Model description

model's version v-06 (current version)

David Nortes Martinez, G-EAU research unit. IRSTEA

February 8, 2018

Contents

The following description of the model is based on the precepts of the ODD protocol for describing individual and agent-based models (Grimm et al. [2006]). Nonetheless the specifications of the model's architecture have required a few adaptations of such protocol.

The present document is organized in 9 sections, and attempts to provide all the information needed to ensure both full comprehension of the model and its replicability. Sections 1 and 2 are dedicated to state the main goals pursued with the model and explain how it is conceived.

Sections 3 to 5 are used to describe thoroughly elements in the model, agents, associations between agents, dynamics and expected consequences of floods. Together with sections 1 and 2, they should be enough to understand how the model works. Section 6 completes the exposition with an overview of impact calculation, indicators built to recover the proper information, and different scales of measure in our model.

Section 7 reviews the model general architecture and each one of the main procedures present. This section is a *must-read* for all of those who want to understand as well how the model works at code level, and/or recode/extend it.

Section 8 covers all the information relative to calibration and hypothesis. The model description finishes with section 9, dedicated to inform of the concrete numerical values provided to the model in the set-up. We expect it to allow researchers to be able to replicate our experiments if wished, as well as feed discussions about the convenience/realism of concrete values.

Different annexes have been also included to ensure the comprehension of the model. First, the reader counts on a glossary of terms, so definitions and meaning rest doubtless. As well, readers not familiar with flowcharting, have available a "legend" of the symbols, thus the task of understanding and follow flowcharts along the text becomes easier.

Last, but not least, to favor the comprehension of the text, we will use the next **convention** when referring to terms:

- Variables, processes, functions and code in general, when part of the text, will be written in `teletypefont`. When summary tables of variables are shown, such variables are written in standard font.
- Words highlighted in blue are references to glossary entries.
- As a special case, when R is written as R, it refers to the programming language in a general way, while when written as `R`, it refers to the environment in which procedures, variables, functions or processes exist. Likewise for Netlogo.
- Numbers highlighted in blue are references to sections, footnotes, figures and/or tables in the text.

1 Purpose

The main objective of this model is to simulate the propagation of flood impacts through an economic productive system.

To do so, on one hand, it simulates a productive system characterized as a star-type network [Tesfatsion, 2006] —figure 1—, where all elements in the system are connected one to each other through a central element. Such kind of organization can be found in the cooperative productive systems. There, all small producers are linked to each other through the cooperative, mutualizing productive means, costs, risk and benefits.

On the other hand, the model includes a flood simulator that exposes the system to floods of different extent. It provide us with data about the disruptions, damages and consequences —direct, indirect, immediate, delayed, intuitive, non-intuitive— caused by those natural disasters on the normal performance of each of the elements in the system, and the own system as a whole.

To illustrate a case of a star-type productive network, we have chosen a case study of vine-growers grouped in a cooperative winery, nourishing it with data from both the Aude and Var regions (southern France).

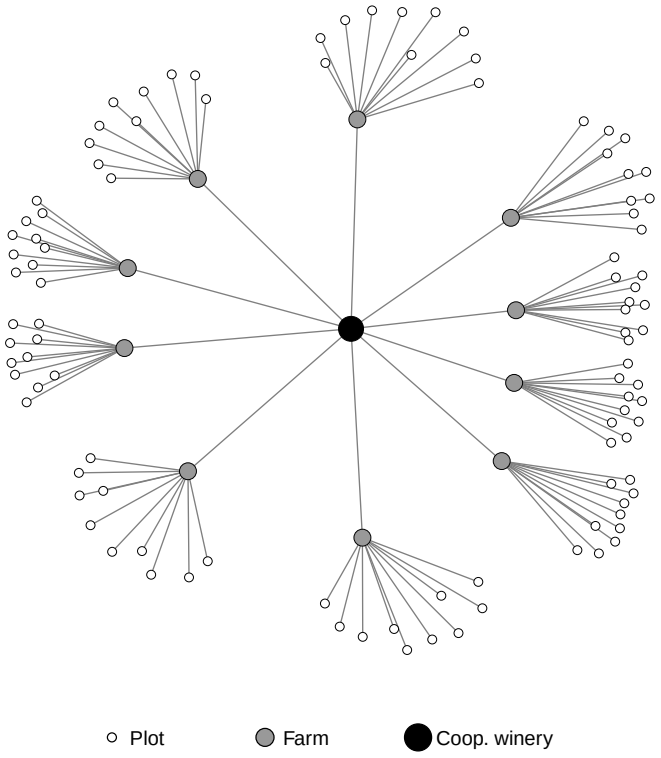


Figure 1: representation of a star-type network, based on a cooperative winery

2 Model conceptualization and design concepts

2.1 Conceptualization

The model is illustrated by a collection of vine-growers linked to each other through a cooperative winery. Although elements in the model will be properly defined in section 3, we provide here a rough description.

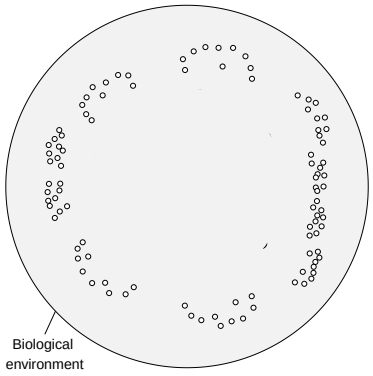
The model is conceptualized as an interaction of two different environments —biophysical and productive-economic (figure 2)— through a productive system composed by three main elements —vineyards (also referred as plots hereafter), vine-growing farms and cooperative winery/ies.

The biophysical environment (figure 2a) is responsible for plant cycles, soil basic productivity and yield availability at plots' level, as well as for floods. The productive-economic environment (figure 2b) uses the referred yield as its basic input and deals with the social, productive and economic functioning. The consequences of the floods in this productive economic system is the result of the interaction between the two environments.

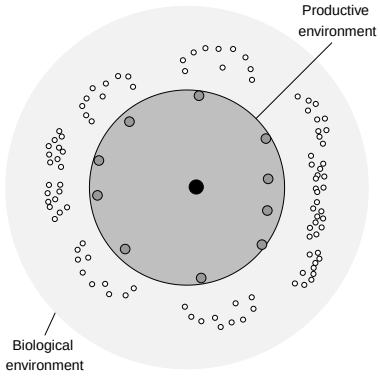
This way, connections between elements in the system (figure 2c) ensure not only the interaction between those same elements, but also the interaction of both environments at different levels and time spans.

At the same time, dealing with floods implies refers to the notion of exposure: which of the elements in the model can be directly impacted by a flood? furthermore, by which flood extent?

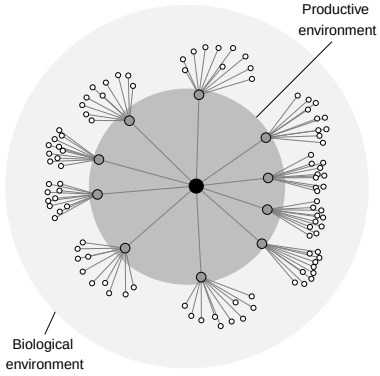
As a result, our model is geolocated. Presence of elements and, if information is available, proportions of elements —over their respective totals— are represented on



(a) Biophysical environment



(b) Productive environment



(c) Resulting bio-productive environment

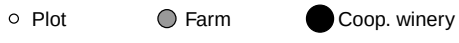


Figure 2: Environments

the terrain as close as possible to real cases. An example of geolocation of the star-type network in figure 2c, distinguishing between prone and non-prone areas is shown in figure 3.

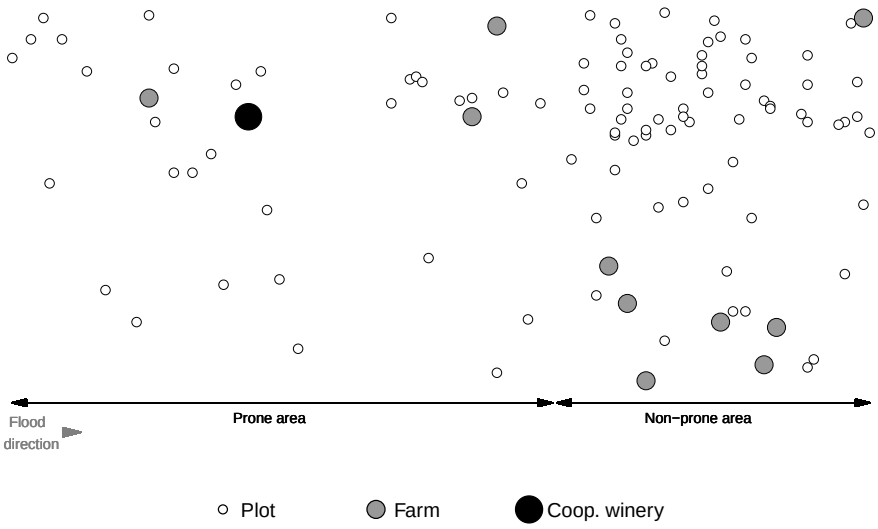


Figure 3: Example of geolocation of a star-type network in the model

2.2 Design concepts

Rationality of agents: Agents are assumed to follow a routine which is considered as their optimal production. Thus the incentive is to keep their productive means *in statu quo ante* and they try to recover as soon as possible after flooding.

It means that, when in a Bussines as Usual scenario or Zero Flood Scenario (BAU), agents will not have any motivation to change their investment-reinvestment-production pattern, whereas when in a Simulated Flood Scenario (SFS), if them or any of their belongings are flooded, their choice in the aftermath of the flood is to reverse to the initial state as soon as possible and minimize the losses.

Emergence: relation between defining features of agents in the system and disruptions, damages and consequences —direct, indirect, induced and/or immediate— of floods, both at individual and system's' level.

Adaptation: when agents are flooded, they can choose between two possible coping tactics: to perform the tasks assuming losses due to lack of means (direct impacts reduce the agent's coping capacity) or to outsource the tasks to be performed during the season they are flooded with an extra cost.

Notwithstanding, their autonomy to choose their coping strategy in SFS has been limited in this version of the model. As a consequence, the respond gets homogenized and the effects of choosing one or another tactic can be compared.

Sensing: sensing capabilities are different depending on each element. Plots sense their own state through 3 key variables: i) `flooded/not flooded`; ii) if so flooded, `destroyed/not destroyed`; and iii) if so flooded and not destroyed, `proportion of harvest lost`.

Wineries will perceive their state through `flooded/not flooded`. It will allow them to start reparations to preserve the *status quo*, and determine whether they are able to perform their tasks. As well, the winery "sense"" which of the farms, and in what amount, has provided it with input for production.

Farms, together with plots and wineries, sense their state through `flooded/not flooded` variable. In this version of the model, when `flooded`, it triggers the need for action: immediate reparation and adoption of coping tactic. Additionally, each farm receive information of the state of its plots —and only its plots; the state of the neighbor's plots cannot be perceive— and of the state of the winery, and its ability to perform tasks.

Interaction: different kinds of interactions can be assumed:

- Among environments: interaction of a productive and a biophysical environments, as already explained in subsection 2.1.
- Among agents:
  - A so-called *direct* interaction: interaction of farms with their plots, and farms with the cooperative winery, following the production links
  - A so-called *indirect*: interaction of farms with farms through the cooperative winery. It is reflected by the fluctuations of costs and revenues from the winery

Stochascity: Flood damages depend on a large amount of factors, which explanation and influence are not among the goals of this model. Thus, to model the consequences over plants depending on a multitude of elements that are unknown inside the model, we have chosen to simulate plant destruction at plot's level through random processes.

Collectives: each star-type network is considered a collective. In the model several collectives can coexists at the same time, and their definition comes preset in the setup of the model.

This version of the model does not include any mechanism of network evolution through time.

Observation: data to be collected is focused on 4 key aspects: production, revenues, costs, and investments (further information in section 6) Such data is collected at agent's level, once every four time steps —or `ticks` in Netlogo terminology—, coinciding with the autumn season, on both BAUs and SFSs. Comparisons between both allow us to analyze the evolution in time and magnitude of the impacts of floods.

3 Entities, state variables, and scales

Two main agents operate and interact in the model along time: farms and cooperative wineries. (Figure 4).

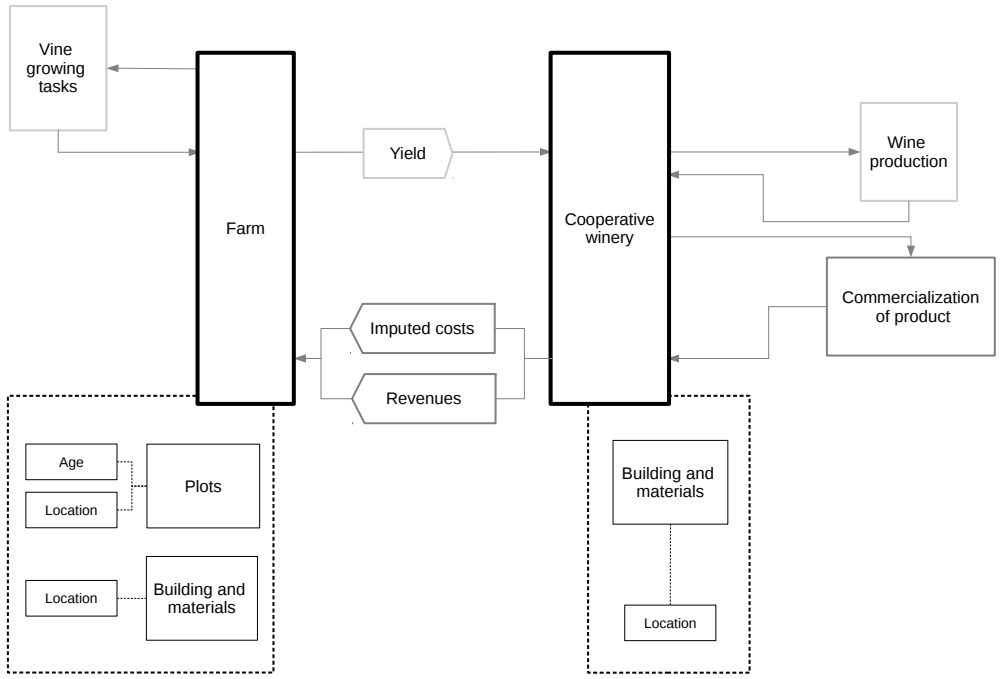


Figure 4: Main elements in model

Farms perform vine-growing tasks over the amount of plots owned, providing, this way, the main productive input to the system. They are considered as the union of two different elements: buildings and plots.

Buildings are considered the core of the farms. They determine where the farm is physically located, thus the level of exposure to floods. Additionally, when in SFS, its state —`flooded/not flooded`— will determine the farm's capacity to perform its inherent duties.

Each plot is defined by: `location` —which establish the `distance to the river`, therefore the exposure to floods— and `age` —which determines whether the plot is productive or not, as well as the investment's lifetime— of the plants. Together with `extent`, they determine the plot's yield in harvesting season. Furthermore, plots are kept with different ages, which has three different consequences: one, there is rotation in crops; two, the production is variable and lower than the potential; three, agents have heterogeneous productions.

As well as it occurs with farm's buildings, in the SFS, its state —`flooded/not flooded; destroyed/not destroyed`— will contribute to determine the amount of yield available

in each one.

Each farm is associated with one, and only one, cooperative winery. These wineries, once they receive the yield from their associated farms, produce the wine and commercialize it in the markets, sharing both revenues and cost with their associates. In the same way it happens in the case of farms, the location of its building over the terrain determines the level of exposure to floods. Again, in SFS, its state —`flooded/not flooded`— will determine the winery's capacity to perform its tasks.

In addition, both wineries and farms have assigned a determined `size`. In case of farms it comes given by the number of plots they own, whilst in the case of wineries it comes given by the sum of maximum potential production of their associate farms. That `size` is used to calculate the initial value of the `structural cost` inside each agent's `cost structure` (see section 9)

The time step has been set to one season. This way, each time step, or `tick` in Netlogo terminology, represents a quarter of a year. Thus each year corresponds to 4 time steps or `ticks`, starting always in winter. Simulations are run over 30 years to take into account damage propagation in time.

4 Process overview and scheduling

Each agent presented in section 3 disposes of its own schedule. For both farm and winery, we count on simplified —and seasonally adjusted— versions of their own real-life complex schedules linked to biological cycles of plants (more details will be given in subsection 8.2). As a result, the global internal schedule in the model is given by the coexistence and interaction of those individual schedules. To illustrate the point, figure 5 outlines the global model schedule and each agent's own schedule when no flood hit the system (BAU scenario). A year begins in winter and ends in autumn.

Assuming we are in year $t = 1$, The dynamic goes as follows:

1. Vine-growing tasks are done over plots during the four seasons (table 1). Such tasks have been translated to hours of labor, then split among seasons following Bremond [2011]

	Winter	Spring	Summer	Autumn	total
Total (hours)	49	18	31.5	10.5	109
Proportion over total	0.45	0.16	0.29	0.1	1

Table 1: Seasonal attribution of vine-growing tasks based on citeBremond2011

2. In winter, the cooperative winery produces wine with the yield obtained from the farms in $t = 0$
3. In spring, the cooperative winery commercializes the wine produced during winter with the yield obtained in $t = 0$

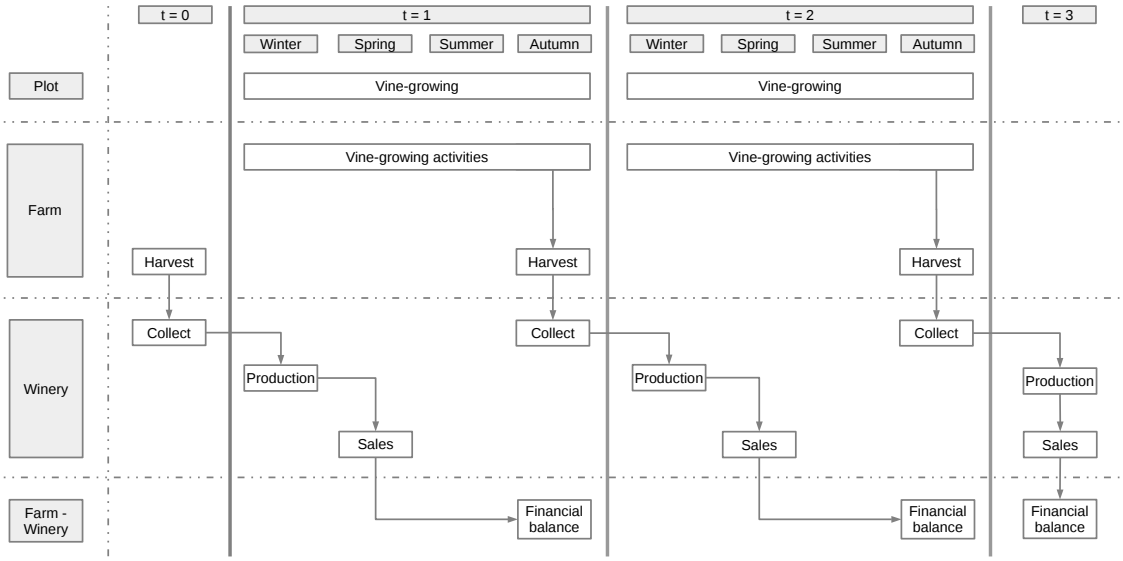


Figure 5: Process overview and schedule in the BAU scenario

4. In spring, once everything is sold, the cooperative winery splits both revenue and cost among farms proportionally to their yield in $t = 0$.
5. In autumn, the farms harvest their plots again
6. In autumn, the cooperative winery collects the yield from the farms.
7. In autumn, both farms and winery make their financial balances. Farm's financial balance includes vine-growing costs of $t = 1$ and revenues of $t = 0$ (cost and revenues are delayed one year). At winery' level this financial balance is done counts on the revenues and wine-making cost of $t = 1$ over input collected in $t = 0$
8. At the end of autumn, plots reaching `age` $= 30$ get replanted and rest unproductive for 5 years (20 time steps or `ticks`). Agents always choose to replant the plot at the end of each plot's investment lifetime (`age` $= 30$), and renew the vineyards.
9. In winter of $t = 2$, the cooperative winery produces wine with the yield obtained from the farms in $t = 1$
10. In spring of $t = 2$, the cooperative winery commercializes the wine produced during winter with the yield obtained in $t = 1$
11. In spring of $t = 2$, once everything is sold, the cooperative winery splits both revenue and cost among farms proportionally to their yield in $t = 1$. Both farms and winery make their financial balances of $t = 1$, where farms' financial balance includes vine-growing costs of $t = 1$

    To split cost and revenues, the cooperative winery proceeds in a proportional way [Biarnés and Touzard, 2003]:

$$TC_i = \left(\frac{F+V}{\sum_{i=1}^{n} q_i} q_i \right) \quad (i = 1, 2...n) \tag{1}$$

$$B_i^o = pq_i - TC_i = pq_i - \left(\frac{F+V}{\sum_{i=1}^{n} q_i} q_i \right) \quad (i = 1, 2...n) \tag{2}$$

Where:

- TC_i is the share of the wine-making cost in the winery for the farm i
- B_i^o is the share of the benefit in the winery for the farm i
- pq_i is the share of revenue of the farm i.
- $\frac{F+V}{\sum_{i=1}^{n} q_i} q_i$ is the decomposed wine-making cost in the winery for the farm i
  - F is the structural wine-making cost
  - V is the operational wine-making costs
  - $\sum_{i=1}^{n} q_i$ is the total production in the cooperative winery, as a sum of the individual productions of the associted farms.
  - q_i is the production of the farm i.

5 Submodels: flood impacts

5.1 Floods

In our model floods are programmed to cover a variable extent of a predefined *potential maximum prone area* (see figure 6) during a given season.

Regarding the time span, two remarks are worth mention at this point: on the one hand, floods hit the system once per season. The model is not ready to simulate two, or more, flood events during the same season. On the other hand, as a convention, we assume floods hit the system at the beginning of the season. Such hypothesis, far from trivial, has consequences on damages, cost variations, etc.

Regarding flood extent, our formulation keeps the flood's y coordinate constant, and equal to the maximum value of y ($y = y_{max}$), while the x coordinate varies in the interval $[0, 100]$. This way, the area covered by floods comes expressed by the function $f(x) = xy_{max}, x \in [0, 100]$. That formulation allows us to liken the value of the flood extent's 'x coordinate with the percentage of the maximum prone area flooded. As well, it simplifies the identification of flooded elements: every entity —plot, farm and cooperative winery— will declare itself `flooded`, always its location is included inside the area covered by the flood. However they will only need to use its x coordinate as reference: when the x coordinate of the entity is less or equal than the x coordinate of the flood extent, entities declare themselves flooded.

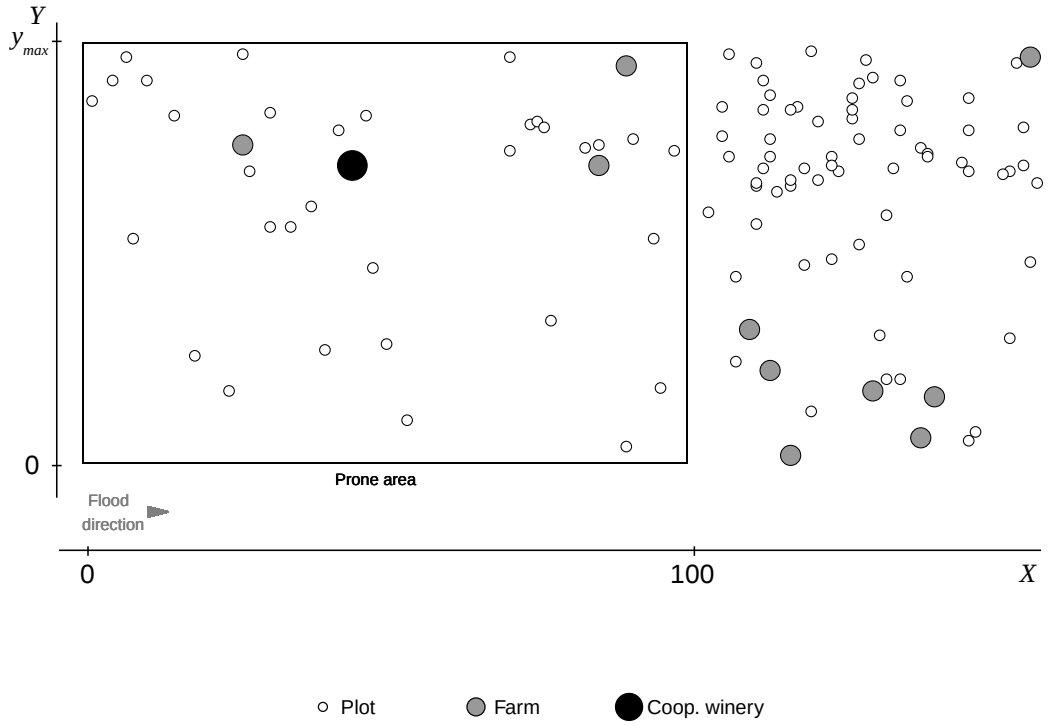


Figure 6: Detail of coordinate axes in the geolocated representation of the start-type network

When a flood hits the system, entities declared `flooded` will register and declare consequences depending on their own damage functions. Thus, it is foreseeable that the normal performance, described in section 4, gets disrupted by those same impacts. Additionally, we expect non-intuitive effects to emerge from the interaction of the different entities and schedules.

The next two sections will explain in detail both each entity's damage functions, and the consequences over the system dynamics.

5.2 Farm's damage function and system dynamics

As stated in section 3, farm units are considered the union of two different elements: plots and farm's buildings and materials. For pedagogical purposes we are going to analyze separately each element's damage functions and consequences for the system's dynamics.

5.2.1 Plot's damage function and system dynamics

Damage function. The damage function at plot's level presents the seasonal behavior detailed in table 2. As we can see, each time a plot is hit by a flood, effects are threefold:

1. The probability that plants result destroyed differs from one season to another:
   - Winter: $p = 0$
   - Spring: $p = 0.5$
   - Summer: $p = 0.2$
   - Autumn: $p = 0.1$
2. The proportion of harvest lost will depend on the season as well, but also on plant destruction:
   - On plots where plants are not destroyed
     - Winter: no loses
     - Spring: 50% of the plot's harvest is lost.
     - Summer and Autumn: the plot loses all its available harvest
   - On plots where plants are destroyed
     - Winter: no losses
     - Spring, Summer and Autumn: the plot loses all its available harvest
3. Soil-conditioning should always be performed after a flood

	Winter	Spring	Summer	Autumn
Probability of plant destruction	$p = 0$	$p = 0.5$	$p = 0.2$	$p = 0.1$
Harvest destroyed if plants not destroyed (%)	0	50	100	100
Harvest destroyed if plants destroyed (%)	0	100	100	100
Soil	reconditioning	reconditioning	reconditioning	reconditioning

Table 2: Plot's damage function

Effects. Regarding system's dynamics, we consider necessary to distinguish the combo spring-summer- autumn (figure 7) from winter (figure 8). As it is shown in figure 7, when the flood hit a plot —let's assume in $t = 1$— two potential situations are possible: i) plants are not destroyed; ii) plants are destroyed.

In the first case, at plot's level, plants keep their integrity but the harvest is lost according the seasonal proportion. At farm's level, all plots impacted owned by the farm

will need soil reconditioning. The yield harvested will depend on the number of plots flooded. At the same time, plots whose yield is completely lost, save vine-growing cost to the farm, due to the fact that tasks not essential for the plant survival are not performed by the farm[1]. At winery's level, as it happens at farm's level, the yield collection will be affected by the number of plots hit owned by the winery's associates, and so will be the annual production and the sales. Ultimately the financial balances of the winery and the farms will reflect the impacts of the flood.

The second case have further ramifications: at plot's level, plants are destroyed, ergo all harvest is lost. At farm's level, impacts in the aftermath of the flood will be of the same nature but different magnitude. However, plant destruction introduces a longer term effect: destroyed plots need to be replanted. Assuming they replanted immediately (next winter), as told in sections 3 and 4, they will need 5 complete years to be considered productive. Therefore, *ceteris paribus*, farm's yield will reflect the impact of the flood during 5 more years. At winery's level, those longer term impacts will be reflected too.

By time spans, damages in soils and harvest will become part of impacts in $t = 1$, as well as variations in vine-growing costs. Variations in production (ergo in revenues and wine-making costs), always plants are not destroyed, will be delayed one year ($t = 2$); if plants are destroyed, they will last until $t = 7$, assuming plots are replanted in $t = 2$.

Winter (figure 8) is an special case. Damage functions limit losses in winter, when plots are hit directly, to soil-reconditioning. It provokes a direct financial impact over farms who own impacted plots (benefits will decrease as a consequences of the extra reconditioning cost), but not further damages over yield, thus production, thus revenues, will take place.

5.2.2 Farm's buildings damage function and system dynamics

Damage function. Table 3 details the damage function for farm's building in the system. It can be split into two kind of consequences: consequences due to buildings and materials flooded, and, once it happens, consequences due to the coping strategy chosen.

Farm's choices and actions. As said in section 2.2, agents (therefore farms) are assumed to be in their optimal production point, thus motivated to preserve their *statu quo*. It means, in absence of constraints, buildings will be repaired and materials substituted right away, so the farm is fully operational next season[2]. Same principle applies to plot's replant: in absence of constraints, it is done first winter season following the flood. But when the building is hit, we assume that part of the vine-growing material is lost/hit. Farms, consequently, will have to pay for reparations and , additionally, they cannot fully perform their seasonal tasks. To cope with the situation, they can choose between two strategies:

[1] Since floods happen at the beginning of the season, plots whose yield is destroyed will not be attended. Thus no vine-growing cost will be paid for them until the next campaign

[2] After the flood hits the farm in the beginning of the season, we assume that, in absence of financial constraints, farms have enough time during the season to repair and be fully operational next one

- Outsourcing: the farm pays external service providers to perform the task in its place. Such strategy saves all the yield in plots since the tasks are fully performed, but increases the seasonal vine-growing costs 80%
- Insourcing: the farm counts on its own resources to perform the seasonal tasks. Since part of the material is lost, we assume the farm can only perform the half of the tasks planned for the season. As a consequence, seasonal vine-growing cost decreases 50% but there is an associated lost in yield.

  For an explanation on the origin of the values, see section 8

			Winter	Spring	Summer	Autumn
Material damage			Building and material all seasons			
Performance	Outsourcing tactic	Δ vine-growing cost	+80	+80	+80	+80
		Yield lost	none	none	none	none
	Insourcing tactic	Δ vine-growing cost	-50	-50	-50	-50
		Yield lost	36.5	18.5	21.5	50

Unit: Percentage (%)

Table 3: Farm's damage function

Effects. Impacts on the system dynamics are outlined in figures 9 and 10. Figure 9 illustrates the process already described: if the farm's building is impacted —x coordinate of building $\leq x$ coordinate of flood—, we assume material damages that will have consequences over the farm's performance, forcing it to choose a coping tactic.

If the coping tactic chosen is *outsourcing* here will not be effects over yield, only over the season's vine-growing cost. On the contrary, if the farm decides to go *insourcing* both vine-growing costs and yield will be impacted. The time span for both impacts is different though: assuming the flood hits the system in year $t = 1$, effects over vine-growing costs become part of impacts in $t = 1$, while effects over yield will be felt in year $t = 2$, once the yield is processed, turned into wine and sold.

Eventually financial balances get affected, but, while the *outsourcing* tactic limits impacts to the year in which flood hits the system, the *insourcing* one generates more persistent impacts.

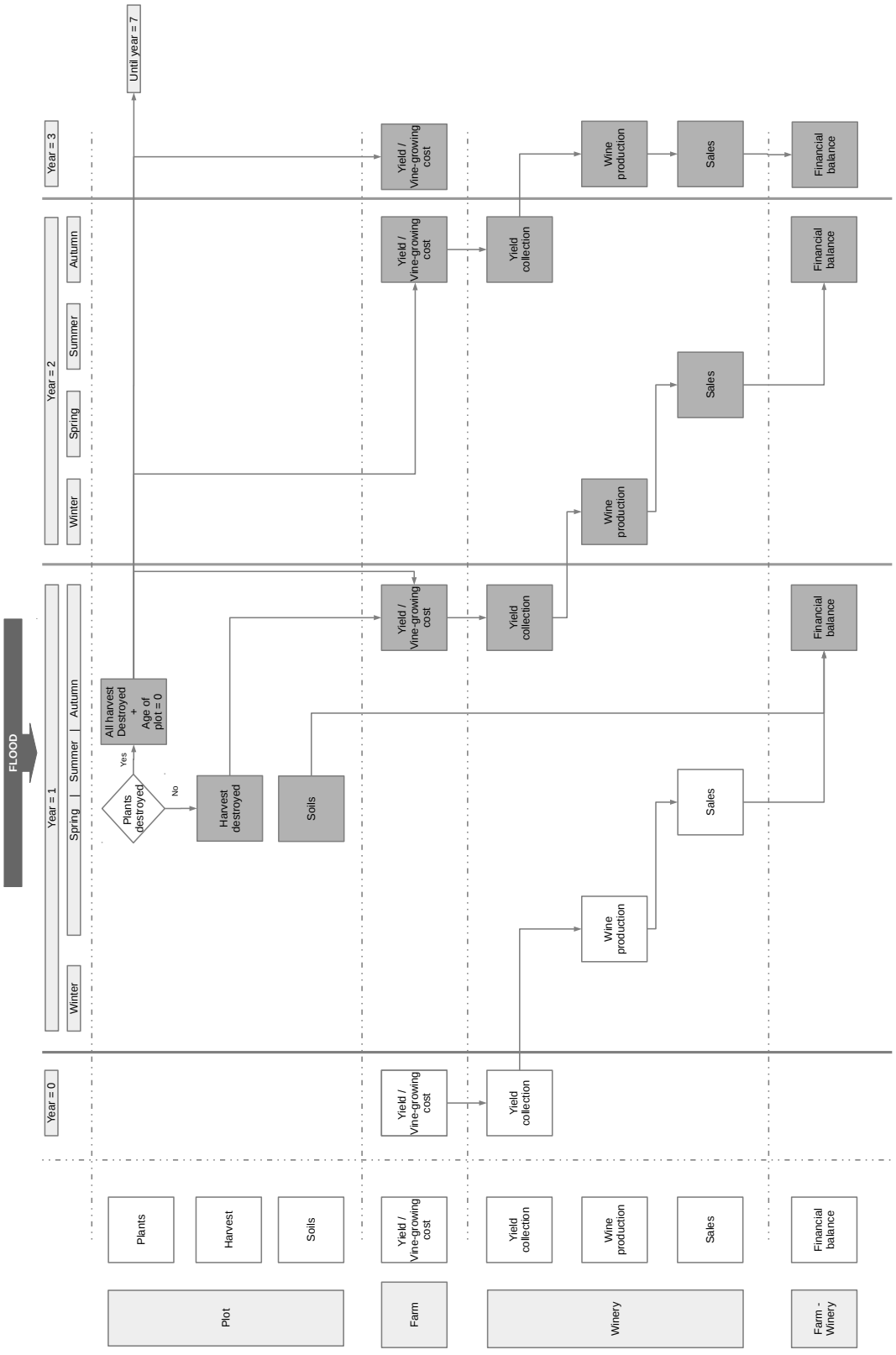


Figure 7: Consequences of a flood over a plot in the system's dynamic. Season spring to autumn

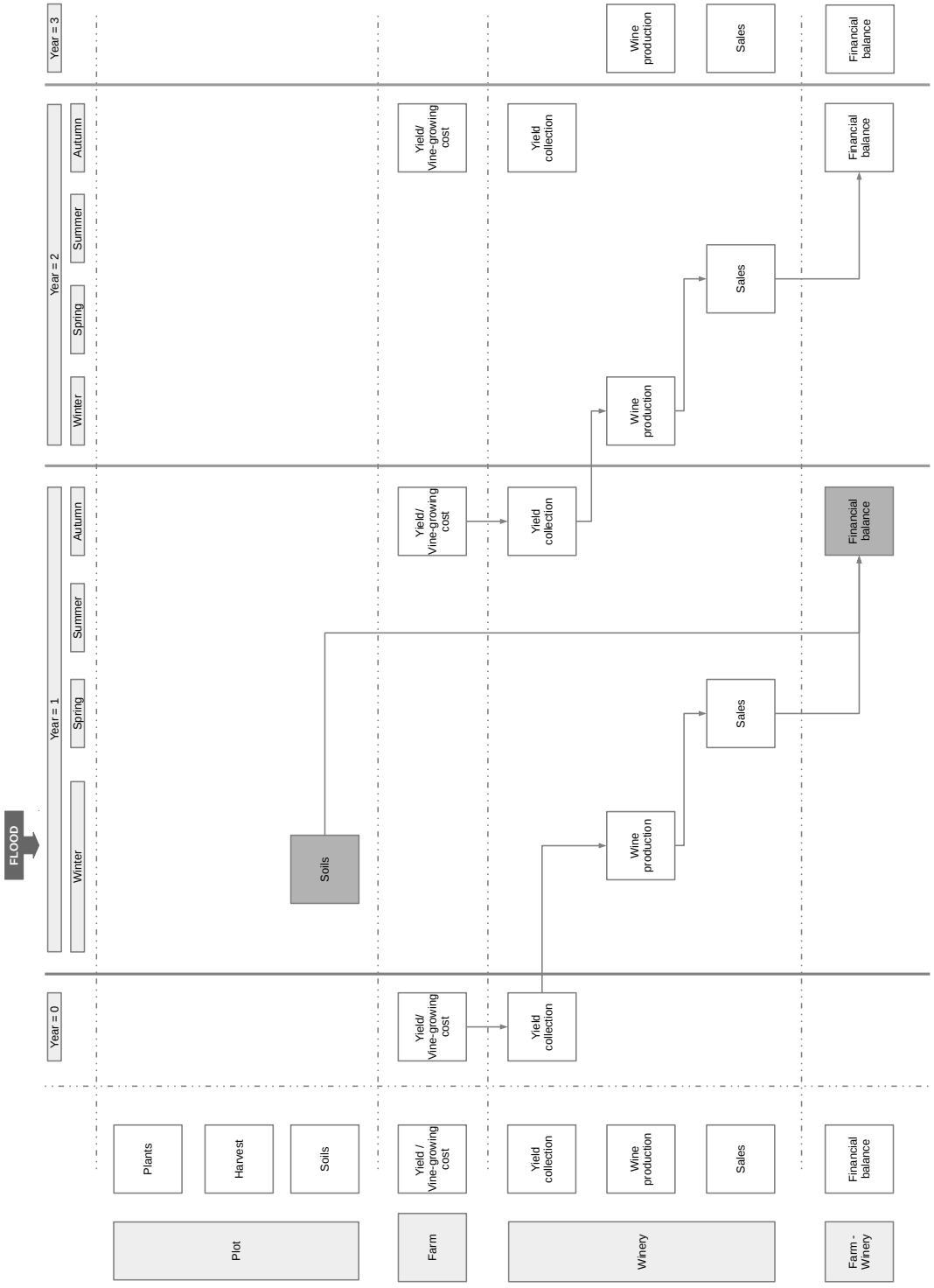


Figure 8: Consequences of a flood over a plot for the system's dynamic. Special case of winter

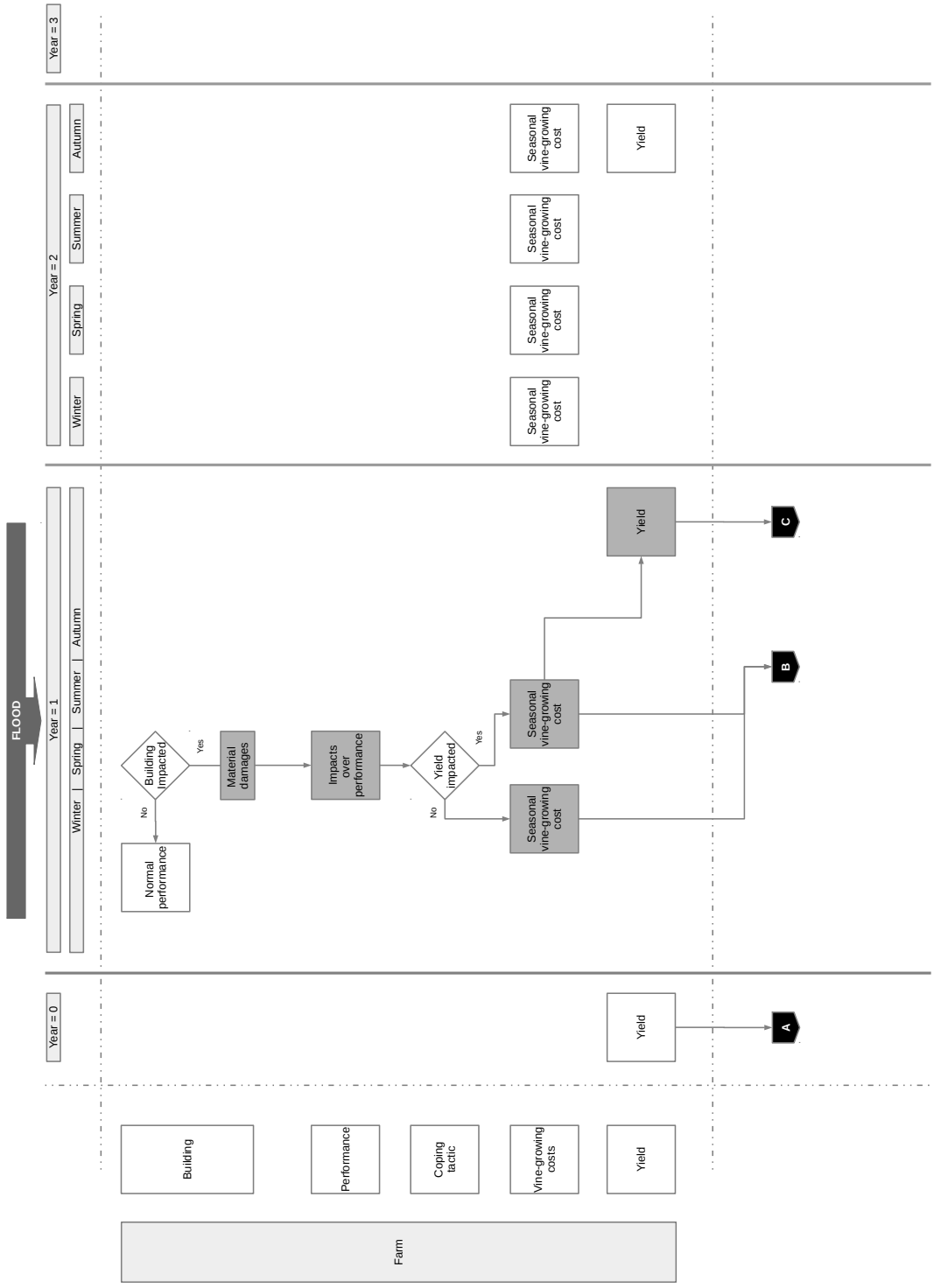


Figure 9: Consequences of a flood over a farm for the system's dynamic. All seasons

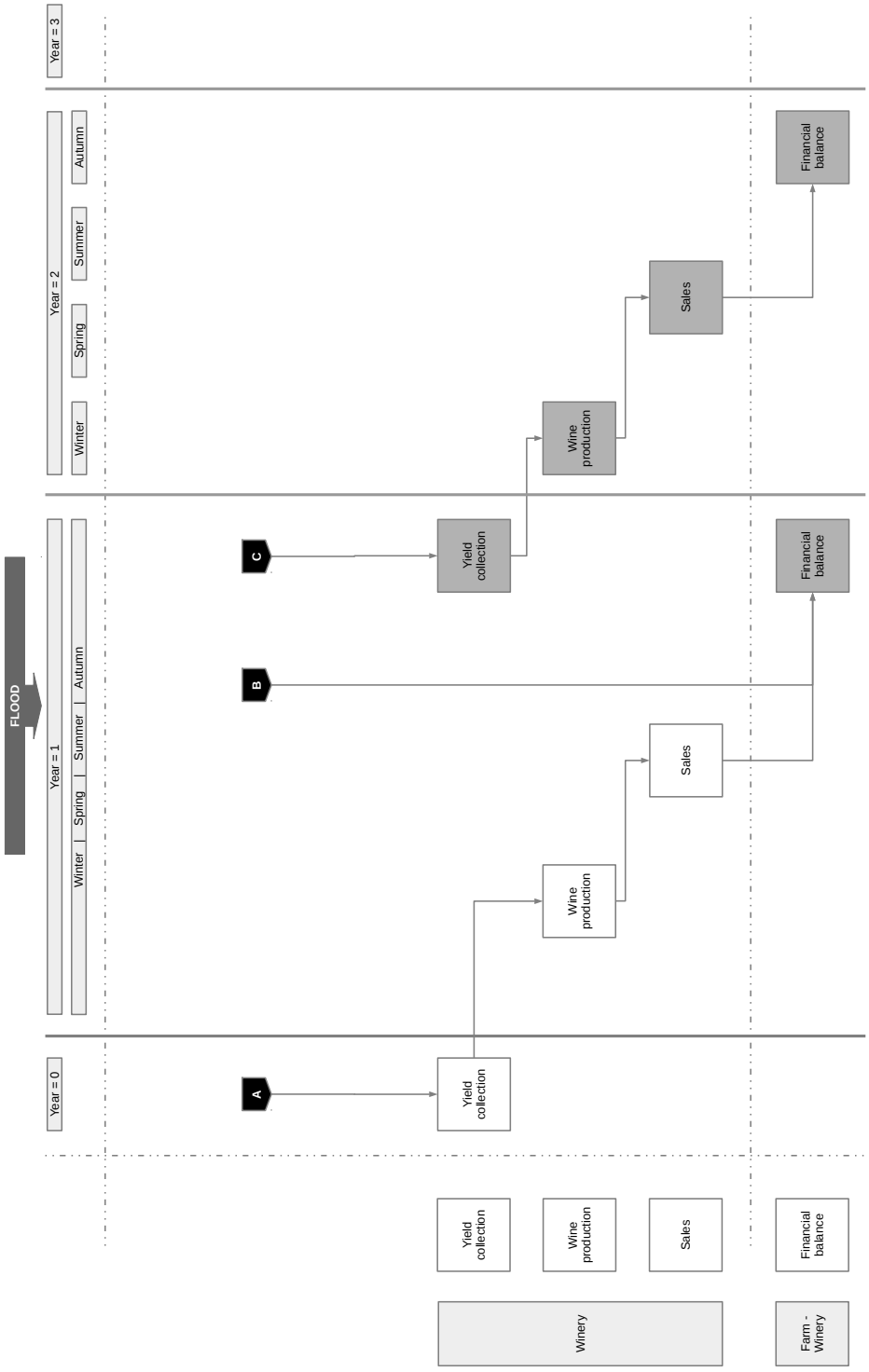


Figure 10: Consequences of a flood over a farm for the system's dynamic. All seasons (continuation)

5.3 Winery's damage function and system dynamics

Damage function. Table 4 displays the damage function for wineries in the system. As with farms, we can differentiate two sequential types of consequences:

- In spite of the season, when a cooperative winery is hit by a flood —x coordinate of building $\leq x$ coordinate of flood—, the model assumes buildings and materials flooded.
- Damages over buildings and materials affect winery's capacity to perform their assigned tasks. Therefore, depending on the season the flood hits the winery, in addition to material damages, the following consequences are assumed:
  - When the flood hits in winter, wine production cannot be accomplished.
  - If the flood takes place in spring, the production is lost and sales cannot be performed
  - Floods in autumn make impossible to collect the yield coming from its associated farms.

	Winter	Spring	Summer	Autumn
Material damage	Building and material all seasons			
Performance	No production	No sales		No collection

Table 4: Winery's damage function

Effects. Dynamics in the system get altered in different ways and time spans, depending on the season the winery is hit. Assuming flood occurs in $t = 1$, figures 11 to 14 display those alterations.

When the winery gets hit during winter, we assume the material damage suffered impedes the winery's normal performance. Therefore it will not be able to process the yield collected during $t = 0$ and produce the wine. As a consequence there will be no production to sell[3], thus no revenues nor wine-making cost, beside the structural cost.

Since all production and sales are done in and through the cooperative winery, all the associated farms will lose all production and revenues. They will be imputed, though, with their share of the structural cost and reparations. Eventually, financial balances will reflect such situation.

If the winery is caught by a flood in spring, we consider wine-making processes finished and production ready to be sold. However, material damages will make the winery lose

[3]Since floods happen at the beginning of the season, the winery will have time to fully functional for the next season, and to perform sales. However, to not be able to produce the wine, has left it with no production to be sold

the production and, as in winter, no revenues over the yield of $t = 0$ will be perceived. Contrary to winter, in spring, since wine-making activities are done, farms will be imputed with all the wine-making cost corresponding to its share plus the reparations needed.

During summer season, wineries are not expected to perform any essential task. Therefore, when they are flooded, impacts are "reduced" to reparations, with no further effect besides the ones over the financial balance of the winery and its associated farms.

Floods over the winery's buildings in autumn, hinders the winery from collecting the yield coming from its associated farms. Under such circumstances, all farms lose their yields, which prevents the system from having input to produce wine during winter of $t = 2$. Without production, effects are the same than the already described for winter, but delayed one period: no sales, ergo no revenues and wine-making cost reduced to the structural cost.

Agent's actions. As we said, when the winery's buildings are flooded, there is always an imputation of cost of reparation to each associated farm. According the disruptions described, we can differentiate two cases: the first one is when the winery is flooded, but production can be done or has been done. In such case, reparation costs are imputed among associated farms according the rule in equation 3

$$R_i = \left(\frac{R}{\sum_{i=1}^{n} q_i} q_i \right) \tag{3}$$

Where:

1. R_i is the reparation costs imputed to farm i
2. R is the total monetary value of reparations
3. $\sum_{i=1}^{n} q_i$ is the total production in the cooperative winery, as a sum of the individual productions of the member farms.
4. q_i is the production of the farm i.

The second case is when the production-commercialization process gets disrupted, and production cannot be done. In this case, wine-making cost is reduced to the winery's structural cost. Added to reparation costs, both are imputed according equation 4

$$CT_i = \frac{R + F}{N} \tag{4}$$

Where:

1. CT_i is the total cost imputed to farm i
2. F is the monetary value of the fixed vinification costs
3. R is the total monetary value of reparations
4. N is the number of farms members in the cooperative winery

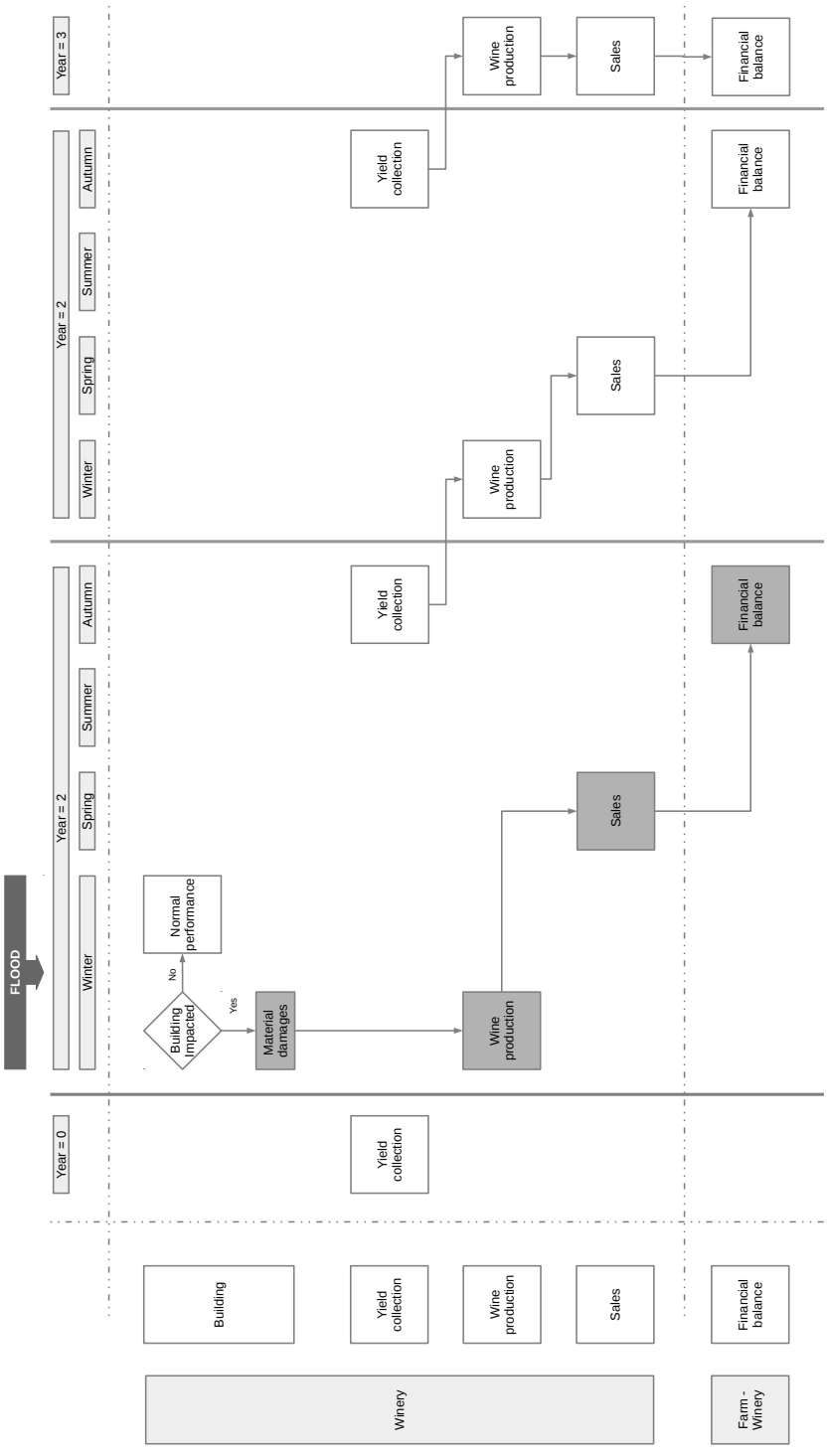


Figure 11: Consequences of a flood over a winery in winter in SFS

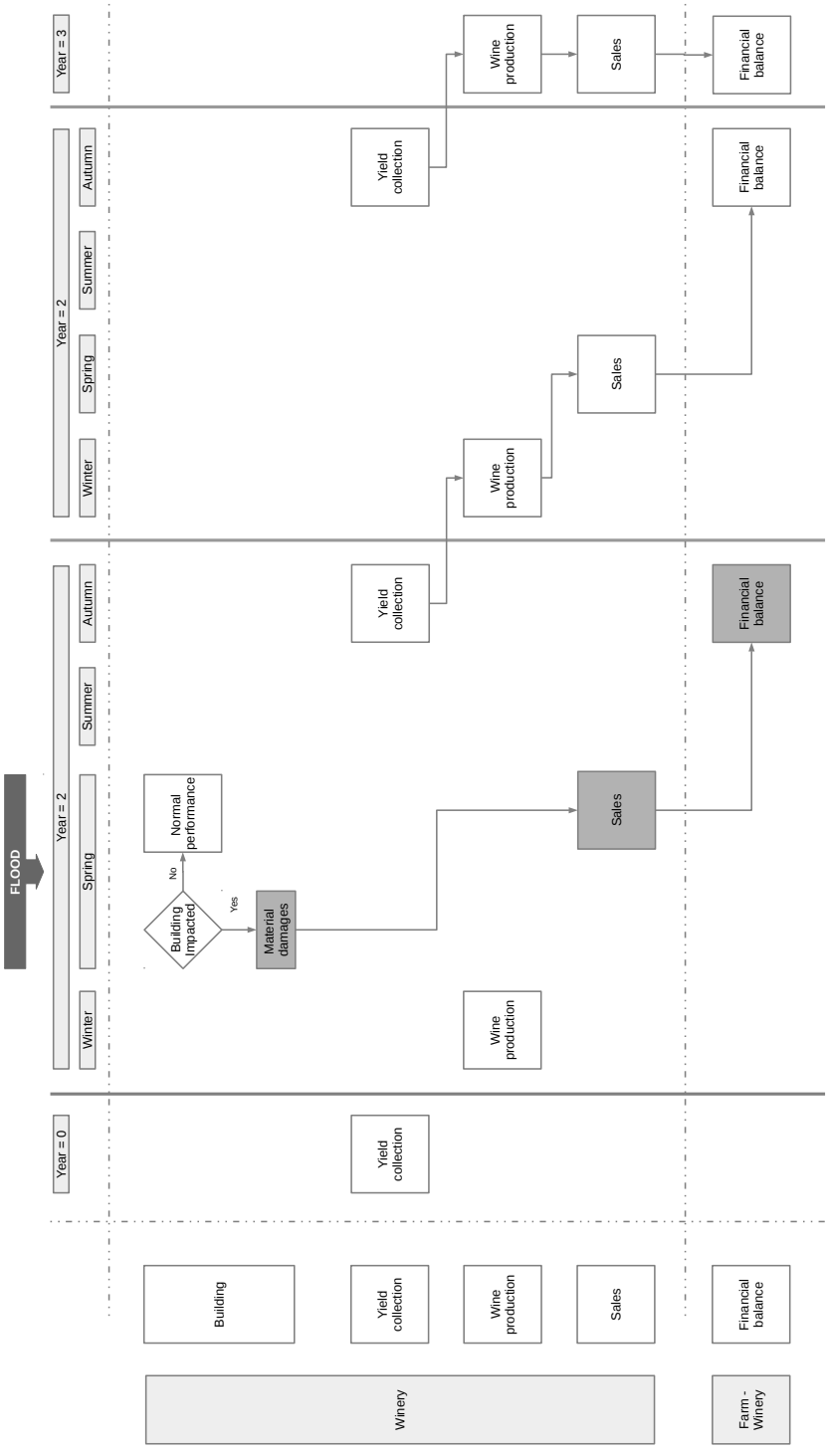


Figure 12: Consequences of a flood over a winery in spring in SFS

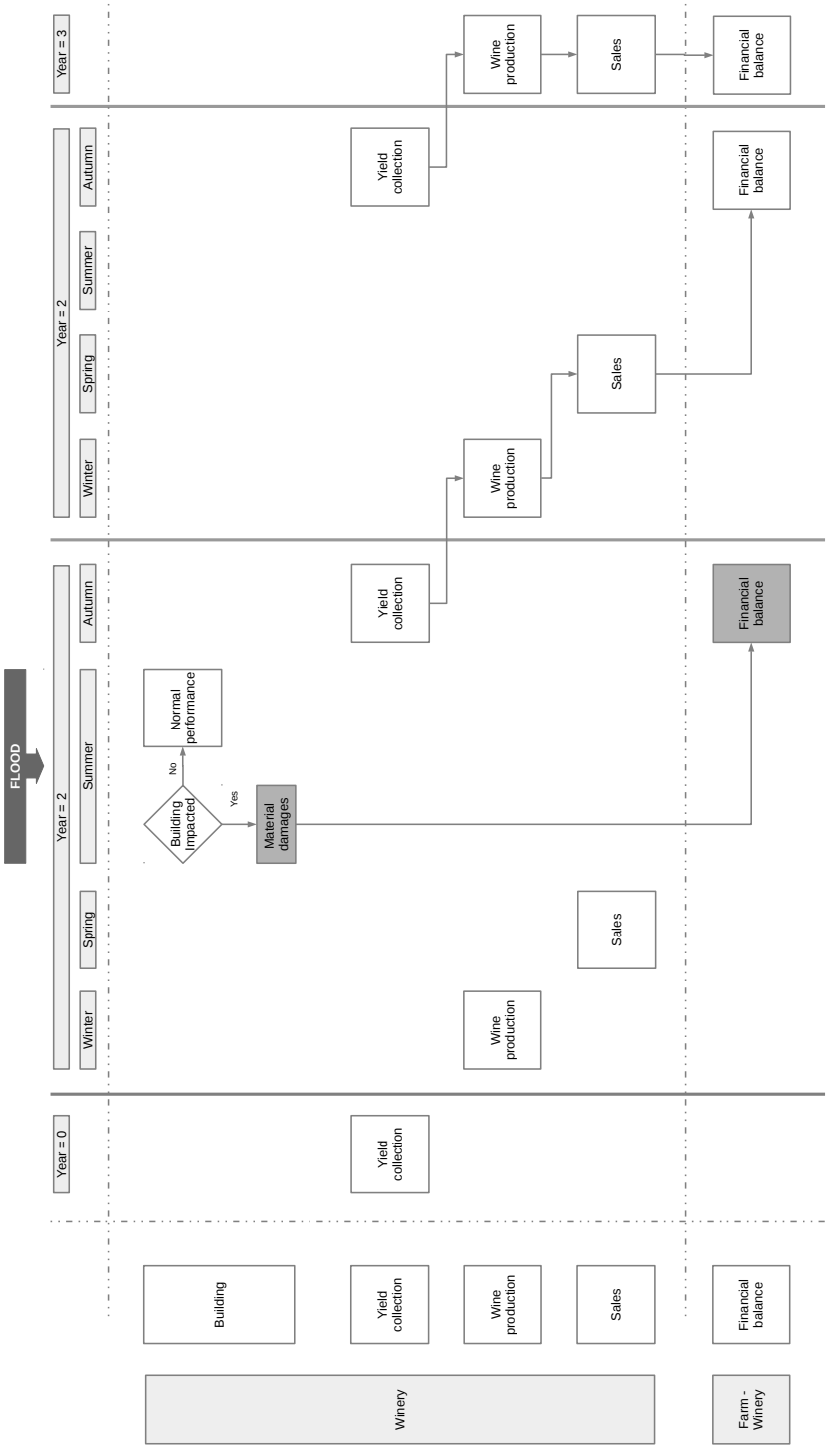


Figure 13: Consequences of a flood over a winery in summer in SFS

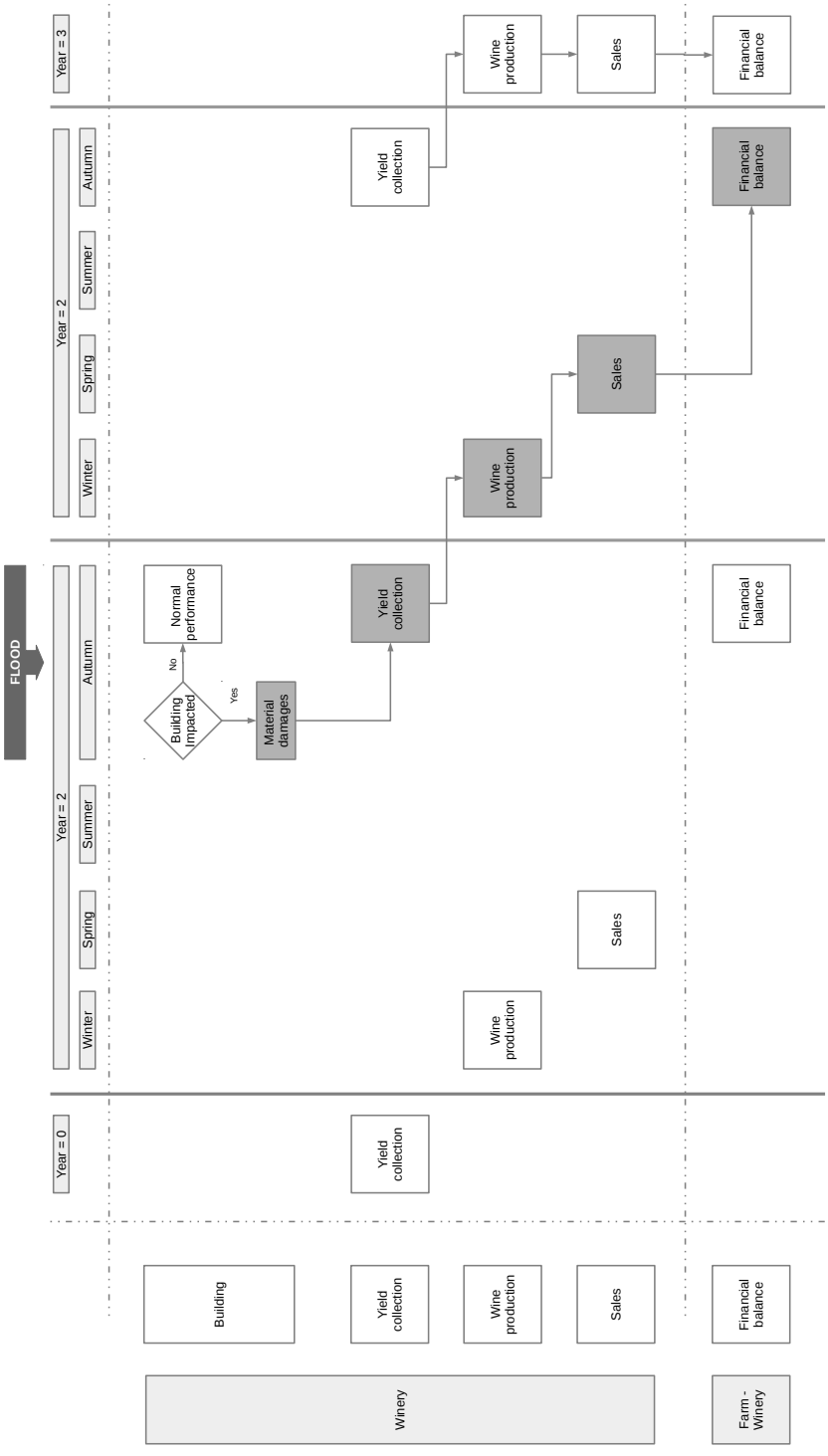


Figure 14: Consequences of a flood over a winery in autumn in SFS

5.4 Combining damage functions

Floods can affect at the same time cooperative wineries, farms and plots. It means that effects described in the prior sections can be summed. Notwithstanding, since in our network, impacts of floods over one entity have effects over every other entity, we have decided to introduce hierarchy levels over the impacts of floods. This way, problems related to double accountability can be avoided, and the the impact can always be scouted to its origin.

Figure 15 sketches out the hierarchy levels by entities, but before we can analyze it , we need to introduce new nomenclature and definitions.

For each productive plot γ_κ, owned by farm i, we can express its yield as

$$q_{i_T\kappa} = q_{i\kappa} + q_{i_D\kappa} \tag{5}$$

Where:

1. $q_{i_T\kappa}$ is the potential harvest in plot γ_κ of farm i
2. $q_{i\kappa}$ is the effective harvest in plot γ_κ of farm i
3. $q_{i_D\kappa}$ is the damaged harvest in plot γ_κ of farm i by the flood

The term $q_{i_D\kappa}$ "stores" the total of harvest damaged, whether its origin is in the direct submersion of the harvest or provoked by plant damages.

In our system, each farm i owns a number n_i of plots. Aggregating all those plots, each farm i owns a total extent Γ_i that can be expressed as:

$$\Gamma_i = \sum_{\kappa=1}^{n_i} \gamma_{i\kappa} \tag{6}$$

Using equation 6, we can express equation 5 at farm level as:

$$\sum_{\kappa=1}^{n_i} q_{i_T\kappa} = \sum_{\kappa=1}^{n_i} q_{i\kappa} + \sum_{\kappa=1}^{n_i} q_{i_D\kappa} \tag{7}$$

Where:

1. $\sum_{\kappa=1}^{n_i} q_{i_T\kappa}$ is the potential yield of farm i
2. $\sum_{\kappa=1}^{n_i} q_{i\kappa}$ is the effective yield of farm i
3. $\sum_{\kappa=1}^{n_i} q_{i_D\kappa}$ is the damaged yield of farm i

And the term $\sum_{\kappa=1}^{n_i} q_{i_D\kappa}$, as in the individual case, "stores" the total of harvest damaged, whether its origin is in the direct submersion of the harvest or provoked by plant damages.

At the same time, we know that, depending on the coping strategy the farm adopts, we can have additional damages over the harvest. To take such effect into account, and, therefore, know the real value of $\sum_{\kappa=1}^{n_i} q_{i\kappa}$, we need to modify equation 5 introducing the new term, $q_{i_\beta\kappa}$:

$$q_{i_T\kappa} = q_{i\kappa} + q_{i_D\kappa} + q_{i_\beta\kappa} \tag{8}$$

Where:

1. $q_{i_T\kappa}$ is the potential harvest in plot γ_κ of farm i
2. $q_{i\kappa}$ is the effective harvest in plot γ_κ of farm i
3. $q_{i_D\kappa}$ is the damaged harvest in plot γ_κ of farm i by the flood
4. $q_{i_\beta\kappa}$ is the damaged harvest in plot γ_κ of farm i caused by the coping strategy of the farm i

Then equation 7 becomes:

$$\sum_{\kappa=1}^{n_i} q_{i_T\kappa} = \sum_{\kappa=1}^{n_i} q_{i\kappa} + \sum_{\kappa=1}^{n_i} q_{i_D\kappa} + \sum_{\kappa=1}^{n_i} q_{i_\beta\kappa} \tag{9}$$

Where:

1. $\sum_{\kappa=1}^{n_i} q_{i_T\kappa}$ is the potential yield of farm i
2. $\sum_{\kappa=1}^{n_i} q_{i\kappa}$ is the effective yield of farm i
3. $\sum_{\kappa=1}^{n_i} q_{i_D\kappa}$ is the damaged yield of farm i
4. $\sum_{\kappa=1}^{n_i} q_{i_\beta\kappa}$ is the damaged yield of farm i caused by the farm i's coping strategy

Or alternatively,

$$q_{i_T} = q_i + q_{i_D} + q_{i_\beta} \tag{10}$$

Where:

$$q_{i_T} = \sum_{\kappa=1}^{n_i} q_{i_T\kappa} \qquad q_i = \sum_{\kappa=1}^{n_i} q_{i\kappa} \qquad q_{i_D} = \sum_{\kappa=1}^{n_i} q_{i_D\kappa} \qquad q_{i_\beta} = \sum_{\kappa=1}^{n_i} q_{i_\beta\kappa} \tag{11}$$

Up-scaling a level in the production chain, we can express the amount of yield provided as input to the cooperative winery, Q_w, as the aggregation of the individual yields of its associates:

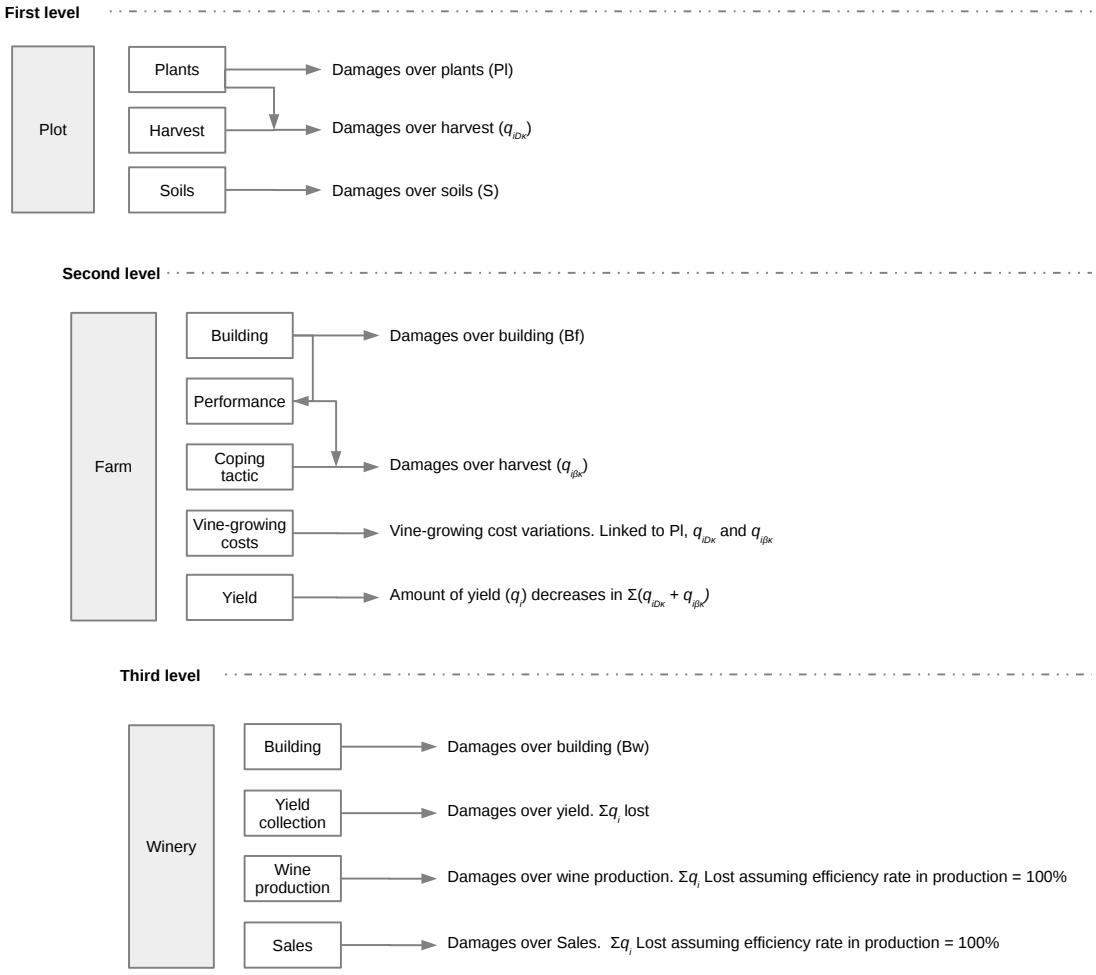


Figure 15: Hierarchy of damages for a flood hitting entities altogether in SFS

$$Q_w = \sum_{i=1}^{n} q_i = \sum_{i=1}^{n} \sum_{\kappa=1}^{n_i} q_{i\kappa} \tag{12}$$

Where n_i is the number of plots, γ_κ, of farm i, and n is the number of farms

Returning to figure 15, we can use the new nomenclature to clearly scout damages when different entities are flooded at the same time. As always let's assume i) the flood hits the system in year $t = 1$, and ii) seasonal sequence is winter-spring-summer-winter. Then, if the flood its the system in:

1. **Winter**. Impacts over plots flooded are reduced to reconditioning of soils (S)

   Impacts over farms flooded include buildings (B1) and performance. If opting for *outsourcing*, $q_{i_\beta \kappa} = 0$, in each plot owned by flooded farms. Therefore in autumn,

when harvest is done, in each productive plot owned by those farms $q_{i\kappa} = q_{i_T\kappa}$, thus $q_i = q_{i_T}$ at farms level for $t = 1$. If opting for *insourcing*, $q_{i_\beta\kappa} > 0$, in each plot owned by flooded farms, so in autumn $q_{i\kappa} < q_{i_T\kappa}$ in each plot owned by flooded farms, and $q_i < q_{i_T}$ at farms level for $t = 1$. In any case, vine-growing cost will vary

Impacts over wineries incorporate damages over buildings (B2) and performance. It will make the system lose Q_w of $t = 0$, but will have no effect over Q_w of $t = 1$. Since Q_w is lost, there will be no revenues for farms in $t = 1$, and the ones expected in $t = 2$ will be linked to the farms coping tactic. Wine-making cost will vary reflecting both situations.

2. **Spring**. Impacts over plots flooded include reconditioning of soils (S), losses of harvest $q_{i_D\kappa} > 0$ and plant destruction (Pl)

   Impacts over farms flooded include buildings (B1) and performance. If opting for *outsourcing*, $q_{i_\beta\kappa} = 0$, in each plot owned by flooded farms. Therefore in autumn $q_i < q_{i_T}$ in the amount given by q_{i_D} at farms level for $t = 1$. If opting for *insourcing*, $q_{i_\beta\kappa} > 0$, therefore in autumn $q_i < q_{i_T}$ too, but in the amount $q_{i_D} + q_{i_\beta}$. As in winter, vine-growing-cost will vary

   Impacts over wineries are the same than for winter. Since in spring destruction of plants is likely to happen, the impacts over wine-making cots and revenues can last longer in time

3. **Summer**. Impacts over plots and farms are the same as exposed for spring, while impacts over wineries are reduced to reparation costs over buildings and materials (B2). Impacts over revenues and wine-making cost in $t = 2$ —and potentially further in time— will reflect the level of destruction in plots and the coping tactics chosen by farms

4. **Autumn**. Impacts over plots and farms are the same as exposed for spring. Impacts over wineries comprise damages over buildings (B2) and performance. It will make the system lose Q_w of $t = 1$.

   As we can see, in $t = 1$ eventually all production gets lost. However but for different reasons:

   - It exists $q_{i_D\kappa} > 0$ at each flooded plot. Therefore at systems level we have $\sum_{i=1}^{n}\sum_{\kappa=1}^{n_i} q_{i_D\kappa} > 0$ provoked by the direct impact of floods over plots
   - If farm's coping tactic is *outsourcing*, then $q_{i_\beta\kappa} = 0$. There is no added damage by the farm, and the yield lost by the winery is:

$$Q_w = \sum_{i=1}^{n}\sum_{\kappa=1}^{n_i} q_{i_T\kappa} - \sum_{i=1}^{n}\sum_{\kappa=1}^{n_i} q_{i_D\kappa} \tag{13}$$

   - If farm's coping tactic is *insourcing*, then $q_{i_\beta\kappa} > 0$, the added damage by each

farm is $\sum_{\kappa=1}^{n_i} q_{i_\beta \kappa}$, and the yield lost by the winery is

$$Q_w = \sum_{i=1}^{n}\sum_{\kappa=1}^{n_i} q_{i_T \kappa} - \sum_{i=1}^{n}\sum_{\kappa=1}^{n_i} q_{i_D \kappa} - \sum_{i=1}^{n}\sum_{\kappa=1}^{n_i} q_{i_\beta \kappa} \tag{14}$$

Revenues in $t = 2$ will be null and wine-making cost will be reduced to the winery's structural cost. Due to plant destruction at plot's level, as it happens in spring and summer, effects over revenues and wine-making cost are expected to last longer in time, reflecting such plant destruction.

6 Output

6.1 Indicators and scales

As said in section 2.2, our productive system rests, both at collective and individual scale, over a vector of four key variables: production —Q_t—, revenues —R_t—, costs —C_{vg} (vine-growing) and C_{wm} (wine-making)— and investments and reinvestments —I_t. This last variable (I_t) serves us to group all reparations to be done in the system after a flood, reinvestments in plants and materials and, also, planed investments independent of the flood.

Every time any element of the system is flooded, as explained in section 5, one or more of those variables are going to experiment certain level of change. Thus, assuming that $B\vec{A}U_t$ and $S\vec{F}S_t$ are two vectors of key variables for their respective BAU and SFS scenarios:

$$B\vec{A}U_t = (I_t, Q_t, R_t, C_{vg_t}, C_{wm_t}) \tag{15}$$

$$S\vec{F}S_t = (I'_t, Q'_t, R'_t, C'_{vg_t}, C'_{wm_t}) \tag{16}$$

We can define the impact of a flood for each moment t as:

$$\vec{Imp}_t = S\vec{F}S_t - B\vec{A}U_t \tag{17}$$

Assuming that each farm is the smallest productive unit in the system, we can define as well C_{vg_t} and C_{wm_t} for each farm i as:

$$C_{vg_{i,t}} = F_{vg_i} + v_{vg_i} q_{i,t} \tag{18}$$

$$C_{wm_t} = \frac{F_{wm}}{\sum_{i=1}^{n} q_{i,t}} + v_{wm} q_{i,t} \tag{19}$$

Where:

1. F_{vg_i} is the structural or fixed vine-growing cost of the farm i. Assumed constant over time
2. v_{vg_i} is the operational or variable vine-growing cost of the farm i. Linked to the impacts over the farm and its coping tactic
3. $q_{i,t}$ is the yield of farm i in the moment t
4. F_{wm} is the structural or fixed cost of the winery or fixed wine-making cost. Assumed constant over time
5. v_{wm} is the operational or variable cost of wine-making. Assumed constant over time
6. $\sum_{i=1}^{n} q_{i,t}$ is the sum of yields of all farm $i \in [1, n]$ in the moment t, where n is the total number of farms

Using equations 17, 18 and 19, we can calculate the impacts for both each farm i, and the whole system, at any moment t (table 5)

Variable	Impact ($Imp_t = \vec{SFS}_t - \vec{BAU}_t$)	
	Individual level	Collective level (system)
I_t	$I'_{i,t} - I_{i,t}$	$I'_t - I_t$
Q_t	$q'_{i,t} - q_{i,t}$	$\sum_{i=1}^{n} q'_{i,t} - \sum_{i=1}^{n} q_{i,t}$
R_t	$p(q'_{i,t} - q_{i,t})$	$p\left(\sum_{i=1}^{n} q'_{i,t} - \sum_{i=1}^{n} q_{i,t}\right)$
Cvg_t	$v_{vg_i}(q'_{i,t} - q_{i,t})$	$v_{vg_i}\left(\sum_{i=1}^{n} q'_{i,t} - \sum_{i=1}^{n} q_{i,t}\right)$
Cwm_t	$v_{wm}(q'_{i,t} - q_{i,t}) + F_{wm}\frac{\sum_{i=1}^{n} q_{i,t} - \sum_{i=1}^{n} q'_{i,t}}{\sum_{i=1}^{n} q'_{i,t} \sum_{i=1}^{n} q_{i,t}}$	$v_{wm}\left(\sum_{i=1}^{n} q'_{i,t} - \sum_{i=1}^{n} q_{i,t}\right)$

I_t = Investment | Q_t = Production | R_t = Revenues
C_{vg} = Vine-growing cost | C_{wm} = Wine-making cost

Table 5: Impacts of floods over investments, production, revenues, vine-growing and wine-making costs, at individual ($\forall$ farm i) and system's level in a moment t

At this moment, it is worth notice that $q'_{i,t} - q_{i,t}$ in table 5 is not the same than q_{i_D} in equation 10. In the equation, we refer only to the yield damaged by the flood, while $q'_{i,t} - q_{i,t}$ also includes the yield lost because of disability of an agent to perform an assigned task due to the flood. That is to say, it includes q_{i_β} and Q_ω

Aggregating the different components of the vector of impacts and regrouping terms, we can express the total impact for each individual farm as in equation 20:

$$Imp_{i,t} = (I'_{i,t} - I_{i,t}) + (p + v_{vg_i} + v_{wm})(q'_{i,t} - q_{i,t}) + F_{wm}\frac{\sum_{i=1}^{n} q_{i,t} - \sum_{i=1}^{n} q'_{i,t}}{\sum_{i=1}^{n} q'_{i,t} \sum_{i=1}^{n} q_{i,t}} \tag{20}$$

And for the whole system as in equation 21:

$$Imp_t = (I'_t - I_t) + (p + v_{vg} + v_{wm})\left(\sum_{i=1}^{n} q'_{i,t} - \sum_{i=1}^{n} q_{i,t}\right) \tag{21}$$

Where p is the market price of the wine produced with the yield of the farm i.

That is, the impact of a flood in any moment t comes given by the differences in investment and yield/production. In addition, at individual level, such impact comprises the redistributing effect driven by the individual share of the winery's fixed cots. In other words, the indirect effect that the winery's financial structure has over its associates. Therefore, for us, the collectivity has not the same properties of the individuals when up-scaling; rather the collectivity is an aggregation of the individuals with their own features involved in such collectivity. As a result, in our model, impacts of floods are level-dependent.

Using Brémond et al. [2013], we are able to build a damage time scale with two time spans: i) immediate impacts —"those ones which occurs during or immediately after the flood event"—, and ii) induced impacts —"those which occur later in time". Such scale will allow us to discriminate and follow up the impacts over elements that cannot be solved immediately, as well as their consequences during the aftermath of the flood in a time span of our choice.

Assuming the flood occurs in $t = t_1$, the mathematical formulation of individual immediate impacts will be as follows

$$\begin{aligned} Imp_{i,t=1} = (I'_{i,t=1} - I_{i,t=1}) + (p + v_{vg_i} + v_{wm})(q'_{i,t=1} - q_{i,t=1}) + \\ + F_{wm} \frac{\sum_{i=1}^{n} q_{i,t=1} - \sum_{i=1}^{n} q'_{i,t=1}}{\sum_{i=1}^{n} q'_{i,t=1} \sum_{i=1}^{n} q_{i,t=1}} \end{aligned} \tag{22}$$

And for the whole system as in equation 23:

$$Imp_{t=1} = (I'_{t=1} - I_{t=1}) + (p + v_{vg} + v_{wm})\left(\sum_{i=1}^{n} q'_{i,t=1} - \sum_{i=1}^{n} q_{i,t=1}\right) \tag{23}$$

For induced impacts, such formulation can be enounced as in equation 24, at individual level, and as in equation 25, at system level:

$$Imp_{i,t\in[t_2,t_n]} = \sum_{t=2}^{t_n}(I'_{i,t} - I_{i,t})(1+r)^{1-t} + (p + v_{vg_i} + v_{wm}) \sum_{t=2}^{t_n}(q'_{i,t} - q_{i,t})(1+r)^{1-t} + \\ + F_{wm} \sum_{t=2}^{t_n} \left(\frac{\sum_{i=1}^{n} q_{i,t} - \sum_{i=1}^{n} q'_{i,t}}{\sum_{i=1}^{n} q'_{i,t} \sum_{i=1}^{n} q_{i,t}} \right)(1+r)^{1-t} \tag{24}$$

$$Imp_{t\in[t_2,t_n]} = \sum_{t=2}^{t_n}(I'_t - I_t)(1+r)^{1-t} + \\ + (p + v_{vg} + v_{wm}) \sum_{t=2}^{t_n} \left(\sum_{i=1}^{n} q'_{i,t} - \sum_{i=1}^{n} q_{i,t} \right)(1+r)^{1-t} \tag{25}$$

Where $(1+r)^{1-t}$ is the discount factor[4] of the period t for a discount rate r.

Brémond et al. [2013] allows us to introduce another scale. Our so-called *spatial scale*, where impacts are identified as direct impacts —those ones "related to direct exposure to the disaster" (physically flooded in our case)— or indirect impacts —"those which occurs in a area that has not been exposed to flooding". Such classification is, nonetheless, agent-dependent (or system-dependent), thus, we are forced to predefine the entity we assume is the elementary unit in the system, before making any potential classification of damages based on this scale. The presence of the two scales gives us the additional possibility of, crossing them, classify impacts in:

- Immediate Direct impacts: impacts due to direct exposure to flood, and manifested during the flood or immediately after.
- Immediate Indirect impacts: impacts occurred outside the flooded area, and manifested during the flood or immediately after
- Induced Direct impacts: impacts due to direct exposure to flood, manifested later in time.
- Induced Indirect impacts: impacts occurred outside the flooded area, manifested later in time

Impact information on those 5 key variables is presented through a collection of 12 different indicators, founded on Barbut et al. [2004], Bremond [2011], Brémond et al. [2013] and Hiete and Merz [2009] (figure 16). Over such battery of indicators, different complementary classifications are possible. The first, and probably the most intuitive one, classifies the indicators by entities —plot, farm and winery (central part of figure 16)—, so it is possible to identify where in the model the impact is originated, or, in other words, which entity has been impacted.

Additionally, following the scales exposed above based on Brémond et al. [2013], indicators present two alternative categorizations. Figure 16 shows, in its left side the resulting classification according our time scale, whereas, in its right side, we have the

[4]Discount factors have been introduce to ensure the comparability of financial flows over time

so-called *spatial scale*, assuming the tandem farm-plot as elementary unit of the system and, therefore, classifying impacts according their point of view.

The structure of indicators in figure 16 can be replicated for any individual entity. Therefore the same collection of indicators is available for, in our case, every individual farm in the system. As it has been said, in our model, the collectivity is an aggregation of the individuals —and their individual features— involved in such collectivity (table 5), rather than an extrapolation. Thus, aggregating each of the individual values, we will be able to replicate the same structure at system's level[5], and impacts would reflect the same values than if the would have been calculated following table 5's formula (figure 17).

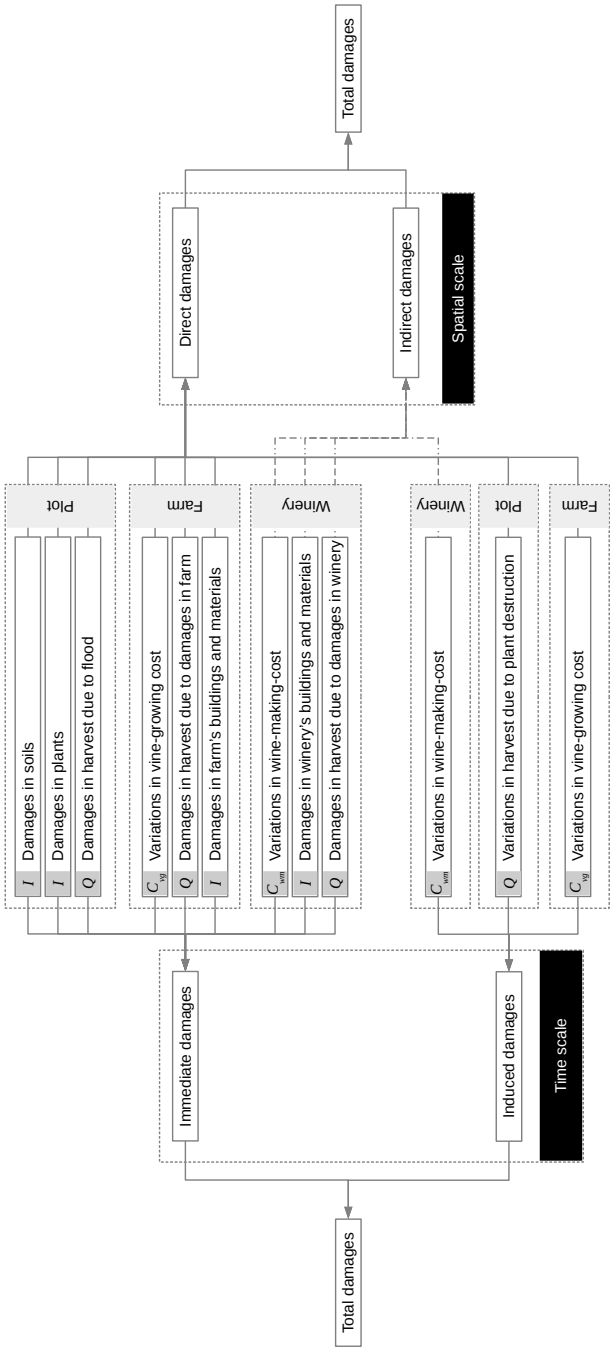
To prevent metrics from showing potential scale effects induced by entities and systems' sizes, we build a synthetic measure of impacts, dividing each indicator by the so-called *yearly potential gross benefit* (equation 26). Under our point of view, it presents three different advantages: i) as metric, the *yearly potential gross benefit* is easy to understand; ii) at the same time, it is also available at all the levels we would like to consider; and iii) it is a metric of the entity/system's annual gross capacity for resource generation. Therefore it provides a final synthetic measure easily interpretable.

$$PB = npv(p - C_{wm} - C_{vg}) \tag{26}$$

Where:

1. PB = Potential gross benefit
2. n = number of plots (all of them. Not only productive ones)
3. p = price of wine
4. C_{wm} = wine-making costs by hl
5. pv = productivity by ha
6. C_{vg} = vine-growing cost per ha

[5]If The system is composed by different cooperative wineries coexisting in the same terrain, the structure is replicable at individual level, winery level and subsystem level.



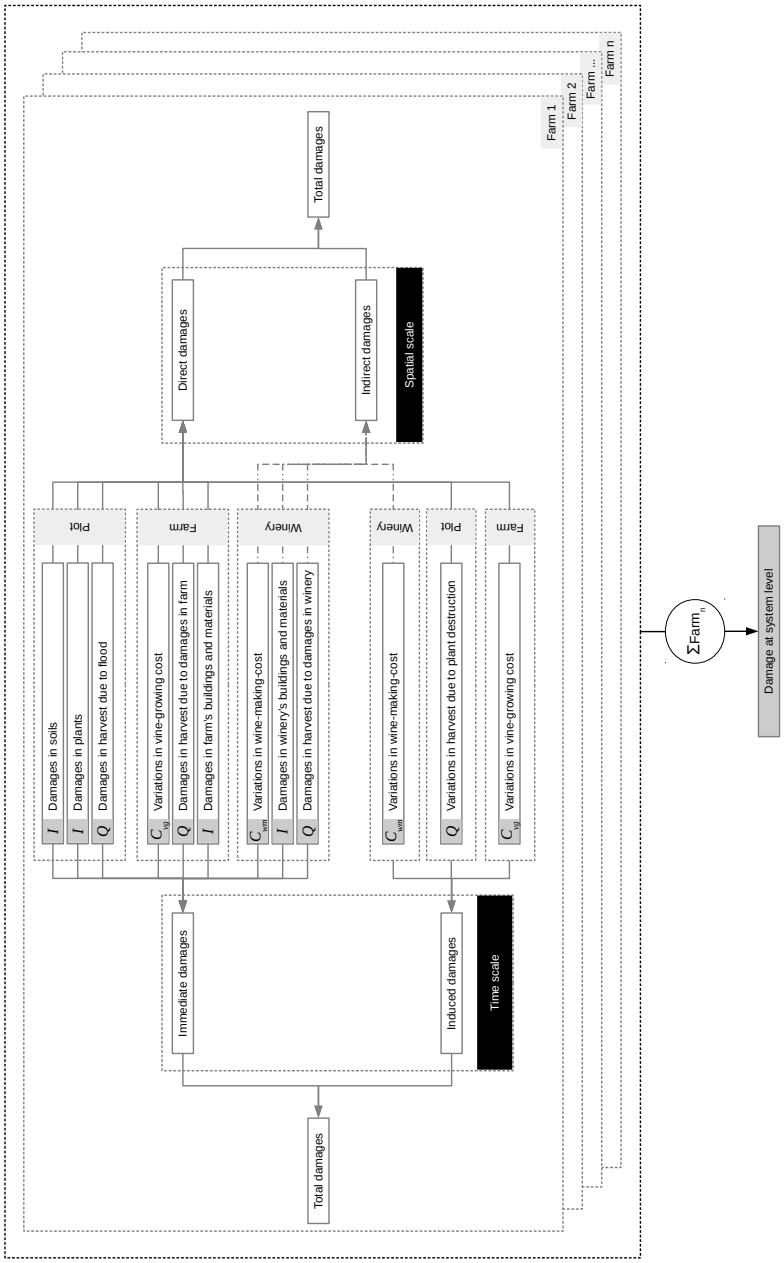
Source: based on Brémond et al. [2013]

Remarks:

Each indicator includes in the left side of its frame the variable to which it refers, according to the nomenclature included at the beginning of this section: I_t = Investment | Q_t = Production | R_t = Revenues | C_{vg} = Vine-growing cost | C_{wm} = Wine-making cost

Spatial scale classified assuming the ensemble of farm and its owned plots as elementary unit of the system.

Figure 16: Indicators



Source: based on Brémond et al. [2013]

Remarks:

Each indicator includes in the left side of its frame the variable to which it refers, according to the nomenclature included at the beginning of this section: I_t = Investment | Q_t = Production | R_t = Revenues | C_{vg} = Vine-growing cost | C_{wm} = Wine-making cost

Spatial scale classified assuming the ensemble of farm and its owned plots as elementary unit of the system.

Figure 17: Individual-global duality of indicators

6.2 Influence of the discount factor over the damage assessment

Our indicators consider discount factors to assess damages along time. While it takes into account the economic idea that assessment of future values is not independent from the moment they occur, its presence will influence the magnitude of induced impacts, hence total ones.

To show the influence of the discount rate over the different variables that conform the indicator, we have, first, isolated the discount factor from any variable. Then tested it over a period of 30 years (biologic cycle of a plot in our model) for values of the discount rate ranging from 0 to 1, with increments of 0.01 units.

Figure 18 displays the value of the sum of discount factors over the 30 years chosen. Numerical values for discount rates from 0 to 0.1 are also provided in the table attached to the figure. As we can see, the most sensitive area is found when $r \in [0, 0.1]$. In this area $Imp_{t\in[t_2,t_30]}$ can drop the 70% of their values. When $r \in [0, 0.05]$, impacts present a faster decreasing evolution —dropping 50% of the value— than values of $r \in (0.05, 0.1]$ —remaining 20%.

The choice we make about the discount rate is far from trivial. It will affect directly the weight future impacts have in relation to the immediate impacts of the flood, and the importance of the induced impacts in the final mix of damages. In the interval $[0, 0.05]$, each percentage point of variation in the discount rate is translated approximately in 10% of variation of the induced damages. With $r \in [0.05, 0.1]$, such multiplier drops to -4 for each percentage point. Values of $r \in [0.1, 0.15]$ will present a multiplier of -2, while when $r \in [0.15, 0.23]$ it will be -1. For values of r beyond 0.23, each percentage point of increment will make variations in the total impacts inferior to -1%.

Discount Rate (r)	$\Delta Imp_{t\in[t_2,t_n]}$ in $t = 1$, $\forall r \in [0.01, 0.1]$ in relation to $r = 0$ (%)
0.01	- 14.05
0.02	- 25.56
0.03	- 35.04
0.04	- 42.92
0.05	- 49.50
0.06	- 55.03
0.07	- 59.72
0.08	- 63.72
0.09	- 67.15
0.10	- 70.11

Figure 18: Variations in damage assessment in t = 1 for discount rates $r \in [0, 1]$ with $\Delta r = 0.01$, over a time span of 30 years

7 Model implementation

The model is implemented combining Netlogo 5.3.1 [Wilensky, 1999] and R 3.4.1 [R Core Team, 2017], through the RNetLogo package [Thiele et al., 2012] in its version 1.0.2.[6]

7.1 Overall structure and processes

The model's code structure —outlined in figure 19— can be split into two different big blocks that will interact, feeding information one to each other, all along the process. Such blocks also correspond to the different languages used to code the model.

Roughly speaking, on one side we have the `R` block, that contains:

- `Input generator` (top left of figure 19)
- `Simulation launcher/iterator` (left side of figure 19)
- `Impact calculator` (left side of figure 19)

  These two last procedures are thoroughly explained and outlined in section 7.3 and figure 22

On the other hand, the `Netlogo`'s one is constituted by the model's core, and so-called `flood simulator` (right side of figure 19. More detailed in section 7.2 and figures 20 and 21).

The very first step in the simulation process pass through the input generator. Its mission is to provide values to the `flood simulator`. To do that, it equips the user with a way to translate the values of the simulation parameters —the so-called `scenario's conf. data` in figures 19 to 22, whose content is summarized in table 6— into information readable by the `flood simulator`. Once prompted (or facilitated by user's scripts) in the `R` terminal and processed, such information is stored with the proper format/order for `Netlogo` in standard `txt` files on the hard disk. This procedure obeys to different objectives:

- Time saving: all simulation parameters and values of a plan of experiments can be created, and stored, prior to the simulation launching.
- Replicability: all simulation parameters and values of a plan of experiments can be replicated numerous times, just by calling the proper file in the `flood simulator`
- Feedback: the stored files grant access to the simulation parameters, so the configuration of a particular simulation or plan of simulations is always accessible for the user.
- Reuse: new simulation parameters files can be done, reusing the ones already done without having to build entire new ones.

[6] Although available, higher versions of Netlogo have included major changes regarding language, and the model has not been yet adapted

- Task sharing and information exchange: configurations of scenario parameters can be shared directly between users.

Global variables	Age at which the plot is considered productive Age at which the plot is replanted Number of farmers in the cooperative winery Number of cooperative wineries Amount of money to spend in soil set up after the flood Price per ha to replace the plants in the plots Amount of money to spend setting up again the farm after the flood Amount of money to spend setting up again the winery after the flood Average productivity by ha Average wine price by hl Periods to simulate (one period = one season) Configuration of links between plots and farms Extent of the prone area (by default = 100) Extent of the flood Season to simulate the flood Global coping tactic in the system
Plot	Internal `Netlogo ID` Position over terrain Owner Productivity State (planted/unplanted) extent Age Operational vine-growing cost associated
Farm	Internal `Netlogo ID` Position over terrain Initial amount of cumulated balance Proportion of structural costs over total vine-growing costs Winery's Internal `Netlogo ID` to be associated to
Winery	Internal `Netlogo ID` Position over terrain Fixed vinification costs proportion Average wine-making cost per hl Proportion of structural cost over total cost Efficiency

Table 6: Summary of parameters that conform the `scenario's conf. data` of the `flood simulator`, whose values need to be provided to the `input generator`. Classification by entity

Although essential for the whole simulation process, the `input generator` is not part of the simulation procedure. It means that when the `simulation launcher/iterator` be-

gins, the `input generator` will not be called at any moment. Only the stored `scenario's conf. data` files generated by it will be.

The `simulation launcher/iterator` starts the simulation procedure. As well as the `input generator`, it will provide simulation parameters to the initialization of the `flood simulator`. Such parameters are the ones whose values are expected to be modified by the `simulation launcher/iterator` in order to complete the experiment plan. An example is provided in table 7.

Parameters	Periods to simulate (one period = one season)
	Configuration of links between plots and farms
	Extent of the prone area (by default = 100)
	Extent of the flood
	Season to simulate the flood
	Global coping tactic in the system

Table 7: example of parameters whose values are provided by the simulator launcher/iterator

As it can be seen, values in table 7 are already included in table 6. The information provided by the `input generator` and the `simulation launcher/iterator` is complementary. It means that parameter values whose effect we wish to test, are expected to be provided through the `simulation launcher/iterator`, whereas values stable values should be passed through the `input generator`.

The `simulation launcher/iterator` should set, additionally, values for two more variables (table 8):

1. `dam_byR`: special boolean variable passed to the `flood simulator`, setting up whether we wish to use the Random Number Generator (RNG) of `Netlogo`, or to provide the damages over plants in plots through the RNG of `R` (the difference between both methods will be explained in section 7.2)
2. Number of iterations of each simulation to be done, due, precisely, to the presence of random effects in the simulations.

When all values are set, the `simulation launcher/iterator` calls the `flood simulator` once per simulation[7], passing the control of the process to `netlogo`. When each simulation is finished, the `flood simulator`, returns control of the process to `R` along with the

[7]Assuming the simulation parameters in table 7, we call simulation to the performance of one system set up by parameters in table 6, with one specific `configuration of links`, during n `periods to simulate`, that uses one `coping tactic` when a flood —defined by one `flood extent` shorter or equal

Special parameters
`dam_byR`
Iterations

Table 8: Special parameters to be provided to the simulator launcher/iterator

BAUs[8]/SFSs, to be processed by the `simulation launcher/iterator`. At this stage three different sequential tasks take place:

1. SFSs/BAUs are stored in the hard disk in R native format file *.rds*[9];
2. Data from SFSs/BAUs is classified into the different time spans considered (see section 6) and stored in auxiliary files associated to each SFSs/BAUs
3. Forward the auxiliary files associated to SFSs/BAUs to the `impact calculator`

Over the those auxiliary files, the `impact calculator` determines the impact of the flood over the SFSs by comparison with BAUs. Impacts are then stored in the hard disk in R native format file *.rds* for further analysis[10]. When all simulations are done, the analyst has two possibilities over the stored results:

- To conduct the automated pre-coded statistical and graphical analysis, and/or
- To conduct their own statistical and graphical analysis

to the system's `extent of the prone area`— hits the system during one of the 4 first periods —set by `season`

[8]By default BAUs are not iterated.

[9]Such format, directly readable by R allows us to store the file already compressed, saving a significant amount of space when cmpared to *.csv* or *.txt*

[10]It is possible, though, to convert them to more standard formats, readable by other software

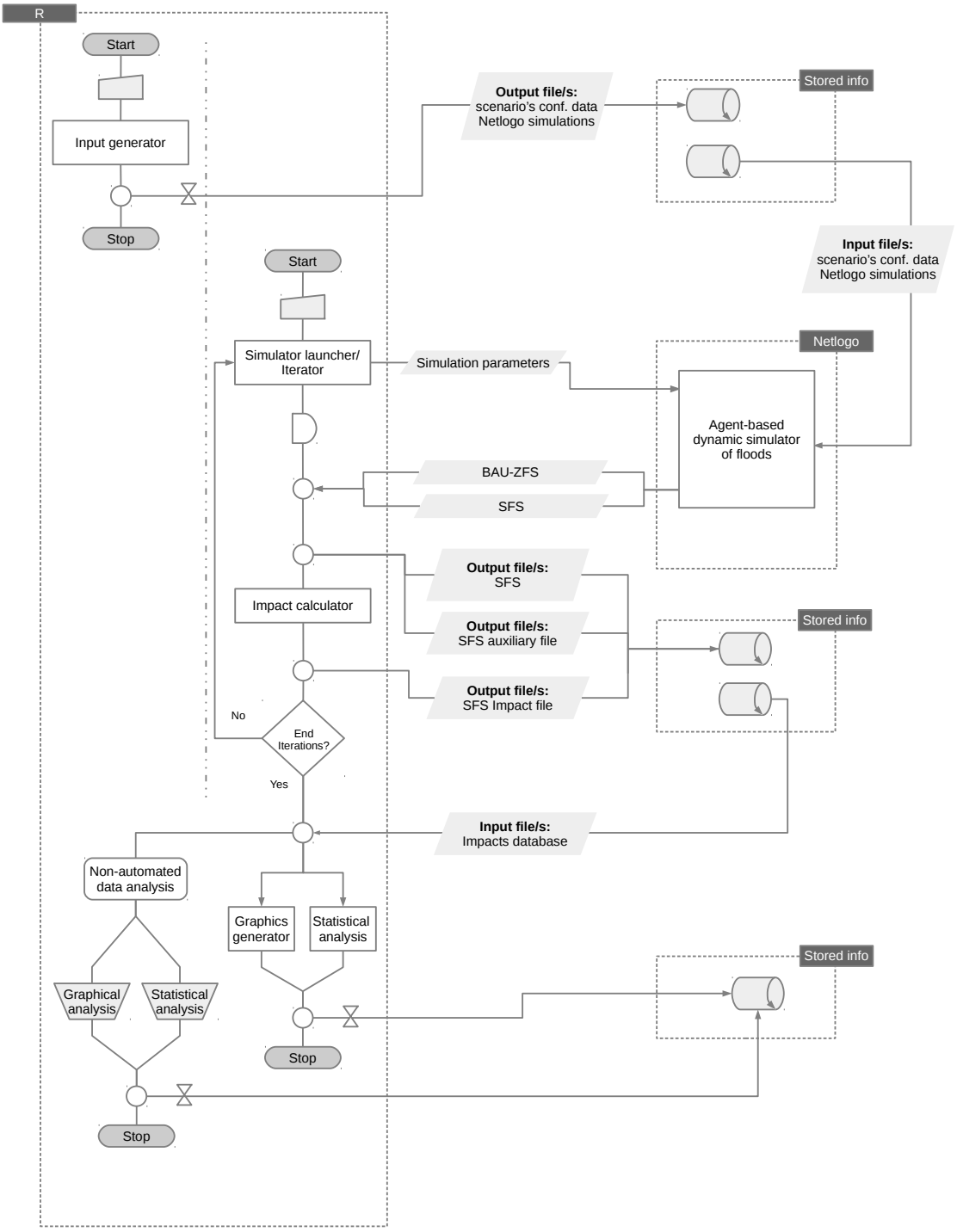


Figure 19: model's general flowchart

7.2 Flood simulator

As it has been said, the `flood simulator` is the core of the simulation process. It is built as an agent-based model (fully written in Netlogo 5.3.1 [Wilensky, 1999]), and developed to be able to work i) as part of the simulation procedure when called by `R` —headless mode in our architecture— , or ii) independently –trough its own GUI[11]—, always the input files with the parameters in table 6 exist, and `dam_byR` is set to `FALSE`.

Either way, the procedure remains the same (figure 20). It starts setting up the simulation information coming from the `scenario's conf.  data` files, and the `simulation launcher/iterator` in `R` when in headless mode. In other words, it displays entities over the terrain, assigns links between them, provides values to the key variables of the system and start the `season sequence` (`winter-spring-summer-autumn`).

The simulation sequence starts first season of the year (usually winter). It then checks if any flood is programmed to happen in such season; assuming no flood will take place, the procedure perform the operations scheduled during such season and advances one position on the `season sequence`.

This new season is compared with the first one in the sequence. When different, the procedure returns to the beginning (checking for programmed floods, etc). When equal, the value of the variables used to calculate the impacts are stored in memory as result of the year (`yearly result`). The procedure starts then a new year and the `season sequence` restarts.

When the procedure arrives to the end of the simulation (indicated by `periods to simulate` in table 6) it returns the collection of values stored in memory. If the model is used in headless mode the values, and the control of the process, are passed directly to the `simulation launcher/iterator`. Otherwise, the user should extract the values and store them himself.

The behavior of the procedure in each season is a little bit more complex and needs a more thorough description. Seasons and years are not independent in our model (see section 4). Instead, they should feed each other with information that ensures the correct performance of the procedure. Assuming the procedure is in year n, the season to simulate is `winter`, and no flood has hit the system, nor will it during n, the procedure's seasonal component can be described as follows:

1. The procedure updates the number of productive plots and other variables such as the vine-growing costs.
2. Plots are replanted (`investment` task)
3. The vine-growing costs of the season are calculated and stored in memory (access 0.1)
4. The wine-making task is done, with the `available amount of input` stored in memory by the `winery's yield collection` task in `autumn` of $n-1$ (access 4)

[11]It exists an alpha version of GUI in `R shiny` that pretends to serve as front end for the whole simulation process, making the model more user-friendly. In development

5. The wine-making task stores in memory `amount to sale` (access 2) and `cost data` (access 3)
6. The procedure advances one season. Now we are in `spring` of year n
7. The vine-growing costs of the season are calculated, added to those stored in access 0.1 and updated in memory (access 0.2)
8. The sales task is done, with the `amount to sale` stored in memory (access 2) from the `wine-making` task in `winter` of n
9. The sales task provides the `revenues data` to the procedure.
10. With the information in 9 and the `cost data` stored in memory (access 3), the cooperative winery splits its cost and revenue among its associates
11. The `financial balance` task adds the `vine-growing cost of the year` of $n-1$ (stored in `autumn`; access 1) to the result of 10 to calculate the final financial balance.
12. The procedure advances one season. Now we are simulating `summer`
13. The vine-growing costs of the season are calculated, added to those stored in access 0.2 and updated in memory (access 0.3)
14. The procedure advances one season. Now we are simulating `autumn`
15. The vine-growing costs of the season are calculated, added to those stored in access 0.3 and updated in memory as `vine-growing cost of the year` of n (access 1)
16. The `harvest` task is done and provides `yield data` to the `winery's yield collection` task
17. The `winery's yield collection` task updates the `available amount of input` (access 4)
18. The procedure advances one season. Now, year n is over and `winter` of $n+1$ will be simulated

The presence of the floods adds a layer of complexity. When a flood is scheduled to hit the system, the target season follows a parallel procedure, outlined in figure 21. Known the value of the `flood extent`, each entity will check its status, reporting `flooded` when its coordinate in the x axis is smaller or equal to the `flood extent`. Together with its status, each entity will report as well the level of individual damages and the consequences over its performance. The algorithm incorporates all the information, updating whichever values are needed (harvest lost per plot, destruction of plants, tactic to follow by flooded farms, etc), and proceeds to calculate.

For instance, let's assume that the flood takes place in `summer` of year n ; let's assume as well that a few of the farms have been hit, and they only count on their own resources to face the aftermath. Steps 1 to 12 will remain the same, whereas from 13 it will be as follows:

13. The task `update values` introduces in the system information about harvest lost and plant destruction on plots, and material impacts on farms along with their

coping tactic.

The vine-growing costs of the season are calculated taken into account the new information: i) those plots destroyed will not pay the vine-growing cost from now on until they are replanted. ii) impacted farms, since they do not have extra support, will not perform all their task, thus vine-growing costs will be smaller this season.

The final seasonal amount of vine-growing costs is added to those stored in access 0.2 and updated in memory (access 0.3)

14. The procedure advances one season. Now we are simulating `autumn`
15. The vine-growing costs of the season are calculated over not destroyed plots, added to those stored in access 0.3 and updated in memory as `vine-growing cost of the year` of n (access 1)
16. The `harvest` task is done and provides `yield data` to the `winery's yield collection` task

    Since some vine-growing tasks could not be performed by the few farms hit in `autumn`, the consequences over the harvest of the not destroyed plots they own, are taken into account (reducing the final amount)
17. The `winery's yield collection` task updates the `available amount of input` (access 4)
18. The procedure advances one season. Now, year n is over and `winter` of $n + 1$ will be simulated

As stated in section 7.1, to determine whether plants in a plot are destroyed, thus the plot, we recur to RNGs. The `flood simulator` is capable to use two different ones depending on the value passed to `dam_byR`. Such feature responds to a need imposed by the replicability of iterations that could not be satisfied by Netlogo: using the netlogo's RNG we get different plots destroyed each iteration, and all of the iterations are independent. If we simulate an interval of m values for a given parameter p, repeating each value n iterations, we get mxn independent simulations for the parameter p.

That procedure impedes users to be sure in what proportion changes during the iteration n are due to variation of p and not to the RNG's behavior. To solve such contingency we make use of the R's RNG to generate the series of destroyed plots in each iteration. Then they are passed to the `flood simulator` as data lists. When `dam_byR` is set to `TRUE`, the `flood simulator` uses the data lists passed by the `simulation launcher/iterator` instead of Netlogo's RNG.

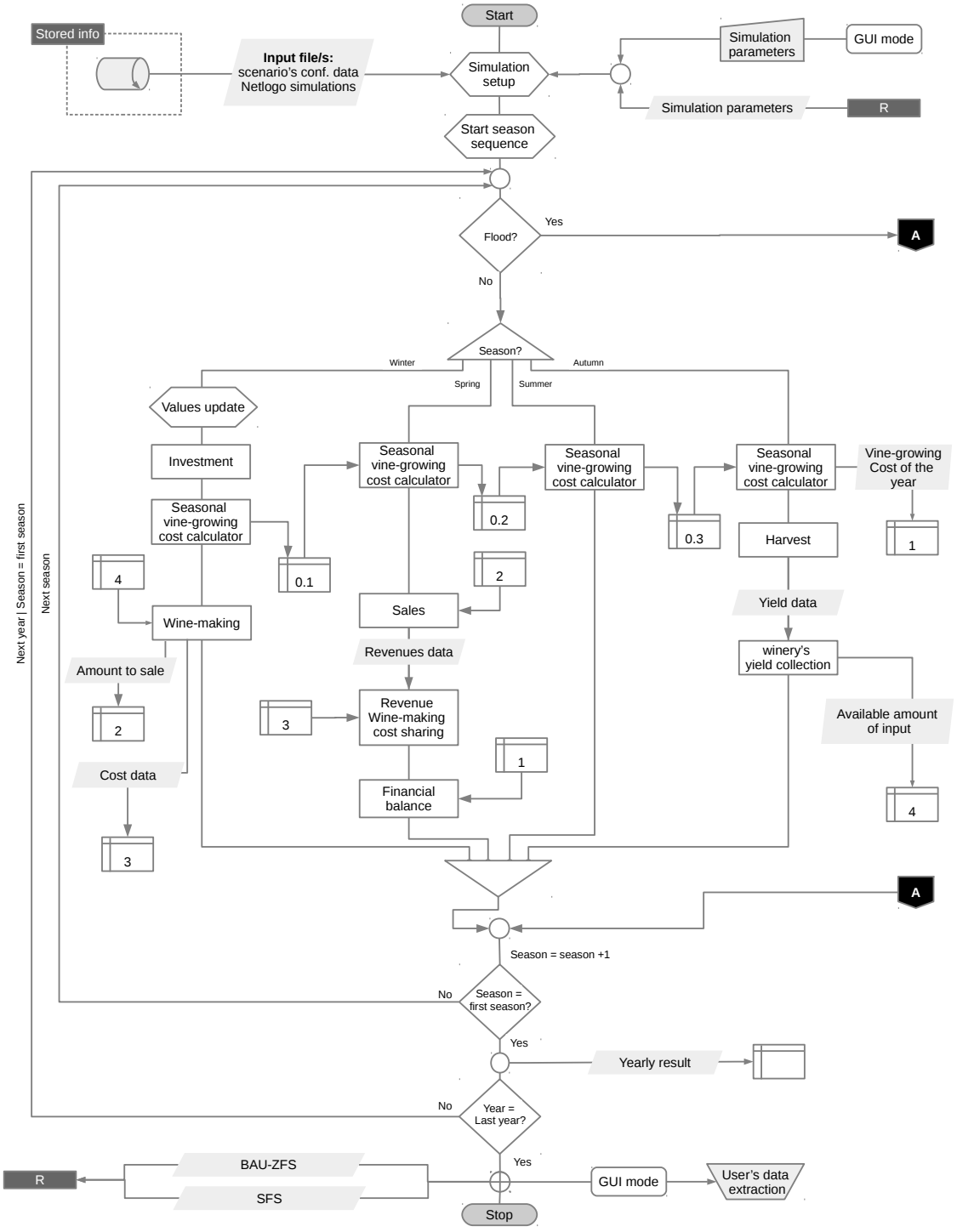


Figure 20: model's core flood simulator flowchart

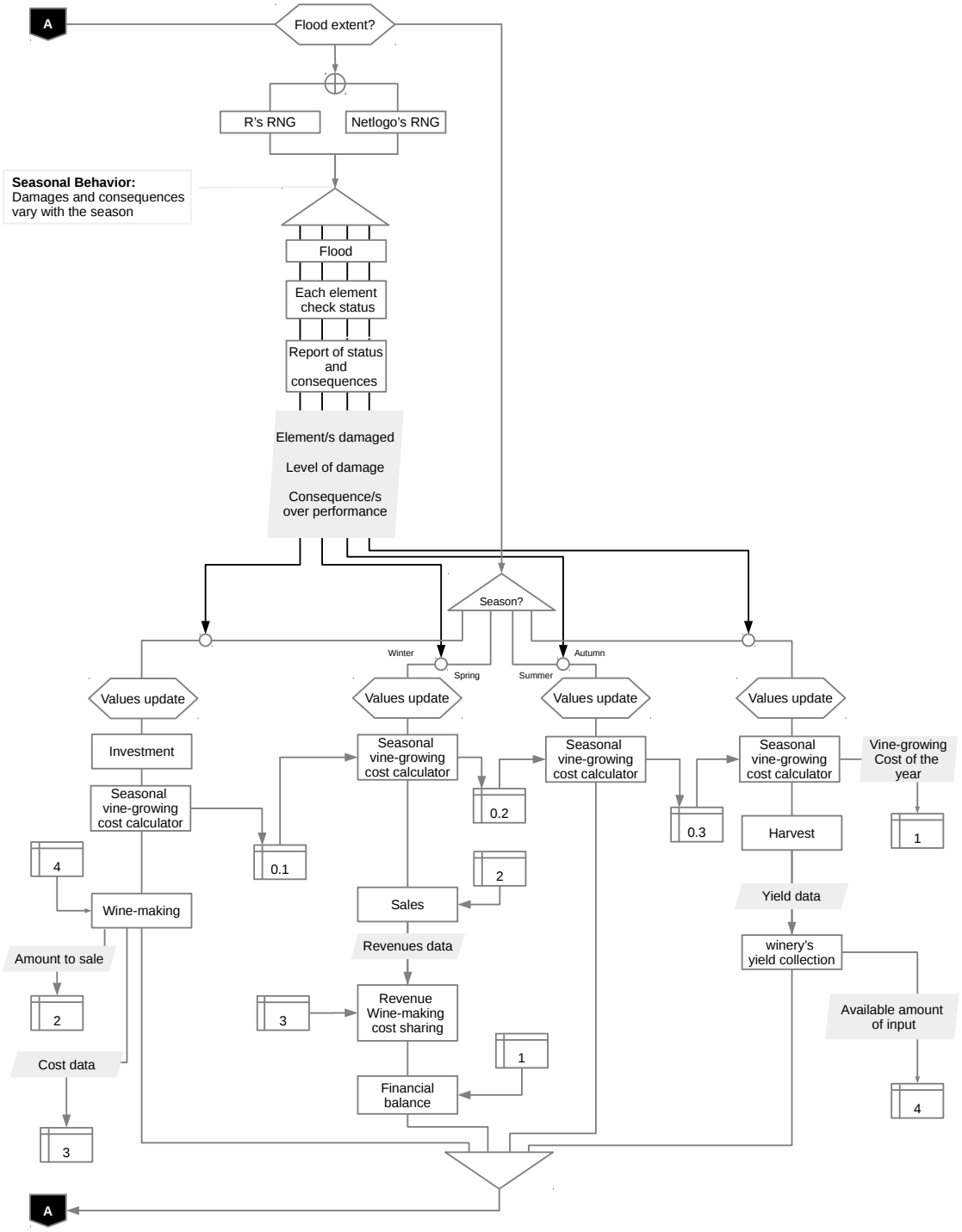


Figure 21: model's core flood simulator outline

7.3 Simulator launcher and impact calculator

The `simulation launcher/iterator` presents an structure more complex than the one offered to describe the general structure of the simulation process. A more detailed outline is offered in figure 22.

The process starts with the user introducing the values of the parameters to simulate (see section 7.1), which are stored in memory for further usage.

The procedure is parallel-ready. Therefore, once the `simulation launcher/iterator` is launched, the number of available CPU cores is detected and used to set up a cluster. Over the cluster, the `flood simulator` is called as many times as CPU cores available in the cluster, which reduces significantly the simulation time.

At this point, it is worth mention that Netlogo needs an specific ID for each of the `flood simulators` called in parallel through RNetLogo —thus tasks can be sent to an specific `flood simulator`. Such IDs should be set up beforehand to avoid unwanted crashes. In our model, is up to users to decide the best strategy to approach such matter. Although possible, it is strongly unadvised to open/close a `flood simulator` each time a new simulation [12] is launched. It reduces considerably the advantages of the parallelization, overcharging the system with unnecessary operations that a good ID strategy can avoid[13]

When the `flood simulator` returns the control and the simulation results to the `simulation launcher/iterator`, this last one executes the processes already described in section 7.1:

1. Storage of the raw SFS data in the hard disk
2. Classification of the raw data according the predefined time scale spans (see section 6), and storage into auxiliary files associated with the simulation in the hard disk.
3. Computation of impacts by comparison of SFSs against BAUs (`sweeper` task), and storage in the hard disk.

Before the procedure initiates the following iteration, all auxiliary objects created during the iteration are erased from the virtual memory. Tests have revealed a considerable usage of RAM memory during each iteration, thus the procedure has been equipped with an "eraser" to prevent crashes and overdemand of resources.

[12] see footnote 7

[13] For instance, assuming we have available 4 CPU cores and we want to simulate floods in each season, we can ID each of the `flood simulators` with one of the seasons. This way, each CPU will open one `flood simulator` with the given ID. All simulations with the same ID (season) will be sent to the same `flood simulator`, over the same core. Once all those simulations are done, the `flood simulator` with that specific ID(season) is closed.

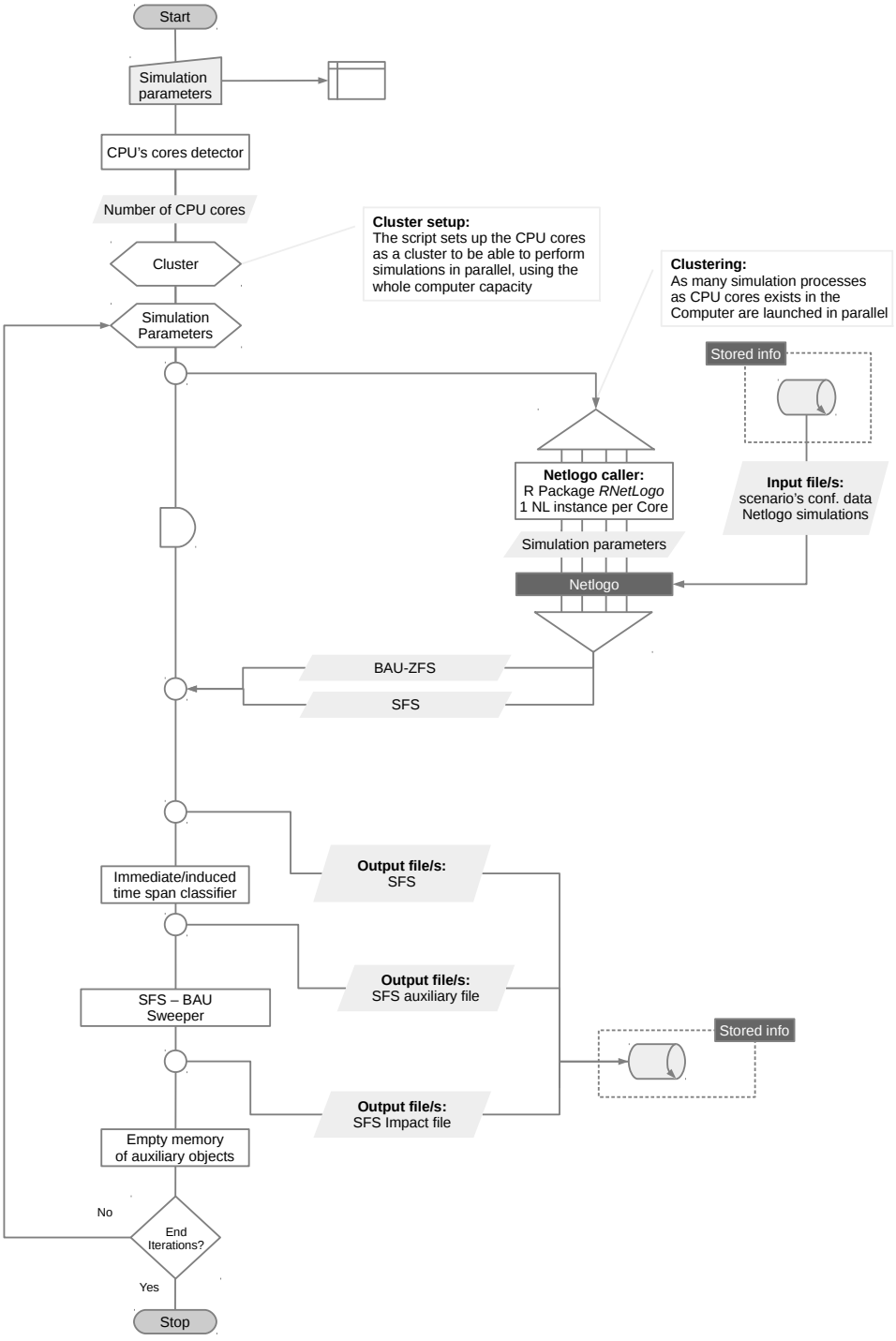


Figure 22: model's core flood simulator flowchart

8 Model calibration

8.1 Terrain

In order to provide realistic configurations to our simulated territory (section **??**), we have integrated in our model geographical information from the Aude and Var regions, located in southeast France.

Our choice has not been arbitrary. Both territories have been impacted by several floods, some of them particularly dreadful and damaging. Such particular circumstances have included them as study cases in the research project *Résilience des territoires face à l'inondation Pour une approche préventive par l'adaptation post-événement*, in which this work is framed.

Our model has three different entities: plots, farms and wineries. For two of them —plots and wineries—, it has been possible to find information to calibrate territorial configurations. Unfortunately, information on geographic locations of farm buildings in our territories (or any other one) was impossible to find. In consequence, we have asked experts in the field to formulate a reasonable hypothesis of territorial distribution. In standard conditions, we assume 20% of farms' building are located in prone area.

Regarding plots, maps in figure 23 display the resulting figures when we superpose the available information on vineyard extent —according the two different sources available : CORINE Land Cover and registre parcellaire graphique—, and the flood prone areas —available through the *Atlas des Zones Inondables de France*— in both territories. Discrepancies between both sources of information are evident, however, we have no found any available criteria to opt for one or another.

In such situation, our choice has been to keep both of them, and produce territorial configurations using the highest proportion of plots in prone area (table 9). This way, using the flexibility that our way to simulate floods provides us with, we can cover all the potential options.

	RPG 2012	CORINE 2012
Aude	18.28	15.08
Var	29.71	25.50

Table 9: Percentage of total ha of vineyards in prone areas according source of information

Productivities in both areas are so different, which is coherent with the different commercial orientations shown in table **??**. Typically AOC lands are less productive than IGP ones with prices for the final product following the opposite path

With reference to wineries, the approach has been similar. Once the prone areas could be established, the exact location[14] of each winery interviewed during the process of

[14]Such exact location has been provided to our maps through google Maps

Department	Average productivity (hl/ha)
Aude	87
Var	46

Source: France Agrimer 2012

Table 10: Average productivity of vineyards in the departments

recollection of information was incorporated. As a result, we can verify that cooperatives can be situated both in and out of prone areas (see figure 24).

8.2 Vine-growing

Tasks. Bremond [2011] identifies 14 different vine-growing tasks, with the annual distribution pattern showed in figure 26, assuming an standard 52 weeks' year.

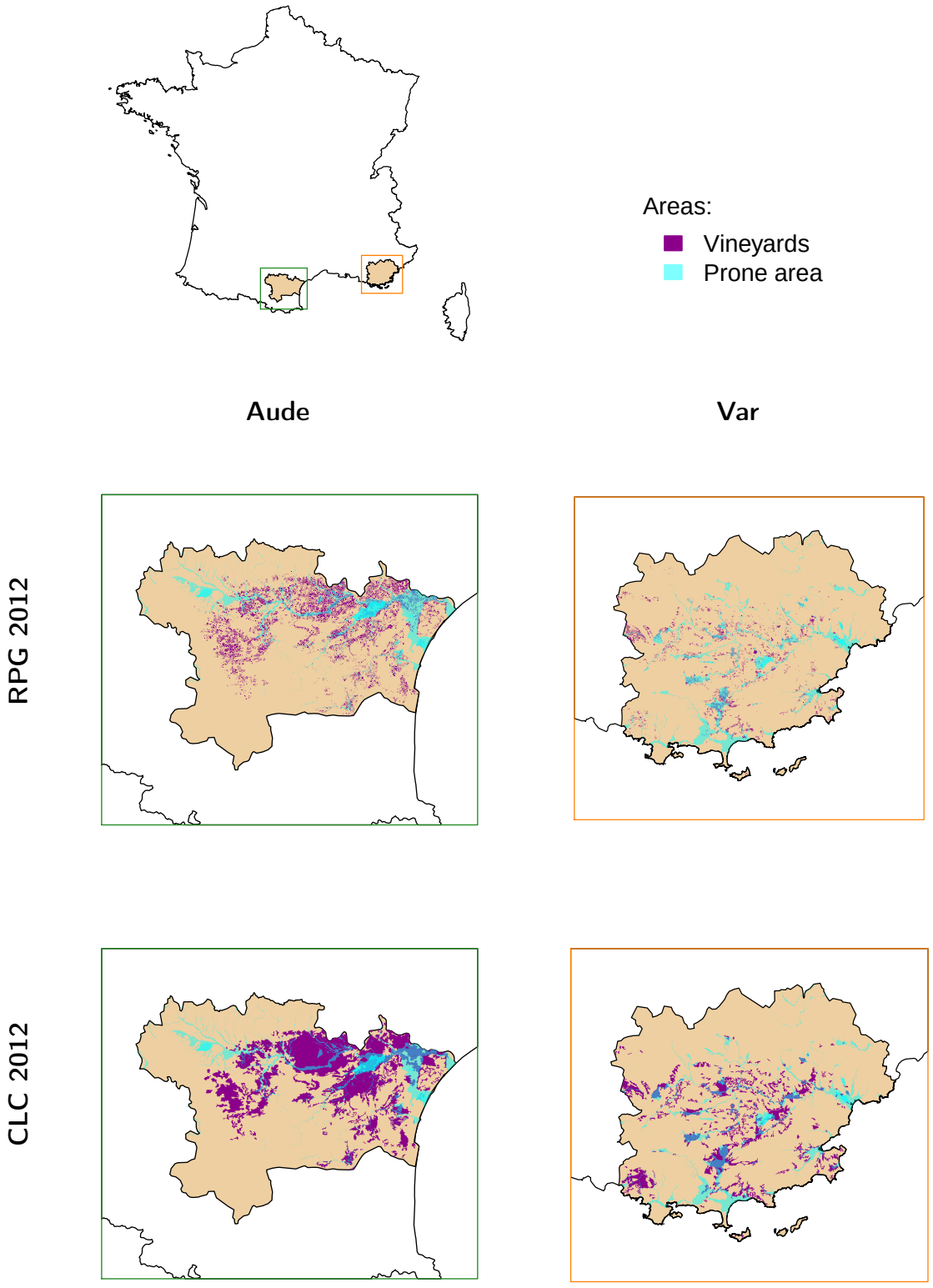
In our model, seasons come defined by whole months, instead of weeks. Thus, we need to find the way to summarize all information in figure 26 in data that can be handle by our model. To do that we follow a two-steps approach: first, over the initial distribution of Bremond [2011], we are able to calculate the number of hours spent per task each month, using the standard ISO week numbers and their monthly correspondence (table 11).

Second, defining seasons as:

- Winter: December - January - February
- Spring: March - April - May
- Summer: June - July - August
- Autumn: September - October - November

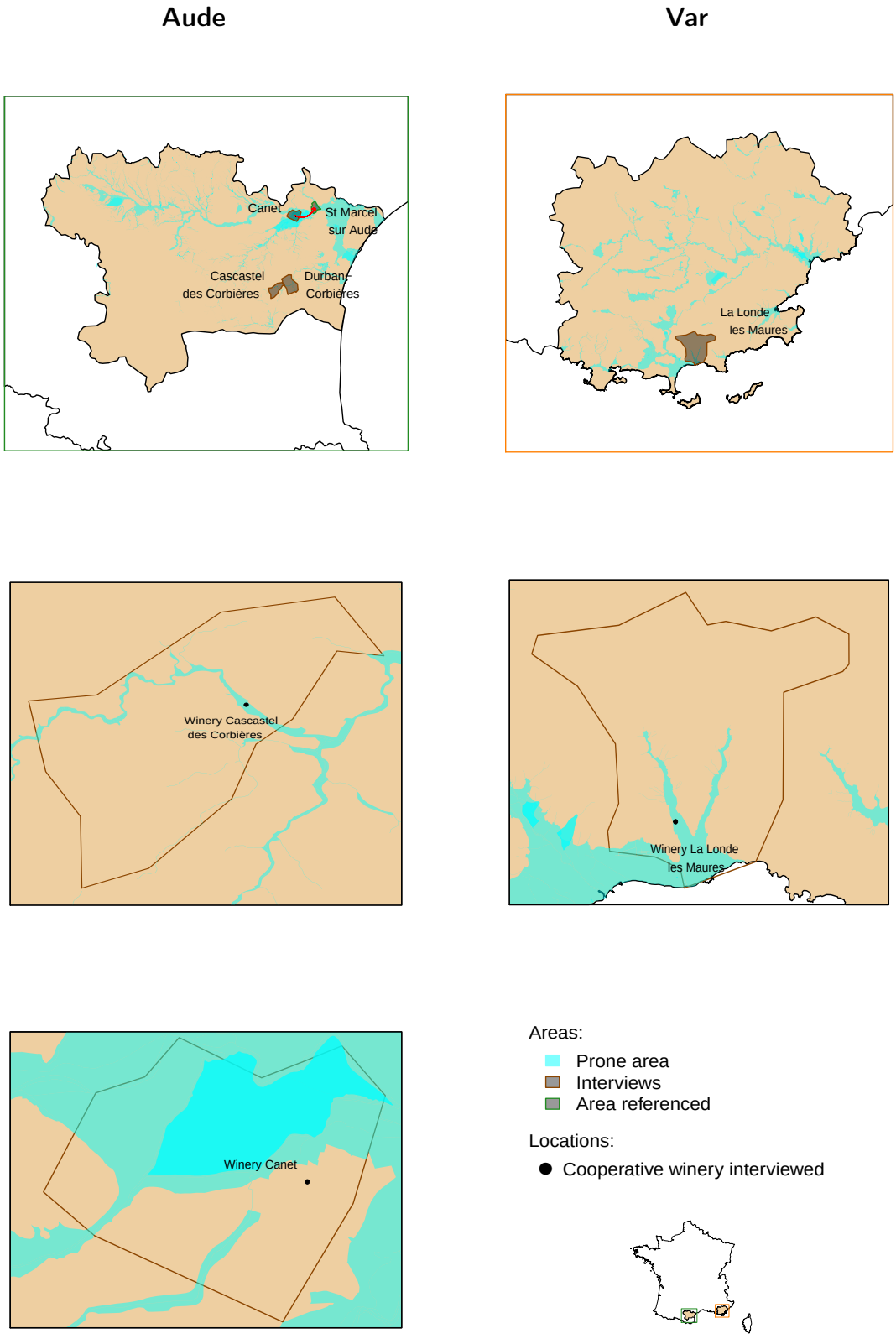
And attributing tasks to the season where they have more working hours [15], we obtain the assignation in table 12

[15] If a task has the same weight over two seasons, the criterion has been to attribute such task to the season it is started



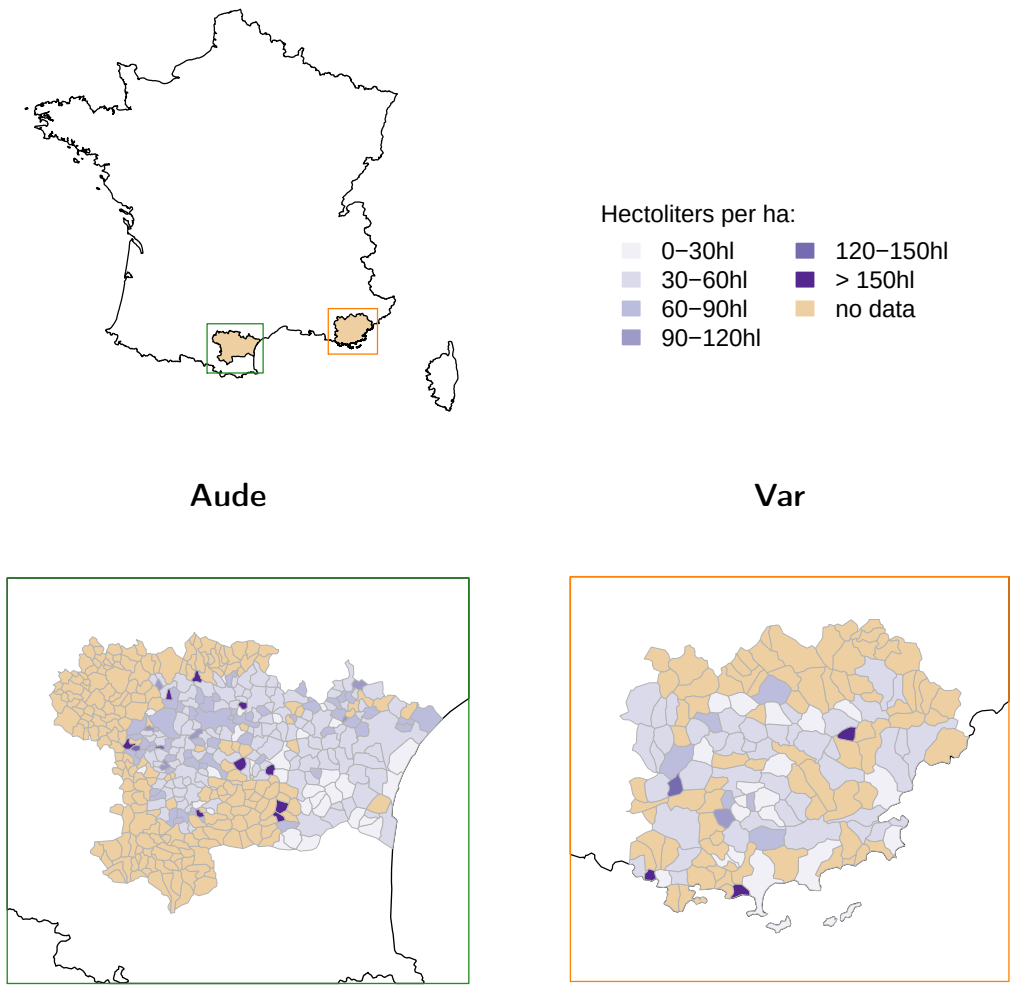
Source: own elaboration with data from CORINE Land Cover (CLC), Registre Parcellaire Graphique (RPG) and *Atlas des Zones Inondables de France*

Figure 23: Prone areas and vineyard extents in Aude and Var territories



Source: own elaboration with data from Google Maps and *Atlas des Zones Inondables de France*

Figure 24: Position of interviewed wineries in relation to prone areas



Source: own elaboration with data from France Agrimer 2012 data

Figure 25: vineyard's productivity per ha. Display by *commune*

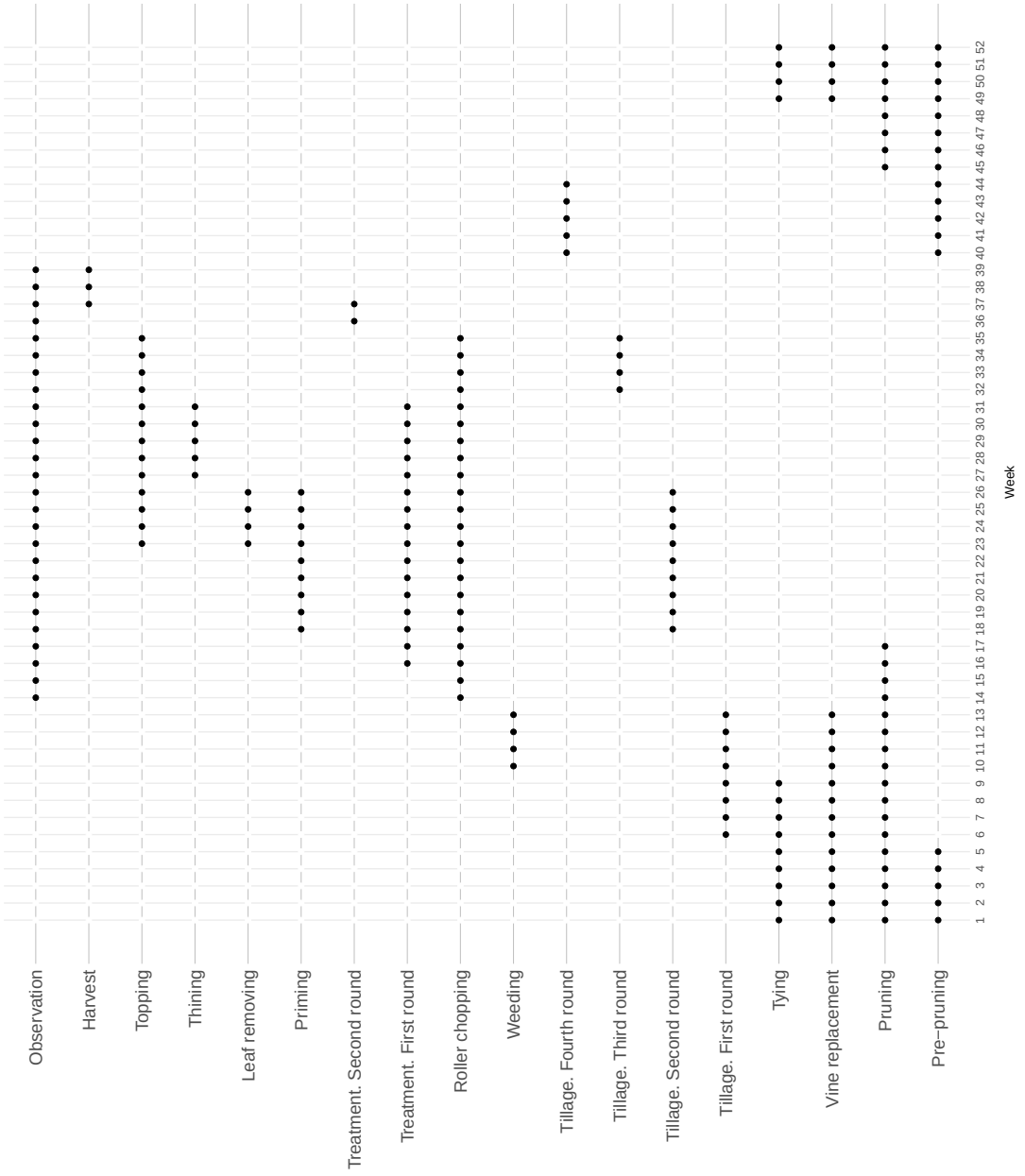


Figure 26: Annual pattern distribution of vine-growing tasks in Bremond [2011]

Weeks	1-4	5-8	9-13	14-17	18-22	23-26	27-30	31-35	36-39	40-43	44-48	49-52	
Month	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	**Total**
Prepruning	0.44	0.11								0.44	0.56	0.56	2
Pruning	6.72	6.72	8.4	6.72							6.72	8.4	42
Vine replacement	0.47	0.47	0.59									0.47	2
Tying	0.92	0.92	0.23									0.92	3
Tillage 1 round		1.5	2.5										4
Tillage 2 round					2.22	1.78							4
Tillage 3 round								2					2
Tillage 4 round										1.6	0.4		2
Weeding			2										2
Chopping				0.91	1.14	0.91	0.91	1.14					5
Treatment 1 round				1.31	3.28	2.62	2.62	0.66					10.5
Treatment 2 round									1.5				1.5
Priming					4.44	3.56							8
Leaf removing						4							4
Thinning							0.8	0.2					1
Topping						0.92	0.92	1.15					3
Harvest									7				7
Observation				0.92	1.15	0.92	0.92	1.15	0.92				6
Total	8.56	9.72	13.72	9.86	12.24	14.71	6.18	6.3	9.42	2.04	7.68	10.35	109

Table 11: Monthly repartition of tasks based on Bremond [2011] and ISO week numbers for standard 52 weeks' year. Unit: hours of labor

	Winter	Spring	Summer	Autumn
Prepruning	2			
Pruning	42			
Vine replacement	2			
Tying	3			
Tillage 1 round		4		
Tillage 2 round		4		
Tillage 3 round			2	
Tillage 4 round				2
Weeding		2		
Chopping			5	
Treatment 1 round			10.5	
Treatment 2 round				1.5
Priming		8		
Leaf removing			4	
Thinning			1	
Topping			3	
Harvest				7
Observation			6	
Total	49	18	31.5	10.5
Proportion over total	0.45	0.16	0.29	0.1

Table 12: Seasonal attribution of vine-growing tasks based on Bremond [2011]. Unit: hours of labor

Damages associated with *insourcing* coping tactic As explained in prior sections, when a farm uses the *insourcing* tactic, we assume such farm do not perform all their tasks, which is going to translate in certain level of damages in each of its productive plots. In Bremond [2011], tasks not performed are translated into losses (table 13).

Task	Damage	Task	Damage
Prepruning	0.1	Chopping	0.01
Pruning	0.4	Treatment 1 round	0.3
Vine replacement		Treatment 2 round	0.3
Tying	0.5	Priming	0.01
Tillage 1 round	0.05	Thinning	0.01
Tillage 2 round	0.05	Topping	0.01
Tillage 3 round	0.05	Harvest	1
Tillage 4 round	0.05	Observation	0.01
Weeding	0.3	Leaf removing	0.1

Table 13: Proportion of yield lost per task, based on Bremond [2011]

To calculate the seasonal attributed damage to each task, we have followed a cumulative method. To illustrate it, let's take winter as reference; according table 12, tasks to be done in this season are:

- Prepruning, which, if not done, provokes losses of 10% of the harvest per plot. Let's call it a
- Pruning. If not performed, losses of 40% of the harvest per plot. Hereafter known as b
- Tying. responsible of losing 50% of the harvest per plot when not done. Hereafter c
- Vine replacement, which, if not done, does not provoke any loss

The cumulative approach used, establishes that total losses can be expressed as:

$$harvest_{flood=winter} = (1-a) - b(1-a) - c((1-a) - b(1-a)) \tag{27}$$

Operating...

$$\begin{aligned} harvest_{flood=winter} &= (1-a) - b(1-a) - c((1-a) - b(1-a)) = \\ &= (1-a)(1+bc-b-c) = \\ &= (1-a)(1-b)(1-c) \end{aligned} \tag{28}$$

Therefore...

$$harvest_{flood=winter} = (1 - 0.1)(1 - 0.4)(1 - 0.5) = 0.27 \Rightarrow losses_{flood=winter} = 0.73 \quad (29)$$

For the rest of the seasons, results are summed up in table 14

	Winter	Spring	Summer	Autumn
Proportion of harvest damaged	0.73	0.37	0.43	1.00

Table 14: Seasonal attribution of damages based on Bremond [2011], in case seasonal vine-growing tasks are not performed

Vine-growing costs variations associated to coping tactics. Consequences of insourcing and outsourcing tactics are represented respectively in tables 15 and 16.

They both display the consequences over vine-growing costs and harvest over one productive plot —which has not been directly hit by the flood— when the farm it belongs to is flooded. Seasonal costs in absence of flood is calculated over a total per ha of 2 312.64€, applying the seasonal proportions of table 1.

First, table 15 shows the situation in which the farm opts for an *insourcing* tactic. The amount of task that can or cannot be done during the season, when the farm is hit by a flood, depends on multiple factors. Those factors, their behavior and the level of detailed analysis they require, are not the objective of our model nor they are implemented on it. Hence, we need a working hypothesis that allow us to compare the different outcomes of coping strategies.

Such working hypothesis has been to fix the amount of tasks the flooded farm is unable to perform to 50%. This way, every time a farm is flooded, automatically half of the tasks cannot be performed. Therefore, half of the vine-growing cost of the season in which the flood occurs will not be spent. Additionally, using table 14, we are able to know the level of damage it will cause to the harvest (all has been summarized in table 3). For instance, when a flood hits the farm in winter, the seasonal costs pass from €1 040.68 to €520.34; annual vine-growing cost then decreases from €2 312.64 to €1 792.30, and the farm loses 29.2 hl of production.

As it happens for *insourcing tactic*, when flooded farms opt for *outsourcing* tactic, we have no information about how much cost can increase[16]. Therefore, we will have to use, as well, working hypothesis to be able to simulate the effect of the tactic. For this case, we have set an increment of seasonal cost of 80% (table 16). Using the same example, now when the farm is flooded in winter, the seasonal cost pass from from €1 040.68 to €1 837.24, while annual vine-growing cost increases to €3 145.19, and the farm does not lose any production.

[16]To normal services prices we would have to add the emergency situation, the potential increment in the demand of such services in the aftermath of the flood, and, as well, the potential solidarity of agents, as it happens in real cases

		Not flooded	Flooded in: Winter	Spring	Summer	Autumn
Vine-growing costs	Winter	1040.68	520.34	1,040.69	1,040.69	1,040.69
	Spring	370.02	370.02	185.01	370.02	370.02
	Summer	670.66	670.67	670.67	335.33	670.67
	Autumn	231.26	231.26	231.26	231.26	115.63
	TOTAL	2312.64	1,792.30	2,127.63	1,977.31	2,197.01
Harvest (hl)		80.00	50.80	65.20	62.8	40.00

Table 15: Consequences on costs and harvest of *insourcing* tactic per productive plot, by flooding season. Units in euros (€); otherwise, explicitly indicated

		Not flooded	Flooded in: Winter	Spring	Summer	Autumn
Vine-growing costs	Winter	1040.68	1,873.24	1,040.69	1,040.69	1,040.69
	Spring	370.02	370.02	666.04	370.02	370.02
	Summer	670.66	670.67	670.67	1,207.20	670.67
	Autumn	231.26	231.26	231.26	231.26	416.28
	TOTAL	2312.64	3,145.19	2,608.66	2,849.17	2,497.65
Harvest (hl)		80.00	80.00	80.00	80.00	80.00

Table 16: Consequences on costs and harvest of *outsourcing* tactic per productive plot, by flooding season. Units in euros (€); otherwise, explicitly indicated

8.3 Financial structure

Both farms and wineries are provided with a simple financial structure, which distinguishes between fixed or structural cost, and variable or operational cost. Calculus for both parts are based on data from CER [2014].

For farms, the referred publication, based on a study of 2010 with 771 vine-growers, states that total cost per ha is €3 522. Of those, for vineyards with an average production of 80 hl per ha, €2 538 correspond to operational costs. Thus over the total cost per ha, structural cost represents the 28%

Those accounts include outsourcing of harvesting services (€310). To be coherent with our reasoning on coping tactics, we proceeded to reduce such total cost per ha in the amount of the outsourcing service. As a result we get a total cost per ha of €3 212, of which 28% corresponds to structural cost and 72% to operational cost.

In the model, each plot has associated an annual operational cost of €2 312.64 (72% of €3 212). To calculate the structural cost of farms, the following mechanism has been implemented:

$$Structural\ cost = 0.28 \text{ x } Number\ of\ ha\ owned \text{ x } Total\ cost\ per\ ha \tag{30}$$

If a farm owns 10 ha, its structural cost will be €8 993.6, whether it has or has not production. To such amount, we will add €2 312.64 each year, per productive plot, and €622 per unproductive plot[17].

For cooperatives, CER [2014] fixes a total price of €20 per hl of wine based on a study conducted in 2008. However no more detailed information is offered for cooperative wineries. Folwell and Castaldi [2004.] offer a detailed wine-making cost structure, from where we get fixed costs represent around 20% of the cost by hl. Using such reference, we implement the following mechanism to calculate the structural cost at winery's level:

$$Structural\ cost = 0.2 \text{ x } wine\ making\ cost\ per\ hl \text{ x } potential\ production\ of\ the\ winery \tag{31}$$

Where

$$potential\ production\ of\ the\ winery = \sum_{i=1}^{n} \sum_{\kappa=1}^{n_i} q_{i_T \kappa} \tag{32}$$

Assuming all $\gamma_{i\kappa}$ plots owned by farm i are productive.

Let's assume we have a cooperative winery with ten associates, and, each of them, own ten plots of extent 1 ha and an average yield of 80hl per ha. Structural cost in that winery will be:

[17] 20% of total cost per ha. Hypothesis made based on the price of phytosanitary products, herbicides, fertilizers, etc

$$Structural\ cost = 0.2 \text{ x } wine\ making\ cost\ per\ hl \text{ x } \sum_{i=1}^{n}\sum_{\kappa=1}^{n_i} q_{i_T\kappa} \tag{33}$$

$$Structural\ cost = 0.2 \text{ x } 20 \text{ x } \sum_{i=1}^{10}\sum_{\kappa=1}^{10} 80$$

$$Structural\ cost = 0.2 \text{ x } 20 \text{ x } 8000 = 32\ 000$$

A winery as that one will have €32 000 of structural cost plus €16 per hl as operational cost

9 Initialization

This section is dedicated to specify the set up value every parameter and variable gets in the model.

Terrain
Terrain's size set to 150 patches from the origin of coordinates.
Prone area set to 100 patches from the origin of coordinates.
Origin of coordinates located in the bottom left corner.
Floods moving from the left edge to the right along the x axis.

Economic environment
Wine prices: €80 per hl[1]

[1] Average price using Agrimer [avril 2016 / mars 2016]. We assume it constant and not endogenously determined by the model.

Plot
Distribution over terrain: random
Extent: 1 ha per plot
Productivity: 80 hl per ha
Initial investment (reposition cost): €19 394 per ha
Life expectancy (investment life): 30 years[1]
Age unproductive: [0, 5) years
Age productive: [5, 30] years
Age: random in the range [0, 30]
State: all plots planted in the initialization
Reparation cost for plants and infrastructure: €19 394 per ha
Reparation cost for soils: €600 per ha[2]
Reparation cost for plot: €19 994 per ha
Owner: either random or preassigned by user

[1] Hypothesis made based on expected life of the investment (25 years) in CER [2014]

[2] Hypothesis made based on the costs of phytosanitaries, herbicides and fertilizers. The sum corresponds to the costs of such element in a normal year. See CER [2014]

Farm
Distribution over terrain: random
Extent owned: Variable
Total costs per ha (when plot productive): €3 212 per ha
Operational cost per ha (when plot productive): 72% of total costs[1]
Structural cost: 28% of total costs[2]
Total costs per ha (when plot unproductive): 20% of total costs[3]
Cumulative balance: €0
Reparation costs of damages: 30 times the value of the potential production[4]

[1] €2 228 per ha

[2] Calculated taking as reference the size of the farm: *Structural costs = 0.28 * Number of croplands * Total Costs per ha*

[3] €622 per ha. Hypothesis made based on operational costs.

[4] €Hypothesis made based on Bremond [2011].

Cooperative wineries
Distribution over terrain: random
Number of associates: Variable
Vinification costs: €20 per hl
Fixed vinification cost proportion: 20%[1]
Variable vinification cost per hl: 80% of the vinification cost[2]
Efficiency : 100% (Every hl harvested corresponds to 1 hl of wine.)
Initial investment of winery: €290 per hl[3]
Reparation costs of damages: 30% of Winery's property value [4]

[1] See section 8

[3] Winery's property value is then calculated in the setup as:

$$Property\ value\ of\ winery = Initial\ investment\ of\ winery\ \text{x}\ \sum_{i=1}^{n}\sum_{\kappa=1}^{n_i} q_{i_T\kappa} \qquad (34)$$

Assuming all $\gamma_{i\kappa}$ plots owned by farm i are productive. Example with cooperative winery with ten associates, and, each of them, own ten plots of extent 1 ha and an average yield of 80hl per ha:

$$Property\ value\ of\ winery = 290 \text{ x } \sum_{i=1}^{10}\sum_{\kappa=1}^{10} 80 \tag{35}$$
$$Property\ value\ of\ winery = 290 \text{ x } 8000 = 2\ 320\ 000$$

[4] Hypothesis based in Folwell and Castaldi [2004.] and interviews with agents. When tested over a winery of around 6000hl, the amount of damage is similar to the amount declared in La Londe's interview.

[1] [2] [3] [4] Adapted from information in Folwell and Castaldi [2004.]

List of abbreviations

BAU Bussines as Usual scenario or Zero Flood Scenario. 6, 7, 9, 10, 30, 31, 41, 48, 66, *Glossary:* Bussines as Usual scenario or Zero Flood Scenario

CLC CORINE Land Cover. 52, *Glossary:* CORINE Land Cover

CPU Central Process Unit. 48, 65, 66, *Glossary:* central process unit

RAM Random Access Memory. 48, *Glossary:* random access memory

RNG Random Number Generator. 40, 45, *Glossary:* random number generator

RPG Registre Parcellaire Graphique. 52, *Glossary:* registre parcellaire graphique

SFS Simulated Flood Scenario. 6–9, 22–25, 28, 30, 31, 41, 48, 65, *Glossary:* Simulated Flood Scenario

List of Terms

Bussines as Usual scenario or Zero Flood Scenario Simulation scenario generated in Netlogo where no flood is simulated. It serves as baseline to be compared with the SFSs and calculate the impacts of the different floods. 6, 65

Central process unit it handles all instructions it receives from hardware and software running on the computer. Also known as processor, central processor, or microprocessor. 65

Cluster Generally speaking a computer cluster refers to a group of computers capable to work as a one single unit. In our context, we are referring to the collection of CPUs used by the parallel processes in R to work simultaneously.
As a particularity R always uses one CPU no matter how many are present on the computer. When we provide our code parallel-ready, R is capable of launching tasks over all the CPUs we indicate when we set up the cluster. 48

CORINE Land Cover Geographic database of biophysical soil occupation and use on the territory of the European Union. 50, 52, 65

Direct impact impacts *related to direct exposure to the disaster* (physically flooded in our case). See Brémond et al. [2013]. 33

GUI Acronym for Graphical User Interface. Type of interface that allows users to interact with electronic devices or software by direct manipulation of graphical elements, for instance icons, instead of typing command lines. 43, 65

Headless mode Feature of certain kinds of software that allows them to work without GUI. 43

Immediate impact impacts which occurs during or immediately after the flood event. See Brémond et al. [2013]. 32

Indirect impact impacts *which occurs in a area that has not been exposed to flooding.* See Brémond et al. [2013]. 33

Induced impact impacts which occurs occur later in time after the flood event. See Brémond et al. [2013]. 32

Netlogo Open source multi-agent programmable modeling environment supported by the Center for Connected Learning and Computer-Based Modeling at Northwestern University. 7, 9, 38, 39, 43, 45, 48, 65, 66

Operational cost costs linked to business' volume of activity. 61–63

Parallel-ready code capable of performing tasks simultaneously, using more than one of CPU cores, instead of performing tasks sequentially over one and only one CPU core. 48, 65

Patch space unit in `Netlogo` terminology. 62

R Open source programming language and software environment for statistical computing and graphics, supported by the R Foundation for Statistical Computing. 3, 38, 41, 45, 65

Random access memory part of the computing device where ongoing tasks store data so it can be quickly reached by the device's processor, since it is much faster to read from and write to than other kinds of storage. 65

Random number generator algorithm for generating sequences of numbers whose properties approximate the properties of sequences of random numbers.. 40, 65

Registre parcellaire graphique Geographic database of agricultural plots in the french territory. 50, 52, 65

RNetLogo Package that embeds NetLogo in R, providing functions to load models, execute commands, and get values from reporters. 38, 48

Simulated Flood Scenario Simulation scenario generated in Netlogo where a flood of a given extent in a given season is simulated. When confronted with the BAUs, they allow disruptions and damages caused by the flood to emerge. 6, 65

Simulation procedure code dedicated to the simulation of impacts of a flood. It includes the `simulation launcher/iterator`, the `flood simulator` and the `impacts calculator`. 39, 40, 43

Simulation process entire process of simulation of impacts of a flood, including generation of inputs and output analysis. 39, 43, 48

Structural cost fixed costs of the business. Resulting from long term decisions and choices. 9, 20, 21, 30, 61–63

Tick Time step in `Netlogo` terminology. 7, 9

References

Agrimer. *Constatation nationale des prix moyens d'achat de vin en vrac. Campagne vitivinicole 2015-2016*, avril 2016 / mars 2016.

Revolution Analytics and Steve Weston. *doParallel: Foreach Parallel Adaptor for the 'parallel' Package*, 2015. URL https://CRAN.R-project.org/package=doParallel. R package version 1.0.10.

Adrian Baddeley, Ege Rubak, and Rolf Turner. *Spatial Point Patterns: Methodology and Applications with R*. Chapman and Hall/CRC Press, 2015.

L. Barbut, N. Bauduceau, and C. Devaux Ros. Vers une evaluation de la vulnérabilité des activités agricoles aux inondations. *Ingénieries - E A T, IRSTEA*, pages 29 – 41, 2004.

A. Biarnés and J.M. Touzard. La construction de la rémunération différenciée du raisin dans les coopératives du languedoc roussillon. In J.M; Touzard and J.F. Draperi, editors, *Les coopératives entre territoires et mondialisation*. L'harmattan, 2003.

Roger Bivand and Nicholas Lewin-Koh. *maptools: Tools for Reading and Handling Spatial Objects*, 2017. URL https://CRAN.R-project.org/package=maptools. R package version 0.8-41.

Roger Bivand and Colin Rundel. *rgeos: Interface to Geometry Engine - Open Source (GEOS)*, 2017. URL https://CRAN.R-project.org/package=rgeos. R package version 0.3-22.

Roger Bivand, Tim Keitt, and Barry Rowlingson. *rgdal: Bindings for the Geospatial Data Abstraction Library*, 2016. URL https://CRAN.R-project.org/package=rgdal. R package version 1.2-5.

Roger S. Bivand, Edzer Pebesma, and Virgilio Gomez-Rubio. *Applied spatial data analysis with R*. Springer, NY, second edition edition, 2013.

P. Bremond. *Caractérisation et Évaluation Économique de la vulnérabilité des exploitations agricoles aux inondations*. PhD thesis, École Doctorale Économie et Gestion de Montpellier, 2011.

P. Brémond, F. Grelot, and A.L. Agenais. Review article: economic evaluation of flood damage to agriculture - review and analysis of existing methods. *Natural Hazards and Earth System Sciences*, 13:p. 2493 – p. 2512, 2013. doi: 10.5194/nhess-13-2493-2013. URL https://hal.archives-ouvertes.fr/hal-00910741.

Francois Briatte. *ggnetwork: Geometries to Plot Networks with 'ggplot2'*, 2016. URL https://CRAN.R-project.org/package=ggnetwork. R package version 0.5.1.

Carter T. Butts. network: a package for managing relational data in r. *Journal of Statistical Software*, 24(2), 2008. URL http://www.jstatsoft.org/v24/i02/paper.

Carter T. Butts. *network: Classes for Relational Data*. The Statnet Project (http://statnet.org), 2015. URL http://CRAN.R-project.org/package=network. R package version 1.13.0.

Carter T. Butts. *sna: Tools for Social Network Analysis*, 2016. URL https://CRAN.R-project.org/package=sna. R package version 2.4.

CER. *Références technico-économiques d'une vigne palissée en Languedoc Roussillon*, 2014.

David B. Dahl. *xtable: Export Tables to LaTeX or HTML*, 2016. URL https://CRAN.R-project.org/package=xtable. R package version 1.8-2.

Matt Dowle and Arun Srinivasan. *data.table: Extension of 'data.frame'*, 2017. URL https://CRAN.R-project.org/package=data.table. R package version 1.10.4.

R. J. Folwell and M. A. Castaldi. Bulk winery investment and operating costs. *Agricultural Research Center, College of Agricultural, Human, and Natural Resources Sciences, Washington State University, Pullman, Washington*, 2004.

GRASS Development Team. *Geographic Resources Analysis Support System (GRASS GIS) Software, Version 7.2*. Open Source Geospatial Foundation, 2017. URL http://grass.osgeo.org.

V. Grimm, U. Bergerb, F. Bastiansena, S. Eliassenc, V. Ginotd, J. Jarl Giskec, J. Goss-Custarde, T. Grandf, S. K. Heinzc, G. Huseg, A. Hutha, J. U. Jepsena, C. JÃÿrgensenc, W. M. Mooijh, B. MÃijllera, G. PeâĂŹeri, C. Pioub, S. F. Railsbackj, A. M. Robbinsk, M. M. Robbinsk, E. Rossmanithl, N. RÃijgera, E. Strandc, S. Souissim, R. A. Stillmane, R. VabÃÿg, U. Vissera, and D. L. DeAngelis. A standard protocol for describing individual-based and agent-based models. *Ecological modelling*, (198):115–126, 2006.

M. Hiete and M. Merz. Indicator framework for the assessment of indirect industrial vulnerabilities. In J. Landgren and S. Ju, editors, *Proceedings of the 6th International ISCRAM Conference*, Gothenburg, Sweden, May 2009.

Robert J. Hijmans. *raster: Geographic Data Analysis and Modeling*, 2016. URL https://CRAN.R-project.org/package=raster. R package version 2.5-8.

Q. Ethan McCallum and Stephen Weston. *Parallel R*. O'Reilly Media Inc, 2012.

Erich Neuwirth. *RColorBrewer: ColorBrewer Palettes*, 2014. URL https://CRAN.R-project.org/package=RColorBrewer. R package version 1.1-2.

Beatriz Pateiro-Lopez, Alberto Rodriguez-Casal, and . *alphahull: Generalization of the Convex Hull of a Sample of Points in the Plane*, 2016. URL https://CRAN.R-project.org/package=alphahull. R package version 2.1.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2017. URL https://www.R-project.org/.

David Robinson. *broom: Convert Statistical Analysis Objects into Tidy Data Frames*, 2017. URL https://CRAN.R-project.org/package=broom. R package version 0.4.2.

L. Tesfatsion. Agent based computational economics. In Leigh Tesfatsion and Kenneth L. Judd, editors, *Handbook of Computational Economics*, volume 2, chapter 30. Elsevier B.V., 2006. doi: 10.1016/S1574-0021(05)02030-7.

Jan C. Thiele, Winfried Kurth, and Volker Grimm. RNetLogo: An R package for running and exploring individual-based models implemented in NetLogo. *Methods in Ecology and Evolution*, 3(3):480–483, 2012. URL http://onlinelibrary.wiley.com/doi/10.1111/j.2041-210X.2011.00180.x/abstract.

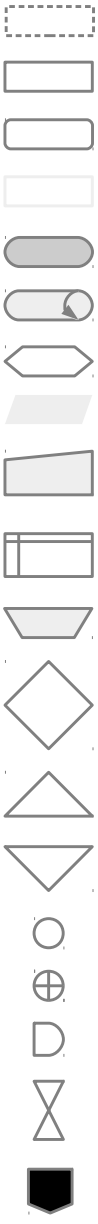
Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2009. ISBN 978-0-387-98140-6. URL http://ggplot2.org.

Hadley Wickham. *scales: Scale Functions for Visualization*, 2016. URL https://CRAN.R-project.org/package=scales. R package version 0.4.1.

U. Wilensky. *NetLogo*. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL., 1999.

Claus O. Wilke. *cowplot: Streamlined Plot Theme and Plot Annotations for 'ggplot2'*, 2016. URL https://CRAN.R-project.org/package=cowplot. R package version 0.7.0.

Annex A Flowchart symbols cheat sheet



Environment

Process, action or task

Alternative process, action or task

Comment

Start/end point of a procedure

Storage on hard drive or other physical device

Preparation/Set up process, action or task

Data input/output

Manual input

Storage on virtual memory

Manual operation

Choice/decision

Split of processes, actions or tasks

Merge of processes, actions or tasks

Connector

Logical OR

Waiting period, delay

Data conversion to standard format

Off-page connector

Annex B Size of farms according available data sources

AGRESTE report for Aude's region. Year 2010

Size	ha (average)
Small	2.9
Medium	11.5
Large	48

Table 17

Agrarian census for Aude's region. Year 2010

Size	ha (average)
Small	2.72
Medium/Large	27.85

Table 18

Canet's cooperative winery interview. Information referred only to its associates

Size	ha (average)
Small	2-3
Medium	20-25
Large	50-60

Table 19