Y
0	0
1	-0.0010101
100	-0.10101

XValue = [Berry].MetFactors.BM_RainTotFlow

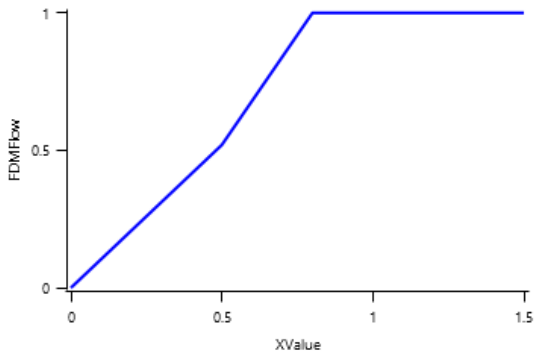
RainTotVer is calculated using linear interpolation.



X	Y
0	0
1	0.0010051
100	0.10051

XValue = [Berry].MetFactors.BM_RainTotVer

FDMFlow is calculated using linear interpolation.



X	Y
0	0
0.5	0.52
0.8	1
1.5	1

XValue = [Berry].MetFactors.BM_CarbonFlow

UpperLimit = 3 ()

NumberFunction = [Grapevine].Structure.MainStemPopn x [Berry].YieldComponent.BunchesPerShoot x [Berry].YieldComponent.BerryNum

StemPopulation = [Grapevine].Structure.MainStemPopn

BunchesPerShoot = [Berry].YieldComponent.BunchesPerShoot

BerriesPerBunch = [Berry].YieldComponent.BerryNum

SingleBerryFW = BetaGrowthFunction

Where:

3.9.2.8 BetaGrowthFunction

a beta growth function of the form y = Ymax * (1 + (te - t)/(te-tm))* (t/te)^(te/(te-tm))

XValue = [Berry].ThermalTimeAfterFlowering

Ymax = [Berry].YieldComponent.BerryMass

TotalBerryFW = [Berry].NumberFunction x [Berry].SingleBerryFW

BerryNumber = [Berry].NumberFunction

BerryFW = [Berry].SingleBerryFW

SingleBerryDW = BetaGrowthFunction

Where:

3.9.2.9 BetaGrowthFunction

a beta growth function of the form y = Ymax * (1 + (te - t)/(te-tm))* (t/te)^(te/(te-tm))

Ymax = 0.56 ()

XValue = [Berry].ThermalTimeAfterFlowering

WaterContent = Constant - AfterFlowering

Where:

Constant = 1 (g/g)

AfterFlowering has a non-zero value between Flowering and LeafFall calcualted as:

3.9.2.10 MinimumFunction

MinimumFunction is calculated as the minimum of *DivideFunction* and *Constant*

Where:

DivideFunction = [Berry].Live.Wt d [Berry].TotalBerryFW

BerryDW = [Berry].Live.Wt

TotalBerryFW = [Berry].TotalBerryFW

Constant = 0.88 (g/g)

DMDemandFunction = LessThanFunction x Constant

Where:

3.9.2.11 LessThanFunction

IF [Weather].DaysSinceWinterSolstice < HarvestTime THEN

AfterFruitSet has a non-zero value between Flowering and LeafFall calcualted as:

MaximumFunction is calculated as the minimum of *DeltaFunction* and *Constant*

Where:

DeltaFunction is the daily differential of

Integral = [Berry].NumberFunction x [Berry].SingleBerryDW

BerryNumber = [Berry].NumberFunction

BerryDM = [Berry].SingleBerryDW

Constant = 0 (g/m2/d)

ELSE

Zero = 0 (g/m2/d)

Constant = 1 (g/m2/d)

3.9.3 DMDemandPriorityFactors

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

Structural = 1.07758 (g/m2)

Metabolic = 0.001 (g/m2)

Storage = 0.001 (g/m2)

YieldPerVine = [Berry].TotalBerryFW d [Grapevine].Population d Constant

Where:

Constant = 1000 ()

TotalBerryFW = [Berry].TotalBerryFW

Population = [Grapevine].Population

3.9.4 Brix

Brix is calculated using specific values or functions for various growth phases. The function will use a value of zero for phases not specified below.

BeforeVeraison has a non-zero value between Flowering and Veraison calcualted as:

Constant = 5 ()

AfterVeraison has a non-zero value between Veraison and LeafFall calcualted as:

3.9.4.1 LessThanFunction

IF [Weather].DaysSinceWinterSolstice < HarvestTime THEN

MaximumFunction is calculated as the minimum of *DivideFunction* and *min*

Where:

DivideFunction = MaximumFunction d constant

Where:

MaximumFunction is calculated as the minimum of *SubtractFunction* and *zero*

Where:

SubtractFunction = constant - [Berry].WaterContent

Where:

constant = 0.944 ()

WaterContent = [Berry].WaterContent

zero = 0 ()

constant = 0.0082 ()

min = 5 ()

ELSE

Zero = 0 ()

3.9.5 TitratableAcid

TitratableAcid is calculated using specific values or functions for various growth phases. The function will use a value of zero for phases not specified below.

BeforeVeraison has a non-zero value between Flowering and Veraison calcualted as:

Constant = 0 ()

AfterVeraison has a non-zero value between Veraison and LeafFall calcualted as:

3.9.5.1 TitratableAcid

An exponential function

XValue = [Berry].ThermalTimeAfterVeraison

3.9.6 ThermalTimeAfterVeraison

Accumulates *ThermalTime* between [Start] and [End]

ThermalTime = [Phenology].ThermalTime

3.9.7 ThermalTimeAfterFlowering

Accumulates *ThermalTime* between [Start] and [End]

ThermalTime = [Phenology].ThermalTime

MaxNConcDailyGrowth = 0.01 ()

NFillingRate = 0 (g/m2/d)

MinimumNConc = 0 (g/g)

MaximumNConc = 0.0001 (g/g)

3.9.8 BiomassRemovalDefaults

This organ will respond to certain management actions by either removing some of its biomass from the system or transferring some of its biomass to the soil surface residues. The following table describes the default proportions of live and dead biomass that are transferred out of the simulation using "Removed" or to soil surface residue using "To Residue" for a range of management actions. The total percentage removed for live or dead must not exceed 100%. The difference between the total and 100% gives the biomass remaining on the plant. These can be changed during a simulation using a manager script.

Method	% Live Removed	% Dead Removed	% Live To Residue	% Dead To Residue
Harvest	0	0	0	0
Cut	0	0	0	0
Prune	0	0	0	0
Graze	0	0	0	0
Thin	0	0	0	0

MaximumPotentialGrainSize = 2.5 ()

DMConversionEfficiency = 0.98 (g/g)

RemobilisationCost = 0 (g/g)

CarbonConcentration = 0.4 (g/g)

MaintenanceRespirationFunction = Qm x ExponentialFunction

Where:

Qm = 0.001 (g/m^2)

3.9.8.1 ExponentialFunction

An exponential function

SubtractFunction = [Weather].MeanT - RefT

Where:

RefT = 20 (g/m^2)

Xvalue = [Weather].MeanT

3.10 Root

The generic root model calculates root growth in terms of rooting depth, biomass accumulation and subsequent root length density in each soil layer.

Root Growth

Roots grow downwards through the soil profile, with initial depth determined by sowing depth and the growth rate determined by RootFrontVelocity. The RootFrontVelocity is modified by multiplying it by the soil's XF value, which represents any resistance posed by the soil to root extension.

Root Depth Increase = RootFrontVelocity x XF_i x RootDepthStressFactor

where i is the index of the soil layer at the rooting front.

Root depth is also constrained by a maximum root depth.

Root length growth is calculated using the daily DM partitioned to roots and a specific root length. Root proliferation in layers is calculated using an approach similar to the generalised equimarginal criterion used in

economics. The uptake of water and N per unit root length is used to partition new root material into layers of higher 'return on investment'. For example, the Root Activity for water is calculated as

$RAw_i = -WaterUptake_i / LiveRootWt_i \times LayerThickness_i \times ProportionThroughLayer$

The amount of root mass partitioned to a layer is then proportional to root activity

$DMAllocated_i = TotalDMAllocated \times RAw_i / TotalRAw$

Dry Matter Demands

A daily DM demand is provided to the organ arbitrator and a DM supply returned. By default, 100% of the dry matter (DM) demanded from the root is structural. The daily loss of roots is calculated using a SenescenceRate function. All senesced material is automatically detached and added to the soil FOM.

Nitrogen Demands

The daily structural N demand from root is the product of total DM demand and the minimum N concentration. Any N above this is considered Storage and can be used for retranslocation and/or reallocation as the respective factors are set to values other then zero.

Nitrogen Uptake

Potential N uptake by the root system is calculated for each soil layer (i) that the roots have extended into. In each layer potential uptake is calculated as the product of the mineral nitrogen in the layer, a factor controlling the rate of extraction (kNO3 or kNH4), the concentration of N form (ppm), and a soil moisture factor (NUptakeSWFactor) which typically decreases as the soil dries.

$NO3\ uptake = NO3_i \times kNO3 \times NO3_{ppm,\ i} \times NUptakeSWFactor$

$NH4\ uptake = NH4_i \times kNH4 \times NH4_{ppm,\ i} \times NUptakeSWFactor$

As can be seen from the above equations, the values of kNO3 and kNH4 equate to the potential fraction of each mineral N pool which can be taken up per day for wet soil when that pool has a concentration of 1 ppm.

Nitrogen uptake demand is limited to the maximum daily potential uptake (MaxDailyNUptake) and the plant's N demand. The former provides a means to constrain N uptake to a maximum value observed in the field for the crop as a whole. The demand for soil N is then passed to the soil arbitrator which determines how much of the N uptake demand each plant instance will be allowed to take up.

Water Uptake

Potential water uptake by the root system is calculated for each soil layer that the roots have extended into. In each layer potential uptake is calculated as the product of the available water in the layer (water above LL limit) and a factor controlling the rate of extraction (KL). The values of both LL and KL are set in the soil interface and KL may be further modified by the crop via the KLModifier function.

$SW\ uptake = (SW_i - LL_i) \times KL_i \times KLModifier$

Note that this organs is parameterised to represents all the fine roots of the plant. The root organ is reponsible for uptake but can also supply both N and DM from its non-structural biomass. The Fibrous root organ is used for simulating biomass storage and remobilisation because the current root class can not handle the biomass retranslocation.

The dynamics of root biomass was based on the root length dynamics measurement done on mature concord vines in UC Davis by comos et al., 2005. both root growth rate and mortality rate have annual cycles.Maintenance was set to zero as it was parameterized insided the senescence rate. the maximum root biomass during the season is around 60 g per plant, estimated by lakso et al., 2008.

the initial weight of the root was increased as we start the simulation in February with no leaves.the start of simulation in February was for correctly simulating the budburst in the first year. the initial weight was given to ensure that the peak of the biomass can cycle around 30 g/m2.

3.10.1 InitialWt

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

Structural = 100 (g/m2)

Metabolic = 0 (g/m2)

Storage = 0 (g/m2)

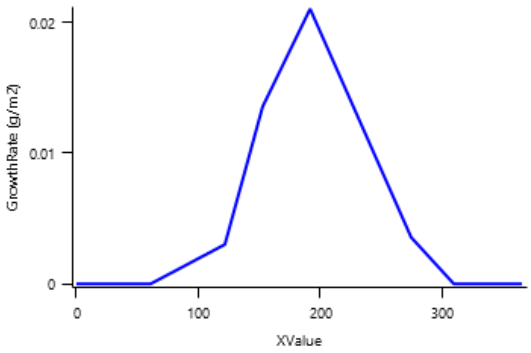
3.10.2 DMDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

Structural = [Root].Live.StructuralWt x GrowthRate x GrowthCostScale

Where:

GrowthRate is calculated using linear interpolation.



X	Y
0	0
61	0
122	0.0030125
153	0.013545
192	0.02102184
275	0.0035542
310	0
366	0

XValue = [Weather].DaysSinceWinterSolstice

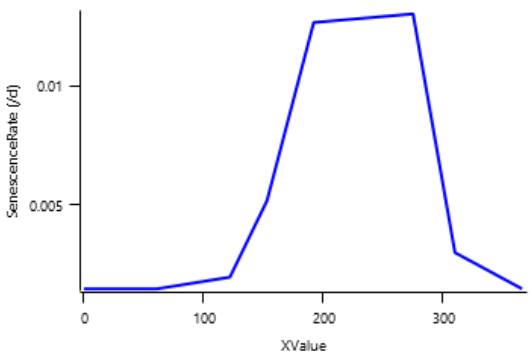
GrowthCostScale = 1.3 (g/m2)

wt = [Root].Live.StructuralWt

Metabolic = 0 (g/m2)

Storage = 0 (g/m2)

SenescenceRate is calculated using linear interpolation.



X	Y
0	0.001484
61	0.001484
122	0.001973
153	0.005178
192	0.01264
275	0.013
310	0.003
366	0.001484

XValue = [Weather].DaysSinceWinterSolstice

DMConversionEfficiency = 0.83 (g/g)

3.10.3 RootShape

DMRetranslocationFactor = 0 ()

DMReallocationFactor = 0 (/d)

NitrogenDemandSwitch has a non-zero value between BudBurst and LeafFall calcualted as:

Constant = 1 ()

MinimumNConc = 0 (g/g)

MaximumNConc = 0.0001 (g/g)

MaximumRootDepth = 1000000 (mm)

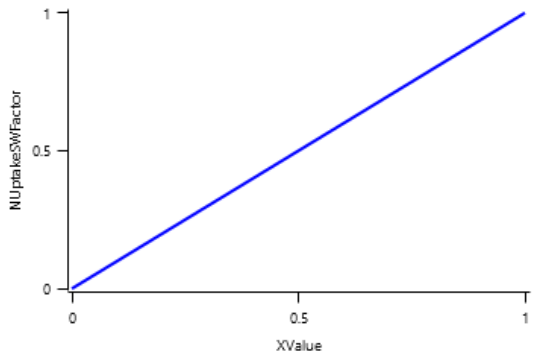
3.10.4 BiomassRemovalDefaults

This organ will respond to certain management actions by either removing some of its biomass from the system or transferring some of its biomass to the soil surface residues. The following table describes the default proportions of live and dead biomass that are transferred out of the simulation using "Removed" or to soil surface residue using "To Residue" for a range of management actions. The total percentage removed for live or dead must not exceed 100%. The difference between the total and 100% gives the biomass remaining on the plant. These can be changed during a simulation using a manager script.

Method	% Live Removed	% Dead Removed	% Live To Residue	% Dead To Residue
Harvest	0	0	0	0
Cut	0	0	0	0
Prune	0	0	0	0
Graze	0	0	0	0

RemobilisationCost = 0 ()

NUptakeSWFactor is calculated using linear interpolation.



X	Y
0	0
1	1

XValue = [Root].RWC

SpecificRootLength = 500 (m/g)

RootFrontVelocity = 20 (mm/d)

KNO3 = 0.02 (/d/ppm)

KNH4 = 0.003 (/d/ppm)

MaxDailyNUptake = 6 (kg N/ha/d)

CarbonConcentration = 0.4 (g/g)

3.10.5 NDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

Structural = [Root].minimumNconc x [Root].potentialDMAllocation.Structural

MinNconc = [Root].minimumNconc

PotentialDMAllocation = [Root].potentialDMAllocation.Structural

Metabolic = MetabolicNconc x [Root].potentialDMAllocation.Structural

Where:

MetabolicNconc = [Root].criticalNConc - [Root].minimumNconc

CritNconc = [Root].criticalNConc

MinNconc = [Root].minimumNconc

PotentialDMAllocation = [Root].potentialDMAllocation.Structural

3.10.5.1 Storage

The partitioning of daily N supply to storage N attempts to bring the organ's N content to the maximum concentration.

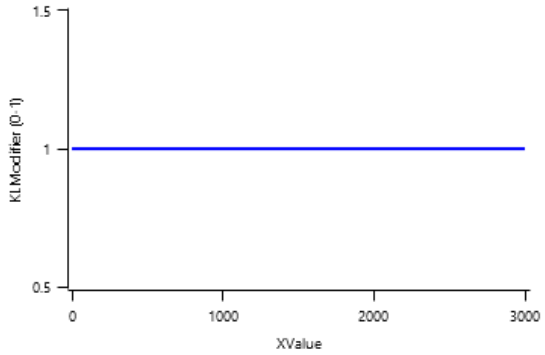
Storage = [Root].maximumNconc × ([Root].Live.Wt + potentialAllocationWt) - [Root].Live.N

The demand for storage N is further reduced by a factor specified by the [Root].NitrogenDemandSwitch.

CriticalNConc = [Root].MinimumNConc

This is important in SLURP as it is set for each species to represent their differences in rooting patterns between crop species

KLModifier is calculated using linear interpolation.



X	Y
0	1
300	1
600	1
900	1
1200	1
1500	1
1800	1
2100	1
2400	1
2700	1
3000	1

XValue = [Root].LayerMidPointDepth

RootDepthStressFactor = 1 (0-1)

3.10.6 DMDemandPriorityFactors

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

Structural = 1.6 (g/m2)

Metabolic = 0 (g/m2)

Storage = 0 (g/m2)

MaintenanceRespirationFunction = Qm x ExponentialFunction

Where:

Qm = 0.01 (/d)

3.10.6.1 ExponentialFunction

An exponential function

SubtractFunction = [Weather].MeanT - RefT

Where:

RefT = 20 (/d)

Xvalue = [Weather].MeanT

MortalityRate = 0 ()

3.11 Arbitrator

The Arbitrator class determines the allocation of dry matter (DM) and Nitrogen between each of the organs in the crop model. Each organ can have up to three different pools of biomass:

- **Structural biomass** which is essential for growth and remains within the organ once it is allocated there.
- **Metabolic biomass** which generally remains within an organ but is able to be re-allocated when the organ senesces and may be retranslocated when demand is high relative to supply.
- **Storage biomass** which is partitioned to organs when supply is high relative to demand and is available for retranslocation to other organs whenever supply from uptake, fixation, or re-allocation is lower than demand.

The process followed for biomass arbitration is shown in Figure 3. Arbitration calculations are triggered by a series of events (shown below) that are raised every day. For these calculations, at each step the Arbitrator exchange information with each organ, so the basic computations of demand and supply are done at the organ level, using their specific parameters.

1. **doPotentialPlantGrowth**. When this event occurs, each organ class executes code to determine their potential growth, biomass supplies and demands. In addition to demands for structural, non-structural and metabolic biomass (DM and N) each organ may have the following biomass supplies:
2. **Fixation supply**. From photosynthesis (DM) or symbiotic fixation (N)
3. **Uptake supply**. Typically uptake of N from the soil by the roots but could also be uptake by other organs (eg foliage application of N).
4. **Retranslocation supply**. Storage biomass that may be moved from organs to meet demands of other organs.
5. **Reallocation supply**. Biomass that can be moved from senescing organs to meet the demands of other organs.
6. **doPotentialPlantPartitioning.** On this event the Arbitrator first executes the DoDMSetup() method to gather the DM supplies and demands from each organ, these values are computed at the organ level. It then executes the DoPotentialDMAllocation() method which works out how much biomass each organ would be allocated assuming N supply is not limiting and sends these allocations to the organs. Each organ then uses their potential DM allocation to determine their N demand (how much N is needed to produce that much DM) and the arbitrator calls DoNSetup() to gather the N supplies and demands from each organ and begin N arbitration. Firstly DoNReallocation() is called to redistribute N that the plant has available from senescing organs. After this step any unmet N demand is considered as plant demand for N uptake from the soil (N Uptake Demand).
7. **doNutrientArbitration.** When this event occurs, the soil arbitrator gets the N uptake demands from each plant (where multiple plants are growing in competition) and their potential uptake from the soil and determines how much of their demand that the soil is able to provide. This value is then passed back to each plant instance as their Nuptake and doNUptakeAllocation() is called to distribute this N between organs.
8. **doActualPlantPartitioning.** On this event the arbitrator call DoNRetranslocation() and DoNFixation() to satisfy any unmet N demands from these sources. Finally, DoActualDMAllocation is called where DM allocations to each organ are reduced if the N allocation is insufficient to achieve the organs minimum N concentration and final allocations are sent to organs.

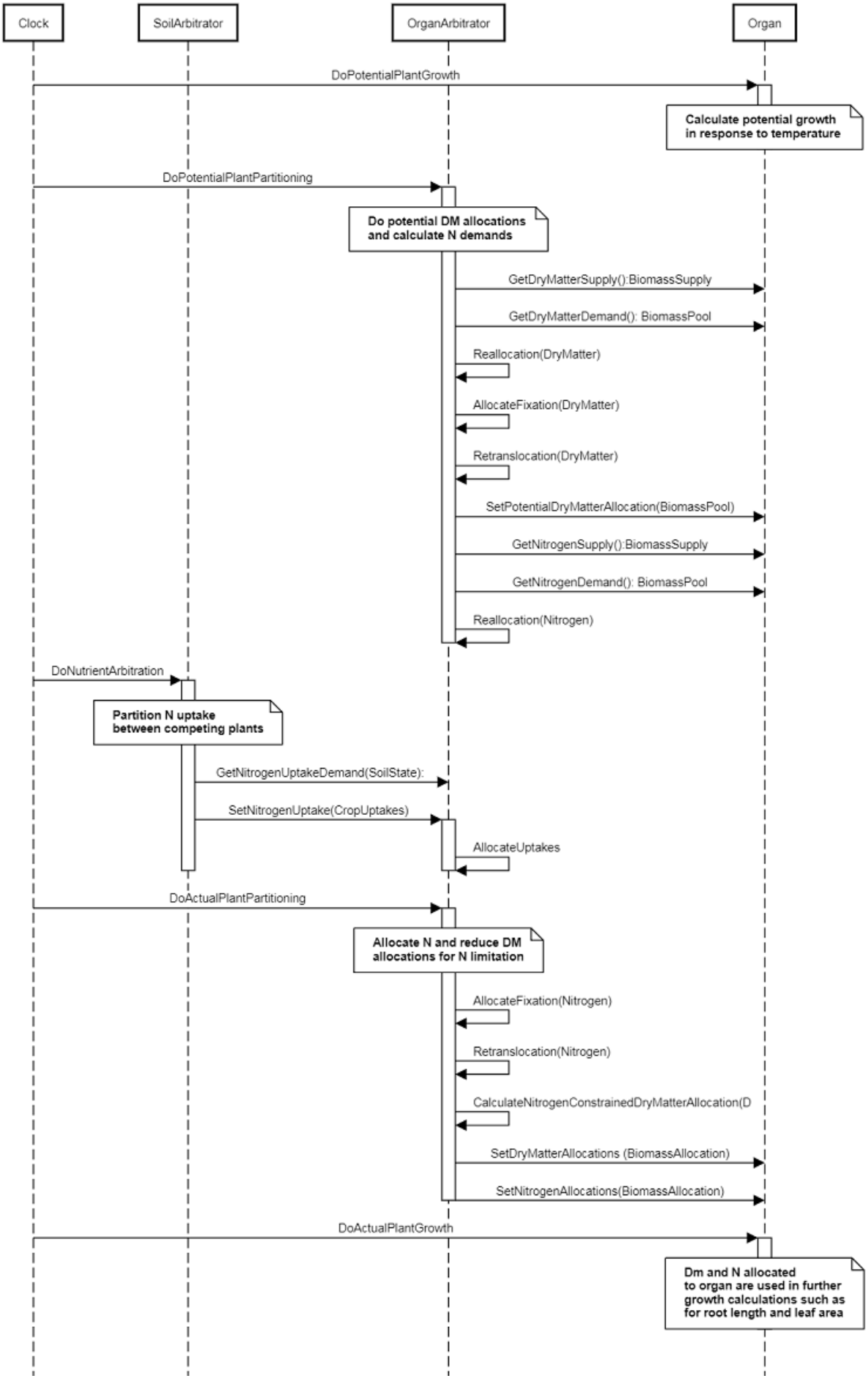


Figure 3: Schematic showing the procedure for arbitration of biomass partitioning. Pink boxes represent events that occur every day and their numbering shows the order of calculations. Blue boxes represent the methods that are called when these events occur. Orange boxes contain properties that make up the organ/arbitrator interface. Green boxes are organ specific properties.

3.12 Varieties

Chardonnay, Merlot, PinotGris, PinotNoir, SauvignonBlanc

3.12.1 SauvignonBlanc

SauvignonBlanc overrides the following properties:

3.12.2 Chardonnay

Chardonnay overrides the following properties:

[Phenology].EndoDormancy.Target.BaseTarget.FixedValue = 52.0647
[Phenology].EndoDormancy.Progression.Response.Xo.FixedValue = 10.98
[Phenology].EndoDormancy.Progression.Response.b.FixedValue = -25.5611
[Phenology].Budding.Target.FixedValue = 906.4065 [Phenology].Budding.Progression.Response.X = 0,0.27668,10.27668,20.27668 [Phenology].ThermalTime.Response.MinTemp = 0.239618
[Phenology].ThermalTime.Response.OptTemp = 27.149 [Phenology].ThermalTime.Response.MaxTemp = 48.75 [Phenology].Flowering.Target.FixedValue = 45.41293
[Phenology].BerryDevelopment.Target.Base.FixedValue = 47.77695

3.12.3 Merlot

Merlot overrides the following properties:

[Phenology].EndoDormancy.Target.BaseTarget.FixedValue = 88.74292
[Phenology].EndoDormancy.Progression.Response.Xo.FixedValue = 15.07995
[Phenology].EndoDormancy.Progression.Response.b.FixedValue = -1.27593
[Phenology].Budding.Target.FixedValue = 767.6113 [Phenology].Budding.Progression.Response.X = 0,0.353281,10.353281,20.353281 [Phenology].ThermalTime.Response.MinTemp = 2.51
[Phenology].ThermalTime.Response.OptTemp = 22.245 [Phenology].ThermalTime.Response.MaxTemp = 40.316 [Phenology].Flowering.Target.FixedValue = 42.007
[Phenology].BerryDevelopment.Target.Base.FixedValue = 50.84

3.12.4 PinotNoir

PinotNoir overrides the following properties:

[Phenology].EndoDormancy.Target.BaseTarget.FixedValue = 72.1
[Phenology].EndoDormancy.Progression.Response.Xo.FixedValue =7.95
[Phenology].EndoDormancy.Progression.Response.b.FixedValue = -45.3083
[Phenology].Budding.Target.FixedValue = 627.0382 [Phenology].Budding.Progression.Response.X = 0,0.244,10.244,20.244 [Phenology].ThermalTime.Response.MinTemp = 1.514
[Phenology].ThermalTime.Response.OptTemp = 26.42341 [Phenology].ThermalTime.Response.MaxTemp = 36.87527 [Phenology].Flowering.Target.FixedValue = 27.5666
[Phenology].BerryDevelopment.Target.Base.FixedValue = 29.75368

3.12.5 PinotGris

PinotGris overrides the following properties:

[Phenology].EndoDormancy.Target.BaseTarget.FixedValue = 74.163
[Phenology].EndoDormancy.Progression.Response.Xo.FixedValue = 12.53467
[Phenology].EndoDormancy.Progression.Response.b.FixedValue = -66.5092
[Phenology].Budding.Target.FixedValue = 519.1688 [Phenology].Budding.Progression.Response.X = 0, 1.93454,11.93454,21.93454 [Phenology].ThermalTime.Response.MinTemp = 1.173588
[Phenology].ThermalTime.Response.OptTemp = 25.6193 [Phenology].ThermalTime.Response.MaxTemp = 35.65306 [Phenology].Flowering.Target.FixedValue = 31.13788
[Phenology].BerryDevelopment.Target.Base.FixedValue = 32.965

A replacements model

The Grapevine model is constructed from the following list of software components. Details of the implementation and model parameterisation are provided in the following sections.

List of Plant Model Components.

Component Name	Component Type
Phenology	Models.PMF.Phen.Phenology
Structure	Models.PMF.Struct.Structure
Leaf	Models.PMF.Organs.Leaf
Shoot	Models.PMF.Organs.GenericOrgan
Cordon	Models.PMF.Organs.GenericOrgan
Trunk	Models.PMF.Organs.GenericOrgan
StructuralRoot	Models.PMF.Organs.GenericOrgan
Berry	Models.PMF.Organs.ReproductiveOrgan
Root	Models.PMF.Organs.Root
MortalityRate	Models.Functions.Constant
Arbitrator	Models.PMF.OrganArbitrator

3.12.6 Phenology

Grapevine's phenological development is simulated as the progression through a series of developmental phases, each bound by distinct growth *stages*.

3.12.7 Phases

List of stages and phases used in the simulation of crop phenological development

Phase Number	Phase Name	Initial Stage	Final Stage
1	EndoDormancy	Endodormancy	Ecodormancy
2	Budding	Ecodormancy	BudBurst
3	Flowering	BudBurst	Flowering
4	FruitSet	Flowering	FruitSet
5	BerryDevelopment	FruitSet	Veraison
6	CanopySenescence	Veraison	LeafFall
7	Dormancy	LeafFall	EndoDormancy

3.12.7.1 EndoDormancy Phase

The *EndoDormancy* phase goes from the *Endodormancy* stage to the *Ecodormancy* stage.

The *Target* for completion is calculated as

Target = BaseTarget - [Phenology].AccChillBefPrune

Where:

BaseTarget = 52.6 (oD)

AccChillBefPrune = [Phenology].AccChillBefPrune

Progression through the *EndoDormancy* phase is calculated daily and accumulated until the *Target* is reached.

Progression is the Average of sub-daily values from a Models.Functions.SigmoidFunction.

Firstly hourly estimates of air temperature (Ta) are interpolated from Tmax, Tmin and daylength (d) usig the method of Goudriaan et al., 1994. During sunlight hours Ta is calculated each hour using a sinusoidal curve fitted to Tmin and Tmax . After sunset Ta is calculated as an exponential decline from Ta at sunset to the Tmin at sunrise the next day. The hour (Th) of sunrise is calculated as Th = 12 - d/2 and Ta is assumed to equal Tmin at this time. Tmax is reached when Th equals 13.5.

Each of the interpolated air temperatures are then passed into the following Response and the Average taken to give daily Progression

Response is calcualted using a sigmoid function of the form y = Xmax * 1 / 1 + $e^{-(Xvalue - Xo) / b}$. \n Ymax is calculated as

Ymax = 1 ()

Xo is calculated as

Xo = 12.42 ()

b is calculated as

b = -3.865 ()

Xvalue is calculated as

Unknown child name: XValue

3.12.7.2 Budding Phase

The *Budding* phase goes from the *Ecodormancy* stage to the *BudBurst* stage.

The *Target* for completion is calculated as

Target = 1011 (oD)

Progression through the *Budding* phase is calculated daily and accumulated until the *Target* is reached.

Progression is the Average of sub-daily values from a Models.Functions.XYPairs.

Firstly hourly estimates of air temperature (Ta) are interpolated from Tmax, Tmin and daylength (d) usig the method of Goudriaan et al., 1994. During sunlight hours Ta is calculated each hour using a sinusoidal curve fitted to Tmin and Tmax . After sunset Ta is calculated as an exponential decline from Ta at sunset to the Tmin at sunrise the next day. The hour (Th) of sunrise is calculated as Th = 12 - d/2 and Ta is assumed to equal Tmin at this time. Tmax is reached when Th equals 13.5.

Each of the interpolated air temperatures are then passed into the following Response and the Average taken to give daily Progression

Response is calculated from an XY matrix (graphed below) which returns a value for Y interpolated from the Xvalue provided.

Graph

3.12.7.3 Flowering Phase

The *Flowering* phase goes from the *BudBurst* stage to the *Flowering* stage.

The *Target* for completion is calculated as

Target = 21.1 (oD)

Progression through the *Flowering* phase is calculated daily and accumulated until the *Target* is reached.

Progression = [Phenology].ThermalTime

3.12.7.4 FruitSet Phase

The *FruitSet* phase goes from the *Flowering* stage to the *FruitSet* stage.

The *Target* for completion is calculated as

Target = 10 (oD)

Progression through the *FruitSet* phase is calculated daily and accumulated until the *Target* is reached.

Progression = [Phenology].ThermalTime

Fruit set (or the proportion of flowers that are retained as berries) represents a change-over from the static condition of the fully developed flower to the rapidly growing condition of the young fruit (Coombe, 1962).

In practice the criteria to define this stage is a bit blur. In current, we set the duration from 50% flowering to fruit set is 10 thermal days based on our field experience and the method for determining berry number by meteorology factors. Fruit set phase is used as the time when the berry number is determined in this model.

3.12.7.5 BerryDevelopment Phase

The *BerryDevelopment* phase goes from the *FruitSet* stage to the *Veraison* stage.

The *Target* for completion is calculated as

Target = 23 (oD)

Progression through the *BerryDevelopment* phase is calculated daily and accumulated until the *Target* is reached.

Progression = [Phenology].ThermalTime

3.12.7.6 CanopySenescence Phase

The *CanopySenescence* phase goes from the *Veraison* stage to the *LeafFall stage* which occurs when all leaves have fully senesced.

ThermalTime = [Phenology].ThermalTime

3.12.7.7 Dormancy

When LeafFall is reached phenology is rewound to EndoDormancy

Phenology is also rewound to the EndoDormant stage by a prun event

3.12.7.8 AccChillBefPrune

Accumulates *ChillBeforePrune* between [Start] and [End]

3.12.7.8.1 ChillBeforePrune

IF [Weather].DaysSinceWinterSolstice < PruningTime THEN

3.12.7.8.2 ChillBeforeAutumn

IF autumn < [Weather].DaysSinceWinterSolstice THEN

3.12.7.8.3 ChillBelowCriticalPhotoperiod

IF PhotoperiodFunction < CriticalPhotoPeriod THEN

ChillingUnit = [Phenology].EndoDormancy.Progression

ELSE

Zero = 0 ()

ELSE

Zero = 0 ()

ELSE

Zero = 0 ()

3.12.8 Structure

The structure model simulates morphological development of the plant to inform the Leaf class when and how many leaves and branches appear and provides an estimate of height.

3.12.8.1 Plant and Main-Stem Population

The *Plant.Population* is set at sowing with information sent from a manager script in the Sow method. The *PrimaryBudNumber* is also sent with the Sow method and the main-stem population (*MainStemPopn*) for Grapevine is calculated as:

MainStemPopn = *Plant.Population* x *PrimaryBudNumber*

Primary bud number is > 1 for crops like potato and grape vine where there are more than one main-stem per plant

3.12.8.2 Main-Stem leaf appearance

Each day the number of main-stem leaf tips appeared (*LeafTipsAppeared*) is calculated as:

LeafTipsAppeared += *DeltaTips*

Where *DeltaTips* is calculated as:

DeltaTips = *ThermalTime/Phyllochron*

Where *Phyllochron* is the thermal time duration between the appearance of leaf tips given by:

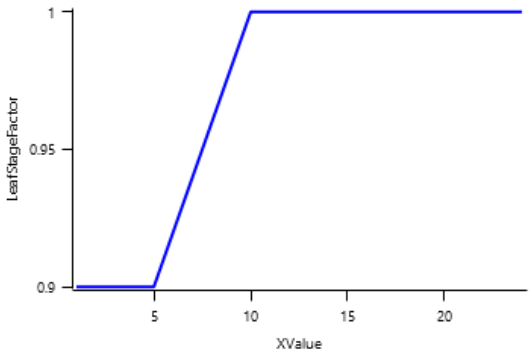
Phyllochron has a non-zero value between BudBurst and Veraison calcualted as:

Phyllochron = MaximumPhyllochron x LeafStageFactor

Where:

MaximumPhyllochron = 1.823 ()

LeafStageFactor is calculated using linear interpolation.



X	Y
1	0.9
5	0.9
10	1
24	1

XValue = [Leaf].AppearedCohortNo

ThermalTime is given by:

ThermalTime = [Phenology].ThermalTime

LeafTipsAppeared continues to increase until *FinalLeafNumber* is reached where *FinalLeafNumber* is calculated as:

FinalLeafNumber = 25 ()

3.12.8.3 Branching and Branch Mortality

The total population of stems (*TotalStemPopn*) is calculated as:

TotalStemPopn = MainStemPopn + NewBranches - NewlyDeadBranches

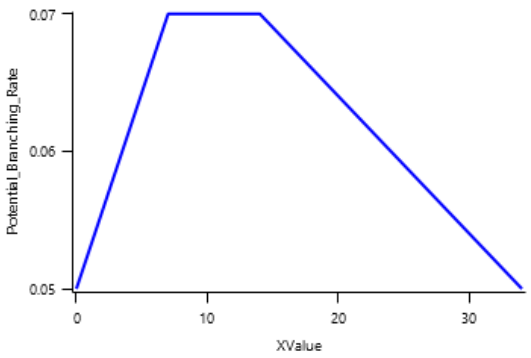
Where *NewBranches = MainStemPopn* x *BranchingRate*

and *BranchingRate* is given by:

BranchingRate = Potential_Branching_Rate x DensityEffect x CanopySizeEffect

Where:

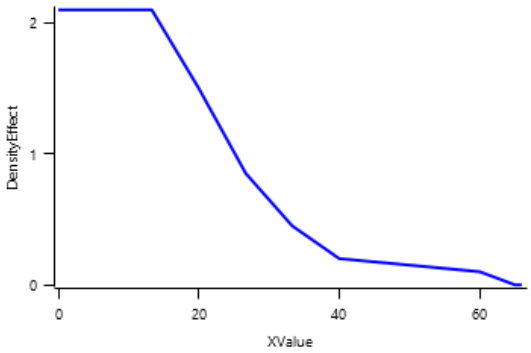
Potential_Branching_Rate is calculated using linear interpolation.



X	Y
0	0.05
7	0.07
8	0.07
13	0.07
14	0.07
34	0.05

XValue = [Structure].LeafTipsAppeared

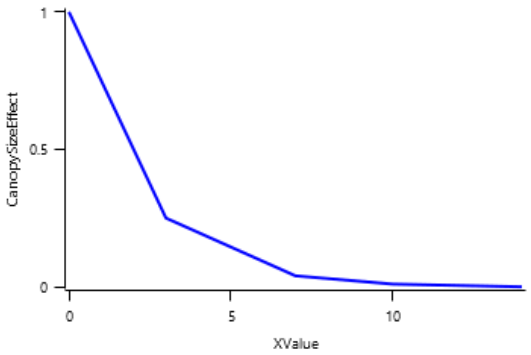
DensityEffect is calculated using linear interpolation.



X	Y
0	2.1
13.3	2.1
20	1.5
26.7	0.85
33.3	0.45
40	0.2
60	0.1
65	0
66	0

XValue = [Structure].BudsPerMeterRow.FixedValue

CanopySizeEffect is calculated using linear interpolation.



X	Y
0	1
3	0.25
7	0.04
10	0.01
14	0

XValue = [Leaf].LAI

NewlyDeadBranches is calcualted as:

NewlyDeadBranches = (*TotalStemPopn - MainStemPopn*) x *BranchMortality*

where *BranchMortality* is given by:

BranchMortality = 0 ()

3.12.8.4 Height

The Height of the crop is calculated by the *HeightModel*:

HeightModel = BaseHeight p ShootHeight

Where:

BaseHeight = 1100 ()

3.12.8.5 ShootHeight

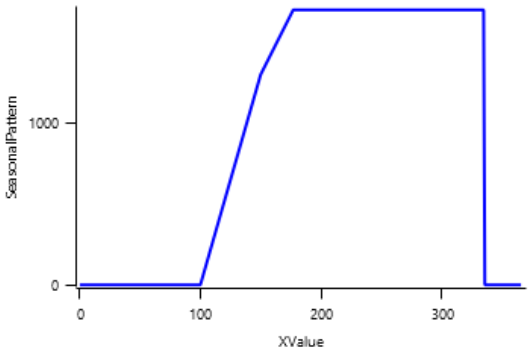
ShootHeight is calculated as the minimum of *ShootLength* and *TrimShootHeight*

Where:

ShootLength = SeasonalPattern x DensityEffect

Where:

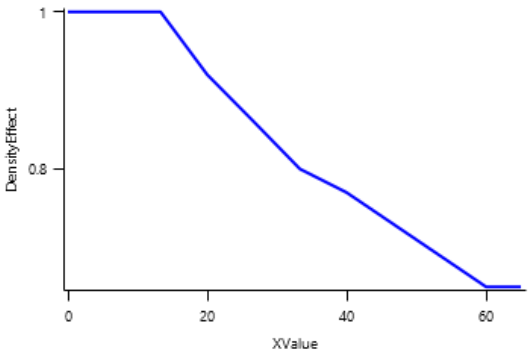
SeasonalPattern is calculated using linear interpolation.



X	Y
0	0
100	0
150	1300
177	1700
335	1700
336	0
366	0

XValue = [Weather].DaysSinceWinterSolstice

DensityEffect is calculated using linear interpolation.



X	Y
0	1
13.3	1
20	0.92
26.7	0.86
33.3	0.8
40	0.77
60	0.65
65	0.65

XValue = [Structure].BudsPerMeterRow.FixedValue

TrimShootHeight = 1100 ()

The structure module defines the bud number per vine, rate of main stem primordia initiation rate, phyllochron (thermal day interval between two successive leaf appearance, 1.823 td in this model), branching rate and branching mortality, final leaf number and canopy height. Phyllochron was defined by a phase look up function in combination with a linear interpolation function. The phase look up function defines when new leaf appearance will happen, which is currently defined between budburst and véraison. The linear interpolation function defines how phyllochron changes with leaf rank based on the data presented in (Greer and Weston, 2010). Branching rate was calculated as the product of three components: potential branching rate, the effect of leaf area per vine, and the effects of retained bud number per meter row

3.12.8.6 BudNumber

Each time BudBurst occurs bud number on each main-stem is set to

*FractionOfBudBurst * SowingData.BudNumber* (from manager at establishment)

FractionOfBudBurst = BudNumberEffect x CordonDiameterEffect

Where:

BudNumberEffect is calcualted using a sigmoid function of the form y = Xmax * 1 / 1 + $e^{-(Xvalue - Xo) / b}$. \n
Ymax is calculated as

Ymax = 1.4 ()

Xo is calculated as

Xo = 100 ()

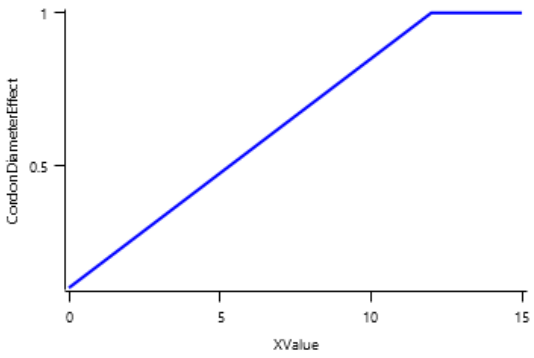
b is calculated as

b = -54.8977 ()

Xvalue is calculated as

XValue = [Structure].PrimaryBudNo

CordonDiameterEffect is calculated using linear interpolation.



X	Y
0	0.1
12	1
15	1

XValue = [Grapevine].Cordon.CordonDiameter

BudsPerMeterRow = 25 ()

3.12.9 Leaf

The leaves are modelled as a set of leaf cohorts and the properties of each of these cohorts are summed to give overall values for the leaf organ. A cohort represents all the leaves of a given main- stem node position including all of the branch leaves appearing at the same time as the given main-stem leaf (Lawless et al., 2005). The number of leaves in each cohort is the product of the number of plants per m^2 and the number of branches per plant. The *Structure* class models the appearance of main-stem leaves and branches. Once cohorts are initiated the *Leaf* class models the area and biomass dynamics of each. It is assumed all the leaves in each cohort have the same size and biomass properties. The modelling of the status and function of individual cohorts is delegated to *LeafCohort* classes.

3.12.9.1 Dry Matter Fixation

The most important DM supply from leaf is the photosynthetic fixation supply. Radiation interception is calculated from LAI using an extinction coefficient of:

ExtinctionCoeff = 0.6 ()

3.12.9.2 Photosynthesis

Biomass fixation is modelled as the product of intercepted radiation and its conversion efficiency, the radiation use efficiency (RUE) (Monteith et al., 1977). This approach simulates net photosynthesis rather than providing separate estimates of growth and respiration. The potential photosynthesis calculated using RUE is then adjusted according to stress factors, these account for plant nutrition (FN), air temperature (FT), vapour pressure deficit (FVPD), water supply (FW) and atmospheric CO_2 concentration (FCO2). NOTE: RUE in this model is expressed as g/MJ for a whole plant basis, including both above and below ground growth.

3.12.9.2.1 RUE

RUE is calculated using specific values or functions for various growth phases. The function will use a value of zero for phases not specified below.

Vegetative has a non-zero value between BudBurst and Flowering calcualted as:

Constant = 0.7 (g/MJ)

Reproductive has a non-zero value between Flowering and LeafFall calcualted as:

Constant = 0.8 (g/MJ)

3.12.9.2.2 FCO2

This model calculates the CO_2 impact on RUE using the approach of Reyenga et al., 1999.

For C3 plants,

$F_{CO2} = (CO_2 - CP) \times (350 + 2 \times CP)/(CO_2 + 2 \times CP) \times (350 - CP)$

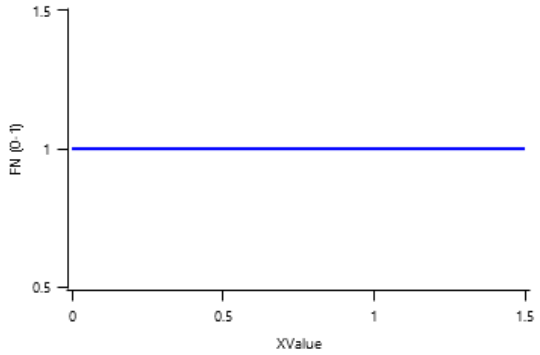
where CP, is the compensation point calculated from daily average temperature (T) as

*CP = (163.0 - T) / (5.0 - 0.1 * T)*

For C4 plants,

$F_{CO2} = 0.000143 * CO_2 + 0.95$

FN is calculated using linear interpolation.

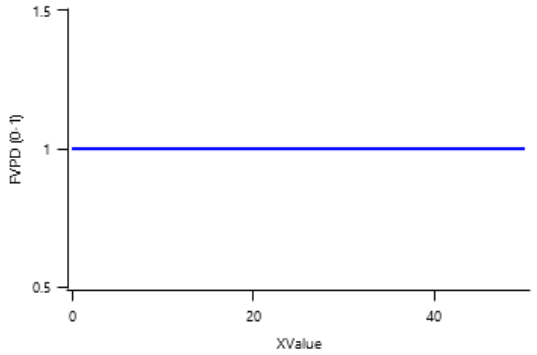


X	Y
0	1
1	1
1.5	1

XValue = [Leaf].Fn

3.12.9.2.3 FT

FVPD is calculated using linear interpolation.



X	Y
0	1
10	1
50	1

XValue = [Leaf].Photosynthesis.VPD

RadnInt = [Leaf].RadiationIntercepted

FW = 1 (0-1)

The grapevine model used the phytomer-based Leaf organ class. It predicted the appearance, expansion and senescence of cohorts of leaves at each position on the primary shoot and estimated how many leaves were present in each cohort based on branching rates. Branch leaves were treated the same as main-stem leaves appeared at the same time.

The maximum area of each cohort was currently set as a function of relative node position in respect to the final leaf number. The growth duration, lag duration, senescence duration and specific minimum and maximum leaf area were parameterised based on our field experiment, see Datasets used for Model calibration and validation. For correctly modelling leaf senescence and the dynamics of leaf area, a photoperiod acceleration effect was added. The photoperiod acceleration effect included and reflected the effects of the time of harvest on leaf senescence as well, as we observed that leaves would stay green much longer if the fruit were not harvested, See Supplementary Method 1 leaf module. Other leaf properties were parameterised to ensure no nitrogen or water stress on leaf expansion in the current grapevine model as the water and nitrogen dynamic part require further calibration.

The potential total DM demand of a leaf was calculated based upon the delta leaf area increase per day (constrained by stress) times the mean of maximum and minimum specific leaf area given in UI. The total DM demand was then divided into structural DM demand and Metabolic DM demand based on the fraction of structural DM in total leaf weight. The priority factor q for leaf structural and metabolic demand was set to a high

value (1.6) as it is the immediate source of photosynthetic and has the highest priority early in the season (Buwalda, 1991; Lakso et al., 2008). Non-structural reserves were not considered in leaf current model. At leaf senescence, 50% of the total leaf DM was reallocated to other organs following the carbon allocation method, the other half was lost due to respiration to generate the energy required for the degradation and export metabolism in the face of declining photosynthesis (Keller, 2015).

The dynamics of seasonal canopy width and canopy depth were defined in the leaf class based on our field observation.

ThermalTime = [Structure].ThermalTime

Area = 1000

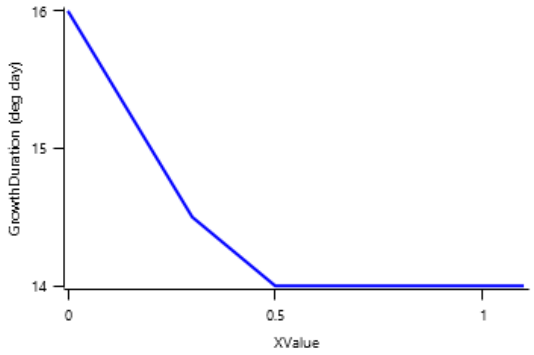
Area = 0

Area = 0

3.12.9.3 Potential Leaf Area index

Leaf area index is calculated as the sum of the area of each cohort of leaves The appearance of a new cohort of leaves occurs each time Structure.LeafTipsAppeared increases by one. From tip appearance the area of each cohort will increase for a certian number of degree days defined by the *GrowthDuration*

GrowthDuration is calculated using linear interpolation.



X	Y
0	16
0.3	14.5
0.5	14
1	14
1.1	14

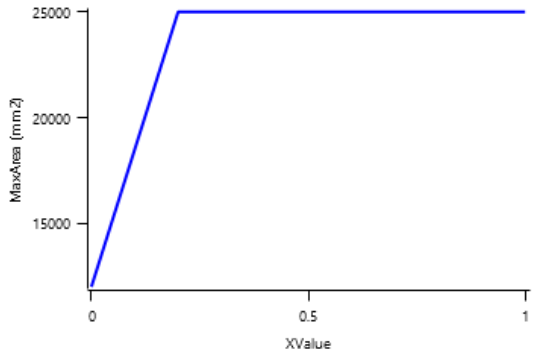
XValue = [Structure].RelativeNodeApperance

If no stress occurs the leaves will reach a Maximum area (*MaxArea*) at the end of the *GrowthDuration*. The *MaxArea* is defined by:

MaxArea = MaxArea x Constant

Where:

MaxArea is calculated using linear interpolation.



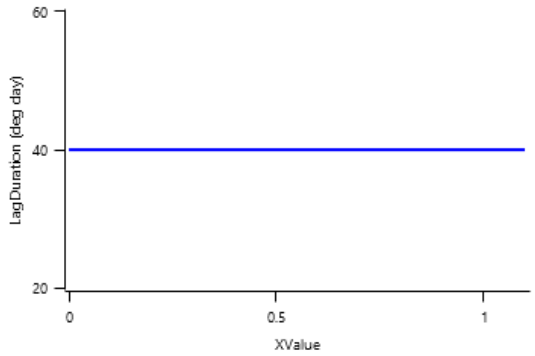
X	Y
0	12000
0.2	25000
1	25000

XValue = [Structure].RelativeNodeApperance

Constant = 0.5 (mm2)

In the absence of stress the leaf will remain at *MaxArea* for a number of degree days set by the *LagDuration* and then area will senesce to zero at the end of the *SenescenceDuration*

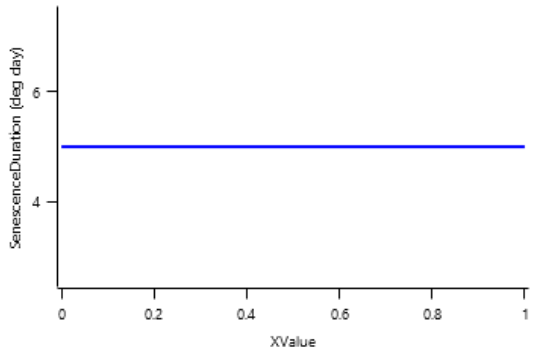
LagDuration is calculated using linear interpolation.



X	Y
0	40
0.15	40
0.3	40
0.4	40
0.73	40
1	40
1.1	40

XValue = [Structure].RelativeNodeApperance

SenescenceDuration is calculated using linear interpolation.

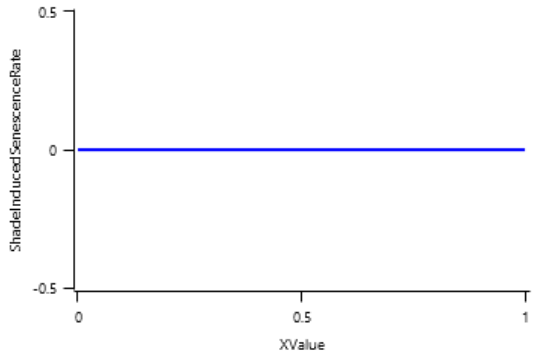


X	Y
0	5
0.2	5
1	5

XValue = [Structure].RelativeNodeApperance

Mutual shading can cause premature senescence of cohorts if the leaf area above them becomes too great. Each cohort models the proportion of its area that is lost to shade induced senescence each day as:

ShadeInducedSenescenceRate is calculated using linear interpolation.



X	Y
0	0
0.5	0
0.8	0
1	0

XValue = [Leaf].CohortCurrentRankCoverAbove

3.12.9.4 Stress effects on Leaf Area Index

Stress reduces leaf area in a number of ways. Firstly, stress occuring prior to the appearance of the cohort can reduce cell division, so reducing the maximum leaf size. Leaf captures this by multiplying the *MaxSize* of each cohort by a *CellDivisionStress* factor which is calculated as:

CellDivisionStress = 1 ()

Leaf.FN quantifys the N stress status of the plant and represents the concentration of metabolic N relative the maximum potentil metabolic N content of the leaf calculated as (*Leaf.NConc - MinimumNConc*)/(*CriticalNConc - MinimumNConc*).

Leaf.FW quantifies water stress and is calculated as *Leaf.Transpiration*/*Leaf.WaterDemand*, where *Leaf.Transpiration* is the minimum of *Leaf.WaterDemand* and *Root.WaterUptake*

Stress during the *GrowthDuration* of the cohort reduces the size increase of the cohort by multiplying the potential increase by a *ExpansionStress* factor:

ExpansionStress = 1 ()

Stresses can also acellerate the onset and rate of senescence in a number of ways. Nitrogen shortage will cause N to be retranslocated out of lower order leaves to support the expansion of higher order leaves and other organs When this happens the lower order cohorts will have their area reduced in proportion to the amount of N that is remobilised out of them.

Water stress hastens senescence by increasing the rate of thermal time accumulation in the lag and senescence phases. This is done by multiplying thermal time accumulation by *DroughtInducedLagAcceleration* and *DroughtInducedSenescenceAcceleration* factors, respectively:

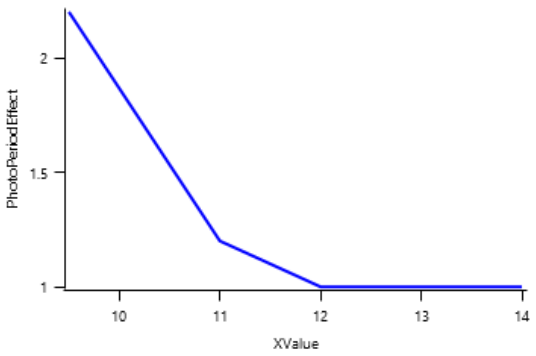
3.12.9.4.1 DroughtInducedLagAcceleration

IF [Weather].DaysSinceWinterSolstice < Autumn THEN

One = 1 ()

ELSE

PhotoPeriodEffect is calculated using linear interpolation.



X	Y
9.5	2.2
11	1.2
12	1
14	1

3.12.9.4.1.1 XValue

Returns the duration of the day, or photoperiod, in hours. This is calculated using the specified latitude (given in the weather file) and twilight sun angle threshold. If a variable called ClimateControl.PhotoPeriod is found in the simulation, it will be used instead.

Twilight = 0 (degrees)

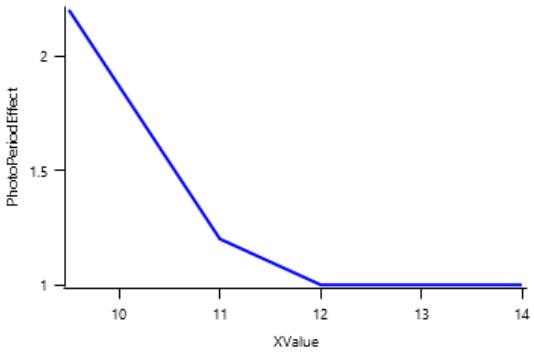
3.12.9.4.2 DroughtInducedSenAcceleration

IF [Weather].DaysSinceWinterSolstice < Autumn THEN

One = 1 ()

ELSE

PhotoPeriodEffect is calculated using linear interpolation.



X	Y
9.5	2.2
11	1.2
12	1
14	1

3.12.9.4.2.1 XValue

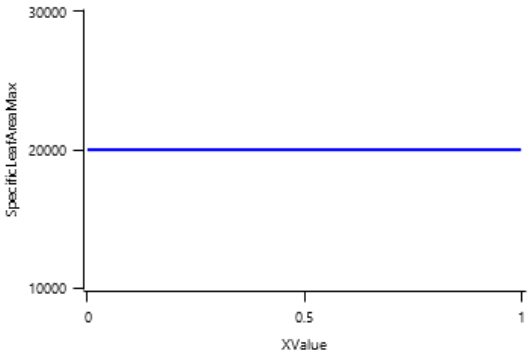
Returns the duration of the day, or photoperiod, in hours. This is calculated using the specified latitude (given in the weather file) and twilight sun angle threshold. If a variable called ClimateControl.PhotoPeriod is found in the simulation, it will be used instead.

Twilight = 0 (degrees)

3.12.9.5 Dry matter Demand

Leaf calculates the DM demand from each cohort as a function of the potential size increment (DeltaPotentialArea) an specific leaf area bounds. Under non stressed conditions the demand for non-storage DM is calculated as *DeltaPotentialArea* divided by the mean of *SpecificLeafAreaMax* and *SpecificLeafAreaMin*. Under stressed conditions it is calculated as *DeltaWaterConstrainedArea* divided by *SpecificLeafAreaMin*.

SpecificLeafAreaMax is calculated using linear interpolation.



X	Y
0	20000
1	20000

XValue = [Structure].RelativeNodeApperance

SpecificLeafAreaMin = 15000 ()

Non-storage DM Demand is then seperated into structural and metabolic DM demands using the *StructuralFraction*:

StructuralFraction = 0.8 ()

The storage DM demand is calculated from the sum of metabolic and structural DM (including todays demands) multiplied by a *NonStructuralFraction*:

Unknown child name: NonStructuralFraction

3.12.9.6 Nitrogen Demand

Leaf calculates the N demand from each cohort as a function of the potential DM increment and N concentration bounds. Structural N demand = *PotentialStructuralDMAllocation* * *MinimumNConc* where:

MinimumNConc = 0.0001 ()

Metabolic N demand is calculated as *PotentialMetabolicDMAllocation* * (*CriticalNConc* - *MinimumNConc*) where:

CriticalNConc = 0.0001 ()

Storage N demand is calculated as the sum of metabolic and structural wt (including todays demands) multiplied by *LuxaryNconc* (*MaximumNConc* - *CriticalNConc*) less the amount of storage N already present. *MaximumNConc* is given by:

MaximumNConc = 0.05 ()

3.12.9.7 Drymatter supply

In additon to photosynthesis, the leaf can also supply DM by reallocation of senescing DM and retranslocation of storgage DM: Reallocation supply is a proportion of the metabolic and non-structural DM that would be senesced each day where the proportion is set by:

DMReallocationFactor = 0.5 ()

Retranslocation supply is calculated as a proportion of the amount of storage DM in each cohort where the proportion is set by :

DMRetranslocationFactor = 1 ()

3.12.9.8 Nitrogen supply

Nitrogen supply from the leaf comes from the reallocation of metabolic and storage N in senescing material and the retranslocation of metabolic and storage N. Reallocation supply is a proportion of the Metabolic and Storage DM that would be senesced each day where the proportion is set by:

NReallocationFactor = 0 ()

Retranslocation supply is calculated as a proportion of the amount of storage and metabolic N in each cohort where the proportion is set by :

NRetranslocationFactor = 0 ()

DetachmentLagDuration = 1 (deg day)

DetachmentDuration = 1 (deg day)

InitialNConc = 0.05 ()

StorageFraction = 0 ()

SenessingLeafRelativeSize = 1 ()

LeafSizeShapeParameter = 0.01 ()

3.12.9.8.1 LagDurationAgeMultiplier

LagDurationAgeMultiplier = 1 1 1

3.12.9.8.2 SenescenceDurationAgeMultiplier

SenescenceDurationAgeMultiplier = 1 1 1

3.12.9.8.3 LeafSizeAgeMultiplier

LeafSizeAgeMultiplier = 1 1 1 1 1 1 1 1 1 1 1

RemobilisationCost = 0 ()

CarbonConcentration = 0.4 ()

MaintenanceRespirationFunction = Qm x ExponentialFunction

Where:

Qm = 0.03 ()

3.12.9.9 ExponentialFunction

An exponential function

SubtractFunction = [Weather].MeanT - RefT

Where:

RefT = 20 ()

Xvalue = [Weather].MeanT

3.12.9.10 FRGRFunction

FRGRFunction is calculated as the minimum of *RUE_FT* and *Others*

Where:

RUE_FT = [Leaf].Photosynthesis.FT

3.12.9.11 Others

Others is calculated as the minimum of *RUE_FN* and *RUE_FVPD*

Where:

RUE_FN = [Leaf].Photosynthesis.FN

RUE_FVPD = [Leaf].Photosynthesis.FVPD

3.12.9.12 Total Biomass

This is a composite biomass class, representing the sum of 1 or more biomass objects.

Total summarises the following biomass objects:

- [Leaf].Live
- [Leaf].Dead

3.12.9.13 BiomassRemovalDefaults

This organ will respond to certain management actions by either removing some of its biomass from the system or transferring some of its biomass to the soil surface residues. The following table describes the default proportions of live and dead biomass that are transferred out of the simulation using "Removed" or to soil surface residue using "To Residue" for a range of management actions. The total percentage removed for live or dead must not exceed 100%. The difference between the total and 100% gives the biomass remaining on the plant. These can be changed during a simulation using a manager script.

Method	% Live Removed	% Dead Removed	% Live To Residue	% Dead To Residue
Harvest	0	0	0	0
Cut	0	0	0	0
Prune	0	0	0	0
Graze	0	0	0	0

DMConversionEfficiency = 0.83 ()

RemobilisationCost = 0 ()

CarbonConcentration = 0.4 ()

StructuralFraction = 0.8 ()

StomatalConductanceCO2Modifier = 1 ()

3.12.9.14 DMDemandPriorityFactors

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

Structural = 1.6 (g/m2)

Metabolic = 1.6 (g/m2)

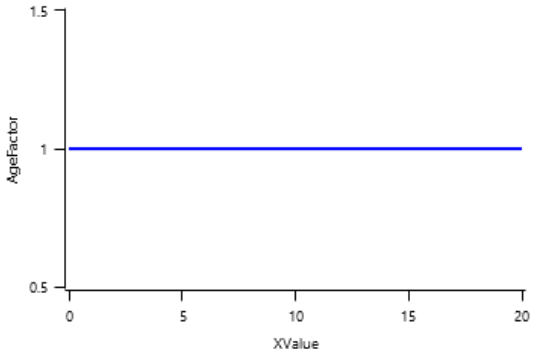
Storage = 0.01 (g/m2)

FrostFraction = 0 ()

WidthFunction = AgeFactor x SeasonalWidthPattern

Where:

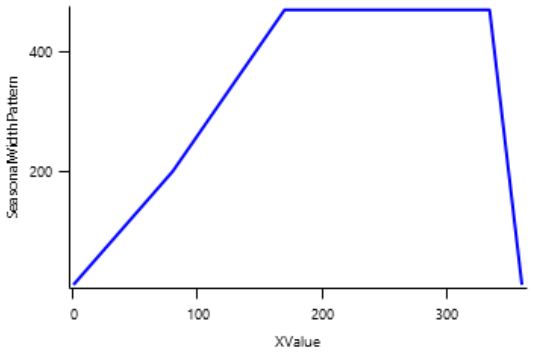
AgeFactor is calculated using linear interpolation.



X	Y
0	1
2	1
10	1
20	1

XValue = [Phenology].Age.Years

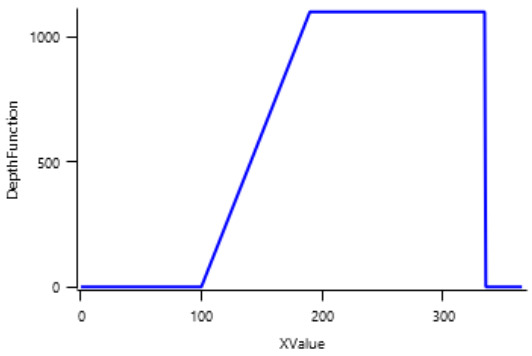
SeasonalWidthPattern is calculated using linear interpolation.



X	Y
0	10
80	200
170	470
335	470
361	10

XValue = [Weather].DaysSinceWinterSolstice

DepthFunction is calculated using linear interpolation.



X	Y
0	1
100	1
190	1100
335	1100
336	1
366	1

XValue = [Weather].DaysSinceWinterSolstice

3.12.10 Shoot

This organ is simulated using a GenericOrgan type. It is parameterised to calculate the growth, senescence, and detachment of any organ that does not have specific functions.

Shoot was represented by a Generic Organ class.

The structural biomass demand of the shoot module was calculated by a delta function that takes the difference of yesterday and today's value of a fitted biomass growth curve. The structural biomass growth curve was represented by a beta growth function (Yin et al., 2003) fitted on individual shoot weight over time and times the shoot population per square meter. Shoot population per square meter was calculated internally based on shoot number per vine and vine density. DMs(t)= SP*DM _max (1+ (t_e-t)/(t_e-t_m)) (t/t_e)^(t_e/(t_e-t_m)) 0=t_m<t_e (Eq. 8) DMs(t) is the DM of all shoots (g m-2) at time t from budburst. SP the shoot population per square meter. DMmax is the fitted maximum DM per shoot under four-cane pruned vines (g). t_e is the time when DMmax was reached (79.2 td), t_m is the time at maximum growth rate (43 td).

Note the shoot dry matter including both the primary and secondary shoots. For capturing the reduction of shoot biomass under high retained bud number per meter row caused both by smaller primary shoot and less lateral shoots (Greven et al., 2014), the priority factor for shoot DM demand was set to a very low value (4e-3).

Parameters for shoot non-structural DM demand, e.g. priority factors were set as half of the trunk and root.

3.12.10.1 Dry Matter Demand

The dry matter demand for the organ is calculated as defined in DMDemands, based on the DMDemandFunction and partition fractions for each biomass pool.

3.12.10.1.1 DMDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

Structural is the daily differential of

Integral = [Structure].MainStemPopn x BetaGrowthFunction

Where:

3.12.10.1.1.1 BetaGrowthFunction

a beta growth function of the form y = Ymax * (1 + (te - t)/(te-tm))* (t/te)^(te/(te-tm))

Ymax = 30 (g/m2)

XValue = [Shoot].ThermalTimeAfterBudBurst

ShootPopulation = [Structure].MainStemPopn

Metabolic = 0 (g/m2)

Storage is calculated as the minimum of *Storage* and *Zero*

Where:

Storage = SubtractFunction x DailySynthesis

Where:

SubtractFunction = StructuralWt - [Grapevine].Shoot.Live.StorageWt

Where:

StructuralWt = [Grapevine].Shoot.Live.Wt x MaxConcentration

Where:

MaxConcentration = 0.26 (g/m2)

StructuralWt = [Grapevine].Shoot.Live.Wt

StorageWt = [Grapevine].Shoot.Live.StorageWt

3.12.10.1.1.2 DailySynthesis

a beta growth function of the form y = Ymax * (1 + (te - t)/(te-tm))* (t/te)^(te/(te-tm))

Ymax = 0.03 (g/m2)

XValue = [Shoot].ThermalTimeAfterBudBurst

Zero = 0 (g/m2)

3.12.10.2 Nitrogen Demand

The N demand is calculated as defined in NDemands, based on DM demand the N concentration of each biomass pool.

3.12.10.2.1 NDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

Structural = [Shoot].minimumNconc x [Shoot].potentialDMAllocation.Structural

MinNconc = [Shoot].minimumNconc

PotentialDMAllocation = [Shoot].potentialDMAllocation.Structural

Metabolic = MetabolicNconc x [Shoot].potentialDMAllocation.Structural

Where:

MetabolicNconc = [Shoot].criticalNConc - [Shoot].minimumNconc

CritNconc = [Shoot].criticalNConc

MinNconc = [Shoot].minimumNconc

PotentialDMAllocation = [Shoot].potentialDMAllocation.Structural

3.12.10.2.1.1 Storage

The partitioning of daily N supply to storage N attempts to bring the organ's N content to the maximum concentration.

Storage = [Shoot].maximumNconc × ([Shoot].Live.Wt + potentialAllocationWt) - [Shoot].Live.N

The demand for storage N is further reduced by a factor specified by the [Shoot].NitrogenDemandSwitch.

MinimumNConc = 0 (g/g)

CriticalNConc = [Shoot].MinimumNConc

MaximumNConc = 0.0001 (g/g)

The demand for N is reduced by a factor specified by the NitrogenDemandSwitch.

NitrogenDemandSwitch has a non-zero value between BudBurst and LeafFall calcualted as:

Constant = 1 ()

3.12.10.3 Dry Matter Supply

Shoot will reallocate 100% of DM that senesces each day.

Shoot will retranslocate 4% of non-structural DM each day.

3.12.10.4 Nitrogen Supply

Shoot can reallocate up to 100% of N that senesces each day if required by the plant arbitrator to meet N demands.

Shoot can retranslocate up to 10% of non-structural N each day if required by the plant arbitrator to meet N demands.

3.12.10.5 Senescence and Detachment

Shoot has senescence parameterised to zero so all biomass in this organ will remain alive.

Shoot has detachment parameterised to zero so all biomass in this organ will remain with the plant until a defoliation or harvest event occurs.

3.12.11 Cordon

This organ is simulated using a GenericOrgan type. It is parameterised to calculate the growth, senescence, and detachment of any organ that does not have specific functions.

Cordon, Trunk and Structural root module were all represented by the Generic Organ class, which has properties of biomass status and daily biomass demand and supply. The structural DM demand of cordon and trunk (Eq. 6) were calculated by their radius (r, unit m) derived from their DM and length, daily growth rate of the radius (dr/dt, m d-1), length (l, m) and wood density (, g m-3) (Cieslak et al., 2011). dStructuralDM _i/dt=q_(str,i) 2prl dr/dt Eq. 6 where i was organ type. Qstr,i was the priority factor for structural DM demand for a certain organ, determined as 0.7 for all three organs see model calibration methods. Daily growth rate of the radius were calculated based on the trunk circumference measurements on the same eight hundreds vines with 14 years' interval. The structural carbon demand of the structural root was calculated by the structural demand of the trunk times the structural root/trunk ratio (set to one in this model). The structural root/trunk ratio may vary between vineyards and training systems.

The non-structural DM demand was modelled as an active competing sink (Cieslak et al., 2011), and the parameters were kept the same for those three organs. The rate of non-structural DM synthesis depended on organ size and limited by overloading. dNonStructuralDM _i/dt=q_(NSC,i) *k_syn* (C_(max,NSC)* StructuralDM _i- NonStructuralDM _i) Eq. 7 The equation limited carbon storage owing to non-structural DM overloading and the organ storage capacity, which was proportional to the current organ structural biomass, Cmax,NSC was the maximum non-structural DM per unit of structural DM, set as 0.26 based on our field trunk and root samples (Greven et al., 2016). qNSC was set to 0.61 throughout the growing season. ksyn was daily non-structural DM synthesis rate, which was represented by a beta growth function Eq. 8 (Yin et al., 2003) for capturing the fast recovery of carbon reserves after flowering (Greven et al., 2016; Seleznyova et al., 2018).

Parameters were optimized based on the measured dynamics of non-structural carbon in trunk and root by Greven et al. (2016), see model calibration methods. k_syn=k_max (1+ (t_e-t)/(t_e-t_m)) (t/t_e)^(t_e/(t_e-t_m)) 0=t_m<t_e (Eq. 8) k_max was the maximum non-structural DM synthesis rate, 0.03 g g-1. t_e was the thermal day after budburst when kmax was reached, 54 td (around véraison). t_m was the time when maximum increase rate occurred, 34.4 td. For potential carbon retranslocation by those storage organs, a value of 3.7% each day (DMRetranslocationFactor) was obtained through optimization. Actual retranslocation depends on the daily carbon supply and demand differences. The biomass of retained cane for following season was reset by the pruning events based on input cane diameter in the UI each year for the cane-pruned vines. The biomass of trunk and fibrous root keep growing each year.

3.12.11.1 Dry Matter Demand

The dry matter demand for the organ is calculated as defined in DMDemands, based on the DMDemandFunction and partition fractions for each biomass pool.

3.12.11.1.1 DMDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

3.12.11.1.1.1 Metabolic

Metabolic is calculated using specific values or functions for various growth phases. The function will use a value of zero for phases not specified below.

GrowingSeason has a non-zero value between BudBurst and LeafFall calcualted as:

Constant = 0 (g/m2)

3.12.11.1.1.2 Structural

Structural is calculated using specific values or functions for various growth phases. The function will use a value of zero for phases not specified below.

GrowingSeason has a non-zero value between BudBurst and LeafFall calcualted as:

Structural = 2 x PI x Radius x Density x RadiusGR x [Cordon].CordonLength x [Grapevine].Population x [Cordon].CordonNumber

Where:

2 = 2 (g/m2)

PI = 3.1415926 (g/m2)

3.12.11.2 Radius

Raises the value of the child to the power of the exponent specified

RadiusSquare = [Cordon].Live.Wt d [Grapevine].Population d [Cordon].CordonNumber d Density d [Cordon].CordonLength d PI

Where:

Density = 500000 (g/m2)

PI = 3.1415926 (g/m2)

Wt = [Cordon].Live.Wt

population = [Grapevine].Population

CordonNumber = [Cordon].CordonNumber

Length = [Cordon].CordonLength

Density = 400000 (g/m2)

RadiusGR = 9E-06 (g/m2)

Length = [Cordon].CordonLength

Population = [Grapevine].Population

CordonNumber = [Cordon].CordonNumber

3.12.11.2.1 Storage

Storage is calculated using specific values or functions for various growth phases. The function will use a value of zero for phases not specified below.

GrowingSeason has a non-zero value between BudBurst and LeafFall calcualted as:

Storage = SubtractFunction x DailySynthesis

Where:

SubtractFunction = StructuralWt - [Grapevine].Cordon.Live.StorageWt

Where:

StructuralWt = [Grapevine].Cordon.Live.StructuralWt x MaxConcentration

Where:

MaxConcentration = 0.26 (g/m2)

StructuralWt = [Grapevine].Cordon.Live.StructuralWt

StorageWt = [Grapevine].Cordon.Live.StorageWt

3.12.11.2.1.1 DailySynthesis

a beta growth function of the form y = Ymax * (1 + (te - t)/(te-tm))* (t/te)^(te/(te-tm))

Ymax = 0.03 (g/m2)

XValue = [Shoot].ThermalTimeAfterBudBurst

3.12.11.3 Nitrogen Demand

The N demand is calculated as defined in NDemands, based on DM demand the N concentration of each biomass pool.

3.12.11.3.1 NDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

Structural = [Cordon].minimumNconc x [Cordon].potentialDMAllocation.Structural

MinNconc = [Cordon].minimumNconc

PotentialDMAllocation = [Cordon].potentialDMAllocation.Structural

Metabolic = MetabolicNconc x [Cordon].potentialDMAllocation.Structural

Where:

MetabolicNconc = [Cordon].criticalNConc - [Cordon].minimumNconc

CritNconc = [Cordon].criticalNConc

MinNconc = [Cordon].minimumNconc

PotentialDMAllocation = [Cordon].potentialDMAllocation.Structural

3.12.11.3.1.1 Storage

The partitioning of daily N supply to storage N attempts to bring the organ's N content to the maximum concentration.

Storage = [Cordon].maximumNconc × ([Cordon].Live.Wt + potentialAllocationWt) - [Cordon].Live.N

The demand for storage N is further reduced by a factor specified by the [Cordon].NitrogenDemandSwitch.

MinimumNConc = 0 (g/g)

CriticalNConc = [Cordon].MinimumNConc

MaximumNConc = 0.0001 (g/g)

The demand for N is reduced by a factor specified by the NitrogenDemandSwitch.

NitrogenDemandSwitch has a non-zero value between BudBurst and LeafFall calcualted as:

Constant = 1 ()

3.12.11.4 Dry Matter Supply

Cordon will reallocate 100% of DM that senesces each day.

Cordon will retranslocate 3.7% of non-structural DM each day.

3.12.11.5 Nitrogen Supply

Cordon can reallocate up to 100% of N that senesces each day if required by the plant arbitrator to meet N demands.

Cordon can retranslocate up to 4% of non-structural N each day if required by the plant arbitrator to meet N demands.

3.12.11.6 Senescence and Detachment

Cordon has senescence parameterised to zero so all biomass in this organ will remain alive.

Cordon has detachment parameterised to zero so all biomass in this organ will remain with the plant until a defoliation or harvest event occurs.

3.12.12 Trunk

This organ is simulated using a GenericOrgan type. It is parameterised to calculate the growth, senescence, and detachment of any organ that does not have specific functions.

3.12.12.1 Dry Matter Demand

The dry matter demand for the organ is calculated as defined in DMDemands, based on the DMDemandFunction and partition fractions for each biomass pool.

3.12.12.1.1 DMDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

3.12.12.1.1.1 Metabolic

Metabolic is calculated using specific values or functions for various growth phases. The function will use a value of zero for phases not specified below.

GrowingSeason has a non-zero value between BudBurst and LeafFall calcualted as:

Constant = 0 (g/m2)

3.12.12.1.1.2 Structural

Structural is calculated using specific values or functions for various growth phases. The function will use a value of zero for phases not specified below.

GrowingSeason has a non-zero value between BudBurst and LeafFall calcualted as:

Structural = 2 x PI x Radius x [Trunk].TrunkLength x Density x RadiusGR x [Grapevine].Population x Age

Where:

2 = 2 (g/m2)

PI = 3.1415926 (g/m2)

3.12.12.2 Radius

Raises the value of the child to the power of the exponent specified

RadiusSquare = [Trunk].Live.Wt d [Grapevine].Population d PI d [Trunk].TrunkLength d Density

Where:

PI = 3.1415926 (g/m2)

Density = 500000 (g/m2)

Wt = [Trunk].Live.Wt

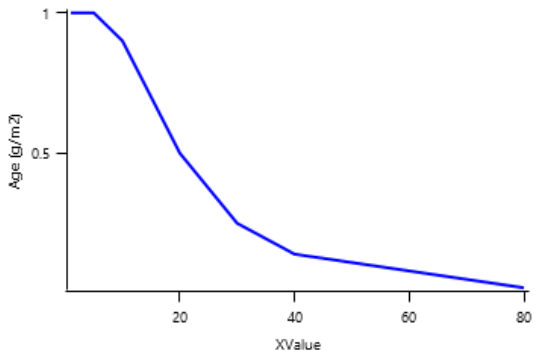
population = [Grapevine].Population

Length = [Trunk].TrunkLength

Density = 500000 (g/m2)

RadiusGR = 8E-06 (g/m2)

Age is calculated using linear interpolation.



X	Y
1	1
5	1
10	0.9
20	0.5
30	0.25
40	0.14
80	0.02

XValue = [Phenology].Age.Years

Length = [Trunk].TrunkLength

population = [Grapevine].Population

3.12.12.2.1 Storage

Storage is calculated using specific values or functions for various growth phases. The function will use a value of zero for phases not specified below.

GrowingSeason has a non-zero value between BudBurst and LeafFall calcualted as:

Storage is calculated as the minimum of *Storage* and *Zero*

Where:

Storage = SubtractFunction x DailySynthesis

Where:

SubtractFunction = StructuralWt - [Grapevine].Trunk.Live.StorageWt

Where:

StructuralWt = [Grapevine].Trunk.Live.StructuralWt x MaxConcentration

Where:

MaxConcentration = 0.26 (g/m2)

StructuralWt = [Grapevine].Trunk.Live.StructuralWt

StorageWt = [Grapevine].Trunk.Live.StorageWt

3.12.12.2.1.1 DailySynthesis

a beta growth function of the form y = Ymax * (1 + (te - t)/(te-tm))* (t/te)^(te/(te-tm))

Ymax = 0.03 (g/m2)

XValue = [Shoot].ThermalTimeAfterBudBurst

Zero = 0 (g/m2)

3.12.12.3 Nitrogen Demand

The N demand is calculated as defined in NDemands, based on DM demand the N concentration of each biomass pool.

3.12.12.3.1 NDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

Structural = [Trunk].minimumNconc x [Trunk].potentialDMAllocation.Structural

MinNconc = [Trunk].minimumNconc

PotentialDMAllocation = [Trunk].potentialDMAllocation.Structural

Metabolic = MetabolicNconc x [Trunk].potentialDMAllocation.Structural

Where:

MetabolicNconc = [Trunk].criticalNConc - [Trunk].minimumNconc

CritNconc = [Trunk].criticalNConc

MinNconc = [Trunk].minimumNconc

PotentialDMAllocation = [Trunk].potentialDMAllocation.Structural

3.12.12.3.1.1 Storage

The partitioning of daily N supply to storage N attempts to bring the organ's N content to the maximum concentration.

Storage = [Trunk].maximumNconc × ([Trunk].Live.Wt + potentialAllocationWt) - [Trunk].Live.N

The demand for storage N is further reduced by a factor specified by the [Trunk].NitrogenDemandSwitch.

MinimumNConc = 0 (g/g)

CriticalNConc = [Trunk].MinimumNConc

MaximumNConc = 0.0001 (g/g)

The demand for N is reduced by a factor specified by the NitrogenDemandSwitch.

NitrogenDemandSwitch has a non-zero value between BudBurst and LeafFall calcualted as:

Constant = 1 ()

3.12.12.4 Dry Matter Supply

Trunk will reallocate 100% of DM that senesces each day.

Trunk will retranslocate 3.7% of non-structural DM each day.

3.12.12.5 Nitrogen Supply

Trunk can reallocate up to 100% of N that senesces each day if required by the plant arbitrator to meet N demands.

Trunk can retranslocate up to 10% of non-structural N each day if required by the plant arbitrator to meet N demands.

3.12.12.6 Senescence and Detachment

Trunk has senescence parameterised to zero so all biomass in this organ will remain alive.

Trunk has detachment parameterised to zero so all biomass in this organ will remain with the plant until a defoliation or harvest event occurs.

3.12.13 StructuralRoot

This organ is simulated using a GenericOrgan type. It is parameterised to calculate the growth, senescence, and detachment of any organ that does not have specific functions.

Note this represents the structural roots of all the plant in the population. The structural root are primarily considered as storage organs, its researves can be made available to boost plant growth in spring and/or following a defoliation.

The structural root is separated from the mian root class because the current root class can not handle the biomass retranslocation while a generic organ can. The biomass of the structural root is expressed as gram per square meter, while the initial dry mass for the root class is expressed at per plant level.

structural roots in grapevine are perennial. It can grow in diamention and biomass. Its structural growth rate was calculated based on the the trunk demand times structural root and trunk ratio in this model. Its storage biomass typicall decrease in spring and refill after veraison.

3.12.13.1 Dry Matter Demand

The dry matter demand for the organ is calculated as defined in DMDemands, based on the DMDemandFunction and partition fractions for each biomass pool.

3.12.13.1.1 DMDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

3.12.13.1.1.1 Metabolic

Metabolic is calculated using specific values or functions for various growth phases. The function will use a value of zero for phases not specified below.

GrowingSeason has a non-zero value between BudBurst and LeafFall calcualted as:

Constant = 0 (g/m2)

3.12.13.1.1.2 Structural

Structural is calculated using specific values or functions for various growth phases. The function will use a value of zero for phases not specified below.

GrowingSeason has a non-zero value between BudBurst and LeafFall calcualted as:

Structural = 2 x PI x Radius x [Trunk].TrunkLength x Density x RadiusGR x [Grapevine].Population x RootTrunkRatio x Age

Where:

2 = 2 (g/m2)

PI = 3.1415926 (g/m2)

3.12.13.2 Radius

Raises the value of the child to the power of the exponent specified

RadiusSquare = [Trunk].Live.Wt d [Grapevine].Population d PI d [Trunk].TrunkLength d Density

Where:

PI = 3.1415926 (g/m2)

Density = 500000 (g/m2)

Wt = [Trunk].Live.Wt

population = [Grapevine].Population

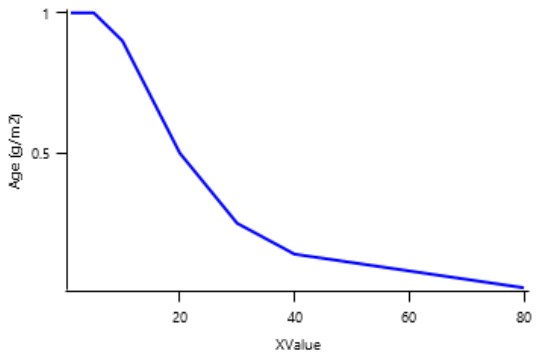
Length = [Trunk].TrunkLength

Density = 500000 (g/m2)

RadiusGR = 8E-06 (g/m2)

RootTrunkRatio = 1 (g/m2)

Age is calculated using linear interpolation.



X	Y
1	1
5	1
10	0.9
20	0.5
30	0.25
40	0.14
80	0.02

XValue = [Phenology].Age.Years

Length = [Trunk].TrunkLength

population = [Grapevine].Population

3.12.13.2.1 Storage

Storage is calculated using specific values or functions for various growth phases. The function will use a value of zero for phases not specified below.

GrowingSeason has a non-zero value between BudBurst and LeafFall calcualted as:

Storage is calculated as the minimum of *Storage* and *Zero*

Where:

Storage = SubtractFunction x DailySynthesis

Where:

SubtractFunction = StructuralWt - [Grapevine].StructuralRoot.Live.StorageWt

Where:

StructuralWt = [Grapevine].StructuralRoot.Live.StructuralWt x MaxConcentration

Where:

MaxConcentration = 0.26 (g/m2)

StructuralWt = [Grapevine].StructuralRoot.Live.StructuralWt

StorageWt = [Grapevine].StructuralRoot.Live.StorageWt

3.12.13.2.1.1 DailySynthesis

a beta growth function of the form y = Ymax * (1 + (te - t)/(te-tm))* (t/te)^(te/(te-tm))

Ymax = 0.03 (g/m2)

XValue = [Shoot].ThermalTimeAfterBudBurst

Zero = 0 (g/m2)

3.12.13.3 Nitrogen Demand

The N demand is calculated as defined in NDemands, based on DM demand the N concentration of each biomass pool.

3.12.13.3.1 NDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

Structural = [Trunk].minimumNconc x [Trunk].potentialDMAllocation.Structural

MinNconc = [Trunk].minimumNconc

PotentialDMAllocation = [Trunk].potentialDMAllocation.Structural

Metabolic = MetabolicNconc x [Trunk].potentialDMAllocation.Structural

Where:

MetabolicNconc = [Trunk].criticalNConc - [Trunk].minimumNconc

CritNconc = [Trunk].criticalNConc

MinNconc = [Trunk].minimumNconc

PotentialDMAllocation = [Trunk].potentialDMAllocation.Structural

3.12.13.3.1.1 Storage

The partitioning of daily N supply to storage N attempts to bring the organ's N content to the maximum concentration.

Storage = [StructuralRoot].maximumNconc × ([StructuralRoot].Live.Wt + potentialAllocationWt) - [StructuralRoot].Live.N

The demand for storage N is further reduced by a factor specified by the [StructuralRoot].NitrogenDemandSwitch.

MinimumNConc = 0 (g/g)

CriticalNConc = [Trunk].MinimumNConc

MaximumNConc = 0.0001 (g/g)

The demand for N is reduced by a factor specified by the NitrogenDemandSwitch.

NitrogenDemandSwitch has a non-zero value between BudBurst and LeafFall calcualted as:

Constant = 1 ()

3.12.13.4 Dry Matter Supply

StructuralRoot will reallocate 100% of DM that senesces each day.

StructuralRoot will retranslocate 3.7% of non-structural DM each day.

3.12.13.5 Nitrogen Supply

StructuralRoot can reallocate up to 100% of N that senesces each day if required by the plant arbitrator to meet N demands.

StructuralRoot can retranslocate up to 10% of non-structural N each day if required by the plant arbitrator to meet N demands.

3.12.13.6 Senescence and Detachment

StructuralRoot has senescence parameterised to zero so all biomass in this organ will remain alive.

StructuralRoot has detachment parameterised to zero so all biomass in this organ will remain with the plant until a defoliation or harvest event occurs.

3.12.14 Berry

This organ uses a generic model for plant reproductive components. Yield is calculated from its components in terms of organ number and size (for example, grain number and grain size).

The final yield was calculated by bunches per shoot, shoots per vine, berries per bunch,and berry fresh or dry weight.

The effects of carbon status on yield compoent has not been included as it requires further calibration. Bunch number, berry number and potential berry fresh weight were determined by weather conditions at critical periods around flowerings of the previous and current season, see details at Zhu et al., 2020 OENO one. Furthermore, in the current model, carbon effects represented by total carbon supply and demand were added in the calculation of bunch number, berry number and potential berry fresh weight.

Berry dry mass accumulation following the source-sink carbon allocation rules. Brix was calculated based on the ratio of berry dry weight to fresh weight. Total titratable acid was simulated based on thermaltime accumulation after veraison follwoing a negative exponetial curve.

A long-term phenology and yield monitoring trial using both two-cane and four-cane trained vertically shoot positioned (VSP) Sauvignon blanc vines was established in four vineyards in Marlborough, New Zealand in 2004, and was used for calibrating the berry module. Phenology, bunch number, berry mass, yield and meteorology records were collated. A multivariable mixed linear model was used to assess the relationship between various yield components and weather conditions. The critical periods for each yield component and weather factor were optimised based on the maximum likelihood returned from the mixed linear model. The optimised critical periods of temperature for all yield components occurred mainly before 50 % flowering either in the previous season (during inflorescence initiation) and the current season, indicating the importance of the pre-flowering period on yield formation. Out of all weather factors, maximum daily temperature had the largest effect on bunch number and overall yield and strongly influenced berry number and bunch mass. Rainfall near flowering time had a negative effect on berry mass and bunch mass, but post-flowering rainfall had a strong positive effect.

3.12.14.1 MetFactors

The final yield was calculated by vines per hectare, shoots per vine, bunches per shoot, flower number per bunch, berries per bunch,and mean berry weight. The effects of temperature and carbon status were or will be considered on bunches per shoot, flower number per bunch, berries per bunch,and mean berry weight.

BUN_TmaxIni = TmaxIni d Time

Where:

A function that accumulates values from child functions

Tmax = [Weather].MaxT

A function that accumulates values from child functions

Days = 1 ()

BUN_RadIni = RadIni d Time

Where:

A function that accumulates values from child functions

Rad = [Weather].Radn

A function that accumulates values from child functions

Days = 1 ()

BUN_CarbonIni = FDMIni d Time

Where:

A function that accumulates values from child functions

FDM = [Grapevine].Arbitrator.FDM

A function that accumulates values from child functions

Days = 1 ()

BEN_TmaxFlow = TmaxFlow d Time

Where:

A function that accumulates values from child functions

Tmax = [Weather].MaxT

A function that accumulates values from child functions

Days = 1 ()

BEN_TminFlow = TminFlow d Time

Where:

A function that accumulates values from child functions

Tmin = [Weather].MinT

A function that accumulates values from child functions

Days = 1 ()

BEN_TmaxIni = TmaxIni d Time

Where:

A function that accumulates values from child functions

Tmax = [Weather].MaxT

A function that accumulates values from child functions

Days = 1 ()

BEN_RainTotFlow = RainTotFlow d Constant

Where:

A function that accumulates values from child functions

RainTot = [Weather].Rain

Constant = 1 ()

BEN_CarbonFlow = FDMFlow d Time

Where:

A function that accumulates values from child functions

FDM = [Grapevine].Arbitrator.FDM

A function that accumulates values from child functions

Days = 1 ()

BM_TmaxFlow = TmaxFlow d Time

Where:

A function that accumulates values from child functions

Tmax = [Weather].MaxT

A function that accumulates values from child functions

Days = 1 ()

BM_RainTotFlow = RainTotFlow d Constant

Where:

A function that accumulates values from child functions

RainTot = [Weather].Rain

Constant = 1 ()

BM_RadFlow = RadFlow d Time

Where:

A function that accumulates values from child functions

Rad = [Weather].Radn

A function that accumulates values from child functions

Days = 1 ()

BM_RainTotVer = RainTotVer d Constant

Where:

A function that accumulates values from child functions

RainTot = [Weather].Rain

Constant = 1 ()

BM_CarbonFlow = FDMFlow d Time

Where:

A function that accumulates values from child functions

FDM = [Grapevine].Arbitrator.FDM

A function that accumulates values from child functions

Days = 1 ()

3.12.14.1.1 BEN_TmaxIni1

BEN_CarbonIni = FDMIni d Time

Where:

A function that accumulates values from child functions

FDM = [Grapevine].Arbitrator.FDM

A function that accumulates values from child functions

Days = 1 ()

3.12.14.2 YieldComponent

3.12.14.2.1 BunchesPerShoot

here we use a trigger function for bunches per shoot as the value of mean bunch initiation temperature in the previous season

Numbers were guessed based on Eltom 2013 Fig. 7.7. y=0.0981*Ta + 0.35

Bunches per shoot is affected by the temperature and CHO during initiation, it is further affected by the CHO during bud burst. The later decided whether the potential initiated inflorescence can grow out successfully or been realized.

For simiplicity, right now, the CHO effect during bud burst is expressed as a function of number of buds retained.

3.12.14.2.2 BunchesPerVine

Trigger the calculation after the determination of bunches per shoot

3.12.14.2.3 BerryNum

here we use a trigger function for bunches per shoot as the value of mean flowering temperature Y0 is updating itself during the growing season, which would cause inconvience if we directly correlates the bunches per shoot and mean flowering temperature Y0.

Numbers were guessed based on Eltom 2013 Fig. 7.7. y=0.0981*Ta + 0.35

Bunches per shoot is affected by the temperature and CHO during initiation, it is further affected by the CHO during bud burst. The later decided whether the potential initiated inflorescence can grow out successfully or been realized.

For simiplicity, right now, the CHO effect during bud burst is expressed as a function of number of buds retained.

3.12.14.2.4 BerryMass

here we use a trigger function for bunches per shoot as the value of mean flowering temperature Y0 is updating itself during the growing season, which would cause inconvience if we directly correlates the bunches per shoot and mean flowering temperature Y0.

Numbers were guessed based on Eltom 2013 Fig. 7.7. y=0.0981*Ta + 0.35

Bunches per shoot is affected by the temperature and CHO during initiation, it is further affected by the CHO during bud burst. The later decided whether the potential initiated inflorescence can grow out successfully or been realized.

For simiplicity, right now, the CHO effect during bud burst is expressed as a function of number of buds retained.

NumberFunction = [Grapevine].Structure.MainStemPopn x [Berry].YieldComponent.BunchesPerShoot x [Berry].YieldComponent.BerryNum

StemPopulation = [Grapevine].Structure.MainStemPopn

BunchesPerShoot = [Berry].YieldComponent.BunchesPerShoot

BerriesPerBunch = [Berry].YieldComponent.BerryNum

SingleBerryFW = BetaGrowthFunction

Where:

3.12.14.2.5 BetaGrowthFunction

a beta growth function of the form y = Ymax * (1 + (te - t)/(te-tm))* (t/te)^(te/(te-tm))

XValue = [Berry].ThermalTimeAfterFlowering

Ymax = [Berry].YieldComponent.BerryMass

TotalBerryFW = [Berry].NumberFunction x [Berry].SingleBerryFW

BerryNumber = [Berry].NumberFunction

BerryFW = [Berry].SingleBerryFW

SingleBerryDW = BetaGrowthFunction

Where:

3.12.14.2.6 BetaGrowthFunction

a beta growth function of the form y = Ymax * (1 + (te - t)/(te-tm))* (t/te)^(te/(te-tm))

Ymax = 0.56 ()

XValue = [Berry].ThermalTimeAfterFlowering

WaterContent = Constant - AfterFlowering

Where:

Constant = 1 (g/g)

AfterFlowering has a non-zero value between Flowering and LeafFall calcualted as:

3.12.14.3 MinimumFunction

MinimumFunction is calculated as the minimum of *DivideFunction* and *Constant*

Where:

DivideFunction = [Berry].Live.Wt d [Berry].TotalBerryFW

BerryDW = [Berry].Live.Wt

TotalBerryFW = [Berry].TotalBerryFW

Constant = 0.88 (g/g)

DMDemandFunction = LessThanFunction x Constant

Where:

3.12.14.3.1 LessThanFunction

IF [Weather].DaysSinceWinterSolstice < HarvestTime THEN

AfterFruitSet has a non-zero value between Flowering and LeafFall calcualted as:

MaximumFunction is calculated as the minimum of *DeltaFunction* and *Constant*

Where:

DeltaFunction is the daily differential of

Integral = [Berry].NumberFunction x [Berry].SingleBerryDW

BerryNumber = [Berry].NumberFunction

BerryDM = [Berry].SingleBerryDW

Constant = 0 (g/m2/d)

ELSE

Zero = 0 (g/m2/d)

Constant = 1 (g/m2/d)

3.12.14.4 DMDemandPriorityFactors

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

Structural = 1.07758 (g/m2)

Metabolic = 0.001 (g/m2)

Storage = 0.001 (g/m2)

YieldPerVine = [Berry].TotalBerryFW d [Grapevine].Population d Constant

Where:

Constant = 1000 ()

TotalBerryFW = [Berry].TotalBerryFW

Population = [Grapevine].Population

3.12.14.5 Brix

Brix is calculated using specific values or functions for various growth phases. The function will use a value of zero for phases not specified below.

BeforeVeraison has a non-zero value between Flowering and Veraison calcualted as:

Constant = 5 ()

AfterVeraison has a non-zero value between Veraison and LeafFall calcualted as:

3.12.14.5.1 LessThanFunction

IF [Weather].DaysSinceWinterSolstice < HarvestTime THEN

MaximumFunction is calculated as the minimum of *DivideFunction* and *min*

Where:

DivideFunction = MaximumFunction d constant

Where:

MaximumFunction is calculated as the minimum of *SubtractFunction* and *zero*

Where:

SubtractFunction = constant - [Berry].WaterContent

Where:

constant = 0.944 ()

WaterContent = [Berry].WaterContent

zero = 0 ()

constant = 0.0082 ()

min = 5 ()

ELSE

Zero = 0 ()

3.12.14.6 TitratableAcid

TitratableAcid is calculated using specific values or functions for various growth phases. The function will use a value of zero for phases not specified below.

BeforeVeraison has a non-zero value between Flowering and Veraison calcualted as:

Constant = 0 ()

AfterVeraison has a non-zero value between Veraison and LeafFall calcualted as:

3.12.14.7 TitratableAcid

An exponential function

XValue = [Berry].ThermalTimeAfterVeraison

3.12.14.8 ThermalTimeAfterVeraison

Accumulates *ThermalTime* between [Start] and [End]

ThermalTime = [Phenology].ThermalTime

3.12.14.9 ThermalTimeAfterFlowering

Accumulates *ThermalTime* between [Start] and [End]

ThermalTime = [Phenology].ThermalTime

MaxNConcDailyGrowth = 0.01 ()

NFillingRate = 0 (g/m2/d)

MinimumNConc = 0 (g/g)

MaximumNConc = 0.0001 (g/g)

3.12.14.10 BiomassRemovalDefaults

This organ will respond to certain management actions by either removing some of its biomass from the system or transferring some of its biomass to the soil surface residues. The following table describes the default proportions of live and dead biomass that are transferred out of the simulation using "Removed" or to soil surface residue using "To Residue" for a range of management actions. The total percentage removed for live or dead must not exceed 100%. The difference between the total and 100% gives the biomass remaining on the plant. These can be changed during a simulation using a manager script.

Method	% Live Removed	% Dead Removed	% Live To Residue	% Dead To Residue
Harvest	0	0	0	0
Cut	0	0	0	0
Prune	0	0	0	0
Graze	0	0	0	0
Thin	0	0	0	0

MaximumPotentialGrainSize = 2.5 ()

DMConversionEfficiency = 0.98 (g/g)

RemobilisationCost = 0 (g/g)

CarbonConcentration = 0.4 (g/g)

MaintenanceRespirationFunction = Qm x ExponentialFunction

Where:

Qm = 0.001 (g/m^2)

3.12.14.11 ExponentialFunction

An exponential function

SubtractFunction = [Weather].MeanT - RefT

Where:

RefT = 20 (g/m^2)

Xvalue = [Weather].MeanT

3.12.15 Root

The generic root model calculates root growth in terms of rooting depth, biomass accumulation and subsequent root length density in each soil layer.

Root Growth

Roots grow downwards through the soil profile, with initial depth determined by sowing depth and the growth rate determined by RootFrontVelocity. The RootFrontVelocity is modified by multiplying it by the soil's XF value, which represents any resistance posed by the soil to root extension.

Root Depth Increase = RootFrontVelocity x XF_i x RootDepthStressFactor

where i is the index of the soil layer at the rooting front.

Root depth is also constrained by a maximum root depth.

Root length growth is calculated using the daily DM partitioned to roots and a specific root length. Root proliferation in layers is calculated using an approach similar to the generalised equimarginal criterion used in economics. The uptake of water and N per unit root length is used to partition new root material into layers of higher 'return on investment'. For example, the Root Activity for water is calculated as

RAw_i = -$WaterUptake_i$ / $LiveRootWt_i$ x $LayerThickness_i$ x ProportionThroughLayer

The amount of root mass partitioned to a layer is then proportional to root activity

$DMAllocated_i$ = TotalDMAllocated x RAw_i / TotalRAw

Dry Matter Demands

A daily DM demand is provided to the organ arbitrator and a DM supply returned. By default, 100% of the dry matter (DM) demanded from the root is structural. The daily loss of roots is calculated using a SenescenceRate function. All senesced material is automatically detached and added to the soil FOM.

Nitrogen Demands

The daily structural N demand from root is the product of total DM demand and the minimum N concentration. Any N above this is considered Storage and can be used for retranslocation and/or reallocation as the respective factors are set to values other then zero.

Nitrogen Uptake

Potential N uptake by the root system is calculated for each soil layer (i) that the roots have extended into. In each layer potential uptake is calculated as the product of the mineral nitrogen in the layer, a factor controlling

the rate of extraction (kNO3 or kNH4), the concentration of N form (ppm), and a soil moisture factor (NUptakeSWFactor) which typically decreases as the soil dries.

NO3 uptake = $NO3_i$ x kNO3 x $NO3_{ppm,\ i}$ x NUptakeSWFactor

NH4 uptake = $NH4_i$ x kNH4 x $NH4_{ppm,\ i}$ x NUptakeSWFactor

As can be seen from the above equations, the values of kNO3 and kNH4 equate to the potential fraction of each mineral N pool which can be taken up per day for wet soil when that pool has a concentration of 1 ppm.

Nitrogen uptake demand is limited to the maximum daily potential uptake (MaxDailyNUptake) and the plant's N demand. The former provides a means to constrain N uptake to a maximum value observed in the field for the crop as a whole. The demand for soil N is then passed to the soil arbitrator which determines how much of the N uptake demand each plant instance will be allowed to take up.

Water Uptake

Potential water uptake by the root system is calculated for each soil layer that the roots have extended into. In each layer potential uptake is calculated as the product of the available water in the layer (water above LL limit) and a factor controlling the rate of extraction (KL). The values of both LL and KL are set in the soil interface and KL may be further modified by the crop via the KLModifier function.

SW uptake = (SW_i - LL_i) x KL_i x KLModifier

Note that this organs is parameterised to represents all the fine roots of the plant. The root organ is reponsible for uptake but can also supply both N and DM from its non-structural biomass. The Fibrous root organ is used for simulating biomass storage and remobilisation because the current root class can not handle the biomass retranslocation.

The dynamics of root biomass was based on the root length dynamics measurement done on mature concord vines in UC Davis by comos et al., 2005. both root growth rate and mortality rate have annual cycles.Maintenance was set to zero as it was parameterized insided the senescence rate. the maximum root biomass during the season is around 60 g per plant, estimated by lakso et al., 2008.

the initial weight of the root was increased as we start the simulation in February with no leaves.the start of simulation in February was for correctly simulating the budburst in the first year. the initial weight was given to ensure that the peak of the biomass can cycle around 30 g/m2.

3.12.15.1 InitialWt

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

Structural = 100 (g/m2)

Metabolic = 0 (g/m2)

Storage = 0 (g/m2)

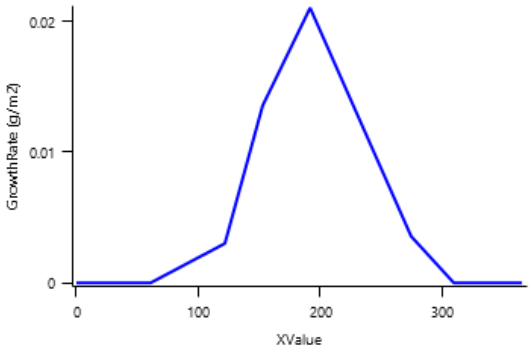
3.12.15.2 DMDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

Structural = [Root].Live.StructuralWt x GrowthRate x GrowthCostScale

Where:

GrowthRate is calculated using linear interpolation.



X	Y
0	0
61	0
122	0.0030125
153	0.013545
192	0.02102184
275	0.0035542
310	0
366	0

XValue = [Weather].DaysSinceWinterSolstice

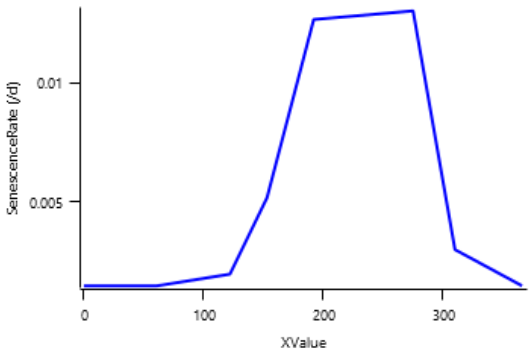
GrowthCostScale = 1.3 (g/m2)

wt = [Root].Live.StructuralWt

Metabolic = 0 (g/m2)

Storage = 0 (g/m2)

SenescenceRate is calculated using linear interpolation.



X	Y
0	0.001484
61	0.001484
122	0.001973
153	0.005178
192	0.01264
275	0.013
310	0.003
366	0.001484

XValue = [Weather].DaysSinceWinterSolstice

DMConversionEfficiency = 0.83 (g/g)

3.12.15.3 RootShape

DMRetranslocationFactor = 0 ()

DMReallocationFactor = 0 (/d)

NitrogenDemandSwitch has a non-zero value between BudBurst and LeafFall calcualted as:

Constant = 1 ()

MinimumNConc = 0 (g/g)

MaximumNConc = 0.0001 (g/g)

MaximumRootDepth = 1000000 (mm)

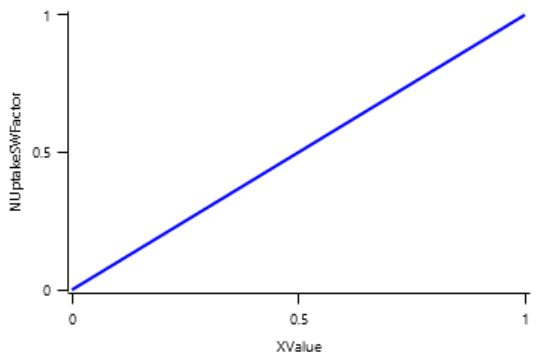
3.12.15.4 BiomassRemovalDefaults

This organ will respond to certain management actions by either removing some of its biomass from the system or transferring some of its biomass to the soil surface residues. The following table describes the default proportions of live and dead biomass that are transferred out of the simulation using "Removed" or to soil surface residue using "To Residue" for a range of management actions. The total percentage removed for live or dead must not exceed 100%. The difference between the total and 100% gives the biomass remaining on the plant. These can be changed during a simulation using a manager script.

Method	% Live Removed	% Dead Removed	% Live To Residue	% Dead To Residue
Harvest	0	0	0	0
Cut	0	0	0	0
Prune	0	0	0	0
Graze	0	0	0	0

RemobilisationCost = 0 ()

NUptakeSWFactor is calculated using linear interpolation.



X	Y
0	0
1	1

XValue = [Root].RWC

SpecificRootLength = 500 (m/g)

RootFrontVelocity = 20 (mm/d)

KNO3 = 0.02 (/d/ppm)

KNH4 = 0.003 (/d/ppm)

MaxDailyNUptake = 6 (kg N/ha/d)

CarbonConcentration = 0.4 (g/g)

3.12.15.5 NDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

Structural = [Root].minimumNconc x [Root].potentialDMAllocation.Structural

MinNconc = [Root].minimumNconc

PotentialDMAllocation = [Root].potentialDMAllocation.Structural

Metabolic = MetabolicNconc x [Root].potentialDMAllocation.Structural

Where:

MetabolicNconc = [Root].criticalNConc - [Root].minimumNconc

CritNconc = [Root].criticalNConc

MinNconc = [Root].minimumNconc

PotentialDMAllocation = [Root].potentialDMAllocation.Structural

3.12.15.5.1 Storage

The partitioning of daily N supply to storage N attempts to bring the organ's N content to the maximum concentration.

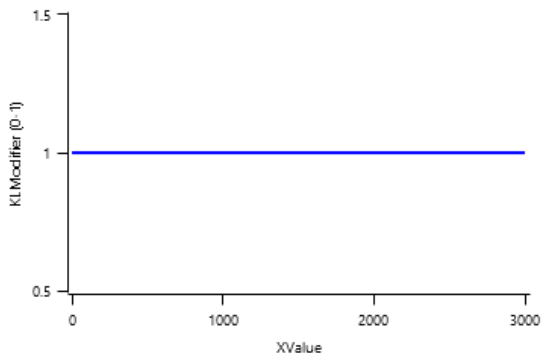
Storage = [Root].maximumNconc × ([Root].Live.Wt + potentialAllocationWt) - [Root].Live.N

The demand for storage N is further reduced by a factor specified by the [Root].NitrogenDemandSwitch.

CriticalNConc = [Root].MinimumNConc

This is important in SLURP as it is set for each species to represent their differences in rooting patterns between crop species

KLModifier is calculated using linear interpolation.



X	Y
0	1
300	1
600	1
900	1
1200	1
1500	1
1800	1
2100	1
2400	1
2700	1
3000	1

XValue = [Root].LayerMidPointDepth

RootDepthStressFactor = 1 (0-1)

3.12.15.6 DMDemandPriorityFactors

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

Structural = 1.6 (g/m2)

Metabolic = 0 (g/m2)

Storage = 0 (g/m2)

MaintenanceRespirationFunction = Qm x ExponentialFunction

Where:

Qm = 0.01 (/d)

3.12.15.7 ExponentialFunction

An exponential function

SubtractFunction = [Weather].MeanT - RefT

Where:

RefT = 20 (/d)

Xvalue = [Weather].MeanT

MortalityRate = 0 ()

3.12.16 Arbitrator

The Arbitrator class determines the allocation of dry matter (DM) and Nitrogen between each of the organs in the crop model. Each organ can have up to three different pools of biomass:

- **Structural biomass** which is essential for growth and remains within the organ once it is allocated there.
- **Metabolic biomass** which generally remains within an organ but is able to be re-allocated when the organ senesces and may be retranslocated when demand is high relative to supply.
- **Storage biomass** which is partitioned to organs when supply is high relative to demand and is available

for retranslocation to other organs whenever supply from uptake, fixation, or re-allocation is lower than demand.

The process followed for biomass arbitration is shown in Figure 4. Arbitration calculations are triggered by a series of events (shown below) that are raised every day. For these calculations, at each step the Arbitrator exchange information with each organ, so the basic computations of demand and supply are done at the organ level, using their specific parameters.

1. **doPotentialPlantGrowth**. When this event occurs, each organ class executes code to determine their potential growth, biomass supplies and demands. In addition to demands for structural, non-structural and metabolic biomass (DM and N) each organ may have the following biomass supplies:
2. **Fixation supply**. From photosynthesis (DM) or symbiotic fixation (N)
3. **Uptake supply**. Typically uptake of N from the soil by the roots but could also be uptake by other organs (eg foliage application of N).
4. **Retranslocation supply**. Storage biomass that may be moved from organs to meet demands of other organs.
5. **Reallocation supply**. Biomass that can be moved from senescing organs to meet the demands of other organs.
6. **doPotentialPlantPartitioning.** On this event the Arbitrator first executes the DoDMSetup() method to gather the DM supplies and demands from each organ, these values are computed at the organ level. It then executes the DoPotentialDMAllocation() method which works out how much biomass each organ would be allocated assuming N supply is not limiting and sends these allocations to the organs. Each organ then uses their potential DM allocation to determine their N demand (how much N is needed to produce that much DM) and the arbitrator calls DoNSetup() to gather the N supplies and demands from each organ and begin N arbitration. Firstly DoNReallocation() is called to redistribute N that the plant has available from senescing organs. After this step any unmet N demand is considered as plant demand for N uptake from the soil (N Uptake Demand).
7. **doNutrientArbitration.** When this event occurs, the soil arbitrator gets the N uptake demands from each plant (where multiple plants are growing in competition) and their potential uptake from the soil and determines how much of their demand that the soil is able to provide. This value is then passed back to each plant instance as their Nuptake and doNUptakeAllocation() is called to distribute this N between organs.
8. **doActualPlantPartitioning.** On this event the arbitrator call DoNRetranslocation() and DoNFixation() to satisfy any unmet N demands from these sources. Finally, DoActualDMAllocation is called where DM allocations to each organ are reduced if the N allocation is insufficient to achieve the organs minimum N concentration and final allocations are sent to organs.

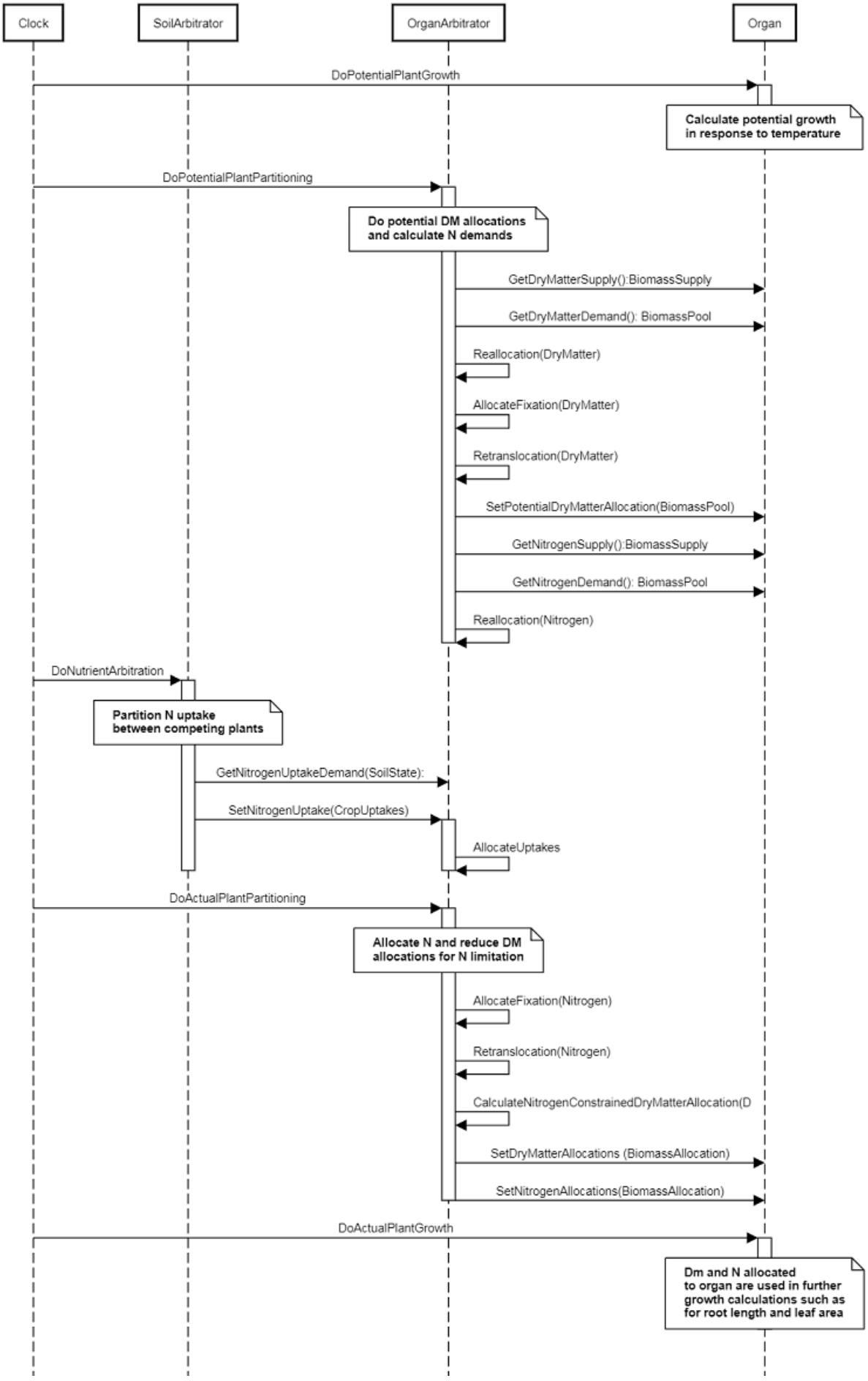


Figure 4: Schematic showing the procedure for arbitration of biomass partitioning. Pink boxes represent events that occur every day and their numbering shows the order of calculations. Blue boxes represent the methods that are called when these events occur. Orange boxes contain properties that make up the organ/arbitrator interface. Green boxes are organ specific properties.

3.12.17 Varieties

Chardonnay, Merlot, PinotGris, PinotNoir, SauvignonBlanc

3.12.17.1 SauvignonBlanc

SauvignonBlanc overrides the following properties:

3.12.17.2 Chardonnay

Chardonnay overrides the following properties:

[Phenology].EndoDormancy.Target.BaseTarget.FixedValue = 52.0647
[Phenology].EndoDormancy.Progression.Response.Xo.FixedValue = 10.98
[Phenology].EndoDormancy.Progression.Response.b.FixedValue = -25.5611
[Phenology].Budding.Target.FixedValue = 906.4065 [Phenology].Budding.Progression.Response.X = 0,0.27668,10.27668,20.27668 [Phenology].ThermalTime.Response.MinTemp = 0.239618
[Phenology].ThermalTime.Response.OptTemp = 27.149 [Phenology].ThermalTime.Response.MaxTemp = 48.75 [Phenology].Flowering.Target.FixedValue = 45.41293
[Phenology].BerryDevelopment.Target.Base.FixedValue = 47.77695

3.12.17.3 Merlot

Merlot overrides the following properties:

[Phenology].EndoDormancy.Target.BaseTarget.FixedValue = 88.74292
[Phenology].EndoDormancy.Progression.Response.Xo.FixedValue = 15.07995
[Phenology].EndoDormancy.Progression.Response.b.FixedValue = -1.27593
[Phenology].Budding.Target.FixedValue = 767.6113 [Phenology].Budding.Progression.Response.X = 0,0.353281,10.353281,20.353281 [Phenology].ThermalTime.Response.MinTemp = 2.51
[Phenology].ThermalTime.Response.OptTemp = 22.245 [Phenology].ThermalTime.Response.MaxTemp = 40.316 [Phenology].Flowering.Target.FixedValue = 42.007
[Phenology].BerryDevelopment.Target.Base.FixedValue = 50.84

3.12.17.4 PinotNoir

PinotNoir overrides the following properties:

[Phenology].EndoDormancy.Target.BaseTarget.FixedValue = 72.1
[Phenology].EndoDormancy.Progression.Response.Xo.FixedValue =7.95
[Phenology].EndoDormancy.Progression.Response.b.FixedValue = -45.3083
[Phenology].Budding.Target.FixedValue = 627.0382 [Phenology].Budding.Progression.Response.X = 0,0.244,10.244,20.244 [Phenology].ThermalTime.Response.MinTemp = 1.514
[Phenology].ThermalTime.Response.OptTemp = 26.42341 [Phenology].ThermalTime.Response.MaxTemp = 36.87527 [Phenology].Flowering.Target.FixedValue = 27.5666
[Phenology].BerryDevelopment.Target.Base.FixedValue = 29.75368

3.12.17.5 PinotGris

PinotGris overrides the following properties:

[Phenology].EndoDormancy.Target.BaseTarget.FixedValue = 74.163
[Phenology].EndoDormancy.Progression.Response.Xo.FixedValue = 12.53467
[Phenology].EndoDormancy.Progression.Response.b.FixedValue = -66.5092
[Phenology].Budding.Target.FixedValue = 519.1688 [Phenology].Budding.Progression.Response.X = 0, 1.93454,11.93454,21.93454 [Phenology].ThermalTime.Response.MinTemp = 1.173588
[Phenology].ThermalTime.Response.OptTemp = 25.6193 [Phenology].ThermalTime.Response.MaxTemp = 35.65306 [Phenology].Flowering.Target.FixedValue = 31.13788
[Phenology].BerryDevelopment.Target.Base.FixedValue = 32.965

A report class for writing output to the data store.

4 DataStore

A storage service for reading and writing to/from a database.

4.1 PhenoObs

Reads the contents of a specific sheet from an EXCEL file and stores into the DataStore.

4.2 PredictedObserved

Reads the contents of a file (in apsim format) and stores into the DataStore. If the file has a column name of 'SimulationName' then this model will only input data for those rows where the data in column 'SimulationName' matches the name of the simulation under which this input model sits. If the file does NOT have a 'SimulationName' column then all data will be input.

4.2.1 PredictedObserved

Variable	n	Slope	Intercept	R2	RMSE	NSE	ME	M
BerryMass								
BerryNum	3409	0.536	27.549	0.404	9.386	0.349	-1.283	6.
BudBurstDOY								
BunchesPerVine								
f.rad.int	768	0.918	0.031	0.644	0.068	0.520	-0.008	0.
FloweringDOY								
psw	789	0.107	260.965	0.041	100.683	-0.622	58.792	84.
transpiration	1591	0.004	0.021	0.016	1.580	-2.222	-1.314	1.
VeraisonDOY								
FloweringDAWS	286	0.773	38.702	0.704	4.433	0.697	0.238	3.
LAIStrip	129	0.928	1.525	0.483	2.371	-0.240	1.186	1.
leaf.area.shoot	37	1.438	-0.042	0.875	0.061	0.421	0.024	0.
leaf.area.shoot.lateral	13	1.513	0.066	0.328	0.088	-42.246	0.083	0.
leaf.area.shoot.main	12	-0.398	0.224	0.133	0.043	-6.627	0.034	0.
leaf.area.vine	129	0.928	2.470	0.483	3.841	-0.240	1.921	2.
leaf.dw.shoot	56	1.093	-0.775	0.813	3.359	0.713	0.312	2.
leaf.dw.vine	22	1.083	-160.535	0.718	145.764	0.086	-101.887	107.
leaf.no.shoot	28	0.861	-0.845	0.770	7.201	0.598	-4.565	5.
leaf.no.shoot.main	93	1.236	-1.523	0.624	4.374	-0.163	1.760	2.
psw	789	0.107	260.965	0.041	100.683	-0.622	58.792	84.
shoot.dw.mean	115	0.713	4.267	0.703	4.548	0.696	0.689	3.
ta	4125	0.135	13.540	0.027	11.969	-0.487	-2.670	6.
total.NSC.concentration.root	41	0.631	0.041	0.370	0.031	0.181	-0.002	0.
total.NSC.concentration.trunk	41	0.533	0.042	0.380	0.040	0.070	-0.020	0.
TotalBerryDW	3367	0.689	150.641	0.772	336.525	0.673	-175.981	240.
TotalBerryFW	3467	0.000	0.000	NaN	5781.383	-5.179	-5292.886	5292.
transpiration	1591	0.004	0.021	0.016	1.580	-2.222	-1.314	1.
VeraisonDAWS	285	0.822	41.287	0.554	6.825	0.421	-0.547	5.
yield.per.vine	266	0.670	1.285	0.475	2.247	0.335	-0.673	1.

5 BaseSim

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

5.1 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.

5.1.1 Pre-timestep events (in order)

Events
DoInitialSummary
StartOfSimulation
CLEMInitialiseResource
CLEMInitialiseActivity
FinalInitialise
CLEMValidate
StartOfMonth
StartOfYear

5.1.2 Timestep events (in order)

Events
DoWeather
DoDailyInitialisation
StartOfDay
DoManagement
DoPestDiseaseDamage
DoEnergyArbitration
DoSoilWaterMovement
DoSoilTemperature
DoSoilOrganicMatter
DoSurfaceOrganicMatterDecomposition
DoUpdateWaterDemand
DoWaterArbitration
PrePhenology
DoPhenology
DoPotentialPlantGrowth
DoPotentialPlantPartioning
DoNutrientArbitration
DoActualPlantPartioning

Events
DoActualPlantGrowth
DoStock
DoLifecycle
DoUpdate
DoManagementCalculations
DoReportCalculations
EndOfDay
DoReport

5.1.3 Post-timestep events (in order)

Events
EndOfSimulation

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.

5.2 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.

A report class for writing output to the data store.

Double Guyôt-trained vines, pruned each year to retain 24 nodes were monitored. Vines were trained using vertical shoot positioning (VSP), with an exposed leaf area height of 1.2 m, and trimmed to maintain a compact canopy. Vines were planted 1.8 m apart within the row, and 2.4 m apart between rows. Average flowering date was determined by monitoring all the inflorescences on one cane on a vine in each plot on a regular basis throughout flowering. Likewise, regular berry samples (at least weekly) were taken from before the date of véraison until harvest at a soluble solids concentration value of 21.5 oBrix.

The Grapevine model is constructed from the following list of software components. Details of the implementation and model parameterisation are provided in the following sections.

List of Plant Model Components.

Component Name	Component Type

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 holding layer structure information

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.

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.

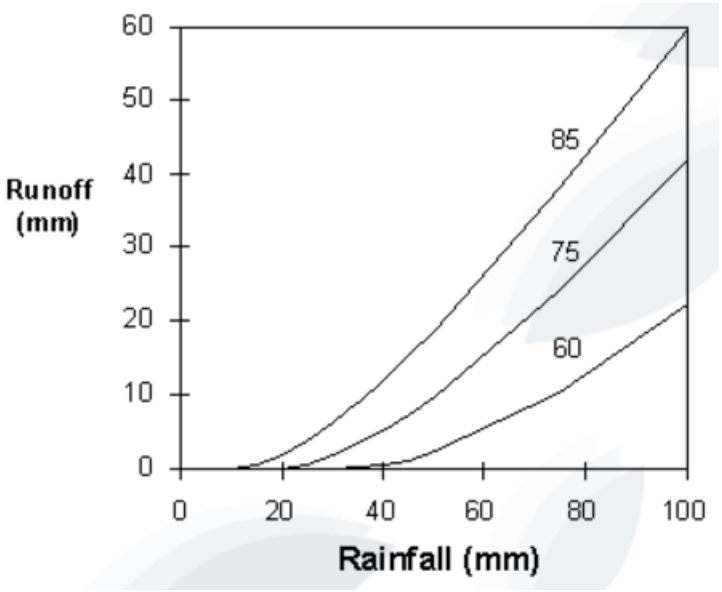
The module is interfaced with SurfaceOrganicMatter and crop modules so that simulation of the soil water balance responds to change in the status of surface residues and crop cover(via tillage, decomposition and crop growth).

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).

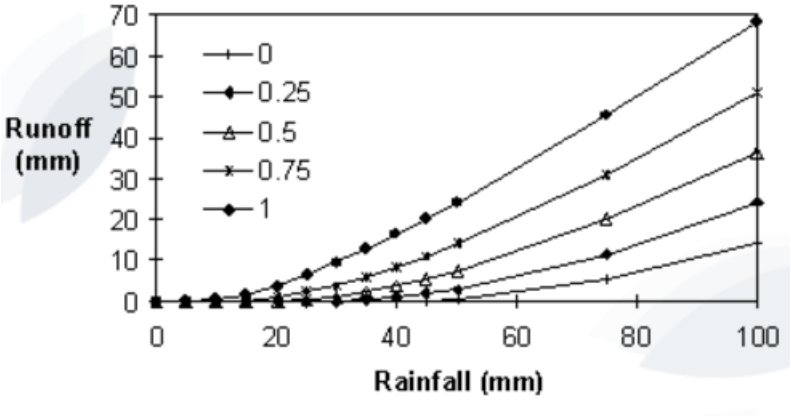
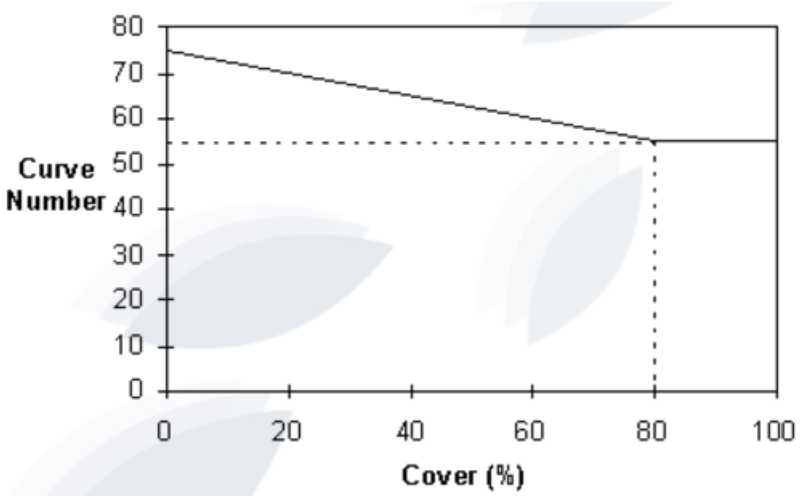


Figure: Residue cover effect on runoff curve number where bare soil curve number is 75 and total reduction in curve number is 20 at 80% cover.

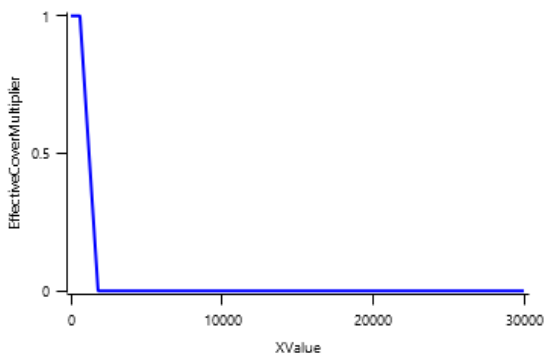


Surface residues inhibit the transport of water across the soil surface during runoff events and so different families of response curves are used according to the amount of crop and residue cover.The extent of the effect on runoff is specified by a threshold surface cover (CNCov), above which there 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.

EffectiveCoverMultiplier is calculated using linear interpolation.



X	Y
0	1
600	1
1800	0
30000	0

XValue = 0 ()

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).

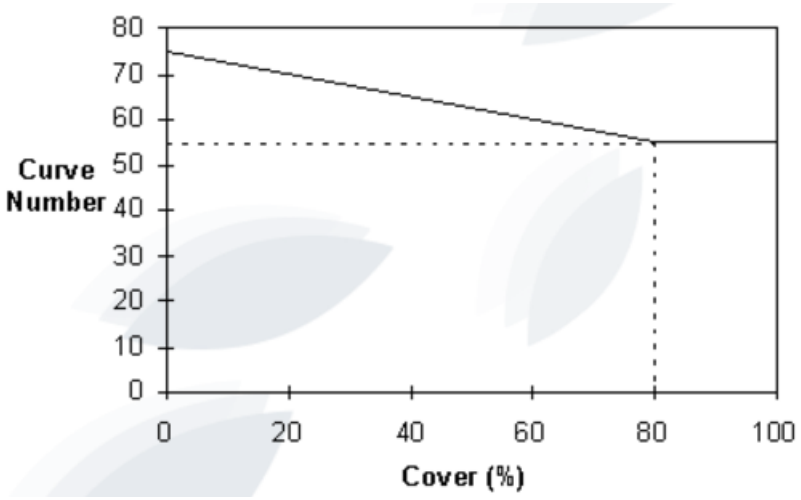
Once the water content of the soil has decreased below a threshold value the rate of supply from the soil will be less than potential evaporation (second stage evaporation). These behaviors are described in SoilWater through the use of two parameters: U and CONA.

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 $t > t_1$ $Es = U \times t + CONA \times Sqrt(t-t_1)$

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

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)

This class takes soil variables simulated at each of the modelled soil layers and maps them onto a new specified layering. The outputs can be used for producing summaries and to facilitate comparison with observed data.

This class encapsulates an operations schedule.

This class encapsulates an operations schedule.

The Slurp model is constructed from the following list of software components. Details of the implementation and model parameterisation are provided in the following sections.

List of Plant Model Components.

Component Name	Component Type
Phenology	Models.PMF.Phen.Phenology
Arbitrator	Models.PMF.OrganArbitrator
Root	Models.PMF.Organs.Root
Leaf	Models.PMF.Organs.SimpleLeaf
MortalityRate	Models.Functions.Constant

5.2.1 SLURP: the Sound of a crop using water

This model has been built using the Plant Modelling Framework (PMF) of Brown et al., 2014 to provide a simple representation of crops. It is usefull for water and nitrogen balance studies where the focus is on soil processes and a very simple crop is adequate. The model does not predict crop growth, development or yields. It simply takes up water and nitrogen. .

SLURP has no phenology so the behaviour of the SLURP is the same throughout its ontology (with the exception of root depth being able to increase). SLURP consists of 2 organs:

1. Leaf which Is represented with a Simpleleaf class and provides a Water uptake demand to the soil arbitrator.

2. Root which extracts water and nitrogen from the soil for plant growth

5.2.2 Inclusion in APSIM simulations

A Slurp crop is included in a simulation the same as any other APSIM crop

- The Slurp object needs to be dragged or copied from the Crop folder in the tool box into the field of your simulation.
- It is then planted with a sowing rule

```
Slurp.Sow(cultivar: StaticCrop, population: 1, depth: 10, rowSpacing: 150);
```

- Note that SLURP has no notion of population or rowSpacing but these parameters are required by the Sow method so filler values are provided
- `depth` in the Sow arguement is the depth that the crop is sown at. While this has no effect on emergence in SLURP it sets the depth that the root system is initialised at. For static crops this depth should be set to the rooting depth that is expected as roots will not grow during the simulation

5.2.3 Altering Slurp properties during runs

In some cases users will wish to change properties of Slurp while the simulation is running. This can be done using a the set method in a manager script.

```
object LAIResetValue = leaflai;
zone.Set("Slurp.Leaf.LAIFunction.Value()", LAIResetValue);
object HeightResetValue = CoverToday * MaximumHeight;
zone.Set("Slurp.Leaf.HeightFunction.Value()", HeightResetValue);
```

5.2.4 Phenology

Slurp's phenological development is simulated as the progression through a series of developmental phases, each bound by distinct growth *stages*.

5.2.4.1 Phases

List of stages and phases used in the simulation of crop phenological development

Phase Number	Phase Name	Initial Stage	Final Stage
1	Initial	Initialisation	Sowing
2	Ongoing	Sowing	Never

5.2.5 Initial Phase

The *Initial* phase goes from the *Initialisation* stage to the *Sowing* stage.

The *Target* for completion is calculated as

Target = 0 (oD)

Progression through the *Initial* phase is calculated daily and accumulated until the *Target* is reached.

Progression = [Phenology].ThermalTime

5.2.6 Ongoing Phase

The *Ongoing* phase goes from the *Sowing* stage to the *Never* stage.

The *Target* for completion is calculated as

Target = 1E+15 (oD)

Progression through the *Ongoing* phase is calculated daily and accumulated until the *Target* is reached.

Progression = [Phenology].ThermalTime

5.2.6.1 Arbitrator

The Arbitrator class determines the allocation of dry matter (DM) and Nitrogen between each of the organs in the crop model. Each organ can have up to three different pools of biomass:

- **Structural biomass** which is essential for growth and remains within the organ once it is allocated there.
- **Metabolic biomass** which generally remains within an organ but is able to be re-allocated when the organ senesces and may be retranslocated when demand is high relative to supply.
- **Storage biomass** which is partitioned to organs when supply is high relative to demand and is available for retranslocation to other organs whenever supply from uptake, fixation, or re-allocation is lower than demand.

The process followed for biomass arbitration is shown in Figure 7. Arbitration calculations are triggered by a series of events (shown below) that are raised every day. For these calculations, at each step the Arbitrator exchange information with each organ, so the basic computations of demand and supply are done at the organ level, using their specific parameters.

1. **doPotentialPlantGrowth**. When this event occurs, each organ class executes code to determine their potential growth, biomass supplies and demands. In addition to demands for structural, non-structural and metabolic biomass (DM and N) each organ may have the following biomass supplies:
2. **Fixation supply**. From photosynthesis (DM) or symbiotic fixation (N)
3. **Uptake supply**. Typically uptake of N from the soil by the roots but could also be uptake by other organs (eg foliage application of N).
4. **Retranslocation supply**. Storage biomass that may be moved from organs to meet demands of other organs.
5. **Reallocation supply**. Biomass that can be moved from senescing organs to meet the demands of other organs.
6. **doPotentialPlantPartitioning.** On this event the Arbitrator first executes the DoDMSetup() method to gather the DM supplies and demands from each organ, these values are computed at the organ level. It then executes the DoPotentialDMAllocation() method which works out how much biomass each organ would be allocated assuming N supply is not limiting and sends these allocations to the organs. Each organ then uses their potential DM allocation to determine their N demand (how much N is needed to produce that much DM) and the arbitrator calls DoNSetup() to gather the N supplies and demands from each organ and begin N arbitration. Firstly DoNReallocation() is called to redistribute N that the plant has available from senescing organs. After this step any unmet N demand is considered as plant demand for N uptake from the soil (N Uptake Demand).
7. **doNutrientArbitration.** When this event occurs, the soil arbitrator gets the N uptake demands from each plant (where multiple plants are growing in competition) and their potential uptake from the soil and determines how much of their demand that the soil is able to provide. This value is then passed back to each plant instance as their Nuptake and doNUptakeAllocation() is called to distribute this N between organs.
8. **doActualPlantPartitioning.** On this event the arbitrator call DoNRetranslocation() and DoNFixation() to satisfy any unmet N demands from these sources. Finally, DoActualDMAllocation is called where DM allocations to each organ are reduced if the N allocation is insufficient to achieve the organs minimum N concentration and final allocations are sent to organs.

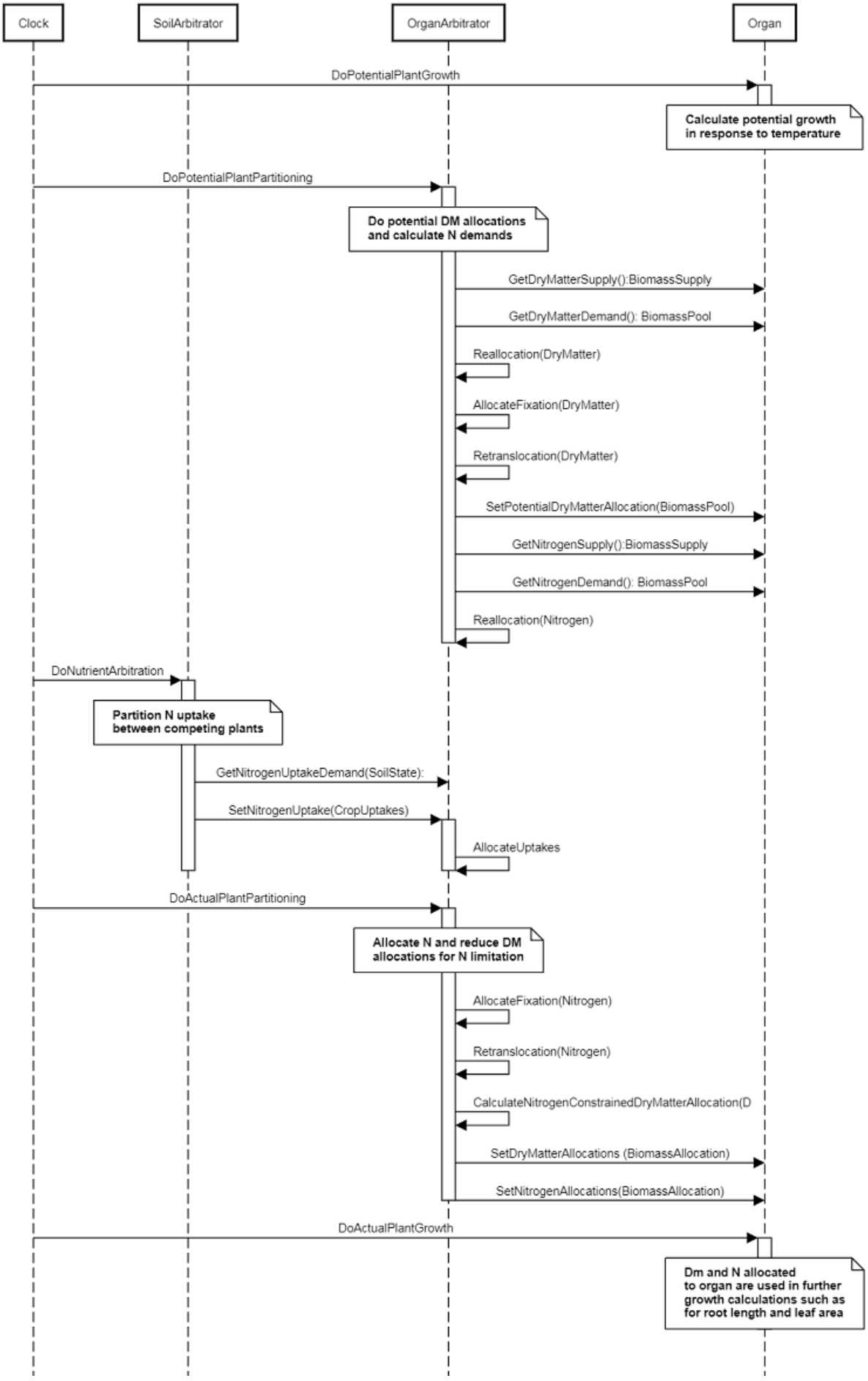


Figure 7: Schematic showing the procedure for arbitration of biomass partitioning. Pink boxes represent events that occur every day and their numbering shows the order of calculations. Blue boxes represent the methods that are called when these events occur. Orange boxes contain properties that make up the organ/arbitrator interface. Green boxes are organ specific properties.

5.2.7 Root

The generic root model calculates root growth in terms of rooting depth, biomass accumulation and subsequent root length density in each soil layer.

Root Growth

Roots grow downwards through the soil profile, with initial depth determined by sowing depth and the growth rate determined by RootFrontVelocity. The RootFrontVelocity is modified by multiplying it by the soil's XF value, which represents any resistance posed by the soil to root extension.

Root Depth Increase = RootFrontVelocity x XF_i x RootDepthStressFactor

where i is the index of the soil layer at the rooting front.

Root depth is also constrained by a maximum root depth.

Root length growth is calculated using the daily DM partitioned to roots and a specific root length. Root proliferation in layers is calculated using an approach similar to the generalised equimarginal criterion used in economics. The uptake of water and N per unit root length is used to partition new root material into layers of higher 'return on investment'. For example, the Root Activity for water is calculated as

$RAw_i = -WaterUptake_i / LiveRootWt_i \times LayerThickness_i \times ProportionThroughLayer$

The amount of root mass partitioned to a layer is then proportional to root activity

$DMAllocated_i = TotalDMAllocated \times RAw_i / TotalRAw$

Dry Matter Demands

A daily DM demand is provided to the organ arbitrator and a DM supply returned. By default, 100% of the dry matter (DM) demanded from the root is structural. The daily loss of roots is calculated using a SenescenceRate function. All senesced material is automatically detached and added to the soil FOM.

Nitrogen Demands

The daily structural N demand from root is the product of total DM demand and the minimum N concentration. Any N above this is considered Storage and can be used for retranslocation and/or reallocation as the respective factors are set to values other then zero.

Nitrogen Uptake

Potential N uptake by the root system is calculated for each soil layer (i) that the roots have extended into. In each layer potential uptake is calculated as the product of the mineral nitrogen in the layer, a factor controlling the rate of extraction (kNO3 or kNH4), the concentration of N form (ppm), and a soil moisture factor (NUptakeSWFactor) which typically decreases as the soil dries.

$NO3\ uptake = NO3_i \times kNO3 \times NO3_{ppm,\,i} \times NUptakeSWFactor$

$NH4\ uptake = NH4_i \times kNH4 \times NH4_{ppm,\,i} \times NUptakeSWFactor$

As can be seen from the above equations, the values of kNO3 and kNH4 equate to the potential fraction of each mineral N pool which can be taken up per day for wet soil when that pool has a concentration of 1 ppm.

Nitrogen uptake demand is limited to the maximum daily potential uptake (MaxDailyNUptake) and the plant's N demand. The former provides a means to constrain N uptake to a maximum value observed in the field for the crop as a whole. The demand for soil N is then passed to the soil arbitrator which determines how much of the N uptake demand each plant instance will be allowed to take up.

Water Uptake

Potential water uptake by the root system is calculated for each soil layer that the roots have extended into. In each layer potential uptake is calculated as the product of the available water in the layer (water above LL limit) and a factor controlling the rate of extraction (KL). The values of both LL and KL are set in the soil interface and KL may be further modified by the crop via the KLModifier function.

$SW\ uptake = (SW_i - LL_i) \times KL_i \times KLModifier$

5.2.7.1 RootShape

RootFrontVelocity = TemperatureEffect x Potential x DMConversionEfficiency

Where:

TemperatureEffect is the Average of sub-daily values from a Models.Functions.XYPairs.

Firstly 3-hourly estimates of air temperature (Ta) are interpolated usig the method of Jones et al., 1986 which assumes a sinusoidal temperature. pattern between Tmax and Tmin.

Each of the interpolated air temperatures are then passed into the following Response and the Average taken to give daily TemperatureEffect

Response is calculated from an XY matrix (graphed below) which returns a value for Y interpolated from the Xvalue provided.

Potential = 0 (mm/d)

DMConversionEfficiency = 1 (mm/d)

MaxDailyNUptake = 1 (kg N/ha/d)

DMConversionEfficiency = 1 (g/g)

RemobilisationCost = 0 ()

MaximumRootDepth = 1000 (mm)

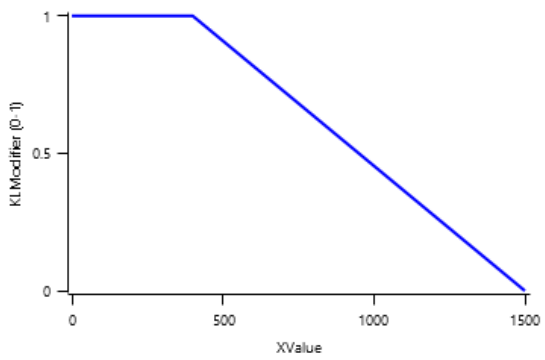
NitrogenDemandSwitch = 1 ()

MaximumNConc = 0.01 (g/g)

MinimumNConc = 0.01 (g/g)

This is important in SLURP as it is set for each species to represent their differences in rooting patterns between crop species

KLModifier is calculated using linear interpolation.



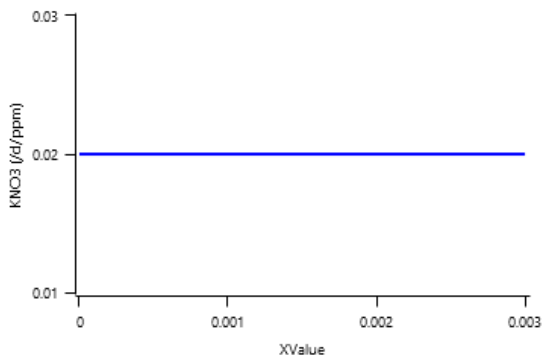
X	Y
0	1
400	1
1500	0

XValue = [Root].LayerMidPointDepth

SenescenceRate = 0 (/d)

SpecificRootLength = 105 (m/g)

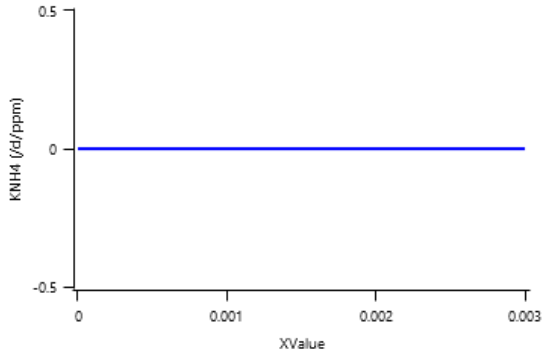
KNO3 is calculated using linear interpolation.



X	Y
0	0.02
0.003	0.02

XValue = [Root].LengthDensity

KNH4 is calculated using linear interpolation.



X	Y
0	0
0.003	0

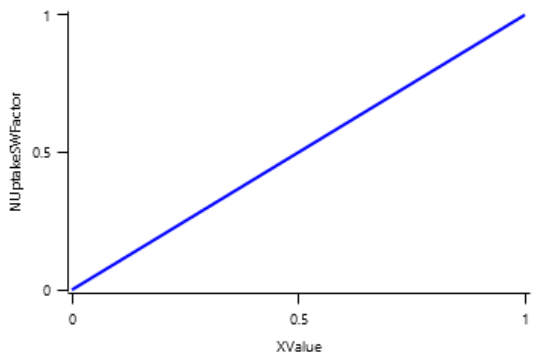
XValue = [Root].LengthDensity

5.2.7.2 BiomassRemovalDefaults

This organ will respond to certain management actions by either removing some of its biomass from the system or transferring some of its biomass to the soil surface residues. The following table describes the default proportions of live and dead biomass that are transferred out of the simulation using "Removed" or to soil surface residue using "To Residue" for a range of management actions. The total percentage removed for live or dead must not exceed 100%. The difference between the total and 100% gives the biomass remaining on the plant. These can be changed during a simulation using a manager script.

Method	% Live Removed	% Dead Removed	% Live To Residue	% Dead To Residue
Harvest	0	0	0	0
Cut	0	0	0	0
Prune	0	0	0	0
Graze	0	0	0	0

NUptakeSWFactor is calculated using linear interpolation.



X	Y
0	0
1	1

XValue = [Root].RWC

CarbonConcentration = 0.4 (g/g)

MaintenanceRespirationFunction = 0 (/d)

5.2.7.3 DMDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

Structural = DMDemandFunction x StructuralFraction

Where:

5.2.7.3.1 DMDemandFunction

DMDemandFunction = PartitionFraction [Arbitrator].DM.TotalFixationSupply

Where:

PartitionFraction = 0 (g/m2)

StructuralFraction = 1 (g/m2)

Metabolic = 0 (g/m2)

5.2.7.3.2 Storage

The partitioning of daily growth to storage biomass is based on a storage fraction.

StorageFraction = 1 - [Root].DMDemands.Structural.StructuralFraction

One = 1 (g/m2)

StructuralFraction = [Root].DMDemands.Structural.StructuralFraction

5.2.7.4 NDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

Structural = [Root].minimumNconc x [Root].potentialDMAllocation.Structural

MinNconc = [Root].minimumNconc

PotentialDMAllocation = [Root].potentialDMAllocation.Structural

Metabolic = MetabolicNconc x [Root].potentialDMAllocation.Structural

Where:

MetabolicNconc = [Root].criticalNConc - [Root].minimumNconc

CritNconc = [Root].criticalNConc

MinNconc = [Root].minimumNconc

PotentialDMAllocation = [Root].potentialDMAllocation.Structural

5.2.7.4.1 Storage

The partitioning of daily N supply to storage N attempts to bring the organ's N content to the maximum concentration.

Storage = [Root].maximumNconc × ([Root].Live.Wt + potentialAllocationWt) - [Root].Live.N

The demand for storage N is further reduced by a factor specified by the [Root].NitrogenDemandSwitch.

CriticalNConc = [Root].MinimumNConc

5.2.7.5 InitialWt

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

Structural = 0.005 (g/m2)

Metabolic = 0 (g/m2)

Storage = 0 (g/m2)

RootDepthStressFactor = 1 (0-1)

5.2.7.6 Leaf

This organ is simulated using a SimpleLeaf organ type. It provides the core functions of intercepting radiation, producing biomass through photosynthesis, and determining the plant's transpiration demand. The model also calculates the growth, senescence, and detachment of leaves. SimpleLeaf does not distinguish leaf cohorts by age or position in the canopy.

Radiation interception and transpiration demand are computed by the MicroClimate model. This model takes into account competition between different plants when more than one is present in the simulation. The values of canopy Cover, LAI, and plant Height (as defined below) are passed daily by SimpleLeaf to the MicroClimate model. MicroClimate uses an implementation of the Beer-Lambert equation to compute light interception and the Penman-Monteith equation to calculate potential evapotranspiration. These values are then given back to SimpleLeaf which uses them to calculate photosynthesis and soil water demand.

InitialWt = InitialPlantWt x [Plant].Population

Where:

InitialPlantWt = 0 (g/m2)

Population = [Plant].Population

5.2.7.6.1 Dry Matter Demand

The dry matter demand for the organ is calculated as defined in DMDemands, based on the DMDemandFunction and partition fractions for each biomass pool.

5.2.7.6.1.1 DMDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

Structural = DMDemandFunction x StructuralFraction

Where:

DMDemandFunction = PartitionFraction [Arbitrator].DM.TotalFixationSupply

Where:

PartitionFraction = 1 (g/m2)

StructuralFraction = 0.5 (g/m2)

Metabolic = 0 (g/m2)

The partitioning of daily growth to storage biomass is based on a storage fraction.

StorageFraction = 1 - [Leaf].DMDemands.Structural.StructuralFraction

One = 1 (g/m2)

StructuralFraction = [Leaf].DMDemands.Structural.StructuralFraction

5.2.7.6.2 Nitrogen Demand

The N demand is calculated as defined in NDemands, based on DM demand the N concentration of each biomass pool.

5.2.7.6.2.1 NDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

Structural = [Leaf].minimumNconc x [Leaf].potentialDMAllocation.Structural

MinNconc = [Leaf].minimumNconc

PotentialDMAllocation = [Leaf].potentialDMAllocation.Structural

Metabolic = MetabolicNconc x [Leaf].potentialDMAllocation.Structural

Where:

MetabolicNconc = [Leaf].criticalNConc - [Leaf].minimumNconc

CritNconc = [Leaf].criticalNConc

MinNconc = [Leaf].minimumNconc

PotentialDMAllocation = [Leaf].potentialDMAllocation.Structural

The partitioning of daily N supply to storage N attempts to bring the organ's N content to the maximum concentration.

Storage = [Leaf].maximumNconc × ([Leaf].Live.Wt + potentialAllocationWt) - [Leaf].Live.N

The demand for storage N is further reduced by a factor specified by the [Leaf].NitrogenDemandSwitch.

MinimumNConc = 0.05 (g/g)

CriticalNConc = [Leaf].MinimumNConc

MaximumNConc = 0.05 (g/g)

5.2.7.6.3 Dry Matter Supply

Leaf does not reallocate DM when senescence of the organ occurs.

Leaf does not retranslocate non-structural DM.

Photosynthesis = 5 ()

5.2.7.6.4 Nitrogen Supply

Leaf does not reallocate N when senescence of the organ occurs.

Leaf does not retranslocate non-structural N.

5.2.7.6.5 Canopy Properties

Leaf has been defined with a LAIFunction, cover is calculated using the Beer-Lambert equation.

Area = 5 ()

ExtinctionCoefficient = 0.7 ()

Tallness = 400 ()

5.2.7.6.6 StomatalConductance

Stomatal Conductance (gs) is calculated for use within the micromet model by adjusting a value provided for an atmospheric CO2 concentration of 350 ppm. The impact of other stresses (e.g. Temperature, N) are captured through the modifier, Frgr.

gs = Gsmax350 x FRGR x stomatalConductanceCO2Modifier

5.2.7.6.6.1 StomatalConductanceCO2Modifier

StomatalConductanceCO2Modifier = 1 ()

5.2.7.6.7 Senescence and Detachment

Leaf has senescence parameterised to zero so all biomass in this organ will remain alive.

Leaf has detachment parameterised to zero so all biomass in this organ will remain with the plant until a defoliation or harvest event occurs.

5.2.8 StaticCrop

StaticCrop overrides the following properties:

[Command]

The static crop is the base crop and uses all the paramaters values specified above.

5.2.9 StaticTree

StaticTree overrides the following properties:

[Root].MaximumRootDepth.FixedValue = 2600 [Leaf].ExtinctionCoefficient.FixedValue = 0.65 [Leaf].Tallness.FixedValue = 5000 [Root].KLModifier.XYPairs.X = 0, 400, 500, 1000, 1500, 2000 [Root].KLModifier.XYPairs.Y = 1, 1, 0.9, 0.6, 0.3, 0.1

The static tree is similar to the static crop but is taller, has a deeper root system and a lower extinction coefficient

5.2.10 Pasture

Pasture overrides the following properties:

[Root].MaximumRootDepth.FixedValue = 800 [Root].RootFrontVelocity.Potential.FixedValue = 10 [Leaf].ExtinctionCoefficient.FixedValue = 0.65 [Root].KLModifier.XYPairs.X = 0, 200, 300, 500, 1000 [Root].KLModifier.XYPairs.Y = 1, 1, 0.2, 0.03, 0.0 [Leaf].Gsmax350 = 0.004 [Leaf].R50 = 175

The pasture differs from the base crop because its has a RootFrontVelocity of 15 (as opposed to zero) which means the roots grow downward from sowing untill the MaximumRootDepth is reached. It ExtinctionCoefficient and MaximumRootDepth are different to the base StaticCrop and it has a different pattern of kl with depth.

5.2.11 Lucerne

Lucerne overrides the following properties:

[Root].MaximumRootDepth.FixedValue = 2500 [Root].RootFrontVelocity.Potential.FixedValue = 11 [Leaf].ExtinctionCoefficient.FixedValue = 0.8 [Root].KLModifier.XYPairs.X = 0, 200, 500, 1000, 1500, 2000 [Root].KLModifier.XYPairs.Y = 1, 1, 0.5, 0.2, 0.08, 0.03 [Leaf].Gsmax350 = 0.006 [Leaf].R50 = 150

Lucerne also grows root depth like pasture but has a greater root depth, different kl pattern and different extinction coefficient

MortalityRate = 0 ()

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 holding layer structure information

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 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).

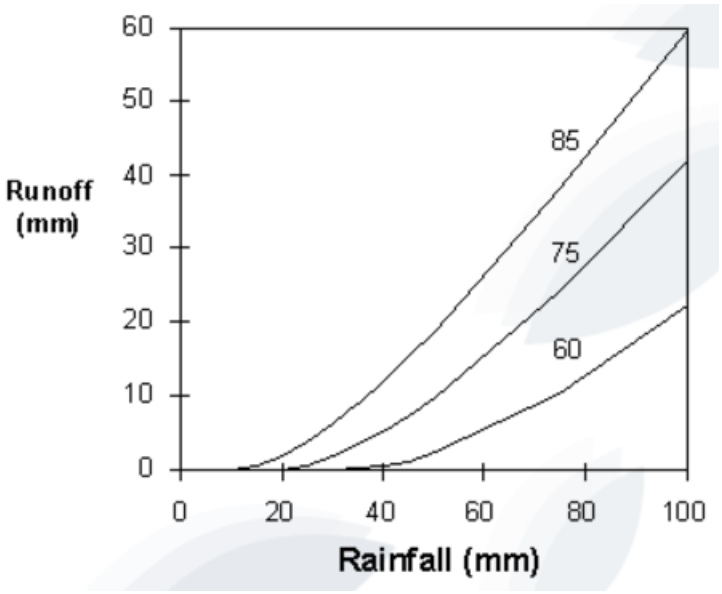
Enhancements beyond CERES and PERFECT include: * the specification of swcon for each layer, being the proportion of soil water above dul that drains in one day * isolation from the code of the coefficients determining diffusivity as a function of soil water (used in calculating unsaturated flow).Choice 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).

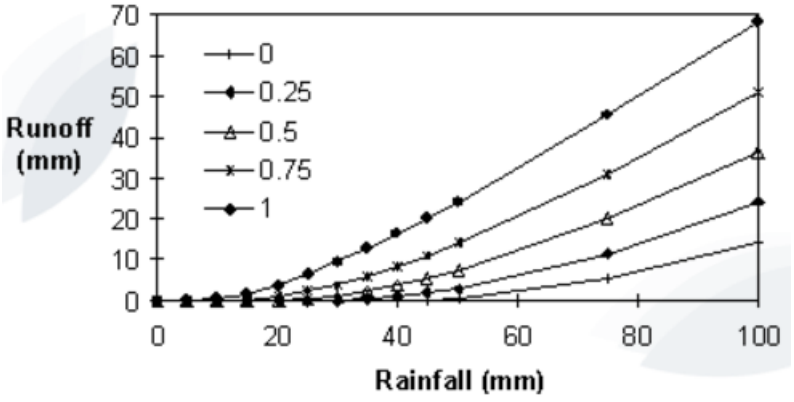
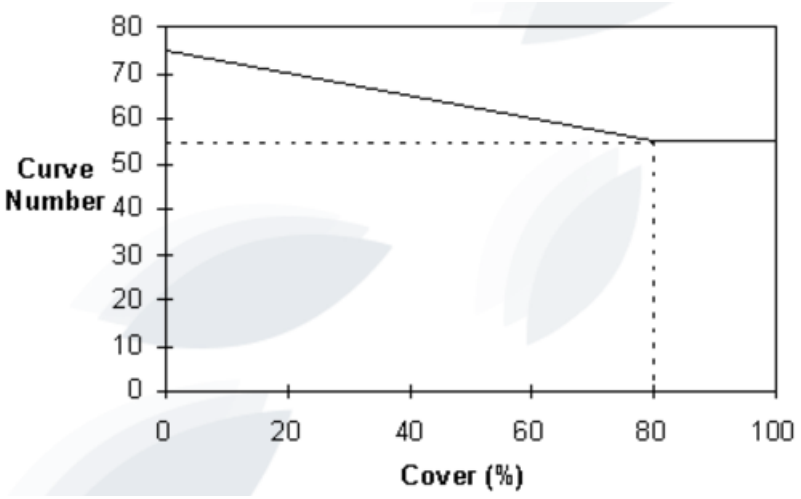


Figure: Residue cover effect on runoff curve number where bare soil curve number is 75 and total reduction in curve number is 20 at 80% cover.

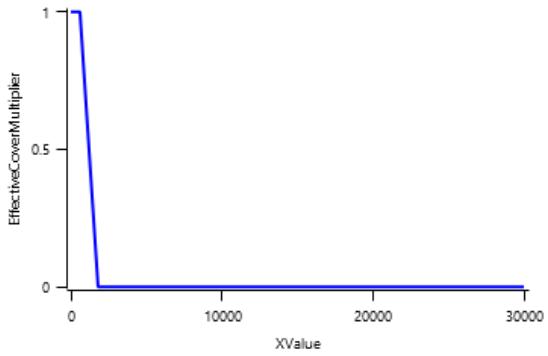


Surface residues inhibit the transport of water across the soil surface during runoff events and so different families of response curves are used according to the amount of crop and residue cover.The extent of the effect on runoff is specified by a threshold surface cover (CNCov), above which there 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.

EffectiveCoverMultiplier is calculated using linear interpolation.



X	Y
0	1
600	1
1800	0
30000	0

XValue = 0 ()

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).

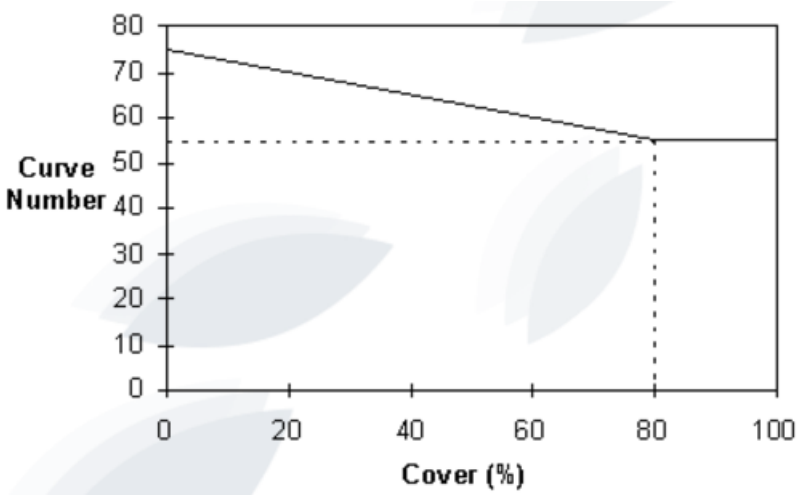
Once the water content of the soil has decreased below a threshold value the rate of supply from the soil will be less than potential evaporation (second stage evaporation). These behaviors are described in SoilWater through the use of two parameters: U and CONA.

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

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)

This class takes soil variables simulated at each of the modelled soil layers and maps them onto a new specified layering. The outputs can be used for producing summaries and to facilitate comparison with observed data.

A model for capturing physical soil parameters

A soil crop parameterization class.

A soil crop parameterization class.

This class encapsulates an operations schedule.

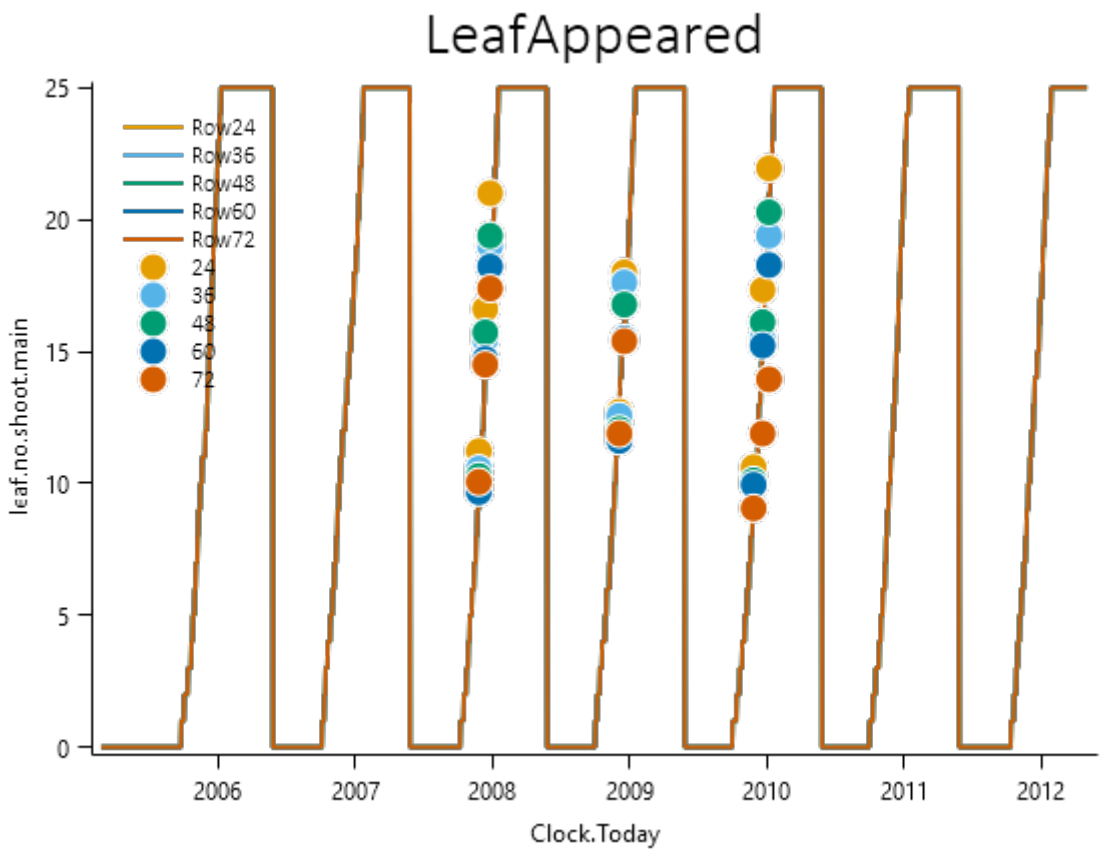
6 Validation

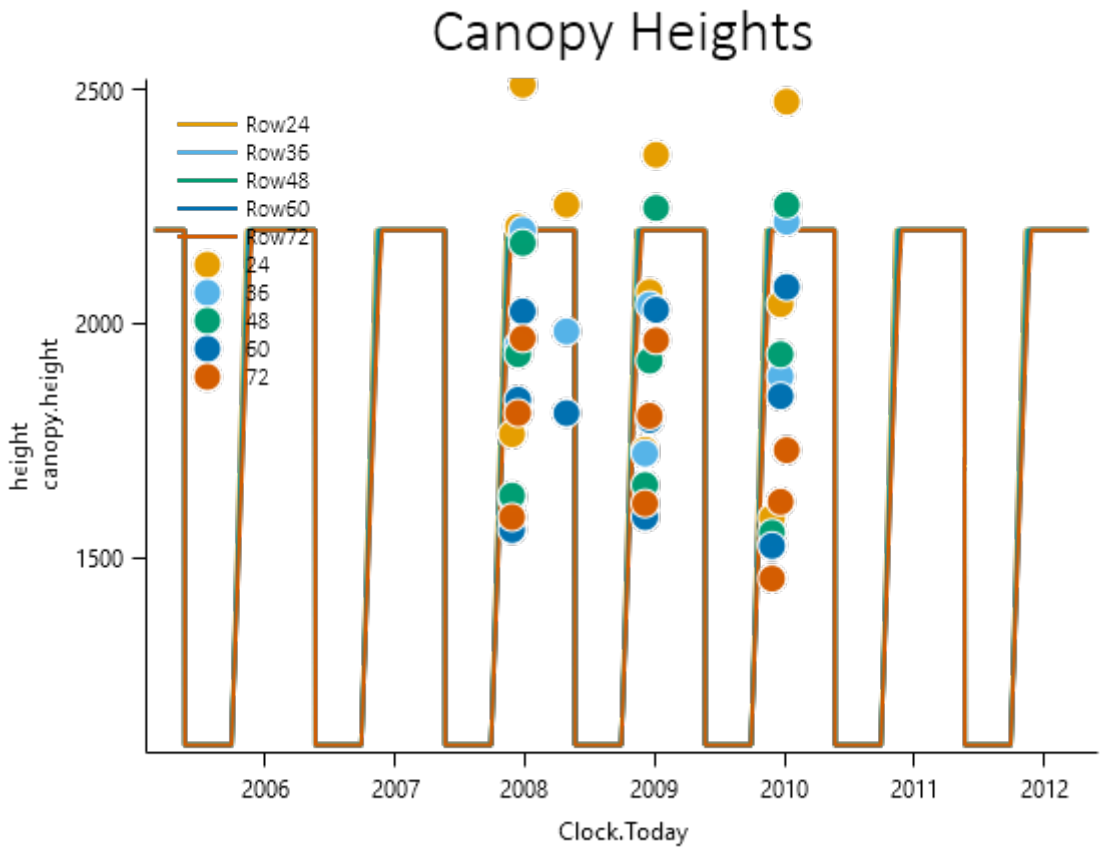
6.1 Sauvignonblanc

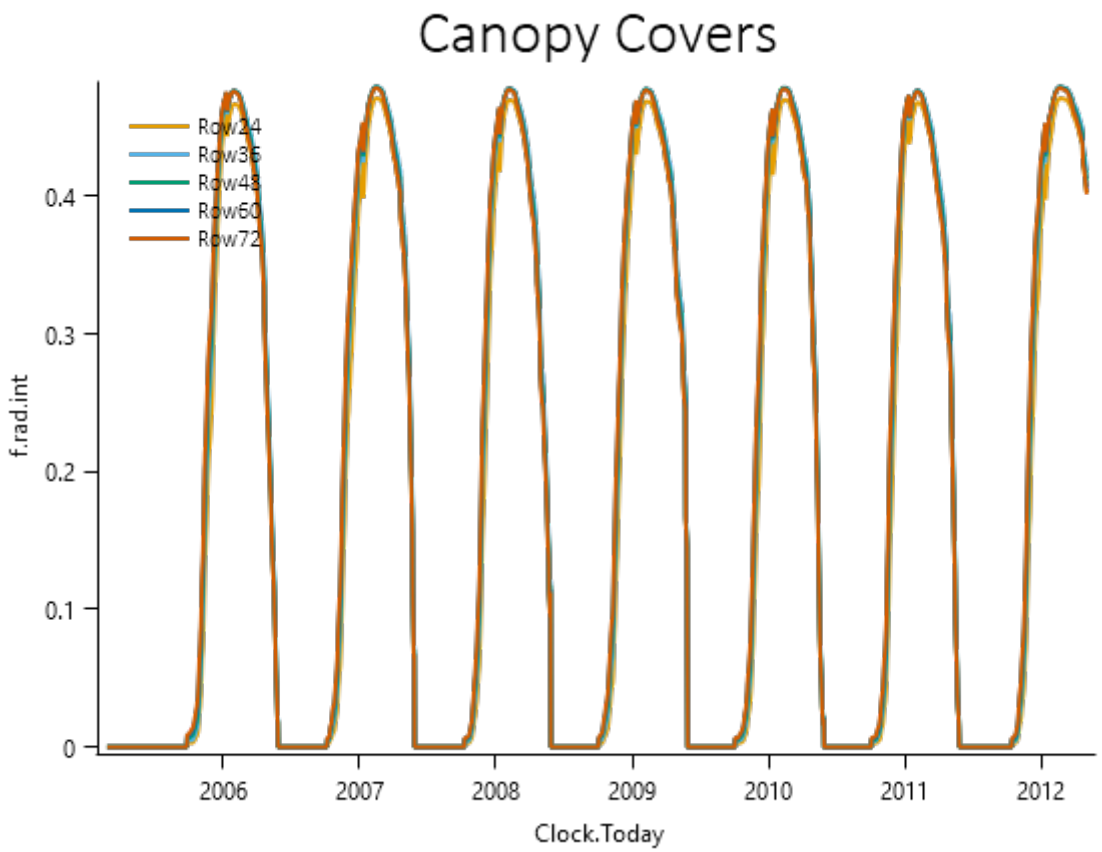
List of experiments.

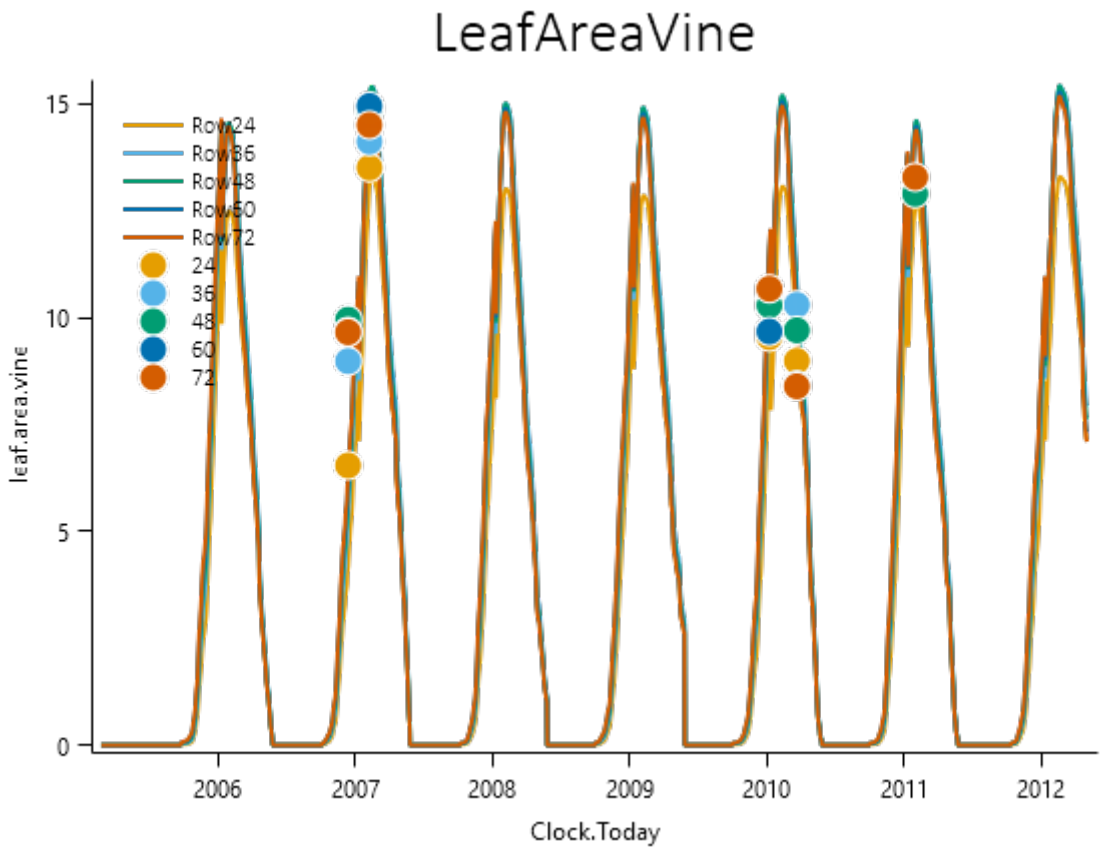
Experiment Name	Design (Number of Treatments)
RetainedNodeNum	Node (5)
Marlborough_2Cane	Climate (4)
Marlborough_3Cane	Climate (2)
Marlborough_4Cane	Climate (4)
Grape_RPC_1	Irri (2)
Grape_RPC_2	Irri (3)
Renwick	Irri (4)
OtherRegions	Climate (5)

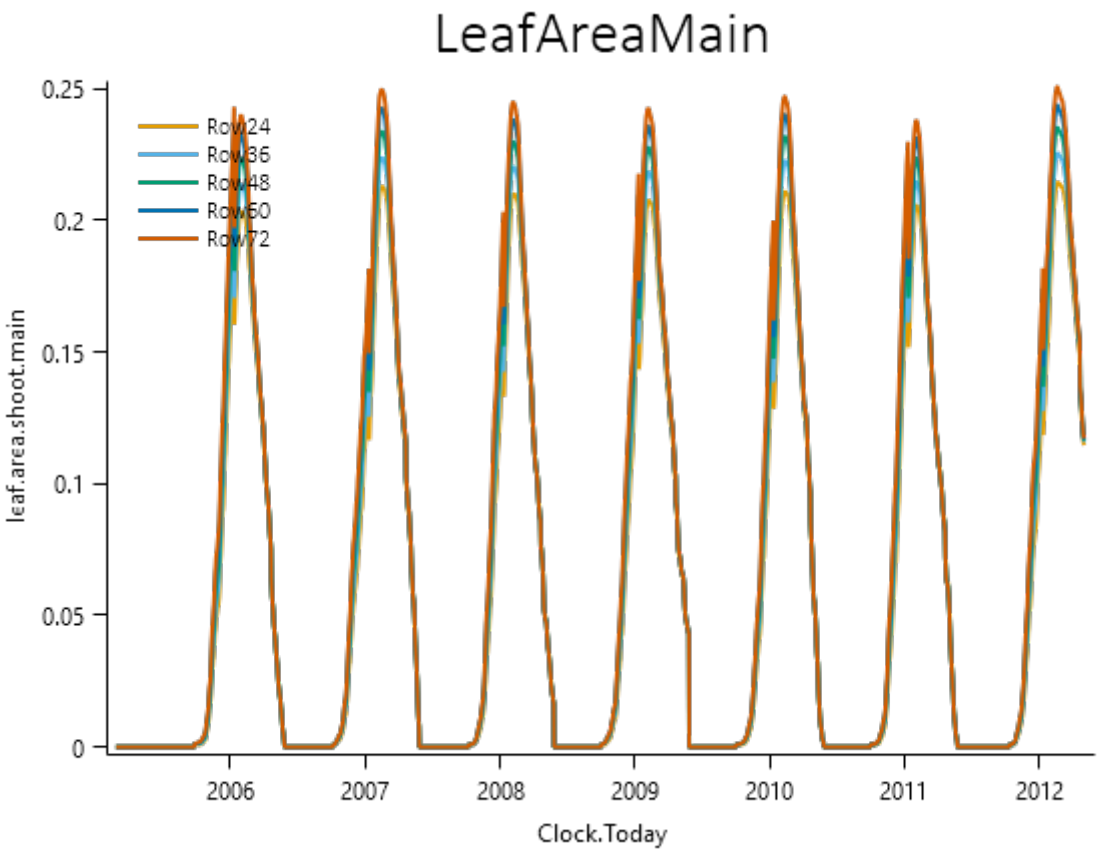
6.1.1 RetainedNodeNum

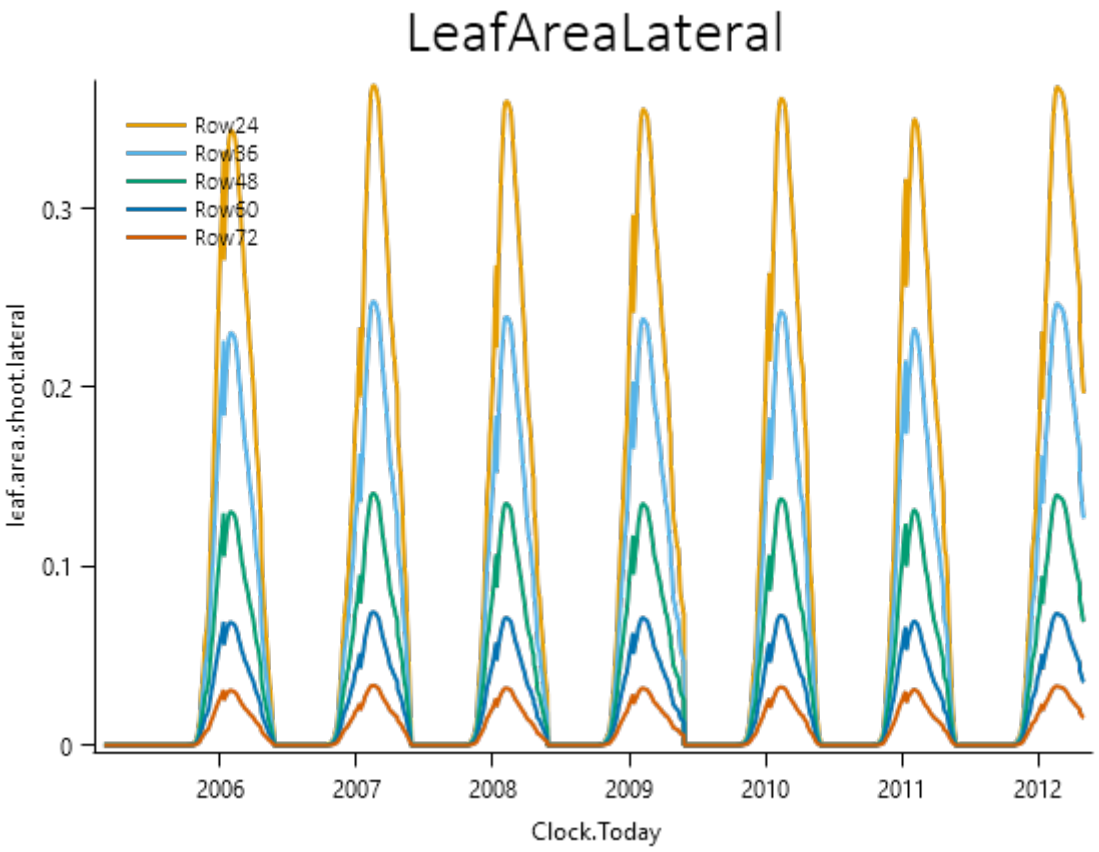


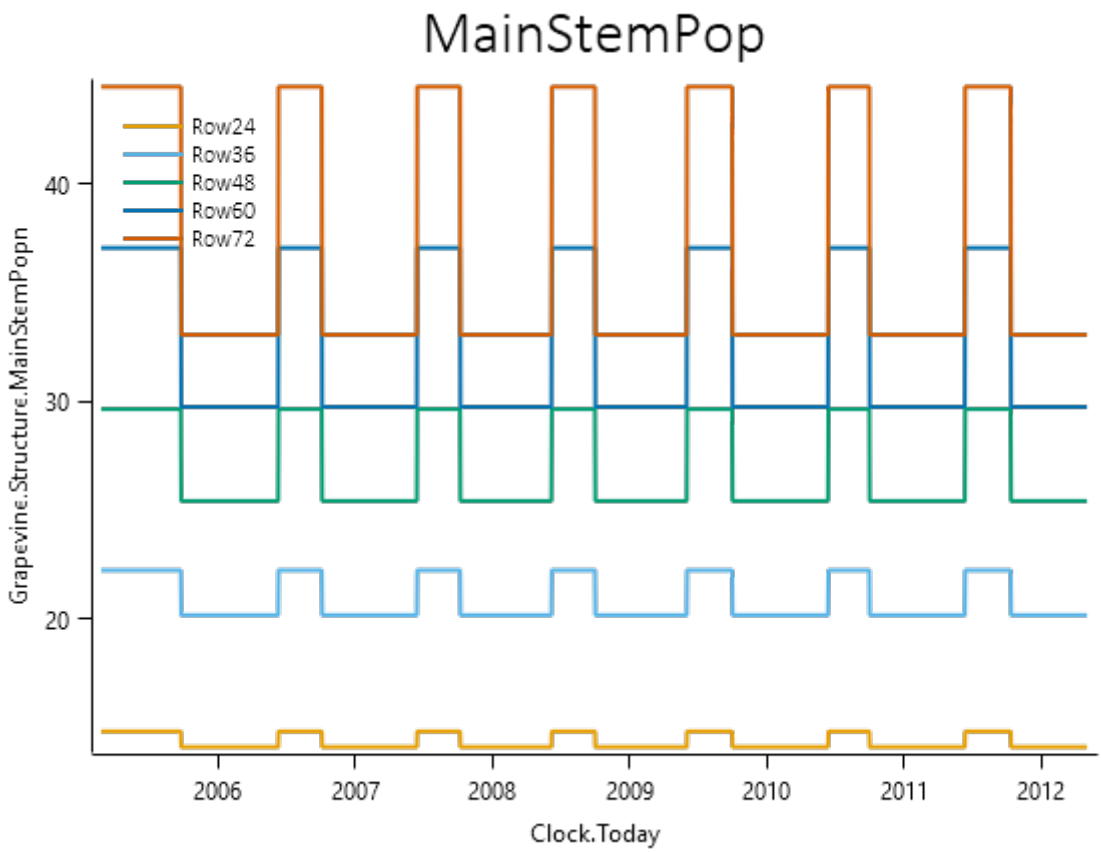


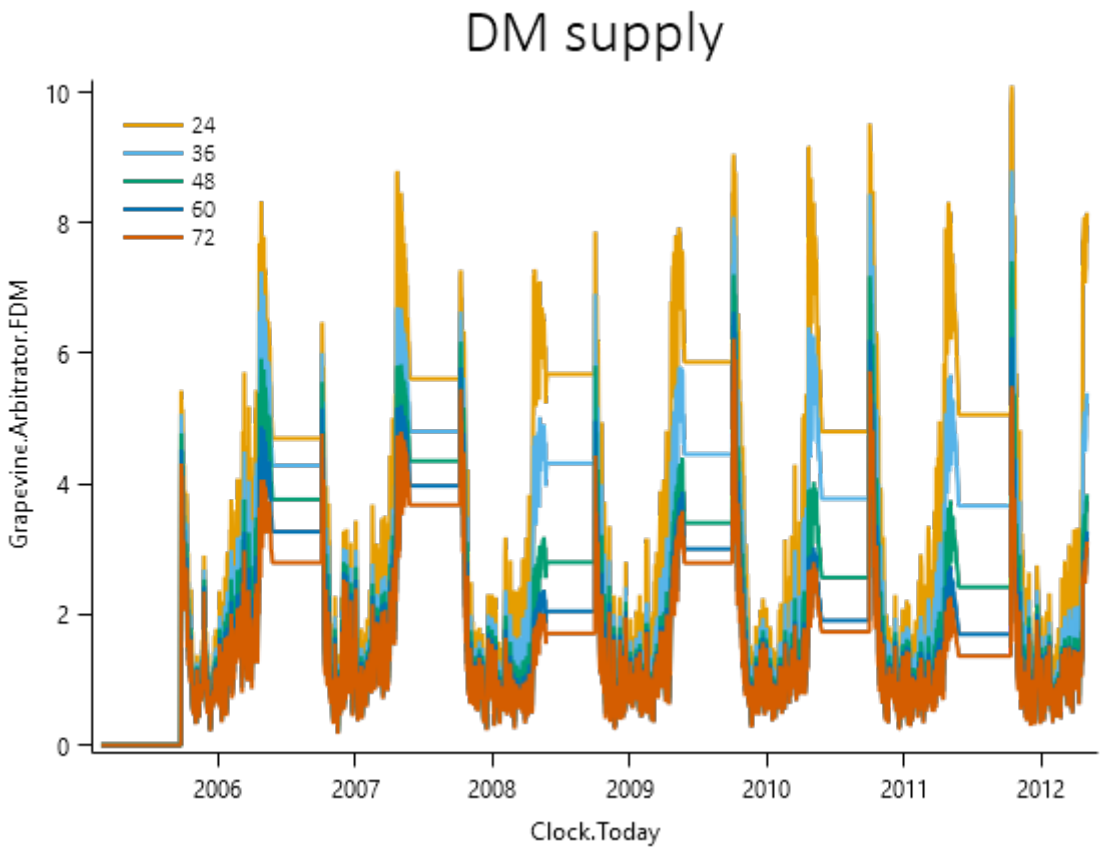

Canopy Covers
Row24
Row36
Row48
Row60
Row72
f.rad.int
0
0.1
0.2
0.3
0.4
2006
2007
2008
2009
2010
2011
2012
Clock.Today

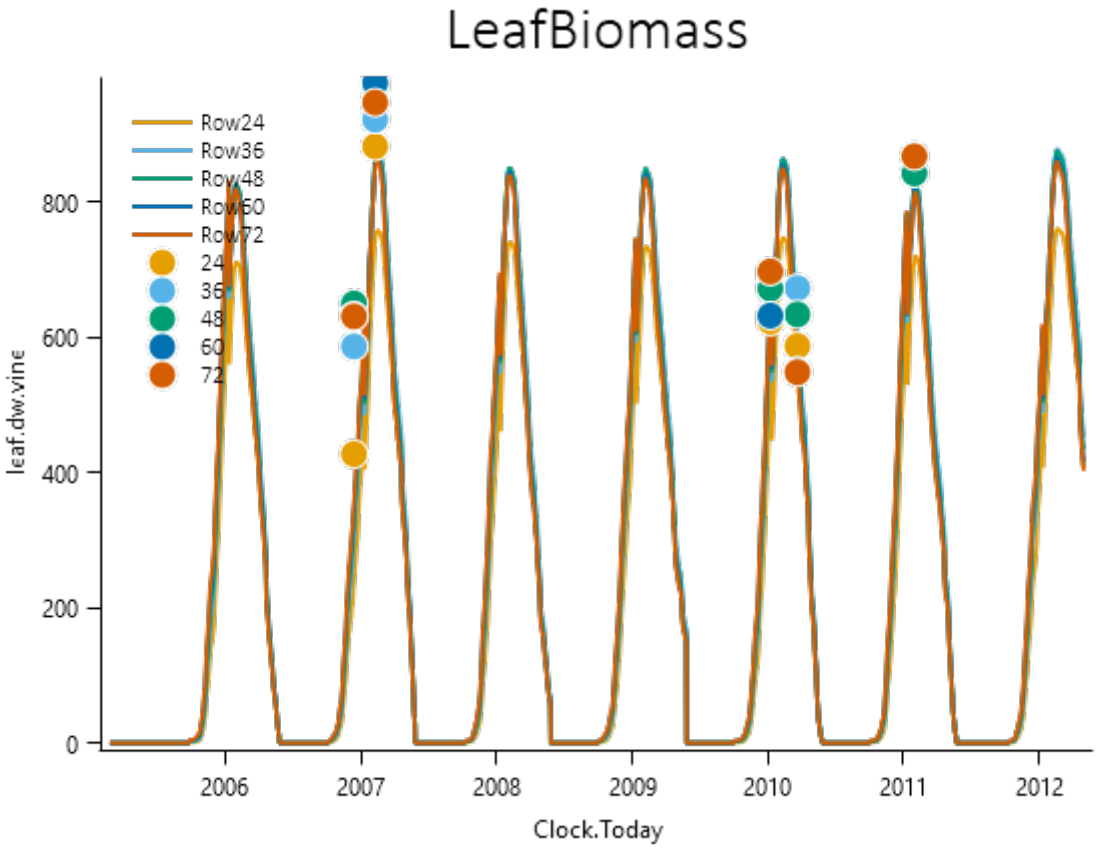

LeafAreaVine
Row24
Row36
Row48
Row60
Row72
24
36
48
60
72
leaf.area.vine
0
5
10
15
2006
2007
2008
2009
2010
2011
2012
Clock.Today

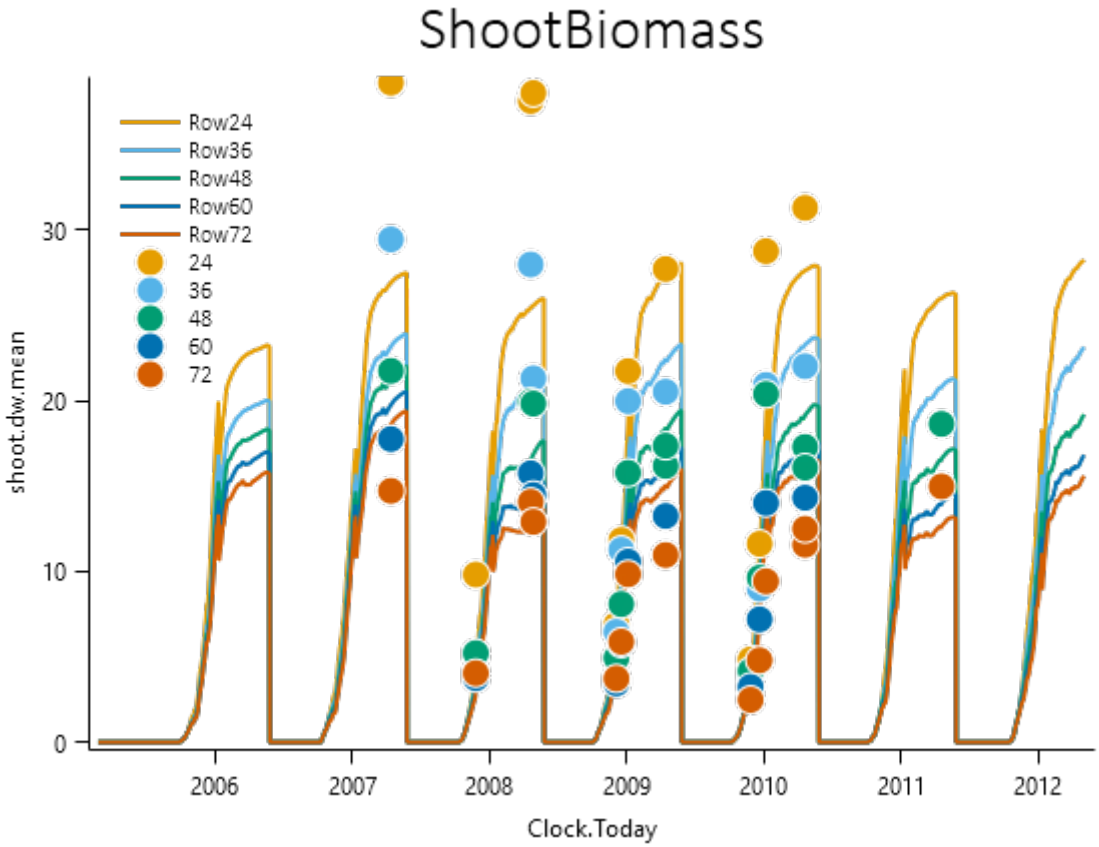

LeafAreaMain
Row24
Row36
Row48
Row60
Row72
leaf.area.shoot.main
0.25
0.2
0.15
0.1
0.05
0
2006
2007
2008
2009
2010
2011
2012
Clock.Today

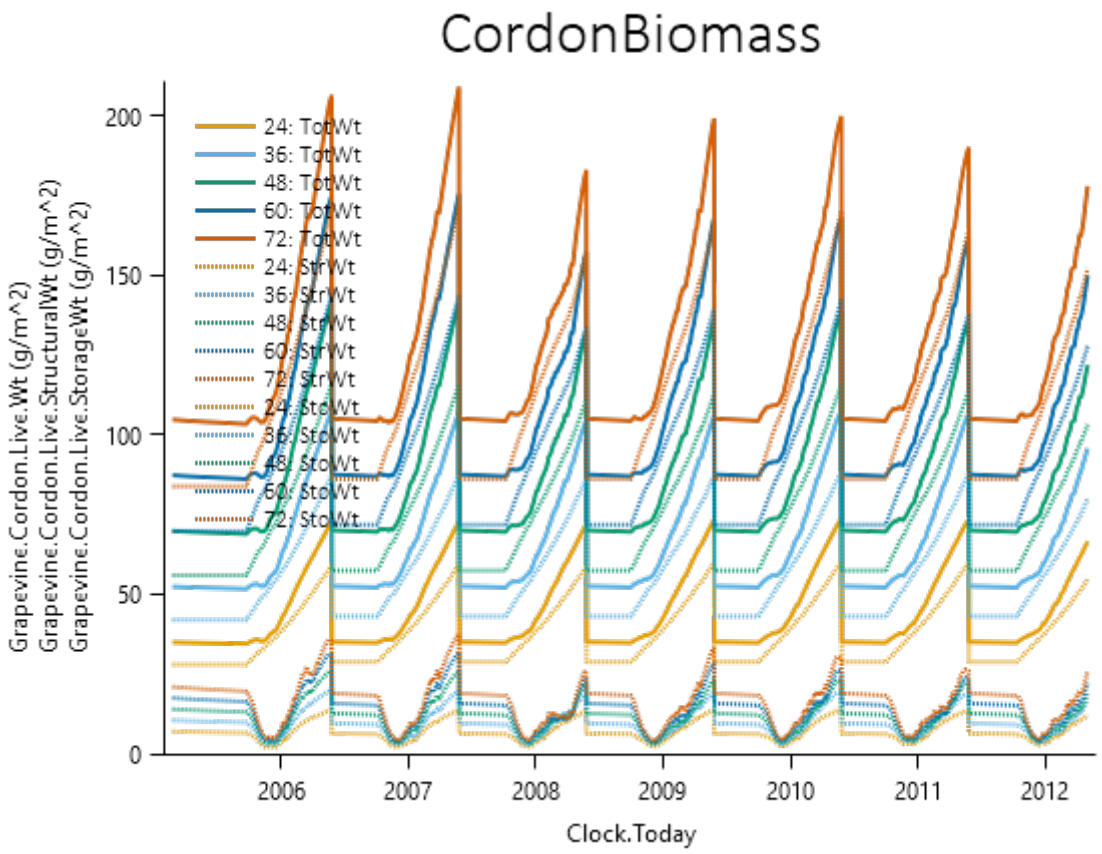

LeafAreaLateral
Row24
Row36
Row48
Row60
Row72
leaf.area.shoot.lateral
0.3
0.2
0.1
0
2006
2007
2008
2009
2010
2011
2012
Clock.Today

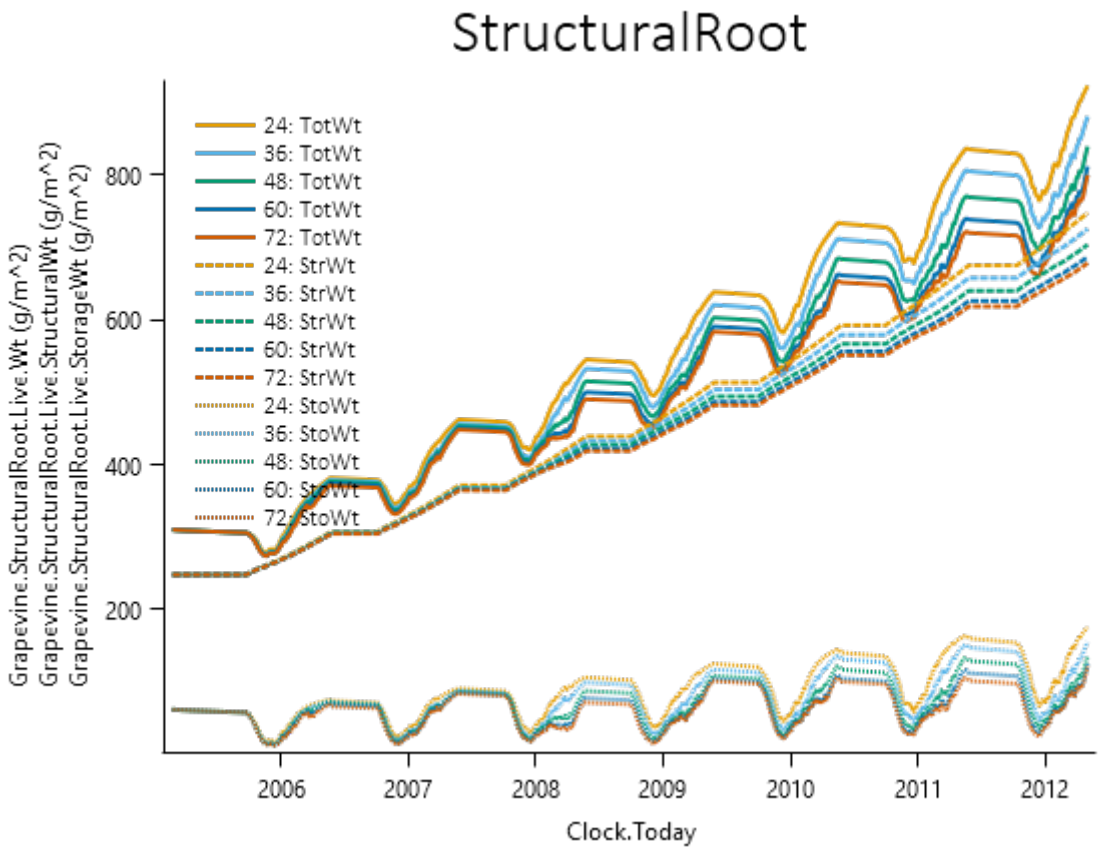

MainStemPop
Row24
Row36
Row48
Row60
Row72
Grapevine.Structure.MainStemPopn
40
30
20
2006
2007
2008
2009
2010
2011
2012
Clock.Today


DM supply
24
36
48
60
72
Grapevine.Arbitrator.FDM
10
8
6
4
2
0
2006
2007
2008
2009
2010
2011
2012
Clock.Today

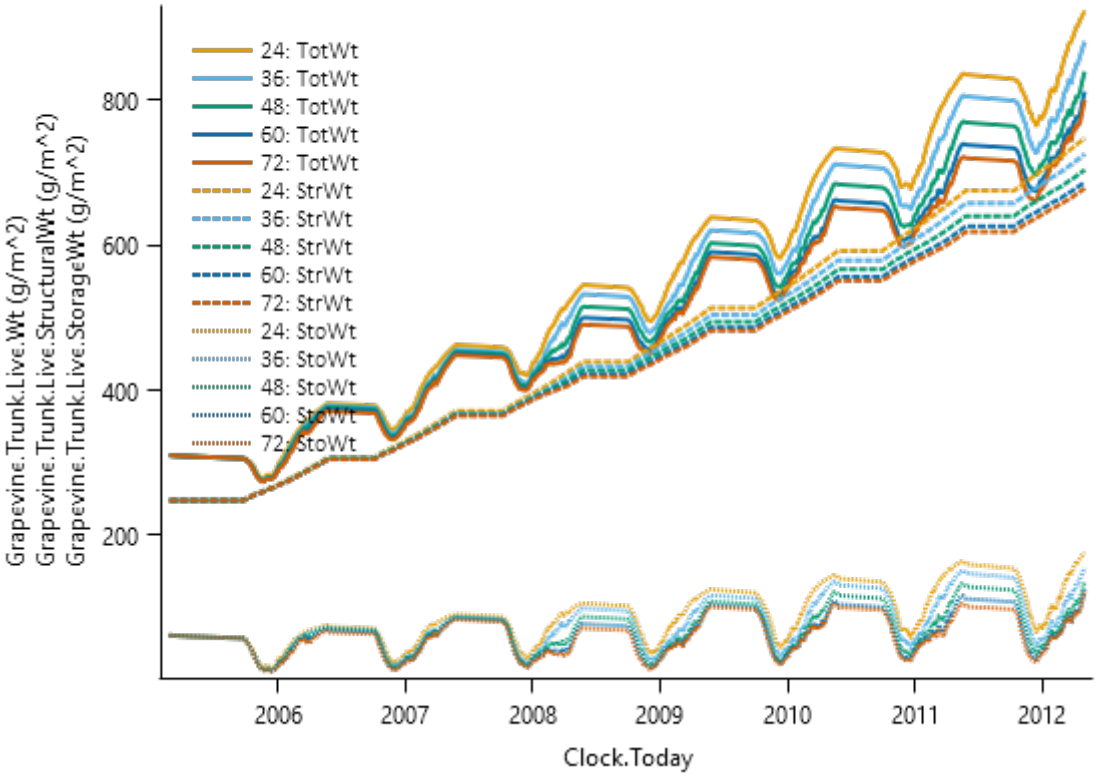

LeafBiomass
Row24
Row36
Row48
Row60
Row72
24
36
48
60
72
leaf.dw.vine
0
200
400
600
800
2006
2007
2008
2009
2010
2011
2012
Clock.Today


ShootBiomass
Row24
Row36
Row48
Row60
Row72
24
36
48
60
72
shoot.dw.mean
0
10
20
30
2006
2007
2008
2009
2010
2011
2012
Clock.Today

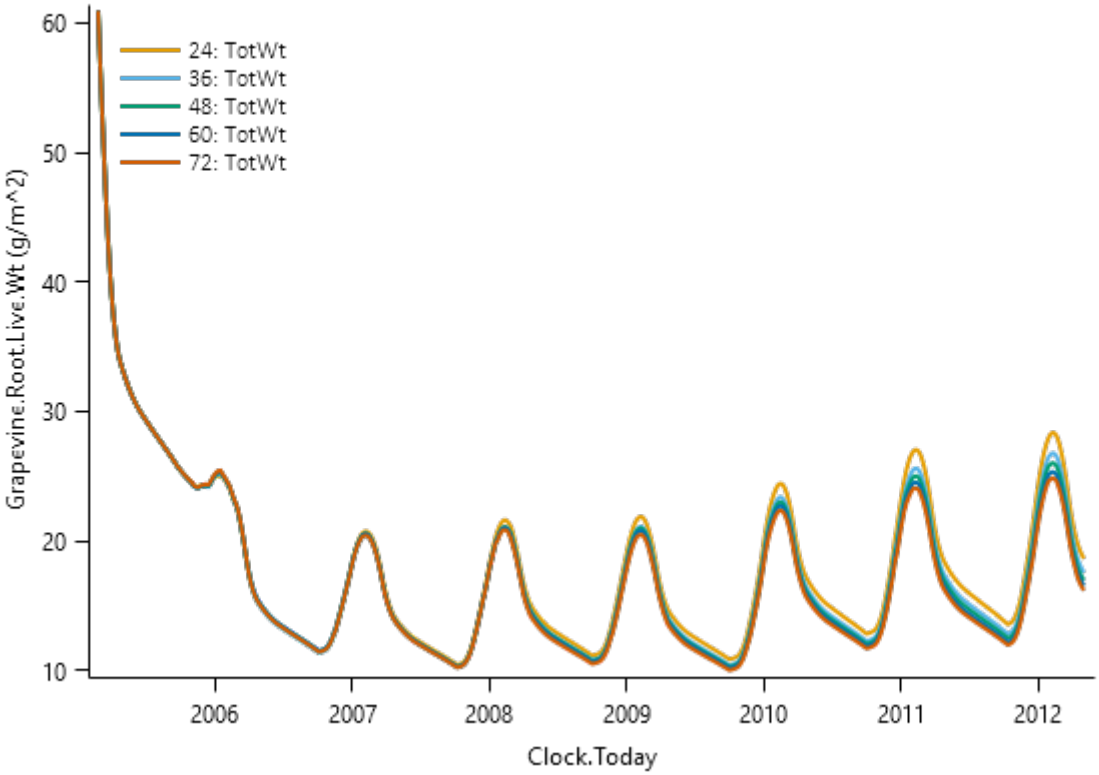

CordonBiomass
Grapevine.Cordon.Live.Wt (g/m^2)
Grapevine.Cordon.Live.StructuralWt (g/m^2)
Grapevine.Cordon.Live.StorageWt (g/m^2)
24: TotWt
36: TotWt
48: TotWt
60: TotWt
72: TotWt
24: StrWt
36: StrWt
48: StrWt
60: StrWt
72: StrWt
24: StoWt
36: StoWt
48: StoWt
60: StoWt
72: StoWt
200
150
100
50
0
2006
2007
2008
2009
2010
2011
2012
Clock.Today

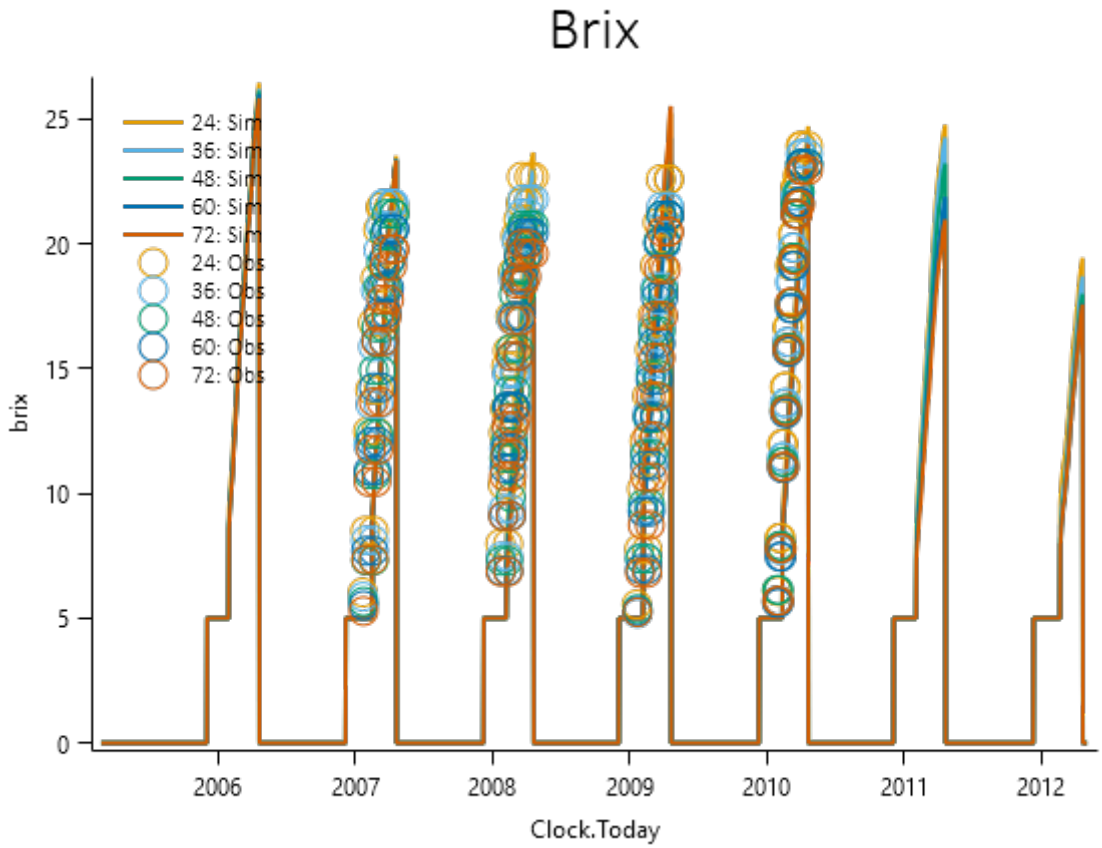

StructuralRoot
Grapevine.StructuralRoot.Live.Wt (g/m^2)
Grapevine.StructuralRoot.Live.StructuralWt (g/m^2)
Grapevine.StructuralRoot.Live.StorageWt (g/m^2)
24: TotWt
36: TotWt
48: TotWt
60: TotWt
72: TotWt
24: StrWt
36: StrWt
48: StrWt
60: StrWt
72: StrWt
24: StoWt
36: StoWt
48: StoWt
60: StoWt
72: StoWt
800
600
400
200
2006
2007
2008
2009
2010
2011
2012
Clock.Today

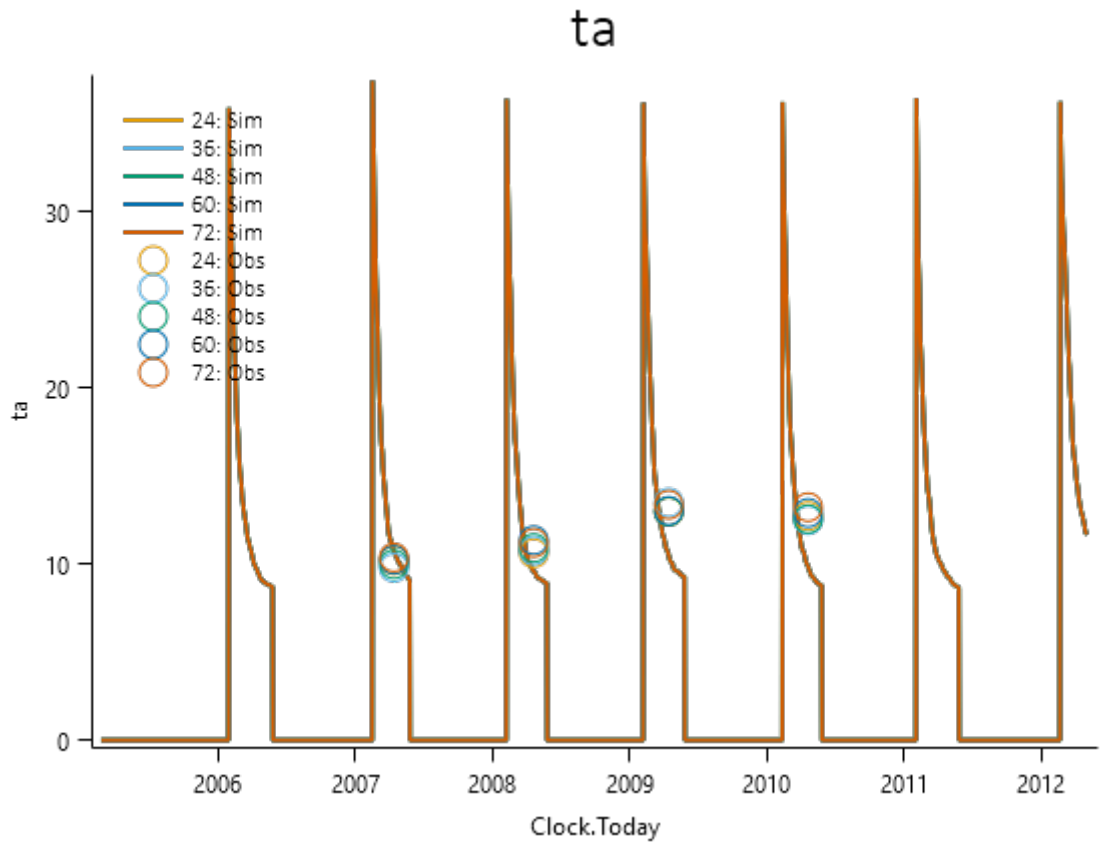
TrunkBiomass

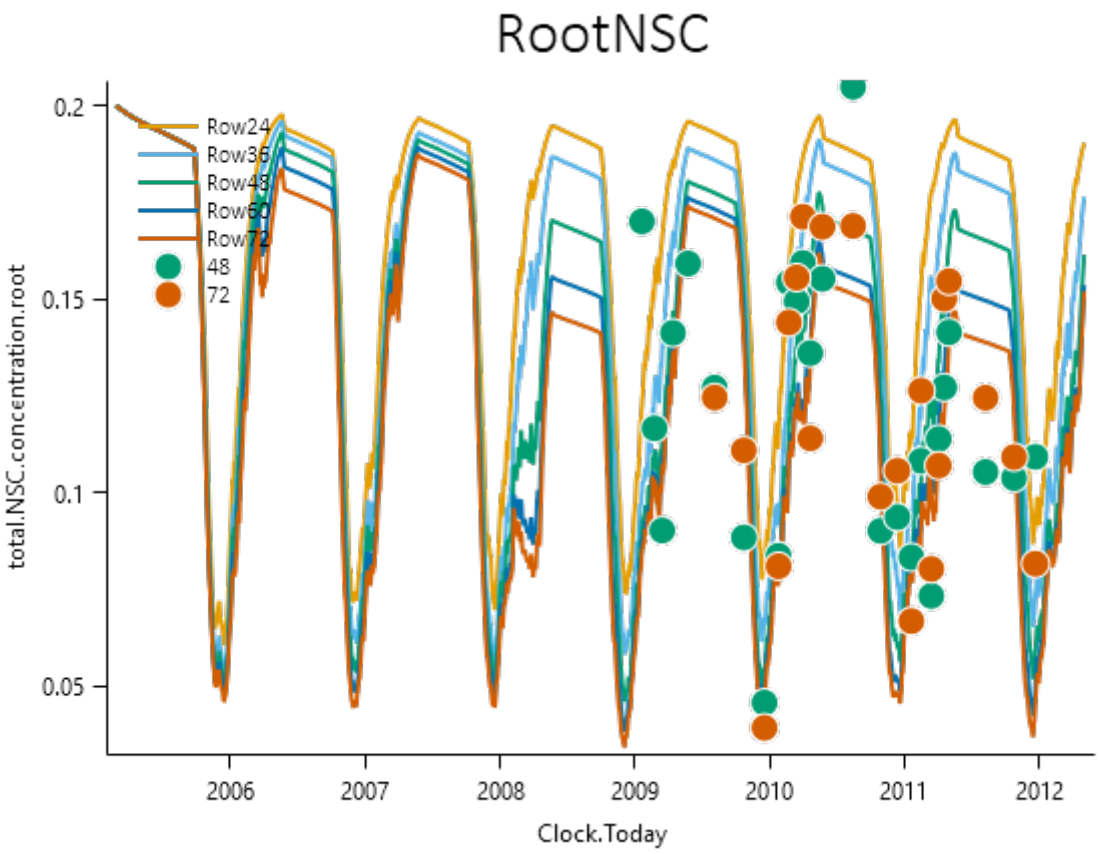


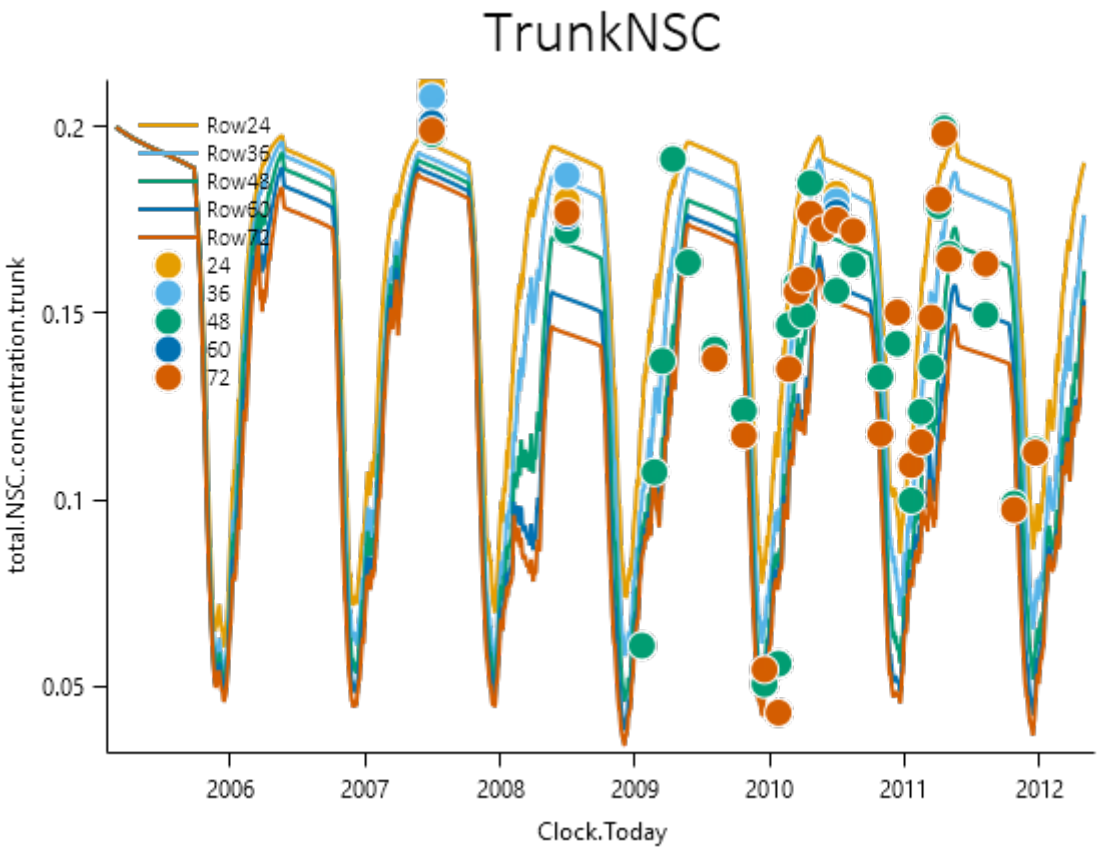
RootBiomass

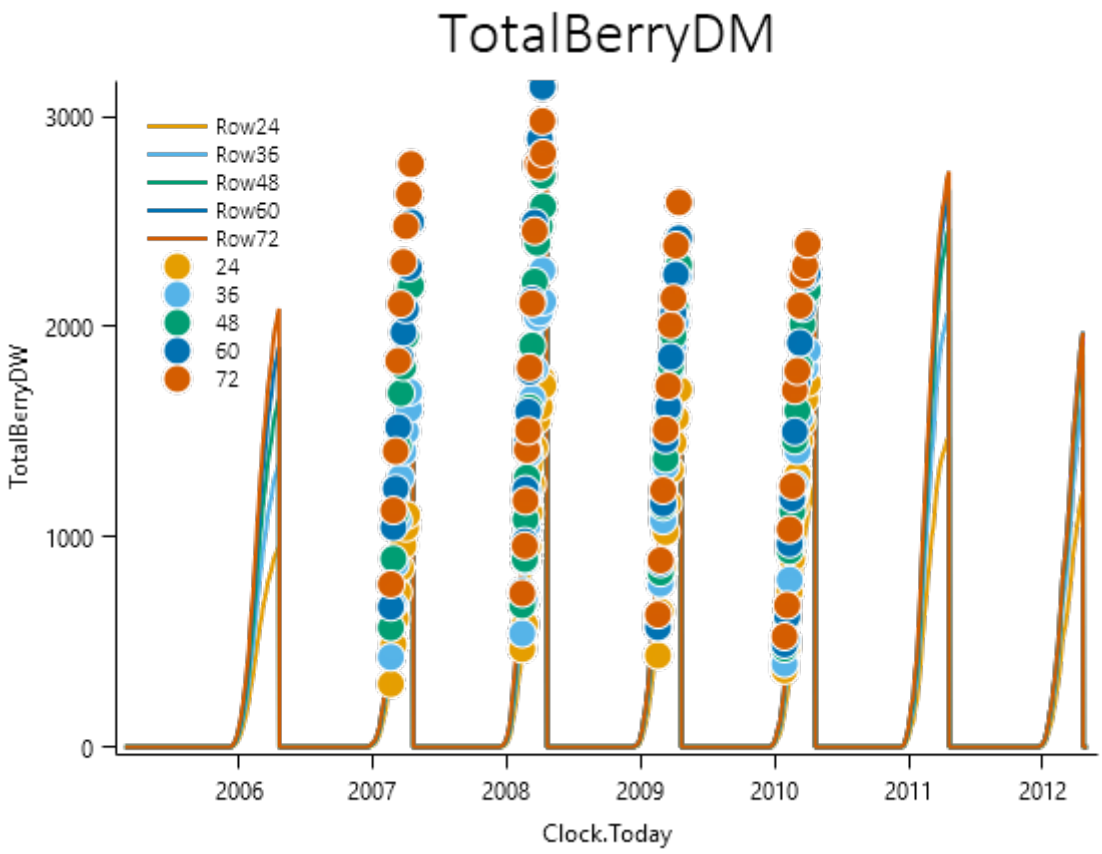


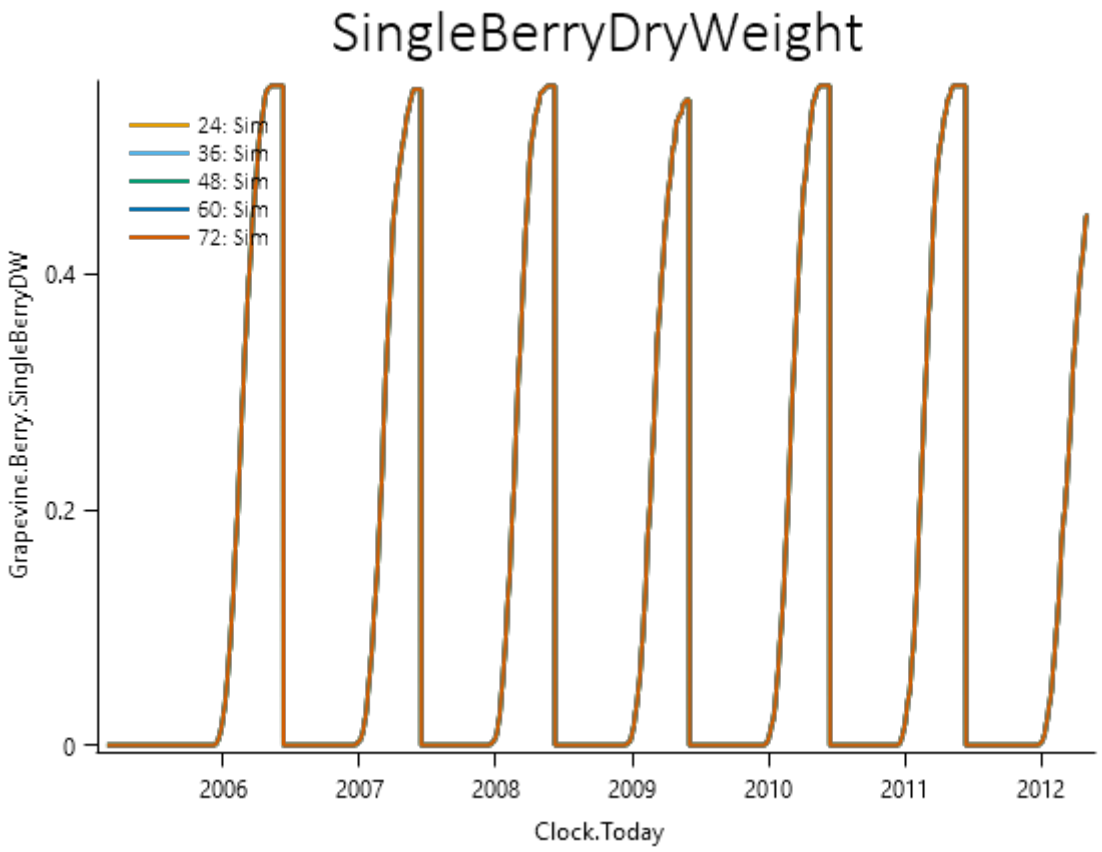

Brix
24: Sim
36: Sim
48: Sim
60: Sim
72: Sim
24: Obs
36: Obs
48: Obs
60: Obs
72: Obs
brix
0
5
10
15
20
25
2006
2007
2008
2009
2010
2011
2012
Clock.Today

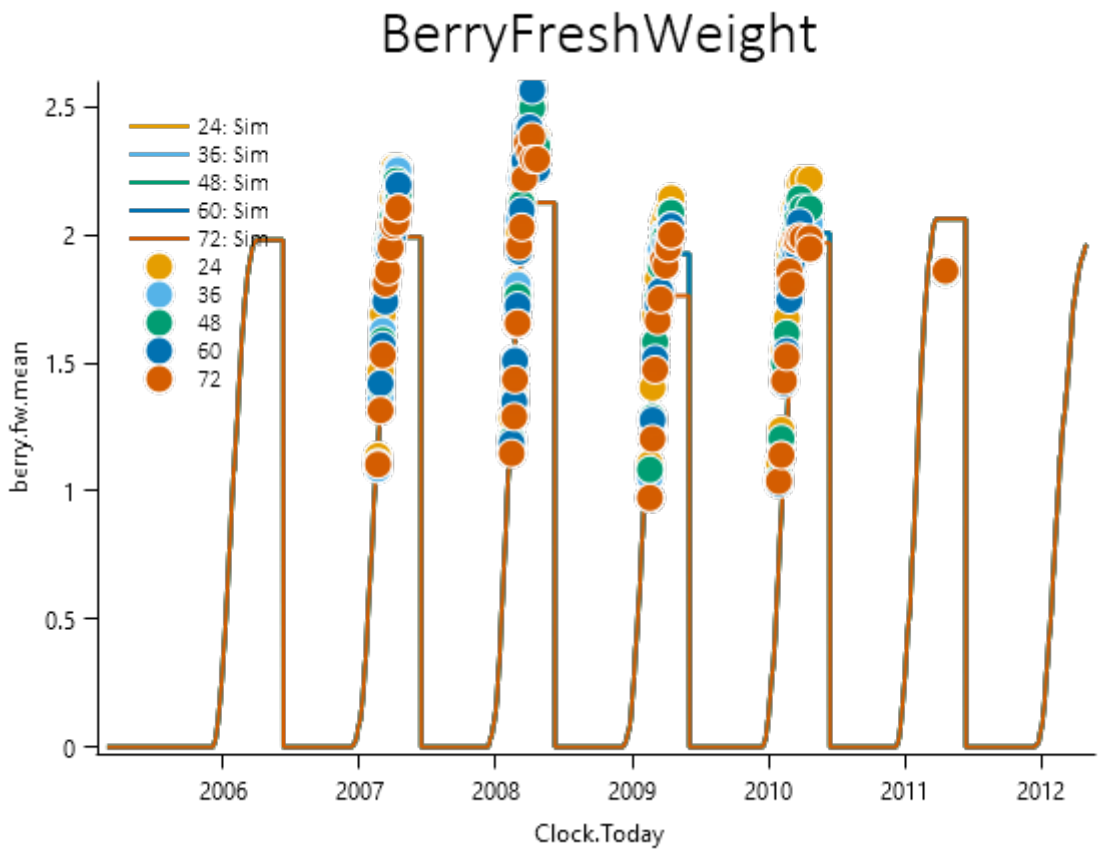

ta
24: Sim
36: Sim
48: Sim
60: Sim
72: Sim
24: Obs
36: Obs
48: Obs
60: Obs
72: Obs
ta
0
10
20
30
2006
2007
2008
2009
2010
2011
2012
Clock.Today

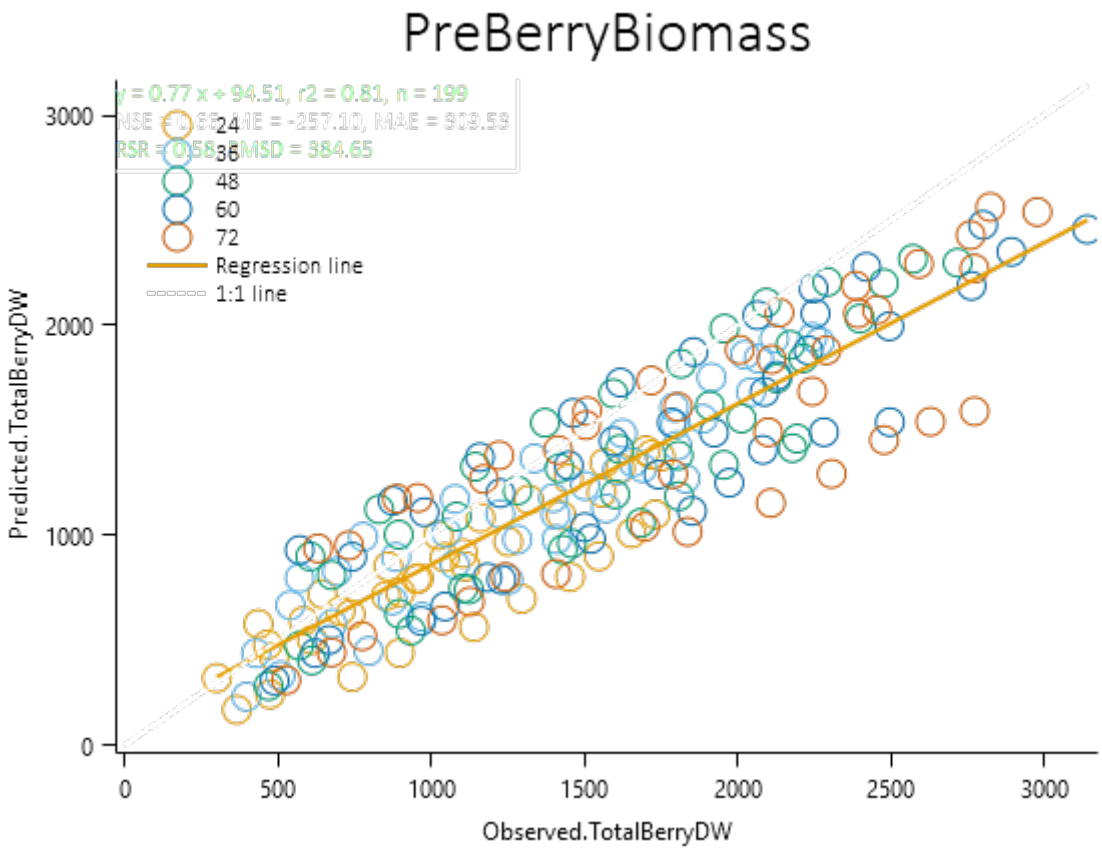

RootNSC
total.NSC.concentration.root
0.2
0.15
0.1
0.05
Row24
Row36
Row48
Row60
Row72
48
72
2006
2007
2008
2009
2010
2011
2012
Clock.Today

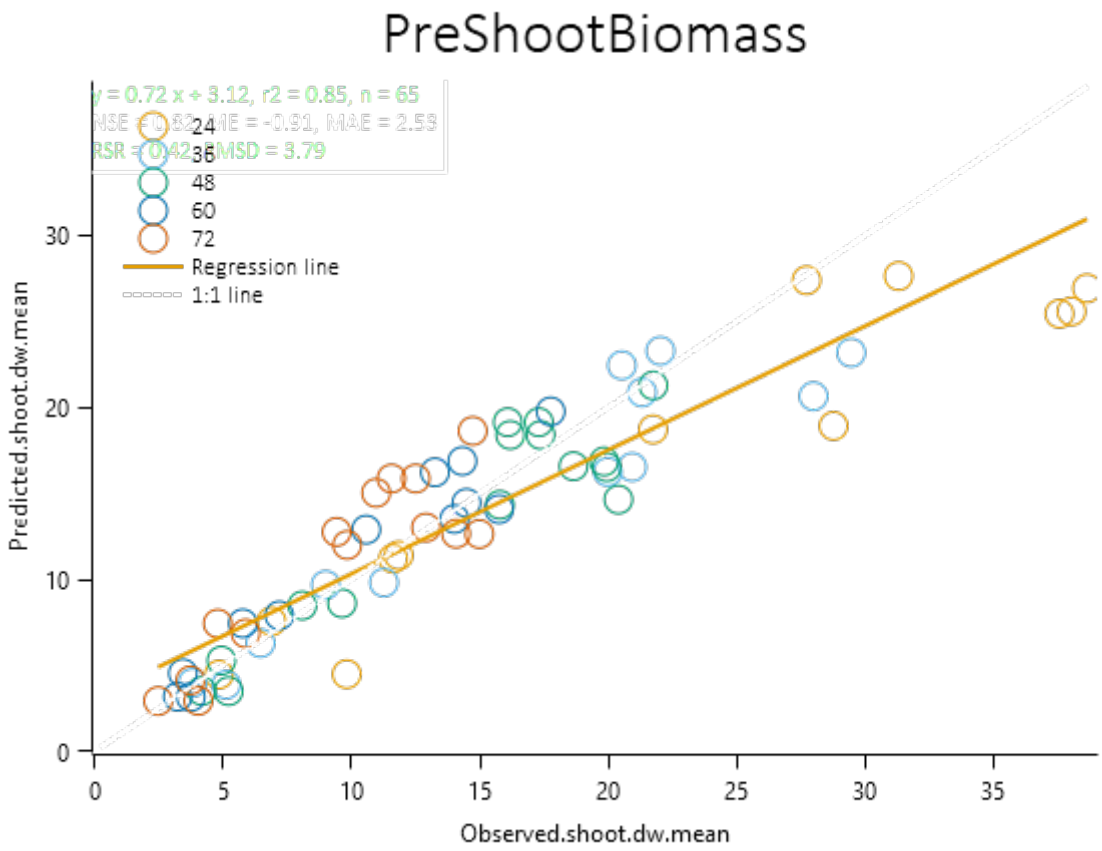

TrunkNSC
total.NSC.concentration.trunk
0.2
0.15
0.1
0.05
Row24
Row36
Row48
Row60
Row72
24
36
48
60
72
2006
2007
2008
2009
2010
2011
2012
Clock.Today

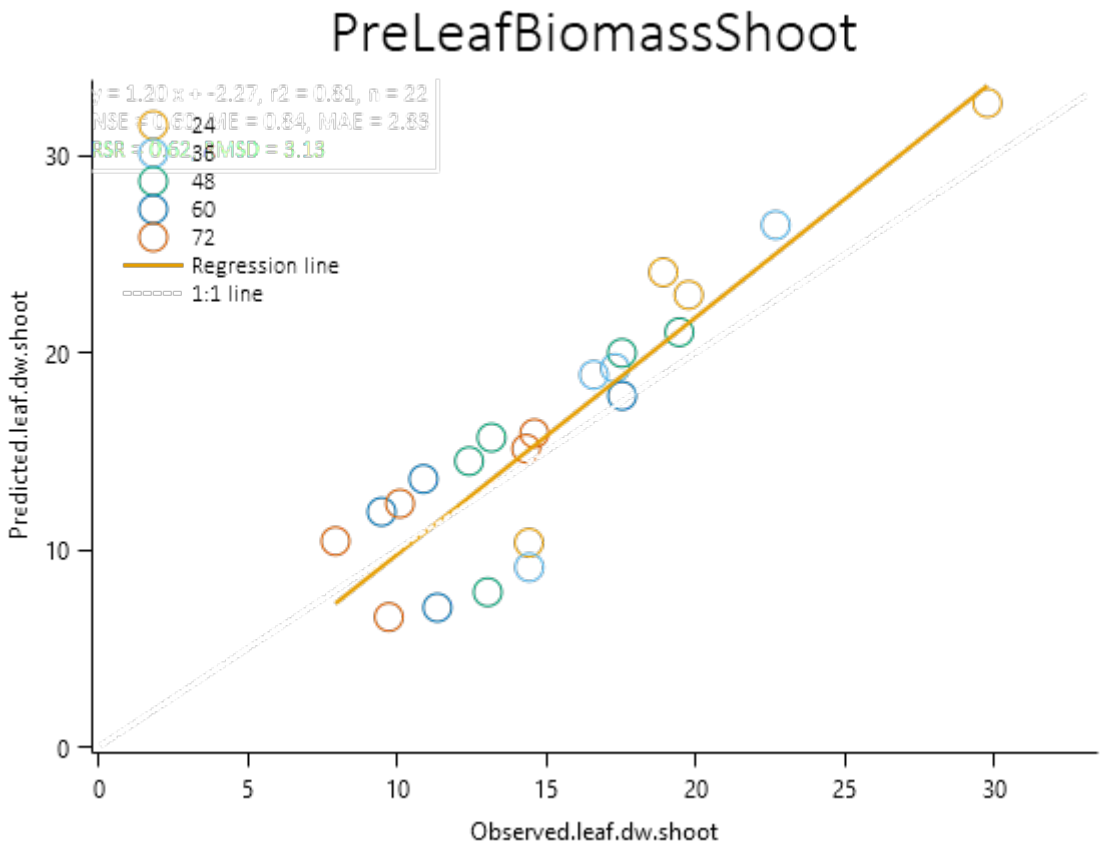

TotalBerryDM
Row24
Row36
Row48
Row60
Row72
24
36
48
60
72
TotalBerryDW
3000
2000
1000
0
2006
2007
2008
2009
2010
2011
2012
Clock.Today

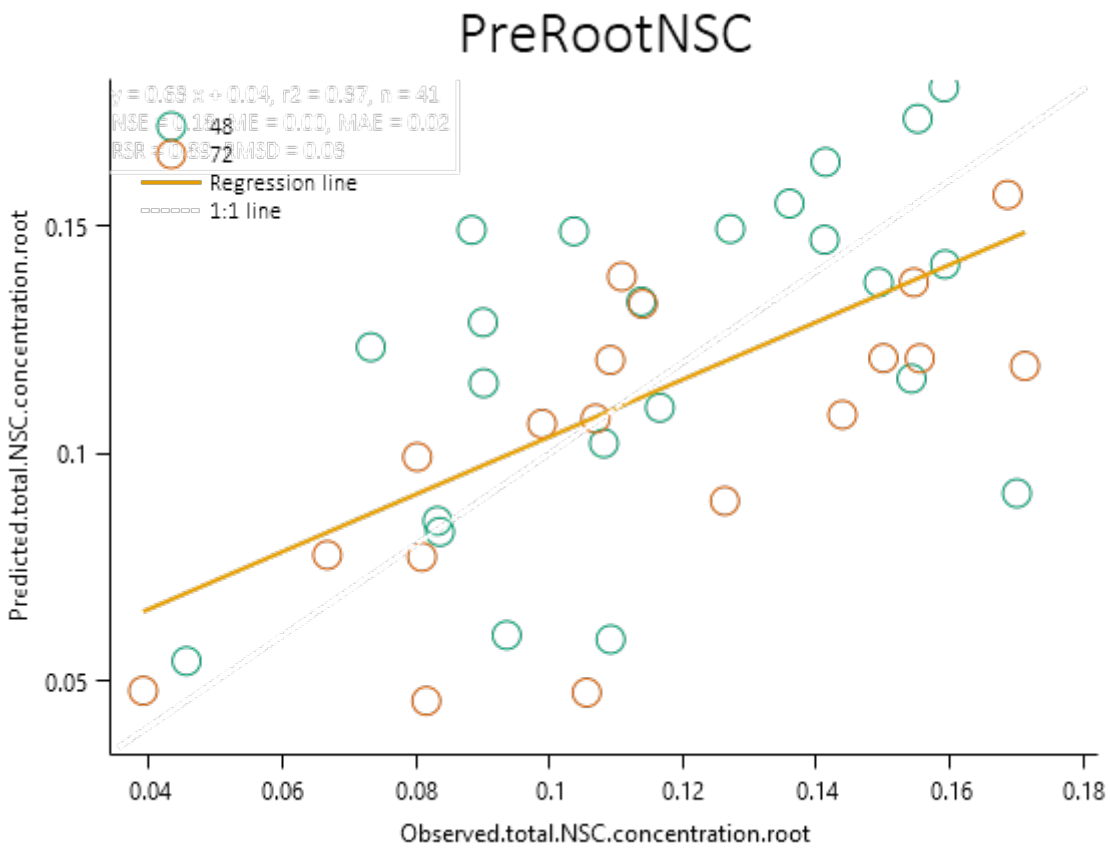

SingleBerryDryWeight
24: Sim
36: Sim
48: Sim
60: Sim
72: Sim
Grapevine.Berry.SingleBerryDW
0.4
0.2
0
2006
2007
2008
2009
2010
2011
2012
Clock.Today

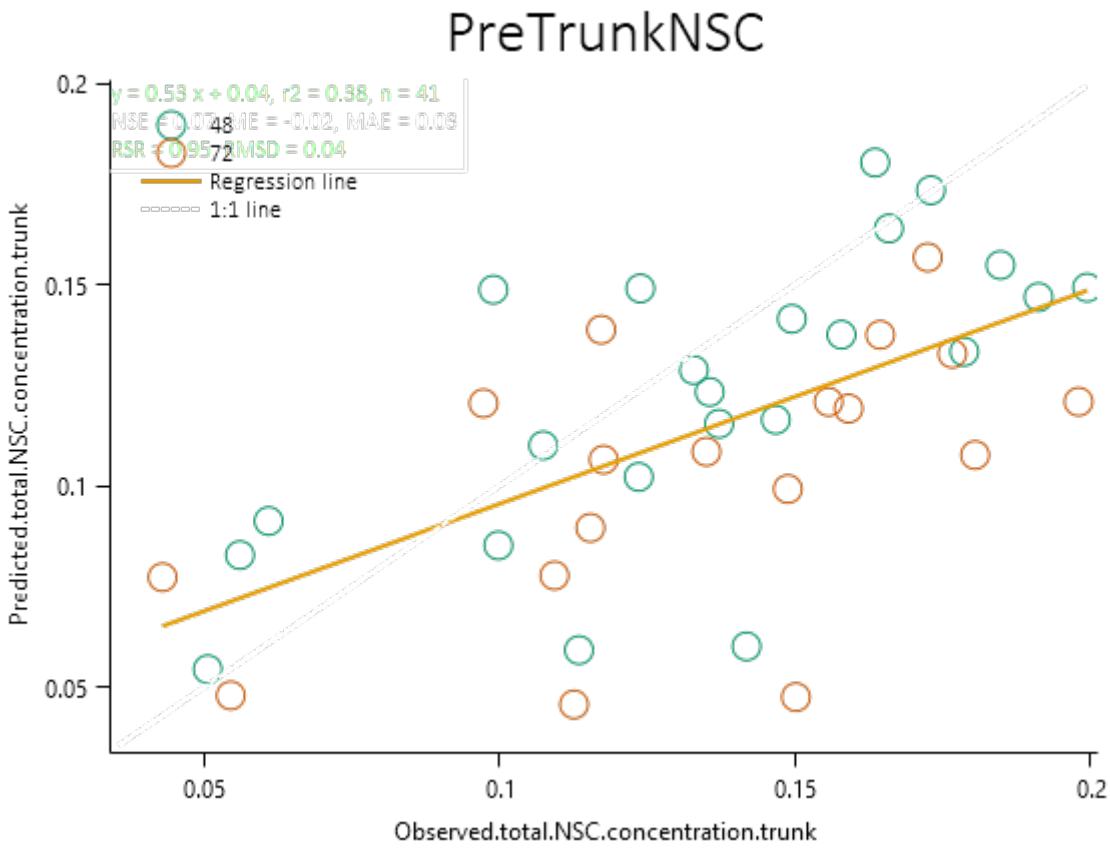

BerryFreshWeight
24: Sim
36: Sim
48: Sim
60: Sim
72: Sim
24
36
48
60
72
berry.fw.mean
Clock.Today
0
0.5
1
1.5
2
2.5
2006
2007
2008
2009
2010
2011
2012


PreBerryBiomass
y = 0.77 x + 94.51, r2 = 0.81, n = 199
RSR = 0.58, RMSD = 384.65
24
36
48
60
72
Regression line
1:1 line
Predicted.TotalBerryDW
Observed.TotalBerryDW
0
1000
2000
3000
0
500
1000
1500
2000
2500
3000


PreShootBiomass
y = 0.72 x + 3.12, r2 = 0.85, n = 65
NSE = 0.82, ME = -0.91, MAE = 2.53
RSR = 0.42, RMSD = 3.79
24
36
48
60
72
Regression line
1:1 line
Predicted.shoot.dw.mean
Observed.shoot.dw.mean
0
10
20
30
0
5
10
15
20
25
30
35

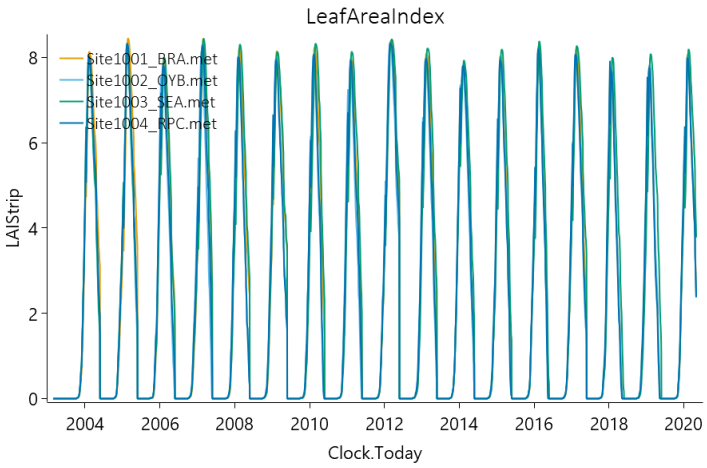

PreLeafBiomassShoot
y = 1.20 x + -2.27, r2 = 0.81, n = 22
NSE = 0.60, ME = 0.84, MAE = 2.69
RSR = 0.62, RMSD = 3.13
24
36
48
60
72
Regression line
1:1 line
Predicted.leaf.dw.shoot
Observed.leaf.dw.shoot
0
10
20
30
0
5
10
15
20
25
30

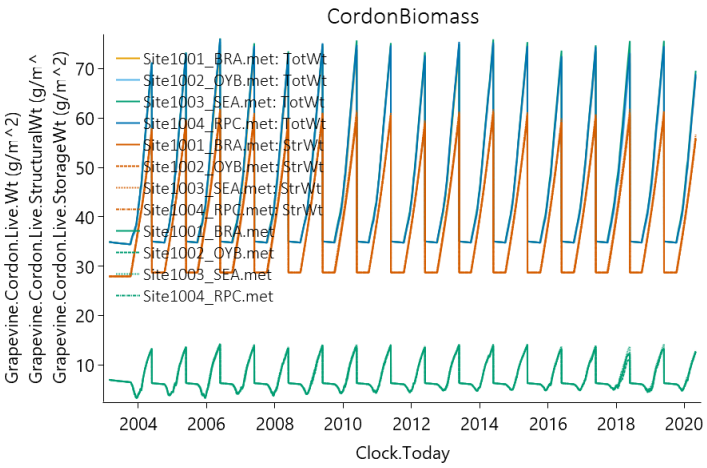

PreRootNSC
y = 0.68 x + 0.04, r2 = 0.37, n = 41
NSE = 0.18, ME = 0.00, MAE = 0.02
RSR = 0.85, MSD = 0.03
48
72
Regression line
1:1 line
Predicted.total.NSC.concentration.root
0.05
0.1
0.15
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
Observed.total.NSC.concentration.root

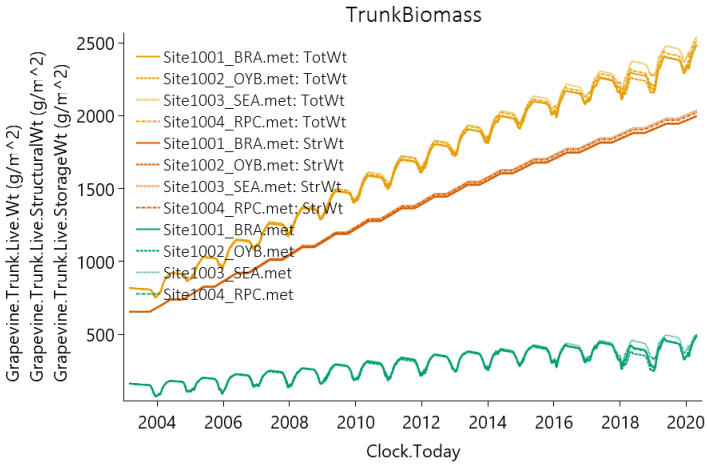

PreTrunkNSC
y = 0.53 x + 0.04, r2 = 0.38, n = 41
NSE = 0.07, ME = -0.02, MAE = 0.03
RSR = 0.95, MSD = 0.04
48
72
Regression line
1:1 line
Predicted.total.NSC.concentration.trunk
0.05
0.1
0.15
0.2
0.05
0.1
0.15
0.2
Observed.total.NSC.concentration.trunk

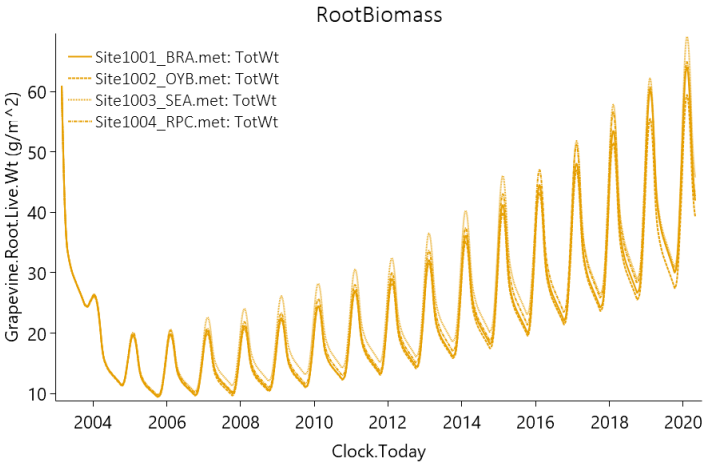
6.1.2 Marlborough_2Cane

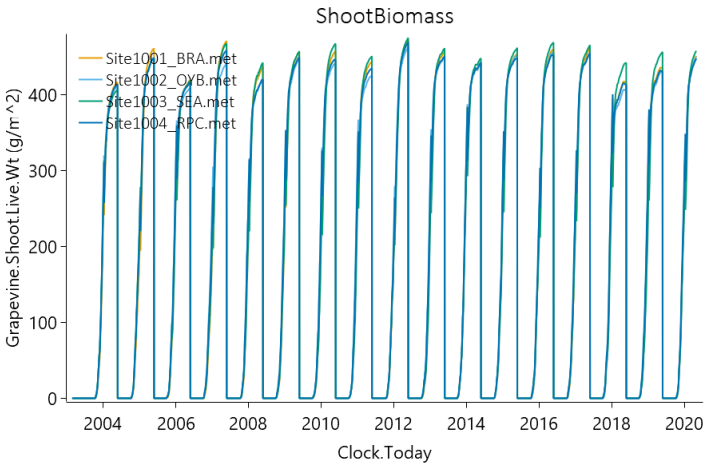
6.1.2.1 Figures

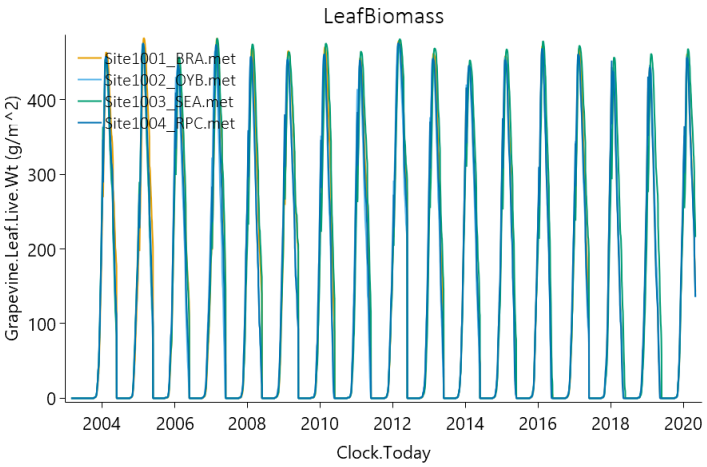




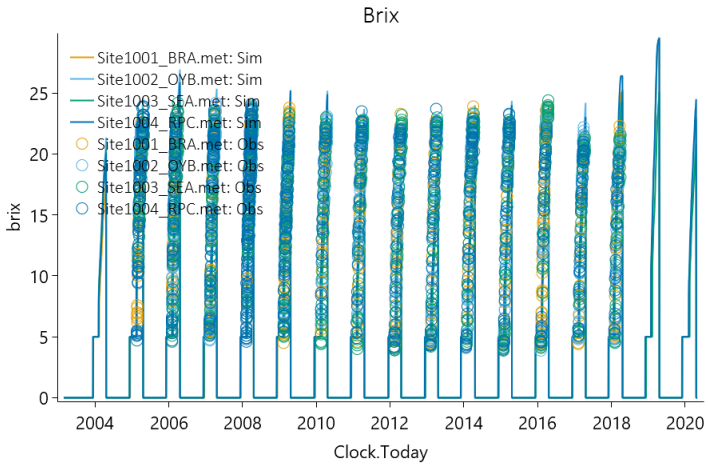


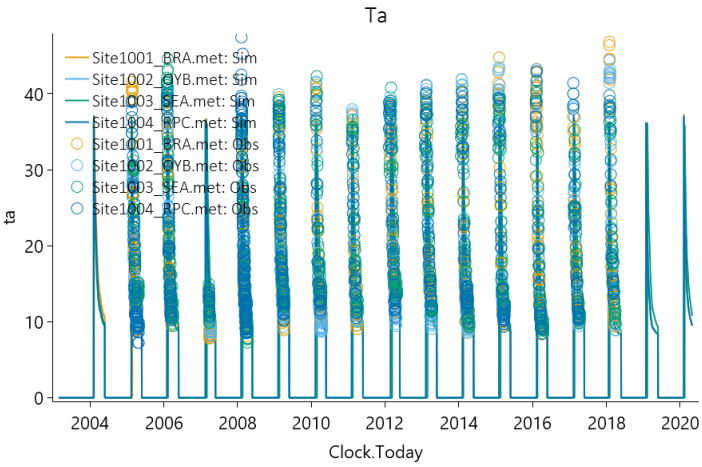


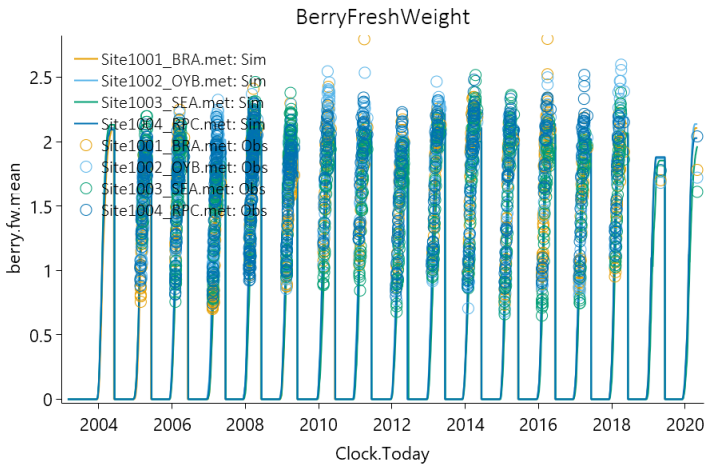


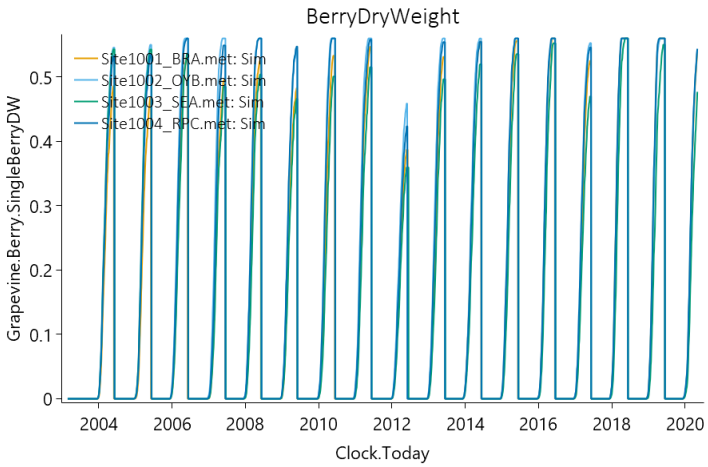


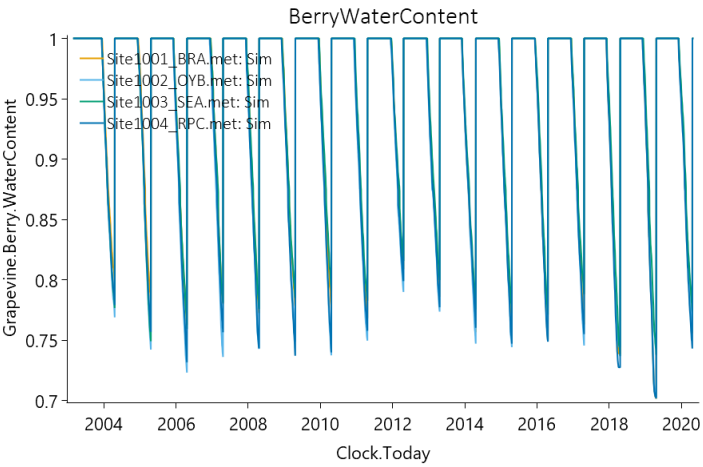
6.1.2.2 YieldQuality

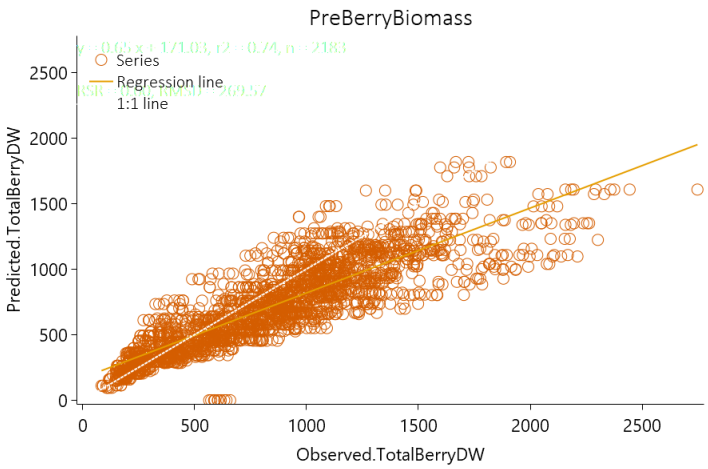


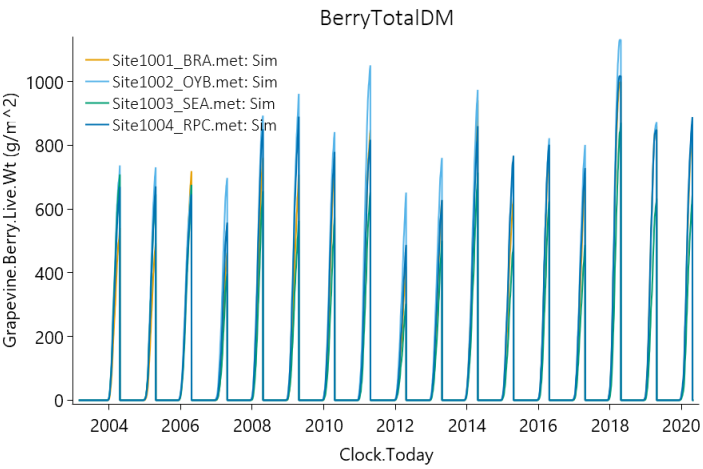


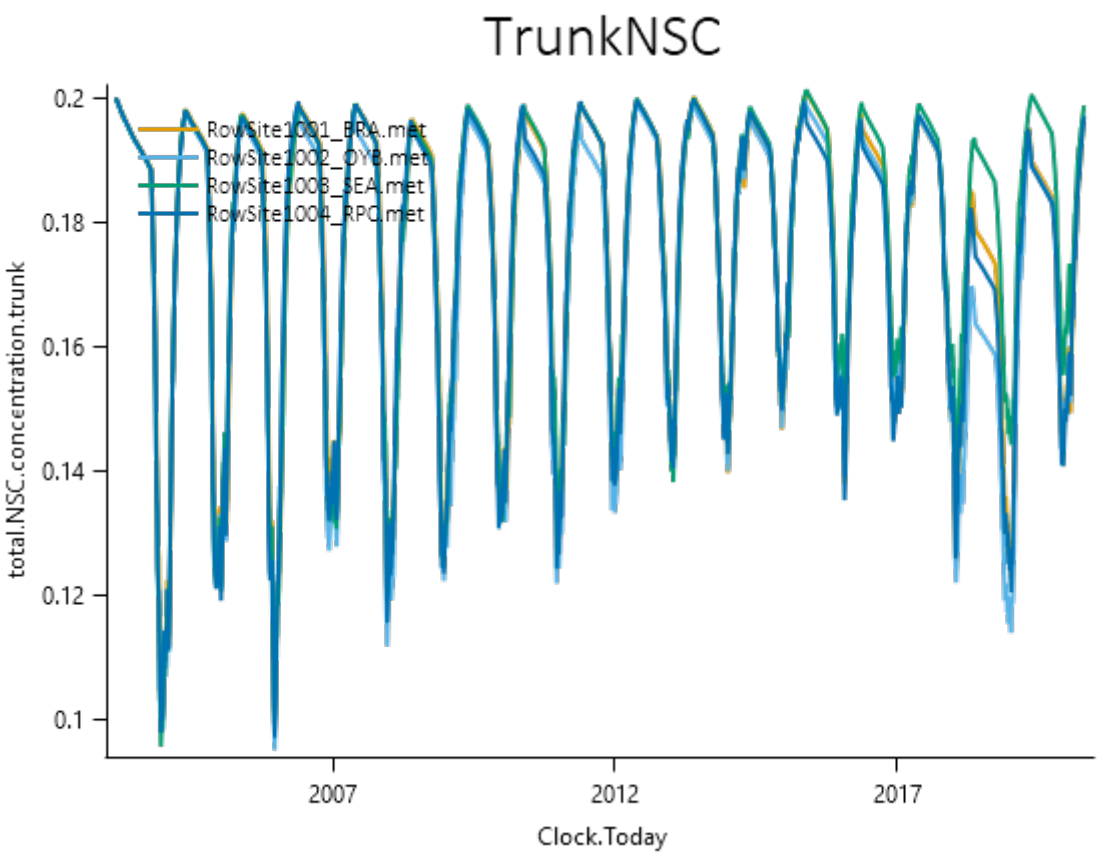






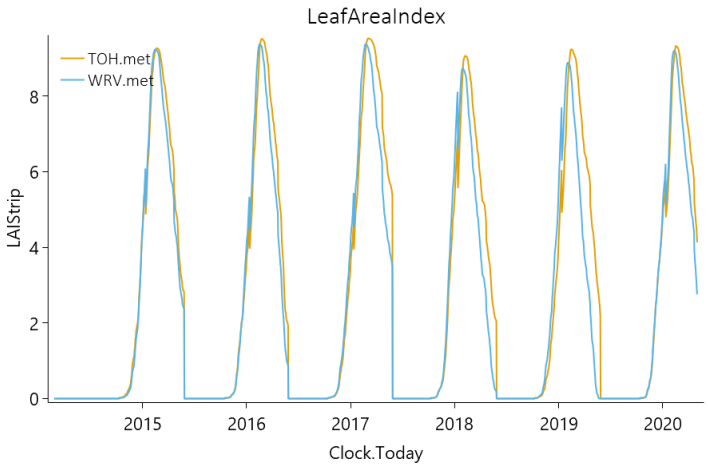


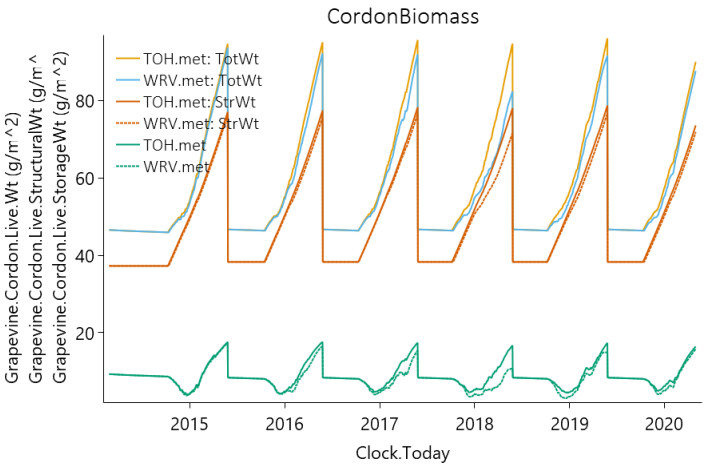


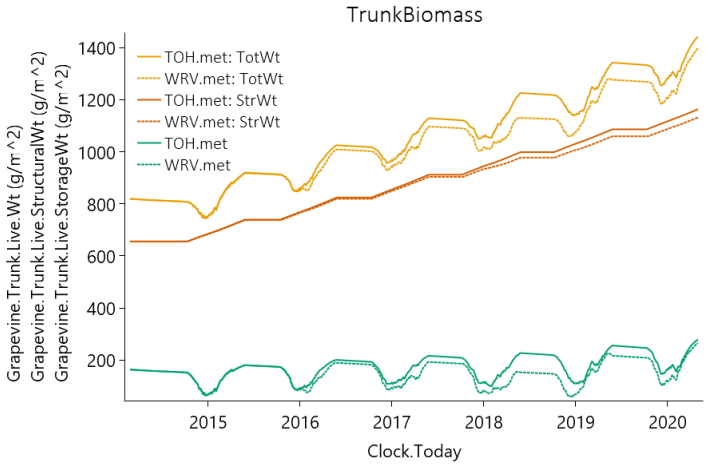

TrunkNSC
RowSite1001_BRA.met
RowSite1002_OYB.met
RowSite1003_SEA.met
RowSite1004_RPC.met
total.NSC.concentration.trunk
0.2
0.18
0.16
0.14
0.12
0.1
2007
2012
2017
Clock.Today

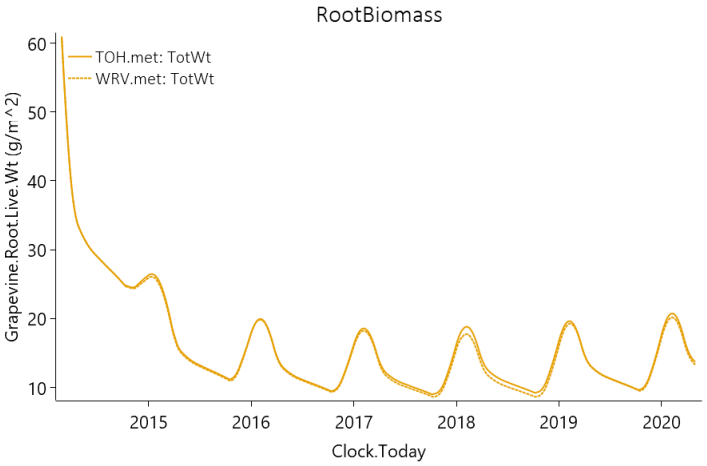
6.1.3 Marlborough_3Cane

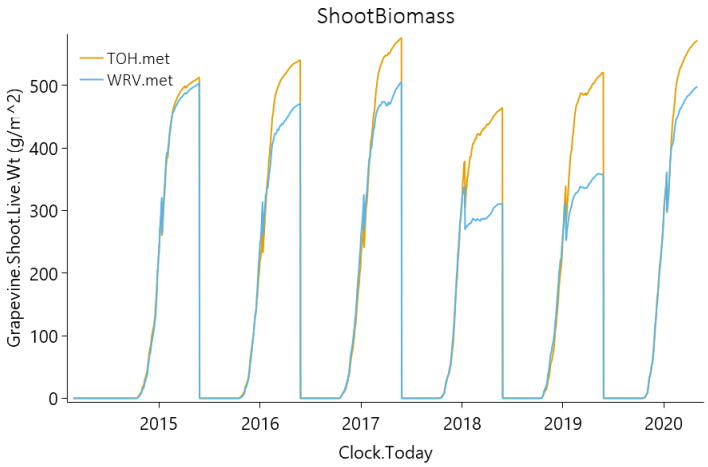
6.1.3.1 Figures

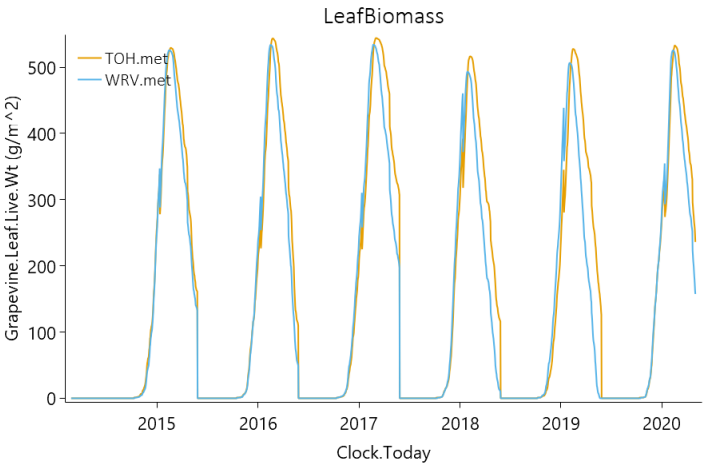




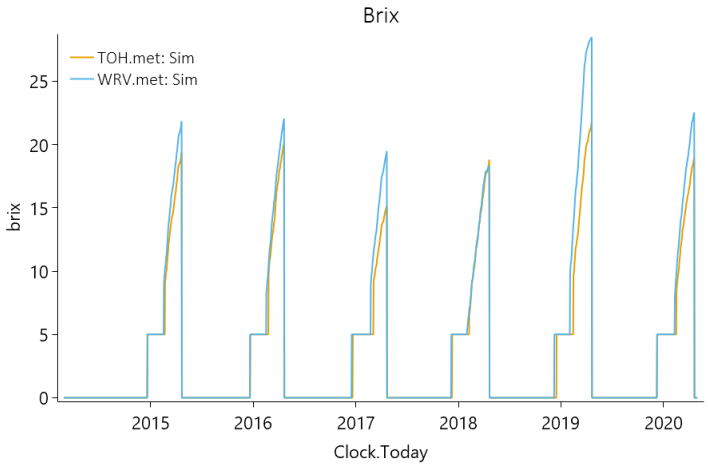


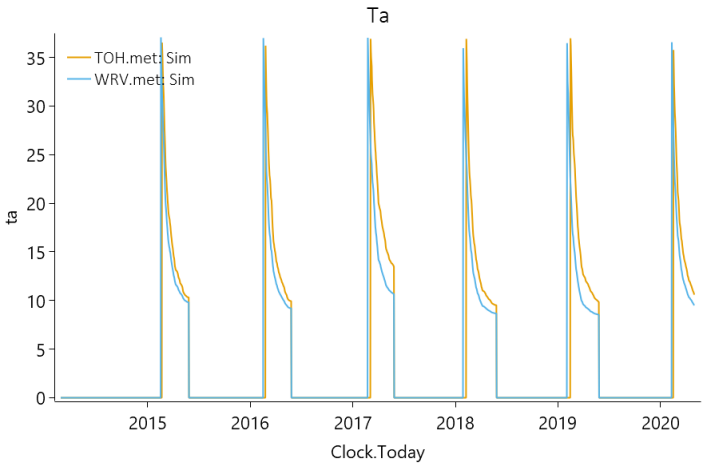


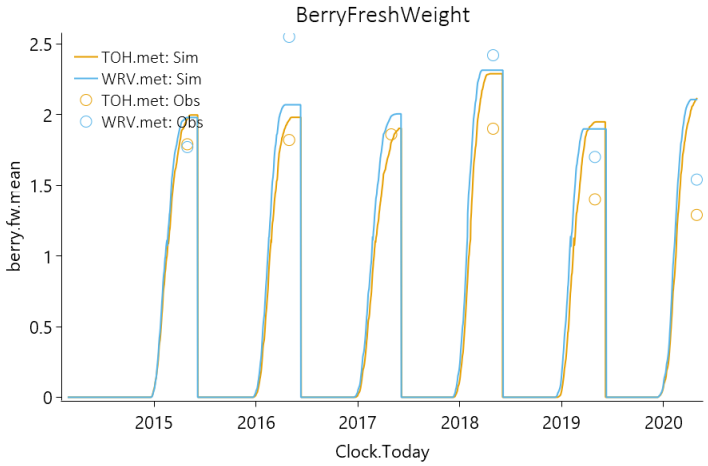


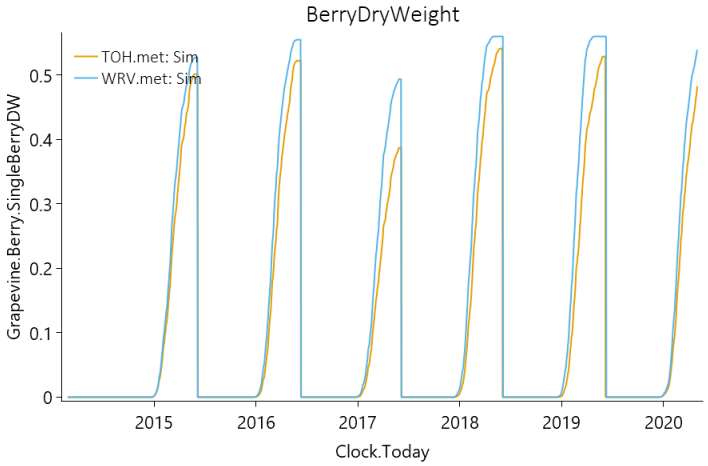


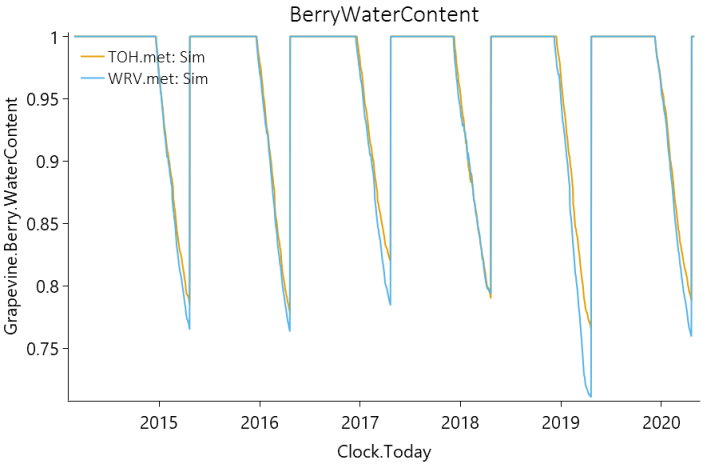
6.1.3.2 YieldQuality





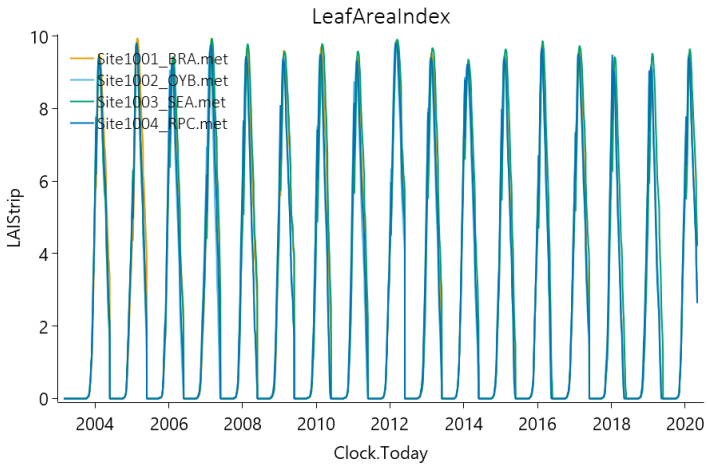


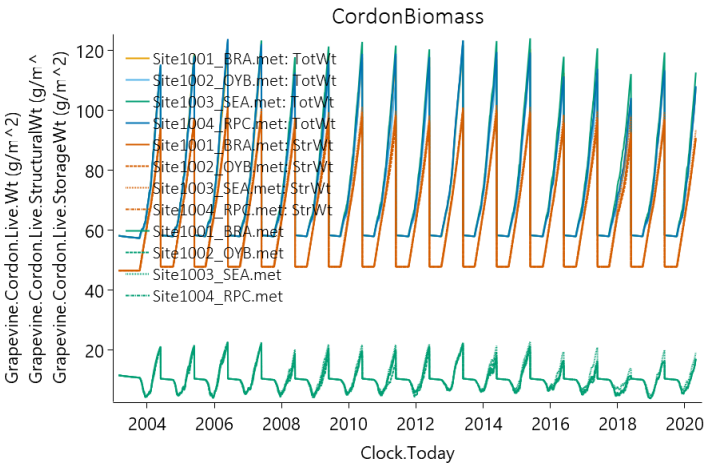


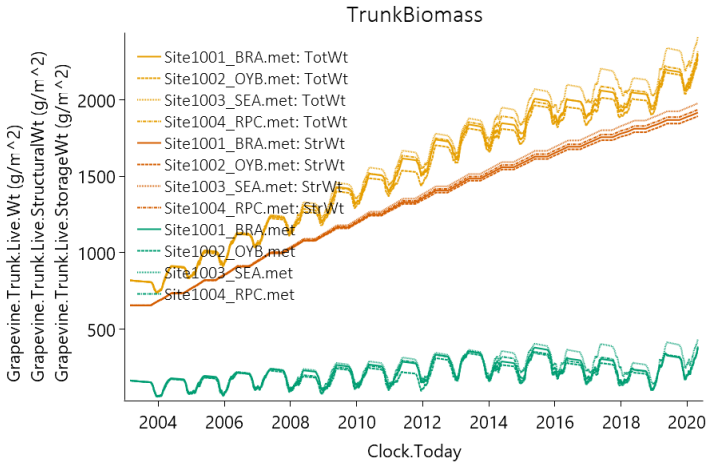


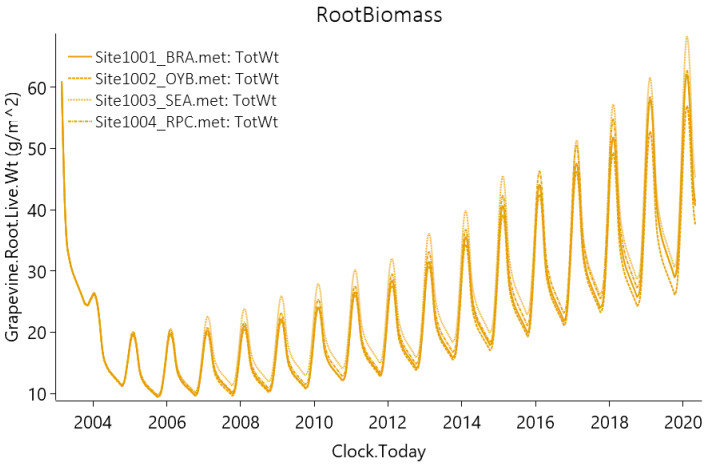
6.1.4 Marlborough_4Cane

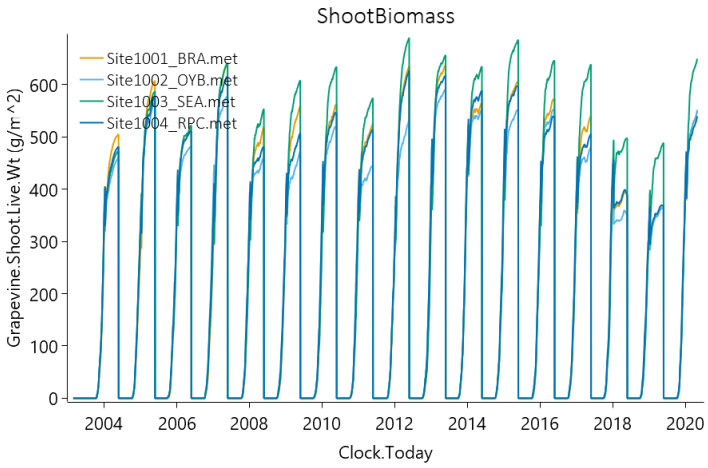
6.1.4.1 Figures

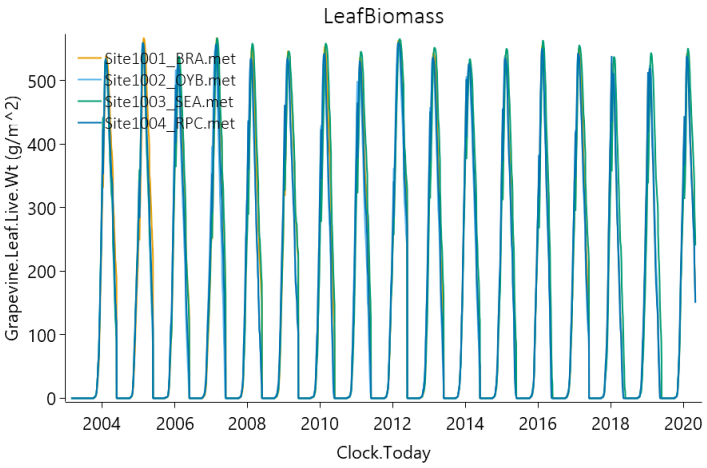




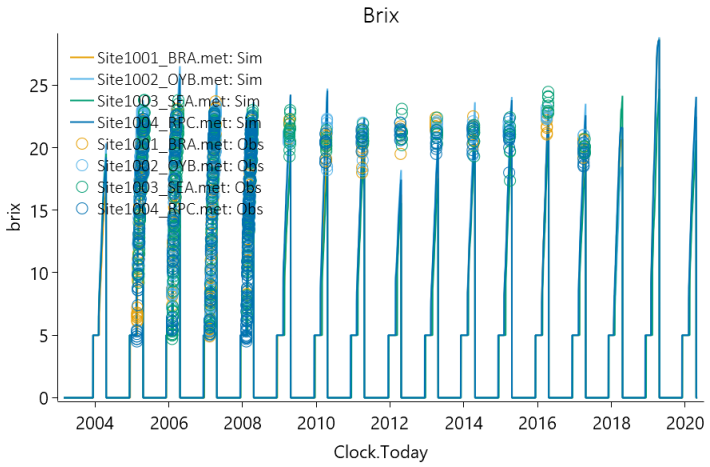


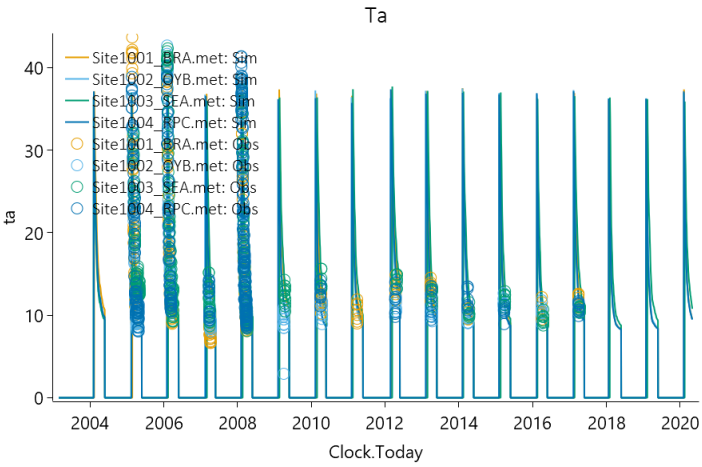


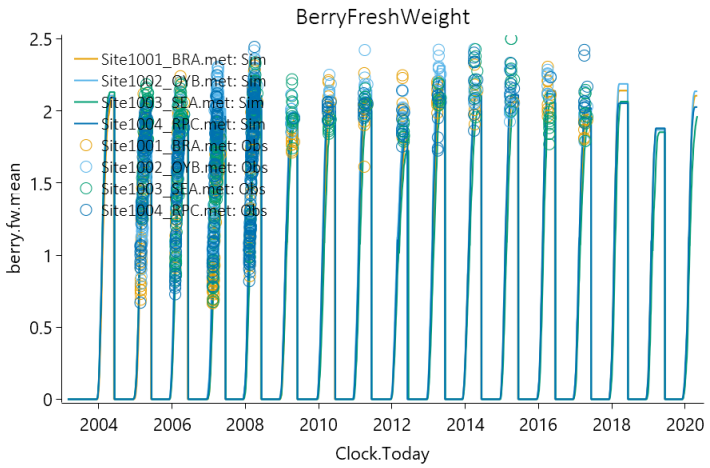


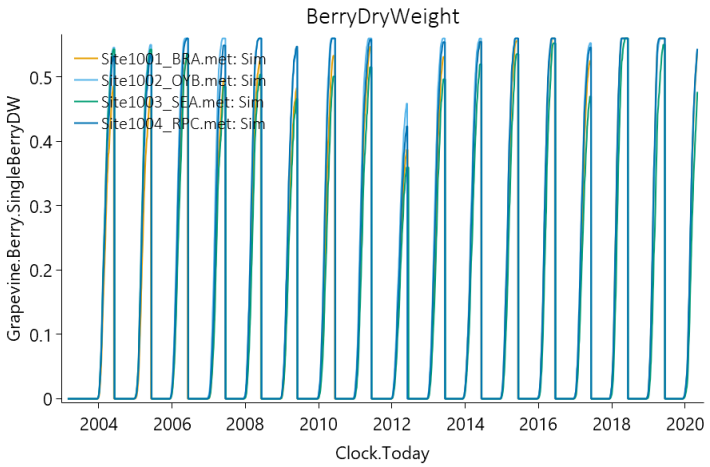


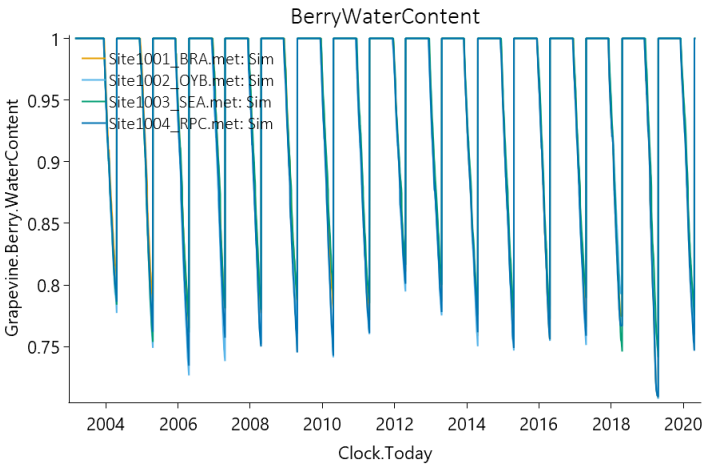
6.1.4.2 YieldQuality

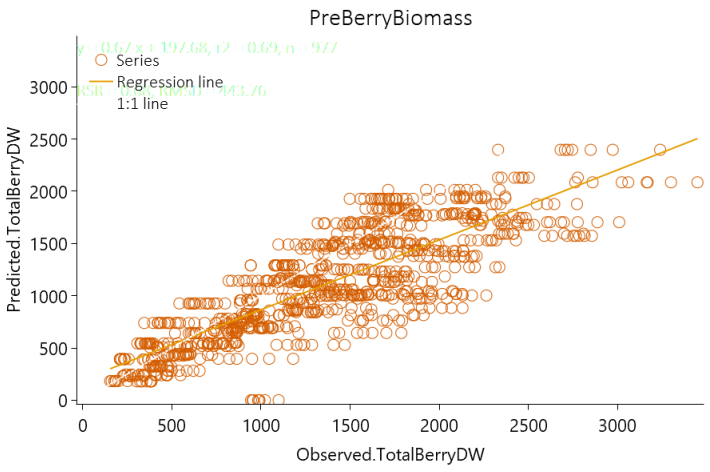




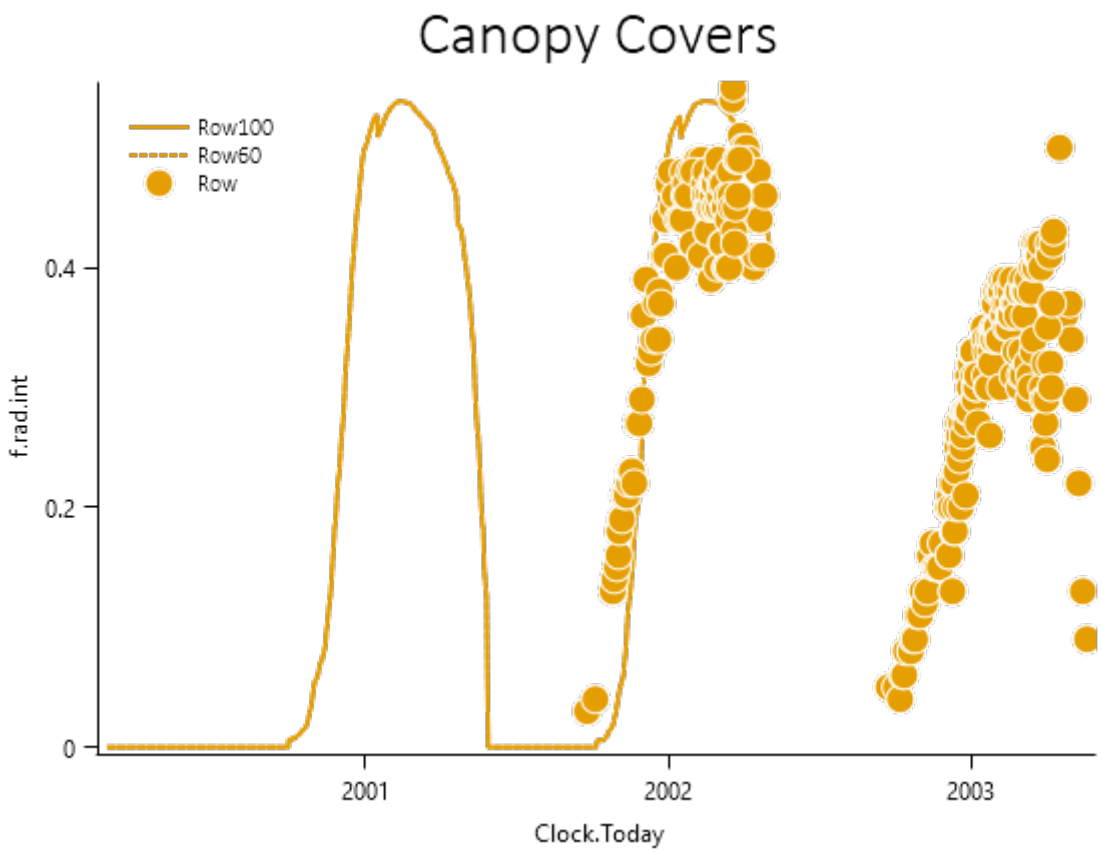


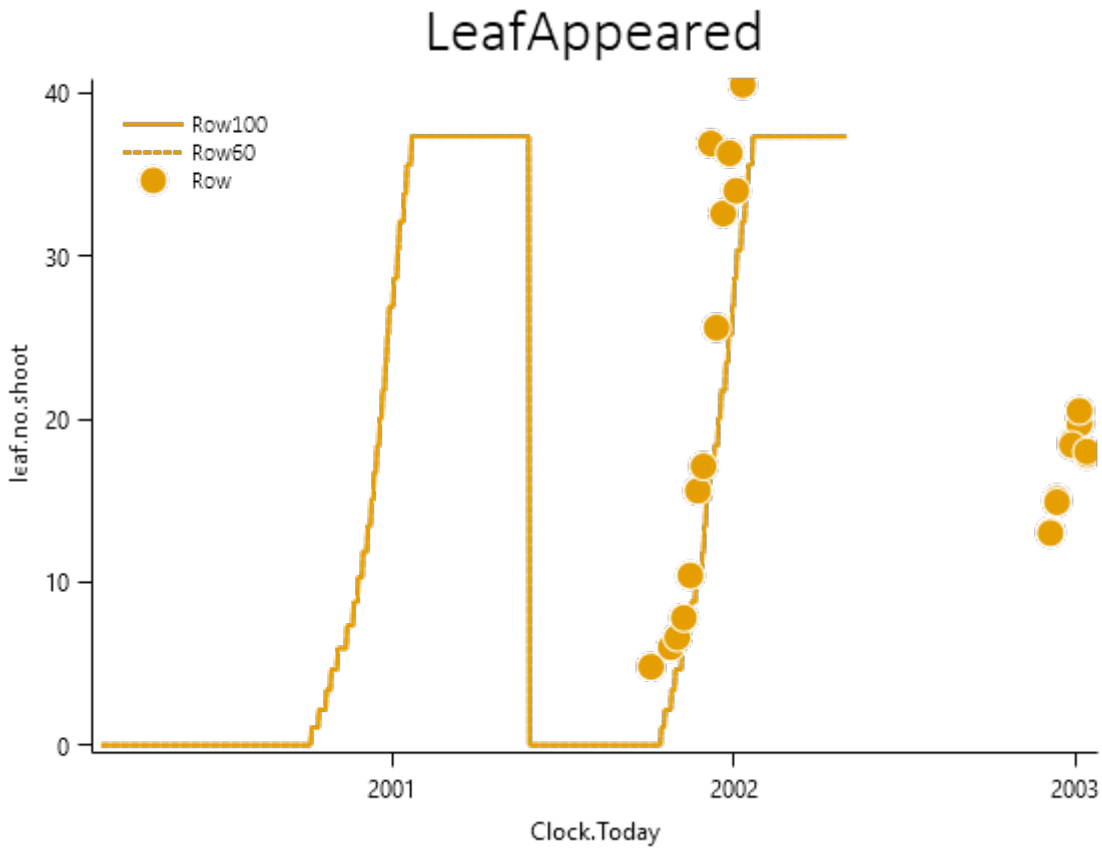


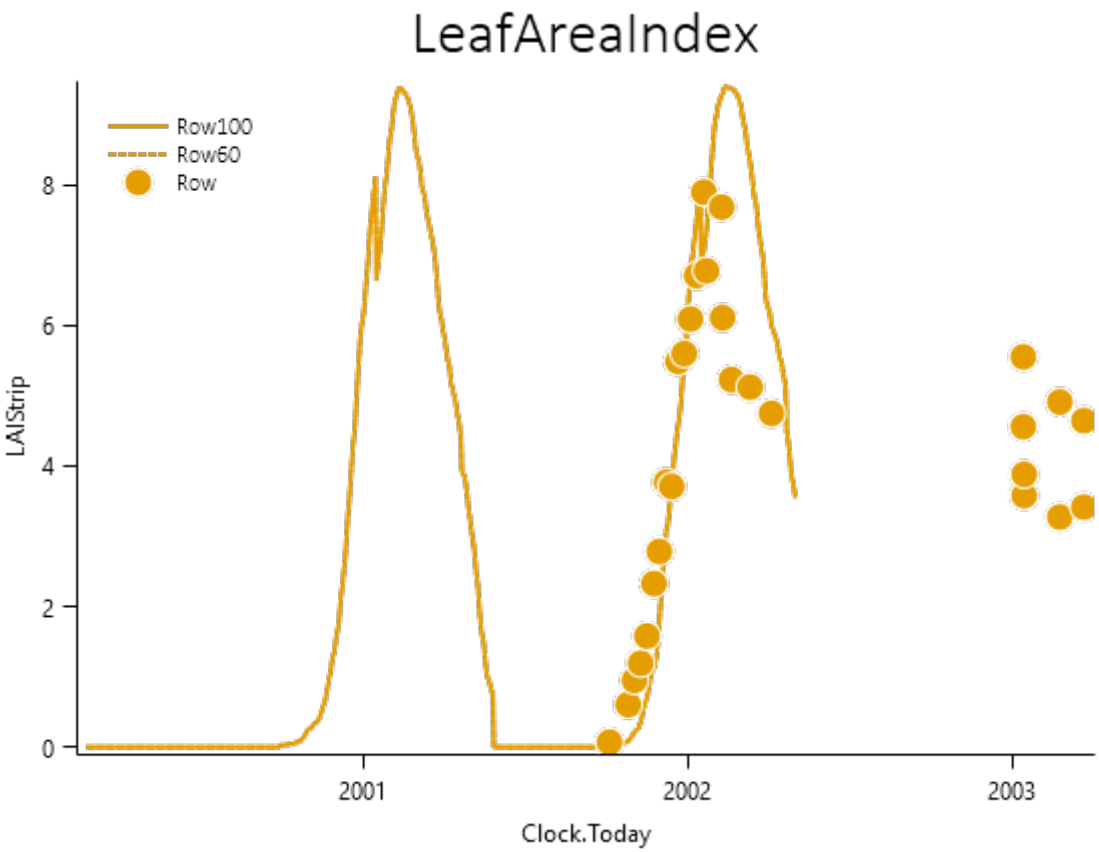


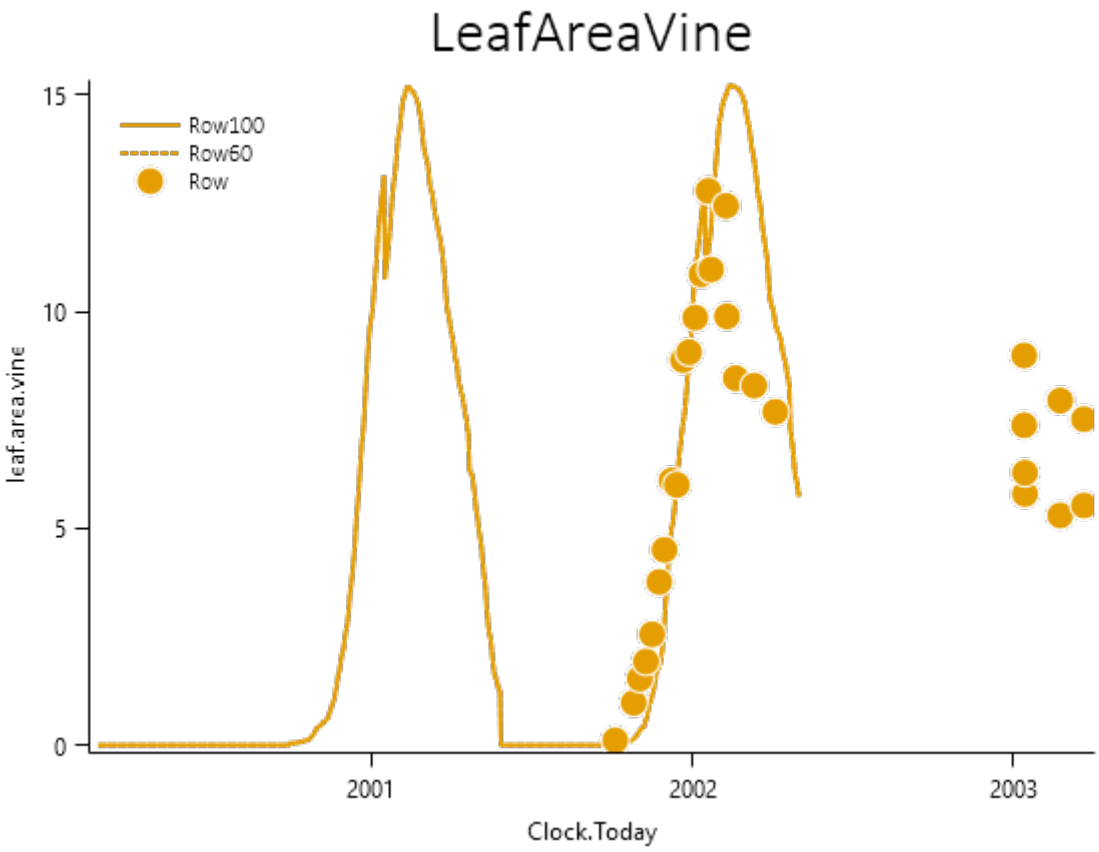


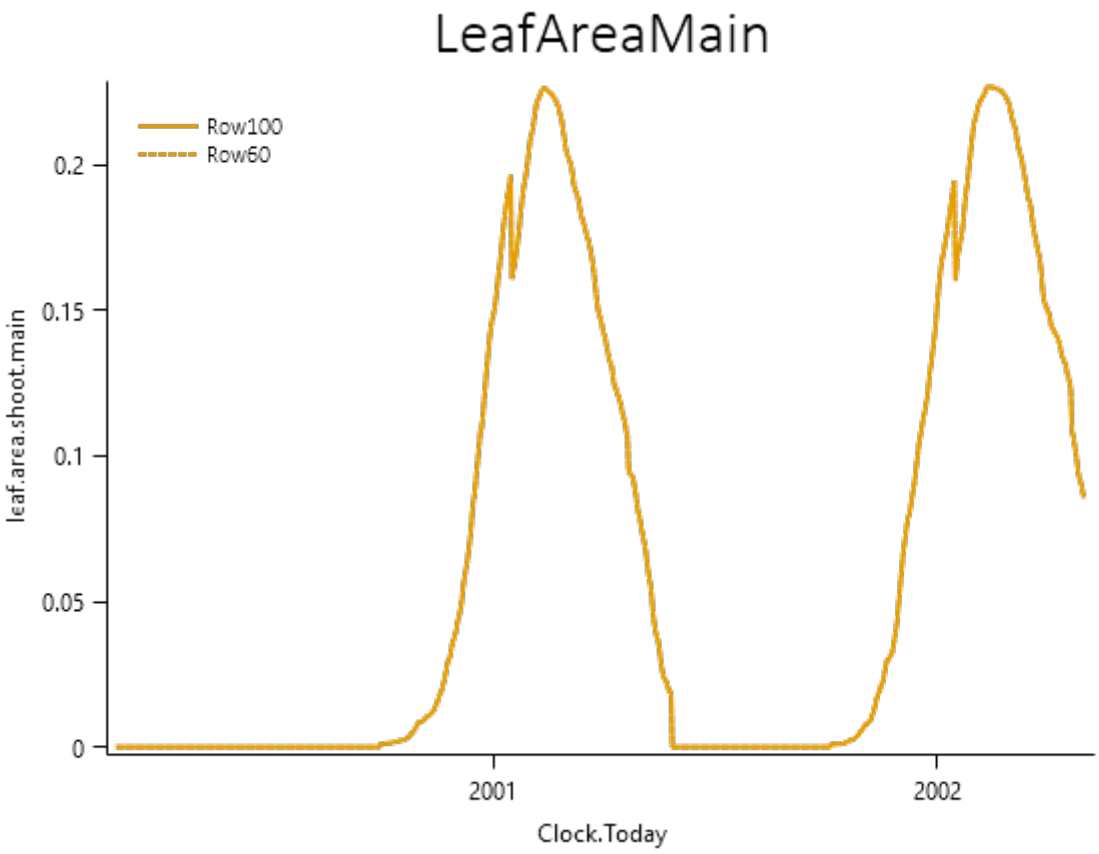
6.1.5 Grape_RPC_1

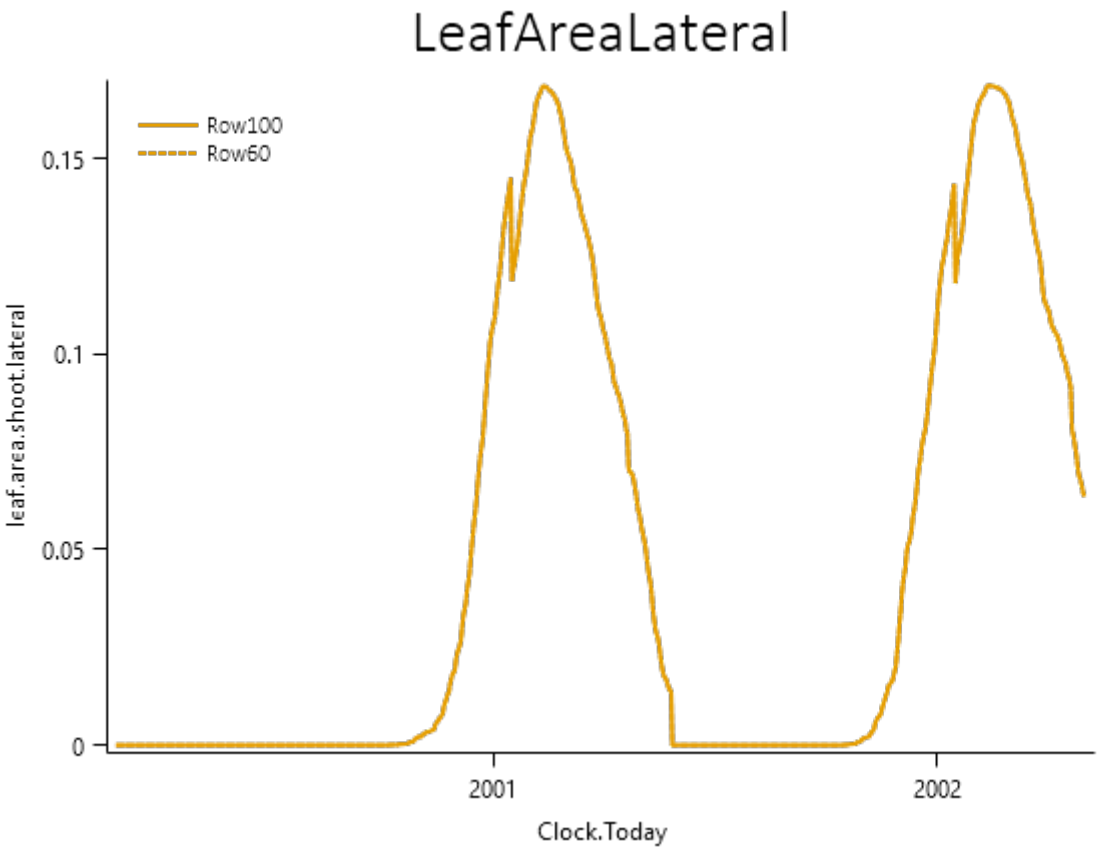


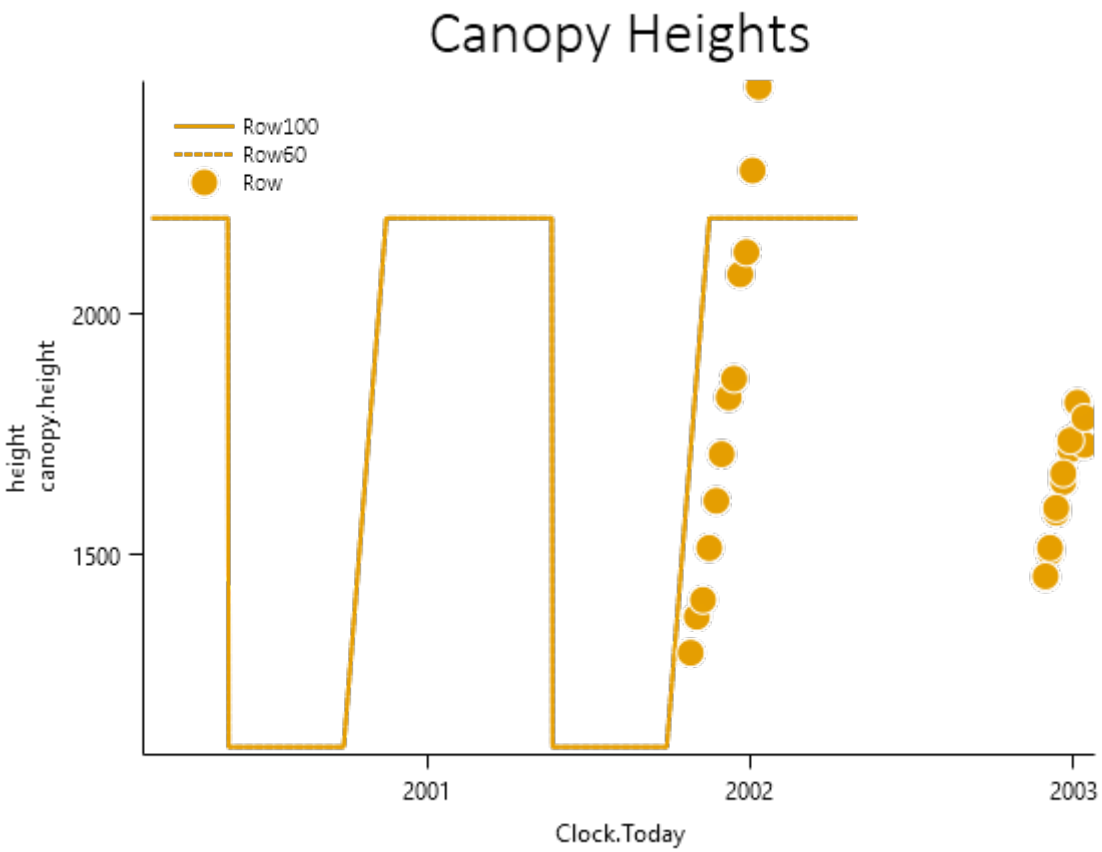


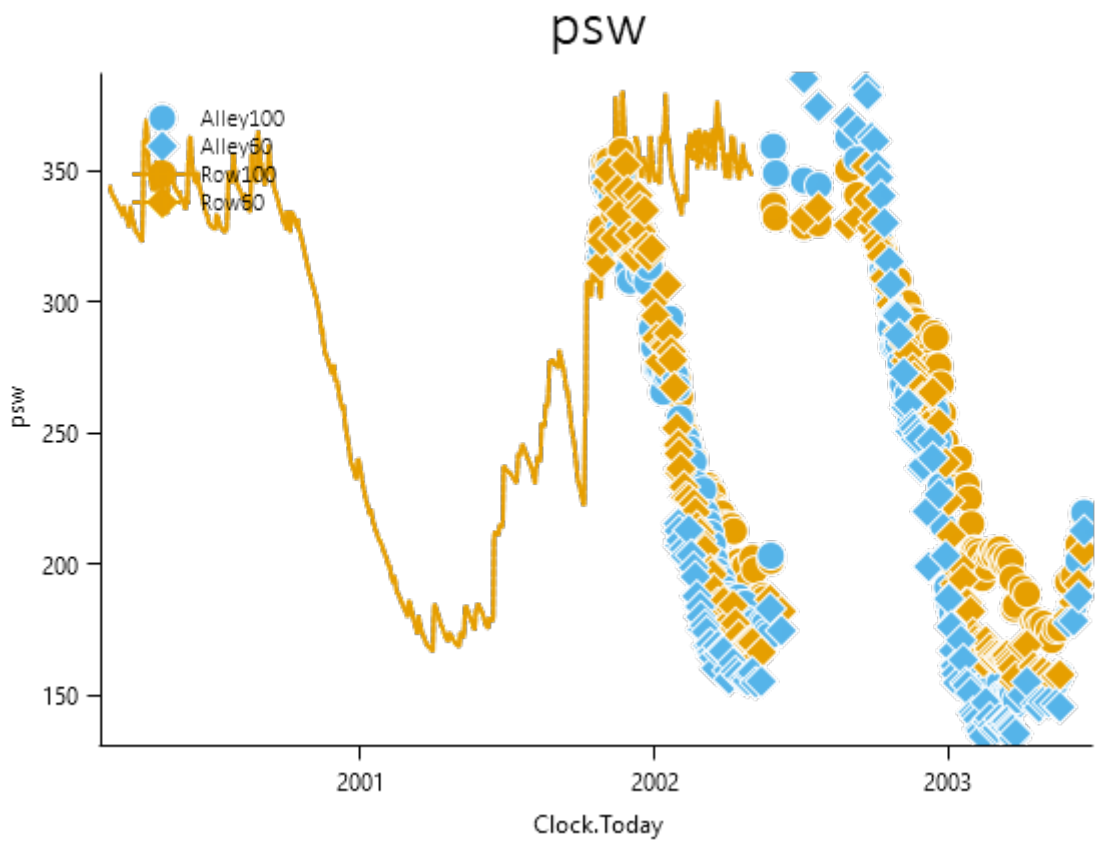

LeafAreaIndex
Row100
Row60
Row
LAIStrip
0
2
4
6
8
2001
2002
2003
Clock.Today

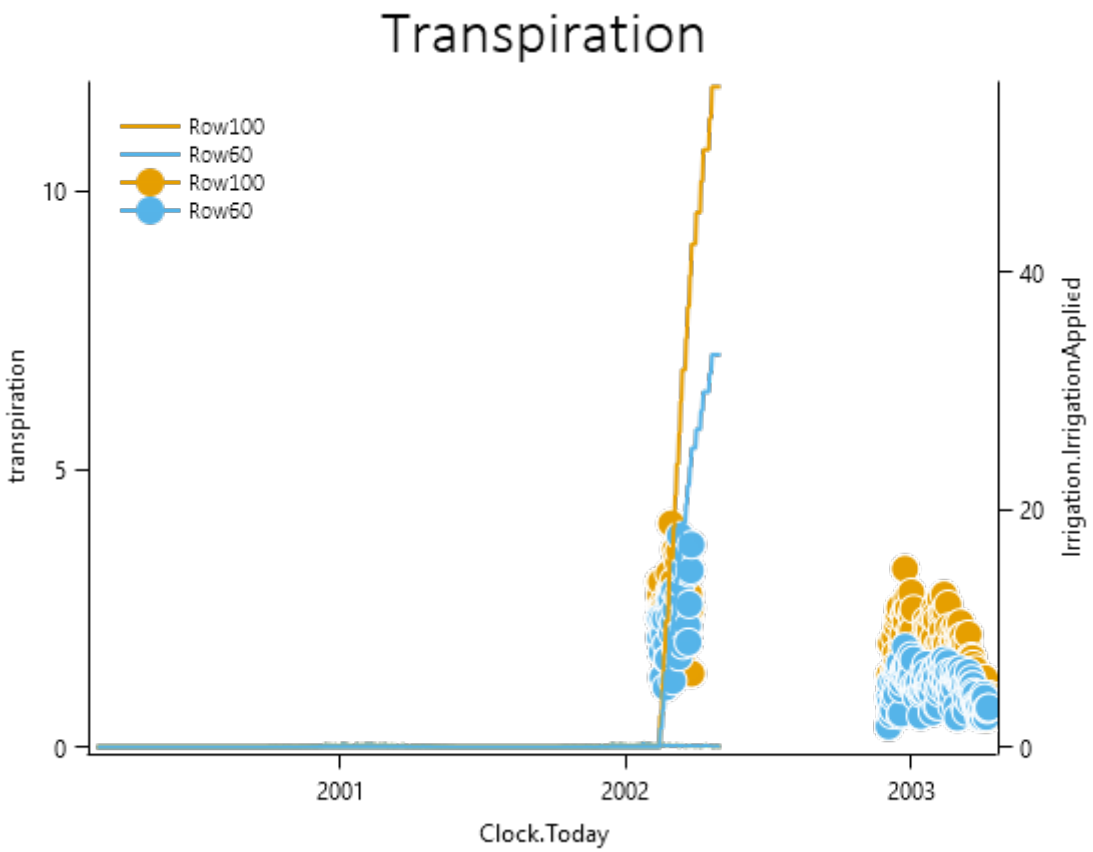

LeafAreaVine
Row100
Row60
Row
leaf.area.vine
0
5
10
15
2001
2002
2003
Clock.Today

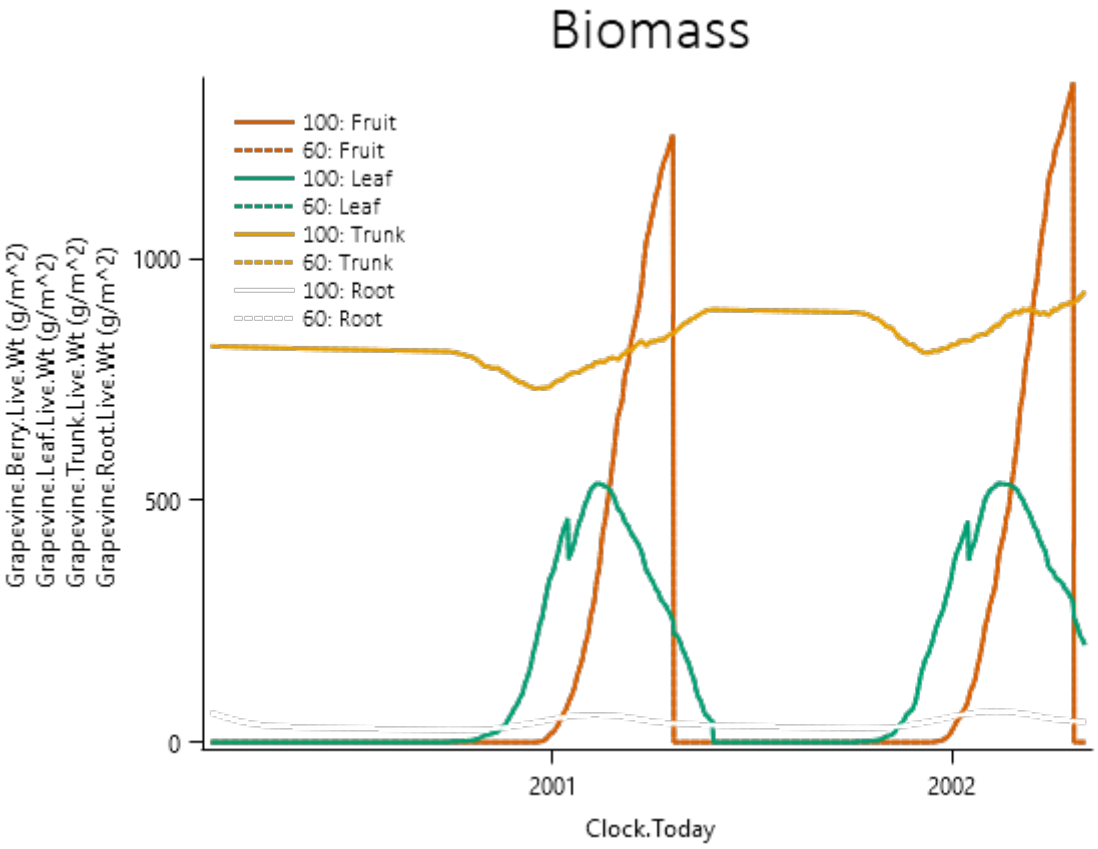

LeafAreaMain
Row100
Row60
0.2
0.15
0.1
0.05
0
leaf.area.shoot.main
2001
2002
Clock.Today

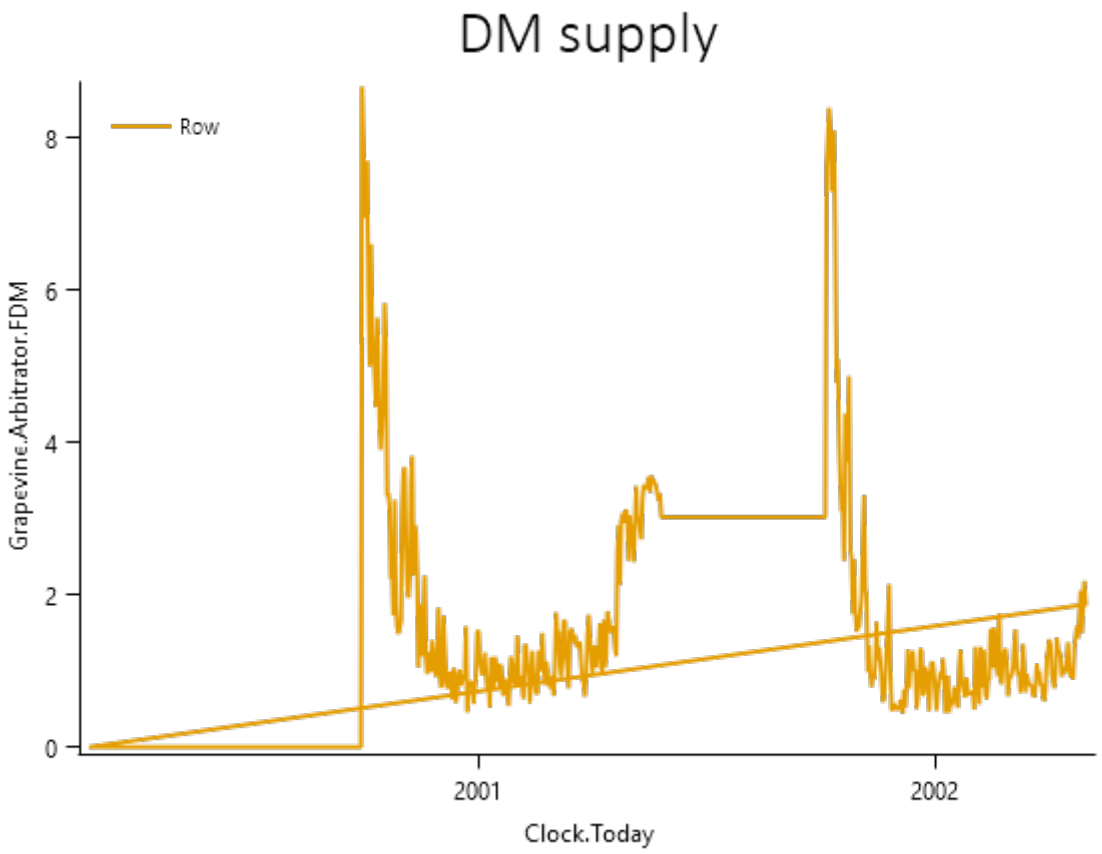

LeafAreaLateral
Row100
Row60
0.15
0.1
0.05
0
leaf.area.shoot.lateral
2001
2002
Clock.Today

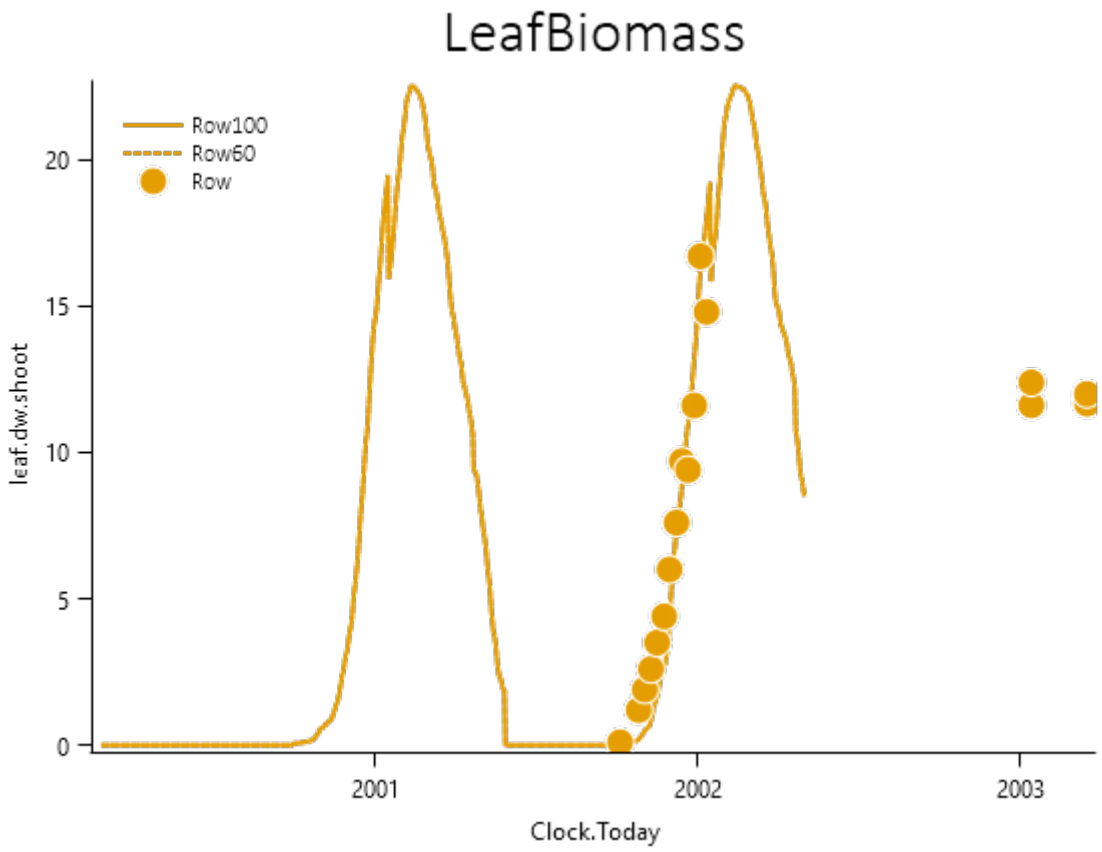

Canopy Heights
Row100
Row60
Row
height
canopy.height
2000
1500
2001
2002
2003
Clock.Today

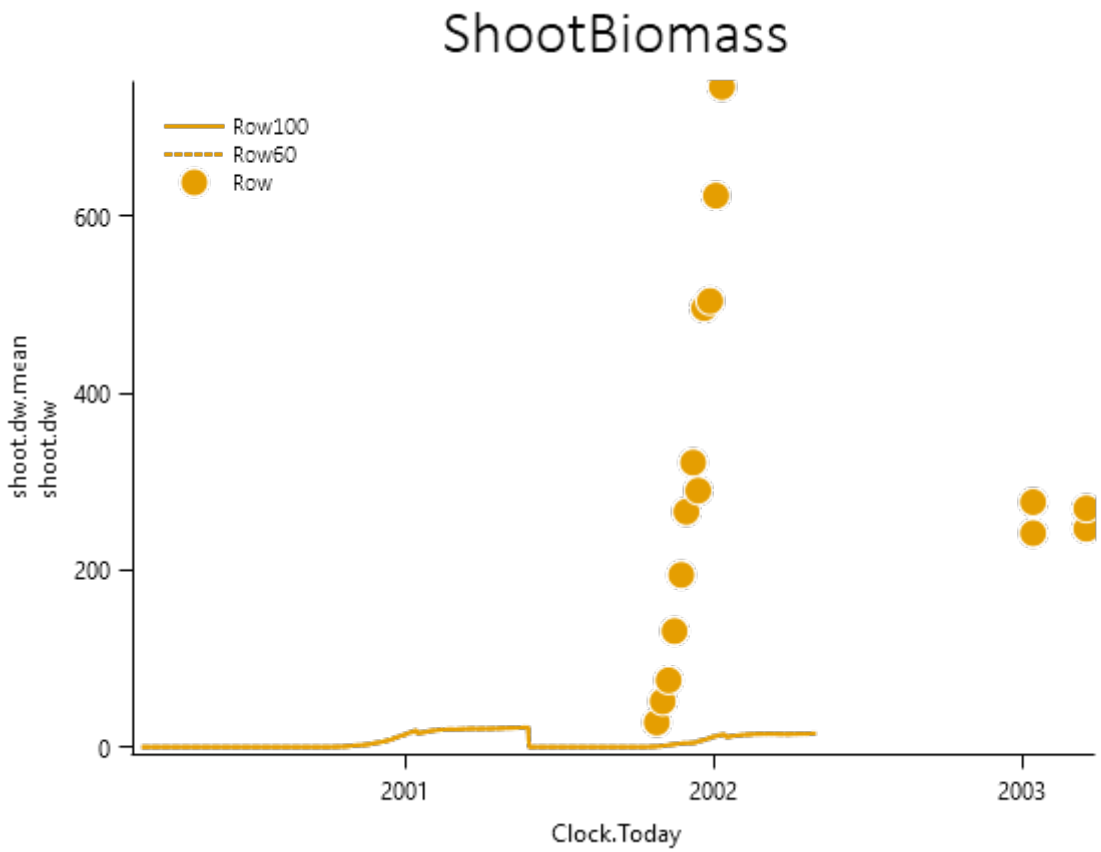

psw
Alley100
Alley60
Row100
Row60
350
300
250
200
150
psw
2001
2002
2003
Clock.Today

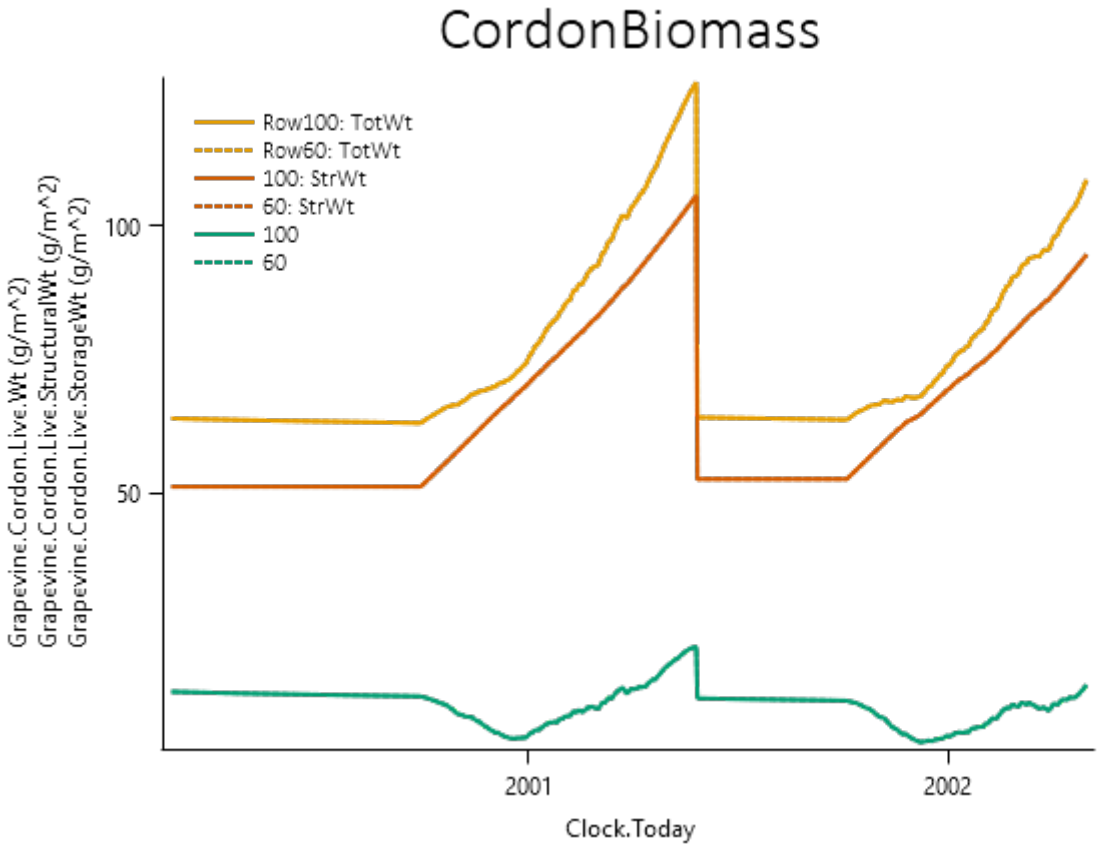

Transpiration
Row100
Row60
Row100
Row60
transpiration
Irrigation.IrrigationApplied
10
5
0
40
20
0
2001
2002
2003
Clock.Today

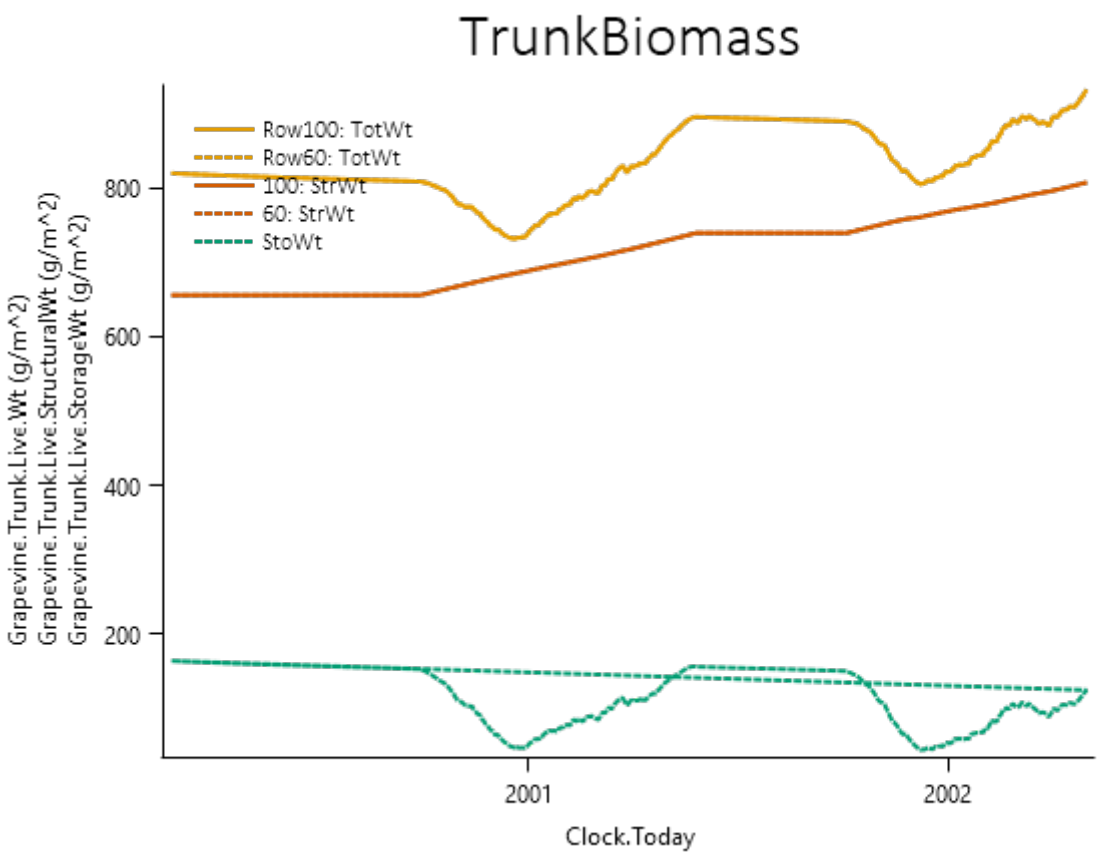

Biomass
100: Fruit
60: Fruit
100: Leaf
60: Leaf
100: Trunk
60: Trunk
100: Root
60: Root
Grapevine.Berry.Live.Wt (g/m^2)
Grapevine.Leaf.Live.Wt (g/m^2)
Grapevine.Trunk.Live.Wt (g/m^2)
Grapevine.Root.Live.Wt (g/m^2)
1000
500
0
2001
2002
Clock.Today

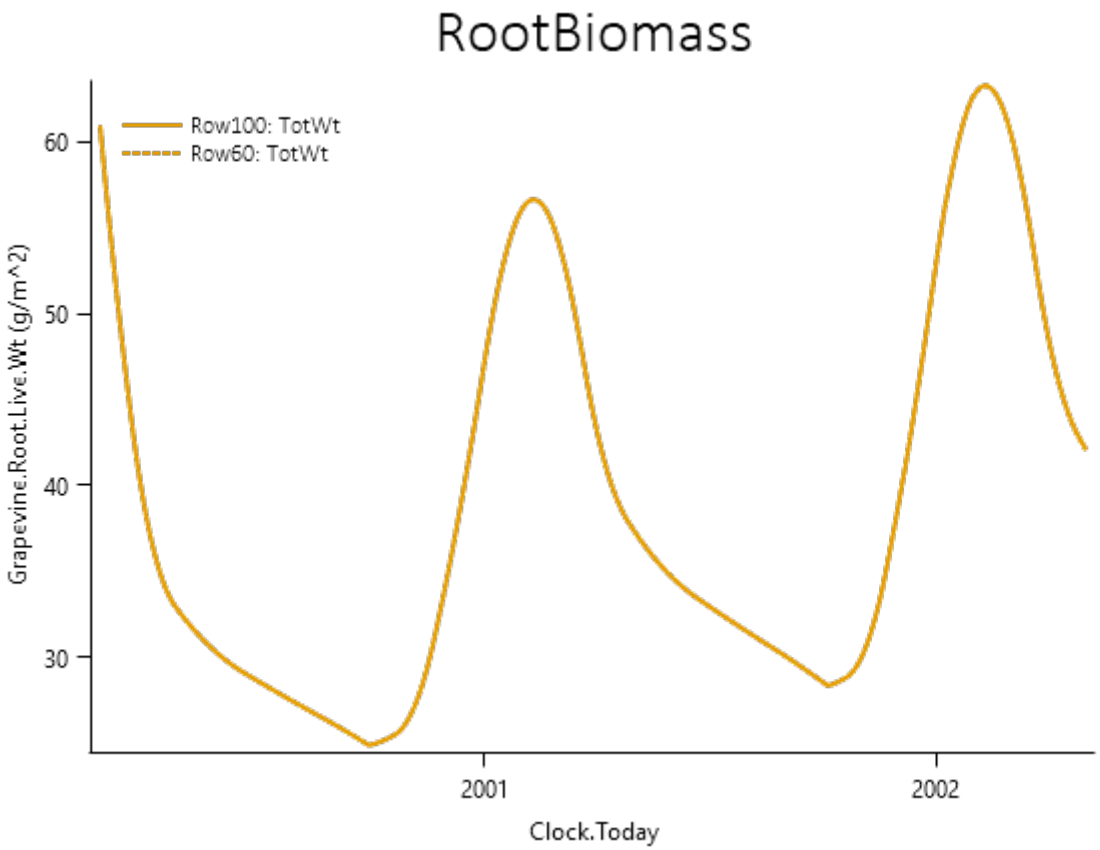

DM supply
Row
Grapevine.Arbitrator.FDM
0
2
4
6
8
2001
2002
Clock.Today

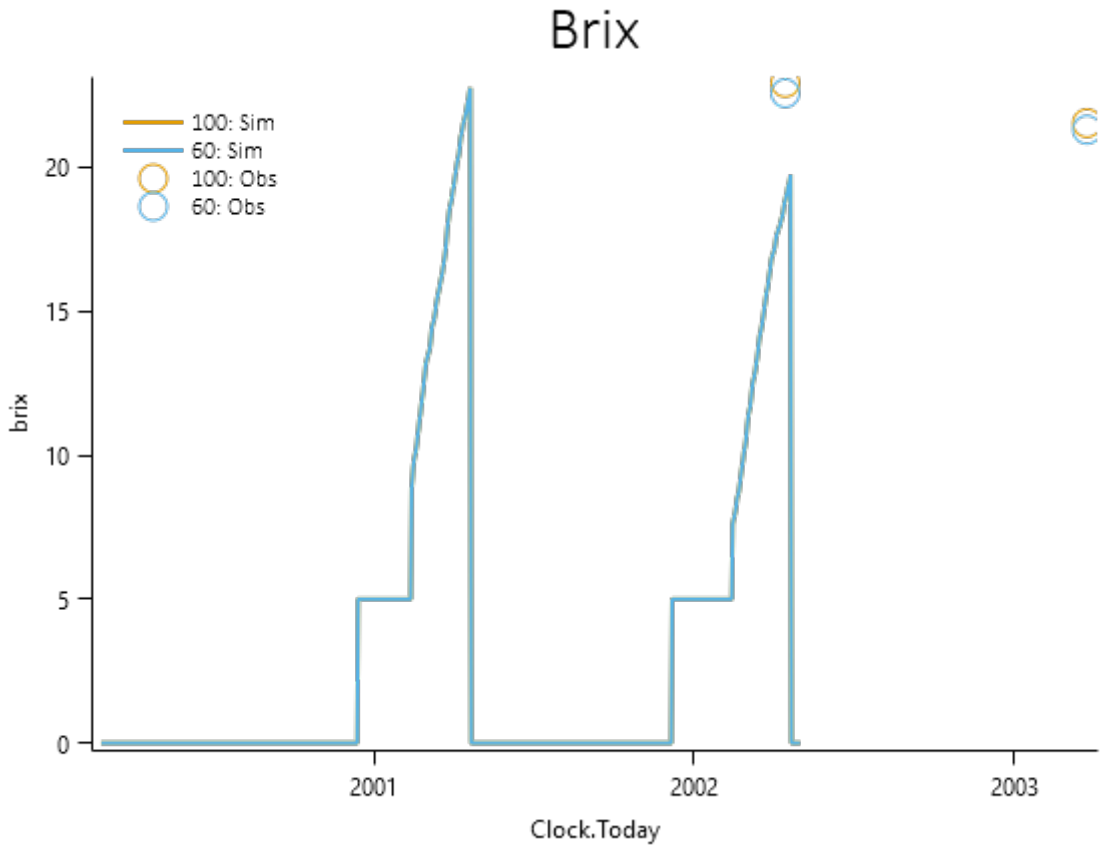

LeafBiomass
Row100
Row60
Row
leaf.dw.shoot
0
5
10
15
20
2001
2002
2003
Clock.Today

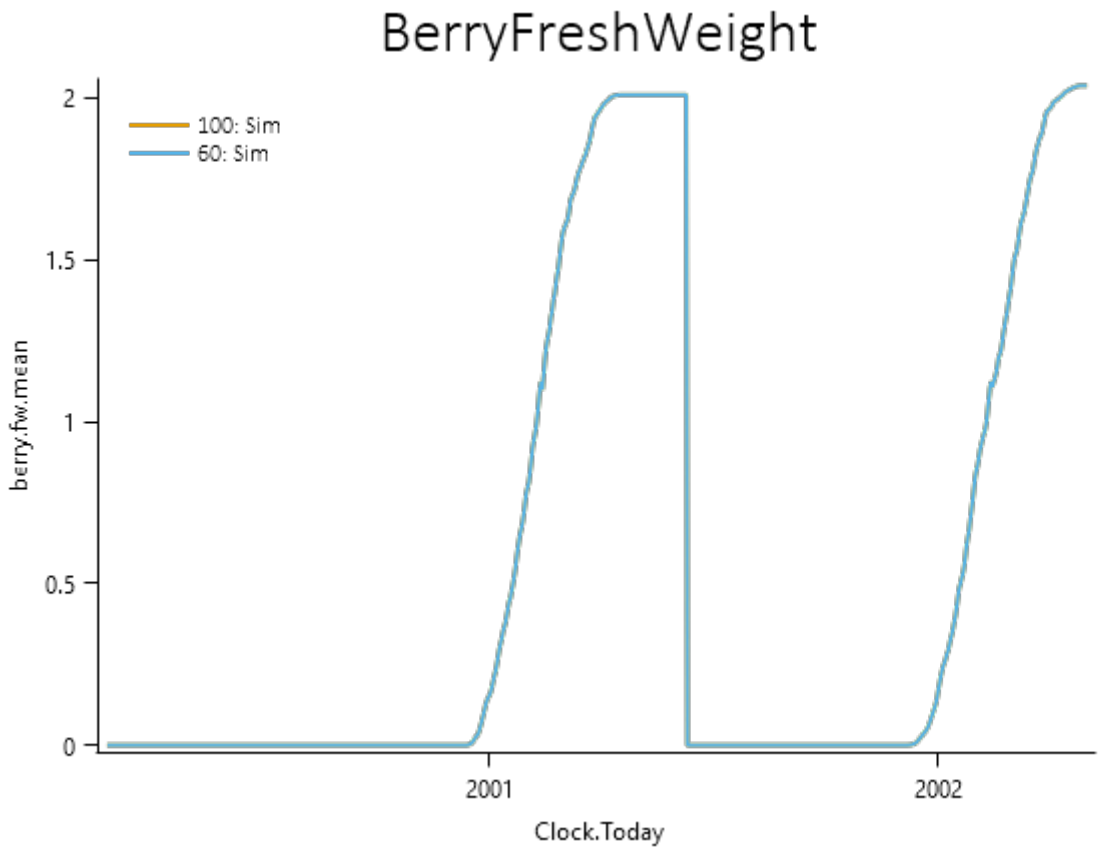

ShootBiomass
Row100
Row60
Row
600
400
200
0
shoot.dw.mean
shoot.dw
2001
2002
2003
Clock.Today

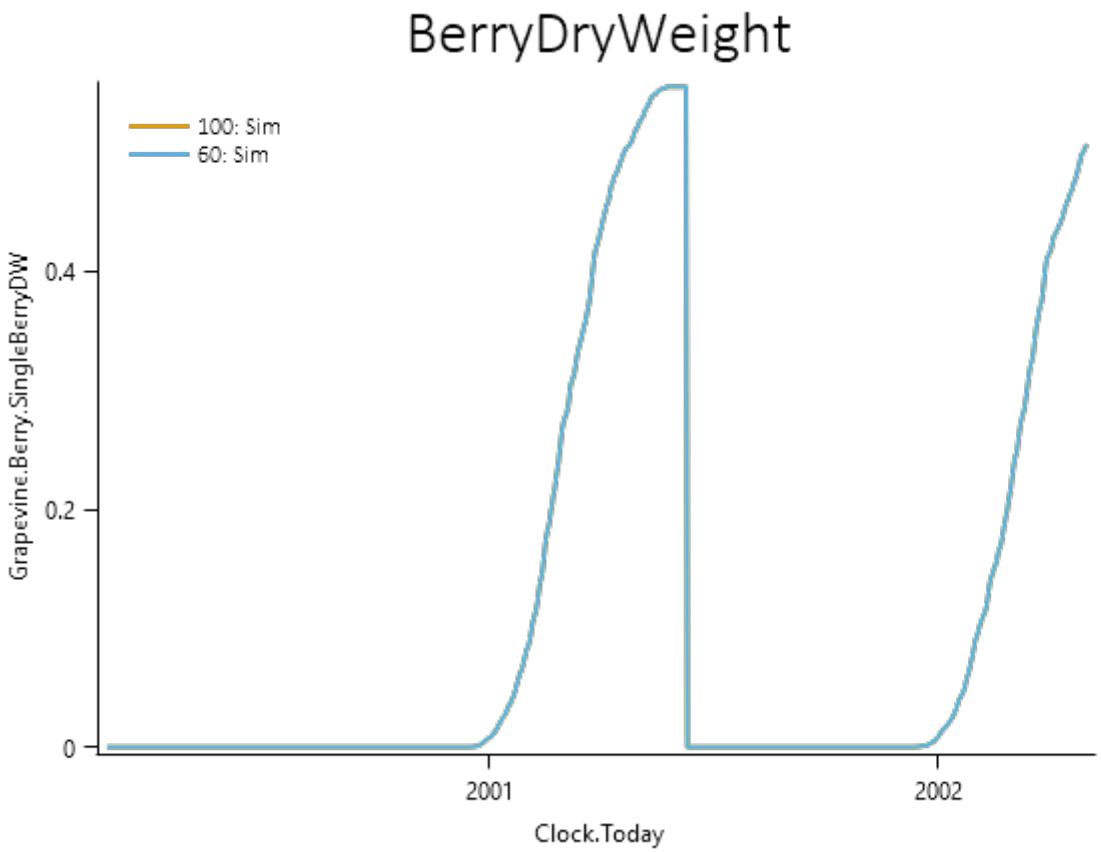

CordonBiomass
Row100: TotWt
Row60: TotWt
100: StrWt
60: StrWt
100
60
100
50
Grapevine.Cordon.Live.Wt (g/m^2)
Grapevine.Cordon.Live.StructuralWt (g/m^2)
Grapevine.Cordon.Live.StorageWt (g/m^2)
2001
2002
Clock.Today

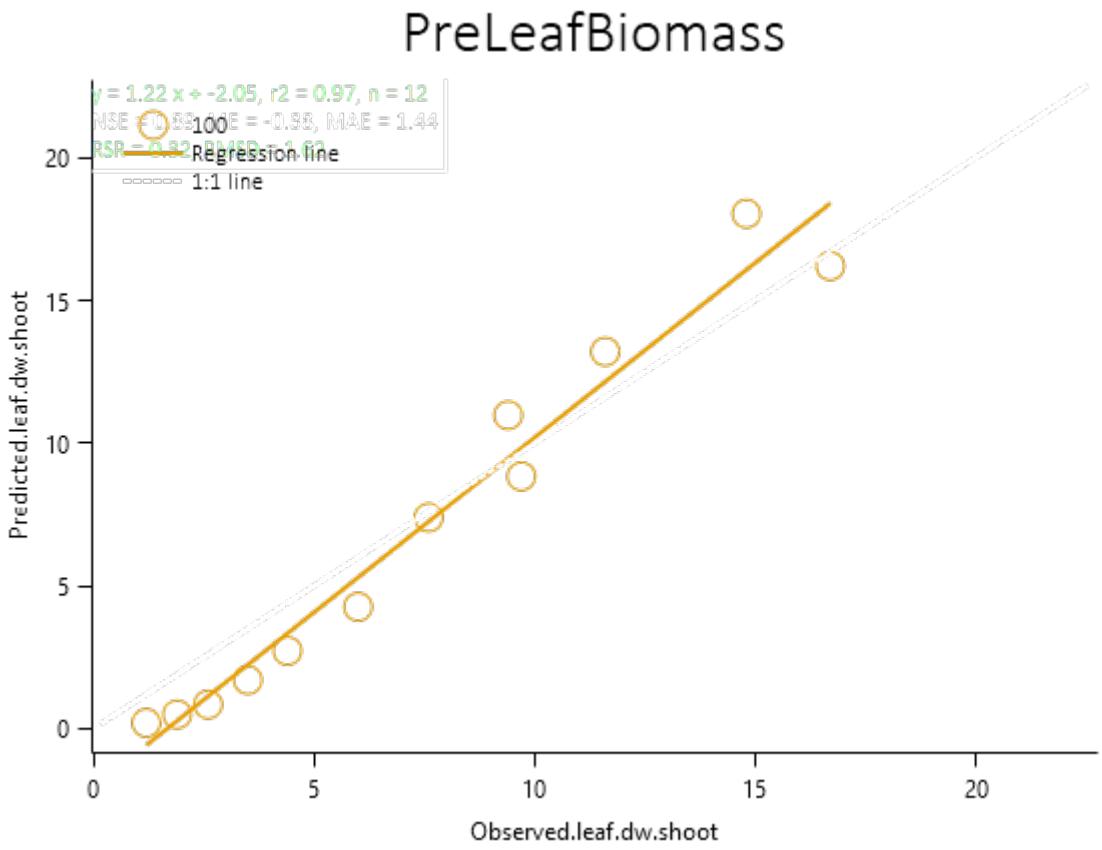

TrunkBiomass
Row100: TotWt
Row60: TotWt
100: StrWt
60: StrWt
StoWt
Grapevine.Trunk.Live.Wt (g/m^2)
Grapevine.Trunk.Live.StructuralWt (g/m^2)
Grapevine.Trunk.Live.StorageWt (g/m^2)
800
600
400
200
2001
2002
Clock.Today

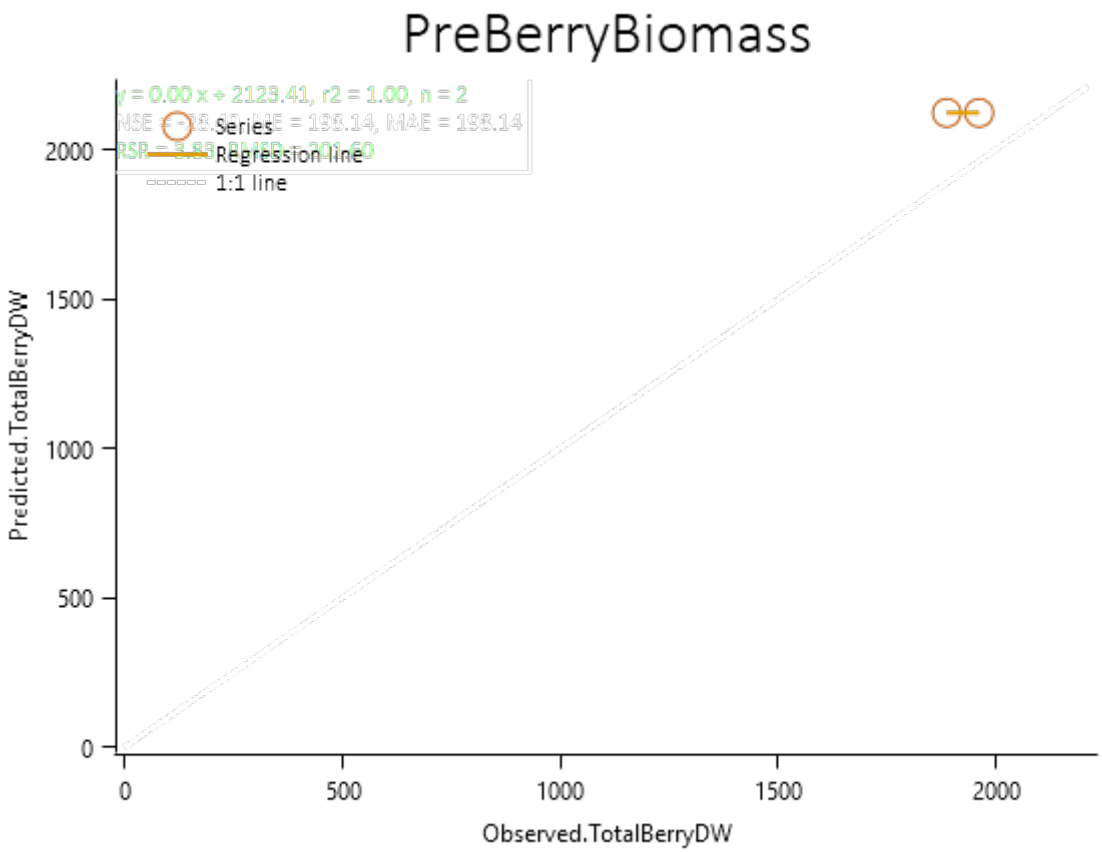

RootBiomass
Row100: TotWt
Row60: TotWt
Grapevine.Root.Live.Wt (g/m^2)
60
50
40
30
2001
2002
Clock.Today

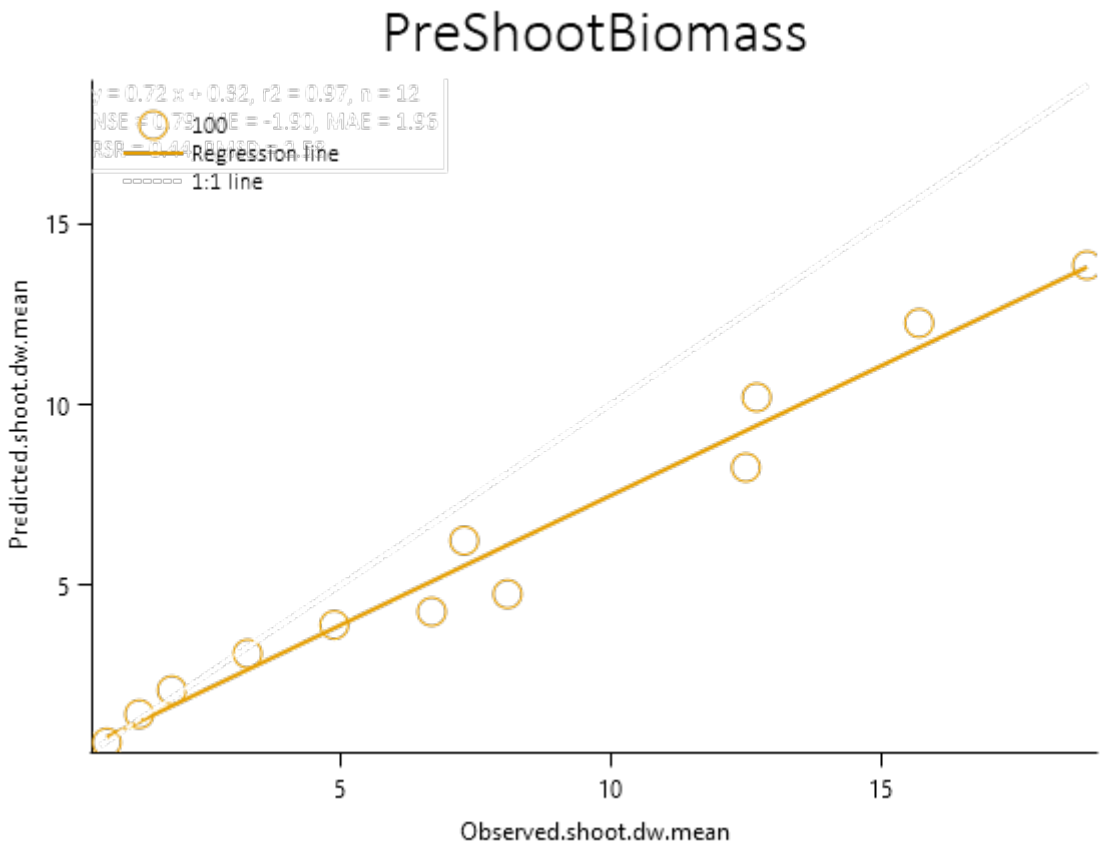

Brix
100: Sim
60: Sim
100: Obs
60: Obs
brix
0
5
10
15
20
2001
2002
2003
Clock.Today


BerryFreshWeight
100: Sim
60: Sim
berry.fw.mean
0
0.5
1
1.5
2
2001
2002
Clock.Today

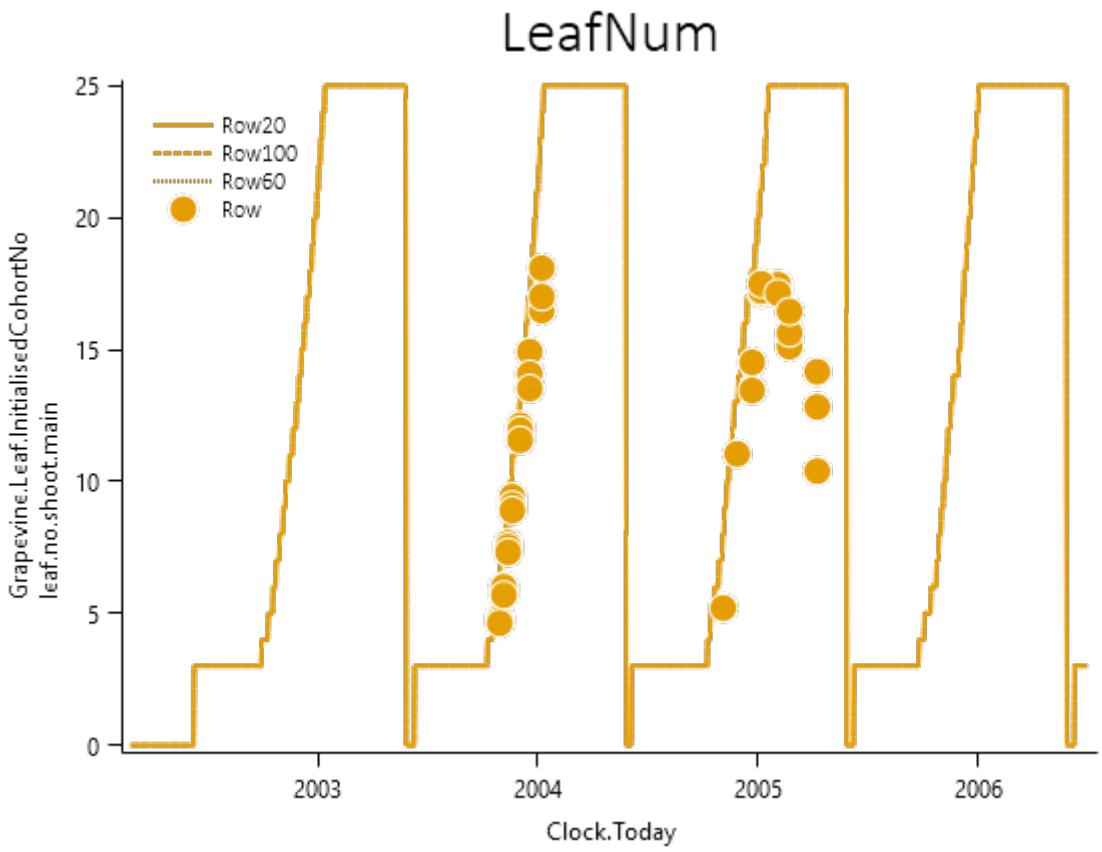

BerryDryWeight
100: Sim
60: Sim
Grapevine.Berry.SingleBerryDW
0.4
0.2
0
2001
2002
Clock.Today

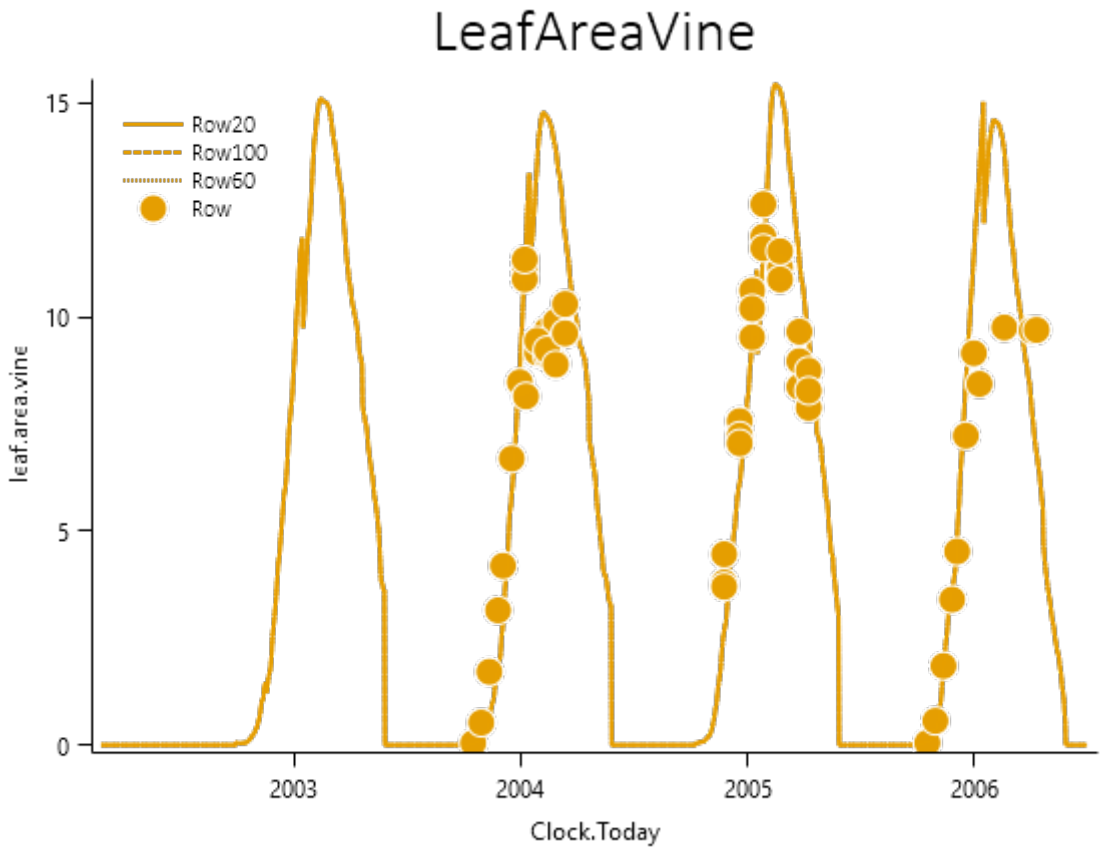

PreLeafBiomass
y = 1.22 x + -2.05, r2 = 0.97, n = 12
100
Regression line
1:1 line
Predicted.leaf.dw.shoot
20
15
10
5
0
0
5
10
15
20
Observed.leaf.dw.shoot

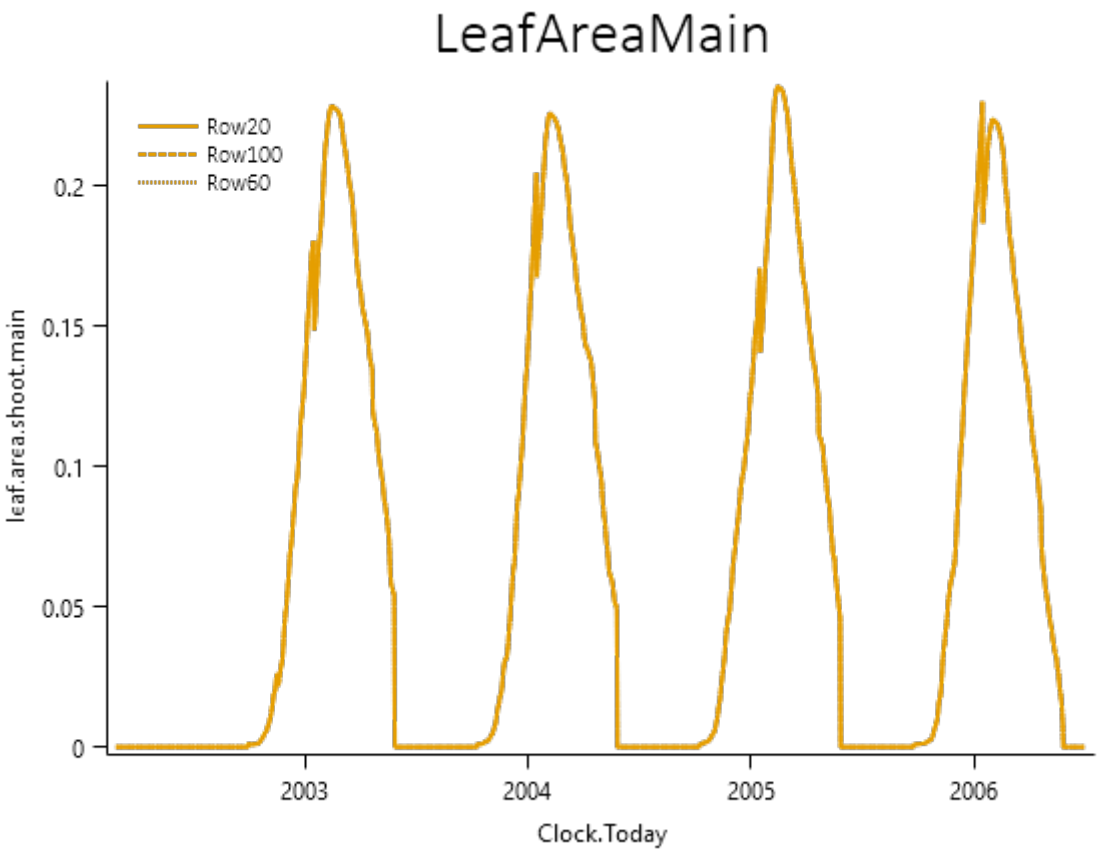

PreBerryBiomass
y = 0.00 x + 2123.41, r2 = 1.00, n = 2
Series
Regression line
1:1 line
Predicted.TotalBerryDW
Observed.TotalBerryDW
0
500
1000
1500
2000

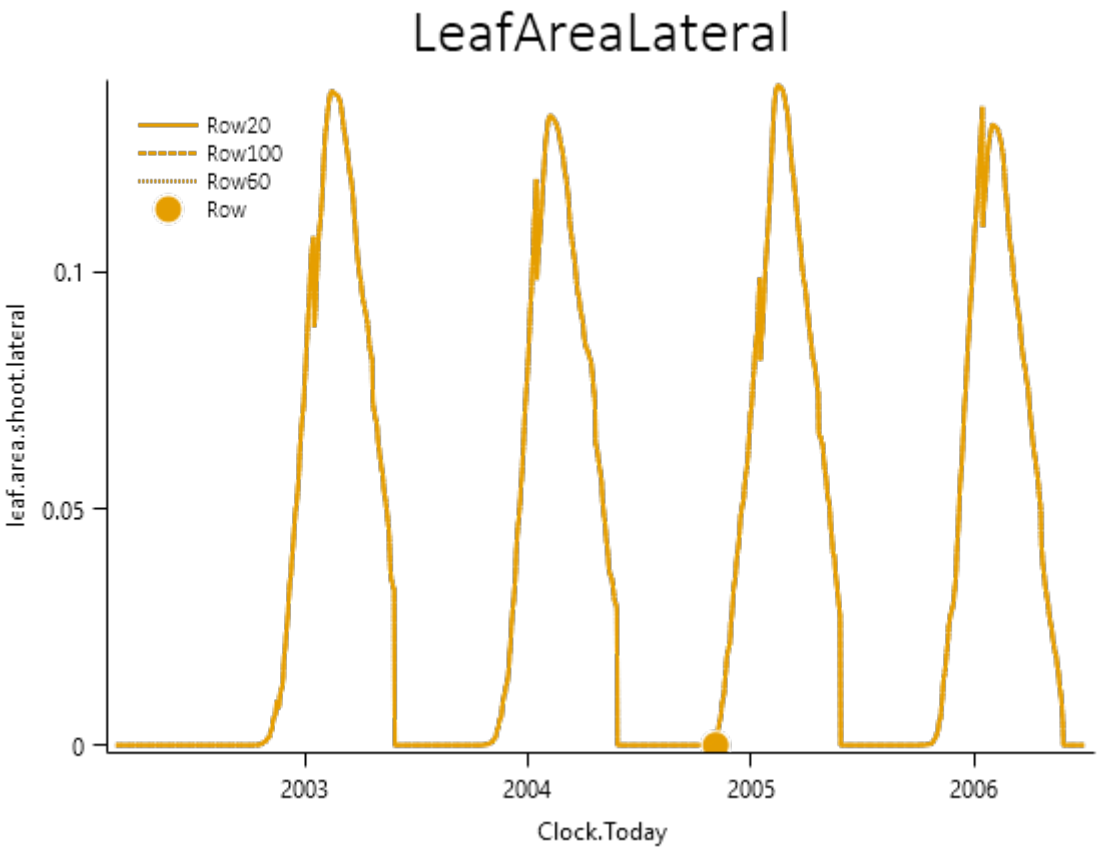

PreShootBiomass
100
Regression line
1:1 line
Predicted.shoot.dw.mean
Observed.shoot.dw.mean
5
10
15

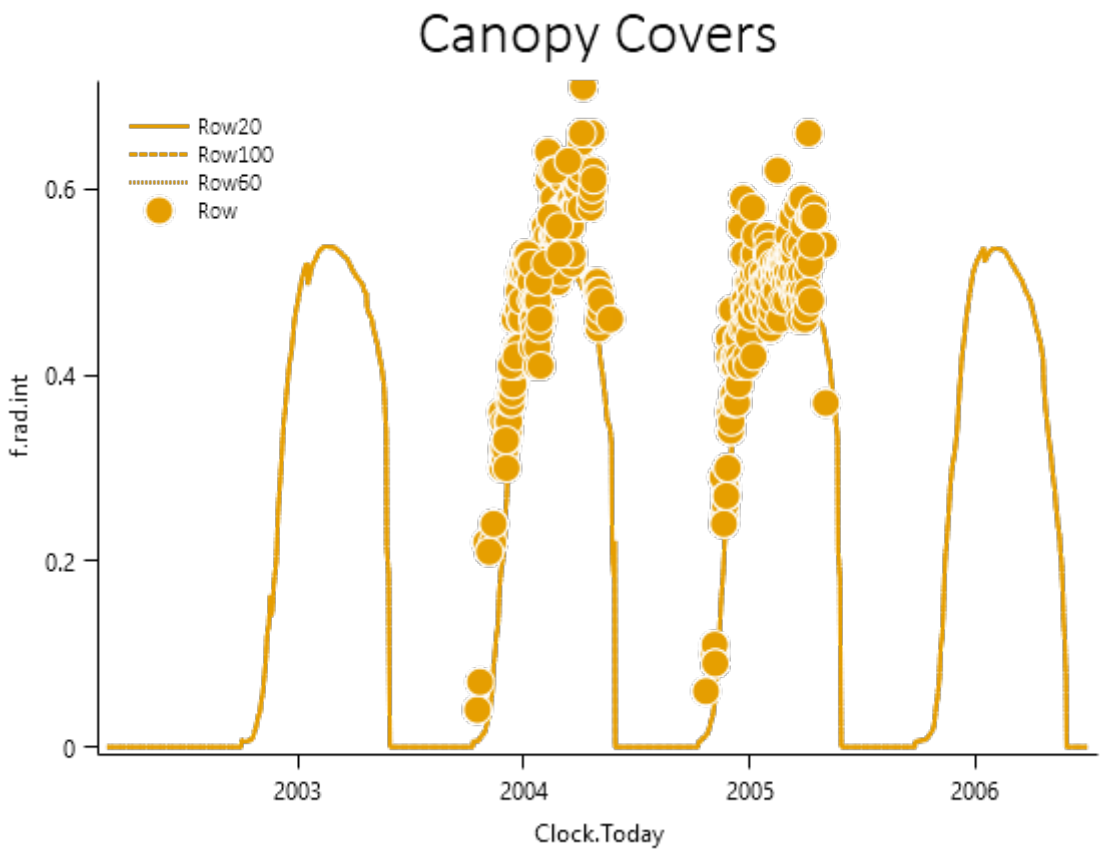
6.1.6 Grape_RPC_2

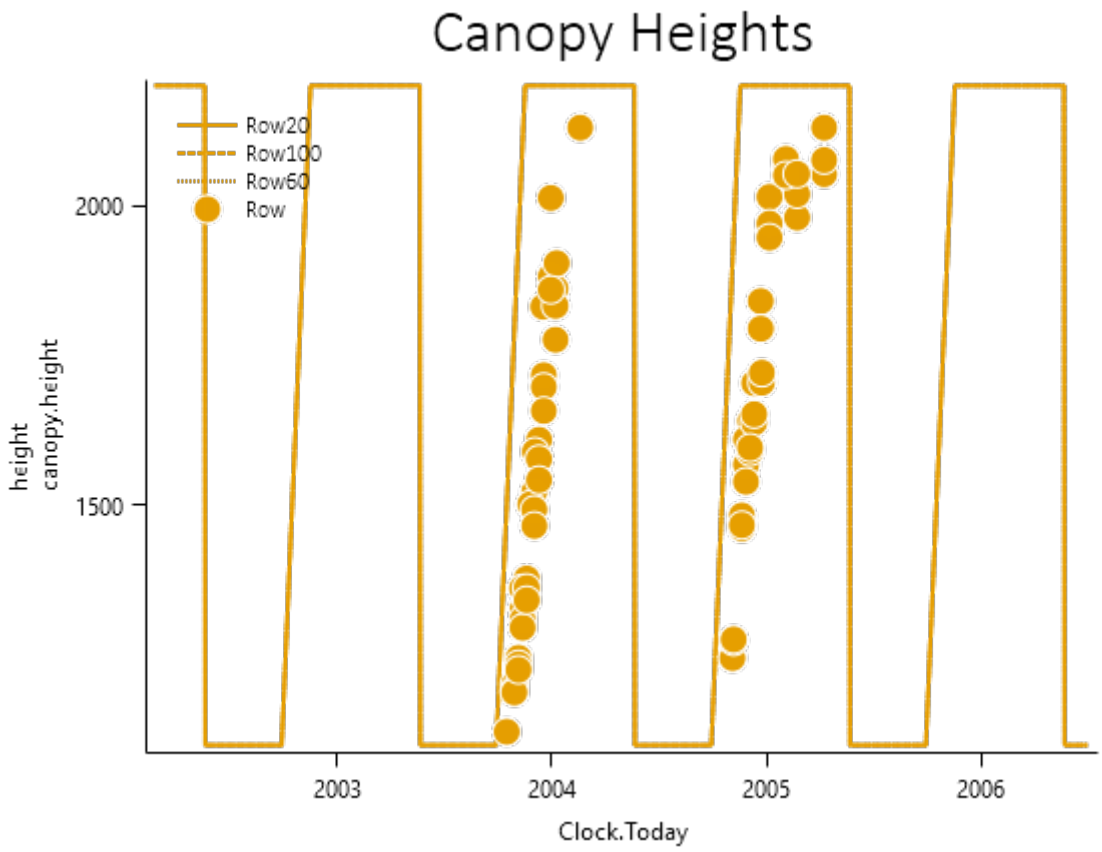


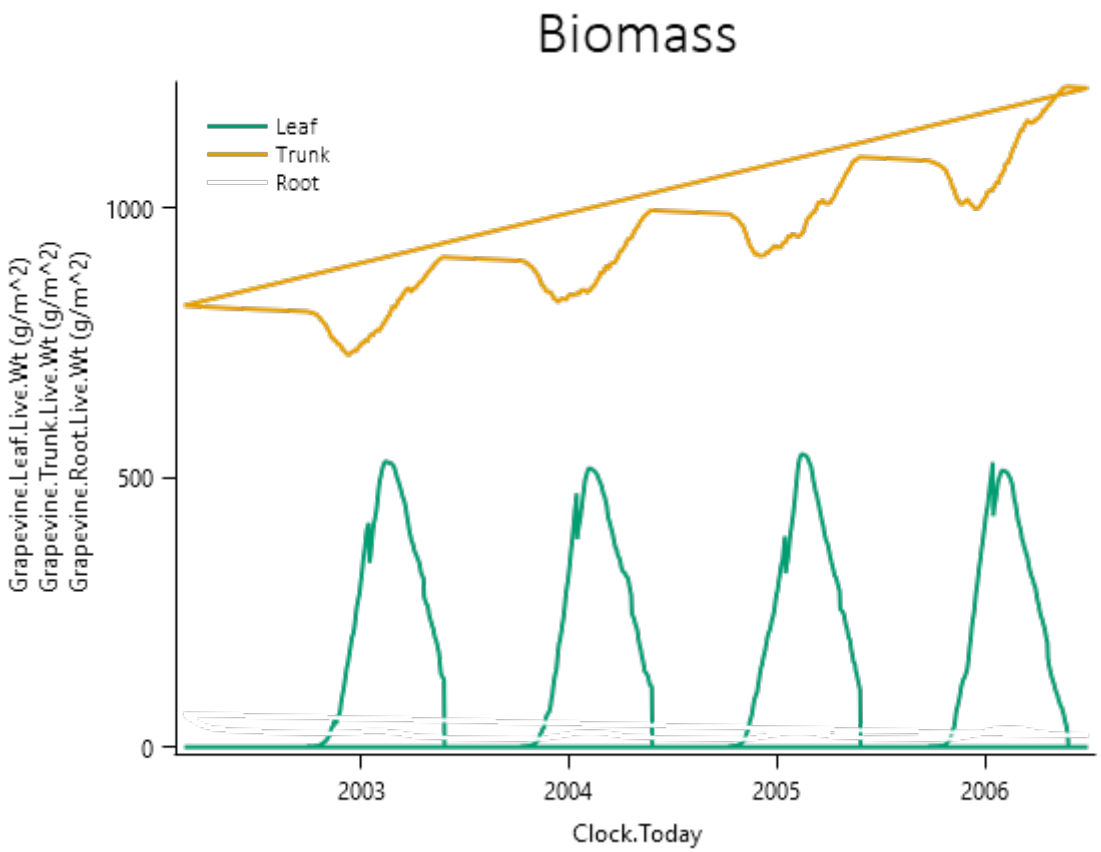


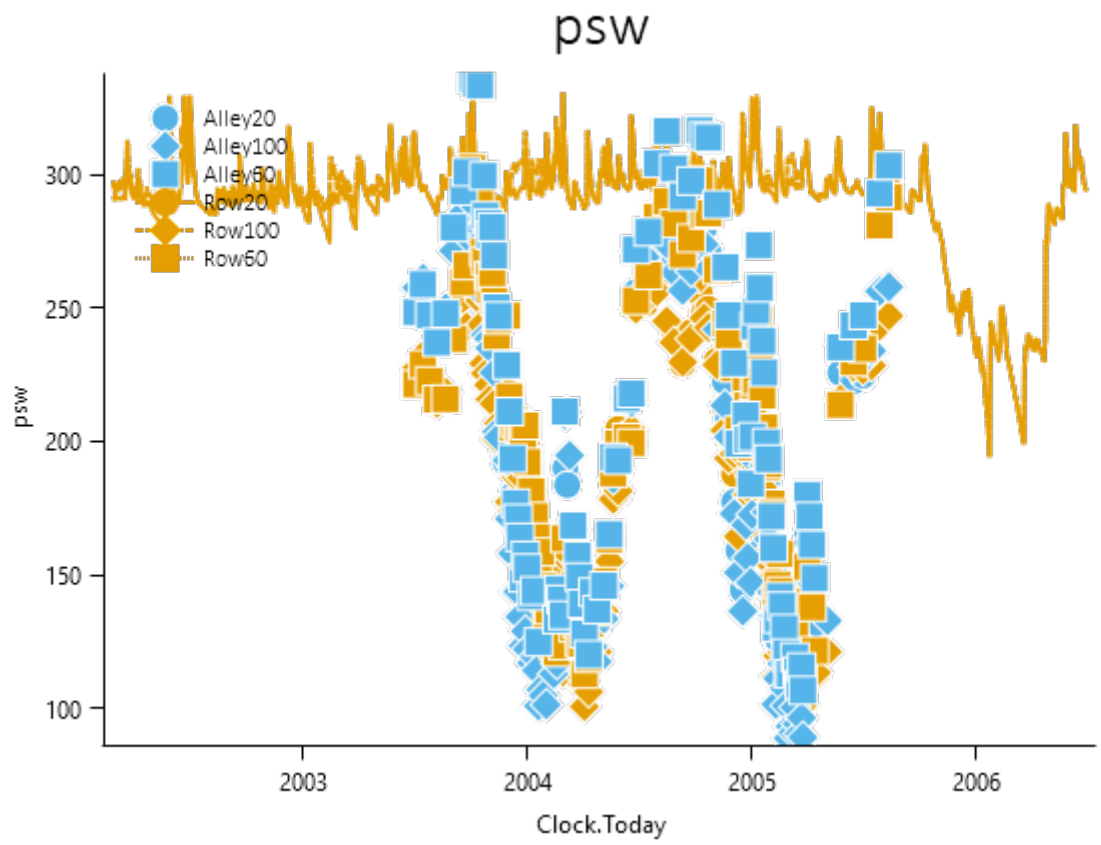

LeafAreaMain
Row20
Row100
Row60
leaf.area.shoot.main
0
0.05
0.1
0.15
0.2
2003
2004
2005
2006
Clock.Today

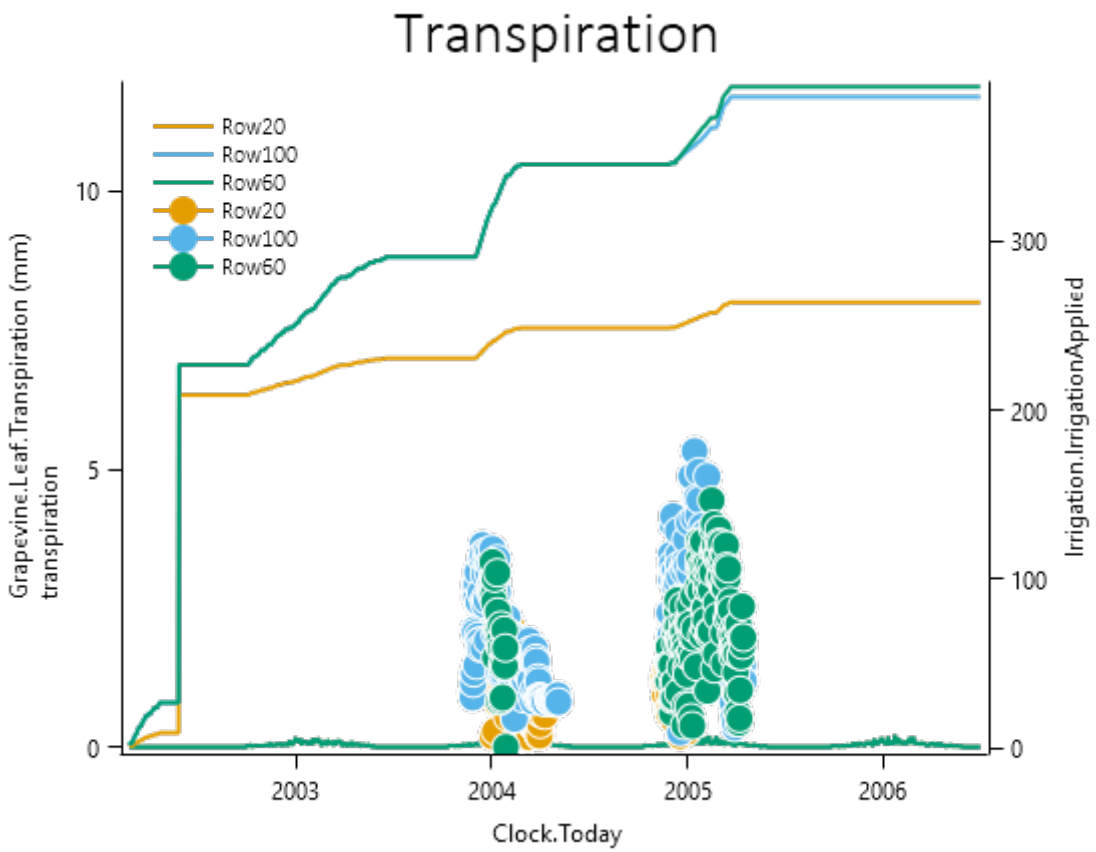

LeafAreaLateral
Row20
Row100
Row60
Row
leaf.area.shoot.lateral
0
0.05
0.1
2003
2004
2005
2006
Clock.Today


Canopy Covers
Row20
Row100
Row60
Row
0.6
0.4
0.2
0
f.rad.int
2003
2004
2005
2006
Clock.Today

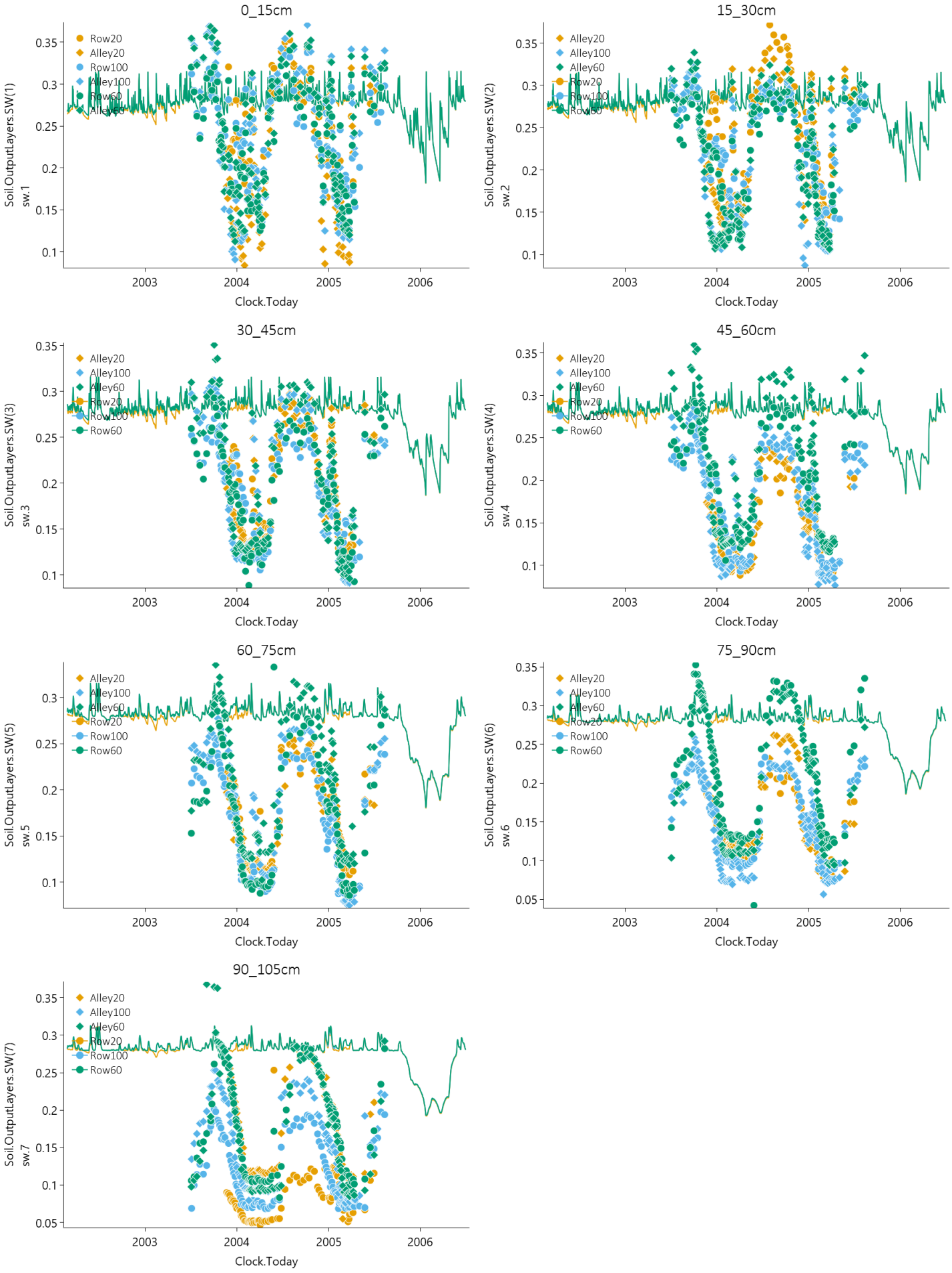

Canopy Heights
Row20
Row100
Row60
Row
2000
1500
height
canopy.height
2003
2004
2005
2006
Clock.Today

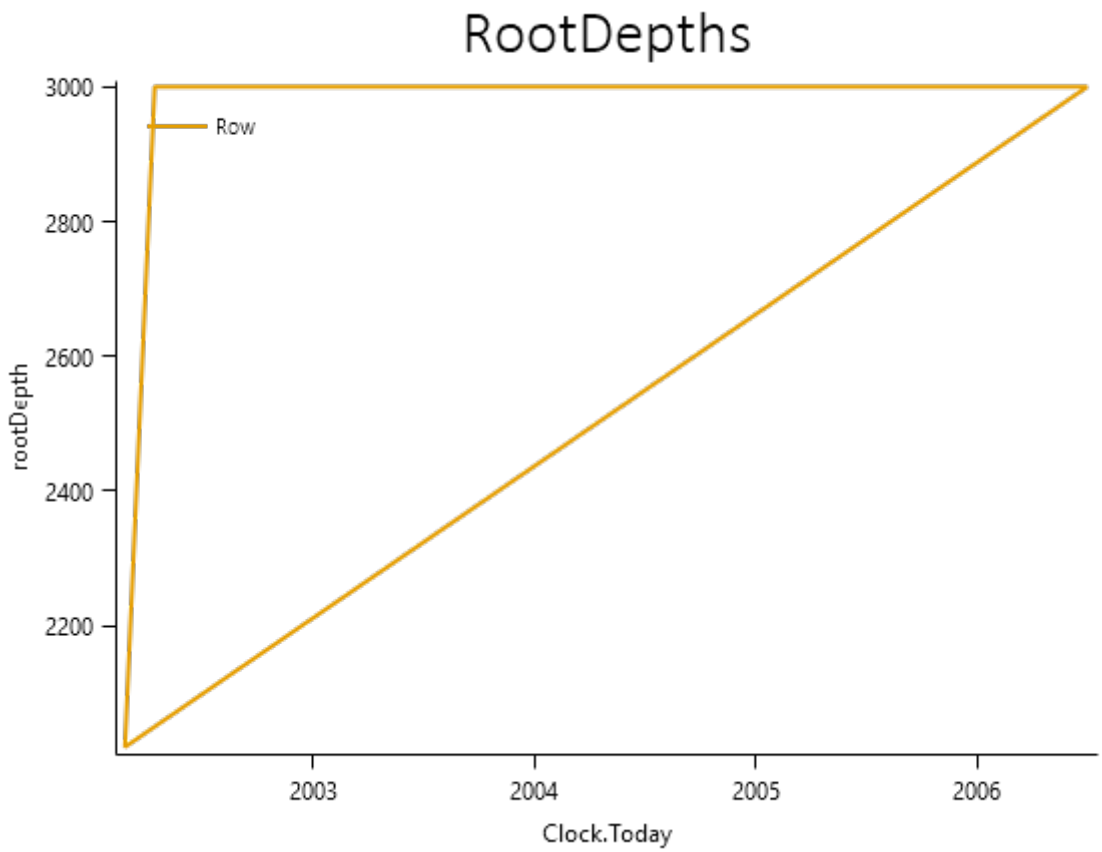

Biomass
Leaf
Trunk
Root
1000
500
0
Grapevine.Leaf.Live.Wt (g/m^2)
Grapevine.Trunk.Live.Wt (g/m^2)
Grapevine.Root.Live.Wt (g/m^2)
2003
2004
2005
2006
Clock.Today

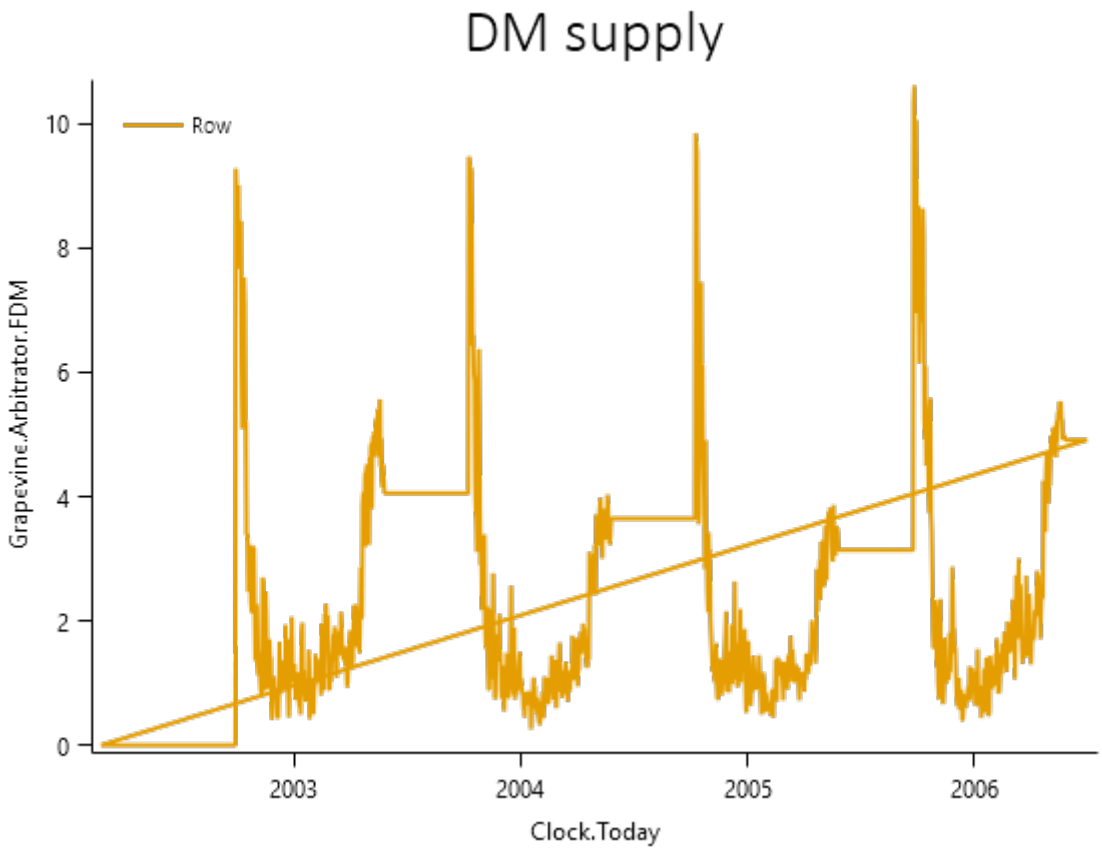

psw
Alley20
Alley100
Alley60
Row20
Row100
Row60
300
250
200
150
100
psw
2003
2004
2005
2006
Clock.Today

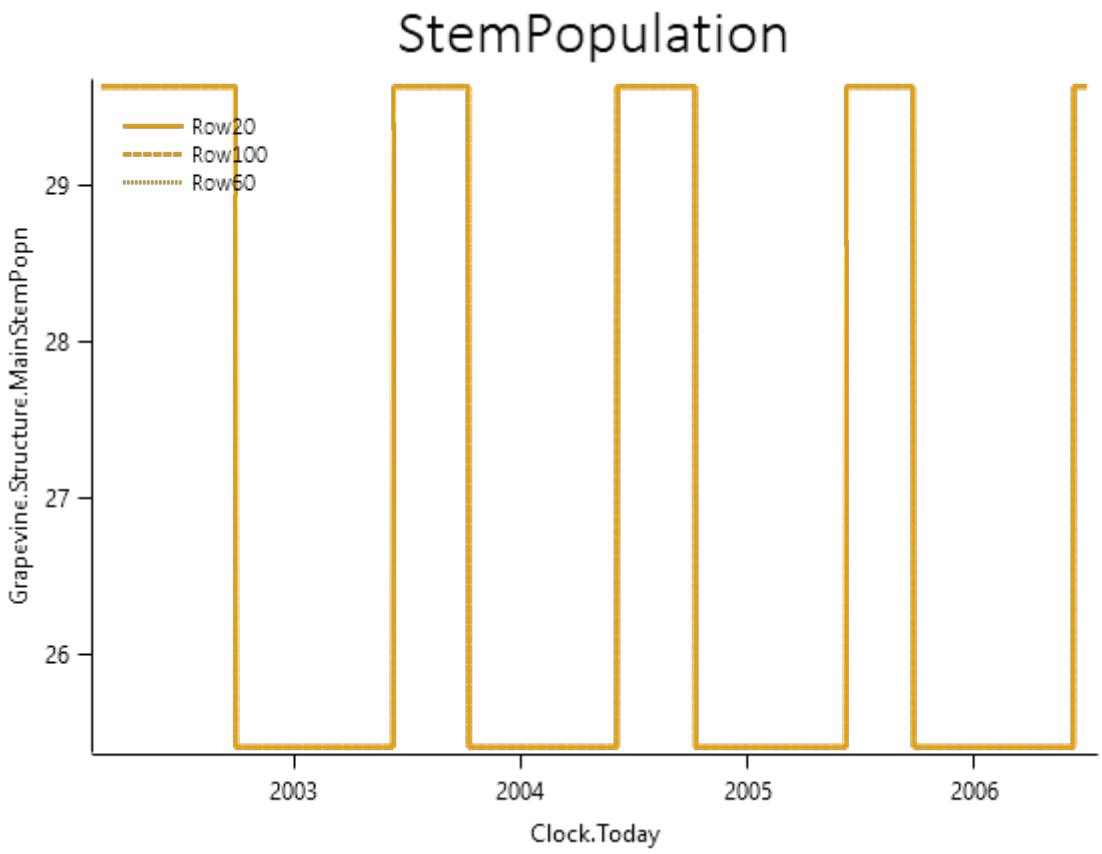

Transpiration
Row20
Row100
Row60
Row20
Row100
Row60
10
5
0
300
200
100
0
2003
2004
2005
2006
Clock.Today
Grapevine.Leaf.Transpiration (mm)
transpiration
Irrigation.IrrigationApplied

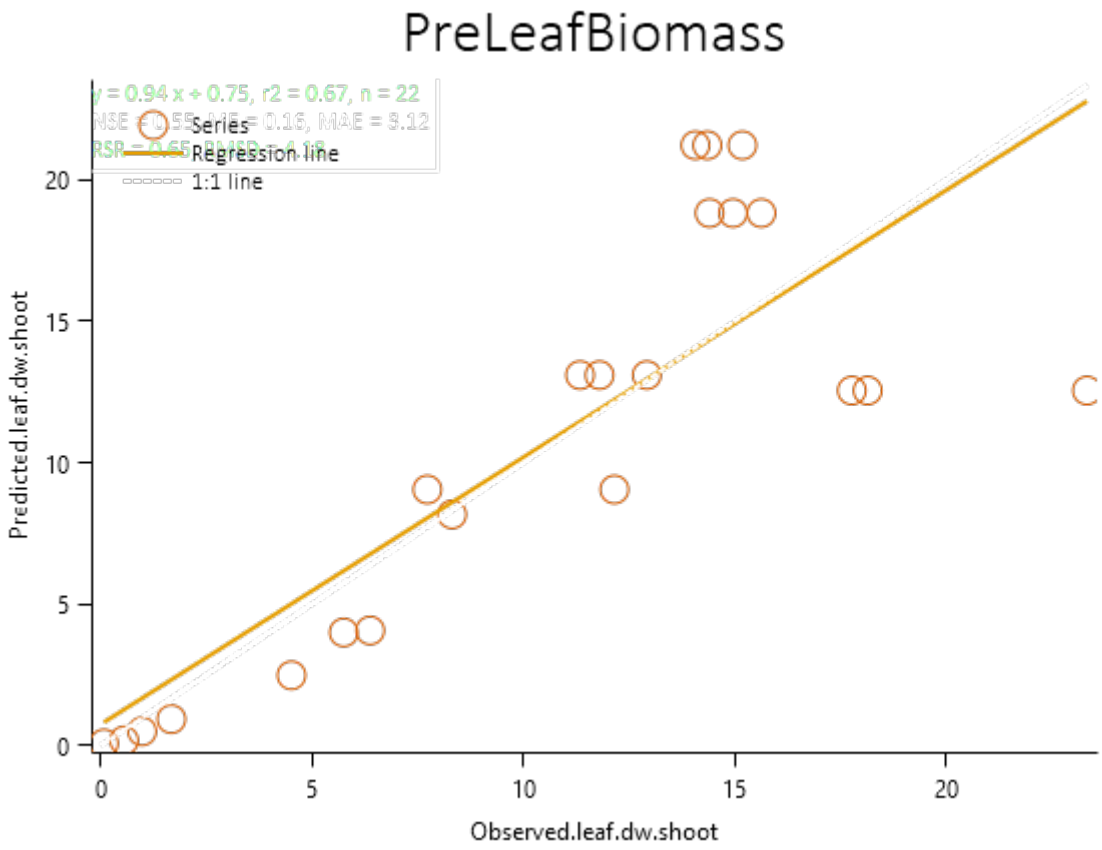
6.1.6.1 SoilWater

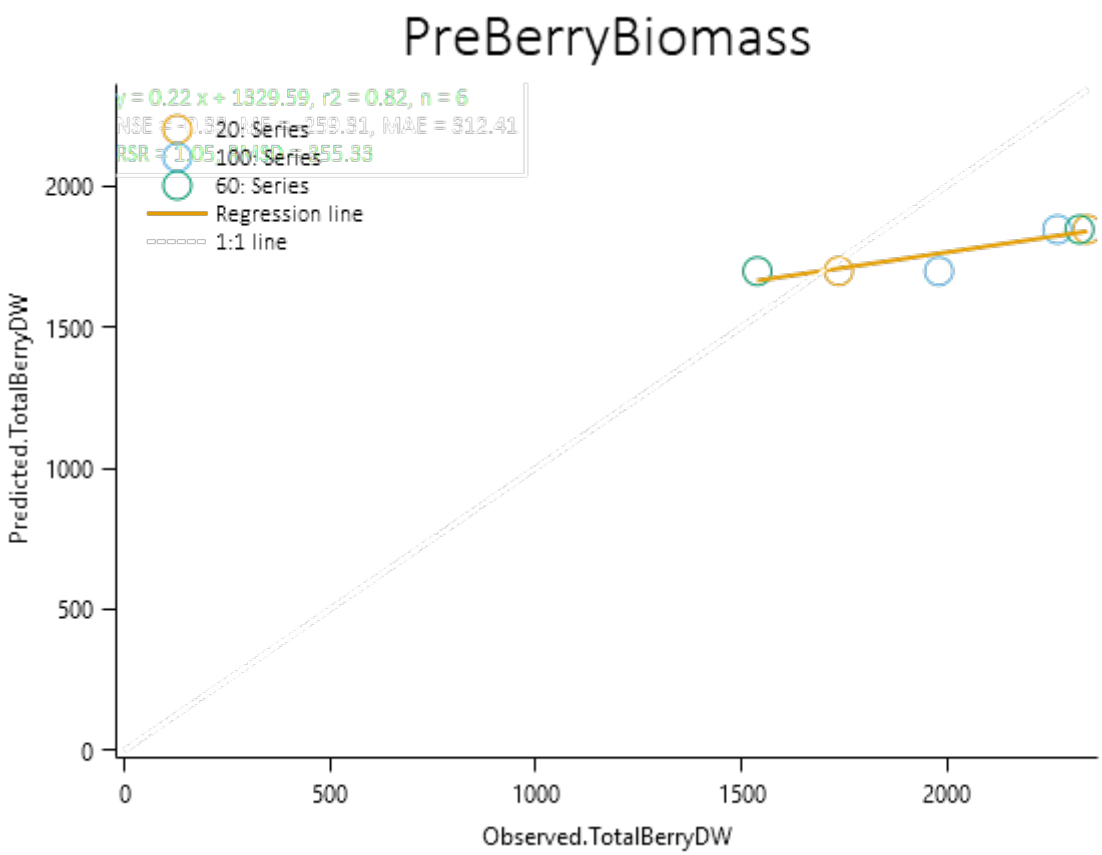


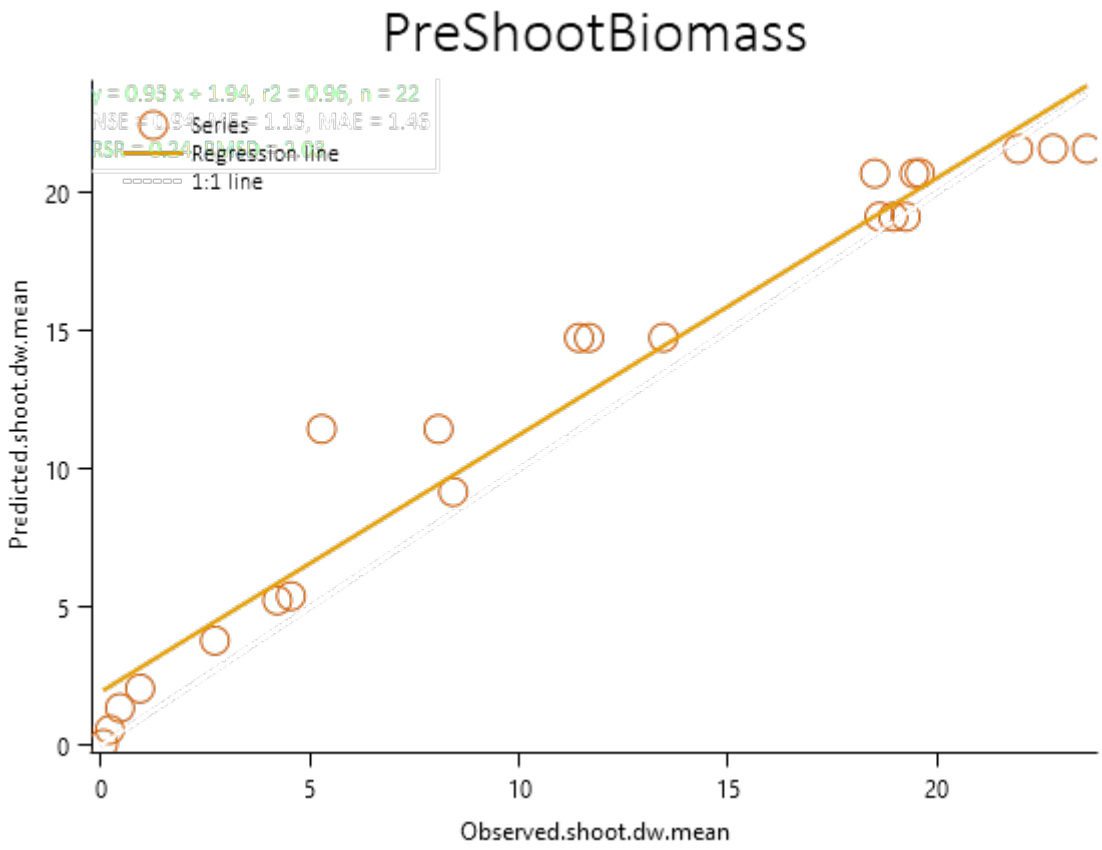

RootDepths
Row
3000
2800
2600
2400
2200
rootDepth
2003
2004
2005
2006
Clock.Today

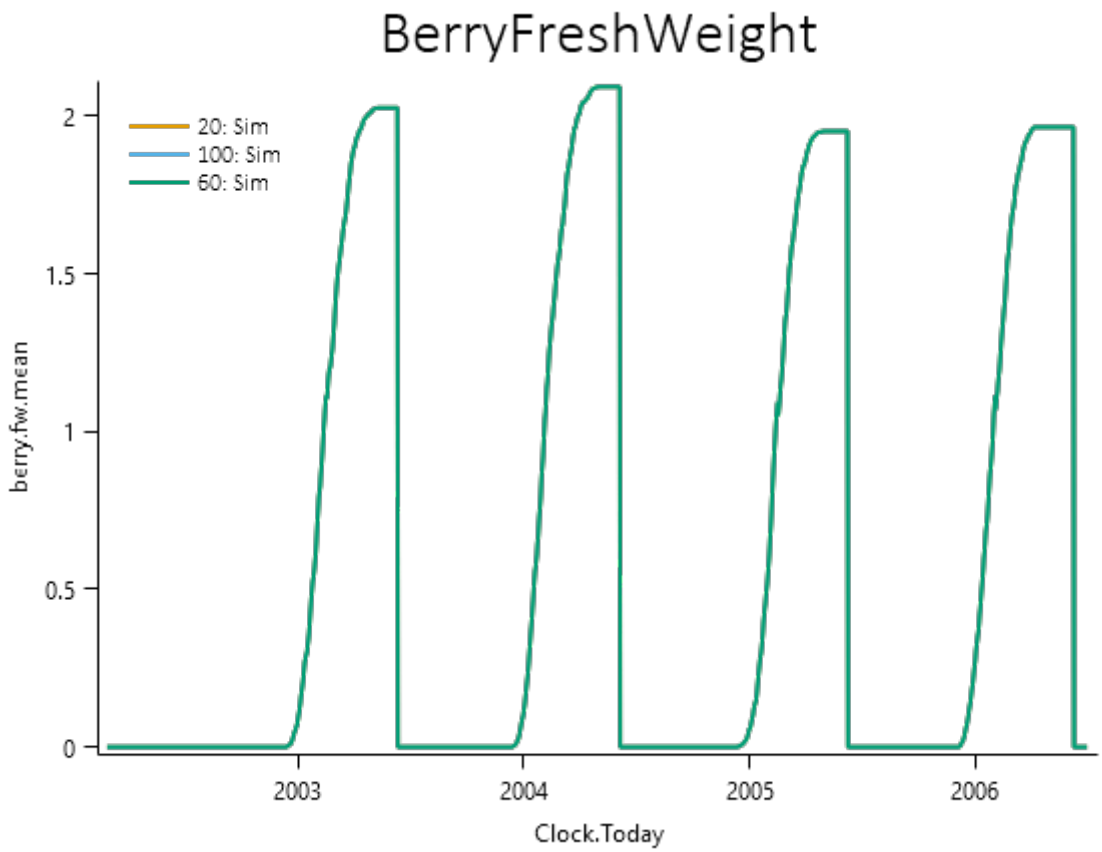

DM supply
Row
10
8
6
4
2
0
Grapevine.Arbitrator.FDM
2003
2004
2005
2006
Clock.Today

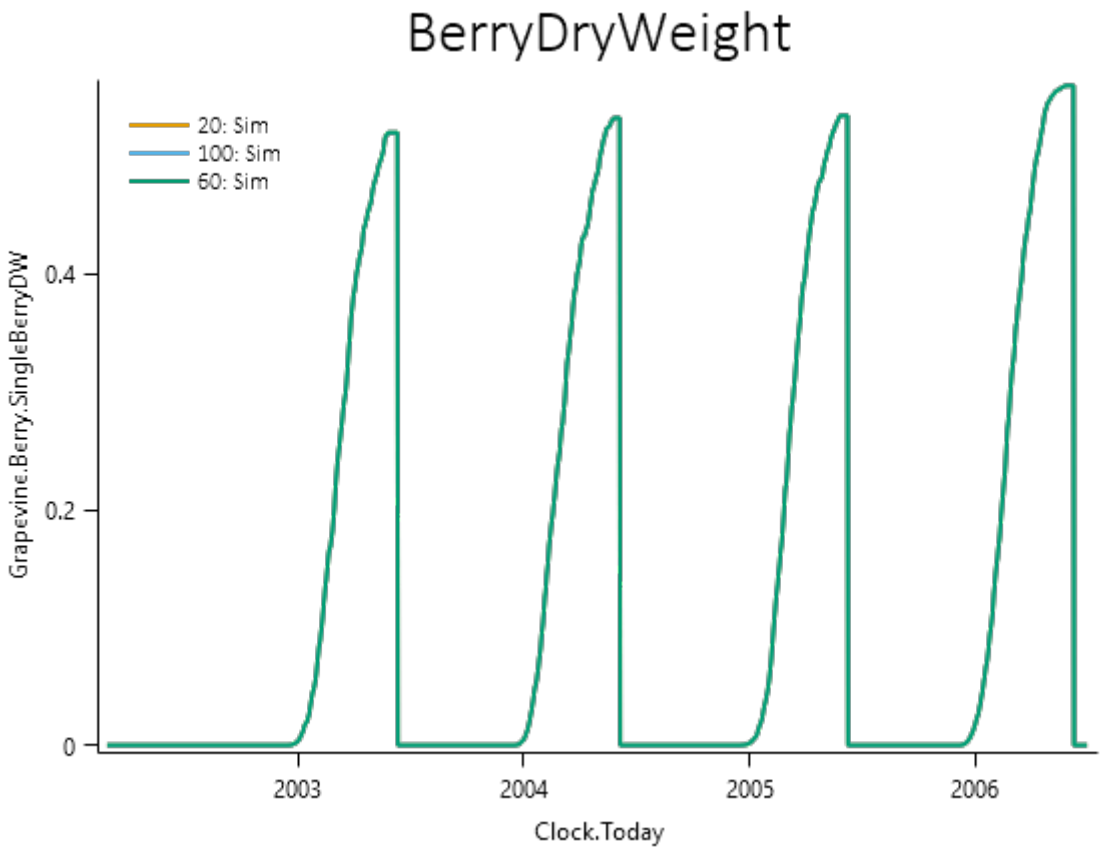

StemPopulation
Row20
Row100
Row60
Grapevine.Structure.MainStemPopn
29
28
27
26
2003
2004
2005
2006
Clock.Today


PreLeafBiomass
y = 0.94 x + 0.75, r2 = 0.67, n = 22
Series
Regression line
1:1 line
Predicted.leaf.dw.shoot
20
15
10
5
0
0
5
10
15
20
Observed.leaf.dw.shoot

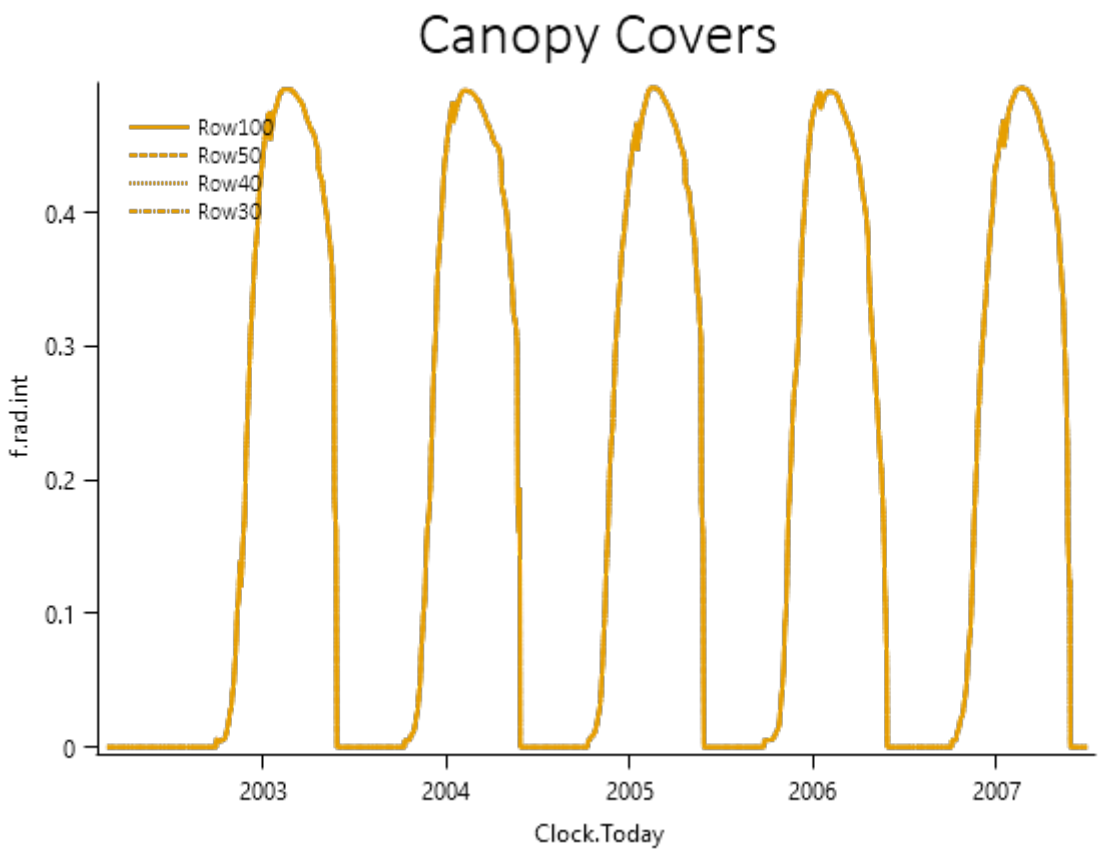

PreBerryBiomass
y = 0.22 x + 1329.59, r2 = 0.82, n = 6
20: Series
100: Series
60: Series
Regression line
1:1 line
Predicted.TotalBerryDW
Observed.TotalBerryDW
0
500
1000
1500
2000

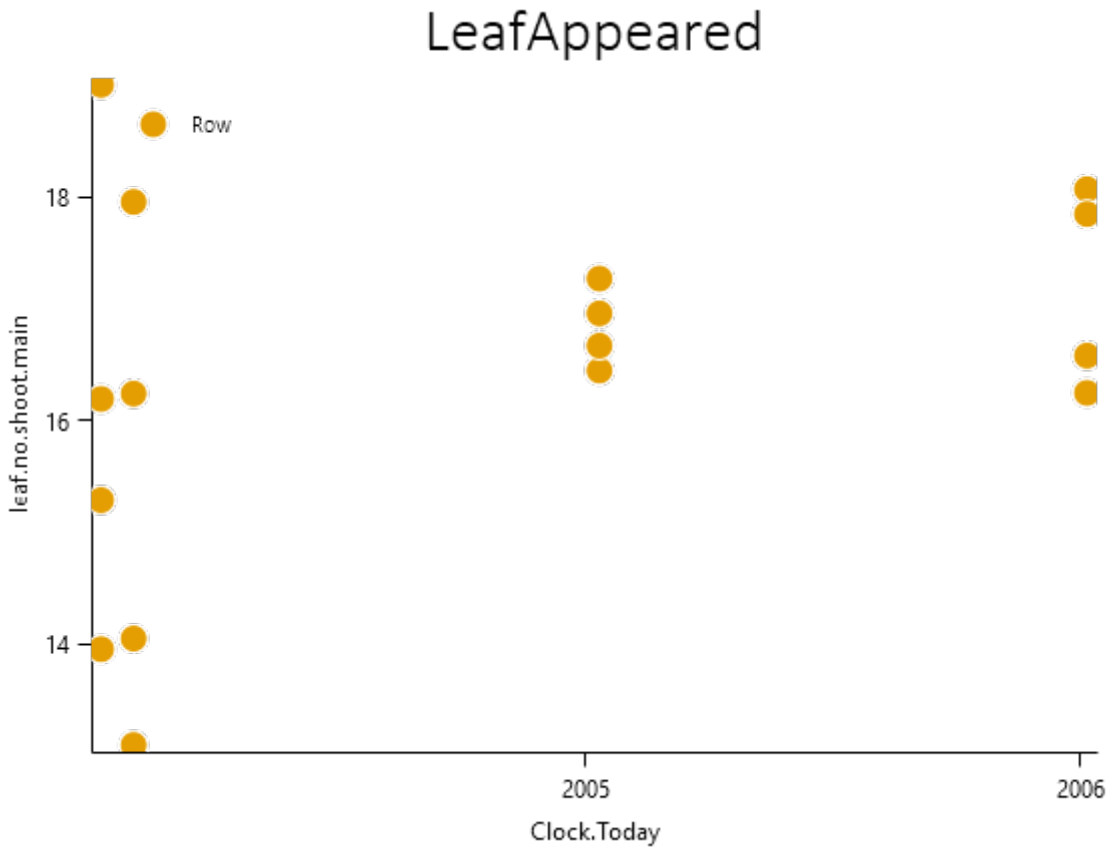

PreShootBiomass
y = 0.93 x + 1.94, r2 = 0.96, n = 22
Series
Regression line
1:1 line
Predicted.shoot.dw.mean
Observed.shoot.dw.mean
0
5
10
15
20

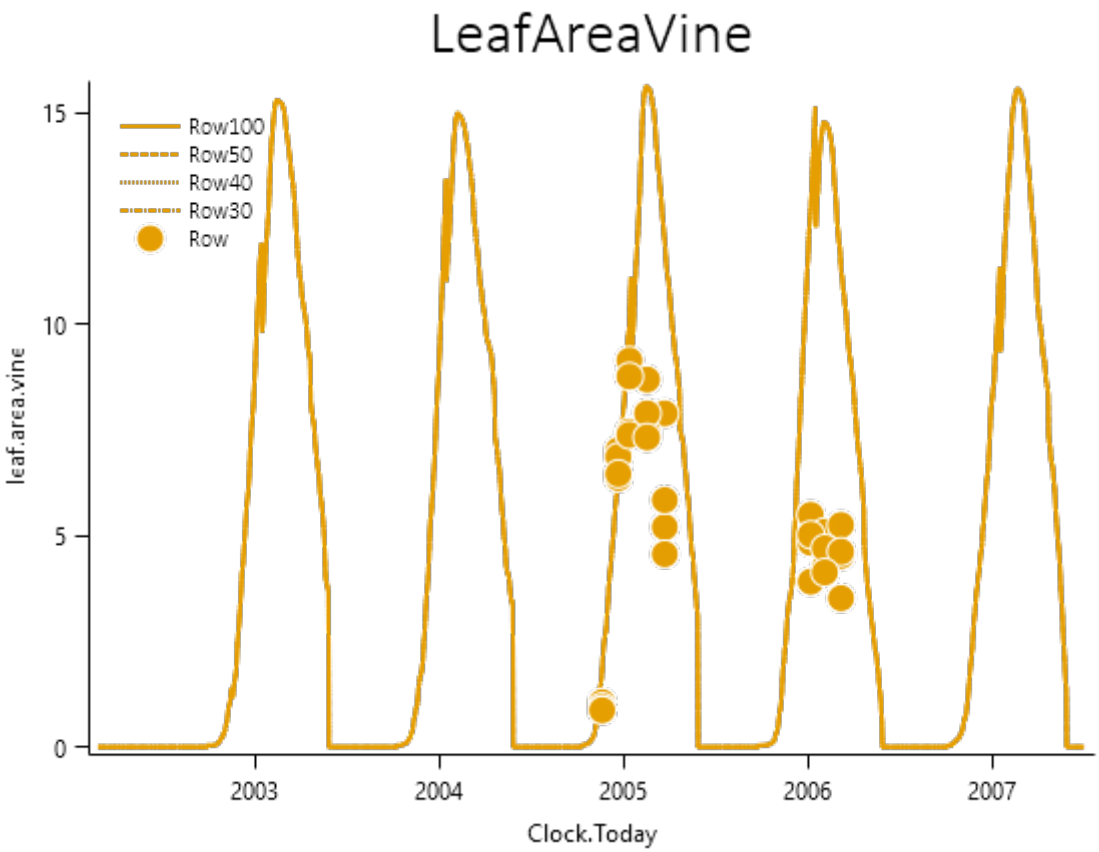

BerryFreshWeight
20: Sim
100: Sim
60: Sim
berry.fw.mean
2
1.5
1
0.5
0
2003
2004
2005
2006
Clock.Today

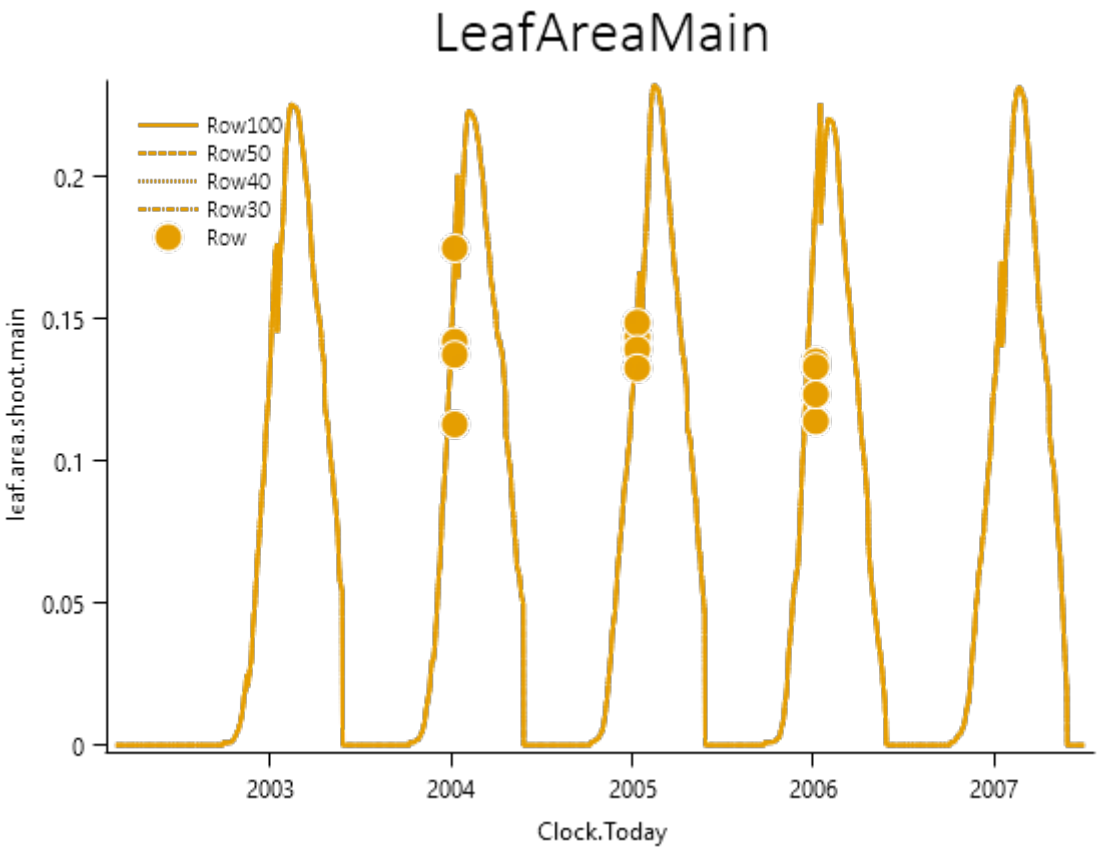

BerryDryWeight
20: Sim
100: Sim
60: Sim
Grapevine.Berry.SingleBerryDW
0.4
0.2
0
2003
2004
2005
2006
Clock.Today

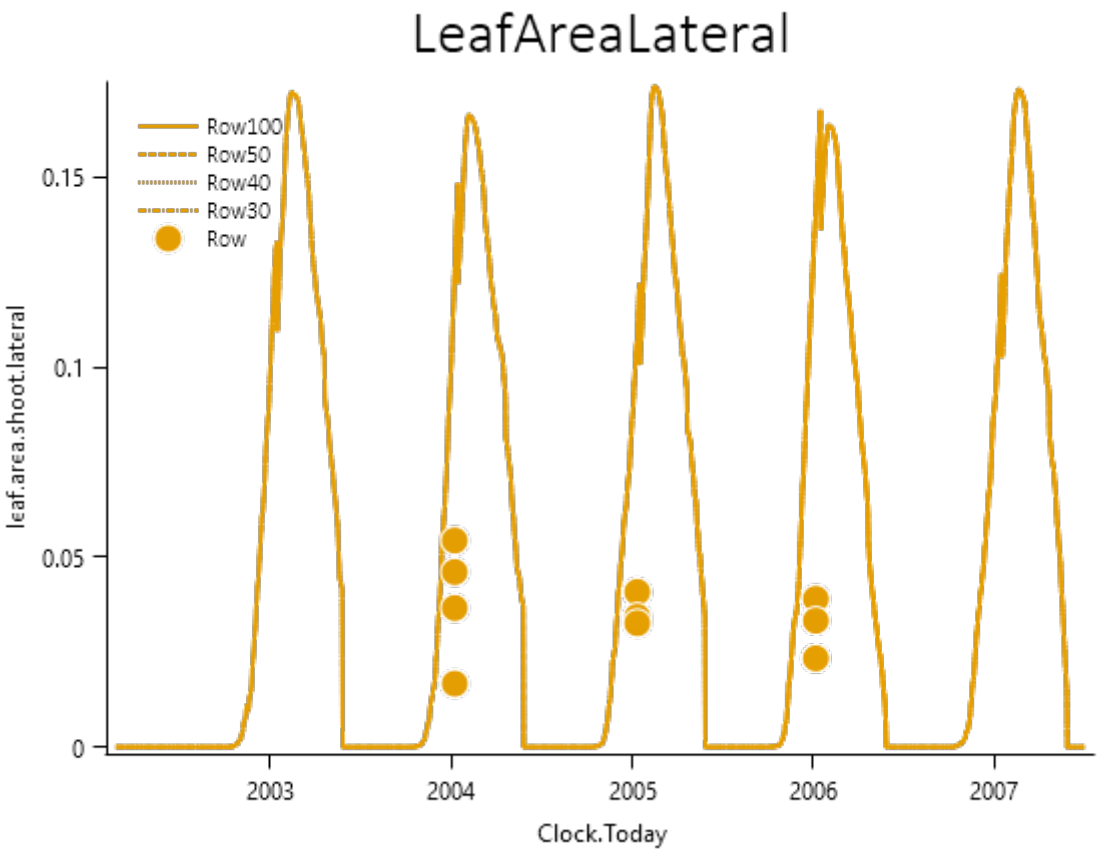
6.1.7 Renwick

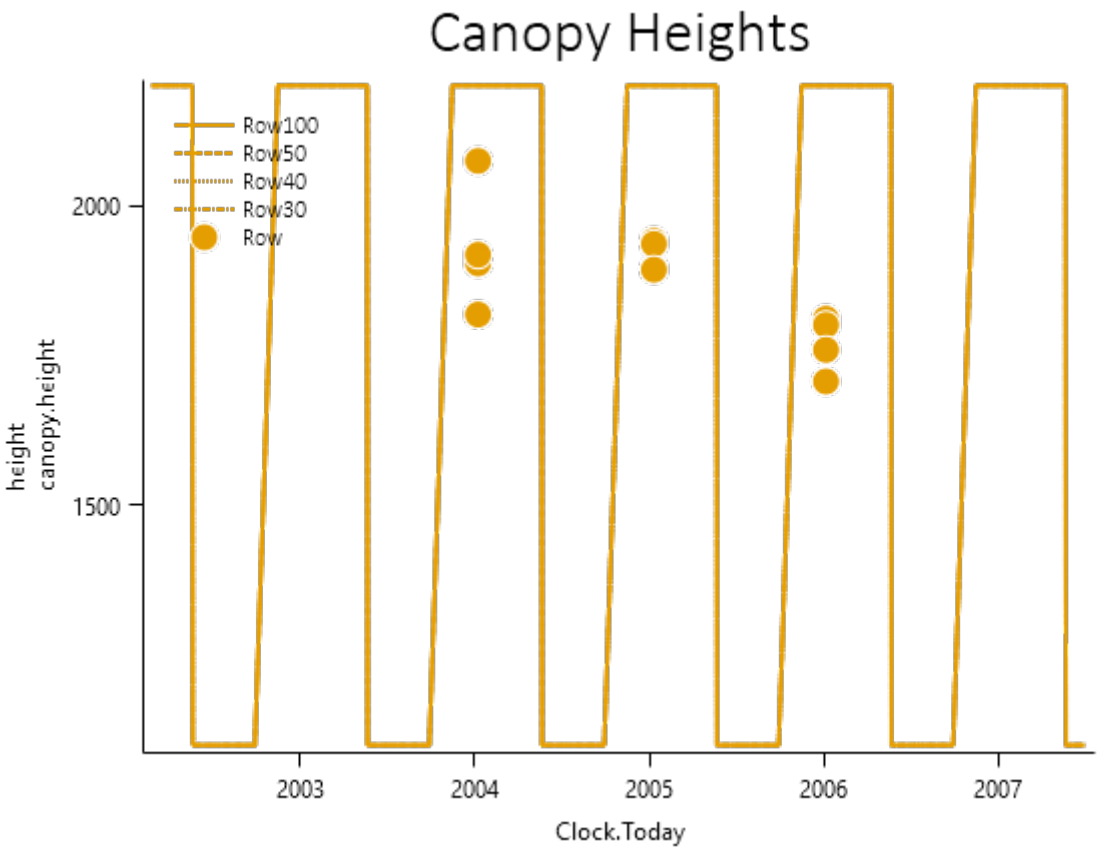

Canopy Covers
Row100
Row50
Row40
Row30
f.rad.int
0.4
0.3
0.2
0.1
0
2003
2004
2005
2006
2007
Clock.Today

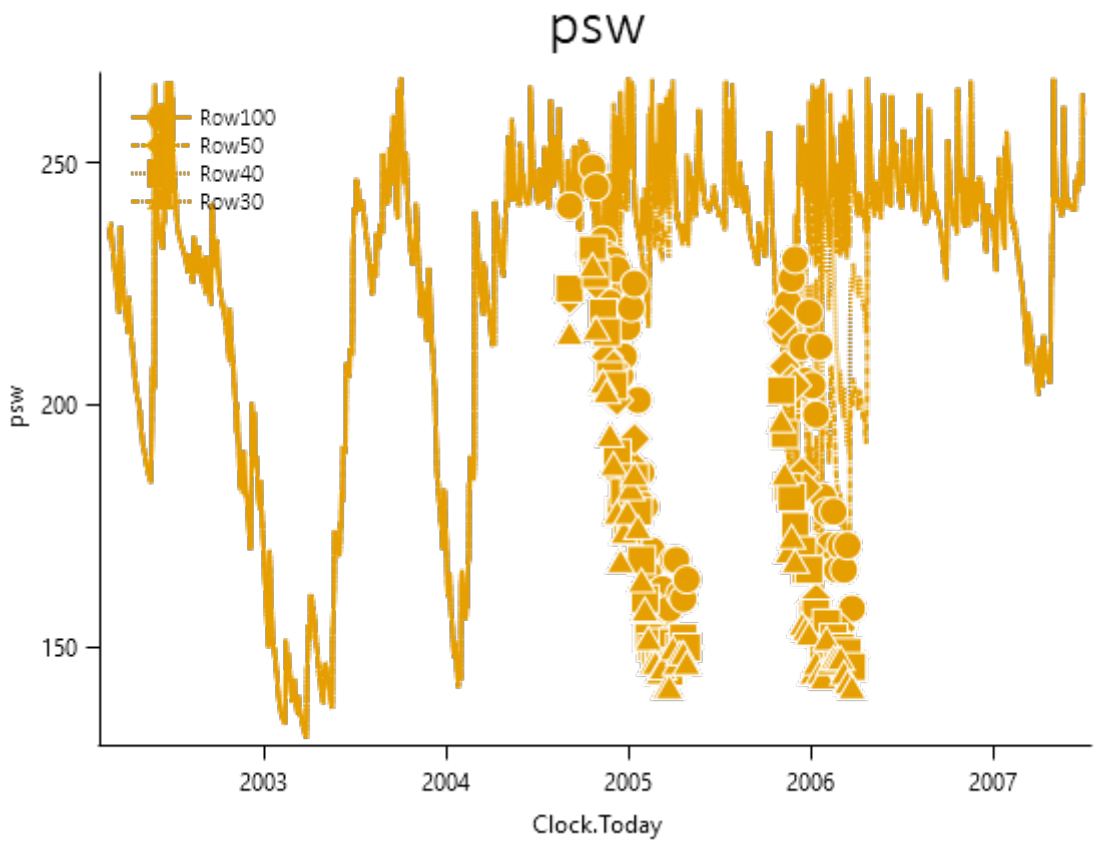

LeafAppeared
Row
leaf.no.shoot.main
18
16
14
2005
2006
Clock.Today

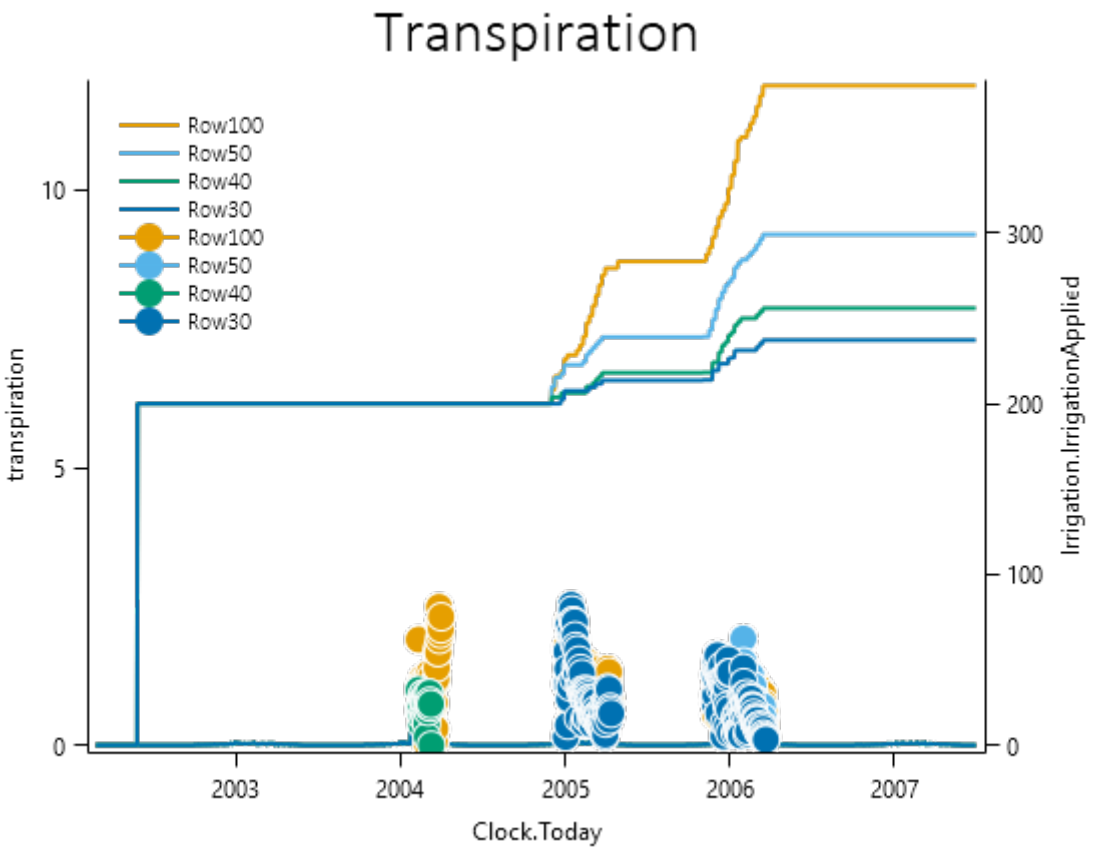

LeafAreaVine
Row100
Row50
Row40
Row30
Row
leaf.area.vine
15
10
5
0
2003
2004
2005
2006
2007
Clock.Today


LeafAreaMain
Row100
Row50
Row40
Row30
Row
leaf.area.shoot.main
0.2
0.15
0.1
0.05
0
2003
2004
2005
2006
2007
Clock.Today

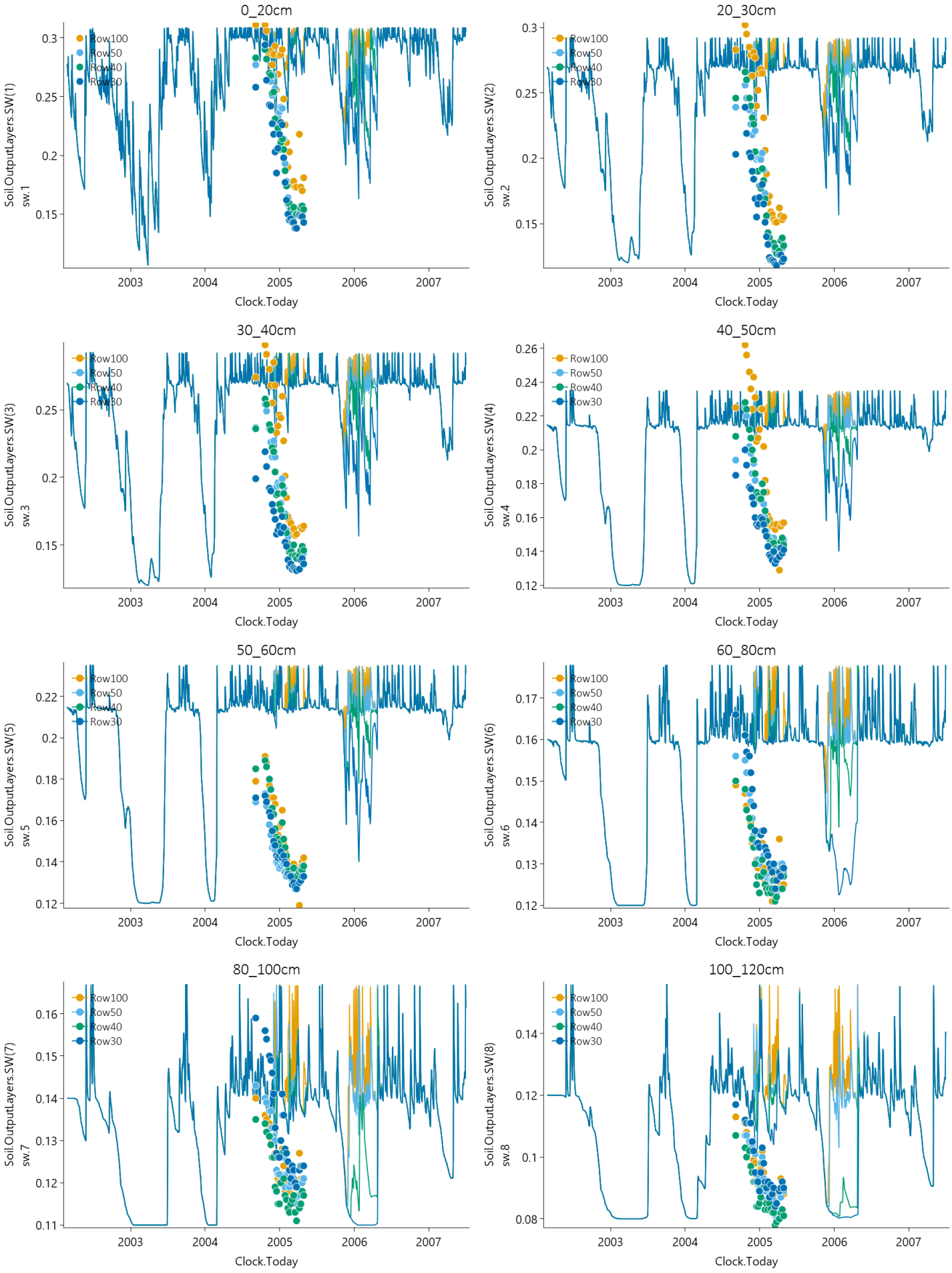

LeafAreaLateral
Row100
Row50
Row40
Row30
Row
0.15
0.1
0.05
0
leaf.area.shoot.lateral
2003
2004
2005
2006
2007
Clock.Today

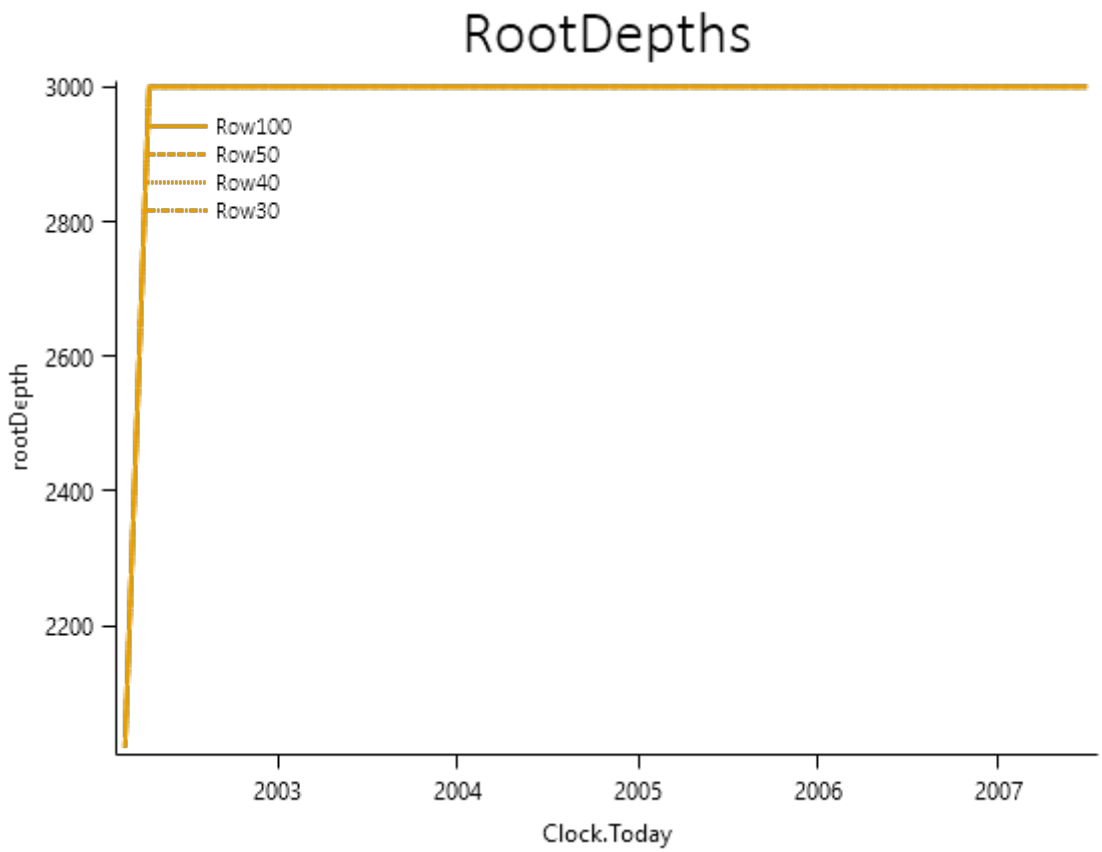

Canopy Heights
Row100
Row50
Row40
Row30
Row
2000
1500
height
canopy.height
2003
2004
2005
2006
2007
Clock.Today

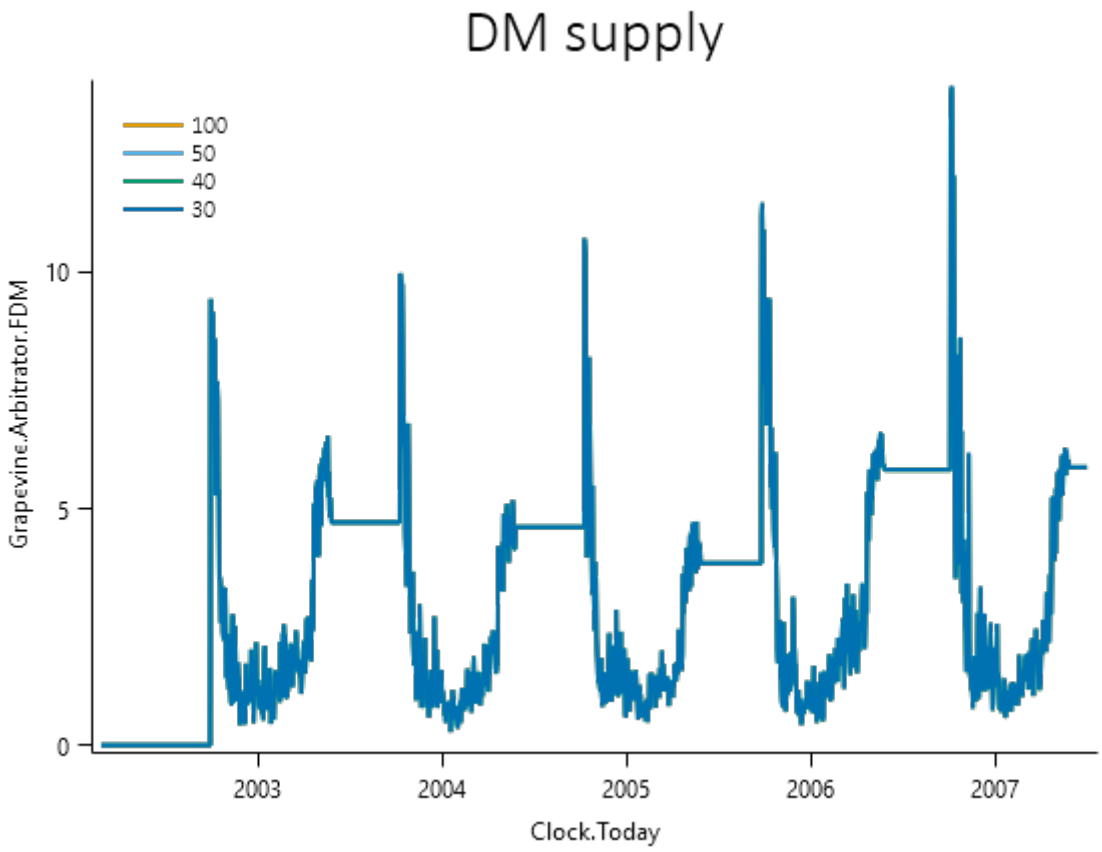

psw
Row100
Row50
Row40
Row30
250
200
150
psw
2003
2004
2005
2006
2007
Clock.Today

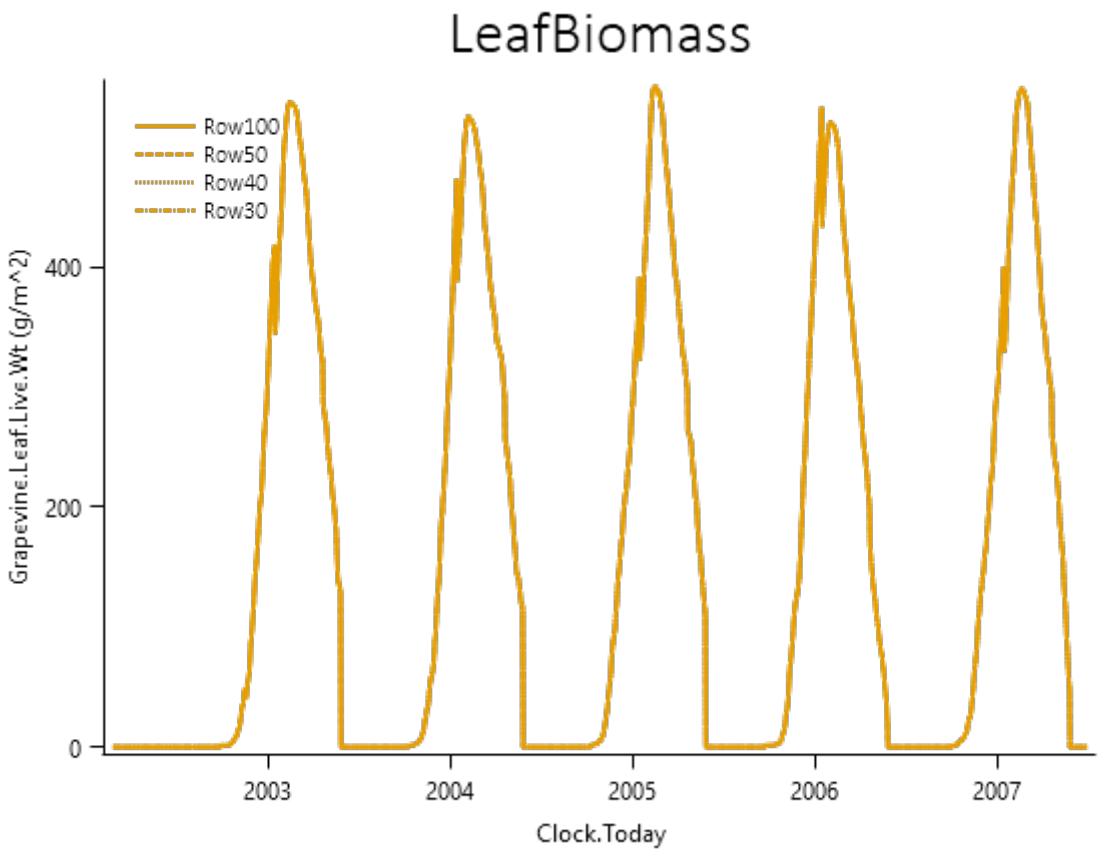

Transpiration
Row100
Row50
Row40
Row30
Row100
Row50
Row40
Row30
10
5
0
transpiration
300
200
100
0
Irrigation.IrrigationApplied
2003
2004
2005
2006
2007
Clock.Today

6.1.7.1 SoilWater

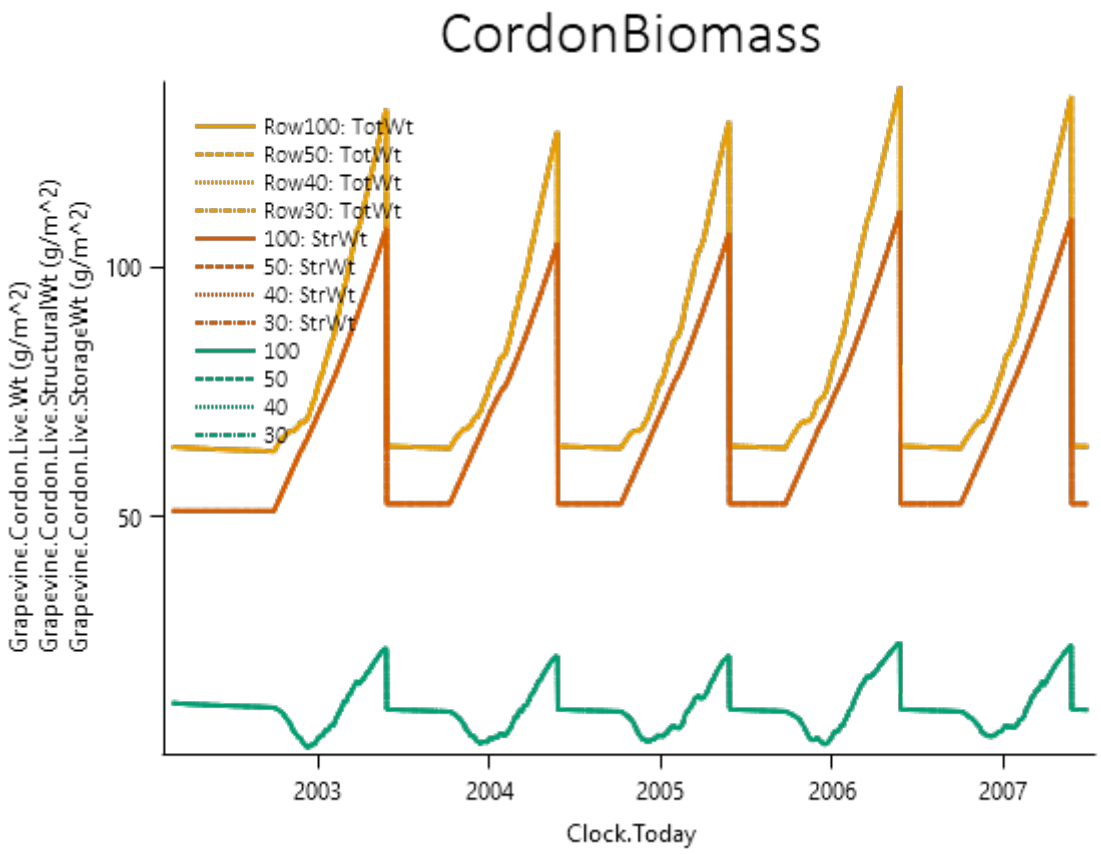


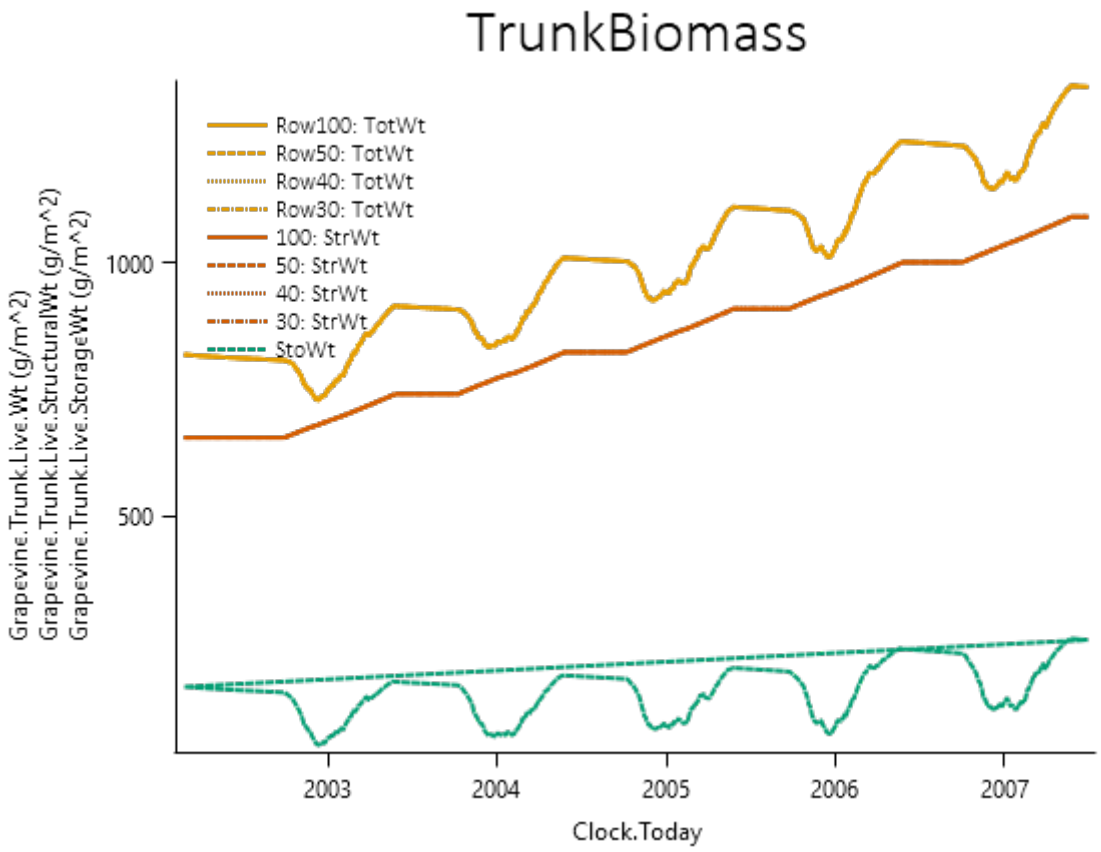

RootDepths
3000
2800
2600
2400
2200
rootDepth
Row100
Row50
Row40
Row30
2003
2004
2005
2006
2007
Clock.Today

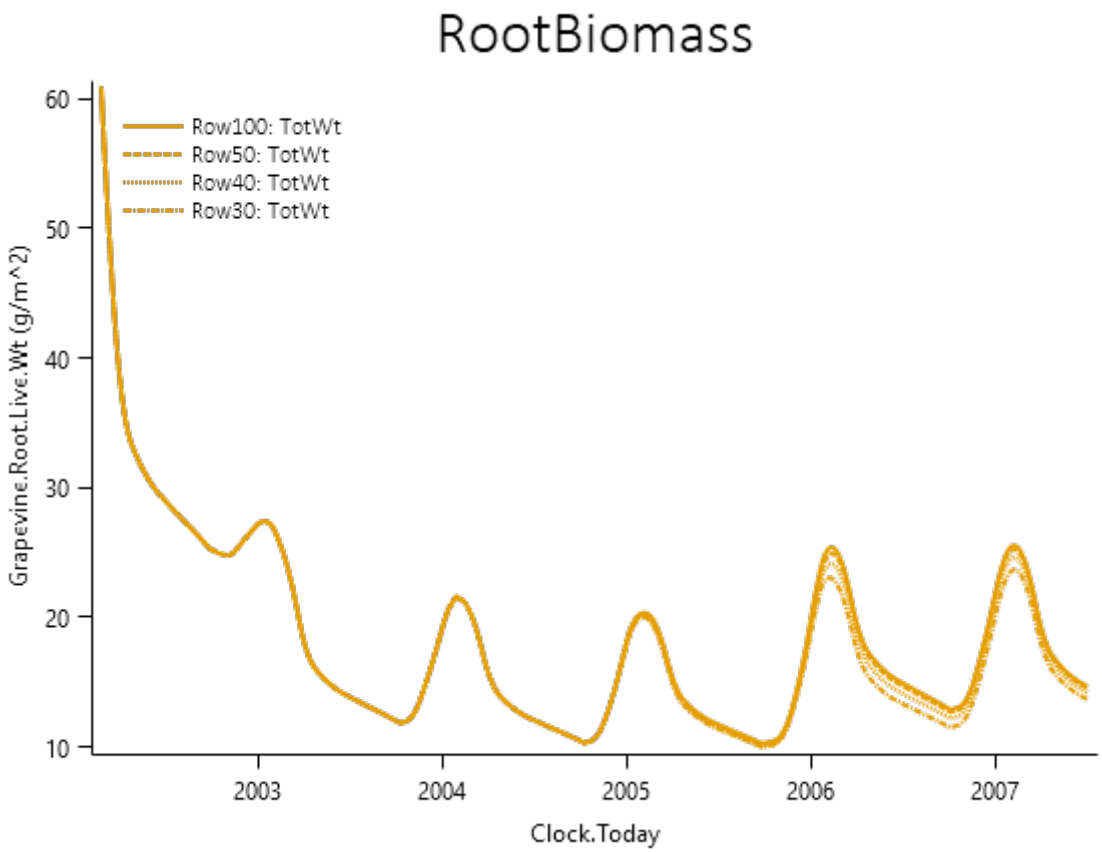

DM supply
100
50
40
30
10
5
0
Grapevine.Arbitrator.FDM
2003
2004
2005
2006
2007
Clock.Today



ShootBiomass


CordonBiomass
Row100: TotWt
Row50: TotWt
Row40: TotWt
Row30: TotWt
100: StrWt
50: StrWt
40: StrWt
30: StrWt
100
50
40
30
Grapevine.Cordon.Live.Wt (g/m^2)
Grapevine.Cordon.Live.StructuralWt (g/m^2)
Grapevine.Cordon.Live.StorageWt (g/m^2)
100
50
2003
2004
2005
2006
2007
Clock.Today


TrunkBiomass
Row100: TotWt
Row50: TotWt
Row40: TotWt
Row30: TotWt
100: StrWt
50: StrWt
40: StrWt
30: StrWt
StoWt
Grapevine.Trunk.Live.Wt (g/m^2)
Grapevine.Trunk.Live.StructuralWt (g/m^2)
Grapevine.Trunk.Live.StorageWt (g/m^2)
1000
500
2003
2004
2005
2006
2007
Clock.Today



6.2 Chardonnay

List of experiments.

Experiment Name	Design (Number of Treatments)
Chardonnay	Climate (6)

6.3 Merlot

List of experiments.

Experiment Name	Design (Number of Treatments)
Merlot	Climate (2)

6.4 PinotNoir

List of experiments.

Experiment Name	Design (Number of Treatments)
PinotNoir	Climate (7)

6.5 PinotGris

List of experiments.

Experiment Name	Design (Number of Treatments)
PinotGris	Climate (3)

6.6 Statistics

6.6.1 PredictedObserved

Variable	n	Slope	Intercept	R2	RMSE	NSE	ME	M
BerryMass								
BerryNum	3409	0.536	27.549	0.404	9.386	0.349	-1.283	6.
BudBurstDOY								
BunchesPerVine								
f.rad.int	768	0.918	0.031	0.644	0.068	0.520	-0.008	0.
FloweringDOY								
psw	789	0.107	260.965	0.041	100.683	-0.622	58.792	84.
transpiration	1591	0.004	0.021	0.016	1.580	-2.222	-1.314	1.
VeraisonDOY								
FloweringDAWS	286	0.773	38.702	0.704	4.433	0.697	0.238	3.
LAIStrip	129	0.928	1.525	0.483	2.371	-0.240	1.186	1.
leaf.area.shoot	37	1.438	-0.042	0.875	0.061	0.421	0.024	0.
leaf.area.shoot.lateral	13	1.513	0.066	0.328	0.088	-42.246	0.083	0.
leaf.area.shoot.main	12	-0.398	0.224	0.133	0.043	-6.627	0.034	0.
leaf.area.vine	129	0.928	2.470	0.483	3.841	-0.240	1.921	2.
leaf.dw.shoot	56	1.093	-0.775	0.813	3.359	0.713	0.312	2.
leaf.dw.vine	22	1.083	-160.535	0.718	145.764	0.086	-101.887	107.
leaf.no.shoot	28	0.861	-0.845	0.770	7.201	0.598	-4.565	5.
leaf.no.shoot.main	93	1.236	-1.523	0.624	4.374	-0.163	1.760	2.
psw	789	0.107	260.965	0.041	100.683	-0.622	58.792	84.
shoot.dw.mean	115	0.713	4.267	0.703	4.548	0.696	0.689	3.
ta	4125	0.135	13.540	0.027	11.969	-0.487	-2.670	6.
total.NSC.concentration.root	41	0.631	0.041	0.370	0.031	0.181	-0.002	0.
total.NSC.concentration.trunk	41	0.533	0.042	0.380	0.040	0.070	-0.020	0.
TotalBerryDW	3367	0.689	150.641	0.772	336.525	0.673	-175.981	240.
TotalBerryFW	3467	0.000	0.000	NaN	5781.383	-5.179	-5292.886	5292.
transpiration	1591	0.004	0.021	0.016	1.580	-2.222	-1.314	1.
VeraisonDAWS	285	0.822	41.287	0.554	6.825	0.421	-0.547	5.
yield.per.vine	266	0.670	1.285	0.475	2.247	0.335	-0.673	1.

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.

Brown, Hamish E., Huth, Neil I., Holzworth, Dean P., Teixeira, Edmar I., Zyskowski, Rob F., Hargreaves, John N. G., Moot, Derrick J., 2014. Plant Modelling Framework: Software for building and running crop models on the APSIM platform. Environmental Modelling and Software 62, 385-398.

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.

Goudriaan, J, van Laar, H H, 1994. Modelling potential crop growth processes.

Holzworth, Dean P., Huth, Neil I., deVoil, Peter G., Zurcher, Eric J., Herrmann, Neville I., McLean, Greg, Chenu, Karine, van Oosterom, Erik J., Snow, Val, Murphy, Chris, Moore, Andrew D., Brown, Hamish, Whish, Jeremy P. M., Verrall, Shaun, Fainges, Justin, Bell, Lindsay W., Peake, Allan S., Poulton, Perry L., Hochman, Zvi, Thorburn, Peter J., Gaydon, Donald S., Dalgliesh, Neal P., Rodriguez, Daniel, Cox, Howard, Chapman, Scott, Doherty, Alastair, Teixeira, Edmar, Sharp, Joanna, Cichota, Rogerio, Vogeler, Iris, Li, Frank Y., Wang, Enli, Hammer, Graeme L., Robertson, Michael J., Dimes, John P., Whitbread, Anthony M., Hunt, James, van Rees, Harm, McClelland, Tim, Carberry, Peter S., Hargreaves, John N. G., MacLeod, Neil, McDonald, Cam, Harsdorf, Justin, Wedgwood, Sara, Keating, Brian A., 2014. APSIM – Evolution towards a new generation of agricultural systems simulation. Environmental Modelling and Software 62, 327-350.

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

Jones, C.A., Kiniry, J.R., Dyke, P.T., 1986. CERES-Maize: a simulation model of maize growth and development.

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

Keating, B. A., Carberry, P. S., Hammer, G. L., Probert, M. E., Robertson, M. J., Holzworth, D., Huth, N. I., Hargreaves, J. N. G., Meinke, H., Hochman, Z., McLean, G., Verburg, K., Snow, V., Dimes, J. P., Silburn, M., Wang, E., Brown, S., Bristow, K. L., Asseng, S., Chapman, S., McCown, R. L., Freebairn, D. M., Smith, C. J., 2003. An overview of APSIM, a model designed for farming systems simulation. European Journal of Agronomy 18 (3-4), 267-288.

Lawless, Conor, Semenov, MA, Jamieson, PD, 2005. A wheat canopy model linking leaf area and phenology. European Journal of Agronomy 22 (1), 19-32.

McCown, R. L., Hammer, G. L., Hargreaves, J. N. G., Holzworth, D., Huth, N. I., 1995. APSIM: an agricultural production system simulation model for operational research. Mathematics and Computers in Simulation 39 (3-4), 225-231.

McCown, R. L., Hammer, G. L., Hargreaves, J. N. G., Holzworth, D. P., Freebairn, D. M., 1996. APSIM: a Novel Software System for Model Development, Model Testing and Simulation in Agricultural Systems Research. Agricultural Systems 50 (3), 255-271.

Monteith, J. L., Moss, C. J., 1977. Climate and the Efficiency of Crop Production in Britain [and Discussion]. Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences 281 (980), 277-294.

Reyenga, P.J., Howden, S. M., Meinke, H., McKeon, G.M., 1999. Modelling global change impacts on wheat cropping in south-east Queensland, Australia. Environmental Modelling & Software 14, 297-306.

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.

Wang, Enli, Engel, Thomas, 1998. Simulation of phenological development of wheat crops. Agricultural Systems 58 (1), 1-24.